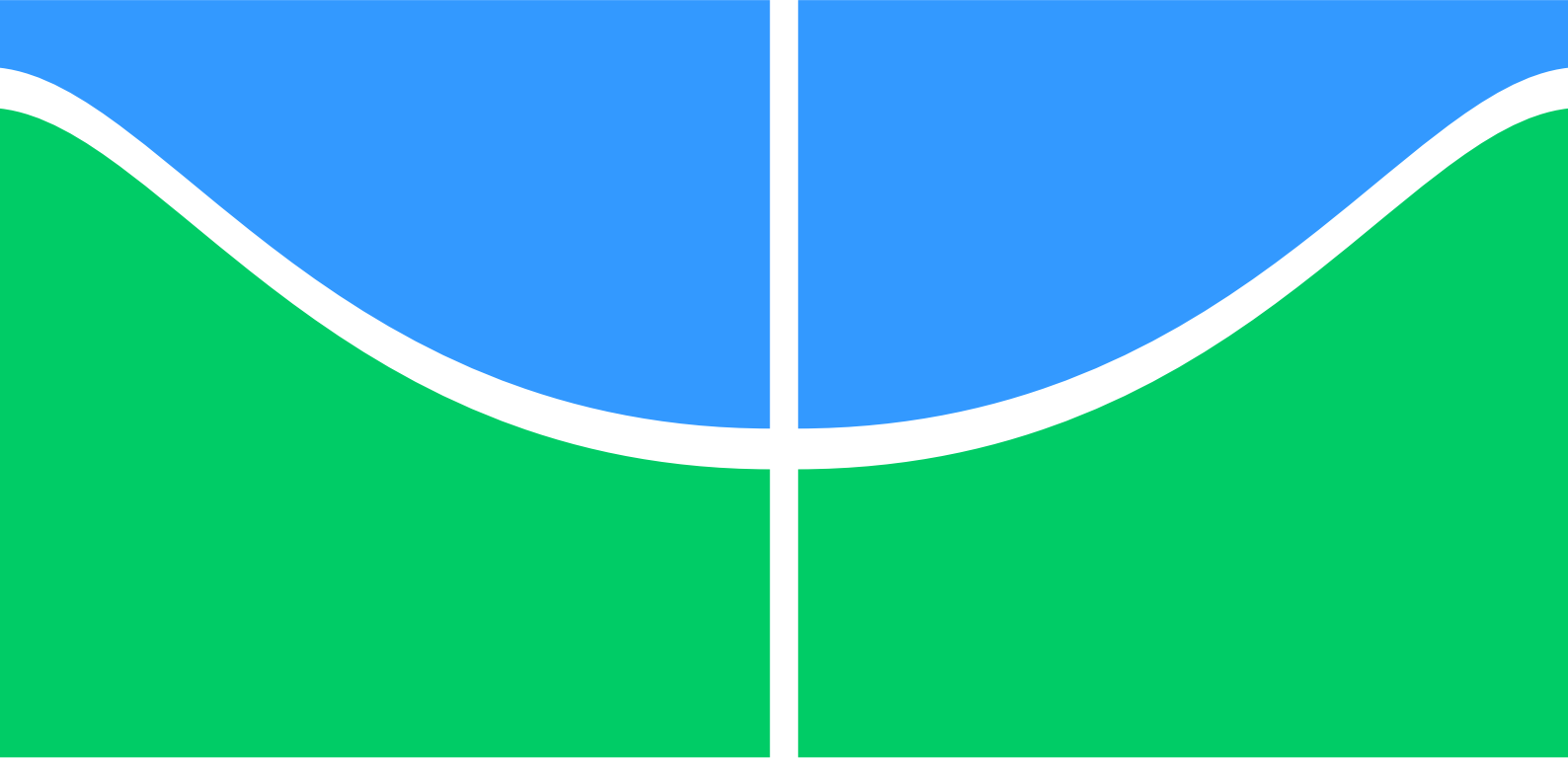

ESTUDO DE LOCALIZAÇÃO COM Wi-Fi NA FAIXA DE 60 GHZ

ARTHUR OLIVEIRA SOUZA DA COSTA

FELIPE SOUSA ROCHA

PROJETO FINAL DE GRADUAÇÃO EM ENGENHARIA DE REDES DE COMUNICAÇÃO

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

ESTUDO DE LOCALIZAÇÃO COM Wi-Fi NA FAIXA DE 60 GHZ

ARTHUR OLIVEIRA SOUZA DA COSTA

FELIPE SOUSA ROCHA

Orientador: PROF. DR. LEONARDO AGUAYO, ENE/UNB

PROJETO FINAL DE GRADUAÇÃO EM ENGENHARIA DE REDES DE COMUNICAÇÃO

BRASÍLIA-DF, 13 DE MAIO DE 2022.

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

ESTUDO DE LOCALIZAÇÃO COM Wi-Fi NA FAIXA DE 60 GHZ

ARTHUR OLIVEIRA SOUZA DA COSTA

FELIPE SOUSA ROCHA

PROJETO FINAL DE GRADUAÇÃO ACADÊMICO SUBMETIDO AO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA, COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE GRADUAÇÃO EM ENGENHARIA DE REDES DE COMUNICAÇÃO.

APROVADA POR:

Prof. Dr. Leonardo Aguayo,
ENE/UnB
Orientador _____

Prof. Dr. Paulo Henrique Portela de Carvalho, ENE/UnB
Examinador interno _____

Prof. Dr. Paulo Roberto de Lira Gondim, ENE/UnB
Examinador interno _____

BRASÍLIA, 13 DE MAIO DE 2022.

FICHA CATALOGRÁFICA
ARTHUR OLIVEIRA SOUZA DA COSTA
FELIPE SOUSA ROCHA
ESTUDO DE LOCALIZAÇÃO COM Wi-Fi NA FAIXA DE 60 GHZ
2022xiii, 63p., 201x297 mm
(ENE/FT/UnB, Graduação, ENGENHARIA DE REDES DE COMUNICAÇÃO, 2022)
Projeto Final de Graduação - Universidade de Brasília
Faculdade de Tecnologia - Departamento de Engenharia Elétrica

REFERÊNCIA BIBLIOGRÁFICA

ARTHUR OLIVEIRA SOUZA DA COSTA; FELIPE SOUSA ROCHA (2022) ESTUDO DE LOCALIZAÇÃO COM Wi-Fi NA FAIXA DE 60 GHZ. Projeto Final de Graduação em ENGENHARIA DE REDES DE COMUNICAÇÃO, Publicação xxx/AAAA, Departamento de ENGENHARIA DE REDES DE COMUNICAÇÃO, Universidade de Brasília, Brasília, DF, 63p.

CESSÃO DE DIREITOS

AUTORES:

Arthur Oliveira Souza da Costa

Felipe Sousa Rocha

TÍTULO: ESTUDO DE LOCALIZAÇÃO COM Wi-Fi NA FAIXA DE 60 GHZ.

GRAU: Graduação ANO: 2022

_____

Arthur Oliveira Souza da Costa
Universidade De Brasília

_____

Felipe Sousa Rocha
Universidade De Brasília

Agradecimentos

Gostaria de agradecer aos meus pais pelo suporte e apoio em toda minha jornada na faculdade. Também gostaria de agradecer aos inúmeros colegas que me acompanharam nos momentos alegres e tristes e também aos professores da Faculdade de Tecnologia pelo suporte e disposição, além dos vários ensinamentos que levarei sempre comigo.

Arthur Costa

Agradeço a todos que contribuíram para essa conquista. Agradeço também a tudo que me deu força nos momentos em que pensei em desistir.

Felipe Sousa

Resumo

Este documento descreve um estudo de localização de dispositivos em redes Wi-Fi IEEE 802.11ay na banda de 60 GHz. Uma das técnicas utilizadas pelo padrão é a formatação de feixes (*beamforming*), que permite o controle do posicionamento angular do feixe principal de um arranjo de antenas. O estudo também explora a informação de *timestamp* nos pacotes para estimar o tempo de propagação do sinal.

A utilização conjunta dos recursos de *beamforming* e de *timestamps* permitiu a elaboração de um método simplificado de localização, avaliado com base na simulação de eventos discretos utilizando a ferramenta ns-3. A simulação considerou 4 (quatro) cenários de controle permitindo concluir que o uso conjunto destes dois recursos permite localizar a estação móvel em condições favoráveis de propagação com linha de visada.

Verificou-se que a marcação de *timestamp* em pacotes inseridos na camada de aplicação por si só não é suficiente para determinar com precisão o tempo de propagação. O método utilizado baseou-se no uso de variáveis de simulação para obter uma melhor estimativa do tempo de atraso entre o momento em que o transmissor envia um pacote e o momento em que este pacote chega ao receptor. Dentro dos cenários analisados, a acurácia na localização foi da ordem de 30 cm.

Abstract

This document describes a study of device location on IEEE 802.11ay Wi-Fi networks in the 60 GHz band. One of the techniques used by the standard is beamforming, which allows controlling the angular positioning of the main beam of an antenna array. The study also exploits the timestamp information in the packets to estimate the signal propagation time.

The joint use of beamforming and timestamps resources allowed the elaboration of a simplified location method, evaluated based on the simulation of discrete events using the ns-3 tool. The simulation considered 4 (four) control scenarios, allowing to conclude that the joint use of these two resources allows locating the mobile station in favorable conditions of propagation with line of sight.

It was found that packet timestamping at the application layer by itself is not enough to determine propagation time. The method used had to rely on simulation variables in order to obtain the time delay between the time transmitter sends a packet and the time that this packet reaches the receiver. Among the analyzed scenarios, the accuracy in the location was of the order of 30 cm.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE CÓDIGOS FONTE

LISTA DE SÍMBOLOS

c Velocidade da luz
λ Comprimento de onda
k Número de onda
θ Ângulo de azimute
φ Ângulo de elevação

LISTA DE TERMOS E SIGLAS

A-BFT	*Association Beamforming Training*
AP	*Access Point*
BHI	*Beacon Header Interval*
BI	*Beacon Interval*
BRP	*Beamforming Protocol*
BTF	*Beamforming Training*
BTI	*Beacon Transmission Interval*
CBAPs	*Contention-Based Access Periods*
CEF	*Channel Estimate Field*
CSMA/CA	*Carrier Sense Multiple Access with Collision Avoidance*
DMG	*Directional Multi-Gigabit*
DTI	*Data Transmission Interval*
EDMG AP	*Enhanced Directional Multi-Gigabit Access Point*
EDMG STA	*Enhanced Directional Multi-Gigabit Station*
GI	*Guard Interval*
I-RXSS	*Responder Transmit Sector Sweep*
I-TXSS	*Initiator Transmit Sector Sweep*
IEEE	*Institute of Electrical and Electronics Engineers*
IoT	*Internet of Things*
L-CEF	*Legacy Channel Estimate Field*
L-STF	*Legacy Short Training Field*

LOS	*Line of Sight*
MCS	*Modulation and Coding Scheme*
MIMO	*Multiple-Input Multiple-Output*
mmWave	*Millimeter-Wave*
MPC	*Multipath Component*
NIST	*National Institute of Standards and Technology*
NLOS	*Non Line of Sight*
OFDM	*Orthogonal Frequency Division Multiplexing*
PAA	*Phased Antenna Array*
RTT	*Round Trip Time*
SC	*Single Carrier*
SINR	*Signal to Interference Plus Noise Ratio*
SISO	*Single-Input Single-Output*
SLS	*Sector Level Sweep*
SNR	*Signal to Noise Ratio*
SPs	*Service Periods*
SSW	*Sector Sweep*
STA	*Station*
STF	*Short Training Field*
WiGig	*Wireless Gigabit*
WLAN	*Wireless Local Area Network*

Capítulo 1

Introdução

1.1 Motivação

Desde que foi estabelecida, a tecnologia *Wireless Local Area Network* (WLAN) tornou-se predominante no cotidiano, seja em ambiente residencial ou profissional. A mais difundida segue o padrão 802.11, que está em constante desenvolvimento a partir dos *Working Groups*, grupos pertencentes ao Institute of Electrical and Electronics Engineers (IEEE), responsáveis pelo desenvolvimento e projeto de padrões em diversas áreas da tecnologia.

No entanto, apesar de amplamente difundido, ainda há bastante espaço para implementação de melhorias e, consequentemente, buscar novas aplicações. Uma das novas perspectivas é a comunicação de dados na faixa de 60 GHz, permitindo velocidades de transferência superiores quando comparado com os padrões em faixas de 2,4 GHz e 5 GHz atuais. No entanto, atingir um desempenho satisfatório não é simples. O menor comprimento de onda das ondas milimétricas (*Millimeter-Wave* - mmWave) impõe dificuldades adicionais devido à dependência da frequência nos efeitos de desvanecimento do sinal, em especial nos ambientes fechados sujeitos a obstruções e múltiplas reflexões.

Posto isto, surge o conceito de *Wireless Gigabit* (WiGig), inicialmente definido a partir do padrão 802.11ad e em seguida, a versão com melhorias, o 802.11ay. O salto tecnológico também se faz presente nas redes celulares e abre portas para novas aplicações e oportunidades ainda não exploradas. O artigo *"6G White Paper on Localization and Sensing"* [Bourdoux et al. 2020] traz novidades relacionadas à aplicações de localização e redes de detecção com precisão a partir da chegada do 6G no futuro, mas que também podem ser exploradas com o 5G, que por sua vez, já é realidade. A ideia é aproveitar as altas frequências das mmWave e outras tecnologias como *Multiple-Input Multiple-Output* (MIMO) e Internet das Coisas (*Internet of Things* - IoT) para desenvolver novos sensores de alerta, navegação e otimização de redes de comunicação, com o eventual auxílio de redes WiGig.

1.2 Objetivos

A padronização do 802.11ay é relativamente recente. Em fevereiro de 2021, a versão de revisão e aprovação do documento que descreve o padrão foi submetida ao comitê avaliador do IEEE, e publicada em Julho do mesmo ano [IEEE 2021c] [IEEE 2021a]. Ainda em 2018, a empresa desenvolvedora de *chipsets* Qualcomm anunciou a chegada dos novos modelos de *chips*, as famílias QCA64x8 e QCA64x1, designadas para viabilizar o desenvolvimento de tecnologias de conectividade na faixa de 60 GHz [Qualcomm 2018].

O difícil acesso aos equipamentos de *hardware* compatíveis com o 802.11ay, motivado pelo curto tempo desde o lançamento desta padronização, contribuiu para a decisão de realizar este trabalho em ambiente simulado. O ns-3 é uma ferramenta de simulação de rede desenvolvida em código aberto amplamente utilizada para replicar o funcionamento de redes de comunicação, por meio da reprodução do comportamento de dispositivos, fluxos de dados e protocolos, por exemplo. Diversas tecnologias de comunicação podem ser exploradas com este simulador. Uma área que oferece bastante recursos de simulação é a parte de Wi-Fi, que já tem suporte para as gerações a, b, g, n, ac e ax, na versão oficial atual do repositório, a 3.35 [ns-3 2021b].

Por ser desenvolvido em código aberto e por seu potencial de utilização, o ns-3 atrai interessados em desenvolver versões simuladas das tecnologias de comunicação existentes. Na edição de 2021 do evento de conferência de *workshop* oficial do ns-3, o WNS3, um grupo vinculado ao *National Institute of Standards and Technology* (NIST) apresentou uma publicação que trata da implementação do 802.11ay a partir do código do ns-3, o artigo *"Implementation and Evaluation of a WLAN IEEE 802.11ay Model in Network Simulator ns-3"* [Assasa et al. 2021]. O código referente a esta publicação está disponível em [wigig-tools 2021c], e conta com simulações de exemplo utilizadas para validar o funcionamento dos recursos presentes no módulo, como MIMO e agrupamento de canais.

Assim, o trabalho irá explorar as funcionalidades disponíveis neste *framework*, chamado Wigig-Module, para fazer estudos de localização. De maneira geral, serão elaborados códigos com o objetivo de obter informações suficientes da conexão simulada entre uma *Station* (STA) e um *Access Point* (AP), para que este segundo consiga estimar com certo nível de precisão a localização do primeiro. Para tanto, será explorado o recurso de *beamforming* para estimativa de direção de feixe e a marcação de tempo (*timestamp*) inserida nos pacotes para estimativa da distância.

1.3 Trabalhos Relacionados

Com a chegada da tecnologia 5G e dos padrões IEEE 802.11 na faixa de frequências mmWave, os sistemas e aplicações podem usufruir de uma maior quantidade de dados em menor tempo, sendo assim novos sistemas inteligentes e dispositivos podem atingir objetivos

e realizar tomada de decisões baseadas em uma base de dados rica ou a partir de medições precisas. O cenário é promissor para aplicações de localização e novas técnicas estão em desenvolvimento atualmente para fazer o melhor uso possível da grande quantidade de informação disponível.

Vários trabalhos e projetos de localização *indoor* usam o Wi-Fi como base, redes locais sem fio estão presentes nos mais diversos ambientes e os sistemas de localização atuais são compatíveis com redes sem fio. As técnicas envolvem parâmetros como ângulos de chegada das ondas, potência recebida, formatação de feixes e implementação de antenas inteligentes. O trabalho [Chabbar and Chami 2017] traz alguns projetos em andamento para localização de usuários e mapeamento de regiões por meio de registros de potência recebida de vários pontos de acesso.

Tratando de técnicas de aproveitamento de dados e treinamento pode-se citar o aprendizado de máquina (*Machine Learning* - ML) como uma poderosa ferramenta. Os algoritmos de classificação surgem como potenciais formas de localização a partir de uma base de dados envolvendo parâmetros do canal, informações transmitidas, informação de ruído e características do ambiente, por exemplo. Sendo assim, vários cenários diferentes podem ser usados para compor banco de dados para treinamento, permitindo ao sistema aprender conforme os dados mudam e se relacionam entre si [Bourdoux et al. 2020].

Em [Chen et al. 2020] são apresentados métodos de aprendizagem de máquina voltados para mapeamento e localização. A partir de uma base de dados sólida composta pelos parâmetros apresentados anteriormente, pode-se implementar sistemas de classificação e até mesmo determinar posições dentro de uma margem de erro de xx metros. A necessidade de uma grande quantidade de dados bem treinados e um alto poder de processamento dos sistemas é um dos principais entraves. Posto isto, a coleta de dados não pode ser muito arbitrária também, mas deve contemplar o máximo de casos e cenários possíveis [Bourdoux et al. 2020].

Novos modelos e abordagens para construção de algoritmos de aprendizagem surgem como principal meta de solução para os principais obstáculos de implementação da técnica. As classificações podem determinar desde posições no espaço através de modelos de regressão até mesmo identificar dispositivos na rede por meio de um sistema de classificação baseado nas características dos dados[Sobehy 2020].

Outro método bastante explorado é a triangulação. Este método baseia-se na medição do ângulo de chegada, valor que indica a posição angular $(\theta,\ \varphi)$ com que a onda eletromagnética chega na antena receptora com a maior potência. Também é válido perceber que o método pode fazer proveito do *beamforming*, de tal modo que o transceptor tenha controle do apontamento da direção dos feixes. Diferentemente do aprendizado de máquina, não há necessidade de coletar dados previamente [Chan and Sohn 2012].

No entanto, na triangulação são necessários no mínimo três dispositivos na rede. Um cenário possível seria com o uso de duas STA e um AP na rede de comunicação: a partir da

distância entre as duas STA e os ângulos de chegada na antena do AP é possível obter um triângulo formado pelas linhas que definem a distância do AP para as estações, permitindo obter sua posição. Outro cenário possível, que conta apenas com colaboração de APs para localizar STA, pode ser como apresentado na Figura 1.1.

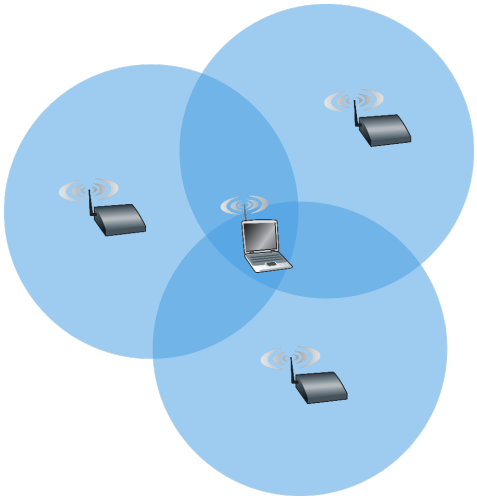

Figura 1.1: Exemplo de triangulação com 3 APs (Adaptado de [James W. Kurose 2021]).

1.4 Organização

O trabalho está organizado como segue. Capítulo 2 apresenta detalhes e explicações sobre a arquitetura e funcionamento do padrão 802.11ay. O Capítulo 3 introduz o método adotado para a localização e os repositórios principais para execução das simulações. No Capítulo 4 estão detalhados os vários processos para atender a proposta, incluindo as tentativas por meio do desenvolvimento das aplicações. Os parâmetros envolvidos nas simulações, os resultados e avaliações dos cenários estão apresentados no Capítulo 5. Por fim, a conclusão apresenta as considerações finais e trabalhos futuros propostos.

Capítulo 2

Redes Locais no padrão IEEE 802.11ay

2.1 Evolução do padrão IEEE 802.11

O padrão 802.11 é a principal referência para redes locais sem fio (*wireless*). O padrão IEEE 802.11 (Wi-Fi) é dividido em várias gerações que compartilham funcionalidades como, por exemplo, o protocolo de acesso ao meio *Carrier Sense Multiple Access with collision avoidance* (CSMA/CA) e a retrocompatibilidade. As diferenças notórias se encontram especialmente na camada física. As versões iniciais 802.11b e 802.11g fazem uso da banda de 2,4 GHz até que a chegada do 802.11n introduziu a característica *dual-band* permitindo, além da banda de 2,4 GHz, a banda de 5 GHz. A Figura 2.1 apresenta o aumento da largura de faixa dos canais utilizados em redes Wi-Fi nas bandas de 2,4 GHz, 5 GHz e 60 GHz.

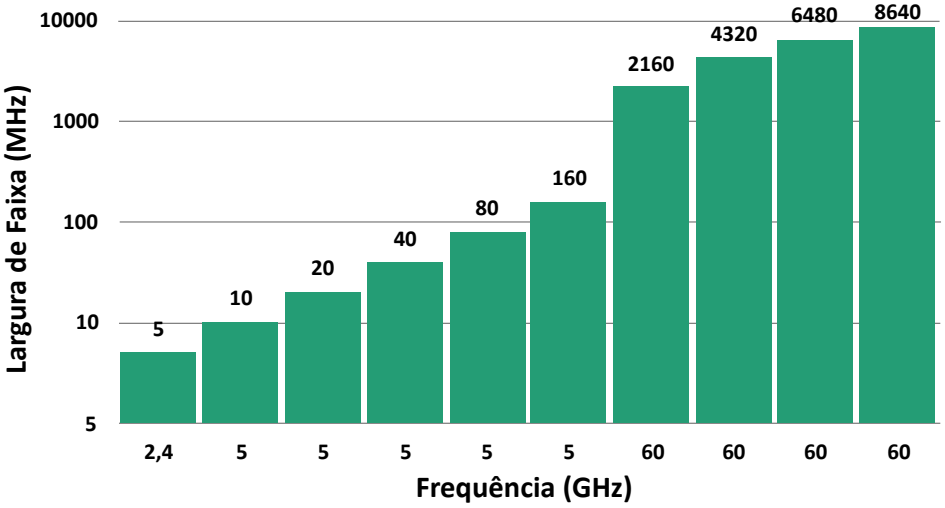

Figura 2.1: Aumento da largura de faixa em redes Wi-Fi.

A faixa de frequências em torno de 60 GHz é explorada por outros padrões do 802.11. Seja c a velocidade da luz aproximada para $3 \times 10^8\, m/s$, o comprimento de onda λ para essa frequência é conforme o cálculo que segue:

$$\lambda = \frac{c}{f} = \frac{3 \cdot 10^8}{60 \cdot 10^9} = 5\,mm. \tag{2.1}$$

Assim, por seu comprimento próximo de 5 milímetros, as ondas nas proximidades da faixa referida são chamadas ondas milimétricas, ou *mmWaves*. O padrão 802.11ad foi o primeiro a operar no espectro das *mmWaves* na frequência de 60 GHz, o que, por si só, muda completamente as condições de propagação quando comparado com outras versões. Inicialmente, os ganhos na taxa de transmissão foram animadores, sendo possível obter até 7 Gbits/s de velocidade em redes de curto alcance [Nitsche et al. 2014]. Contudo, as transmissões em 60 GHz têm alcance menor ao consumir a mesma energia. Tomando a equação de perda no espaço livre, onde d é a distância do transmissor ao receptor:

$$L_{fs_{[dB]}} = 92,4 + 20\log_{10} f_{[GHz]} + 20\log_{10} d_{[km]}. \tag{2.2}$$

para a distância de 1 metro, é possível verificar que os sinais de 2,4 GHz, 5 GHz e 60 GHz sofrem atenuação L_{fs} de 40; $46,38$; e $67,96$ dB, respectivamente. Desta forma, a alta atenuação presente na frequência de 60 GHz, em comparação com as demais citadas, reduz significativamente a capacidade de penetração de obstáculos e a da comunicação de dispositivos fora do campo de visão. A Figura 2.2 apresenta um gráfico comparando as atenuações em espaço livre para as frequências citadas com relação à distância.

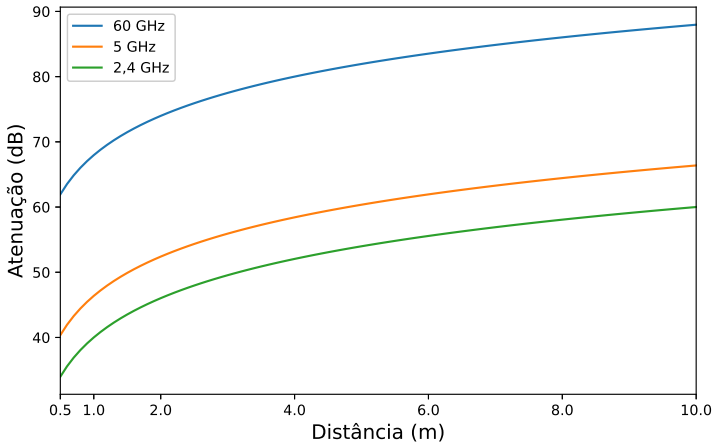

Figura 2.2: Atenuação no espaço livre para 60, 5 e 2.4 GHz.

Para descrever a geometria geral do enlace de comunicação, dois termos são usualmente empregados: LOS (*Line of Sight*) e NLOS (*Non Line of Sight*). O primeiro nomeia os cenários onde não há estruturas que bloqueiem o caminho entre transmissor e receptor. O segundo nomeia o caso contrário.

Desta forma, o sinal *mmWave* aplicado em transmissores de baixo consumo terá curto alcance, o que pode ser visto como vantajoso ao considerar a redução de interferência co-canal em dispositivos que estejam mais distantes. Entretanto, dado que o principal objetivo

é transmitir dados, técnicas como o agrupamento de canais e a formatação de feixes serão aplicadas para aumentar a eficiência da comunicação. Múltiplos elementos de antena transmitindo ao mesmo tempo permitem formar um feixe mais estreito que pode ser direcionado para diferentes posições por meio do deslocamento de fase do sinal transmitido.

O 802.11ay representa o próximo passo da evolução da tecnologia operando na banda de 60 GHz, dos sistemas e dispositivos *Directional Multi-Gigabit* (DMG), representando a base dos sistemas de comunicação *Enhanced Directional Multi-Gigabit* (EDMG) [Assasa et al. 2021]. O padrão aproveita várias das especificações de seu antecessor, o 802.11ad, mas o detalhe está no aproveitamento de uma maior largura de banda e a implementação de duas cadeias de rádio frequência *Radio Frequency Chains* por antena permitindo até 8 fluxos espaciais, possibilitando taxas de transmissão de dados combinadas de até 100 Gbits/s [IEEE 2021b]. O repositório no ns-3, no entanto, possui uma limitação de apenas uma *RF chain* por antena, sendo assim, apenas 4 fluxos espaciais podem ser configurados [Assasa et al. 2021]. Os dispositivos compatíveis com o padrão 802.11ay são definidos como *Enhanced Directional Multi-Gigabit Stations* (EDMG STA) e *Enhanced Directional Multi-Gigabit Access Point* (EDMG AP). O canal de 60 GHz possui uma largura de banda não licenciada de aproximadamente 14 GHz, dividida em canais de 2,16 GHz, 4,32 GHz, 6,48 GHz e 8,64 GHz. Originalmente, apenas 4 canais de 2,16 GHz estavam disponíveis para uso individual, mas por meio da agregação de canal, canais de até 8,64 GHz estão disponíveis para uso. Indo além, o padrão define, no mínimo, o agrupamento de canais (*channel bonding*) de dois canais de 2,16 GHz. A disposições dos canais na banda e os possíveis agrupamentos são conforme apresentado na Figura 2.3.

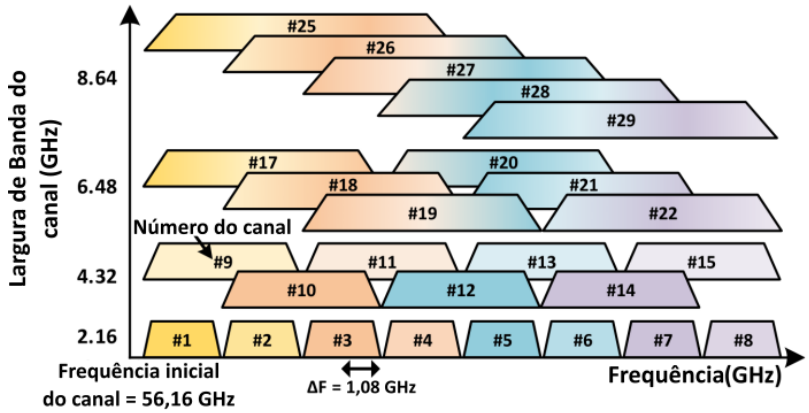

Figura 2.3: Configurações de canal EDMG do 802.11ay [Assasa et al. 2021].

2.2 Camada Física

Ao tratar da camada física, o padrão 802.11ay suporta três modos principais de operação: controle, *Single Carrier* (SC) e *Orthogonal Frequency Division Multiplexing* (OFDM). São componentes importantes durante a comunicação de dados e influenciam diretamente nos

parâmetros de qualidade como *throughput* e taxa de erro de pacote. O modo de controle é dedicado para a transmissão de quadros de gerenciamento, sendo que os mais importantes são os quadros *Beacons* e os quadros de treino *Beamforming Training* (BTF). O modo de controle também é designado para operar em condições com relação sinal-ruído (*Signal to Noise Ratio* - SNR) relativamente baixa (em comparação com os outros modos) e priorizar o treinamento de *beamforming*.

Os outros modos restantes são dedicados para a transmissão de dados, ambos definem um conjunto expandido de *Modulation and Coding Scheme* (MCS) que podem ser configurados livremente para atender uma determinada aplicação. Enquanto o modo SC representa uma boa opção para dispositivos com menos complexidade de *hardware* e limitados em potência, a opção OFDM se destaca pelas maiores taxas de transmissão de dados, porém é mais complexa de ser implementada. Os MCS básicos definidos para o 802.11ay são compostos pelas modulações BPSK, QPSK, 16QAM e 64QAM e o código de verificação *Low Density Parity Check* LDPC com as taxas de código de 1/2, 5/8, 3/4, 13/16 e 7/8 disponíveis [Ghasempour et al. 2017]. Nota-se que a vazão máxima alcançada pelo repositório para cada MCS depende do número de feixes $1 \leq N_{SS} \leq 4$ e do número de canais de 2,16 GHz utilizados $1 \leq N_{CB} \leq 4$. O número de feixes é determinado pelo número de *RF chains* configurados por antena.

A combinação dos principais fatores como MCS, utilização de *channel bonding*, modo de operação da camada física e a implementação de MIMO reflete no *throughput* e nas taxas de erro de pacote. Tal comportamento pode ser observado facilmente com a modulação escolhida, sua eficiência depende fortemente das condições de SNR do canal. Uma vez que o modo de operação SC é suficiente para atender a simulação desejada, isto é, um dispositivo consegue mandar e receber dados de outro por meio de comunicação sem fio, optou-se por utilizá-lo. O documento de padronização do IEEE [IEEE 2021a] traz uma tabela relacionando o *throughput* máximo possível para um determinado MCS e o modo de operação SC baseado no *channel bonding* e considerando apenas um fluxo e está representada na Tabela 2.1. A taxa varia conforme o número de *channel bonding* que pode chegar até 4 e também baseado em três tipos de *Guard Interval* (GI). O valor N_{CBPS} indica o número de bits codificados em símbolos (*Coded Bits per Symbol*).

Basicamente o intervalo de guarda indica um tempo de espera e está presente na família de padrões 802.11. Seu propósito é proteger transmissões distintas de eventuais interferências que podem ocorrer devido ao desvanecimento e os multipercursos das ondas eletromagnéticas. Considerando ambos os modos SC e OFDM a duração dos tipos de *guard interval* são definidos conforme a tabela Tabela 2.2. Por exemplo, ao considerar para o MCS 2 um *guard interval* do tipo normal, o cálculo é definido como 770 Mbps multiplicado pelo número de *channel bonding* (N_{CB}). Assim, é possível que um único fluxo alcance até 3080 Mbps.

Índice	Modulação	N_{CBPS}	Repetição	Taxa de código	Throughput(Mbps)		
					Normal GI	Short GI	Long GI
1	π/2-BPSK	1	2	1/2	N_{CB}×385.00	N_{CB}×412.50	N_{CB}×330.00
2	π/2-BPSK	1	1	1/2	N_{CB}×770.00	N_{CB}×825.00	N_{CB}×660.00
3	π/2-BPSK	1	1	5/8	N_{CB}×962.50	N_{CB}×1031.25	N_{CB}×825.00
4	π/2-BPSK	1	1	3/4	N_{CB}×1155.00	N_{CB}×1237.50	N_{CB}×990.00
5	π/2-BPSK	1	1	13/16	N_{CB}×1251.25	N_{CB}×1340.63	N_{CB}×1072.50
6	π/2-BPSK	1	1	7/8	N_{CB}×1347.50	N_{CB}×1443.75	N_{CB}×1155.00
7	π/2-QPSK	2	1	1/2	N_{CB}×1540.00	N_{CB}×1650.00	N_{CB}×1320.00
8	π/2-QPSK	2	1	5/8	N_{CB}×1925.00	N_{CB}×2062.50	N_{CB}×1650.00
9	π/2-QPSK	2	1	3/4	N_{CB}×2310.00	N_{CB}×2475.00	N_{CB}×1980.00
10	π/2-QPSK	2	1	13/16	N_{CB}×2502.50	N_{CB}×2681.25	N_{CB}×2145.00
11	π/2-QPSK	2	1	7/8	N_{CB}×2695.00	N_{CB}×2887.50	N_{CB}×2310.00
12	π/2-16-QAM	4	1	1/2	N_{CB}×3080.00	N_{CB}×3300.00	N_{CB}×2640.00
13	π/2-16-QAM	4	1	5/8	N_{CB}×3850.00	N_{CB}×4125.00	N_{CB}×3300.00
14	π/2-16-QAM	4	1	3/4	N_{CB}×4620.00	N_{CB}×4950.00	N_{CB}×3960.00
15	π/2-16-QAM	4	1	13/16	N_{CB}×5005.00	N_{CB}×5362.50	N_{CB}×4290.00
16	π/2-16-QAM	4	1	7/8	N_{CB}×5390.00	N_{CB}×5775.00	N_{CB}×4620.00
17	π/2-64-QAM	6	1	1/2	N_{CB}×4620.00	N_{CB}×4950.00	N_{CB}×3960.00
18	π/2-64-QAM	6	1	5/8	N_{CB}×5775.00	N_{CB}×6187.50	N_{CB}×4950.00
19	π/2-64-QAM	6	1	3/4	N_{CB}×6930.00	N_{CB}×7425.00	N_{CB}×5940.00
20	π/2-64-QAM	6	1	13/16	N_{CB}×7507.50	N_{CB}×8043.75	N_{CB}×6435.00
21	π/2-64-QAM	6	1	7/8	N_{CB}×8085.00	N_{CB}×8662.50	N_{CB}×6930.00

Tabela 2.1: Throughput EDMG-MCS em modo SC [IEEE 2021b].

Guard Interval		
Short GI	Normal GI	Long GI
18,18 ns	36,36 ns	72,72 ns

Tabela 2.2: Tempos de guarda [IEEE 2021b].

2.3 Camada MAC

2.3.1 Quadro Enhanced Directional Multi-Gigabit

Analisando o quadro EDMG percebe-se uma organização de cabeçalhos que definem importantes processos, como a manutenção da compatibilidade reversa com o IEEE 802.11ad, além de carregar informações de controle e de auxílio para implementação das técnicas de *channel bonding* e MIMO. Posto isto, o quadro 802.11ay também apresenta campos necessários para atender a arquitetura dos dispositivos EDMG.

O quadro 802.11ay pode ser dividido em duas partes, como mostrado na Figura 2.4. A primeira é compatível com o padrão 802.11ad e pode ser reconhecida prontamente por dispositivos DMG. A parte seguinte é composta por alguns campos reconhecíveis apenas por dispositivos EDMG e inclui informações de MCS, número de transmissões no espaço, emprego ou não de *channel bonding* e se há utilização de MIMO. Estas informações são divididas em dois campos de cabeçalho (*headers A e B*), campos EDMG-CEF (*Channel Estimate Field*) e EDMG-STF (*Short Training Field*) [IEEE 2021b].

O quadro é finalizado com o campo de dados contento o *payload* a ser transmitido e o *Training Field* referente ao *Beam Refinement Protocol* (BRP). Este protocolo é utilizado para refinar e completar o treinamento inicial das antenas que ocorre durante as primeiras fases da transmissão, mais detalhes sobre o treinamento *beamforming* serão explicados nas subseções seguintes. A Figura 2.4 ilustra a estrutura do quadro 802.11ay.

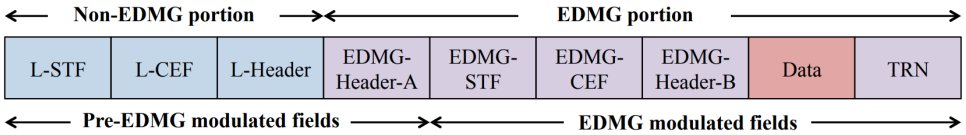

Figura 2.4: Quadro 802.11ay [Ghasempour et al. 2017].

2.3.2 Intervalo de *Beacon*

O 802.11ay organiza o acesso ao meio no tempo em *beacon intervals* (BIs). Cada BI é iniciado com a transmissão de um quadro *beacon* pelo AP, o qual serve para divulgar no meio a presença de uma rede Wi-Fi e para que as estações consigam se associar ao mesmo. Uma vez associado ao AP, o dispositivo pode começar a enviar e receber quadros de dados pela conexão estabelecida [James W. Kurose 2021].

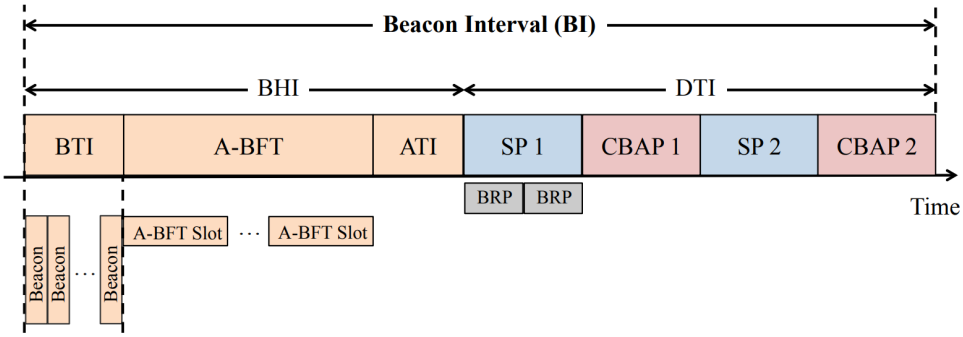

Figura 2.5: Estrutura de *Beacon Interval* do 802.11ay [Ghasempour et al. 2017].

A Figura 2.5 ilustra como está subdividido o BI. A divisão principal é feita em *Beacon Header Interval* (BHI) e *Data Transmission Interval* (DTI). No primeiro subintervalo há três períodos: o *Beacon Transmission Interval* (BTI), usado para a transmissão de quadros de *beacon*; o *Association Beamforming Trainig* (A-BFT), usado pelas DMG/EDMG STAs para treinar suas configurações de transmissão e recepção das antenas; e o *Announcement Transmission Interval* (ATI), usado para gerenciamento de troca de quadros entre o AP e as STAs que já realizaram o *beamforming training*.

O espaço no tempo reservado para o A-BFT contém até 40 divisões. As estações escolhem aleatoriamente uma dessas para transmitir seu quadro de *sector sweep* (SSW), que serve para auxiliar as estações envolvidas na comunicação a determinarem o melhor setor de transmissão e recepção de sinal. O número de divisões é também uma melhoria com relação

ao 802.11ad, que conta com apenas oito e, portanto, reduz as chances de colisão que ocorreriam caso estações distintas escolhessem o mesmo *slot* de tempo para transmitir seu quadro de varredura [Ghasempour et al. 2017].

O segundo intervalo, o DTI, é composto de um ou mais períodos de *Contention-Based Access Periods* (CBAPs) e agendados *Service Periods* (SPs), para reduzir colisões. No primeiro período o aceso ao canal é feito de modo distribuído, e no segundo por meio de agendamento [Ghasempour et al. 2017][Nitsche et al. 2014].

2.3.3 Acesso Múltiplo aos Canais

Uma vez que o controle de acesso ao meio em WLANs do IEEE 802.11 é feito através da prevenção de colisão, é exigida das estações a capacidade de realizar dois tipos de sensoriamento: o *Virtual Carrier Sensing* e o *Physical Carrier Sensing*. O primeiro é realizado obrigatoriamente apenas no canal de frequência em que a estação opera. A estação que deseja transmitir utilizará o valor do campo de duração presente em algum quandro capturado para criar um contador decrescente chamado *Network Allocation Vector*. Enquanto o valor do contador for maior que 0, a estação deve continuar esperando pela oportunidade de requisitar acesso ao canal.

O segundo tipo de sensoriamento consiste em avaliar se a quantidade de energia de rádio frequência detectada no meio indica a presença de outra comunicação acontecendo naquele instante de tempo, e deve ser feito também nos canais adjacentes. As estações economizam energia ao não terem que realizar essa medição quando o *Network Allocation Vector* está diferente de 0 [Tom Carpenter 2007].

2.4 Antena de Arranjo Faseado

A estrutura de arranjo de múltiplas antenas *Phased Antenna Array* (PAA) é uma excelente solução para os padrões de tecnologia *wireless* na faixa mmWave para transmissão de feixes direcionados. A ideia consiste em alimentar individualmente ou em grupos, os elementos de antena com as cadeias RF e controlar a direção dos feixes gerados explorando o deslocamento de fase das ondas. Posto isto, um arranjo PAA é capaz de transmitir feixes com maior EIRP e direcionados para uma determinada região ótima, evitando transmissões para direções indesejadas e melhorando o aproveitamento da energia consumida. Na Figura 2.6 está a representação do diagrama de radiação de um arranjo PAA, onde A representa o arranjo linear, θ o ângulo de apontamento desejado e determinado pelo módulo de controle C e φ a compensação de fase da excitação dos elementos do arranjo.

Apesar do padrão 802.11ad permitir várias PAA conectadas a uma única cadeia RF, apenas uma PAA pode ser utilizada por vez o que resulta em apenas uma transmissão de feixe. Porém, o padrão 802.11ay tem suporte para tecnologia MIMO, sendo assim, os arranjos

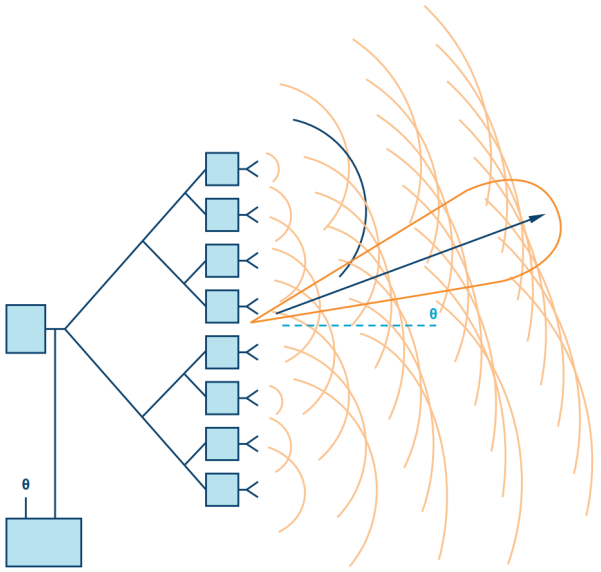

Figura 2.6: Antena de arranjo faseado [Benson 2019].

devem transmitir concorrentemente, possibilitando vários fluxos ao mesmo tempo. A geometria do arranjo e o padrão de radiação dos elementos de antena devem ser considerados como fatores importantes para garantir baixa interferência entre os múltiplos feixes, o que nem sempre é possível alcançar. Com relação à geometria, são várias as opções de escolha para a montagem, mantendo a estrutura dos vários elementos em uma organização uniforme. Atualmente, as configurações mais estudadas e implementadas são baseadas em três modelos:

- ULA (Uniform Linear Array)
- URA (Uniform Retangular Array)
- UCA (Uniform Circular Array)

A Figura 2.7 apresenta, nesta mesma ordem, exemplos de representações antenas de arranjo faseado gerados com MATLAB.

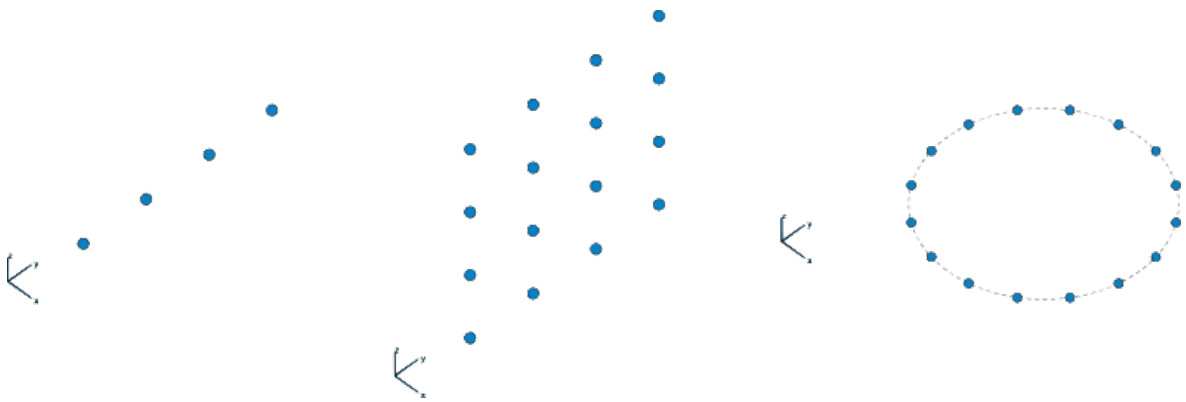

Figura 2.7: Exemplos de representações de antenas de arranjo faseado.

O arranjo ULA é o modelo mais básico, os elementos são distribuídos no eixo y do plano por exemplo. O modelo sofre limitações na capacidade de direcionamento dos feixes quando é desejado ter controle tanto do direcionamento vertical quanto do horizontal, fatores essenciais de uma PAA. No modelo UCA os elementos são distribuídos conforme um anel circular, de tal forma que os feixes podem ser igualmente distribuídos no plano azimute. Por último, o modelo URA combina alto poder de diretividade com alto aproveitamento na potência transmitida distribuindo os elementos de maneira bidimensional.

2.5 Formatação de feixes

Um dos principais mecanismos para superar o problema envolvendo a perda de percurso na banda de 60 GHz é estabelecer enlaces de comunicação de modo que a potência irradiada esteja concentrada em feixes com abertura angular estreita. Para tal abordagem, o *beamforming* é uma das abordagens para trabalhar com ondas na faixa mmWave. O procedimento é imprescindível e já estava presente no padrão 802.11ad. O processo se inicia durante a transmissão de quadros de varredura (SSW) e os quadros beacons nas fases BTI e A-BFT. Basicamente, emissor e receptor devem trocar informações contendo a melhor configuração de suas antenas transmissoras. Esta configuração determinada pela seleção, em cada um dos dois nós, do melhor apontamento (θ, φ) do respectivo feixe que resulta na melhor SNR para o enlace. Uma configuração específica de apontamento será denominada "setor"e, portanto, um enlace caracterizado pela escolha de dois identificadores. Após ambos dispositivos compartilharem informações de avaliação de setores, a fase inicial de treinamento é encerrada através de quadros de *feedback*, o receptor envia um pacote ACK através da antena escolhida até o emissor.

Para encerrar o treinamento, ainda falta parear a melhor antena transmissora com a melhor antena receptora. O treinamento das antenas receptoras é feito através de quadros de treinamento do *Beam Refinement Protocol*. Assim que as combinações são feitas, o melhor enlace dentre as combinações deve ser escolhido como o link de comunicação estabelecido entre os dispositivos. A Figura 2.8 indica as fases e subfases principais do treinamento de *beamforming*, a fase inicial é chamada de *Sector Level Sweep* (SLS) e pode ser dividida em duas subfases definidas como *Initiator Transmit Sector Sweep* (I-TXSS) e *Responder Transmit Sector Sweep* (R-TXSS) [Zhou et al. 2018], indicando a troca de quadros de treinamento referente às antenas transmissoras.

Por último, a fase BRP composta por outras duas subfases, *Multiple Sector ID Detection* (MID) e *Beam Combining* responsáveis pelo treinamento das antenas receptoras e montar as combinações possíveis entre os pares de transmissão e recepção por meio da troca de quadros BRP [Zhou et al. 2018]. Por ser compatível com a tecnologia *Single User* (SU-MIMO) e *Multiple User* MU-MIMO, o treinamento de *beamforming* assume um papel bastante importante de suporte e permite ao padrão 802.11ay alcançar transmissões robustas através de

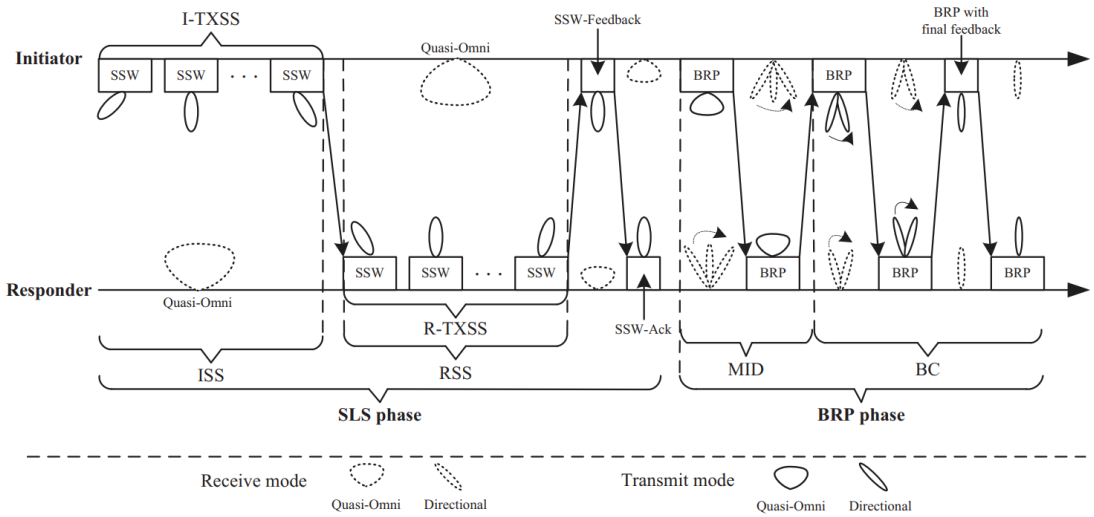

Figura 2.8: Treinamento *beamforming* [Zhou et al. 2018].

fluxos de dados múltiplos e direcionados simultaneamente.

Para estabelecer a comunicação SU-MIMO, o treinamento segue conforme as especificações do padrão, porém para avaliar melhor o processo como um todo são estabelecidas duas fases principais definidas como SU-MIMO SISO e SU-MIMO MIMO [Zhou et al. 2018]. Durante a fase inicial SU-MIMO SISO, os mesmos passos definidos para o treinamento de *beamforming* são executados através do processo SLS, portanto as subfases I-TXSS e R-TXSS estão envolvidas. A troca de pacotes BRP encerra a fase SU-MIMO SISO, de tal modo que o melhor setor com feixe individual entre antena transmissora e receptora pode ser estabelecido. A Figura 2.9 ilustra o processo da fase SISO.

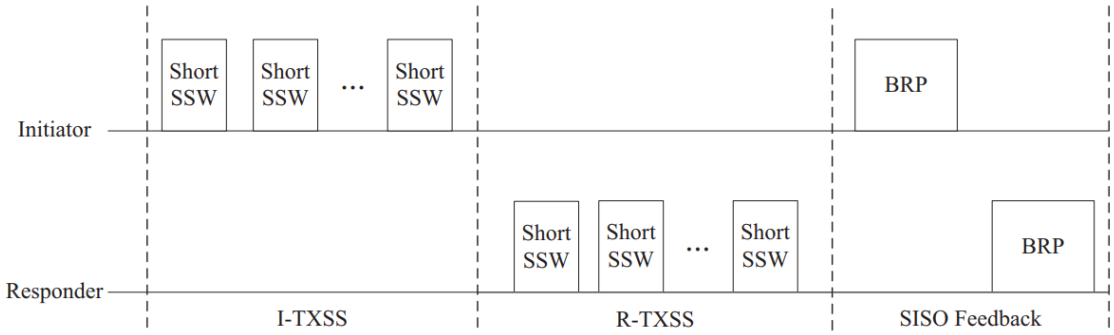

Figura 2.9: SU-MIMO SISO [Zhou et al. 2018].

Em seguida é iniciada a fase SU-MIMO MIMO cujo grau de complexidade é maior, pois dependendo da configuração das antenas a interferência entre transmissões simultâneas deve ser considerada no momento de escolha dos setores, indicando que trabalhar com *Signal to Interference Plus Noise Ratio* (SINR) seja mais efetivo. Recentes trabalhos discutem algoritmos capazes de definir as melhores combinações de transmissões simultâneas, considerando o ambiente de simulação ns-3 o artigo [Assasa et al. 2021] traz uma abordagem interessante

de utilizar os parâmetros da fase SU-MIMO SISO como entrada para um algorítimo de seleção das melhores K combinações possíveis resultando em um conjunto de medições de SINR.

Considerando o trabalho, a simulação no ns-3 utiliza as implementações das classes e objetos criados em [wigig-tools 2021c] para habilitar o SU-MIMO. A fase SU-MIMO SISO descrita anteriormente fornece a avaliação do melhor setor para uma determinada configuração de posição dos nós na rede, esta informação será a base para o método de localização. Através da checagem do setor escolhido é possível rastrear os ângulos de chegada.

Existem duas abordagens para implementar o *beamforming* em um arranjo PAA, digital e analógico. Recentemente, o interesse em implementar *beamforming* digital cresce devido a manipulação mais flexível dos feixes. Estudos recentes buscam implementar a formatação de feixes por meio de *Field Programmable Gate Arrays* (FPGAs), diferentes configurações e novos algoritmos podem ser disponibilizados através de atualizações de *firmware* [Benson 2019]. Considerando o caso, o 802.11ay trabalha com um protocolo híbrido analógico e digital de *beamforming* para compensar perdas de performance e garantir altos ganhos nas taxas quando utiliza o MIMO [Zhou et al. 2018].

Capítulo 3

Proposta de Método de Localização

3.1 Introdução

3.1.1 Técnica Proposta

O trabalho apresenta uma proposta de localização envolvendo a discretização dos ângulos e a implementação de *timestamp* para marcação de tempo dos pacotes. A partir dos ambientes simulados as principais informações necessárias são os ângulos de chegada e o cálculo do tempo de propagação a partir do *timestamp* obtido. A distância entre os dispositivos é definida pelo vetor r que pode ser obtido multiplicando a velocidade da luz pelo tempo de propagação t:

$$r = 3 \cdot 10^8 \cdot t \ \ (m). \tag{3.1}$$

Quanto ao sincronismo, o ns-3 garante sincronia entre os dispositivos. Os detalhes de como obter as medições necessárias a partir dos arquivos de simulação do ns-3 e MATLAB para a localização serão abordados no Capítulo 4 com mais clareza, além dos cenários construídos e parâmetros bases da simulação, além das instruções e alterações necessárias para que o repositório consiga atender a demanda proposta.

3.1.2 Sistema esférico de coordenadas

O sistema esférico de coordenadas pode ser definido pelas medições angulares θ e φ e o vetor r projetado pela origem até qualquer direção no plano, sendo assim, a partir dos valores (r, θ, φ). A Figura 3.1 ilustra a organização comumente utilizada, quando os ângulos crescem positivamente em uma determinada direção no plano horizontal e vertical. Essas marcações são representadas pelas variáveis θ_1 e φ_1, em que o ângulo θ_1 refere-se ao ângulo de azimute formado entre qualquer ponto e o centro de coordenadas na horizontal. O ângulo de elevação

é definido por φ_1.

Para uniformizar as fórmulas e resultados obtidos, será utilizado o sistema de coordenadas esféricas que utiliza sinal negativo e positivo para indicar o sentido de giro do ângulo. Assim, no plano azimute (X, Y) é medido o ângulo θ com relação ao eixo X. A variação de θ no sentido horário é representada por $-\theta$ e no sentido anti-horário por $+\theta$. De forma similar, para o plano de elevação (Y, Z) o φ indica as medições de ângulo com relação ao eixo Y. A variação de φ no sentido horário é representada por $-\varphi$ e no sentido anti-horário por $+\varphi$. Nessa convenção, é preciso fazer ajustes para que as fórmulas que envolvem senoides continuem válidas. Basta considerar que $\theta_1 = 2\pi + \theta$ e $\varphi_1 = \frac{\pi}{2} - \varphi$.

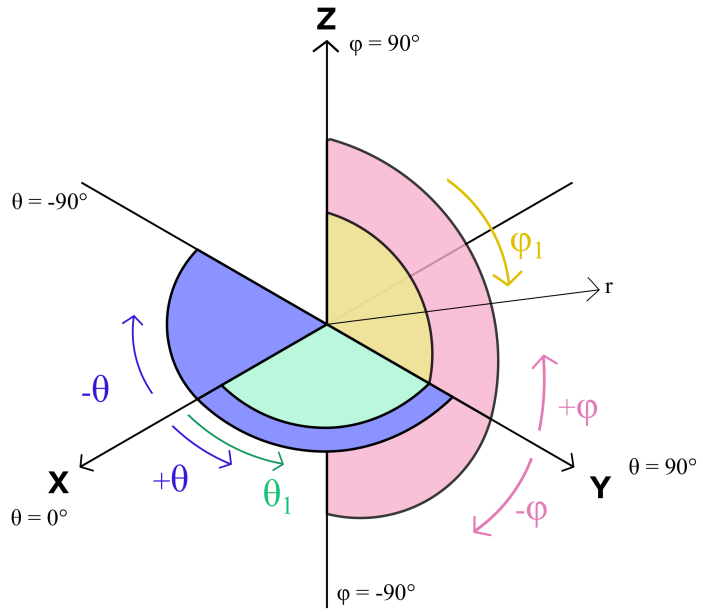

Figura 3.1: Sistema esférico de coordenadas.

As medições dos ângulos de referência com melhor aproveitamento do canal chegando no receptor e o cálculo do tempo de propagação dos pacotes pode ser uma solução viável pois, após a definição do centro de coordenadas, basta fazer uso das relações abaixo para identificar um ponto no plano:

$$x = r \cdot cos(\theta_1) \cdot sin(\varphi_1) \tag{3.2a}$$

$$y = r \cdot sin(\theta_1) \cdot sin(\varphi_1) \tag{3.2b}$$

$$z = r \cdot cos(\varphi_1). \tag{3.2c}$$

3.1.3 Fluxograma do Método

O método irá seguir o seguinte caminho: gerar os cenários em MATLAB, com reflexões ou não, contendo as posições dos nós e encaminhar os resultados para a simulação no ns-3. O resultado da simulação no ns-3 indica o melhor setor atribuído dependendo da posição dos nós no cenário e também os registros de *timestamp* obtidos no receptor. Em seguida. é feita a comparação entre os cenários com e sem reflexões. Caso o mesmo setor seja obtido, pode-se concluir, devido às condições da simulação, poucos nós na rede, ausência de obstáculos, que o tempo de propagação equivale ao tempo de propagação. Com as informações obtidas é feita a estimativa de localização com o *timestamp* e os ângulos de chegada. A Figura 3.2 ilustra o fluxograma do método de simulação que será realizado.

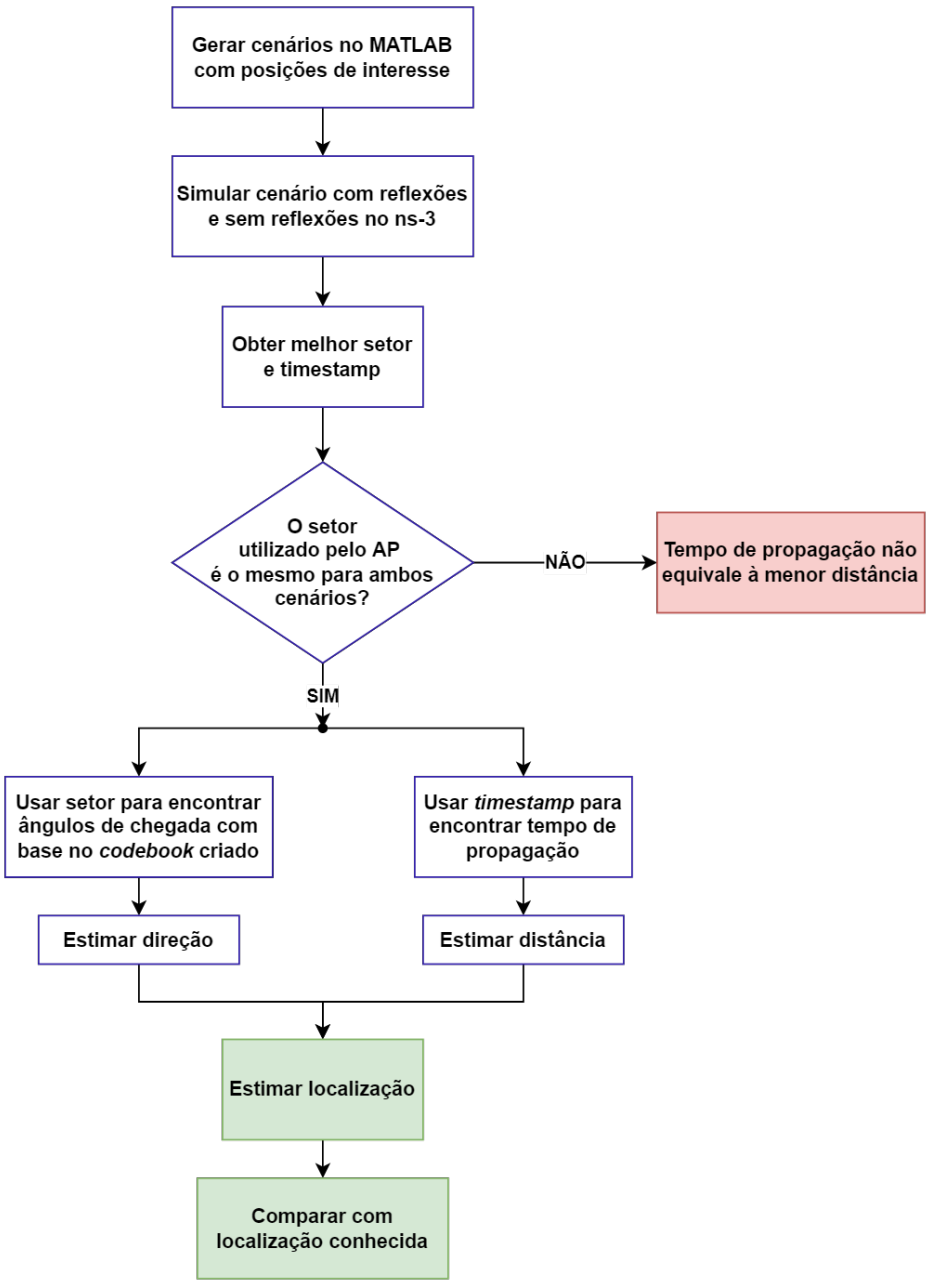

Figura 3.2: Fluxograma do método.

3.2 Repositórios do WiGig Tools

O principal material desse trabalho é o conjunto de repositórios de código aberto chamado WiGig Tools, disponível no GitHub [Hany Assasa 2019]. O conjunto possui ferramentas para simular redes do IEEE 802.11ay utilizando o simulador ns-3. Também há bastantes funcionalidades voltadas para o IEEE 802.11ad, já que o código base foi desenvolvido para esta versão do protocolo.

A Figura 3.3 ilustra como estão relacionados os repositórios do WiGig Tools, com os detalhes das principais funções de cada um.

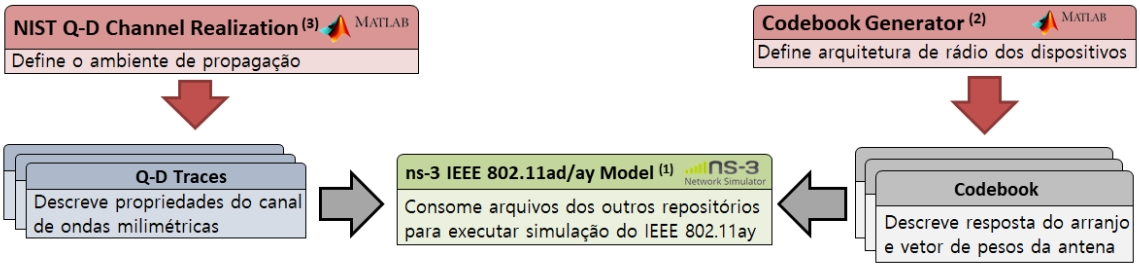

Figura 3.3: Representação de relação entre os repositórios do WiGig Tools (Adaptado de [wigig-tools 2021c]).

Em apresentação breve, o "NIST Q-D Channel Realization Software" contém códigos em MATLAB para, por meio do tracejado de raios (*ray-tracing*), fazer a modelagem de canal *mmWave*. Do outro lado, o "Codebook Generator" é responsável por gerar o arquivo que define os padrões de *beam* da *Phased Antenna Array* (PAA). Por fim, o "ns-3 IEEE 802.11ad/ay model" é o repositório de código baseado em ns-3 que consome a saída destes outros repositórios para compor a implementação das camadas física e MAC do 802.11ay, além da implementação do modelo do canal *mmWave*.

O resultado disso tudo é a possibilidade de simular no ns-3 a comunicação de dispositivos que operam de acordo com o padrão 802.11ay, com atendimento das especificações validado no documento [Assasa et al. 2021], que foi apresentado no *Workshop on ns-3* (WNS3) de 2021, evento oficial dos desenvolvedores do ns-3.

Os detalhes específicos dos repositórios serão abordados na Seção 3.4.

3.3 Preparação de ferramentas e repositórios

O programa MATLAB é necessário para execução dos códigos presentes no repositório "Q-D Realization". A versão utilizada neste trabalho foi a R2022a, versão *trial* de 30 dias. Além da instalação básica do programa, é necessário instalar os seguintes *Add-Ons*:

- Parallel Computing Toolbox

- Signal Processing Toobox
- Phased Array Sytem Toolbox
- DSP System Toolbox
- Statistics and Machine Learning Toolbox
- Mapping Toolbox

Os demais detalhes de preparação de ferramentas e repositórios são abordados na Seção I.1 do Apêndice.

3.4 Detalhes sobre os repositórios

3.4.1 NIST Q-D Channel Realization Software

A proposta do Q-D Channel Realization é ser um modelador de canal quase-determinístico voltado para o ns-3, compatível com aplicações responsáveis por fazer tracejado de raios. Os cenários modelados podem ser para quando os dispositivos estão em LOS ou mesmo em NLOS, e também há possibilidade de seleção de frequência do canal. No primeiro momento, o modelador surge como uma alternativa para melhorar a fidelidade do modelo do IEEE 802.11ad disponível para simulação no ns-3, mas já com expectativa de expansão para uso com o IEEE 802.11ay [Assasa et al. 2019].

A Figura 3.4 apresenta a visão geral do fluxo de execução do Q-D Realization Software, e será melhor explorada com o auxílio das figuras seguintes. A entrada do programa consiste de arquivos com dados e arquivos com configurações de parâmetros.

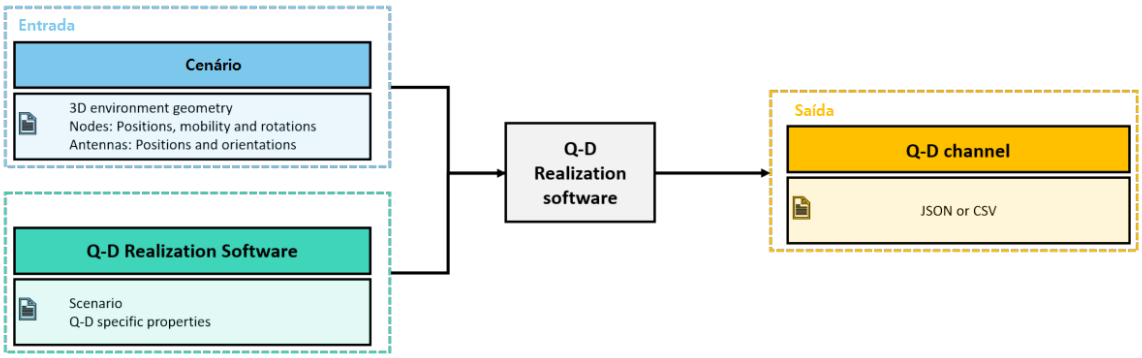

Figura 3.4: Fluxo de execução do Q-D Realization Software[wigig-tools 2021b].

Os valores de medida de dimensões do ambiente estarão dentro do arquivo `enviromentFileName.xml` apresentado na Figura 3.5. Dentro dos arquivos `nodePositionX.dat` e `nodeRotationX.dat` estarão os dados que descrevem a posição e rotação de um nó `X` durante o curso da simulação. Ainda, o `NodePaaX.dat` terá os valores de posição das antenas do nó, que são gerados tomando uma distância fixa do centro

do dispositivo em cada posição em que o mesmo se encontra. Porém, embora exista a possibilidade de configurar movimentação nesta etapa, essa funcionalidade não será explorada por falta de compatibilidade total com o ns-3. Nas simulações será considerada apenas a posição inicial definida para cada cenário.

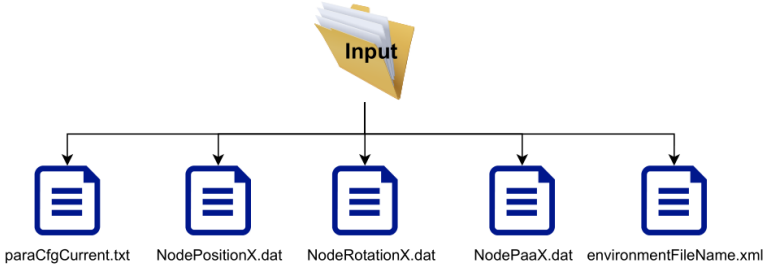

Figura 3.5: Arquivos de entrada do Q-D Realization Software [wigig-tools 2021b].

O arquivo com configuração de parâmetros é o `paraCfgCurrent.txt`, também apresentado na Figura 3.5. Nele estão definidos os caminhos para os arquivos que definem o ambiente e as características físicas dos materiais, o tempo total de simulação, o número de divisões do tempo, a frequência da portadora, o número da ordem de reflexões, além de outras variáveis utilizadas para fazer a modelagem do canal. Exemplos de uso e modificações que podem ser feitas neste arquivo podem ser encontrados na documentação do repositório, que está na pasta `docs` do [wigig-tools 2021b].

A parte central da Figura 3.4 é detalhada na Figura 3.6. O código de execução `main.m` buscará, por meio dos componentes `parameterCfg.m` e `nodeProfileCfg.m`, os parâmetros de simulação e de nó, que são os dados dos arquivos de entrada explorados na Figura 3.5.

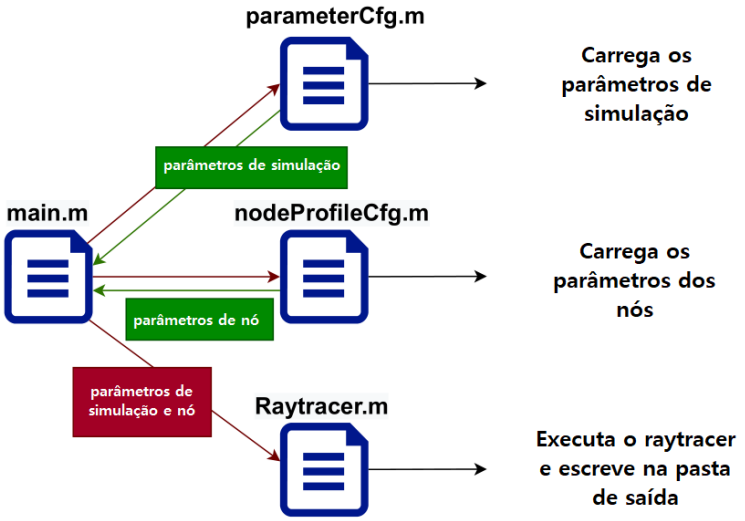

Figura 3.6: Estrutura do código do Q-D Realization Software.

Uma vez com os dados carregados, o componente `Raytracer.m` é acionado para executar o tracejado de raios e escrever os arquivos de saída do programa. Após a execução,

na pasta de saída, os resultados são apresentados no formato compatível com ns-3 e no formato para usar no Visualizer, que é uma aplicação, disponibilizada no mesmo repositório, desenvolvida para prover visualização do cenário, raios e representação espacial do ângulos de partida e chegada.

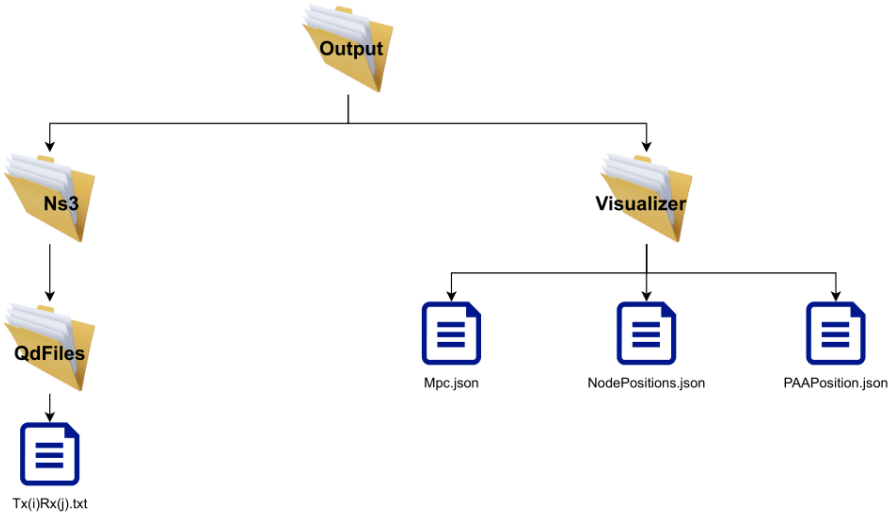

Figura 3.7: Arquivos de saída do Q-D Realization Software [wigig-tools 2021b].

Conforme apresentado na Figura 3.7, os arquivos de saída compatíveis com o ns-3 têm nomenclatura `Tx(i)Rx(j).txt`, o que indica que este arquivo trata sobre a comunicação do dispositivo `i` transmitindo para o dispositivo `j`. A sequência de informações escritas se inicia com o número de raios e o restante, para cada raio, é como segue:

- Atraso em segundos
- Perda de caminho em dB
- Desvio de fase em radianos
- Ângulo de partida em graus (elevação)
- Ângulo de partida em graus (azimute)
- Ângulo de chegada em graus (elevação)
- Ângulo de chegada em graus (azimute)

Esse bloco de 8 linhas se repete para cada divisão de tempo da simulação e para cada permutação de combinação de antenas transmissoras e receptoras. Por exemplo, o conteúdo de saída para o caso de dois dispositivos com duas antenas cada, definindo trinta divisões no tempo, terá $2 \times 2 \times 30 \times 8 = 960$ linhas. Na Listagem 1 é apresentado o arquivo que descreve essa estrutura para o terceiro cenário que será abordado na parte de simulações. É válido notar que cada linha de dados apresenta apenas um valor por se tratar da versão sem reflexão e, então, apenas uma possibilidade de caminho é modelada. Nos casos com reflexão,

os dados de cada feixe aparecem lado a lado. Aparecem 4 blocos nesta listagem porque a modelagem é de 2 antenas para 2 antenas, logo, 4 permutações.

Alternativamente, pode-se optar por obter esses dados todos em um único arquivo no formato JSON, que contém as mesmas informações, porém organizado de forma otimizada para representar as permutações de antena e sentido da comunicação no mesmo arquivo. O número de raios gerados para cada posição dependerá de um parâmetro denominado "ordem de reflexão"e das características do material que compõe as superfícies refletoras do ambiente. Sobre ordem de reflexão, o tracejado de raios considera reflexão de ordem 1 aquelas em que as componentes especulares geradas podem refletir até uma vez em alguma superfície antes de atingir o dispositivo alvo. Quando considerado reflexões de ordem dois, os raios especulares refletem em até duas superfícies antes de atingir o alvo, e assim sucessivamente.

Para a pasta de saída voltada para usar no Visualizer, as informações estão organizadas de forma diferente. No arquivo `NodePositions.json` estão descritas as posições e rotações de cada dispositivo. No arquivo `PAAPositions.json` estão descritas a orientação e as posições das antenas.

Ainda, o `Mpc.json` traz o restante das informações necessárias para deixar a representação do cenário semelhantemente ao apresentado na Figura 3.8. A essência dos dados deste arquivo é semelhante ao detalhado para a saída que atende o ns-3, porém com a adaptação de nomenclatura: nesse contexto, os raios são chamados de *Multipath Components* (MPCs), ou componentes de multicaminhos, em tradução livre. As componentes de multicaminhos que podem aparecer na saída são as componentes especulares, o caminho direto, e as componentes difusas de reflexão. Essas últimas só surgem quando consideradas as propriedades dos materiais no âmbito das imperfeições das superfícies refletoras.

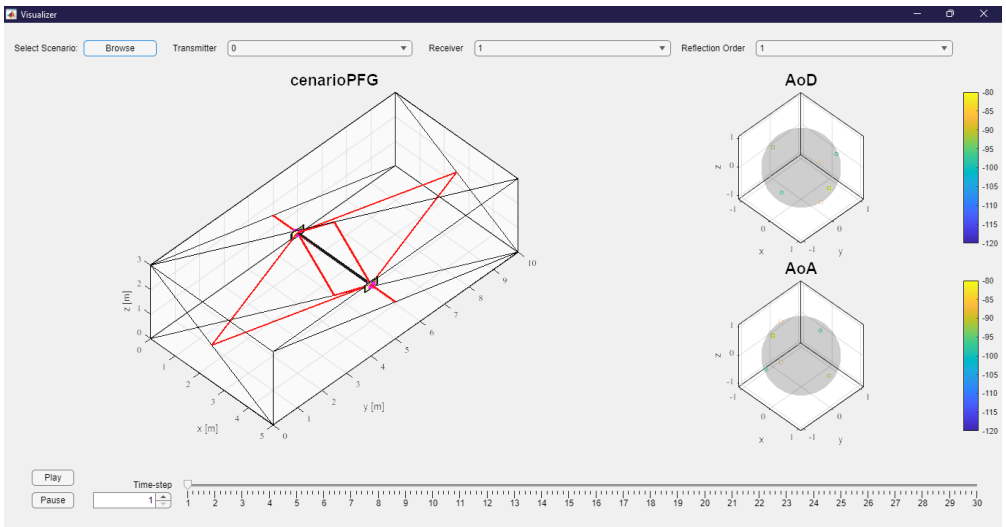

Figura 3.8: Componente Visualizer do Q-D Realization Software.

A Figura 3.8 ilustra a interface do Visualizer ao abrir os arquivos gerados para um cenário chamado “cenarioPFG”. No centro esquerdo da figura aparece o esboço geométrico da cena. Trata-se de uma sala de $5m \times 10m \times 3m$. Na sala, há dois dispositivos, dos quais saem os

raios vermelhos, que são as componentes especulares, e um raio preto de maior espessura, que é o caminho direto.

No centro direito há representações dos ângulos de partida e de chegada. A linha do tempo na parte inferior serve para atualizar a representação para um instante específico, o que é útil para observar mudanças provocadas por movimentação dos dispositivos, por exemplo.

3.4.2 Codebook Generator

Este repositório está aberto para distribuir o código da aplicação em MATLAB criada para gerar *codebooks*. Um elemento de um *codebook* corresponde a uma configuração específica de apontamento (θ, φ) para o arranjo. Em essência, um elemento de um *codebook* é um vetor com a informação de atraso de fase que deve ser aplicada a cada elemento do arranjo. Segundo a proposta, os usuários podem gerar múltiplas instancias de *codebook* para utilizar em dispositivos de rede. O principal conteúdo contido no arquivo de saída da aplicação apresenta a resposta em fase de cada elemento do arranjo, quando uma onda incidente em uma determinada o direção atinge. O mesmo arquivo lista quais são os elementos do *codebook* a serem utilizados. Entretanto, o código gerador destas informações não está disponível, o que trouxe a necessidade de geração de outros codebooks por intermédio do software Matlab [Assasa et al. 2019][wigig-tools 2021a].

3.4.3 ns-3 IEEE 802.11ad/ay model

Este repositório é o que concentra todos os resultados dos materiais citados neste capítulo para executar simulações em um simulador baseado na versão 3.31 do ns-3. No simulador, foram disponibilizados recursos capazes de modelar redes sem fio baseadas nos padrões IEEE 802.11ad/ay. Estão dentre os principais destaques de capacidades a implementação das camadas MAC e PHY, técnica de *beamforming* e tecnologia MIMO. A maioria das simulações disponíveis dentro do repositório visa testar o desempenho com relação ao *throughput* do padrão 802.11ay e a avaliação do mesmo com base, em alguns casos, nos parâmetros dentro do próprio ns-3 que são mais limitadas visto que a implementação do canal de 60GHz no simulador não está pronto ainda.

Já outros códigos disponíveis possuem suporte para consumir os dados gerados pelo modelador de canal Q-D Channel Realization, permitindo o programa ler o bloco de informação contendo informações a respeito da propagação de energia do canal simulado no MATLAB. Este processo é realizado por meio das classes `QdPropagationEngine` e `MultiModelSpectrumChannel`, componentes principais para a montagem do canal mmWave dentro do ns-3.

Códigos baseados em ns-3 oferecem dois perfis de montagem dos arquivos de execução:

o modo *debug* e o modo *optimized*. Quando em *debug*, as saídas dos códigos oferecem a possibilidade de acompanhar detalhes como nome das funções que são executadas, visualização de parâmetros e *logs*, e informações extras sobre erros. Entretanto, execuções neste modo são lentas, o que é ruim para simulações longas. No modo *optimized*, o código é montado de forma otimizada, como sugere o nome. Abre-se mão de alguns *logs* do compilador, porém as saídas de dados da simulação são mantidas e o ganho em tempo de execução é expressivo.

Visando adaptar parte do código para atender a proposta do trabalho foi selecionado um dos códigos disponíveis para adequação a novos cenários de simulação. O código `evaluate_11ay_su_mimo` recebe como dados de entrada as informações presentes no arquivo referente ao cenário escolhido, que por sua vez, contém a posição dos nós, esboço do ambiente *indoor* ou *outdoor* e suas dimensões, posições das antenas, e outros dados.

Capítulo 4

Configuração do Ambiente de Simulações

4.1 Adaptação do Ambiente de Simulação

Além da implementação do *timestamp*, outras alternativas foram testadas com o intuito de se obter o tempo de propagação dos pacotes. A análise inicial envolveu as aplicações disponíveis no código, OnOff e Bulk, esperava-se que o conjunto de funções que definem ambas as classes auxiliasse no processo de obter o tempo de propagação. Outra abordagem testada foi implementar uma aplicação mais simples de ping e fazer uso dos tempos de RTT para observar o tempo dos pacotes. Para abordar a implementação do *timestamp* foi necessário criar uma aplicação nova, uma vez que, o módulo `seqTsSizeHeader` está presente apenas na versão 3.33, a versão mais recente disponível, enquanto que o repositório Wigig-Module foi montado usando a versão 3.31 do ns-3. O módulo é fundamental para implementação do *timestamp* nos pacotes e obtenção do mesmo na saída do código para avaliação.

O código base `evaluate_11ay_su_mimo` foi selecionado como referência pois configurações básicas, como receber os arquivos referente à simulação no MATLAB e implementação do *beamforming* durante o processo SU-MIMO, já se encontravam prontas. Portanto, o foco de ação foi direcionado em duas questões, obter informações sobre tempo de transmissão e recepção dos pacotes usando configurações disponíveis nas classes de aplicação do ns-3. Inicialmente a análise de logs foi essencial para o entendimento do processo de execução das várias funções do código e quais seriam úteis posteriormente para gerar a saída com os dados necessários. É importante entender e gerar novos arquivos de codebook, garantido assim o controle sobre o mesmo. Os arquivos de codebook contêm os dados principais de configuração dos arranjos das antenas PAA, isto é, como são organizados os vetores de peso que permitem que a antena possa ser direcionada em uma determinada direção com o deslocamento de fase.

4.1.1 Visualização de logs

No modo *debug* do ns-3, o recurso que ajuda a ter visão minuciosa dos passos da simulação é o uso de *logs*. Através destes, é possível acompanhar as funções chamadas, os parâmetros passados, mensagens de erro e também mensagens informativas que podem ser adicionadas aos códigos de execução. Para tornar isso possível, os módulos da simulação devem ter a definição de componente que segue o padrão `NS_LOG_COMPONENT_DEFINE ("NomeDoModulo");`. Desta forma, inserir o comando `LogComponentEnable ("NomeDoModulo", LOG_LEVEL_ALL);` no código principal fará com que a saída apresente todos os *logs* possíveis do módulo `NomeDoModulo`. Detalhes adicionais e as variações deste comando podem ser encontrados no manual do ns-3, disponível em [ns-3 2021a].

Dado que o planejamento previa modificações nos códigos do repositório para fazer simulação de localização, foi necessário entender tudo o que já estava pronto e como tirar proveito disso. Além de contribuir para o entendimento de etapas não abordadas com detalhe em documentações, a visualização de *logs* permitiu entender o fluxo dos dados de entrada vindos dos repositórios MATLAB, nomeadamente o QD-Trace e o arquivo de Codebook.

4.2 Configuração do Posicionamento de Nós

A configuração de posicionamento dos nós é feita através de um código em MATLAB que gera como saída um arquivo `.dat` para recepção do código principal `main.m`. O posicionamento pode ser fixo, bastando apenas definir as posições nos eixos x, y e z, mas o programa também pode ler movimentação dos nós. O arquivo gerado pode descrever a movimentação do nó no espaço, para tal, o arquivo de configuração `generateNodeposition.m` deve contemplar todas as posições organizadas como uma matriz, na qual cada posição é definida como um passo, sendo assim, o tempo de cada passo é configurado livremente no arquivo `paraCfgCurrent`.

Para verificar o funcionamento do algorítimo de seleção de melhor setor, é interessante modificar as posições iniciais dos dispositivos e observar a alteração no direcionamento dos feixes para melhor desempenho. Esta informação também é essencial para a simulação de localização do nó na rede, as informações de ângulos de chegada definida no setor escolhido são vitais para a simulação. A geometria e dimensões serão mantidas em todos os cenários porque é o ambiente disponível mais simples de simulação *indoor*. Na Figura 4.1 é apresentado o segundo cenário montado, que agora conta com uma rotação de +90º do posicionamento da STA com relação ao AP no eixo horizontal.

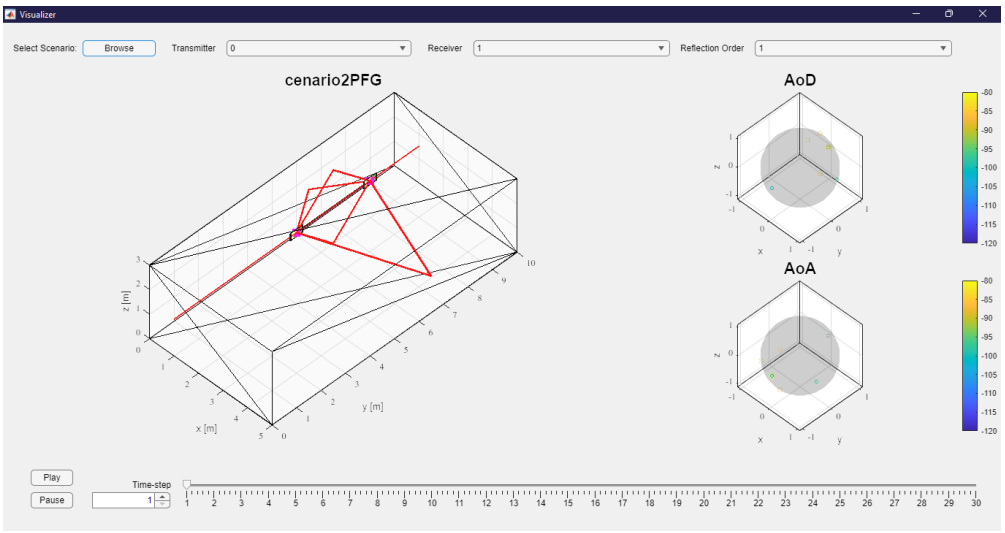

Figura 4.1: Posicionamento da STA +90º de Azimute com relação ao posicionamento da Figura 3.8.

4.3 Diagrama de Radiação e *Codebook*

Conforme pode ser encontrado em [Assasa et al. 2019], há a implementação de uma classe chamada Codebook que agrega as funcionalidades relacionadas ao treinamento de *beamforming* no ns-3, o que compreende as etapas de iniciar o treinamento, rastrear setores e antenas ativos, e fazer a comutação de setores. É definido que o arquivo de *codebook* é desenvolvido para um número de *RF chains*, onde cada *RF chain* pode ser conectada à uma ou mais antenas de arranjo faseado. Descrevendo de forma geral, o arquivo de *codebook* traz a descrição do padrão de radiação do arranjo, que é resultante do produto do padrão de radiação de um elemento da antena pelo fator de arranjo, quando todos elementos de antena forem idênticos. Seja Y o padrão de radiação de um arranjo de antena faseado, então:

$$Y(k, \theta, \varphi) = R(\theta, \varphi) W^T S(k, \theta, \varphi) \tag{4.1}$$

onde $k = 2\pi/\lambda$, $R(\theta, \varphi)$ é o padrão de radiação de um único elemento, W^T é o vetor de pesos transposto e $S(k, \theta, \varphi)$ é a representação do atraso de fase entre os elementos da antena para cada onda plana eletromagnética que incide sobre o arranjo, chamado *steering vector*. O *steering vector* depende da geometria do arranjo, do posicionamento dos elementos e a frequência da portadora, parâmetros que não serão alterados.

Se por um lado ter o Visualizer permitiu que alterações fossem feitas no posicionamento dos nós, entender e ser capaz de manipular as antenas não foi tão direto assim. A falta dos códigos do repositório Codebook Generator tornaram difícil ter certeza sobre os dados de resposta do arranjo e os vetores de peso, por exemplo. Superar esta etapa de preparo funcionou com a aplicação do MATLAB chamada Phased Array System Toolbox e com a construção de novos vetores de pesos no arquivo de *codebook* utilizado na simulação do ns-3.

A Phased Array System Toolbox possui uma interface executável através do comando `sensorArrayAnalyzer` que permite modelagem de arranjos e visualização das repostas em 2D e em 3D. Foi possível fazer a modelagem compatível com os dados que já estavam em uso nas simulações porque os parâmetros obrigatórios já eram conhecidos. É preciso do tamanho do arranjo, que é 2x8; espaçamento entre os elementos, $\lambda/2$; a velocidade de propagação, $3 \cdot 10^8\ m/s$; e também a frequência do sinal, 60 GHz. No fim, é possível exportar um *script* MATLAB que faz as representações obtidas, semelhante ao início do código apresentado na Listagem 2. No código gerado, algumas modificações foram feitas para incluir a diretividade da antena de arranjo faseado. Nas linhas 6 e 7, foram determinados os intervalos de varredura dos ângulos θ e φ. Posteriormente, são computados os vetores de pesos (*weights*) que atendem o direcionamento de *beam* determinado pelos ângulos de azimute e elevação.

Adotando a nomenclatura que chama de "setor" a configuração do arranjo que faz apontamento do *beam* para determinados ângulos θ e φ, o mapeamento dos setores feito na Listagem 2 é conforme apresentado na Figura 4.2.

		Azimute (θ)											
		-90°	-75°	-60°	-45°	-30°	-15°	0°	15°	30°	45°	60°	75°
Elevação (ϕ)	-30°	1	2	3	4	5	6	7	8	9	10	11	12
	-15°	13	14	15	16	17	18	19	20	21	22	23	24
	0°	25	26	27	28	29	30	31	32	33	34	35	36
	15°	37	38	39	40	41	42	43	44	45	46	47	48
	30°	49	50	51	52	53	54	55	56	57	58	59	60

Figura 4.2: Relação de setores com os ângulos θ e φ para o *codebook* de 60 setores.

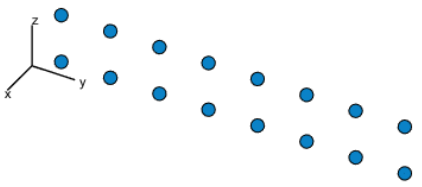

Figura 4.3: Orientação do arranjo 2x8.

Como pode ser observado, quando o *beam* tem direção perpendicular ao arranjo, isto é, $(\theta,\ \varphi) = (0°, 0°)$, o setor que representa o comportamento do padrão de radiação é o 31. As relações ficam melhor ilustradas com as figuras seguintes. Na Figura 4.3 apresenta-se a orientação do arranjo com relação aos eixos (x, y, z) nas considerações feitas neste trabalho. Já na Figura 4.4 está a representação da diretividade para o setor 1, que tem uma alta concentração no *beam* direcionado para $(\theta,\ \varphi) = (-90°, -30°)$.

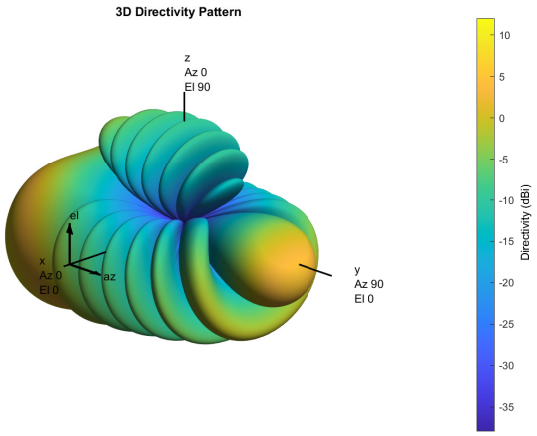

Figura 4.4: Padrão de radiação para o setor 1 do *codebook* de 60 setores.

Ainda sobre a Listagem 2, a parte central do código é responsável por gerar os diagramas de cada setor, gerar a representação dos cortes de azimute no plano horizontal, e salvar tudo isso como imagens. O final do código realiza a conversão dos pesos da forma real e imaginário para a forma polar, que é o formato esperado pelo ns-3.

Como a estrutura de dados que representa o padrão de diretividade de um determinado setor é no formato de matriz, uma comparação de matrizes foi feita para obter os dados que formam a Figura 4.5. As matrizes são de colunas indicando o ângulo horizontal (θ) e linhas indicando o ângulo vertical (φ), e o valor de cada célula indica um valor de diretividade para os ângulos correspondentes. Dado isso, montou-se uma matriz na qual para cada par de ângulos tem-se o mapeamento de qual setor possui o maior valor de diretividade, medida em dBi, e a representação visual desta é o que aparece na figura, com eixos adaptados para os intervalos de observação de interesse.

O código original conta com alguns *codebooks* prontos. O *codebook* que estava em uso até então mapeava 27 setores. Antes de conseguir mapear 60 setores, também houve a tentativa de utilizar 90 setores, partindo do pressuposto que um número grande de *beams* bem distribuídos no espaço aumentaria a acurácia da estimativa de direção do feixe. Entretanto, foi encontrado que há um limite previsto de 64 setores para cada antena, determinado pelo padrão [Assasa et al. 2019]. Por assim ser, o procedimento seguinte foi adicionar os vetores de pesos obtidos para mapear 60 setores substituindo no arquivo de 27 setores os parâmetros correspondentes. Assim, já que os setores mapeiam ângulos, uma das metas neste ponto é obter esse parâmetro para estimar os ângulos do feixe recebido.

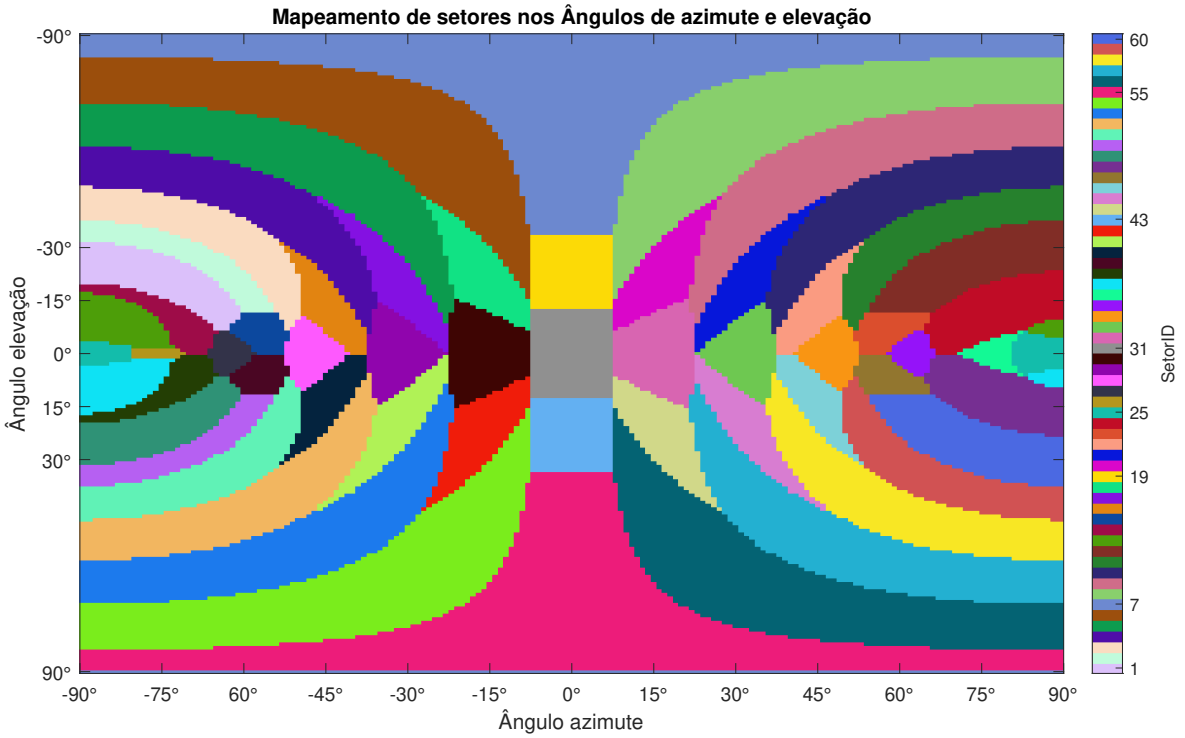

Figura 4.5: Mapeamento de diretividade do *codebook* de 60 setores.

4.4 Configuração dos Dispositivos no ns-3

A configuração dos dispositivos focou principalmente em definir as especificações do padrão 802.11ay, sobretudo nas camadas física e MAC. No entanto, o modelo não foge muito das abordagens mais comuns de criação e configuração de dispositivos de redes, na Listagem 8 está o código base, percebe-se que a partir da função `wifi.Setcodebook` está a definição do arquivo de leitura que pode ser escrita em código como um tipo string. Observa-se também a classe `codebook-parametric.cc`, responsável pela montagem do codebook nos dispositivos. A camada MAC dos nós é configurada através da função `wifiMac.SetType` que recebe por sua vez um objeto da classe `DmgApWifiMac` ou `DmgStaWifiMac`, sendo assim, são definidos os dispositivos AP e STA. A configuração de camada física é mais simples, a classe principal envolvida para a montagem é a `spectrumWifiPhy`. Conforme a Listagem 7, destaca-se a variável `txpower`, a potência de transmissão.

Com relação ao posicionamento dos nós, no código, o ns-3 deve considerar as posições como constantes conforme descreve a Listagem 9 por meio da classe `MobilityHelper`, enquanto que a determinação dos "passos"na simulação são definidas por meio de um índice. Na Listagem 6, o atributo `StartIndex` permite a simulação iniciar em qualquer passo, lembrando que, os passos são como amostras configuradas no arquivo `paraCfgCurrent.txt`. Caso a simulação em MATLAB dure 21 segundos e cada amostra ocorra a cada 0,7 segundos, serão geradas 30 amostras com posições diferentes ou não para os nós. Para que

a simulação no ns-3 possa acompanhar as amostras, basta adicionar o comando `qdPropagationEngine->SetAttribute ("Interval", TimeValue(Seconds(0.7)));`.

Complementando o processo de construção do canal mmWave, uma vez que, o ns-3 ainda não possui classes próprias desenvolvidas para a montagem do canal, sendo assim, toda montagem do canal de 60GHz é feito através dos arquivos de saída gerados pela simulação do MATLAB, na Listagem 6 segue os passos da classe `QdPropagationEngine`. Por meio da linha `qdPropagationEngine->SetAttribute` a classe recebe os dados compatíveis para o ns-3 em formato `.txt`. Em seguida a finalização por meio das definições referentes à perda no espaço livre, o modelo é definido por meio da classe auxiliar `MultiModelSpectrumChannel`.

4.5 Avaliação de Métodos para Cálculo da Distância

A próxima etapa de preparos foi a escolha de uma aplicação para gerar pacotes entre transmissor e receptor.

4.5.1 OnOff e Bulk

Duas aplicações legadas de códigos já presentes no repositório são OnOff e Bulk. A primeira tentativa de escolha de aplicação compreendia avaliar se seria possível obter dados que permitissem aferir o tempo de propagação dos pacotes com o uso de alguma dessas duas. Por se tratarem de aplicações já prontas para uso, a primeira abordagem adotada foi estudar as classes e funções disponíveis e verificar as possibilidades de satisfazer a proposta de localização.

Primeiramente a aplicação OnOff, cuja classe permite introduzir um controle no fluxo de pacotes UDP, ou seja, permite configurar seções ativas e de espera. Um problema recorrente é a filtragem dos pacotes que na maioria dos casos é imprecisa, múltiplos pacotes são reconhecidos no dispositivo receptor no mesmo tempo. Posto isto, a aplicação OnOff poderia ser uma solução possível, controlando os tempos ativos da aplicação espera-se que a filtragem dos pacotes seja mais precisa. Então, uma aplicação OnOff foi configurada, deixando o tempo ativo em 1 segundo e inativo em 3 segundos. Aqui, o tempo de percurso compreende o período desde o instante em que o pacote é gerado na aplicação transmissora até o instante em que ele chega por completo na aplicação receptora. Os tempos de percurso de cada pacote foram obtidos para plotar o gráfico apresentado na Figura 4.6.

Analisando a imagem pode-se notar alguns períodos onde o atraso dos pacotes cresce muito, representado pelos picos do gráfico, isso se deve aos períodos de contenção CBAP do padrão 802.11ay. Excluindo este evento em particular, os pacotes possuem um atraso acumulado devido a soma dos vários atrasos envolvidos no processo da transmissão. Separar

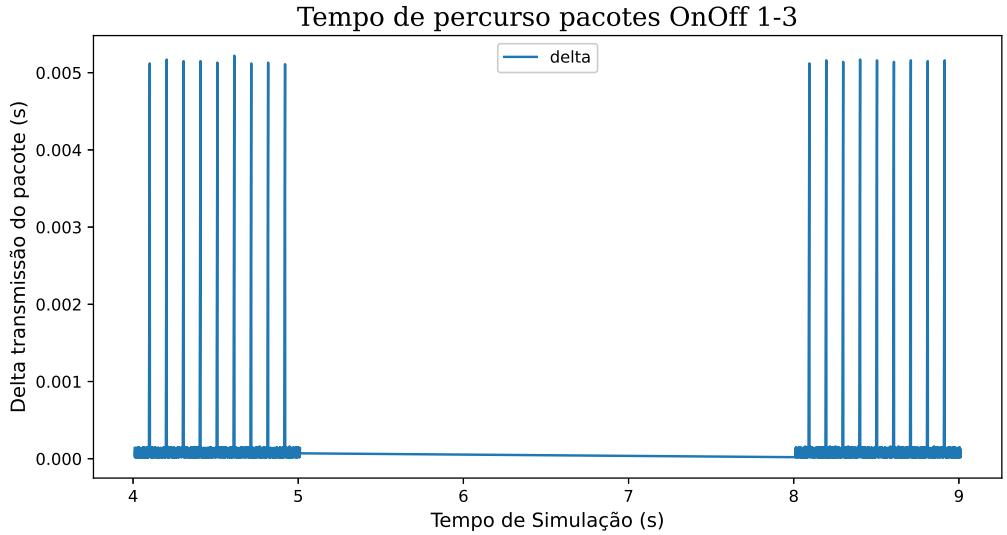

Figura 4.6: Tempo de percurso pacotes OnOff 1-3.

os vários atrasos para isolar o tempo de propagação foi a principal tentativa, porém a precisão se mostrou insuficiente, pois o tempo de propagação é da ordem de nanosegundos, enquanto que na classe OnOff os pacotes tinham tempos de percurso da ordem de microssegundos.

De modo similar a aplicação Bulk foi explorada, porém a ideia central era observar os efeitos do protocolo TCP na transmissão. De maneira similar, os resultados não foram proveitosos por conta da unidade das medidas obtidas.

4.5.2 Uso de Ping e Round Trip Time

Uma vez que não pareceu proveitoso tentar obter os atrasos de propagação através das aplicações anteriores, a próxima tentativa foi utilizar a aplicação de *ping*, a qual faz troca de pacotes ICMP entre transmissor e receptor, e conta com a métrica que mede o tempo de ida e volta de um pacote, o valor de *Round Trip Time* (RTT).

O código para a aplicação de *ping* é conforme apresentada na Listagem 3. No teste feito, o AP transmite pacotes para uma STA a cada $0,1024$ segundos, que foi o tempo encontrado para evitar as congestões periódicas apontadas na seção anterior. Ainda, no final desta Listagem aparece a linha de *callback* do evento `rtt`, responsável por chamar a função `PingRtt` toda vez que um novo RTT for computado. Esta função é apresentada na Listagem 4, toda vez que acionada, a função imprime o instante de tempo da chamada em segundos e o RTT em nanosegundos, assim como mostrado na Figura 4.7. Como pode ser observado, o valor recorrente do tempo de ida e volta se estabiliza em torno de $35,64$ μs. Neste caso, a distância entre os dispositivos era de 3 metros, e apenas com o RTT não seria possível retirar essa informação.

```
No segundo 1.00032 computou o RTT de um pacote. Valor encontrado: 321062 ns
No segundo 1.10244 computou o RTT de um pacote. Valor encontrado: 35642 ns
No segundo 1.20484 computou o RTT de um pacote. Valor encontrado: 35642 ns
No segundo 1.30724 computou o RTT de um pacote. Valor encontrado: 35642 ns
No segundo 1.40964 computou o RTT de um pacote. Valor encontrado: 35642 ns
No segundo 1.51204 computou o RTT de um pacote. Valor encontrado: 35642 ns
No segundo 1.61444 computou o RTT de um pacote. Valor encontrado: 35642 ns
No segundo 1.71684 computou o RTT de um pacote. Valor encontrado: 35642 ns
No segundo 1.81924 computou o RTT de um pacote. Valor encontrado: 35642 ns
No segundo 1.92164 computou o RTT de um pacote. Valor encontrado: 35642 ns
No segundo 2.02404 computou o RTT de um pacote. Valor encontrado: 35642 ns
No segundo 2.12644 computou o RTT de um pacote. Valor encontrado: 35642 ns
No segundo 2.22884 computou o RTT de um pacote. Valor encontrado: 35642 ns
```

Figura 4.7: Saída da aplicação *ping* na *console*.

4.5.3 Uso do Timestamp

A última aplicação avaliada para compor as simulações surgiu após encontrar um recurso disponível em uma versão mais recente do ns-3, o componente `seqTsSizeHeader`, responsável viabilizar a criação de pacotes em que o cabeçalho apresenta o número de sequência, o tamanho do pacote e uma marcação de tempo que indica o instante em que esse pacote foi criado, o *timestamp*. Conforme dito no início desta seção, a diferença de versões exigiu a inserção do componente ao código principal, o ns-3 IEEE 802.11ad/ay model. Fazer esse ajuste só foi possível porque o componente é relativamente independente dos demais presentes no simulador.

O próximo passo foi implementar uma aplicação que fizesse uso do `seqTsSizeHeader`. Inspirado nas aplicações existentes, foi criado o módulo chamado `TsLocation`, que, por sua vez, permitiu a criação do `TsLocationHelper`, que fornece funções para abstrair as operações de baixo nível e facilitar o uso do módulo em aplicações. A principal função do `TsLocation` é apresentada na Listagem 5. Em resumo, na primeira metade do código ocorre a criação de pacote e acréscimo das informações ao cabeçalho. Em seguida, o endereço de destino é obtido para então enviar o pacote. Após as linhas que configuram *logs* para acompanhamento em modo *debug*, eventos de envio são agendados para acontecer com o intervalo determinado pela aplicação que fizer a chamada desta função.

Na Listagem 10 aparece como foi configurada a aplicação `TsLocation` no código final, que foi feita utilizando o mesmo nome do módulo criado. A ideia desta aplicação é tornar possível a localização pelo método de estimativa de distância e direção, através da marcação de tempo incluída no pacote, que pode ser utilizada para mensurar o tempo de propagação no espaço e, consequentemente, a distância percorrida.

Como o objetivo é que o AP consiga fazer a estimativa de localização da STA, a aplicação de envio de pacotes é instalada na STA, e no AP é instalado a aplicação `PacketSink` que já está embutida no ns-3 e é responsável pela recepção de pacotes. A primeira metade do código da Listagem 10 mostra a configuração da STA. É válido notar que, no processo de adaptação do ns-3 para incluir o recurso de marcação de tempo nos pacotes, o componente

do `PacketSink` também precisou ser alterado para que na recepção essa marcação fosse reconhecida. Ambos, transmissor e receptor, habilitam o *timestamping* através do atributo `EnableSeqTsSizeHeader`.

Ainda no final da configuração da STA, um *stream* de escrita em arquivo é programado para coletar dados da simulação, o que ocorrerá sempre que acontecer a recepção de um pacote com *timestamp*. Nas linhas em seguida aparece a configuração do AP. De diferente, temos que definir o atributo *interval*, que dita o tempo entre pacotes, e o tamanho do pacote. O intervalo de $0,1024$ segundos foi escolhido estrategicamente para que a aplicação não fosse influenciada pelo período de contenção. O valor de 120 bytes define o tamanho do pacote, escolhido arbitrariamente, dos quais 20 bytes compõem o cabeçalho.

4.6 Validação da Camada de Aplicação

Pelo exposto na seção anterior, a aplicação de *timestamp* foi a escolhida para compor as simulações a serem realizadas. Com a marcação de tempo no pacote no transmissor, era esperado que, no receptor, fosse possível estimar o tempo que esse pacote demorou para percorrer a distância entre os dois dispositivos. Com base nos dados obtidos percebeu-se que, isoladamente, o *timestamp* não é suficiente para medir o tempo de propagação. Para tentar contornar este problema uma última implementação foi tentada, criação de períodos de serviço, com base no previsto pelo padrão e apresentado na Subseção 2.3.2. Basicamente, os períodos de serviço permitem reservar o canal para transmissão entre os dispositivos, sendo assim, trata-se de uma técnica interessante quando muitos nós estiverem competindo pelo canal, porém a tentativa a ser explorada é verificar se o tempo de reserva pode melhorar a precisão do *timestamp*. Através do processo de análise dos *logs* comentada na Subseção 4.1.1, outros atrasos, principalmente atrasos de backoff na camada MAC impedem de isolar completamente o atraso de propagação, mesmo com a implementação dos períodos de serviço, que não permite uma transmissão direta do pacote. Esse problema é ilustrado na Figura 4.8, que traz uma representação de como ocorre o fluxo de transmissão de pacotes nas camadas e no meio, considerando transmissor e receptor.

Representando o transmissor à esquerda, é visto que na camada de aplicação ocorre a marcação do pacote, que percorre a camada de enlace e depois a camada física, onde ocorre um intervalo de duração variável até que ocorra a transmissão no *link* de comunicação. Então, o pacote chega no receptor e passa pelas camadas até chegar na camada de aplicação e ser lido. O grande problema aqui é que a informação do intervalo variável no meio da transmissão não é capturada nem divulgada de um dispositivo para o outro. Dessa forma, a alternativa encontrada para proceder com as simulações foi utilizar das métricas controladas pelo simulador. Essas métricas são obtidas através dos dois primeiros *callbacks* apresentados no início da Listagem 12, os quais armazenam nas variáveis RxStarted e TxStarted os tempos de início de recepção e de início de transmissão nas camadas físicas dos dispositi-

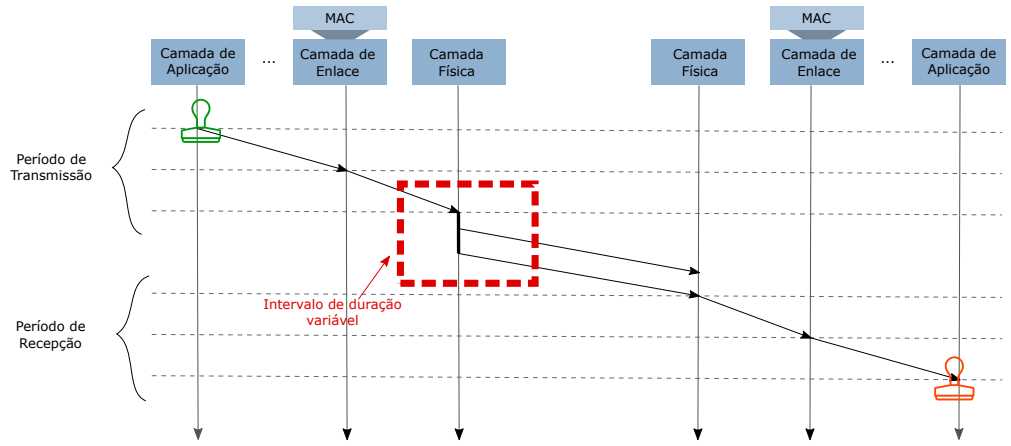

Figura 4.8: Progresso da transmissão de pacotes com *timestamp*.

vos. Assim, obter a aproximação do tempo de propagação é feito subtraindo TxStarted de RxStarted. Vale notar que, como a precisão do ns-3 é da ordem de nanosegundos, é esperado erro de até 30 centímetros na conversão de tempo de propagação para distância, pois:

$$(1ns) \cdot (3 \cdot 10^8 m/s) = 0,3m. \tag{4.2}$$

Outra alternativa tentada foi verificar a possibilidade de configurar o *Fine Time Measurement* (FTM) na simulação do ns-3. O FTM é um protocolo do padrão IEEE 802.11 usado para estimar a distância entre dois dispositivos na rede sem fio [Barral et al. 2022]. A ideia segue o mesmo conceito, marcar com um *timestamp* os pacotes FTM e verificar o tempo de RTT dos pacotes. O documento de implementação do FTM [wifi-ftm-ns3 2021] recomenda para uma melhor precisão nas medições alterar o tempo de resolução do ns-3 para picosegundos. Após testes, no entanto, a configuração acaba afetando outros objetos e funções que possuem unidades já definidas, isto é, o programa define uma variável em microsegundos que interfere diretamente com o comando solicitando valores em picosegundos.

Capítulo 5

Resultados

Este capítulo descreve as simulações realizadas e apresenta os resultados obtidos. Por padrão, as medidas de tempo “Timestamp”, “’TxStarted’, “RxStarted”, “RxCompleted” e “Tempo de Propagação” estão medidas em nanosegundos. A distância é sempre medida em metros.

5.1 Simulação de localização no ns-3

A simulação final para tentar localizar é a junção das configurações base discutidas na Seção 4.4 e na Subseção 4.5.3 envolvendo os dispositivos com duas antenas PAA cada e o canal mmWave, além da definição do *codebook* de 60 setores, elaborado conforme a Seção 4.3. Como já abordado anteriormente, a aplicação `TsLocation` usufrui dos recursos da classe `seqTsSizeHeader`, permitindo a marcação do tempo no momento em que os pacotes são gerados. As linhas *callback* presentes na Listagem 11 indicam algumas marcações importantes para checagem do andamento da simulação, pode-se observar desde a associação entre STA e AP e a varredura de setores durante a fase SISO. Já na Listagem 12 se encontram os comandos para extração dos dados referentes aos eventos de interesse. Além do *timestamp*, o simulador permite que os eventos de transmissão e recepção na camada física sejam obtidos, as funções `SignalArrival` e `MonitorSnifferTx` vão auxiliar nesse aspecto. Em seguida, serão apresentados os cenários realizados.

5.1.1 Cenário 1

5.1.1.1 Posicionamento dos Nós

- **Posições**: AP (1, 5, 1.5) e STA (4, 5, 1.5)
- **Codebook**: 60 setores
- **Ambiente**: Sala retangular *indoor*, dimensões (5x10x3)

A simulação possui um tempo de 2,5 segundos de duração, os períodos de associação e varredura ocorrem rapidamente, de tal modo que a aplicação pode ser configurada para iniciar no tempo de 1 segundo da simulação, as antenas também possuem uma identificação por meio dos IDs 0 e 1 para identificar as duas antenas.

5.1.1.2 Com Reflexões Ativadas

A configuração obtida pelo processo SISO indica os IDs das antenas selecionadas e o respectivo SNR medido, considerando os parâmetros iniciais, o link principal é formado pelos IDs 0 de ambos dispositivos com uma SNR de $27,6097$ dB. O principal arquivo de saída é uma tabela que armazena os principais dados de interesse. Além do *timestamp*, a saída computa também os tempos TxStarted e RxStarted, eventos que são registrados assim que um pacote chega na camada física do transmissor e receptor, enquanto que o tempo RxCompleted refere-se ao tempo que o pacote foi recebido na camada de aplicação no receptor. A Tabela 5.5 contém os dados do arquivo de saída da simulação no ns-3 referente aos primeiros 6 pacotes.

Cenário 1 - Com reflexões							
SeqNumber	BytesPayload	Timestamp	TxStarted	RxStarted	RxCompleted	Tempo de propagação	Distância
0	100	1000000000	1000114574	1000114583	1000120000	9	2.7
1	100	1102400000	1102401864	1102401873	1102407290	9	2.7
2	100	1204800000	1204801864	1204801873	1204807290	9	2.7
3	100	1307200000	1307201864	1307201873	1307207290	9	2.7
4	100	1409600000	1409601864	1409601873	1409607290	9	2.7
5	100	1512000000	1512001864	1512001873	1512007290	9	2.7

Tabela 5.1: Dados de TSLocation cenário 1.

Conforme os dados obtidos o tempo de propagação aproximado resultante foi de 9 nanosegundos. Em seguida, multiplicando o tempo obtido pela velocidade da luz, a distância será de 2,7 metros aproximadamente.

5.1.1.3 Com Reflexões Desativadas

O cenário 1 com as reflexões desabilitadas é apresentado na Figura 5.1. A simulação tem como saída os dados apresentados na Tabela 5.2. Percebe-se que não há diferenças significativas, indicando que o mesmo direcionamento foi selecionado em ambas simulações.

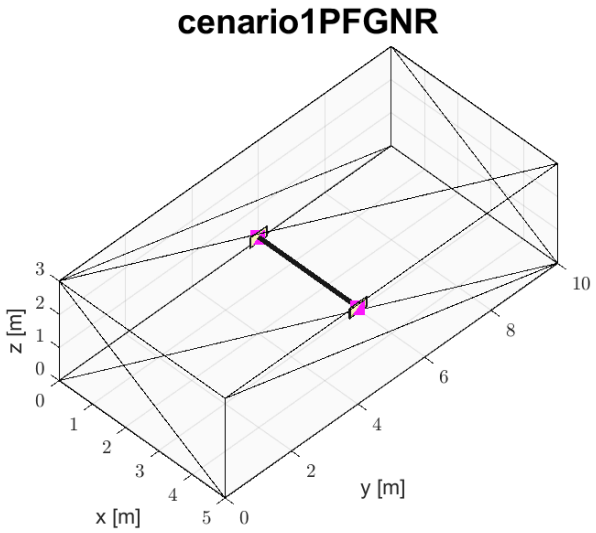

Figura 5.1: Cenário 1 sem reflexão.

Cenário 1 - Sem reflexões							
SeqNumber	BytesPayload	Timestamp	TxStarted	RxStarted	RxCompleted	Tempo de propagação	Distância
0	100	1000000000	1000114574	1000114583	1000120000	9	2.7
1	100	1102400000	1102401864	1102401873	1102407290	9	2.7
2	100	1204800000	1204801864	1204801873	1204807290	9	2.7
3	100	1307200000	1307201864	1307201873	1307207290	9	2.7
4	100	1409600000	1409601864	1409601873	1409607290	9	2.7
5	100	1512000000	1512001864	1512001873	1512007290	9	2.7

Tabela 5.2: Dados de TSLocation cenário 1 sem reflexão.

5.1.1.4 Estimativa de Localização

O próximo passo é verificar o melhor setor escolhido, conforme a organização do *codebook*, os ângulos de chegada dos feixes devem seguir a estrutura de direcionamento. Para o cenário 1, o setor 31 é o escolhido tanto para o AP quanto para a STA, ambos com SNR de 27.6097 dB, sendo assim, os ângulos θ e φ são ambos 0º. A representação do padrão de radiação desse setor é apresentada na Figura 5.2, onde é possível notar alta concentração de radiação no sentido positivo do eixo X, que é a porção do diagrama chamada de lóbulo principal, cuja definição é: a parte do digrama que corresponde ao ângulo em que a antena concentra a maior parte da sua energia. Ainda, na direção oposta deste está o lóbulo traseiro.

Na Figura 5.3 é apresentado o diagrama de cortes de azimute do setor 31. Os tracejados de -15º e +15º estão sobrepostos, por serem iguais. Nota-se que o corte equivalente à elevação de 0º prevalece no ângulo de 0º de azimute, com respeito à diretividade. Como os ângulos de diretividade oposta também estão representados no diagrama, isto é, (θ, φ) = (180º, 0º), também é possível visualizar a representação do lóbulo traseiro.

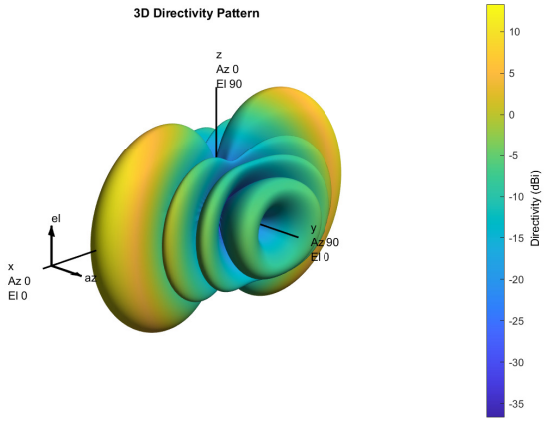

Figura 5.2: Padrão de radiação para o setor 31 do *codebook* de 60 setores.

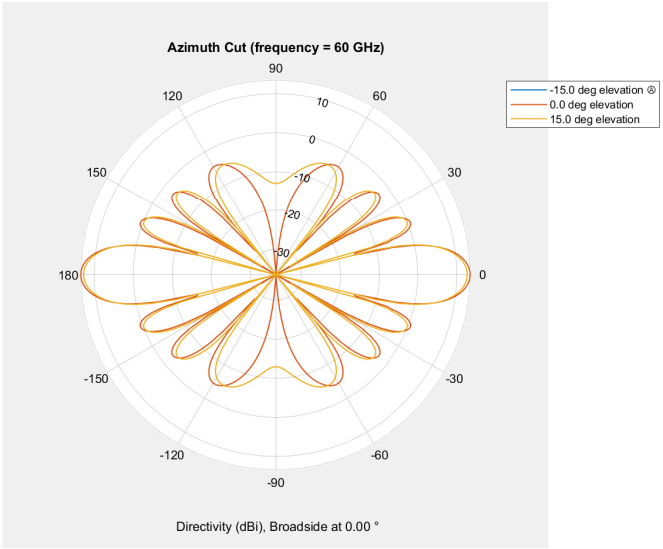

Figura 5.3: Cortes de azimute para o setor 31 do *codebook* de 60 setores.

Contudo, ainda é necessário verificar a convenção explicada na Subseção 3.1.2 para uso das equações de localização do sistema de coordenadas esférico. Posto isto, os ângulos (θ_1,φ_1) são (2π, $\frac{\pi}{2}$). Com este dado em mãos, basta utilizar as equações, sendo r = 2,7, a localização da STA (x,y,z) é como segue:

$$x = 2,7 \cdot cos(2\pi) \cdot sin(\frac{\pi}{2}) \tag{5.1a}$$

$$y = 2,7 \cdot sin(2\pi) \cdot sin(\frac{\pi}{2}) \tag{5.1b}$$

$$z = 2,7 \cdot cos(\frac{\pi}{2}) \tag{5.1c}$$

Por meio das equações a posição da STA obtida é (2.7, 0, 0):

- $x = 2,7 \cdot 1 = 2,7$
- $y = 2,7 \cdot 0 = 0$
- $z = 2,7 \cdot 0 = 0$

O centro de coordenadas é o AP (1, 5, 1.5), logo, é necessário fazer uma correção para checar a posição verdadeira em relação ao centro escolhido. Após esse ajuste, a localização do ponto obtida é (3.7, 5, 1.5), enquanto que a posição original é (4, 5, 1.5).

5.1.2 Cenário 2

5.1.2.1 Posicionamento dos Nós

A visualização de como está o posicionamento dos nós nesta simulação pode ser verificada na Figura 4.1. Citando novamente, esta simulação difere da anterior pela mudança de local da STA. Segue os parâmetros:

- **Posições**: AP (1, 5, 1.5) e STA (1, 8, 1.5)
- **Codebook**: 60 setores
- **Ambiente**: Sala retangular *indoor*, dimensões (5x10x3)

Assim, a menor distância entre os dispositivos permanece 3 metros. Entretanto, a STA está +90° de azimute com relação ao AP, de acordo com a convenção apresentada na Figura 3.1, o que torna esse caso interessante, já que não houve um setor projetado para ser direcionado ao valor de $\theta = 90°$, como pode ser observado na Figura 4.2.

5.1.2.2 Com Reflexões Ativadas

Nos resultados obtidos para as estações, foi encontrado que o AP utilizou a antena de ID 0 para fechar *link* com a antena de ID 1 da STA. Ambos utilizaram o setor 25, e os valores de SNR foram $26,7724$ dB e $26,7091$ dB, respectivamente. Com relação aos dados de saída referentes aos atrasos, por se tratar da mesma distância os valores são os mesmos da Tabela 5.5 indicando uma distância de 2,7 metros entre os dispositivos. Os ângulos de chegada já convertidos são ($\frac{3\pi}{4}$, $\frac{\pi}{2}$).

Na Figura 5.5 é apresentado o diagrama de cortes de azimute do setor 25. Os tracejados de -15º e +15º estão sobrepostos, por serem iguais. Nota-se que o corte equivalente à elevação de 0º prevalece no ângulo de -90º de azimute, com respeito à diretividade. Como os ângulos de diretividade oposta também estão representados no diagrama, isto é, (θ, φ) = (90º, 0º), também é possível visualizar a representação do lóbulo traseiro.

Cenário 2 - Com reflexões							
SeqNumber	BytesPayload	Timestamp	TxStarted	RxStarted	RxCompleted	Tempo de propagação	Distância
0	100	1000000000	1000114574	1000114583	1000120000	9	2.7
1	100	1102400000	1102401864	1102401873	1102407290	9	2.7
2	100	1204800000	1204801864	1204801873	1204807290	9	2.7
3	100	1307200000	1307201864	1307201873	1307207290	9	2.7
4	100	1409600000	1409601864	1409601873	1409607290	9	2.7
5	100	1512000000	1512001864	1512001873	1512007290	9	2.7

Tabela 5.3: Dados de TSLocation cenário 2.

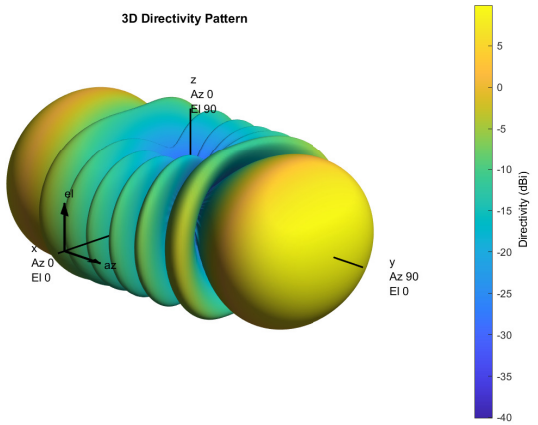

Figura 5.4: Padrão de radiação para o setor 25 do *codebook* de 60 setores.

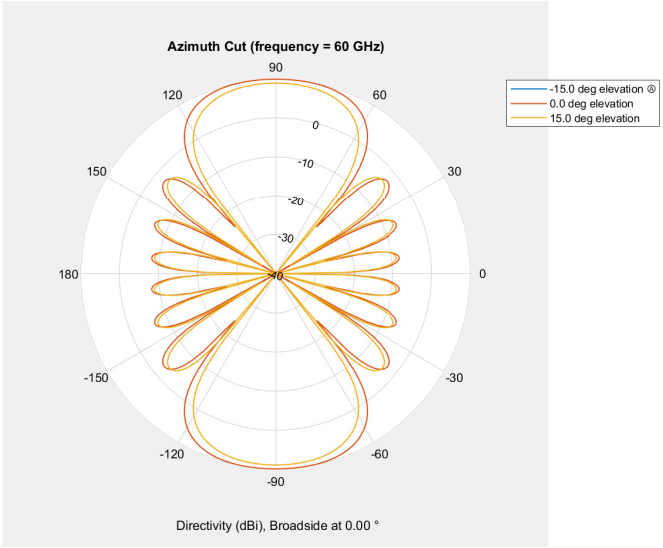

Figura 5.5: Cortes de azimute para o setor 25 do *codebook* de 60 setores.

5.1.2.3 Com Reflexões Desativadas

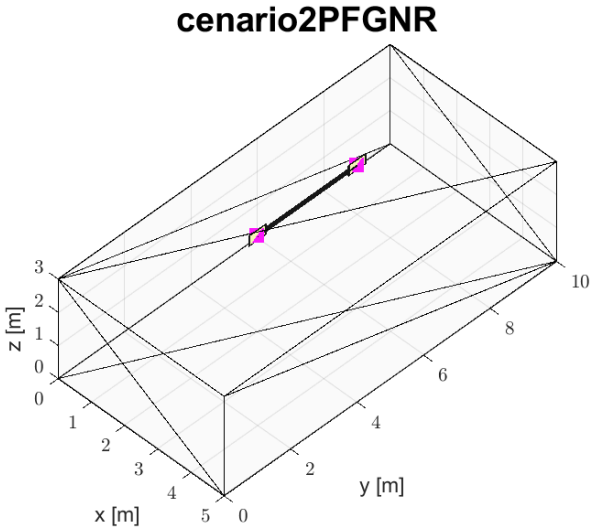

Figura 5.6: Cenário 2 sem reflexão.

O cenário 2 com as reflexões desabilitadas é apresentado na Figura 5.6. A simulação tem como saída os dados apresentados na Tabela 5.4. Novamente, a distância estimada pelo tempo de propagação foi de 2,7 metros, e as demais medidas que aparecem na tabela não sofreram modificações em relação ao cenário equivalente com reflexões ativadas. O link no sentido STA para a AP possui uma SNR medida de $26,6056$ dB, o sentido inverso possui uma SNR de $26,5183$ dB. Também, o setor escolhido pelo AP para se comunicar com a STA foi o 25, bem como no caso inverso.

Cenário 2 - Sem reflexões							
SeqNumber	BytesPayload	Timestamp	TxStarted	RxStarted	RxCompleted	Tempo de propagação	Distância
0	100	1000000000	1000114574	1000114583	1000120000	9	2.7
1	100	1102400000	1102401864	1102401873	1102407290	9	2.7
2	100	1204800000	1204801864	1204801873	1204807290	9	2.7
3	100	1307200000	1307201864	1307201873	1307207290	9	2.7
4	100	1409600000	1409601864	1409601873	1409607290	9	2.7
5	100	1512000000	1512001864	1512001873	1512007290	9	2.7

Tabela 5.4: Dados de TSLocation cenário 2 sem reflexão.

5.1.2.4 Estimativa de Localização

A posição da STA calculada é (0, -2.7, 0):

- $x = 2,7 \cdot 0 = 0$
- $y = 2,7 \cdot -1 = -2,7$
- $z = 2,7 \cdot 0 = 0$

Seguindo o processo estabelecido para localização do nó, segue o cálculo das coordenadas considerando o AP como centro de coordenadas. Após esse ajuste, a localização da STA obtida é (1, 2.3, 1.5), enquanto que a posição original é (1, 8, 1.5). Este caso é interessante pois o *codebook* da simulação não possui um setor próprio de apontamento para o ângulo de 90° no plano azimute, sendo assim, as antenas foram orientadas para o setor 25 com apontamento no sentido de -90°, o qual possui lóbulo traseiro apontado para o ângulo de 90°. A inversão de 180° impactou diretamente na estimativa da posição do nó.

5.1.3 Cenário 3

5.1.3.1 Posicionamento dos Nós

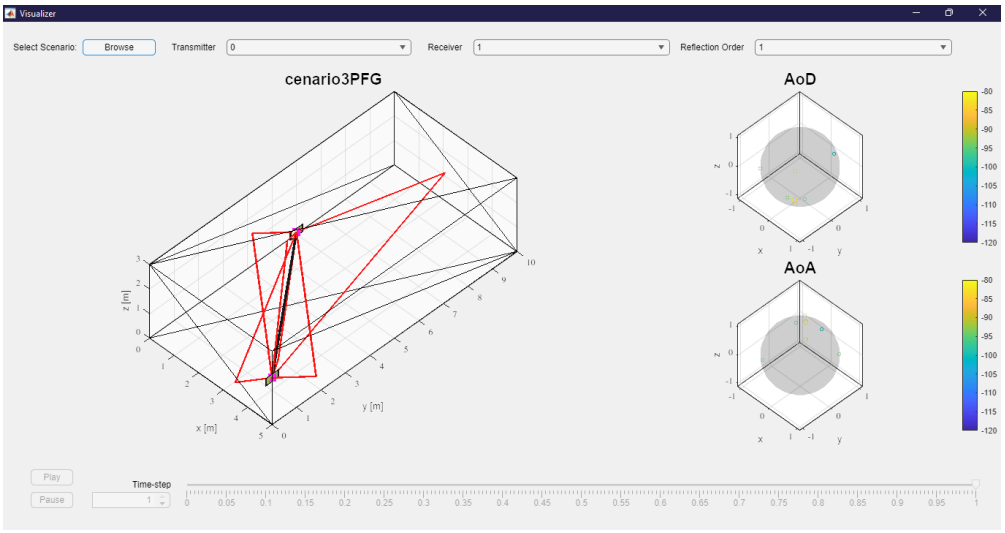

Figura 5.7: Posicionamento dos dispositivos no cenário 3.

A visualização da posição dos nós neste cenário é apresentada na Figura 5.7. Os parâmetros de posição foram definidos assim:

- **Posições**: AP (1, 5, 1.5) e STA (4, 1, 0.5)
- **Codebook**: 60 setores
- **Ambiente**: Sala retangular *indoor*, dimensões (5x10x3)

5.1.3.2 Com Reflexões Ativadas

Sobre os dispositivos, foi encontrado que ambos utilizaram suas antenas de ID 0 para fechar o *link* de comunicação. A distância entre os dispositivos é de aproximadamente 5,1 metros. O AP utilizou o setor 15 e a STA o 34. Os valores de SNR foram $21,2267$ dB e $21,7873$ dB, respectivamente. A partir do setor definido para o AP, os ângulos de chegada θ

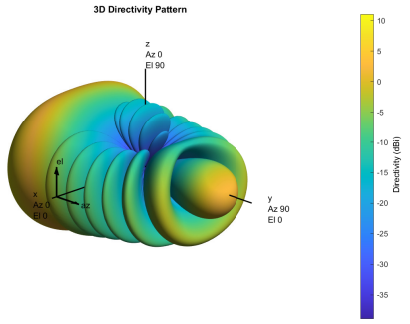

Figura 5.8: Padrão de radiação para o setor 15 do *codebook* de 60 setores.

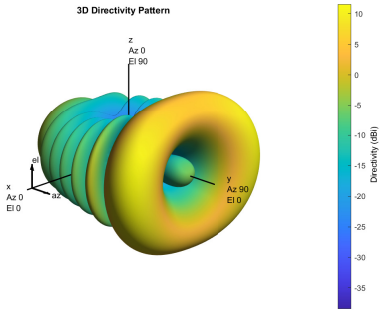

Figura 5.9: Padrão de radiação para o setor 34 do *codebook* de 60 setores.

e φ já convertidos são $(\frac{5\pi}{3}, \frac{7\pi}{12})$. Os padrões de radiação para os setores citados aparecem na Figura 5.8 e na Figura 5.9.

Na Figura 5.10 é apresentado o diagrama de cortes de azimute do setor 15. Nota-se que o corte equivalente à elevação de -15º prevalece no ângulo de -60º de azimute, com respeito à diretividade. Os ângulos de diretividade oposta (θ, φ) = (120º, 15º) não são representados no mesmo diagrama. Por isso o lóbulo traseiro não aparece representado.

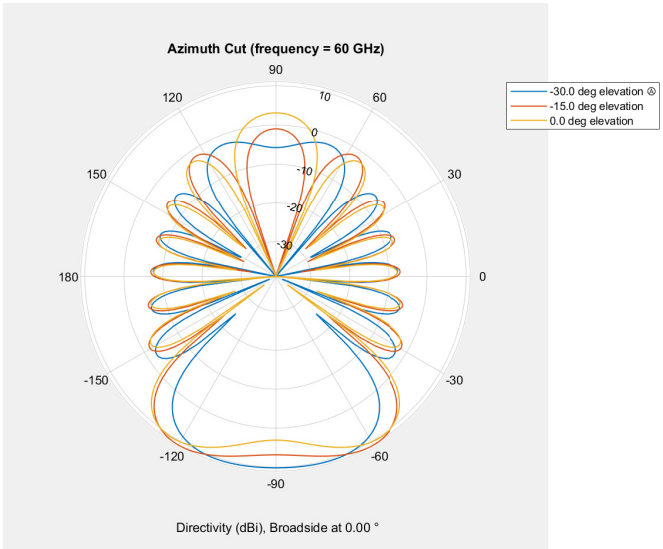

Figura 5.10: Cortes de azimute para o setor 15 do *codebook* de 60 setores.

Os dados de saída da simulação foram conforme a Tabela 5.5. Pela medição do tempo de propagação, a distância entre os dispositivos é de 4,8 metros.

Cenário 3 - Com reflexões							
SeqNumber	BytesPayload	Timestamp	TxStarted	RxStarted	RxCompleted	Tempo de propagação	Distância
0	100	1000000000	1000114595	1000114611	1000120028	16	4.8
1	100	1102400000	1102401871	1102401887	1102407304	16	4.8
2	100	1204800000	1204801871	1204801887	1204807304	16	4.8
3	100	1307200000	1307201871	1307201887	1307207304	16	4.8
4	100	1409600000	1409601871	1409601887	1409607304	16	4.8
5	100	1512000000	1512001871	1512001887	1512007304	16	4.8

Tabela 5.5: Dados de TSLocation cenário 3.

5.1.3.3 Com Reflexões Desativadas

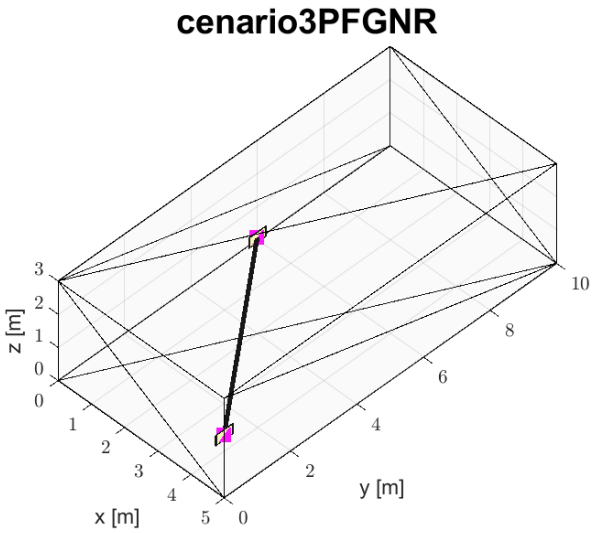

Figura 5.11: Cenário 3 sem reflexão.

O cenário 3 com as reflexões desabilitadas é apresentado na Figura 5.11. A simulação tem como saída os dados apresentados na Tabela 5.6. Em comparação com a Tabela 5.5, é possível verificar que os valores resultantes são os mesmos.

Cenário 3 - Sem reflexões							
SeqNumber	BytesPayload	Timestamp	TxStarted	RxStarted	RxCompleted	Tempo de propagação	Distância
0	100	1000000000	1000114595	1000114611	1000120028	16	4.8
1	100	1102400000	1102401871	1102401887	1102407304	16	4.8
2	100	1204800000	1204801871	1204801887	1204807304	16	4.8
3	100	1307200000	1307201871	1307201887	1307207304	16	4.8
4	100	1409600000	1409601871	1409601887	1409607304	16	4.8
5	100	1512000000	1512001871	1512001887	1512007304	16	4.8

Tabela 5.6: Dados de TSLocation cenário 3 sem reflexão.

Entretanto, a configuração de setores escolhidos pelos dispositivos sofreu alterações. O AP continua se comunicando por meio de setor 15, com $20,9824$ dB de SNR, mas a STA agora utiliza o setor 47, com mesmo valor de SNR. Como apenas o setor escolhido pelo AP é considerado na hora de estimar, o cálculo levará em conta o setor 15, o mesmo encontrado quando as reflexões estavam ativadas.

5.1.3.4 Estimativa de Localização

Seguindo os passos estabelecidos para estimar a localização, a posição da STA calculada é (2.31, -4.01, -1.24):

- $x = 4,8 \cdot 0,48 = 2,31$
- $y = 4,8 \cdot -0,83 = -4,01$
- $z = 4,8 \cdot -0,26 = -1,24$

Após os ajustes para considerar o AP o centro de coordenadas, a posição obtida é (3.31, 0.99, 0.26), enquanto que a posição original é (4, 1, 0.5).

5.1.4 Cenário 4

5.1.4.1 Posicionamento dos Nós

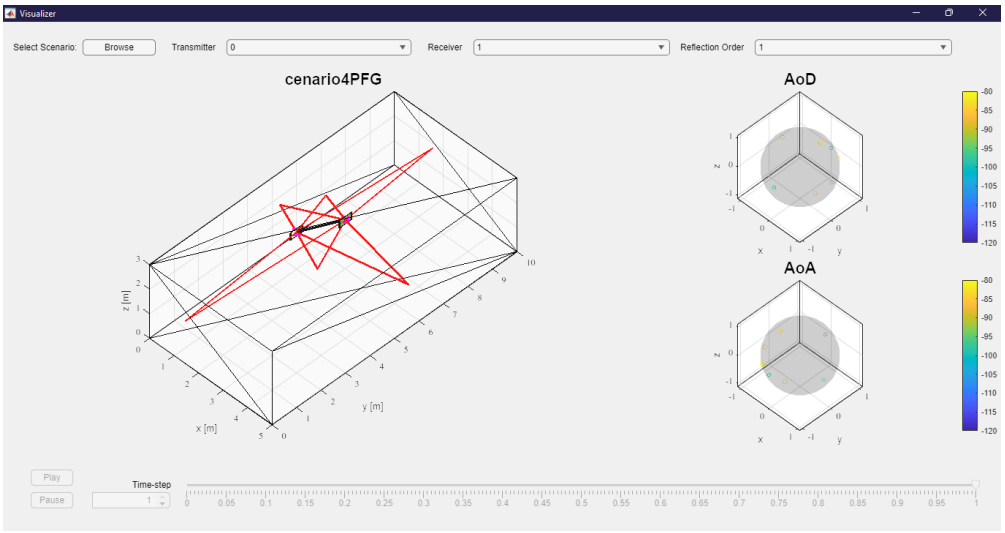

Figura 5.12: Posicionamento dos dispositivos no cenário 4.

A visualização da posição dos nós neste cenário é apresentada na Figura 5.12. Os parâmetros de posição foram definidos assim:

- **Posições**: AP (1, 5, 1.5) e STA (2, 6, 2)
- **Codebook**: 60 setores
- **Ambiente**: Sala retangular *indoor*, dimensões (5x10x3)

5.1.4.2 Com reflexões ativadas

Sobre os dispositivos, foi encontrado que o AP utilizou a antena de ID 0 e setor 46 para fechar *link* com a antena de ID 1 da STA, que utilizou o setor 16. Os valores de SNR foram $32,95$ dB e $32,94$ dB, respectivamente. A distância entre os dispositivos é de 1,5 metros e, com base nas medições com o atraso de propagação, a distância obtida é de aproximadamente 1,2 metros. Os ângulos convertidos para este cenário são $(\frac{9\pi}{4}, \frac{5\pi}{12})$.

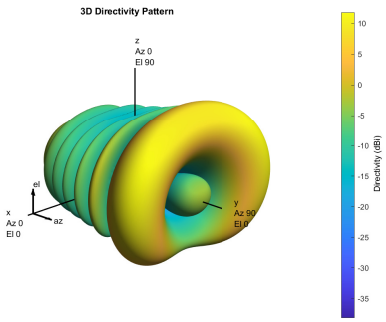

Figura 5.13: Padrão de radiação para o setor 46 do *codebook* de 60 setores.

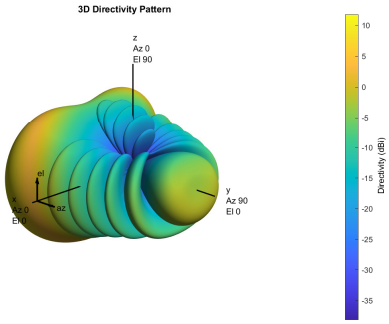

Figura 5.14: Padrão de radiação para o setor 16 do *codebook* de 60 setores.

Na Figura 5.15 é apresentado o diagrama de cortes de azimute do setor 46. Nota-se que o corte equivalente à elevação de 15º prevalece no ângulo de 45º de azimute, com respeito à diretividade. Os ângulos de diretividade oposta (θ, φ) = (-105º, -15º) não são representados no mesmo diagrama. Por isso o lóbulo traseiro não aparece representado.

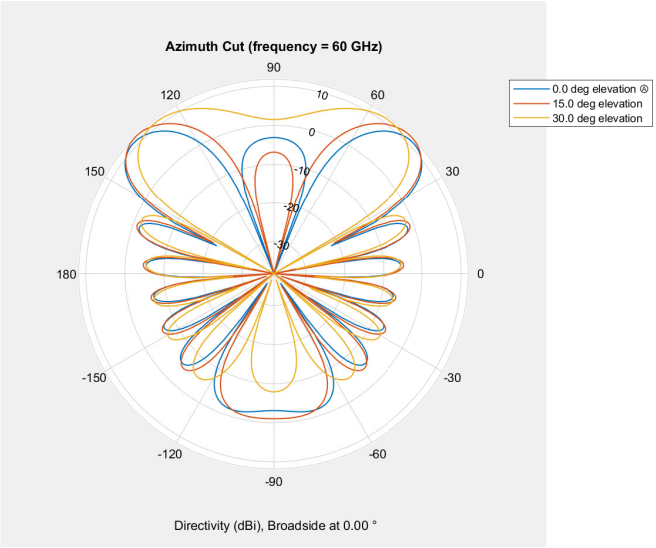

Figura 5.15: Cortes de azimute para o setor 46 do *codebook* de 60 setores.

Cenário 4 - Com reflexões							
SeqNumber	BytesPayload	Timestamp	TxStarted	RxStarted	RxCompleted	Tempo de propagação	Distância
0	100	1000000000	1000114559	1000114563	1000119980	4	1.2
1	100	1102400000	1102401859	1102401863	1102407280	4	1.2
2	100	1204800000	1204801859	1204801863	1204807280	4	1.2
3	100	1307200000	1307201859	1307201863	1307207280	4	1.2
4	100	1409600000	1409601859	1409601863	1409607280	4	1.2
5	100	1512000000	1512001859	1512001863	1512007280	4	1.2

Tabela 5.7: Dados de TSLocation cenário 4.

5.1.4.3 Com reflexões Desativadas

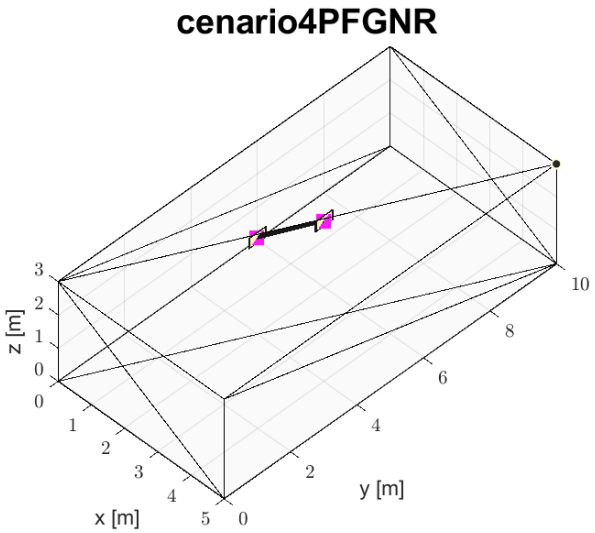

Figura 5.16: Cenário 4 sem reflexão.

O cenário 4 com as reflexões desabilitadas é apresentado na Figura 5.16. A simulação tem como saída os dados apresentados na Tabela 5.8. A mesma configuração de setores foi obtida, isto é, o mesmo direcionamento foi adotado. O link no sentido STA para a AP possui uma SNR medida de $32,9314$dB, o sentido inverso possui uma SNR de $32,9453$dB.

Cenário 4 - Sem reflexões							
SeqNumber	BytesPayload	Timestamp	TxStarted	RxStarted	RxCompleted	Tempo de propagação	Distância
0	100	1000000000	1000114559	1000114563	1000119980	4	1.2
1	100	1102400000	1102401859	1102401863	1102407280	4	1.2
2	100	1204800000	1204801859	1204801863	1204807280	4	1.2
3	100	1307200000	1307201859	1307201863	1307207280	4	1.2
4	100	1409600000	1409601859	1409601863	1409607280	4	1.2
5	100	1512000000	1512001859	1512001863	1512007280	4	1.2

Tabela 5.8: Dados de TSLocation cenário 4 sem reflexão.

5.1.4.4 Estimativa de localização

Seguindo os passos estabelecidos. A posição da STA calculada é (0.82, 0.82, 0.31):

- $x = 1,2 \cdot 0,68 = 0,82$

- $y = 1,2 \cdot 0,68 = 0,82$
- $z = 1,2 \cdot 0,26 = 0,31$

Após os ajustes, a posição obtida é (1.82, 5.82, 1.81), enquanto que a posição original é (2, 6, 2).

5.2 Avaliação geral

Nota-se que o cálculo do atraso de propagação foi aproximado, o que resultou na localização não precisa do dispositivo. Além do atraso, o modelo do *codebook* também é um fator de limitação, sua estrutura é de certa forma simples e o repositório possui um limite de até 64 setores possíveis, sendo que o caso ideal seria melhorar a varredura com mais antenas cobrindo mais ângulos. [wigig-tools 2021c].

Como foi mostrado, a localização não é precisa mesmo com a marcação de tempos na camada física dos dispositivos com a classe `seqTsSizeHeader`, ainda existem imprecisões nas medições. O *timestamp* não foi suficiente devido aos vários atrasos envolvidos, nem mesmo a configuração de períodos de serviço foi capaz de tornar a medição direta, uma vez que, os pacotes não são enviados assim que são criados na camada de aplicação, como foi verificado na análise de *logs*. O codebook também é um fator importante que deve ser considerado, implementação de arranjos mais robustos devem melhorar o mapeamento dos ângulos para localização mais precisa.

Ativar e desativar reflexões nas simulações serviu para sustentar a hipótese de que o AP estaria utilizando o setor que tem melhor SNR na direção do menor caminho em direção à STA. Como trata-se de cenários em que os dispositivos estão em LOS , o conhecimento da diretividade dos setores juntamente com a distância estimada foram utilizados para estimar a posição da STA. De forma geral, as distâncias encontradas ficaram dentro do intervalo de erro previsto, $0,3$ metros.

Conclusão

O trabalho explorou, em ambiente simulado, como pode ser feita a localização de um dispositivo por um AP em uma rede WLAN do padrão IEEE 802.11ay, que opera na faixa de 60 GHz. Para as simulações, foi utilizado um *framework* chamado WiGig Module, que conta com repositórios baseados em ns-3 e MATLAB, os quais viabilizam a modelagem de cenários em que os dispositivos se comunicam com base nesse padrão de redes sem fio. O sistema de coordenadas esféricas juntamente com o cálculo do atraso de propagação são a base do método de localização proposto, sendo assim, o controle do direcionamento dos feixes por meio da configuração do codebook e a marcação de tempo dos pacotes são processos chave.

Com base na análise dos resultados obtidos percebeu-se que a implementação de um *timestamp* para cálculo do atraso de propagação seria mais efetivo se o tempo marcado fosse exatamente o tempo em que o pacote foi transmitido, e não o tempo em que o pacote foi criado. Outra alternativa seria a implementação das marcações de tempo TxStarted e RxStarted exploradas na simulação, mas que em cenários práticos ainda não há uma implementação concreta de marcações precisas nos *timestamps* dos pacotes.

Para trabalhos futuros, a primeira sugestão é explorar o recurso de MIMO presente nos módulos do ns-3 IEEE 802.11ad/ay model. Isso seria mais eficiente porque utilizaria todas as antenas dos dispositivos. Entretanto, certa complexidade é adicionada, uma vez que, com o MIMO, a consideração de interferência mútua das antenas pode fazer com que o setor escolhido não tenha diretividade equivalente ao menor caminho entre os dispositivos.

Também fica como sugestão desenvolver estudos similares que explorem a presença de múltiplos dispositivos no ambiente e até mesmo condições de NLOS, que são desafios comuns em comunicações sem fio. Outra possibilidade de continuação é explorar a facilidade de geração de cenários para fazer uma base de dados que contemple várias situações e, eventualmente, usar os resultados obtidos para alimentar algoritmos de aprendizado de máquina.

De contribuição científica, espera-se que o trabalho sirva para evidenciar a possibilidade de explorar técnicas de localização na faixa de 60 GHz, além de ajudar apontar dificuldades encontradas para executá-las. Isso pode servir para justificar a iniciativa de pensar em melhorias que podem ser adicionadas ao padrão e aos dispositivos visando esse avanço tecnológico.

Referências Bibliográficas

[Assasa et al. 2021] Assasa, H., Grosheva, N., Ropitault, T., Blandino, S., Golmie, N., and Widmer, J. (2021). Implementation and evaluation of a wlan ieee 802.11ay model in network simulator ns-3. In *Proceedings of the Workshop on Ns-3*, WNS3 '21, page 9–16, New York, NY, USA. Association for Computing Machinery.

[Assasa et al. 2019] Assasa, H., Widmer, J., Ropitault, T., and Golmie, N. (2019). Enhancing the ns-3 ieee 802.11ad model fidelity: Beam codebooks, multi-antenna beamforming training, and quasi-deterministic mmwave channel. In *Proceedings of the 2019 Workshop on Ns-3*, WNS3 2019, page 33–40, New York, NY, USA.

[Barral et al. 2022] Barral, V., Campos, O., Domínguez-Bolaño, T., Escudero, C. J., and García-Naya, J. A. (2022). Fine time measurement for the internet of things: A practical approach using esp32. *IEEE Internet of Things Journal*, pages 1–1.

[Benson 2019] Benson, K. (2019). *Phased Array Beamforming ICs Simplify Antenna Design*, chapter 45. Analog Dialogues.

[Bourdoux et al. 2020] Bourdoux, A., Barreto, A. N., van Liempd, B., Lima, C., Dardari, D., Belot, D., Lohan, E.-S., Seco-Granados, G., Sarieddeen, H., Wymeersch, H., Suutala, J., Saloranta, J., Guillaud, M., Isomursu, M., Valkama, M., Aziz, M., Berkvens, R., Sanguanpuak, T., Svensson, T., and Miao, Y. (2020). 6g white paper on localization and sensing.

[Chabbar and Chami 2017] Chabbar, H. and Chami, M. (2017). Indoor localization using wi-fi method based on fingerprinting technique. In *2017 International Conference on Wireless Technologies, Embedded and Intelligent Systems (WITS)*, pages 1–5.

[Chan and Sohn 2012] Chan, S. and Sohn, G. (2012). Indoor localization using wi-fi based fingerprinting and trilateration techiques for lbs applications. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXVIII-4/C26:1–5.

[Chen et al. 2020] Chen, C., Wang, B., Lu, C. X., Trigoni, N., and Markham, A. (2020). A survey on deep learning for localization and mapping: Towards the age of spatial machine intelligence.

[Ghasempour et al. 2017] Ghasempour, Y., da Silva, C. R. C. M., Cordeiro, C., and Knightly, E. W. (2017). Ieee 802.11ay: Next-generation 60 ghz communication for 100 gb/s wi-fi. *IEEE Communications Magazine*, 55(12):186–192.

[Hany Assasa 2019] Hany Assasa (2019). wigig-tools. `https://github.com/wigig-tools/`. Acesso em: 25-01-2022.

[IEEE 2021a] IEEE (2021a). IEEE 802.11ay-2021 - IEEE Standard for Information Technology–Telecommunications and Information Exchange between Systems Local and Metropolitan Area Networks–Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 2: Enhanced Throughput for Operation in License-exempt Bands above 45 GHz. `https://standards.ieee.org/standard/802_11ay-2021.html`. Acesso em: 22-01-2022.

[IEEE 2021b] IEEE (2021b). IEEE Standard for Information Technology–Telecommunications and Information Exchange between Systems Local and Metropolitan Area Networks–Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 2: Enhanced Throughput for Operation in License-exempt Bands above 45 GHz.

[IEEE 2021c] IEEE (2021c). Status of Project IEEE 802.11ay. `https://www.ieee802.org/11/Reports/tgay_update.htm`. Acesso em: 22-01-2022.

[James W. Kurose 2021] James W. Kurose, K. W. R. (2021). *Computer Networking: A Top-Down Approach*. Pearson, 8 edition.

[Nitsche et al. 2014] Nitsche, T., Cordeiro, C., Flores, A. B., Knightly, E. W., Perahia, E., and Widmer, J. C. (2014). Ieee 802.11ad: directional 60 ghz communication for multi-gigabit-per-second wi-fi [invited paper]. *IEEE Communications Magazine*, 52(12):132–141.

[ns-3 2021a] ns-3 (2021a). ns-3 Manual. `https://www.nsnam.org/docs/manual/html/index.html`. Acesso em: 21-03-2022.

[ns-3 2021b] ns-3 (2021b). ns-3 Releases. `https://www.nsnam.org/releases/`. Acesso em: 25-01-2022.

[Qualcomm 2018] Qualcomm (2018). Qualcomm Dramatically Extends Wi-Fi Experiences to the 5G Era with 60GHz 802.11ay Solutions. `https://www.qualcomm.com/news/releases/2018/10/16/qualcomm-dramatically-extends-wi-fi-experiences-5g-era-60ghz-80211ay`. Acesso em: 22-01-2022.

[Sobehy 2020] Sobehy, A. (2020). *Machine learning based localization in 5G*. PhD thesis, Institut Polytechnique de Paris.

[Tom Carpenter 2007] Tom Carpenter, J. B. (2007). *CWNA Certified Wireless Network Administrator Official Study Guide (Exam PW0-100), Fourth Edition*. McGraw-Hill, Inc., USA, 4 edition.

[wifi-ftm-ns3 2021] wifi-ftm-ns3 (2021). FTM-ns3. `https://github.com/tkn-tub/wifi-ftm-ns3`. Acesso em: 20-04-2022.

[wigig-tools 2021a] wigig-tools (2021a). codebook-generator. `https://github.com/wigig-tools/codebook-generator`. Acesso em: 25-01-2022.

[wigig-tools 2021b] wigig-tools (2021b). qd-realization. `https://github.com/wigig-tools/qd-realization`. Acesso em: 25-01-2022.

[wigig-tools 2021c] wigig-tools (2021c). wigig-module. `https://github.com/wigig-tools/wigig-module`. Acesso em: 25-01-2022.

[Zhou et al. 2018] Zhou, P., Cheng, K., Han, X., Fang, X., Fang, Y., He, R., Long, Y., and Liu, Y. (2018). Ieee 802.11ay-based mmwave wlans: Design challenges and solutions. *IEEE Communications Surveys Tutorials*, 20(3):1654–1681.

Apêndice

I.1 Preparação de ferramentas e repositórios

Esta seção especifica quais ferramentas e repositórios precisam ser obtidos para a executar as ações expostas neste material, com os detalhes de procedimentos a serem realizados.

I.1.1 NIST Q-D Channel Realization Software

Para obter o código deste componente, basta clonar o repositório com o comando:

```
git clone https://github.com/wigig-tools/qd-realization
```

Após adquirir os arquivos, a execução será conforme especificado na Subseção 3.4.1.

I.1.2 ns-3 IEEE 802.11ad/ay Model

Para obter o código deste componente, o primeiro passo é clonar o repositório com o comando:

```
git clone https://github.com/wigig-tools/wigig-module.git
```

Dentro da pasta adquirida ao clonar o repositório, a montagem em modo *debug* pode ser feita com a execução dos seguintes comandos:

```
./waf configure --disable-examples --disable-tests --disable-python
↪  --enable-modules='core','applications','wifi','spectrum','flow-moni⌋
↪  tor','point-to-point','buildings'

./waf build
```

Já para a montagem em modo *optimized*, os comandos são como segue:

```
./waf configure --disable-examples --disable-tests --disable-python
↪  --enable-modules='applications','core','internet','point-to-point',
↪  'wifi','flow-monitor','spectrum' --enable-static -d
↪  optimized

./waf build
```

Assim como nas versões oficiais do ns-3, os códigos de execução de simulações estão localizados na pasta `scratch`. Para executar um código chamado `Programa.cc`, por exemplo, o comando é como segue:

```
./waf --run Programa1
```

II.2 Arquivos

Listagem 1: Arquivo de Modelagem do Canal Para o Cenário 3 Sem Reflexão.

```
1
1.69967e-08
-82.1545
5.05107
101.31
306.87
78.6901
126.87
1
1.70753e-08
-82.1946
3.25122
101.257
306.665
78.7428
126.665
1
1.69184e-08
-82.1144
0.653046
101.363
307.077
78.637
127.077
1
1.69967e-08
-82.1545
5.05107
101.31
306.87
78.6901
126.87
```

III.3 Códigos

III.3.1 Script Sensor Array Analyzer

Listagem 2: Script Sensor Array Analyzer.

```
% Create a uniform rectangular array
dth = 15;
th = -90: dth:90-dth;
ph = zeros(size(th));

th = [th th th th th];
ph = [-30+ph -15+ph ph 15+ph 30+ph];

%Assign steering angles, frequencies and propagation speed
SA = [th;ph];
f = 60000000000;
PS = 300000000;

h = phased.URA;
h.Size = [2 8];
h.ElementSpacing = [0.5 0.5]*PS/f;
h.Lattice = 'Rectangular';
%Calculate Row Taper
rwind = ones(1,8);
rwind = repmat(rwind,2,1);
%Calculate Column Taper
cwind = ones(1,2);
cwind = repmat(cwind.',1,8);
%Calculate taper
wind = rwind.*cwind;
h.Taper = wind;
%Create Isotropic Antenna Element
el = phased.IsotropicAntennaElement;
h.Element = el;

%Create figure, panel, and axes
fig = figure('Position', [100 100 900 600]);
panel = uipanel('Parent',fig);
hAxes = axes('Parent',panel,'Color','none');
%Calculate Steering Weights
w = zeros(getNumElements(h), length(th));
SV = phased.SteeringVector('SensorArray',h, 'PropagationSpeed', PS);

%Find the weights
catMatrix = [];
for idx = 1:length(th)
    w(:, idx) = step(SV, f, SA(:, idx));

    % plot polar azimuth cut 0 degree
    if ph(idx) == 0
        cutAngle = 0;
        pattern(h, f, -180:180, cutAngle, 'PropagationSpeed', PS, 'Type', ...
            'directivity', 'CoordinateSystem', fmt ,'weights', w(:,idx));
        % save images
        ax = gca;
        image_name = strcat('imagens/cut', num2str(idx), '.png');
        image_name_eps = strcat('imagens/cut', num2str(idx), '.eps');
        exportgraphics(ax,image_name);
        exportgraphics(ax,image_name_eps);
    end
```

```
56
57      %Plot 3d graph
58      fmt = 'polar';
59      pattern(h, f, 'PropagationSpeed', PS, 'Type','directivity', ...
60            'CoordinateSystem', fmt, 'weights', w(:,idx));
61
62      currentMatrix = pattern(h, f, 'PropagationSpeed', PS, 'Type','directivity', ...
63            'CoordinateSystem', fmt, 'weights', w(:,idx));
64      catMatrix = cat(3, catMatrix, currentMatrix);
65
66      % save images
67      ax = gca;
68      image_name = strcat('imagens/sector', num2str(idx), '.png');
69      image_name_eps = strcat('imagens/sector', num2str(idx), '.eps');
70      exportgraphics(ax,image_name);
71      exportgraphics(ax,image_name_eps);
72      drawnow();
73  end
74
75  mod_fase = kron(abs(w'),[1 0]) + kron(angle(w'),[0 1]);
76
77  [lines, max_nl] = size(ph);
78
79  file_wights = ("pesos.txt");
80  fout = fopen(file_wights,'w');
81
82  fprintf(fout,'%d\n', max_nl);
83
84  for nl=1:max_nl
85      fprintf(fout,'%d\n', nl);
86      fprintf(fout,'%d\n', 2);
87      fprintf(fout,'%d\n', 2);
88      for nc = 1:31
89          fprintf(fout,'%f', mod_fase(nl,nc));
90          fprintf(fout,'%s',',');
91
92      end
93      fprintf(fout,'%f\n', mod_fase(nc,32));
94  end
```

III.3.2 *Ping*

Listagem 3: Código da aplicação *ping*.

```
V4PingHelper ping = V4PingHelper (apInterface.GetAddress (0));
ping.SetAttribute ("Interval", TimeValue(Seconds(0.1024)));
ApplicationContainer pingApp = ping.Install (staWifiNode);
pingApp.Start (Seconds(appStartTime));
pingApp.Stop (Seconds (appEndTime));

Config::Connect ("/NodeList/*/ApplicationList/*/$ns3::V4Ping/Rtt", MakeCallback (&PingRtt))
    ;
```

Listagem 4: Função para imprimir os valores de RTT do *ping*.

```
PingRtt (std::string context, Time rtt)
{
    std::cout << "No segundo " << Simulator::Now().GetSeconds () << " computou o RTT de um
        pacote. Valor encontrado: ";
    std::cout << rtt.GetNanoSeconds () << " ns" << std::endl;
}
```

III.3.3 Módulo *TS Location*

Listagem 5: Função SendPackets da aplicação TS Location.

```
void
TsLocation::SendPacket (void)
{
    NS_LOG_FUNCTION (this);
    NS_ASSERT (m_sendEvent.IsExpired ());

    Ptr<Packet> packet;

    if (m_enableSeqTsSizeHeader)
    {
        Address from, to;
        m_socket->GetSockName (from);
        m_socket->GetPeerName (to);
        SeqTsSizeHeader header;
        header.SetSeq (m_seq++);
        header.SetSize (m_pktSize);
        NS_ABORT_IF (m_pktSize < header.GetSerializedSize ());
        packet = Create<Packet> (m_pktSize - header.GetSerializedSize ());
        // Trace before adding header, for consistency with PacketSink
        m_txTraceWithSeqTsSize (packet, from, to, header);
        packet->AddHeader (header);
    }
    else
    {
        packet = Create<Packet> (m_pktSize);
    }

    std::stringstream peerAddressStringStream;
    if (Ipv4Address::IsMatchingType (m_peerAddress))
    {
        peerAddressStringStream << Ipv4Address::ConvertFrom (m_peerAddress);
    }
    else if (Ipv6Address::IsMatchingType (m_peerAddress))
```

```
34      {
35          peerAddressStringStream << Ipv6Address::ConvertFrom (m_peerAddress);
36      }
37
38      if ((m_socket->Send (packet)) >= 0)
39      {
40        NS_LOG_INFO ("TraceDelay TX " << m_pktSize << " bytes to "
41                                      << peerAddressStringStream.str () << " Uid: "
42                                      << packet->GetUid () << " Time: "
43                                      << (Simulator::Now ()).GetSeconds ());
44
45      }
46      else
47      {
48        NS_LOG_INFO ("Error while sending " << m_pktSize << " bytes to "
49                                            << peerAddressStringStream.str ());
50      }
51      m_sendEvent = Simulator::Schedule (m_interval, &TsLocation::SendPacket, this);
52  }
```

III.3.4 Aplicação Final

Listagem 6: Configuração do canal do código final.

```
1   /**** Set up Channel ****/
2   Ptr<MultiModelSpectrumChannel> spectrumChannel = CreateObject<MultiModelSpectrumChannel>
        ();
3   qdPropagationEngine = CreateObject<QdPropagationEngine> ();
4   qdPropagationEngine->SetAttribute ("QDModelFolder", StringValue ("WigigFiles/QdChannel/"
        + qdChannelFolder + "/"));
5   Ptr<QdPropagationLossModel> lossModelRaytracing = CreateObject<QdPropagationLossModel> (
        qdPropagationEngine);
6   Ptr<QdPropagationDelayModel> propagationDelayRayTracing = CreateObject<
        QdPropagationDelayModel> (qdPropagationEngine);
7   spectrumChannel->AddSpectrumPropagationLossModel (lossModelRaytracing);
8   spectrumChannel->SetPropagationDelayModel (propagationDelayRayTracing);
9   qdPropagationEngine->SetAttribute ("StartIndex", UintegerValue (0));
10  //qdPropagationEngine->SetAttribute ("Interval", TimeValue(Seconds(0.7)));
```

Listagem 7: Configuração da camada física do código final.

```
1   /**** Setup physical layer ****/
2   SpectrumDmgWifiPhyHelper spectrumWifiPhy = SpectrumDmgWifiPhyHelper::Default ();
3   spectrumWifiPhy.SetChannel (spectrumChannel);
4   /* All nodes transmit at 10 dBm == 10 mW, no adaptation */
5   spectrumWifiPhy.Set ("TxPowerStart", DoubleValue (txPower));
6   spectrumWifiPhy.Set ("TxPowerEnd", DoubleValue (txPower));
7   spectrumWifiPhy.Set ("TxPowerLevels", UintegerValue (1));
8   /* Add a preamble detection model based on thresholds for the RSSI and SINR of the
        preamble. */
9   if (preambleDetection)
10  {
11      spectrumWifiPhy.Set ("PreambleDetectionModel", StringValue ("ns3::
            ThresholdPreambleDetectionModel"));
12      Config::SetDefault ("ns3::ThresholdPreambleDetectionModel::MinimumRssi", DoubleValue (
            preambleMinRssi));
13      Config::SetDefault ("ns3::ThresholdPreambleDetectionModel::Threshold", DoubleValue (
            preambleSnrThreshold));
14  }
```

```
/* Set operating channel */
EDMG_CHANNEL_CONFIG config = FindChannelConfiguration (channelNumber);
spectrumWifiPhy.Set ("ChannelNumber", UintegerValue (config.chNumber));
spectrumWifiPhy.Set ("PrimaryChannelNumber", UintegerValue (config.primayChannel));
/* Set the correct error model */
spectrumWifiPhy.SetErrorRateModel ("ns3::DmgErrorModel",
                                   "FileName", StringValue ("WigigFiles/ErrorModel/
                                       LookupTable_1458_ay.txt"));
```

Listagem 8: Configuração das estações do código final.

```
/* Install DMG PCP/AP Node */
Ssid ssid = Ssid ("SU-MIMO");
wifiMac.SetType ("ns3::DmgApWifiMac",
                 "Ssid", SsidValue (ssid),
                 "BE_MaxAmpduSize", StringValue (mpduAggSize),
                 "BE_MaxAmsduSize", StringValue (msduAggSize),
                 "SSSlotsPerABFT", UintegerValue (8), "SSFramesPerSlot", UintegerValue
                     (16),
                 "BeaconInterval", TimeValue (MicroSeconds (102400)),
                 "EDMGSupported", BooleanValue (true));

/* Set Parametric Codebook for the EDMG AP */
wifi.SetCodebook ("ns3::CodebookParametric",
                  "MimoCodebook", BooleanValue (true),
                  "TotalAntennas", UintegerValue (numStreams),
                  "FileName", StringValue ("WigigFiles/Codebook/CODEBOOK_URA_AP_" +
                      arrayConfig + ".txt"));

/* Create Wifi Network Devices (WifiNetDevice) */
NetDeviceContainer apDevice;
apDevice = wifi.Install (spectrumWifiPhy, wifiMac, apWifiNode);

wifiMac.SetType ("ns3::DmgStaWifiMac",
                 "Ssid", SsidValue (ssid), "ActiveProbing", BooleanValue (false),
                 "BE_MaxAmpduSize", StringValue (mpduAggSize),
                 "BE_MaxAmsduSize", StringValue (msduAggSize),
                 "EDMGSupported", BooleanValue (true));

/* Set Parametric Codebook for the EDMG STA */
wifi.SetCodebook ("ns3::CodebookParametric",
                  "MimoCodebook", BooleanValue (true),
                  "TotalAntennas", UintegerValue (numStreams),
                  "FileName", StringValue ("WigigFiles/Codebook/CODEBOOK_URA_STA_" +
                      arrayConfig + ".txt"));

staDevices = wifi.Install (spectrumWifiPhy, wifiMac, staWifiNode);
```

Listagem 9: Configuração de posicionamento nativa do ns-3 no código final.

```
/* Setting mobility model */
MobilityHelper mobility;
mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
mobility.Install (wifiNodes);
```

Listagem 10: Aplicação TsLocation no código final.

```
/* Install Sink Application on the EDMG STA */
PacketSinkHelper sinkHelper (socketType, InetSocketAddress (Ipv4Address::GetAny (), 9999));
sinkHelper.SetAttribute ("EnableSeqTsSizeHeader", BooleanValue (true));
ApplicationContainer sinkApp = sinkHelper.Install (apWifiNode);
packetSink = DynamicCast<PacketSink> (sinkApp.Get (0));

Ptr<OutputStreamWrapper> timestamp_file = ascii.CreateFileStream ("timestamp.csv");
*timestamp_file->GetStream () << "SeqNumber, BytesPayload, PcktTimestamp,"
                                 << "TxStarted, RxStarted, RxCompleted, Propagation,
                                     Distance" << std::endl;

packetSink->TraceConnectWithoutContext ("RxWithSeqTsSize", MakeBoundCallback (&ReceiveRx,
    timestamp_file));
sinkApp.Start (Seconds(appStartTime));
sinkApp.Stop (Seconds(simulationTime));

/* Install TsLocation Application on the EDMG AP */
Address dest (InetSocketAddress (apInterface.GetAddress (0), 9999));

TsLocationHelper tslApp (socketType, dest);
tslApp.SetAttribute ("Interval", TimeValue(Seconds(0.1024)));
tslApp.SetAttribute ("PacketSize", UintegerValue (packetSize));
tslApp.SetAttribute ("EnableSeqTsSizeHeader", BooleanValue (true));

ApplicationContainer srcApp = tslApp.Install (staWifiNode);
tslocation = DynamicCast<TsLocation> (srcApp.Get (0));

srcApp.Start (Seconds(appStartTime));
srcApp.Stop (Seconds (appEndTime));
```

Listagem 11: CallBacks de configuração inicial das estações no código final.

```
  /* EDMG AP Traces */
  beamformingTracerHelper->ConnectTrace (apWifiMac);
  apWifiMac->TraceConnectWithoutContext ("SLSCompleted", MakeBoundCallback (&SLSCompleted,
      apWifiMac));
  apWifiMac->TraceConnectWithoutContext ("SuMimoSisoPhaseCompleted", MakeBoundCallback (&
      SuMimoSisoPhaseCompleted, apWifiMac));

  /* EDMG STA Traces */
  beamformingTracerHelper->ConnectTrace (staWifiMac);
  staWifiMac->TraceConnectWithoutContext ("Assoc", MakeBoundCallback (&StationAssoicated,
      staWifiMac));
  staWifiMac->TraceConnectWithoutContext ("SLSCompleted", MakeBoundCallback (&SLSCompleted,
       staWifiMac));
  staWifiMac->TraceConnectWithoutContext ("DTIStarted", MakeBoundCallback (&
      DataTransmissionIntervalStarted, staWifiMac));
  staWifiMac->TraceConnectWithoutContext ("SuMimoSisoPhaseCompleted", MakeBoundCallback (&
      SuMimoSisoPhaseCompleted, staWifiMac));
```

Listagem 12: Fim da simulção e CallBacks de coleta de resultados e criação de saídas no código final.

```
/* Callbacks reception and transmission time*/
apWifiPhy->TraceConnectWithoutContext ("SignalArrival", MakeCallback (&APSignalArrival));
staWifiPhy->TraceConnectWithoutContext ("MonitorSnifferTx", MakeCallback (&STAPcktTx));


/* CalculateSNR callbacks*/
apWifiPhy->TraceConnectWithoutContext ("PhyRxEnd", MakeCallback (&PhyRxEnd));
apWifiPhy->TraceConnectWithoutContext ("PhyRxDrop", MakeCallback (&PhyRxDrop));

staWifiPhy->TraceConnectWithoutContext ("PhyTxEnd", MakeCallback (&PhyTxEnd));
staRemoteStationManager->TraceConnectWithoutContext ("MacTxDataFailed", MakeCallback (&
    MacTxDataFailed));


/* Get SNR Traces */
Ptr<OutputStreamWrapper> snrStream = ascii.CreateFileStream (tracesFolder + "snrValues.csv"
    );
apRemoteStationManager->TraceConnectWithoutContext ("MacRxOK", MakeBoundCallback (&
    MacRxOk_AP, snrStream));


Simulator::Stop (Seconds (simulationTime + 0.101));
Simulator::Run ();
Simulator::Destroy ();
```