FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Impacto de Slotted Aloha nas comunicações LoRa com baixo consumo

Eduardo Soqueiro Barbosa

Mestrado em Engenharia Eletrotécnica e de Computadores

Orientador: Prof. Luís de Almeida

27 de julho de 2026

Resumo

LoRa é uma tecnologia de comunicação desenvolvida para redes LPWAN que possibilita a implementação de aplicações de IoT em grandes áreas geográficas, desde ambientes urbanos até zonas rurais. Um dos domínios onde pode ser aplicada é na monitorização e controlo em tempo real de campos petrolíferos. No entanto, este domínio de aplicação impõe requisitos de tempo real que o LoRa não foi originalmente concebido para suportar.

Vários trabalhos têm sido desenvolvidos com o objetivo de introduzir capacidades de tempo real em LoRa, recorrendo essencialmente a mecanismos de sincronização rigorosa para resolver problemas de contenção no acesso ao meio. Contudo, estas técnicas destinadas a melhorar o comportamento em tempo real conduzem também a um aumento do consumo energético, o que pode comprometer a autonomia e o tempo de vida útil do sistema.

Esta dissertação propõe o protocolo CW-S-ALOHA, uma variante do Slotted ALOHA que preserva o mecanismo de ressincronização por *timestamps* piggybacked no *downlink*, substituindo o *backoff* reativo por uma dispersão proativa das transmissões numa janela de contenção configurável. A avaliação experimental, conduzida com quatro nós LoRaWAN em hardware de baixo custo e com servidor de rede, compara o CW-S-ALOHA com o Pure ALOHA em três regimes de carga crescente. Os resultados confirmam melhorias consistentes no PDR, com ganhos que atingem cerca de 10 pontos percentuais no regime de carga mais elevado, confirmando o ganho teórico previsto pela duplicação do *throughput* máximo do Slotted ALOHA face ao Pure ALOHA.

Palavras-chave: LoRa • LoRaWAN • IoT • Pure ALOHA • Slotted ALOHA • CW-S-ALOHA • PDR • Sincronização Temporal • Energia

UN Sustainable Development Goals

The United Nations Sustainable Development Goals (SDGs) provide a global framework to achieve a better and more sustainable future for all. They include 17 goals[1] to address the world's most pressing challenges, including poverty, inequality, climate change, environmental degradation, peace, and justice.

This dissertation contributes to two SDGs, summarized below.

SDG 9 Build resilient infrastructure, promote inclusive and sustainable industrialization and foster innovation

SDG 7 Ensure access to affordable, reliable, sustainable and modern energy for all

SDG	Target	Contribution	Performance Indicators and Metrics
9	9.1	Proposes and validates CW-S-ALOHA, a reliable medium-access mechanism for LoRaWAN built on low-cost commercial hardware, improving the quality and resilience of wireless communication infrastructure for industrial IoT applications.	PDR gain (pp), normalized throughput gain (%) over Pure ALOHA across three load regimes
	9.5	Contributes original experimental methodology and open-source firmware and server implementation, supporting further research on LPWAN MAC protocols.	Public repository availability; reproducibility of results
7	7.3	Quantifies the energy cost of improved communication reliability in battery-powered IoT devices, providing data to inform energy-aware protocol design.	Average current consumption (mA), battery autonomy estimate (h)

[1] https://sdgs.un.org/goals

Agradecimentos

Gostaria de expressar o meu sincero agradecimento a todas as pessoas que, de alguma forma, contribuíram para a realização desta dissertação. Um agradecimento muito especial ao meu excelentíssimo professor orientador, Luís Almeida, por toda a orientação, partilha de conhecimento e paciência ao longo de todo este percurso.

Aos meus colegas de laboratório, Miguel Almeida e José Felisberto, agradeço o excelente ambiente de trabalho e a entreajuda constante. Dirijo também um profundo agradecimento ao Onoriode Akporherhe por ter despendido tantas horas para me apoiar e por ter disponibilizado generosamente o seu próprio material para que eu me conseguisse ambientar à componente prática deste projeto.

À minha mãe, ao meu pai, ao meu irmão e à minha namorada, agradeço o amor incondicional e o suporte de sempre, pois sem o vosso apoio constante este caminho teria sido muito mais difícil. Por fim, não posso deixar de deixar uma palavra de carinho aos meus irmãos felinos, o José e o Júlio, pelas longas horas de companhia silenciosa e por me confortarem sempre nos momentos de maior cansaço.

"O caos é uma ordem por decifrar."

José Saramago, *O Homem Duplicado* (2002)

Índice

Lista de Figuras

Lista de Tabelas

Lista de Códigos

Lista de Acrónimos

ABP	Activation By Personalization
ACK	Acknowledgment
ADR	Adaptive Data Rate
AES-128	Advanced Encryption Standard-128 bits
AppEUI	Application Extended Unique Identifier
AppKey	Application Key
AppSKey	Application Session Key
BER	Bit Error Rate
BW	Bandwidth
CR	Coding Rate
CSS	Chirp Spread Spectrum
CW	Contention Window
CW-S-ALOHA	Contention Window Slotted ALOHA
DevAddr	Device Address
DevEUI	Device Extended Unique Identifier
DevNonce	Device Nonce
DL	Downlink
DR	Data Rate
ED	End Device
ETSI	European Telecommunications Standards Institute
EU868	European 868 MHz Frequency Band
FCnt	Frame Counter
FCtrl	Frame Control
FHDR	Frame Header
FOpts	Frame Options
FPort	Frame Port
FRMPayload	Frame Payload
GPIO	General Purpose Input/Output
gRPC	Google Remote Procedure Call
GW	Gateway
HTTP	Hypertext Transfer Protocol
HVPM	High Voltage Power Monitor
IoT	Internet of Things
ISM	Industrial, Scientific and Medical
JoinEUI	Join Extended Unique Identifier
LoRa	Long Range
LoRaWAN	Long Range Wide Area Network
LPWAN	Low-Power Wide-Area Network
MAC	Medium Access Control
MCU	Microcontroller Unit
MHDR	MAC Header
MIC	Message Integrity Code
NwkSKey	Network Session Key
OLED	Organic Light-Emitting Diode
OTAA	Over-The-Air Activation
PDR	Packet Delivery Ratio
PHY	Physical Layer
RF	Radio Frequency
RSSI	Received Signal Strength Indicator
RTOS	Real-Time Operating System
RX	Receive Window
S-ALOHA	Slotted ALOHA
SF	Spreading Factor
SNR	Signal-to-Noise Ratio
SoC	System on Chip
SPI	Serial Peripheral Interface
TDMA	Time Division Multiple Access
ToA	Time-on-Air
TX	Transmit
UART	Universal Asynchronous Receiver-Transmitter
UDP	User Datagram Protocol
UL	Uplink
UTC	Coordinated Universal Time

Capítulo 1

Introdução

A Internet das Coisas (IoT) permitiu a ligação de dispositivos físicos, como sensores, atuadores e sistemas embebidos, a redes de comunicação, possibilitando a recolha de dados e a atuação remota sobre processos em tempo real através da Internet. Este paradigma suporta aplicações em domínios críticos como a monitorização industrial, o controlo de processos e a gestão de infraestruturas, nos quais as tecnologias de comunicação de longo alcance sem fios assumem um papel fundamental.

As aplicações IoT apresentam requisitos distintos dos das redes de comunicação tradicionais. Muitos dispositivos operam em ambientes remotos, são alimentados por bateria, transmitem pequenas quantidades de dados e devem manter-se operacionais durante anos sem intervenção humana. Estas características motivaram o desenvolvimento das *Low-Power Wide-Area Networks* (LPWAN), concebidas para suportar alcances típicos entre 5 e 30 km, potências de transmissão inferiores a 100 mW, consumos energéticos em modo de repouso tipicamente inferiores a 10 µW, taxas de dados inferiores a 50 kbit/s e custos de comunicação por nó da ordem de alguns dólares, permitindo autonomias de vários anos em dispositivos alimentados a bateria [21].

A tecnologia LoRa destacou-se entre as tecnologias LPWAN pela sua ampla adoção, pela utilização de bandas de frequência não licenciadas e pela especificação aberta LoRaWAN, mantida pela LoRa Alliance [2]. A modulação LoRa, baseada em *Chirp Spread Spectrum*, permite comunicações a distâncias de vários quilómetros com consumos energéticos compatíveis com anos de operação autónoma [4]. Contudo, a especificação LoRaWAN não foi concebida para suportar comunicação em tempo real, introduzindo desafios significativos quando aplicada a cenários industriais críticos onde latências previsíveis e garantias temporais são essenciais.

1.1 Declaração do Problema

O protocolo LoRaWAN foi concebido para cenários caracterizados por tráfego esporádico, comunicação predominantemente ascendente e tolerância a latências elevadas. Estas características resultam de opções do protocolo, como o acesso ao meio baseado em Pure ALOHA, a ausência de sincronização temporal global e a disponibilidade limitada de *downlink*. Embora adequadas para

muitas aplicações IoT, estas opções dificultam a utilização direta de LoRaWAN em sistemas que requerem comportamento temporal previsível.

Este problema torna-se particularmente relevante em aplicações industriais distribuídas, como a monitorização e controlo de campos petrolíferos, onde sensores e atuadores se encontram dispersos por áreas extensas e devem comunicar eventos críticos dentro de limites temporais definidos. Nestes contextos, atrasos imprevisíveis na comunicação podem comprometer a eficiência operacional e, em situações extremas, a segurança de pessoas e infraestruturas.

Para responder a estas limitações, diversos trabalhos propuseram extensões ao LoRaWAN que introduzem sincronização temporal e mecanismos de acesso coordenado ao meio. Contudo, estas soluções tendem a aumentar o consumo energético, criando um compromisso entre previsibilidade temporal e eficiência energética que importa caracterizar experimentalmente.

1.2 Objetivos

Esta dissertação tem como objetivo avaliar o impacto da introdução de acesso ao meio baseado em Slotted ALOHA sobre uma infraestrutura LoRaWAN real, tomando o Pure ALOHA como referência de base. A avaliação é conduzida experimentalmente sobre hardware comercial de baixo custo, com foco nas métricas de fiabilidade de entrega, débito efetivo e consumo energético.

Os principais objetivos deste trabalho são os seguintes:

- Desenvolver uma infraestrutura experimental baseada em LoRaWAN que permita avaliar e comparar diferentes mecanismos de acesso ao meio, utilizando quatro nós terminais, um *gateway* comercial e um servidor de rede;
- Testar o Pure ALOHA sobre LoRaWAN 1.0.3 como protocolo de referência, caracterizando o seu desempenho em três regimes de carga de rede distintos;
- Propor e implementar a variante CW-Slotted ALOHA (CW-S-ALOHA) sobre LoRaWAN 1.1, introduzindo sincronização temporal por *timestamps* e dispersão *a priori* da janela de contenção;
- Avaliar experimentalmente ambos os protocolos, analisando métricas de desempenho como a taxa de entrega de pacotes, o débito normalizado, o erro de sincronização temporal e o consumo energético;
- Confrontar os resultados obtidos com os modelos analíticos teóricos, identificando as causas de desvio e os limites práticos de cada protocolo;
- Validar o compromisso entre fiabilidade do protocolo e eficiência energética nas soluções estudadas, quantificando o ganho introduzido pelo mecanismo de Slotted ALOHA face ao Pure ALOHA em condições reais.

1.3 Esboço da Dissertação

Este documento encontra-se organizado da seguinte forma:

- Capítulo 2 – Fundamentos Tecnológicos: apresenta os principais conceitos associados à tecnologia LoRa e ao protocolo LoRaWAN, descrevendo o funcionamento da modulação *Chirp Spread Spectrum*, os parâmetros de desempenho relevantes, a arquitetura da rede, as classes de dispositivos e os mecanismos de segurança e ativação. O capítulo termina com uma síntese das limitações estruturais do LoRaWAN que motivam as extensões estudadas nesta dissertação.
- Capítulo 3 – Estado da Arte e Protocolo Proposto: analisa os fundamentos teóricos do Pure ALOHA e do Slotted ALOHA no contexto LoRaWAN, descreve o S-LoRaWAN de referência proposto por Polonelli et al. e apresenta a variante CW-S-ALOHA desenvolvida nesta dissertação, incluindo o seu modelo analítico de débito e a justificação da opção pela Classe A.
- Capítulo 4 – Implementação e Avaliação Experimental: descreve a infraestrutura experimental utilizada, a arquitetura de firmware comum aos dois protocolos e os detalhes de implementação de cada um, incluindo o dimensionamento dos parâmetros, o processamento no servidor Flask e a metodologia de avaliação adotada.
- Capítulo 5 – Resultados e Discussão: apresenta e analisa os resultados experimentais do Pure ALOHA e do CW-S-ALOHA nos três regimes de carga avaliados, confrontando os dados obtidos com as previsões teóricas e comparando diretamente os dois protocolos em termos de fiabilidade, débito, sincronização e consumo energético.
- Capítulo 6 – Conclusões e Trabalho Futuro: resume as principais contribuições da dissertação, discute as limitações do trabalho realizado e apresenta direções para investigação futura.

Capítulo 2

Fundamentos Tecnológicos

O desempenho das comunicações em redes LoRa depende de um conjunto de fatores tecnológicos que importa compreender antes de analisar os protocolos de acesso ao meio estudados nesta dissertação. O capítulo começa pelos fundamentos da modulação CSS e do SF (*Spreading Factor*), passando pela ortogonalidade entre canais, colisões e *capture effect*. Seguem-se os parâmetros de desempenho mais relevantes, o ToA (*Time on Air*) e o ADR (*Adaptive Data Rate*), a arquitetura e a pilha protocolar LoRaWAN, a estrutura das mensagens, as classes de dispositivos e os mecanismos de segurança e ativação, e as limitações estruturais do protocolo que motivam as extensões estudadas nos capítulos seguintes.

2.1 Visão Geral da Tecnologia LoRa e LoRaWAN

LoRa é uma tecnologia de comunicação sem fios baseada na modulação *Chirp Spread Spectrum* (CSS), concebida para transmissões de longo alcance com baixo consumo energético. O seu principal campo de aplicação são dispositivos alimentados por bateria que enviam pequenas quantidades de dados de forma esporádica. Em ambiente urbano, uma única *gateway* cobre tipicamente 2 a 5 km, podendo atingir 15 a 20 km em zonas rurais com linha de vista [4]. A taxa de transmissão máxima é de 5,4 kbit/s com SF7 e BW = 125 kHz, descendo até 250 bit/s com SF12 em troca de maior alcance [26]. A autonomia de bateria típica atinge vários anos em regimes de transmissão esporádica [8].

A utilização do espetro na banda ISM de 863 a 870 MHz está regulamentada pela norma ETSI EN 300.220 [11], que estabelece limites de *duty-cycle* para evitar que um nó monopolize o meio de comunicação. Na maioria dos canais LoRaWAN da banda EU868, o limite é de 1%, o que significa que cada dispositivo apenas pode transmitir durante uma pequena fração do tempo. A Tabela 2.1 apresenta os limites para as sub-bandas mais utilizadas.

Sub-banda [MHz]	Duty-cycle	Tempo TX [s/h]	Uso típico
863.0 – 868.0	1%	36	Uplink
868.0 – 868.6	1%	36	Uplink
868.7 – 869.2	0,1%	3,6	Uplink
869.4 – 869.65	10%	360	Downlink (alta potência)
869.7 – 870.0	1%	36	Uplink

Tabela 2.1: Limites de *duty-cycle* nas sub-bandas EU868 (ETSI EN 300.220).

Para um dispositivo que transmita com um ToA de 200 ms num canal com *duty-cycle* de 1%, o tempo mínimo entre transmissões consecutivas é de 20 s. Esta restrição condiciona severamente a capacidade de retransmissão em caso de colisões, motivando protocolos MAC que minimizem a probabilidade de colisão através de coordenação temporal, conforme analisado no Capítulo 3.

2.2 Arquitetura LoRaWAN

A arquitetura LoRaWAN baseia-se numa topologia designada por *star-of-stars*. Nesta arquitetura, os dispositivos terminais comunicam diretamente com um ou mais *gateways*, que atuam como pontos de acesso à rede.

Os *gateways* encaminham os pacotes recebidos para um *network server*, responsável pela gestão da rede e pelo encaminhamento das mensagens para os respetivos *application servers*.

A Figura 2.1 ilustra a topologia *star-of-stars* e o fluxo de comunicação entre dispositivos terminais, *gateways* e servidores.

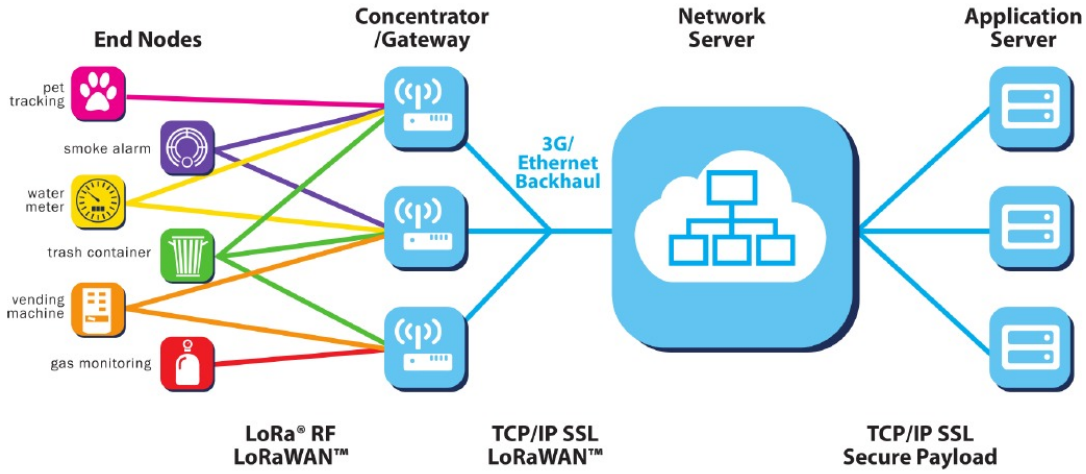


Figura 2.1: Arquitetura LoRaWAN ([3]).

A comunicação entre os nós terminais e os *gateways* é bidirecional, permitindo tanto o envio de dados dos dispositivos para a rede (*uplink*) como a receção de comandos ou dados da rede para os dispositivos (*downlink*). Os *gateways* funcionam como simples pontes de comunicação,

reencaminhando transparentemente os pacotes sem processar o seu conteúdo, sendo toda a lógica de gestão centralizada no *network server*. Em configurações com múltiplos *gateways*, o mesmo pacote *uplink* pode ser recebido e encaminhado por vários deles simultaneamente; o *network server* é responsável por evitar duplicados, retendo apenas a cópia da melhor qualidade de sinal [16]. Na infraestrutura experimental desta dissertação, constituída por um único *gateway*, este mecanismo não é aplicável e cada pacote recebido corresponde a uma entrega única ao servidor (ver Secção 4.1).

2.3 Pilha Protocolar LoRa/LoRaWAN

A tecnologia LoRa pode ser organizada em diferentes camadas funcionais. A camada física é responsável pela modulação e transmissão do sinal rádio, enquanto o protocolo LoRaWAN define as camadas superiores responsáveis pela comunicação entre dispositivos e servidores [13].

A separação entre estas camadas é ilustrada na Figura 2.2.

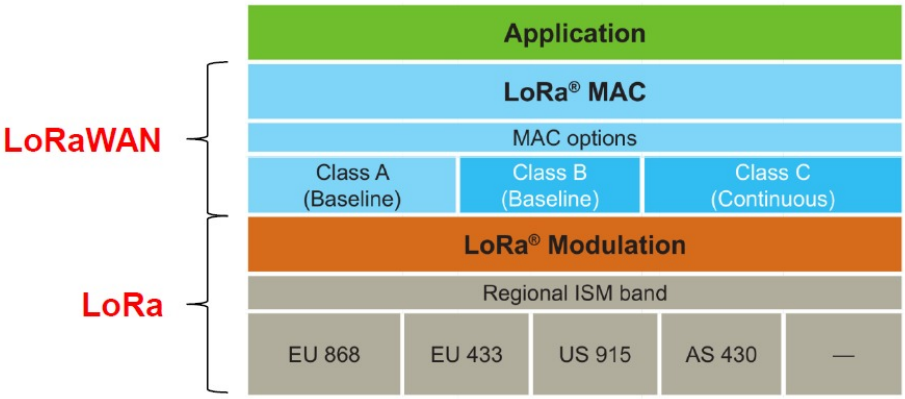


Figura 2.2: Pilha de protocolos LoRa/LoRaWAN ([3]).

A separação entre a camada física LoRa e o protocolo LoRaWAN permite uma clara divisão de responsabilidades. A camada física assegura a transmissão eficiente do sinal rádio, enquanto as camadas superiores gerem aspetos como endereçamento, controlo de acesso ao meio, segurança e gestão da rede. No topo da pilha encontra-se a camada de aplicação, onde são implementadas as lógicas específicas de cada caso de uso.

O protocolo LoRaWAN define três classes de dispositivos (A, B e C) que diferem no compromisso entre latência de *downlink* e consumo energético, analisadas em detalhe na Secção 2.7.

Esta arquitetura modular facilita a evolução independente de cada camada e permite adaptar o comportamento da rede a diferentes requisitos de aplicação, constituindo a base para o desenvolvimento de extensões e otimizações ao protocolo.

2.4 Modulação LoRa

A camada física de LoRa assenta na modulação CSS, cujo comportamento é controlado principalmente pelo SF. Esta secção descreve os princípios da modulação e os compromissos que a escolha do SF implica em termos de alcance, taxa de transmissão e probabilidade de colisão.

2.4.1 Chirp Spread Spectrum

A modulação LoRa baseia-se na técnica CSS. Através desta técnica, a informação é transmitida por meio de sinais cuja frequência varia de forma contínua ao longo do tempo.

Os sinais utilizados na modulação LoRa são denominados como *chirps*. Durante a transmissão de um símbolo, a frequência do sinal varia linearmente ao longo de toda a largura de banda disponível. A informação é codificada através do deslocamento temporal destes *chirps*.

A Figura 2.3 apresenta um espectrograma que ilustra a variação de frequência e o comportamento temporal dos sinais LoRa. É possível observar as linhas diagonais ascendentes características dos *up-chirps*, onde a frequência cresce linearmente desde a frequência mínima até à frequência máxima dentro da largura de banda configurada. À medida que o SF aumenta, a duração do símbolo também aumenta, conferindo maior robustez à transmissão à custa de menor taxa de transmissão de dados.

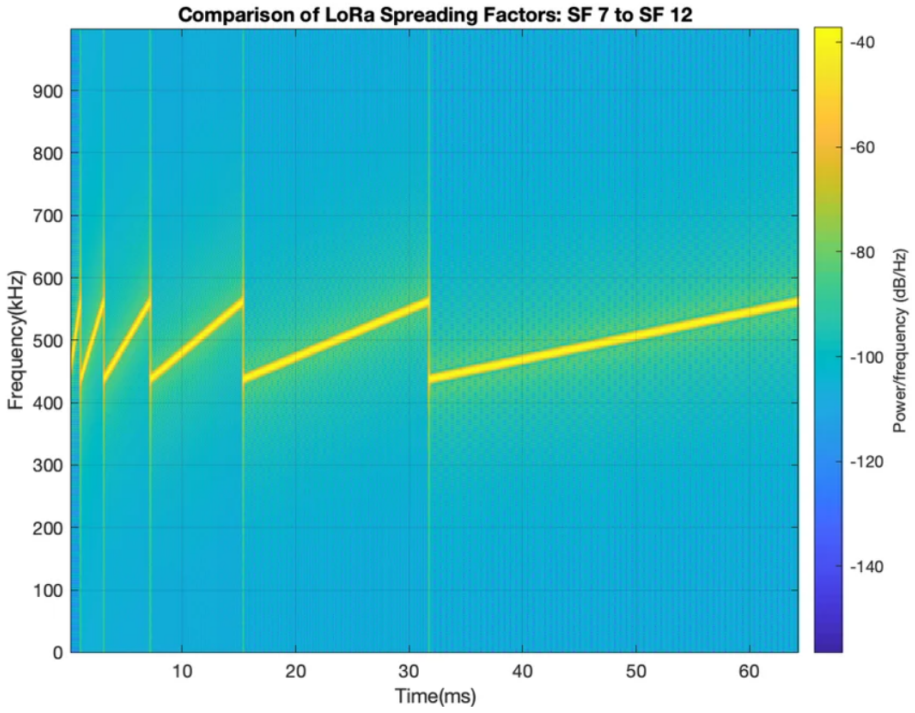


Figura 2.3: Espectrograma de sinais LoRa (SF7 a SF12). [14].

A robustez face a interferência e ruído, a elevada sensibilidade do recetor, a resistência a efeitos de multipercurso e a possibilidade de coexistência de transmissões com diferentes parâmetros de

modulação tornam a modulação CSS particularmente adequada para aplicações IoT em ambientes de comunicação adversos [26].

2.4.2 Spreading Factor

O *Spreading Factor* (SF) é um dos parâmetros fundamentais da modulação LoRa, definindo o número de chips utilizados para representar cada símbolo transmitido. Este parâmetro controla diretamente o compromisso entre taxa de transmissão e alcance da comunicação. Na prática, o SF assume valores inteiros entre 7 e 12.

Valores mais baixos de SF permitem maiores taxas de transmissão e menor duração das transmissões, sendo adequados para comunicações em curtas distâncias ou em condições de canal favoráveis. Por outro lado, valores mais elevados aumentam a sensibilidade do recetor, permitindo comunicações a maiores distâncias ou em ambientes com maior atenuação do sinal, à custa de uma redução significativa da taxa de transmissão.

A taxa de símbolos depende da largura de banda (BW) e do *Spreading Factor*, podendo ser expressa por [26]:

$$R_s = \frac{BW}{2^{SF}} \tag{2.1}$$

onde R_s representa a taxa de símbolos transmitidos.

A taxa de chips da modulação LoRa é dada por:

$$R_c = R_s \cdot 2^{SF} \tag{2.2}$$

onde R_c corresponde à *chip rate*. Substituindo $R_s = BW/2^{SF}$, obtém-se $R_c = BW$, confirmando que a *chip rate* é constante e igual à largura de banda, propriedade fundamental da modulação CSS.

A taxa de bits efetiva depende ainda da taxa de codificação (CR), sendo expressa por:

$$R_b = SF \cdot \frac{BW}{2^{SF}} \cdot CR \tag{2.3}$$

onde R_b representa a taxa de bits da transmissão.

O aumento do *Spreading Factor* reduz a taxa de transmissão e aumenta o tempo de ocupação do canal, conhecido como *Time-on-Air* (ToA). Este parâmetro influencia diretamente a latência da comunicação, o consumo energético dos dispositivos e a probabilidade de colisões entre transmissões.

A Figura 2.4 ilustra o compromisso existente entre alcance de comunicação, taxa de transmissão e consumo energético para diferentes valores de *Spreading Factor*. Observa-se que valores elevados de SF aumentam o alcance e a sensibilidade do recetor, mas reduzem o *bitrate* e aumentam o tempo de transmissão.

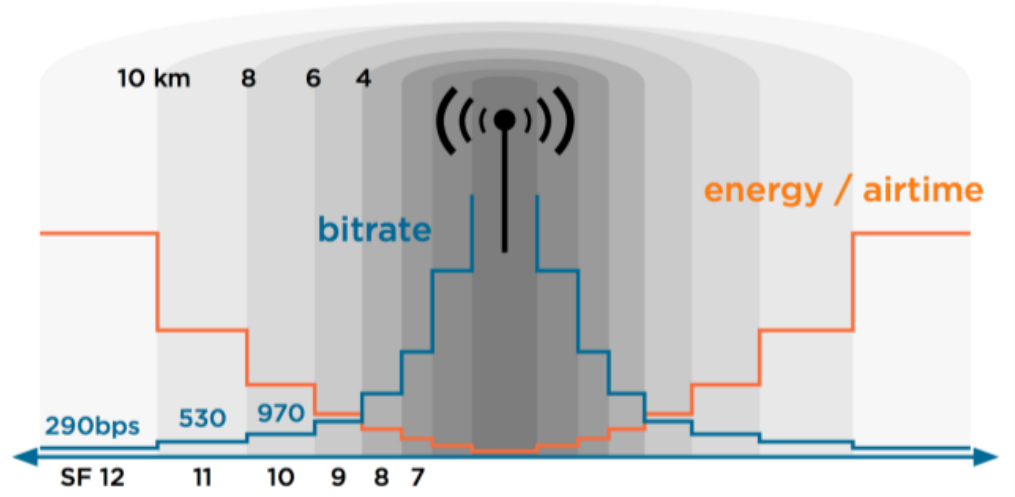


Figura 2.4: Compromisso entre SF, taxa de transmissão e consumo energético ([29]).

A Tabela 2.2 apresenta a relação entre o *Spreading Factor*, a taxa de transmissão e a sensibilidade do recetor.

SF	Bit rate [kbit/s]	Sensibilidade Rx [dBm]
7	5,4	-124
8	3,1	-126
9	1,7	-129
10	0,98	-132
11	0,44	-133
12	0,25	-137

Tabela 2.2: Relação entre *Spreading Factor*, taxa de bits e sensibilidade do recetor [27]

Os valores de sensibilidade são retirados do SX1261/2 DS Rev 2.2 [27]. A progressão exata de 2,5 dB por incremento de SF aplica-se aos limiares mínimos de SNR (Tabela 2.3), a partir dos quais a sensibilidade pode ser estimada. Observa-se que valores mais elevados de SF reduzem significativamente a taxa de transmissão, mas aumentam a sensibilidade do recetor, permitindo comunicações a maiores distâncias. No entanto, este aumento do *ToA* implica maior latência, maior consumo energético e maior probabilidade de colisões. Assim, a escolha do *SF* representa um compromisso entre alcance da comunicação, eficiência energética e capacidade da rede.

Na presente dissertação, o SF é o parâmetro de maior impacto, pois determina o ToA e a probabilidade de colisão. A carga da rede é o segundo fator mais relevante, condicionando diretamente o desempenho dos protocolos MAC avaliados. Os restantes parâmetros como: BW, CR e potência de transmissão, são mantidos fixos em todos os ensaios, conforme descrito no Capítulo 4.

2.4.3 Ortogonalidade entre Spreading Factors

Uma propriedade fundamental da modulação LoRa é a quase-ortogonalidade entre diferentes *Spreading Factors*. Esta característica permite que transmissões simultâneas com SFs distintos coexistam no mesmo canal de frequência sem causar interferência mútua significativa, aumentando substancialmente a capacidade da rede [28].

A ortogonalidade resulta da estrutura temporal dos símbolos: um símbolo transmitido com SF12 tem duração $2^{12}/BW$, enquanto um símbolo SF7 dura apenas $2^{7}/BW$, o que representa uma diferença de $32\times$ no tempo de símbolo. O recetor, por outro lado, sincronizado para um SF específico consegue extrair o sinal desejado mesmo na presença de outros SFs, através de correlação com o padrão de *chirp* correspondente [28].

No entanto, a ortogonalidade não é perfeita. Bor et al. [5] demonstram experimentalmente que transmissões simultâneas com SFs diferentes introduzem interferência residual, particularmente quando a diferença de potência recebida é elevada (*near-far effect*). Croce et al. [9] demonstram que o limiar de SIR abaixo do qual a interferência entre SFs distintos causa perda de pacotes se situa em aproximadamente -16 dB. Esta propriedade de quase-ortogonalidade é explorada pelo mecanismo ADR (Secção 2.5.4) para distribuir dispositivos por diferentes SFs, maximizando a capacidade da rede. Colisões determinísticas ocorrem entre transmissões com o mesmo SF, motivando estratégias de acesso coordenado como as analisadas no Capítulo 3.

2.4.4 Colisões e Capture Effect

Em redes LoRaWAN baseadas em acesso ALOHA, colisões ocorrem quando dois ou mais dispositivos transmitem simultaneamente no mesmo canal e com o mesmo SF. A probabilidade de colisão aumenta com a densidade de dispositivos e com a carga oferecida ao canal. O *capture effect* é um fenómeno físico observado em modulações de espetro expandido onde, em caso de colisão *entre transmissões com o mesmo SF*, o recetor consegue desmodular corretamente o sinal mais forte se a diferença de potência recebida exceder um limiar mínimo. Em LoRa, este limiar situa-se tipicamente entre 6 e 10 dB [5, 31], sendo distinto do limiar de SIR de -16 dB que Croce et al. [9] identificam para interferência entre SFs diferentes (Secção 2.4.3). Este efeito melhora significativamente a taxa de receção em cenários reais comparativamente aos modelos teóricos de colisão, onde qualquer sobreposição temporal resulta em perda de ambos os pacotes.

A presença de *capture effect* implica que dispositivos próximos do *gateway* (sinal forte) têm maior probabilidade de sucesso mesmo em condições de colisão, enquanto dispositivos distantes (sinal fraco) são mais suscetíveis a perdas. Este comportamento contribui para o problema de *near-far*, onde dispositivos próximos dominam o canal em detrimento de dispositivos distantes [5].

Adicionalmente, em colisões parciais, onde apenas parte dos pacotes se sobrepõe temporalmente, a receção pode ainda ser bem-sucedida se o preâmbulo e o cabeçalho não forem afetados, permitindo ao recetor sincronizar e desmodular a parte restante do pacote [5].

Estes fenómenos explicam discrepâncias frequentemente observadas entre desempenho experimental e previsões teóricas baseadas em ALOHA puro. Na infraestrutura experimental desta

dissertação, a diferença de RSSI entre nós atinge aproximadamente 15 dB, valor superior ao limiar típico de *capture effect* (6 a 10 dB), pelo que este fenómeno pode contribuir para a receção bem-sucedida de pacotes em colisão. Esta observação é retomada na análise de resultados (Capítulo 5).

2.5 Métricas e Parâmetros de Desempenho

O desempenho das transmissões LoRa é caracterizado por um conjunto de métricas e parâmetros complementares, descritos nas subsecções seguintes.

2.5.1 RSSI e SNR

O *Received Signal Strength Indicator* (RSSI) indica a potência total do sinal recebido, expressa em dBm. Embora útil para estimar a cobertura e detetar perdas grosseiras de propagação, não distingue entre potência de sinal útil e ruído ou interferência, tornando-o um preditor menos fiável em cenários com níveis de ruído elevados. O *Signal-to-Noise Ratio* (SNR) colmata esta limitação ao representar a relação entre a potência do sinal útil e o ruído presente no canal, expresso em dB. Em modulações convencionais, o limiar mínimo de demodulação corresponde a um SNR positivo; a modulação LoRa distingue-se por permitir operação com SNR negativo, graças ao ganho de processamento da correlação com o *chirp* de referência. A Tabela 2.3 apresenta os limiares mínimos de SNR para cada SF.

SF	SNR mínimo [dB]	Sensibilidade Rx [dBm]
7	$-7{,}5$	-124
8	$-10{,}0$	-126
9	$-12{,}5$	-129
10	$-15{,}0$	-132
11	$-17{,}5$	-133
12	$-20{,}0$	-137

Tabela 2.3: Limiares mínimos de SNR e sensibilidade do recetor por *Spreading Factor*, para BW = 125 kHz [27].

Os limiares decrescem 2,5 dB por incremento de SF, sendo que os valores de sensibilidade correspondem ao transcetor SX1261/2 utilizado na infraestrutura experimental. Para SF12, o recetor consegue desmodular corretamente com SNR de -20 dB, correspondendo a um sinal 100 vezes menos potente do que o ruído. A distinção entre RSSI e SNR é relevante para a interpretação dos resultados desta dissertação. Dois pacotes com RSSI idêntico de -110 dBm podem ter qualidades de ligação muito distintas consoante o nível de ruído local: se o ruído for -120 dBm, o SNR é $+10$ dB e a ligação é robusta; se for de -112 dBm, o SNR é de apenas $+2$ dB e a ligação está próxima do colapso para SF7. Por este motivo, os resultados experimentais privilegiam o SNR como métrica de qualidade do canal (Capítulo 5).

2.5.2 PDR

O *Packet Delivery Ratio* (PDR) corresponde à proporção de pacotes transmitidos recebidos com sucesso pelo *gateway*, calculado como $\text{PDR} = N_{\text{rx}}/N_{\text{tx}}$. O PDR integra o efeito conjunto da qualidade do canal, da carga da rede e da probabilidade de colisões, sendo a métrica de mais alto nível avaliada nesta dissertação.

2.5.3 Time-on-Air e Ocupação do Canal

O ToA representa a duração temporal de uma transmissão LoRa, correspondendo ao intervalo durante o qual o canal rádio permanece ocupado. Este parâmetro é fundamental para avaliar a ocupação do canal, o consumo energético dos dispositivos e a probabilidade de colisões em redes densas.

O ToA depende de múltiplos parâmetros de configuração da modulação LoRa e pode ser calculado através da expressão [25]:

$$T_{\text{ToA}} = T_{\text{preamble}} + T_{\text{payload}} \tag{2.4}$$

onde:

$$T_{\text{preamble}} = \left(n_{\text{preamble}} + 4{,}25\right) \cdot T_s \tag{2.5}$$

$$T_{\text{payload}} = n_{\text{payload}} \cdot T_s \tag{2.6}$$

sendo $T_s = \frac{2^{SF}}{BW}$ o tempo de duração de um símbolo, e n_{payload} o número de símbolos necessários para transportar o *payload*, calculado através de:

$$n_{\text{payload}} = 8 + \max\left(\left\lceil \frac{8PL - 4SF + 28 + 16 \cdot \text{CRC} - 20H}{4(SF - 2DE)} \right\rceil \cdot (4 + CR), 0\right) \tag{2.7}$$

onde PL representa o tamanho do *payload* em bytes; $H = 0$ para cabeçalho explícito (modo padrão) e $H = 1$ para cabeçalho implícito; DE indica a ativação da otimização para baixas taxas de dados, obrigatória para $\text{SF} \geq 11$ com $\text{BW} = 125\,\text{kHz}$ [25]; e $CR \in \{1,2,3,4\}$ é o parâmetro de taxa de codificação (4/5, 4/6, 4/7 e 4/8, respetivamente), de modo que o fator $(4 + CR) \in \{5,6,7,8\}$ representa o número de chips codificados por bloco de quatro bits de dados.

A Tabela 2.4 apresenta valores típicos de ToA para diferentes configurações, assumindo BW=125 kHz, CR=4/5, preâmbulo de 8 símbolos e cabeçalho explícito.

SF	Payload [bytes]	ToA [ms]	Throughput útil [bit/s]
7	10	41,2	1941
7	50	97,5	4101
10	10	288,8	277
10	50	616,4	649
12	10	991,2	81
12	50	2302,0	174

Tabela 2.4: Exemplos de *Time-on-Air* para diferentes configurações [28].

A Figura 2.5 ilustra visualmente este crescimento, evidenciando que cada incremento unitário de SF duplica aproximadamente o ToA, independentemente do tamanho do *payload*.

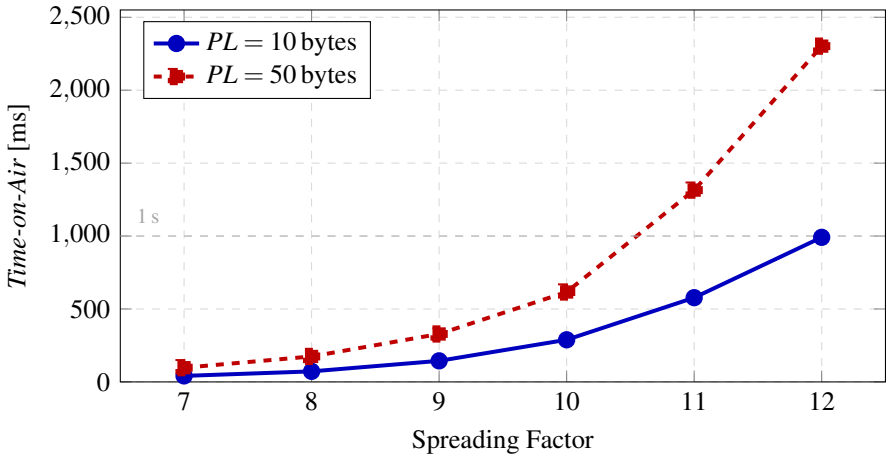


Figura 2.5: *Time-on-Air* em função do *Spreading Factor* para dois tamanhos de *payload* (BW = 125 kHz, CR = 4/5, preâmbulo de 8 símbolos, cabeçalho explícito).

O crescimento é aproximadamente exponencial: cada incremento unitário de SF duplica o ToA, condicionando a capacidade da rede e a latência de comunicação, em particular sob o limite de *duty-cycle* de 1%. Para SF12 e 50 bytes, o ToA atinge aproximadamente 2,3 s, valor que condiciona severamente a capacidade da rede e a latência da comunicação, especialmente sob restrições de *duty-cycle* de 1% (tempo mínimo entre transmissões superior a 230 s).

O ToA condiciona diretamente o consumo energético, pois um tempo de transmissão mais longo esgota mais rapidamente a bateria; a probabilidade de colisão, dado que transmissões mais longas têm maior probabilidade de sobreposição temporal; e a capacidade da rede, na medida em que reduz o número de transmissões possíveis por unidade de tempo.

Este compromisso entre alcance (aumenta com SF) e ocupação do canal (aumenta com ToA) é central na operação de redes LoRaWAN, motivando estratégias de *Adaptive Data Rate* (ADR) e protocolos MAC otimizados como os analisados no Capítulo 3.

2.5.4 Adaptive Data Rate (ADR)

O mecanismo *Adaptive Data Rate* (ADR) é uma funcionalidade opcional do protocolo LoRaWAN que permite ao *network server* otimizar dinamicamente o SF e a potência de transmissão de cada dispositivo em função das condições do canal, com o objetivo de maximizar o débito e minimizar o consumo energético [15]. O algoritmo monitoriza o SNR das últimas 20 transmissões e ajusta o SF em passos de 3 dB de margem disponível, sendo que, se o dispositivo não receber confirmação durante um número configurável de *uplinks* consecutivos, incrementa autonomamente o SF e aumenta a potência de transmissão [16, 17].

A Tabela 2.5 sintetiza as vantagens e limitações do mecanismo.

Vantagens	Limitações
Redução de ToA (SF mais baixos)	Dependente de *downlink* após *uplink*
Menor consumo energético	Convergência lenta (20+ *uplinks*)
Maior capacidade da rede	Instável em dispositivos móveis
Menos colisões (TX mais curtas)	Heterogeneidade de SF dificulta comparação de protocolos

Tabela 2.5: Vantagens e limitações do ADR.

Na presente dissertação, o ADR é desativado em todos os dispositivos terminais, os quais operam com SF e potência fixos ao longo de toda a campanha experimental (Capítulo 4). Esta opção garante um SF idêntico em todos os nós, condição necessária para a comparação rigorosa entre protocolos, e elimina a variabilidade dinâmica que a convergência lenta do algoritmo introduziria nos resultados. Note-se que, com todos os nós em linha de vista a aproximadamente 10 m do *gateway*, o SF7 corresponde já ao SF ótimo, pelo que a desativação do ADR não implica penalização energética adicional.

2.6 Estrutura das Mensagens em LoRaWAN

Cada transmissão LoRaWAN consiste num *PHYPayload* composto por três campos: MHDR (1 byte, tipo de mensagem), MACPayload (informação de controlo e dados da aplicação) e MIC (4 bytes, integridade). O MACPayload subdivide-se em FHDR, FPort e FRMPayload, conforme ilustrado na Figura 2.6.

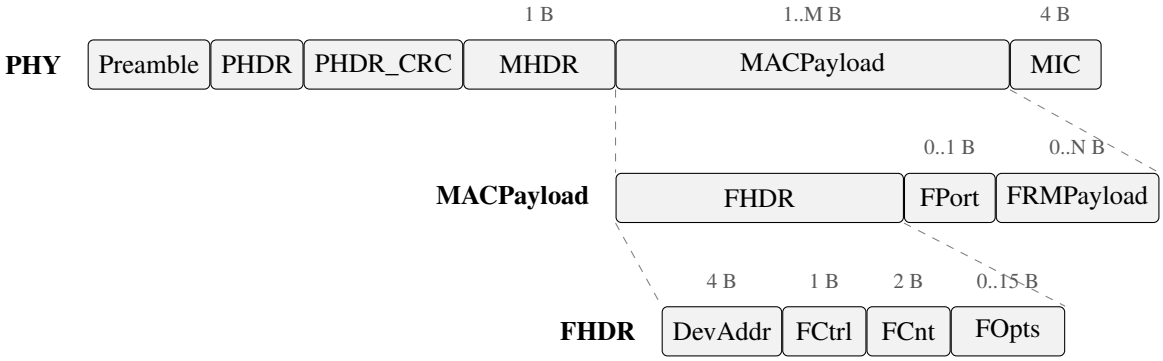


Figura 2.6: Estrutura hierárquica de uma mensagem LoRaWAN ([30]).

O FHDR transporta o endereço do dispositivo (*DevAddr*, 4 bytes), o campo de controlo (*FCtrl*), o contador de *frames* (*FCnt*, base da deteção de perdas e proteção contra repetição [6]) e comandos MAC opcionais (*FOpts*, até 15 bytes). O *FPort* distingue dados de aplicação (valores 1 a 223) de comandos MAC (*FPort* = 0).

As mensagens confirmadas requerem *ACK* explícito nas janelas RX1 ou RX2 seguintes, aumentando a fiabilidade à custa de maior latência [7]. Nesta dissertação utilizam-se exclusivamente mensagens não confirmadas, eliminando o *overhead* de retransmissão e simplificando o cálculo da carga oferecida *G*.

2.7 Classes dos dispositivos terminais

A estrutura das mensagens LoRaWAN, analisada na secção anterior, é comum a todos os dispositivos da rede. No entanto, o comportamento temporal da comunicação, nomeadamente a disponibilidade de janelas de receção para *downlink*, varia significativamente consoante a classe do dispositivo.

O protocolo LoRaWAN define três classes de dispositivos (A, B e C) que diferem no compromisso entre consumo energético e latência de comunicação, determinando quando e como os dispositivos podem receber mensagens da rede.

2.7.1 Classe A

A Classe A constitui o modo de operação base definido pela especificação LoRaWAN e é obrigatória para todos os dispositivos. Esta classe é particularmente adequada para dispositivos alimentados por bateria, uma vez que maximiza a eficiência energética ao manter os dispositivos em modo de baixo consumo durante a maior parte do tempo.

Nos dispositivos Classe A, a comunicação é iniciada pelo próprio dispositivo terminal (*end device*). Quando existe informação a transmitir, o dispositivo envia uma mensagem *uplink* para uma ou mais *gateways*. Após o fim da transmissão, o dispositivo abre duas janelas de receção consecutivas, denominadas *RX1* e *RX2*, durante as quais pode receber mensagens de *downlink*

provenientes do *network server*. Caso não seja recebida qualquer mensagem nessas janelas, o dispositivo regressa ao modo de baixo consumo até ocorrer uma nova transmissão.

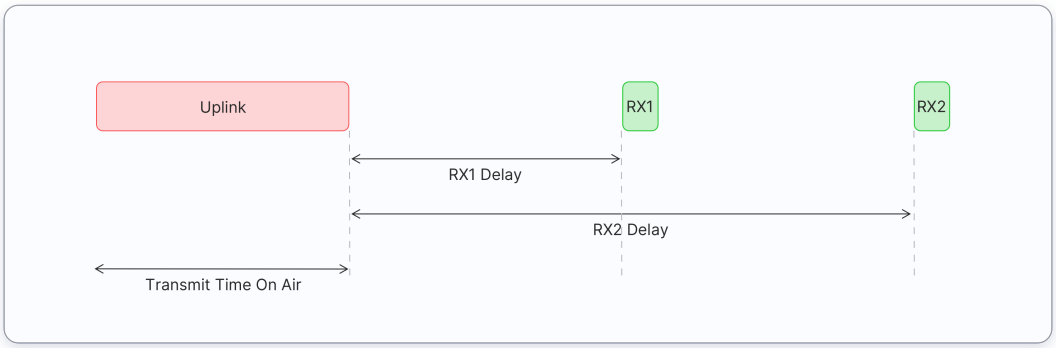


Figura 2.7: Sequência temporal da Classe A ([30]).

A primeira janela de receção (*RX1*) é aberta tipicamente após *RxDelay1* (geralmente 1 s) e utiliza parâmetros derivados do canal do *uplink*. Se não houver receção, abre-se a segunda janela (*RX2*) após *RxDelay2* (tipicamente 2 s), utilizando frequência e *Spreading Factor* predefinidos pela configuração regional. A Figura 2.7 ilustra esta sequência temporal.

Devido a este mecanismo, a disponibilidade de *downlink* depende diretamente da periodicidade das transmissões *uplink*. Esta característica limita a latência e a capacidade de comunicação bidirecional frequente, uma vez que o envio de confirmações (*ACK*) ou comandos da rede apenas é possível após a iniciativa do dispositivo. Em contrapartida, este comportamento permite elevada eficiência energética, tornando a Classe A adequada para aplicações com transmissões esporádicas e tolerantes a latência variável.

2.7.2 Classe B

A Classe B introduz sincronização temporal para melhorar a previsibilidade das comunicações em *downlink*, oferecendo maior consistência e robustez face à Classe A.

Os dispositivos sincronizam-se com *beacons* periódicos enviados pelos *gateways* e abrem janelas de receção adicionais (*ping slots*) em instantes previamente definidos. Isto permite à rede enviar mensagens *downlink* sem depender exclusivamente de transmissões *uplink*, superando uma das principais limitações da Classe A.

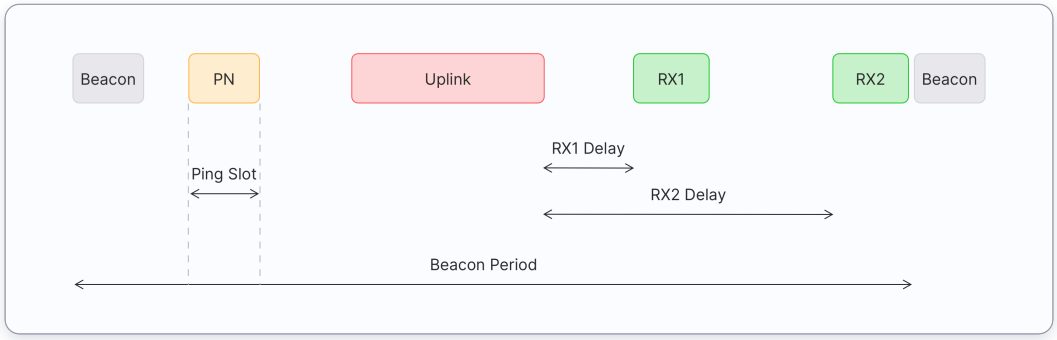


Figura 2.8: Sequência temporal da Classe B com *beacons* e *ping slots* ([30]).

Comparativamente à Classe A, a Classe B reduz a latência de *downlink* ao abrir janelas de receção adicionais em instantes programados por *beacons* periódicos do *gateway*, à custa de maior consumo energético pela ativação periódica do recetor.

Assim, a Classe B estabelece um compromisso direto entre eficiência energética e desempenho temporal: melhora a previsibilidade e reduz a latência de *downlink* face à Classe A, à custa de um aumento do consumo energético devido à sincronização periódica e à abertura de *ping slots*.

Este compromisso torna-a adequada para cenários onde é necessário equilibrar robustez da comunicação e autonomia dos dispositivos, particularmente em aplicações que requerem atuação remota com latência controlada.

2.7.3 Classe C

A Classe C foi concebida para minimizar a latência de *downlink*, oferecendo a maior disponibilidade de comunicação entre todas as classes LoRaWAN.

Neste modo de operação, os dispositivos mantêm o recetor ativo de forma quase contínua, permitindo a receção de mensagens da rede a qualquer instante. A única exceção ocorre durante a transmissão de um *uplink*, após a qual o dispositivo regressa imediatamente ao modo de receção.

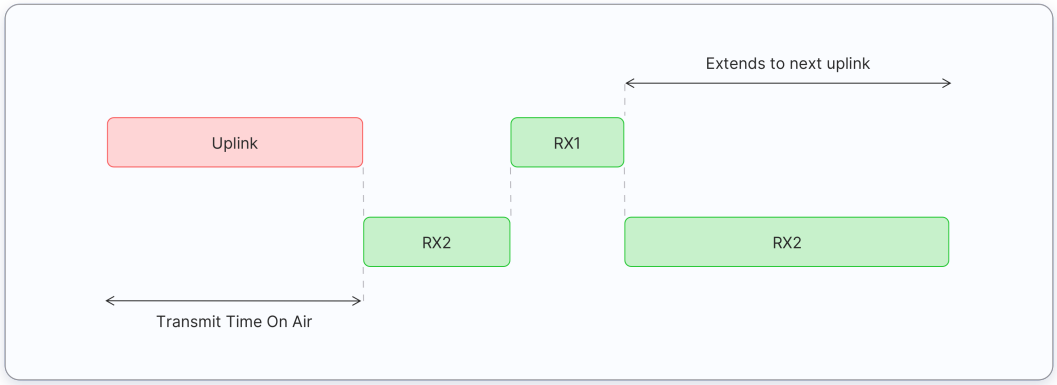


Figura 2.9: Sequência temporal da Classe C com receção quase contínua ([30]).

Comparativamente às Classes A e B, a Classe C elimina a dependência de janelas de receção discretas, permitindo latência mínima e comunicação praticamente imediata em *downlink*. No entanto, este comportamento implica um consumo energético significativamente superior, uma vez que o recetor permanece ativo durante a maior parte do tempo.

A Classe C representa assim o extremo do compromisso entre desempenho temporal e eficiência energética, maximizando a disponibilidade e minimizando a latência de *downlink* à custa de consumo energético significativamente superior, sendo adequada para dispositivos com alimentação contínua ou aplicações críticas que requerem controlo em tempo real, como sistemas de atuação industrial ou monitorização de infraestruturas críticas.

A Tabela 2.6 sintetiza a latência de *downlink* característica de cada classe [17], evidenciando o compromisso entre eficiência energética e disponibilidade de comunicação descendente que motiva a escolha da classe adequada a cada aplicação.

Classe	**Latência downlink**	**Disponibilidade RX**	**Consumo relativo**
A	Variável; depende do próximo *uplink*	Apenas RX1/RX2 pós-*uplink*	Mínimo
B	Limitada; máximo ≈128 s (período do *beacon*)	*Ping slots* periódicos	Médio
C	Quase imediata ($<$ RX2 delay ≈ 2 s)	Contínua (exceto durante TX)	Máximo

Tabela 2.6: Latência de *downlink* e disponibilidade de receção por classe de dispositivo LoRaWAN.

Na presente dissertação, todos os dispositivos operam em Classe A, modo que maximiza a eficiência energética e é suficiente para o protocolo CW-S-ALOHA, cujo *downlink* com *timestamp* de sincronização é tipicamente entregue na janela RX1 imediatamente a seguir a cada *uplink*; em caso de falha de entrega, o nó retoma a sincronização no *uplink* seguinte sem interromper o ciclo de transmissão.

2.8 Segurança e Ativação

A segurança no LoRaWAN assenta em criptografia simétrica AES-128 [16, 17]. Cada dispositivo possui uma chave raiz permanente e um conjunto de chaves de sessão derivadas no momento da ativação, que garantem a integridade e a confidencialidade das mensagens em duas camadas independentes: a camada de rede e a camada de aplicação.

O mecanismo de ativação adotado nesta dissertação é o OTAA (*Over-The-Air Activation*), no qual o dispositivo negoceia as chaves de sessão dinamicamente com o servidor de rede através de uma troca *Join* no início de cada sessão [17]. Esta abordagem é preferida face à ativação por personalização (ABP), que recorre a chaves estáticas gravadas diretamente no dispositivo e é considerada mais vulnerável a ataques de repetição [6].

As versões 1.0.3 e 1.1 do protocolo diferem no número e na função das chaves de sessão geradas, conforme se resume na Tabela 2.7. Na versão 1.0.3, utilizada no protocolo Pure ALOHA, são derivadas duas chaves a partir de uma única chave raiz: uma destinada à integridade das mensagens

de rede e outra à cifra do conteúdo da aplicação. Na versão 1.1, adotada no protocolo CW-S-ALOHA, a chave raiz é dividida em duas chaves distintas, o que permite separar as responsabilidades do operador de rede das do proprietário da aplicação. Esta separação reforça o isolamento entre as duas entidades sem alterações visíveis ao nível do comportamento do dispositivo.

Versão	Chave de sessão	Derivada de	Função
1.0.3	`NwkSKey`	`AppKey`	Integridade MIC e cifra MAC
	`AppSKey`	`AppKey`	Cifra do payload de aplicação
1.1	`FNwkSIntKey`	`NwkKey`	Integridade MIC em uplink
	`SNwkSIntKey`	`NwkKey`	Integridade MIC em downlink
	`NwkSEncKey`	`NwkKey`	Cifra dos comandos MAC
	`AppSKey`	`AppKey`	Cifra do payload de aplicação

Tabela 2.7: Chaves de sessão AES-128 por versão LoRaWAN em modo OTAA.

Para os efeitos desta dissertação, as diferenças criptográficas entre as duas versões não influenciam as métricas avaliadas, nomeadamente o débito útil, a taxa de entrega de pacotes e o consumo energético. A escolha de cada versão resultou de requisitos de compatibilidade com o firmware implementado, conforme detalhado na Secção 4.2.

2.9 Limitações do LoRaWAN e Motivação para Extensões

Apesar das vantagens do LoRaWAN, como o longo alcance e o baixo consumo energético, o protocolo apresenta limitações estruturais que condicionam o seu desempenho em cenários com maiores exigências. O acesso ao meio baseado em *ALOHA* conduz a transmissões não coordenadas, o que aumenta a probabilidade de colisões à medida que a densidade da rede cresce. Em simultâneo, a ausência de sincronização temporal entre dispositivos impede a implementação de mecanismos determinísticos, dificultando a garantia de latência previsível.

Adicionalmente, restrições regulamentares como o *duty-cycle* limitam a frequência de transmissão, reduzindo a capacidade da rede e aumentando os tempos de espera. Na avaliação experimental desta dissertação, os regimes de carga mais elevada implicam uma violação deliberada deste limite, conforme justificado na Secção 4.5.1. A limitada disponibilidade de *downlink*, especialmente na Classe A, reforça esta limitação ao tornar a comunicação dependente de eventos *uplink*. Neste contexto, têm sido propostas diversas extensões ao protocolo com o objetivo de melhorar o desempenho, nomeadamente através da introdução de sincronização temporal e mecanismos de acesso coordenado ao meio.

Entre estas abordagens destacam-se soluções baseadas em TDMA [32] e em acesso aleatório coordenado, que visam reduzir colisões e aumentar a previsibilidade da comunicação. Estas extensões serão analisadas no Capítulo 3, com foco no compromisso entre robustez do protocolo e eficiência energética.

2.10 Sumário

Este capítulo estabeleceu os fundamentos tecnológicos essenciais para a avaliação experimental desenvolvida nas etapas seguintes do trabalho, destacando-se os seguintes aspetos estruturais. A modulação CSS com SF7, BW de 125 kHz e CR 4/5 fixa o ToA em 97,5 ms para *payloads* de 50 bytes, definindo o patamar mínimo de ocupação do canal.

No que concerne ao comportamento em rádio, a quase ortogonalidade mitiga a interferência entre SFs distintos, mas as colisões tornam-se determinísticas quando ocorre sobreposição no mesmo canal e com o mesmo SF. Nestes casos, o *capture effect* assume um papel crucial na recuperação do pacote mais forte, sendo esta uma dinâmica relevante face aos 15 dB de dispersão de RSSI observados na infraestrutura de testes. No plano regulamentar, o limite de *duty-cycle* de 1 % na sub-banda g1 exige um intervalo mínimo de 9,75 s entre transmissões, o que faz com que os regimes de carga moderada e elevada aqui testados constituam verdadeiros ensaios de *stress* laboratorial.

Por fim, o modelo operacional baseia-se estritamente na Classe A com ADR desativado para preservação do perfil de rádio, recorrendo ao procedimento de ativação OTAA sob as especificações LoRaWAN 1.0.3 para o Pure ALOHA e LoRaWAN 1.1 para a variante CW-S-ALOHA.

Capítulo 3

Estado da Arte e Protocolo Proposto

3.1 Introdução e Fundamentos do Acesso Aleatório

O protocolo LoRaWAN, conforme analisado no Capítulo 2, foi concebido para maximizar a eficiência energética e o alcance das comunicações, recorrendo a um mecanismo de acesso ao meio baseado em ALOHA puro. Esta abordagem, embora adequada para aplicações com transmissões esporádicas e tolerantes a latência variável, apresenta limitações significativas em cenários que exigem maior previsibilidade temporal e elevada densidade de dispositivos.

As principais limitações identificadas incluem: elevada probabilidade de colisões à medida que a densidade da rede aumenta, ausência de mecanismos de coordenação temporal entre dispositivos, e disponibilidade limitada de *downlink* na Classe A, dependente da periodicidade das transmissões *uplink*. Adicionalmente, restrições regulamentares como o *duty-cycle* que limitam a cadência de transmissão, agravando o impacto das colisões e comprometendo a fiabilidade da comunicação.

Neste contexto, ao longo das décadas têm sido propostas diversas extensões ao protocolo LoRaWAN com o objetivo de melhorar o desempenho em cenários com requisitos temporais mais exigentes, mantendo as características fundamentais de baixo consumo energético e longo alcance. Entre as abordagens mais relevantes destacam-se soluções baseadas em coordenação temporal através de TDMA com escalonamento centralizado [32], protocolos inspirados em TSCH adaptados à camada física LoRa [12], e mecanismos de escalonamento simples que operam sobre infraestrutura LoRaWAN standard sem modificar os dispositivos terminais [22]. A presente dissertação centra-se nesta última família, que visa reduzir colisões sem comprometer a eficiência energética ou requerer sincronização ao nível do bit.

Este capítulo apresenta os fundamentos teóricos dos protocolos de acesso aleatório utilizados nesta dissertação e o enquadramento do trabalho de referência que lhe serve de base. A presente secção analisa o Pure ALOHA e as suas limitações no contexto LoRaWAN. A Secção 3.2 apresenta o Slotted ALOHA, o S-LoRaWAN de referência, nomeadamente a sua validação experimental e posteriormente a variante CW-S-ALOHA proposta, terminando com a justificação da opção pela Classe A.

3.1.1 Pure ALOHA: Contexto e Fundamentos Teóricos

O protocolo ALOHA, proposto originalmente por Norman Abramson em 1970 no contexto do sistema ALOHANET da Universidade do Havai [1], constitui o mecanismo de acesso ao meio mais simples possível para redes de comunicação por pacotes. O sistema ALOHANET foi desenvolvido para interligar computadores dispersos pelas ilhas havaianas através de canais rádio UHF (*Ultra High Frequency*), enfrentando o desafio fundamental de coordenar o acesso partilhado ao meio sem infraestrutura centralizada ou sincronização entre estações.

A solução proposta por Abramson baseou-se num princípio de extrema simplicidade: cada estação transmite imediatamente quando tem dados disponíveis, sem qualquer verificação prévia do estado do canal ou coordenação com outras estações. Se a transmissão for bem-sucedida (confirmada por *acknowledgment* do destinatário), a estação prossegue para o próximo pacote. Se ocorrer colisão (ausência de confirmação), a estação aguarda um intervalo aleatório antes de retransmitir.

Esta abordagem revelou-se fundamental para o desenvolvimento posterior das redes de pacotes, incluindo Ethernet e protocolos *wireless* modernos [1]. A simplicidade do Pure ALOHA traduz-se em baixíssima complexidade de implementação, ausência de *overhead* de coordenação, e consumo energético mínimo, correspondendo assim a características essenciais para dispositivos IoT alimentados por bateria.

3.1.2 Modelo Analítico do Pure ALOHA

O desempenho do protocolo Pure ALOHA pode ser caracterizado analiticamente através do conceito de *throughput* normalizado S, definido como a fração de tempo durante a qual o canal está a transmitir pacotes com sucesso. Para um canal onde chegam pacotes segundo um processo de Poisson com taxa média G (carga oferecida ao canal, expressa em número médio de pacotes por unidade de tempo normalizada), o *throughput* do Pure ALOHA é dado pela expressão clássica derivada por Abramson [1]:

$$S = G \cdot e^{-2G} \tag{3.1}$$

A derivação baseia-se no conceito de período de vulnerabilidade, assumindo pacotes de duração fixa T. Uma transmissão iniciada em t_0 tem sucesso apenas se nenhuma outra começar no intervalo $[t_0 - T, t_0 + T]$. Este intervalo de duração $2T$ constitui o período de vulnerabilidade, conforme ilustrado na Figura 3.1: qualquer transmissão iniciada até T antes de t_0 ainda estará a decorrer durante a transmissão atual, causando colisão; simetricamente, qualquer transmissão iniciada até T após t_0 colidirá com a transmissão atual que ainda estará em curso.

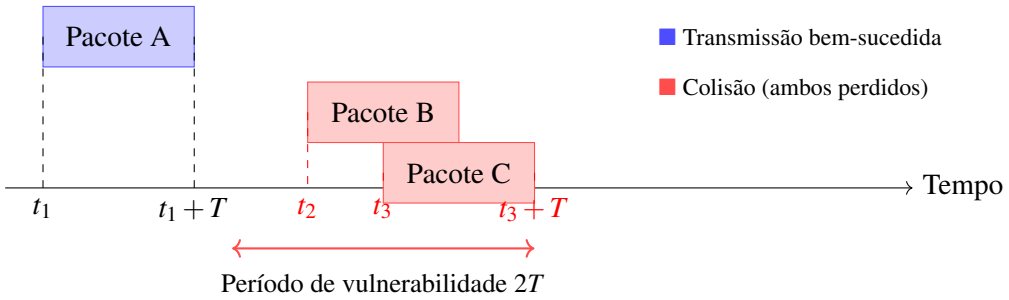


Figura 3.1: Colisões em Pure ALOHA.

Assumindo um processo de chegadas de Poisson com taxa λ (pacotes por segundo), a probabilidade de não ocorrerem chegadas num intervalo de duração $2T$ é dada por $e^{-\lambda \cdot 2T}$. Normalizando o tempo tal que a duração de um pacote seja a unidade ($T = 1$), e definindo $G = \lambda$ como a carga oferecida, obtém-se diretamente a Equação 3.1.

Esta expressão demonstra que o *throughput* máximo teórico do Pure ALOHA é de aproximadamente 18,4%, atingido quando $G = 0{,}5$ (demonstrado através de derivação $\frac{dS}{dG} = 0$):

$$S_{\max} = 0{,}5 \cdot e^{-1} \approx 0{,}184 \tag{3.2}$$

Este valor representa uma limitação fundamental intrínseca ao protocolo: mesmo em condições ideais, mais de 80% da capacidade do canal é desperdiçada devido a colisões ou períodos de inatividade. Para valores de G superiores a 0,5, o *throughput* decresce acentuadamente devido ao aumento exponencial de colisões, caracterizando o fenómeno de instabilidade inerente ao Pure ALOHA.

A Figura 3.2 ilustra esta curva, evidenciando o ponto de operação ótimo em $G = 0{,}5$ e a degradação em regime de sobrecarga. Importa, contudo, enquadrar esta curva no âmbito experimental da presente dissertação. Com apenas quatro nós e sob as restrições de *duty-cycle* discutidas no Capítulo 4, a carga oferecida atingível situa-se na gama $0{,}026 \lesssim G \lesssim 0{,}122$, correspondente ao ascendente inicial da curva. Os ensaios desta dissertação percorrem, portanto, apenas a região de baixa carga, onde o *throughput* cresce de forma aproximadamente linear com G e as colisões são pouco frequentes. O regime de sobrecarga ($G > 0{,}5$), embora central na caracterização teórica do protocolo, não é alcançável com a dimensão de rede disponível, onde o ideal seria o uso de 10 a 24 nós.

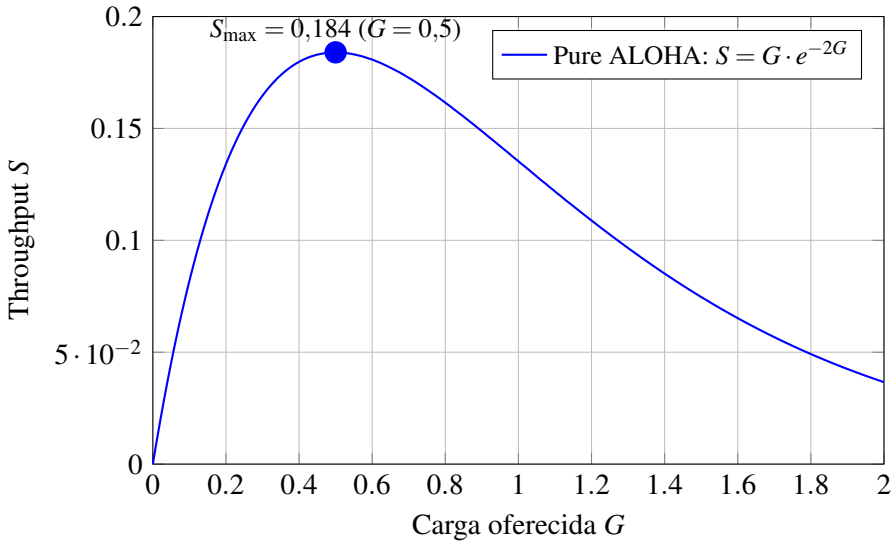


Figura 3.2: Débito normalizado do Pure ALOHA em função da carga oferecida.

3.1.3 Pure ALOHA em Redes LoRaWAN

Em redes LoRaWAN Classe A, o acesso ao meio segue rigorosamente o modelo Pure ALOHA [16]: cada dispositivo transmite quando tem dados disponíveis, sem verificação prévia do estado do canal ou coordenação com outros dispositivos. As janelas de receção RX1 e RX2 que se seguem a cada transmissão *uplink* (descritas na Secção 2.7) permitem ao dispositivo receber confirmações (*ACK*) ou comandos do servidor de rede, mas não alteram o carácter fundamentalmente não coordenado do acesso ao meio.

A aplicação do modelo Pure ALOHA ao contexto LoRaWAN requer adaptações que considerem as características específicas da tecnologia. Em particular, o conceito de colisão adquire nuances importantes devido às propriedades da modulação LoRa descritas no Capítulo 2.

A modulação LoRa oferece quase ortogonalidade entre diferentes *Spreading Factors*, permitindo que transmissões simultâneas com SFs distintos coexistam no mesmo canal com interferência reduzida (Secção 2.4.3). Esta propriedade mitiga parcialmente o impacto de colisões em redes heterogéneas onde dispositivos operam com SFs diferentes. No entanto, em redes homogéneas, onde todos os dispositivos utilizam o mesmo SF, como será o caso na validação experimental desta dissertação, esta vantagem desaparece, e o comportamento aproxima-se do modelo ALOHA clássico.

Conforme analisado na Secção 2.4.4, a modulação LoRa apresenta *capture effect* significativo: quando dois pacotes colidem mas um é recebido com potência substancialmente superior ao outro (tipicamente diferença > 6 dB [5, 31]), o descodificador consegue frequentemente recuperar o pacote mais forte, tratando o mais fraco como ruído de fundo. Este fenómeno melhora o *throughput* efetivo face às previsões do modelo teórico puro, particularmente em cenários onde existe diversidade significativa de distâncias dispositivo-*gateway*. Importa, contudo, notar que este fenómeno é mais relevante em cenários de carga elevada com diversidade de distâncias.

Em regime de baixa carga ($G \ll 1$), o desempenho é dominado pela raridade de colisões. Importa precisar o que se entende por colisão na avaliação do Pure ALOHA. Em LoRaWAN, a perda de um pacote nunca é observada diretamente: é inferida a partir de descontinuidades no contador de *frames* (*FCnt*), atribuído sequencialmente pelo nó a cada transmissão. O servidor regista o último *FCnt* de cada nó e, ao receber um novo *frame*, contabiliza como perdas a diferença que exceda a unidade. Na ausência de qualquer coordenação temporal, e dado que todos os nós partilham o mesmo SF, assume-se que toda a perda assim inferida resulta de colisão. Trata-se de uma métrica de perda global, que não distingue colisão de outras causas como sinal fraco, mas que constitui a única evidência efetiva de que um *frame* não foi entregue.

Ao contrário do modelo ALOHA clássico que assume pacotes de duração fixa, em LoRaWAN o *Time-on-Air* (ToA) varia significativamente em função do *Spreading Factor*, tamanho do *payload*, e taxa de codificação (Secção 2.5.3). Para um *payload* de aplicação de 50 bytes transmitido com SF7, BW=125 kHz e CR=4/5, o ToA é de aproximadamente 97,5 ms. Este valor é utilizado ao longo desta dissertação como referência para o cálculo da carga oferecida G, garantindo consistência na comparação entre Pure ALOHA e CW-S-ALOHA. Importa, contudo, notar que o *frame* LoRaWAN efetivamente transmitido no meio físico inclui cabeçalhos e campos de controlo adicionais, resultando num tempo de ocupação real do canal superior ao considerado no modelo simplificado. Esta diferença não afeta a comparação relativa entre protocolos, mas deve ser tida em conta na interpretação de valores absolutos de carga e ocupação do canal. Ao contrário da formulação histórica com retransmissão, o Pure ALOHA implementado opera em modo *unconfirmed*: não há retransmissão automática e a deteção de perdas é feita exclusivamente por gaps de FCnt. Em modo confirmado, o G inclui retransmissões; em modo *unconfirmed*, G corresponde exatamente à taxa de chegadas λ, sem componente de retransmissão. A curva teórica $S = G \cdot e^{-2G}$ mantém-se válida como referência e, em regime de baixa carga ($G \ll 1$), esta simplificação não altera a previsão de primeira ordem do *throughput*.

A carga oferecida G em redes LoRaWAN pode ser expressa em função do número de dispositivos ativos N, do ToA médio dos pacotes T_{ToA}, e do intervalo médio entre transmissões T_{int}:

$$G = \frac{N \cdot T_{ToA}}{T_{\text{int}}} \tag{3.3}$$

Na análise experimental (Capítulo 5), G é calculado como:

$$G = \frac{\text{tentativas} \times T_{\text{ToA}}}{T_{\text{sessão}}} \tag{3.4}$$

onde as tentativas correspondem ao campo `tx_attempt_count` transportado no *payload*, que contabiliza todas as tentativas de transmissão independentemente do resultado, garantindo compatibilidade com a Equação 3.3.

Esta expressão é fulcral para validação de resultados e permite calcular a carga oferecida em qualquer cenário LoRaWAN, prevendo o *throughput* esperado através da Equação 3.1, assumindo,

no pior caso, que efeitos de captura são negligenciáveis e que a ortogonalidade entre SFs não está disponível, por todos os nós partilharem o mesmo SF.

As limitações do Pure ALOHA tornam-se particularmente evidentes em cenários de IoT industrial ou de infraestruturas críticas. Estes ambientes apresentam frequentemente requisitos que o protocolo não consegue satisfazer de forma adequada. Em primeiro lugar, plantas industriais ou redes de monitorização urbana podem incluir centenas ou milhares de sensores concentrados numa área limitada, resultando em carga oferecida $G \gg 0{,}5$ e *throughput* severamente degradado. Em segundo lugar, aplicações de controlo ou atuação remota requerem garantias de latência máxima, incompatíveis com o comportamento probabilístico e potencialmente instável do Pure ALOHA. Por último, sistemas críticos exigem taxas de entrega de pacotes (PDR) superiores a 95% ou mesmo 99%, valores inalcançáveis em Pure ALOHA sob carga moderada a elevada.

Estas limitações motivam a investigação de extensões ao protocolo LoRaWAN que introduzam coordenação temporal, reduzindo colisões sem comprometer a eficiência energética ou a simplicidade de implementação. A secção seguinte apresenta o Slotted ALOHA e a variante CW-Slotted ALOHA proposta nesta dissertação, que aumentam significativamente o *throughput* máximo através da introdução de estrutura temporal discreta ao acesso ao meio.

3.2 Slotted ALOHA, S-LoRaWAN e Variante CW-S-ALOHA

3.2.1 Slotted ALOHA: Fundamentos

O protocolo Slotted ALOHA [23], proposto por Lawrence G.Roberts, constitui uma evolução natural do Pure ALOHA que introduz estrutura temporal discreta ao acesso ao meio. A modificação fundamental consiste em dividir o tempo em intervalos discretos (*slots*) de duração fixa T, correspondente ao tempo de transmissão de um pacote, e restringir o início de todas as transmissões ao início destes *slots*.

Esta restrição temporal reduz o período de vulnerabilidade de cada transmissão de $2T$ para T, uma vez que colisões apenas podem ocorrer entre transmissões iniciadas no mesmo *slot*. Transmissões em *slots* adjacentes não colidem, eliminando o fator 2 no período de vulnerabilidade. O *throughput* do Slotted ALOHA é dado por:

$$S = G \cdot e^{-G} \tag{3.5}$$

O *throughput* máximo é atingido em $G = 1$, resultando em:

$$S_{\max} = e^{-1} \approx 0{,}368 \tag{3.6}$$

Este valor representa uma duplicação do *throughput* máximo face ao Pure ALOHA (18,4% $\rightarrow$ 36,8%), demonstrando o impacto da coordenação temporal mesmo quando esta se limita à sincronização ao nível do *slot*, não requerendo sincronização ao nível do bit ou do símbolo. A Figura 3.3 apresenta a comparação entre ambos os protocolos.

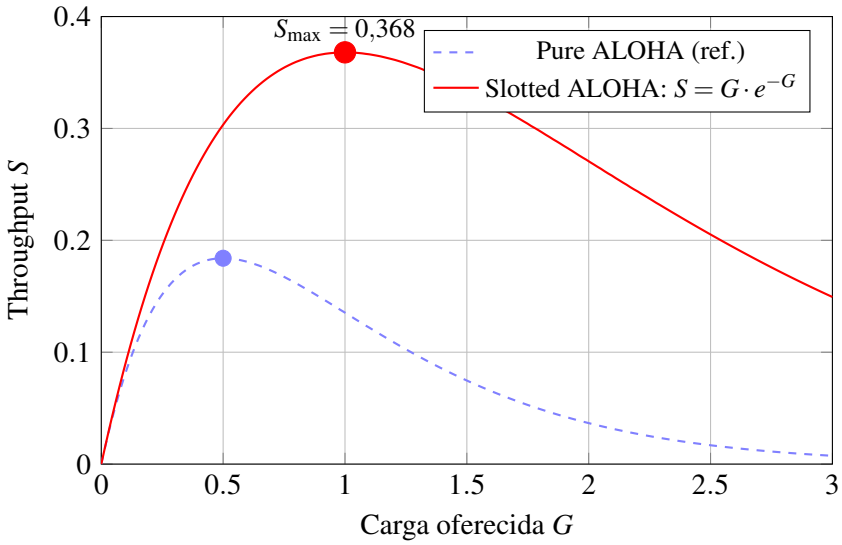


Figura 3.3: Comparação de *throughput* entre Pure ALOHA e Slotted ALOHA em função da carga oferecida G.

3.2.2 S-LoRaWAN (Polonelli et al.)

O S-LoRaWAN, proposto por Polonelli et al. [20], é a primeira adaptação do Slotted ALOHA à especificação LoRaWAN Classe A. O princípio central consiste em dividir o tempo em *slots* discretos sincronizados e restringir o início de cada transmissão ao início do *slot* corrente, reduzindo o período de vulnerabilidade de $2T$ para T. Para manter compatibilidade com LoRaWAN, cada *slot* encapsula o ciclo MAC completo: *uplink*, RX1 Delay obrigatório de 1 s e *downlink* de confirmação. A duração total do *slot* é dada por:

$$T = T_r + T_b \tag{3.7}$$

onde T_r engloba o ToA do *uplink*, o RX1 Delay e o ToA do ACK, e T_b é o intervalo de tolerância para absorver imprecisões de sincronização e deriva temporal. Para a configuração experimental reportada pelos autores, com SF8, BW 125 kHz, CR 4/5 e *payload* de 200 bytes, obtém-se $T_r = 1{,}615$ s. O intervalo de tolerância foi dimensionado em $T_b = 385$ ms, resultando numa duração total de *slot* $T = 2{,}0$ s, com um rácio $T_b/T_r \approx 19\%$ [20]. A Figura 3.4 ilustra a estrutura temporal de um *slot* e os possíveis estados de ocupação.

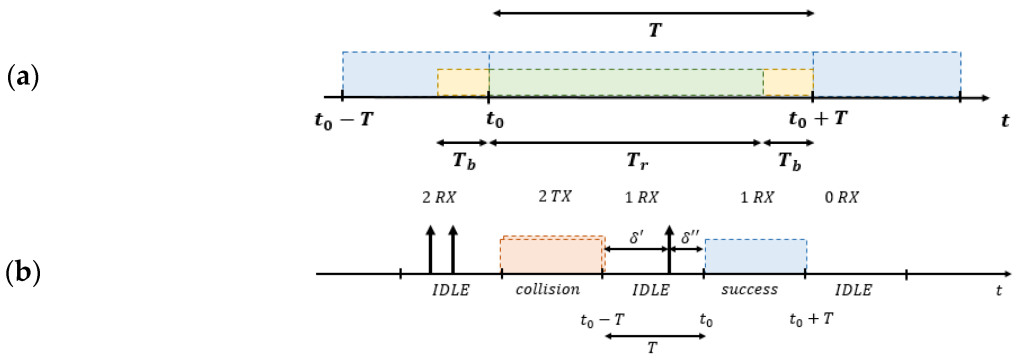


Figura 3.4: Definição da duração do *slot* T, composto pelo tempo de transmissão T_r e pelo intervalo de tolerância T_b, e exemplos de ocupação: *slot* livre, *slot* com colisão e *slot* com transmissão bem-sucedida [20].

A eficácia do S-LoRaWAN depende da capacidade dos dispositivos manterem uma referência temporal comum à estrutura de *slots*. Esta referência é mantida através de relógios internos de baixo consumo, baseados em osciladores de cristal que apresentam desvios (*drift*) típicos na ordem de 20 a 80 ppm. Para um cristal de 80 ppm com um intervalo de ressincronização de 80 minutos, o desvio acumulado é de aproximadamente:

$$\Delta t = 80 \times 10^{-6} \times (80 \times 60) = 384 \text{ ms} \tag{3.8}$$

Este valor justifica o dimensionamento de $T_b = 385$ ms, confirmando a adequação do intervalo de tolerância ao pior caso de deriva considerado pelos autores [20]. Para corrigir periodicamente esta deriva, Polonelli et al. propõem um mecanismo de ressincronização baseado em *timestamps* transmitidos pelo *gateway* através das mensagens de *downlink* da janela RX1, sem introduzir comunicação adicional. O procedimento, ilustrado na Figura 3.5, decorre em quatro passos: o *gateway* marca o instante de fim da receção do pacote *uplink* com precisão de ± 20 μs, limitação do concentrador SX1301; o *timestamp* é incluído no *downlink* enviado na janela RX1, com um *overhead* de apenas 8 bytes; o dispositivo calcula o desvio entre o instante de transmissão registado localmente e o *timestamp* recebido; e atualiza o relógio interno com a referência temporal do *gateway*, corrigindo a deriva acumulada.

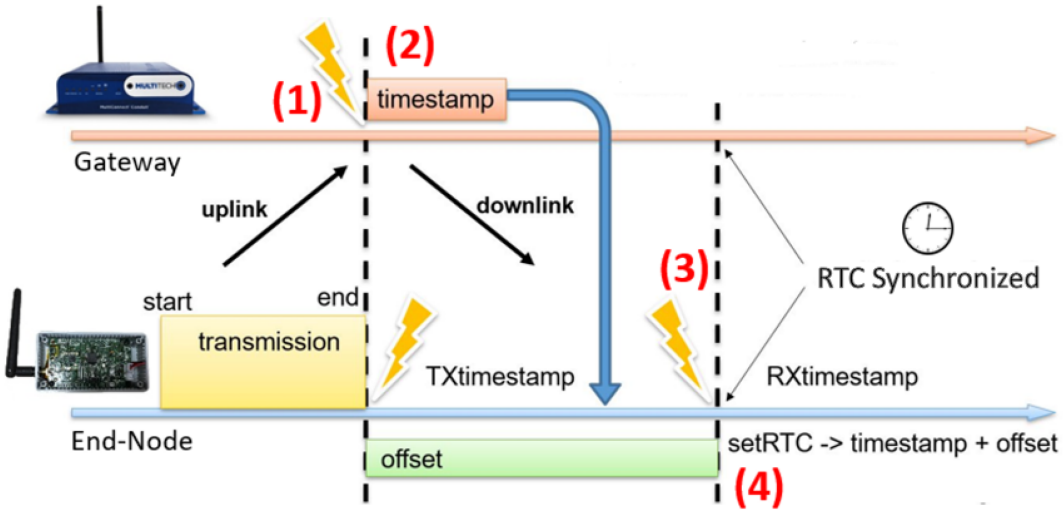


Figura 3.5: Procedimento de sincronização temporal por *timestamps* piggybacked no *downlink*, proposto pelo S-LoRaWAN e adotado no CW-S-ALOHA [20].

Polonelli et al. [20] validaram experimentalmente o S-LoRaWAN em ambiente controlado com 24 nós baseados em microcontrolador STM32L476 e transceptor RFM95W, realizando três ensaios com configurações de tráfego distintas, cujos resultados se apresentam na Tabela 3.1.

	Ensaio 1	Ensaio 2	Ensaio 3
Duração do ensaio [s]	10273	6977	3524
Pacotes enviados	2708	3285	2894
Número de nós	10	18	24
Período de transmissão [s]	15	15	13
PDR [%]	69	55	43
Carga oferecida G	0,264	0,471	0,773
Throughput S	0,184	0,275	0,332

Tabela 3.1: Resultados experimentais do S-LoRaWAN reportados por Polonelli et al. [20].

Os resultados confirmam uma tendência consistente com o modelo teórico do Slotted ALOHA: o *throughput* cresce com a carga oferecida até próximo do máximo teórico, enquanto a taxa de sucesso por pacote decresce com o aumento da contenção. Na comparação direta com o LoRaWAN padrão em condições equivalentes, com 24 nós e período de transmissão de 13 s, o S-LoRaWAN atingiu um *throughput* normalizado de $S = 0{,}332$ face a $S \approx 0{,}057$ do LoRaWAN Classe A, correspondendo a um ganho de $5{,}8\times$ em *throughput* efetivo. O valor superior ao ganho teórico de $2\times$ explica-se pelo facto de o LoRaWAN estar a operar em regime de sobrecarga, onde o Pure ALOHA se encontra já na zona de instabilidade da curva $S(G)$ [20].

A análise de sincronização realizada no ensaio 3 revelou apenas 5 falhas no total dos *slots* transmitidos pelos 24 nós, correspondendo a uma taxa de sucesso de sincronização de 99,6%. O erro máximo observado foi de 37 ms, confirmando o dimensionamento adequado de $T_b = 385$ ms [20].

A validação apresenta, contudo, limitações: o ensaio foi conduzido com infraestrutura dedicada e hardware customizado, não contemplando o comportamento sobre plataformas comerciais de baixo custo nem sobre servidores LoRaWAN de código aberto.

Importa sublinhar que a configuração experimental do S-LoRaWAN difere substancialmente da adotada nesta dissertação. Polonelli et al. operam em SF8 com *payload* de 200 bytes forçado num único canal (canal 6) e em modo confirmado com *backoff* reativo, ao passo que o CW-S-ALOHA aqui proposto usa SF7, *payload* de 50 bytes, múltiplos canais e modo *unconfirmed* com dispersão *a priori*. Esta diferença de parametrização explica a disparidade nos valores absolutos de T_r, G e *throughput* entre os dois trabalhos, e reforça a necessidade da validação independente conduzida no Capítulo 4.

3.2.3 Variante Proposta: CW-S-ALOHA

A presente dissertação propõe uma variante do S-LoRaWAN, designada *Contention Window Slotted ALOHA* (CW-S-ALOHA), que herda o princípio de *slots* discretos sincronizados e o mecanismo de ressincronização por *timestamps*, mas substitui o *backoff* reativo (retransmissão após deteção de colisão) por uma aleatorização *a priori* do *slot* de transmissão.

No instante em que um nó tem dados disponíveis (evento de aplicação), não transmite no *slot* imediatamente seguinte: seleciona aleatoriamente um *slot*-alvo dentro de uma janela de contenção (*Contention Window*, CW), enfileira o pacote, e a transmissão efetiva ocorre quando o relógio sincronizado atinge o *slot* escolhido:

$$\text{slot}_{\text{alvo}} = \text{slot}_{\text{atual}} + 1 + \text{CW}, \quad \text{CW} \in \{0, 1, \ldots, N_{\text{CW}} - 1\} \tag{3.9}$$

onde CW é um inteiro uniformemente distribuído. O *slot*-alvo é sempre, no mínimo, o *slot* seguinte, podendo ser adiado até $N_{\text{CW}} - 1$ *slots* adicionais. A latência de escalonamento distribui-se assim uniformemente entre T e $N_{\text{CW}} \cdot T$, constituindo uma garantia de previsibilidade temporal ausente no Pure ALOHA.

A Figura 3.6 ilustra o mecanismo: o evento de aplicação ocorre num instante arbitrário, o nó sorteia um *slot*-alvo dentro da janela de contenção, e a transmissão efetiva inicia-se apenas na fronteira desse *slot*, após o período de guarda.

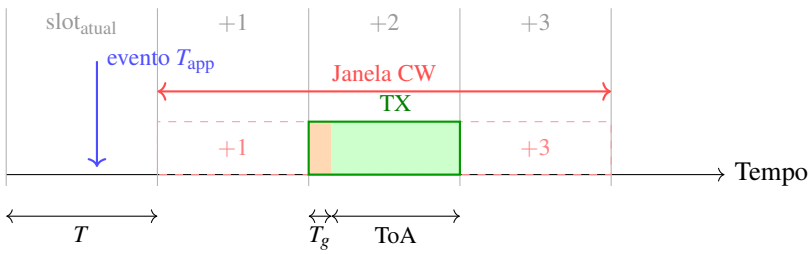


Figura 3.6: Mecanismo temporal do CW-S-ALOHA (*slot*-alvo +2 a título ilustrativo).

A vantagem da dispersão *a priori* é quantificável. Num esquema de *backoff* reativo, N nós que geram tráfego em simultâneo competem primeiro pelo mesmo *slot*, redistribuindo-se apenas após a colisão detetada, ao custo de ter uma ronda perdida e da latência de deteção. O CW-S-ALOHA dispersa as transmissões antes de qualquer contenção, reduzindo a probabilidade de colisão inicial para:

$$P_{\text{col}} = 1 - \left(1 - \frac{1}{N_{\text{CW}}}\right)^{N-1} \tag{3.10}$$

Esta expressão representa o pior caso em que todos os N nós geram um pacote simultaneamente no mesmo ciclo de contenção. No regime de Poisson de baixa carga (operação normal), os eventos de aplicação estão distribuídos no tempo e a probabilidade de colisão por pacote é melhor aproximada por $1 - e^{-G/N_{\text{CW}}}$, consistente com o modelo de *throughput* da Equação 3.11. A expressão evidencia que P_{col} cresce com N e decresce com N_{CW}, motivando o dimensionamento da janela de contenção.

A estrutura em *slots* permite uma classificação de colisões mais rigorosa do que a do Pure ALOHA. À deteção de perdas por descontinuidade no contador de *frames* (*FCnt*), única evidência efetiva de que um *frame* não foi entregue, acrescenta-se a deteção de sobreposição temporal por *slot*. Como os nós operam com SF homogéneo, a única diversidade capaz de separar duas transmissões simultâneas é a frequência, dado que o plano EU868 dispõe de múltiplos canais. Distinguem-se assim dois casos: quando dois nós transmitem no mesmo *slot* e no mesmo canal, ocorre uma colisão real; quando transmitem no mesmo *slot* mas em canais distintos, não há colisão destrutiva. Uma colisão real não implica necessariamente perda, dependendo das condições de SNR, o receptor pode ainda descodificar um dos *frames*. As duas métricas são complementares: o *FCnt* prova a perda, enquanto a sobreposição temporal caracteriza a sua causa.

No CW-S-ALOHA, a ressincronização ocorre a cada transmissão acompanhada de *downlink*, pelo que o intervalo entre sincronizações é limitado pelo período de aplicação T_{app}. Para T_{app} na ordem de segundos a dezenas de segundos, a deriva acumulada entre sincronizações consecutivas é negligenciável face ao período de guarda, ao contrário do S-LoRaWAN onde o pior caso era de 80 minutos. O dimensionamento concreto de T_g e a sua relação com o *jitter* são apresentados no Capítulo 4.

3.2.4 Modelo Analítico e Throughput

Esta subsecção desenvolve o modelo de *throughput* do CW-S-ALOHA, partindo da curva do Slotted ALOHA estabelecida na Secção 3.2.1 ($S = G \cdot e^{-G}$) e aplicando-lhe a penalização introduzida pelo período de guarda. Na literatura, Polonelli et al. [20] propõem uma adaptação do modelo Pure ALOHA que incorpora o *overhead* LoRaWAN em modo confirmado, obtendo um *throughput* máximo efetivo de $\approx 8\%$ para $G = 0{,}25$. A presente dissertação opera em modo *unconfirmed*, pelo que adota o modelo clássico $S = G \cdot e^{-2G}$ como referência para o Pure ALOHA.

Como referido, no CW-S-ALOHA o *slot* é apenas a unidade de escalonamento e não encapsula o ciclo LoRaWAN completo, ao contrário do S-LoRaWAN, onde o *slot* continha o ciclo confirmado

inteiro ($T_r = 1{,}615$ s). Esta distinção permite reduzir substancialmente a duração do *slot* sem violar a especificação LoRaWAN Classe A, transferindo o ciclo rádio para fora da estrutura de *slots*.

A estrutura temporal do *slot* (Figura 4.8) e a condição de não sobreposição entre *slots* adjacentes (Equação 4.8) são detalhadas na Secção 4.4.6 do Capítulo 4.

O CW distribui cada evento de aplicação uniformemente por N_{CW} *slots*, reduzindo a carga efetiva por *slot* para G/N_{CW}. Aplicando o modelo do Slotted ALOHA a cada *slot* e somando os N_{CW} *slots*:

$$S_{\mathrm{CW}} = N_{\mathrm{CW}} \cdot \frac{G}{N_{\mathrm{CW}}} \cdot e^{-G/N_{\mathrm{CW}}} = G \cdot e^{-G/N_{\mathrm{CW}}} \tag{3.11}$$

Esta expressão confirma a vantagem analítica central do CW-S-ALOHA: para o mesmo G, $e^{-G/N_{\mathrm{CW}}} > e^{-G}$, pelo que o *throughput* é sempre superior ao do Slotted ALOHA puro, a dispersão *a priori* reduz diretamente a probabilidade de colisão por *slot*. O fator G está normalizado ao ToA ($G = N \cdot T_{\mathrm{ToA}}/T_{\mathrm{app}}$), pelo que não é necessário um fator de eficiência de guarda adicional.

O débito máximo por *slot* é atingido em $G/N_{\mathrm{CW}} = 1$, ou seja, $G = N_{\mathrm{CW}}$, com valor:

$$S_{\mathrm{CW,max}} = N_{\mathrm{CW}} \cdot e^{-1} \tag{3.12}$$

Para $N_{\mathrm{CW}} = 3$, obtém-se $S_{\mathrm{CW,max}} \approx 1{,}10$. Este valor excede a unidade porque S está aqui normalizado ao ToA e não à duração total do *slot*: a expressão (3.11) contabiliza a fração do tempo de canal ocupada por transmissões bem-sucedidas em todos os N_{CW} *slots* em conjunto, podendo atingir N_{CW} vezes o débito de um *slot* singular. Na prática, a capacidade física do canal impõe $S \leq 1$, limite que é atingido antes de $G = N_{\mathrm{CW}}$ por efeito do *overhead* das janelas Classe A e das restrições de *duty-cycle*. Os ensaios desta dissertação operam em $G \ll N_{\mathrm{CW}}$, zona onde o modelo é válido e $S < 1$ em todos os pontos de operação (ver Tabela 5.8).

Ao impor $T_{\mathrm{app}} \gg T$, o CW-S-ALOHA opera deliberadamente em regime de baixa carga ($G \ll 1$), privilegiando a fiabilidade e a previsibilidade temporal. A Equação 3.11 caracteriza o comportamento na zona ascendente da curva $S(G)$, onde o aumento de carga se traduz diretamente num aumento de *throughput* sem instabilidade. A Tabela 3.2 sintetiza os três protocolos, evidenciando o ganho do Slotted ALOHA face ao Pure ALOHA.

Protocolo	**Modelo**	G^* **(pico)**	S^* **(máximo)**
Pure ALOHA	Ge^{-2G}	0,5	0,184
Slotted ALOHA	Ge^{-G}	1,0	0,368
CW-S-ALOHA ($N_{CW} = 3$)	$Ge^{-G/3}$	3,0	1,104

Tabela 3.2: Comparação de throughput teórico entre Pure ALOHA, Slotted ALOHA e CW-S-ALOHA ($N_{CW} = 3$).

O CW-S-ALOHA supera o Slotted ALOHA puro para qualquer $G > 0$, com o pico teórico de $S_{\mathrm{CW,max}} = N_{CW}\, e^{-1} \approx 1{,}10$ em $G = N_{CW} = 3$ a representar o dobro do máximo do Slotted ALOHA

($e^{-1} \approx 0{,}368$ em $G = 1$). Na zona de operação desta dissertação ($G \leq 0{,}122$), o CW-S-ALOHA produz $S_{\mathrm{CW}} \leq 0{,}11 \ll 1$, confirmando que o sistema opera abaixo da saturação do canal.

3.2.5 Compromisso entre Desempenho e Eficiência Energética

O CW-S-ALOHA introduz três compromissos adicionais face ao Pure ALOHA. Valores elevados de N_{CW} reduzem P_{col} mas aumentam a latência máxima de acesso ao canal, a qual é limitada ao intervalo de T a $N_{\mathrm{CW}} \cdot T$, sendo esta determinística e previsível, ao contrário da latência nula mas sem garantia do Pure ALOHA. Valores reduzidos de T_g maximizam o *throughput* mas aumentam o risco de sobreposição entre *slots* adjacentes. A frequência de ressincronização estabelece um compromisso análogo, o qual é mitigado pelo facto de o mecanismo *piggybacked* manter o *overhead* em apenas 8 bytes por *downlink* [20].

Esta análise revela uma tensão metodológica relevante, uma vez que a região de interesse das curvas $S(G)$, onde as colisões se tornam frequentes, situa-se em cargas elevadas que a restrição de *duty-cycle* de 1 % da regulamentação ETSI torna difíceis de atingir com poucos nós. A resolução desta tensão e as suas implicações para o desenho experimental são discutidas no Capítulo 4.

3.3 Sumário

Este capítulo apresentou os fundamentos teóricos e o estado da arte dos protocolos de acesso aleatório aplicados a redes LoRaWAN. O LoRaWAN Classe A, baseado em Pure ALOHA, limita o *throughput* máximo teórico a 18,4 % por via do período de vulnerabilidade $2T$; em modo *unconfirmed*, adotado nesta dissertação, este valor constitui a referência analítica.

O S-LoRaWAN [20] demonstrou que a introdução de *slots* sincronizados reduz o período de vulnerabilidade de $2T$ para T, duplicando o *throughput* teórico sem modificar a infraestrutura LoRaWAN. A variante CW-S-ALOHA proposta preserva o mecanismo de ressincronização por *timestamps* piggybacked no *downlink* RX1 e substitui o *backoff* reativo por dispersão *a priori* através de uma janela de contenção. A implementação restringe-se à Classe A, cujo ciclo de receção é suficiente para entregar o *timestamp* de sincronização sem recurso a *beacons* de Classe B. Outras abordagens, nomeadamente protocolos com TDMA, como o TS-LoRa [32], ficam fora do âmbito desta dissertação e constituem direções naturais de investigação futura.

A literatura existente sobre S-LoRaWAN baseia-se em hardware dedicado e infraestrutura própria, sem avaliar o protocolo sobre plataformas comerciais de baixo custo nem servidores de rede de código aberto. O Capítulo 4 preenche esta lacuna, descrevendo a infraestrutura utilizada e confrontando os resultados experimentais com os modelos analíticos desenvolvidos neste capítulo em três regimes de carga progressivamente mais elevada.

Capítulo 4

Implementação e Avaliação Experimental

Este capítulo descreve a implementação experimental dos dois protocolos de acesso ao meio avaliados nesta dissertação: Pure ALOHA, que serve de referência, e CW-Slotted ALOHA. A Secção 4.1 descreve a infraestrutura experimental partilhada. A Secção 4.2 documenta a arquitetura de firmware comum. As Secções 4.3 e 4.4 apresentam a implementação de cada protocolo, incluindo o dimensionamento dos parâmetros e o processamento no servidor Flask correspondente. A Secção 4.5 descreve os regimes de carga, as métricas, o procedimento experimental e o ambiente de desenvolvimento utilizado.

4.1 Infraestrutura Experimental

A infraestrutura experimental segue uma topologia em estrela composta por três camadas, conforme ilustrado na Figura 4.1. Na camada de campo, quatro nós terminais DFRobot DFR1195 transmitem *uplinks* LoRa para um *gateway* SenseCAP Indoor Gateway. Na camada de rede, o *gateway* reencaminha os pacotes via UDP para um servidor ChirpStack v4.16.2 a correr numa Raspberry Pi na mesma sub-rede local. Na camada de aplicação, o ChirpStack entrega os eventos de *uplink* ao servidor Flask por *webhook* HTTP. O fluxo descendente difere entre protocolos: no Pure ALOHA, o servidor Flask não enfileira *downlinks* (sendo os MAC commands gerados automaticamente pelo ChirpStack, fora do controlo do servidor); no CW-S-ALOHA, enfileira um *timestamp* de sincronização após cada *uplink* via gRPC para o ChirpStack, que o entrega ao nó na janela RX1. A Tabela 4.1 reúne todos os endereços, identificadores e credenciais da infraestrutura.

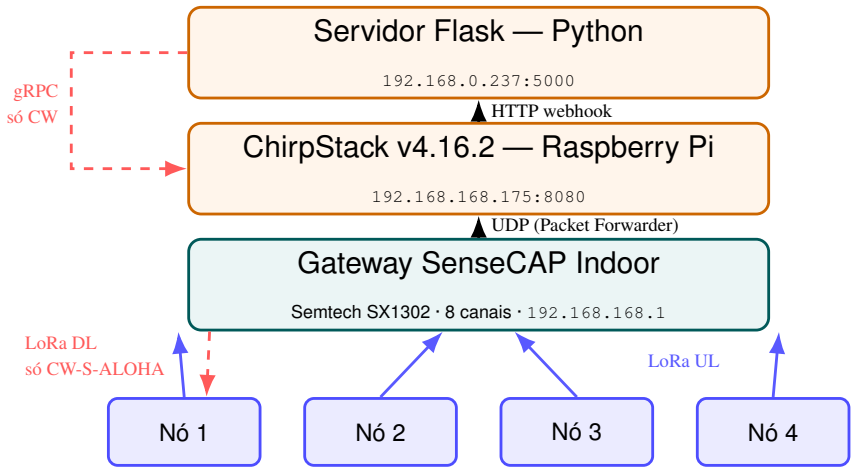


Figura 4.1: Topologia da infraestrutura experimental em estrela.

Componente	Parâmetro	Valor
Gateway SenseCAP	Endereço IP	`192.168.168.1`
	Identificador	`2cf7f1107500016d`
	Packet forwarder	HAL 5.0.1 / concentratord 4.6.0
ChirpStack v4.16.2	Endereço IP	`192.168.168.175`
	Porta gRPC	8080
	DR fixo	DR5 (SF7, BW 125 kHz, ADR desativado)
Servidor Flask	Endereço IP	`192.168.0.237`
	Porta / *endpoint*	5000 / `POST /uplink`
fPorts reservados	Dados de aplicação	fPort = 1
	Sincronização CW-S-ALOHA	fPort = 200
Pure ALOHA	APP_ID ChirpStack	`7bbd73af-7f65-4c7d-bfc8-3ea7e4d12e33`
	LoRaWAN	1.0.3
	Nó 1 DevEUI	`9e70e30d570f93f5`
	Nó 2 DevEUI	`788410651972aca8`
	Nó 3 DevEUI	`dc63358a5a95d62f`
	Nó 4 DevEUI	`1573451f0dff21d1`
CW-S-ALOHA	APP_ID ChirpStack	`c34433fe-5bf6-4c79-8207-2dddfc9d3391`
	LoRaWAN	1.1
	Nó 1 DevEUI	`a64f0db71afc5ccd`
	Nó 2 DevEUI	`cb1c242c24fbc268`
	Nó 3 DevEUI	`00dcaf9e9160b8f5`
	Nó 4 DevEUI	`901c8ecacb49de13`

Tabela 4.1: Configuração da infraestrutura experimental.

4.1.1 Nós Terminais (DFRobot DFR1195)

Os nós terminais são baseados na plataforma DFRobot DFR1195, que integra num único módulo um microcontrolador ESP32-S3 e um transcetor LoRa SX1262. O ESP32-S3 é um SoC de duplo núcleo Xtensa LX7 a 240 MHz com 512 KB de SRAM e gerador de números aleatórios por hardware, utilizado na aleatorização da janela de contenção do CW-S-ALOHA. O ecrã OLED integrado foi mantido inativo durante todos os ensaios para evitar conflitos no barramento SPI partilhado com o transcetor.

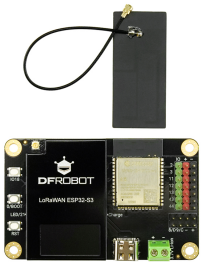


Figura 4.2: Nó terminal DFRobot DFR1195 utilizado nos ensaios ([10]).

O transcetor SX1262 opera na banda EU868 (863–870 MHz), com suporte a transmissão até $+22$ dBm e sensibilidade de receção até -148 dBm. A ligação ao ESP32-S3 é realizada via barramento SPI dedicado (FSPI), com o mapeamento de pinos descrito na Secção 4.2.1. O firmware foi desenvolvido com o *framework* Arduino e a biblioteca RadioLib 6.6.0, que gere a camada física LoRa e o protocolo LoRaWAN. Todos os nós operam em modo Classe A com ativação OTAA. Os parâmetros de comunicação LoRa são fixos em todas as campanhas experimentais e encontram-se resumidos na Tabela 4.2. A escolha de SF7 minimiza o *Time-on-Air* (ToA), reduzindo a probabilidade de colisão e a latência de transmissão. Com estes parâmetros, o ToA de um *payload* de aplicação de 50 bytes é de aproximadamente 97,5 ms (Secção 4.2.4).

Parâmetro	**Valor**
Spreading Factor (SF)	7
Largura de banda (BW)	125 kHz
Taxa de codificação (CR)	4/5
Tamanho do *payload* de aplicação	50 bytes
Plano de frequências	EU868

Tabela 4.2: Parâmetros de comunicação LoRa utilizados em todos os ensaios.

4.1.2 Gateway LoRaWAN (SenseCAP Indoor Gateway)

Figura 4.3: Gateway SenseCAP Indoor Gateway utilizado nos ensaios [24].

O *gateway* utilizado é o Seeed Studio SenseCAP Indoor Gateway [24], um concentrador LoRa de oito canais baseado no chip Semtech SX1302, processador MT7628 a 580 MHz com 128 MB de RAM. O SX1302 suporta receção simultânea nos oito canais EU868 com múltiplos *Spreading Factors*, sendo adequado para cenários com vários nós a transmitir em paralelo. A sensibilidade de receção atinge -125 dBm com SF7 e -139 dBm com SF12 (BW = 125 kHz), e a potência máxima de transmissão é de $+26$ dBm. O *gateway* executa um *packet forwarder* (HAL 5.0.1, concentratord 4.6.0) configurado para o plano EU868, reencaminhando os pacotes recebidos via UDP para o servidor de rede. Os endereços e identificadores do *gateway* encontram-se na Tabela 4.1.

O servidor de rede ChirpStack v4.16.2 corre numa Raspberry Pi distinta na mesma sub-rede local do *gateway*. No contexto desta dissertação, o ChirpStack desempenha três funções: gestão do processo de ativação OTAA, incluindo validação das mensagens *Join Request* e geração das chaves de sessão; desencriptação e reencaminhamento dos pacotes de *uplink* para o servidor Flask via *webhook* HTTP; e receção e entrega de mensagens de *downlink* enfileiradas pelo servidor de aplicação via gRPC. Os perfis de dispositivo no ChirpStack foram configurados com ADR desativado e DR fixo em 5 (SF7, BW 125 kHz), garantindo que todos os nós operam com o mesmo SF ao longo de toda a campanha.

4.1.3 Servidor de Aplicação Flask

O servidor de aplicação é desenvolvido em Python com o *framework* Flask, num ambiente Visual Studio Code, e executado no endereço `192.168.0.237`, porta 5000. Em ambos os protocolos, o servidor expõe o *endpoint* `POST /uplink`, que recebe os eventos reencaminhados pelo ChirpStack via *webhook* HTTP, e exporta no final de cada sessão os dados recolhidos para ficheiros CSV destinados a análise posterior. A estrutura interna do servidor varia consoante o protocolo: o Pure ALOHA utiliza um único ficheiro `server.py`, enquanto o CW-S-ALOHA é composto pelos módulos `server.py` (ponto de entrada), `cw_aloha.py` (`CWAlohaManager`),

`gateway_api.py` (comunicação gRPC) e `config.py`. O detalhe do processamento de cada servidor é apresentado nas Secções 4.3.5 e 4.4.8, junto da implementação do respetivo protocolo.

4.1.4 Instrumento de Medição de Energia

O consumo energético dos nós é medido com o Monsoon High Voltage Power Monitor (HVPM) [18], referência AAA10F, controlado pelo software PowerTool 5.0.0.25 [19] (HW rev G, FW ver 32). O equipamento funciona simultaneamente como fonte de alimentação regulada e como instrumento de medição de corrente de alta precisão: substitui a alimentação USB do nó, fornece uma tensão de saída regulável entre 0,8 e 13,5 V com corrente contínua até 6 A, e amostra a corrente consumida a uma taxa fixa de 5000 amostras por segundo com resolução de 0,9 μA na gama de baixa corrente (0 a 60 mA) e de 125 μA na gama de alta corrente (60 mA a 6 A). A precisão é de $\pm 1\%$ ou ± 50 μA (o que for maior) na gama fina, e de $\pm 1\%$ ou ± 1 mA na gama grosseira [19]. Cada amostra é integrada sobre um período de 200 μs, permitindo capturar corretamente picos de corrente de curta duração típicos do ciclo de transmissão LoRa. O equipamento é auto-calibrado ciclicamente: um ciclo em cada 1000 é reservado a calibração de zero ou de referência, compensando variações de temperatura ao longo da sessão de medição [19].

Figura 4.4: Monsoon High Voltage Power Monitor (HVPM) utilizado nas sessões de medição de energia [18].

Nas sessões de medição desta dissertação, o nó é alimentado através do canal USB do HVPM, com tensão medida de $V_{\text{USB}} = 4{,}97$ V, valor típico de uma porta USB 2.0. Esta configuração mede o consumo real do nó em condições de alimentação USB, sem necessidade de modificação da bateria do dispositivo. As formas de onda de corrente, os valores estatísticos (corrente média, potência média, energia consumida e autonomia estimada) e os dados de amostragem são visualizados em tempo real no PowerTool e exportados em formato CSV para análise posterior no Microsoft Excel.

4.2 Arquitetura de Firmware Comum

Os dois protocolos partilham a mesma fundação de firmware, desenvolvida em *framework* Arduino sobre a biblioteca RadioLib 6.6.0. Esta secção documenta os componentes comuns e as particularidades de cada protocolo são descritas nas secções respetivas.

4.2.1 Plataforma e Configuração de Hardware

O transcetor SX1262 é ligado ao ESP32-S3 através de um barramento SPI dedicado (FSPI), instanciado em modo exclusivo para evitar conflitos com o display OLED:

```
SPIClass SPI_LORA(FSPI);
SX1262 radio = new Module(10, 4, 41, 40, SPI_LORA);
// CS=10, DIO1=4, RST=41, BUSY=40
```

Listing 1: Inicialização e configuração do transcetor RF e do nó LoRaWAN.

O barramento FSPI é iniciado com o mapeamento SCK=GPIO7, MISO=GPIO5, MOSI=GPIO6, CS=GPIO10. Na inicialização, o pino GPIO42 (RXEN) é colocado em estado lógico alto para habilitar o caminho de receção do SX1262, sem esta configuração, as janelas RX1 e RX2 falham silenciosamente.

4.2.2 Períodos de Transmissão e Regimes de Carga

Ambos os protocolos operam com períodos de transmissão organizados entre os quatro nós, reduzindo a probabilidade de bloqueio de fase persistente entre ciclos. Os valores base correspondem ao Regime 1 (baixa carga, conforme ETSI); os Regimes 2 e 3 utilizam períodos progressivamente mais curtos, mantendo a mesma proporção relativa entre nós.

Regime	G_{alvo}	N1 (s)	N2 (s)	N3 (s)	N4 (s)
T1 / C1	$\approx 0{,}026$	14,0	14,5	15,0	15,5
T2 / C2	$\approx 0{,}072$	4,8	5,2	5,6	6,0
T3 / C3	$\approx 0{,}122$	2,8	3,05	3,30	3,55

Tabela 4.3: Períodos nominais por nó e por regime de carga, comuns ao Pure ALOHA (T1–T3) e ao CW-S-ALOHA (C1–C3).

Os períodos do Pure ALOHA e do CW-S-ALOHA são ajustados para igualar a carga efetiva G em cada regime, isolando assim o efeito do mecanismo de acesso ao meio. O escalonamento entre nós distribui o tráfego ao longo do tempo sem eliminar completamente a possibilidade de sobreposição, representando condições realistas de operação.

4.2.3 Ativação OTAA

A ativação na rede é realizada por OTAA, a sequência de arranque executa `node.clearSession()` seguido de `node.beginOTAA()`, e tenta `node.activateOTAA()` em ciclo com intervalos de 5 segundos.

Após *join* bem-sucedido, o ADR é imediatamente desativado e a taxa de dados fixada em DR5 (SF7, BW 125 kHz). As credenciais de cada nó (`JoinEUI`, `DevEUI`, `AppKey`) estão isoladas em ficheiros `credentials_nodeN.h`, permitindo compilar o mesmo código para os quatro nós alterando apenas o ficheiro incluído. Com SF7 fixo e ToA de 97,5 ms, o tempo mínimo entre transmissões imposto pelo *duty-cycle* de 1% é de aproximadamente 9,75 s por nó, e os *uplinks* utilizam exclusivamente os oito canais EU868 padrão (867,1–868,5 MHz).

```
node.setADR(false);
node.setDatarate(5);  // DR5 = SF7 BW125
```

Listing 2: Desativação do ADR e fixação de DR5 (SF7, BW 125 kHz).

A desativação do ADR é essencial para garantir SF idêntico em todos os nós durante toda a campanha. Em testes preliminares sem esta medida, o ChirpStack alterava dinamicamente o SF dos nós para valores superiores, invalidando a comparação de métricas como o ToA e *G*.

A recuperação de falhas em tempo de execução difere entre os dois protocolos. O Pure ALOHA implementa um mecanismo de *rejoin* automático após três falhas consecutivas de transmissão, descrito na Secção 4.3.4. O CW-S-ALOHA não implementa *rejoin* automático: uma falha de `sendReceive()` é registada e o ciclo retoma no evento de *slot* seguinte, sem interrupção do temporizador de aplicação.

4.2.4 Determinação do Time-on-Air

O ToA é determinístico para a configuração fixa do protocolo. Para SF7, BW 125 kHz, CR 4/5 e 50 bytes de *payload* de aplicação, o valor analítico é:

$$T_{\text{ToA}} = 12{,}544 + 84{,}992 = 97{,}536 \text{ ms} \tag{4.1}$$

O ToA poderia ser medido em tempo-real via `radio.getTimeOnAir()`, mas essa abordagem apresenta dois problemas práticos neste contexto. Em primeiro lugar, o momento de leitura é incorreto: quando invocado após `sendReceive()`, o rádio está internamente configurado para a janela RX2 (SF12, BW 125 kHz), pelo que `getTimeOnAir()` retorna ≈2300 ms em vez dos ≈97 ms do *uplink* SF7, o que representa um erro de um fator 24. Chamar a função antes do envio obrigaria a reconfigurar manualmente os parâmetros do *uplink*, eliminando qualquer vantagem da medição dinâmica. Em segundo lugar, os parâmetros são fixos, tornando a medição redundante. O ToA é uma função determinística de SF, BW, CR e comprimento de *payload*, sendo todos estes elementos constantes nesta implementação.

A fórmula LoRa garante que o resultado é sempre 97,536 ms, independentemente do nó ou da sessão, pelo que medir algo determinístico introduziria variabilidade de leitura sem ganho de precisão. Por estes motivos, o ToA é fixado analiticamente. O *firmware* arredonda este valor para `toa_real_ms = 97` ms, transmitido como campo de telemetria no *payload*, e o servidor utiliza 97,5 ms nas equações de G e S para maior precisão, ignorando o campo de telemetria para este cálculo.

4.3 Pure ALOHA

O Pure ALOHA serve de referência nesta dissertação: implementa o acesso ao meio sem qualquer coordenação temporal, sincronização ou mecanismo de confirmação. O firmware opera sobre a especificação LoRaWAN 1.0.3, onde o `DevNonce` é gerado aleatoriamente a cada tentativa de *join*. Esta aleatoriedade implica que ciclos frequentes podem gerar `DevNonce` repetidos, resolvidos pelo mecanismo de *flush* descrito na Secção 4.3.5. A implementação é deliberadamente minimalista e serve de linha de base para avaliar o ganho introduzido pelo CW-S-ALOHA.

4.3.1 Mecanismo de Acesso

O protocolo opera em modo *fire-and-forget*: cada nó transmite assim que o temporizador de aplicação expira, sem verificar o canal, sem janela de contenção e sem retransmissão automática. O *uplink* é enviado em modo *unconfirmed*, mas a chamada `sendReceive()` é utilizada para respeitar o ciclo Classe A, abrindo as janelas obrigatórias RX1 e RX2. O valor de retorno segue a convenção do RadioLib, onde `state == 0` indica transmissão concluída sem *downlink* de aplicação e `RADIOLIB_ERR_RX_TIMEOUT` indica que nenhuma janela recebeu tráfego. Ambos são tratados como ciclos válidos, pelo que o limiar de falha `tx_failed = (state < -1199)` exclui os *timeouts* normais de receção.

A duração de `sendReceive()` varia entre 1300 ms e 2900 ms: ciclos sem *downlink* esgotam ambas as janelas (cerca de 2027 ms), enquanto ciclos com comandos MAC do ChirpStack retornam logo após a RX1 (cerca de 1200 ms pós-transmissão). A métrica `sr_elevado` (`sr` superior a 2000 ms) distingue estes dois regimes, indicando que a maioria dos ciclos completou ambas as janelas sem receber tráfego, o que confirma que a ausência de *downlinks* de aplicação (`rx_success = 0`) é o comportamento esperado e não uma falha de rádio.

4.3.2 Ciclo de Firmware

O firmware é significativamente mais simples que o do CW-S-ALOHA: não existe fila, não existe gestão de *slots* nem sincronização por *downlink*. O percurso completo do firmware está representado na Figura 4.5. Após o arranque, o nó executa a ativação OTAA em ciclo até obter uma sessão válida, seguindo-se a configuração de desativação do ADR e do SF, iniciando-se depois o ciclo de transmissão permanente.

Nesse ciclo, cada iteração constrói o *payload*, invoca `sendReceive()` e segue um de dois ramos: em caso de sucesso, contabiliza a transmissão e aguarda o período de espera em modo `DEBUG_MODE` (ou de poupança de energia) ou em caso de falha, incrementa o contador de tentativas, acionando o mecanismo de recuperação descrito na Secção 4.3.4. O ciclo principal resume-se a:

```
void loop() {
    uint32_t now_ms = millis();
    if (last_tx_ms > 0)
        interval_real_ms = now_ms - last_tx_ms;
    tx_attempt_count++;
    build_payload();
    uint32_t tx_start_ms = millis();
    int16_t state = node.sendReceive(payload, PAYLOAD_SIZE, 1, false);
    uint32_t sendreceive_ms = millis() - tx_start_ms;
    if (tx_failed || sendreceive_ms > 8000) {   // falha ou timeout
        retry_count++;
        if (retry_count >= 3) rejoin();          // rejoin apos 3 falhas
        delay(TX_PERIOD_MS);
        return;
    }
    last_tx_ms = tx_start_ms;   // referencia para interval_real_ms
    tx_count++;
    retry_count = 0;
#ifdef DEBUG_MODE
    delay(TX_PERIOD_MS);
#else
    esp_sleep_enable_timer_wakeup((uint64_t)TX_PERIOD_MS * 1000ULL);
    esp_light_sleep_start();
#endif
}
```

Listing 3: Ciclo principal do Pure ALOHA (`pure_aloha_node.ino`).

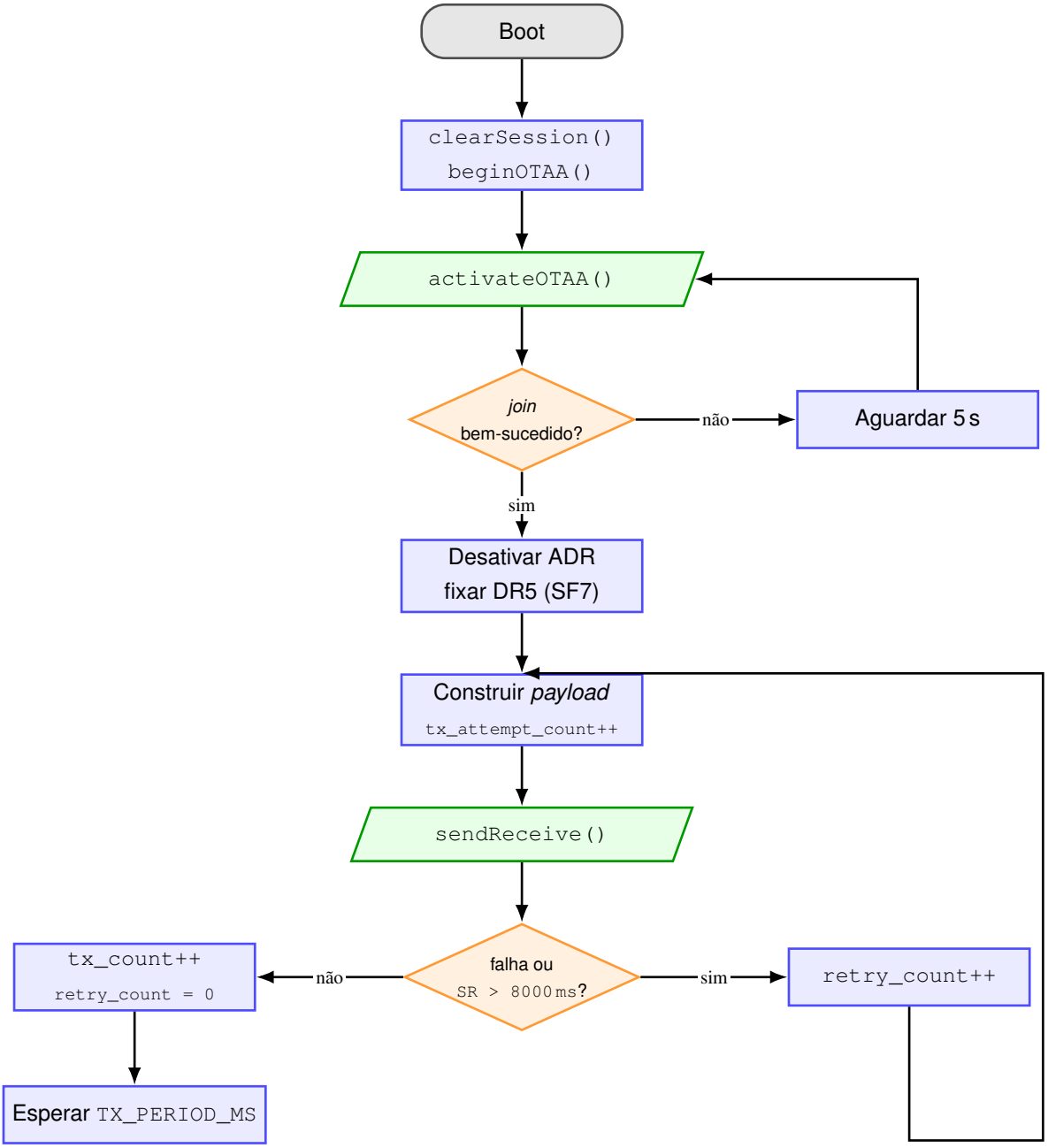


Figura 4.5: Fluxograma do firmware do Pure ALOHA, do arranque ao ciclo de transmissão permanente. O mecanismo de *rejoin* está detalhado na Secção 4.3.4.

Para que o período efetivo de transmissão permaneça próximo de `TX_PERIOD_MS` independentemente da duração de `sendReceive()`, o firmware subtrai o tempo real de execução antes de aguardar:

```
int32_t wait_ms = (int32_t)TX_PERIOD_MS - (int32_t)sendreceive_ms;
if (wait_ms < 200) wait_ms = 200;
```

Desta forma, $T_{\text{real}} \approx$ `TX_PERIOD_MS` quer `sendReceive()` demore 1300 ms (MAC em RX1) quer demore 2027 ms (sem *downlink*). Sem esta compensação, T_{real} seria `TX_PERIOD_MS` mais

`sendreceive_ms`, inflacionando G calculado em cerca de 14%. A guarda de 200 ms é o único caso em que a compensação falha: quando `sendreceive_ms` excede `TX_PERIOD_MS` − 200 ms, o período efetivo fica aquém do nominal. A base temporal utilizada é `millis()`, visto que não é necessária a função `micros64()` utilizada pelo CW-S-ALOHA, uma vez que o Pure ALOHA não requer resolução de microssegundos nem sincronização de *slots*.

4.3.3 Estrutura do Payload

O *payload* de aplicação tem dimensão fixa de 50 bytes (`PAYLOAD_SIZE = 50`), construído explicitamente com `memcpy()` antes de cada chamada a `sendReceive()`. A Tabela 4.4 detalha a sua estrutura campo a campo, indicando para cada um o *offset*, o tamanho em *bytes* e o significado. Os campos transportam telemetria que o servidor usa para reconstruir as métricas: contadores de transmissão (`tx_count`, `tx_attempt_count`), tempos medidos pelo nó (`interval_real_ms`, `sendreceive_ms`, `node_millis`) e o ToA fixo, terminando num bloco de preenchimento que perfaz os 50 bytes.

Offset	Tamanho	Campo	Descrição
0	1 B	`node_id`	Identificador do nó (1–4)
1–4	4 B	`tx_count`	Transmissões bem-sucedidas (sem falha de RF)
5–8	4 B	`rx_expected`	*Downlinks* esperados (= `tx_count` em cada TX)
9–12	4 B	`rx_success`	*Downlinks* recebidos (estruturalmente 0 — sem DL)
13–16	4 B	`toa_real_ms`	ToA fixo: 97 ms (SF7; `getTimeOnAir()` desativado)
17–20	4 B	`interval_real_ms`	Intervalo entre TXs medido pelo nó (ms)
21–24	4 B	`airtime_total_ms`	Tempo de ar acumulado na sessão (ms)
25	1 B	`retry_count`	Falhas consecutivas (reset em 3 → rejoin)
26–29	4 B	`tx_attempt_count`	Total de tentativas (base do cálculo de G)
30–33	4 B	`node_millis`	Relógio do nó no instante do TX (ms)
34–37	4 B	`sendreceive_ms`	Duração do `sendReceive()` anterior (ms)
38–49	12 B	`padding`	Preenchimento fixo (`0x00`)

Tabela 4.4: Estrutura do *payload* do Pure ALOHA (50 bytes, *little-endian*).

Dois campos merecem atenção particular, sendo que o campo `tx_attempt_count` é incrementado antes de cada chamada a `sendReceive()`, contabilizando todas as tentativas de transmissão independentemente do resultado, o que constitui a base do cálculo de G medido no servidor. Por sua vez, o campo `sendreceive_ms` regista a duração da chamada anterior, permitindo ao servidor cruzar o intervalo de *gateway* com o intervalo de aplicação e detetar ciclos anómalos. Distinguem-se dois limiares com propósitos diferentes: uma duração superior a 2000 ms é marcada

como SR elevado, um aviso de atividade anómala em janela RX; uma duração superior a 8000 ms é tratada como falha de transmissão, disparando o mecanismo de *retry* (Secção 4.3.4). O valor `toa_real_ms = 97` é definido por código (linha 90 do `.ino`) e não lido do transcetor. Após `sendReceive()`, a função `radio.getTimeOnAir()` retorna o SF da janela RX2 (SF12) em vez do SF do *uplink*, produzindo valores incorretos (≈2300 ms); o valor analítico fixo evita este problema.

4.3.4 Gestão de Falhas e Rejoin Automático

Uma falha de transmissão é detetada por dois critérios independentes: código de erro de RF (`state < -1199`) ou duração de `sendReceive()` superior a 8000 ms. Após três falhas consecutivas (`retry_count` $\geq$ 3), o firmware executa um *rejoin* OTAA completo, com até cinco tentativas espaçadas de 5 s. Em caso de sucesso, o ADR é imediatamente desativado e o DR fixado em DR5, e os contadores de falha são repostos. Se as cinco tentativas falharem, o nó aguarda 60 s antes de retomar o ciclo normal.

Na prática, porém, este mecanismo nunca operou de forma autónoma nas sessões experimentais, uma vez que o ChirpStack rejeita JoinRequests com *dev nonces* já registados, e o respetivo *flush* requer uma chamada gRPC externa (`flush_dev_nonces()` em `gateway_api.py`) que o próprio nó não pode desencadear. A gestão de *rejoin* foi por isso feita manualmente, pelo que, sempre que um nó perdia sessão, executava-se o *flush* de nonces diretamente na API do ChirpStack antes de reiniciar o nó.

Após um rejoin bem-sucedido, o firmware envia um *uplink* de 1 byte (`0xFF`) para validar a nova sessão LoRaWAN antes de retomar a operação normal. Este frame aparece no servidor com *payload* incompleto (inferior aos 26 bytes mínimos esperados) e é registado com $FCnt = 0$ da nova sessão. Cada rejoin bem-sucedido implica portanto um join OTAA, um *uplink* de validação e a retoma do ciclo normal.

4.3.5 Processamento no Servidor (`server.py`)

O servidor `server.py` (Flask, porta 5000) recebe eventos do ChirpStack via *webhook* `POST /uplink`. Apenas eventos `event=up` são processados; `event=join` é tratado separadamente e os restantes são ignorados. O servidor mantém um contador de *joins* por nó (`session_id`); quando este atinge três (`session_id` ≥ 3), o servidor invoca um *flush* de `DevNonce` na API gRPC do ChirpStack para limpar nonces repetidos acumulados por ciclos de reprogramação frequentes. Esta limpeza preventiva assegura que os nós experimentais conseguem restabelecer sessão sem bloqueios administrativos no servidor de rede. Não há processamento de `event=txack`, uma vez que o servidor opera estritamente em modo unidirecional e nunca enfileira *downlinks* de aplicação que exijam confirmação de transmissão por parte do *gateway*.

Os nós do Pure ALOHA estão registados na aplicação ChirpStack `7bbd73af-7f65-4c7d-bfc8-3ea7e4d1`
A deteção de perdas por `fCnt` processa-se da seguinte forma: para cada trama recebida, o servidor

compara o `fCnt` atual com o último valor registado para aquele dispositivo. Uma diferença > 1 indica a perda de Δ`fCnt` tramas:

```
expected_fcnt = nc['last_fcnt'] + 1
if f_cnt > expected_fcnt:
    collisions_detected = f_cnt - expected_fcnt
    nc['collisions'] += collisions_detected
    TOTAL_COLLISIONS  += collisions_detected
nc['last_fcnt'] = f_cnt
nc['rx_count'] += 1
```

Listing 4: Deteção de perdas por gap de `fCnt` (`server.py`).

O PDR por nó é calculado de forma dinâmica sobre o `fCnt` corrente:

$$\mathrm{PDR}_{\mathrm{PA}}(t) = \frac{\texttt{fCnt}(t) + 1 - \texttt{collisions}(t)}{\texttt{fCnt}(t) + 1} \tag{4.2}$$

Esta forma instantânea é equivalente à expressão cumulativa PDR $= N_{\mathrm{rx}}/(N_{\mathrm{rx}} + \Delta\texttt{fCnt})$ utilizada na Secção 4.5.4, onde $N_{\mathrm{rx}} = \texttt{rx\_count}(t)$ e $\Delta\texttt{fCnt} = \texttt{collisions}(t)$. O PDR final não é calculado por sessão individual: o servidor acumula os *frames* de todas as sessões, incluindo rejoins, somando o *FCnt* corrente ao total de *frames* transmitidos em sessões anteriores. O valor exportado no *summary* reflete portanto o desempenho de toda a campanha experimental, não de uma única sessão contínua.

No cálculo de G medido, o servidor implementa dois métodos de cálculo de G, com preferência pelo método direto se o *payload* v3.0 estiver disponível:

```
for dev_eui, stats in NODE_COUNTERS.items():
    toa_s        = stats['toa_teorico_ms'] / 1000.0
    last_attempt = stats.get('last_tx_attempt_count')
    if last_attempt and duration_s > 0:
        G_real += (last_attempt * toa_s) / duration_s
    else:
        avg_interval_s = mean(stats['interval_real_list']) / 1000.0
        G_real += toa_s / avg_interval_s
```

Listing 5: Cálculo de G por `tx_attempt_count` (preferido) e por intervalo (fallback).

O método direto contabiliza todas as tentativas e é independente do período nominal, incluindo o *overhead* do `sendReceive()`. É o método mais fiel à definição teórica $G = \lambda \cdot T_{\mathrm{ToA}}$.

O servidor calcula explicitamente o débito observado e o débito teórico para cada sessão:

$$S_{\text{obs}} = \frac{N_{\text{rx, sem colisão}} \cdot \overline{T_{\text{ToA}}}}{T_{\text{sessão}}} \tag{4.3}$$

$$S_{\text{teórico}} = G_{\text{real}} \cdot e^{-2G_{\text{real}}} \tag{4.4}$$

$$P_{\text{colisão, teo}} = \left(1 - e^{-2G_{\text{real}}}\right) \times 100\% \tag{4.5}$$

O *summary* exporta estas três grandezas como base da comparação experimental do Capítulo 5. Para cada *uplink* recebido, o servidor executa quatro operações em sequência:

1. Filtrar evento: processar apenas `event=up`;
2. Detetar perdas por `fCnt`: gap > 1 incrementa `collisions` e alimenta o PDR;
3. Detetar sobreposição temporal: verificar se dois *frames* chegaram no mesmo canal com $\Delta t < T_{\text{ToA}}$ (métrica diagnóstica, independente do PDR);
4. Atualizar métricas acumuladas: G_{real}, S_{obs}, drift, airtime.

A Tabela 4.5 resume as duas métricas de perda e as suas semânticas.

	`collision_rate_fcnt_%`	**`collision_rate_temporal_%`**
Origem	Gap de `fCnt`	$\Delta t < T_{\text{ToA}}$, mesmo canal
Ficheiro	`collisions_*.csv`	`temporal_collisions_*.csv`
Semântica	Perda confirmada	Colisão provável (*capture effect* possível)
Entra no PDR	Sim	Não

Tabela 4.5: Comparação das duas métricas de perda do servidor Pure ALOHA.

A presença de oito canais EU868 introduz uma diferença importante face ao modelo teórico de canal único. Quando dois nós transmitem simultaneamente em canais distintos, não existe colisão RF, uma vez que o *gateway* recebe ambos os *frames* independentemente. O servidor já exclui estes pares do contador de sobreposições temporais, mas o modelo $S = Ge^{-2G}$ assume a partilha de um único canal, pelo que a probabilidade de colisão efetiva é inferior à prevista pela teoria, na proporção dos pares que caem em canais diferentes. Com oito canais de igual probabilidade, a fração de pares em canal comum é de $1/8 = 12{,}5\,\%$, o que atenua o impacto das colisões e explica parte do ganho de PDR observado experimentalmente face ao valor teórico.

A calibração do relógio do nó é feita pelo servidor, que computa, por regressão linear, a taxa de *drift* do oscilador de cada nó face ao relógio do *gateway*. O *offset* bruto $\Delta_t = t_{\text{GW,rx}} - \texttt{node\_millis} - T_{\text{ToA}}$ é registado a cada trama; a regressão sobre os pares $(\text{elapsed\_s}, \Delta_t)$ produz o *slope* em ms/s (equivalente a ppm) e o desvio padrão dos resíduos. Os limiares de classificação foram definidos com base nas características do hardware, onde $|\text{drift}| < 50$ ppm é conservador face à tolerância típica do cristal externo de 40 MHz do ESP32-S3 (de ± 20 a 30 ppm de fábrica), acomodando a variação de temperatura; por sua vez, um resíduo < 5 ms corresponde ao patamar

de ruído do método de medição, o qual é dominado pelo *jitter* do encaminhamento HTTP entre o *gateway* e o servidor Flask. Um nó que exceda qualquer um dos limiares é classificado como *deriva*. Quanto à exportação de resultados, no final de cada sessão (comando `S` ou `CTRL+C`) o servidor exporta para a pasta `teste_pure_aloha/` um conjunto de onze ficheiros CSV, organizados por granularidade.

A Tabela 4.6 enumera-os: um ficheiro trama a trama com todos os campos e as janelas deslizantes de G e S, um sumário global da sessão, e ficheiros agregados por nó e por categoria de diagnóstico (colisões, *joins*, distribuição de SF, sobreposições temporais e calibração de relógio), que servem de base à análise do Capítulo 5:

Ficheiro	Conteúdo
`pure_aloha_{ts}.csv`	Trama a trama: todos os campos + G/S por janela deslizante
`summary_{ts}.csv`	Sumário global: G, S, PDR, colisões, RSSI/SNR, P_{col}
`nodes_{ts}.csv`	Por nó: PDR, colisões, sessões, RSSI, *dl_pdr*
`metrics_{ts}.csv`	Por nó: *duty-cycle*, período, *jitter*, SR elevado, drift
`airtime_{ts}.csv`	Por nó: airtime acumulado, utilização do canal, conformidade ETSI
`collisions_{ts}.csv`	Eventos de `fCnt` gap (timestamp, nó, gap)
`joins_{ts}.csv`	Eventos de *join*/*rejoin* (timestamp, `devEUI`)
`sf_distribution_{ts}.csv`	SF por trama (deteção de alterações ADR ao longo da sessão)
`temporal_collisions_{ts}.csv`	Sobreposições temporais (nó, overlap, canal)
`global_timestamps_{ts}.csv`	Todos os *frames* ordenados por tempo, com Δt e *overlap_toa*
`clock_calibration_{ts}.csv`	Drift (ppm, ms/h), offset, resíduo por nó

Tabela 4.6: Ficheiros CSV exportados pelo `server.py` no final de sessão.

O ficheiro `pure_aloha_{ts}.csv` inclui adicionalmente as colunas `G_window` e `S_window`, calculadas sobre uma janela deslizante de 20 tramas, permitindo analisar a evolução temporal da carga oferecida e do *throughput* ao longo da sessão.

4.4 CW-Slotted ALOHA

A implementação do CW-S-ALOHA é construída sobre a arquitetura comum da Secção 4.2 e opera sobre a especificação LoRaWAN 1.1, que introduz um `DevNonce` sequencial e persistente entre resets, eliminando falhas de *join* por repetição e conferindo maior estabilidade a sessões longas com ressincronização frequente. O protocolo acrescenta três mecanismos à base comum: sincronização temporal por *timestamp* piggybacked no *downlink* RX1, seleção aleatória *a priori* do *slot* dentro de uma janela de contenção $N_{\text{CW}} = 3$, e uma fila circular que desacopla a geração do pacote da sua transmissão efetiva. As subsecções seguintes detalham cada componente. O protocolo CW-S-ALOHA exige uma base temporal monótona de microssegundos. A função `micros()` do Arduino sofre *rollover* aos 71 minutos (contador de 32 bits a 1 MHz), pelo que o firmware implementa uma versão estendida de 64 bits. Esta extensão de resolução temporal expande o limite do temporizador, assegurando a continuidade do agendamento de *slots*.

```
static uint64_t micros64() {
  static uint32_t last32 = 0;
  static uint64_t high   = 0;
  uint32_t now32 = (uint32_t)micros();
  if (now32 < last32) high += (uint64_t)1 << 32;
  last32 = now32;
  return high | (uint64_t)now32;
}
```

Listing 6: Base temporal de 64 bits sem *rollover*.

Esta função é exclusiva do firmware CW-S-ALOHA e é a base do seu relógio sincronizado.

4.4.1 Modelo Event-Driven

O acesso ao meio é gerido por dois temporizadores em paralelo. O temporizador de aplicação dispara a cada período T_{app} (14,0/14,5/15,0/15,5 s por nó no Regime 1) e gera um pacote; o temporizador de *slot* dispara a cada $T = 300$ ms e verifica se existe pacote pendente para o *slot* corrente. O nó aguarda sempre o evento mais próximo, permitindo baixo consumo energético entre eventos:

```
last_P = last_S = clock_sync_now();
while (1) {
  next_P = last_P + APP_PERIOD_US;   // proximo evento de aplicacao
  next_S = last_S + SLOT_LEN_US;     // proximo boundary de slot (300 ms)
  next   = min(next_P, next_S);
  espera(next - agora);
  if (next_S <= next_P) {
    process_slot(next_S / SLOT_LEN_US);  last_S = next_S;
  } else {
    process_app();                       last_P = next_P;
  }
}
```

Listing 7: Estrutura do ciclo *event-driven*.

A arquitetura *event-driven* trata o *overhead* do `sendReceive()` de forma diferente do Pure ALOHA. No Pure ALOHA, o período efetivo inclui a duração do `sendReceive()` porque este precede diretamente o `delay()`. No CW-S-ALOHA, o temporizador de aplicação (`next_P`) avança independentemente enquanto `sendReceive()` bloqueia; quando o ciclo retoma, `next_P` já passou ou está iminente, pelo que o evento de aplicação dispara de imediato. O *overhead* de $\approx$1185 ms não se acumula ao período: o intervalo real entre transmissões consecutivas observado

nos CSV ($\approx T_{\text{app}}$) confirma este comportamento, em contraste com o Pure ALOHA onde $T_{\text{real}} \approx T_{\text{app}} + T_{\text{sendReceive}}$.

Este desacoplamento entre a geração do pacote e a sua transmissão é detalhado de forma completa na Figura 4.6, que cobre todo o ciclo do firmware, da sincronização inicial ao evento de slot.

4.4.2 Sincronização Temporal

Todos os nós partilham uma referência temporal comum baseada no mecanismo herdado do S-LoRaWAN (Secção 3.2.2). Após cada *uplink*, o servidor devolve um *downlink* com `fPort`=200 contendo o *timestamp* UTC do *gateway* no instante de receção, serializado em 8 bytes *little-endian* (uint64, microsegundos). O nó calcula o desvio entre esse valor e o seu relógio local. Enquanto o relógio não está sincronizado, o nó envia periodicamente (a cada 30 s) um *probe* de um único byte (`0xAA`). O servidor identifica estes *frames* pelo tamanho do *payload* (inferior a 8 bytes) e pela ausência de *flag* válida (`flag=unknown`), descartando-os de todas as métricas de desempenho; ao receber o *downlink* de resposta, o nó calcula o *offset* inicial e transita para o ciclo CW-S-ALOHA.

A implementação está em `clock_sync.cpp`:

```
void clock_sync_update(uint64_t gw_ts_us, uint64_t rx_end_us) {
  // UTC no momento rx_end = gw_ts + RX1_delay(1s) + ToA_DL(57ms)
  const uint64_t RX1_DELAY_US  = 1000000ULL;
  const uint64_t TOA_DL_US     =   57000ULL;
  const uint64_t DL_LATENCY_US = RX1_DELAY_US + TOA_DL_US;  // 1 057 ms

  const uint64_t net_now_us = gw_ts_us + DL_LATENCY_US;
  g_offset_us = (int64_t)net_now_us - (int64_t)rx_end_us;
  g_valid     = true;
}
```

Listing 8: Atualização do *offset* de sincronização (`clock_sync.cpp`).

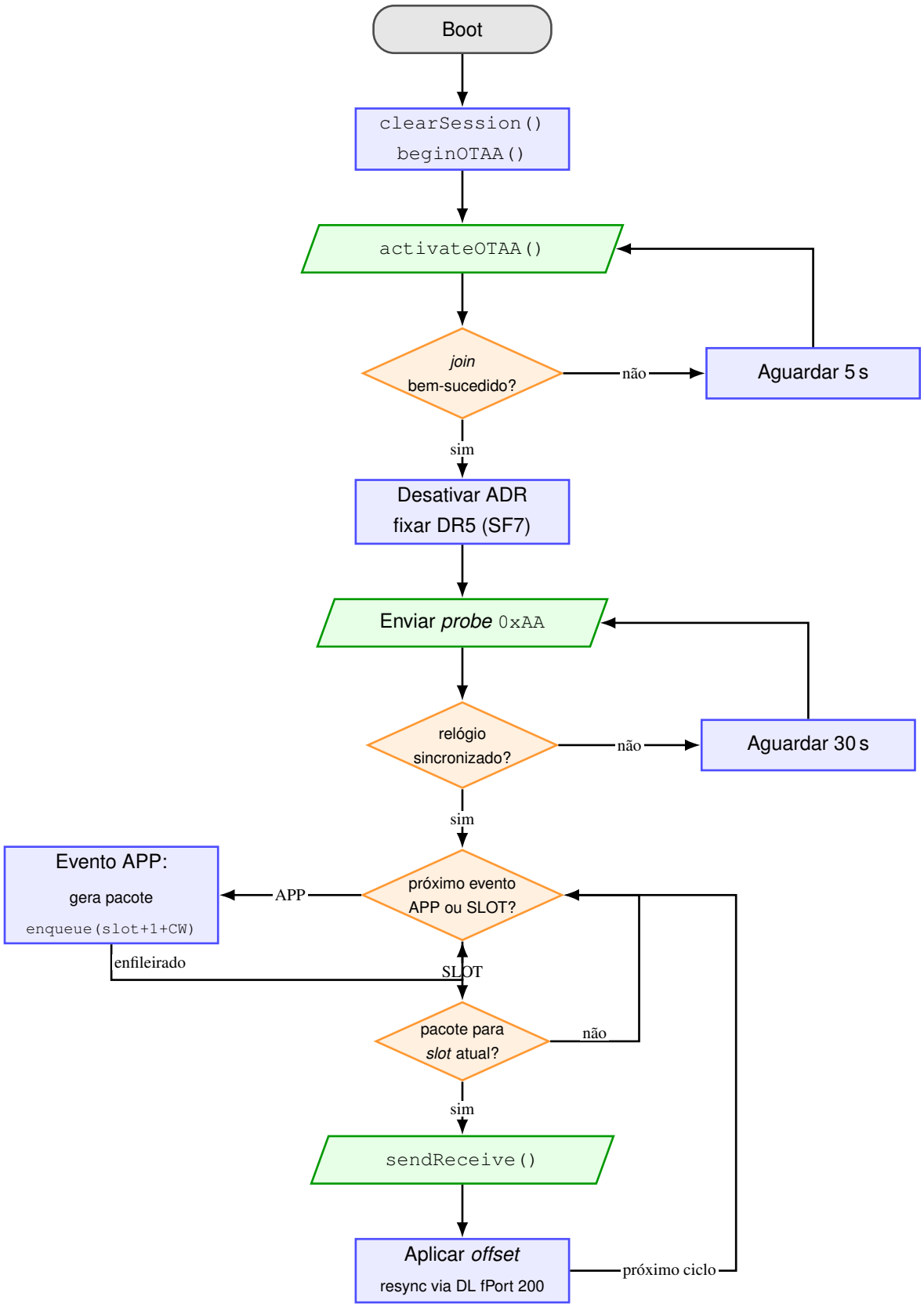


Figura 4.6: Fluxograma do firmware do CW-S-ALOHA, do arranque ao ciclo *event-driven*.

O offset é recalibrado a cada uplink com downlink recebido, com uma cadência que mantém a deriva do cristal muito abaixo do período de guarda dimensionado (Secção 4.4.6).

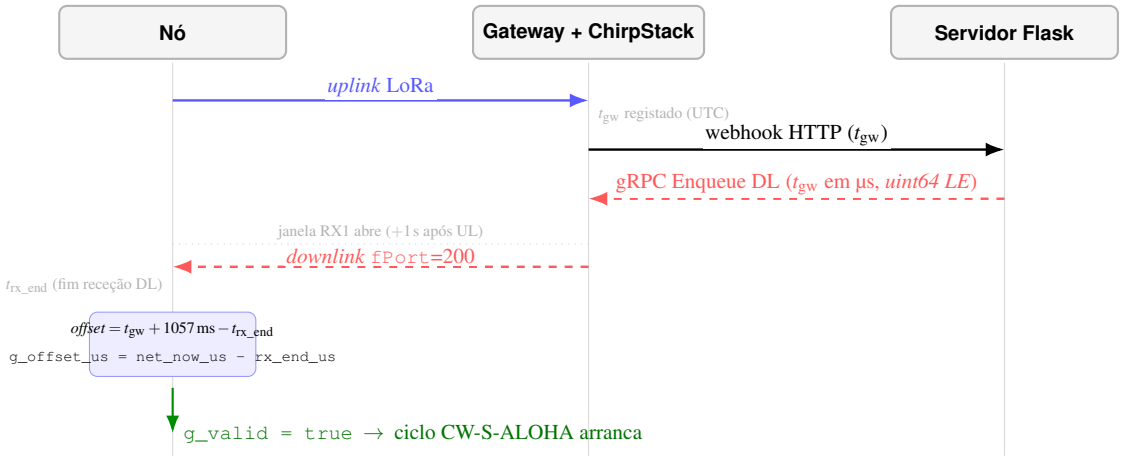


Figura 4.7: Sequência de sincronização temporal no CW-S-ALOHA: do *uplink* ao cálculo do *offset*.

O *firmware* atualiza o relógio para qualquer *downlink* recebido: o *timestamp* é validado antes de ser aplicado, sendo rejeitado se o desvio face ao relógio local estiver fora da janela $[-3\,s, +5\,s]$. Fora desta janela, o *offset* anterior é mantido e o evento é registado como ignorado. Esta validação protege contra *timestamps* antigos entregues com atraso elevado, nomeadamente em situações de congestionamento ou de reinício do ChirpStack.

4.4.3 Slots Absolutos e Janela de Contenção

Com o relógio sincronizado, o *slot* absoluto corrente é dado por `current_slot() = clock_sync_now() / SLOT_LEN_US`, com `SLOT_LEN_US = 300 000` μs. Todos os nós sincronizados partilham a mesma numeração, contígua desde a *epoch* Unix. Quando o evento de aplicação dispara, o nó seleciona o *slot*-alvo:

```
1 uint8_t  cw          = esp_random() % CW_MAX;    // CW in {0, 1, 2}
2 uint64_t slot_target = current_slot() + 1 + cw; // minimo: next slot
```

Listing 9: Seleção aleatória do *slot*-alvo na janela de contenção.

O `esp_random()` utiliza o gerador de *hardware* do ESP32-S3, para produzir uma distribuição uniforme em $\{0,1,2\}$. A latência de escalonamento distribui-se uniformemente entre 300 ms (CW=0) e 900 ms (CW=2). No instante de transmissão, o erro temporal face ao início ideal do *slot* é registado como diagnóstico:

$$\text{slot\_error}_{\mu s} = \texttt{clock\_sync\_now()} - (\texttt{slot\_now} \times \texttt{SLOT\_LEN\_US} + \texttt{GUARD\_US}) \quad (4.6)$$

O servidor calcula o erro de sincronização com base no *timestamp* do *gateway* e no início esperado do *slot*-alvo:

$$\mathrm{SE}_{ms} = \left(t_{\mathrm{gw}} - t_{\mathrm{slot\_target}}\right) \times 1000 - \left(T_g + T_{\mathrm{ToA}}\right) \tag{4.7}$$

onde t_{gw} é o *timestamp* UTC do *gateway* em segundos, $t_{\mathrm{slot\_target}}$ é o instante de início do *slot*-alvo e o termo $(T_g + T_{\mathrm{ToA}}) = 30 + 97{,}5 \approx 128$ ms corresponde à latência esperada entre o início do *slot* e a chegada do último símbolo ao *gateway*. Sem esta subtração, uma transmissão perfeita produziria sempre um erro aparente de $+128$ ms; com ela, o valor esperado numa transmissão perfeita é 0 ms. O desvio médio observado experimentalmente (≈ 60 ms) é atribuível ao overhead interno do RadioLib que não está contabilizado no valor fixo de (≈ 128 ms)

A transmissão bloqueia o firmware durante aproximadamente 1 185 ms ($T_g + T_{\mathrm{ToA}} + T_{\mathrm{RX1}} + T_{\mathrm{ToA,DL}} = 30 + 97{,}5 + 1000 + 57 \approx 1\,185$ ms, correspondendo a $\approx 3{,}95$ *slots*; o Pure ALOHA, sem período de guarda nem *downlink* de sincronização, bloqueia $\approx 97{,}5 + 2 \times 1000 \approx 2100$ ms em ciclos sem *downlink* ou ≈ 1100 ms em ciclos com MAC em RX1. O servidor aceita como bem-sucedidas as tramas com desvio de *slot* $\Delta \in \{0, 1\}$. A razão primária não é a deriva do relógio, mas a semântica do campo `time` do ChirpStack: este regista o instante em que o último símbolo foi recebido pela antena, não o início da transmissão. Uma trama que começa a emitir em `slot_target` $\times T$ chega ao *gateway* em `slot_target` $\times T + T_{\mathrm{ToA}}$, ou seja, $+97{,}5$ ms após o início do *slot*. Com período de guarda de 30 ms, o fim da emissão situa-se em $+127{,}5$ ms, dentro da fronteira de 300 ms.

Contudo, uma ligeira antecipação de sincronização pode fazer com que o *timestamp* caia após a fronteira (`slot_target` $+ 1) \times T$, classificando a trama no *slot* seguinte ($\Delta = +1$). Aceitar $\Delta = 1$ como sucesso cobre este caso sem introduzir permissividade excessiva. A Figura 4.8 representa a estrutura temporal de um *slot* com os valores adotados. O eixo mostra quatro *slots* consecutivos de 300 ms; um evento de aplicação que ocorra no *slot n* sorteia a janela de contenção sobre os três *slots* seguintes (CW $\in \{0, 1, 2\}$). No *slot*-alvo escolhido, a transmissão só começa após o período de guarda $T_g = 30$ ms e ocupa o canal durante o *Time-on-Air* de 97,5 ms, restando ainda folga até ao fim do *slot*, o que ilustra graficamente o invariante $T > T_{\mathrm{ToA}} + T_g$.

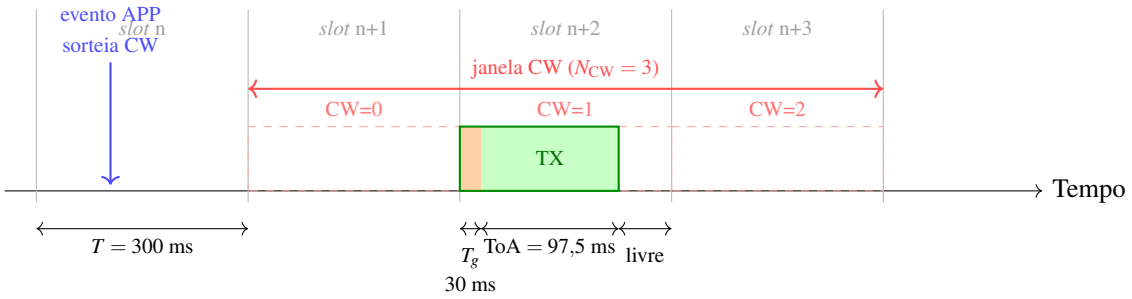


Figura 4.8: Estrutura temporal de um *slot* no CW-S-ALOHA (*slot*-alvo $n{+}2$ a título ilustrativo).

Após o retorno de `sendReceive()`, o ponteiro `last_S` é realinhado ao *slot* corrente através da *flag* interna `g_reset_loop` processada na iteração seguinte do ciclo, aplicando desta forma

este bloqueio sem acumulação de erros; `last_P` não é alterado, preservando a integridade do período de aplicação.

Esta operação é necessária porque `sendReceive()` bloqueia o ciclo durante aproximadamente 1185 ms, o equivalente a ≈3,95 *slots*. Durante esse bloqueio, todos os eventos de *slot* são perdidos e `last_S` continua a apontar para o *slot* que originou a transmissão. Sem realinhamento, a iteração seguinte calcularia `next_S = last_S + SLOT_LEN_US`, um instante já no passado, e dispararia três a quatro eventos de *slot* em cascata num único ciclo. O `g_reset_loop` previne esta acumulação de dívida de *slots* ao forçar `last_S` para o instante atual imediatamente após o retorno de `sendReceive()`.

4.4.4 Fila Circular

Os pacotes gerados são armazenados numa fila circular de oito posições (`QUEUE_SIZE = 8`). O `process_slot()` descarta entradas expiradas (`slot_target < slot_now`) e transmite a primeira entrada cujo `slot_target` coincide com o *slot* corrente. Em regime permanente, a fila contém no máximo uma entrada por nó, bem dentro da capacidade disponível. Em caso de *overflow* (fila completamente preenchida), o firmware descarta o pacote mais antigo (posição *head*) para dar lugar ao novo, garantindo que o nó transmite sempre dados recentes. Esta política prioriza a recência sobre a garantia de entrega: num cenário de carga muito elevada onde os pacotes se acumulam mais depressa do que são transmitidos, o nó continua a reportar o estado atual em vez de tentar entregar pacotes com vários ciclos de atraso. A Figura 4.9 ilustra este funcionamento. As oito posições dispõem-se em anel; as entradas a cheio representam pacotes enfileirados, cada um com o seu *slot*-alvo, e as posições a tracejado estão livres. O ponteiro de *enqueue* insere novos pacotes à medida que os eventos de aplicação ocorrem, enquanto o `process_slot()` percorre a fila a cada fronteira de *slot* para transmitir o pacote correspondente, deixando confortável a margem face às oito posições.

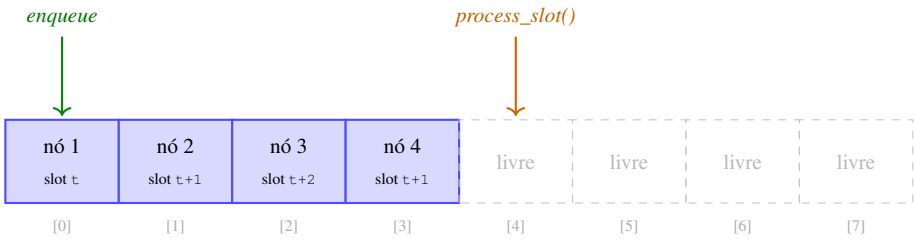


Figura 4.9: Fila circular de oito posições (`QUEUE_SIZE` = 8) do CW-S-ALOHA.

4.4.5 Estrutura do Payload e Flags

O *payload* de 50 bytes inclui uma *flag* que distingue o tipo de evento. No momento do *enqueue*, o pacote é marcado com `FLAG_APP_ENQUEUE` (0x02); imediatamente antes da transmissão em `process_slot()`, a *flag* é substituída por `FLAG_SLOT_TX` (0x01). O servidor filtra do cálculo de métricas todos os *frames* que não carreguem `FLAG_SLOT_TX`, excluindo automaticamente os

probes (1 byte, sem *flag* válida) e eventuais pacotes de *enqueue* não convertidos. A Tabela 4.7 detalha a estrutura do *payload* de 50 bytes. O campo `uplink_cnt` (bytes 1–4) é o equivalente ao `tx_attempt_count` do Pure ALOHA e serve de base ao cálculo de G medido. Os bytes 6 a 8, exclusivos do CW-S-ALOHA, transportam a CW sorteada (byte 6) e o *slot* absoluto alvo em dois bytes *little-endian* (bytes 7 a 8, `uint16`), lidos pelo servidor como `payload[7] | (payload[8] « 8)`.

Uma consequência direta do mecanismo de substituição de *flag* é que o servidor nunca recebe um *frame* com `flag=APP_ENQUEUE`: a substituição por `FLAG_SLOT_TX` ocorre sempre antes de qualquer transmissão. Por este motivo, a coluna `tx_app` nos ficheiros CSV exportados apresenta o valor zero em todas as sessões e em todos os nós, o que é o comportamento esperado e não um erro de contagem.

Tabela 4.7: Estrutura do *payload* do CW-S-ALOHA (50 bytes).

Offset	Tamanho	Tipo	Campo	Descrição
0	1 byte	`uint8_t`	`node_id`	Identificador do nó (1–4)
1	4 bytes	`uint32_t`	`uplink_cnt`	Contador de tentativas de TX
5	1 byte	`uint8_t`	`flag`	`0x01=SLOT_TX` / `0x02=APP_ENQUEUE`
6	1 byte	`uint8_t`	`cw`	Valor da CW sorteada ($\in \{0,1,2\}$)
7	1 byte	`uint8_t`	`slot_target_lo`	Byte baixo do *slot* alvo (*uint16 LE*)
8	1 byte	`uint8_t`	`slot_target_hi`	Byte alto do *slot* alvo (*uint16 LE*)
9	41 bytes	`uint8_t[41]`	—	Preenchimento fixo (`0x00`)

4.4.6 Dimensionamento do Slot e do Período de Guarda

Para que dois nós em *slots* adjacentes não se sobreponham, a duração do *slot* deve satisfazer a condição necessária:

$$T > T_{\text{ToA}} + T_g \quad \Longrightarrow \quad 300 > 97{,}5 + 30 = 127{,}5 \text{ ms}, \tag{4.8}$$

o que define um limite inferior de $T_{\min} = 127{,}5$ ms e uma margem disponível de $300/127{,}5 \approx 2{,}35\times$, suficiente para absorver *jitter* de escalonamento e variações de ToA. Para os parâmetros do sistema (SF7, BW 125 kHz, *payload* de 50 bytes), o ToA analítico é $T_{\text{ToA}} = 97{,}5$ ms (Secção 4.2.4). O período de guarda foi fixado em $T_g = 30$ ms para absorver o *jitter* de escalonamento de *software* e não a deriva do relógio, uma vez que, com ressincronização a cada 14 a 15,5 s e cristal de ≈ 100 ppm, a deriva acumulada por ciclo é da ordem de $100 \times 10^{-6} \times 15{,}5 \approx 1{,}5$ ms, valor muito inferior ao período de guarda. O *jitter* medido em modo `DEBUG_MODE` é inferior a 10 ms, pelo que a margem $T_g/jitter_{\max} = 30/10 = 3\times$ é suficiente. A escolha de $T = 300$ ms satisfaz a condição (4.8) com uma margem de $300 - 127{,}5 = 172{,}5$ ms e foi determinada por três critérios adicionais. Em primeiro lugar, mantém a proporção $T_g/T = 10\,\%$, próxima da adotada por Polonelli et al. [20] no S-LoRaWAN ($T_b/T \approx 19\,\%$), garantindo uma fração útil do *slot* de:

$$\frac{T_u}{T} = \frac{T - T_g}{T} = \frac{300 - 30}{300} = 0{,}9. \tag{4.9}$$

Em segundo lugar, com $T_{\text{app}} = 14$ s no Regime 1, o *duty-cycle* por nó é $T_{\text{ToA}}/T_{\text{app}} = 97{,}5/14000 \approx 0{,}70\,\%$, abaixo do limite regulamentar de 1 % da ETSI EU868. Em terceiro lugar, a granularidade de 300 ms é compatível com a resolução do *timestamp* do *gateway* ($\pm 20\ \mu$s no SX1302) e com o *jitter* de escalonamento medido, garantindo um alinhamento fiável entre nós.

4.4.7 Parâmetros de Configuração

A Tabela 4.8 reúne os parâmetros que definem o comportamento temporal do protocolo. Os três primeiros fixam a estrutura de *slots*: a duração do *slot* (`SLOT_LEN_US`), a largura da janela de contenção (`CW_MAX`) e o período de guarda (`GUARD_US`). Seguem-se a capacidade da fila circular (`QUEUE_SIZE`), o intervalo entre *probes* de *bootstrap* (`PROBE_INTERVAL`), o período de aplicação por nó no Regime 1 (`APP_PERIOD`) e o porto reservado ao *downlink* de sincronização (`DL_FPORT`). O dimensionamento destes valores é justificado nas Secções 4.4.6 e seguintes.

Parâmetro	Valor	Descrição
`SLOT_LEN_US`	300 000 μs (300 ms)	Duração do *slot* T
`CW_MAX`	3	Janela de contenção: CW $\in \{0,1,2\}$
`GUARD_US`	30 000 μs (30 ms)	Período de guarda T_g
`QUEUE_SIZE`	8	Capacidade da fila circular
`PROBE_INTERVAL`	30 s	Intervalo do *probe* de bootstrap
`DL_FPORT`	200	Porto do *downlink* de sincronização

Tabela 4.8: Parâmetros de configuração do CW-S-ALOHA.

4.4.8 Processamento no Servidor (`server.py` + `cw_aloha.py`)

O servidor CW-S-ALOHA é constituído por dois ficheiros com responsabilidades distintas: `server.py` contém o ponto de entrada Flask e a lógica de comunicação com o ChirpStack; `cw_aloha.py` contém a classe `CWAlohaManager`, responsável pelo cálculo de todas as métricas. Para cada *uplink* recebido, o servidor executa quatro operações em sequência. A primeira operação é a classificação do *frame*. O servidor verifica se o campo *flag* do *payload* é `FLAG_SLOT_TX` (0x01). Caso contrário, o *frame* é registado com tipo *unknown* e excluído de todas as métricas de desempenho (*probes* `0xAA` e pacotes `APP_ENQUEUE` entram nesta categoria). A segunda operação é o envio do *downlink* de sincronização.

Os nós do CW-S-ALOHA estão registados na aplicação ChirpStack cujo identificador consta da Tabela 4.1. O servidor extrai o *timestamp* do campo `time` do *webhook*, converte-o para Unix em microsegundos e enfileira um *downlink* com `fPort`=200 via gRPC. Para evitar acumulação de itens na fila do ChirpStack em períodos de carga elevada, o servidor mantém uma *flag* `dl_pending` por

nó: o *downlink* só é enfileirado se não existir já um pendente, e a *flag* é reposta a *False* quando o *gateway* confirma a transmissão (`event=txack`) ou o item expira (`event=log/EXPIRED`):

```
# server.py
if not NODE_COUNTERS[dev_eui].get('dl_pending'):
    ts_s = parse_ns_time(timestamp)             # timestamp UTC do webhook
     ↪  (s)
    enqueue_timestamp_downlink(ts_s, dev_eui)   # gRPC via gateway_api.py
    NODE_COUNTERS[dev_eui]['dl_pending'] = True
# gateway_api.py --> serializacao e envio via gRPC
req.queue_item.data       = struct.pack("<Q", int(ts_s * 1_000_000))   #
 ↪  uint64 LE (us)
req.queue_item.f_port     = DOWNLINK_FPORT    # 200
req.queue_item.expires_at = now_utc + timedelta(seconds=10)
client.Enqueue(req, metadata=[("authorization", f"Bearer {API_KEY}")])
```

Listing 10: Controlo da fila de *downlink* de sincronização.

O prazo de expiração de 10 s é generoso face à janela RX1 (1 s após o *uplink*), acomodando latências de escalonamento sem risco de o ChirpStack não encontrar o item na fila. O tempo de processamento total medido experimentalmente é consistentemente inferior a 1 s. A terceira operação é a deteção de perdas via `fCnt`. Para cada *frame* recebido com `FLAG_SLOT_TX`, o servidor compara o `fCnt` com o último valor registado para aquele nó; uma diferença superior a um indica a perda de `fCnt_gap` *frames*. Este contador alimenta o cálculo do PDR, de forma idêntica ao Pure ALOHA:

$$\mathrm{PDR} = \frac{\texttt{tx\_slot}}{\texttt{tx\_slot} + \texttt{tx\_collision}} \tag{4.10}$$

onde `tx_slot` é o número de *frames* `SLOT_TX` recebidos e `tx_collision` são as perdas estimadas por gap de `fCnt`. A quarta operação é a deteção de sobreposições temporais. Para cada *uplink* recebido, o servidor calcula o número de *slot* absoluto com base no campo `time` do *webhook*. Se dois *frames* de nós distintos chegarem no mesmo *slot* e no mesmo canal, é registada uma colisão temporal:

```
# cw_aloha.py
arrivals = self._slot_arrivals[slot]  # lista de (dev_eui, canal)
multi_node   = len(set(d for d, _ in arrivals)) > 1
same_channel = len(set(c for _, c in arrivals)) < len(arrivals)
collision    = multi_node and same_channel
```

Listing 11: Lógica de deteção de sobreposição temporal (pseudocódigo simplificado).

Para verificar se a trama chegou ao *slot* correto, o servidor reconstrói o *slot*-alvo completo a partir dos 16 bits transmitidos no *payload* (`uint16`, alcance de $2^{16} \times 0{,}3\text{ s} \approx 5{,}5\text{ h}$):

```
for offset in range(-1, CW_MAX + 2):   # offsets: -1, 0, 1, 2, 3, 4
    candidate = slot_chegada + offset
    if (candidate & 0xFFFF) == slot_target_raw:
        slot_target_full = candidate
        break
```

Listing 12: Reconstrução do *slot*-alvo completo a partir dos dois bytes *uint16* do *payload*.

O offset -1 cobre o caso $\Delta = +1$ (trama classificada pelo *gateway* no *slot* seguinte ao alvo); os offsets 0 a 2 cobrem os valores normais da janela de contenção ($\text{CW} \in \{0,1,2\}$); os offsets 3 e 4 constituem margem adicional. Com o *slot*-alvo reconstruído, o servidor calcula $\Delta = \texttt{slot\_chegada} - \texttt{slot\_target\_full}$ e classifica a trama como bem-sucedida se $\Delta \in \{0, 1\}$.

O critério de *mesmo canal* é essencial: dois *frames* recebidos no mesmo *slot* mas em canais distintos não constituem colisão RF, através da ortogonalidade dos canais EU868. Esta métrica é mantida separada do PDR e nunca entra no cálculo da Equação 4.10. A distinção é fundamental: o `fCnt` gap prova a perda de um *frame*, enquanto a sobreposição temporal caracteriza a causa provável dessa perda mas não constitui prova de entrega falhada (o *capture effect* pode recuperar um dos *frames* colidentes). Nas campanhas realizadas, a taxa de sobreposição temporal foi consistentemente inferior à taxa de perdas por `fCnt` gap, confirmando que a sobreposição não é a causa dominante de perda. À semelhança do Pure ALOHA, no final de cada sessão (comando `S` ou `CTRL+C`) o servidor consolida os contadores e exporta os dados para ficheiros CSV, incluindo um sumário por nó (`cw_nodes_*.csv`) e ficheiros de diagnóstico de *slot*, sincronização e sobreposição temporal, para posterior análise no Capítulo 5.

4.5 Metodologia de Avaliação

4.5.1 Carga Oferecida e Regimes Experimentais

A carga oferecida G é a variável independente central da avaliação. Para um conjunto de N nós com período médio de aplicação $\bar{T}_{\text{app}}$, a carga agregada normalizada é dada pela Equação 3.3, que para os quatro nós desta dissertação se reduz a $G = 4\,T_{\text{ToA}}/\bar{T}_{\text{app}}$. Variando o período de aplicação, percorre-se a curva $S(G)$ desde o regime de baixa carga, onde as colisões são raras, até ao regime de contenção elevada, onde os protocolos se distinguem de forma mais acentuada. A Tabela 4.3 situa cada regime na curva teórica do Pure ALOHA $S = Ge^{-2G}$, cujo máximo é $S_{\max} = 0{,}184$ em $G = 0{,}5$. O Regime 1 ($G \approx 0{,}026$) produz $S \approx 0{,}025$, operando a apenas 13% da capacidade máxima, sendo esta uma consequência direta do limite ETSI de 1% de *duty-cycle*. Os Regimes 2 e 3 ($G \approx 0{,}072$ e $G \approx 0{,}122$) atingem $S \approx 0{,}062$ e $S \approx 0{,}094$, correspondendo a 34% e 51% da capacidade máxima, ainda muito aquém do ótimo teórico em $G = 0{,}5$, mas suficientes para expor

diferenças substanciais entre os protocolos e situar a comparação numa região com colisões não negligenciáveis.

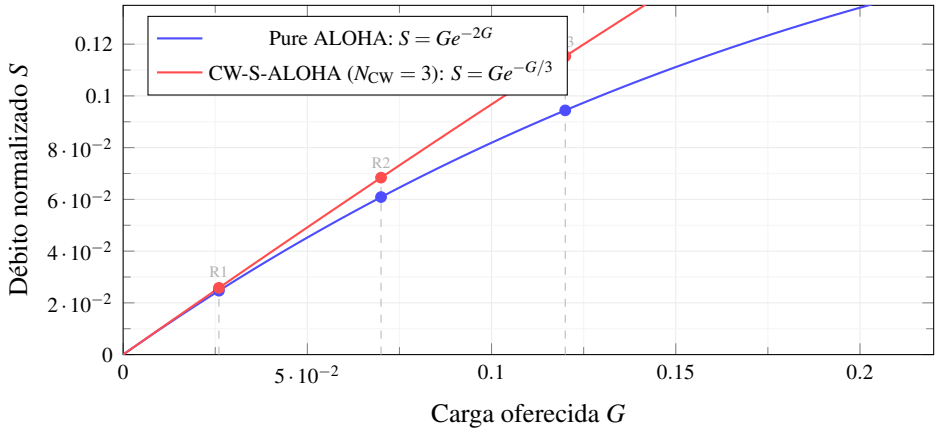


Figura 4.10: Curva $S(G)$ teórica com os três regimes experimentais assinalados.

Para cada regime, o ganho teórico do CW-S-ALOHA face ao Pure ALOHA, medido pelo rácio S_{CW}/S_{PA}, é de $\approx 4\%$ em R1, $\approx 12\%$ em R2 e $\approx 22\%$ em R3. A probabilidade teórica de colisão no Pure ALOHA, dada por $P_{col,PA} = (1 - e^{-2G}) \times 100\%$, é de $\approx 5\%$ em R1, $\approx 13\%$ em R2 e $\approx 21\%$ em R3; no CW-S-ALOHA, dada por $P_{col,CW} = (1 - e^{-G/3}) \times 100\%$, é de $\approx 1\%$, $\approx 2\%$ e $\approx 4\%$, respetivamente. Estes valores constituem a referência teórica face à qual os resultados experimentais do Capítulo 5 são avaliados.

O Regime 1 opera dentro do limite regulamentar de *duty-cycle* de 1% imposto pela ETSI para a banda EU868. O período mínimo conforme é $T_{min} = T_{ToA}/0{,}01 \approx 9{,}75$ s por nó, pelo que, com quatro nós, a carga máxima legal é $G_{max} \approx 0{,}04$. Os Regimes 2 e 3, necessários para observar o regime de contenção das curvas $S(G)$, exigem períodos inferiores a este limiar e implicam, portanto, uma violação deliberada do limite de *duty-cycle*. Esta violação é aceitável apenas num ambiente laboratorial controlado e isolado, constituindo uma não conformidade regulamentar numa rede real. Importa notar que, para uma mesma carga G, os períodos adotados em cada protocolo não são necessariamente idênticos. O ciclo de transmissão do CW-S-ALOHA inclui o bloqueio da janela RX1 e do *downlink* de sincronização (Secção 4.4.3), o que impõe um período de aplicação mínimo praticável superior ao do Pure ALOHA. Nos regimes de carga mais elevada, os períodos do CW-S-ALOHA são por isso ligeiramente mais longos do que os do Pure ALOHA, ajustando-se de modo a igualar a carga efetiva G. A comparação é assim conduzida a igual G e com períodos semelhantes, e não a períodos estritamente iguais.

4.5.2 Ferramentas e Ambiente de Desenvolvimento

A Tabela 4.9 resume o software utilizado em cada componente do sistema experimental.

Componente	Software	Versão / Detalhe
Nós terminais	*framework* Arduino + RadioLib	6.6.0
	compilação e gravação	Arduino IDE
	monitorização em série	Serial Monitor (modo `DEBUG_MODE`)
Gateway	*packet forwarder* (HAL)	5.0.1
	concentratord	4.6.0
	ChirpStack Network Server	v4.16.2
	sincronização de relógio	`sync_time.py` (SSH, `date`+`hwclock`)
Servidor Flask	Python + Flask	3.x
	ambiente de desenvolvimento	Visual Studio Code
	comunicação *downlink*	gRPC (`chirpstack-api`)
Análise de resultados	Microsoft Excel	a partir dos CSV exportados
Medição de energia	Monsoon HVPM + PowerTool	$V_{out} = 4{,}97$ V

Tabela 4.9: Software utilizado por componente do sistema experimental.

O Serial Monitor do Arduino IDE é utilizável apenas em modo `DEBUG_MODE`: neste modo, o firmware substitui o *light sleep* por chamadas a `delay()`, mantendo o controlador UART do ESP32-S3 ativo e a porta série funcional.

Em modo de produção, o firmware ativa `esp_light_sleep_start()` entre eventos, o que suspende o UART durante o período de sono e torna o Serial Monitor inutilizável. O modo de produção é reservado exclusivamente às sessões de medição de energia com o Monsoon HVPM; todas as campanhas de desempenho foram realizadas em modo `DEBUG_MODE`.

4.5.3 Procedimento Experimental

Todos os ensaios foram realizados em ambiente interior, num laboratório, sem obstáculos entre os nós e o *gateway*. Os quatro nós foram posicionados a uma distância de aproximadamente 10 m do *gateway* SenseCAP. Nestas condições, o RSSI recebido foi consistentemente elevado ($\gtrsim -80$ dBm) e nunca constituiu um fator limitante. Quaisquer tipo de perdas observadas são atribuíveis ao mecanismo de acesso ao meio e não à propagação. Cada campanha de medição segue uma sequência de passos reproduzível.

As diferenças entre os dois protocolos são indicadas explicitamente e os restantes passos são idênticos para ambos. O primeiro passo é a preparação do firmware de cada nó: seleciona-se o ficheiro de credenciais `credentials_nodeN.h`, ajusta-se o período de aplicação para o regime de carga pretendido (Tabela 4.3) e compila-se e grava-se o firmware no Arduino IDE, repetindo o processo para os quatro nós. Antes de ligar os nós, garante-se que o *gateway* SenseCAP está operacional e que o relógio do servidor de rede foi alinhado ao do computador de análise através do script `sync_time.py`, que envia a hora UTC por SSH e a grava no relógio de hardware. De seguida arranca-se o servidor Flask com `python server.py`, esta ordem é deliberada, pois assegura que o servidor está pronto a receber o primeiro *uplink* e, no caso do CW-S-ALOHA, a responder com o *downlink* de sincronização que os nós aguardam para arrancar o ciclo de *slots*.

Ligados os nós, cada um executa a ativação OTAA e fixa o SF em DR5; no Pure ALOHA, os nós entram de imediato no ciclo de transmissão periódica, enquanto que no CW-S-ALOHA aguardam o primeiro *downlink* de sincronização, enviando *probes* de *bootstrap* até o relógio estar alinhado. A sessão decorre durante um intervalo prolongado, suficiente para acumular um número de tramas estatisticamente representativo em cada regime, com os quatro nós a transmitir em paralelo sob a mesma carga G.

Por fim, a sessão é terminada pelo operador (comando `S` ou `CTRL+C`), momento em que o servidor consolida os contadores e exporta os ficheiros CSV para a pasta dos testes. Os ficheiros são posteriormente importados para o Microsoft Excel, onde se calculam o PDR, a carga oferecida medida, o *throughput* e as métricas de diagnóstico de cada protocolo, e se confrontam os dois protocolos a igual G. As campanhas de medição de energia seguem o procedimento de ativação do nó e arranque do servidor descrito anteriormente, mas em modo de produção, pelo que a ativação OTAA e o registo no ChirpStack decorrem de forma idêntica, alterando-se apenas o método de observação do funcionamento do nó. Deste modo, em vez de se acompanhar a telemetria através do Serial Monitor, o consumo é observado em tempo real através do gráfico de corrente apresentado pelo *software* PowerTool e registado pelo Monsoon HVPM.

4.5.4 Métricas

As métricas de desempenho são calculadas de forma idêntica nos dois protocolos, garantindo que existe comparação direta. A taxa de entrega de pacotes (PDR) é determinada a partir de saltos no contador de *frames* (`fCnt`), única evidência efetiva de que um *frame* não foi entregue.

Para o Pure ALOHA, o PDR é dado pela Equação 4.2 e para o CW-S-ALOHA, pela Equação 4.10. As duas expressões medem a mesma grandeza e são por isso diretamente comparáveis.

A carga oferecida G é calculada de duas formas complementares. O valor teórico é obtido pela Equação 3.3 com os períodos nominais. O valor medido é derivado do campo em que se contam as tentativas de transporte no *payload* de cada nó, nomeadamente através de `tx_attempt_count` no Pure ALOHA e de `uplink_cnt` no CW-S-ALOHA, acumulando todas as tentativas de transmissão independentemente do resultado. O *throughput* S é exportado em duas variantes. A variante *estrita* (S_{obs}) contabiliza apenas os *frames* recebidos no *slot* correto ($\Delta = 0$) e sem gap de `fCnt`:

$$S_{\text{obs}} = \frac{N_{\Delta=0,\,\text{sem gap}} \cdot T_{\text{ToA}}}{T_{\text{sessão}}} \tag{4.11}$$

A variante *comparável* (S_{comp}) contabiliza todos os *frames* sem gap de `fCnt`, independentemente do desvio de *slot*. A diferença entre as duas variantes quantifica o impacto do *jitter* de sincronização no *throughput* efetivo; sendo que S_{comp} é a métrica comparável com o PDR do Pure ALOHA à mesma carga G. O erro de sincronização (`slot_error_ms`) é uma métrica exclusiva do CW-S-ALOHA, calculada pela Equação 4.6 e registada a cada transmissão. Permite verificar experimentalmente que o *jitter* real é inferior ao período de guarda dimensionado. A sobreposição temporal é calculada em ambos os protocolos como métrica diagnóstica, nunca entrando no cálculo do PDR. No Pure ALOHA, quantifica a colisão esperada da ausência de coordenação; no CW-S-ALOHA, sinaliza

falha de escalonamento ou janela de contenção insuficiente. O período de arranque de cada sessão é excluído da análise de forma automática. No Pure ALOHA, as primeiras tramas podem apresentar `fCnt=0` ou `fCnt=1` resultantes do *join*; o servidor apenas inicia o cálculo de PDR a partir da segunda trama recebida de cada nó, quando já existe um `fCnt` de referência. No CW-S-ALOHA, os *probes* de *bootstrap* enviados antes da primeira sincronização são descartados pelo servidor com base na *flag* `unknown` (Secção 4.4.2); as métricas são calculadas exclusivamente sobre tramas com `FLAG_SLOT_TX`, pelo que o período de sincronização inicial (tipicamente inferior a 30 s) não contamina os resultados. Em ambos os protocolos, o instante de início da sessão registado no nome do ficheiro CSV corresponde ao arranque do servidor, não ao primeiro *uplink* válido; a duração efetiva da sessão com dados válidos é ligeiramente inferior à duração total da campanha.

4.6 Sumário

Este capítulo descreveu a implementação dos dois protocolos sobre a mesma infraestrutura experimental. A arquitetura de firmware comum (RadioLib 6.6.0, OTAA, ADR desativado, SF7 fixo) garante que as diferenças de desempenho observadas são atribuíveis exclusivamente ao mecanismo de acesso ao meio e não a variações de hardware ou configuração. O Pure ALOHA estabelece a referência sem qualquer coordenação temporal; o CW-S-ALOHA introduz sincronização por *slot* de 300 ms e aleatorização *a priori* da janela de contenção, com parâmetros dimensionados para respeitar o invariante $T > T_{\text{ToA}} + T_g$. O PDR é calculado de forma idêntica nos dois protocolos, via descontinuidades de `fCnt`, garantindo comparabilidade direta a igual carga G. A avaliação percorre três regimes de carga, do regime conforme de baixa carga ao regime de contenção elevada, cujos resultados são apresentados e analisados no Capítulo 5.

Capítulo 5

Resultados e Discussão

Este capítulo apresenta os resultados experimentais dos dois protocolos implementados no Capítulo 4, confrontando o Pure ALOHA (referência) com o CW-Slotted ALOHA proposto. A avaliação percorre três regimes de carga: $G \approx 0{,}026$ (baixa, conforme ao *duty-cycle* ETSI), $G \approx 0{,}072$ (moderada) e $G \approx 0{,}12$ (elevada), obtidos pela redução progressiva do período de aplicação. Em cada regime, ambos os protocolos operam sobre a mesma infraestrutura LoRaWAN com SF7 e BW 125 kHz, sendo os períodos ajustados de modo a igualar G; a comparação isola assim o efeito do mecanismo de acesso ao meio. A Secção 5.1 apresenta os resultados do Pure ALOHA, a Secção 5.2 os do CW-S-ALOHA, e a Secção 5.3 confronta diretamente os dois protocolos e relaciona os resultados com as previsões teóricas do Capítulo 3.

5.1 Pure ALOHA: Resultados Experimentais

As sessões T1, T2 e T3 foram desenhadas para cobrir três pontos de operação distintos do canal Pure ALOHA: tráfego leve conforme ETSI ($G = 0{,}027$), carga intermédia ($G = 0{,}072$) e carga de *stress* ($G = 0{,}121$), esta última intencionalmente acima do limite de 1 % de *duty-cycle* para expor o regime de saturação do protocolo. Todos os nós operam em SF7, BW 125 kHz e $T_{\text{oA}} = 97{,}536$ ms; o cálculo de G baseia-se no contador `tx_attempt_count` transportado no *payload*, conforme descrito na Secção 4.3.5.

5.1.1 Carga Oferecida: Teórica vs Medida

A carga oferecida teórica G_{teo} é calculada pela expressão

$$G_{\text{teo}} = T_{\text{oA}} \sum_{i=1}^{4} \frac{1}{T_i}, \tag{5.1}$$

onde T_i é o período efetivo de cada nó e $T_{\text{oA}} = 97{,}536$ ms.

Sessão	Duração	Período efetivo por nó (s)				G_{teo}	G_{med}
	(min)	N1	N2	N3	N4		
T1	298,7	14,0	14,5	15,0	15,5	0,0265	0,0265
T2	110,4	4,80	5,20	5,60	6,00	0,0728	0,0724
T3	96,1	2,80	3,05	3,30	3,55	0,1238	0,1214

Tabela 5.1: Parâmetros de carga das sessões Pure ALOHA.

Em T1 e T2, os períodos nominais são suficientemente longos para que o mecanismo de guarda de 200 ms nunca seja ativado, resultando em desvios $\leq 0{,}5\,\%$ entre G_{teo} e G_{med}. Em T3, os períodos mais curtos (2,80–3,55 s) fazem com que alguns ciclos atinjam o limite mínimo de espera.

5.1.2 PDR

O PDR é calculado pelo servidor a partir do contador de *frames* transmitidos e recebidos pelo *gateway*, acumulando todas as sessões incluindo *rejoins* (ver Secção 4.3.5).

Sessão	Nó	TX	RX	Perdas	PDR (%)
T1 (G=0,027)	N1	1277	1234	43	96,63
	N2	1234	1165	68	94,41
	N3	1194	1168	26	97,82
	N4	1156	1108	48	95,85
	Total	4861	4675	185	96,17
T2 (G=0,072)	N1	1372	1083	289	78,94
	N2	1269	1126	143	88,73
	N3	1178	1086	92	92,19
	N4	1102	972	130	88,20
	Total	4921	4267	654	86,71
T3 (G=0,121)	N1	1955	1316	639	67,31
	N2	1866	1260	606	67,52
	N3	1741	1236	505	70,99
	N4	1620	1129	491	69,69
	Total	7182	4941	2241	68,80

Tabela 5.2: PDR e contagem de perdas por nó e por sessão (Pure ALOHA).

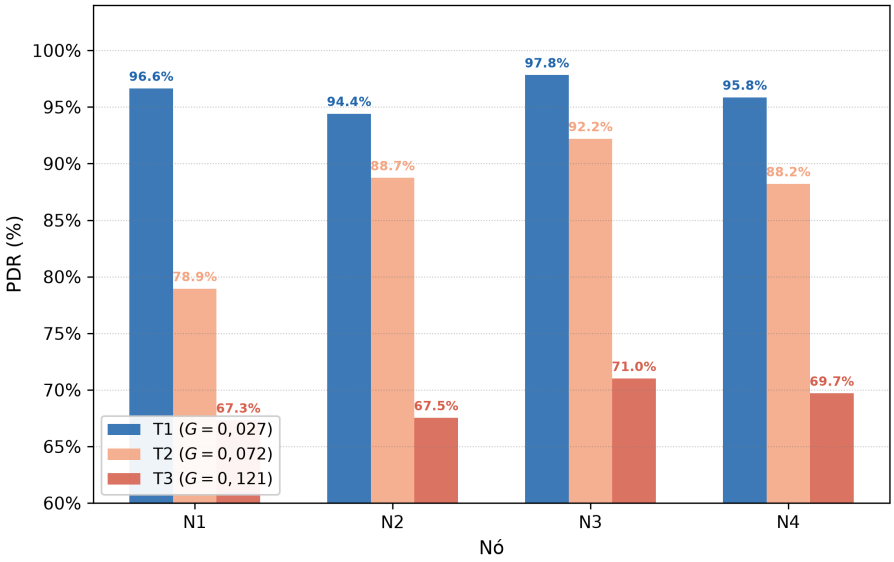


Figura 5.1: PDR por nó nas três sessões Pure ALOHA.

A assimetria mais saliente ocorre em T2, onde N1 apresenta PDR de 78,9 % face a 88–92 % nos restantes nós, penalizado pela cadência mais elevada e pelo RSSI intermédio. Em T3, todos os nós convergem para 67–71 %, evidenciando saturação homogénea a carga elevada.

Em T1, as perdas medidas por *fCnt gap* (3,8 %) ficam abaixo da previsão teórica $P_{\text{col}} = 1 - e^{-2G} = 5{,}2\,\%$. A diferença de $-1{,}4$ pp resulta de dois fatores complementares: a diversidade de 8 canais EU868 reduz por um fator $\approx 1/8$ a probabilidade de colisão intra-canal, e o efeito de captura resolve os choques restantes em favor do nó com RSSI dominante. Com diferenças inter-nó de 7–8 dB em T1 e um limiar de captura típico de SF7 de 3–6 dB, a recuperação é quase garantida. Em T2, as perdas medidas (13,3 %) estão em acordo quase perfeito com a teoria (13,5 %, desvio de $-0{,}2$ pp), validando o modelo de Pure ALOHA neste ponto de operação. Em T3, as perdas medidas (31,2 %) excedem a teoria (21,5 %) em $+9{,}7$ pp: o *fCnt gap* agrega colisões RF e perdas por saturação e interferência acumulada, enquanto a fórmula teórica modela apenas colisões num canal único com tráfego de Poisson. A Figura 5.1 torna visível a assimetria intra-sessão em T2: N1, com o período mais curto (4,8 s), apresenta PDR de 78,9 %, consideravelmente inferior aos restantes nós.

Esta penalização tem origem dupla: a cadência mais elevada expõe N1 a um maior número de colisões por unidade de tempo, e o seu RSSI médio ($-55{,}5$ dBm) não beneficia da vantagem de captura que N3 detém ($-45{,}8$ dBm, nó mais próximo do *gateway*), razão pela qual N3 apresenta o PDR mais elevado de T2 (92,2 %). Em T3, a maior carga global uniformiza as perdas entre nós (67–71 %), atenuando a assimetria observada em T2. Deste modo, demonstra-se que o mecanismo de captura dita a distribuição de perdas em carga moderada, mas perde eficácia protetora quando o canal atinge a saturação generalizada.

5.1.3 Débito Medido

O débito normalizado S representa a fração do tempo de canal utilizada para transmissão bem-sucedida.

Sessão	S_{obs}	$S_{\text{teo}} = Ge^{-2G}$	$S_{\text{obs}}/S_{\text{teo}}$
T1	0,0245	0,0251	97,6 %
T2	0,0540	0,0627	86,1 %
T3	0,0514	0,0953	53,9 %

Tabela 5.3: Débito normalizado: Pure ALOHA (valores de G na Tabela 5.1).

Em T1, o acordo com a teoria (97,6 %) valida a metodologia de medição e confirma que o modelo de Pure ALOHA descreve corretamente o sistema a baixa carga, sendo o resultado mais relevante a quase-estagnação de S_{obs} entre T2 e T3. De facto, apesar de G aumentar 67,7 %, S_{obs} desce apenas de 0,054 para 0,051 ($-4{,}8\,\%$), um colapso que ocorre a $G = 0{,}121$, o que representa apenas 24 % do pico teórico $G^* = 0{,}5$. Este comportamento resulta de dois mecanismos concorrentes: por um lado, o *overhead* das janelas RX1+RX2 (≈ 2 s) ocupa 55–69 % do período em T2–T3, comprimindo o tempo disponível para transmissão útil; por outro lado, com quatro nós a partilhar 8 canais EU868, a probabilidade de colisão intra-canal cresce mais rapidamente do que o previsto pelo modelo de canal único Ge^{-2G}, explicando o aumento do desvio $S_{\text{obs}}/S_{\text{teo}}$ de 86,1 % em T2 para 53,9 % em T3.

5.1.4 Distribuição Temporal de Colisões

O servidor implementa dois métodos de deteção de perdas (ver Secção 4.3.5): *fCnt gap* e sobreposição temporal no mesmo canal. A Figura 5.2 compara os dois métodos nas três sessões.

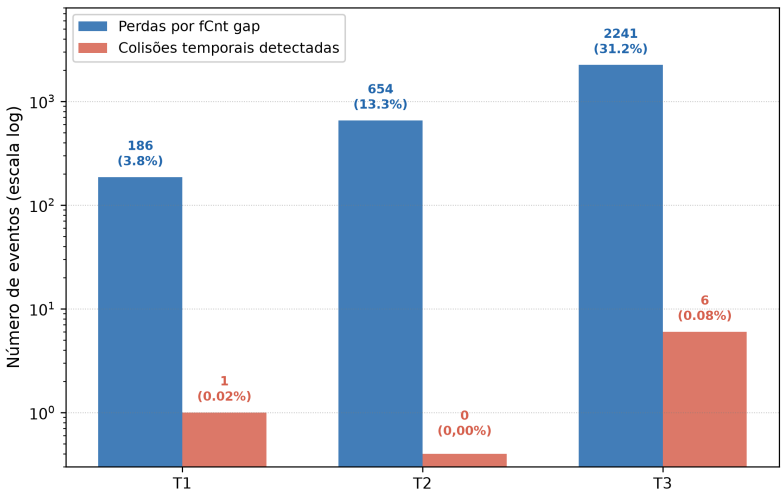


Figura 5.2: Perdas por *fCnt gap* vs colisões por sobreposição temporal (Pure ALOHA, escala logarítmica).

A discrepância entre os dois métodos é estrutural. Com 8 canais EU868 equiprováveis, apenas $1/8$ dos pares de *frames* sobrepostos no tempo partilham o mesmo canal. Em T2, se as 654 perdas por *fCnt gap* fossem exclusivamente devidas a colisões diretas, seriam esperadas $\approx 654/8 \approx 82$ deteções temporais; foram observadas zero. Esta ausência indica que as perdas detetadas por *fCnt gap* não resultam predominantemente de colisões diretas no mesmo canal, uma vez que o *fCnt gap* agrega todos os tipos de perda (colisões RF, janelas de receção perdidas e atrasos de escalonamento no ChirpStack), enquanto o detetor de sobreposição temporal captura apenas os pares de *frames* que chegam simultaneamente ao mesmo canal sem resolução por captura. Com diferenças de RSSI inter-nó de 7 a 14 dB (Tabela 5.4) e um limiar de captura típico de SF7 de 3 a 6 dB, as sobreposições RF que ocorrem são na sua maioria resolvidas em favor do nó dominante, contribuindo para o acordo entre perdas medidas (13,3 %) e teóricas (13,5 %) em T2. As perdas residuais distribuem-se por causas não-RF que o modelo de Pure ALOHA não captura mas que, a esta carga são impactantes. Em T3, as 6 colisões temporais detetadas, face a 2241 perdas por *fCnt gap*, mostram que o efeito de captura continua a dominar, mas a maior densidade de tráfego origina ocasionalmente pares de *frames* com RSSI suficientemente próximo para que nenhum seja descodificado. O *fCnt gap* permanece, por isso, a métrica primária de perdas, pois captura a totalidade das falhas independentemente da causa.

5.1.5 Qualidade de Sinal e Calibração de Relógio

Sessão	Nó	RSSI (dBm)	SNR (dB)	Deriva (ppm)	Resíduo (ms)
T1	N1	−52,6	13,1	+9,1	64,0
	N2	−58,9	13,0	+6,9	62,5
	N3	−60,1	12,9	+7,4	62,1
	N4	−58,5	13,0	+6,1	65,0
T2	N1	−55,5	13,1	+12,7	81,4
	N2	−55,1	13,0	+9,5	62,1
	N3	−45,8	13,1	+9,0	45,4
	N4	−60,0	12,9	+8,7	47,4
T3	N1	−56,9	13,1	+9,4	36,4
	N2	−55,3	12,8	+7,8	31,1
	N3	−41,4	13,1	+8,4	33,4
	N4	−54,8	12,9	+6,5	37,0

Tabela 5.4: Qualidade de sinal e deriva de relógio: Pure ALOHA (médias de sessão).

O SNR estável (12,8 a 13,1 dB em todas as sessões e nós) confirma que as perdas têm origem exclusivamente protocolar: os nós operam em linha de vista com o *gateway* em ambiente controlado, e a margem de ligação é suficiente para que nenhuma perda seja atribuível à propagação. A deriva

de relógio situa-se entre +6 e +13 ppm, coerente com a especificação do oscilador interno do ESP32 (±20 ppm).

O facto de todos os valores de deriva serem positivos sugere um offset sistemático comum aos quatro osciladores, o qual é provavelmente de origem térmica, uma vez que os nós operam em condições de temperatura estável em laboratório, e não uma flutuação aleatória simétrica em torno de zero.

O padrão decrescente do resíduo entre sessões, de T1 (62–65 ms) para T2 (45–81 ms) e T3 (31–37 ms), constitui por si um resultado: à medida que o período diminui, o `sendReceive()` torna-se mais consistente, reduzindo o *jitter* não capturado pelo modelo linear. A exceção é N1 em T2 (81 ms), cujo resíduo elevado reflete a alta variabilidade de `sendReceive()` nessa sessão ($\sigma = 671$ ms): a alternância entre retorno antecipado em RX1 (≈ 1300 ms, quando o ChirpStack entrega um MAC *command*) e espera completa em RX2 (≈ 2000 ms) cria uma distribuição bimodal de `wait_ms` que o modelo de regressão linear não captura. Em T3, a cadência mais elevada impede o ChirpStack de entregar comandos MAC em RX1 sistematicamente, pois o intervalo entre transmissões (≈ 3 s) é insuficiente para enfileirar e transmitir um *downlink*, tornando o `sendReceive()` unimodal e o resíduo mínimo.

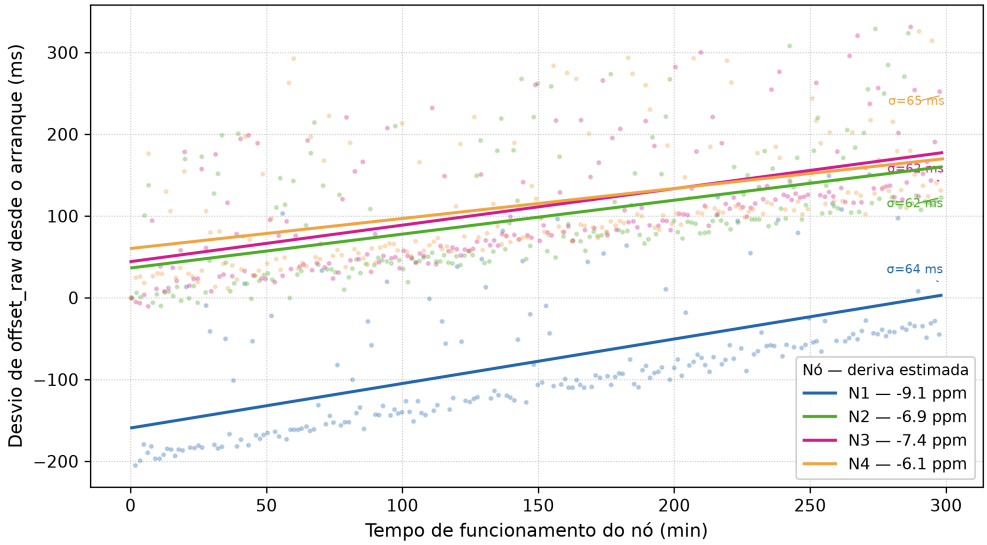
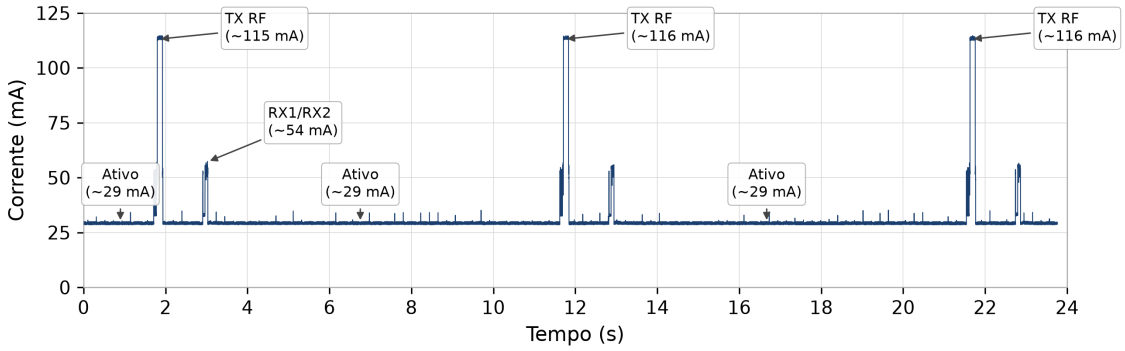


Figura 5.3: Desvio normalizado de `offset_raw` vs tempo de funcionamento (sessão T1, 298 min).

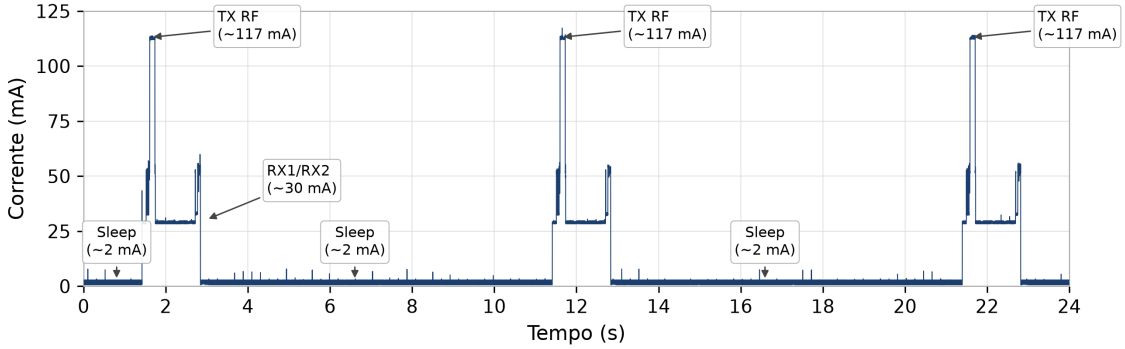
O padrão linear da deriva confirma que o oscilador interno do ESP32 tem comportamento estável ao longo do tempo, com inclinação constante coerente com os valores de ppm reportados na Tabela 5.4. A ausência de saltos ou descontinuidades valida a metodologia de calibração por regressão linear utilizada pelo servidor: um modelo não linear ou a presença de *outliers* indicaria fontes de erro adicionais, como variações térmicas abruptas ou perda intermitente de sincronização, nenhuma das quais se observa nesta sessão. Este comportamento sustenta a fiabilidade da deriva estimada como métrica de diagnóstico ao longo de toda a campanha experimental.

5.1.6 Modos de Operação e Energia

Foi conduzido um ensaio independente com um único nó a período fixo de $T = 10\,\text{s}$, comparando o `DEBUG_MODE` com o modo *light sleep*. A Figura 5.4 apresenta o perfil de corrente medido para três ciclos completos em cada modo.



(a) `DEBUG_MODE`: corrente de base $\approx 29\,\text{mA}$ durante o intervalo entre transmissões.



(b) *Light sleep*: corrente de $\approx 2\,\text{mA}$ durante o intervalo entre transmissões ($\approx 82{,}5\,\%$ do ciclo).

Figura 5.4: Perfil de corrente para três ciclos completos ($T = 10\,\text{s}$, Pure ALOHA).

O pico de TX (≈ 115–$117\,\text{mA}$, $\approx 127\,\text{ms}$) é idêntico em ambos os modos; o pico secundário corresponde às janelas RX1/RX2 do ciclo Classe A ($\approx 2\,\text{s}$ após cada TX). A corrente de base distingue os dois modos: $\approx 29\,\text{mA}$ em `DEBUG_MODE` e $\approx 2\,\text{mA}$ em *light sleep*. A corrente média por ciclo é obtida por integração do perfil de corrente sobre N ciclos completos,

$$I_{\text{med}} = \frac{1}{NT}\int_0^{NT} i(t)\,\mathrm{d}t, \tag{5.2}$$

a partir da qual se extrapola a autonomia para qualquer capacidade de bateria segundo $A = C_{\text{bat}}/I_{\text{med}}$.

Modo	I_{med} (mA)	I_{sleep} (mA)	Fração *sleep* (%)	Autonomia* (h)
`DEBUG_MODE`	30,28	—	0	33
Light sleep	7,25	2,0	82,5	138

Tabela 5.5: Corrente média e estimativa de autonomia com bateria LiPo de 1000 mAh ($^*A = C_{\text{bat}}/I_{\text{med}}$).

O modo *light sleep* reduz a corrente média em $\approx 76\,\%$ (de 30,28 para 7,25 mA), correspondendo a um fator de autonomia de 4,2× (33 para 138 horas com 1000 mAh). Os picos de TX e de receção são idênticos em ambos os modos, confirmando que a alteração do modo de operação não afeta o comportamento protocolar. A comparação energética com o CW-S-ALOHA é apresentada na Secção 5.3.4. Em síntese, os três pontos de operação validam o modelo de Pure ALOHA em regime leve (T1, acordo de 97,6 % com a teoria), identificam o único ponto conforme com o ETSI ($G = 0{,}027$) e caracterizam o regime de saturação na transição T2 para T3: o débito colapsa a $G = 0{,}121$, apenas 24 % do pico teórico $G^* = 0{,}5$, muito antes do regime de máxima capacidade. Estes resultados estabelecem a linha de referência para a comparação com o CW-S-ALOHA.

5.2 CW-Slotted ALOHA: Resultados Experimentais

As sessões C1, C2 e C3 reproduzem exatamente os três pontos de operação do Pure ALOHA ($G \approx 0{,}026$, 0,072 e 0,122), partilhando os mesmos períodos nominais por nó, o mesmo SF7 e BW 125 kHz, e a mesma infraestrutura LoRaWAN. O protocolo CW-S-ALOHA foi parametrizado com $N_{CW} = 3$, $T_{\text{slot}} = 300$ ms e $T_g = 30$ ms; os quatro nós operam com períodos escalonados entre si de modo a distribuir o tráfego ao longo do tempo e evitar bloqueio de fase.

5.2.1 Carga Oferecida: Teórica vs Medida

A carga teórica é calculada pela mesma expressão utilizada para o Pure ALOHA (Equação 5.1), com $T_{\text{oA}} = 97{,}536$ ms. A Tabela 5.6 apresenta os parâmetros das três sessões; as métricas de débito e PDR são detalhadas nas Secções 5.2.3 e 5.2.2, respetivamente.

Nível	Períodos nominais (s)	$\bar{T}_{\text{nom}}$ (s)	G_{teo}	G_{obs}	Desvio (%)	Duração (min)
C1	14,0 / 14,5 / 15,0 / 15,5	14,75	0,0265	0,0264	−0,4	302,3
C2	4,8 / 5,2 / 5,6 / 6,0	5,40	0,0723	0,0715	−1,1	132,7
C3	2,80 / 3,05 / 3,30 / 3,55	3,18	0,1228	0,1218	−0,8	86,2

Tabela 5.6: Parâmetros de carga das sessões CW-S-ALOHA.

O desvio entre G_{teo} e G_{obs} é inferior a 1,2 % em todos os níveis, confirmando a qualidade da calibração e a estabilidade do gerador de tráfego. Em C1 e C2, os períodos nominais são suficientemente longos para que o `sendReceive()` nunca ultrapasse o limiar de guarda de

200 ms, resultando em desvios $\leq 0{,}5\,\%$. Em C3, o bloqueio de `sendReceive()` ($\approx 1{,}2$ s por ciclo) sobrepõe-se ocasionalmente ao temporizador de período, expandindo ligeiramente o intervalo efetivo entre pacotes e reduzindo o tráfego injetado ($-0{,}8\,\%$), o mesmo fenómeno observado em T3 do Pure ALOHA, ainda que com impacto ligeiramente menor graças ao *jitter* introduzido pela espera de *slot* $E[\mathrm{CW}] \cdot T_{\mathrm{slot}}$.

5.2.2 PDR

O PDR é calculado pelo servidor a partir do contador de sequência LoRaWAN (`fCnt`), acumulando todas as transmissões incluindo eventuais *rejoins*, da mesma forma que no Pure ALOHA (Secção 4.3.5).

Nó	PDR C1 (%)	PDR C2 (%)	PDR C3 (%)
1	96,90	94,56	78,66
2	97,91	92,87	75,99
3	98,51	93,50	77,55
4	98,20	93,87	81,84
Sistema	**97,86**	**93,72**	**78,41**

Tabela 5.7: PDR por nó e nível de carga: CW-S-ALOHA.

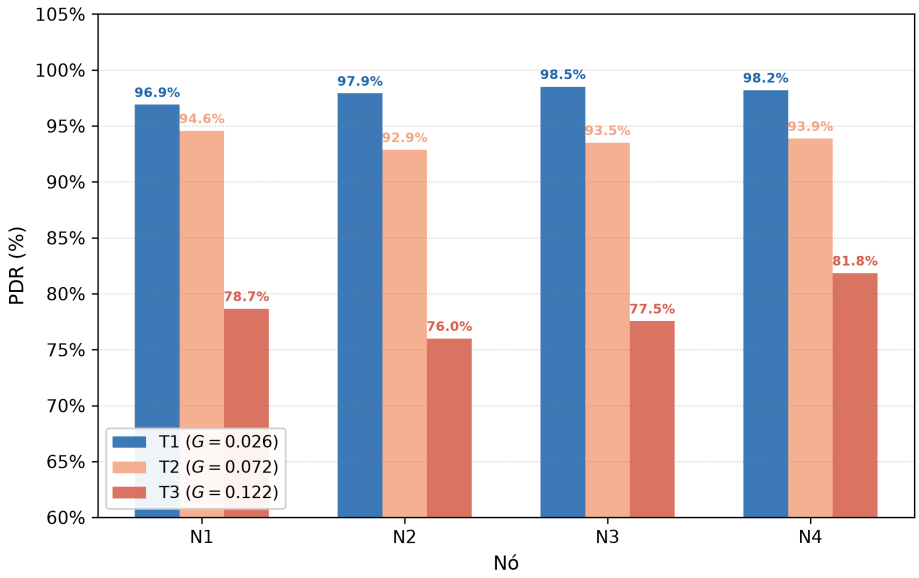


Figura 5.5: PDR por nó nas três sessões CW-S-ALOHA.

A dispersão inter-nós é reduzida em todos os níveis (máximo 1,7 pp em C2), contrastando com os 13,3 pp observados no Pure ALOHA em T2. Em C3, o nó 4 ($T_{\mathrm{nom}} = 3{,}55$ s) apresenta a PDR

mais elevada (81,8 %), evidenciando que o mecanismo de perda dominante é a expiração de *slot* durante o bloqueio de `sendReceive()`.

Em C1, a PDR situa-se entre 96,9 e 98,5 %, com uma dispersão inter-nós de apenas 1,6 pp. A ausência de assimetria reflete que, a carga baixa, cada transmissão raramente entra em contenção e o escalonamento de períodos distribui uniformemente a carga rádio entre os nós. O valor de sistema (97,9 %) é 1,7 pp superior ao do Pure ALOHA em T1 (96,2 %), uma diferença já mensurável mesmo a $G \approx 0{,}026$, confirmando que a contenção por janela elimina as raras sobreposições simultâneas de dois ou mais nós no mesmo *slot* temporal. Em C2, a PDR desce para 92,9–94,6 %, com dispersão inter-nós de 1,7 pp. Este resultado contrasta fortemente com o Pure ALOHA a carga equivalente (T2), onde N1 apresentou PDR de 78,9 % e os restantes se situaram entre 88,2 e 92,2 %, produzindo uma dispersão inter-nós de 13,3 pp. No CW-S-ALOHA, a dispersão por $N_{CW} = 3$ *slots* elimina a penalização associada ao período mais curto: ao sortear um dos três *slots* antes de transmitir, todos os nós enfrentam a mesma probabilidade de contenção, independentemente da sua cadência individual. Esta uniformização é a consequência mais visível do mecanismo de acesso ao meio proposto e é confirmada pela Figura 5.5. Em C3, a PDR desce para 76,0–81,8 %. O gradiente entre nós inverte-se parcialmente: o nó 4 ($T_{\text{nom}} = 3{,}55$ s, PDR 81,8 %) supera o nó 1 ($T_{\text{nom}} = 2{,}80$ s, PDR 78,7 %). Esta inversão aponta para um mecanismo de perda diferente do observado em C2: com períodos tão curtos, a duração de `sendReceive()` ($\approx 1{,}2$ s) representa uma fração significativa do ciclo (39–43 %), e o `slot_target` calculado antes do início de `sendReceive()` pode já ter expirado quando o nó regressa ao ciclo principal. Nós com período mais longo têm maior margem de tempo entre a saída de `sendReceive()` e a próxima janela de transmissão, reduzindo as expirações de *slot*.

5.2.3 Débito e Mecanismo Dominante de Perdas

O débito normalizado S representa a fração do tempo de canal utilizada para transmissão bem-sucedida. O modelo teórico para o CW-S-ALOHA com $N_{CW} = 3$ é:

$$S_{\text{CW}} = G e^{-G/N_{CW}}, \tag{5.3}$$

que assume ausência de perdas MAC e tráfego de Poisson perfeito.

Nível	S_{obs}	S_{teo}	$S_{\text{obs}}/S_{\text{teo}}$ (%)	*Duty-cycle* RF (%)
C1	0,0249	0,0262	95,0	0,629–0,697
C2	0,0605	0,0698	86,7	1,626–2,032
C3	0,0635	0,1170	54,3	2,747–3,483

Tabela 5.8: Débito normalizado CW-S-ALOHA.

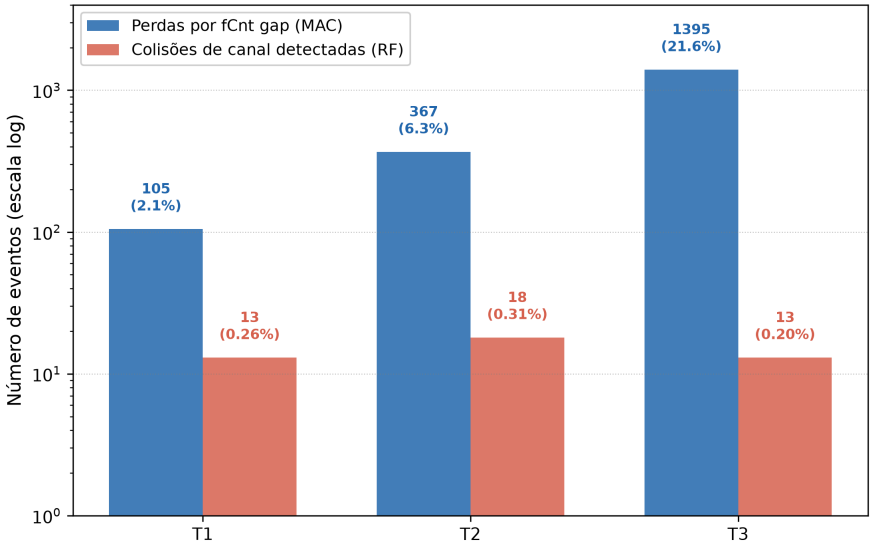


Figura 5.6: Perdas por *fCnt gap* (MAC) vs colisões de canal (RF) nas três sessões CW-S-ALOHA (escala logarítmica).

Em C1, o débito observado ($S_{\text{obs}} = 0{,}0249$) representa 95 % do previsto pelo modelo (5.3), um acordo que valida simultaneamente a metodologia de medição e a aplicabilidade do modelo a baixa carga. As 13 colisões de canal detetadas em 302 min (Figura 5.6) correspondem a uma taxa de 0,09 % face ao total de transmissões, muito próxima do nível de ruído de medição. O *duty-cycle* máximo de 0,697 % respeita o limite ETSI de 1 % em todos os nós, tornando C1 o único nível de carga regulamentarmente conforme. Em C2, $S_{\text{obs}} = 0{,}0605$ com desvio de 13,3 % face ao modelo. A maior divergência em relação a C1 resulta da combinação de dois fatores: o aumento da probabilidade de contenção intra-*slot* e o crescente peso do *overhead* das janelas Classe A, que comprime a janela disponível para transmissão útil, exatamente como em T2 do Pure ALOHA. O *duty-cycle* excede o limite ETSI em todos os nós (1,63–2,03 %); à semelhança do descrito para o Pure ALOHA, C2 e C3 devem ser interpretadas como ensaios de *stress* e não como configurações de implantação real. Em C3 observa-se a divergência mais acentuada: $S_{\text{obs}} = 0{,}0635$ representa apenas 54,3 % do previsto pela Equação (5.3). A Figura 5.6 é crucial para interpretar este resultado: apenas 13 colisões de canal foram detetadas em 86 min (0,11 % das transmissões). A causa da divergência não são colisões RF, pois o protocolo CW-S-ALOHA elimina-as com sucesso, mas perdas MAC por expiração de *slot*. Com $T_{\text{app}} \approx 3$ s e `sendReceive()` a bloquear $\approx 1{,}2$ s por ciclo, os pacotes cujo `slot_target` cai dentro da janela de bloqueio são descartados silenciosamente pelo *firmware* sem tentativa de retransmissão. Esta limitação é inerente à arquitetura de bloqueio síncrono da pilha LoRaWAN e manifesta-se quando a condição de operação eficiente

$$T_{\text{app}} \gg T_{\text{sendReceive}} + N_{CW} \cdot T_{\text{slot}} \approx 1{,}2 + 0{,}9 = 2{,}1\,\text{s} \tag{5.4}$$

deixa de ser satisfeita. Em C3 ($T_{\text{app}} \approx 3{,}18$ s), a margem disponível é de apenas 1,1 s, insuficiente para absorver a variabilidade de `sendReceive()`, ao passo que em C1 ($T_{\text{app}} = 14{,}75$ s) e C2

($T_{\text{app}} = 5{,}40\,\text{s}$) a condição é amplamente respeitada. O resultado mais significativo de C3 é que a degradação do débito resulta da arquitetura de implementação e não do mecanismo de acesso ao meio, o que distingue o CW-S-ALOHA do Pure ALOHA, onde a degradação em T3 resulta de colisões genuínas que o efeito de captura apenas parcialmente mitiga. Em síntese, as três sessões confirmam que o CW-S-ALOHA opera de acordo com o modelo teórico em C1 (95 %) e C2 (87 %), com a divergência em C3 atribuível a perdas MAC por arquitetura de implementação e não a falhas do protocolo de acesso ao meio.

5.2.4 Precisão de Sincronização Temporal

O erro de sincronização temporal e_{slot} quantifica o desvio entre o instante de chegada do pacote ao *gateway* e o instante alvo esperado dentro do *slot* atribuído:

$$e_{\text{slot}} = \left(t_{\text{arr}} - t_{\text{slot\_start}}\right) \times 10^3 - L_{\text{MAC}}, \tag{5.5}$$

onde $L_{\text{MAC}} = T_g + T_{\text{oA}} = 30 + 97{,}5 \approx 128\,\text{ms}$. Um pacote cuja transmissão inicia exatamente no instante $t_{\text{slot\_start}} + T_g$ resulta em $e_{\text{slot}} = 0\,\text{ms}$; a janela válida é $e_{\text{slot}} \in [-128,\, 172]\,\text{ms}$.

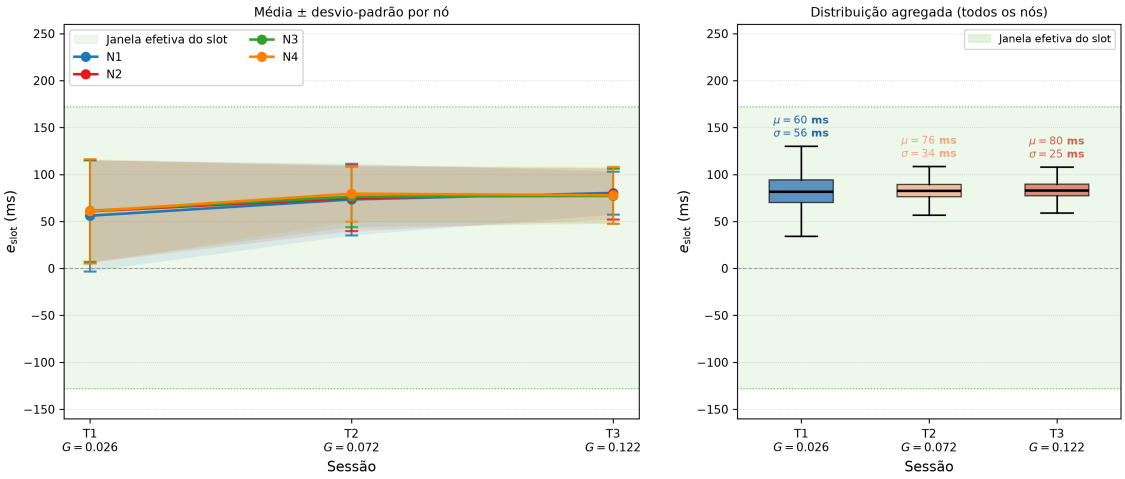


Figura 5.7: Erro de *slot* e_{slot} nas três sessões CW-S-ALOHA. Esquerda: evolução de $\mu \pm \sigma$ por nó. Direita: distribuição agregada por sessão.

Os quatro nós comportam-se de forma quase idêntica em todas as sessões e a banda de incerteza estreita-se com a carga. A banda a verde delimita a janela efetiva $[-128, 172]$ ms; todos os eventos SLOT_TX ficam dentro desta janela em todos os regimes.

Tabela 5.9: Estatísticas de e_{slot} (ms) e deslocamento médio em *slots* ($\bar{\Delta}_{\text{slots}}$): CW-S-ALOHA.

	C1 ($G = 0{,}026$)		C2 ($G = 0{,}072$)		C3 ($G = 0{,}122$)	
Nó	μ (ms)	σ (ms)	μ (ms)	σ (ms)	μ (ms)	σ (ms)
1	56,1	59,2	73,4	38,1	80,4	22,8
2	60,8	54,6	74,9	35,1	79,3	27,1
3	61,4	54,3	76,3	32,2	77,2	29,6
4	60,9	55,7	79,6	29,8	78,0	30,4
Sistema	60,0	55,7	75,9	34,2	80,1	25,0
$\bar{\Delta}_{\text{slots}}$	0,09–0,13		0,04–0,05		0,02–0,04	

Da Tabela 5.9 e da Figura 5.7 emergem dois padrões que, à primeira vista, são contra-intuitivos. Em primeiro lugar, μ cresce com a carga ($60 \rightarrow 76 \rightarrow 80$ ms). Em C1, com períodos de 14–15 s, o nó passa ≈ 11 s em *light sleep* entre transmissões; o acerto do relógio interno acumula *jitter* nas transições de entrada e saída do modo de sono, deslocando o instante real de saída da janela de guarda em relação ao alvo teórico. À medida que os períodos diminuem (C2: 5 s; C3: 3 s), o nó tem cada vez menos tempo de dormência e retoma o ciclo quase imediatamente após `sendReceive()`: a espera de *slot* predomina e fixa o instante de transmissão a ≈ 80 ms após $t_{\text{slot\_start}} + T_g$, valor que corresponde ao atraso médio de processamento do *firmware* antes de chamar o transcetor. Em segundo lugar, σ decresce com a carga ($55{,}7 \rightarrow 34{,}2 \rightarrow 25{,}0$ ms). A dispersão em C1 é dominada pelo *jitter* do ciclo de dormência: cada nó acorda com um desvio diferente do relógio interno, tornando o instante de transmissão variável. Em C3, a ausência efetiva de dormência torna a sequência `wait_slot` $\rightarrow$ TX praticamente determinista, com *jitter* reduzido à variância de escalonamento do RTOS e ao tempo de acesso ao canal LoRa ($\sigma \approx 25$ ms). O resultado mais importante é que todos os pacotes registados como SLOT_TX se situam dentro da janela efetiva $[-128, 172]$ ms em todos os níveis de carga. O valor de $\bar{\Delta}_{\text{slots}} \in [0{,}02, 0{,}13]$ mostra que a esmagadora maioria das chegadas ocorre no *slot* alvo ($\Delta = 0$) ou, por efeito da marcação temporal do *gateway* no fim da receção, no *slot* seguinte ($\Delta = 1$), ambos classificados pelo servidor como transmissão bem-sucedida no *slot* correto. Em C3, o valor de $\bar{\Delta}_{\text{slots}} \approx 0{,}03$ é o mais baixo das três sessões, confirmando que a temporização é mais precisa exatamente quando a carga é maior.

Ao contrário do Pure ALOHA em T2, onde N1 apresentou uma distribuição bimodal de `sendReceive()` (modos em ≈ 1300 e ≈ 2000 ms), todas as distribuições de e_{slot} no CW-S-ALOHA são unimodais. Isto deve-se a dois fatores: a espera de *slot* absorve o atraso variável de `sendReceive()`, regularizando o instante de transmissão; e a cadência mais elevada das sessões C2–C3 impede o ChirpStack de enfileirar MAC *commands* sistematicamente em RX1, eliminando a alternância entre retorno antecipado e espera completa que produzia a bimodalidade no Pure ALOHA.

5.2.5 Fiabilidade do Downlink

A PDR de *downlink* (DL-PDR) complementa a PDR de uplink ao validar o caminho inverso: para cada uplink recebido pelo *gateway*, o ChirpStack enfileira um ACK que deve ser entregue ao nó nas janelas RX1 (+1 s) ou RX2 (+2 s) do ciclo Classe A. O nó sinaliza a receção do *downlink* através do campo `flag` do *payload* do uplink seguinte (bit `DL_RECEIVED`, Secção 4.4.5); o servidor cruza este bit com o registo de envio do ChirpStack para calcular a DL-PDR, definida como

$$\text{DL-PDR} = \frac{\text{DL confirmados pelo nó}}{\text{DL tentados pelo } \textit{gateway}}, \tag{5.6}$$

sendo, portanto, condicional: mede apenas os *downlinks* efetivamente tentados, não a totalidade dos uplinks recebidos.

Tabela 5.10: Fiabilidade do downlink por sessão — CW-S-ALOHA.

Sessão	UL recebidos	DL tentados	DL confirmados	DL-PDR (%)
C1	4808	4774	4740	99,3
C2	5482	4908	4901	99,9
C3	5074	4706	4705	99,98

O PDR do downlink situa-se entre 99,3 e 99,98 % nos três níveis de carga, confirmando que as janelas RX1/RX2 são alcançadas com fiabilidade em toda a gama de G testada. O valor aparentemente contra-intuitivo de DL-PDR crescer com a carga (99,3 $\rightarrow$ 99,98 %) é explicado pelo mesmo mecanismo que reduz $\sigma_{e_{\text{slot}}}$ na Secção 5.2.4: em C2–C3, o ciclo mais curto elimina a dormência entre transmissões, tornando a abertura das janelas RX mais determinista e reduzindo a probabilidade de o nó perder o ACK pelo *jitter*.

A coluna UL recebidos estabelece a âncora de referência: os DL tentados correspondem a 89–99 % dos uplinks recebidos pelo servidor, com o défice restante atribuível a pacotes recebidos nos limites de sessão (antes do servidor completar o registo) e a indisponibilidades pontuais do *scheduler* do ChirpStack quando os uplinks chegam com intervalo inferior a 1 s. Em nenhum caso esta diferença afeta a PDR de uplink, que é calculada independentemente a partir do `fCnt`.

A elevada DL-PDR valida ainda que o mecanismo de espera de *slot* do CW-S-ALOHA não perturba o ciclo Classe A: o nó transmite no *slot* alvo, aguarda RX1 (+1 s) e RX2 (+2 s) normalmente, e só então recalcula o próximo *slot* — a sequência é preservada mesmo a $G = 0{,}122$.

5.2.6 Qualidade de Sinal e Estabilidade Temporal

Tabela 5.11: Qualidade de sinal e *jitter* de período: CW-S-ALOHA (médias de sessão).

Sessão	Nó	RSSI (dBm)	SNR (dB)	σ_T (s)	$\sigma_{e_{\text{slot}}}$ (ms)
C1	N1	−66,0	13,0	0,38	59,2
	N2	−67,7	13,1	0,38	54,6
	N3	−53,5	13,2	0,35	54,3
	N4	−59,2	13,1	0,37	55,7
C2	N1	−62,8	13,1	0,33	38,1
	N2	−64,7	12,9	0,37	35,1
	N3	−67,8	13,0	0,36	32,2
	N4	−59,6	12,9	0,33	29,8
C3	N1	−65,2	12,9	0,58	22,8
	N2	−69,1	13,0	0,59	27,1
	N3	−61,1	13,1	0,65	29,6
	N4	−61,3	13,0	0,64	30,4

O SNR mantém-se entre 12,9 e 13,2 dB em todos os nós e sessões, confirmando margem de ligação superior a 50 dB; as perdas observadas têm, por isso, origem exclusivamente protocolar.

O RSSI varia 8–14 dB entre nós, mas a PDR difere apenas 1,6 pp em C1: ao distribuir as transmissões por $N_{CW} = 3$ *slots*, o protocolo elimina a correlação entre RSSI e PDR, pois a probabilidade de dois nós sortearem o mesmo *slot* é $1/N_{CW} = 1/3$, independentemente da potência de sinal.

O *jitter* de período σ_T cresce com a carga: 0,33– 0,38 s em C1–C2 (2–7 % do período), refletindo a variabilidade normal de `sendReceive()`; em C3, sobe para 0,58–0,65 s (18–23 %), por `sendReceive()` tornar-se bimodal entre retorno via RX1 ($\approx$ 1,3 s, com *downlink*) e fecho de RX2 ($\approx$ 2,0 s, sem *downlink*). Este *jitter* propaga-se para $\sigma_{e_{\text{slot}}}$ (25–30 ms em C3), mas é absorvido pela janela do *slot* sem causar transmissões fora de banda.

O *firmware* CW não exporta `offset_raw` para estimação de deriva por regressão linear, ao contrário do Pure ALOHA. A deriva do oscilador ($\pm$20 ppm) introduziria um desvio máximo de $\approx$ 6 ms por *slot*, muito inferior à janela de 300 ms; o seu efeito acumulado está contido em $\sigma_{e_{\text{slot}}}$, métrica agregada de incerteza temporal no CW-S-ALOHA.

5.2.7 Dispersão na Janela de Contenção

O valor de CW de cada pacote é selecionado uniformemente em $\{0, 1, 2\}$ pelo gerador de números pseudo-aleatórios por hardware da ESP32-S3 (`esp_random() % N_CW`). A uniformidade da distribuição é condição necessária para que os N_{CW} *slots* reduzam a colisão por um fator $1/N_{CW}$: se um *slot* fosse sistematicamente favorecido, a vantagem teórica não se materializaria.

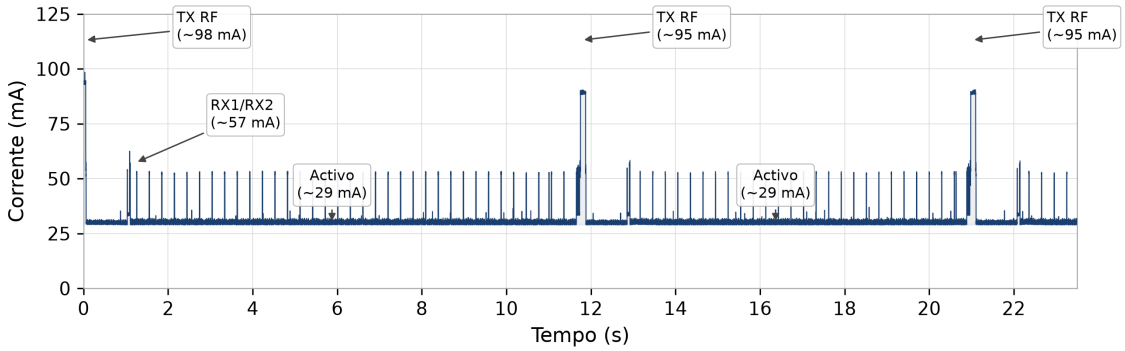
	C1 ($G = 0{,}026$)			C2 ($G = 0{,}072$)			C3 ($G = 0{,}122$)		
Nó	CW0	CW1	CW2	CW0	CW1	CW2	CW0	CW1	CW2
1	34,8	34,5	30,7	33,2	32,3	34,5	33,0	32,8	34,2
2	31,3	32,3	36,4	32,4	34,4	33,1	33,3	32,9	33,8
3	35,0	31,0	34,0	32,9	33,9	33,2	33,2	31,2	35,6
4	33,8	33,6	32,6	32,2	33,1	34,7	30,4	32,0	37,6

Tabela 5.12: Distribuição das transmissões por valor de CW (%): CW-S-ALOHA.

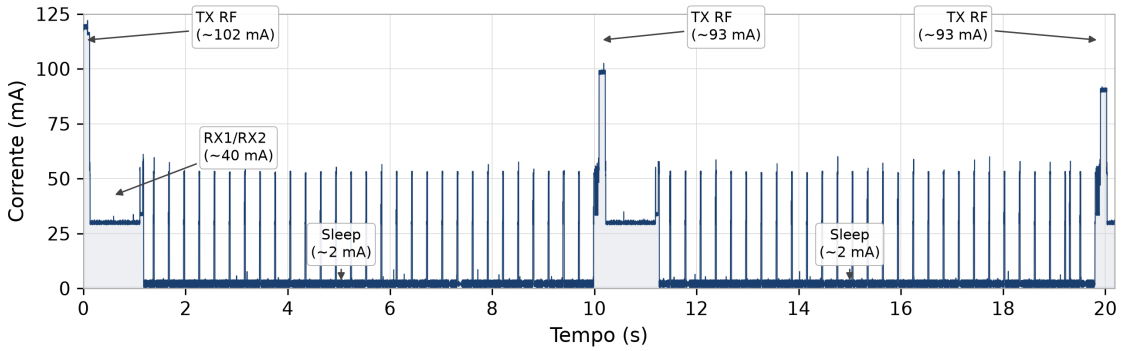
Os desvios máximos face a 33,3 % são de ±3,2 pp em C1 e C2 e de ±4,3 pp em C3, valores compatíveis com a variabilidade estatística. O desvio mais elevado, de 37,6 % no nó 4 para CW=2 em C3, decorre da redução do número de amostras nesta sessão ($N \approx 1100$) e não de um enviesamento sistemático.

5.2.8 Modos de Operação e Energia

Foi conduzido um ensaio independente com um único nó a período fixo de $T = 10$ s, comparando `DEBUG_MODE` com *light sleep*, cujo perfil de corrente é apresentado na Figura 5.8.



(a) `DEBUG_MODE`: corrente de base ≈ 30 mA, com picos periódicos de processamento de *slot* a cada 300 ms. $I_{\text{med}} = 31{,}4$ mA, $T_{\text{obs}} = 11{,}75$ s.



(b) *Light sleep*: corrente de base ≈ 2,1 mA (≈ 81,1 % do ciclo), com picos periódicos de processamento de *slot* a cada 300 ms. $I_{\text{med}} = 10{,}2$ mA, $T_{\text{obs}} = 10{,}09$ s.

Figura 5.8: Perfil de corrente para dois ciclos completos ($T = 10$ s, $T_{\text{slot}} = 300$ ms, $N_{CW} = 3$).

O pico de transmissão (entre 98 e 122 mA e ≈ 115 ms) é idêntico em ambos os modos e compatível com os valores medidos no Pure ALOHA, confirmando que o modo de operação não afeta o comportamento protocolar. O pico secundário visível ≈ 1 s após cada TX corresponde às janelas RX1/RX2 do ciclo Classe A (≈ 61 mA). A corrente de base distingue claramente os dois modos: ≈ 30 mA em `DEBUG_MODE` e $\approx 2{,}1$ mA em *light sleep*. Em ambos os modos são visíveis picos periódicos de curta duração associados ao processamento dos eventos de slot, a cada 300 ms, característica exclusiva do CW-S-ALOHA relativamente ao Pure ALOHA.

O período observado em `DEBUG_MODE` ($T_{\text{obs}} = 11{,}75$ s) é 17,5 % superior ao nominal (10 s), consequência do overhead de processamento de slot sem dormência: cada *wake up* de 300 ms acumula atraso no ciclo de temporização do firmware. Em *light sleep*, o período observado (10,09 s) é praticamente coincidente com o nominal, uma vez que o MCU adormece entre eventos e o temporizador de período não é afetado pelo overhead de slot.

A corrente média por ciclo é obtida por integração trapezoidal sobre N ciclos completos (margem pré-TX excluída do cálculo),

$$I_{\text{med}} = \frac{1}{NT} \int_0^{NT} i(t)\,\mathrm{d}t, \tag{5.7}$$

da qual se extrapola a autonomia para qualquer capacidade de bateria segundo $A = C_{\text{bat}}/I_{\text{med}}$.

Modo	I_{med} (mA)	I_{sleep} (mA)	Fração *sleep* (%)	Autonomia* (h)
`DEBUG_MODE`	31,4	—	0	32
Light sleep	10,2	2,1	81,1	98

Tabela 5.13: Corrente média e estimativa de autonomia com bateria LiPo de 1000 mAh (*$A = C_{\text{bat}}/I_{\text{med}}$) — CW-S-ALOHA, $T = 10$ s.

O modo *light sleep* reduz a corrente média em $\approx 67{,}5$ % (de 31,4 para 10,2 mA), correspondendo a um ganho de autonomia de $3{,}1\times$ (32 para 98 h com 1000 mAh). Este ganho é inferior ao obtido no Pure ALOHA ($4{,}2\times$), o que se explica pelo overhead de processamento dos eventos de slot do CW-S-ALOHA: o nó retoma a atividade a cada $T_{\text{slot}} = 300$ ms para verificar a fila de transmissão, mesmo quando não há pacote a enviar, reduzindo a fração de sleep efetiva face ao Pure ALOHA (81,1 % vs. 82,5 %) e elevando a corrente média de base. A comparação energética direta com o Pure ALOHA é apresentada na Secção 5.3.4.

5.3 Análise Comparativa

Esta secção confronta o CW-S-ALOHA com o Pure ALOHA a cargas equivalentes ($G \approx 0{,}026$, 0,072 e 0,122), sobre a mesma infraestrutura e com os mesmos parâmetros de rádio, isolando assim o contributo exclusivo do mecanismo de acesso ao meio.

5.3.1 PDR: CW-S-ALOHA vs Pure ALOHA

Regime	G_{obs}	PDR_{CW} (%)	PDR_{PA} (%)	ΔPDR (pp)
T1/C1	0,026	97,86	96,17	+1,69
T2/C2	0,072	93,72	86,71	+7,01
T3/C3	0,122	78,41	68,80	+9,61

Tabela 5.14: Comparação de PDR entre CW-S-ALOHA e Pure ALOHA a igual carga.

A vantagem do CW-S-ALOHA cresce monotonicamente com a carga: +1,7 pp em T1/C1, +7,0 pp em T2/C2 e +9,6 pp em T3/C3. A progressão é coerente com a redução teórica de colisão por um fator $N_{CW} = 5{,}8\,\%$ ($P_{\text{col}}^{\text{PA}} \approx 13{,}3\,\%$ vs $P_{\text{col}}^{\text{CW}} \approx 2{,}3\,\%$ em T2/C2). Em T3/C3, o CW-S-ALOHA preserva a vantagem por um motivo diferente: as perdas adicionais por expiração de *slot* distribuem-se uniformemente entre nós, ao contrário das colisões RF do Pure ALOHA, que penalizam desproporcionalmente os nós com RSSI inferior.

5.3.2 Débito: CW-S-ALOHA vs Pure ALOHA

Regime	G	$S_{\text{CW,obs}}$	$S_{\text{PA,obs}}$	Ganho CW (%)	$S_{\text{CW,teo}}$	$S_{\text{PA,teo}}$
T1/C1	0,026	0,0249	0,0245	+1,6	0,0262	0,0251
T2/C2	0,072	0,0605	0,0540	+12,0	0,0698	0,0623
T3/C3	0,122	0,0635	0,0514	+23,5	0,1170	0,0953

Tabela 5.15: Débito normalizado observado e teórico: CW-S-ALOHA vs Pure ALOHA.

O ganho de débito acompanha o ganho de PDR: praticamente nulo em T1/C1, +12,0 % em T2/C2 e +23,5 % em T3/C3. A divergência de ambos face à teoria em T3/C3 tem causas distintas: colisões RF amplificadas pelo *overhead* Classe A no Pure ALOHA; perdas MAC por expiração de *slot* no CW-S-ALOHA (Secção 5.2.3).

5.3.3 Colisões de Canal: CW-S-ALOHA vs Pure ALOHA

	T1/C1 ($G \approx 0{,}026$)		T2/C2 ($G \approx 0{,}072$)		T3/C3 ($G \approx 0{,}122$)	
	CW	PA	CW	PA	CW	PA
Perdas *fCnt* (%)	2,14	3,83	6,28	13,27	21,59	31,20
Colisões canal (eventos)	13	1	18	13	13	6

Tabela 5.16: Perdas e colisões de canal: CW-S-ALOHA vs Pure ALOHA.

As perdas por *fCnt* do CW-S-ALOHA são sistematicamente inferiores às do Pure ALOHA em todos os regimes (ex.: $6,3\%$ vs. $13,3\%$ em T2/C2). A contagem de colisões de canal não é diretamente comparável entre os protocolos, visto que o Pure ALOHA deteta sobreposição temporal no mesmo canal e o CW-S-ALOHA compara *slot targets* declarados.

5.3.4 Consumo Energético: CW-S-ALOHA vs Pure ALOHA

Protocolo	I_{med} (mA)	I_{sleep} (mA)	Fração *sleep* (%)	Autonomia* (h)
Pure ALOHA	7,25	2,0	82,5	138
CW-S-ALOHA	10,2	2,1	81,1	98
Δ (CW$-$PA)	$+2,95$	$+0,1$	$-1,4$ pp	-40

Tabela 5.17: Comparação energética em modo *light sleep*: Pure ALOHA vs CW-S-ALOHA ($T = 10$ s, bateria LiPo 1000 mAh, $^*A = C_{\text{bat}}/I_{\text{med}}$).

O CW-S-ALOHA consome $+40,7\,\%$ mais corrente média (10,2 vs 7,25 mA), reduzindo a autonomia de 138 para 98 h. A corrente de *sleep* é quase idêntica nos dois protocolos (2,0 vs 2,1 mA); o custo adicional resulta da frequência das transições de sono de *slot* ($T_{\text{slot}} = 300$ ms), não da sua duração individual (≈ 4 ms cada). Este custo é o preço da fiabilidade adicional: em T2/C2, cada ponto percentual de ganho de PDR custa $\approx 5,8\,\%$ de corrente média extra. Para carga baixa ($G \approx 0,026$, ganho de PDR de apenas $+1,7$ pp), o Pure ALOHA permanece a opção mais eficiente energeticamente.

5.3.5 Síntese Comparativa

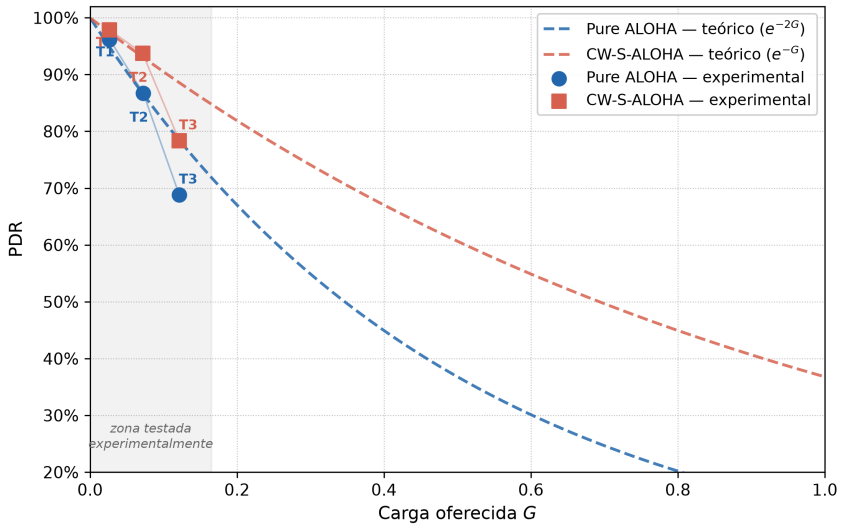


Figura 5.9: PDR experimental vs G: Pure ALOHA e CW-S-ALOHA ($N_{\text{CW}} = 3$), com curvas teóricas.

A Figura 5.9 resume a comparação. O Pure ALOHA acompanha a curva teórica em T1–T2 e desvia-se abaixo em T3, quando o bloqueio de `sendReceive()` deixa de ser negligenciável face ao período. O CW-S-ALOHA supera o Pure ALOHA nos três regimes ($+1{,}7$ a $+9{,}6$ pp de PDR), mas também se desvia da sua própria curva teórica com G crescente, não por colisão RF, mas sim por expiração de pacotes durante o bloqueio síncrono de `sendReceive()`.

5.4 Sumário

O CW-S-ALOHA supera o Pure ALOHA em todos os regimes avaliados, com ganhos de PDR que crescem com a carga: $+1{,}7$ pp em T1/C1 ($G \approx 0{,}026$), $+7{,}0$ pp em T2/C2 ($G \approx 0{,}072$) e $+9{,}6$ pp em T3/C3 ($G \approx 0{,}122$), com correspondentes incrementos de débito de 1,6 %, 12,0 % e 23,5 %. A dispersão inter-nós, de 13,3 pp no Pure ALOHA em T2, é reduzida para 1,7 pp pelo mecanismo de janela de contenção, enquanto a sincronização temporal mantém e_{slot} dentro da janela válida em todos os regimes, validando o mecanismo de *timestamp* piggybacked sem recurso a *beacons* de Classe B.

Por outro lado, este desempenho tem um custo energético de $+40{,}7\,\%$ de corrente média, reduzindo a autonomia de 138 para 98 h com 1000 mAh, resultante das transições periódicas de modo de sono a cada 300 ms. Por conseguinte, em carga baixa, onde o ganho de PDR é marginal, o Pure ALOHA permanece a opção preferível.

Capítulo 6

Conclusões e Trabalho Futuro

6.1 Conclusões

Esta dissertação teve como objetivo principal a avaliação experimental do desempenho de comunicações em tempo real sobre redes LoRaWAN de baixo consumo e longo alcance. Para esse efeito, foram implementados e comparados dois protocolos de acesso ao meio sobre hardware comercial de baixo custo: o Pure ALOHA, que serve de base por reproduzir o comportamento nativo do LoRaWAN Classe A, e o CW-Slotted ALOHA (CW-S-ALOHA), proposto nesta dissertação como extensão do S-LoRaWAN de Polonelli et al. [20]. A questão central foi determinar se a introdução de uma janela de contenção com dispersão temporal *a priori* produz ganhos mensuráveis em fiabilidade de entrega e débito efetivo, sem comprometer a compatibilidade com a especificação LoRaWAN Classe A nem requerer alterações à infraestrutura de rede existente.

A plataforma experimental, descrita em detalhe no Capítulo 4, integrou quatro dispositivos Classe A em SF7/EU868, abrangendo o intervalo de carga $G \approx 0{,}026$ a $G \approx 0{,}122$ em condições de laboratório controladas.

Antes da análise comparativa, a implementação do CW-S-ALOHA foi validada em dois aspetos. O erro de sincronização temporal situou-se dentro do intervalo válido $e_{\text{slot}} \in [-128,\ 172]$ ms em todos os cenários, com uma média entre 60 e 80 ms e um desvio-padrão entre 25 e 56 ms, o que confirma que o mecanismo de ressincronização por *timestamp* piggybacked é suficiente para alinhar *slots* de 300 ms sem recurso a *beacons* de Classe B. Por sua vez, a distribuição de *backoff* apresentou um desvio máximo de $\pm 3{,}2$ pp face à distribuição uniforme teórica, confirmando a correção do gerador de números aleatórios por hardware do ESP32-S3.

A comparação direta entre os dois protocolos revelou que o CW-S-ALOHA superou o Pure ALOHA em PDR e em débito normalizado em todos os cenários avaliados. No regime de carga reduzida (T1/C1, $G \approx 0{,}026$), a diferença em PDR foi de 1,69 pontos percentuais (97,86 % face a 96,17 %) e o incremento de débito foi de 1,6 %, valores expectavelmente modestos uma vez que a probabilidade de colisão é intrinsecamente baixa neste regime. No regime de carga moderada (T2/C2, $G \approx 0{,}072$), o CW-S-ALOHA obteve 93,72 % de PDR face a 86,71 % do Pure ALOHA, correspondendo a um ganho de 7,01 pontos percentuais e a um incremento de débito de 12,0 %.

No regime de carga elevada (T3/C3, $G \approx 0{,}122$), o ganho em PDR atingiu 9,61 pontos percentuais (78,41 % face a 68,80 %) e o débito cresceu 23,5 %. A progressão destes ganhos com a carga confirma que o mecanismo de dispersão *a priori* é tanto mais eficaz quanto maior a densidade de transmissões concorrentes, respondendo diretamente à questão da investigação inicial.

A comparação dos débitos medidos com os modelos teóricos $S_{\mathrm{PA}} = G e^{-2G}$ e $S_{\mathrm{CW}} = G e^{-G/N_{\mathrm{CW}}}$ revelou que ambos os protocolos operam sistematicamente abaixo das curvas analíticas, com rácios $S_{\mathrm{obs}}/S_{\mathrm{teo}}$ entre 53,9 % e 97,6 % para o Pure ALOHA e entre 54,3 % e 95,0 % para o CW-S-ALOHA. O desvio face ao modelo teórico resulta do carácter bloqueante da função `sendReceive()`, que impõe um intervalo mínimo de aproximadamente 1,2 s entre transmissões consecutivas, e das imprecisões de sincronização que originam transmissões marginalmente fora do *slot* nominal. Estes desvios são coerentes com as limitações de implementação identificadas no Capítulo 4 e não comprometem a validade dos modelos analíticos.

A análise de conformidade regulamentar ao abrigo do limite de *duty-cycle* de 1 % para a sub-banda g1 do plano EU868 (ETSI EN 300.220 [11]) mostrou que apenas o regime T1/C1 satisfaz os requisitos em ambos os protocolos, com valores medidos entre 0,60 % e 0,70 %. Os regimes T2 e T3 excedem esse limite em ambos os protocolos, atingindo valores máximos de 2,23 % no Pure ALOHA e de 3,48 % no CW-S-ALOHA em T3. Em todos os regimes, o CW-S-ALOHA apresentou *duty-cycle* superior ao do Pure ALOHA, consequência do *overhead* do *downlink* de sincronização adicionado em cada ciclo e, em T3/C3, das perdas por expiração de *slot* que forçam transmissões adicionais por unidade de tempo.

A análise energética em modo *light sleep* mostrou que o CW-S-ALOHA apresenta uma corrente média de 10,2 mA face a 7,25 mA do Pure ALOHA, correspondendo a um acréscimo de 40,7 % e a uma redução de autonomia de 138 para 98 h com uma bateria de 1000 mAh. Este custo resulta estruturalmente dos wake ups periódicos a cada $T_{\mathrm{slot}} = 300$ ms necessários para o processamento da fila de *slot*, independentemente da existência de pacote a transmitir. O compromisso energia e fiabilidade é favorável nos regimes de carga moderada e elevada, onde cada ponto percentual de ganho de PDR em T2/C2 corresponde a um acréscimo de aproximadamente 5,8 % na corrente média; para carga baixa, onde o ganho de PDR é de apenas 1,69 pp, o Pure ALOHA permanece a opção mais eficiente.

Em síntese, os resultados demonstram que o CW-S-ALOHA proporciona ganhos consistentes de fiabilidade e débito relativamente ao Pure ALOHA, com vantagens que crescem progressivamente com a carga oferecida e se tornam expressivas nos regimes de carga moderada e elevada. A implementação em hardware comercial de baixo custo sobre LoRaWAN Classe A valida a compatibilidade do protocolo com o ciclo de receção previsto pela especificação LoRaWAN, sem requerer qualquer modificação à camada física, à estrutura de tramas ou à infraestrutura de rede. As contribuições principais desta dissertação são a conceção e implementação do CW-S-ALOHA sobre hardware acessível, a validação experimental dos modelos analíticos em condições de laboratório controladas, a caracterização do compromisso energia e fiabilidade e a análise de conformidade com os limites de *duty-cycle* ETSI para ambos os protocolos avaliados.

6.2 Limitações do Trabalho Realizado

Os resultados apresentados foram obtidos numa configuração experimental deliberadamente controlada que, embora favoreça a reprodutibilidade e o isolamento de variáveis, introduz limitações à generalização das conclusões para cenários tipicamente reais.

A rede experimental incluiu apenas quatro nós, número não representativo de implementações LoRaWAN reais em contextos urbanos ou industriais, onde podem coexistir dezenas a centenas de dispositivos por *gateway*. O intervalo de carga explorado ($G \approx 0{,}026$ a $0{,}122$) representa menos de 5 % do pico teórico do CW-S-ALOHA ($G^* = N_{\mathrm{CW}} = 3$), pelo que o comportamento do protocolo na região de saturação, onde a janela de contenção fixa poderá deixar de ser suficiente para dispersar a carga oferecida, não foi caracterizado experimentalmente.

Os ensaios decorreram em ambiente interior, com distâncias inferiores a 10 m entre os nós e o *gateway*, recorrendo a um único *gateway* fixo. Esta configuração elimina a variabilidade de propagação mas não é representativa de implementações exteriores, onde condições de multipercurso, penetração de estruturas e variação de SNR podem degradar a fiabilidade de entrega de forma significativa, e onde a presença de múltiplos *gateways* introduz diversidade de receção que o modelo teórico de canal único não contempla.

Apenas o SF7 foi utilizado nos ensaios. SFs superiores (de SF8 a SF12) implicam ToA consideravelmente mais elevados, agravando o risco de violação do limite de *duty-cycle* e alterando as condições de sobreposição temporal entre transmissões simultâneas. A extensão dos resultados a redes com ADR ativo e múltiplos SFs em operação simultânea requer campanhas experimentais adicionais.

A *stack* LoRaWAN utilizada (RadioLib 6.6.0 em modo `sendReceive()`) bloqueia o microcontrolador durante a transmissão e a receção nas janelas RX1 e RX2, impondo um intervalo mínimo entre transmissões de aproximadamente 1,2 s. Este comportamento limita a carga máxima explorável sem modificação da *stack* e afasta o comportamento da implementação do modelo teórico, que não contempla esta restrição operacional.

Os regimes T2 e T3 excedem o limite de 1 % de *duty-cycle* da sub-banda g1. Embora estes regimes sejam válidos para efeitos de comparação entre protocolos em ambiente laboratorial controlado, não são diretamente transponíveis para implementações regulamentadas na União Europeia sem mitigação adicional, como a distribuição de tráfego por múltiplas sub-bandas ou mecanismos de controlo de acesso ao espetro.

A análise de consumo energético foi realizada com o monitor Monsoon HVPM [18] através do canal USB a 5000 amostras/s. Os valores de autonomia estimados assumem uma bateria de 1000 mAh e tensão constante, sem considerar a variação de tensão ao longo da descarga nem o comportamento do regulador de tensão interno do módulo DFR1195.

Por último, a duração de cada ensaio foi da ordem de horas. A deriva do oscilador de cristal foi quantificada e considerada negligenciável à escala de um ciclo de sincronização (Secção 4.4.6), mas o seu efeito acumulado ao longo de campanhas de duração muito superior, bem como outros fenómenos de longo prazo, nomeadamente os efeitos de limitação por *duty-cycle* acumulado ou a

eventual degradação de desempenho por saturação de filas de transmissão sob carga sustentada, não foram caracterizados neste trabalho.

6.3 Trabalho Futuro

Os resultados desta dissertação apontam para várias extensões que permitem superar as limitações identificadas e aprofundar a compreensão do desempenho de protocolos MAC em redes LPWAN.

A substituição da *stack* bloqueante por uma implementação assíncrona é a extensão de maior impacto imediato. Uma arquitetura baseada em *callbacks* da RadioLib ou numa integração do LoRaMac-node para a plataforma ESP32-S3 permitiria desacoplar o ciclo de transmissão do ciclo principal de aplicação, reduzir o intervalo mínimo entre transmissões e aproximar o comportamento da implementação do modelo teórico. Esta modificação resolveria diretamente a principal causa de divergência identificada em T3/C3, onde o bloqueio de `sendReceive()` ($\approx 1{,}2$ s) responde pela quase totalidade do desvio de aproximadamente 46 % entre o débito observado e o previsto pelo modelo analítico, e é um pré-requisito para a exploração de cargas superiores a $G \approx 0{,}2$ em redes com maior número de nós.

A repetição dos ensaios com um número superior de dispositivos, de 8 a 16 nós, é necessária para aumentar a validade estatística das métricas medidas, validar os modelos analíticos em regimes de carga mais elevada e estudar o comportamento do sistema na região de saturação. Com mais nós, torna-se ainda possível analisar o impacto do desfasamento de fase entre os ciclos de transmissão de diferentes nós na distribuição de colisões e na estabilidade do protocolo.

Relacionada com o ponto anterior, a caracterização experimental da própria região de saturação, com G aproximando-se do pico teórico $G^* = N_{\mathrm{CW}} = 3$, permitiria validar se o modelo $S_{\mathrm{CW}} = Ge^{-G/N_{\mathrm{CW}}}$ se mantém preditivo muito além da gama testada nesta dissertação ($G \leq 0{,}122$, menos de 5 % do pico teórico), ou se emergem, a cargas mais elevadas, mecanismos de degradação adicionais que não foram observados nos regimes avaliados, como por exemplo a saturação da fila circular do nó ou um aumento da probabilidade de colisão intra-*slot* quando o número de nós ativos se aproxima ou excede N_{CW}.

A extensão dos ensaios a múltiplos fatores de alargamento, com e sem ADR ativo, permitiria avaliar o CW-S-ALOHA em condições mais próximas das implementações LoRaWAN reais. Neste contexto, a quase-ortogonalidade entre SFs do LoRa reduz a probabilidade de colisão entre nós que operam em SFs distintos, propriedade que deve ser integrada no modelo analítico para uma comparação rigorosa entre cenários heterogéneos.

A passagem do ambiente de laboratório controlado para um cenário de campo é uma das extensões mais relevantes deste trabalho. A realização de ensaios em ambiente exterior, em cenários urbanos e suburbanos com distâncias ao *gateway* entre 100 m e 1 km, permitiria caracterizar o impacto da propagação real sobre as métricas fundamentais de qualidade de sinal (RSSI e SNR) e relacioná-las diretamente com a fiabilidade de entrega (PDR) de cada protocolo. Em particular, importa verificar se a maior variabilidade de RSSI entre nós, típica de cenários exteriores, degrada

de forma mais acentuada o Pure ALOHA do que o CW-S-ALOHA, e se os ganhos de fiabilidade observados em laboratório se mantêm, atenuam ou acentuam em condições de propagação realistas.

O desenvolvimento de uma versão adaptativa do protocolo, em que o valor de N_{CW} é ajustado dinamicamente com base numa estimativa da carga corrente da rede, representa uma extensão natural deste trabalho. A estimativa de carga pode ser obtida a partir do histórico local de colisões de cada nó, de mensagens de controlo emitidas pelo servidor de rede ou de técnicas de estimação por taxa de ocupação do canal. Uma janela de contenção adaptativa permitiria aproximar o débito do máximo teórico independentemente da densidade momentânea de nós ativos, e constituiria uma contribuição original relevante para a comunidade LoRaWAN.

A caracterização energética em condições de alimentação a bateria, com medições de corrente a alta frequência em ciclos de transmissão reais a diferentes períodos, é necessária para quantificar o custo dos wake ups periódicos de *slot* em função da carga e da duração do ciclo. Esta caracterização é particularmente relevante para avaliar a autonomia de dispositivos em cenários de implementação de longa duração.

Uma alternativa de sincronização a investigar é a migração para dispositivos LoRaWAN Classe B, os quais recebem *beacons* periódicos do *gateway* para abertura de janelas de receção programadas. Esta abordagem eliminaria a dependência do mecanismo de *timestamp* piggybacked no *downlink* de aplicação, reduzindo potencialmente o erro de sincronização medido nesta dissertação ($\mu \in [60, 80]$ ms) para a precisão tipicamente sub-milissegundo dos *beacons* de Classe B. O *tradeoff* a quantificar experimentalmente é, no entanto, significativo: a Classe B exige suporte explícito na infraestrutura de *gateway* e de servidor de rede, nem sempre disponível em implementações de baixo custo, e impõe um consumo energético adicional pela necessidade de manter o recetor ativo nas janelas de *beacon* (*ping slots*), de forma análoga ao custo de $+40{,}7\,\%$ de corrente média já identificado nesta dissertação para os wake ups de *slot* do CW-S-ALOHA em Classe A. Resta por isso determinar, em trabalho futuro, se o ganho de precisão de sincronização da Classe B se traduz em ganhos de PDR e de débito suficientes para justificar o seu custo energético e de infraestrutura adicional, ou se o mecanismo de *timestamp* piggybacked desenvolvido nesta dissertação, mais simples e sem requisitos de infraestrutura adicional, já representa um compromisso eficiente para a generalidade dos cenários de aplicação.

Por fim, a comparação com outros protocolos MAC propostos para redes LPWAN, como CSMA/CA com controlo de *duty-cycle*, TDMA com sincronização por GPS ou protocolos baseados em reserva de *slot* por negociação, como o TS-LoRa [32], permitiria posicionar o CW-S-ALOHA no espaço de soluções disponíveis. Esta comparação beneficiaria da mesma plataforma experimental desenvolvida nesta dissertação, reduzindo a variabilidade introduzida por diferenças de hardware e de configuração de rede, e contribuiria para uma avaliação mais abrangente das alternativas ao acesso aleatório puro em redes LoRaWAN.

Referências

[1] Norman Abramson. «The ALOHA System: Another Alternative for Computer Communications». Em: *Proceedings of the Fall Joint Computer Conference*. ACM, 1970, pp. 281–285. DOI: 10.1145/1478462.1478502.

[2] F. Adelantado et al. «Understanding the Limits of LoRaWAN». Em: *IEEE Communications Magazine* 55.9 (set. de 2017), pp. 34–40. ISSN: 0163-6804. DOI: 10.1109/MCOM.2017.1600613.

[3] Luís Almeida e Pedro Santos. *Industrial Communications — Lecture 21: Long-Range Vertical Solutions*. Lecture slides, Faculty of Engineering, University of Porto (FEUP). Adapted figures used in Sections 2.2 and 2.3. 2024.

[4] A. Augustin et al. «A Study of LoRa: Long Range & Low Power Networks for the Internet of Things». Em: *Sensors* 16.9 (2016), p. 1466.

[5] M. C. Bor et al. «Do LoRa Low-Power Wide-Area Networks Scale?» Em: *Proceedings of the 19th ACM MSWiM*. 2016.

[6] I. Butun, N. Pereira e M. Gidlund. «Security Risk Analysis of LoRaWAN and Future Directions». Em: *Future Internet* 11.1 (2019), p. 3.

[7] M. Capuzzo, D. Magrin e A. Zanella. «Confirmed Traffic in LoRaWAN: Pitfalls and Countermeasures». Em: *2020 17th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. 2020, pp. 1–9.

[8] L. Casals et al. «Modeling the Energy Performance of LoRaWAN». Em: *Sensors* 17.10 (2017), p. 2364.

[9] Daniele Croce et al. «Impact of LoRa Imperfect Orthogonality: Analysis of Link-Level Performance». Em: *IEEE Communications Letters* 22.4 (2018), pp. 796–799. DOI: 10.1109/LCOMM.2018.2797057.

[10] DFRobot. *DFR1195 – ESP32-S3 LoRaWAN Development Board*. Acedido em abril de 2026. 2024. URL: https://wiki.dfrobot.com/dfr1195/.

[11] *ETSI EN 300 220-1: Short Range Devices (SRD); Radio equipment to be used in the 25 MHz to 1 000 MHz frequency range with power levels ranging up to 500 mW*. ETSI, 2017.

[12] M. Haubro et al. «TSCHover-LoRA: Long Range and Reliable IPv6 Multi-Hop Networks for the Internet of Things». Em: *Internet Technology Letters* 3.4 (mai. de 2020), e165. DOI: 10.1002/itl2.165.

[13] J. Haxhibeqiri et al. «A Survey of LoRaWAN for IoT: From Technology to Application». Em: *Sensors* 18.11 (2018), p. 3995.

[14] Dong-Hoon Kim, Eun-Kyu Lee e Jibum Kim. «Experiencing LoRa Network Establishment on a Smart Energy Campus Testbed». Em: *2019 International Conference on Information and Communication Technology Convergence (ICTC)*. 2019, pp. 1–6. DOI: 10.1109/ICTC46691.2019.8939963.

[15] Rachel Kufakunesu, Gerhard P. Hancke e Adnan M. Abu-Mahfouz. «A Survey on Adaptive Data Rate Optimization in LoRaWAN: Recent Solutions and Major Challenges». Em: *Sensors* 20.18 (2020), p. 5044. DOI: 10.3390/s20185044.

[16] LoRa Alliance. *LoRaWAN Specification v1.0.3*. Acedido em março de 2026. 2018. URL: https://lora-alliance.org/wp-content/uploads/2020/11/lorawan1.0.3.pdf.

[17] LoRa Alliance. *LoRaWAN Specification v1.1*. Acedido em março de 2026. 2017. URL: https://lora-alliance.org/wp-content/uploads/2020/11/lorawantm_specification_-v1.1.pdf.

[18] Monsoon Solutions, Inc. *High Voltage Power Monitor (HVPM)*. Acedido em junho de 2025. 2023. URL: https://www.msoon.com/online-store/High-Voltage-Power-Monitor-p90002590.

[19] Monsoon Solutions, Inc. *Mobile Device Power Monitor Manual, Ver. 1.20*. Acedido em junho de 2025. 2019. URL: https://msoon.github.io/powermonitor/PowerTool/doc/Power%20Monitor%20Manual.pdf.

[20] T. Polonelli et al. «Slotted ALOHA on LoRaWAN—Design, Analysis, and Deployment». Em: *Sensors* 19.4 (2019), p. 838.

[21] U. Raza, P. Kulkarni e M. Sooriyabandara. «Low Power Wide Area Networks: An Overview». Em: *IEEE Communications Surveys & Tutorials* 19.2 (2017), pp. 855–873.

[22] B. Reynders et al. «Improving Reliability and Scalability of LoRaWANs Through Lightweight Scheduling». Em: *IEEE Internet of Things Journal* 5.3 (2018), pp. 1830–1842.

[23] Lawrence G. Roberts. «ALOHA packet system with and without slots and capture». Em: *ACM SIGCOMM Computer Communication Review* 5.2 (1975). Apresentado originalmente na ARPANET Conference, 1972, pp. 28–42. DOI: 10.1145/1024916.1024920.

[24] Seeed Studio. *SenseCAP Indoor Gateway Datasheet*. Acedido em junho de 2025. 2023. URL: https://files.seeedstudio.com/Bazaar/product_pdf/114992981.pdf.

[25] Semtech Corporation. *LoRa Modem Designer's Guide: AN1200.13*. Rel. téc. AN1200.13. Rev. 1. Semtech Corporation, 2013.

[26] Semtech Corporation. *LoRa Modulation Basics: AN1200.22*. Rel. téc. AN1200.22. Semtech Corporation, 2015.

[27] Semtech Corporation. *SX1261/2 Long Range, Low Power, sub-GHz RF Transceiver – Datasheet*. Rel. téc. DS.SX1261-2.W.APP. Rev. 2.2. Semtech Corporation, 2024.

[28] Semtech Corporation. *SX1272/73 - 860 MHz to 1020 MHz Low Power Long Range Transceiver*. Datasheet Rev. 3.1. Datasheet disponível no website do fabricante. Accessed: 2024-01-15. Semtech Corporation, mar. de 2013. URL: https://www.semtech.com/products/wireless-rf/lora-transceivers.

[29] J. Shuda, A. J. Rix e M. J. Booysen. «Module-Level Monitoring of Solar PV Plants Using Wireless Sensor Networks». Em: *IEEE Sensors Journal* 18.6 (2018), pp. 2550–2558. DOI: 10.1109/JSEN.2018.2799597.

[30] The Things Network. *LoRaWAN Classes Overview*. Acedido em março de 2026. 2024. URL: https://www.thethingsnetwork.org/.

[31] T. Voigt et al. «Mitigating Inter-Network Interference in LoRa Networks». Em: *Proceedings of EWSN*. 2017.

[32] D. Zorbas et al. «TS-LoRa: Time-Slotted LoRaWAN for the Industrial Internet of Things». Em: *Computer Communications* 153 (2020), pp. 1–10.

DECLARAÇÃO DE HONRA

Declaro que o presente trabalho de dissertação é de minha autoria e não foi previamente apresentado e avaliado noutro curso ou unidade curricular, desta ou de outra instituição.

As referências a outros autores (afirmações, ideias, pensamentos), assim como a utilização de trabalhos publicados respeitam escrupulosamente as regras da atribuição, e encontram-se devidamente indicadas no texto e nas referências bibliográficas, de acordo com as normas de referenciação. Tenho consciência de que a prática de plágio e auto-plágio constitui um ilícito académico.

Declaro, ainda, que, sempre que utilizei ferramentas de Inteligência Artificial como apoio a parte(s) da presente dissertação, essa utilização e seus resultados estão devidamente assinalados e foram objeto de cuidada verificação para deteção de erros, imprecisões, atribuições infundadas ou outras formas de enviesamento de conteúdos e argumentos, bem como de determinação de autoria.

Eduardo Soqueiro Barbosa