

UNIVERSIDADE FEDERAL FLUMINENSE

JOELIAS SILVA PINTO JUNIOR

**O Consumo Energético da Comunicação
Dispositivo-a-Dispositivo em Redes Celulares**

NITERÓI

2019

UNIVERSIDADE FEDERAL FLUMINENSE

JOELIAS SILVA PINTO JUNIOR

O Consumo Energético da Comunicação Dispositivo-a-Dispositivo em Redes Celulares

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: SISTEMAS DE COMPUTAÇÃO.

Orientador:
IGOR MONTEIRO MORAES

NITERÓI

2019

Ficha catalográfica automática - SDC/BEE
Gerada com informações fornecidas pelo autor

P659c Pinto junior, Joelias Silva
O Consumo Energético da Comunicação Dispositivo-a-
Dispositivo em Redes Celulares / Joelias Silva Pinto junior ;
Igor Monteiro Moraes, orientador. Niterói, 2019.
198 p. : il.

Dissertação (mestrado)-Universidade Federal Fluminense,
Niterói, 2019.

DOI: <http://dx.doi.org/10.22409/PGC.2019.m.00195461142>

1. Consumo de energia. 2. Rede de comunicação de
computadores. 3. Redes de telecomunicação. 4. Computação
móvel. 5. Produção intelectual. I. Monteiro Moraes, Igor,
orientador. II. Universidade Federal Fluminense. Instituto de
Computação. III. Título.

CDD -

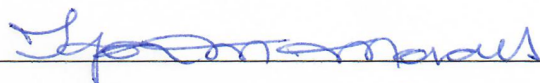
JOELIAS SILVA PINTO JUNIOR

O Consumo Energético da Comunicação Dispositivo-a-Dispositivo em Redes Celulares

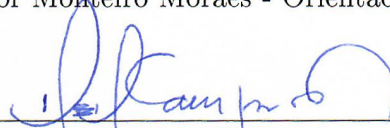
Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: SISTEMAS DE COMPUTAÇÃO.

Aprovada em julho de 2019.

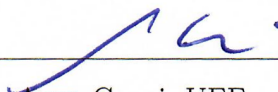
BANCA EXAMINADORA



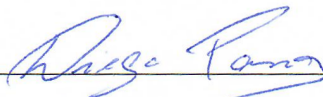
Prof. Igor Monteiro Moraes - Orientador, UFF



Prof. Miguel Elias Mitre Campista, UFRJ



Profª. Aura Conci, UFF



Prof. Diego Gimenez Passos, UFF

Niterói

2019

Aos meus pais e à minha irmã por serem sempre minha base.

A todos professores que me inspiraram à ciência.

Agradecimentos

À minha família e meu namorado por me aturarem nos momentos em que a cabeça estava cheia e a paciência curta. Por sempre me motivarem e acreditarem em mim, mesmo nos momentos que eu quis desistir. O nome disso é AMOR, e é recíproco! Vocês foram meu suporte incondicional.

Aos amigos Douglas, Claudineia e Kássio, que cotidianamente me ouviram, analisaram códigos e ideias comigo. Cada conversa me evoluiu um pouquinho e me motivou a continuar. Em nome da Bia, Kamila e Luiz, agradeço também aos demais amigos que sempre me apoiaram, cada um de sua forma.

Aos colegas de mestrado Erick e Simon, que se tornaram amigos pra vida. Em especial, Wilian e Clerison, vocês foram essenciais para que eu chegasse até aqui. Dividir a rotina com vocês tornou este caminho mais leve.

Ao meu orientador, que colocou os valores “motivação” e “persistência” acima das habilidades técnicas. Visões humanas como a sua transformam a motivação de seus orientados e têm um impacto maior do que possa imaginar. Por mim e por todos que tiveram o privilégio de trabalhar sob sua orientação, nunca se desvencilhe destes valores. Nunca deixe de acreditar mais nas pessoas do que nas técnicas. Imensa gratidão! :-)

Aos professores Aura, Isabel Rosseti, Diego, Lúcia, Simone e Fábio, por terem demonstrado paixão pelo que fazem e se esforçado pra nos instruir o máximo. Os ensinamentos de vocês me deram uma nova visão sobre cada área técnica que lecionaram e, hoje, eu os replico em minhas aulas.

À Marister por ter me adotado sem perceber e ser a mãe que tanto precisamos quando estamos longe de casa, família, amigos e sob a pressão dos estudos e prazos. Obrigado por tanto cuidado e pelos inúmeros momentos de descontração.

À Renata, a quem serei eternamente grato, por ter sido minha mãe na academia e se tornado amiga pra vida. Por ser a professora que mais me motivou a seguir a carreira acadêmica e me inspirou ser professor. Por me aconselhar e incentivar em cada projeto,

artigo e etapa de minha vida. Em cada título meu, haverá sempre um pedacinho seu.

Ao Laboratório MídiaCom, em nome da Prof^a Débora, por humanizar as frias relações acadêmicas, por proporcionar um espaço aconchegante e descontraído que nos faz ter vontade de frequentar.

Aos professores da UFMT Campus Universitário do Araguaia, Linder e Ivairton, por tão prontamente me atenderem e me conduzirem a reflexões indispensáveis para finalização de minha pesquisa.

Ao IFMT e seus dirigentes pela realização do MINTER como meio de investir na capacitação de seus servidores.

Em nome da UFF, agradeço a todas instituições de ensino públicas que me formaram. O conhecimento transformou a minha vida, amadureceu meu ser e deu sentido à minha existência. Que o ensino público de qualidade nunca deixe de ser prioridade!

“O rápido crescimento de redes sem fio pode, em breve, conduzir a uma verdadeira computação distribuída e pervasiva... O desafio para os engenheiros de software será o de desenvolver sistemas e software aplicativo que permitam que dispositivos móveis, computadores pessoais e sistemas corporativos se comuniquem através de extensas redes.”

Roger S. Pressman[46]

Resumo

Os últimos anos têm apresentado um acelerado ritmo de crescimento em número de dispositivos, tráfego de dados e quantidade de informações produzidas na Internet. De 2007 a 2017, por exemplo, o tráfego global da Internet cresceu de 2 TB para 46,6 TB por segundo. A categoria de dispositivos que mais cresceu neste período foi a de telefones móveis e a previsão é que até 2022 mais de 50% do tráfego da Internet seja proveniente destes dispositivos. Ao mesmo tempo, quanto ao tipo de comunicação, na Dispositivo-a-Dispositivo (D2D) se espera um crescimento de 8,5 bilhões de conexões entre 2017 a 2022. Com estas características de crescimento, em breve as redes 4G excederão suas capacidades de quantidade de dispositivos conectados e tráfego de dados. Além disso, a 4G oferece outras limitações que impossibilitam a implantação de novos serviços. Por exemplo, estima-se que um veículo autônomo precise enviar dados a uma taxa de 1 GB/s, com latência próxima de zero. A 4G torna inviável este tipo de aplicação pois, tecnicamente, pode atingir velocidade de apenas 100 Mbps, com alta latência e com necessidade de melhorar a eficiência energética, crítica para os dispositivos móveis. Por isto, a nova geração de telefonia móvel, a 5G, tem sido pensada para atingir altas velocidades de conexão, com baixa latência e para permitir a interconexão de milhões de dispositivos. Para tanto, várias tecnologias serão necessárias, dentre elas, a D2D, pela qual dispositivos podem se comunicar diretamente entre si, atingindo maiores taxas de vazão, menor latência e reduzir o tráfego da estação base. Diversos são os benefícios que podem ser elencados sobre a D2D. No entanto, o que ainda pouco se discute sobre este tipo de comunicação são os impactos no consumo energético dos nós encaminhadores. Em função das baterias dos dispositivos móveis, que são limitadas, é essencial o desenvolvimento de estudos que possibilitem analisar formas de economizar energia. Por isto, este trabalho focou em realizar simulações de comunicação D2D em redes móveis, com o simulador SimuLTE, para identificar comportamentos de consumo de energia da encaminhamento na comunicação D2D. Foram definidos 8 experimentos com 1 célula de comunicação D2D gerenciada pela operadora, com diferentes tipos de protocolos, aplicações e mobilidade entre eles. Para cada experimento, fizemos 6 variações no cenário, referentes à quantidade de nós encaminhadores, variando na forma de 0, 10, 25, 50, 75 e 100% de encaminhadores. Dos resultados obtidos, foram analisadas a quantidade de mensagens enviadas e recebidas por nó e por cada configuração de rede, que possibilitaram compreender o consumo energético neste tipo de rede, com e sem encaminhamento. Em alguns cenários verificou-se que a inserção dos encaminhadores aumenta o consumo energético da rede e, em outros, reduz. Estes resultados, em que se consome menos energia, rompem com o empirismo de que a encaminhamento consumirá sempre mais energia, e se postula como método de redução de consumo energético para novas pesquisas em redes móveis. Já os resultados que demonstram maior consumo energético são úteis para analisar o perfil de consumo da rede conforme a quantidade de encaminhadores inseridos na mesma.

Palavras-chave: D2D, encaminhamento, consumo energético, 5G.

Abstract

In recent years we have seen a fast growth in the number of devices, data traffic and the amount of information produced in the Internet. From 2007 to 2017, for instance, global Internet traffic grew from 2TB to 46.6TB per second. The fastest growing category of devices in this period was mobile phones, and by 2022 it is predicted that more than 50% of Internet traffic will come from mobile devices only. Regarding the type of communication, Device-to-Device (D2D) is the one that stands out, as it is expected to grow 8.5 billion between 2017 and 2022. With these growth characteristics, 4G networks will exceed their capability in terms of connected devices and data traffic. In addition, 4G offers other limitations that make it impossible to deploy new services. For example, it is estimated that a stand-alone vehicle needs to send data at a rate of 1 GB/s, with latency close to zero. 4G makes this type of application impractical because, technically, it can reach speeds of only 100Mbps, with high latency and without considering energy efficiency, critical for mobile devices. Hereby, the new generation of mobile cellular communications, the 5G, has been designed to achieve high connection speeds, with low latency and enable this necessary growth. To do so, several technologies will be required, among them, D2D, by which devices can communicate directly with each other, reaching higher data rates, lower latency and reduce base station traffic. Several benefits can be listed about D2D. However, the impacts on the energy consumption of the relay nodes in this type of communication are still little discussed. Due to limited mobile device batteries, it is essential to develop studies that allow us to analyze ways to save energy. Therefore, this work focused on simulating D2D communication in mobile networks, using the SimuLTE simulator, to identify the energy consumption behavior on relaying in D2D communication. Eight experiments were defined with 1 cell of D2D communication managed by the operator, with different types of protocols, applications and mobility between them. For each experiment, we made 6 variations in the scenario, referring to the number of relaying nodes, varying between 0, 10, 25, 50, 75 and 100% of relays. From the results obtained, we analyzed the number of messages sent and received per node and for each network configuration, which made it possible to understand the energy consumption in this type of network, with and without retransmission. In some scenarios we find out introducing relays increases the energy consumption of the network and in others, it reduces. These results, that more relays may less energy, break with the empiricism that retransmission will always waste more energy, and is postulated as a method of reducing energy consumption for new research in mobile networks. The results that demonstrate higher energy consumption are useful to analyze the consumption profile of the network according to the amount of relays introduced.

Keywords: D2D, relaying, power consumption, 5G.

Lista de Figuras

1.1	Previsão de tráfego até 2022. Fonte: Cisco VNI Forecast 2017–2022 [16]. . .	2
1.2	Taxas de crescimento de tráfego de Internet móvel e fixa em 2017. Fonte: Cisco VNI Forecast 2017–2022 [16].	2
1.3	Previsão de taxa de crescimento de tráfego por tipos de dispositivos entre 2017 a 2022. Fonte: Cisco VNI Forecast 2017–2022 [16].	3
1.4	Previsão de taxa de crescimento de conexões M2M entre 2017 a 2022. Fonte: Cisco VNI Forecast 2017–2022 [16].	4
2.1	Esquema de conexão entre em redes 4G.	10
2.2	Arquitetura do protocolo de rádio LTE. Adaptado de [53].	11
2.3	Arquitetura do <i>User Plane</i> e do <i>Data Plane</i>	12
2.4	Demonstração das técnicas FDD e TDD.	13
2.5	Cronograma previsto para padronização da 5G. Fonte: <i>Tentative 3GPP timeline for 5G</i> [4].	14
2.6	Fases da 5G. Fonte: 3GPP [8].	15
2.7	Esquema de comunicação em duas camadas.	16
2.8	Esquema de comunicação com BS em nuvem.	18
3.1	Célula D2D controlada pela eNB.	22
3.2	Curva de depleção de bateria de íon de lítio da Panasonic Sanyo. Adaptado de [42].	25
4.1	Posicionamento do módulo power na estrutura de diretórios do INET. . . .	32
5.1	Vsualização gráfica da arquitetura de rede D2D base.	40
5.2	Esquema de comunicação com e sem encaminhamento.	47

5.3	Número de bytes transmitidos e recebidos no experimento multicast com 4 UEs.	51
5.4	Consumo energético da rede no experimento multicast.	52
5.5	Consumo energético dos UEs Rx no experimento TCP.	53
5.6	Consumo energético dos UEs Tx no experimento TCP.	53
5.7	Consumo energético dos UEs Tx no experimento TCP com 4 encaminhadores.	54
5.8	Consumo energético dos UEs Tx no experimento TCP com 10 encaminhadores.	54
5.9	Consumo energético dos UEs Tx no Experimento TCP com 20 encaminhadores.	55
5.10	Consumo energético dos UEs Tx no experimento TCP com 30 encaminhadores.	55
5.11	Consumo energético dos UEs Tx no experimento TCP com 40 encaminhadores.	56
5.12	Consumo energético dos UEs Rx no experimento TCP com 4 encaminhadores.	57
5.13	Consumo energético dos UEs Rx no experimento TCP com 10 encaminhadores.	57
5.14	Consumo energético dos UEs Rx no experimento TCP com 20 encaminhadores.	58
5.15	Consumo energético dos UEs Rx no experimento TCP com 30 encaminhadores.	58
5.16	Consumo energético dos UEs Rx no experimento TCP com 40 encaminhadores.	59
5.17	Consumo energético total da rede e dos UEs Rx e Tx no Experimento TCP para cada Cenário.	60
5.18	Consumo energético dos UEs Rx no experimento UDP 2 m/s.	61
5.19	Consumo energético dos UEs Tx no experimento UDP 2 m/s.	61
5.20	Consumo dos UEs Tx no experimento UDP 2 m/s com 10 encaminhadores.	62
5.21	Consumo dos UEs Rx no experimento UDP 2 m/s com 10 encaminhadores.	63

5.22	Consumo energético dos UEs Tx no experimento UDP 2 m/s com 4 encaminhadores.	63
5.23	Consumo energético dos UEs Tx no experimento UDP 2 m/s com 10 encaminhadores.	64
5.24	Consumo energético dos UEs Tx no experimento UDP 2 m/s com 20 encaminhadores.	64
5.25	Consumo energético dos UEs Tx no experimento UDP 2 m/s com 30 encaminhadores.	65
5.26	Consumo energético dos UEs Tx no experimento UDP 2 m/s com 40 encaminhadores.	65
5.27	Consumo energético dos UEs Rx no experimento UDP 2 m/s com 4 encaminhadores.	66
5.28	Consumo energético dos UEs Rx no experimento UDP 2 m/s com 10 encaminhadores.	66
5.29	Consumo energético dos UEs Rx no experimento UDP 2 m/s com 20 encaminhadores.	67
5.30	Consumo energético dos UEs Rx no experimento UDP 2 m/s com 30 encaminhadores.	67
5.31	Consumo energético dos UEs Rx no experimento UDP 2 m/s com 40 encaminhadores.	68
5.32	Consumo energético total da rede e dos UEs Rx e Tx no experimento UDP 2 m/s para cada cenário.	69
5.33	Consumo energético dos UEs Rx no experimento UDP 1 m/s.	70
5.34	Consumo energético dos UEs Tx no experimento UDP 1 m/s.	70
5.35	Consumo energético dos UEs Tx no experimento UDP 1 m/s com 4 encaminhadores.	71
5.36	Consumo energético dos UEs Tx no experimento UDP 1 m/s com 10 encaminhadores.	71
5.37	Consumo energético dos UEs Tx no experimento UDP 1 m/s com 20 encaminhadores.	72

5.38 Consumo energético dos UEs Tx no experimento UDP 1 m/s com 30 encaminhadores.	72
5.39 Consumo energético dos UEs Tx no experimento UDP 1 m/s com 40 encaminhadores.	73
5.40 Consumo energético dos UEs Rx no experimento UDP 1 m/s com 4 encaminhadores.	73
5.41 Consumo energético dos UEs Rx no experimento UDP 1 m/s com 10 encaminhadores.	74
5.42 Consumo energético dos UEs Rx no experimento UDP 1 m/s com 20 encaminhadores.	74
5.43 Consumo energético dos UEs Rx no experimento UDP 1 m/s com 30 encaminhadores.	75
5.44 Consumo energético dos UEs Rx no experimento UDP 1 m/s com 40 encaminhadores.	75
5.45 Consumo energético total da rede e dos UEs Rx e Tx no experimento UDP 1 m/s para cada cenário.	76
5.46 Consumo energético dos UEs Rx no experimento UDP 5 m/s.	77
5.47 Consumo energético dos UEs Tx no experimento UDP 5 m/s.	77
5.48 Consumo energético dos UEs Tx no experimento UDP 5 m/s com 4 encaminhadores.	78
5.49 Consumo energético dos UEs Tx no experimento UDP 5 m/s com 10 encaminhadores.	78
5.50 Consumo energético dos UEs Tx no experimento UDP 5 m/s com 20 encaminhadores.	79
5.51 Consumo energético dos UEs Tx no experimento UDP 5 m/s com 30 encaminhadores.	79
5.52 Consumo energético dos UEs Tx no experimento UDP 5 m/s com 40 encaminhadores.	80
5.53 Consumo energético dos UEs Rx no experimento UDP 5 m/s com 4 encaminhadores.	80

5.54	Consumo energético dos UEs Rx no experimento UDP 5 m/s com 10 encaminhadores.	81
5.55	Consumo energético dos UEs Rx no experimento UDP 5 m/s com 20 encaminhadores.	81
5.56	Consumo energético dos UEs Rx no experimento UDP 5 m/s com 30 encaminhadores.	82
5.57	Consumo energético dos UEs Rx no experimento UDP 5 m/s com 40 encaminhadores.	82
5.58	Consumo energético total da rede e dos UEs Rx e Tx no experimento UDP 5 m/s para cada cenário.	83
5.59	Consumo energético do experimento UDP com mobilidade de 1 a 5 m/s.	84
5.60	Consumo energético do experimento TCP e UDP com mobilidade de 2 m/s.	85
5.61	Consumo energético dos experimento de Streaming com 0 a 40 encaminhadores.	86
B.1	Transmissão com 4 UEs no experimento Multicast	114
B.2	Transmissão com 10 UEs no experimento multicast.	115
B.3	Transmissão com 20 UEs no experimento multicast.	115
B.4	Transmissão com 30 UEs no experimento multicast.	116
B.5	Transmissão com 40 UEs no experimento multicast.	116
B.6	Transmissão dos nós Rx no experimento TCP com 4 D2D.	117
B.7	Transmissão dos nós Tx no experimento TCP com 4 D2D.	117
B.8	Transmissão dos nós Rx no experimento TCP com 10 D2D.	118
B.9	Transmissão dos nós Tx no experimento TCP com 10 D2D.	118
B.10	Transmissão dos nós Rx no experimento TCP com 20 D2D.	119
B.11	Transmissão dos nós Tx no experimento TCP com 20 D2D.	119
B.12	Transmissão dos nós Rx no experimento TCP com 30 D2D.	120
B.13	Transmissão dos nós Tx no experimento TCP com 30 D2D.	120
B.14	Transmissão dos nós Rx no experimento TCP com 40 D2D.	121

B.15 Transmissão dos nós Tx no experimento TCP com 40 D2D.	121
B.16 Transmissão dos nós Rx no experimento UDP VoIP 1 m/s com 4 D2D. . .	122
B.17 Transmissão dos nós Tx no experimento UDP VoIP 1 m/s com 4 D2D. . .	122
B.18 Transmissão dos nós Rx no experimento UDP VoIP 1 m/s com 10 D2D. . .	123
B.19 Transmissão dos nós Tx no experimento UDP VoIP 1 m/s com 10 D2D. . .	123
B.20 Transmissão dos nós Rx no experimento UDP VoIP 1 m/s com 20 D2D. . .	124
B.21 Transmissão dos nós Tx no experimento UDP VoIP 1 m/s com 20 D2D. . .	124
B.22 Transmissão dos nós Rx no experimento UDP VoIP 1 m/s com 30 D2D. . .	125
B.23 Transmissão dos nós Tx no experimento UDP VoIP 1 m/s com 30 D2D. . .	125
B.24 Transmissão dos nós Rx no experimento UDP VoIP 1 m/s com 40 D2D. . .	126
B.25 Transmissão dos nós Tx no experimento UDP VoIP 1 m/s com 40 D2D. . .	126
B.26 Transmissão dos nós Rx no experimento UDP VoIP 2 m/s com 4 D2D. . .	127
B.27 Transmissão dos nós Tx no experimento UDP VoIP 2 m/s com 4 D2D. . .	127
B.28 Transmissão dos nós Rx no experimento UDP VoIP 2 m/s com 10 D2D. . .	128
B.29 Transmissão dos nós Tx no experimento UDP VoIP 2 m/s com 10 D2D. . .	128
B.30 Transmissão dos nós Rx no experimento UDP VoIP 2 m/s com 20 D2D. . .	129
B.31 Transmissão dos nós Tx no experimento UDP VoIP 2 m/s com 20 D2D. . .	129
B.32 Transmissão dos nós Rx no experimento UDP VoIP 2 m/s com 30 D2D. . .	130
B.33 Transmissão dos nós Tx no experimento UDP VoIP 2 m/s com 30 D2D. . .	130
B.34 Transmissão dos nós Rx no experimento UDP VoIP 2 m/s com 40 D2D. . .	131
B.35 Transmissão dos nós Tx no experimento UDP VoIP 2 m/s com 40 D2D. . .	131
B.36 Transmissão dos nós Rx no experimento UDP VoIP 5 m/s com 4 D2D. . .	132
B.37 Transmissão dos nós Tx no experimento UDP VoIP 5 m/s com 4 D2D. . .	132
B.38 Transmissão dos nós Rx no experimento UDP VoIP 5 m/s com 10 D2D. . .	133
B.39 Transmissão dos nós Tx no experimento UDP VoIP 5 m/s com 10 D2D. . .	133
B.40 Transmissão dos nós Rx no experimento UDP VoIP 5 m/s com 20 D2D. . .	134

B.41	Transmissão dos nós Tx no experimento UDP VoIP 5 m/s com 20 D2D. . .	134
B.42	Transmissão dos nós Rx no experimento UDP VoIP 5 m/s com 30 D2D. . .	135
B.43	Transmissão dos nós Tx no experimento UDP VoIP 5 m/s com 30 D2D. . .	135
B.44	Transmissão dos nós Rx no experimento UDP VoIP 5 m/s com 40 D2D. . .	136
B.45	Transmissão dos nós Tx no experimento UDP VoIP 5 m/s com 40 D2D. . .	136
B.46	Transmissão dos nós Rx no experimento UDP streaming 256B com 4 D2D.	137
B.47	Transmissão dos nós Tx no experimento UDP streaming 256B com 4 D2D.	137
B.48	Transmissão dos nós Rx no experimento UDP streaming 256B com 10 D2D.	138
B.49	Transmissão dos nós Tx no experimento UDP streaming 256B com 10 D2D.	138
B.50	Transmissão dos nós Rx no experimento UDP streaming 256B com 20 D2D.	139
B.51	Transmissão dos nós Tx no experimento UDP streaming 256B com 20 D2D.	139
B.52	Transmissão dos nós Rx no experimento UDP streaming 256B com 30 D2D.	140
B.53	Transmissão dos nós Tx no experimento UDP streaming 256B com 30 D2D.	140
B.54	Transmissão dos nós Rx no experimento UDP streaming 256B com 40 D2D.	141
B.55	Transmissão dos nós Tx no experimento UDP streaming 256B com 40 D2D.	141
B.56	Transmissão dos nós Rx no experimento UDP streaming 512B com 4 D2D.	142
B.57	Transmissão dos nós Tx no experimento UDP streaming 512B com 4 D2D.	142
B.58	Transmissão dos nós Rx no experimento UDP streaming 512B com 10 D2D.	143
B.59	Transmissão dos nós Tx no experimento UDP streaming 512B com 10 D2D.	143
B.60	Transmissão dos nós Rx no experimento UDP streaming 512B com 20 D2D.	144
B.61	Transmissão dos nós Tx no experimento UDP streaming 512B com 20 D2D.	144
B.62	Transmissão dos nós Rx no experimento UDP streaming 512B com 30 D2D.	145
B.63	Transmissão dos nós Tx no experimento UDP streaming 512B com 30 D2D.	145
B.64	Transmissão dos nós Rx no experimento UDP streaming 512B com 40 D2D.	146
B.65	Transmissão dos nós Tx no experimento UDP streaming 512B com 40 D2D.	146
B.66	Transmissão dos nós Rx no experimento UDP streaming 1024B com 4 D2D.	147

B.67	Transmissão dos nós Tx no experimento UDP streaming 1024B com 4 D2D.	147
B.68	Transmissão dos nós Rx no experimento UDP streaming 1024B com 10 D2D.	148
B.69	Transmissão dos nós Tx no experimento UDP streaming 1024B com 10 D2D.	148
B.70	Transmissão dos nós Rx no experimento UDP streaming 1024B com 20 D2D.	149
B.71	Transmissão dos nós Tx no experimento UDP streaming 1024B com 20 D2D.	149
B.72	Transmissão dos nós Rx no experimento UDP streaming 1024B com 30 D2D.	150
B.73	Transmissão dos nós Tx no experimento UDP streaming 1024B com 30 D2D.	150
B.74	Transmissão dos nós Rx no experimento UDP streaming 1024B com 40 D2D.	151
B.75	Transmissão dos nós Tx no experimento UDP streaming 1024B com 40 D2D.	151
C.1	Consumo energético dos UEs Rx no experimento streaming 256 B.	152
C.2	Consumo energético dos UEs Tx no experimento streaming 256 B.	153
C.3	Consumo energético dos UEs Tx no experimento streaming 256 B com 4 encaminhadores.	153
C.4	Consumo energético dos UEs Tx no experimento streaming 256 B com 10 encaminhadores.	154
C.5	Consumo energético dos UEs Tx no experimento streaming 256 B com 20 encaminhadores.	154
C.6	Consumo energético dos UEs Tx no experimento streaming 256 B com 30 encaminhadores.	155
C.7	Consumo energético dos UEs Tx no experimento streaming 256 B com 40 encaminhadores.	155
C.8	Consumo energético dos UEs Rx no experimento streaming 256 B com 4 encaminhadores.	156
C.9	Consumo energético dos UEs Rx no experimento streaming 256 B com 10 encaminhadores.	156
C.10	Consumo energético dos UEs Rx no experimento streaming 256 B com 20 encaminhadores.	157
C.11	Consumo energético dos UEs Rx no experimento streaming 256 B com 30 encaminhadores.	157

C.12 Consumo energético dos UEs Rx no experimento streaming 256 B com 40 encaminhadores.	158
C.13 Consumo energético total da rede e dos UEs Rx e Tx no experimento streaming 256 B para cada cenário.	158
C.14 Consumo energético dos UEs Rx no experimento streaming 256 B.	159
C.15 Consumo energético dos UEs Tx no experimento streaming 256 B.	159
C.16 Consumo energético dos UEs Tx no experimento streaming 256 B com 4 encaminhadores.	160
C.17 Consumo energético dos UEs Tx no experimento streaming 256 B com 10 encaminhadores.	160
C.18 Consumo energético dos UEs Tx no experimento streaming 256 B com 210 encaminhadores.	161
C.19 Consumo energético dos UEs Tx no experimento streaming 256 B com 30 encaminhadores.	161
C.20 Consumo energético dos UEs Tx no experimento streaming 256 B com 40 encaminhadores.	162
C.21 Consumo energético dos UEs Rx no experimento streaming 256 B com 4 encaminhadores.	162
C.22 Consumo energético dos UEs Rx no experimento streaming 256 B com 10 encaminhadores.	163
C.23 Consumo energético dos UEs Rx no experimento streaming 256 B com 210 encaminhadores.	163
C.24 Consumo energético dos UEs Rx no experimento streaming 256 B com 30 encaminhadores.	164
C.25 Consumo energético dos UEs Rx no experimento streaming 256 B com 40 encaminhadores.	164
C.26 Consumo Energético Total da Rede e dos UEs Rx e Tx no Experimento Streaming 256 B para cada Cenário.	165
C.27 Consumo energético dos UEs Rx no experimento streaming 256 B.	166
C.28 Consumo energético dos UEs Tx no experimento streaming 256 B.	166

C.29 Consumo energético dos UEs Tx no experimento streaming 256 B com 4 encaminhadores.	167
C.30 Consumo energético dos UEs Tx no experimento streaming 256 B com 10 encaminhadores.	167
C.31 Consumo energético dos UEs Tx no experimento streaming 256 B com 210 encaminhadores.	168
C.32 Consumo energético dos UEs Tx no experimento streaming 256 B com 30 encaminhadores.	168
C.33 Consumo energético dos UEs Tx no experimento streaming 256 B com 40 encaminhadores.	169
C.34 Consumo energético dos UEs Rx no experimento streaming 256 B com 4 encaminhadores.	169
C.35 Consumo energético dos UEs Rx no experimento streaming 256 B com 10 encaminhadores.	170
C.36 Consumo energético dos UEs Rx no experimento streaming 256 B com 210 encaminhadores.	170
C.37 Consumo energético dos UEs Rx no experimento streaming 256 B com 30 encaminhadores.	171
C.38 Consumo energético dos UEs Rx no experimento streaming 256 B com 40 encaminhadores.	171
C.39 Consumo energético total da rede e dos UEs Rx e Tx no experimento streaming 256 B para cada cenário.	172

Lista de Tabelas

1.1	Crescimento histórico do tráfego global da Internet em duas décadas. Adaptado de [16].	3
5.1	Parâmetros de simulação.	46
5.2	Quantidades de experimentos.	47
5.3	Descrição dos experimentos sem comunicação D2D.	49
5.4	Descrição dos experimentos com comunicação D2D.	50
5.5	Consumo energético dos cenário do experimento <i>multicast</i>	51

Lista de Abreviaturas e Siglas

2G	: <i>2nd Generation</i> - 2 ^a Geração de Internet Móvel;
3G	: <i>3rd Generation</i> - 3 ^a Geração de Internet Móvel;
3GPP	: <i>3rd Generation Partnership Project</i> ;
4G	: <i>4th Generation</i> - 4 ^a Geração de Internet Móvel;
5G	: <i>5th Generation</i> - 5 ^a Geração de Internet Móvel;
BS	: <i>Base Station</i> - Estação Base;
CDN	: <i>Content Distribution Network</i> ;
CQI	: <i>Channel Quality Indicator</i> ;
CRN	: <i>Cognitive Radio Network</i> ;
C-RAN	: <i>Cloud-based Radio Access Networks</i> ;
D2D	: <i>Device-to-Device</i> ;
DL	: <i>Downlink</i> ;
eNodeB ou eNB	: <i>Evolved Node B</i> ;
FDD	: <i>Frequency Division Duplex</i> ;
IEEE	: <i>Institute of Electrical and Electronics Engineers</i> ;
IoT	: <i>Internet of Things</i> ;
IP	: <i>Internet Protocol</i> ;
IPv4	: <i>Internet Protocol version 4</i> ;
LAN	: <i>Local Area Network</i> ;
LTE	: <i>Long Term Evolution</i> ;
LTE-A	: <i>Long Term Evolution-Advanced</i> ;
MAC	: <i>Medium Access Control</i> ;
M2M	: <i>Machine-to-Machine</i> ;
MCC	: <i>Mobile Cloud Computing</i> (Computação Móvel em Nuvem);
MEC	: <i>Mobile Edge Computing</i> (Computação Móvel de Borda);
MIMO	: <i>Multiple Input, Multiple Output</i> ;
mMIMO	: <i>Massive MIMO</i> ;
mmWave	: <i>Milimeter Wave</i> ;
NAS	: <i>Non-Access Stratum</i> ;

NED	: <i>Network Description;</i>
NFV	: <i>Network Function Virtualization;</i>
NIC	: <i>Network Interface Card;</i>
OFDMA	: <i>Orthogonal Frequency-Division Multiple Access;</i>
PDCP	: <i>Packet Data Convergence Protocol;</i>
PGW	: <i>Packet Data Network Gateway;</i>
QoS	: <i>Quality of Service;</i>
RB	: <i>Resource Block;</i>
Rx	: <i>Receiver;</i>
RLC	: <i>Radio Link Control;</i>
RRC	: <i>Radio Resource Control;</i>
SDN	: <i>Software Defined Network;</i>
SL	: <i>Sidelink;</i>
SINR	: <i>Signal-to-Interference-plus-Noise Ratio;</i>
TDD	: <i>Time Division Duplex;</i>
Tx	: <i>Transmitter;</i>
UE	: <i>User Equipment;</i>
UL	: <i>Uplink;</i>
V2X	: <i>Vehicles-to-things;</i>
VNI	: <i>Visual Networking Index;</i>
VoIP	: <i>Voice over IP;</i>

Sumário

1	Introdução	1
1.1	O Aumento na Demanda por Tráfego nas Redes	1
1.2	A Solução 5G	4
1.2.1	Comunicação Dispositivo-a-Dispositivo - D2D	5
1.3	O Consumo de Energia nos Dispositivos Móveis	6
2	Redes Móveis	8
2.1	A Quarta Geração	8
2.2	<i>Releases</i> transitórios da 3GPP	11
2.3	5G	13
2.4	Tecnologias para Redes Celulares 5G	14
2.4.1	Células em Duas Camadas (<i>Two-Tier</i>)	15
2.4.2	Comunicação Dispositivo-a-Dispositivo (D2D)	16
2.4.3	Comunicação Baseada em Redes de Rádios Cognitivos	17
2.4.4	Estação Base em Nuvem	18
2.4.5	MIMO Maciço (mMIMO)	19
2.4.6	Transmissão em Ondas Milimétricas (mmWave)	19
2.4.7	Redes Definidas por Software e Virtualização de Funções de Rede	19
2.4.8	Armazenamento na Rede (<i>caching</i>)	20
3	Estado-da-Arte	21
3.1	Comunicação Dispositivo-a-Dispositivo	21

3.1.1	Descarregamento de Tráfego	23
3.1.2	Consumo de Energia	24
3.2	Simuladores	26
3.2.1	Modelo LTE D2D para ns-3	26
3.2.2	Um Módulo do ns-3 para Consumo de Energia em dispositivos LTE	26
3.2.3	OMNeT++ com SimuLTE	27
4	Desenvolvimento	30
4.1	Dificuldades Encontradas	30
4.1.1	Módulo de Energia do INET	30
4.1.2	<i>Port</i> do MiXiM para o SimuLTE	34
4.1.3	Junção de Projetos do ns-3	34
4.1.4	Adaptação de Módulo de Energia do INET à Camada Física do SimuLTE	35
4.1.5	Criação de Módulo de Energia no SimuLTE	36
4.2	Análise do Consumo Energético	36
5	Simulações e Resultados	39
5.1	Simulações	39
5.1.0.1	Configuração da Rede	43
5.1.1	Configuração das simulações	46
5.2	Resultados	48
5.2.1	Aplicação <i>Multicast</i>	50
5.2.2	Experimento: TCP	52
5.2.3	Experimento: UDP VoIP 2 m/s	60
5.2.4	Experimento: UDP VoIP 1 m/s	69
5.2.5	Experimento: UDP VoIP 5 m/s	76
5.2.6	Comparação entre os experimentos UDP com diferente mobilidade .	83

5.2.7	Comparação entre os experimentos TCP e UDP 2 m/s	84
5.2.8	Comparação entre os experimentos UDP de <i>Streaming</i> de Vídeo . . .	85
6	Conclusão	87
	Referências	90
	Apêndice A – Arquivos de Configuração do SimuLTE	95
	Arquitetura do Binder	95
	Arquivo de Descrição de Rede - NED	95
	Arquivo de Codificação das Funcionalidades - C++	95
	Arquivo de Biblioteca (.h) para as Funcionalidades	105
	Arquitetura da Estação Base - eNodeB	109
	Arquivo de Descrição de Rede da eNodeB- NED	109
	Arquitetura de Encaminhamento	112
	Arquivo de Descrição de Rede D2D - NED	112
	Apêndice B – Gráficos dos Resultados de Transmissão de Dados em cada Cenário	114
	B.1 experimento: Multicast	114
	B.2 experimento: TCP	117
	B.3 experimento: UDP VoIP 1 m/s	122
	B.4 experimento: UDP VoIP 2 m/s	127
	B.5 experimento: UDP VoIP 5 m/s	132
	B.6 experimento: UDP streaming 256B	137
	B.7 experimento: UDP streaming 512B	142
	B.8 experimento: UDP streaming 1024B	147
	Apêndice C – Gráficos do Consumo Energético dos Experimentos de Streaming de Vídeo	152

C.1	Gráficos de Consumo Energético do Experimento: UDP <i>Streaming</i> 256 B	. 152
C.1.1	Consumo Energético do Caso Base do Experimento <i>Streaming</i> 256 B	152
C.1.2	Consumo Energético dos UEs Tx no Experimento <i>Streaming</i> 256 B	153
C.1.3	Consumo Energético dos UEs Rx no Experimento <i>Streaming</i> 256 B	156
C.1.4	Consumo Energético Total da Rede no Experimento <i>Streaming</i> 256 B	158
C.2	Gráficos de Consumo Energético do Experimento: UDP <i>Streaming</i> 256 B	. 159
C.2.1	Consumo Energético do Caso Base do Experimento <i>Streaming</i> 256 B	159
C.2.2	Consumo Energético dos UEs Tx no Experimento <i>Streaming</i> 256 B	160
C.2.3	Consumo Energético dos UEs Rx no Experimento <i>Streaming</i> 256 B	162
C.2.4	Consumo Energético Total da Rede no Experimento <i>Streaming</i> 256 B	165
C.3	Gráficos de Consumo Energético do Experimento: UDP <i>Streaming</i> 256 B	. 166
C.3.1	Consumo Energético do Caso Base do Experimento <i>Streaming</i> 256 B	166
C.3.2	Consumo Energético dos UEs Tx no Experimento <i>Streaming</i> 256 B	167
C.3.3	Consumo Energético dos UEs Rx no Experimento <i>Streaming</i> 256 B	169
C.3.4	Consumo Energético Total da Rede no Experimento <i>Streaming</i> 256 B	172

Capítulo 1

Introdução

Estima-se que a demanda dos usuários por tráfego exceda a capacidade das redes 4G em um futuro próximo [30]. As redes 4G foram propostas com o objetivo de aumentar as taxas de transmissão em relação a 3G, prover interoperabilidade entre os diferentes sistemas de redes e permitir que um conjunto específico de serviços possa ser utilizado independentemente do sistema e contexto que o usuário esteja. Para atender a essas demandas, novas tecnologias foram introduzidas nas redes 4G, principalmente para aumentar as taxas de transmissão e solucionar o problema da comunicação entre diferentes tipos de sistemas e tecnologias.

Entretanto, ainda há um aumento dramático no número de usuários com interesse em utilizar redes móveis. A capacidade de processamento de *smartphones* e *notebooks* está cada vez maior e tais dispositivos cada vez mais populares, assim como os requisitos das aplicações de interesse dos usuários desses dispositivos. Assim, o crescimento de dispositivos que se conectam a redes móveis se tornará superior à capacidade da tecnologia atual, a 4G, de suportá-los.

1.1 O Aumento na Demanda por Tráfego nas Redes

A Cisco, baseada em seu projeto *Cisco Visual Networking Index* (VNI) [16] estima que o tráfego IP global irá triplicar de 122 EB (Exabytes)/mês em 2017 para 396 EB/mês em 2022. Este crescimento já representa um aumento em relação às expectativas de crescimento estimadas pelo projeto nos últimos anos. O aumento tanto do fluxo de dados como um todo, quanto das expectativas de crescimento é atribuído a expansão do tráfego móvel [16]. Em 2017 diversos países, como Japão, Coreia do Sul, Canadá, Alemanha e Suécia tiveram uma taxa de crescimento maior em suas redes de telefonia móvel do que

nas redes fixas.

Na Figura 1.1 podemos verificar o crescimento de tráfego IP mensal previsto pela Cisco até 2022. Já na Figura 1.2 podemos ver a relação de crescimento de Internet para dispositivos móveis (*Mobile Growth*) e dispositivos fixos (*Fixed Growth*) no período de 2017.

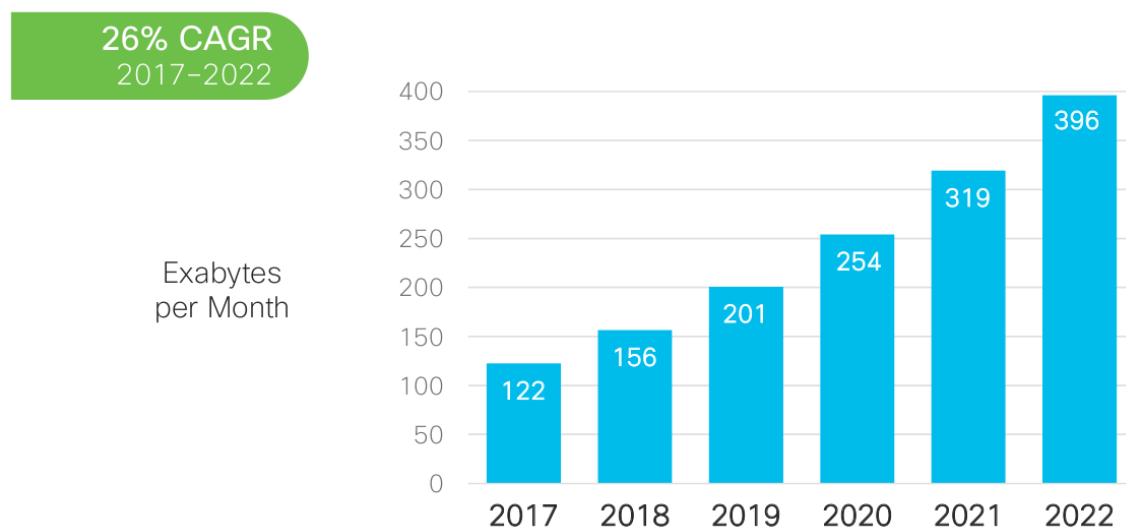


Figura 1.1: Previsão de tráfego até 2022. Fonte: Cisco VNI Forecast 2017–2022 [16].

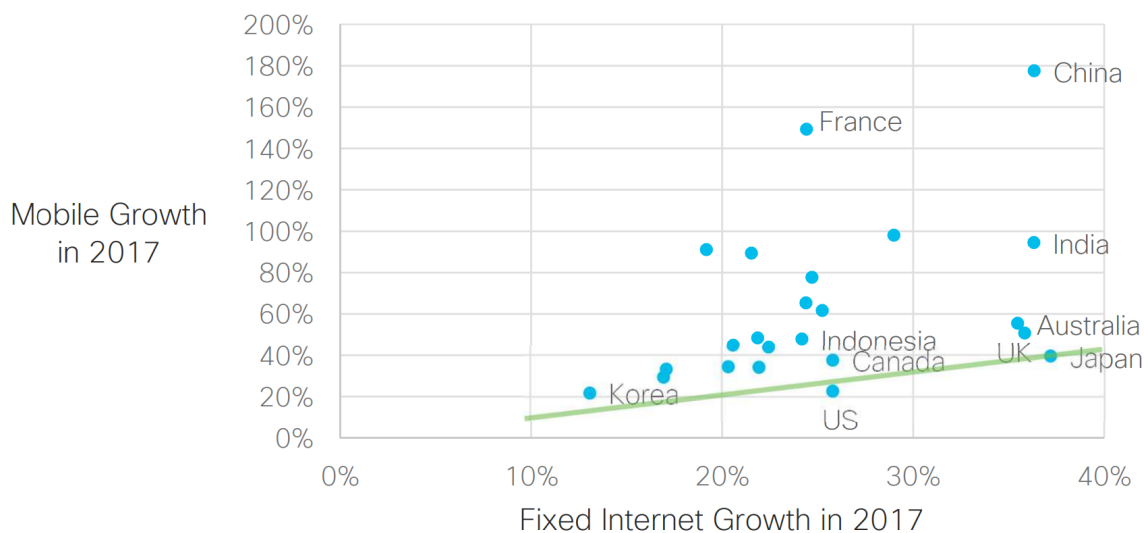


Figura 1.2: Taxas de crescimento de tráfego de Internet móvel e fixa em 2017. Fonte: Cisco VNI Forecast 2017–2022 [16].

Historicamente, este crescimento não é novidade. Nas últimas duas décadas o crescimento de tráfego global da Internet foi marcante, partindo de 100 GB por dia em 1992

com previsão de atingir 150.7 TB por segundo em 2022, como é possível verificar na Tabela 1.1.

Tabela 1.1: Crescimento histórico do tráfego global da Internet em duas décadas. Adaptado de [16].

Ano	Tráfego Global da Internet
1992	100 GB por dia
1997	100 GB por hora
2002	100 GB por segundo
2007	2.000 GB por segundo
2017	46.600 GB por segundo
2022	150.700 GB por segundo

Nesta curva de crescimento se destaca a análise do tipo de dispositivos e também a de tipo de comunicação. No primeiro caso, quando se analisa o tráfego IP global por dispositivos, a categoria de dispositivos que mais cresceu foi a de telefones móveis (*smartphones*). Como podemos ver na Figura 1.3, a previsão é que, seguindo esta tendência, em 2022 mais de 50% do tráfego será proveniente somente de *smartphones* e *tablets*.

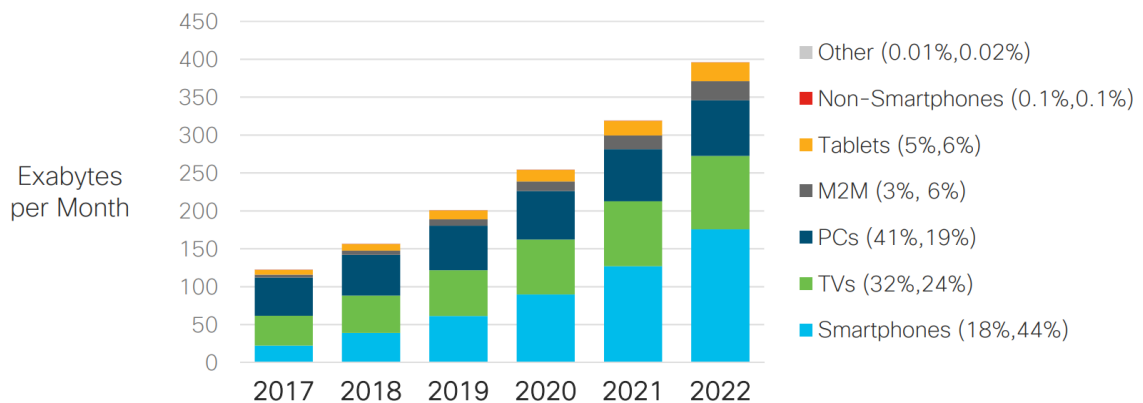


Figura 1.3: Previsão de taxa de crescimento de tráfego por tipos de dispositivos entre 2017 a 2022. Fonte: Cisco VNI Forecast 2017–2022 [16].

Em segundo caso, ao analisar o tipo de comunicação, o destaque fica para o M2M (*Machine-to-Machine*, no português, Máquina-a-Máquina). Este tipo de comunicação se assemelha a comunicação dispositivo-a-dispositivo (D2D - *Device-to-Device*), por permitir que máquinas se comuniquem diretamente entre si, e é utilizado pelas indústrias para

acelerar o crescimento global da Internet das Coisas (IoT - *Internet of Things*). Na Figura 1.4 visualizamos uma expectativa de crescimento de 6,1 bilhões de conexões em 2017 para 14,6 bilhões em 2022. Um número correspondente a 1,8 conexões M2M para cada habitante é esperado para o mundo em 2022.

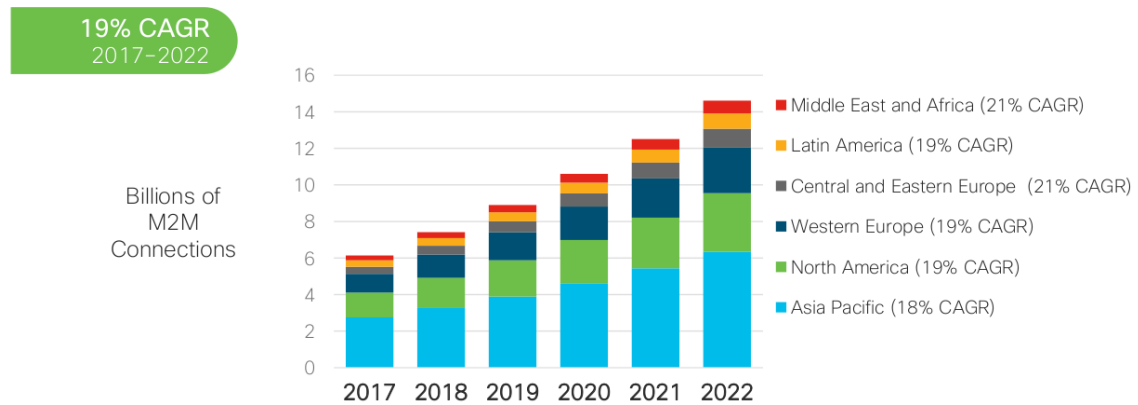


Figura 1.4: Previsão de taxa de crescimento de conexões M2M entre 2017 a 2022. Fonte: Cisco VNI Forecast 2017–2022 [16].

1.2 A Solução 5G

Para conseguir lidar com esse crescimento massivo em quantidade de dispositivos, em tráfego de dados e em quantidade de informações produzidas, necessitamos novas soluções tanto para redes móveis quanto para redes fixas. Nas redes móveis, percebe-se que a tecnologia 4G já não é suficiente para atender aos requisitos dos usuários móveis. Estima-se, por exemplo, que um veículo autônomo precise enviar dados a uma taxa de 1 GB/s com latência mínima, próxima de zero; enquanto que em um contexto de *eHealth* [38], um hospital inteligente e uma fábrica conectada geram, respectivamente, 4 TB/dia e 1 PB/dia [21]. A 4G tecnicamente pode atingir velocidades de até 100Mbps, no entanto, na realidade as conexões 4G têm velocidades inferiores a esta, com alta latência e não consideram a eficiência energética. Mesmo se este padrão atingisse sua capacidade técnica de 100Mbps, ainda seria pouco, considerando-se os tipos de transmissão visados atualmente, como os exemplos citados previamente. Ainda, outra limitação está na infraestrutura das redes 4G. Como ela é projetada hoje, para melhorar a conectividade em áreas urbanas as operadoras precisam instalar novos equipamentos (antenas). Além do custo adicional de implantação, com a adição de novas torres, novos enlaces e novos pontos de encaminhamento, isto pode gerar uma latência ainda maior. Isto gera a necessidade por uma quinta geração (5G) que traga as soluções necessárias.

Pelo menos desde 2016, a 3GPP (*3rd Generation Partnership Project*) tem concentrado esforços para guiar o mercado em busca de tais soluções para a próxima geração de telefonia móvel, a 5G [2]. Ainda não há um consenso sobre a arquitetura das redes 5G. Porém, para que a 5G resolva necessidades eminentes, como ser uma solução de comunicação viável para Internet das Coisas e Redes Veiculares [18] [59], além de atender ao crescimento de número de usuários e da banda demandada por esses usuários, é desejável que a taxa de transmissão seja até 1000 vezes maior que na 4G [23] e que a latência seja da ordem de poucos milissegundos, comumente chamada de latência zero [43]. Com uma rede com essas características, é possível desenvolver aplicações, como a comunicação entre veículos e a Internet das Coisas, para interconectar bilhões de dispositivos [18] [59]. O aumento da taxa de transmissão e redução da latência se darão em virtude do uso de diferentes tecnologias, que vão desde a camada física até a camada de rede. Entre essas tecnologias, estão o desenvolvimento de novos padrões LTE (*Long Term Evolution*) e IEEE 802.11, o uso de pequenas células (*small cells*), transmissão em ondas milimétricas (mmWave), MIMO massivo, compartilhamento de espectro licenciado e não licenciado, entre outras [12] [23] [43] [60]. A 5G vem sendo colocada em pauta nas discussões e planejamentos da 3GPP desde o *Release* 12 [2], e o *Release* 16 [7], o qual sua agenda finaliza em 2020, será o final, para lançamento desta tecnologia com todas suas especificações.

1.2.1 Comunicação Dispositivo-a-Dispositivo - D2D

Outra tecnologia presente na maioria das propostas de rede para a 5G, é a comunicação Dispositivo-a-Dispositivo (*Device-to-Device* - D2D) [22]. Com a comunicação D2D, dispositivos são capazes de se comunicar diretamente sem a necessidade do uso da estação base. Assim, podem experimentar maior vazão e menor latência em virtude da proximidade dos dispositivos, bem como, podem reduzir a carga de tráfego na estação base. Por outro lado, novos desafios são introduzidos, como o aumento da interferência [13], a segurança e privacidade da rede [45], a alocação de canais [12], o controle da comunicação D2D por parte da estação base [22], o consumo de energia [26], a tarifação [15], entre outros.

Quando se utiliza aplicações e serviços baseados em localização, explorar a comunicação entre dispositivos próximos possibilita ganhos na utilização de espectro, vazão e consumo de energia [31]. Estas formas de comunicação são classificadas em 4 diferentes, segundo Tehrani et al. [36] e se diferenciam por ter 1. dispositivos encaminhando com controle da operadora; 2. comunicação D2D direta com controle do operador; 3. dispo-

sitivos encaminhando com controle pelos dispositivos; 4. comunicação D2D direta com controle pelos dispositivos. As classificações de arquitetura para 5G serão discutidas com mais detalhes na seção 2.4.

Neste contexto de encaminhamento com controle da operadora, este trabalho tem por objetivo avaliar o impacto da comunicação D2D no consumo de energia dos nós e da rede LTE. Nos experimentos aqui realizados e descritos seus resultados no Capítulo 5 podemos perceber situações em que a comunicação D2D aumenta o consumo energético, mas também foram identificados cenários e configurações de rede que possibilitaram redução do consumo energético mesmo sob comunicação D2D. Assim, percebemos que uma boa escolha de encaminhadores na comunicação de dispositivo a dispositivo é uma das possibilidades de se economizar bateria dos dispositivos envolvidos no processo de comunicação e estender seu tempo de vida com a mesma carga [32].

1.3 O Consumo de Energia nos Dispositivos Móveis

Em função das baterias dos dispositivos móveis que são limitadas, é essencial o desenvolvimento de técnicas que possam gerar economia de energia. Assim, a modelagem do consumo de energia tem se tornado cada vez mais importante, com o crescimento do número de dispositivos móveis e embarcados, como dispositivos pessoais de monitoramento médico, dispositivos de monitoramento ambiental, veículos elétricos, painéis solares, sensores *wireless* de baixa capacidade de energia, etc. [40].

Simulações quanto ao consumo de energia permitirão o desenvolvimento de protocolos de roteamento sensíveis a consumo energético, que, por consequência, resultarão em dispositivos mais eficientes energeticamente.

Neste sentido, este trabalho realizou simulações de comunicação D2D no simulador de redes móveis SimuLTE [57]. Foram definidos 8 experimentos com diferentes tipos de protocolos, aplicações e mobilidade entre eles. Para cada experimento, fizemos 6 variações no cenário, referentes a quantidade de nós encaminhadores, variando na forma de 0, 10, 25, 50, 75 e 100% de encaminhadores. Para cada uma das variações, dentro de cada cenário, executamos 10 rodadas. Consideramos uma rede com 1 célula, com a comunicação D2D gerenciada pela operadora (estação base) e os dispositivos foram distribuídos nesta de forma estacionária e aleatória. Dos resultados obtidos nas simulações analisamos a quantidade de mensagens enviadas e recebidas por nó e por cada configuração de rede. Estes resultados possibilitam compreender o consumo energético neste tipo de rede, com e sem

encaminhamento. Identificamos alguns cenários em que a inserção dos encaminhadores aumenta o consumo energético da rede e, outros, em que inclusive se consegue reduzir o consumo energético, dependendo da combinação de quantidade de encaminhadores e protocolos utilizados. Estes resultados em que se consome menos energia, em especial, rompem com o empirismo que leva a pensar que a encaminhamento consumirá sempre mais energia e se postula como método de redução de consumo energético para novas pesquisas em redes móveis. Os resultados que demonstram maior consumo energético já são úteis para analisar, principalmente, o perfil de consumo energético da rede conforme a quantidade de encaminhadores inseridos na mesma.

Além desta introdução, o restante deste texto está estruturado da seguinte forma. O Capítulo 2 traz uma contextualização histórica sobre Redes Móveis, citando os *releases* da 3GPP referentes ao LTE e a 5G, seguido das propostas de arquitetura e tecnologias para compor a 5G. Desta forma, no Capítulo 3 será possível compreender o estado-da-arte quanto às redes móveis com encaminhamento. Nele é explicado sobre a comunicação dispositivo-a-dispositivo e suas classificações, a importância da escolha do encaminhador ideal, como esta comunicação pode ser eficaz no descarregamento da rede e as considerações de energia referentes à mesma. Ainda, é tratado sobre os simuladores de rede LTE e suas limitações para este trabalho. No Capítulo 4 é discutido o desenvolvimento da pesquisa, apontando as dificuldades encontradas quanto aos simuladores e modelagem de energia, bem como a descrição da metodologia e ferramentas adotadas. As simulações e seus respectivos resultados obtidos são apresentados no Capítulo 5, demonstrando a análise de consumo energético dos experimentos, bem como algumas comparações dos resultados entre eles. Por fim, no Capítulo 6 encontram-se as conclusões, os apontamentos de pontos de pesquisa em aberto e expectativas de trabalhos futuros.

Capítulo 2

Redes Móveis

3GPP é a sigla para *3rd Generation Partnership Project*, que em português poderia ser entendido como Projeto de Padronização da 3ª Geração. É uma associação que foi criada com intuito de padronizar a 3ª geração de telefonia móvel, mas que se manteve como entidade que guia e padroniza as evoluções das redes móveis no cenário global. A 3GPP une sete organizações de desenvolvimento de padrões de telecomunicações: ARIB, ATIS, CCSA, ETSI, TSDSI, TTA e TTC; e provê aos seus membros um ambiente para produção de *Reports* e Especificações que definem as tecnologias 3GPP [8].

Desta forma, é a 3GPP que está à frente da padronização. Seus padrões são lançados em “pacotes”, chamados de *releases*. Devido a abrangência e complexidade propostas à 5G, sua implementação foi pensada em médio prazo e iniciou sua preparação já no *Release* 12 [2], ainda em 2015. Este capítulo apresenta a 4G, as padronizações lançadas pela 3GPP para promover a evolução da 4G para a 5G e, também, as definições que já existem e expectativas sobre a 5G.

2.1 A Quarta Geração

O padrão LTE (*Long Term Evolution*) foi a arquitetura proposta para a quarta geração de telefonia celular. Como motivações básicas para este padrão, a 3GPP cita [39]:

- a necessidade de garantir a competitividade e continuidade dos sistemas de redes móveis para o futuro;
- as demandas por parte dos usuários por taxas transmissão de dados mais altas e QoS (*Quality of Service* - Qualidade de Serviço);

- um sistema otimizado para a comutação de pacotes;
- a demanda contínua por redução de custos;
- a baixa complexidade;
- a busca para se evitar a fragmentação desnecessária de tecnologias.

Com a evolução do LTE após seu lançamento, vários *releases* de melhoria foram lançados e serão tratados na próxima seção. Se começou a pensar o LTE como uma evolução contínua que pudesse durar por um prazo maior que o intervalo entre as gerações anteriores e aguardar até a chegada de sua sucessora, a 5G, em 2020. Neste caminho para a sucessão, o papel da 4G era convergir para dar base e compatibilidade a tecnologias da quinta geração.

O LTE foi lançado então, como uma rede mais simplificada, baseada em Estações Base, aqui chamadas de *evolved NodeB* (*eNB*¹). Não há controladores centralizados e as eNBs podem ser interconectadas entre si e com o núcleo da rede [39]. As estações base se conectam por redes cabeadas e toda a rede é baseada em IP. A rede também é composta por UEs (*User Equipments* - Equipamentos de Usuários), que são todos os tipos de dispositivos móveis que os usuários eventualmente utilizam para se conectar à rede, como notebook, *smartphones*, *tablets*, etc. Estes dispositivos, por sua vez, se conectam apenas às estações, por meio de uma interface de rede móvel [13]. A Figura 2.1 demonstra como se dão estas conexões.

No LTE os pacotes de dados e de controle são transmitidos em canais diferentes. A diferenciação desta informação não está nos cabeçalhos, como em outros padrões. A arquitetura do protocolo do rádio LTE é separada em *control plane* e *user plane*, como se pode ver na Figura 2.2. No *user plane* a aplicação cria pacotes de dados que são processados por protocolos como TCP, UDP e IP. Já no *control plane*, o protocolo NAS (*Non-Access Stratum*) promove autenticação e registro dos dispositivos, enquanto o protocolo do controlador de recursos de rádio (*Radio Resource Control* - *RRC*) cria as mensagens de sinalização que são trocadas entre a estação base e o dispositivo móvel. Em ambas as situações a informação é processada pelo protocolo de convergência de pacotes de dados (*Packet Data Convergence Protocol* - *PDCP*), o protocolo de controle do canal de rádio (*Radio Link Control* - *RLC*) e o protocolo de controle de acesso ao meio (*Medium Access*

¹eNodeB, eNB ou Evolved Node B é um elemento do padrão LTE que corresponde ao elemento Node B do padrão UMTS ou à BS (Base Station) do GSM.

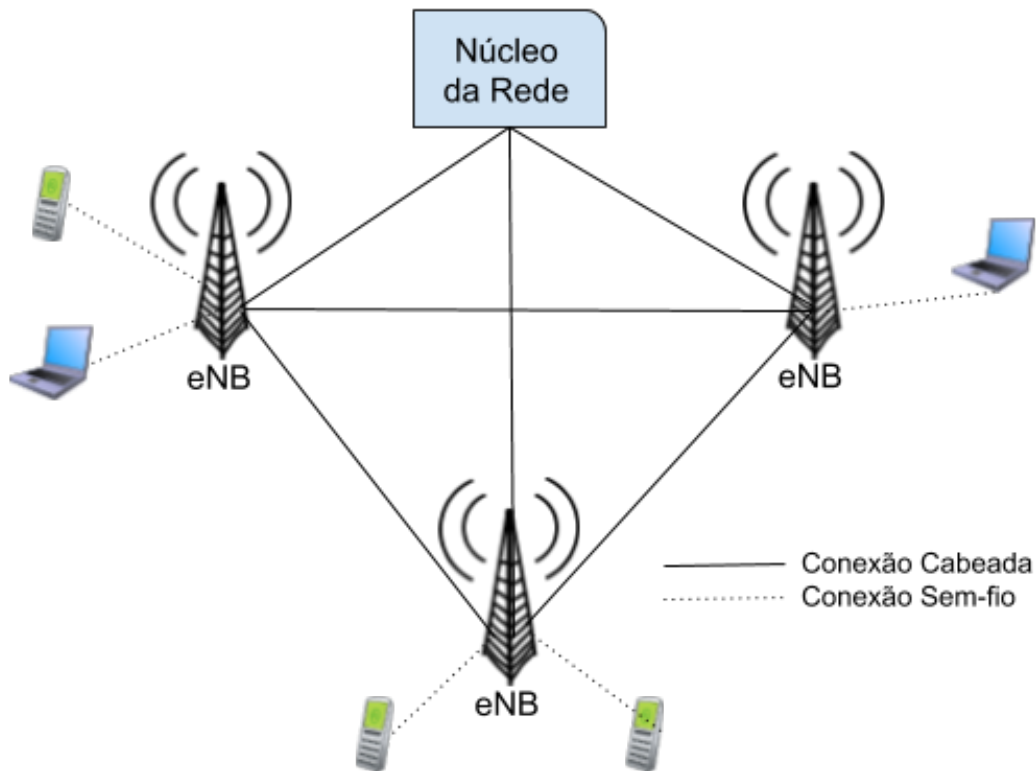


Figura 2.1: Esquema de conexão entre em redes 4G.

Control - MAC), antes de ser entregue à camada física para transmissão [53]. Na Figura 2.3 podemos ver uma demonstração das arquiteturas do *user* e *control plane*.

As redes LTE foram pensadas para conseguir comutar pacotes mesmo com a movimentação dos usuários, sem prejuízo a seus critérios de funcionamento. O LTE opera com as tecnologias FDD (*Frequency Division Duplex*), que transmite os pacotes de *Downlink* (DL) e *Uplink* (UL) ao mesmo tempo, em frequências diferentes, e TDD (*Time Division Duplex*), que transmite pacotes UL e DL na mesma frequência, dividindo o tempo. A Figura 2.4 ilustra este procedimento. Estas tecnologias foram lançadas no Release 12 do 3GPP, que será explicado posteriormente. [51].

A frequência utilizada pelo LTE tem seu espectro dividido em vários canais e estes canais são utilizados para, separadamente, transmitirem mensagens de dados ou mensagens de controle de rede. São transmitidos quadros (*frames*) de 10 ms, onde para cada um destes quadros há a divisão em outros 10 sub-quadros, cada de 1 ms. Estes sub-quadros são mais uma vez divididos em frações menores, chamadas de *slots*, que duram 0.5 ms.

São utilizadas técnicas de acesso ao meio diferentes, sendo que para *downlink* é utili-

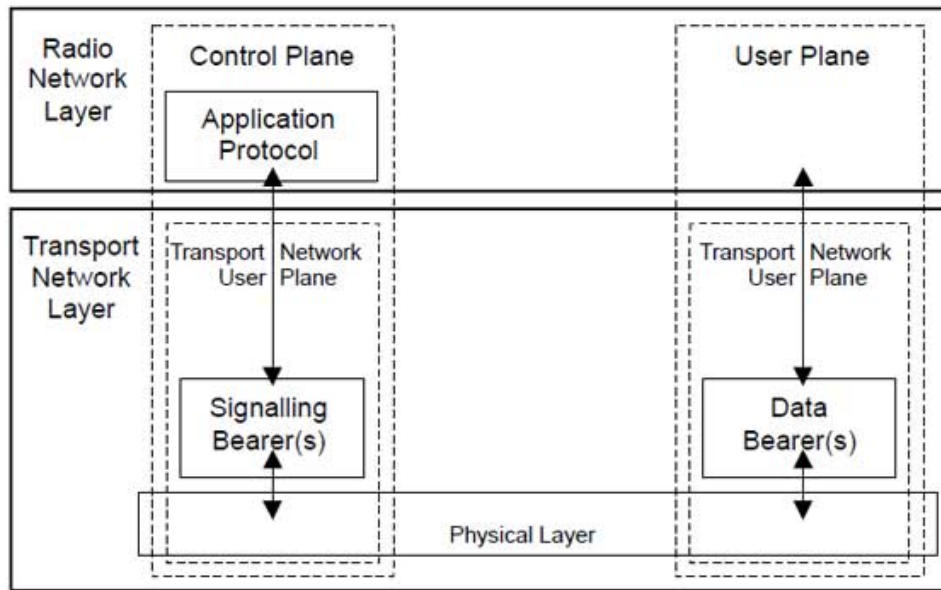


Figura 2.2: Arquitetura do protocolo de rádio LTE. Adaptado de [53].

zado o esquema de modulação digital OFDMA (*Orthogonal Frequency-Division Multiple Access*) e para *uplink* a SC-FDMA (*Single-Carrier Frequency-Division Multiple Access*). Todas as alocações de recursos são controladas pela eNB. Ela identifica todos os dispositivos que fazem parte de sua célula (aqueles que estão conectados à ela) e controla seus recursos para que possam enviar e receber dados.

O OFDMA foi uma das novas tecnologias introduzidas nos sistemas 4G para aumentar as taxas de transmissão. Outra é o uso de múltiplas antenas tanto pelo transmissor como pelo receptor para explorar o fenômeno de múltiplos caminhos na propagação do sinal (*Multiple-Input and Multiple-Output*, - MIMO). Para a solução da interoperabilidade, é proposto o uso da comutação de pacotes através do IP, o qual atua como uma tecnologia comum entre os ecossistemas no acesso de serviços como vídeo por IP, voz sobre IP (VoIP) e dados multimídia [23].

2.2 Releases transitórios da 3GPP

De início, se pensou em uma solução transitiva, que pudesse vigorar até o lançamento da 5G e atendesse as demandas do mundo móvel neste ínterim. Esta foi o LTE (*Long Term Evolution*), que teve suas especificações iniciais no *Release 8* e no *Release 12* [2], em 2015, começou-se a implantar melhorias com prioridade em preparar o padrão LTE compatível com potenciais tecnologias para a 5G.

Funcionalidades importantes que foram finalizadas e inclusas neste *release* foram:

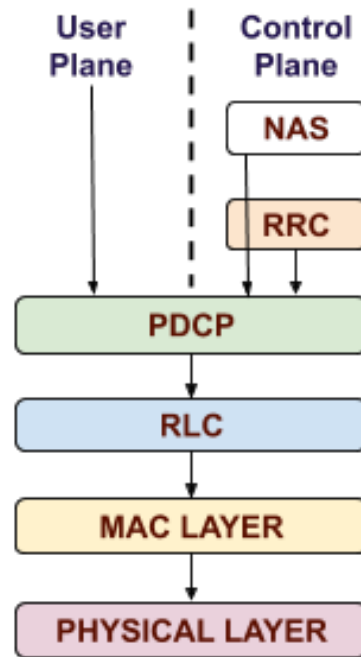


Figura 2.3: Arquitetura do *User Plane* e do *Data Plane*.

small cells e densificação de redes, D2D e operação conjunta TDD-FDD em LTE.

O *Release 13* [3] traz cerca de 170 funcionalidades e estudos, dentre elas, algumas essenciais para a evolução do LTE. Neste lançamento já constam também estudos, que se estenderam à 5G, de mecanismos de co-existência de operação do LTE em espectro não licenciado.

Na conferência de Phoenix (US), em 2015, que é parte deste *release*, iniciou-se o trabalho na próxima geração de tecnologia celular, com o propósito de submeter tecnologias candidatas para o projeto oficial de lançamento da 5G, o IMT-2020, a realizar-se em 2020.

Tecnologias com potencial de integrar a 5G foram integradas ao *Release 14* [5] para suporte ao LTE, o que facilitaria o serviço futuro na 5G. Uma destas tecnologias que merece destaque é o suporte a serviços de comunicação veículos-para-coisas (V2X - *vehicles-to-things*).

Outro ponto de desenvolvimento interessante deste *release* é o cronograma previsto para a padronização da 5G, com um resumo dos principais pontos de controle, expectativas de procedimentos para os grupos de trabalho da 3GPP e resumo das atividades a serem desenvolvidas. A Figura 2.5 mostra este cronograma.

O *Release 15* [6] focou seus trabalhos na primeira fase de entregas da 5G. As principais entregas aqui foram as especificações de interoperabilidade e compatibilidade.

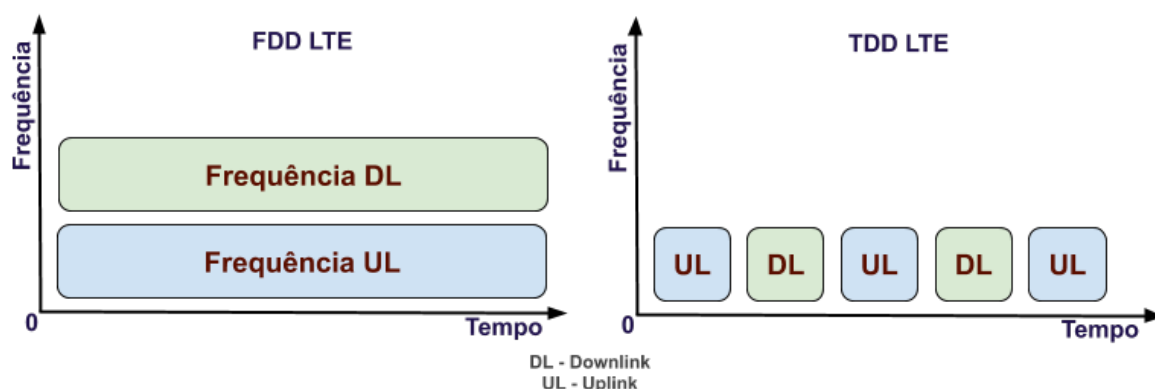


Figura 2.4: Demonstração das técnicas FDD e TDD.

Outros dois pontos importantes foram o ajuste do cronograma para a segunda etapa de especificações da 5G; e as instruções da 3GPP 5G para os grupos de avaliação.

A agenda do *Release* 15 finaliza em 2019.

O *Release* 16 [7] será a última para a 5G e sua agenda finaliza em 2020. Nela está composto o lançamento da Fase 2 da 5G, em dezembro de 2019.

Este lançamento será o marco inicial dos sistemas 5G completos. Neste *release* foram inclusos também estudos necessários aos requisitos de interoperabilidade, interconectividade e inteligência, da 5G, sendo eles de: segurança, *codes* e serviços de *streaming*, interoperabilidade de Redes de Área Local (LAN - *Local Area Network*), fatiamento de rede e Internet das Coisas (IoT).

Apesar de alguns sites de notícias utilizarem informações que dá-se a entender que a 5G já foi oficialmente lançada e está em plena operação, como nesta matéria do Olhar Digital [17], a realidade é outra. Algumas especificações do 5G já foram lançadas nos *releases* anteriores e são estas especificações que já estão em testes. A primeira fase da 5G foi finalizada, no entanto, o *Release* 16, o último para o lançamento oficial da 5G só finalizará em 2020.

2.3 5G

Hoje há um aumento dramático no número de usuários com interesse em utilizar os sistemas de banda larga móveis. A capacidade de processamento de *smartphones* e *notebooks* está cada vez maior e tais dispositivos cada vez mais populares, assim como os requisitos dos serviços multimídia utilizados. Desta forma, o LTE já não é suficiente

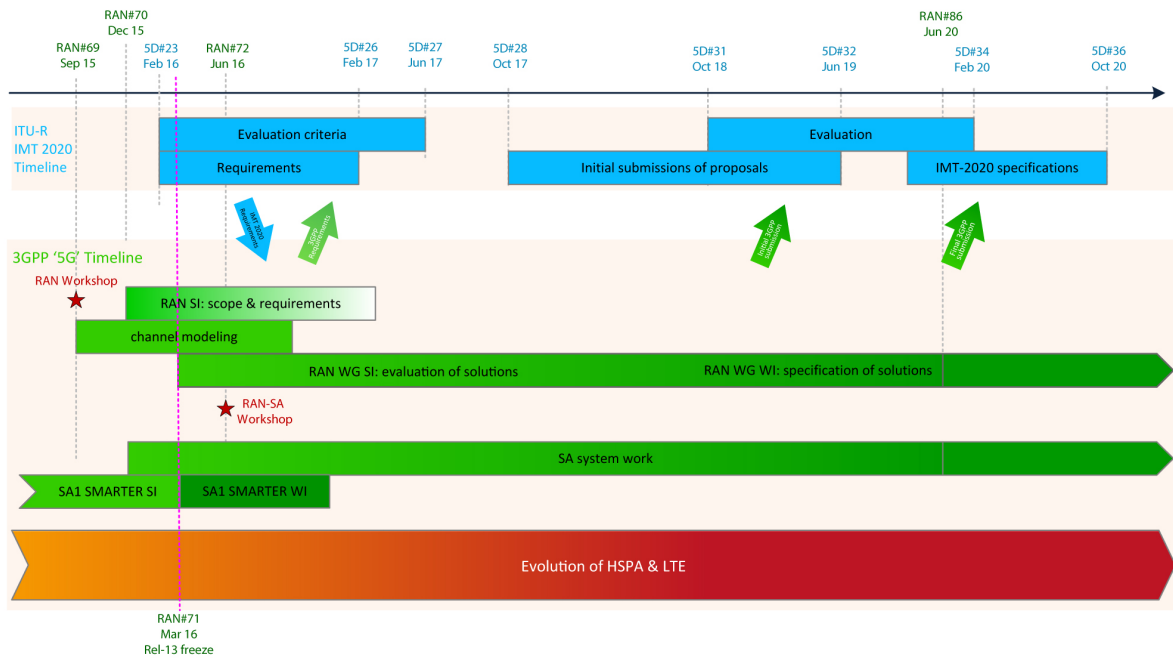


Figura 2.5: Cronograma previsto para padronização da 5G. Fonte: *Tentative 3GPP timeline for 5G* [4].

para atender aos requisitos dos usuários móveis, o que gera a necessidade por uma quinta geração.

As mudanças para a 5G são muitas em relação à 4G e a quantidade, abrangência e impacto das mudanças é superior às mudanças ocorridas nas gerações anteriores. Para que a 5G resolva necessidades eminentes, como ser uma solução de comunicação viável para Internet das Coisas e Redes Veiculares [18] [59], além de atender ao crescimento de número de usuários e da banda demandada por esses usuários, é preciso, mesmo com a expansão da rede, conseguir aumentar a taxa de transmissão da mesma e otimizar o consumo de energia. Exemplos de tecnologias que se tornam viáveis e serão possibilitadas pela 5G são os Veículos Conectados [29], a *e-Health* [38] e as casas inteligentes. Ambas situadas na ambiência da Internet das Coisas [47] e *smart grids*. A segurança e a privacidade da rede e dos dispositivos também são preocupações que merecem cuidado, principalmente nos cenários em que passa a existir comunicação direta (D2D) ou encaminhada entre dispositivos e máquinas.

2.4 Tecnologias para Redes Celulares 5G

Diversas revisões de literatura apontam que variadas tecnologias poderão compor as Redes Celulares 5G. Quanto ao esquema de comunicação nestas redes quatro tipos

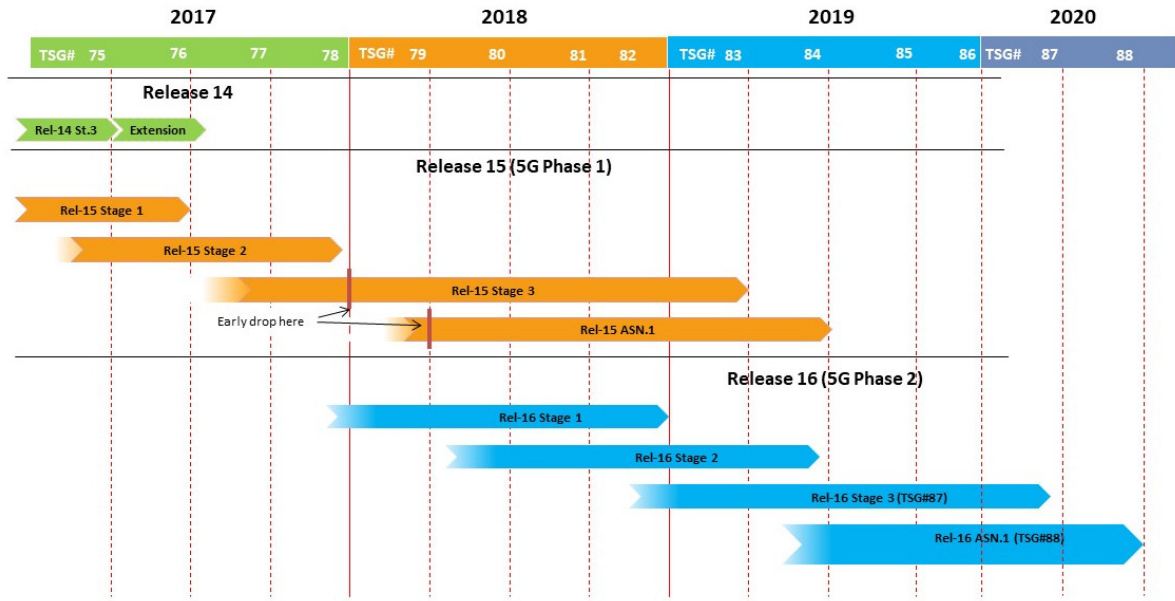


Figura 2.6: Fases da 5G. Fonte: 3GPP [8].

diferentes são os que mais se destacam, como nos surveys [60] [43] [23]: em duas camadas, baseada em comunicação D2D, baseada em rádios cognitivos e baseadas em nuvem.

2.4.1 Células em Duas Camadas (*Two-Tier*)

As células de redes móveis organizadas para se comunicar em duas camadas possuem a *Macrocell Tier* e a *Device Tier* [36]. A Figura 2.8 representa este esquema. A *Macrocell Tier* engloba desde cada dispositivo móvel isolado até aqueles que se comunicam com as estações base (BS - *Base Station*), e as próprias estações base. Na *Device Tier* estão todos os dispositivos, de forma que grupos de dispositivos classificados por listas, grupos de usuários, região geográfica, entre outros, formam células de comunicação dentro dessa camada. Para cada uma das células da *Device Tier* pode haver comunicação direta com a estação base, comunicação de múltiplos saltos ou apenas comunicação entre os dispositivos. Os últimos dois tipos de comunicação serão discutidos na seção seguinte, na qual é tratada a comunicação D2D.



Figura 2.7: Esquema de comunicação em duas camadas.

As vantagens desse modelo em duas camadas são inúmeras e incluem facilidade para distribuição espacial de carga de rede, identificação de usuários *indoor* e *outdoor*, facilidade para o processo de *handoff*, entre outros [11].

2.4.2 Comunicação Dispositivo-a-Dispositivo (D2D)

Alguns trabalhos definem a comunicação baseada em D2D e muitos outros citam tal tecnologia como parte de outras propostas para as redes 5G. Tehrani et al. [36] classificam as formas de comunicação D2D em quatro e propõem uma organização em duas camadas (citadas na seção anterior): uma para dispositivos e outra que compreende dispositivos e a estação base, sendo que ambas devem funcionar sobre a mesma banda, a faixa licenciada definida pela *macrocell*. São elas:

1. **DR-OC** (*Device Relaying with Operator Control*): dispositivo na camada *device tier* pode fazer comunicação diretamente com BS (*macrocell tier*) ou através de encaminhamento por outro dispositivo. O canal de controle é estabelecido pelo operador.
2. **DC-OC** (*Direct D2D communication with Operator Control*): comunicação acontece direta e exclusivamente entre dispositivos (*device tier*) e somente o canal de controle é estabelecido pelo operador (*macrocell tier*).

3. **DR-DC** (*Device Relaying with Device Control*): aqui a comunicação acontece apenas na *device tier*, pois os dispositivos encaminham dados através de múltiplos saltos da origem até o destino e também estabelecem o canal de controle.
4. **DC-DC** (*Direct D2D communication with Device Control*): comunicação acontece de forma direta entre os dispositivos de origem e destino, assim atendo-se também somente a *device tier* e com canal de controle estabelecido pelos dispositivos.

2.4.3 Comunicação Baseada em Redes de Rádios Cognitivos

As discussões sobre *Cognitive Radio Network* (CRN) em 5G tratam da utilização oportunista do espectro em uma rede de rádios cognitivos [9]. Há nós nessas redes, chamados *secondary users* (SUs), que proativamente monitoram o ambiente espectral em uma faixa fora da banda licenciada da rede celular e dinamicamente estabelecem enlaces sem fio ao perceber espaços não utilizados na faixa monitorada. As propostas baseadas em redes de rádios cognitivos podem ser categorizadas em dois tipos: não-cooperativo, onde a rede cognitiva cria uma rede sobreposta à rede celular licenciada e a estação base pode possuir ambas interfaces e atuar em ambas as redes; e cooperativo, no qual é possível estabelecer enlaces sem fio heterogêneos e combinar comunicações de pequena e longa distâncias até que o dado seja entregue ao seu destino final [25]. Os rádios cognitivos oferecem uma solução parcial para a escassez do espectro licenciado das redes celulares. Esta tecnologia possibilita reduzir o uso do espectro licenciado para comunicações de pequena e média distâncias e fornecer um espectro complementar a ser utilizado oportunisticamente em momentos de pico nas redes 5G.

No modo não-cooperativo há duas interfaces de rádio operando, uma na banda licenciada e outra na banda complementar à licenciada. Dessa forma, a rede cognitiva cria uma rede sobreposta à rede celular licenciada. Neste caso, a estação base pode possuir ambas interfaces e atuar em ambas as redes. A rede cognitiva, em virtude de restrições na potência de transmissão de seus dispositivos, se torna mais adequada para comunicações de pequenas e médias distâncias. Enquanto isso, a rede celular licenciada realiza as comunicações de longa distância.

No modo cooperativo tanto o espectro licenciado da rede celular como o espectro complementar são integrados em uma única camada física ao utilizar princípios de comunicação cooperativa. Ao separar a comunicação ponto-a-ponto em etapas é possível estabelecer enlaces sem fio heterogêneos e combinar comunicações de pequena e longa distâncias até que o dado seja entregue ao seu destino final. Ao alocar os canais de forma

inteligente esta proposta aproveita as características do espectro licenciado e do espectro complementar para tornar as comunicações de longas distâncias mais eficientes.

2.4.4 Estação Base em Nuvem

Os esquemas de comunicação baseados em nuvem são chamados de *Cloud-based Architectures* ou *Cloud-based radio access networks* – C-RANs [9]. Eles derivam da computação em nuvem tradicional e estendem seus benefícios, como acesso fácil, em demanda e escalável a um ambiente compartilhado de recursos configuráveis [36]. Assim, o objetivo de uma C-RAN é executar as funções de uma BS na nuvem.

De acordo com a China Mobile [14] as C-RAN se postulam como uma evolução natural para as estações base. Uma estação base em nuvem é composta de um rádio na estação base, que encaminha o processamento para uma unidade remota. Assim, é possível que para as redes 5G se tenha os processos de roteamento e alocação de recursos e serviços gerenciados nestas C-RANs. As funcionalidades do D2D podem ser importantes aqui, para reduzir a demanda de comunicação dos dispositivos com esta nuvem e, invés disto, adquirirem informações de seus pares.

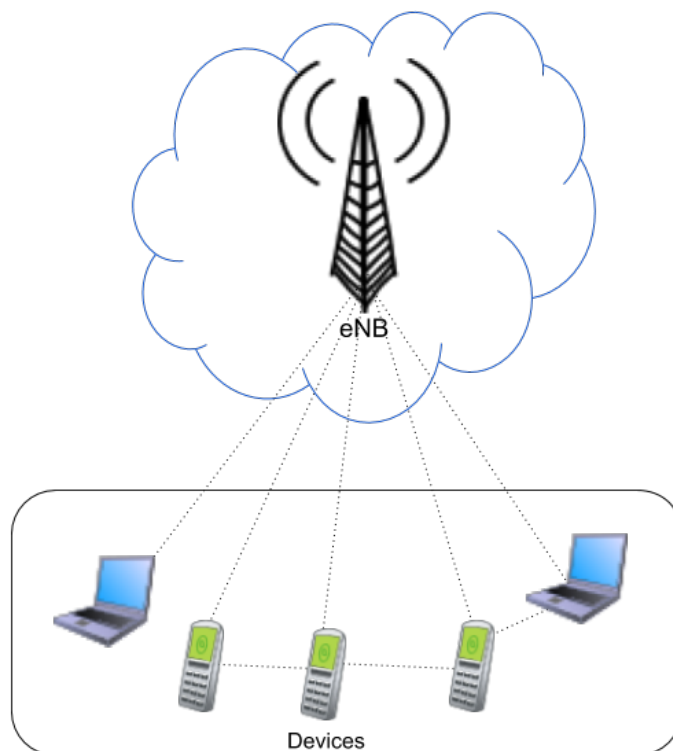


Figura 2.8: Esquema de comunicação com BS em nuvem.

2.4.5 MIMO Maciço (mMIMO)

A próxima geração de redes sem fio necessita acomodar um tráfego de dados cerca de 1000 vezes maior e 50 vezes mais dispositivo móveis comparado às redes móveis atuais [30]. Visto que o recurso espectral é escasso, particularmente em bandas adequadas para coberturas mais amplas, os principais aperfeiçoamentos na 5G precisam focar em um reuso espacial mais agressivo do espectro. Isto é, muito mais transmissões devem ocorrer simultaneamente por unidade de área. Essa característica pode ser alcançada pela tecnologia de MIMO maciço, na qual os pontos de acesso passam a ser equipados com centenas de antenas e podem atender até dezenas de usuários em cada recurso de tempo-frequência através da multiplexação espacial [36]. As diversas antenas fornecem a separação dos usuários no domínio espacial, o qual é visto como uma mudança de paradigma das tecnologias multi-usuários que se apoiam na separação dos usuários nos domínios do tempo ou da frequência.

2.4.6 Transmissão em Ondas Milimétricas (mmWave)

As comunicações por ondas em frequências milimétricas definem uma nova era da comunicação sem fio. As bandas de mmWave oferecem uma quantidade muito maior de canais para comunicação comparadas às bandas existentes nos sistemas sem fio comerciais. WLANs já estão atualmente explorando a banda 60 GHz mmWave, enquanto que os sistemas celulares 5G provavelmente irão operar entre 30 e 300 GHz [43]. Em razão dos enormes vetores de antenas, diferentes modelos de canais e novas restrições de hardware, a maneira como os sinais serão processados será diferente nos sistemas de comunicação mmWave.

2.4.7 Redes Definidas por Software e Virtualização de Funções de Rede

As redes 5G possuem como objetivos de projeto a eficiência, escalabilidade e a versatilidade. Elas precisarão ser extremamente eficientes em termos de energia e gerenciamento de recursos. Conectar um número gigantesco de dispositivos móveis necessita o desenvolvimento de funções de rede escaláveis e versáteis que atendam a um amplo número de serviços, incluindo baixo consumo, altas taxas de dados multimídia e aplicações sensíveis ao atraso. Tais requisitos direcionam os objetivos das redes 5G na inovação da virtualização das funções de rede (NFV). Em termos gerais, NFV pode superar os desafios das

redes 5G ao auxiliar na otimização do provisionamento de recursos, gerenciar e melhor aproveitar os recursos de *hardware* disponíveis e garantir certos parâmetros de desempenho [24]. As redes definidas por *software* (SDN) auxiliarão na virtualização e interconexão dos componentes responsáveis pela virtualização das funções de rede.

2.4.8 Armazenamento na Rede (*caching*)

O congestionamento da rede é um dos principais problemas da Internet hoje. No modelo cliente-servidor, uma página web é recuperada do mesmo servidor por cada usuário da Internet, modelo esse que em condições de tráfego intenso ao servidor causa gargalo e problemas de escalabilidade. Soluções para esses problemas, como *proxy* e redes de distribuição de conteúdos (CDNs), foram propostas ao longo dos anos. A ideia principal é replicar os conteúdos (vídeos, imagem, páginas web e etc) em servidores geograficamente próximos aos usuários e, dessa forma, reduzir o atraso fim-a-fim e o uso desnecessário da infraestrutura de rede. Atualmente, os pesquisadores estão revisitando o mesmo desafio no contexto das redes sem fio 5G. Na verdade, trazer cópias dos conteúdos para a borda de rede, como em estações base e seus usuários, é uma maneira promissora de aliviar a infraestrutura e reduzir o atraso fim-a-fim na recuperação dos conteúdos. Embora a motivação principal de trazer cópias dos conteúdos para a borda das redes celulares seja similar a das redes cabeadas, diversos desafios técnicos permanecem não resolvidos e envolvem multidisciplinaridade, como teoria da informação, aprendizado de máquina e comunicação sem fio [61].

Capítulo 3

Estado-da-Arte

Existem muitos desafios para o desenvolvimento e adoção da 5G em virtude das características de escalabilidade e mobilidade das redes celulares e dos objetivos a serem alcançados em termos de vazão e latência [12] [23] [22] [43] [60]. Entre os desafios estão a conservação de energia, a alocação de canais, o gerenciamento de interferência, modelos econômicos e de tarifação, a segurança, entre outros. Neste trabalho escolhemos focar em avaliar o consumo de energia com a introdução da comunicação D2D.

3.1 Comunicação Dispositivo-a-Dispositivo

Alguns trabalhos definem arquiteturas baseadas na comunicação D2D e muitos outros citam tal tecnologia como parte de outras arquiteturas para as redes 5G. Esse motivo levantou o interesse por tal investigação para esta pesquisa.

Como principais benefícios, a comunicação D2D proporciona melhoria na eficiência de espectro, pois permite que mais dispositivos utilizem um mesmo espectro com qualidade. Assim, se obtém um descarregamento de carga de dados das estações base da rede celular e melhores taxas de tráfego para os dispositivos. Também, estende o alcance da área de comunicação/cobertura das redes móveis [32]. Os dispositivos que atuam no intermédio dessa comunicação entre outros dispositivos podem auxiliar no estabelecimento destas conexões e são chamados de encaminhadores.

Para explicar o funcionamento da comunicação Dispositivo-a-Dispositivo, façamos uma analogia com a tecnologia LTE-Advanced (LTE-A), especificada pelo *Release* 10 [1] da 3GPP em 2011 e presente nas redes de telefonia atuais. Nela, os UEs comunicam-se somente com a Estação Base, chamada de eNB, a que estão associados, tanto para comu-

nicação em direção *downlink* (DL) quanto *uplink* (UL) [37]. D2D é um novo paradigma de comunicação que possibilita que dois ou mais terminais de uma rede celular se comuniquem diretamente, sem encaminhamento pela eNB. Esse caminho de troca de dados direta é chamado de *sidelink* (SL) e tem como vantagem não precisar fazer dois saltos, passando pelo eNB, para realizar sua comunicação. Sua comunicação é direta, em apenas um salto, o que reduz a latência e economiza recursos.

Devido a proximidade dos nós, tornam-se possíveis serviços de comunicação *unicast* e *multicast* em diversos cenários, como Redes Veiculares, Internet das Coisas e aplicações Máquina-a-Máquina [37]. E apesar de a comunicação poder ser feita direta, o eNB ainda pode ser responsável por estabelecer um canal que controle a alocação de recursos [36]. A Figura 3.1 exemplifica esta situação. Há uma célula de comunicação D2D, na qual os UEs se comunicam entre si. No entanto, a eNB, baseada em algum critério, como o de proximidade, escolheu o UE3 como aquele que estabeleceria um canal de controle. Ele gerenciará aquela célula e a ele ela enviará as informações de alocação de recursos.

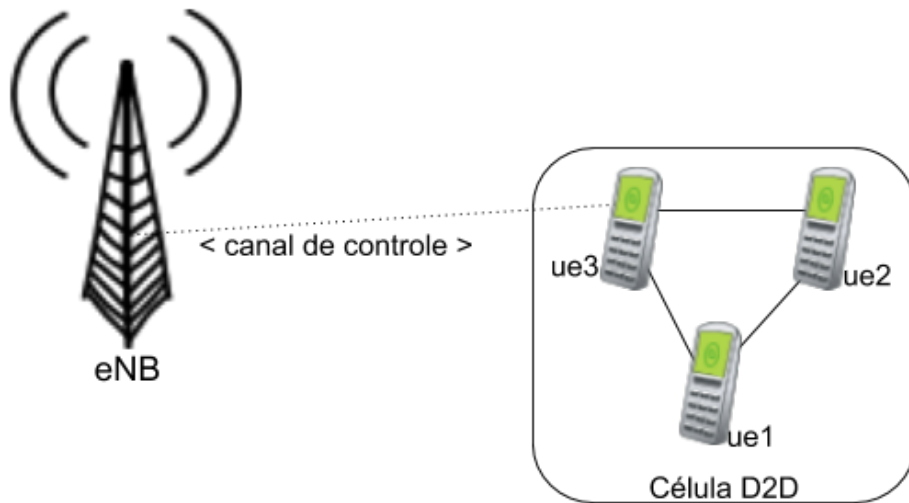


Figura 3.1: Célula D2D controlada pela eNB.

Nas classificações DR-OC e DR-DC, em que dispositivos atuam como encaminhadores, abre-se um leque de possibilidades sobre como esses dispositivos devem atuar e como escolhê-los.

Whitbeck et al. [62] discutem a elegibilidade de nós a partir da utilização de funções objetivo que escolhem quando um nó deve se tornar encaminhador. Ainda, é considerada a participação compulsória dos usuários como encaminhadores, se assim forem escolhidos.

Rebecchi et al. [48] dividem os nós entre consumidores e receptores. Em seguida, usam uma função matemática para determinar qual fração dentre os nós consumidores deve ser promovida a encaminhadores, e quando isso deve acontecer. Há em comum entre ambas propostas o fato de serem soluções encontradas a partir de pequenos cenários, de uma célula só.

Contudo, os trabalhos que discutem a elegibilidade de nós encaminhadores se direcionam por analisar a qualidade do enlace do encaminhador com a eNB. Ainda não foram encontrados trabalhos que, por exemplo, estabeleçam os critérios de elegibilidade baseados na disponibilidade de recursos físicos dos nós, como memória, CPU, bateria e vazão de rede.

Um outro fator que precisa ser considerado é a capacidade de conectividade de cada nó. Devido a questões de bateria, processamento e a própria limitação de hardware das interfaces de rede, há uma quantidade limitada de dispositivos que poderão se conectar a cada encaminhador. Assim, a escolha do encaminhador poderia considerar mais de um critério. Por exemplo, em determinados contextos, poderia ser permitido que dispositivos se conectem em encaminhadores que estejam com baixa carga de bateria, desde que outros que estejam com carga mais alta não tenham mais disponibilidade para aceitar novas conexões.

3.1.1 Descarregamento de Tráfego

As demandas de *streaming* de vídeo têm crescido cada vez mais recentemente. Seja pela demanda de serviços de conteúdo em vídeo ou pelo vídeo ao vivo em chamadas e transmissões [50]. Um outro fator comum no comportamento dos usuários de Internet atual é a “viralização” de conteúdo. Isto é, quando um assunto é repentinamente descoberto e velozmente compartilhado.

Situações como estas, em que o número de usuários multiplica significativamente em um espaço de tempo muito curto, geram sobrecargas nas redes que podem ocasionar lentidão e indisponibilidade. Uma das formas de implementar o descarregamento é utilizando a D2D. Sua arquitetura já se configura de forma distribuída. Então, se em pontos estratégicos de uma rede D2D se criarem *caches* destes conteúdos, o tráfego será distribuído para as bordas das redes e os momentos de sobrecarga são rapidamente amenizados.

Karunakaran et al.[28] produziram um trabalho que traz uma solução para este problema. Os autores elaboraram estratégias cooperativas centradas em conteúdo para re-

dução do consumo energético. Consideraram nos cenários analisados para formar grupos comuns (*mobile cloud*) de consumidores, o tipo de conteúdo consumido por aqueles usuários. Eles demonstraram que nestes grupos a cooperação gera redução de consumo energético. Quando todos usuários de um grupo requisitam os mesmos conteúdos, a taxa de transferência é maior entre o grupo cooperativo do que deles para com a rede celular. Isto aumenta os tempos de transmissão ociosos, proporcionando redução energética.

3.1.2 Consumo de Energia

Para discutir consumo de bateria, é essencial entender como se dá o funcionamento básico das baterias dos dispositivos móveis. O site The Statistics Portal realizou no primeiro trimestre de 2018 uma pesquisa de *market share* global das baterias de íon de lítio [52]. A Panasonic Sanyo foi a empresa apontada em primeiro lugar, com 21,1% das vendas deste mercado. Por isto, tomaremos como referência as funcionalidades de baterias desta empresa.

No site da Panasonic está disponível um documento com as especificações e funcionalidades de sua bateria de íon de lítio [42]. Nele, é possível observar a curva de depleção da bateria. A depleção é o processo pelo qual passa um material semicondutor de medição da densidade de portadores de carga abaixo do seu nível e do nível de dopagem em uma temperatura específica [20].

Ainda neste documento, este fabricante afirma que suas baterias usam um carbono especial em sua composição que permite a produção de baterias estáveis com perfil de descarga menos acentuado durante o tempo de descarga da bateria. Mesmo assim, supondo que esta seja, de fato, uma bateria de boa qualidade, ela ainda apresenta uma curva de depleção variável, como é possível observar na Figura 3.2.

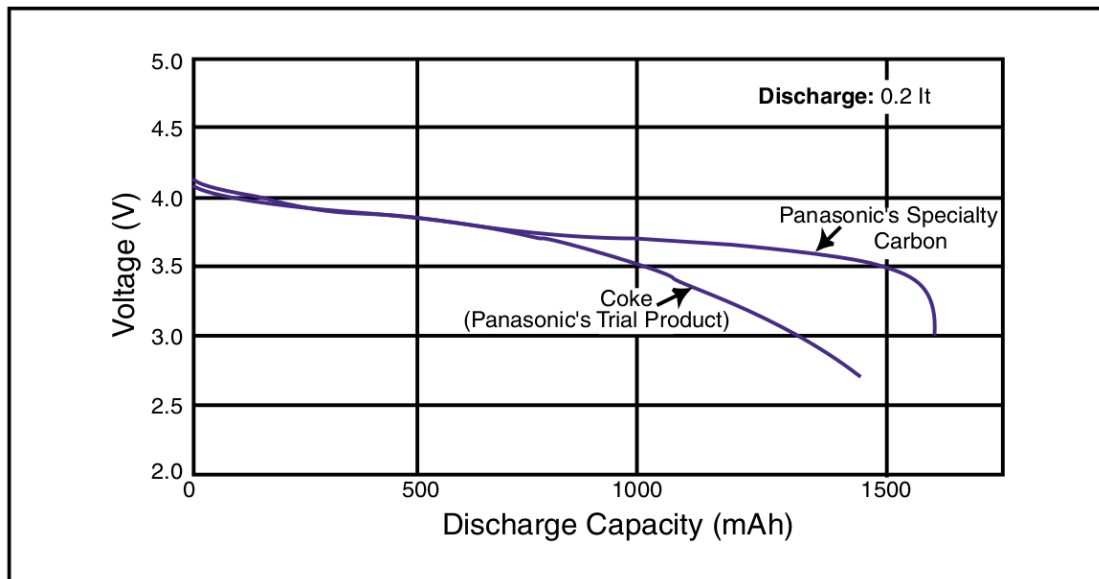


Figura 3.2: Curva de depleção de bateria de íon de lítio da Panasonic Sanyo. Adaptado de [42].

Nesta figura, a curva de depleção não é linear, mas variável. É possível perceber, por exemplo, uma acentuada queda nesta curva após os 500 mAh e nova acentuação na queda após os 1000 mAh. Esta característica é ocasionada pelas condições físicas e químicas deste tipo de bateria. Além disso, a temperatura também é outro fator que pode influenciar na depleção [19]. Ainda, devido às diferenças nas formulações química e física esta curva de depleção será diferente ainda em baterias de outras marcas. E para os casos de todas as baterias, a curva também se altera com o uso da bateria e o passar dos ciclos de vida.

Toda esta complexidade envolvida no processo de produção das baterias, que tem direta influência em seus mecanismos de carga e descarga, torna a literatura insipiente. Pouquíssimos trabalhos discutem abertamente este problema de implementação de um modelo de bateria não-linear, assim como é a depleção das baterias. Menos ainda são aqueles que explicam a dificuldade desta implementação, devido a especificidades de cunho físico-químico.

Logo, como já mencionado, não foram encontrados simuladores com modelos de baterias implementados que simulem D2D em redes móveis. Assim, a solução que adotamos para conseguir avaliar o consumo de energia das comunicações D2D nas redes celulares, foi realizar as simulações no SimuLTE, que implementa a comunicação D2D no contexto de redes móveis LTE. A partir dos dados das simulações, coletamos a quantidade de bytes enviados e recebidos e realizamos um modelagem matemática para transformá-los em

consumo energético. A Seção 4.2 trata desta modelagem em detalhes.

3.2 Simuladores

Para o desenvolvimento das simulações deste trabalho não foi encontrado nenhum simulador que pudesse atender totalmente à necessidade da avaliação pretendida: simular consumo energético de dispositivos móveis sob comunicação D2D em redes LTE.

No entanto, defrontamo-nos com três soluções que poderiam atender parcialmente as necessidades do projeto: Modelo LTE de D2D para ns-3, Módulo do ns-3 para Consumo de Energia em dispositivos LTE e o SimuLTE, os quais detalhamos a seguir.

3.2.1 Modelo LTE D2D para ns-3

O *National Institute of Standards and Technology*, nos Estados Unidos, criou um *fork* do projeto do ns-3 e produziu, a partir dele, uma versão do simulador com suporte a D2D. O projeto [49] se trata de uma derivação do modelo LTE já existente no ns-3 que implementa o D2D focado em Serviços de Proximidade (ProSe - *Proximity Services*). No entanto, este trabalho não considera o consumo de energia e, portanto, não acrescenta nenhum módulo de bateria.

3.2.2 Um Módulo do ns-3 para Consumo de Energia em dispositivos LTE

Um módulo do ns-3 para consumo de energia em dispositivos LTE está disponível [44]. Neste trabalho o *fork* que foi feito do projeto original do ns-3 acrescenta as funcionalidades do módulo de bateria, mas não adiciona nenhuma possibilidade de comunicação D2D.

Inicialmente, ao analisar que a falta deste projeto [49] é complementar a do anterior, [44] analisamos a viabilidade de integrá-los, o que foi desconsiderado após perceber que cada um tem sua implementação em forma de *forks* diferentes do ns-3 e não são modulares. Assim, foi incompatível reunir funções desejadas dos dois projetos, comunicação D2D e módulo de energia, em um projeto só.

3.2.3 OMNeT++ com SimuLTE

A última ferramenta analisada, e mais completa, foi o SimuLTE [57], que é baseado em OMNeT e INET, as quais serão explicadas a seguir.

OMNeT++ é um *framework* modular orientado a objetos para simulação de funções de redes [55]. Sua arquitetura é genérica e pode ser utilizada em vários domínios, como: modelagem de redes de comunicação com fio e sem fio; modelagem de protocolos; modelagem de sistemas distribuídos; validação de arquiteturas de hardware, entre outros.

O OMNeT++ não é um simulador por si só, mas sim, provê infraestrutura e ferramentas para simulação. Uma de suas principais vantagens, é sua arquitetura baseada em componentes para modelos de simulação. Assim, modelos permitem ser reusados por seus componentes. Módulos podem ser combinados de várias formas, como blocos.

Estes módulos podem ser conectados através de interfaces do tipo *gates* e combinados para formar módulos compostos. Módulos comunicam através de mensagens, estas, que carregam estruturas de dados variadas. Os módulos podem enviar mensagens para uma central de visibilidade pública ou diretamente para seus destinatários. Estes módulos são programados em C++ e utilizam as bibliotecas de simulação.

As simulações podem ser executadas em uma interface gráfica, animada, que facilita demonstrações e processos de *debugging* ou em linha de comando para execução com maior necessidade de desempenho.

Tanto o simulador quanto sua interface são portáteis e podem funcionar em sistemas operacionais como Linux, Mac OS/X e Windows. Esta versão do OMNeT++ é livre para uso acadêmico e sem fins lucrativos.

O INET é um *framework* de Redes para OMNeT++ que contém implementações dos protocolos IPv4, IPv6, TCP, SCTP e UDP, e vários modelos de aplicação. O *framework* possui os modelos PPP, Ether e 802.11 da camada de enlace [40].

O INET suporta simulações de redes móveis e sem fio e o roteamento estático pode ser configurado usando autoconfiguradores de rede ou implementações de protocolos de roteamento. Ele também tem um modelo de bateria com módulos de consumo, geração e armazenamento de energia. No entanto, sua implementação é atrelada ao rádio de *WiFi* configurado na camada física do simulador. Por isto, não pode ser estendido ou utilizado em outros modelos/*frameworks* do OMNeT++.

O SimuLTE [54], simulador utilizado neste trabalho, é um *framework* baseado no

INET e OMNeT++. Ele roda sobre o OMNeT++, e depende também de modelos de rede do INET, dos quais herda funcionalidades e, em algumas, faz implementações complementares. O SimuLTE é um dos modelos disponíveis no OMNeT++, que provê um *framework* de simulações para o Plano de Dados de redes LTE-A. Uma vantagem de se utilizar o SimuLTE é sua implementação como um *Network Interface Card* (NIC), pois facilita sua integração com qualquer outro módulo OMNeT++. As simulações são providas com a composição de diferentes módulos que interagem entre si através da troca de mensagens [37].

Optamos utilizar este simulador, porque ele tem os serviços de D2D implementados em contexto de redes móveis LTE-A. Mesmo não tendo funções de energia disponíveis, foi o mais completo dentre os analisados e mais fácil de adicionar novas funcionalidades, por ser modular.

Ele possibilita simulações LTE/LTE-A em modo FDD (Frequency Division Duplexing), com redes heterogêneas e possui modelos para os UEs e BSs. Nele é possível, inclusive, simular comunicações dispositivo-a-dispositivo (D2D - *Device-to-Device*) em modo unicast ou multicast [57][58].

O SimuLTE implementa simulações de camada de enlace, usadas tipicamente para testar as propriedades dos canais para links D2D, em virtude da falta de ferramentas para avaliar o desempenho D2D em nível de enlace. Por isso o SimuLTE, com suporte ao D2D, se mostra uma ferramenta eficaz para realizar testes de desempenho de novos algoritmos e serviços em diferentes cenários e configurações. [37]

Nativamente o SimuLTE permite aplicar diferentes potências de encaminhamento para o UL (Tx) e o D2D. Isso possibilita, por exemplo, avaliar a distância máxima ou a energia consumida por uma comunicação direta sob uma combinação específica de CQI (*Channel Quality Indicator*) e potência de transmissão (*transmission power*). Uma vez que a comunicação D2D possibilita a reutilização de frequência espacial, também é possível realizar testes de interferência e de desempenho de algoritmos que exploram reuso de frequência e coexistência de comunicação D2D e tradicional, por exemplo.

Quando dois UEs estão habilitados para comunicação D2D, o canal de comunicação entre eles é chamado de *sidelink* (SL). Nesse caso, a comunicação pode ser tanto no modo D2D, quanto no modo tradicional, infraestruturada. Um exemplo, de quando pode ser trocada a comunicação D2D pela infraestruturada, é se determinado UE se distancia de outros com quais tem comunicação, reduzindo o SINR nesse *sidelink*, ou se um eNB quer otimizar a alocação de recursos de acordo com suas próprias políticas de funcionamento.

No SimuLTE, a transmissão de dados e alocação de recursos é separada. A alocação de recursos é feita por um módulo central, o Binder, o qual monitora os blocos de recursos no sistema. O Binder é um módulo oráculo, que tem visibilidade completa do sistema, de todos os nós e pode ser solicitado por qualquer nó que queira obter alguma informação compartilhada, ou seja, ele mantém registros de cada alocação de recursos feita por cada nó, quando a eNB transmite em *downlink* (DL) e quando os UEs transmitem em *uplink* (UL). O fluxo de dados é modelado por trocas de mensagens entre módulos. A logística entre trocas de mensagens e alocação de recursos é mantida pelo Binder. Ele associa uma quantidade de blocos de alocação de recursos para cada mensagem.[56]

Capítulo 4

Desenvolvimento

Não havendo até o momento em que se realiza esta pesquisa nenhum simulador de redes que possua função de simular consumo de bateria em comunicação D2D em redes móveis LTE-A, a ferramenta escolhida foi o SimuLTE. Além dos fatores já mencionados sobre esta ferramenta, ele tem um mínimo de referencial bibliográfico disponível.

As simulações em D2D nesta ferramenta possibilitam configurações flexíveis e têm se tornado populares na comunidade acadêmica. No entanto, falta a funcionalidade de simulação de funções de energia/bateria. Por isso, tentamos implementar as funcionalidades de consumo de energia/bateria no SimuLTE. Mas as dificuldades foram várias.

4.1 Dificuldades Encontradas

Alguns dos obstáculos enfrentados na busca por um simulador ideal, a princípio, pareciam possíveis de solucionar, mas com o entendimento dos problemas, descobríamos inviabilidades. Não encontramos nas bibliografias relatos claros destas dificuldades. Assim, é de pontuar aqui as principais dificuldades, para que outros pesquisadores possam tomá-las como experiências, partir suas pesquisas deste ponto, e evoluir um pouco mais a temática.

4.1.1 Módulo de Energia do INET

O INET [27], *framework* de redes do simulador OMNeT++, e o qual serve como base para o SimuLTE, que utiliza de seus modelos para estender as funcionalidades LTE, dispõem de um módulo de modelos de energia que consiste de três componentes: consumo, geração e armazenamento de energia.

Os modelos de consumo de energia são dois:

- **AlternatingEnergyGenerator**: um modelo estatístico trivial, baseado em consumo de energia; e
- **StateBasedEnergyConsumer**: um modelo de consumo de energia baseado no modo do rádio do INET e seus estados de transmissão e recepção.

Já os modelos de armazenamento de energia são:

- **IdealEnergyStorage**: modelo ideal com capacidade de energia e fluxo de energia infinitos;
- **SimpleEnergyStorage**: é um modelo não-trivial, que integra as diferenças entre o total de energia consumida e o total de energia gerada ao longo do tempo;
- **SimpleBattery**: é um modelo mais realístico, baseado em carga e corrente utilizando fonte de tensão ideal independente e resistência interna.

Para este estudo, foi considerada inexistente a geração de energia na rede.

Este módulo de energia do INET é separado de outros modelos de simulação. Esta separação faz o modelo de energia extensível. Assim, diante a necessidade de inferir consumo energético de encaminhadores em redes móveis, a primeira solução planejada foi a de se aproveitar o módulo e criar, no SimuLTE, uma interface de acesso a ele.

Veja na Figura 4.1 o posicionamento do módulo Power do INET em sua estrutura de diretórios.

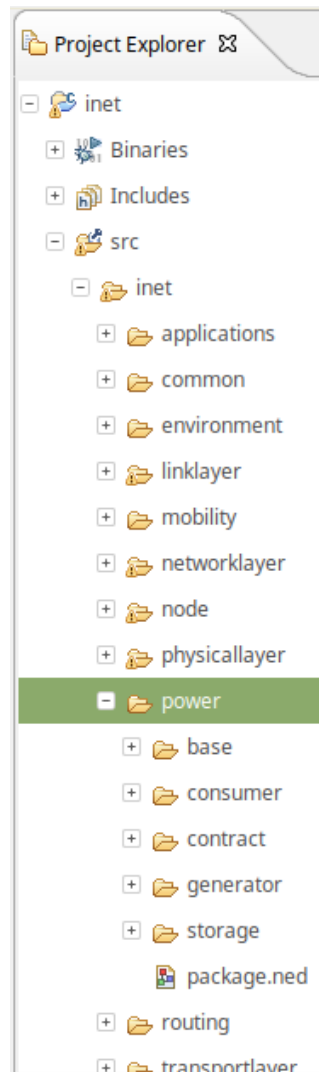


Figura 4.1: Posicionamento do módulo power na estrutura de diretórios do INET.

Utilizando a sistemática de interface definida pela API do INET/SimuLTE, na classe de definição dos equipamentos de usuários (UEs - *User Equipments*) foram adicionados submódulos de armazenamento, geração e consumo de energia, estendendo o módulo “power” do INET, como pode ser visto no trecho de código 4.1 do arquivo de descrição de rede que cria os UEs.

Código 4.1: Adicionando módulos de energia do INET ao SimuLTE.

```

1
2 module Ue
3 {
4     parameters:
5         .
6         .
7         .
8     //# Energy capability
9     bool hasStatus = default(false);
10    string energyStorageType = default("");
11    string energyGeneratorType = default("");

```

```

12     *.energySourceModule = default (energyStorageType != ""
13         ? absPath(".energyStorage") : "");
14     string energyConsumerType =
15         default ("StateBasedEnergyConsumer");
16     .
17     .
18     .
19     submodules:
20     .
21     .
22     .
23     status: NodeStatus if hasStatus { //HostState
24         @display("p=49.068,225.504");
25     energyStorage: <energyStorageType> like IEnergyStorage if
26         energyStorageType != "" {
27     parameters:
28         @display("p=49.068,335.124;i=block/plug;is=s");
29     }
30     energyGenerator: <energyGeneratorType> like
31         IEnergyGenerator if
32         energyGeneratorType != "" {
33     parameters:
34         @display("p=49.068,280.836;i=block/
35         plug;is=s");
36     }
37     energyConsumer: <energyConsumerType> like
38         IEnergyConsumer if
39         energyConsumerType != "" {
40     parameters:
41         @display("p=100,350");
42     }

```

Este módulo seria especialmente útil, utilizando o modelo de consumo de energia baseado em estado em conjunto com modelo de armazenamento *SimpleEnergyStorage*, para que pudesse ser reduzida da bateria toda energia consumida em cada transmissão de um pacote, considerando os diferentes valores de consumo de acordo com cada estado de consumo. No entanto, ao instanciar o módulo (veja no Código 4.2) nos arquivos de configuração da rede, ao executar as simulações foi obtido erro de *casting* entre a camada física do SimuLTE e a do INET. Investigando o erro, percebemos que a implementação do módulo estava diretamente atrelada, de forma não modular, à implementação do rádio da interface de rede 802.11 do INET, para que pudessem ser captados os estados do rádio.

Código 4.2: Instanciando módulos de energia do INET no SimuLTE.

```

1
2 # Setting power consumption
3 **.energyStorageType = "SimpleEnergyStorage"
4 **.energyConsumerType = "StateBasedEnergyConsumer"
5 **.energyStorage.nominalCapacity = 0.05J
6 **.energyStorage.nodeShutdownCapacity = 0J
7 **.energyStorage.nodeStartCapacity = 0.5 * this.nominalCapacity
8 **.energyStorage.initialCapacity=uniform(0J,this.nominalCapacity)
9 **.energyGeneratorType = "AlternatingEnergyGenerator"
10 **.energyGenerator.energySinkModule = "^energyStorage"
11 **.energyGenerator.powerGeneration = 0mW #mW
12 **.energyGenerator.sleepInterval = exponential(10s)
13 **.energyGenerator.generationInterval = exponential(10s)

```

Neste momento, se tornou difícil a solução de usar uma interface no SimuLTE para o módulo do INET, tendo em vista a impossibilidade de desacoplar o módulo de energia do modelo de rádio definido no INET.

4.1.2 *Port* do MiXiM para o SimuLTE

O MiXiM [35] é um framework do simulador OMNeT++ para simular redes móveis e fixas, que permite, inclusive simular consumo de energia, através de um módulo de bateria próprio.

Analizando a arquitetura deste módulo, percebemos uma estrutura modular, para qual pode ser implementada uma interface de comunicação e integrada ao SimuLTE. Assim, foi tentado fazer um *port* do módulo de energia do MiXiM para o SimuLTE.

A inviabilidade deste *port* foi percebida quando na tentativa de compilar o código portado, se iniciaram os problemas de dependência de biblioteca já inexistentes e ausência de definições de *kernel* criadas para o MiXiM e já não suportadas mais na versão atual do OMNeT++.

Alguns problemas de dependência não foram possíveis de resolver, devido a defasagem deste projeto, com funções e bibliotecas antigas que já não existem mais ou foram substituídas. O MiXiM fora criado com compatibilidade para o OMNeT++ versão 1. Atualmente o OMNeT está na versão 5.2 e o MiXiM fora descontinuado, o que tornou esse *port* impraticável.

4.1.3 Junção de Projetos do ns-3

Uma outra tentativa de solucionar o problema com tecnologias já existentes foi juntar os projetos [49] [44], citados nas Seções 3.2.1 e 3.2.2. Uma vez que o primeiro implementa o D2D para LTE e o segundo apresenta um módulo de consumo de energia para LTE, a junção das duas soluções seria o ideal para este projeto.

No entanto, isto também não foi possível pela forma como os projetos foram implementados. Cada um é um *fork* independente do projeto ns-3 e tem dependências do código principal difíceis de serem desfeitas, modularizadas e integradas entre os projetos.

4.1.4 Adaptação de Módulo de Energia do INET à Camada Física do SimuLTE

Diante o insucesso das tentativas anteriores e já ciente do problema de *casting* que impossibilitava a utilização direta do módulo de estados do INET, tentamos uma solução mais simples: aproveitar a função de consumo de energia do INET, instanciá-la no SimuLTE e incluir uma chamada para essa função na camada física do rádio LTE toda vez que houvesse um evento de transmissão ou recepção de um pacote por um dos nós.

Para tanto, importamos a biblioteca *"StateBasedEnergyConsumer.h"* do INET para os arquivos SimuLTE *"LtePhyUe.h"* e *"LtePhyUeD2D.h"*, essas por sua vez, bibliotecas de *"LtePhyUe.cc"* e *"LtePhyUeD2D.cc"*, nas quais estão os eventos de transmissão UL/DL e SL, respectivamente. Os arquivos mencionados neste parágrafo podem ser encontrados no Apêndice A.

Criamos uma função *"useBattery"* que acionava o uso da bateria à partir do arquivo de inicialização da rede e a adicionamos nas funções das classes .cc do UE. Nesta função, *rxAmount* e *txAmount* representam os valores de recepção e transmissão que deveriam ser decrementados da bateria. Estes valores também são instanciados no arquivo de configuração da rede e admitem valores diferentes, além de para transmissão e recepção, também para SL e DL/UL. No Código 4.3 é possível verificar como a função *useBattery* foi inserida na camada física do SimuLTE.

Código 4.3: Funcao useBattery.

```

1
2 // Instanciando a funcao de atualizacao do
3 // modulo de bateria do INET.
4 updatePowerConsumption=inet::power::AlternatingEnergyConsumer.
5     updatePowerConsumption(W amount)
6
7 // Funcao de decremento da bateria a partir
8 // da energia consumida para transmissao.
9 amount = txAmount;
10 if (useBattery_){
11     updatePowerConsumption(amount);
12 }
13
14 // Funcao de decremento da bateria a partir
15 // da energia consumida para recepcao.
16 amount = rxAmount;
17 if (useBattery_){
18     updatePowerConsumption(amount);
19 }

```

Não houve erros de compilação nestas implementações, a chamada da função acontecia, no entanto, a bateria não decrementava. Descobrimos a posteriori que isso deve-se ao fato de que as funções nestes simuladores estavam se comunicando por meio da troca de

mensagens padrão do OMNeT++.

É preciso então, primeiro implementar o processo de leitura e envio de mensagens para a central de mensagens do simulador. E, em seguida, entender como fazer comunicar as mensagens das funções de dois espaços de trabalho diferentes.

A busca de soluções para este problema ocorreu em paralelo com a tentativa de implementação do módulo de bateria diretamente no SimuLTE, descrita na próxima seção. No entanto, ambas tentativas foram interrompidas devido a inviabilidade descoberta quando ocorreu o entendimento da problemática de depleção de bateria, já explicada no Capítulo 3.

4.1.5 Criação de Módulo de Energia no SimuLTE

Visto o trabalho necessário no final da seção anterior, para solucionar a comunicação por mensagens, foi idealizada a implementação de um módulo de bateria no SimuLTE que tivesse modelos de simulação de consumo e armazenamento de energia. Uma vez solucionada a dinâmica de comunicação por mensagens partindo de espaços de trabalhos diferentes, poderíamos implementar o módulo no próprio SimuLTE, sem necessidade de alterar código do INET, de maneira a reduzir as dependências.

Nesta etapa de pesquisa, buscamos mais sobre modelagem de baterias na literatura para entender seu funcionamento, para definir como poderia ser feito o projeto.

Como resultado destas pesquisas e conversas com outros pesquisadores, veio o conhecimento da problemática da depleção de bateria, já mencionada na Seção 3.1.2. Então, esta proposta foi inviabilizada pela tamanha complexidade envolvida na implementação de um modelo realístico de bateria de íons de lítio e o tempo restante para finalização desta pesquisa.

4.2 Análise do Consumo Energético

Depois de feitas todas essas tentativas com abordagens diferentes, percebemos que as implementações ficaram não-funcionais ou simples demais para a complexidade que o problema apresentou ao longo da pesquisa.

Logo, não havendo disponível um simulador que além de prover comunicação D2D, implementasse um modelo de consumo de energia não linear, a ação seguinte foi realizar simulações de D2D e modelar um consumo de energia por cada nó multiplicando a

quantidade de bytes transmitidos e recebidos por uma constante de consumo de energia.

Para analisar o consumo energético, consideramos a quantidade e tipos de pacotes trafegados em cada nó. O consumo energético de uma interface LTE para realizar transmissão e recepção de pacotes se diferem. De acordo com Höyhty et al [26] para recepção de pacotes o consumo é de $51,97mW/(Mb/s)$ e para transmissão de $438,39mW/(Mb/s)$.

Considerando o tempo total de transmissão de uma simulação e a quantidade de pacotes que a interface de um dispositivo enviou e recebeu naquele período, é possível modelar o consumo energético desta forma:

$$C(p_{rx}) = \left(\frac{\left(\frac{Q_B(p_{rx})}{1e-6} \right)}{t_{rx}} \right) * \alpha \quad \alpha = 51,97mW/(Mb/s) \quad (4.1)$$

A Equação 4.1 apresenta o consumo energético para um nó receptor ($C(p_{rx})$) em $mW/Mbps$, na qual $Q_B(p_{rx})$ é a quantidade, em grandeza de armazenamento computacional, de dados recebidos por um nó p receptor (r_x). Os resultados de simulação retornam este dado em bytes, o qual dividimos por $1e-6$ para transformar em megabits. Como o consumo desejado para o nó é dado por mW por Mb por segundo, dividimos por t_{rx} que é o tempo pelo qual o nó p receptor rx recebeu pacotes. Por fim, multiplicamos por α para ter o consumo total do nó de acordo com a quantidade de megabits transmitidos no intervalo de tempo [26].

$$C(p_{tx}) = \left(\frac{\left(\frac{Q_B(p_{tx})}{1e-6} \right)}{t_{tx}} \right) * \beta \quad \beta = 438,39mW/(Mb/s) \quad (4.2)$$

Na Equação 4.2 o princípio é o mesmo que na Equação 4.1, alterando apenas os nós receptores (p_{rx}) para transmissores (p_{tx}), o tempo de consumo na recepção (t_{rx}), para o tempo de consumo na transmissão (t_{tx}) e a taxa de consumo, também para a de transmissão (β).

$$C(R) = \sum_{C(p_{rx_0})}^{C(p_{rx_{n-1}})} + \sum_{C(p_{tx_0})}^{C(p_{tx_{n-1}})} \quad (4.3)$$

Assim, podemos calcular o consumo energético total da rede $C(R)$ realizando uma adição da somatória do consumo de cada nó receptor [$C(p_{rx_1})..C(p_{rx_n})$], com a somatória de consumo dos nós transmissores [$C(p_{tx_1})..C(p_{tx_n})$], como demonstrado na Equação 4.3.

Utilizamos esta modelagem nos resultados dos experimentos realizados, e no Capítulo 5 detalhamos a análise de consumo energético de cada experimento com os respectivos gráficos, para melhor visualização.

Capítulo 5

Simulações e Resultados

5.1 Simulações

As tecnologias para uma nova geração de telefonia celular nascem sempre de experimentações teóricas antes de serem testadas em campo. As simulações são uma das maneiras de experimentos mais eficazes para tal, pois há ambientes de simulações que oferecem configurações com restrições e variáveis próximas do real, possibilitando resultados realísticos. Por esse motivo, este trabalho realiza simulações de comunicação dispositivo-a-dispositivo em redes LTE-A, para identificar como se dá o consumo de energia nestes cenários.

Essa problemática de quando usar a comunicação D2D e quando usar a infraestrutura tem ganhado atenção da comunidade acadêmica e, consequentemente, aumentaram as simulações com esse tipo de comunicação. O intuito é descobrir algoritmos e técnicas que aumentem, por exemplo, o desempenho tanto do roteamento quanto descarregamento de tráfego de rede [34] [63].

No simulador SimuLTE, nos baseamos na arquitetura de rede nomeada "Single-Cell_D2DMulticast". A Figura 5.1 é um exemplo de uso do SimuLTE que traz uma demonstração gráfica desta configuração, e o Código 5.1 traz a programação em NED (*Network Description*) linguagem de descrição de rede do OMNeT++.

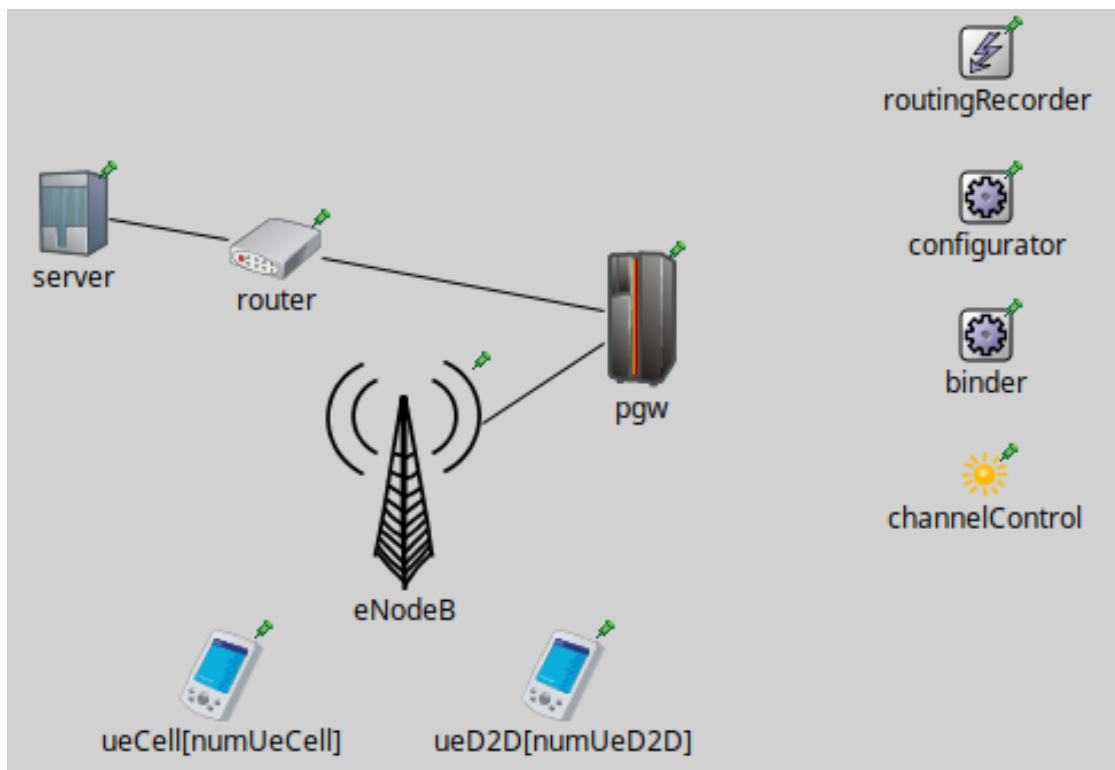


Figura 5.1: Visualização gráfica da arquitetura de rede D2D base.

Código 5.1: Arquivo de Descrição.

```

1
2 package lte.simulations.networks;
3 .
4 .
5 .
6 network SingleCell_D2DMulticast
7 {
8     parameters:
9         int numUeCell = default(1);
10        int numUeD2D = default(0);
11        @display("i=block/network2;bgb=991,558;bgi=background/
12        budapest");
13    submodules:
14        channelControl: LteChannelControl {
15            @display("p=705.20624,258.7;is=s");
16        }
17        routingRecorder: RoutingTableRecorder {
18            @display("p=706.45,29.85;is=s");
19        }
20        configurator: IPv4NetworkConfigurator {
21            @display("p=706.45,108.206245");
22            config = xmldoc("demo.xml");
23        }
24        binder: LteBinder {
25            @display("p=706.45,182.83125;is=s");
26        }
27        server: StandardHost {
28            @display("p=212,118;is=n;i=device/server");
29        }
30        router: Router {
31            @display("p=321,136;i=device/smallrouter");
32        }

```

```

33     pgw: PgwStandardSimplified {
34         nodeType = "PGW";
35         @display ("p=519,175;is=l");
36     }
37     eNodeB: eNodeB {
38         @display ("p=391,259;is=v1");
39     }
40     ueCell[numUeCell]: Ue {
41         @display ("p=292,367");
42     }
43     ueD2D[numUeD2D]: Ue {
44         @display ("p=477,367");
45     }
46     connections:
47         server.pppg++ <==> Eth10G <==> router.pppg++;
48         router.pppg++ <==> Eth10G <==> pgw.filterGate;
49         pgw.pppg++ <==> Eth10G <==> eNodeB.ppp;
50 }

```

Verificando-os, é possível perceber que é instanciado um parâmetro para criação de células na rede e outro para criação de nós UEs com capacidade D2D.

Também são instanciados submódulos necessários para controle do canal LTE, gravação de tabela de roteamento, configurador de rede com protocolo IPv4, roteador de rede, *gateway* de pacotes de dados da rede (*PGW - Packet Data Network Gateway*), um *host* servidor, para rodar as aplicações, estação base da rede LTE (eNodeB), células de UEs para a rede LTE, nós para a rede LTE e, por fim, o Binder, que é o módulo criado pelo SimuLTE para monitorar e fornecer informações sobre a alocação de recursos na rede.

Ainda, são estabelecidas conexões Ethernet de 10 Gb/s do servidor para o roteador, do roteador para o *gateway* e do *gateway* para a estação base. Observando que a estação base se conectará aos dispositivos móveis por meio de rádio sem fio LTE. Para melhor entendimento, vale observarmos também como está arquitetado o módulo que cria os nós UEs. Ele é descrito no Código 5.2.

Código 5.2: Arquitetura do módulo de nós UE.

```

1
2 package lte.corenetwork.nodes;
3
4 import inet.applications.contract.ITCPApp;
5 import inet.applications.contract.IUDPAApp;
6 import inet.mobility.contract.IMobility;
7 import inet.networklayer.common.InterfaceTable;
8 import inet.networklayer.contract.IRoutingTable;
9 import inet.networklayer.contract.INetworkLayer;
10 import inet.transportlayer.tcp.TCP;
11 import inet.transportlayer.udp.UDP;
12 import lte.stack.phy.LteNicUe;
13 import lte.stack.phy.LteNicUeD2D;
14
15 module Ue
16 {
17     parameters:
18         @networkNode();
19         @display ("i=device/pocketpc;bgb=400,518");

```

```

20
21     ///# Mobility
22     string mobilityType = default("StationaryMobility");
23
24     ///# Apps
25     int numTcpApps = default(0);
26     int numUdpApps = default(0);
27
28     ///# Node specs
29     string nodeType = "UE";    // DO NOT CHANGE
30     int masterId;
31     int macNodeId = default(0);
32     int macCellId = default(0);
33
34     ///# D2D capability
35     bool d2dCapable = default(false);
36
37     ///# Network Layer specs
38     string networkLayerType = default("IPv4NetworkLayer");
39     string routingTableType = default("IPv4RoutingTable");
40     *.interfaceTableModule = default(absPath(".interfaceTable"));
41     *.routingTableModule = default(absPath(".routingTable"));
42
43     gates:
44         input radioIn @directIn;    // connection to master
45
46     submodules:
47         interfaceTable: InterfaceTable {
48             @display("p=50,75;is=s");
49         }
50         // routing table
51         routingTable: <routingTableType> like IRoutingTable if
52             routingTableType != "" {
53             parameters:
54                 @display("p=50,125;is=s");
55             }
56         mobility: <mobilityType> like IMobility {
57             @display("p=50,175;is=s");
58         }
59         tcpApp[numTcpApps]: <> like ITCPApp {
60             @display("p=175,50,row");
61         }
62         tcp: TCP if numTcpApps>0 {
63             @display("p=175,150");
64         }
65         udpApp[numUdpApps]: <> like IUdpApp {
66             @display("p=325,50,row");
67         }
68         udp: UDP if numUdpApps>0 {
69             @display("p=325,150");
70         }
71         /// NOTE: instance must be named "nic"
72         nic: LteNicUe if !d2dCapable {
73             nodeType = nodeType;
74             d2dCapable = d2dCapable;
75             @display("p=250,407");
76         }
77         /// NOTE: instance must be named "nic"
78         nic: LteNicUeD2D if d2dCapable {
79             nodeType = nodeType;
80             d2dCapable = d2dCapable;
81             @display("p=250,407");
82         }
83         /// network layer

```

```

84     networkLayer: <networkLayerType> like INetworkLayer {
85         parameters:
86             @display("p=250,258");
87     }
88     connections allowunconnected:
89     /// Internal TCP/UDP applications connections with IP stack
90
91     for i=0..numTcpApps-1 {
92         tcpApp[i].tcpOut → tcp.appIn++;
93         tcpApp[i].tcpIn ← tcp.appOut++;
94     }
95
96     tcp.ipOut → networkLayer.transportIn++ if numTcpApps>0;
97     tcp.ipIn ← networkLayer.transportOut++ if numTcpApps>0;
98
99     for i=0..numUdpApps-1 {
100         udpApp[i].udpOut → udp.appIn++;
101         udpApp[i].udpIn ← udp.appOut++;
102     }
103
104     udp.ipOut → networkLayer.transportIn++ if numUdpApps>0;
105     udp.ipIn ← networkLayer.transportOut++ if numUdpApps>0;
106
107     nic.radioIn ← radioIn;
108
109     networkLayer.ifOut++ → nic.upperLayerIn;
110     networkLayer.ifIn++ ← nic.upperLayerOut;
111 }

```

Outras codificações de módulos e arquitetura da rede estão inseridas no Apêndice A para melhor vislumbrar como se dá seu funcionamento.

Assim, explicaremos a seguir a configuração de inicialização da rede que foi realizada para as simulações.

5.1.0.1 Configuração da Rede

O arquivo de configuração está dividido em diferentes seções, identificadas pela utilização de uma palavra identificadora dentro de colchetes. A primeira seção abaixo, "General", faz as configurações básicas do simulador, como indicar caminhos para buscar e guardar informações, além das configurações básicas da simulação, como indicar duração do experimento e quantidade de repetições. Também é nela que são especificados parâmetros de canal, da camada física e de mobilidade.

```

1
2 [General]
3 image-path=../../images
4 tkenv-plugin-path = ../../inet/etc/plugins
5 output-scalar-file-append = false
6 debug-on-errors = false
7 tkenv-default-config =
8 sim-time-limit=1000s
9 warmup-period=0s
10 repeat = 10
11 **.routingRecorder.enabled = false
12

```

```

13 ##### Statistics #####
14 output-scalar-file = ${resultdir}/${configname}/${repetition}.sca
15 output-vector-file = ${resultdir}/${configname}/${repetition}.vec
16 seed-set = ${repetition}
17 **.vector-recording = true
18
19 ##### Channel parameters #####
20 **.channelControl.pMax = 10W
21 **.channelControl.alpha = 1.0
22 **.channelControl.carrierFrequency = 2100e+6Hz
23
24 ##### PhyLayer parameters #####
25 **.nic.phy.channelModel=xmldoc("config_channel.xml")
26 **.feedbackComputation = xmldoc("config_channel.xml")
27
28 ##### Mobility parameters #####
29 **.mobility.constraintAreaMinZ = 0m
30 **.mobility.constraintAreaMaxZ = 0m
31
32 ##### Deployer parameters #####
33 # UEs attached to eNB
34 **.fbDelay = 1
35
36 ##### AMC MODULE PARAMETERS #####
37 **.rbAllocationType = "localized"
38 **.deployer.numRbDl = 50
39 **.deployer.numRbUl = 50
40 **.numBands = 50

```

A seção “MultiplePairs” estende a arquitetura de rede "SingleCell", já explicada anteriormente, onde há uma só célula LTE. Aqui, são adicionados a esta arquitetura configurações de posicionamento e mobilidade da eNB e dos UEs, bem como quantidade destes e identificadores.

```

1
2 [ Config MultiplePairs]
3 network=lte.simulations.networks.SingleCell_D2D
4
5 ### eNodeBs configuration ###
6 *.eNodeB.mobility.initFromDisplayString = false
7 *.eNodeB.mobility.initialX = 300m
8 *.eNodeB.mobility.initialY = 300m
9
10 ### UEs configuration ###
11 *.numUeCell = 0
12 #*.numUeD2DTx = ${numPairs=5,20,50}
13 *.numUeD2DTx = ${numPairs=40}
14 *.numUeD2DRx = ${numPairs}
15
16 *.ue[*].macCellId = 1
17 *.ue[*].masterId = 1
18 *.ue[*].mobility.initFromDisplayString = false
19 *.ue[*].mobilityType = "LinearMobility"
20 *.ue[*].mobility.speed = 2mps
21
22 # Place D2D endpoints far from the eNodeB (~50m)
23 # and close to each other
24 *.ueD2D[*].mobility.initialX = uniform(290m,310m)
25 *.ueD2D[*].mobility.initialY = uniform(340m,350m)

```

Uma vez já criada a rede, a seção "Config MultiplePairs-UDP-Infra", que estende

uma arquitetura de rede de múltiplos pares já definida no simulador, trata de prover as configurações básicas necessárias para o funcionamento do protocolo e aplicação que serão utilizados.

Podemos verificar a configuração de uma aplicação do tipo VoIP (*Voice over IP*) e sua habilitação para todos os nós transmissores e receptores da rede.

```

1
2 [ Config MultiplePairs-UDP-Infra ]
3 extends=MultiplePairs
4
5 ### Traffic configuration ###
6 *.ueD2D[*].numUdpApps = 1
7
8 # Traffic between pairs of UEs (e.g. ueD2DTx[0] —> ueD2DRx[0])
9 # Transmitters
10 *.ueD2DTx[*].udpApp[*].typename = "VoIPSender"
11 *.ueD2DTx[*].udpApp[*].localPort = 3088+ancestorIndex(0)
12 *.ueD2DTx[*].udpApp[*].startTime = uniform(0s,0.02s)
13 *.ueD2DTx[*].udpApp[*].destAddress =
14     "ueD2DRx[" + string(ancestorIndex(1)) + "]"
15 *.ueD2DTx[*].udpApp[*].destPort = 1000
16 # Receivers
17 *.ueD2DRx[*].udpApp[*].typename = "VoIPReceiver"
18 *.ueD2DRx[*].udpApp[*].localPort = 1000
19
20 # Disable D2D for the eNodeB and the UEs
21 *.eNodeB.d2dCapable = false
22 *.ueD2D[*].d2dCapable = false
23 **.*.amcMode = "AUTO"

```

Agora, a seção "Config MultiplePairs-UDP-D2D" estende as configurações da seção anterior, e adiciona a habilitação da comunicação D2D. A estação base e os nós são habilitados para comunicação D2D, de acordo com os parâmetros de CQI. Um nós decidirá por estabelecer comunicação por D2D quando seu CQI atingir o valor estabelecido nesta seção.

Aqui, além de habilitar os UEs à comunicação D2D, também é necessário estabelecer quais serão os pares. Portanto, cada vez que se quiser mudar a quantidade de nós encaminhadores e quais serão estes, é necessário realizar esta configuração nesta seção.

Neste arquivo, por exemplo, todos os nós estão habilitados a executar comunicação D2D, no entanto, os pareamentos foram estabelecidos apenas entre os nós Tx e Rx de 1 a 10.

```

1
2 [ Config MultiplePairs-UDP-D2D ]
3 extends=MultiplePairs-UDP-Infra
4
5 # Enable D2D for the eNodeB and the UEs involved
6 # in direct communications
7 *.eNodeB.d2dCapable = true
8 *.ueD2D[*].d2dCapable = true
9 **.*.amcMode = "D2D"
10

```



```

11 # — Set the D2D peering capabilities — #
12 string(ancestorIndex(1)) + "]"
13 *.ueD2DTx[1..10].nic.d2dPeerAddresses =
14     "ueD2DRx[" + string(ancestorIndex(1)) + "]"
15
16 # — Select CQI for D2D transmissions — #
17 *.eNodeB.nic.phy.enableD2DCqiReporting = true
18 **.usePreconfiguredTxParams = false
19 **.d2dCqi = 7

```

5.1.1 Configuração das simulações

O objetivo das simulações é avaliar o consumo de energia em uma rede LTE com a presença de UEs que podem se comunicar diretamente. Para tanto, dos resultados das simulações consideramos a quantidade de mensagens enviadas e recebidas para calcular a quantidade total de mensagens, bem como a energia consumida por nó e no cenário, em diferentes cenários: sem UEs encaminhadores, com alguns UEs encaminhadores e com todos UEs encaminhando.

O encaminhamento é o processo de encaminhamento feito por um nó habilitado pela comunicação D2D a receber uma mensagem da estação base e encaminhá-la a um nó ou vice-versa.

Os parâmetros adotados para realizar as simulações estão especificados na Tabela 5.1. Para cada um dos experimentos foram definidos 6 cenários. Para cada cenário foram executadas 10 rodadas, como pode ser verificado na Tabela 5.2.

Tabela 5.1: Parâmetros de simulação.

Parâmetro	Valor
Largura da Banda	5 MHz
Frequência da Portadora	2 GHz
Modelo de perda	ITU Urban Macro
Modelo de desvanecimento	Jakes
Modelo de mobilidade	Linear Mobility (Modelo do OMNeT++)
Velocidade	2 m/s
Ruído	5 dB
Distância eNB - UE	500, 1000, 1500, 2000 m
Tempo de simulação	1000 s

A rede foi configurada com 80 dispositivos móveis, distribuídos de forma estacionária

e aleatória em uma área densa, de 200 m^2 . Consideramos uma rede com uma célula, como na classificação DR-OC [36] apresentada, ou seja, a comunicação D2D é dentro da banda LTE, gerenciada pela estação base. A Figura 5.2 ilustra como é, neste caso, a diferença da rede sem e com encaminhador. Os dispositivos se comunicaram de forma simultânea, utilizando, em alguns cenários, aplicação VoIP sobre protocolo UDP e, em outros, aplicação TCP. Os pares de comunicação são definidos para cada experimentos e serão explicados a seguir ao detalhar sobre cada simulação. A escolha do encaminhador é feita a partir de um algoritmo de cálculo de CQI, que prefere aquele de melhor CQI.

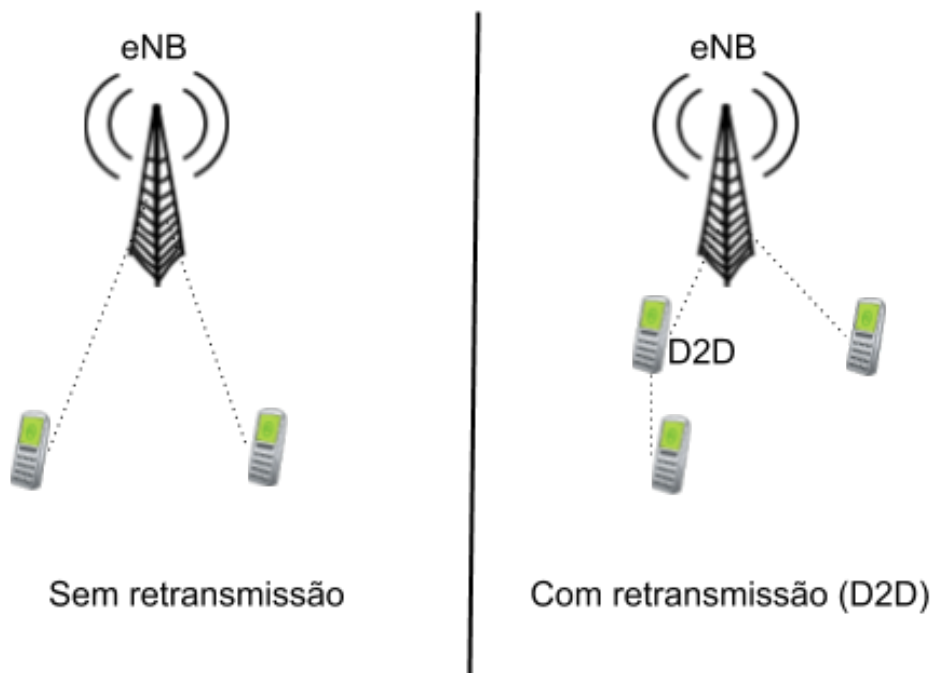


Figura 5.2: Esquema de comunicação com e sem encaminhamento.

Tabela 5.2: Quantidades de experimentos.

Experimento	Nós UEs	Nós encaminhadores (D2D)
1	80	40 (100%)
2	80	30 (75%)
3	80	20 (50%)
4	80	10 (25%)
5	80	04 (10%)
6	80	00 (0%)

Foi escolhida a utilização do modelo de mobilidade *Linear Mobility*. Este é um modelo aleatório para definir de forma linear a movimentação dos usuários móveis e também sua velocidade de forma constante [41]. O funcionamento padrão do modelo é o seguinte: os nós são posicionados aleatoriamente no cenário, com um ângulo de inicialização também aleatório. Iniciada a simulação, todos os nós se movem por uma linha reta com uma velocidade constante. Eles seguem até atingirem a área de limite definida e então, adquirem um novo ângulo, também aleatório, para traçar nova reta.

5.2 Resultados

Para avaliar o consumo de energia com presença de nós que podem se comunicar usando D2D, definem-se diferentes experimentos com diferentes configurações. Variam-se o tipo de aplicação, o protocolo de transporte, velocidade dos nós e o tamanho dos pacotes. Os experimentos realizados neste trabalho são descritos a seguir e detalhados adiante:

1. Transmissão *multicast* usando UDP à partir do Nó [0] para outros nós da rede, com velocidade de 2 m/s;
2. Comunicação *unicast* utilizando protocolo TCP com transmissão dos nós Tx para Rx, com velocidade de 2 m/s;
3. Protocolo UDP com utilização de aplicação VoIP transmitindo de nós Tx para Rx, com velocidade de 1 m/s;
4. Semelhante ao anterior, com velocidade de 2 m/s;
5. Semelhante ao 3, mas com velocidade de 5 m/s;
6. *Streaming* de arquivo de vídeo com tamanho de 4 MB dos nós Tx para o Rx correspondente, utilizando pacotes de 256 B;
7. Semelhante ao anterior, mas utilizando pacotes de 512 B;
8. Semelhante ao 6, alterando o tamanho dos pacotes para 1024 B.

A Tabela 5.3 aponta como foram configurados os casos base de cada experimento. Nestes cenários, não havia encaminhamento, apenas os protocolos e aplicações apontados, utilizando a comunicação infraestruturada convencional. Já na Tabela 5.4 detalhamos as

variações realizadas em cada experimento em função da quantidade de UEs Tx, Rx e encaminhadores. Para os Experimentos 3, 4 e 5, são executados os 6 cenários em cada um (um cenário sem encaminhamento e outros cinco com encaminhamento), assim como nos experimentos 6, 7 e 8 também são executados os 6 cenários especificados. Os resultados dessas simulações servem para comparar os resultados obtidos com a comunicação D2D.

Tabela 5.3: Descrição dos experimentos sem comunicação D2D.

Aplicação	Transporte	Velocidade	Tam. Pacotes
1. <i>Multicast</i>	UDP	2 m/s	10 B
2. <i>Unicast</i>	TCP	2 m/s	-
3. VoIP	UDP	1 m/s	40 B
4. VoIP	UDP	2 m/s	40 B
5. VoIP	UDP	5 m/s	40 B
6. Vídeo	UDP	2 m/s	256 B
7. Vídeo	UDP	2 m/s	512 B
8. Vídeo	UDP	2 m/s	1024 B

Tabela 5.4: Descrição dos experimentos com comunicação D2D.

	Cenário	Qtd Tx	Qtd Rx	D2D
1. <i>Multicast</i> (velocidade: 2 m/s)	1	0	0	4
	2	0	0	10
	3	0	0	20
	4	0	0	30
	5	0	0	40
2. TCP (velocidade: 2 m/s)	1	40	40	Tx[0..03]
	2	40	40	Tx[0..09]
	3	40	40	Tx[0..19]
	4	40	40	Tx[0..29]
	5	40	40	Tx[0..39]
3. UDP VoIP (velocidade: 1 m/s) e	1	40	40	Tx[0..03]
	2	40	40	Tx[0..09]
4. UDP VoIP (velocidade: 2 m/s) e	3	40	40	Tx[0..19]
	4	40	40	Tx[0..29]
5. UDP VoIP (velocidade: 5 m/s)	5	40	40	Tx[0..39]
6. UDP <i>Streaming</i> (pacotes: 256 B) e	1	4	4	Tx[0..03]
	2	10	10	Tx[0..09]
7. UDP <i>Streaming</i> (pacotes: 512 B) e	3	20	20	Tx[0..19]
	4	30	30	Tx[0..29]
8. UDP <i>Streaming</i> (pacotes: 1024 B)	5	40	40	Tx[0..39]

A seguir, apresentamos primeiro cada experimento em específico e, em seguida, discutiremos algumas comparações entre experimentos.

Os gráficos referente ao Tráfego de Dados em Bytes, de cada cenário, de todos os experimentos, foram inseridos ao final deste trabalho, no Apêndice B. Neste apêndice eles poderão ser consultados de forma mais detalhada, de forma que aqui focaremos na análise de consumo energético.

5.2.1 Aplicação *Multicast*

No caso base, somente o Nó [0] recebe e transmite pacotes para estação base. Os demais somente recebem cópias das mensagens, encaminhadas pelo Nó [0]. Neste cenário,

foram transmitidos 223009,2 B e recebidos 362696,2 B.

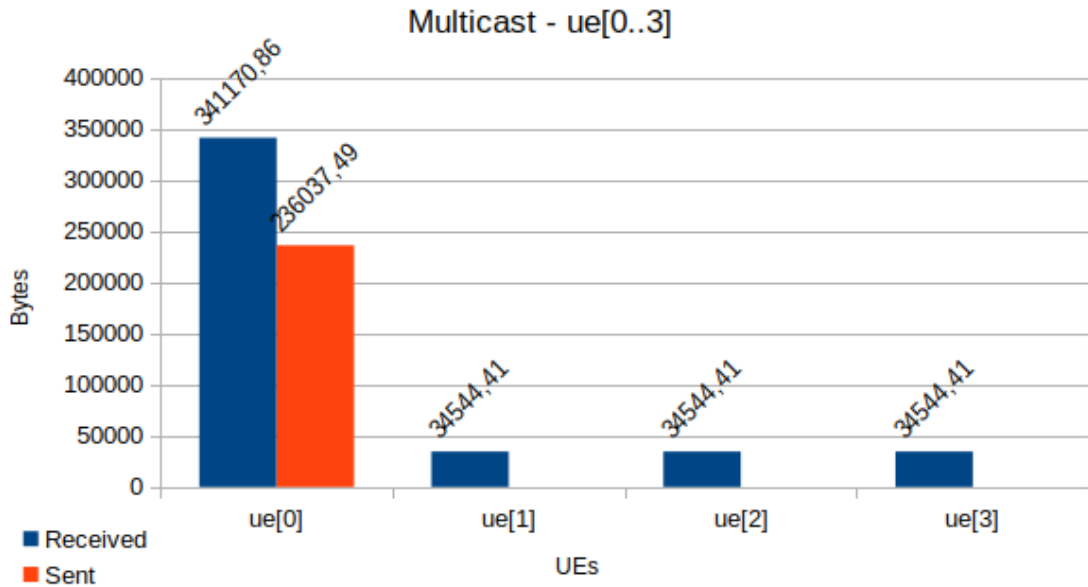


Figura 5.3: Número de bytes transmitidos e recebidos no experimento multicast com 4 UEs.

Já realizando encaminhamentos, para o cenário com 4 nós, demonstrado na Figura 5.3, o nó [0] envia 236037,49 B e recebe 341170,86 B. Os demais nós somente recebem dados que, por serem enviados em *multicast* têm todos o mesmo valor, de 34544,41 B.

Realizando a transformação energética, verificamos que neste experimento altera-se o consumo apenas se estiver sendo realizado o encaminhamento. Não importando se com 4, 10, 20, 30 ou 40 nós, o consumo é o mesmo em todos os cenários em que há encaminhamento. Diferindo apenas do cenário que não há, como se vê na Tabela 5.5.

Tabela 5.5: Consumo energético dos cenário do experimento *multicast*.

	Enviados ue[0]	Recebidos ue[0]	ue[1..n]
Caso Base (mW/Mbps)	0,098	0,019	00
Cenários D2D (mW/Mbps)	0,103	0,018	0,0002

Desta forma, na Figura 5.4, podemos verificar a evolução de consumo energético do experimento Multicast, primeiramente quando há 80 nós, mas nenhum participando do encaminhamento e depois os demais cenários onde os nós participam. Quanto mais encaminhadores são inseridos, maior o consumo energético.

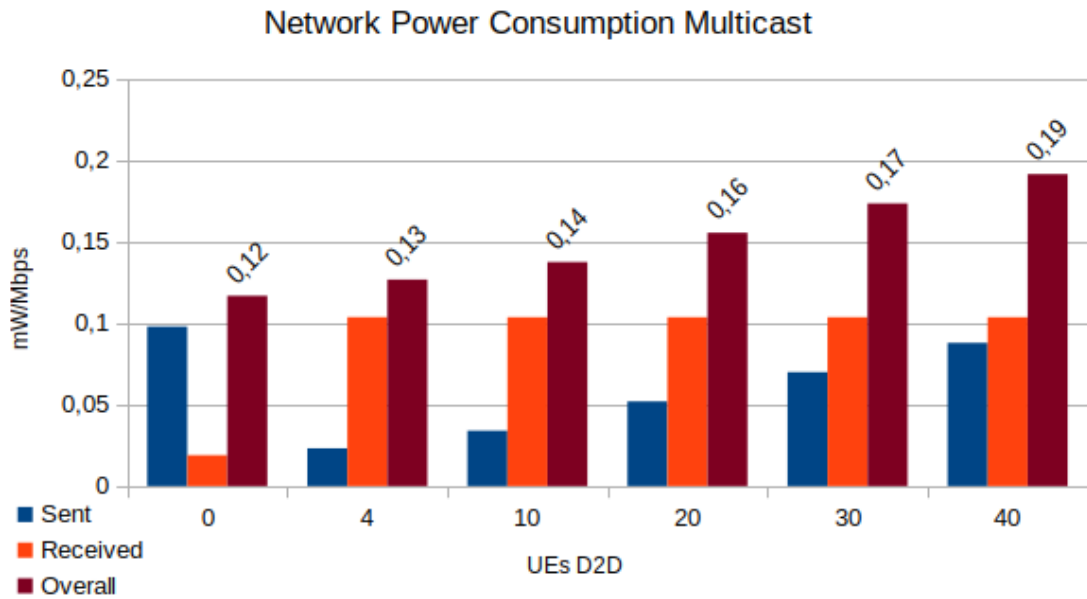


Figura 5.4: Consumo energético da rede no experimento multicast.

5.2.2 Experimento: TCP

No experimento denominado TCP, temos uma aplicação TCP, que abre uma conexão de um nó para a estação base, envia um dado número de bytes e em seguida encerra essa conexão. A velocidade de mobilidade utilizada foi de 2 m/s, como já mencionado na Tabela 5.3. Em todos os cenários deste caso, há 80 nós – 40 do tipo Tx e 40 do tipo Rx –, variando apenas a quantidade de encaminhadores: 0, 4, 10, 20, 30 e 40.

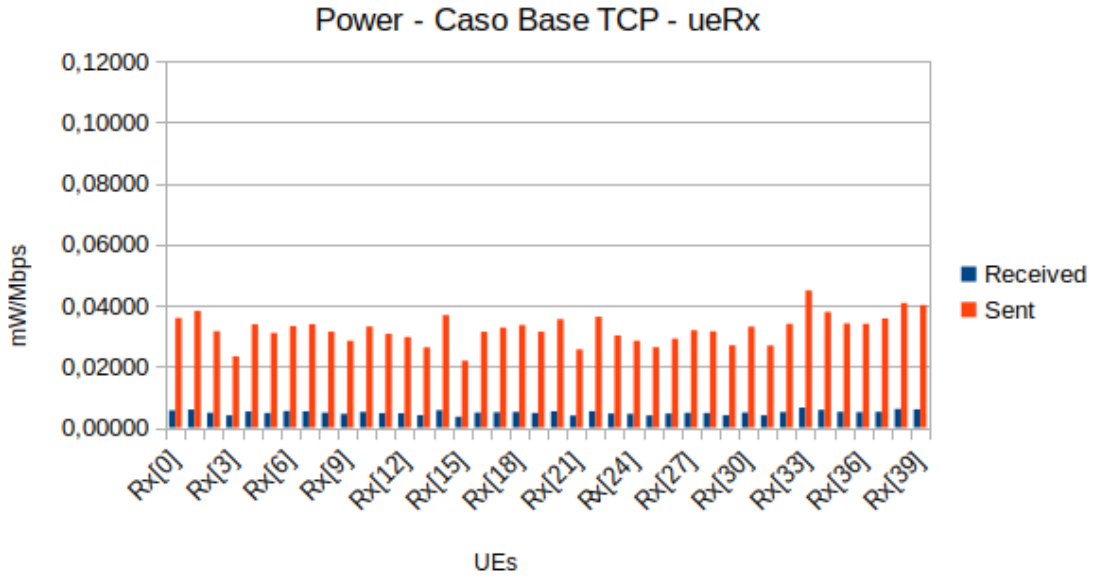


Figura 5.5: Consumo energético dos UEs Rx no experimento TCP.

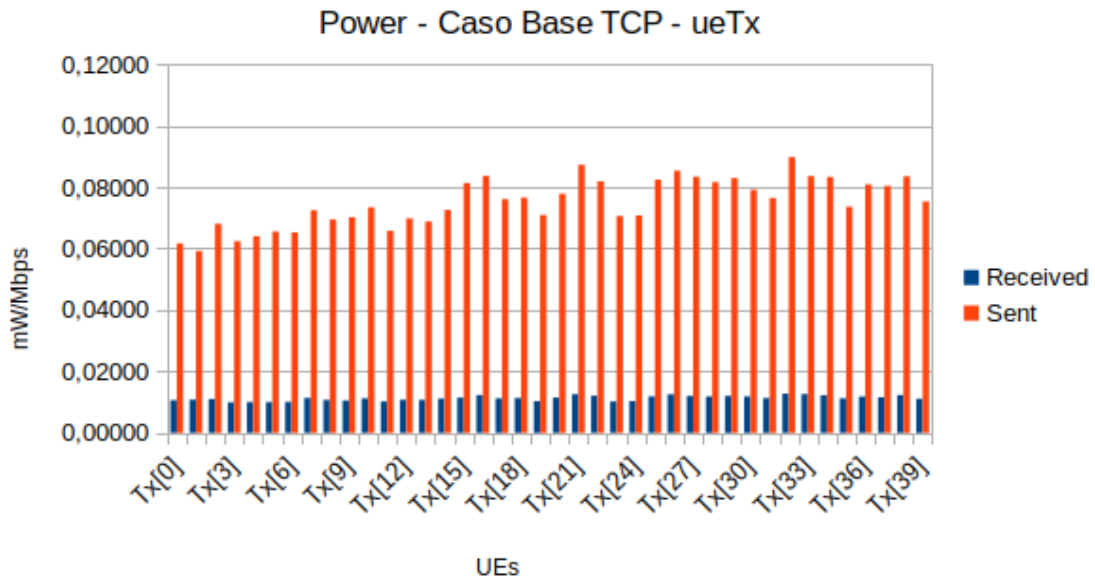


Figura 5.6: Consumo energético dos UEs Tx no experimento TCP.

O caso base deste experimento pode ser verificado na Figura 5.5, que demonstra o consumo dos nós Rx e na Figura 5.6, que apresenta os nós Tx.

Analisando os nós Tx, podemos perceber nas Figuras 5.7, 5.8, 5.9, 5.10 e 5.11, que o nó que obteve o maior consumo energético está sempre no grupo de encaminhadores. É possível perceber também que o consumo energético para a maioria dos nós encaminha-

dores é ligeiramente maior que para os demais. Repare que nestas figuras, assim como nas demais que contêm diferentes quantidades de encaminhadores, os nós encaminhadores são indicados no título do gráfico. Os encaminhadores serão identificados sempre de 0 até $n - 1$, sendo n a quantidade de encaminhadores naquele cenário.

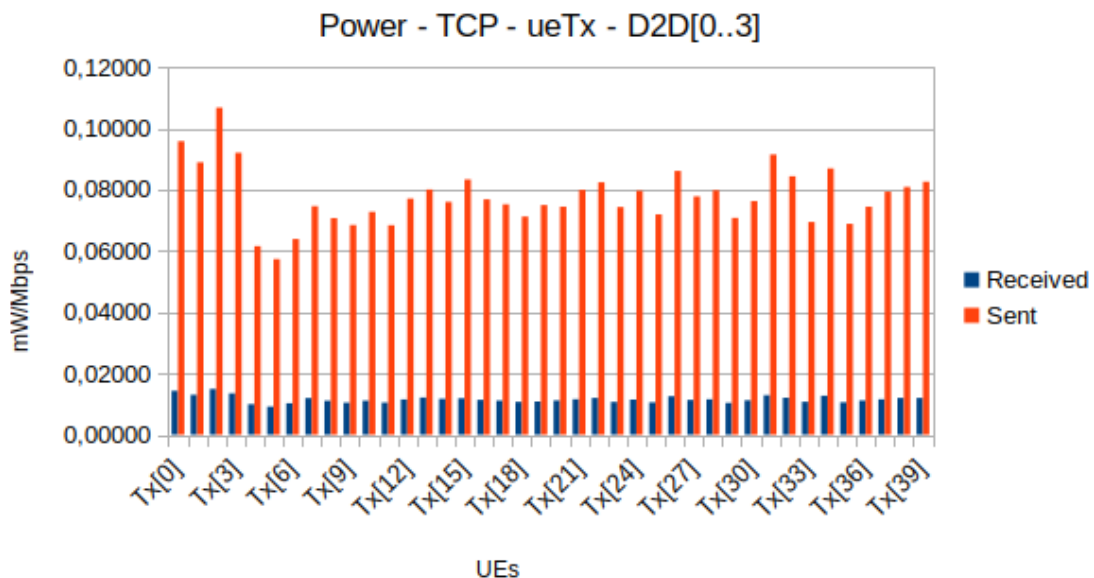


Figura 5.7: Consumo energético dos UEs Tx no experimento TCP com 4 encaminhadores.

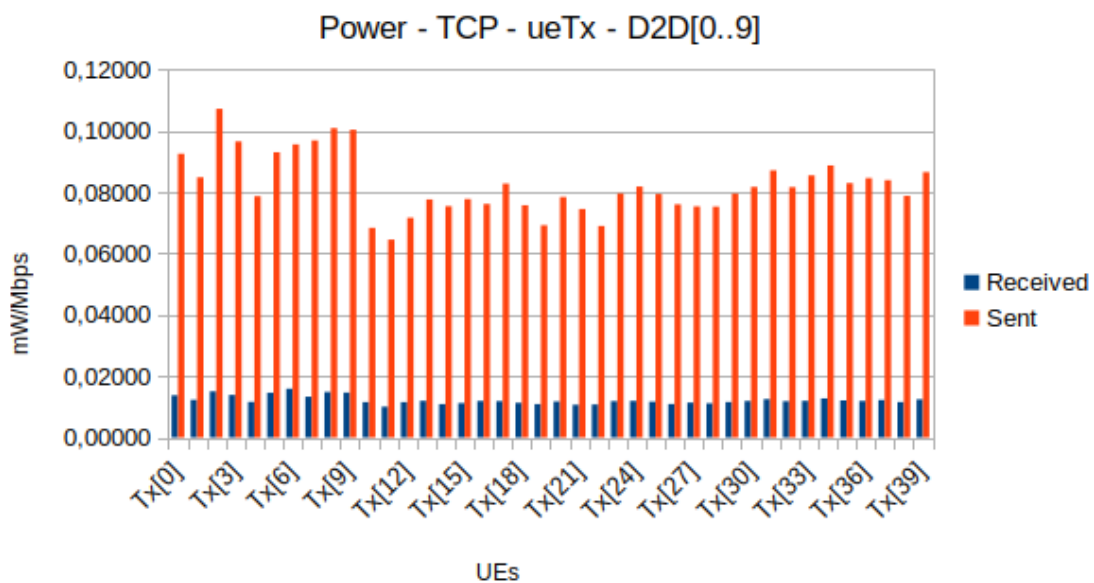


Figura 5.8: Consumo energético dos UEs Tx no experimento TCP com 10 encaminhadores.

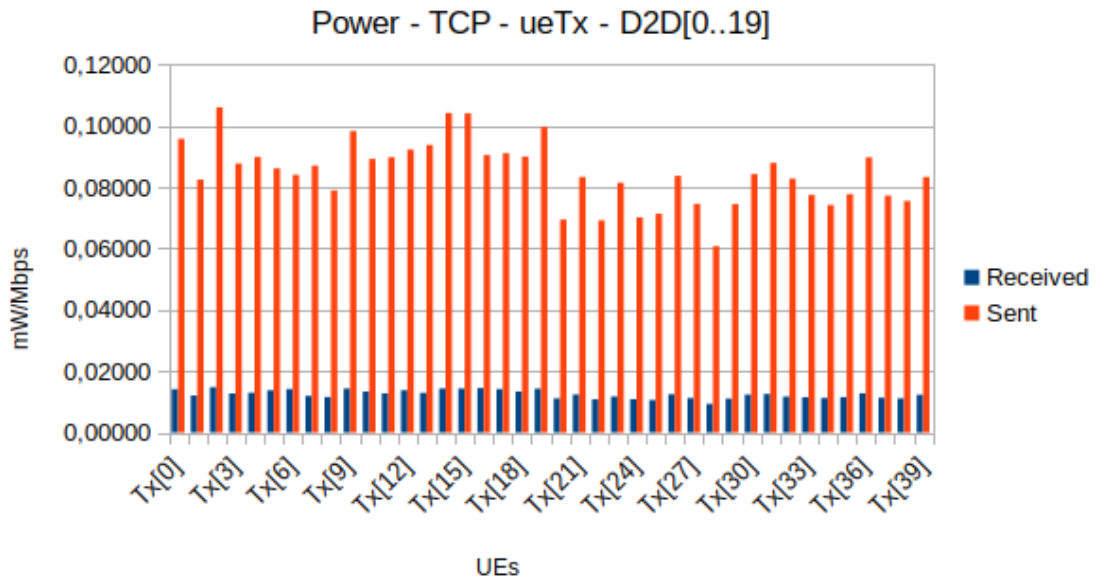


Figura 5.9: Consumo energético dos UEs Tx no Experimento TCP com 20 encaminhadores.

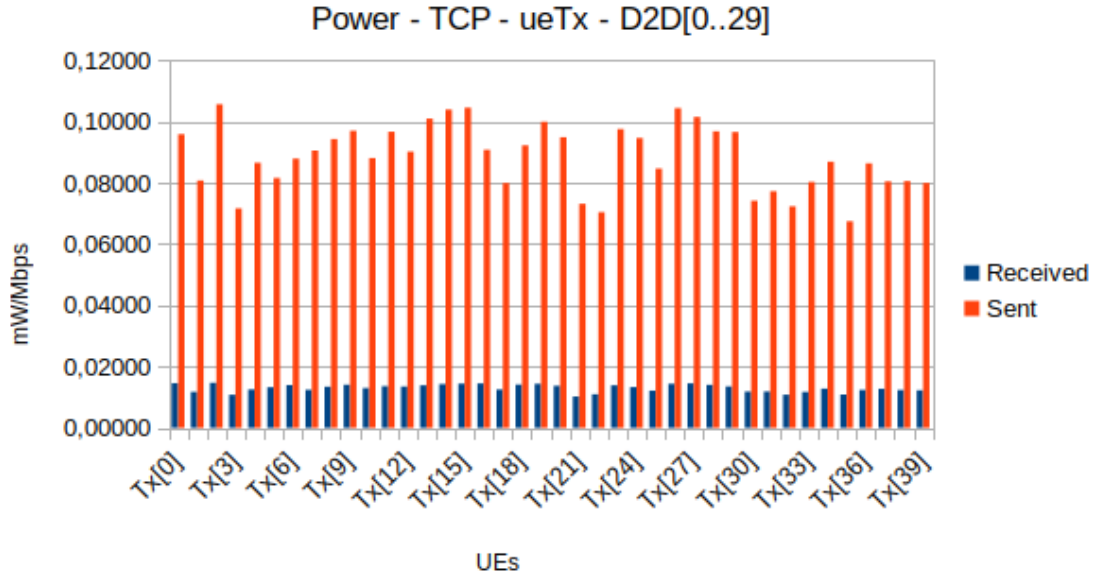


Figura 5.10: Consumo energético dos UEs Tx no experimento TCP com 30 encaminhadores.

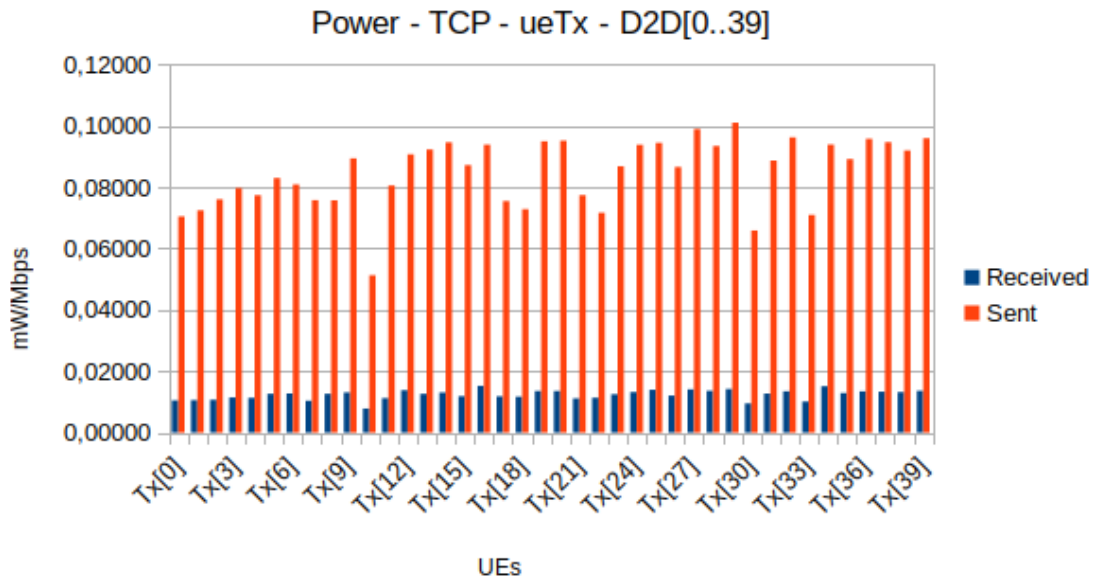


Figura 5.11: Consumo energético dos UEs Tx no experimento TCP com 40 encaminhadores.

As Figuras 5.12, 5.13, 5.14, 5.15 e 5.16 mostram o consumo energético dos nós Rx, nas quais os comportamentos são ligeiramente diferentes. Aqui também percebemos que há um aumento do consumo energético pelos nós que encaminham dados para outros nós, no entanto, em 2 cenários podemos notar os menores consumos energéticos do cenário em nós Rx, sendo: cenário com 4 encaminhadores, no qual o nó Rx[1] foi o que obteve o menor consumo; e cenário com 10 encaminhadores, no qual o nó que menos consumiu energia foi o Rx[9].

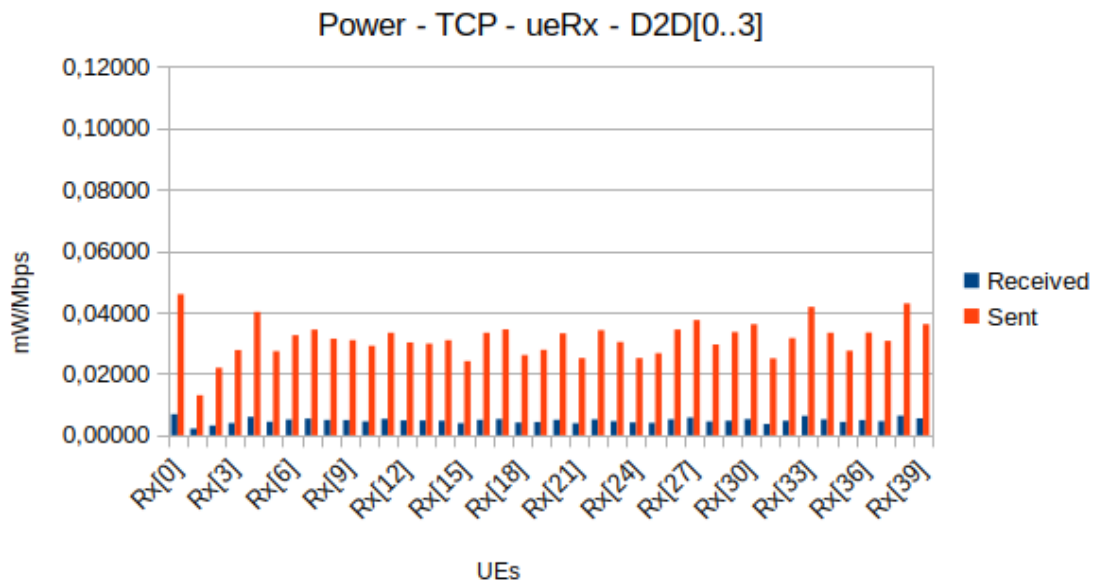


Figura 5.12: Consumo energético dos UEs Rx no experimento TCP com 4 encaminhadores.

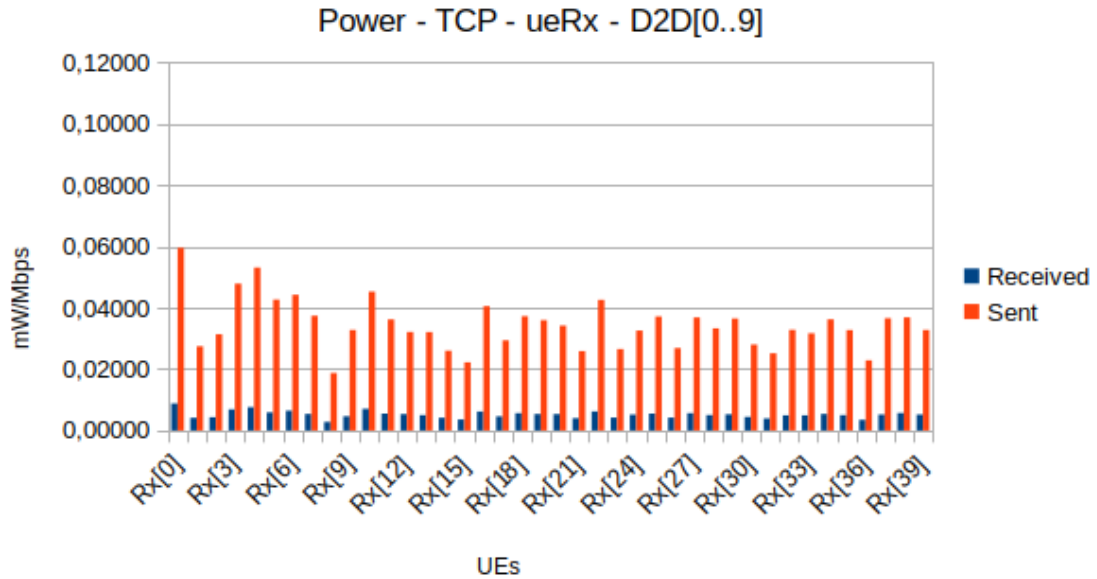


Figura 5.13: Consumo energético dos UEs Rx no experimento TCP com 10 encaminhadores.

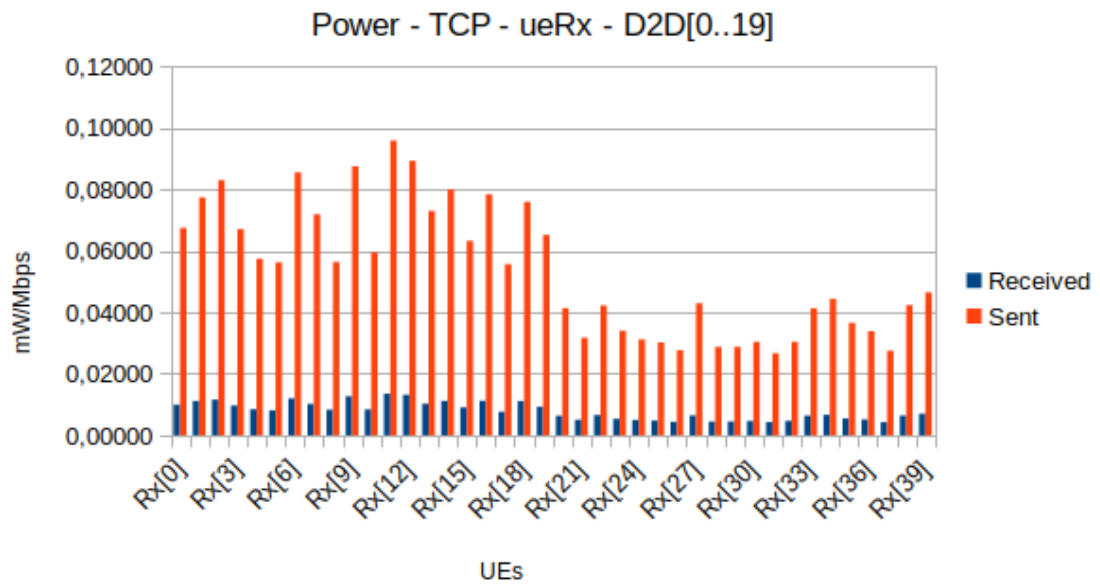


Figura 5.14: Consumo energético dos UEs Rx no experimento TCP com 20 encaminhadores.

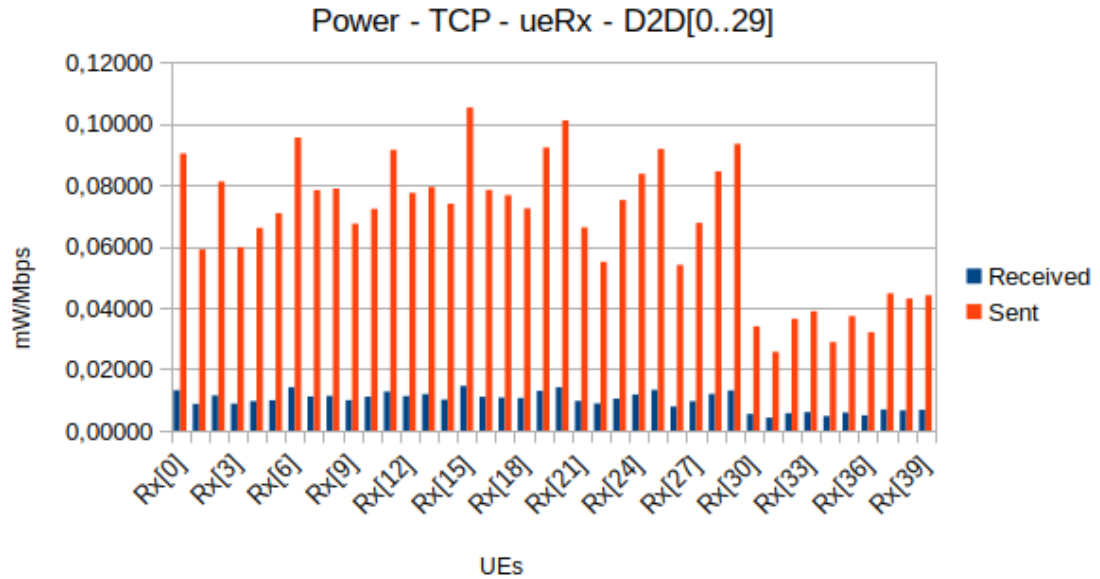


Figura 5.15: Consumo energético dos UEs Rx no experimento TCP com 30 encaminhadores.

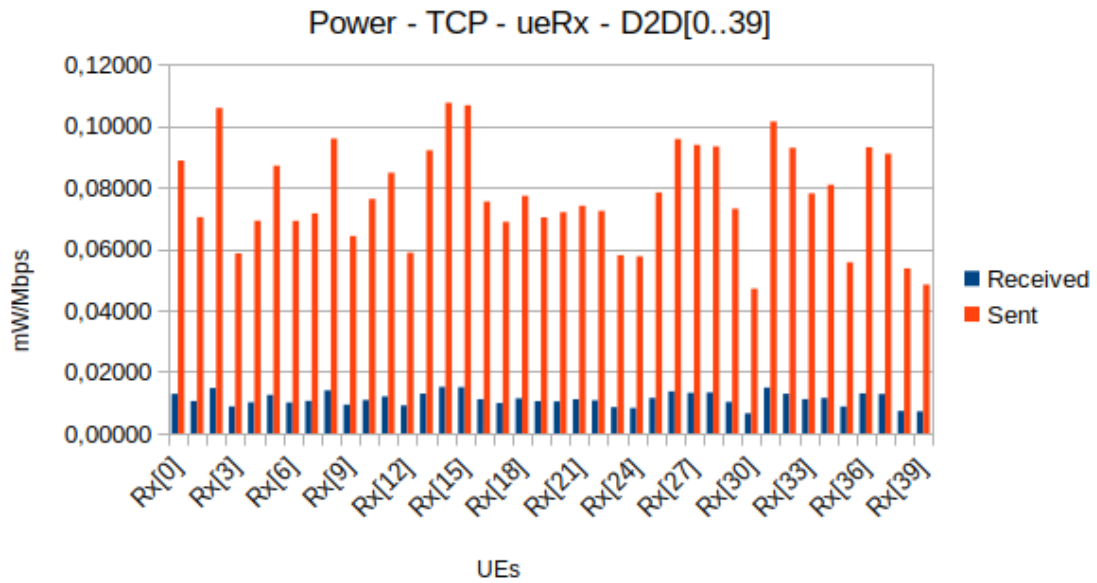


Figura 5.16: Consumo energético dos UEs Rx no experimento TCP com 40 encaminhadores.

Desta forma, é possível afirmar que no Experimento TCP há nós que participam do encaminhamento que, eventualmente, possuem consumo energético menor do que outros que não participam. No entanto, o consumo total da rede – considerando apenas os nós, mas não a estação base – nos cenários que há encaminhamento é sempre maior do que no caso base, onde não há. Conforme se aumenta o número de encaminhadores, o consumo energético também cresce junto. A Figura 5.17 mostra a soma de energia consumida para envio e recepção tanto dos nós Rx quanto Tx em cada cenário. Na cor vinho e com a descrição numérica, é mostrada ainda a quantidade total de energia consumida no cenário.

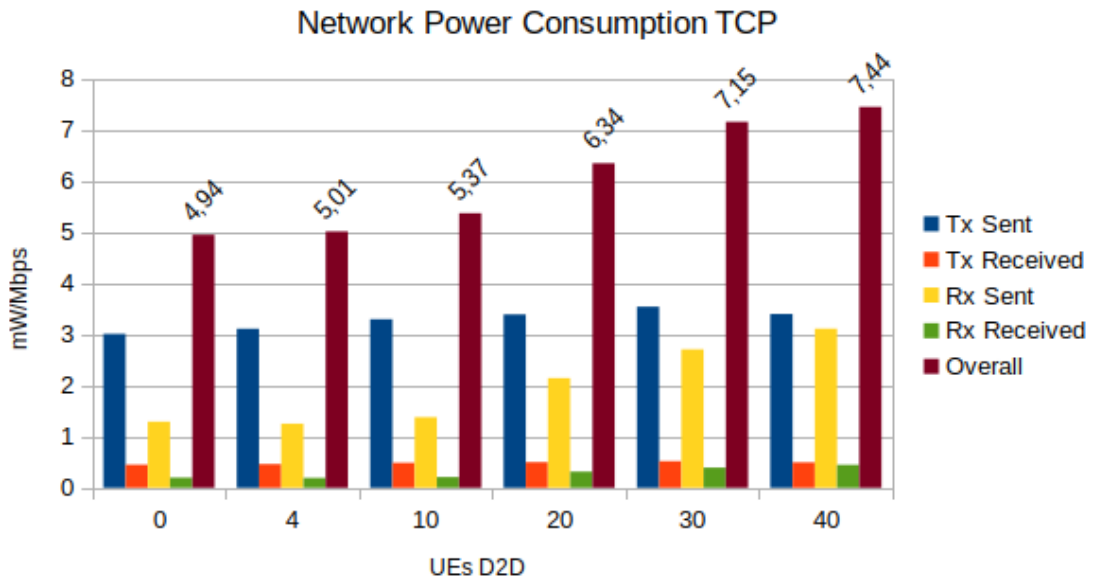


Figura 5.17: Consumo energético total da rede e dos UEs Rx e Tx no Experimento TCP para cada Cenário.

5.2.3 Experimento: UDP VoIP 2 m/s

Para simular um caso no qual pessoas se comunicam por chamada telefônica através da Internet, simulamos *streaming* de voz por meio da aplicação VoIP padrão do OMNeT++ [41]. A velocidade de mobilidade utilizada neste primeiro experimento foi de 2 m/s (metros por segundo), que segundo Mammeri [33] é compatível à velocidade de uma pessoa caminhando. Em seguida, testaremos o comportamento desta aplicação com diferentes velocidade de mobilidade. Alteraremos para 1 e 5 m/s, nos dois próximos experimentos, respectivamente. Em todos os cenários deste experimentos e dos próximos dois com VoIP, há 80 nós, variando a quantidade de encaminhadores em cada um dos 6 cenários: 0, 4, 10, 20, 30 e 40.

As Figuras 5.18 e 5.19 demonstram os gráficos do caso base deste experimento. O caso base é aquele com 0 nós encaminhadores, em que não há nenhum tipo de comunicação D2D, foi o que a rede mais consumiu energia total. Os experimentos possuem nós do tipo Tx, que encaminham a comunicação D2D da estação base para nós do tipo Rx e dos nós Rx para estação base. A origem da comunicação parte sempre de um nó Tx para um Rx, passando pela estação base e eventualmente, pelos encaminhadores. Para os nós do tipo Tx podemos notar uma pequena diferença de consumo energético, sendo que todos se situam entre 0,07 e 0,08 mW/Mbps para envio e entre 0,01 e 0,015 para recepção. Já

entre os nós Rx não há tal linearidade na energia consumida para envio, que varia de 0,002 a 0,0053 mW/Mbps. Já com a energia gasta para recepção a variação é menor e fica entre 0,0002 a 0,0006 mW/Mbps.

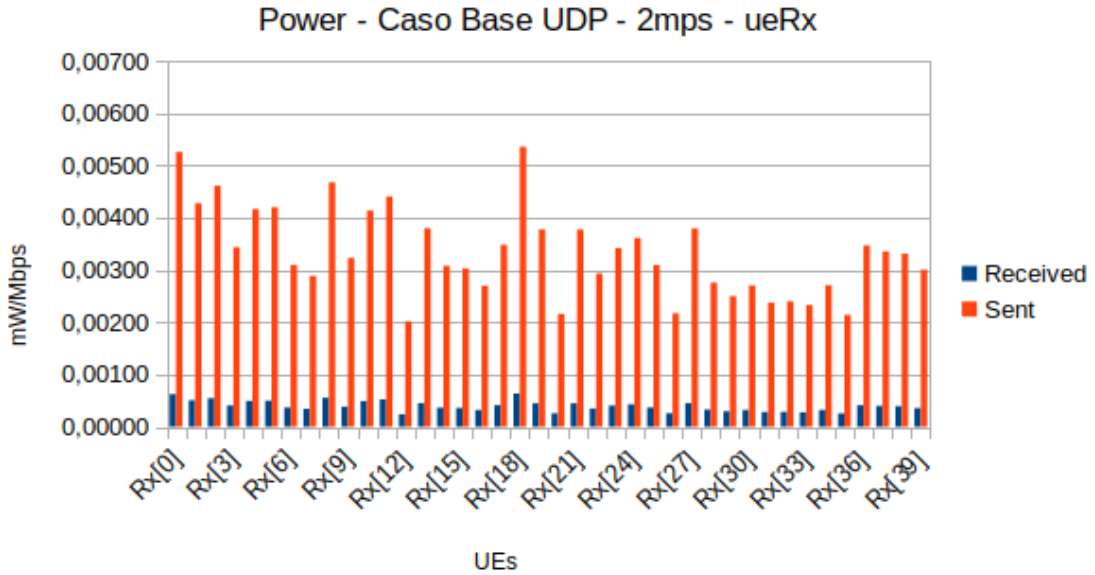


Figura 5.18: Consumo energético dos UEs Rx no experimento UDP 2 m/s.

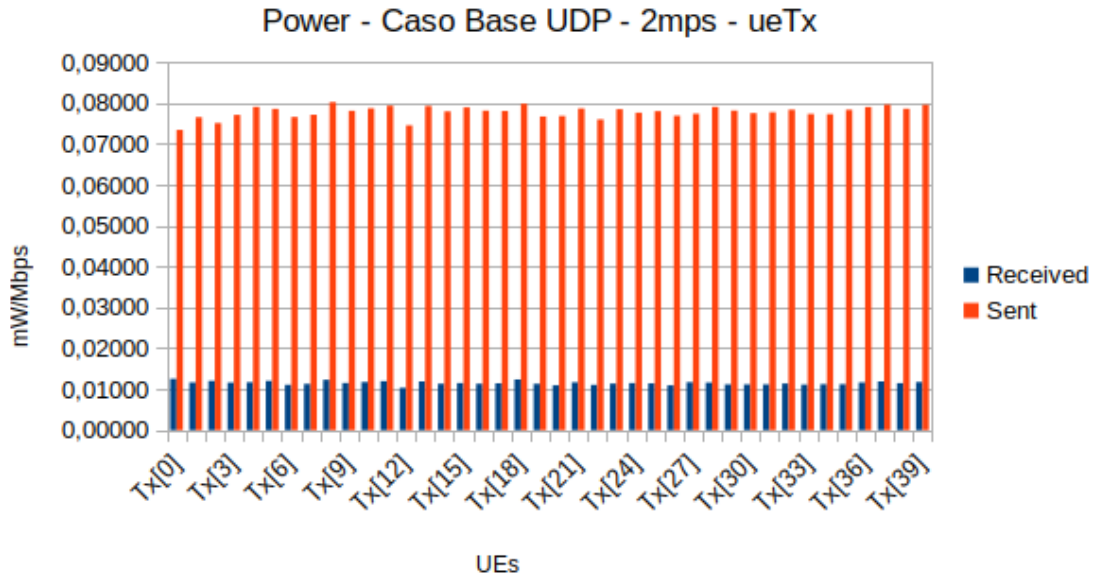


Figura 5.19: Consumo energético dos UEs Tx no experimento UDP 2 m/s.

As Figuras 5.20 e 5.21 mostram os gráficos resultantes do experimento com 10 nós Tx, encaminhadores, e 10 nós Rx. Nestes gráficos destacamos na linha lilás horizontal a

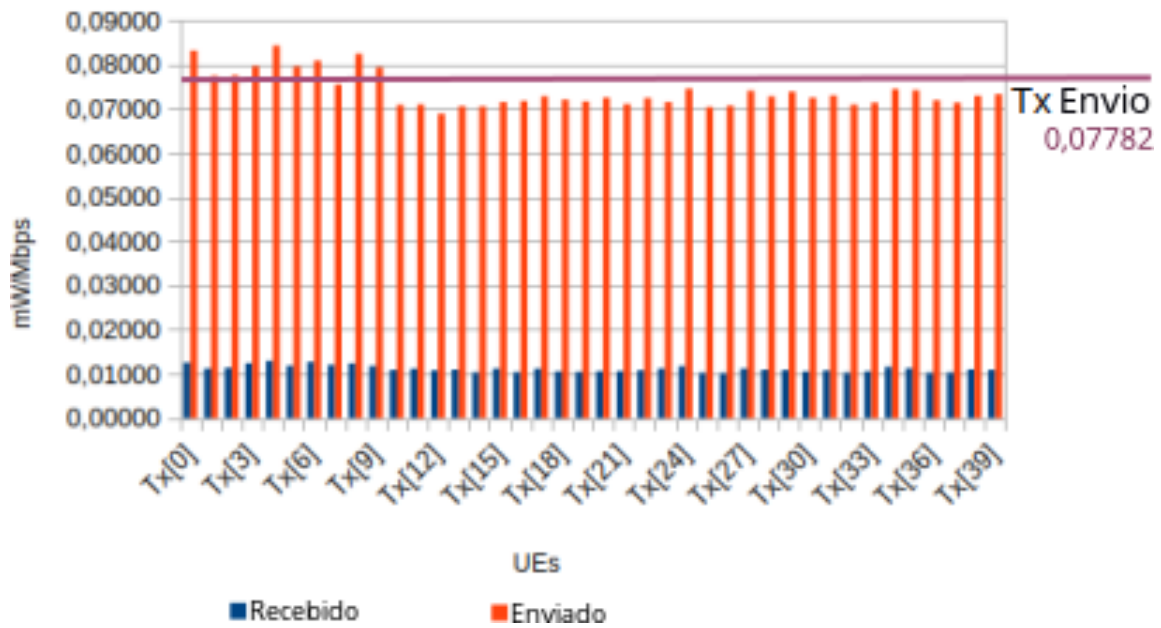


Figura 5.20: Consumo dos UEs Tx no experimento UDP 2 m/s com 10 encaminhadores.

média de envio do caso base para cada um dos cenários. Este cenário foi o que a rede mais consumiu energia total. Para os nós do tipo Tx podemos notar uma pequena diferença de consumo energético, sendo que todos se situam entre 0,07 e 0,08 mW/Mbps para envio e entre 0,01 e 0,015 para recepção. Já entre os nós Rx não há tal linearidade na energia consumida para envio, que varia de 0,002 a 0,0053 mW/Mbps. Já com a energia gasta para recepção a variação é menor e fica entre 0,0002 a 0,0006 mW/Mbps.

Analisando os demais cenários com encaminhamento, podemos verificar uma redução no consumo energético da rede, em relação ao caso base. É possível perceber que o Cenário com 30 encaminhadores é o que atinge os menores valores de consumo de energia, mas não conseguimos identificar um padrão de evolução entre os cenários.

As Figuras 5.22, 5.23, 5.24, 5.25 e 5.26 apresentam o consumo energéticos dos nós do tipo Tx em cada cenário, enquanto as Figuras 5.27, 5.28, 5.29, 5.30 e 5.31 apresentam o consumo dos nós Rx.

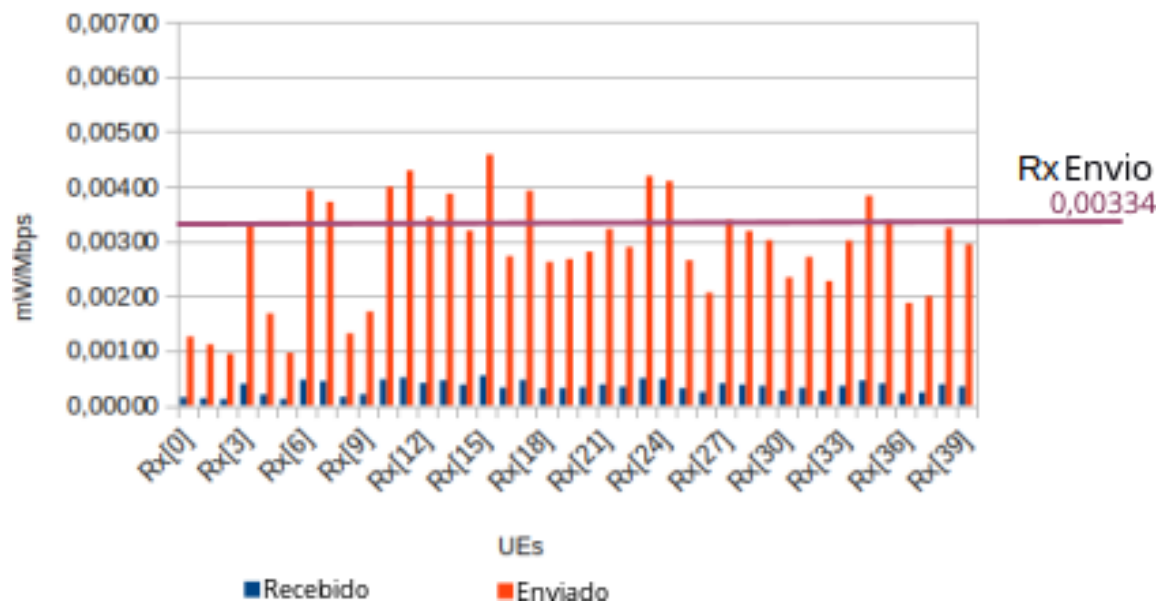


Figura 5.21: Consumo dos UEs Rx no experimento UDP 2 m/s com 10 encaminhadores.

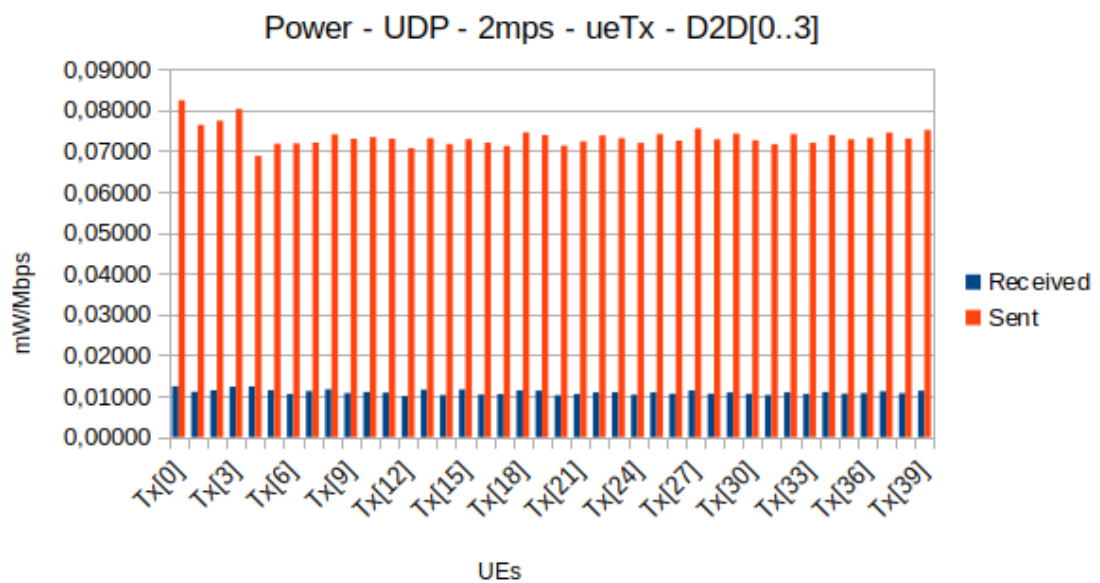


Figura 5.22: Consumo energético dos UEs Tx no experimento UDP 2 m/s com 4 encaminhadores.

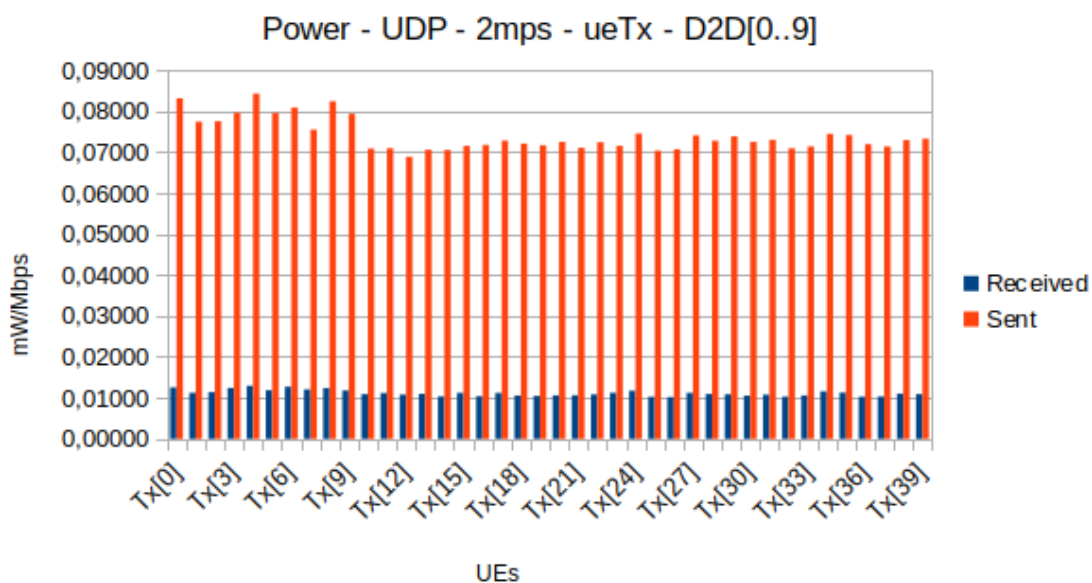


Figura 5.23: Consumo energético dos UEs Tx no experimento UDP 2 m/s com 10 enca-minhadores.

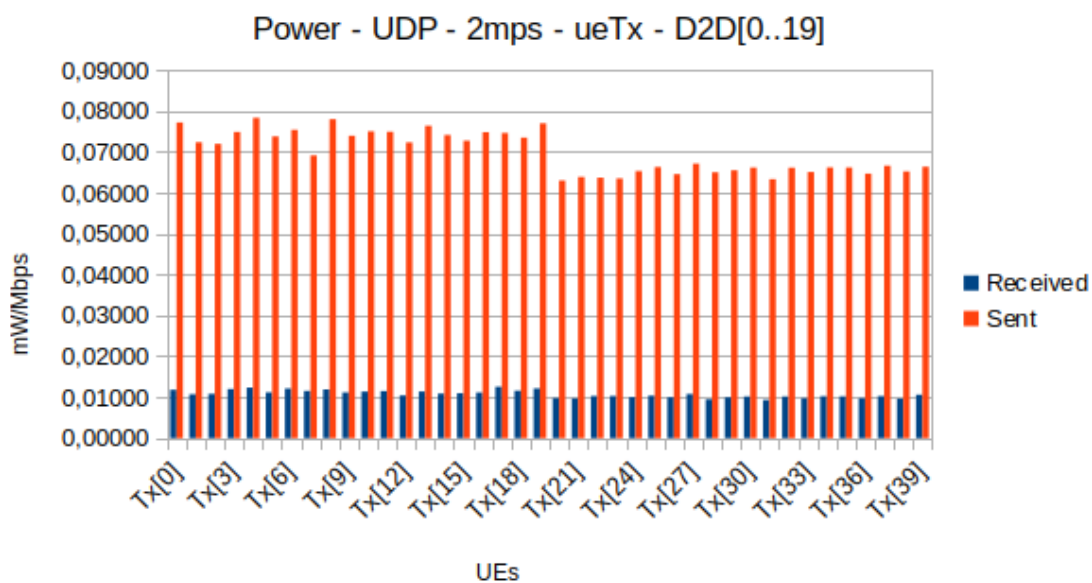


Figura 5.24: Consumo energético dos UEs Tx no experimento UDP 2 m/s com 20 enca-minhadores.

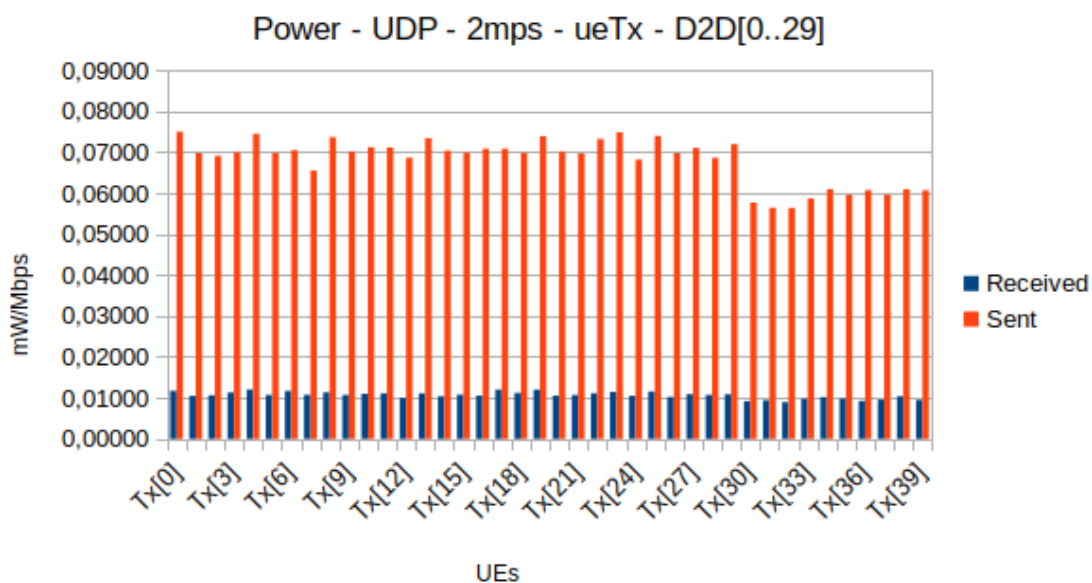


Figura 5.25: Consumo energético dos UEs Tx no experimento UDP 2 m/s com 30 enca-minhadores.

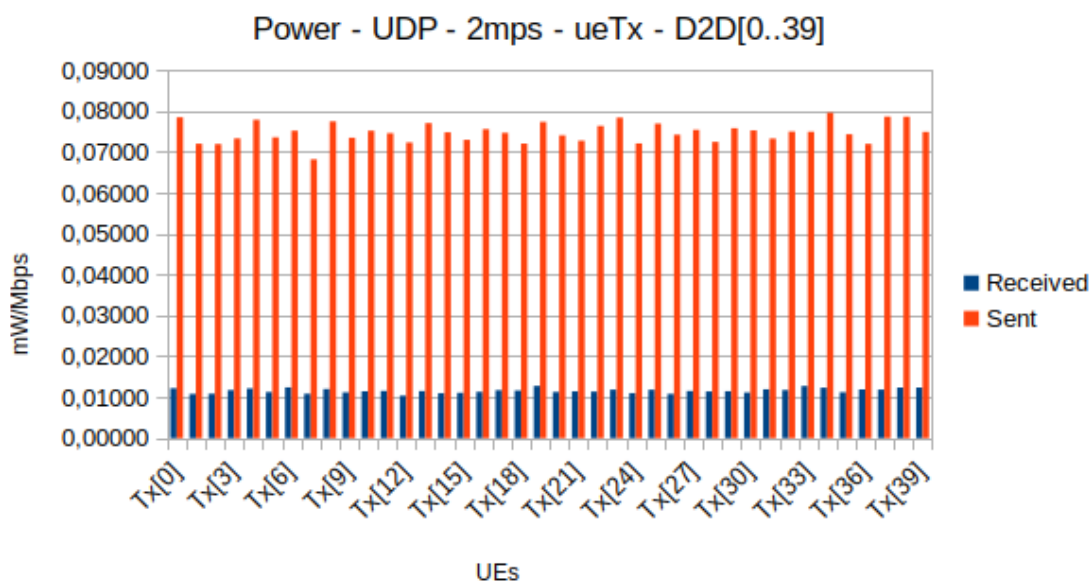


Figura 5.26: Consumo energético dos UEs Tx no experimento UDP 2 m/s com 40 enca-minhadores.

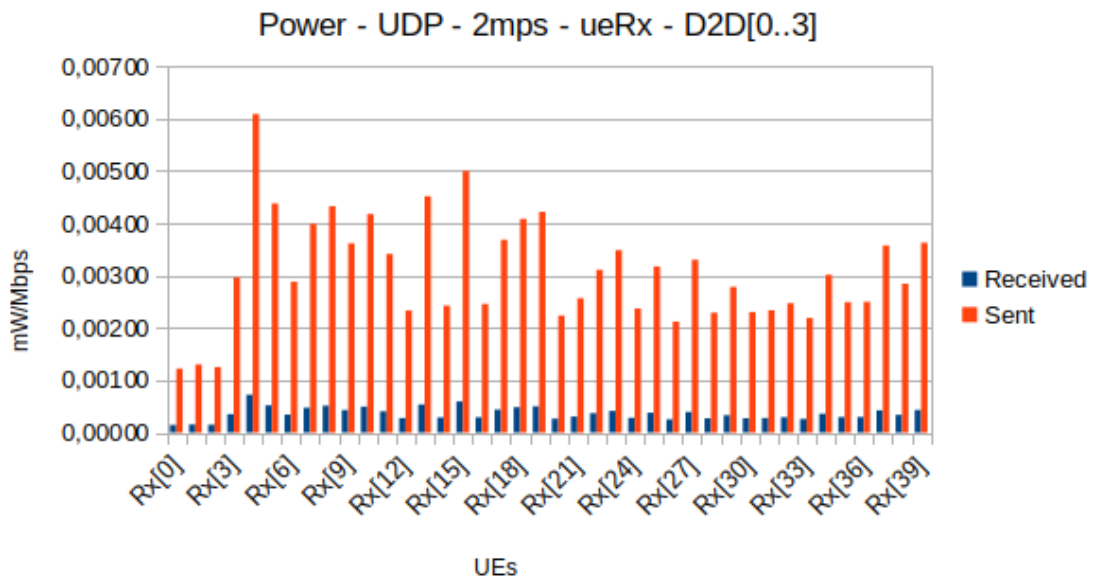


Figura 5.27: Consumo energético dos UEs Rx no experimento UDP 2 m/s com 4 enca-minhadores.

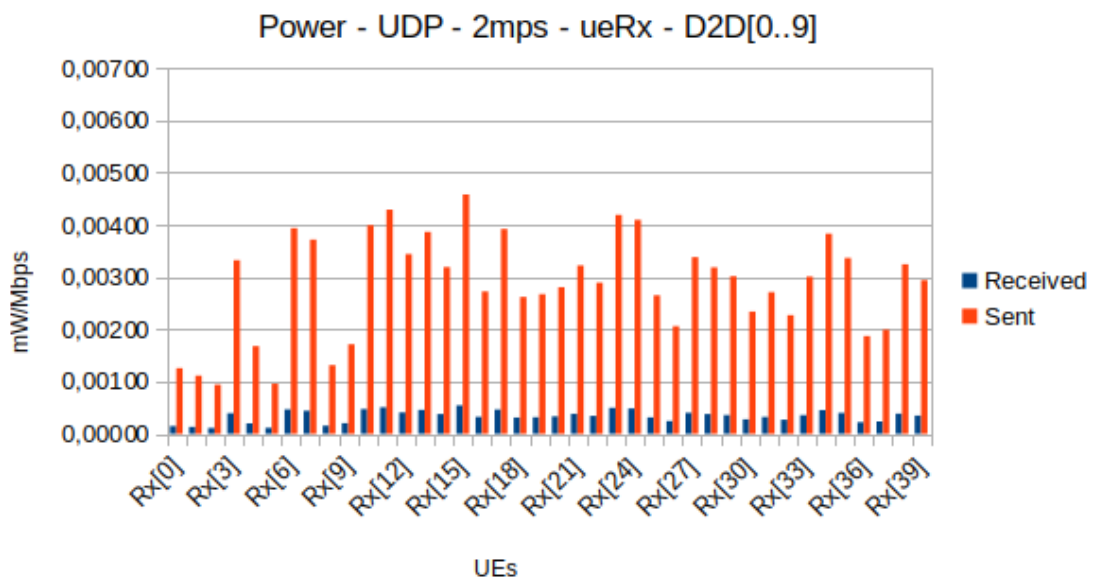


Figura 5.28: Consumo energético dos UEs Rx no experimento UDP 2 m/s com 10 enca-minhadores.

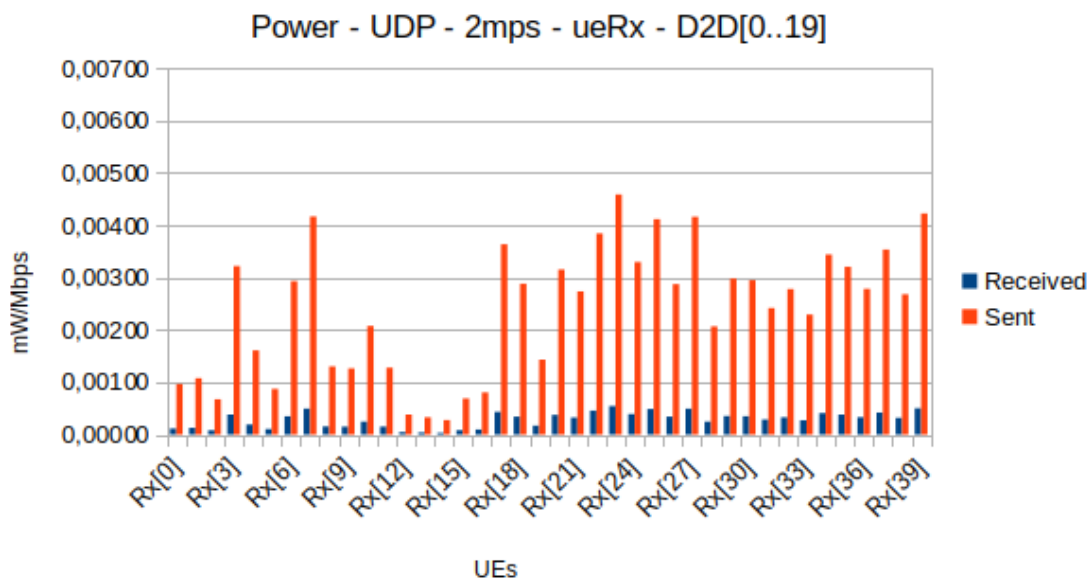


Figura 5.29: Consumo energético dos UEs Rx no experimento UDP 2 m/s com 20 enca-minhadores.

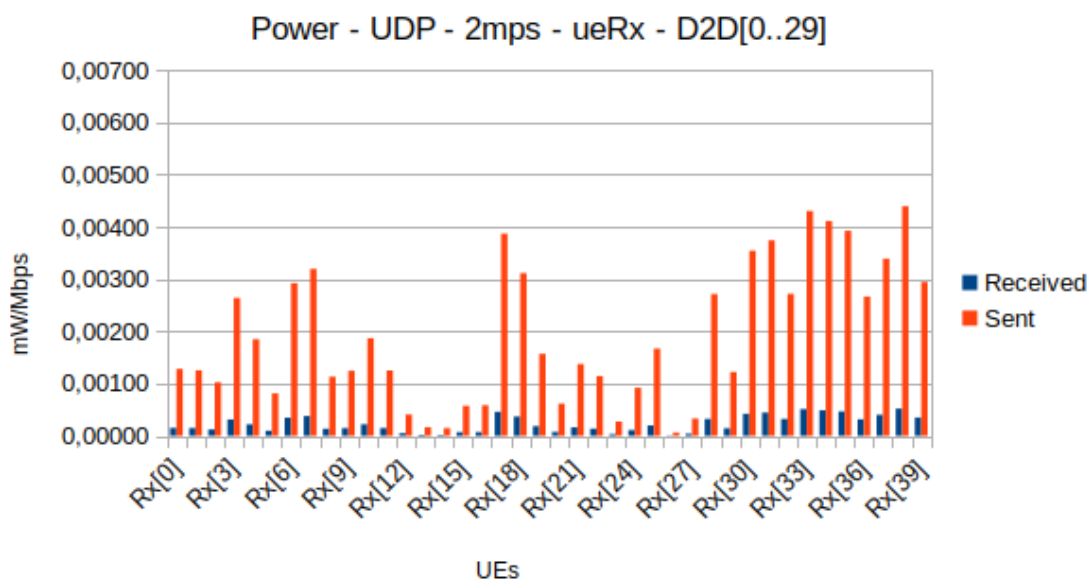


Figura 5.30: Consumo energético dos UEs Rx no experimento UDP 2 m/s com 30 enca-minhadores.

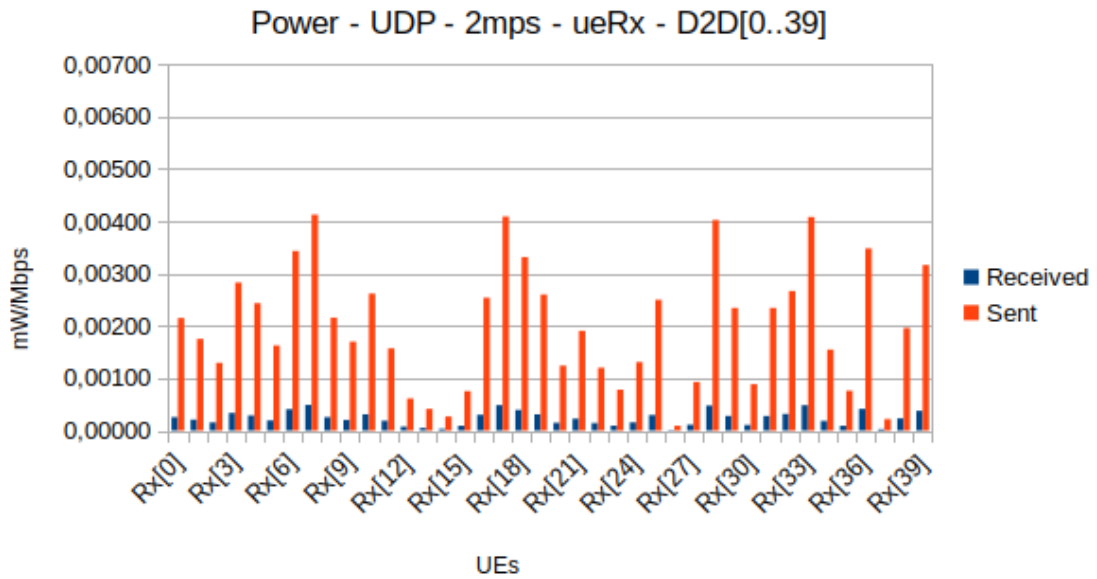


Figura 5.31: Consumo energético dos UEs Rx no experimento UDP 2 m/s com 40 enca-minhadores.

Se analisarmos a Figura 5.32 que demonstra a soma da energia consumida por trans-missão e recepção para todos os nós do tipo Rx e Tx, constatamos que o caso base só não foi superior para recepção nos nós Tx, pois no cenário com 80 nós a soma da energia consumida pelos nós Tx para recepção foi 0,003 mW/Mbps superior. No entanto, em relação ao consumo total da rede o caso base foi o que mais consumiu e superou todos os cenários.

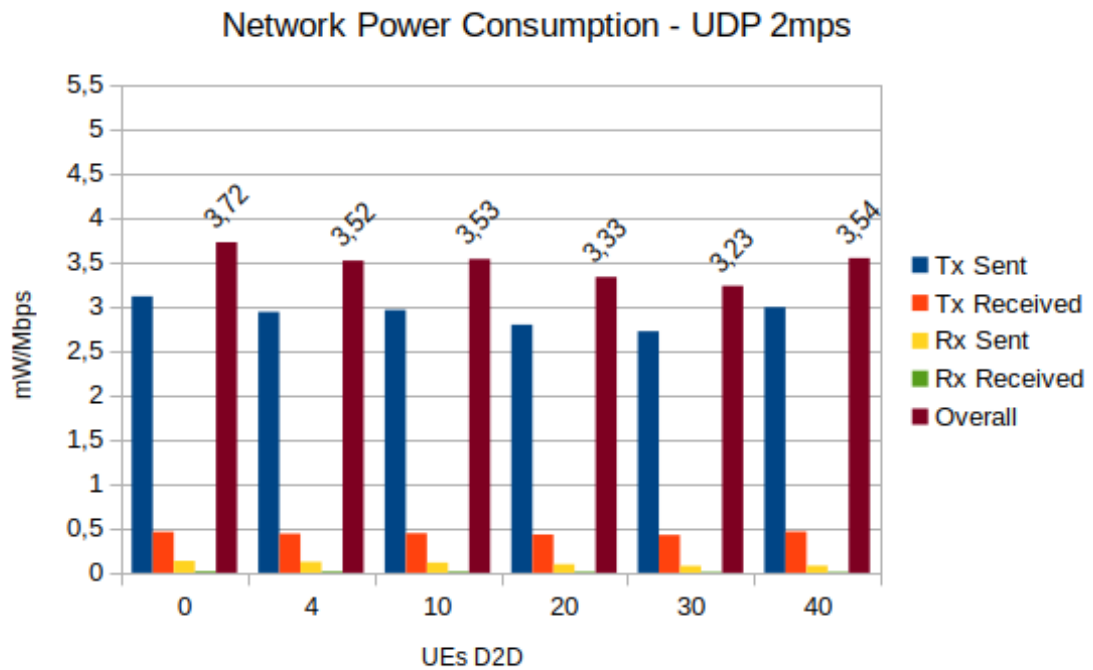


Figura 5.32: Consumo energético total da rede e dos UEs Rx e Tx no experimento UDP 2 m/s para cada cenário.

5.2.4 Experimento: UDP VoIP 1 m/s

Neste experimento, que chamamos de UDP VoIP 1 m/s, todas as configurações de cenário se assemelham às do experimento anterior, exceto pela velocidade de mobilidade utilizada, que agora foi alterada para 1 m/s. Resolvemos executar este experimento com menor velocidade, para simular uma pessoa que anda de forma mais lenta, como em um passeio, observando paisagens ou vitrines.

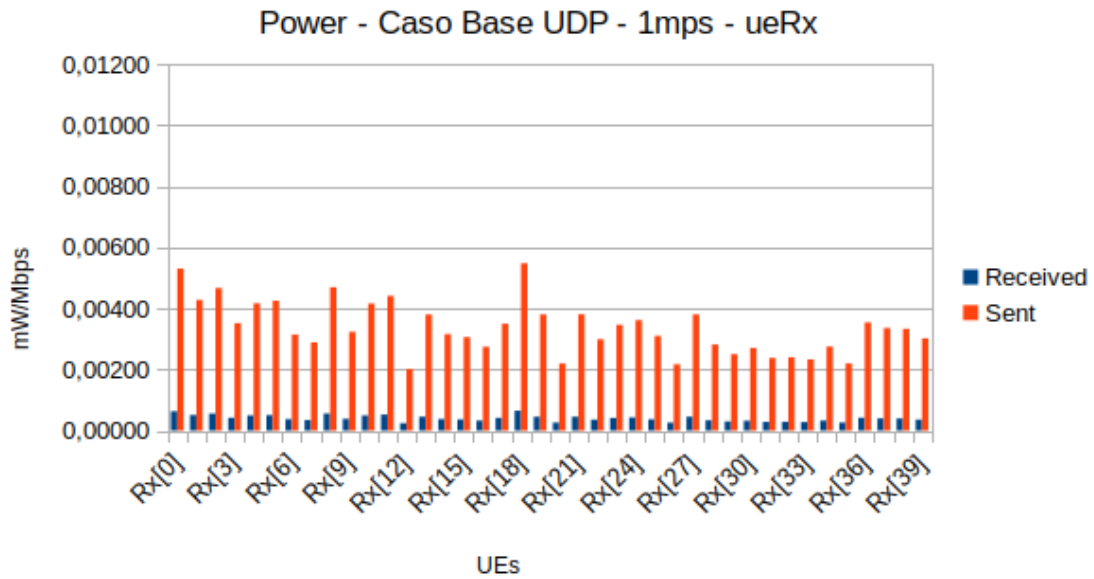


Figura 5.33: Consumo energético dos UEs Rx no experimento UDP 1 m/s.

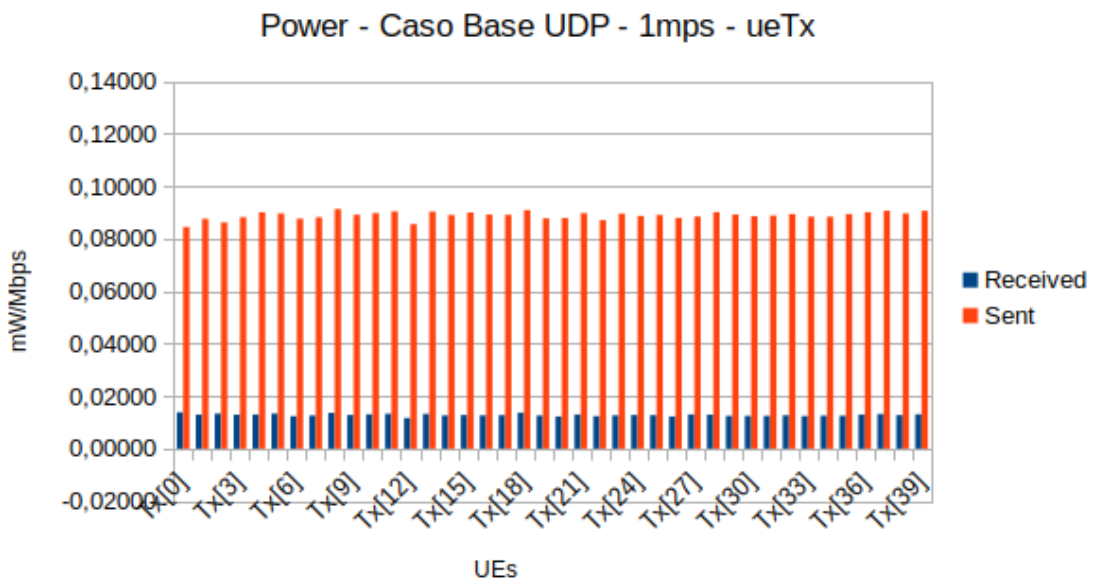


Figura 5.34: Consumo energético dos UEs Tx no experimento UDP 1 m/s.

Em seus casos base, tanto os nós tipo Tx (Figura 5.33) quanto os nós tipo Rx (Figura 5.34) apresentaram pequenas variações de consumo entre si. No entanto, quando inserirmos os nós encaminhadores, percebemos dois comportamentos diferentes: 1. aumenta-se o consumo dos nós Tx que participam dos encaminhamentos; e 2. reduz-se o consumo da maioria dos nós Rx que participam dos encaminhamentos. Para visualizar os comporta-

mentos citados, verifique as Figuras 5.35 a 5.39 para os nós do tipo Tx e as Figuras 5.40 a 5.44 para os de tipo Rx.

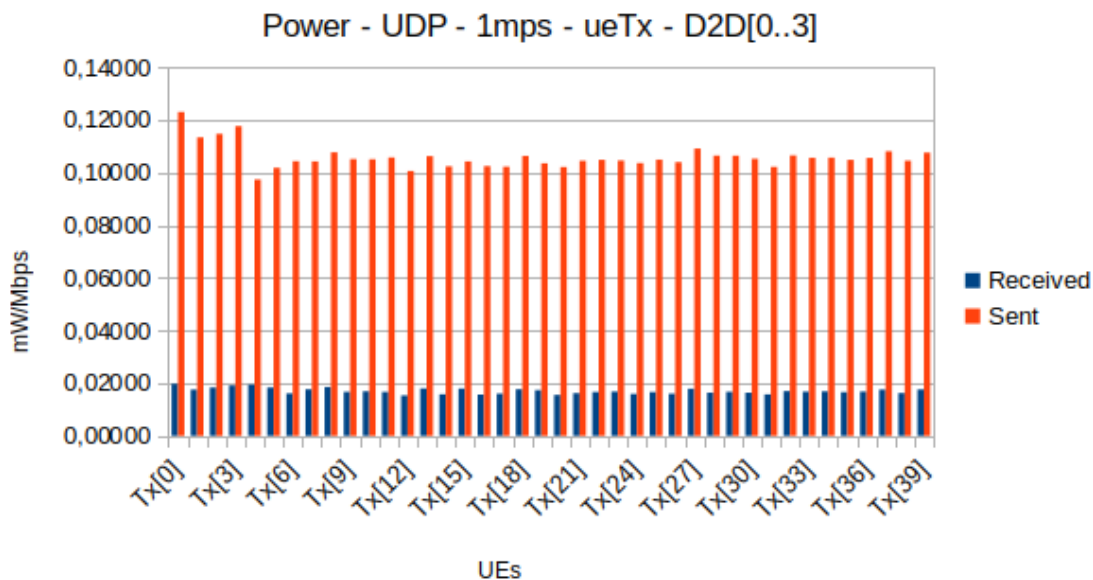


Figura 5.35: Consumo energético dos UEs Tx no experimento UDP 1 m/s com 4 encaminhadores.

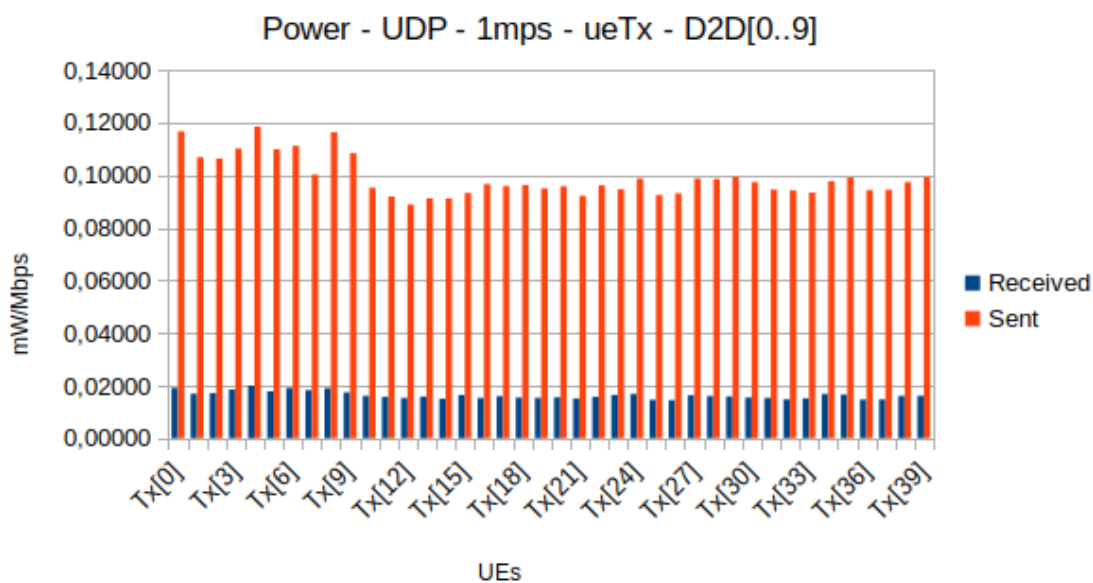


Figura 5.36: Consumo energético dos UEs Tx no experimento UDP 1 m/s com 10 encaminhadores.

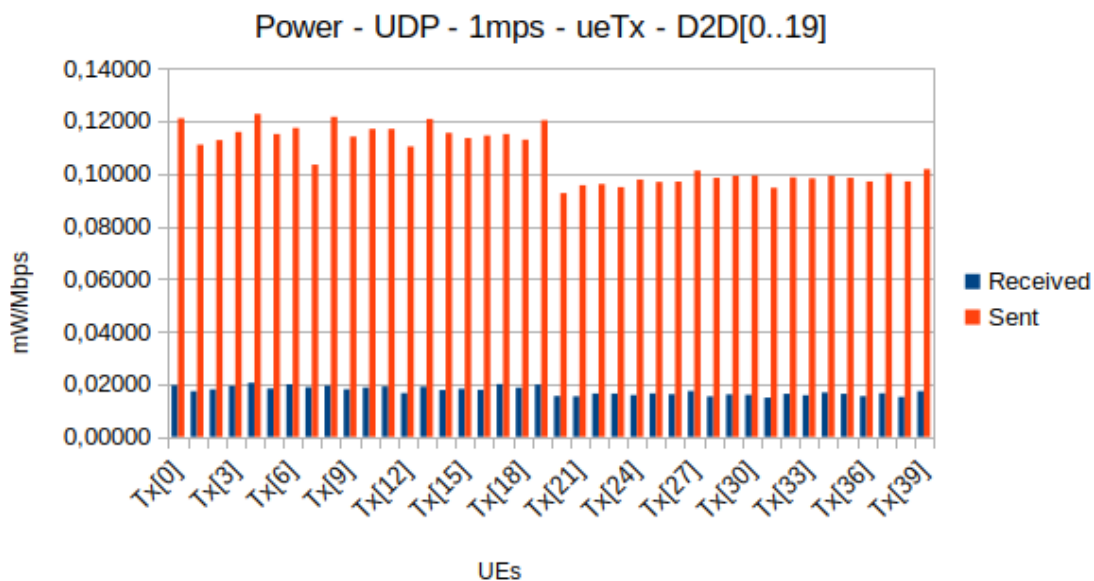


Figura 5.37: Consumo energético dos UEs Tx no experimento UDP 1 m/s com 20 enca-minhadores.

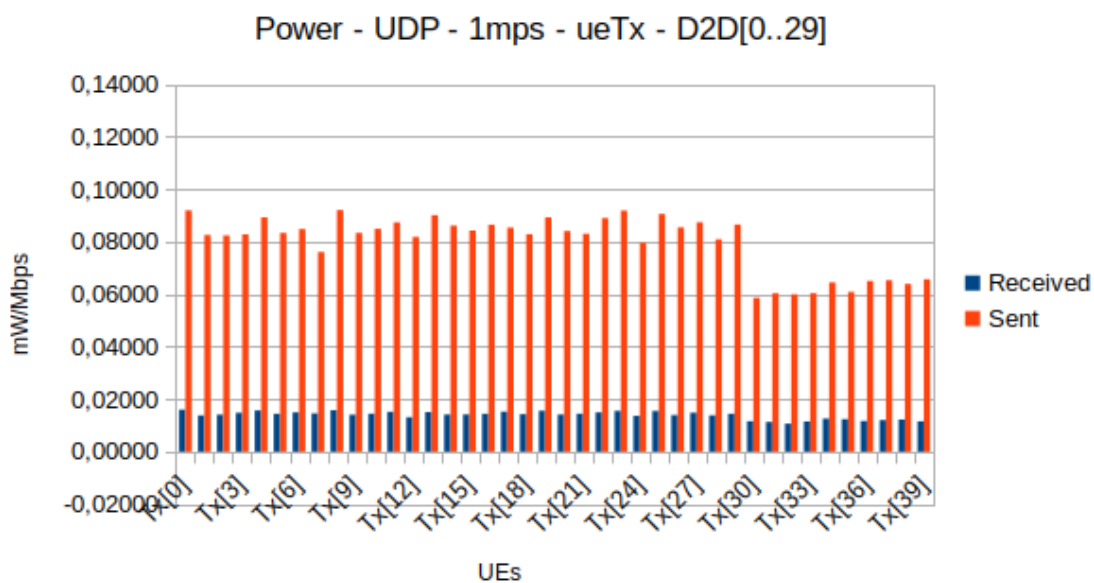


Figura 5.38: Consumo energético dos UEs Tx no experimento UDP 1 m/s com 30 enca-minhadores.

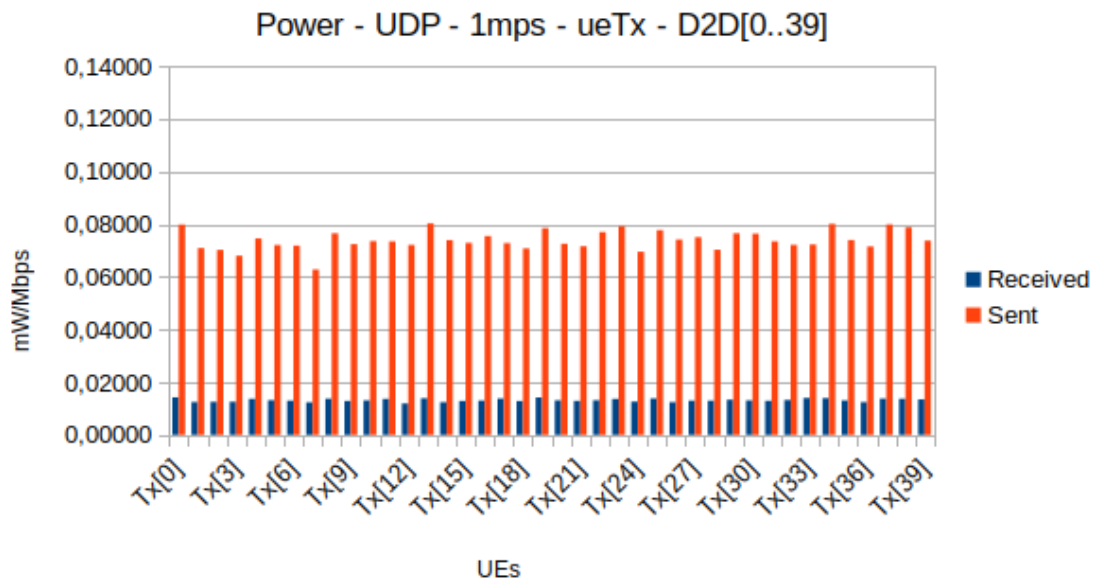


Figura 5.39: Consumo energético dos UEs Tx no experimento UDP 1 m/s com 40 enca-minhadores.

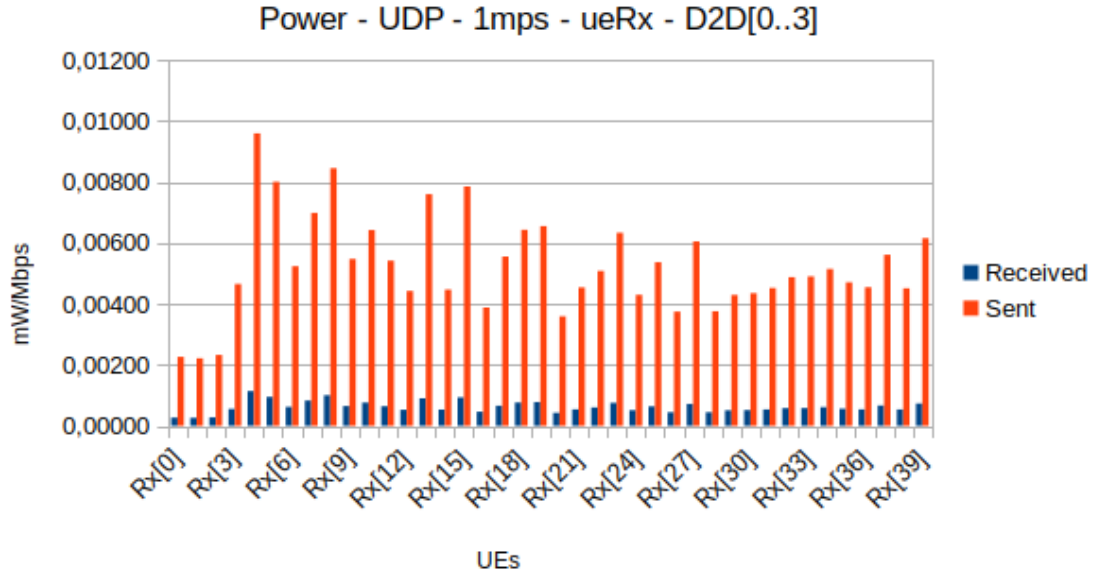


Figura 5.40: Consumo energético dos UEs Rx no experimento UDP 1 m/s com 4 enca-minhadores.

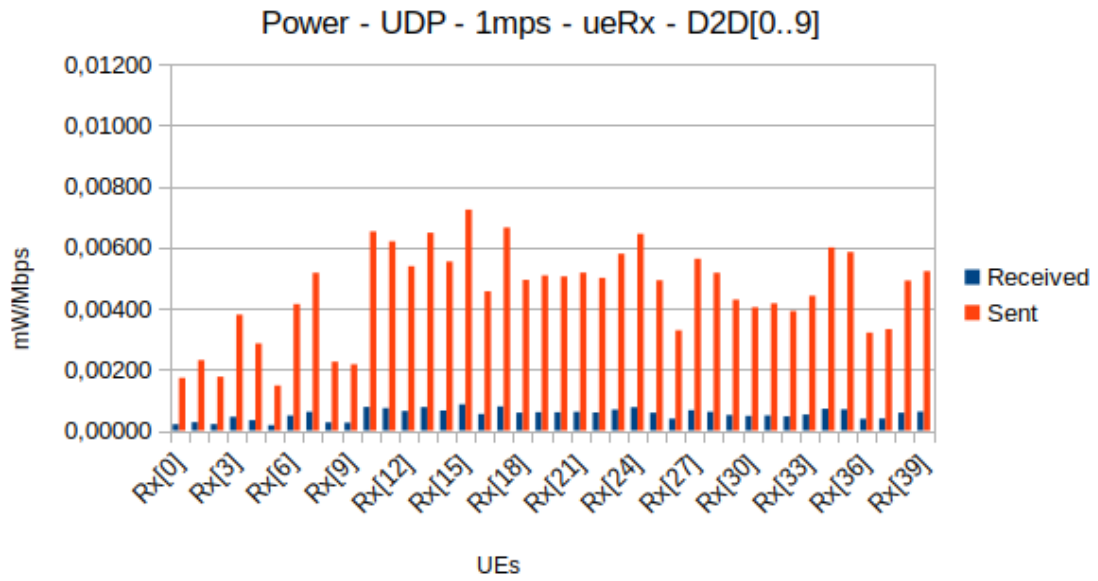


Figura 5.41: Consumo energético dos UEs Rx no experimento UDP 1 m/s com 10 enca-minhadores.

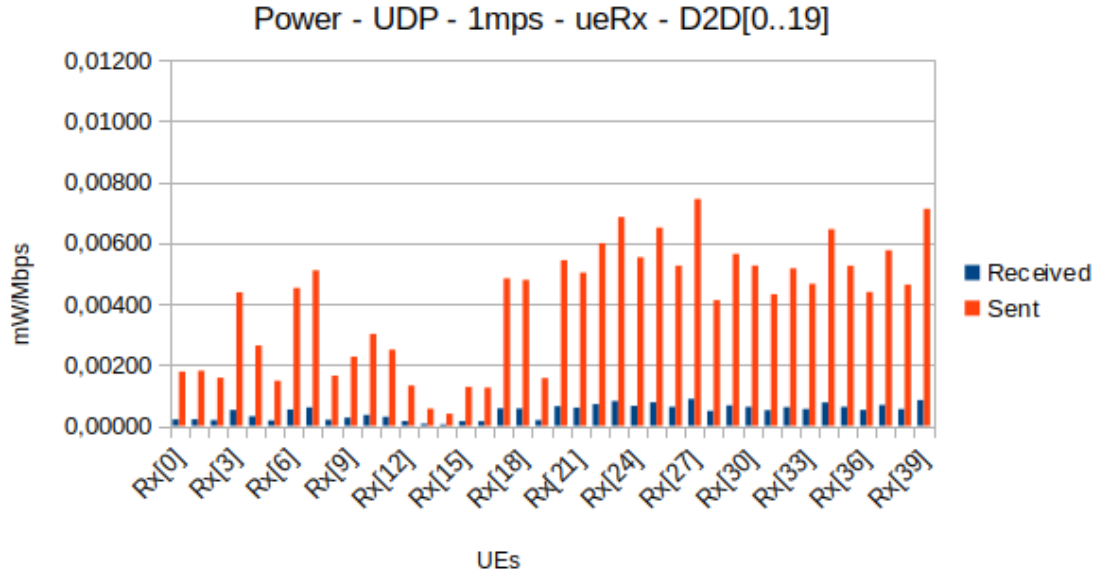


Figura 5.42: Consumo energético dos UEs Rx no experimento UDP 1 m/s com 20 enca-minhadores.

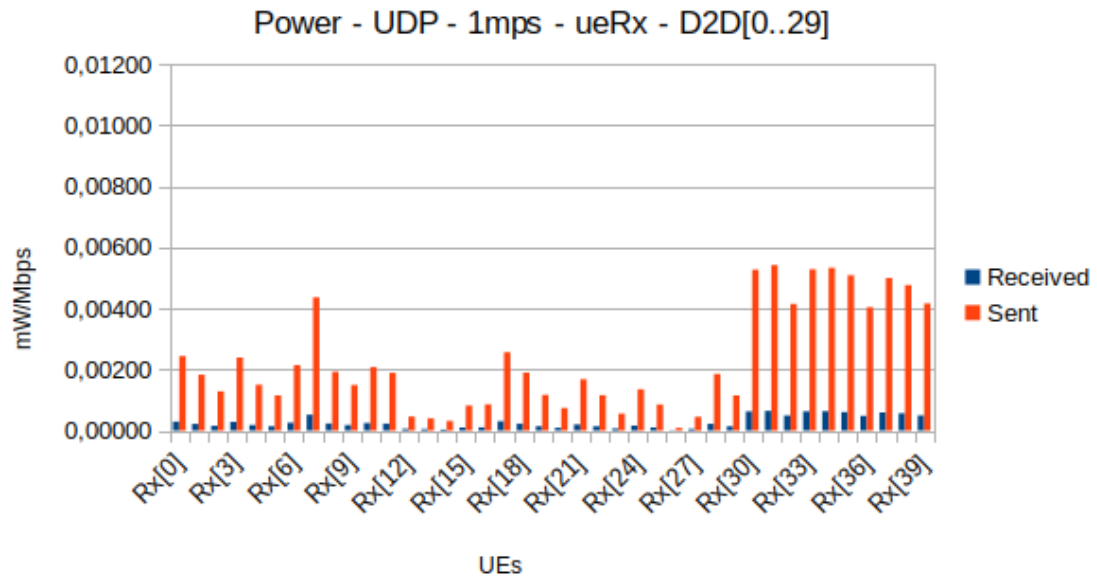


Figura 5.43: Consumo energético dos UEs Rx no experimento UDP 1 m/s com 30 encaminhadores.

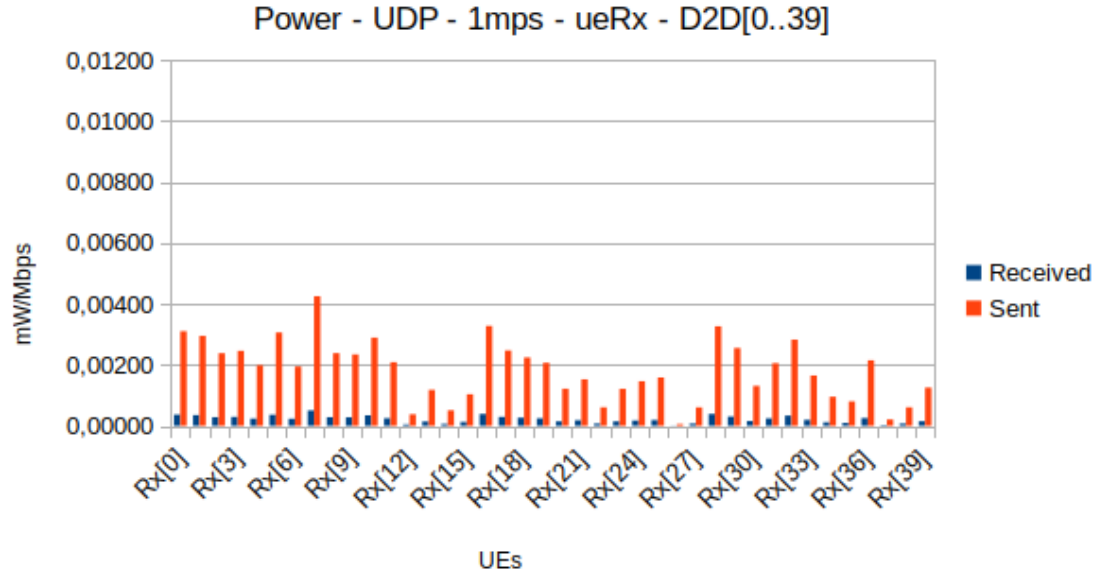


Figura 5.44: Consumo energético dos UEs Rx no experimento UDP 1 m/s com 40 encaminhadores.

Na Figura 5.45 podemos ver o gráfico de consumo energético da rede para este experimento. Diferente do experimento com 2 m/s de velocidade de mobilidade, neste o cenário que menos consumiu energia foi o com 40 nós encaminhando. Os maiores consumo foram

constatados no cenários com 10 e 30 encaminhadores, que se posicionaram como pontos de convergência. Foram o pico de consumo e após eles, o consumo só caiu.

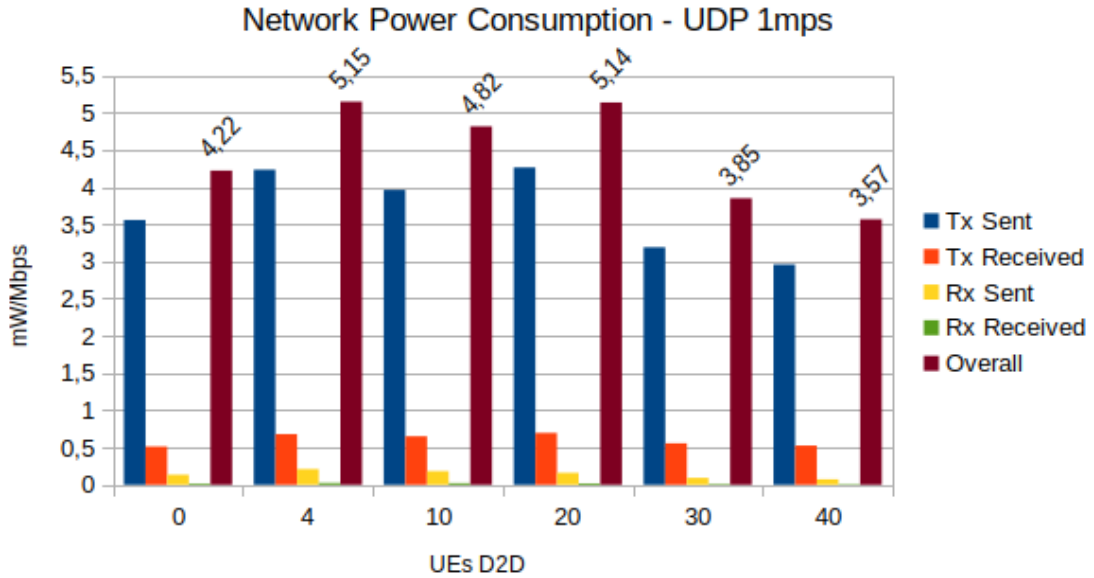


Figura 5.45: Consumo energético total da rede e dos UEs Rx e Tx no experimento UDP 1 m/s para cada cenário.

5.2.5 Experimento: UDP VoIP 5 m/s

Tendo verificado os resultados de mobilidade em velocidades de caminhada de 1 e 2 m/s, optamos por alterar a velocidade de mobilidade para 5 m/s. Assim, podemos perceber como a mesma comunicação se comportaria em uma situação em que pessoas portando seus UEs estivessem realizando uma leve corrida.

No caso base deste cenário os nós Rx destoam um pouco mais do que no anterior. Na Figura 5.46, os Nós de [0] a [11] registram um consumo mais alto do que a maioria dos restantes. Já os nós Tx (Figura 5.47), como no experimento anterior, têm um alto consumo energético, mas com uma pequena variação, entre 0,07 a 0,08 mW/Mbps.

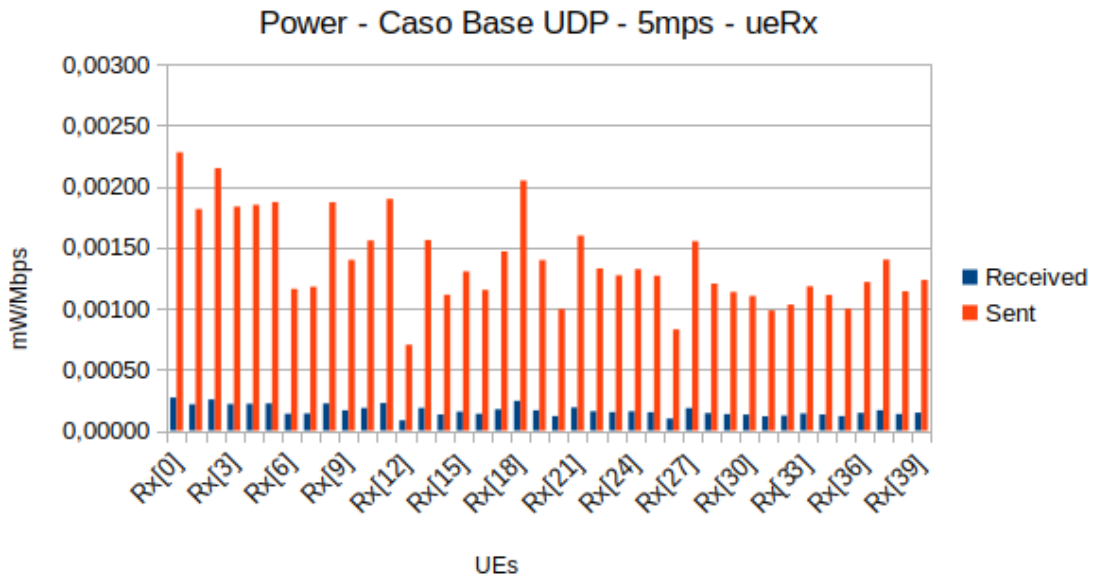


Figura 5.46: Consumo energético dos UEs Rx no experimento UDP 5 m/s.

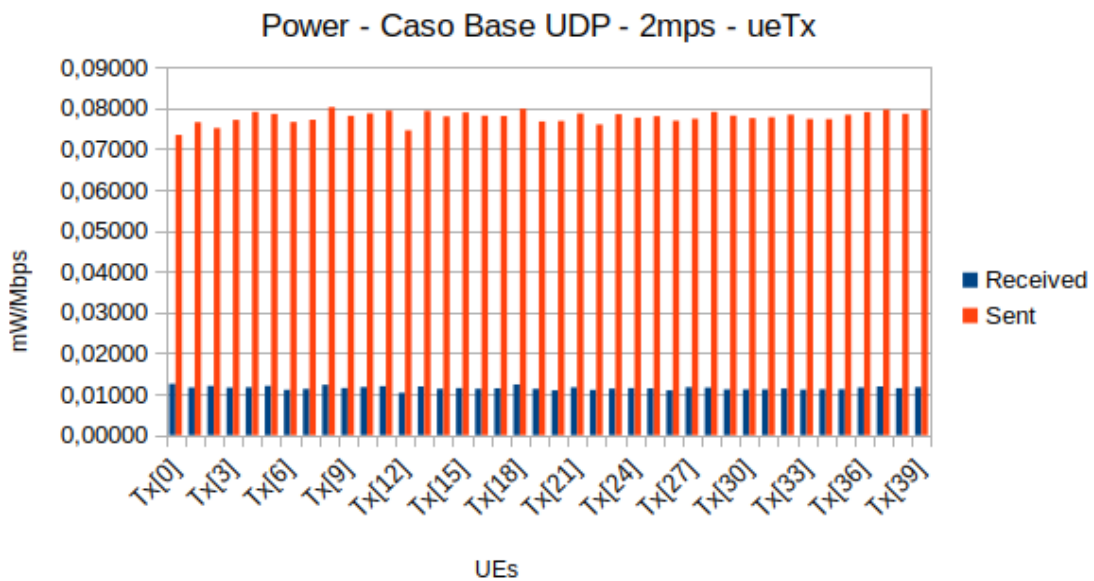


Figura 5.47: Consumo energético dos UEs Tx no experimento UDP 5 m/s.

Analisando os nós do tipo Rx e Tx ao longo dos cenários, acontece algo semelhante ao do experimento anterior: podemos perceber redução do consumo entre a maioria dos nós Rx que participam do encaminhamento e aumento do consumo nos Tx. Os gráficos resultantes dos nós Tx podem ser vistos nas Figuras 5.48 a 5.52 e os dos nós Rx podem ser encontrados nas Figuras 5.53 a 5.57.

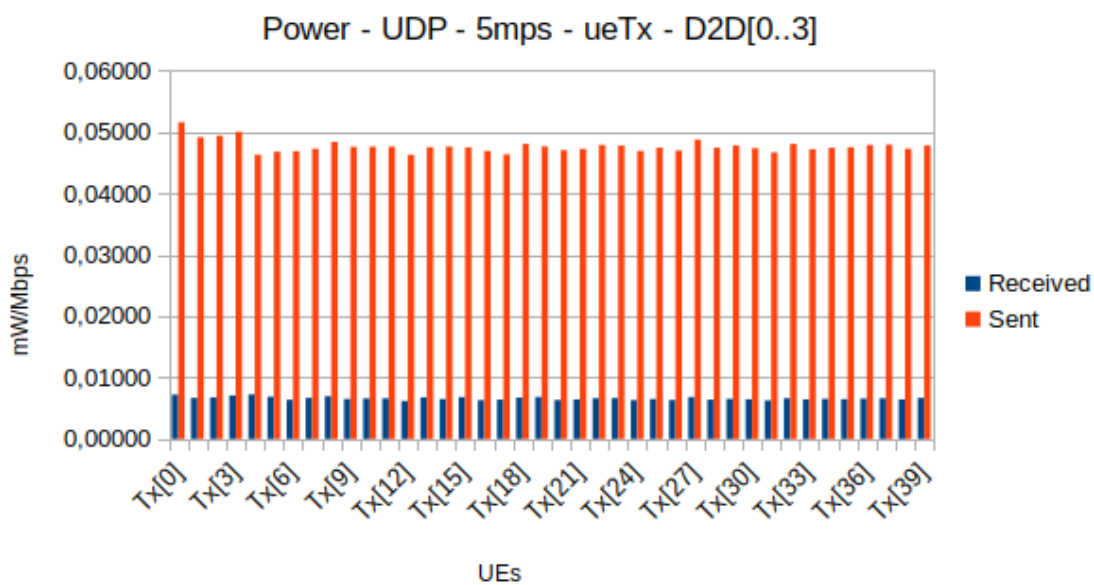


Figura 5.48: Consumo energético dos UEs Tx no experimento UDP 5 m/s com 4 encaminhadores.

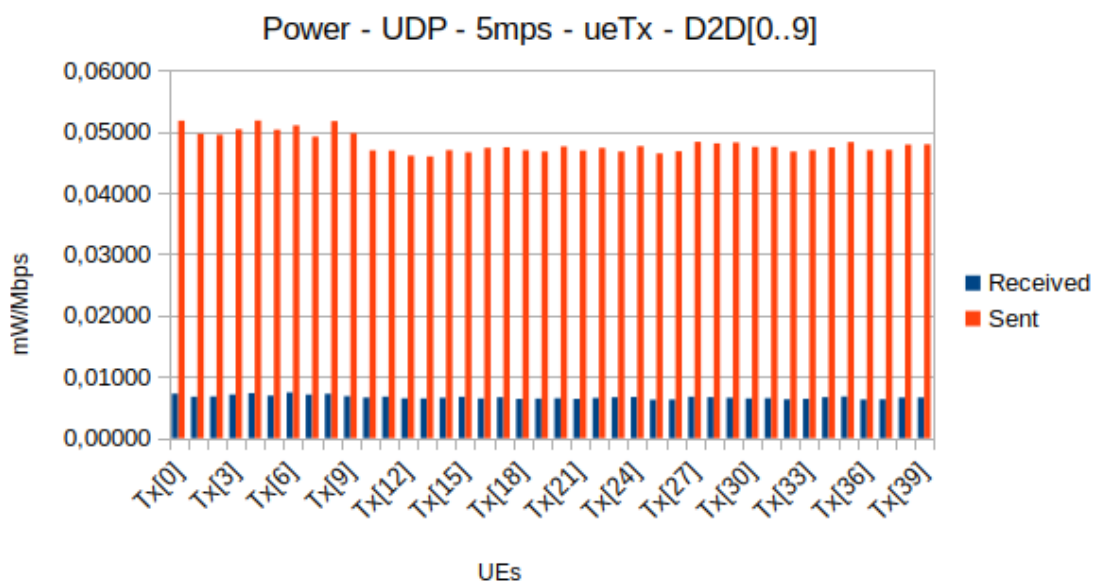


Figura 5.49: Consumo energético dos UEs Tx no experimento UDP 5 m/s com 10 encaminhadores.

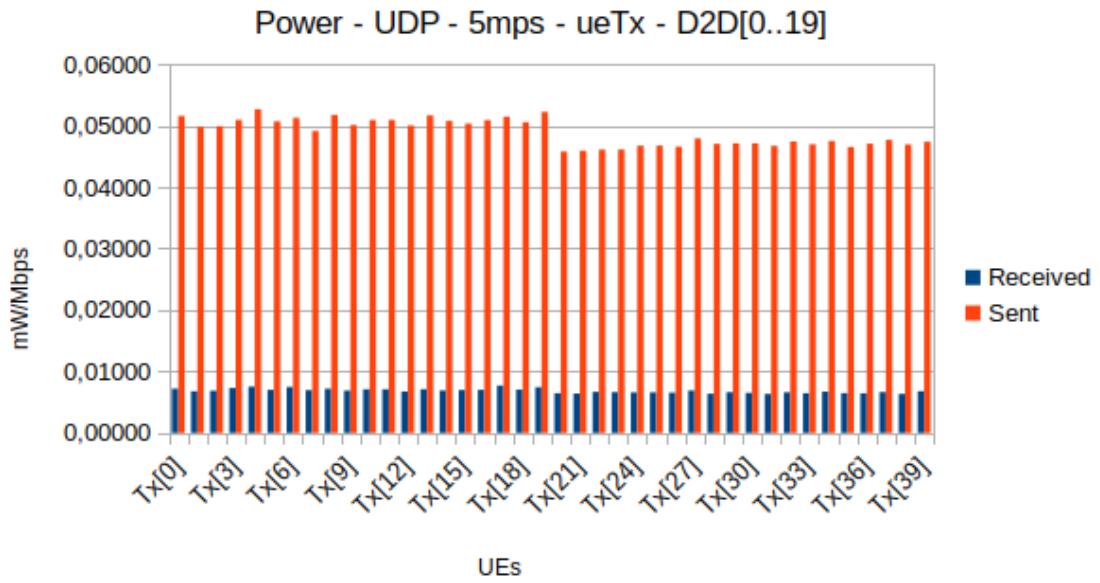


Figura 5.50: Consumo energético dos UEs Tx no experimento UDP 5 m/s com 20 enca-minhadores.

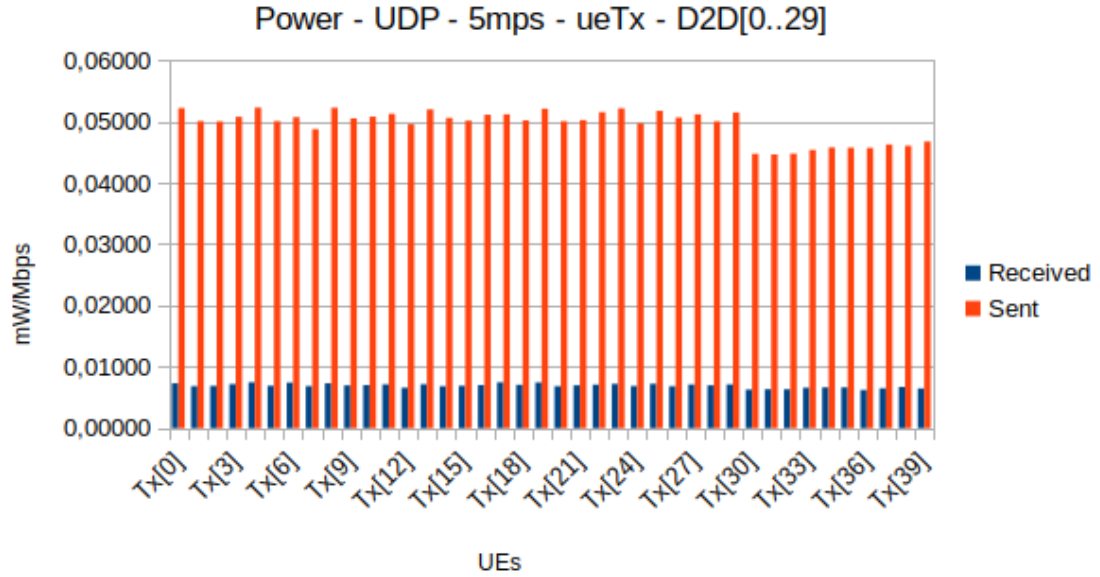


Figura 5.51: Consumo energético dos UEs Tx no experimento UDP 5 m/s com 30 enca-minhadores.

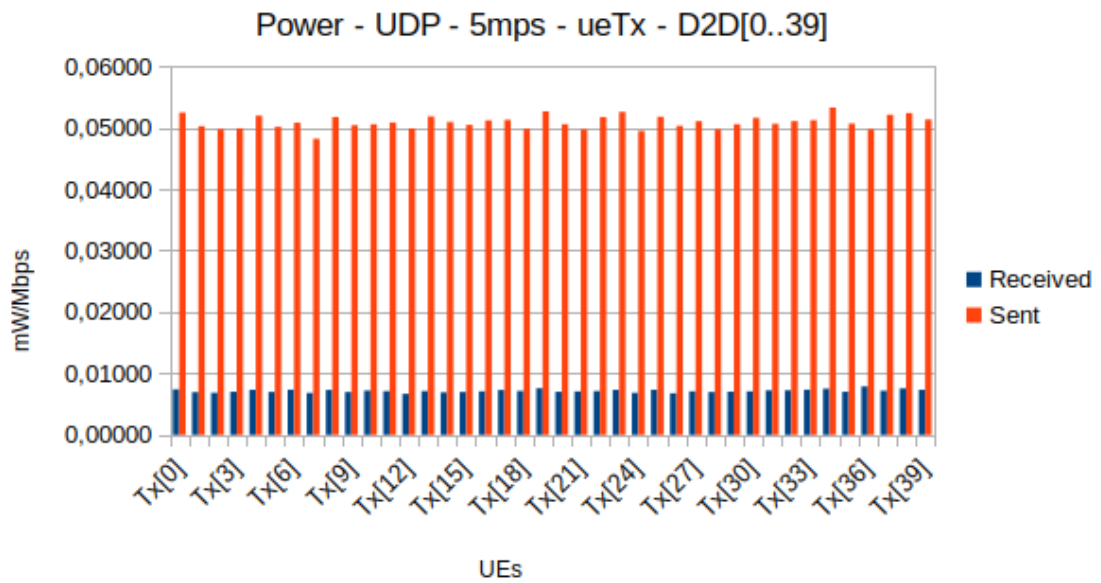


Figura 5.52: Consumo energético dos UEs Tx no experimento UDP 5 m/s com 40 enca-minhadores.

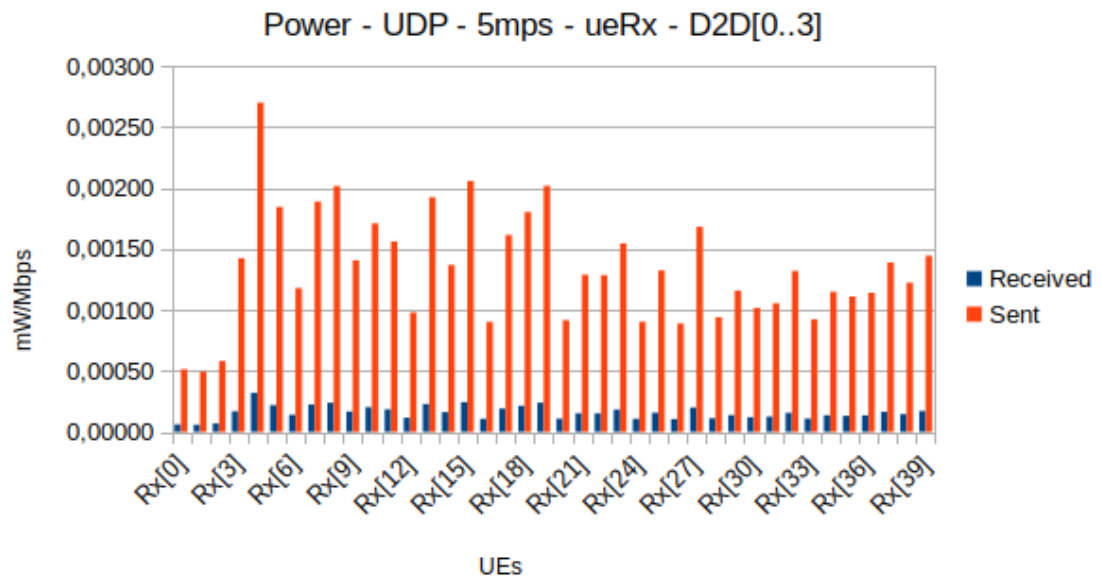


Figura 5.53: Consumo energético dos UEs Rx no experimento UDP 5 m/s com 4 enca-minhadores.

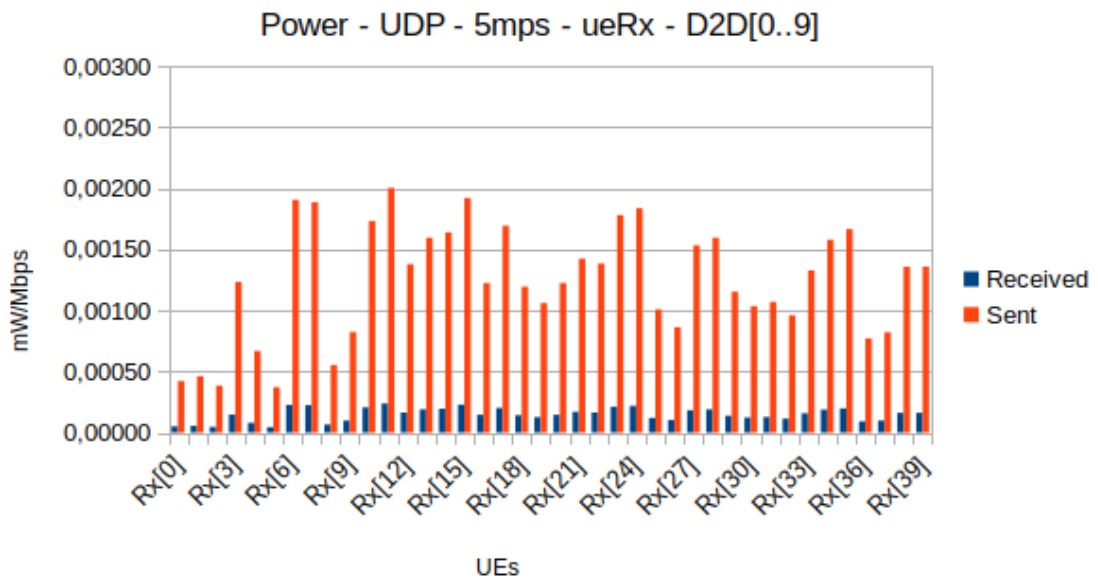


Figura 5.54: Consumo energético dos UEs Rx no experimento UDP 5 m/s com 10 enca-minhadores.

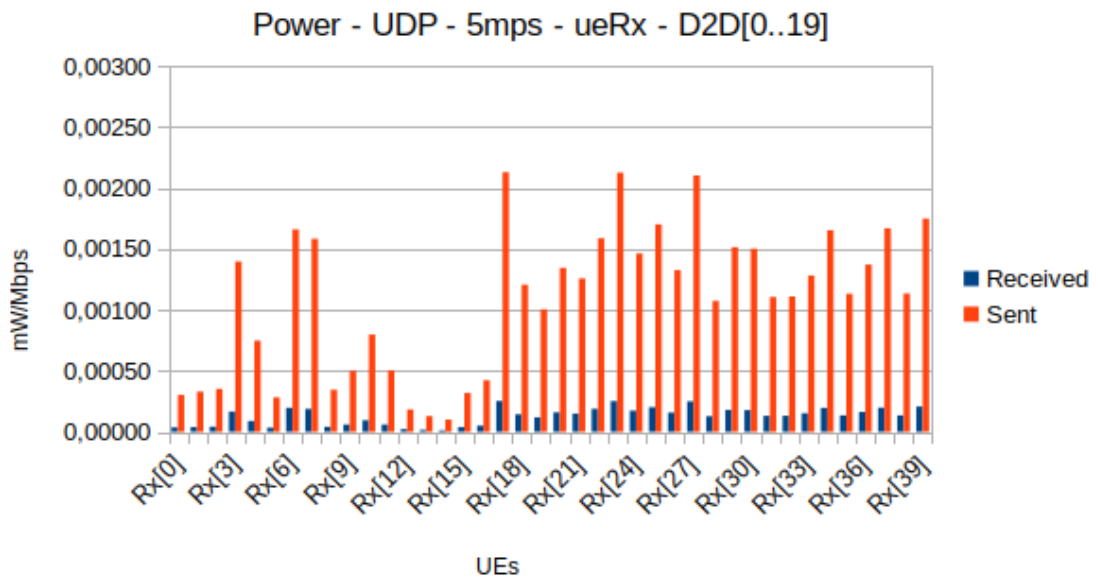


Figura 5.55: Consumo energético dos UEs Rx no experimento UDP 5 m/s com 20 enca-minhadores.

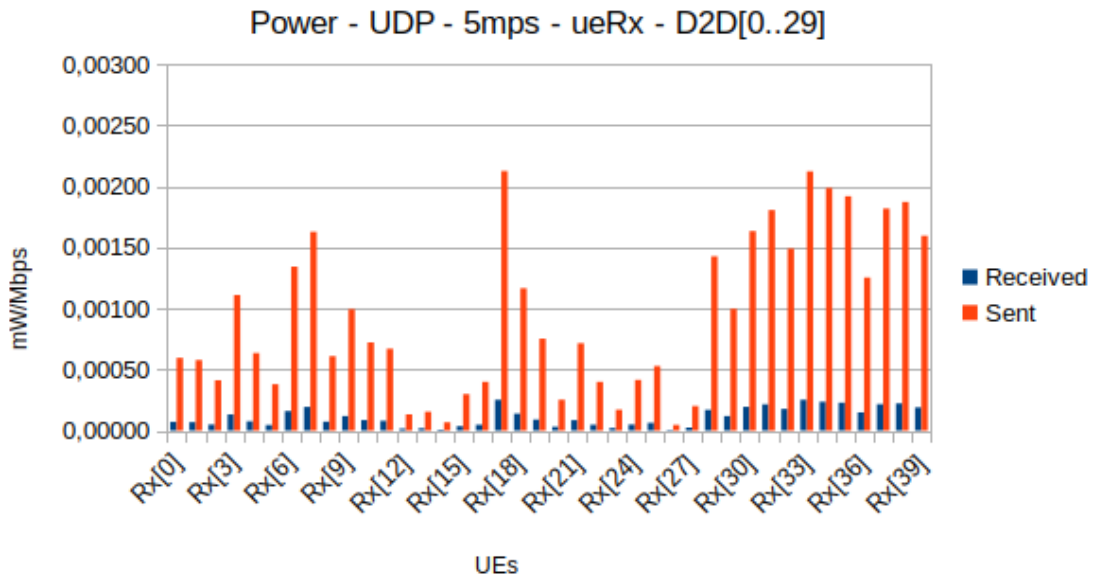


Figura 5.56: Consumo energético dos UEs Rx no experimento UDP 5 m/s com 30 encaminhadores.

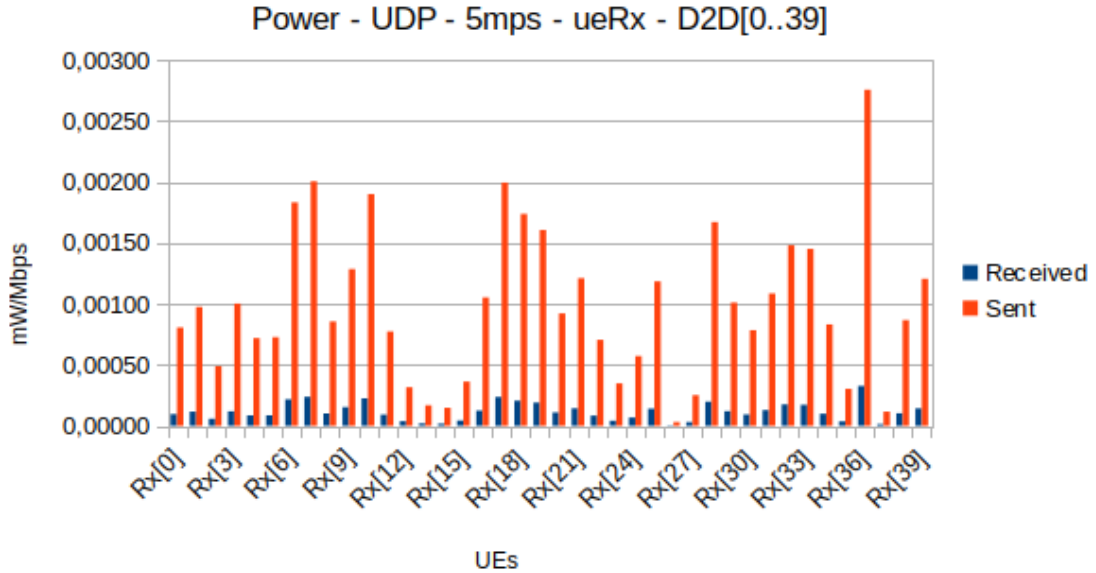


Figura 5.57: Consumo energético dos UEs Rx no experimento UDP 5 m/s com 40 encaminhadores.

No gráfico apresentado à seguir, as barras azuis se referem ao consumo de envio dos nós Tx, que podem encaminhar comunicação; as barras em laranja trazem o consumo de recepção destes mesmos nós. Já as cores amarelo e verde se referem ao consumo para

envio e recepção, respectivamente, dos nós que estão habilitados a receberem comunicação encaminhada. As barras em vinho referem-se à quantidade total de energia que foi gasta naquele cenário. O eixo X se refere aos diferentes cenários, com 0, 4, 10, 20, 30 e 40 encaminhadores, e o eixo Y se refere à quantidade de energia consumida, conforme a modelagem já apresentada.

Considerando a Figura 5.58, que apresenta o gráfico de consumo total da rede nos cenários deste experimento, verificamos que novamente o maior consumo é do cenário sem encaminhamento. No entanto, agora a diferença total de consumo entre os cenários com encaminhamento foi sensivelmente pequena, de apenas 0,14 mW/Mbps. Desta vez o cenário com menor consumo energético foi o que possuía 4 encaminhadores.

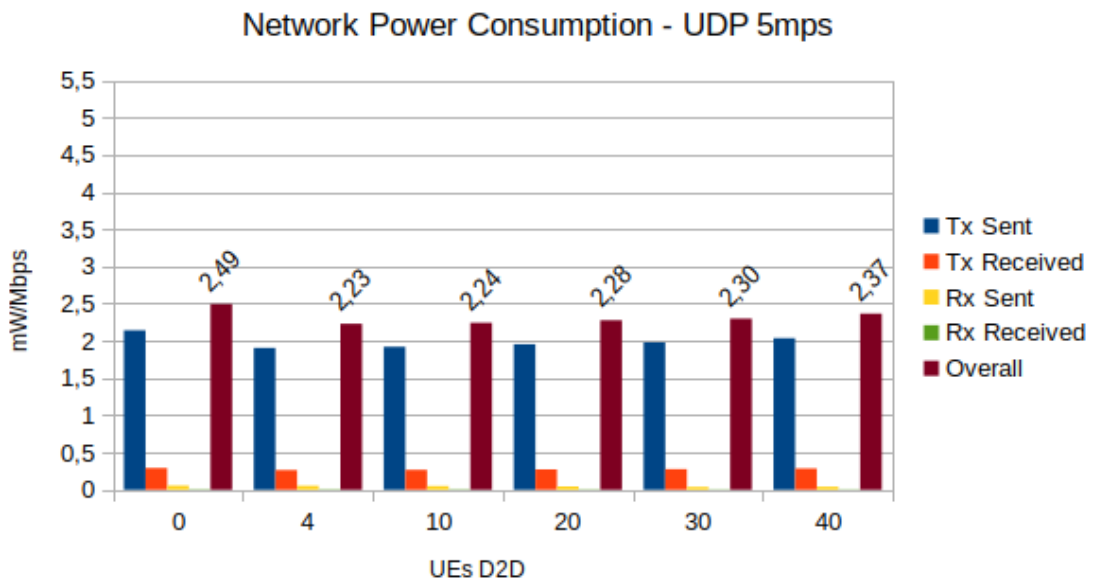


Figura 5.58: Consumo energético total da rede e dos UEs Rx e Tx no experimento UDP 5 m/s para cada cenário.

5.2.6 Comparação entre os experimentos UDP com diferente mobilidade

Façamos agora uma comparação utilizando os experimentos que rodaram uma aplicação VoIP sob protocolo UDP. Uma vez que estes experimentos têm configurações de rede e de variações de quantidade de encaminhadores semelhantes, diferenciando entre si apenas a velocidade de mobilidade de 1, 2 e 5 m/s, utilizaremos essa variação para realizar uma comparação entre eles. Para cada cenário com uma das variações de velocidade, variamos também a quantidade de encaminhadores em 100, 75, 50, 25 e 10% e, mais uma variação,

sem nenhum encaminhador.

A Figura 5.59 traz o gráfico comparativo das variações de velocidade de 1, 2 e 5 m/s. Analisando-o, podemos perceber que o cenário com maior velocidade de mobilidade, 5 m/s, foi o que obteve os menores consumos de energia. Nele, todos experimentos com encaminhamento obtiveram um menor consumo energético geral da rede, sendo o cenário com 4 encaminhadores o mais eficiente.

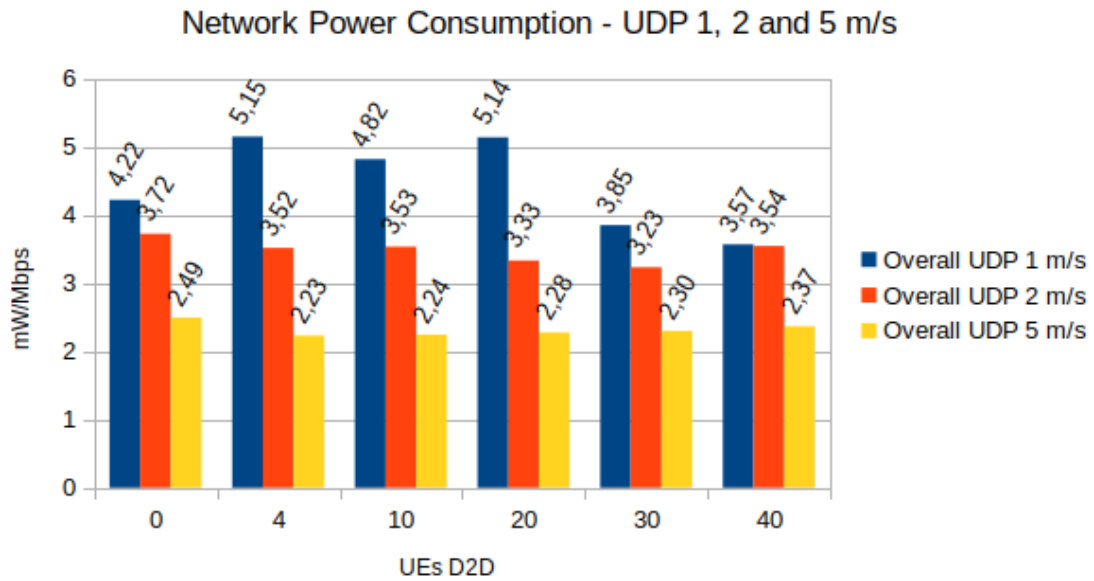


Figura 5.59: Consumo energético do experimento UDP com mobilidade de 1 a 5 m/s.

5.2.7 Comparação entre os experimentos TCP e UDP 2 m/s

Em outro cenário, utilizamos, invés de UDP, uma aplicação TCP que abre uma conexão, envia um dado número de bytes e fecha. Neste cenário a mobilidade também foi de 2 m/s. A Figura 5.60 mostra uma comparação dos resultados do experimento TCP com os resultados provenientes do Cenário UDP com mobilidade de 2 m/s, no qual percebemos que naquele ainda se obteve menores consumos energéticos.

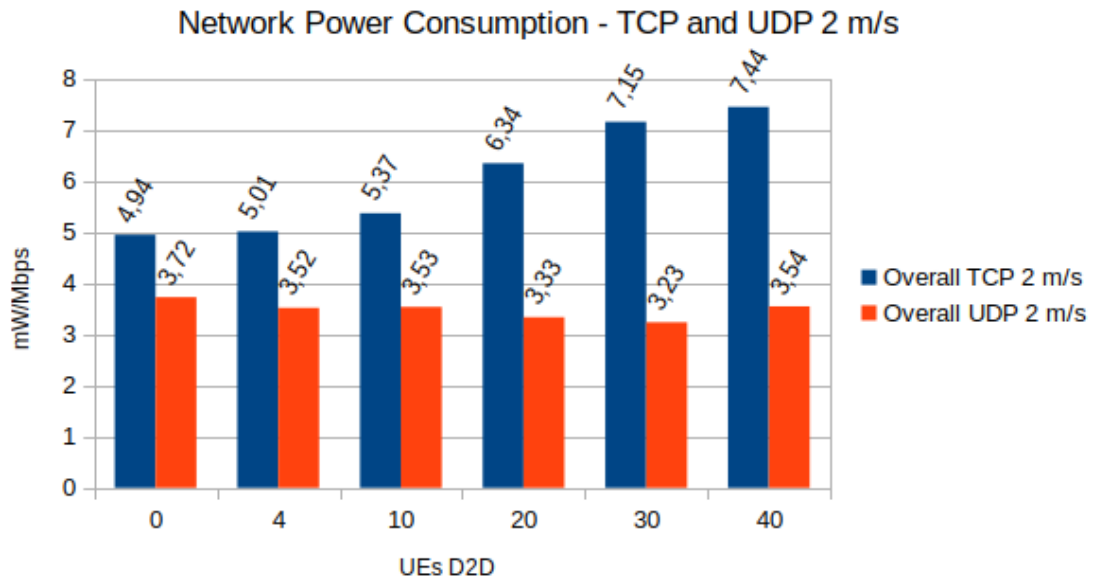


Figura 5.60: Consumo energético do experimento TCP e UDP com mobilidade de 2 m/s.

Analisando apenas o cenário com TCP, é possível enxergarmos um padrão de consumo energético diferente: quanto mais encaminhadores inserimos nos experimentos, maior o consumo total da rede. A quantidade de pacotes transmitidos tanto pelos nós Tx quanto Rx demonstram claramente este crescimento. Para os nós Rx o crescimento sofre um salto maior a partir de 20 nós habilitados em D2D.

5.2.8 Comparação entre os experimentos UDP de *Streaming* de Vídeo

Nestes experimentos, utilizamos sob o protocolo UDP a aplicação de *streaming* de vídeo padrão do OMNeT++, variando além da quantidade de encaminhadores, o tamanho dos pacotes enviados entre 256, 512 e 1024 B. O arquivo transmitido foi de tamanho 4MiB e não houve intervalo no envio. A velocidade de mobilidade foi mantida em 2 m/s para todos os casos.

Devido a limitações do simulador e dos modelos de simulação, estes experimentos de *streaming de vídeo* foram realizados com cada cenário contendo apenas os nós encaminhadores. Neste tipo de configuração o simulador sempre retornava erro ao tentar inserir nós não encaminhadores. Desta forma, foi mais interessante que realizássemos uma análise do comportamento de cada configuração de cenário nos diferentes experimentos. Para cada um dos três experimentos há um cenário com 80 nós, sem nenhum encaminhador e outros

5 cenários que contêm apenas nós encaminhadores, cada um com 4, 10, 20, 30 e 40 nós. Para visualizar de maneira mais fácil os gráficos desta seção, cada um deles se referirá a um destes cenários mencionados e o eixo X descreverá os 3 diferentes tamanhos de pacotes utilizados naquele cenário. Os gráficos específicos de cada experimento poderão ser verificados em maior detalhe no Apêndice C.

A Figura 5.61 apresenta o gráfico com os resultados do consumo da rede para cada um dos cenários dos experimentos de *streaming* de vídeo com tamanho dos pacotes enviados de 256, 512 e 1024 Bytes. Com exceção dos cenários com 10 e 30 nós encaminhadores, no experimento com pacotes de 512 B, nos demais casos podemos observar que o tamanho dos pacotes pouco influenciou o consumo de energia, já que os valores foram todos muito próximos.

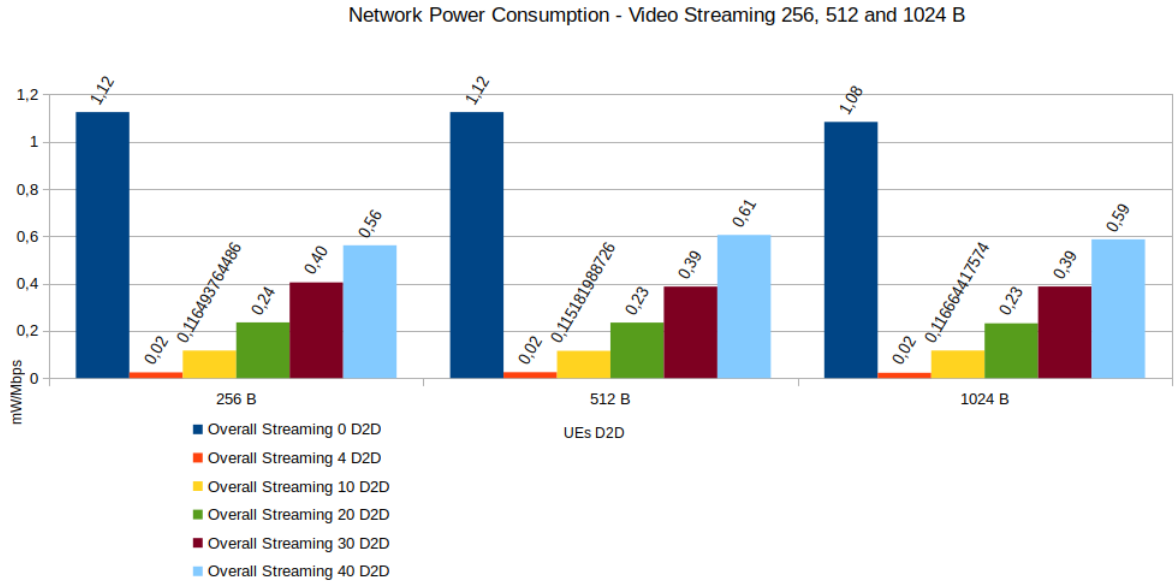


Figura 5.61: Consumo energético dos experimento de Streaming com 0 a 40 encaminhadores.

Capítulo 6

Conclusão

Nesta pesquisa investigamos o consumo de energia da comunicação dispositivo a dispositivo, em redes celulares. Neste contexto pudemos perceber a 5G como uma próxima geração de telefonia que integra de maneira eficaz a comunicação D2D à comunicação de rede. E, assim, ela se posiciona como uma solução não só para telefonia móvel como para o aumento de tráfego na Internet como um todo. Foi discutido como a 5G se conectará a diversos sistemas e tecnologias e formará uma rede única com as redes fixas, não se restringindo somente aos sistemas móveis.

As principais dificuldades encontradas foram em respeito a ausência de simuladores neste contexto com funções de energia já implementadas e, depois, em implementar um módulo de energia devido a tamanha complexidade de codificar as especificidades de descarga e depleção de baterias. Esperamos que outros pesquisadores possam se inteirar destas dificuldades como experiências, partir suas pesquisas deste ponto, e evoluir um pouco mais a temática.

Realizamos uma modelagem matemática, à partir da quantidade de dados trafegados ao longo do tempo, para avaliar o consumo energético dos encaminhadores. Como resultados, pudemos perceber que a comunicação D2D, dependendo da aplicação, protocolo e quantidade de encaminhadores, pode ocasionar tanto economia quanto gasto adicional de energia. O principal fator para alterar este comportamento foi o protocolo, no entanto, a velocidade de mobilidade e quantidade de nós participantes do encaminhamento também afetou.

No experimento com Multicast, pudemos perceber que a inserção de encaminhamento sempre ocasionará em aumento de consumo energético. Da mesma forma, o experimento com TCP aumentou o consumo de energia todas as vezes que incluíamos mais encami-

nhadores no cenário.

Já quando utilizamos UDP, realizamos experimentos com VoIP e *streaming* de vídeo. Em ambos não foi possível identificar resultados que pareçam apontar para uma função que tenha um ponto ótimo bem definido. Os cenários apresentaram resultados dinâmicos, em que ora se tinha maior consumo energético com uma quantidade de nós, ora menor, dependendo da variação de mobilidade e tamanhos de pacote, para o caso de *streaming* de vídeo.

Comparando os experimentos de *streaming* de voz, com suas variações de velocidade de mobilidade, obteve destaque o experimento com velocidade de 5 m/s. Este foi o que obteve os menores valores de consumo energético, inclusive no cenário sem encaminhadores, se comparado-o com os outros experimentos de velocidade de 1 e 2 m/s.

Nos experimentos de *streaming* de vídeo o comportamento de menor consumo foi variado. Pudemos perceber que, independente do cenário, o tamanho dos pacotes pouco influenciou os resultados.

Diante tantas nuances e variações nas análises destes resultados, é possível perceber que é grande a gama de simulações que se pode realizar para cada vez entender mais o consumo energético nestes contextos. Se poderia, por exemplo, realizar ainda mais variações em novas simulações com outros tipos de aplicação, outras velocidades de mobilidade e tamanhos de pacote e arquivos diferentes.

Como trabalhos futuros, além da pretensão de realizar novas simulações para refinar cada vez mais as conclusões de cada tipo de comportamento energético, há ainda alguns tópicos de interesse como:

- realizar simulações considerando a geração de energia na rede;
- verificar o comportamento de consumo energético com outros modelos de mobilidade;
- implementar um módulo de bateria que possibilite, pelo menos, considerar estados diferentes no consumo de energia;
- considerar alocação de recursos pra economizar energia.

Além destes tópicos, há outros pontos de pesquisa em aberto sobre esta temática que poderão ser considerados como trabalhos futuros, como:

- Mecanismos de incentivos ao encaminhamento em redes móveis [15].
- A segurança e a privacidade da rede e dos dispositivos também são preocupações que merecem cuidado, principalmente nos cenários em que passa a existir comunicação direta (D2D) ou encaminhada entre dispositivos e máquinas e um contexto de Internet das Coisas [45].
- O impacto do descarregamento de tráfego e distribuição de conteúdo para as bordas [10].

Diante destas discussões, é possível visualizar uma outra tecnologia em que a comunicação D2D na 5G será de suma importância, principalmente quando considerando a eficiência energética: a Internet das Coisas (IoT). A 5G passará a ser parte integrante da Internet das Coisas, integrando diversos tipos diferentes de dispositivos, formando uma rede em comum por meio de uma conexão com a Internet [45]. Na IoT é precípua a comunicação D2D entre os dispositivos autônomos e também deles com os dispositivos de interação humana. À 5G fica o desafio de prover conectividade para tais equipamentos em diversificados contextos, com o menor consumo de energia possível.

As comunicações entre dispositivos abrem uma gama de novas possibilidades para as redes de computadores. O lançamento oficial da 5G em 2020 e o advento do crescimento da IoT são potencializadores para a adesão a este tipo de comunicação. Nos próximos anos, uma série de novidades e desafios poderão ser esperados. Do autor desta dissertação fica o mais sincero desejo de que este trabalho possa servir como auxílio para outros pesquisadores que também embarcam neste complexo emaranhado de tecnologias de nova geração.

Referências

- [1] 3GPP. 3GPP Release 10. Disponível em <https://www.3gpp.org/specifications/releases/70-release-10>, 2011. Acessado em: 03/19.
- [2] 3GPP. 3GPP Release 12. Disponível em <https://www.3gpp.org/specifications/releases/68-release-12>, 2015. Acessado em: 03/19.
- [3] 3GPP. 3GPP Release 13. Disponível em <https://www.3gpp.org/release-13>, 2015. Acessado em: 03/19.
- [4] 3GPP. Tentative 3GPP timeline for 5G. Disponível em https://www.3gpp.org/news-events/3gpp-news/1674-timeline_5g, 2015. Acessado em: 03/19.
- [5] 3GPP. 3GPP Release 14. Disponível em <https://www.3gpp.org/release-14>, 2017. Acessado em: 03/19.
- [6] 3GPP. 3GPP Release 15. Disponível em <https://www.3gpp.org/release-15>, 2018. Acessado em: 03/19.
- [7] 3GPP. 3GPP Release 16. Disponível em <https://www.3gpp.org/release-16>, 2018. Acessado em: 03/19.
- [8] 3GPP. About 3GPP. Disponível em <https://www.3gpp.org/about-3gpp/about-3gpp>, 2019. Acessado em: 03/19.
- [9] AKYILDIZ, I. F., LEE, W.-Y., CHOWDHURY, K. R. CRAHNs: Cognitive radio ad hoc networks. *Elsevier: Ad Hoc Networks* 7, 5 (julho de 2009), 810–836.
- [10] BASTOS, I. V., MORAES, I. M., THI-MAI-TRANG, N., PUJOLLE, G. Modelo e Avaliação da Recuperação de Conteúdos Através de Funções de Rede Virtuais na Arquitetura de Computação na Borda em Redes Móveis. *Anais do XXXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC 2019* (2019), 766–779.
- [11] BHUSHAN, N., OTHERS. Network densification: the dominant theme for wireless evolution into 5g. In *IEEE Communication Magazine* (2014).
- [12] BORGES, V. C. M., CARDOSO, K. V., CERQUEIRA, E., NOGUEIRA, M., SANTOS, A. Aspirations, challenges, and open issues for software-based 5G networks in extremely dense and heterogeneous scenarios. *EURASIP Journal on Wireless Communications and Networking* 2015, 1 (2015), 164.
- [13] CARVALHO, M., BRITTO, E. D., SILVA, V. F., MACEDO, D. F. QD4G : QoE para Vídeo em Redes D2D / 4G com Aprendizado de Máquina. *Anais do XXXVII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC 2019* (2019), 188–201.

- [14] CHINA MOBILE RESEARCH INSTITUTE. C-ran: The road towards green ran. Disponível em http://labs.chinamobile.com/cran/wp-content/uploads/CRAN_white_paper_v2_5_EN.pdf, 2011. Acessado em: 03/19.
- [15] CHRISTIANI, D. D. M. C., ROCHA, A. D. A., CAMPOS, C. A. V. Um mecanismo distribuído de incentivo baseado em crédito para redes oportunistas.
- [16] CISCO. Cisco Visual Networking Index: Forecast and Trends, 2017–2022 White Paper. Disponível em https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html#\#}_{_}Toc532256795, 2019. Acessado em: 03/18.
- [17] DIGITAL, O. Primeiras redes 5G entram em operação no mundo. Disponível em <https://olhardigital.com.br/video/primeiras-redes-5g-entram-em-operacao-no-mundo/84448>, 2019. Acessado em: 04/19.
- [18] DUA, A., KUMAR, N., BAWA, S. QoS-aware data dissemination for dense urban regions in vehicular ad hoc networks. *Mobile Networks and Applications* 20, 6 (dezembro de 2015), 773–780.
- [19] FEENEY, L. M. Towards a better battery model for inet. *Proceedings of the OM-NeT++ Community Summit 2016* (2016), 1–5.
- [20] FEENEY, L. M., ROHNER, C., GUNNINGBERG, P., LINDGREN, A., ANDERSSON, L. How do the dynamics of battery discharge affect sensor lifetime? In *2014 11th Annual Conference on Wireless On-demand Network Systems and Services (WONS)* (April 2014), p. 49–56.
- [21] FORBES MEDIA LLC. AI and machine learning take center stage at Intel analytics summit. Disponível em <http://www.forbes.com/sites/gilpress/2016/08/16/ai-and-machine-learning-take-center-stage-at-intel-analytics-summit/#3cb58e88108e>, 2016. Acessado em: 10/17.
- [22] GANDOTRA, P., JHA, R. K. Device-to-device communication in cellular networks: A survey. *Journal of Network and Computer Applications* 71 (agosto de 2016), 99–117.
- [23] GUPTA, A., JHA, R. K. A survey of 5G network: Architecture and emerging technologies. *IEEE Access* 3 (2015), 1206–1232.
- [24] HAN, B., GOPALAKRISHNAN, V., JI, L., LEE, S. Network function virtualization: Challenges and opportunities for innovations. *IEEE Communication Magazine* 53, 2 (fevereiro de 2015), 90–97.
- [25] HONG, X., OTHERS. Cognitive radio in 5G: a perspective on energy-spectral efficiency trade-off, jul 2014.
- [26] HÖYHTYÄ, M., APILO, O., LASANEN, M. Energy Consumption Analysis of D2D Communication in 5G Systems: Latest Advances in 3GPP Standardization 4.
- [27] INET API. Modeling Power Consumption. Disponível em <https://inet.omnetpp.org/docs/users-guide/ch-power.html>, 2019. Acessado em: 04/19.

- [28] KARUNAKARAN, P., BAGHERI, H., KATZ, M. Energy efficient multicast data delivery using cooperative mobile clouds. *European Wireless, 2012. 18th European Wireless Conference* (2012), 1–5.
- [29] LEBRE, M.-A., OTHERS. VANET applications: Hot use cases. Relatório Técnico, Cornell University Library, 2014.
- [30] LEE, J., YI, Y., CHONG, S., JIN, Y. Economics of WiFi offloading: Trading delay for cellular capacity. Relatório Técnico, Cornell University Library, 2012.
- [31] LIN, X., ANDREWS, J. G., GHOSH, A., RATASUK, R. An overview of 3GPP device-to-device proximity services. *IEEE Communications Magazine* 52, 4 (2014), 40–48.
- [32] MA, B., SHAH-MANSOURI, H., WONG, V. W. S. A matching approach for power efficient relay selection in full duplex D2D networks. *2016 IEEE International Conference on Communications, ICC 2016* (2016), 0–5.
- [33] MAMMERI, Z. *Wireless and Mobile Networking: IFIP Joint Conference on Mobile Wireless Communications Networks (MWCN'2008) and Personal Wireless Communications (PWC'2008), Toulouse, France, September 30 - October 2, 2008*. IFIP Advances in Information and Communication Technology. Springer US, 2010.
- [34] MASOUDI, M., CAVDAR, C. Cloud vs edge computing for mobile services: Delay-aware decision making to minimize energy consumption. Disponível em <http://arxiv.org/abs/1711.03771>, 2017. Acessado em: 01-2019.
- [35] MIXIM. MiXiM project. Disponível em <http://mixim.sourceforge.net/>, 2019. Acessado em: 09/18.
- [36] MOHSEN NADER TEHRANI, M. U., YANIKOMEROGLU, H. Device-to-device communication in 5g cellular networks: Challenges, solutions, and future directions. *IEEE Communication Magazine* 52, 5 (maio de 2014), 86–92.
- [37] NARDINI, G., VIRDIS, A., STEA, G. Simulating device-to-device communications in OMNeT++ with SimuLTE : scenarios and configurations. *OMNeT++ Community Summit 2016* (2016), 2–5.
- [38] NETO, A., JUNIOR, J., NEUMAN, J., CERQUEIRA, E. Context-aware ehealth information approach for the brazilian primary healthcare system. In *e-Health Networking, Applications & Services (Healthcom), 2013 IEEE 15th International Conference on* (2013), IEEE, p. 274–276.
- [39] NOHRBORG, M. LTE Overview. Disponível em <https://www.3gpp.org/technologies/keywords-acronyms/98-lte>, 2011. Acessado em: 03/19.
- [40] OMNET++. Manual INET Framework for OMNeT++, 2016.
- [41] OMNET API. Mobility models. Disponível em <https://inet.omnetpp.org/docs/showcases/mobility/basic/doc/index.html>, 2019. Acessado em: 09/18.

- [42] PANASONIC. Panasonic overview lithium-ion batteries. Disponível em <https://industrial.panasonic.com/ww/products/batteries/secondary-batteries/lithium-ion>, 2007. Acessado em: 02/19.
- [43] PANWAR, N., SHARMA, S., SINGH, A. K. A survey on 5G: The next generation of mobile communication. *Elsevier Physical Communication* 18, 2 (2016), 64–84.
- [44] PASCA, S. T. V., AKILESH, B., ANAND, A. V., TAMMA, B. R. A NS-3 module for LTE UE energy consumption. *2016 IEEE International Conference on Advanced Networks and Telecommunications Systems, ANTS 2016*, November 2017 (2017).
- [45] PINTO JUNIOR, J. S., SANTOS E SILVA, C. D., DOMINGOS XAVIER, D., OTHERS. Segurança em internet das coisas: Um survey de soluções lightweight. *Revista de Sistemas e Computação-RSC* 7, 2 (2017).
- [46] PRESSMAN, R. *Engenharia de Software - 7.ed.* McGraw Hill Brasil, 2011. Acessado em: 05/18.
- [47] PRIYANKA RAWAT, K. D. S., BONNIN, J. M. Cognitive radio for M2M and internet of things: A survey. *Elsevier Computer Communications* 94 (novembro de 2016), 1–29.
- [48] REBECCHI, F., DE AMORIM, M. D., CONAN, V. Should I seed or should I not: On the remuneration of seeders in D2D offloading. In *2016 IEEE 17th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)* (jun 2016), IEEE, p. 1–9.
- [49] ROUIL, R., CINTRÓN, F. J., BEN MOSBAH, A., GAMBOA, S. Implementation and Validation of an LTE D2D Model for ns-3. *Workshop on ns-3 (WNS3)* (2017), 55–62.
- [50] RÜCKERT, J., RICHERZHAGEN, B., LIDANSKI, E., STEINMETZ, R., HAUSHEER, D. TOPT: Supporting flash crowd events in hybrid overlay-based live streaming. *Proceedings of 2015 14th IFIP Networking Conference, IFIP Networking 2015* (2015).
- [51] SEKIJIMA, L. R. F., FONSECA, N. L. S. Estudo de simulações realistas em redes de acesso móveis LTE / LTE-A. Relatório Técnico, Universidade Estadual de Campinas, 2018.
- [52] THE STATISTICS PORTAL. Global market share of lithium ion battery makers in the 1st quarter of 2018. Disponível em <https://www.statista.com/statistics/235323/lithium-batteries-top-manufacturers/>, 2018. Acessado em: 02/19.
- [53] TUTORIALPOINT. LTE Radio Protocol Architecture. Disponível em https://www.tutorialspoint.com/lte/lte_radio_protocol_architecture.htm, 2019. Acessado em: 05-2019.
- [54] VARGA, A., HORNIG, R. AN OVERVIEW OF THE OMNeT++ SIMULATION ENVIRONMENT. *Proceedings of the First International ICST Conference on Simulation Tools and Techniques for Communications Networks and Systems* (2008).
- [55] VARGAS, A., OPENSIM. OMNeT++ Simulation Manual, 2016.

- [56] VIRDIS, A., NARDINI, G., STEA, G. Modeling unicast device-to-device communications with SimuLTE. *Proceedings of 1st International Workshop on Link and System Level Simulations, IWLS 2016* (2016), 29–34.
- [57] VIRDIS, A., STEA, G., NARDINI, G. SimuLTE – A Modular System-level Simulator for LTE / LTE-A Networks based on OMNeT ++. *Proceedings of SimulTech* (2014), 28–30.
- [58] VIRDIS, A., STEA, G., NARDINI, G. Simulating LTE / LTE-Advanced Networks with SimuLTE. *Simulation and Modeling Methodologies, Technologies and Applications 402*, January (2015), 83–105.
- [59] VUKOBRATOVIC, D., JAKOVETIC, D., SKACHEK, V., BAJOVIC, D., SEJDINOVIC, D., KURT, G. K., HOLLANTI, C., FISCHER, I. CONDENSE: A reconfigurable knowledge acquisition architecture for future 5G IoT. *IEEE Access* 4 (2016), 3360–3378.
- [60] WANG, C. X., HAIDER, F., GAO, X., YOU, X. H., YANG, Y., YUAN, D., AGGOUNE, H. M., HAAS, H., FLETCHER, S., HEPSAYDIR, E. Cellular architecture and key technologies for 5G wireless communication networks. *IEEE Communication Magazine* 52, 2 (fevereiro de 2014), 122–130.
- [61] WANG, X., OTHERS. Cache in the air: Exploiting content caching and delivery techniques for 5G systems. *IEEE Communication Magazine* 52, 2 (fevereiro de 2014), 131–139.
- [62] WHITBECK, J., LOPEZ, Y., LEGUAY, J., CONAN, V., DE AMORIM, M. D. Push-and-track: Saving infrastructure bandwidth through opportunistic forwarding. *Pervasive and Mobile Computing* 8, 5 (2012), 682–697.
- [63] ZHANG, J., XIA, W., YAN, F., SHEN, L. Joint Computation Offloading and Resource Allocation Optimization in Heterogeneous Networks With Mobile Edge Computing.

APÊNDICE A - Arquivos de Configuração do SimuLTE

Arquitetura do Binder

Arquivo de Descrição de Rede - NED

```

1 package lte.corenetwork.binder;
2
3 simple LteBinder{
4     parameters:
5         //QoS Parameters (strings)
6         string priority = "2_4_3_5_1_6_7_8_9";
7         string packetDelayBudget = "0.1_0.15_0.05_0.3_0.1_0.3_0.1
8         _0.3_0.3"; // @unit(s)
9         string packetErrorLossRate = "1e-2_1e-3_1e-3_1e-6_1e-6
10        _1e-6_1e-3_1e-6_1e-6";
11
12         @display("i=block/cogwheel");
13 }

```

Arquivo de Codificação das Funcionalidades - C++

```

1 #include "LteBinder.h"
2 #include "LteDeployer.h"
3 #include "L3AddressResolver.h"
4 #include <cctype>
5 #include "InternetMux.h"
6
7 using namespace std;
8
9 Define_Module(LteBinder);
10
11 void LteBinder::registerDeployer(LteDeployer* pDeployer,
12     MacCellId macCellId)
13 {
14     deployersMap_[macCellId] = pDeployer;
15 }
16
17 void LteBinder::setTransportAppPort(cModule* module, unsigned int
18     counter, cXMLAttributeMap attr)
19 {
20     cXMLAttributeMap::iterator jt;
21     jt = attr.find("type");
22     std::string appType = jt->second;
23
24     std::string basePort;

```

```

25     int port;
26
27     if (appType == "VoIPSender")
28     {
29         basePort = attr.find("baseDestPort")->second;
30         port = atoi(basePort.c_str()) + counter;
31         EV << "LteBinder::setTransportAppPort_ setting_dest_port
32 .....to_" << port << endl;
33
34         module->par("destPort") = port;
35     }
36     else if (appType == "VoIPReceiver")
37     {
38         basePort = attr.find("baseLocalPort")->second;
39         port = atoi(basePort.c_str()) + counter;
40         EV << "LteBinder::setTransportAppPort_ setting_Local_port
41 .....to_" << port << endl;
42
43         module->par("localPort") = port;
44     }
45 }
46
47 void LteBinder::parseParam(cModule* module, cXMLAttributeMap attr)
48 {
49     cXMLAttributeMap::iterator it;
50     for (it = attr.begin(); it != attr.end(); ++it)
51     {
52         if (it->first == "num" || it->first == "type" || it->first
53 == "baseLocalPort" || it->first == "baseDestPort")
54             continue;
55         if (isdigit((it->second)[0]))
56         {
57             if (it->second.find(".") != string::npos)
58                 module->par((it->first).c_str()) =
59                 atof((it->second).c_str());
60             else
61                 module->par((it->first).c_str()) =
62                 atoi((it->second).c_str());
63         }
64         else
65             module->par((it->first).c_str()) = it->second;
66     }
67 }
68
69 cModule* LteBinder::createNodeB(EnbType type)
70 {
71     cModule* lteInternet = getSimulation()->getModuleByPath
72 ("lteInternet");
73
74     cModuleType *mt = cModuleType::get
75 ("lte.corenetwork.nodes.eNodeB");
76
77     cDatarateChannel *iChOut = cDatarateChannel::create
78 ("internetChannelOut");
79     cDatarateChannel *iChIn = cDatarateChannel::create
80 ("internetChannelIn");
81
82     iChOut->setDatarate(0.0);
83     iChIn->setDatarate(0.0);
84
85     cModule *enodeb = mt->create("enodeb", getSimulation()->
86 getSystemModule());
87     MacNodeId cellId = registerNode(enodeb, ENODEB);

```

```

88
89     lteInternet->setGateSize("peerLteIp", (lteInternet->
90     gateSize("peerLteIp") + 1));
91
92     enodeb->gate("peerLteIp$o")->connectTo(lteInternet->gate
93     ("peerLteIp$i", lteInternet->gateSize("peerLteIp$i") - 1),
94     iChOut);
95     lteInternet->gate("peerLteIp$o", lteInternet->gateSize
96     ("peerLteIp$o") - 1)->connectTo(enodeb->gate
97     ("peerLteIp$i"), iChIn);
98
99     cModule *mux = lteInternet->getSubmodule("mux");
100
101     mux->setGateSize("inExt", (mux->gateSize("inExt") + 1));
102     mux->setGateSize("outExt", (mux->gateSize("outExt") + 1));
103
104     cModuleType *it = cModuleType::get
105     ("lte.corenetwork.lteip.InternetQueue");
106     cModule *iQueue = it->create("internetqueue", lteInternet);
107
108     lteInternet->gate("peerLteIp$i", lteInternet->gateSize
109     ("peerLteIp$i") - 1)->connectTo(mux->gate("inExt",
110     mux->gateSize("inExt") - 1));
111     mux->gate("outExt", mux->gateSize("outExt") - 1)->connectTo
112     (iQueue->gate("lteIpIn"));
113     iQueue->gate("internetChannelOut")->connectTo(
114     lteInternet->gate("peerLteIp$o",
115     lteInternet->gateSize("peerLteIp$o") - 1));
116
117     (dynamic_cast<InternetMux*>(mux))->setRoutingEntry(cellId,
118     mux->gate("outExt", mux->gateSize("outExt") - 1));
119
120     enodeb->finalizeParameters();
121     enodeb->buildInside();
122     enodeb->scheduleStart(simTime());
123
124     iQueue->finalizeParameters();
125     iQueue->buildInside();
126     iQueue->scheduleStart(simTime());
127
128     iQueue->callInitialize();
129
130     iChOut->callInitialize();
131     iChIn->callInitialize();
132
133     LteDeployer * deployer = check_and_cast<LteDeployer*>(
134     enodeb->getSubmodule("deployer"));
135     registerDeployer(deployer, cellId);
136
137     EV << "LteBinder::createNodeB_-_enbType_set_to_"
138     << type << endl;
139     deployer->setEnbType(type);
140
141     deployer->preInitialize();
142
143     return enodeb;
144 }
145
146 void LteBinder::transportAppAttach(cModule* parentModule,
147     cModule* appModule, std::string transport)
148 {
149     string appTgateOut, appTgateIn;
150

```

```

151     if (transport == "udp" || transport == "tcp")
152     {
153         appTgateOut = transport + "Out";
154         appTgateIn = transport + "In";
155     }
156     else
157         throw cRuntimeError("LteBinder::transportAppAttach():
158 unrecognized_transport_layer_%s", transport.c_str());
159
160     cModule* tLayer = parentModule->getSubmodule(transport.c_str());
161
162     tLayer->setGateSize("appIn", (tLayer->gateSize("appIn") + 1));
163     tLayer->setGateSize("appOut", (tLayer->gateSize("appOut") + 1));
164
165     appModule->gate(appTgateOut.c_str())->connectTo(
166         tLayer->gate("appIn", (tLayer->gateSize("appIn") - 1)));
167
168     tLayer->gate("appOut", (tLayer->gateSize("appOut") - 1))->
169     connectTo(appModule->gate(appTgateIn.c_str()));
170 }
171
172 void LteBinder::attachAppModule(cModule *parentModule,
173 std::string IPAddr, cXMLAttributeMap attr, int counter)
174 {
175     cXMLAttributeMap::iterator jt;
176     jt = attr.find("type");
177     std::string appType = jt->second;
178
179     EV << NOW << "_LteBinder::attachAppModule_" << appType <<
180     "_from_" << parentModule->getName() << "_to_IP_"
181     << IPAddr.c_str() << endl;
182
183     cModuleType *mt = NULL;
184     cModule *module = NULL;
185
186     if (appType == "UDPSink")
187     {
188         mt = cModuleType::get("inet.applications.udpapp.UDPSink");
189         module = mt->create("udpApp", parentModule);
190         transportAppAttach(parentModule, module, string("udp"));
191     }
192     else if (appType == "UDPBasicApp")
193     {
194         mt = cModuleType::get("inet.applications.udpapp.
195 unrecognized_UDPBasicApp");
196         module = mt->create("udpApp", parentModule);
197         module->par("destAddresses") = IPAddr;
198         transportAppAttach(parentModule, module, string("udp"));
199     }
200     else if (appType == "VoIPSender")
201     {
202         mt = cModuleType::get("VoIP.VoIPSender");
203         module = mt->create("udpApp", parentModule);
204         module->par("destAddress") = IPAddr;
205         transportAppAttach(parentModule, module, string("udp"));
206         if (counter != -1)
207             setTransportAppPort(module, counter, attr);
208     }
209     else if (appType == "VoIPReceiver")
210     {
211         mt = cModuleType::get("VoIP.VoIPReceiver");
212         std::stringstream str;

```

```

213         str << "udpApp";
214         if (counter != -1)
215             str << counter;
216         module = mt->create(str.str().c_str(), parentModule);
217         transportAppAttach(parentModule, module, string("udp"));
218         if (counter != -1)
219             setTransportAppPort(module, counter, attr);
220     }
221     else if (appType == "TCPBasicClientApp")
222     {
223         mt = cModuleType::get("inet.applications.tcpapp.
224         TCPBasicClientApp");
225         module = mt->create("tcpApp", parentModule);
226         module->par("connectAddress") = IPAddr;
227         transportAppAttach(parentModule, module, string("tcp"));
228     }
229     else if (appType == "TCPSinkApp")
230     {
231         mt = cModuleType::get("inet.applications.tcpapp.
232         TCPSinkApp");
233         module = mt->create("tcpApp", parentModule);
234         transportAppAttach(parentModule, module, string("tcp"));
235     }
236     else if (appType == "VoDUDPServer")
237     {
238         mt = cModuleType::get("vod.VoDUDPServer");
239         module = mt->create("vodServer", parentModule);
240         module->par("destAddresses") = IPAddr;
241         transportAppAttach(parentModule, module, string("udp"));
242     }
243     else if (appType == "VoDUDPClient")
244     {
245         mt = cModuleType::get("vod.VoDUDPClient");
246         module = mt->create("vodClient", parentModule);
247         transportAppAttach(parentModule, module, string("udp"));
248     }
249     else if (appType == "gaming")
250     {
251         mt = cModuleType::get("gaming.gaming");
252         module = mt->create("gaming", parentModule);
253         module->par("destAddresses") = IPAddr;
254         transportAppAttach(parentModule, module, string("udp"));
255     }
256     else if (appType == "ftp_3gpp_sender")
257     {
258         mt = cModuleType::get("ftp_3gpp.ftp_3gpp_sender");
259         module = mt->create("ftp_3gpp_sender", parentModule);
260         module->par("destAddresses") = IPAddr;
261         transportAppAttach(parentModule, module, string("udp"));
262     }
263     else if (appType == "ftp_3gpp_sink")
264     {
265         mt = cModuleType::get("ftp_3gpp.ftp_3gpp_sink");
266         module = mt->create("udpApp", parentModule);
267         transportAppAttach(parentModule, module, string("udp"));
268     }
269     else
270     {
271         throw cRuntimeError("LteBinder::attachAppModule():
272         unrecognized_application_type_%s", appType.c_str());

```

```

273     }
274
275     parseParam(module, attr);
276     module->finalizeParameters();
277     module->buildInside();
278     module->scheduleStart(simTime());
279 }
280
281 void LteBinder::unregisterNode(MacNodeId id){
282     if(nodeIds_.erase(id) != 1){
283         EV_ERROR << "Cannot_unregister_node_-_node_id_\\"
284             << id << "\"_not_found";
285     }
286     std::map<IPv4Address, MacNodeId>::iterator it;
287     for(it = macNodeIdToIPAddr_.begin(); it !=
288         macNodeIdToIPAddr_.end(); ){
289         if(it->second == id){
290             macNodeIdToIPAddr_.erase(it++);
291         } else {
292             it++;
293         }
294     }
295 }
296
297 MacNodeId LteBinder::registerNode(cModule *module, LteNodeType
298     type, MacNodeId masterId)
299 {
300     Enter_Method("registerNode");
301
302     MacNodeId macNodeId = -1;
303
304     if (type == UE)
305     {
306         macNodeId = macNodeIdCounter_[2]++;
307     }
308     else if (type == RELAY)
309     {
310         macNodeId = macNodeIdCounter_[1]++;
311     }
312     else if (type == ENODEB)
313     {
314         macNodeId = macNodeIdCounter_[0]++;
315     }
316
317     EV << "LteBinder::Assigning_to_module_" << module->getName()
318         << "_with_OmnetId_" << module->getId()
319         << "_and_MacNodeId_" << macNodeId
320         << "\n";
321
322     nodeIds_[macNodeId] = module->getId();
323
324     module->par("macNodeId") = macNodeId;
325
326     if (type == RELAY || type == UE)
327     {
328         registerNextHop(masterId, macNodeId);
329     }
330     else if (type == ENODEB)
331     {
332         module->par("macCellId") = macNodeId;
333         registerNextHop(macNodeId, macNodeId);
334     }

```

```

335     return macNodeId;
336 }
337
338 void LteBinder::registerNextHop (MacNodeId masterId ,
339     MacNodeId slaveId)
340 {
341     Enter_Method("registerNextHop");
342     EV << "LteBinder::Registering_slave_" << slaveId <<
343     "_to_master_" << masterId << "\n";
344
345     if (masterId != slaveId)
346     {
347         dMap_[masterId][slaveId] = true;
348     }
349
350     if (nextHop_.size() <= slaveId)
351         nextHop_.resize(slaveId + 1);
352     nextHop_[slaveId] = masterId;
353 }
354
355 void LteBinder::initialize(int stage)
356 {
357     if (stage == 0)
358     {
359         const char * stringa;
360
361         std::vector<int> apppriority;
362         std::vector<double> appdelay;
363         std::vector<double> applossrate;
364
365         stringa = par("priority");
366         apppriority = cStringTokenizer(stringa).asIntVector();
367         stringa = par("packetDelayBudget");
368         appdelay = cStringTokenizer(stringa).asDoubleVector();
369         stringa = par("packetErrorLossRate");
370         applossrate = cStringTokenizer(stringa).asDoubleVector();
371
372         for (int i = 0; i < LTE_QCI_CLASSES; i++)
373         {
374             QCIParam_[i].priority = apppriority[i];
375             QCIParam_[i].packetDelayBudget = appdelay[i];
376             QCIParam_[i].packetErrorLossRate = applossrate[i];
377         }
378         nodesConfigured_ = false;
379     }
380     if (stage == 1)
381     {
382         MacNodeId maxUe = macNodeIdCounter_[2];
383         d2dPeeringCapability_ = new bool*[maxUe];
384         for (int i=0; i<maxUe; i++)
385         {
386             d2dPeeringCapability_[i] = new bool[maxUe];
387             for (int j=0; j<maxUe; j++)
388             {
389                 d2dPeeringCapability_[i][j] = false;
390             }
391         }
392     }
393 }
394
395 std::string LteBinder::increment_address(
396     const char* address_string)

```



```

397 {
398     IPv4Address addr(address_string);
399     return IPv4Address(addr.getInt() + 1).str();
400 }
401
402 int LteBinder::getQCIPriority(int QCI)
403 {
404     return QCIParam_[QCI - 1].priority;
405 }
406
407 double LteBinder::getPacketDelayBudget(int QCI)
408 {
409     return QCIParam_[QCI - 1].packetDelayBudget;
410 }
411
412 double LteBinder::getPacketErrorLossRate(int QCI)
413 {
414     return QCIParam_[QCI - 1].packetErrorLossRate;
415 }
416
417 void LteBinder::unregisterNextHop(MacNodeId masterId,
418     MacNodeId slaveId)
419 {
420     Enter_Method("unregisterNextHop");
421     EV << "LteBinder::Unregistering_slave_" << slaveId <<
422     "_from_master_" << masterId << "\n";
423     dMap_[masterId][slaveId] = false;
424
425     if (nextHop_.size() <= slaveId)
426         return;
427     nextHop_[slaveId] = 0;
428 }
429
430 OmnetId LteBinder::getOmnetId(MacNodeId nodeId)
431 {
432     std::map<int, OmnetId>::iterator it = nodeIds_.find(nodeId);
433     if (it != nodeIds_.end()) {
434         return it->second;
435     } else {
436         return 0;
437     }
438 }
439
440 MacNodeId LteBinder::getMacNodeIdFromOmnetId(OmnetId id) {
441     std::map<int, OmnetId>::iterator it;
442     for (it = nodeIds_.begin(); it != nodeIds_.end(); ++it) {
443         if (it->second == id) {
444             return it->first;
445         }
446     }
447     return 0;
448 }
449
450 MacNodeId LteBinder::getNextHop(MacNodeId slaveId)
451 {
452     Enter_Method("getNextHop");
453     if (slaveId >= nextHop_.size())
454         throw cRuntimeError("LteBinder::getNextHop():
455         bad_slave_id_%d", slaveId);
456     return nextHop_[slaveId];
457 }
458

```

```

459 void LteBinder::registerName(MacNodeId nodeId, const char* moduleName)
460 {
461     int len = strlen(moduleName);
462     macNodeIdToModuleName_[nodeId] = new char[len+1];
463     strcpy(macNodeIdToModuleName_[nodeId], moduleName);
464 }
465
466 const char* LteBinder::getModuleNameByMacNodeId(MacNodeId nodeId)
467 {
468     if (macNodeIdToModuleName_.find(nodeId) ==
469         macNodeIdToModuleName_.end())
470         throw cRuntimeError("LteBinder::getModuleNameByMacNodeId
471 ....._node_ID_not_found");
472     return macNodeIdToModuleName_[nodeId];
473 }
474
475 ConnectedUesMap LteBinder::getDeployedUes(MacNodeId localId,
476     Direction dir)
477 {
478     Enter_Method("getDeployedUes");
479     return dMap_[localId];
480 }
481
482 void LteBinder::registerX2Port(X2NodeId nodeId, int port)
483 {
484     if (x2ListeningPorts_.find(nodeId) == x2ListeningPorts_.end() )
485     {
486         std::list<int> ports;
487         ports.push_back(port);
488         x2ListeningPorts_[nodeId] = ports;
489     }
490     else
491     {
492         x2ListeningPorts_[nodeId].push_back(port);
493     }
494 }
495
496 int LteBinder::getX2Port(X2NodeId nodeId)
497 {
498     if (x2ListeningPorts_.find(nodeId) == x2ListeningPorts_.end() )
499         throw cRuntimeError("LteBinder::getX2Port_-
500 .....No_ports_available_on_node_%d", nodeId);
501
502     int port = x2ListeningPorts_[nodeId].front();
503     x2ListeningPorts_[nodeId].pop_front();
504     return port;
505 }
506
507 Cqi LteBinder::meanCqi(std::vector<Cqi> bandCqi, MacNodeId id,
508     Direction dir)
509 {
510     std::vector<Cqi>::iterator it;
511     Cqi mean=0;
512     for (it=bandCqi.begin(); it!=bandCqi.end(); ++it)
513     {
514         mean+=*it;
515     }
516     mean/=bandCqi.size();
517
518     if (mean==0)
519         mean = 1;
520

```

```

521     return mean;
522 }
523
524 void LteBinder::addD2DCapability(MacNodeId src , MacNodeId dst)
525 {
526     if (src < UE_MIN_ID || src >= macNodeIdCounter_[2] || dst
527         < UE_MIN_ID || dst >= macNodeIdCounter_[2])
528         throw cRuntimeError("LteBinder::addD2DCapability_-
529 .....NodeId_not_valid_._Src_%d_Dst_%d", src , dst);
530
531     d2dPeeringCapability_[src][dst] = true;
532     d2dPeeringMode_[src][dst] = DM;
533
534     EV << "LteBinder::addD2DCapability_-UE_" << src <<
535         "_may_transmit_to_UE_" << dst << "_using_D2D" << endl;
536 }
537
538 bool LteBinder::checkD2DCapability(MacNodeId src , MacNodeId dst)
539 {
540     if (src < UE_MIN_ID || src >= macNodeIdCounter_[2] || dst <
541         UE_MIN_ID || dst >= macNodeIdCounter_[2])
542         throw cRuntimeError("LteBinder::checkD2DCapability_-
543 .....NodeId_not_valid_._Src_%d_Dst_%d", src , dst);
544
545     return d2dPeeringCapability_[src][dst];
546 }
547
548 std::map<MacNodeId , std::map<MacNodeId , LteD2DMode> >*
549     LteBinder::getD2DPeeringModeMap()
550 {
551     return &d2dPeeringMode_ ;
552 }
553
554 LteD2DMode LteBinder::getD2DMode(MacNodeId src , MacNodeId dst)
555 {
556     if (src < UE_MIN_ID || src >= macNodeIdCounter_[2] ||
557         dst < UE_MIN_ID || dst >= macNodeIdCounter_[2])
558         throw cRuntimeError("LteBinder::getD2DMode_-
559 .....NodeId_not_valid_._Src_%d_Dst_%d", src , dst);
560
561     return d2dPeeringMode_[src][dst];
562 }
563
564 void LteBinder::registerMulticastGroup(MacNodeId nodeId ,
565     int32 groupId)
566 {
567     if (multicastGroupMap_.find(nodeId) ==
568         multicastGroupMap_.end())
569     {
570         MulticastGroupIdSet newSet;
571         newSet.insert(groupId);
572         multicastGroupMap_[nodeId] = newSet;
573     }
574     else
575     {
576         multicastGroupMap_[nodeId].insert(groupId);
577     }
578 }
579
580 bool LteBinder::isInMulticastGroup(MacNodeId nodeId ,
581     int32 groupId)
582 {

```

```

583     if (multicastGroupMap_.find(nodeId) ==
584         multicastGroupMap_.end())
585         return false;
586     if (multicastGroupMap_[nodeId].find(groupId) ==
587         multicastGroupMap_[nodeId].end())
588         return false;
589
590     return true;
591 }
592
593 void LteBinder::updateUeInfoCellId(MacNodeId id,
594     MacCellId newCellId)
595 {
596     std::vector<UeInfo*>::iterator it = ueList_.begin();
597     for (; it != ueList_.end(); ++it)
598     {
599         if ((*it)->id == id)
600         {
601             (*it)->cellId = newCellId;
602             return;
603         }
604     }
605 }
606
607 void LteBinder::addUeHandoverTriggered(MacNodeId nodeId)
608 {
609     ueHandoverTriggered_.insert(nodeId);
610 }
611
612 bool LteBinder::hasUeHandoverTriggered(MacNodeId nodeId)
613 {
614     if (ueHandoverTriggered_.find(nodeId) ==
615         ueHandoverTriggered_.end())
616         return false;
617     return true;
618 }
619
620 void LteBinder::removeUeHandoverTriggered(MacNodeId nodeId)
621 {
622     ueHandoverTriggered_.erase(nodeId);
623 }

```

Arquivo de Biblioteca (.h) para as Funcionalidades

```

1  #ifndef LTE_LTEBINDER_H
2  #define _LTE_LTEBINDER_H_
3
4  #include <omnetpp.h>
5  #include <string>
6  #include "LteCommon.h"
7  #include "IPv4Address.h"
8  #include "L3Address.h"
9  #include "PhyPisaData.h"
10 #include "ExtCell.h"
11
12 using namespace inet;
13
14 class LteBinder : public cSimpleModule
15 {
16     private:
17         typedef std::map<MacNodeId, std::map<MacNodeId, bool>
18             > DeployedUesMap;
19         typedef std::map<MacCellId, LteDeployer*> DeployerList;

```

```

20
21     std::map<IPv4Address, MacNodeId> macNodeIdToIPAddress_;
22     std::map<MacNodeId, char*> macNodeIdToModuleName_;
23     DeployerList deployersMap_;
24     std::vector<MacNodeId> nextHop_;
25     std::map<int, OmnetId> nodeIds_;
26
27     ExtCellList extCellList_;
28
29     std::vector<EnbInfo*> enbList_;
30
31     std::vector<UeInfo*> ueList_;
32
33     MacNodeId macNodeIdCounter_[3];
34     DeployedUesMap dMap_;
35     QCIParameters QCIParam_[LTE_QCI_CLASSES];
36
37     bool nodesConfigured_;
38
39     std::string increment_address(const char* address_string);
40
41     typedef std::map<X2NodeId, std::list<int>
42 > X2ListeningPortMap;
43     X2ListeningPortMap x2ListeningPorts_;
44
45     std::map<MacNodeId, std::map<MacNodeId, L3Address>
46 > x2PeerAddress_;
47
48     bool **d2dPeeringCapability_;
49     std::map<MacNodeId, std::map<MacNodeId, LteD2DMode>
50 > d2dPeeringMode_;
51
52     typedef std::set<uint32> MulticastGroupIdSet;
53     std::map<MacNodeId, MulticastGroupIdSet> multicastGroupMap_;
54
55     std::set<MacNodeId> ueHandoverTriggered_;
56 protected:
57     virtual void initialize(int stages);
58
59     virtual int numInitStages() const { return INITSTAGE_LAST; }
60
61     virtual void handleMessage(cMessage *msg)
62     {
63     }
64
65     void attachAppModule(cModule *parentModule, std::string IPAddr,
66         cXMLAttributeMap attr, int counter);
67
68     void transportAppAttach(cModule* parentModule, cModule*
69 appModule, std::string transport);
70
71     void setTransportAppPort(cModule* module, unsigned int
72 counter, cXMLAttributeMap attr);
73
74     void parseParam(cModule* module, cXMLAttributeMap attr);
75
76 public:
77     LteBinder()
78     {
79         macNodeIdCounter_[0] = ENB_MIN_ID;
80         macNodeIdCounter_[1] = RELAY_MIN_ID;
81         macNodeIdCounter_[2] = UE_MIN_ID;
82     }
83
84     void registerDeployer(LteDeployer* pDeployer,
85         MacCellId macCellId);

```

```

86
87 virtual ~LteBinder()
88 {
89     while(enbList_.size() > 0){
90         delete enbList_.back();
91         enbList_.pop_back();
92     }
93 }
94 int getQCIPriority(int);
95 double getPacketDelayBudget(int);
96 double getPacketErrorLossRate(int);
97
98 cModule* createNodeB(EnbType type);
99
100 MacNodeId registerNode(cModule *module, LteNodeType type,
101 MacNodeId masterId = 0);
102
103 void unregisterNode(MacNodeId id);
104
105 void registerNextHop(MacNodeId masterId, MacNodeId slaveId);
106
107 void unregisterNextHop(MacNodeId masterId, MacNodeId slaveId);
108
109 OmnetId getOmnetId(MacNodeId nodeId);
110
111 MacNodeId getMacNodeIdFromOmnetId(OmnetId id);
112
113 MacNodeId getNextHop(MacNodeId slaveId);
114
115 MacNodeId getMacNodeId(IPv4Address address)
116 {
117     if (macNodeIdToIPAddress_.find(address) ==
118         macNodeIdToIPAddress_.end())
119         return 0;
120     return macNodeIdToIPAddress_[address];
121 }
122
123 X2NodeId getX2NodeId(IPv4Address address)
124 {
125     return getMacNodeId(address);
126 }
127
128 void setMacNodeId(IPv4Address address, MacNodeId nodeId)
129 {
130     macNodeIdToIPAddress_[address] = nodeId;
131 }
132
133 void setX2NodeId(IPv4Address address, X2NodeId nodeId)
134 {
135     setMacNodeId(address, nodeId);
136 }
137 L3Address getX2PeerAddress(X2NodeId srcId, X2NodeId destId)
138 {
139     return x2PeerAddress_[srcId][destId];
140 }
141 void setX2PeerAddress(X2NodeId srcId, X2NodeId destId,
142 L3Address interfAddr)
143 {
144     std::pair<X2NodeId, L3Address> p(destId, interfAddr);
145     x2PeerAddress_[srcId].insert(p);
146 }
147

```

```

148     void registerName(MacNodeId nodeId, const char* moduleName);
149
150     const char* getModuleNameByMacNodeId(MacNodeId nodeId);
151
152     ConnectedUesMap getDeployedUes(MacNodeId localId, Direction dir);
153     PhyPisaData phyPisaData;
154
155     int getNodeCount() {
156         return nodeIds_.size();
157     }
158
159     int addExtCell(ExtCell* extCell)
160     {
161         extCellList_.push_back(extCell);
162         return extCellList_.size() - 1;
163     }
164
165     ExtCellList getExtCellList()
166     {
167         return extCellList_;
168     }
169
170     void addEnbInfo(EnbInfo* info)
171     {
172         enbList_.push_back(info);
173     }
174
175     std::vector<EnbInfo*> * getEnbList()
176     {
177         return &enbList_;
178     }
179
180     void addUeInfo(UeInfo* info)
181     {
182         ueList_.push_back(info);
183     }
184
185     std::vector<UeInfo*> * getUeList()
186     {
187         return &ueList_;
188     }
189
190     Cqi meanCqi(std::vector<Cqi> bandCqi, MacNodeId id, Direction dir);
191
192     void registerX2Port(X2NodeId nodeId, int port);
193     int getX2Port(X2NodeId nodeId);
194
195     void addD2DCapability(MacNodeId src, MacNodeId dst);
196     bool checkD2DCapability(MacNodeId src, MacNodeId dst);
197     std::map<MacNodeId, std::map<MacNodeId, LteD2DMode>>*
198     getD2DPeeringModeMap();
199     void setD2DMode(MacNodeId src, MacNodeId dst, LteD2DMode mode);
200     LteD2DMode getD2DMode(MacNodeId src, MacNodeId dst);
201
202     void registerMulticastGroup(MacNodeId nodeId, int32 groupId);
203     bool isInMulticastGroup(MacNodeId nodeId, int32 groupId);
204
205     void addUeHandoverTriggered(MacNodeId nodeId);
206     bool hasUeHandoverTriggered(MacNodeId nodeId);
207     void removeUeHandoverTriggered(MacNodeId nodeId);
208     void updateUeInfoCellId(MacNodeId nodeId, MacCellId cellId);
209 };
210

```

211 **#endif**

Arquitetura da Estação Base - eNodeB

Arquivo de Descrição de Rede da eNodeB- NED

```

1 package lte.corenetwork.nodes;
2
3 import inet.linklayer.ppp.PPPInterface;
4 import inet.mobility.static.StationaryMobility;
5 import inet.networklayer.common.InterfaceTable;
6 import inet.networklayer.ipv4.IPv4RoutingTable;
7 import inet.networklayer.contract.IRoutingTable;
8 import inet.networklayer.contract.INetworkLayer;
9 import inet.applications.contract.IUDPApp;
10 import inet.applications.contract.ITCPApp;
11 import inet.applications.contract.ISCTPApp;
12 import inet.transportlayer.udp.UDP;
13 import inet.transportlayer.tcp.TCP;
14 import inet.transportlayer.sctp.SCTP;
15 import lte.x2.LteX2App;
16 import lte.stack.phy.LteNicEnb;
17 import lte.stack.phy.LteNicEnbD2D;
18 import lte.corenetwork.deployer.LteDeployer;
19 import lte.epc.gtp.GtpUserSimplified;
20 import lte.epc.TrafficFlowFilterSimplified;
21 import lte.epc.gtp.GtpUserX2;
22
23 module eNodeB
24 {
25     parameters:
26         @networkNode();
27         @display("i=device/antennatower;bgb=814,437");
28
29         /// Node specs
30         string nodeType = "ENODEB";
31         int macNodeId = default(0);
32         int macCellId = default(0);
33         double txPower @unit(mw) = default(100mw);
34         bool d2dCapable = default(false);
35
36         /// Network Layer specs
37         string networkLayerType = default("IPv4NetworkLayer");
38         string routingTableType = default("IPv4RoutingTable");
39         *.interfaceTableModule = default(absPath(".interfaceTable"));
40         *.routingTableModule = default(routingTableType != "" ?
41         absPath(".routingTable") : "");
42
43         /// Num apps
44         int numTcpApps = default(0);
45         int numUdpApps = default(0);
46         int numX2Apps = default(0);
47
48     gates:
49         inout ppp;
50         input radioIn @directIn;
51         inout x2[];
52
53     submodules:
54
55         interfaceTable: InterfaceTable {
56             @display("p=50,75;is=s");
57         }

```



```

58
59     routingTable: <routingTableType> like IRoutingTable
60     if routingTableType != "" {
61         parameters:
62             @display("p=50,125;is=s");
63     }
64     mobility: StationaryMobility {
65         @display("p=50,175;is=s");
66     }
67
68     deployer: LteDeployer {
69         @display("p=50,225;is=s");
70     }
71
72     nic: LteNicEnbD2D if d2dCapable {
73         nodeType = nodeType;
74         d2dCapable = d2dCapable;
75         @display("p=330,363");
76     }
77     nic: LteNicEnb if !d2dCapable {
78         nodeType = nodeType;
79         d2dCapable = d2dCapable;
80         @display("p=330,363");
81     }
82     ppp: PPPInterface {
83         @display("p=501,363");
84     }
85     x2ppp[sizeof(x2)]: PPPInterface {
86         @display("p=683,363");
87     }
88
89     networkLayer: <networkLayerType> like INetworkLayer {
90         parameters:
91             @display("p=501,269;q=queue");
92     }
93
94     // ===== UDP ===== //
95     udpApp[numUdpApps]: <> like IUdpApp {
96         @display("p=555,48,row");
97     }
98     udp: UDP {
99         @display("p=501,148");
100     }
101
102     // ===== TCP ===== //
103     tcpApp[numTcpApps]: <> like ITcpApp {
104         @display("p=683,48,row");
105     }
106     tcp: TCP if numTcpApps>0 {
107         @display("p=683,148");
108     }
109
110     // ===== X2AP ===== //
111     x2App[numX2Apps]: LteX2App {
112         @display("p=155,48,row");
113     }
114     sctp: SCTP {
115         @display("p=275,148");
116     }
117
118     trafficFlowFilter: TrafficFlowFilterSimplified {
119         ownerType = nodeType;

```

```

120         @display ("p=330,48");
121     }
122     gtpUser: GtpUserSimplified {
123         @display ("p=434,48");
124     }
125
126     gtpUserX2: GtpUserX2 {
127         @display ("p=410,363");
128     }
129
130     connections allowunconnected:
131
132         nic.radioIn <== radioIn;
133         nic.upperLayerOut ==> trafficFlowFilter.internetFilterGateIn;
134         nic.upperLayerIn <== gtpUser.pppGate;
135         trafficFlowFilter.gtpUserGateOut ==>
136         gtpUser.trafficFlowFilterGate;
137         gtpUser.udpOut ==> udp.appIn++;
138         gtpUser.udpIn <== udp.appOut++;
139
140         ppp.phys <==> ppp;
141         ppp.upperLayerOut ==> networkLayer.ifIn++;
142         ppp.upperLayerIn <== networkLayer.ifOut++;
143
144
145         gtpUserX2.lteStackOut ==> nic.x2$i++;
146         gtpUserX2.lteStackIn <== nic.x2$o++;
147         gtpUserX2.udpOut ==> udp.appIn++;
148         gtpUserX2.udpIn <== udp.appOut++;
149
150         for i=0..sizeof(x2)-1 {
151             x2ppp[i].phys <==> x2[i];
152             x2ppp[i].upperLayerOut ==> networkLayer.ifIn++;
153             x2ppp[i].upperLayerIn <== networkLayer.ifOut++;
154         }
155
156         for i=0..numTcpApps-1 {
157             tcpApp[i].tcpOut ==> tcp.appIn++;
158             tcpApp[i].tcpIn <== tcp.appOut++;
159         }
160         for i=0..numUdpApps-1 {
161             udpApp[i].udpOut ==> udp.appIn++;
162             udpApp[i].udpIn <== udp.appOut++;
163         }
164         for i=0..numX2Apps-1 {
165             x2App[i].sctpOut[0] ==> sctp.from_appl++;
166             x2App[i].sctpIn[0] <== sctp.to_appl++;
167
168             x2App[i].sctpOut[1] ==> sctp.from_appl++;
169             x2App[i].sctpIn[1] <== sctp.to_appl++;
170             x2App[i].x2ManagerIn <== nic.x2$o++;
171             x2App[i].x2ManagerOut ==> nic.x2$i++;
172         }
173
174         udp.ipOut ==> networkLayer.transportIn++;
175         udp.ipIn <== networkLayer.transportOut++;
176         tcp.ipOut ==> networkLayer.transportIn++ if numTcpApps>0;
177         tcp.ipIn <== networkLayer.transportOut++ if numTcpApps>0;
178         sctp.to_ip ==> networkLayer.transportIn++ if numX2Apps>0;
179         sctp.from_ip <== networkLayer.transportOut++ if numX2Apps>0;
180     }

```

Arquitetura de Encaminhamento

Arquivo de Descrição de Rede D2D - NED

```

1 package lte.corenetwork.nodes;
2
3 import inet.mobility.static.StationaryMobility;
4 import lte.stack.phy.LteNicRelay;
5
6 // Relay Module
7 module Relay
8 {
9     parameters:
10         @networkNode;
11         @display("i=device/receiverdish;bgb=250,200");
12
13         // Node specs
14         string nodeType = "RELAY";           // DO NOT CHANGE
15         int masterId = default(0);
16         int macNodeId = default(0);
17         int macCellId = default(0);
18
19     gates:
20         input radioIn[2] @directIn;           // connection to master
21
22     submodules:
23         mobility: StationaryMobility {
24             @display("p=50,75;is=s");
25         }
26         nicUe: LteNicRelay {
27             @display("p=100,150");
28             LtePdcPrrcType = "LtePdcPrrcRelayUe";
29             LteMacType = "LteMacRelayUe";
30         }
31         nicEnb: LteNicRelay {
32             @display("p=200,150");
33             LtePdcPrrcType = "LtePdcPrrcRelayEnb";
34             LteMacType = "LteMacRelayEnb";
35         }
36
37     connections allowunconnected:
38         // Internal Lte connections
39         // nicUe.upperLayerIn <-- nicEnb.upperLayerOut;
40         // nicUe.upperLayerOut --> nicEnb.upperLayerIn;
41
42         // External connection to air interface
43
44         radioIn[0] --> nicUe.radioIn;
45         radioIn[1] --> nicEnb.radioIn;
46     }
47
48 simple UnUuMux
49 {
50     parameters:
51         @display("i=block/dispatch");
52
53     gates:
54         inout Un;           // to/from ueStack
55         inout Uu;           // to/from enbStack
56

```

```
60      inout lowerGate;    // to/from NIC
61  }
```

APÊNDICE B - Gráficos dos Resultados de Transmissão de Dados em cada Cenário

B.1 experimento: Multicast

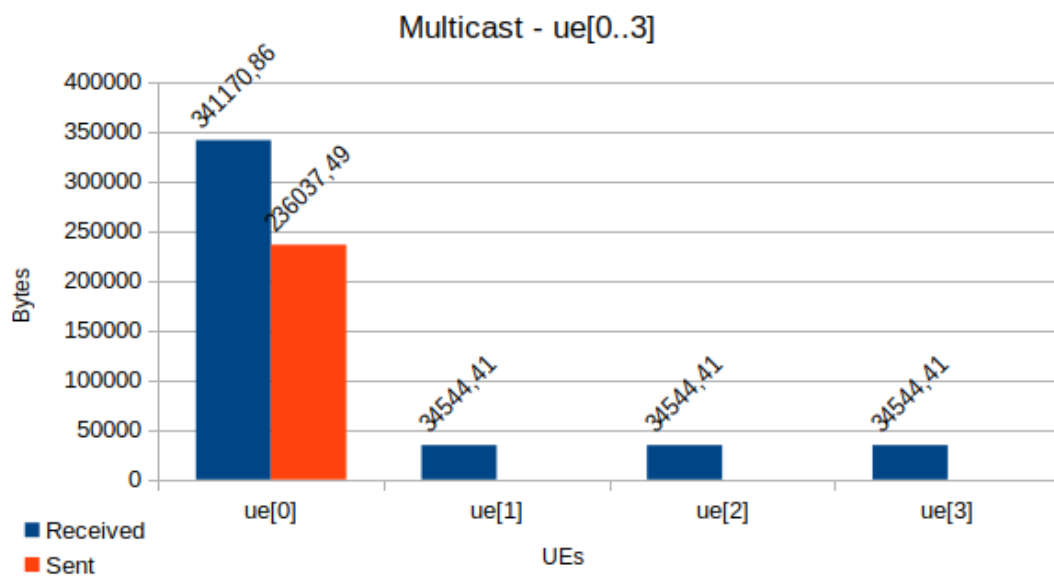


Figura B.1: Transmissão com 4 UEs no experimento Multicast

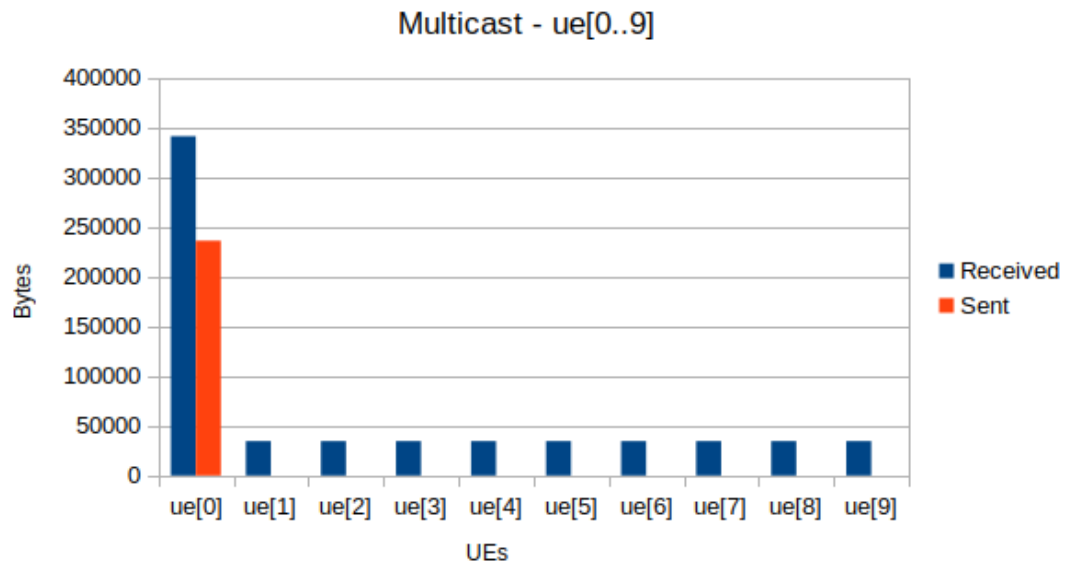


Figura B.2: Transmissão com 10 UEs no experimento multicast.

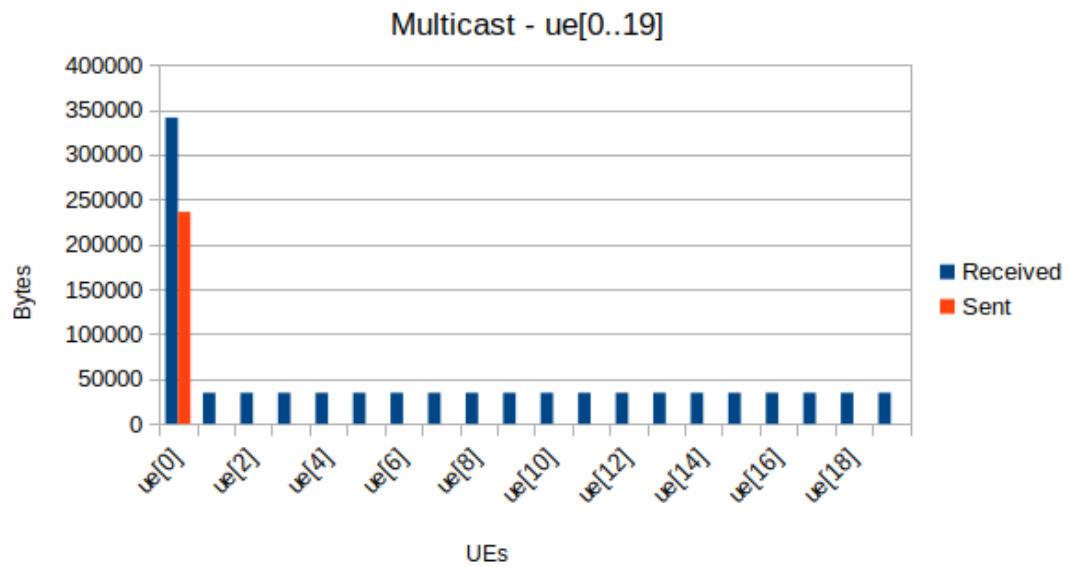


Figura B.3: Transmissão com 20 UEs no experimento multicast.

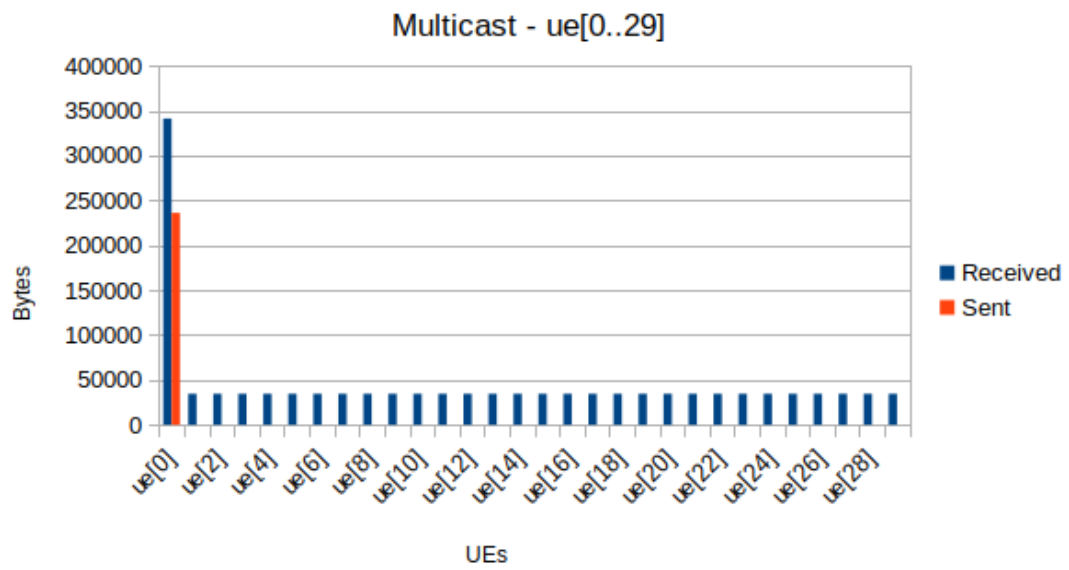


Figura B.4: Transmissão com 30 UEs no experimento multicast.

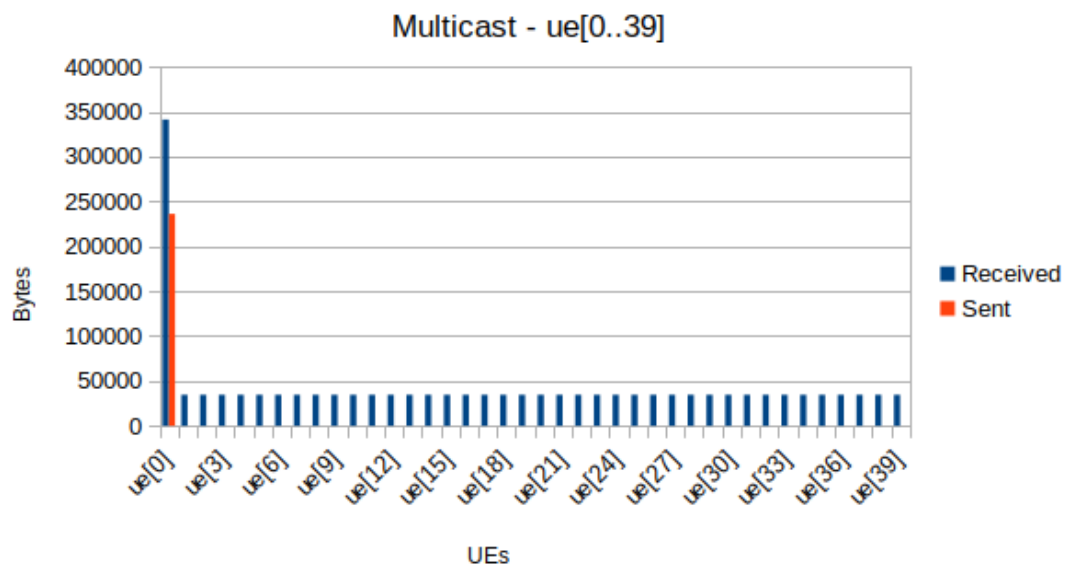


Figura B.5: Transmissão com 40 UEs no experimento multicast.

B.2 experimento: TCP

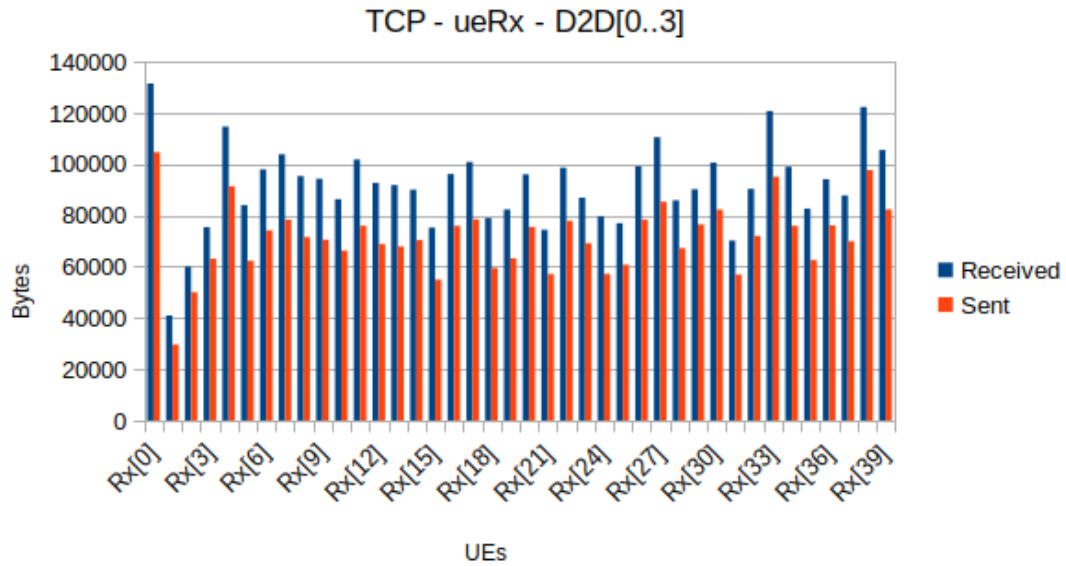


Figura B.6: Transmissão dos nós Rx no experimento TCP com 4 D2D.

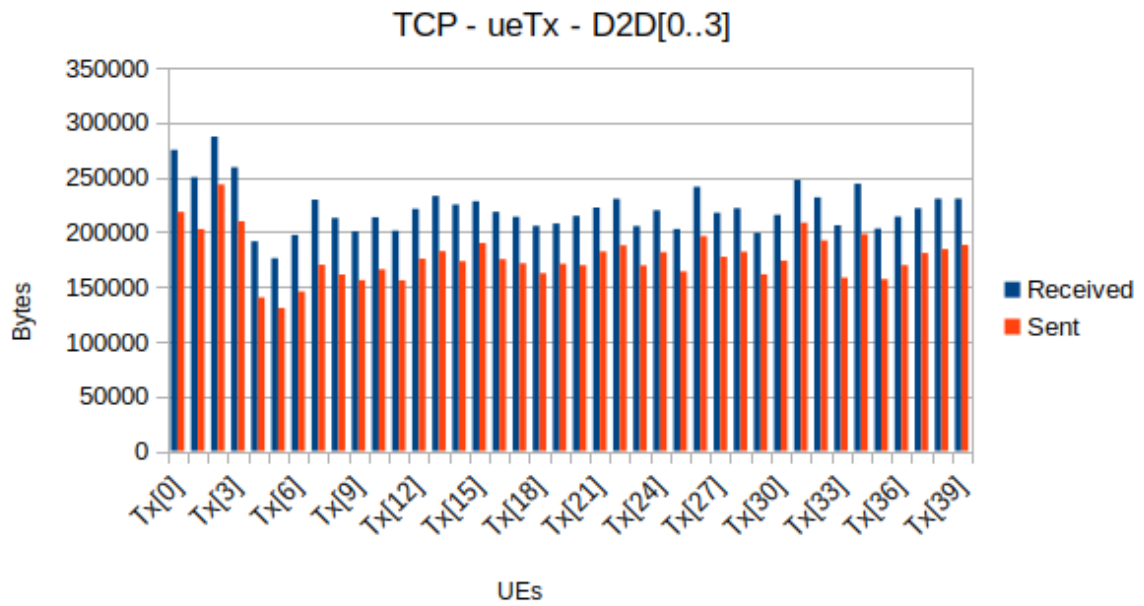


Figura B.7: Transmissão dos nós Tx no experimento TCP com 4 D2D.

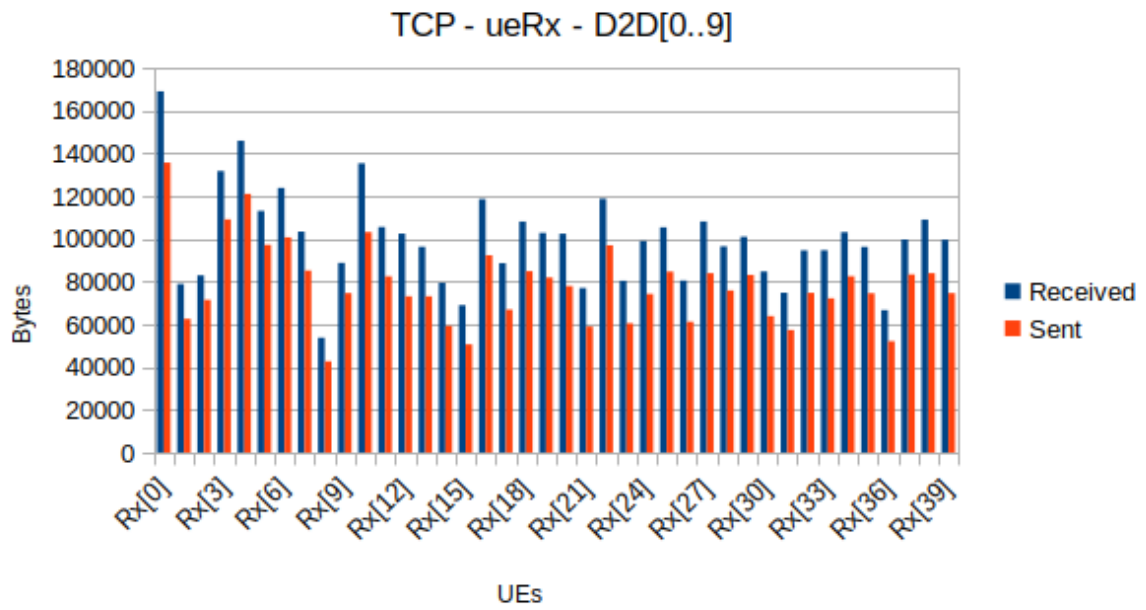


Figura B.8: Transmissão dos nós Rx no experimento TCP com 10 D2D.

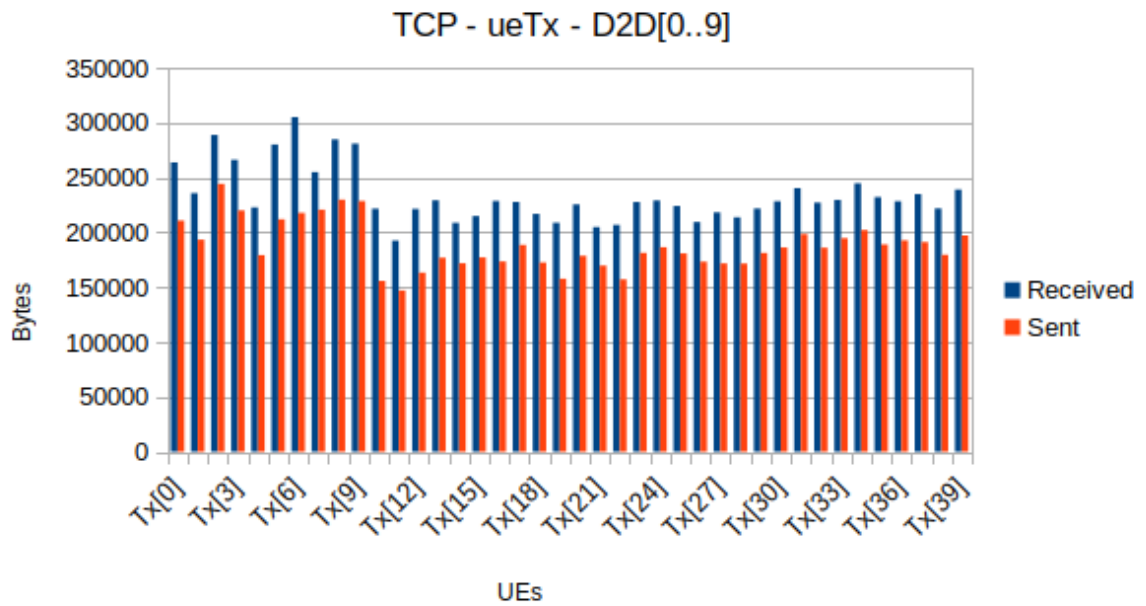


Figura B.9: Transmissão dos nós Tx no experimento TCP com 10 D2D.

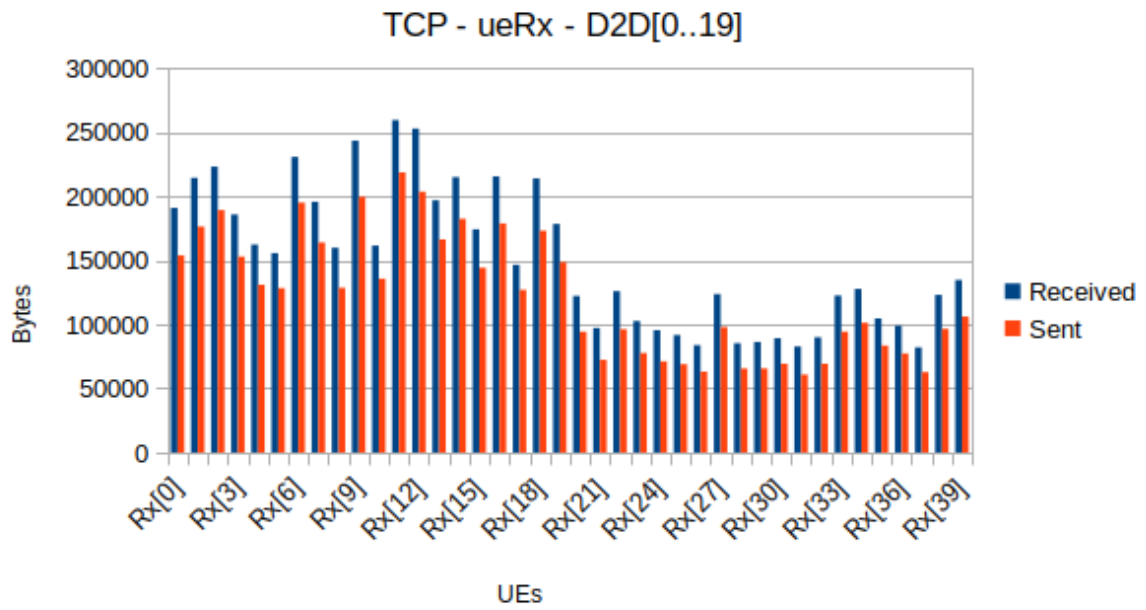


Figura B.10: Transmissão dos nós Rx no experimento TCP com 20 D2D.

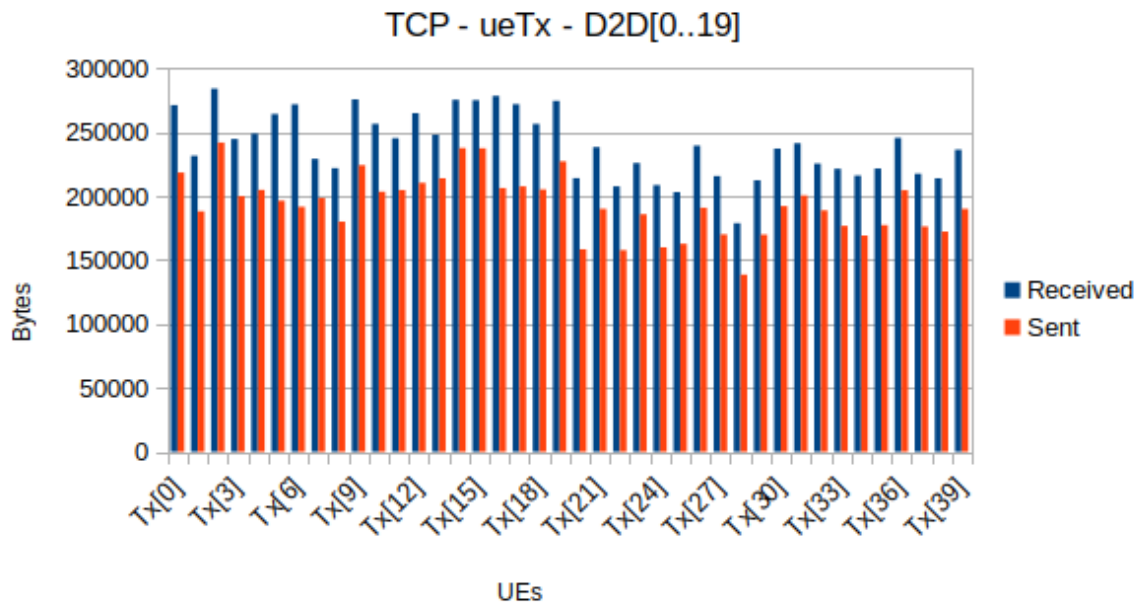


Figura B.11: Transmissão dos nós Tx no experimento TCP com 20 D2D.

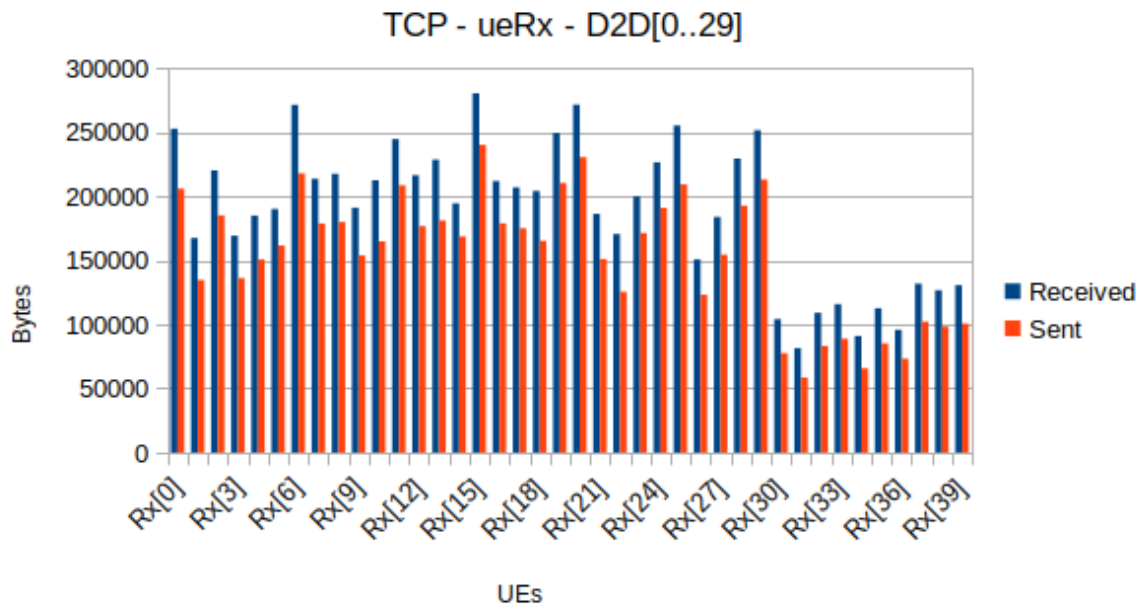


Figura B.12: Transmissão dos nós Rx no experimento TCP com 30 D2D.

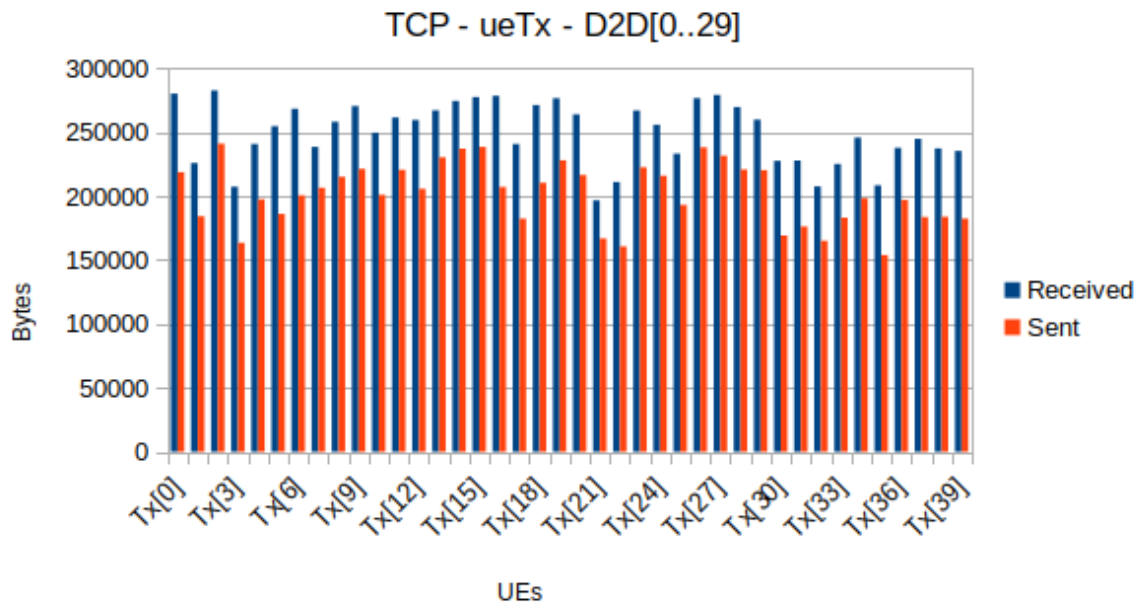


Figura B.13: Transmissão dos nós Tx no experimento TCP com 30 D2D.

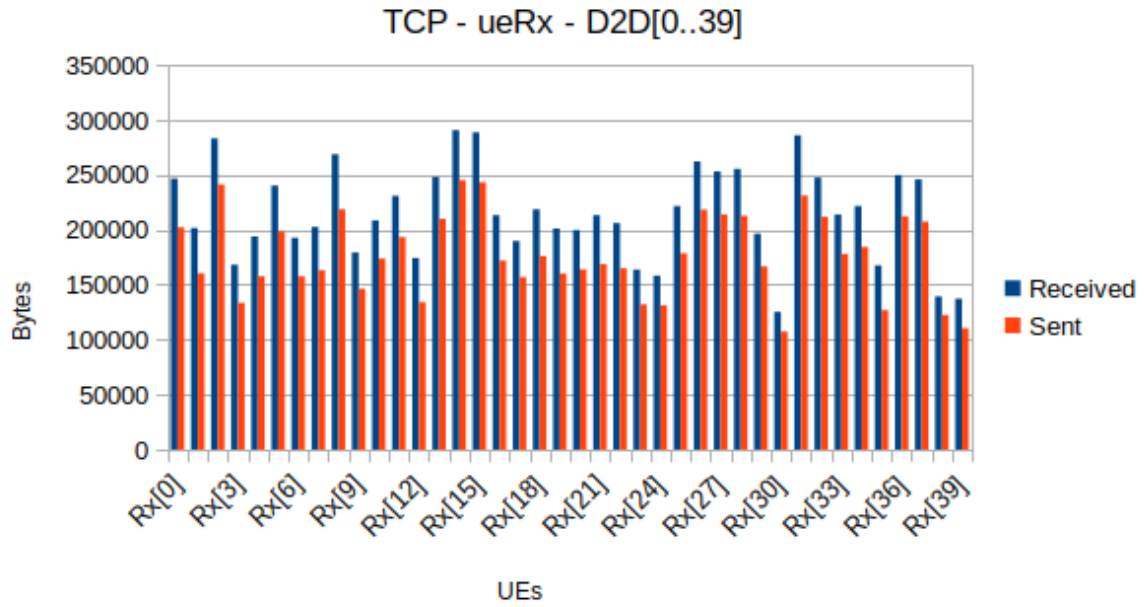


Figura B.14: Transmissão dos nós Rx no experimento TCP com 40 D2D.

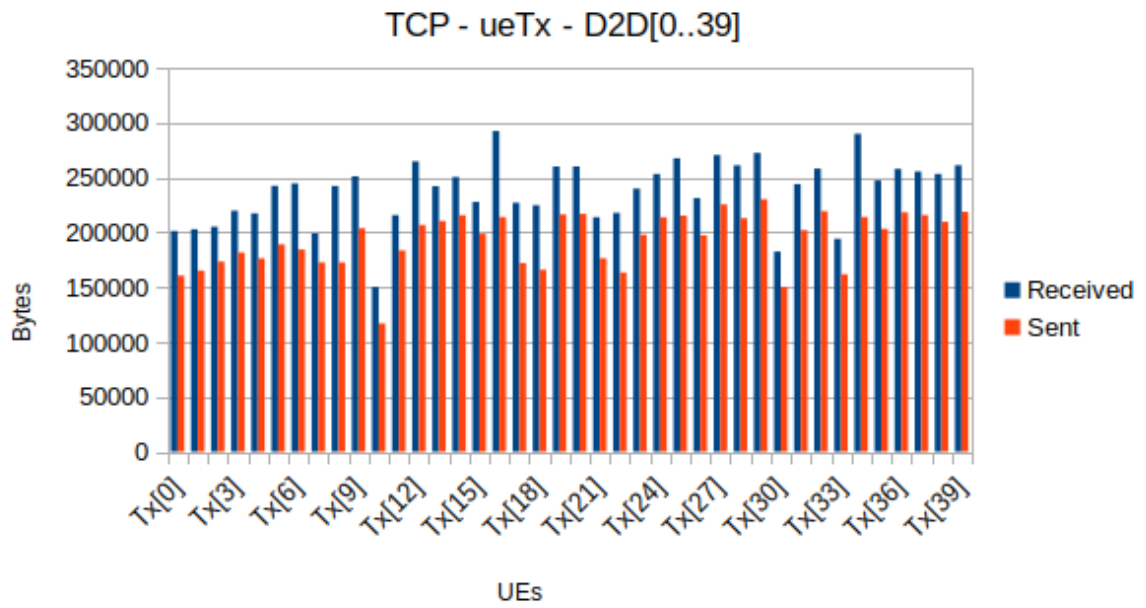


Figura B.15: Transmissão dos nós Tx no experimento TCP com 40 D2D.

B.3 experimento: UDP VoIP 1 m/s

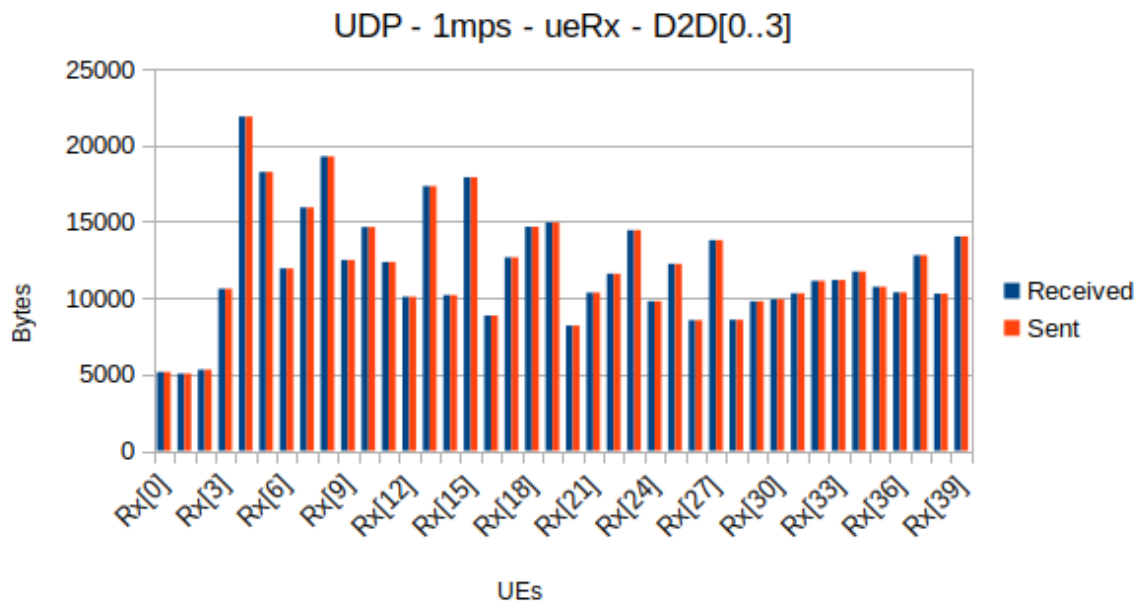


Figura B.16: Transmissão dos nós Rx no experimento UDP VoIP 1 m/s com 4 D2D.

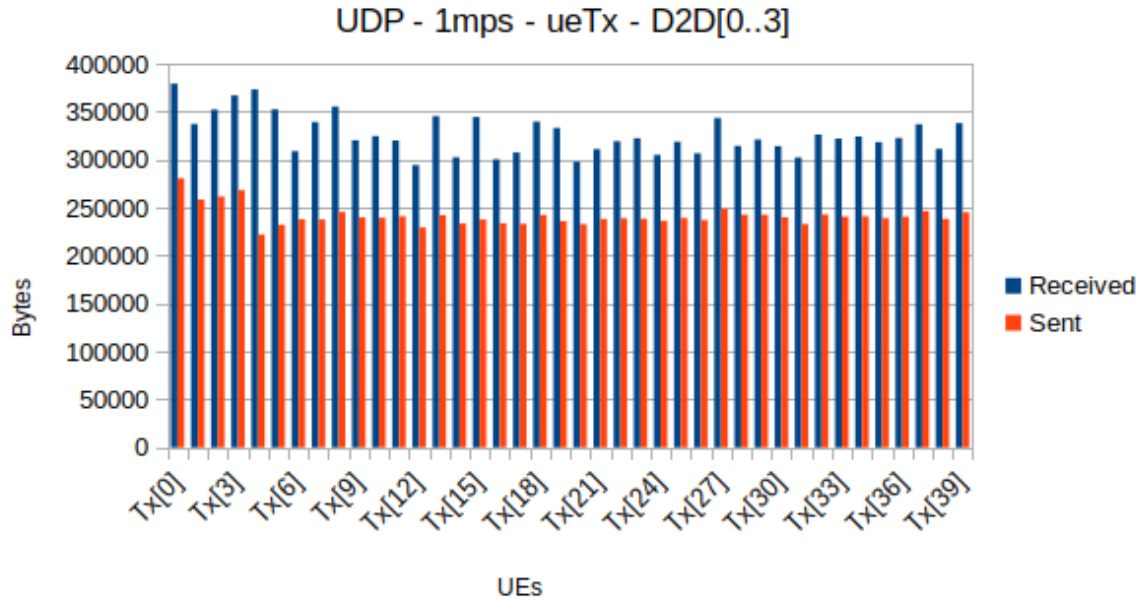


Figura B.17: Transmissão dos nós Tx no experimento UDP VoIP 1 m/s com 4 D2D.

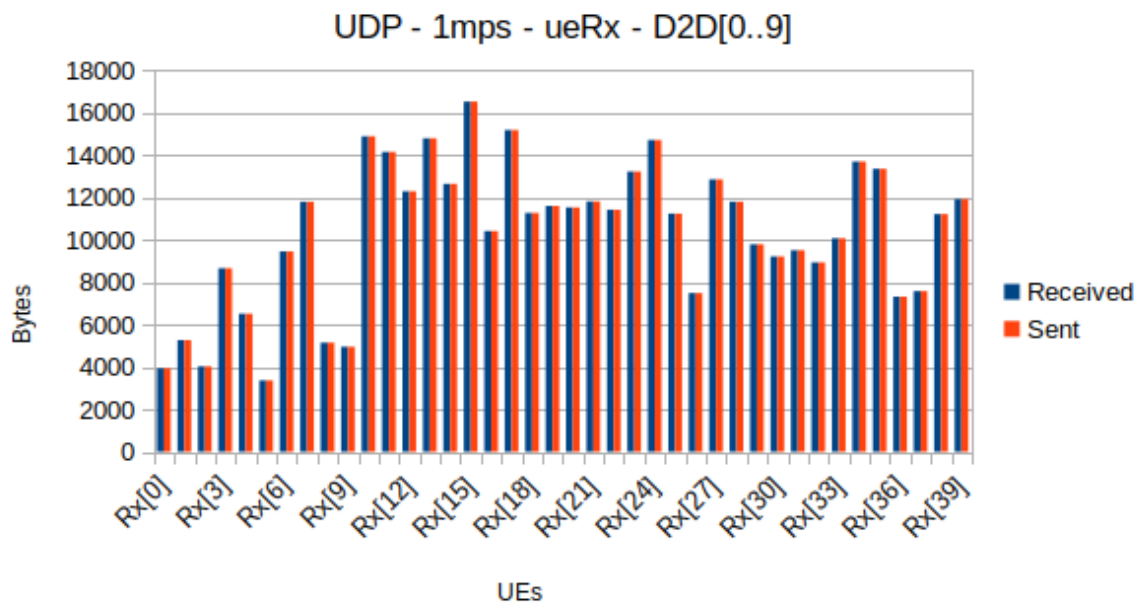


Figura B.18: Transmissão dos nós Rx no experimento UDP VoIP 1 m/s com 10 D2D.

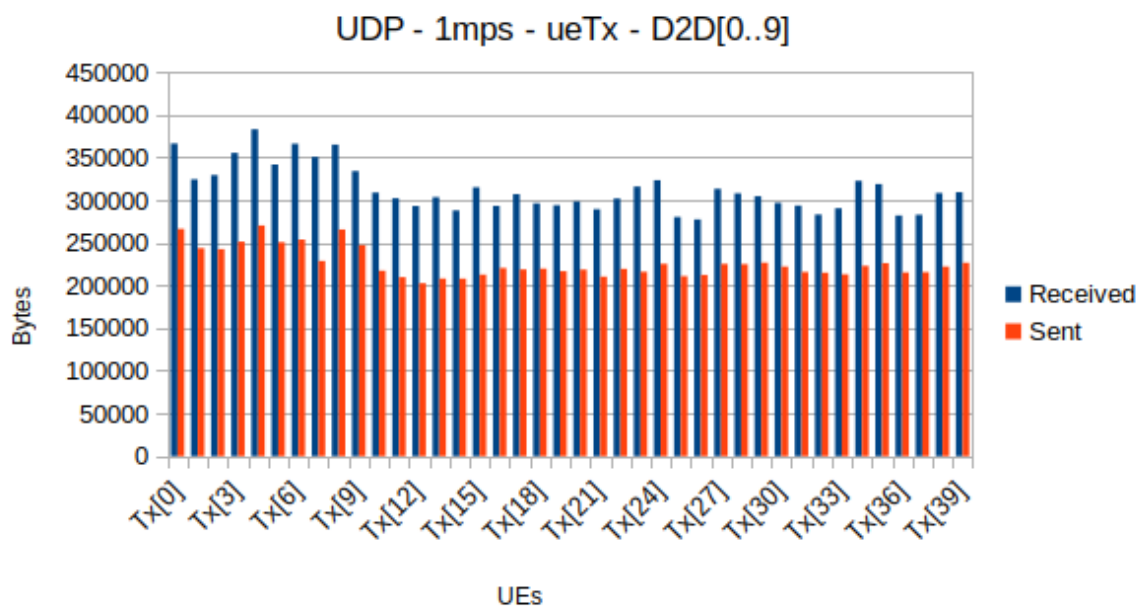


Figura B.19: Transmissão dos nós Tx no experimento UDP VoIP 1 m/s com 10 D2D.

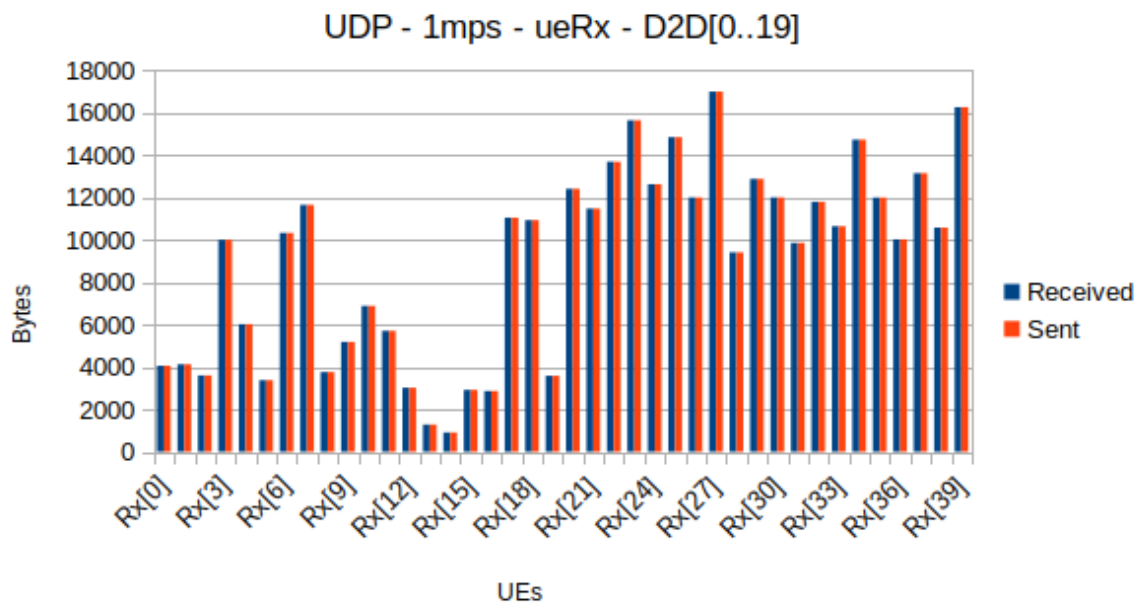


Figura B.20: Transmissão dos nós Rx no experimento UDP VoIP 1 m/s com 20 D2D.

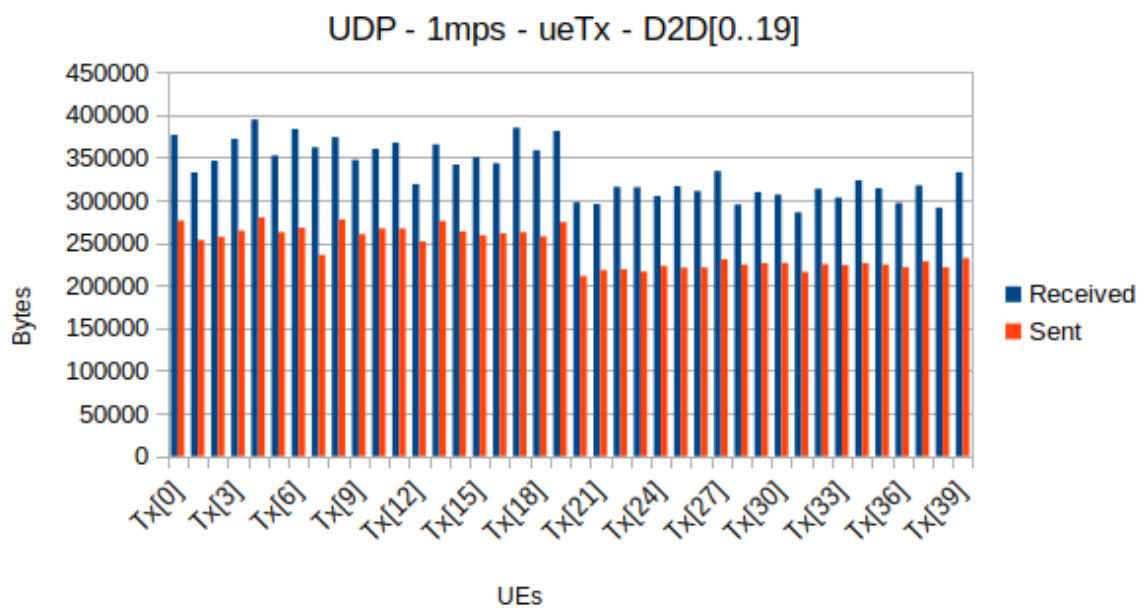


Figura B.21: Transmissão dos nós Tx no experimento UDP VoIP 1 m/s com 20 D2D.

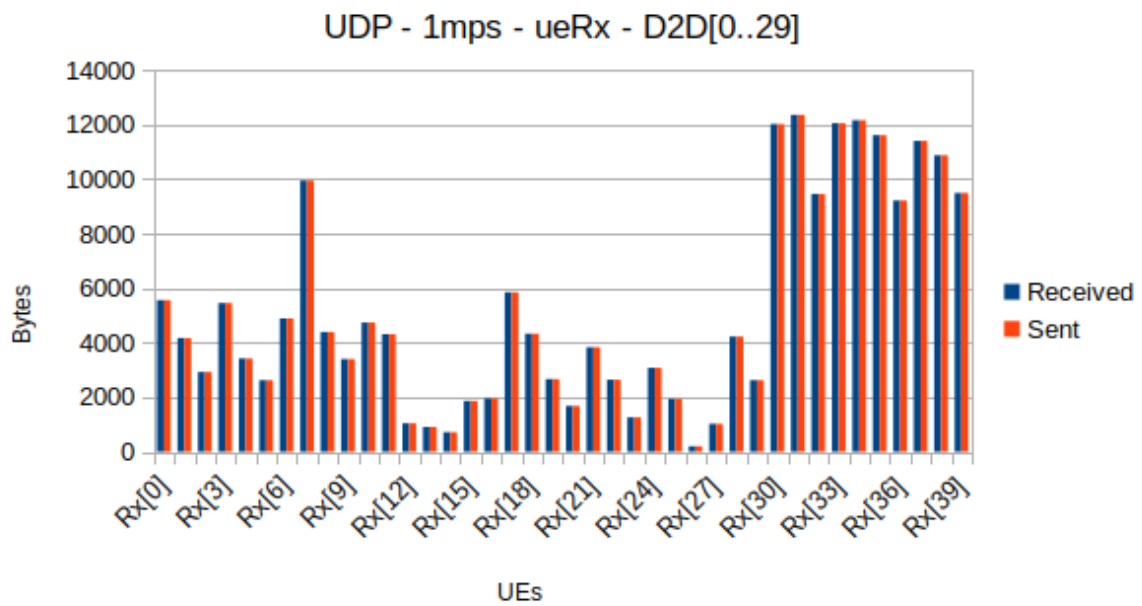


Figura B.22: Transmissão dos nós Rx no experimento UDP VoIP 1 m/s com 30 D2D.

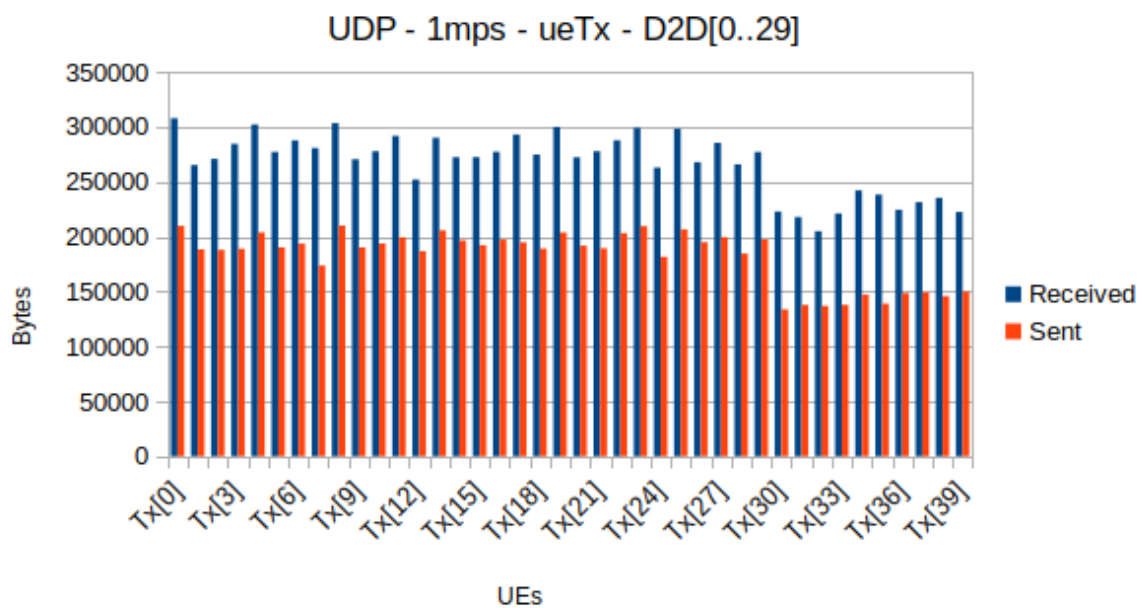


Figura B.23: Transmissão dos nós Tx no experimento UDP VoIP 1 m/s com 30 D2D.

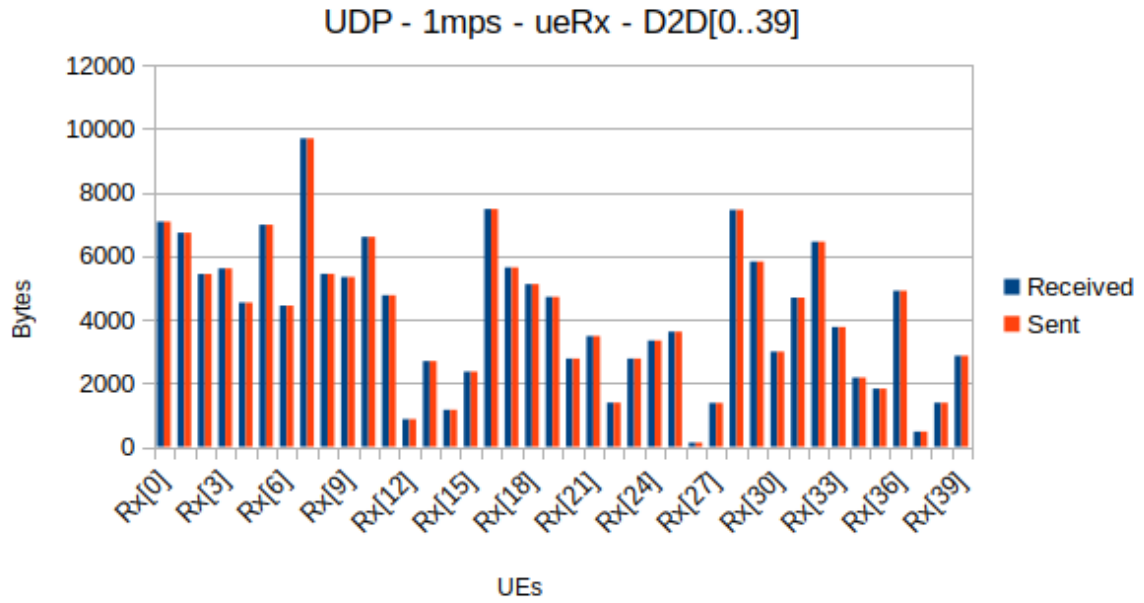


Figura B.24: Transmissão dos nós Rx no experimento UDP VoIP 1 m/s com 40 D2D.

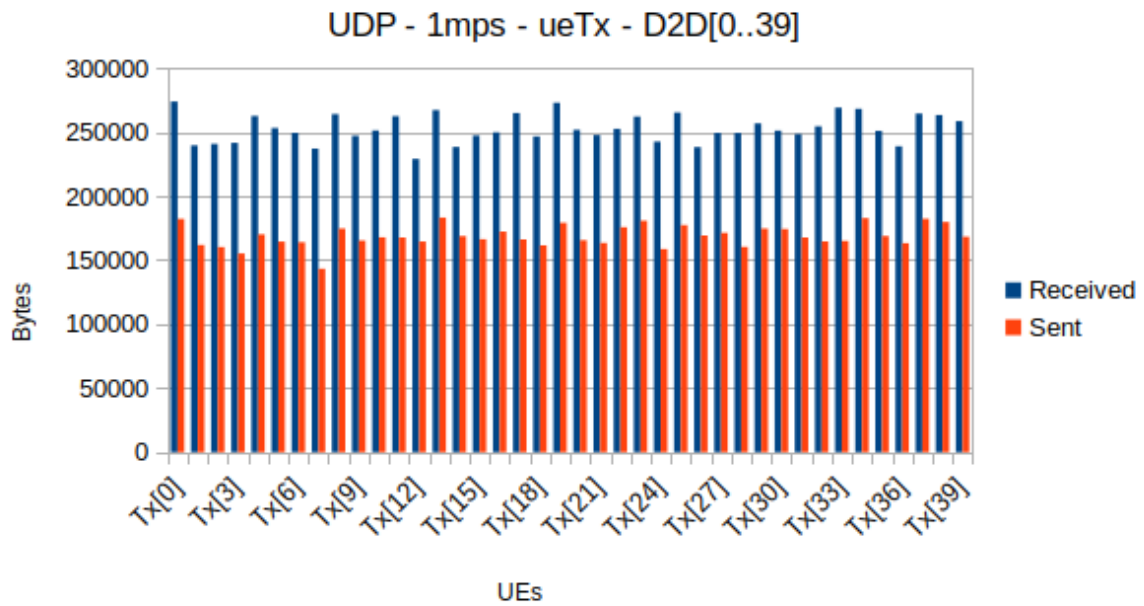


Figura B.25: Transmissão dos nós Tx no experimento UDP VoIP 1 m/s com 40 D2D.

B.4 experimento: UDP VoIP 2 m/s

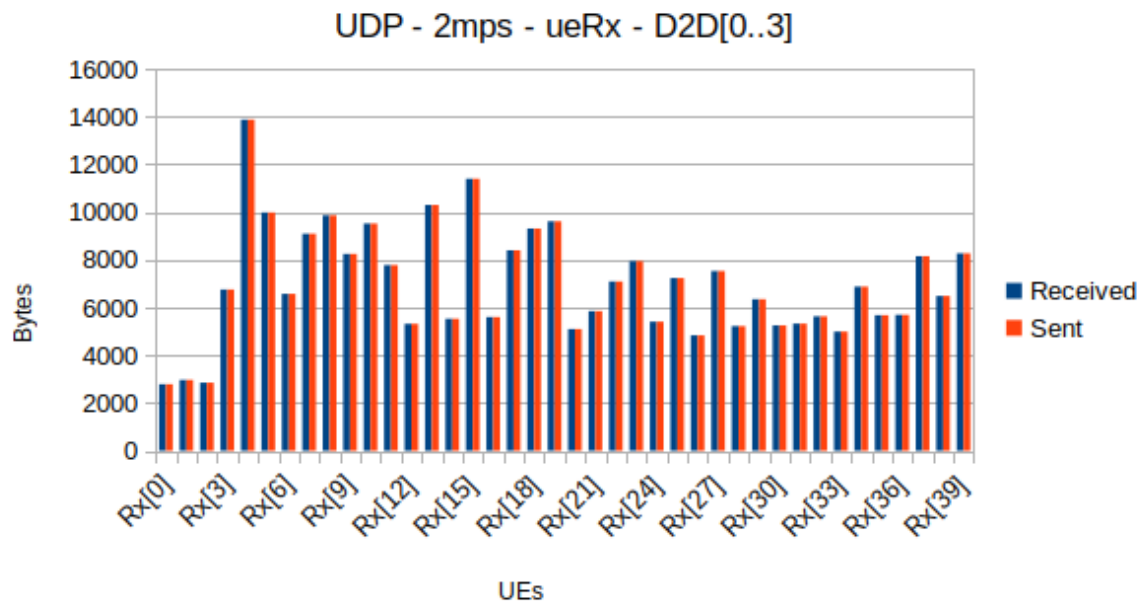


Figura B.26: Transmissão dos nós Rx no experimento UDP VoIP 2 m/s com 4 D2D.

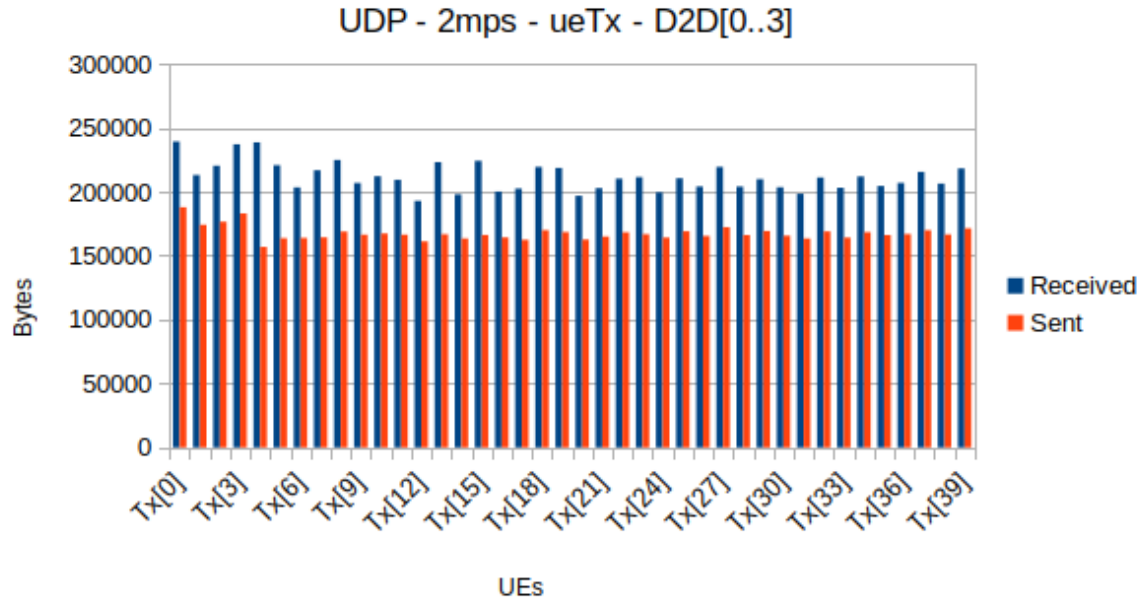


Figura B.27: Transmissão dos nós Tx no experimento UDP VoIP 2 m/s com 4 D2D.

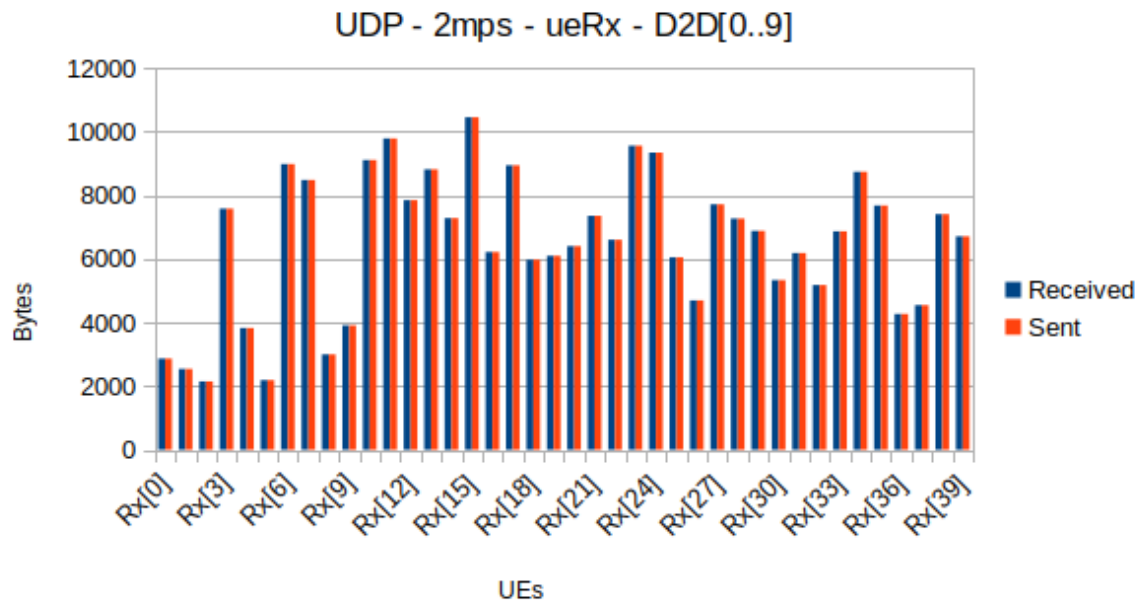


Figura B.28: Transmissão dos nós Rx no experimento UDP VoIP 2 m/s com 10 D2D.

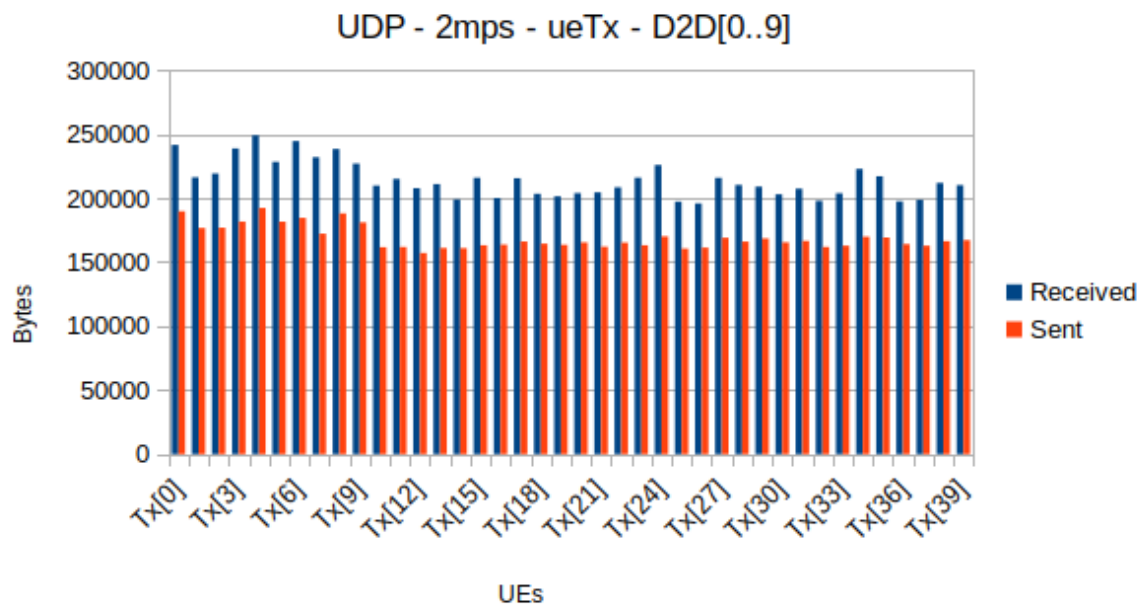


Figura B.29: Transmissão dos nós Tx no experimento UDP VoIP 2 m/s com 10 D2D.

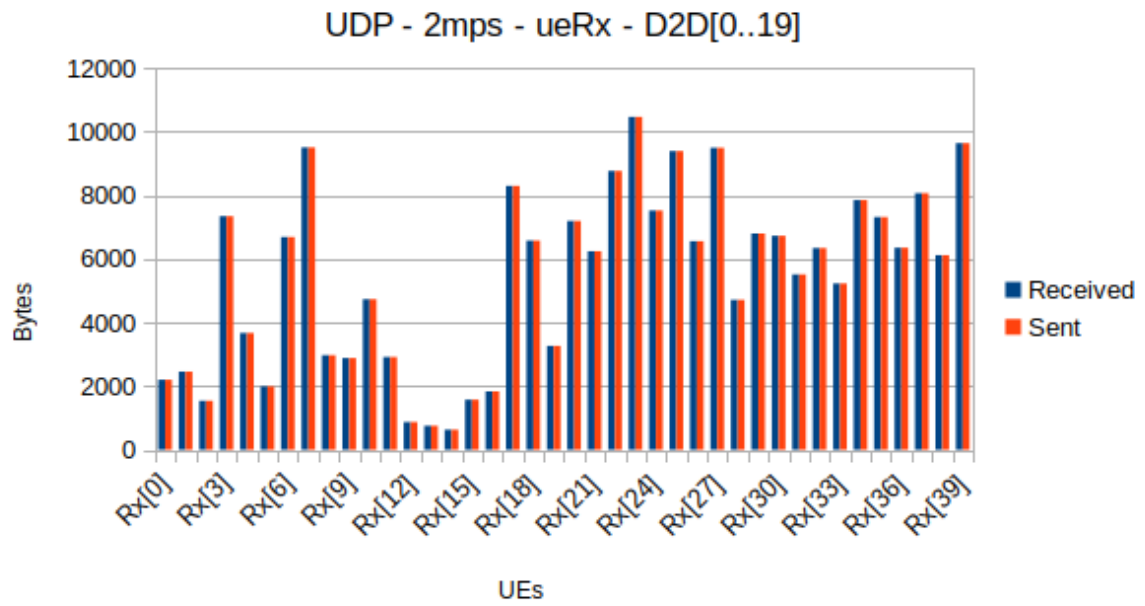


Figura B.30: Transmissão dos nós Rx no experimento UDP VoIP 2 m/s com 20 D2D.

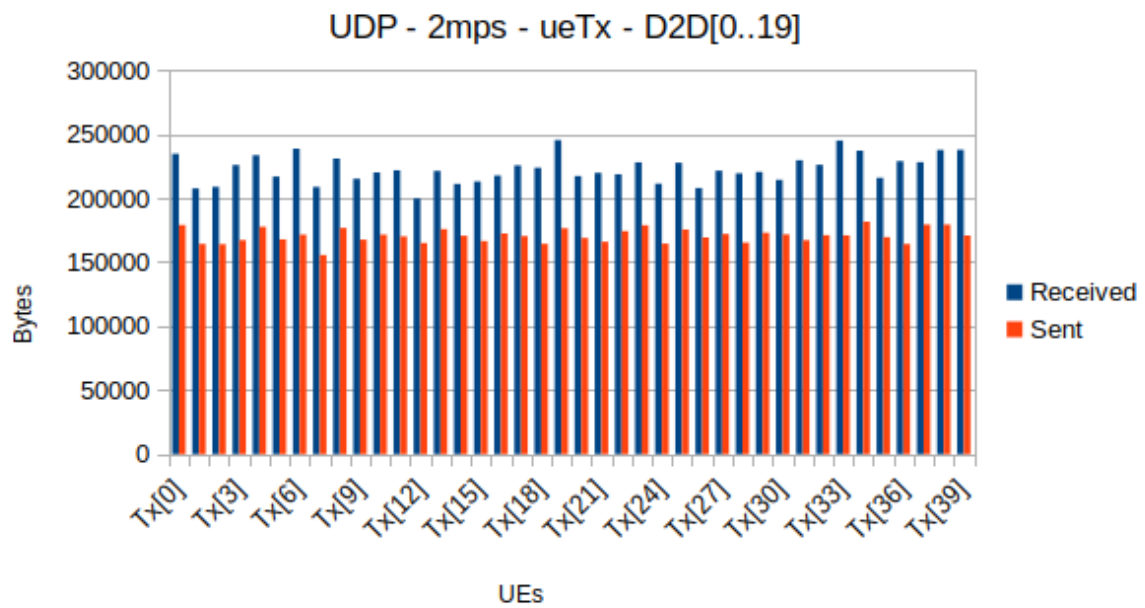


Figura B.31: Transmissão dos nós Tx no experimento UDP VoIP 2 m/s com 20 D2D.

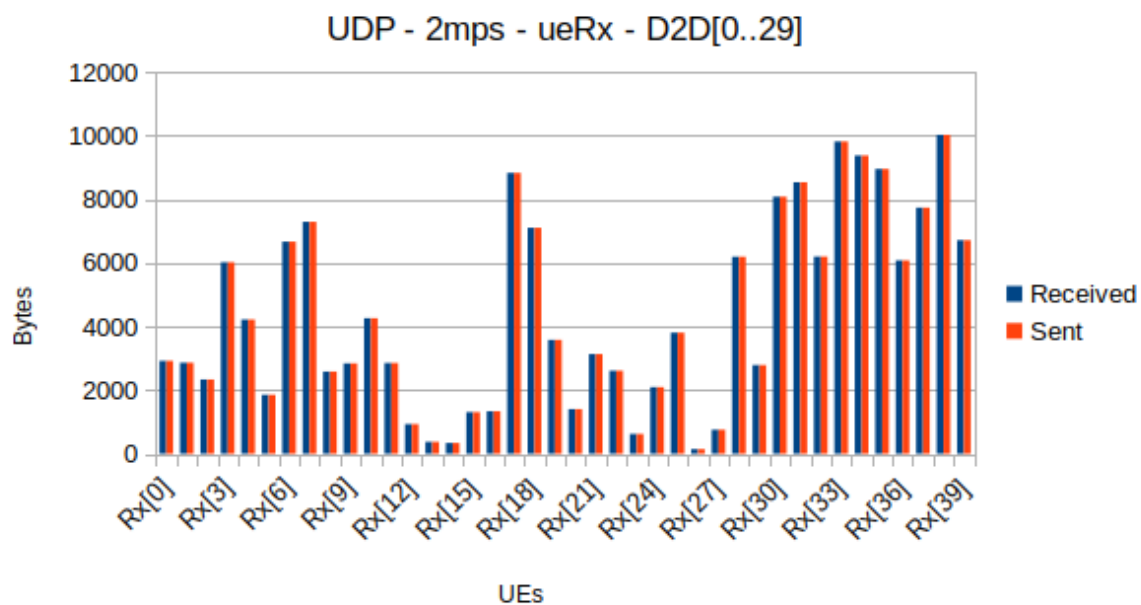


Figura B.32: Transmissão dos nós Rx no experimento UDP VoIP 2 m/s com 30 D2D.

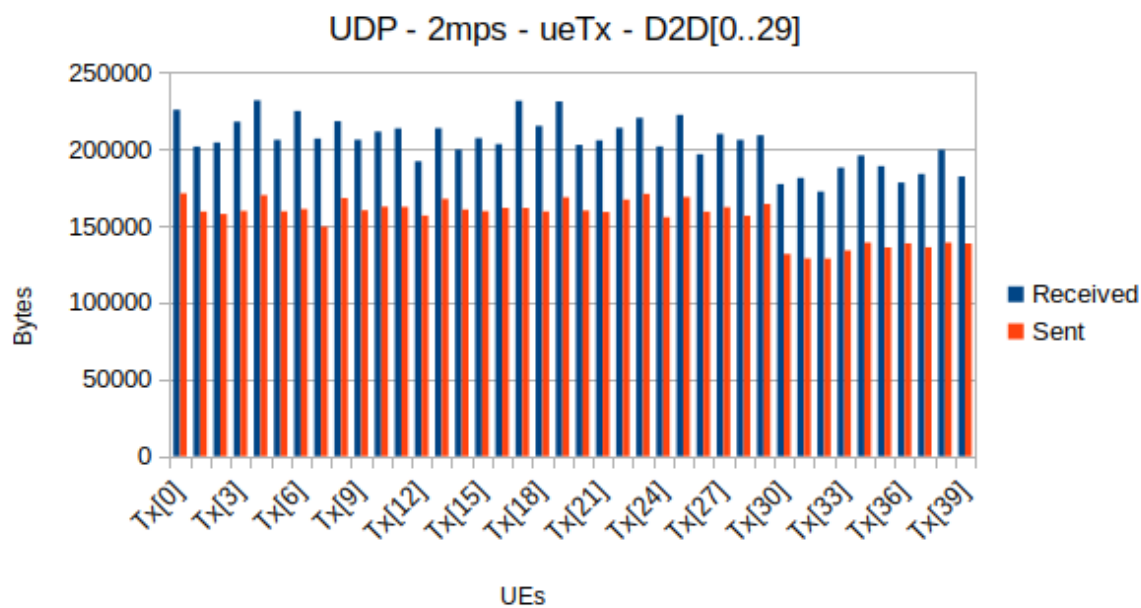


Figura B.33: Transmissão dos nós Tx no experimento UDP VoIP 2 m/s com 30 D2D.

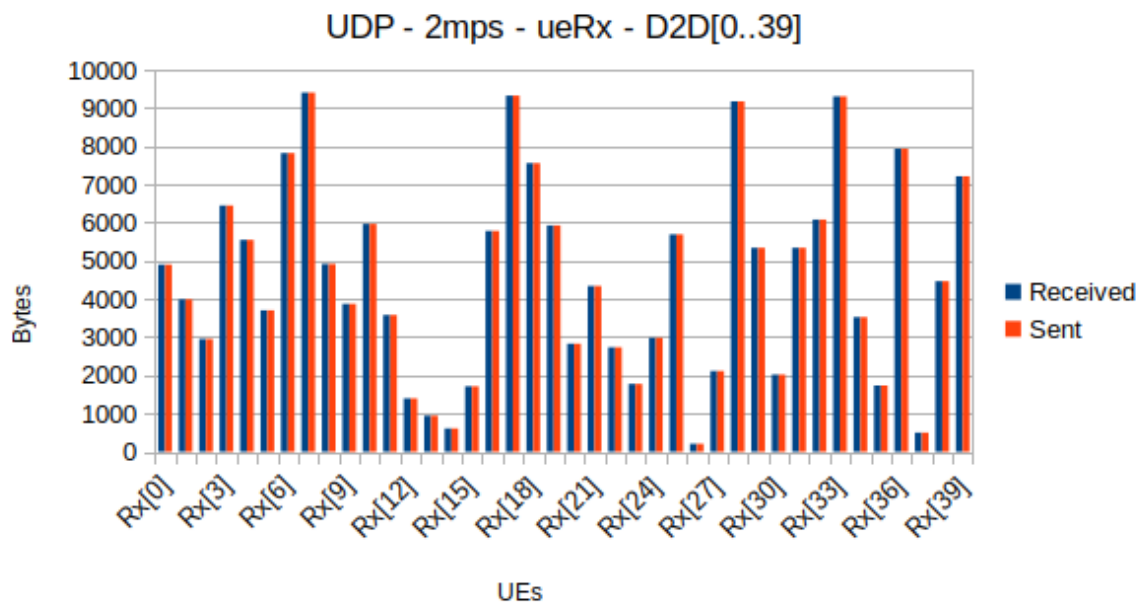


Figura B.34: Transmissão dos nós Rx no experimento UDP VoIP 2 m/s com 40 D2D.

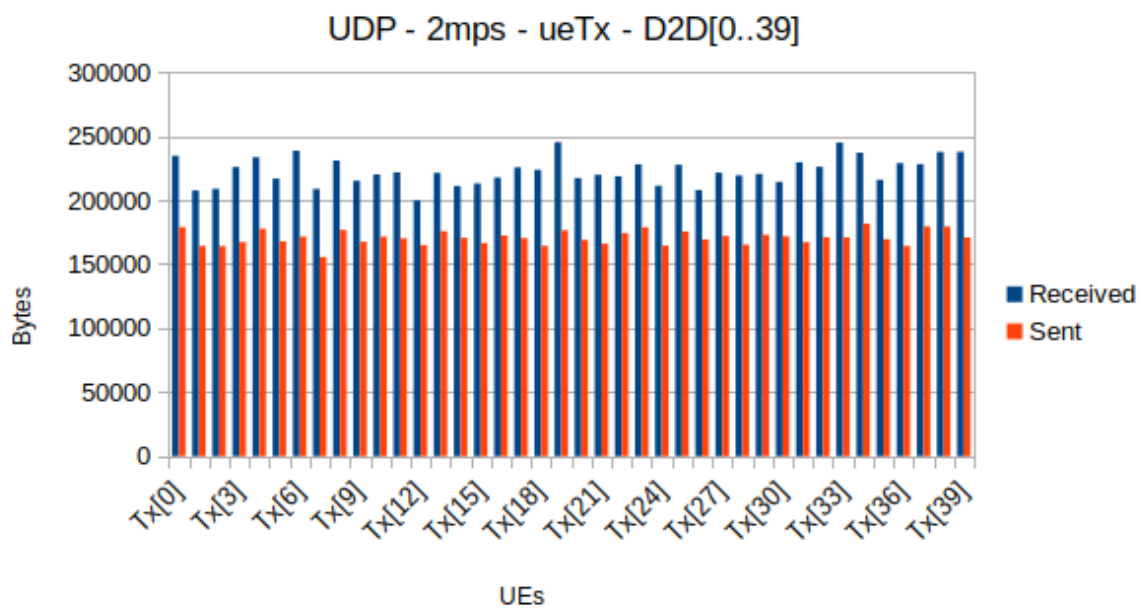


Figura B.35: Transmissão dos nós Tx no experimento UDP VoIP 2 m/s com 40 D2D.

B.5 experimento: UDP VoIP 5 m/s

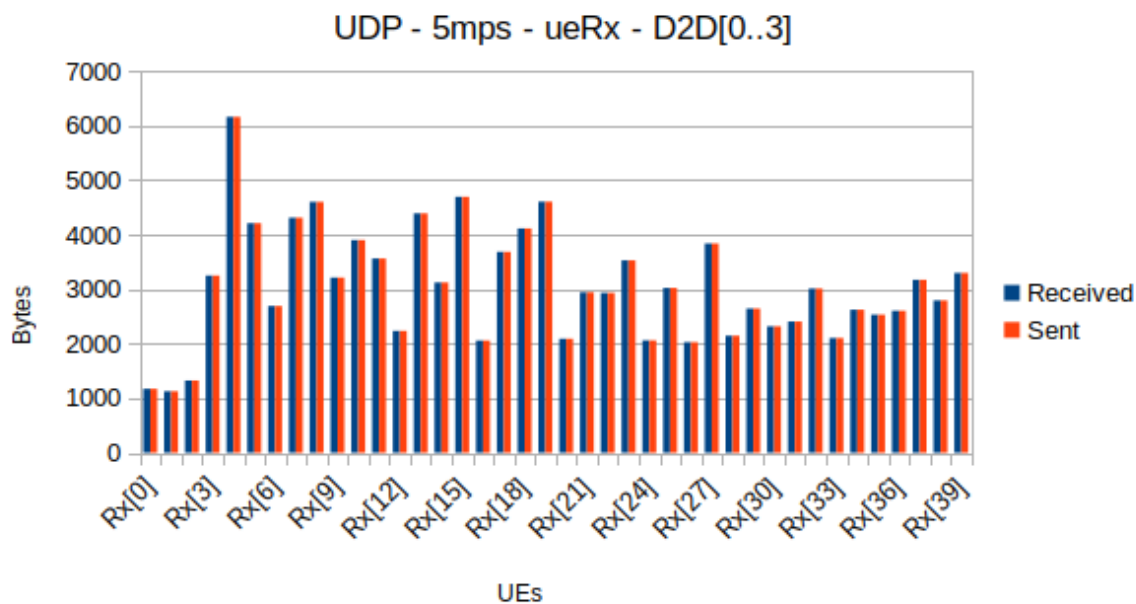


Figura B.36: Transmissão dos nós Rx no experimento UDP VoIP 5 m/s com 4 D2D.

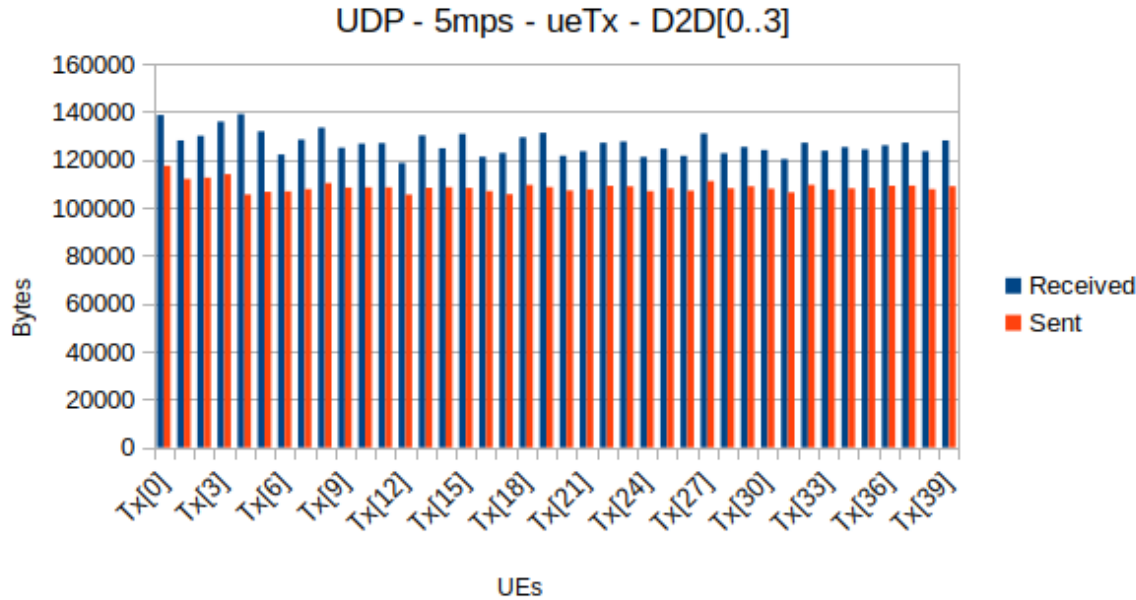


Figura B.37: Transmissão dos nós Tx no experimento UDP VoIP 5 m/s com 4 D2D.

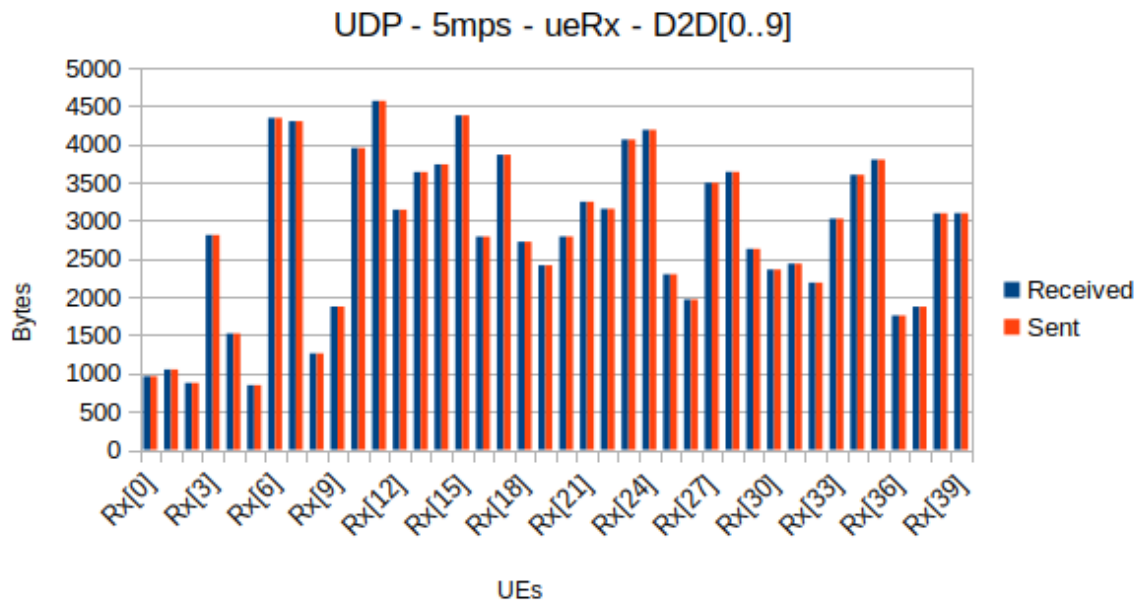


Figura B.38: Transmissão dos nós Rx no experimento UDP VoIP 5 m/s com 10 D2D.

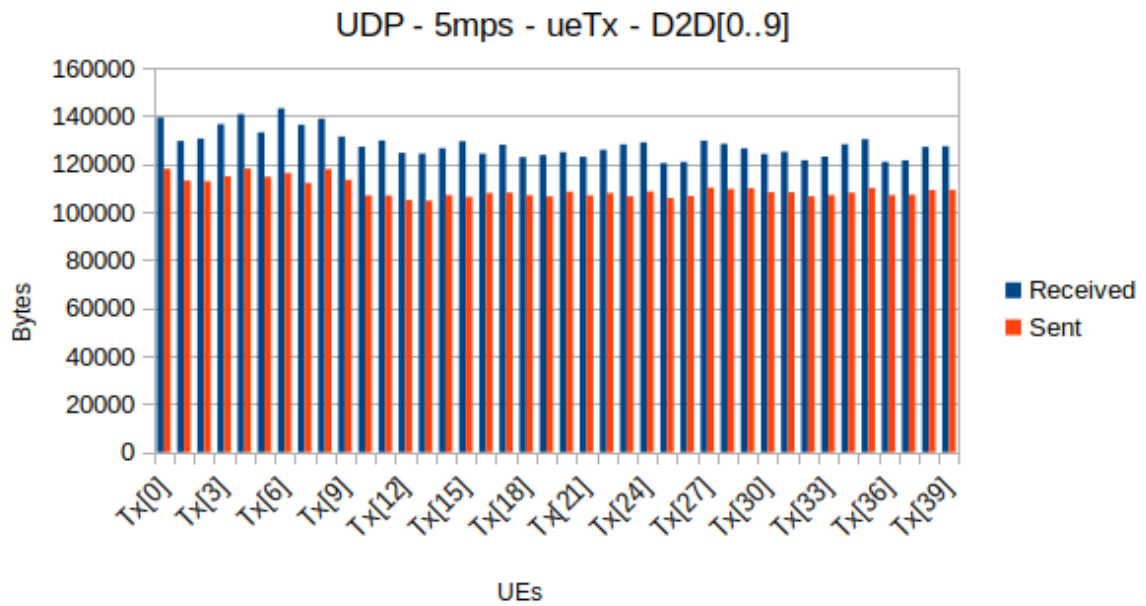


Figura B.39: Transmissão dos nós Tx no experimento UDP VoIP 5 m/s com 10 D2D.

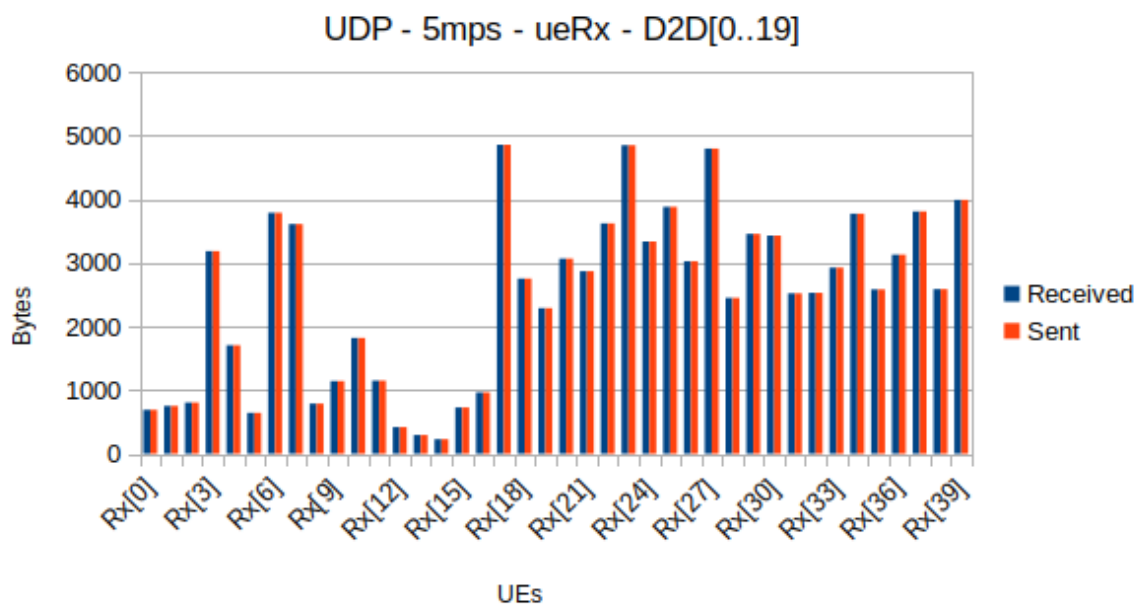


Figura B.40: Transmissão dos nós Rx no experimento UDP VoIP 5 m/s com 20 D2D.

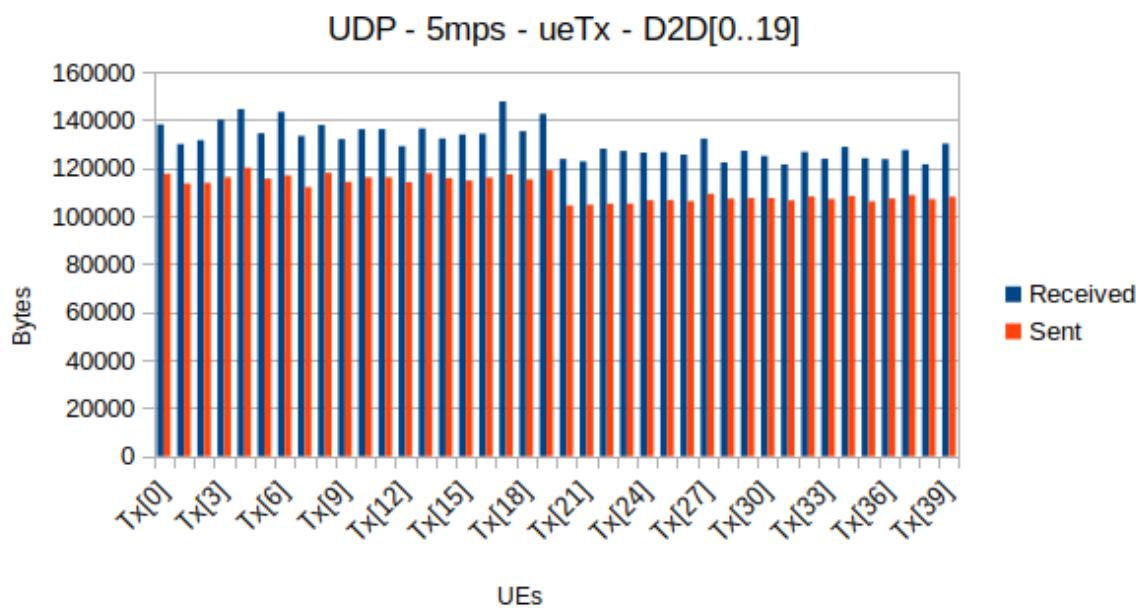


Figura B.41: Transmissão dos nós Tx no experimento UDP VoIP 5 m/s com 20 D2D.

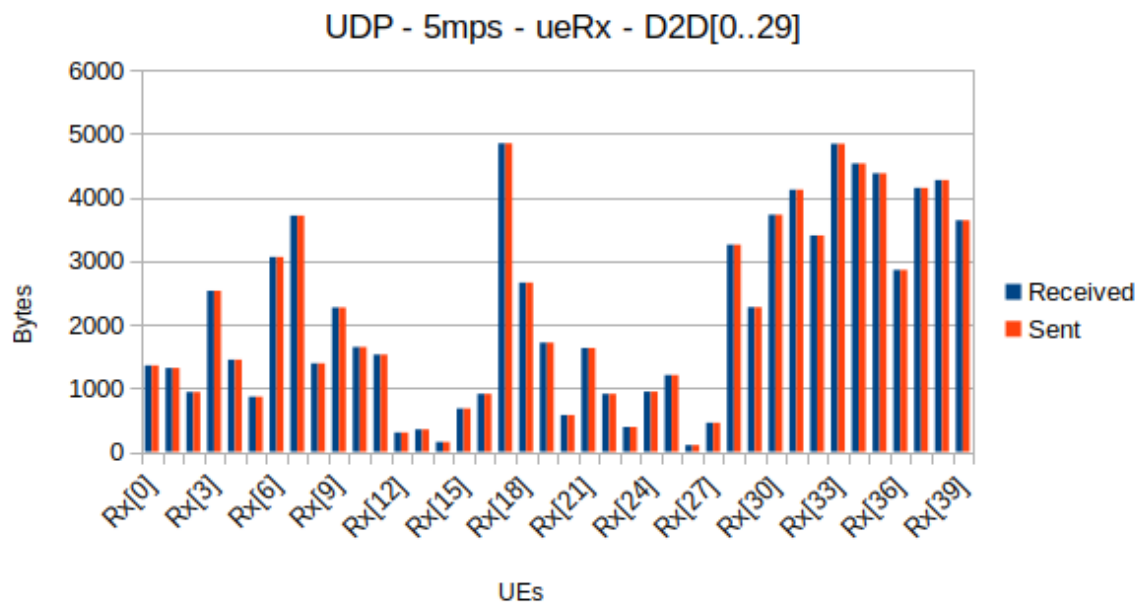


Figura B.42: Transmissão dos nós Rx no experimento UDP VoIP 5 m/s com 30 D2D.

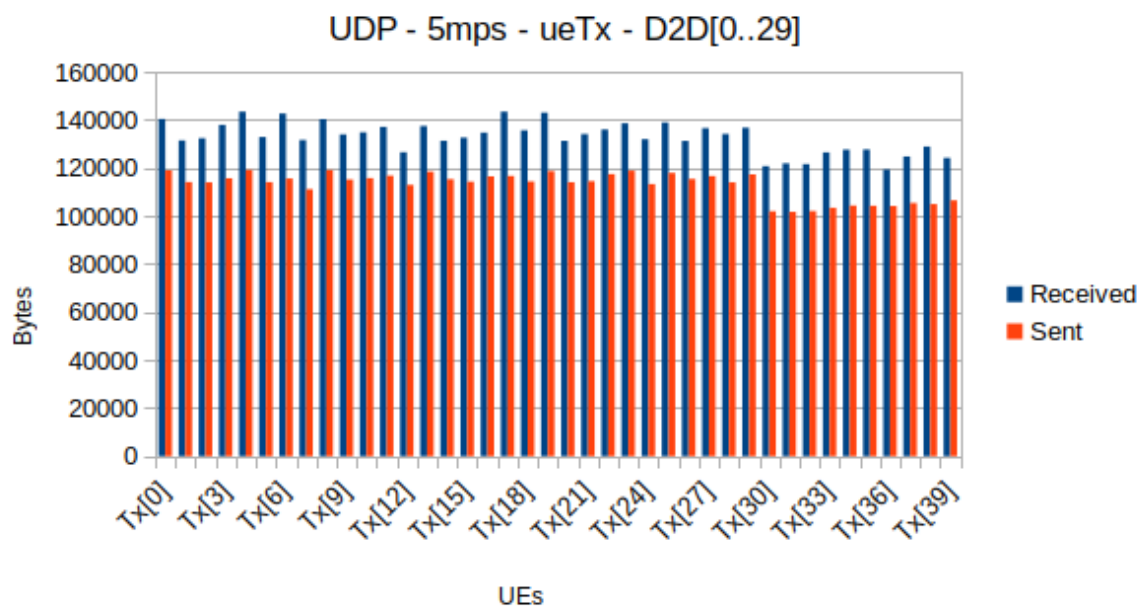


Figura B.43: Transmissão dos nós Tx no experimento UDP VoIP 5 m/s com 30 D2D.

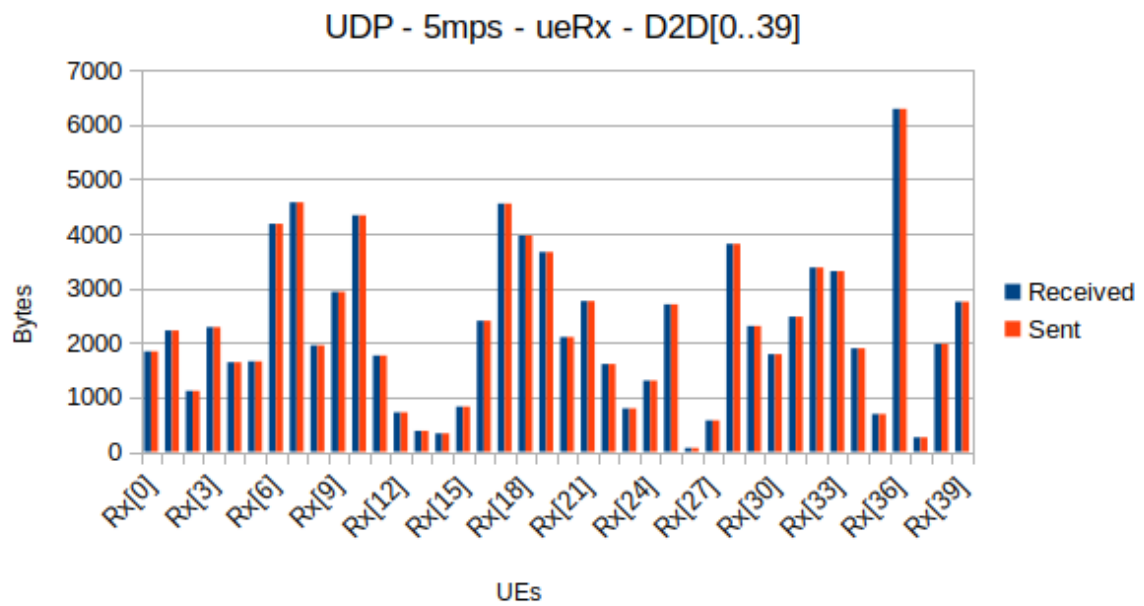


Figura B.44: Transmissão dos nós Rx no experimento UDP VoIP 5 m/s com 40 D2D.

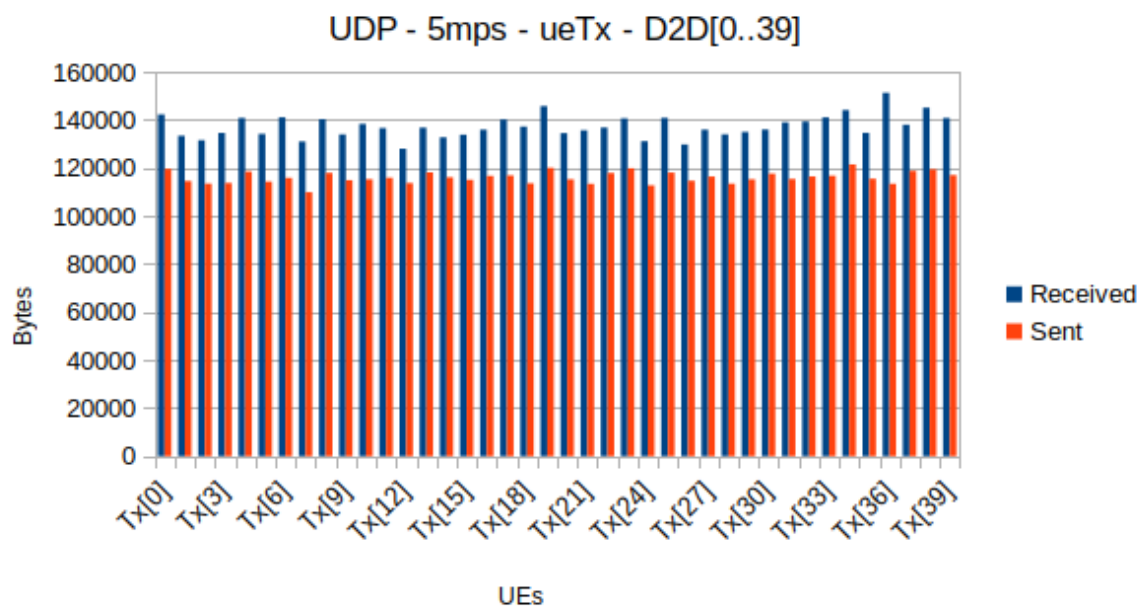


Figura B.45: Transmissão dos nós Tx no experimento UDP VoIP 5 m/s com 40 D2D.

B.6 experimento: UDP streaming 256B

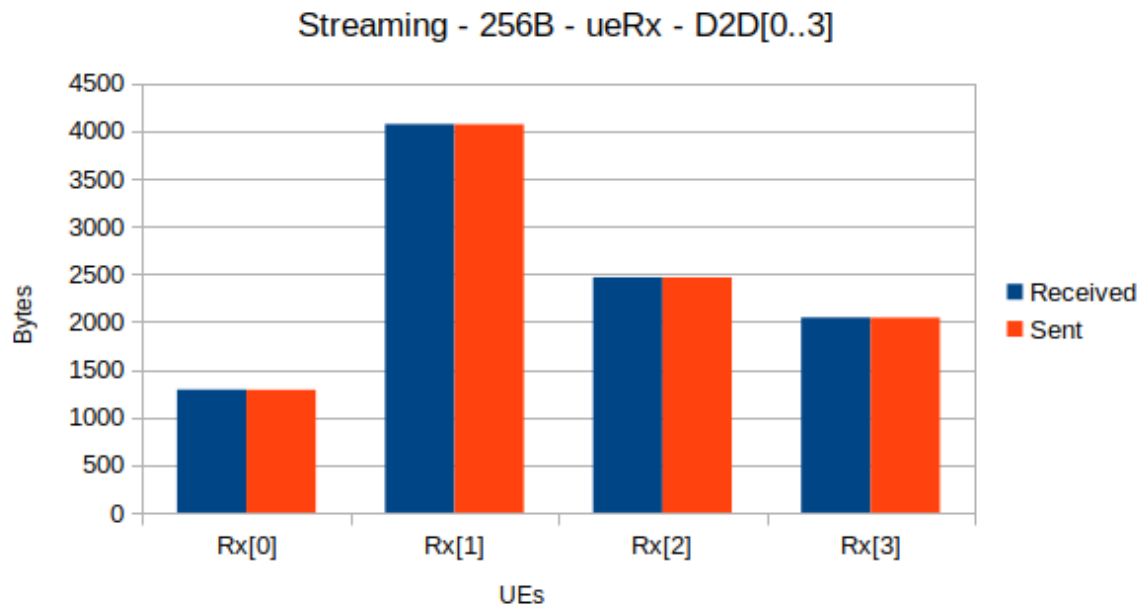


Figura B.46: Transmissão dos nós Rx no experimento UDP streaming 256B com 4 D2D.

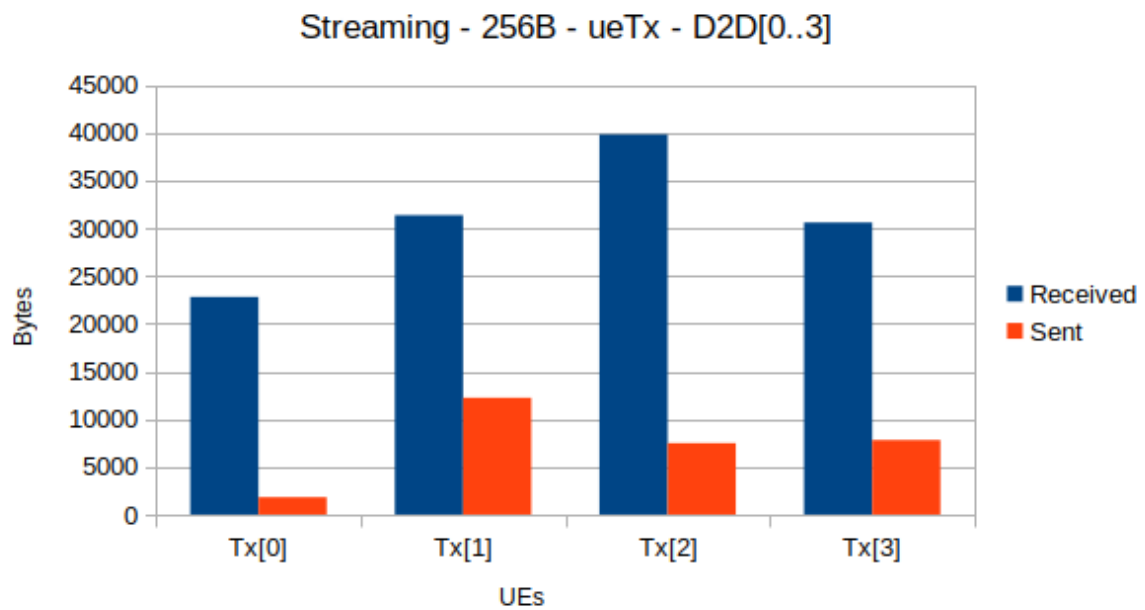


Figura B.47: Transmissão dos nós Tx no experimento UDP streaming 256B com 4 D2D.

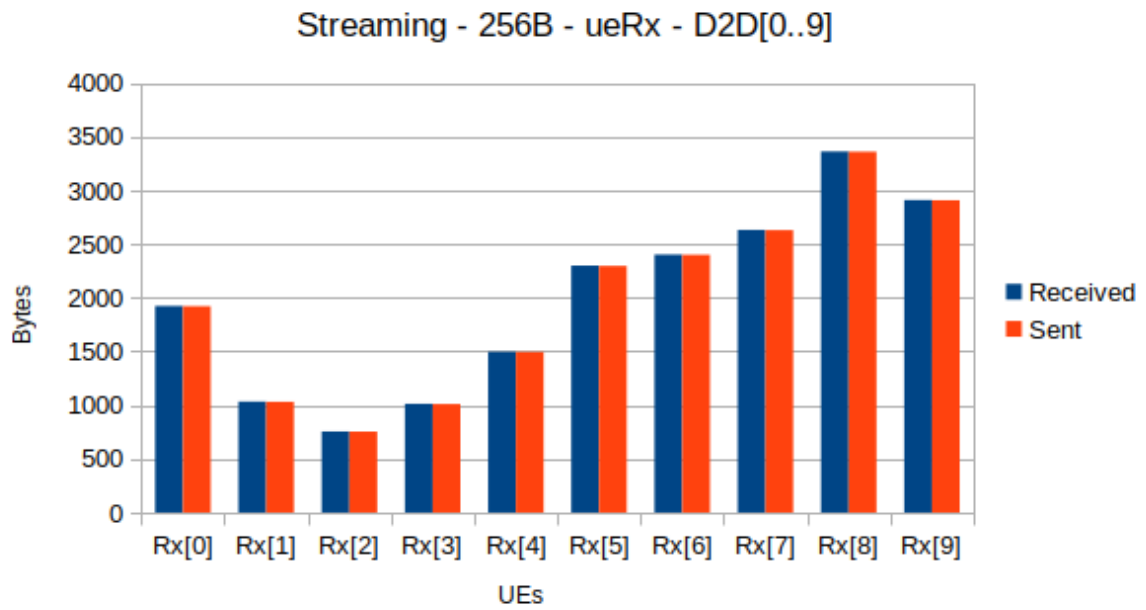


Figura B.48: Transmissão dos nós Rx no experimento UDP streaming 256B com 10 D2D.

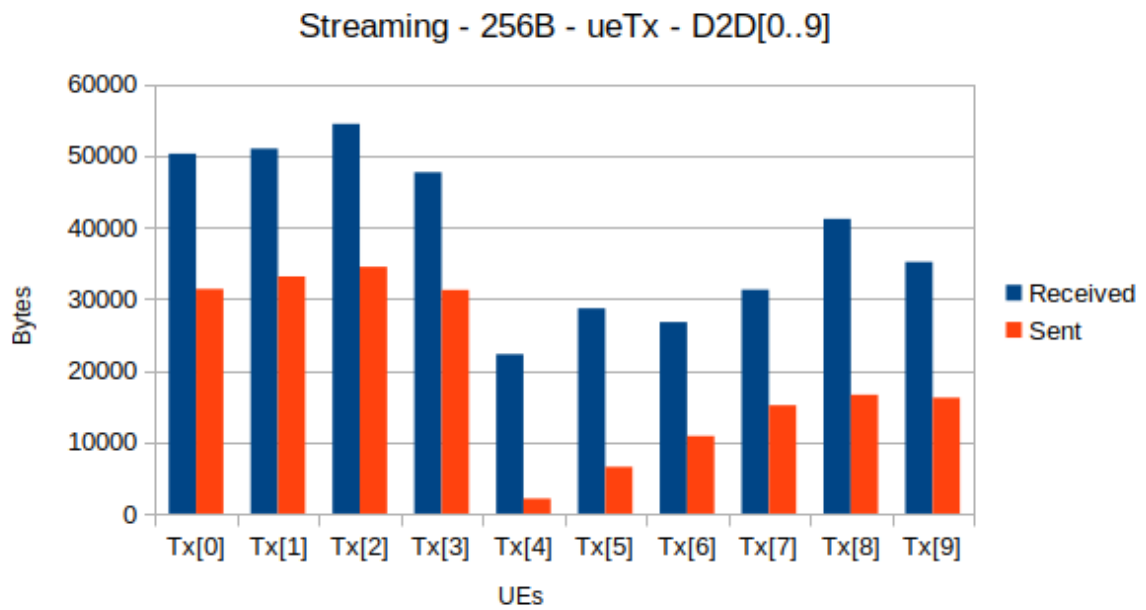


Figura B.49: Transmissão dos nós Tx no experimento UDP streaming 256B com 10 D2D.

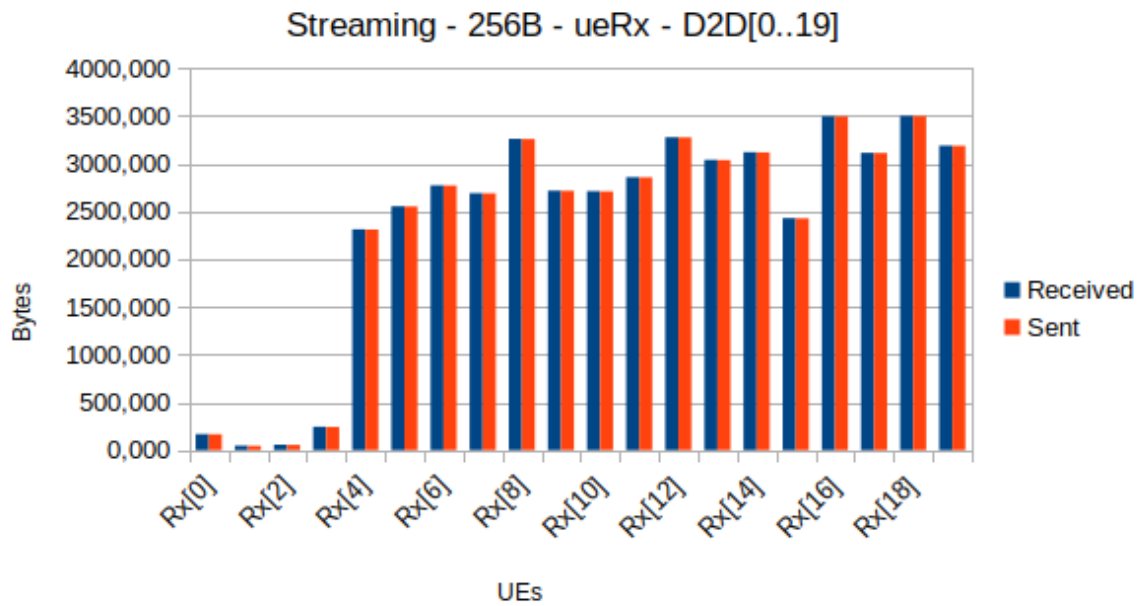


Figura B.50: Transmissão dos nós Rx no experimento UDP streaming 256B com 20 D2D.

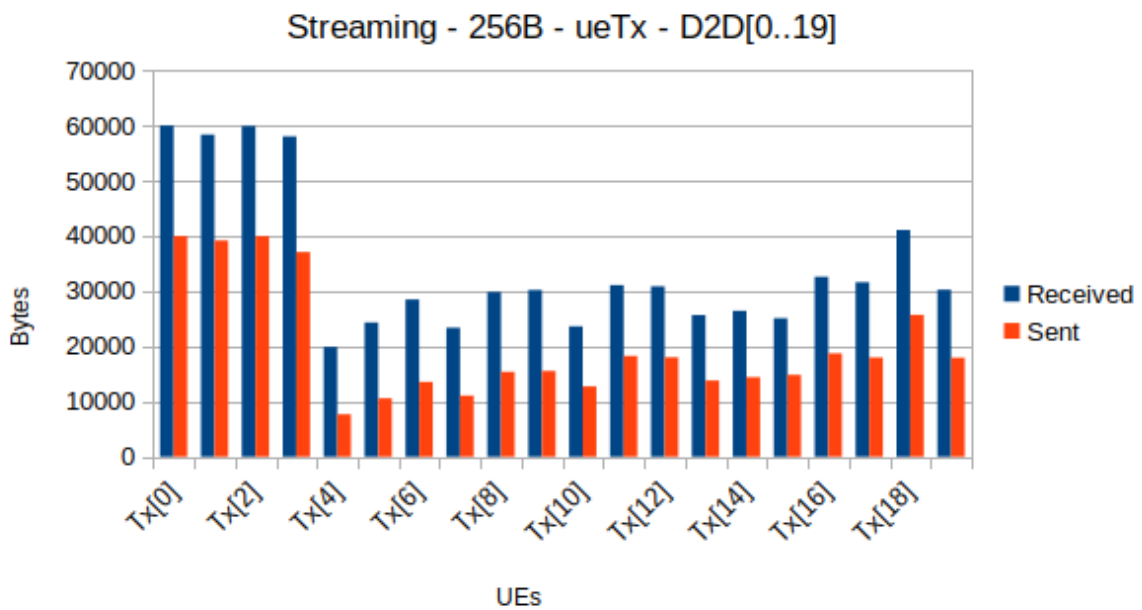


Figura B.51: Transmissão dos nós Tx no experimento UDP streaming 256B com 20 D2D.

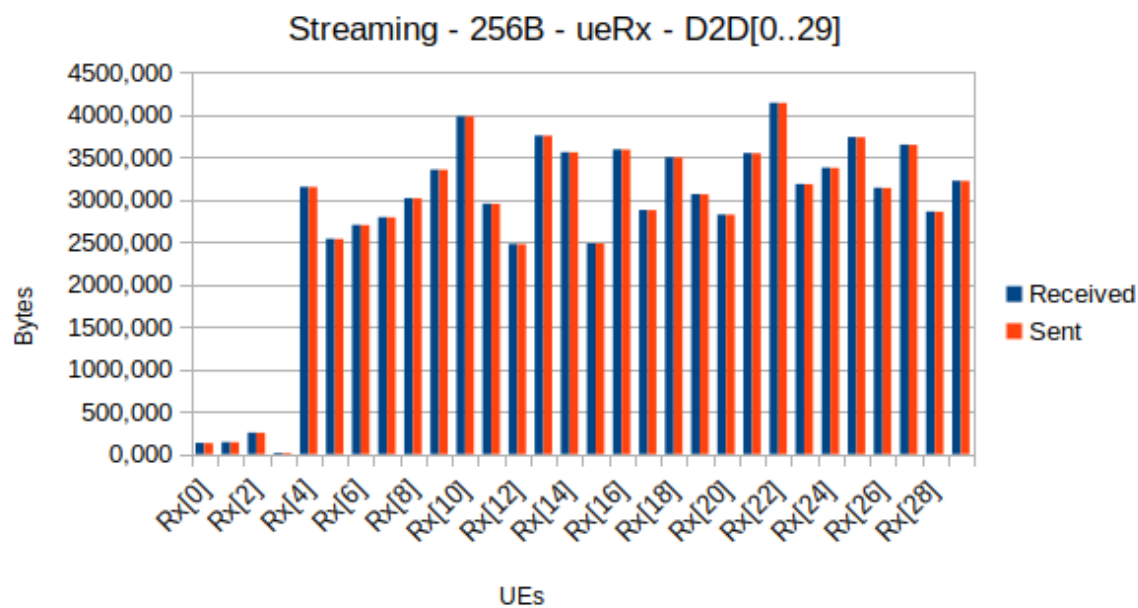


Figura B.52: Transmissão dos nós Rx no experimento UDP streaming 256B com 30 D2D.

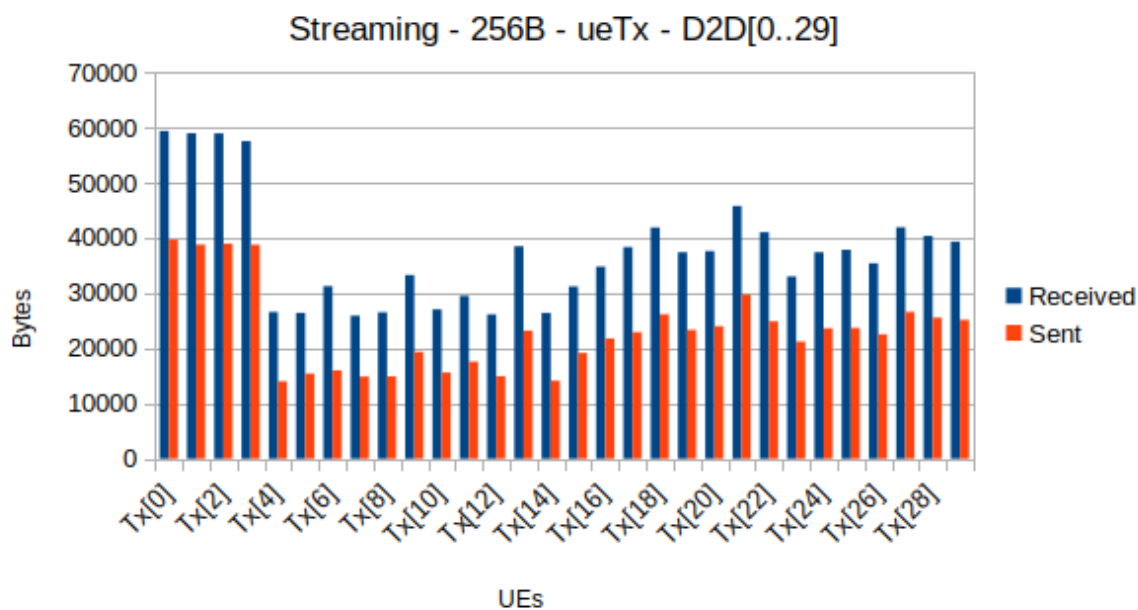


Figura B.53: Transmissão dos nós Tx no experimento UDP streaming 256B com 30 D2D.

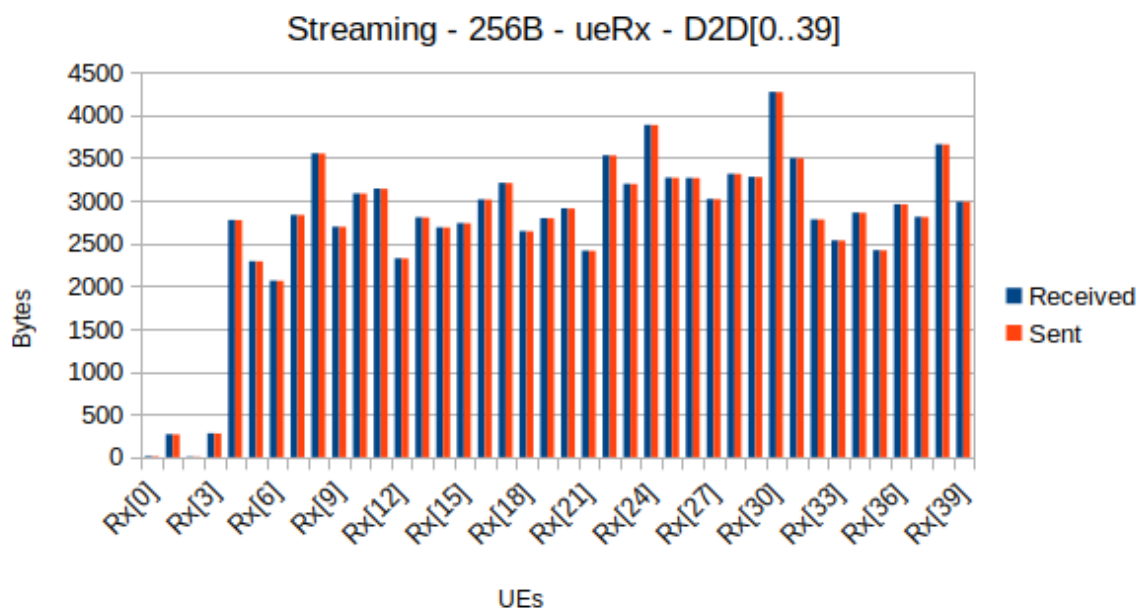


Figura B.54: Transmissão dos nós Rx no experimento UDP streaming 256B com 40 D2D.

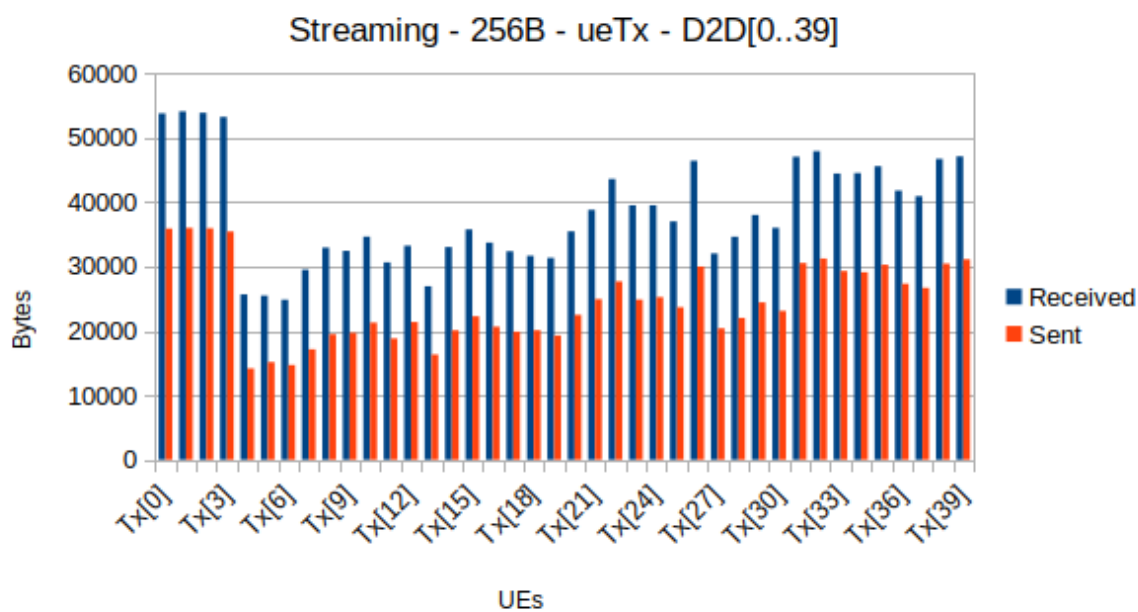


Figura B.55: Transmissão dos nós Tx no experimento UDP streaming 256B com 40 D2D.

B.7 experimento: UDP streaming 512B

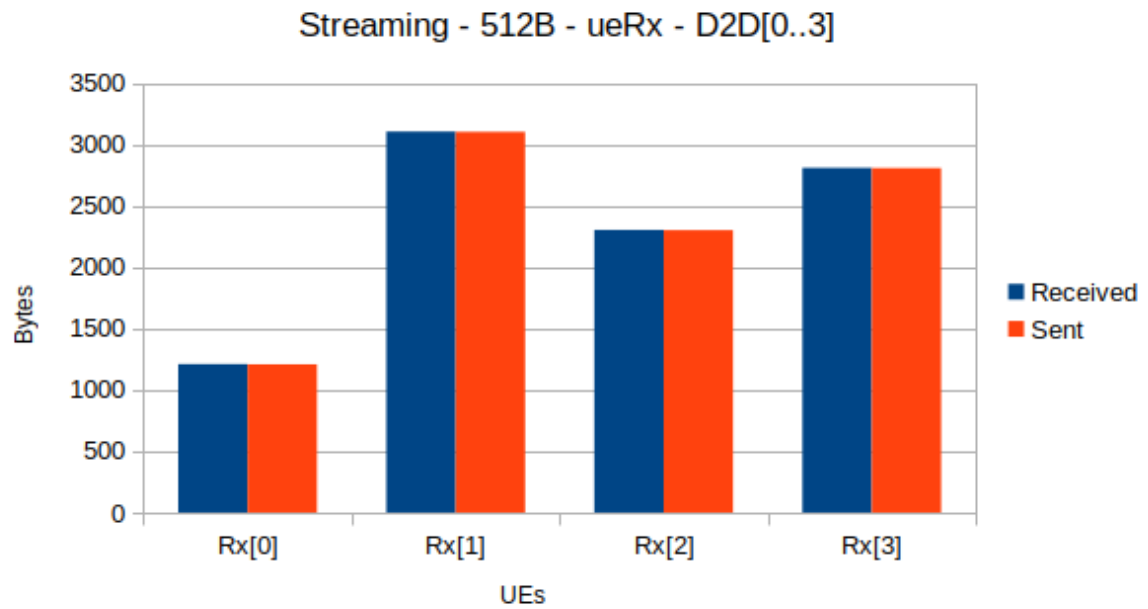


Figura B.56: Transmissão dos nós Rx no experimento UDP streaming 512B com 4 D2D.

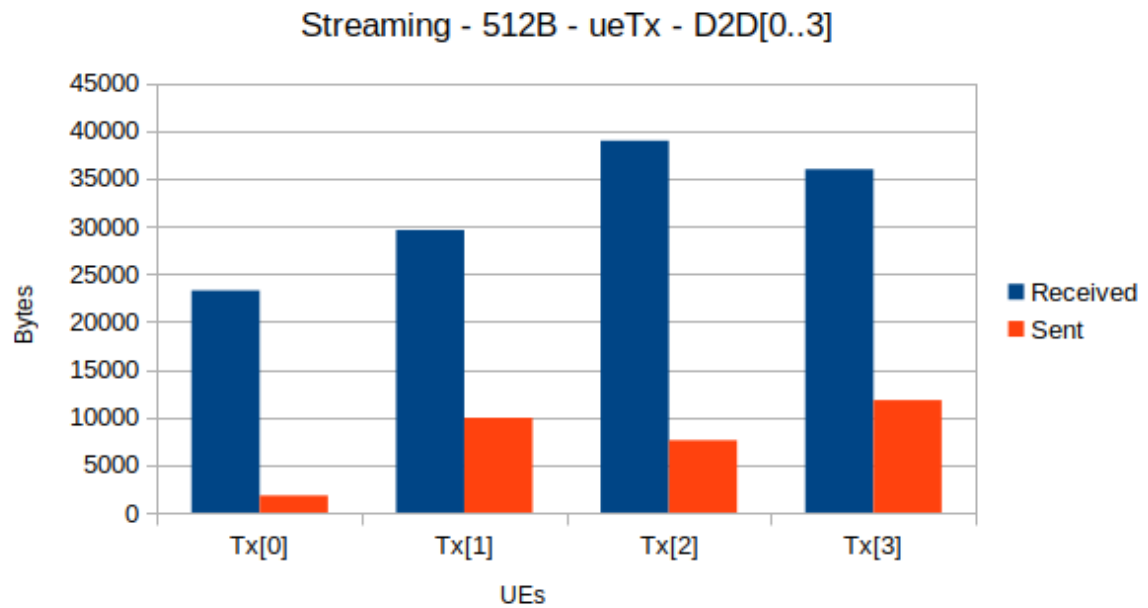


Figura B.57: Transmissão dos nós Tx no experimento UDP streaming 512B com 4 D2D.

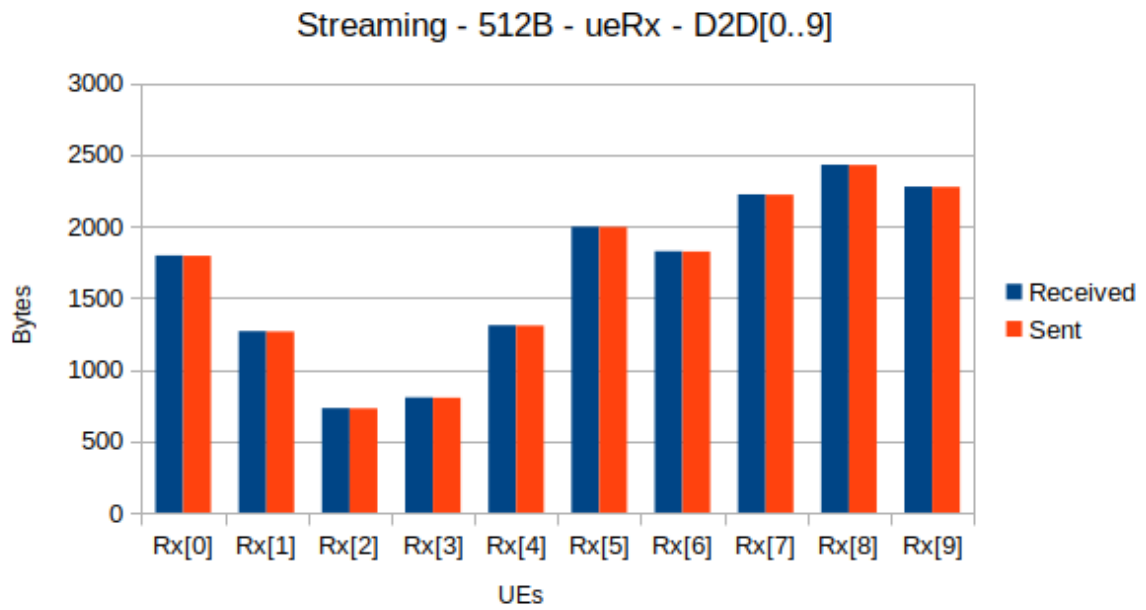


Figura B.58: Transmissão dos nós Rx no experimento UDP streaming 512B com 10 D2D.

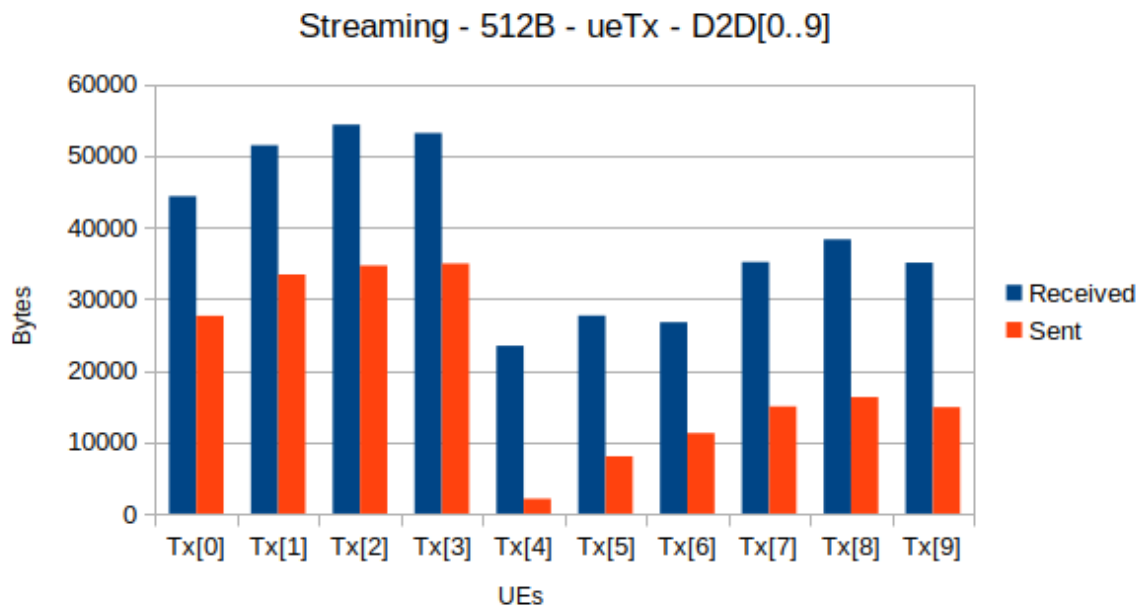


Figura B.59: Transmissão dos nós Tx no experimento UDP streaming 512B com 10 D2D.

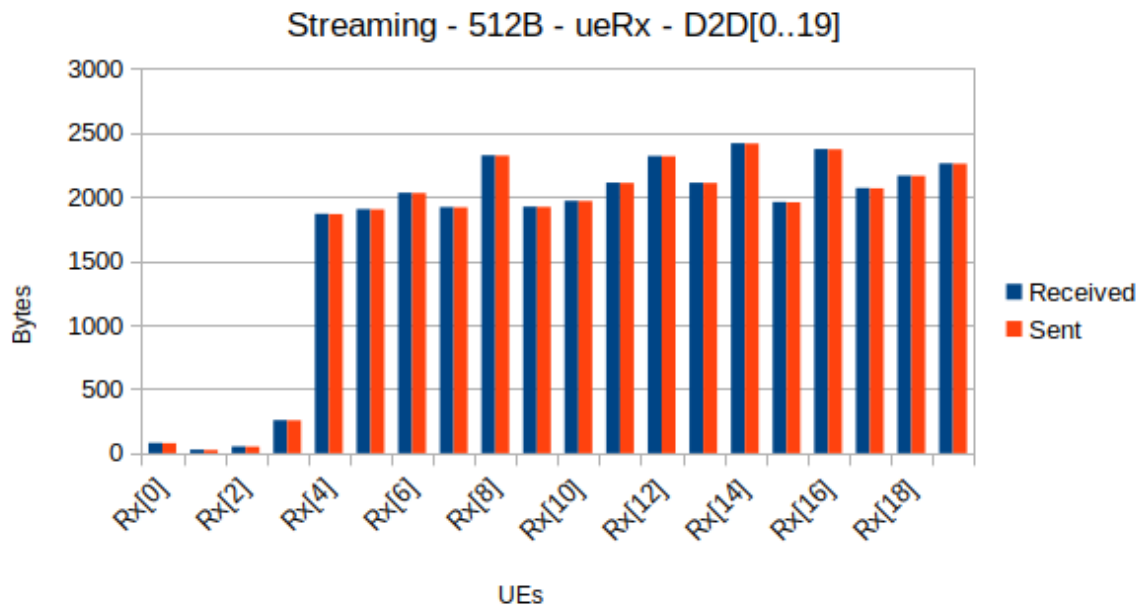


Figura B.60: Transmissão dos nós Rx no experimento UDP streaming 512B com 20 D2D.

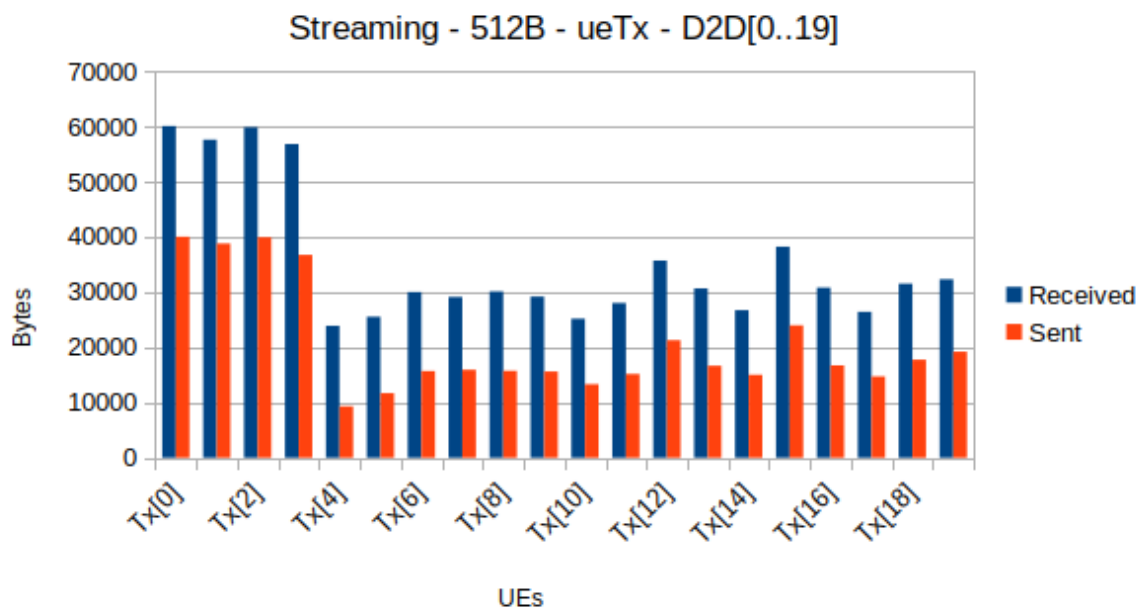


Figura B.61: Transmissão dos nós Tx no experimento UDP streaming 512B com 20 D2D.

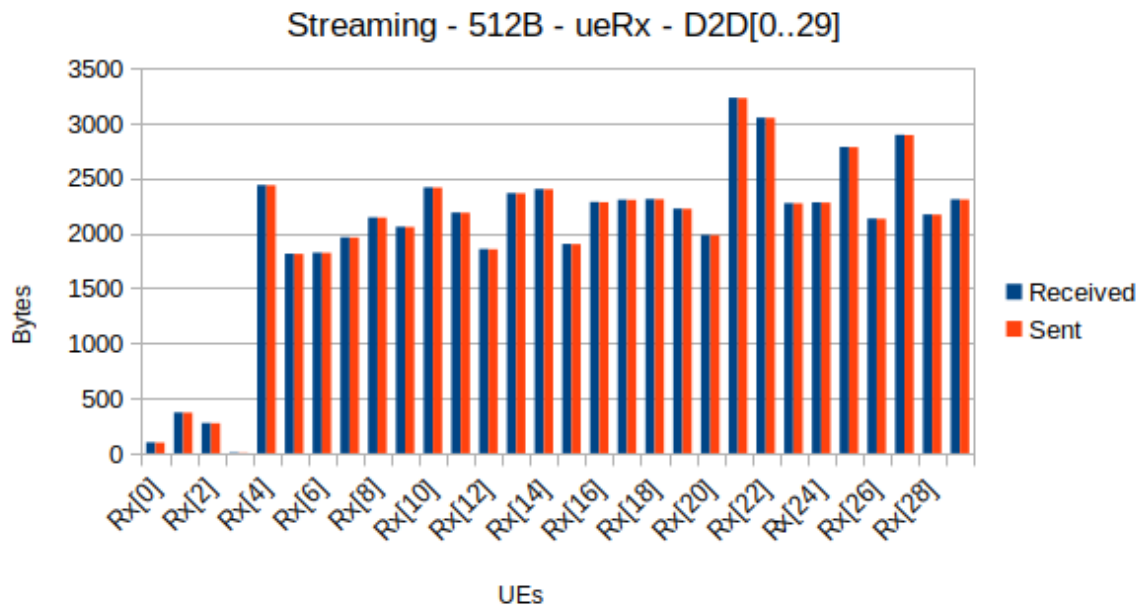


Figura B.62: Transmissão dos nós Rx no experimento UDP streaming 512B com 30 D2D.

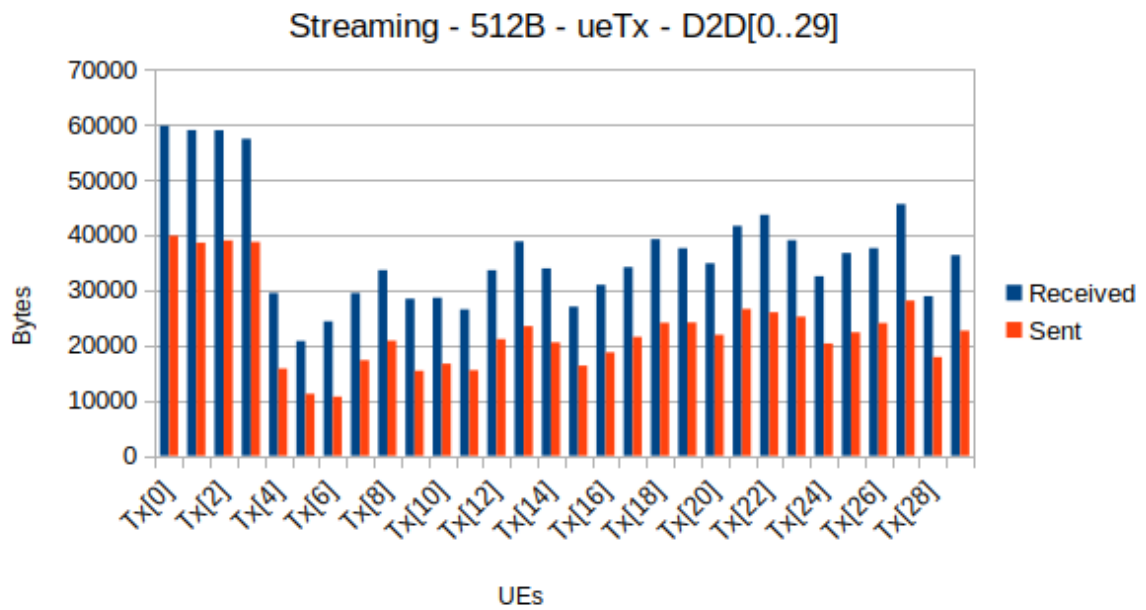


Figura B.63: Transmissão dos nós Tx no experimento UDP streaming 512B com 30 D2D.

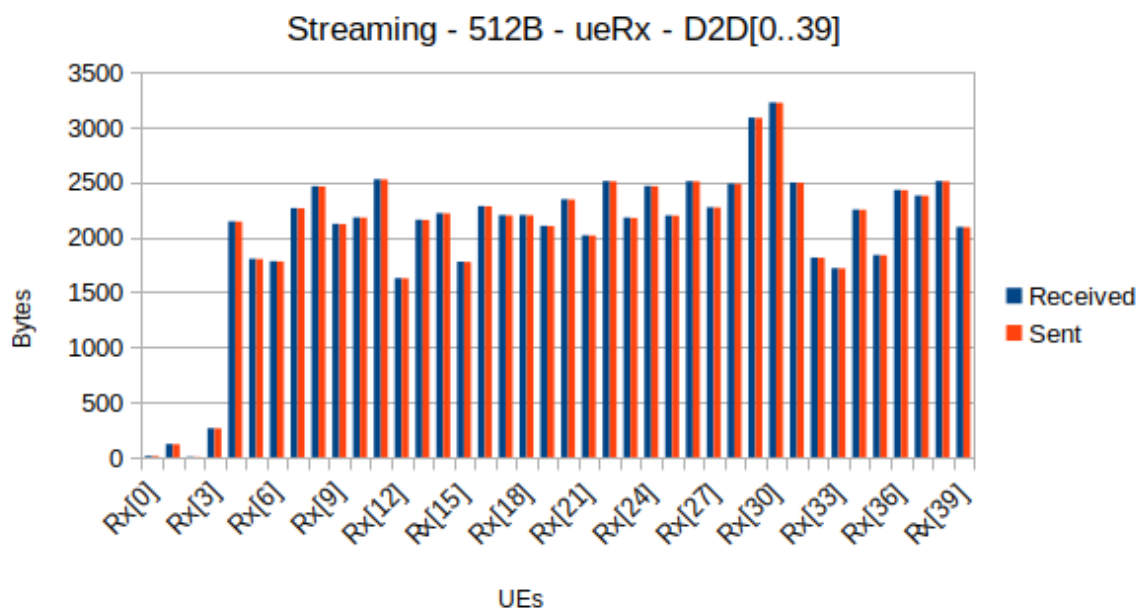


Figura B.64: Transmissão dos nós Rx no experimento UDP streaming 512B com 40 D2D.

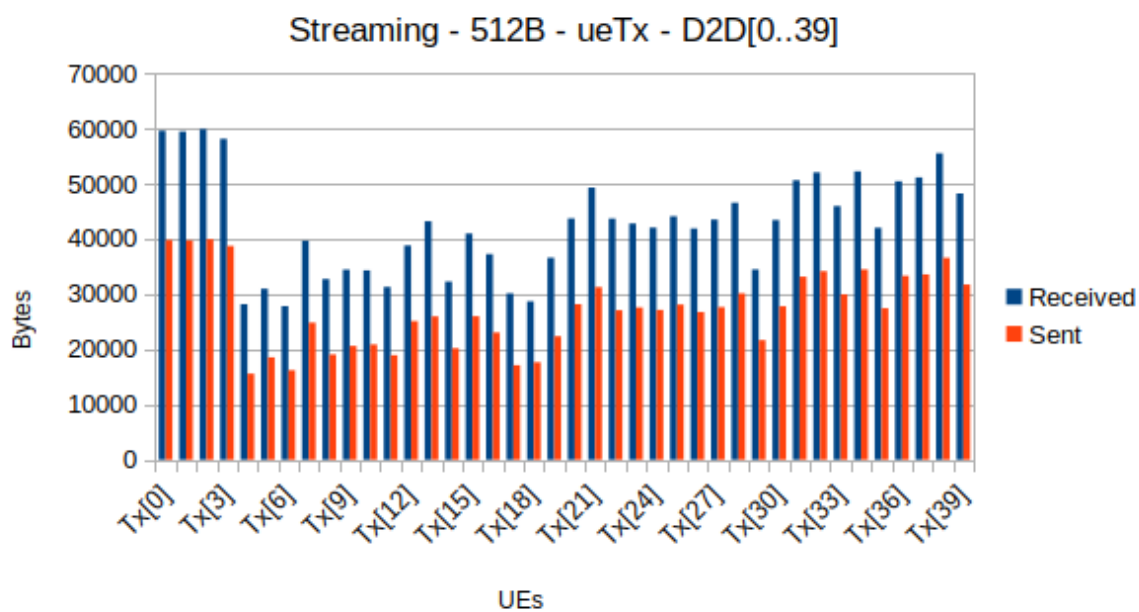


Figura B.65: Transmissão dos nós Tx no experimento UDP streaming 512B com 40 D2D.

B.8 experimento: UDP streaming 1024B

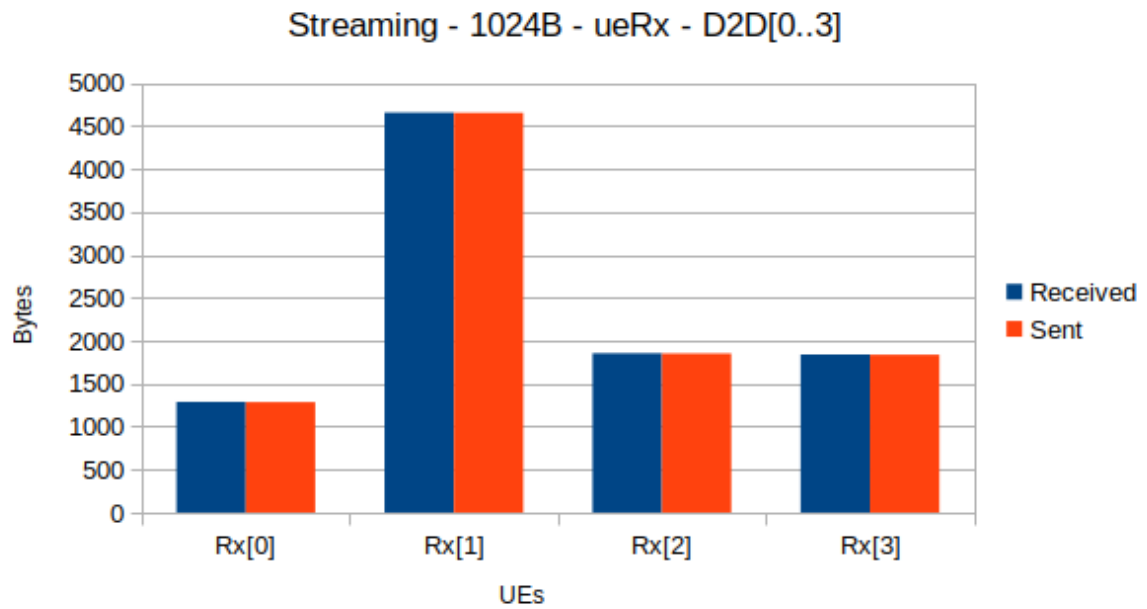


Figura B.66: Transmissão dos nós Rx no experimento UDP streaming 1024B com 4 D2D.

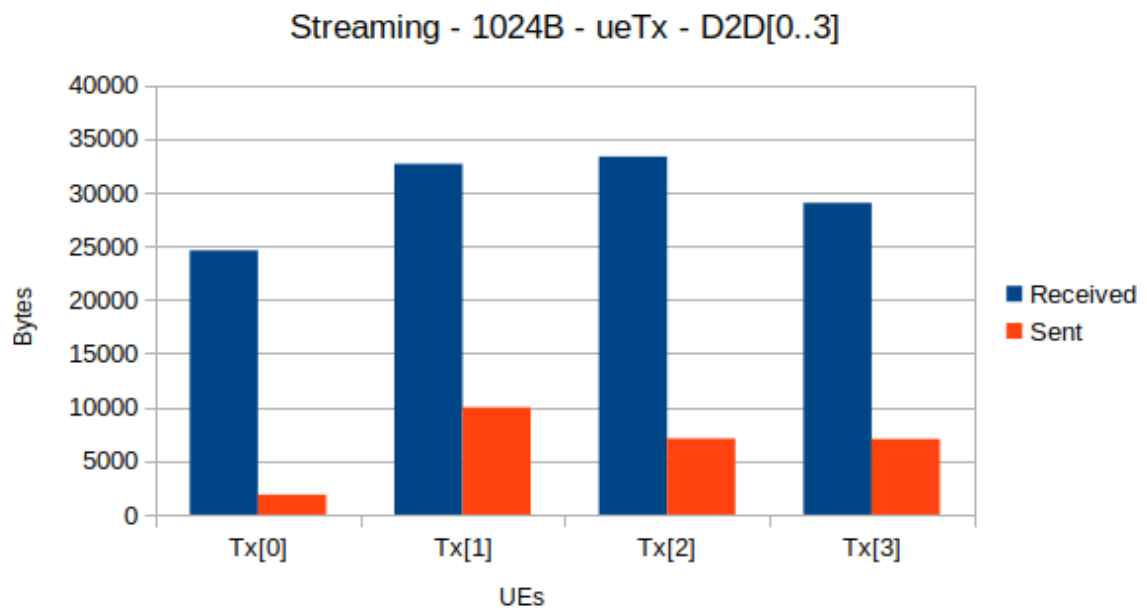


Figura B.67: Transmissão dos nós Tx no experimento UDP streaming 1024B com 4 D2D.

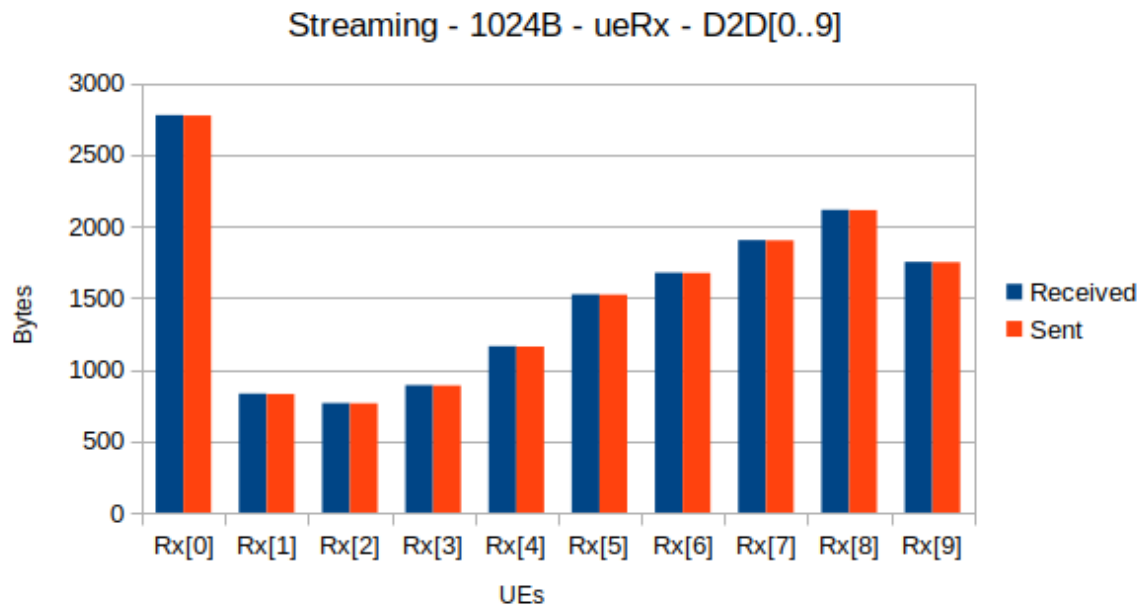


Figura B.68: Transmissão dos nós Rx no experimento UDP streaming 1024B com 10 D2D.

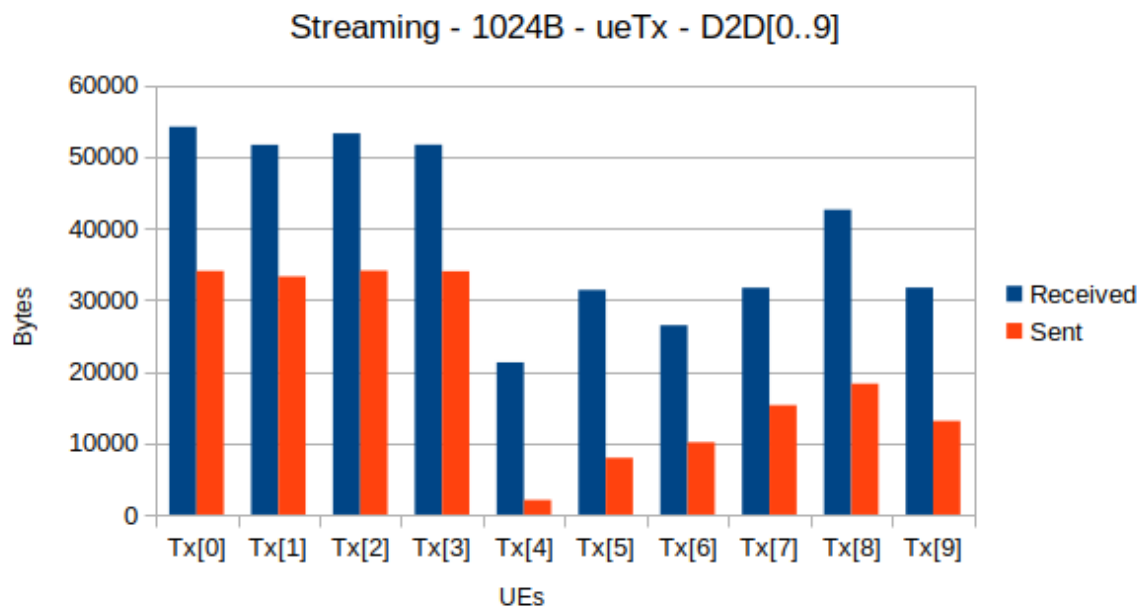


Figura B.69: Transmissão dos nós Tx no experimento UDP streaming 1024B com 10 D2D.

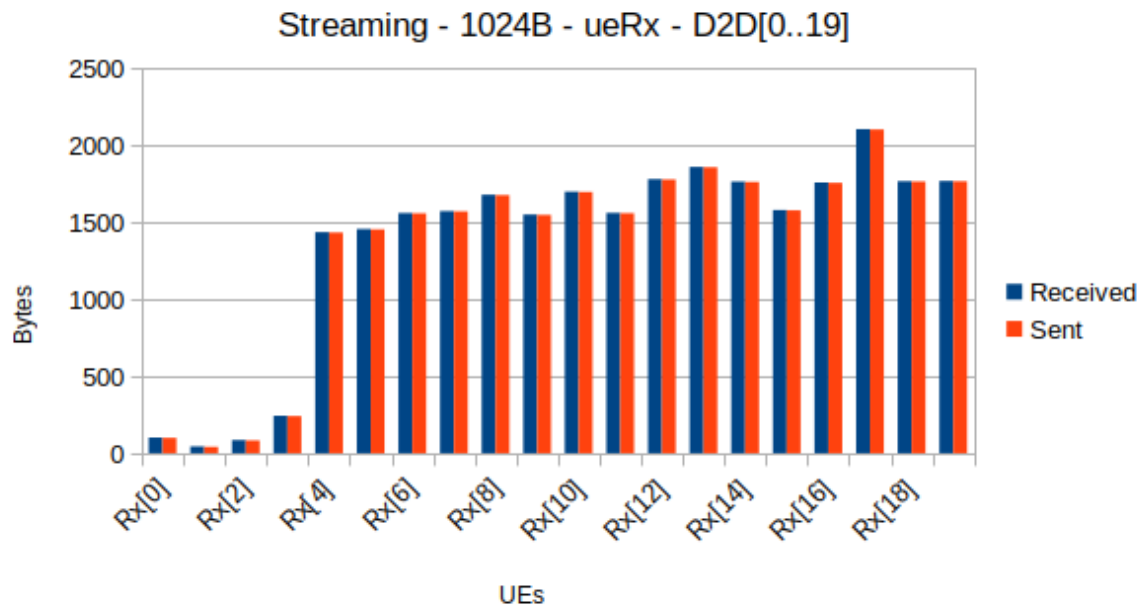


Figura B.70: Transmissão dos nós Rx no experimento UDP streaming 1024B com 20 D2D.

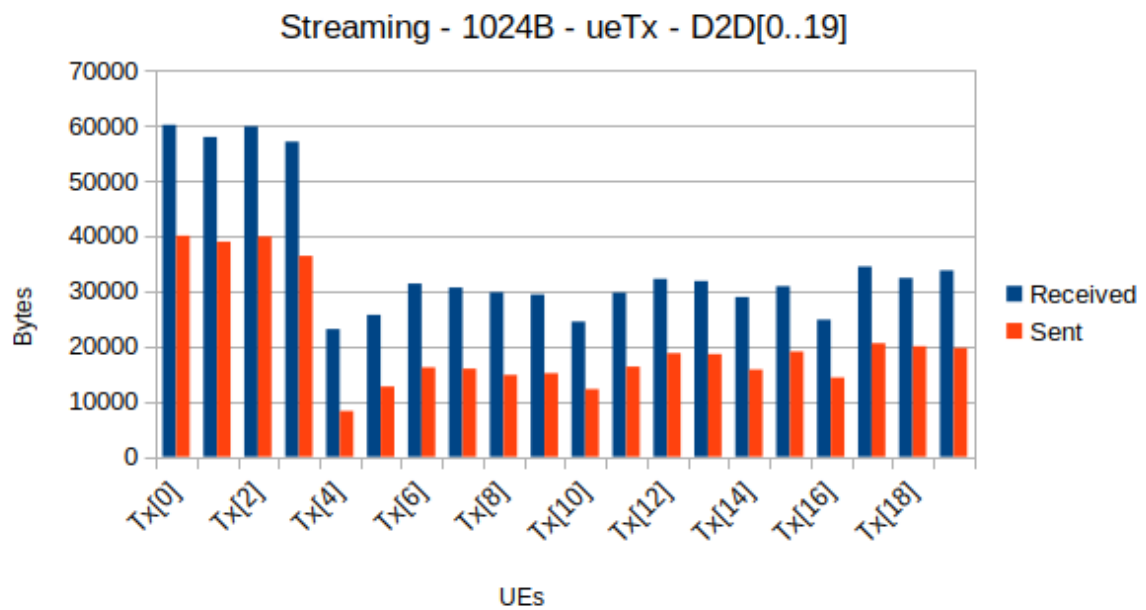


Figura B.71: Transmissão dos nós Tx no experimento UDP streaming 1024B com 20 D2D.

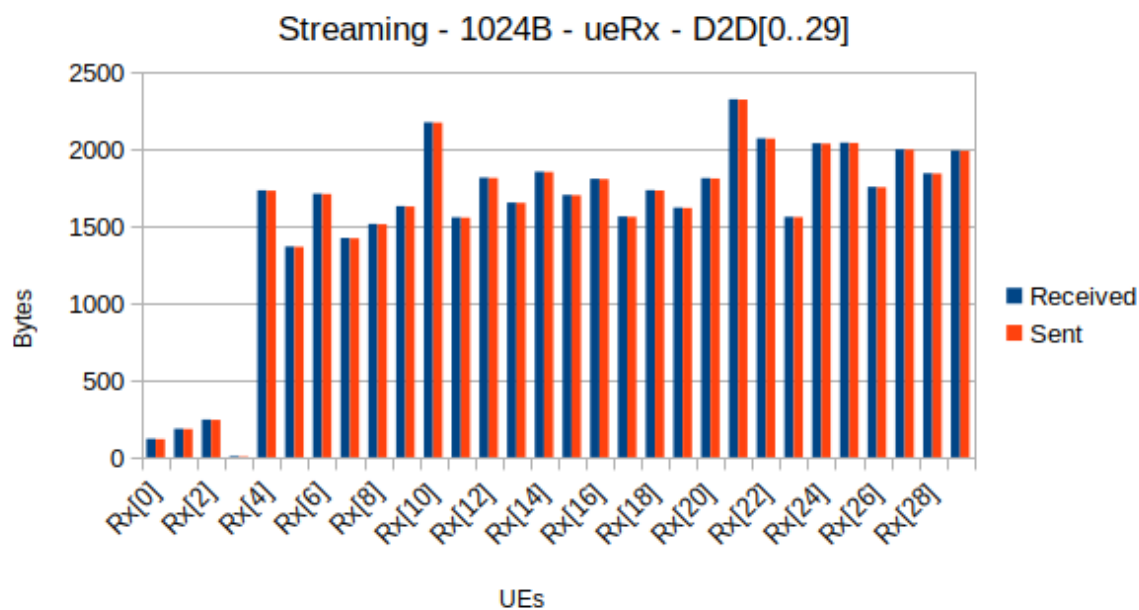


Figura B.72: Transmissão dos nós Rx no experimento UDP streaming 1024B com 30 D2D.

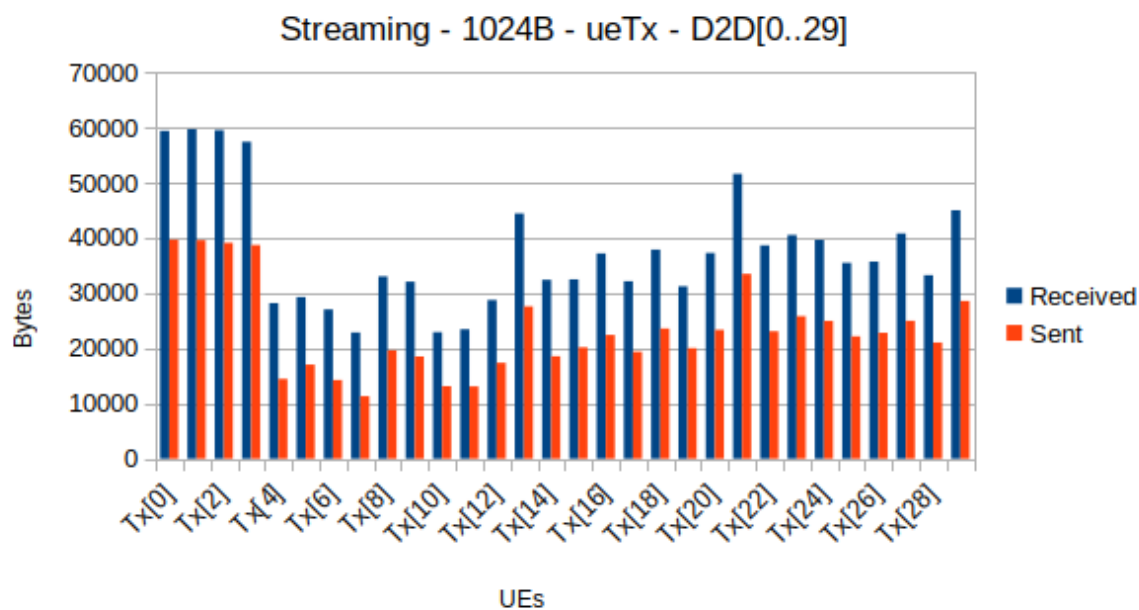


Figura B.73: Transmissão dos nós Tx no experimento UDP streaming 1024B com 30 D2D.

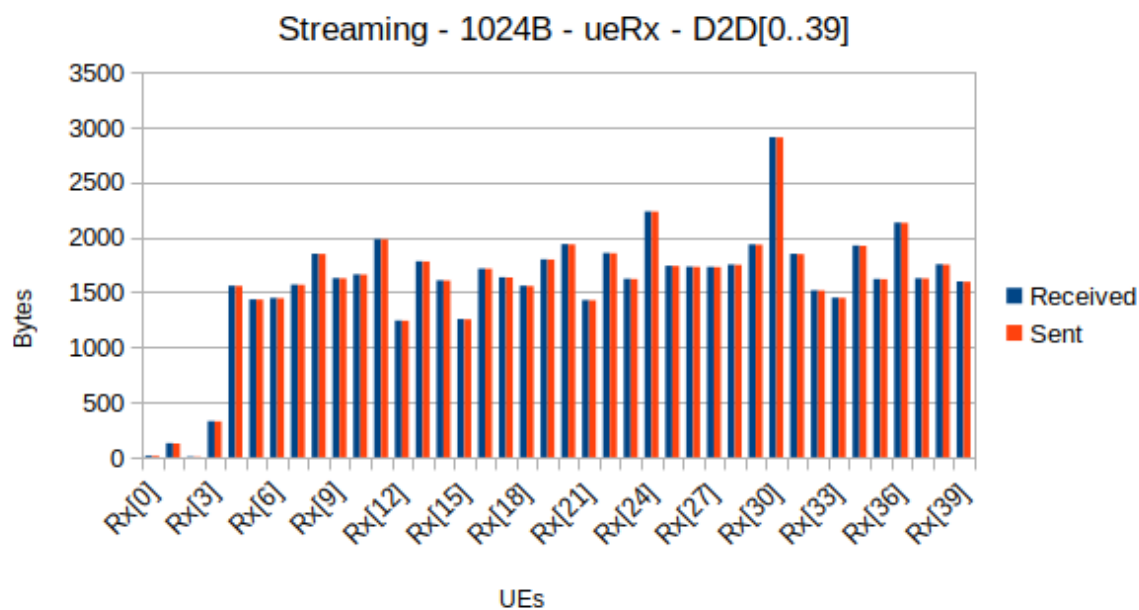


Figura B.74: Transmissão dos nós Rx no experimento UDP streaming 1024B com 40 D2D.

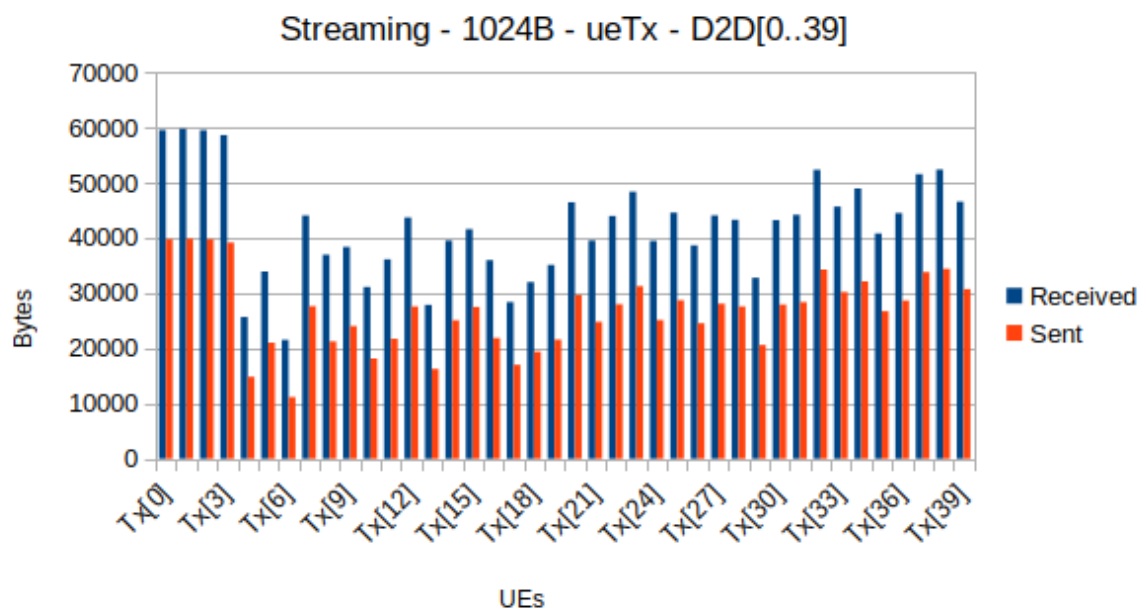


Figura B.75: Transmissão dos nós Tx no experimento UDP streaming 1024B com 40 D2D.

APÊNDICE C – Gráficos do Consumo Energético dos Experimentos de Streaming de Vídeo

C.1 Gráficos de Consumo Energético do Experimento: UDP *Streaming* 256 B

C.1.1 Consumo Energético do Caso Base do Experimento *Streaming* 256 B

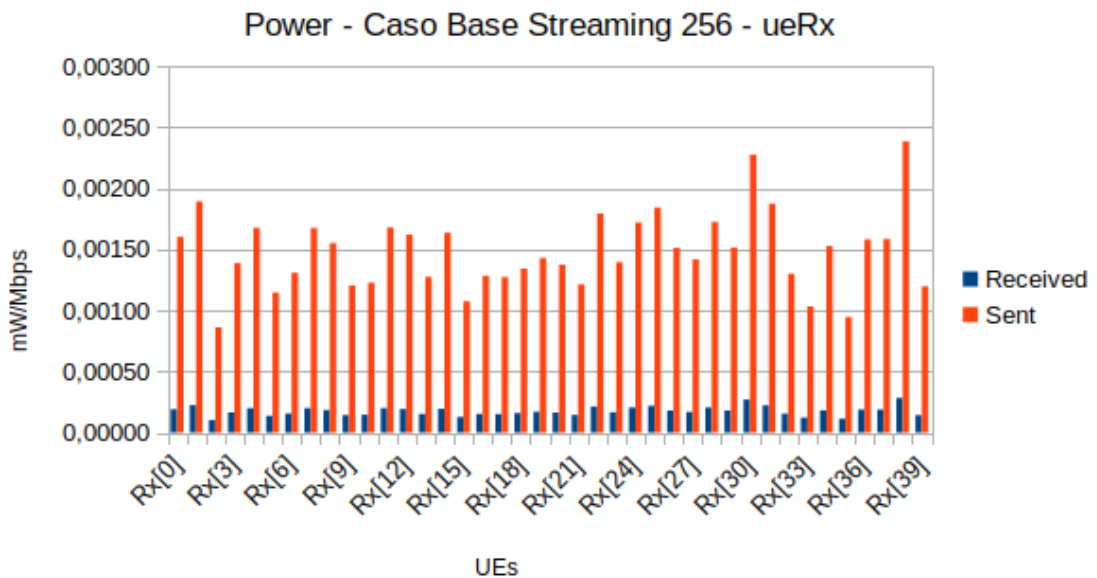


Figura C.1: Consumo energético dos UEs Rx no experimento streaming 256 B.

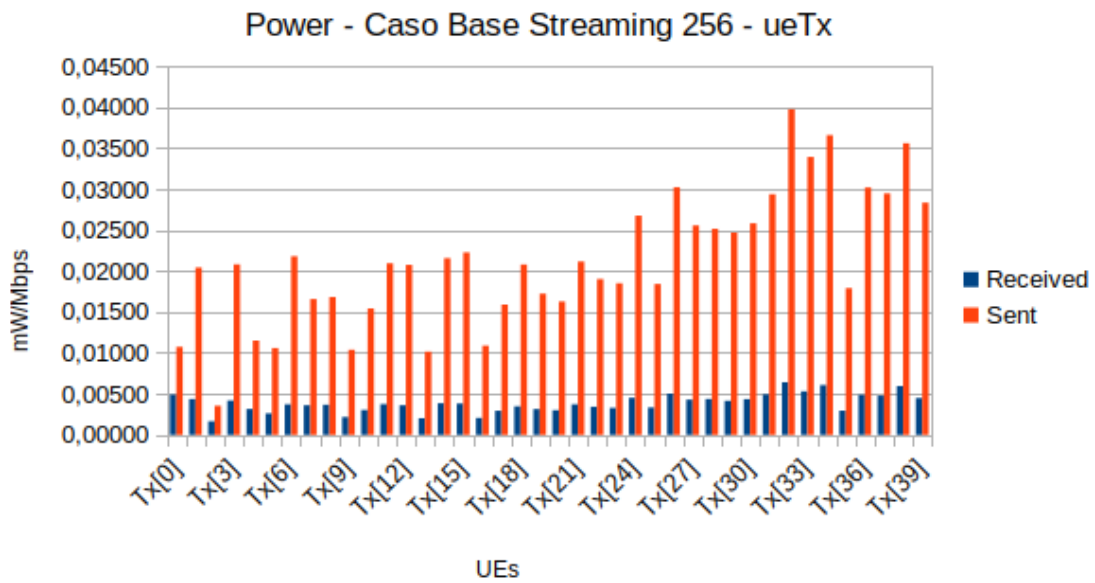


Figura C.2: Consumo energético dos UEs Tx no experimento streaming 256 B.

C.1.2 Consumo Energético dos UEs Tx no Experimento *Streaming* 256 B

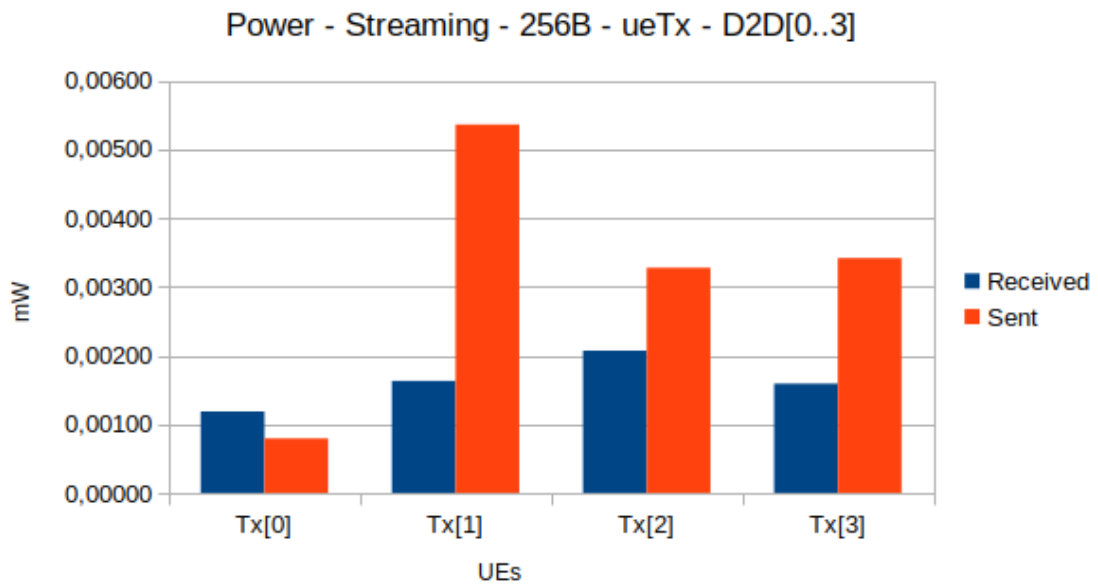


Figura C.3: Consumo energético dos UEs Tx no experimento streaming 256 B com 4 encaminhadores.

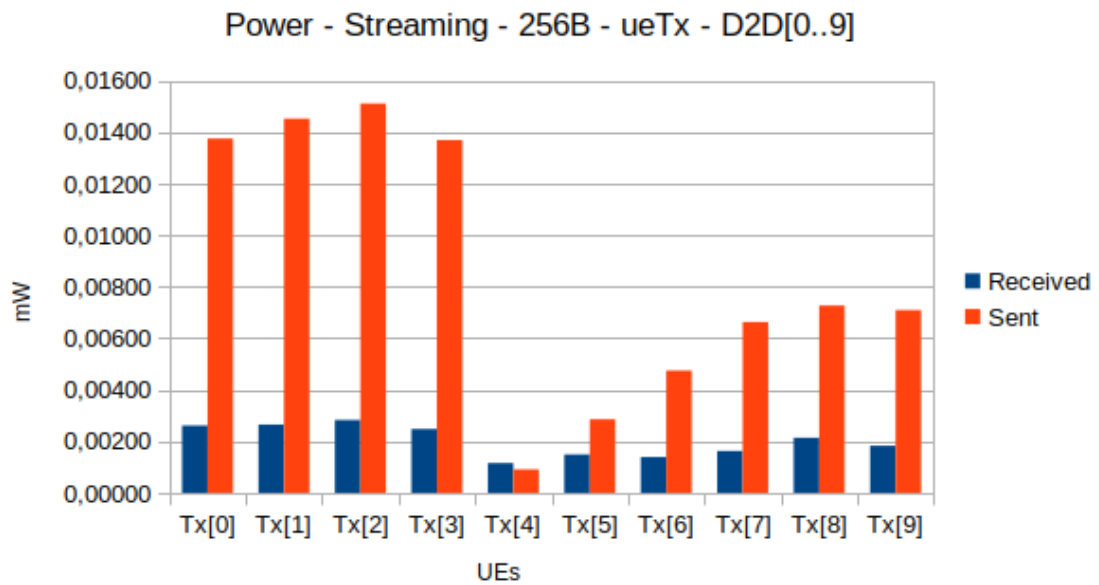


Figura C.4: Consumo energético dos UEs Tx no experimento streaming 256 B com 10 encaminhadores.

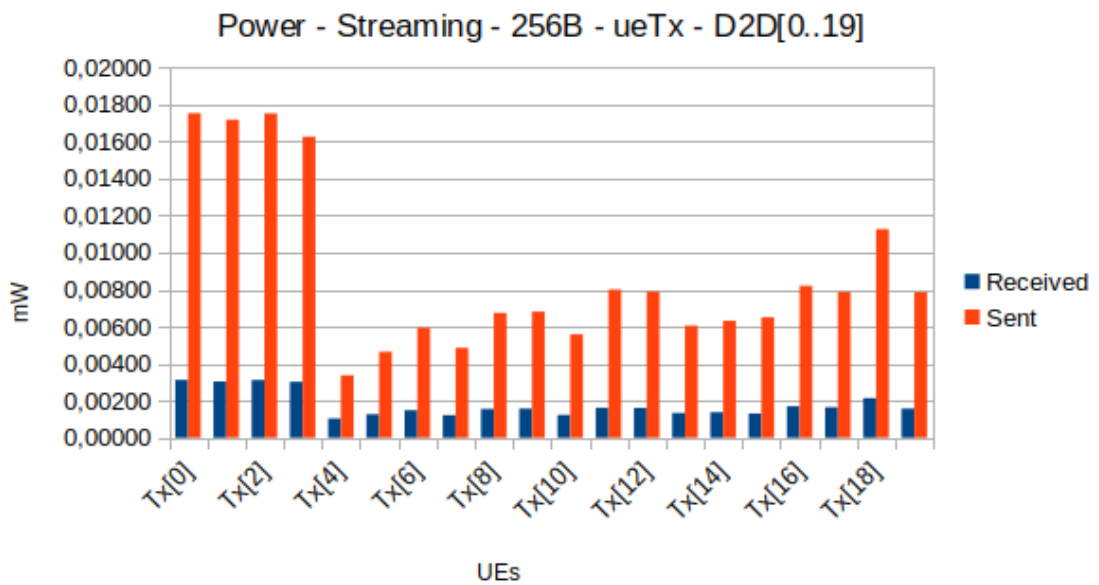


Figura C.5: Consumo energético dos UEs Tx no experimento streaming 256 B com 210 encaminhadores.

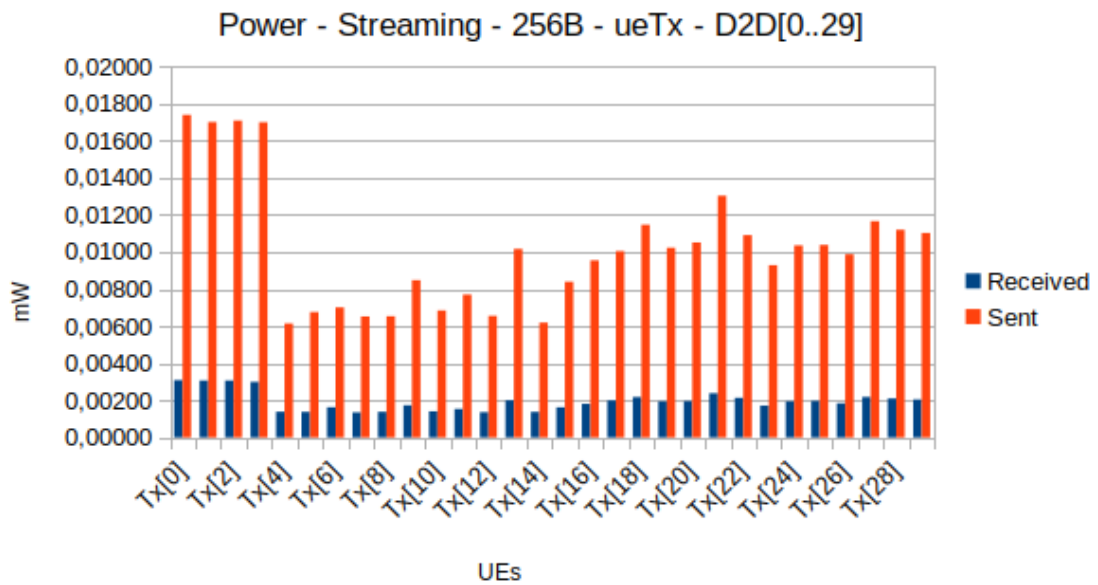


Figura C.6: Consumo energético dos UEs Tx no experimento streaming 256 B com 30 encaminhadores.

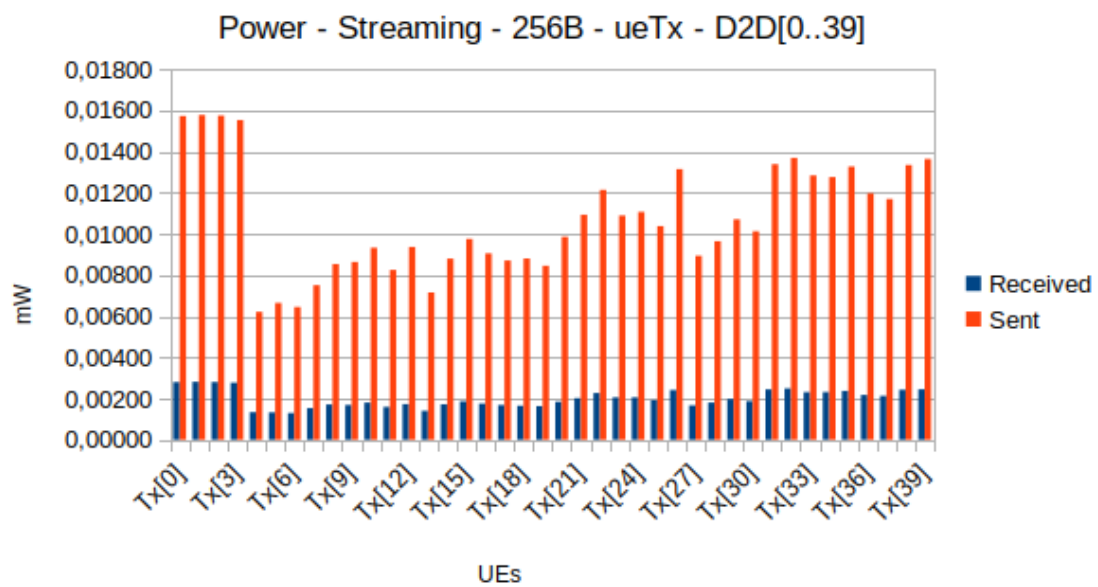


Figura C.7: Consumo energético dos UEs Tx no experimento streaming 256 B com 40 encaminhadores.

C.1.3 Consumo Energético dos UEs Rx no Experimento *Streaming* 256 B

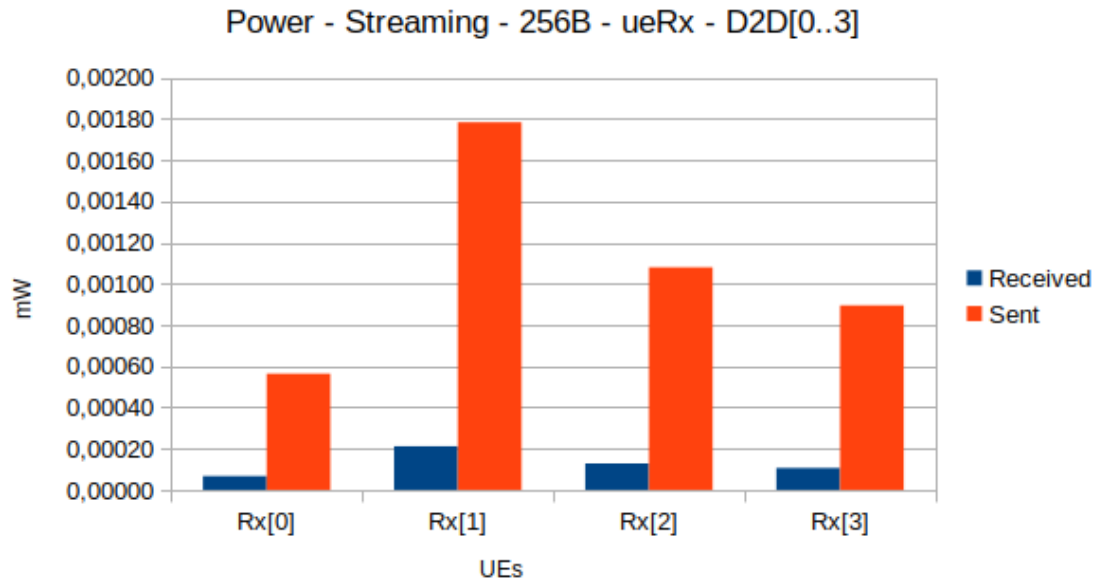


Figura C.8: Consumo energético dos UEs Rx no experimento streaming 256 B com 4 encaminhadores.

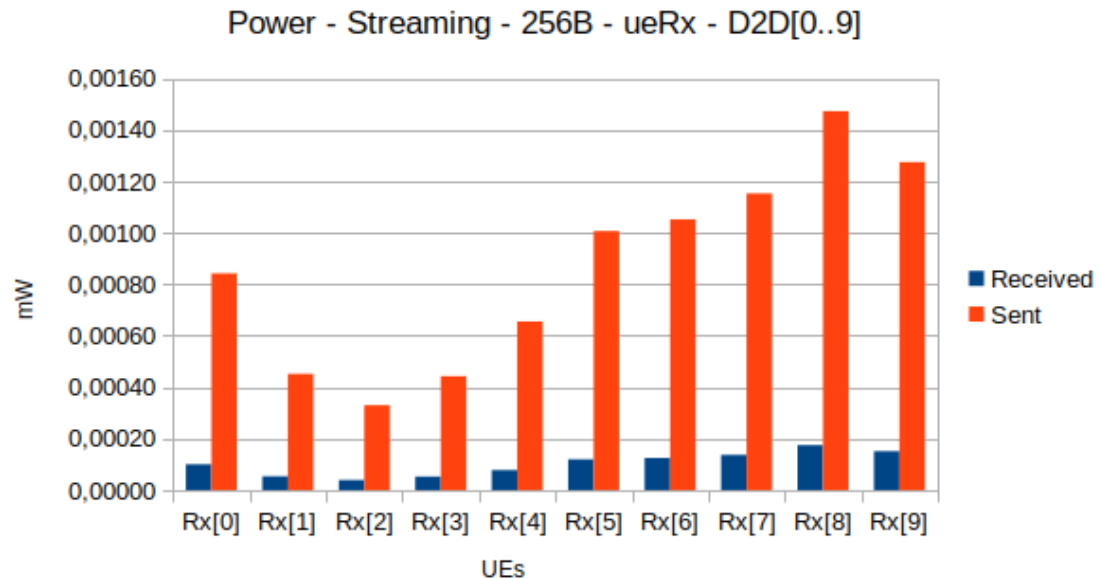


Figura C.9: Consumo energético dos UEs Rx no experimento streaming 256 B com 10 encaminhadores.

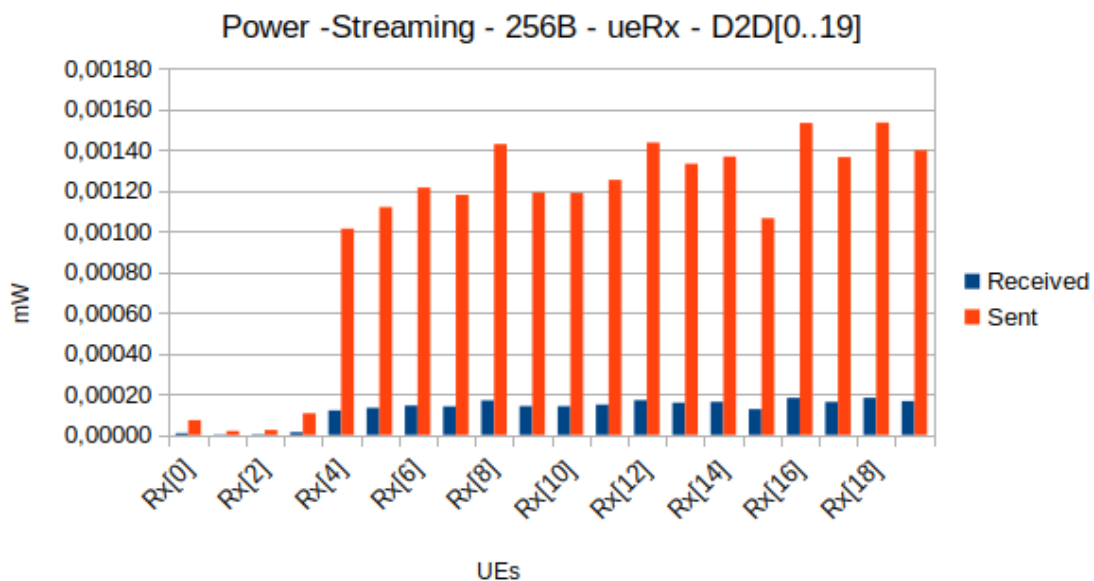


Figura C.10: Consumo energético dos UEs Rx no experimento streaming 256 B com 210 encaminhadores.

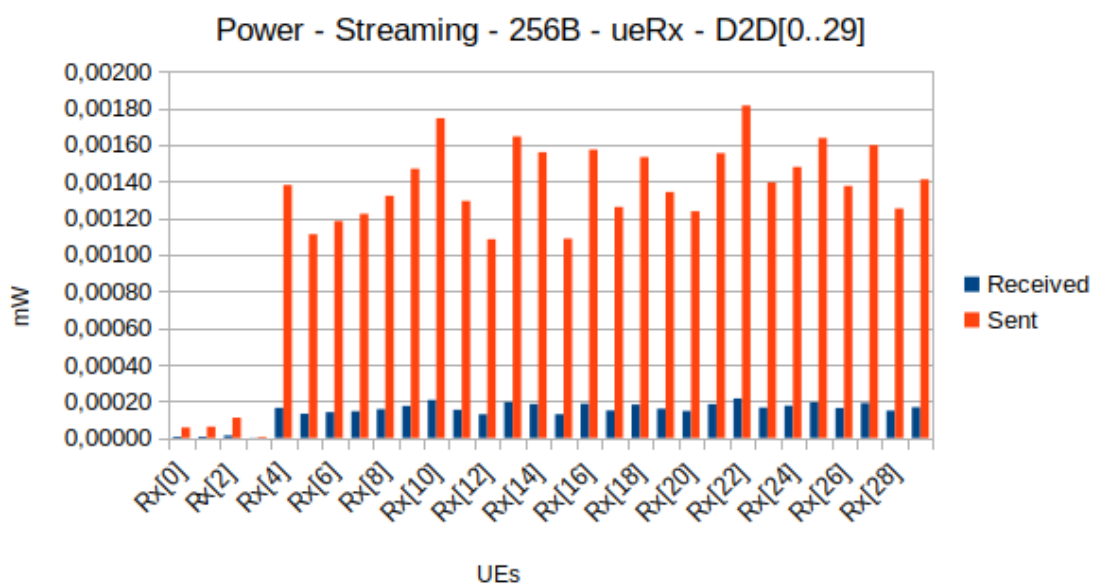


Figura C.11: Consumo energético dos UEs Rx no experimento streaming 256 B com 30 encaminhadores.

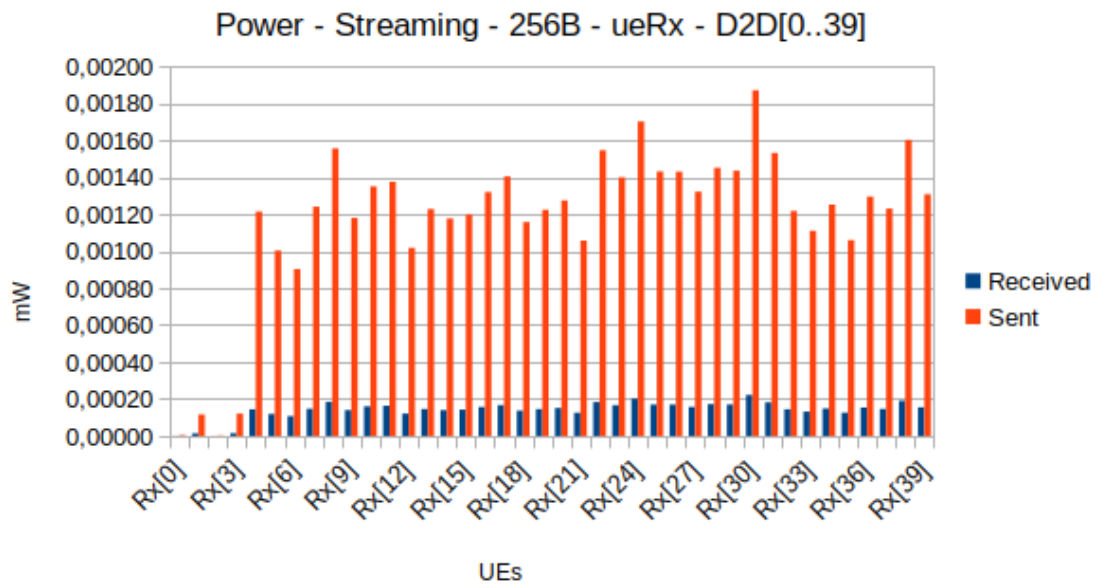


Figura C.12: Consumo energético dos UEs Rx no experimento streaming 256 B com 40 encaminhadores.

C.1.4 Consumo Energético Total da Rede no Experimento *Streaming* 256 B

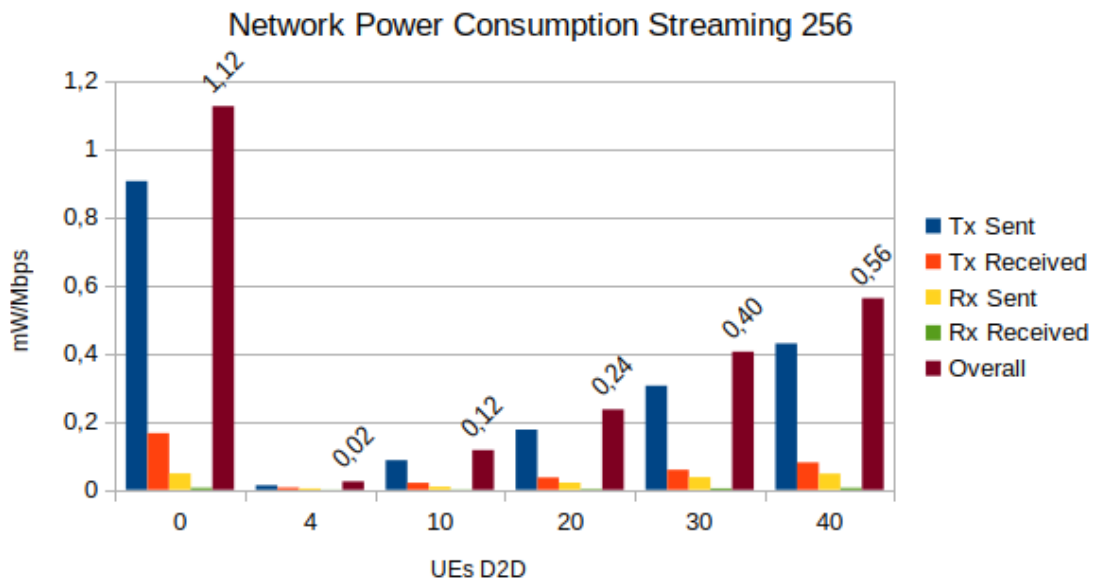


Figura C.13: Consumo energético total da rede e dos UEs Rx e Tx no experimento streaming 256 B para cada cenário.

C.2 Gráficos de Consumo Energético do Experimento: UDP *Streaming* 256 B

C.2.1 Consumo Energético do Caso Base do Experimento *Streaming* 256 B

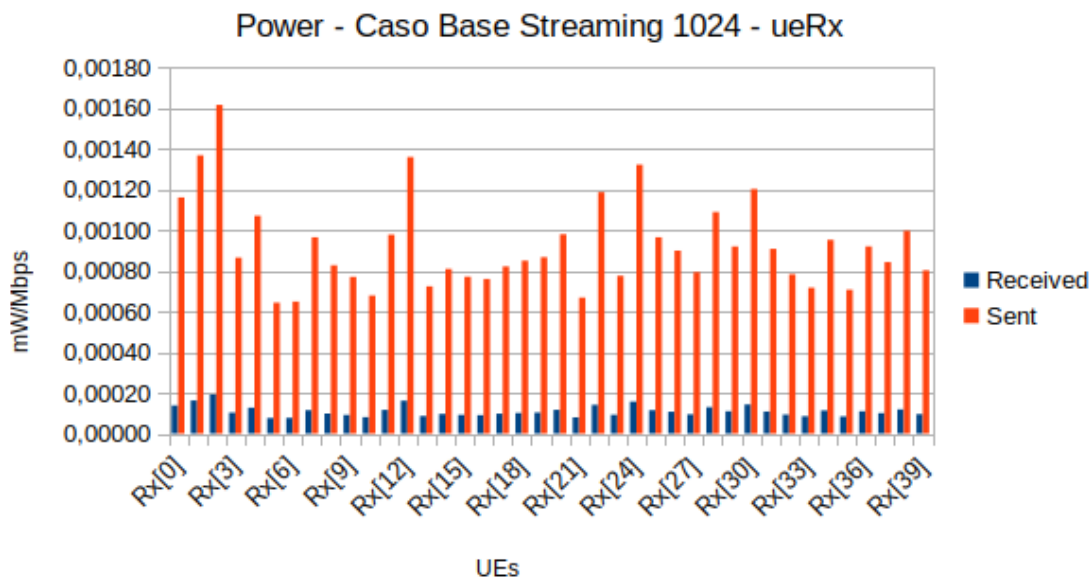


Figura C.14: Consumo energético dos UEs Rx no experimento streaming 256 B.

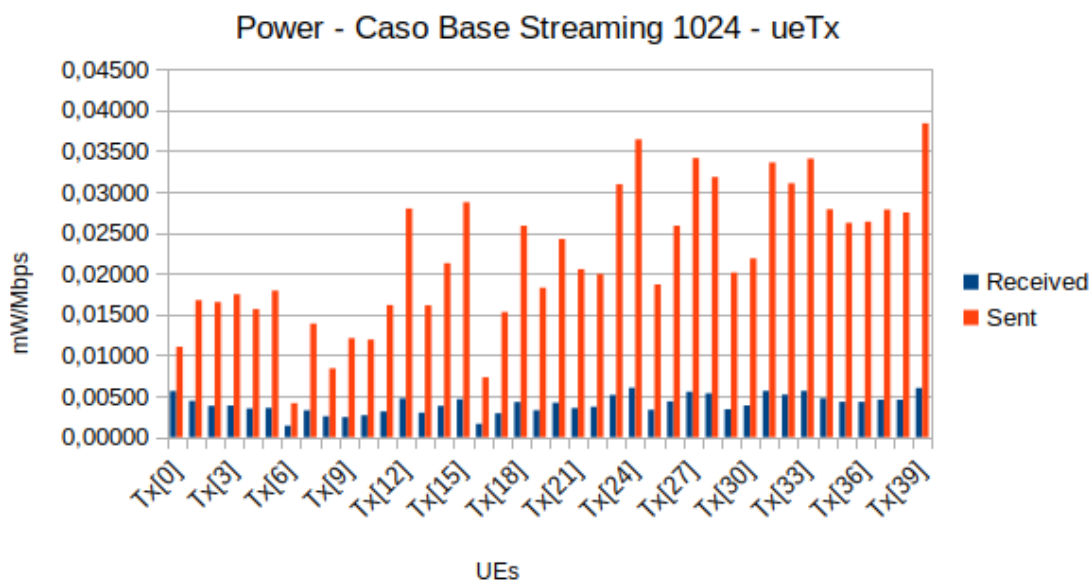


Figura C.15: Consumo energético dos UEs Tx no experimento streaming 256 B.

C.2.2 Consumo Energético dos UEs Tx no Experimento *Streaming* 256 B

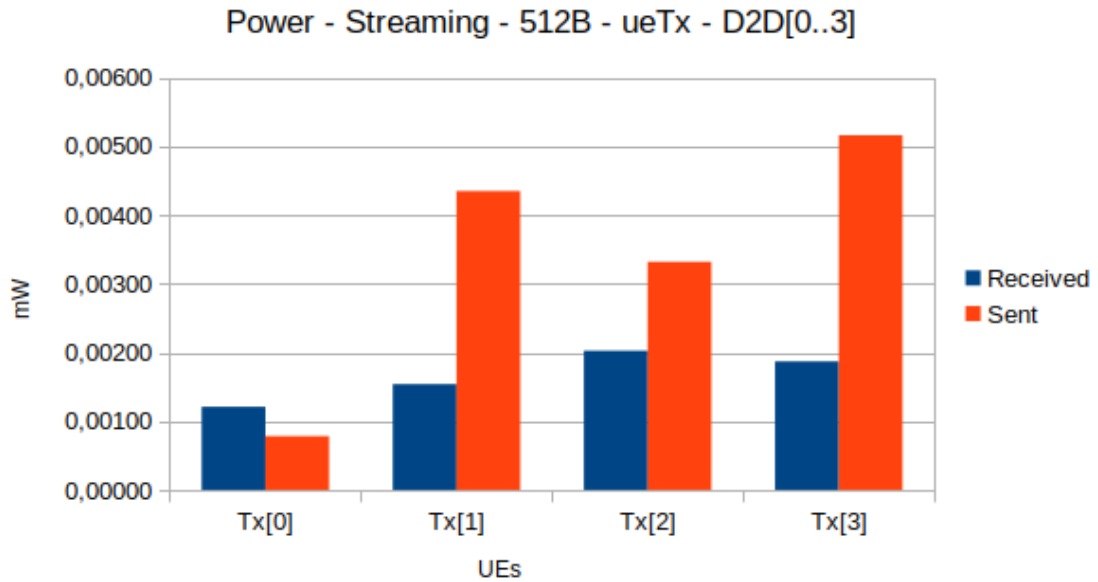


Figura C.16: Consumo energético dos UEs Tx no experimento streaming 256 B com 4 encaminhadores.

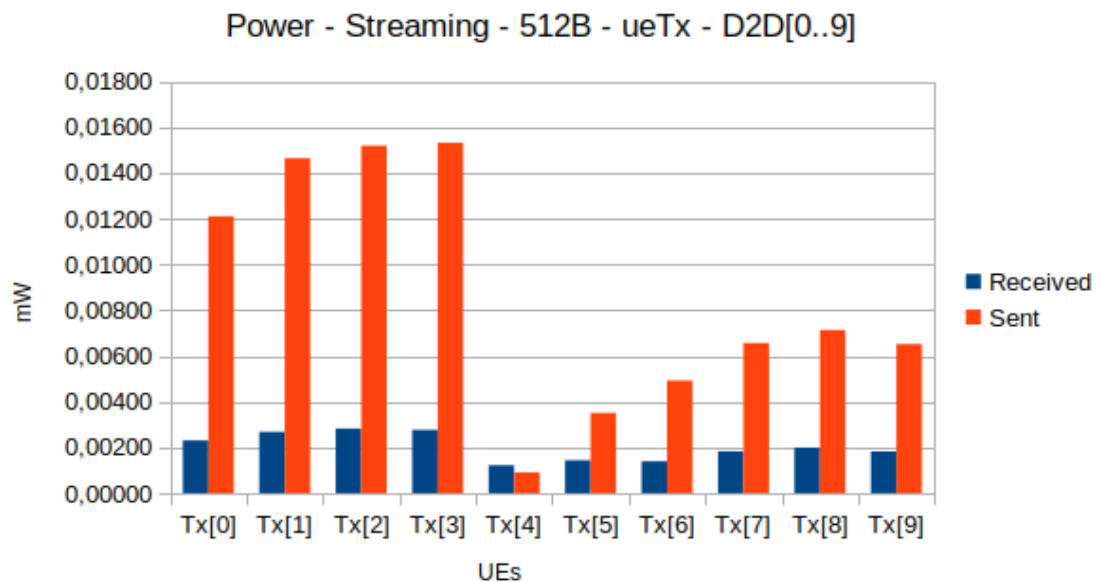


Figura C.17: Consumo energético dos UEs Tx no experimento streaming 256 B com 10 encaminhadores.

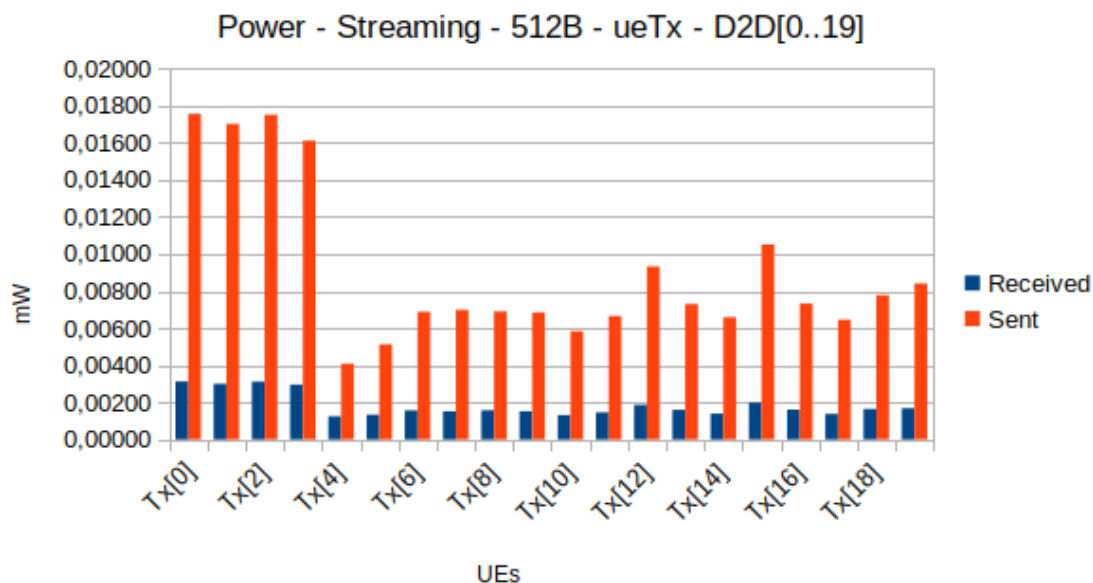


Figura C.18: Consumo energético dos UEs Tx no experimento streaming 256 B com 210 encaminhadores.

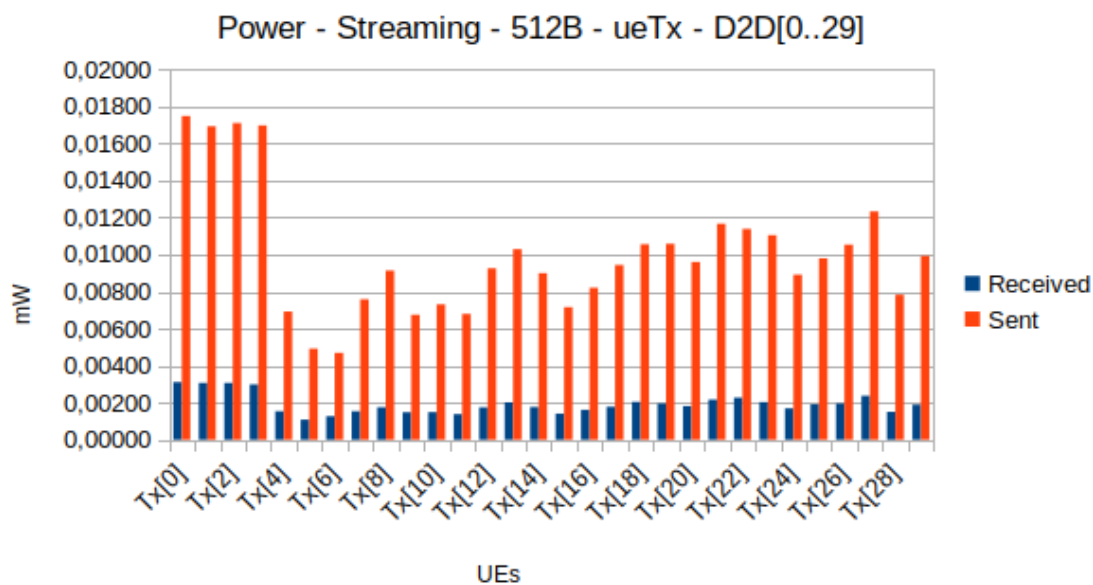


Figura C.19: Consumo energético dos UEs Tx no experimento streaming 256 B com 30 encaminhadores.

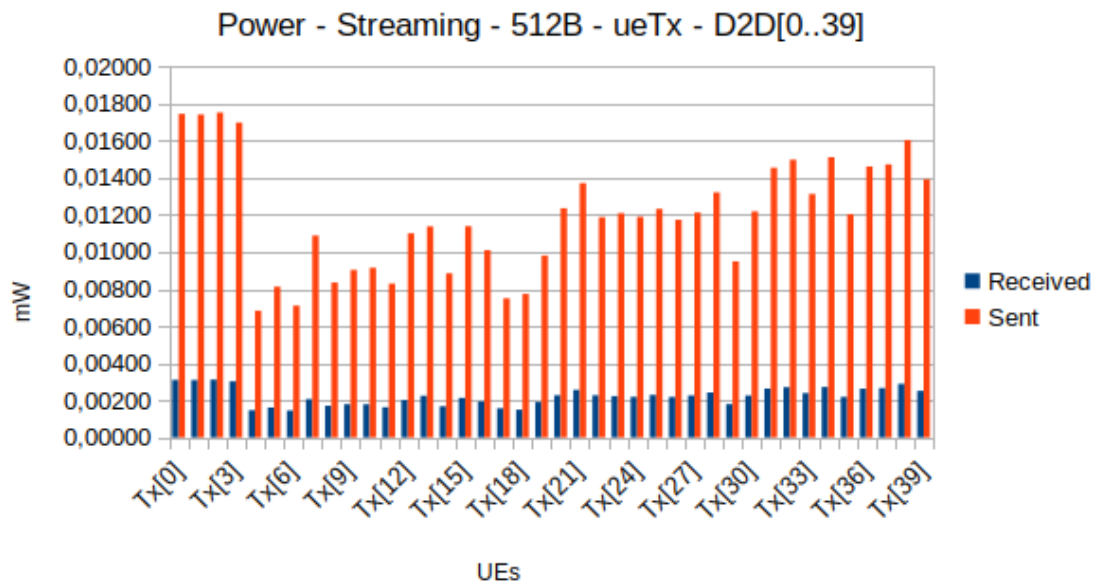


Figura C.20: Consumo energético dos UEs Tx no experimento streaming 256 B com 40 encaminhadores.

C.2.3 Consumo Energético dos UEs Rx no Experimento *Streaming* 256 B

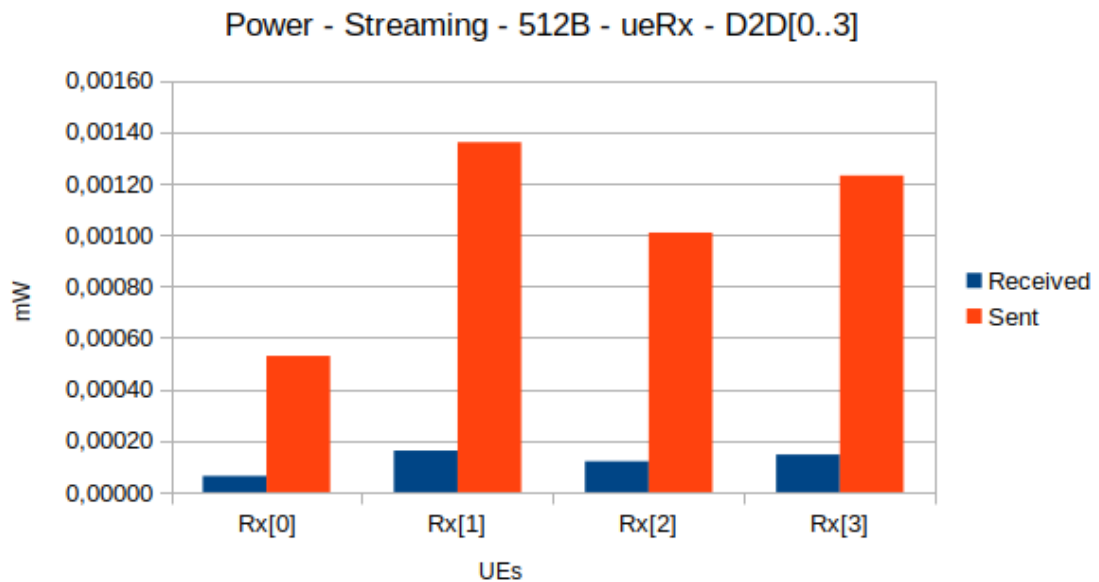


Figura C.21: Consumo energético dos UEs Rx no experimento streaming 256 B com 4 encaminhadores.

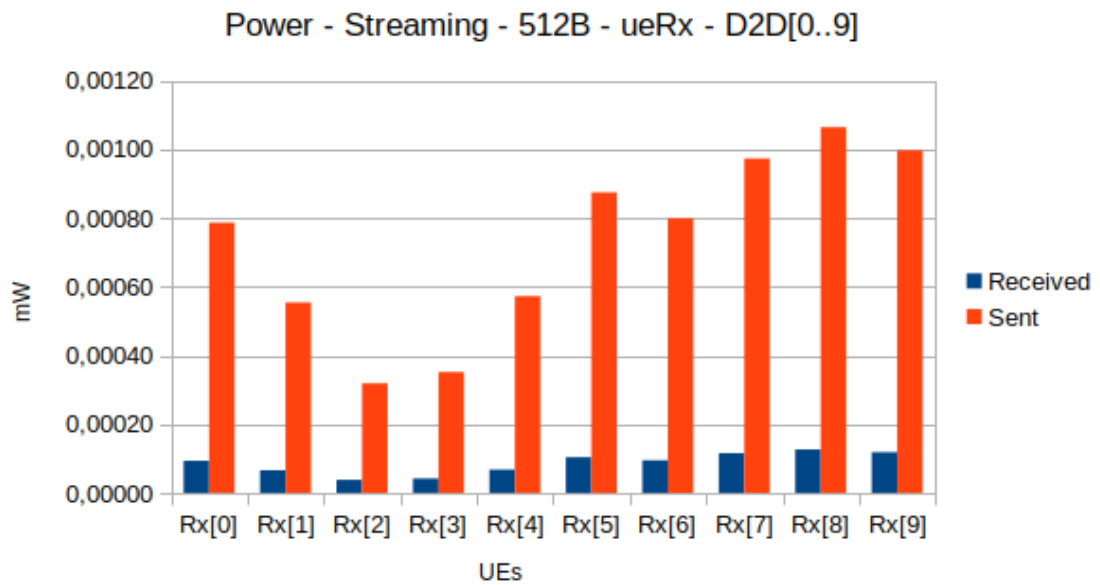


Figura C.22: Consumo energético dos UEs Rx no experimento streaming 256 B com 10 encaminhadores.

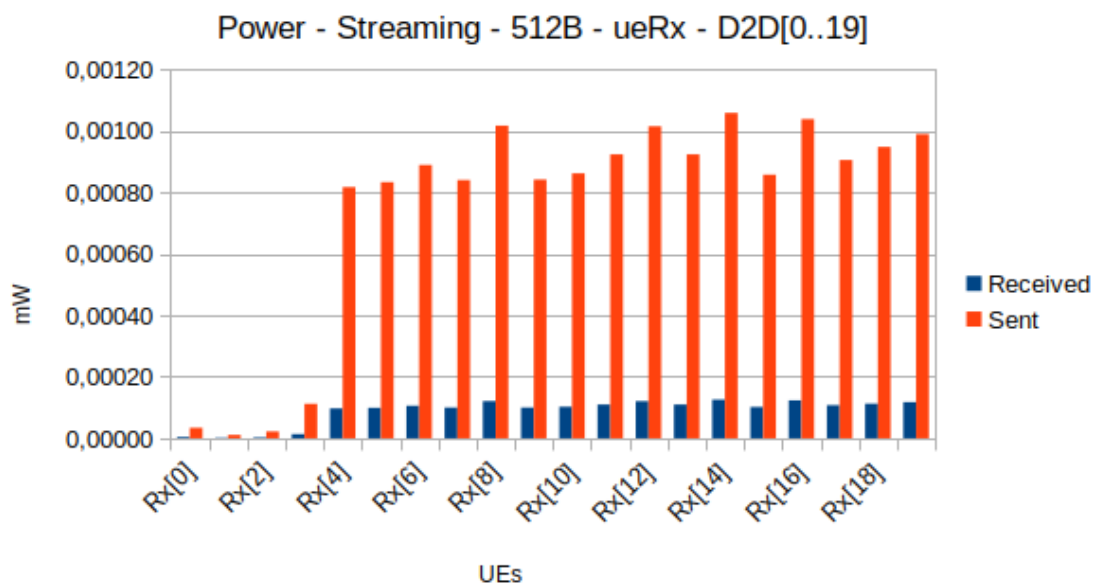


Figura C.23: Consumo energético dos UEs Rx no experimento streaming 256 B com 210 encaminhadores.

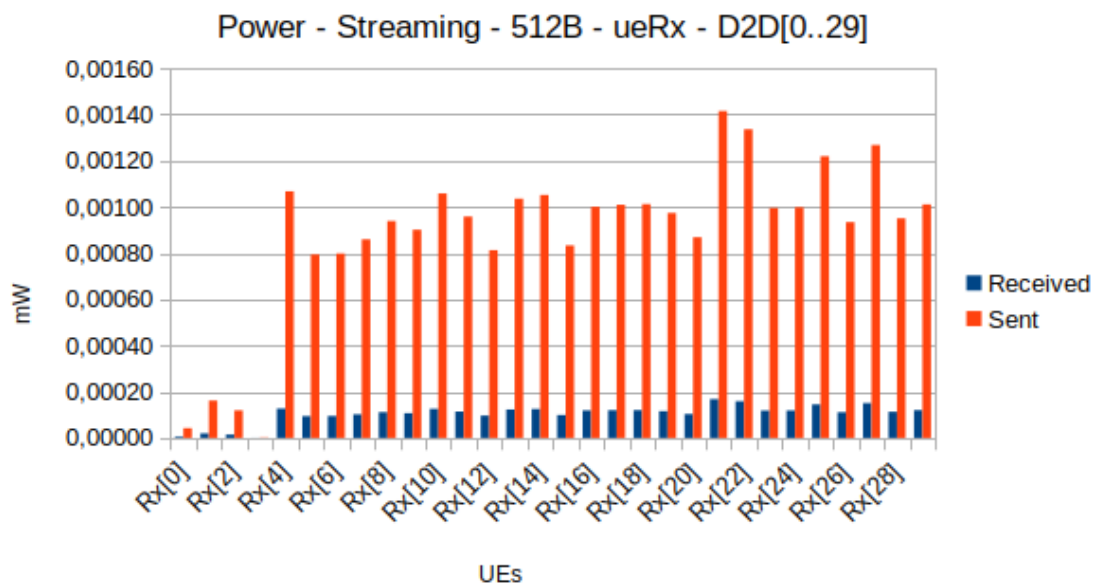


Figura C.24: Consumo energético dos UEs Rx no experimento streaming 256 B com 30 encaminhadores.

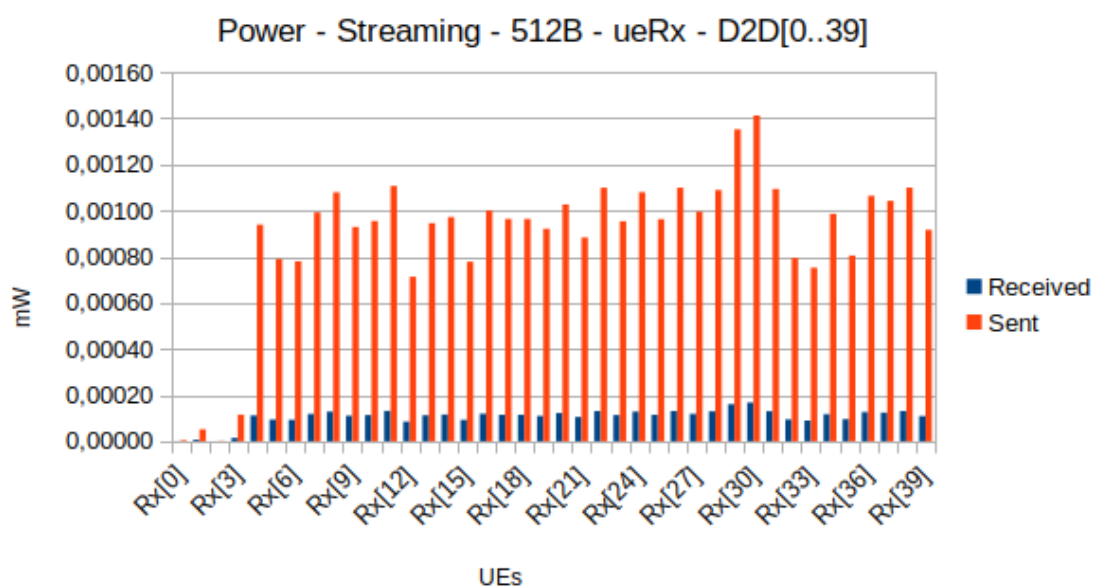


Figura C.25: Consumo energético dos UEs Rx no experimento streaming 256 B com 40 encaminhadores.

C.2.4 Consumo Energético Total da Rede no Experimento *Streaming* 256 B

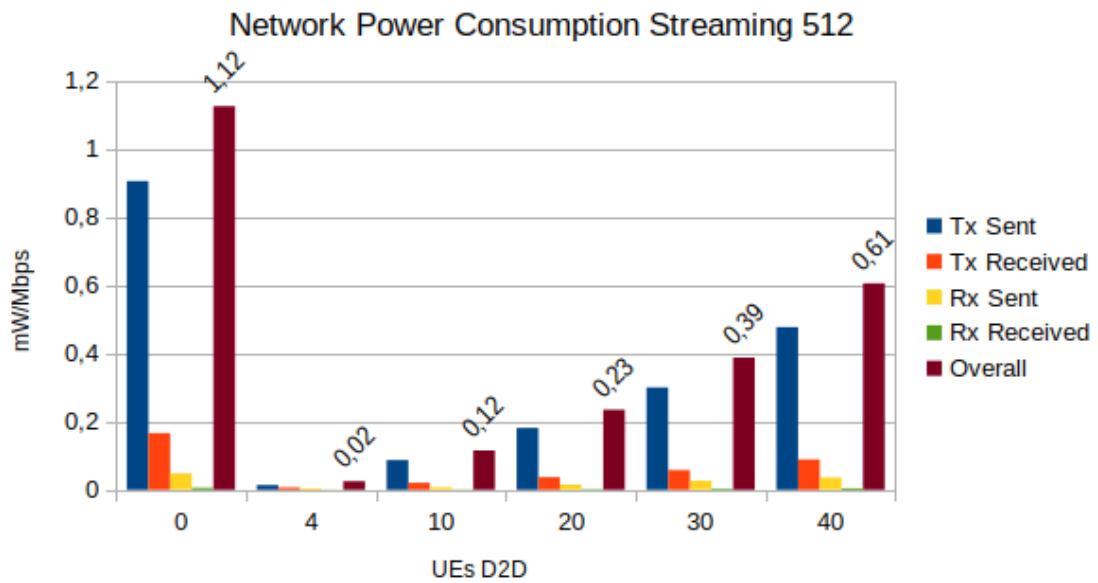


Figura C.26: Consumo Energético Total da Rede e dos UEs Rx e Tx no Experimento Streaming 256 B para cada Cenário.

C.3 Gráficos de Consumo Energético do Experimento: UDP *Streaming* 256 B

C.3.1 Consumo Energético do Caso Base do Experimento *Streaming* 256 B

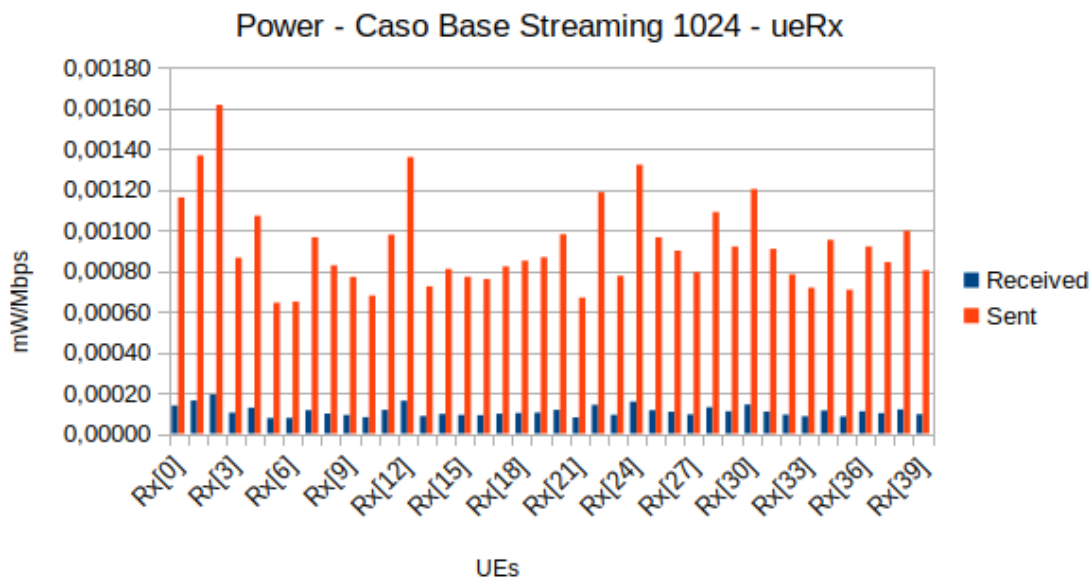


Figura C.27: Consumo energético dos UEs Rx no experimento streaming 256 B.

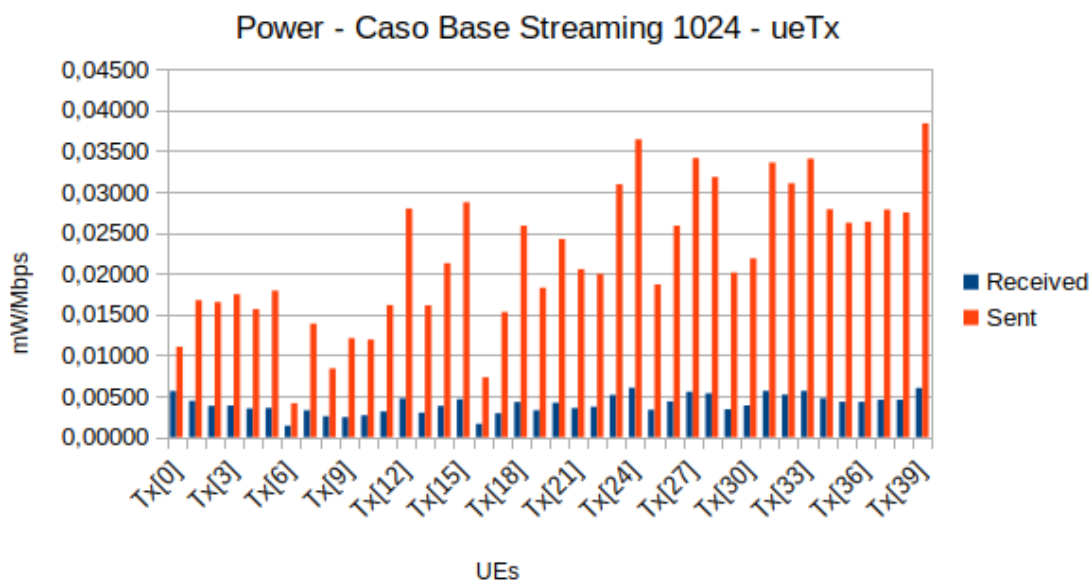


Figura C.28: Consumo energético dos UEs Tx no experimento streaming 256 B.

C.3.2 Consumo Energético dos UEs Tx no Experimento *Streaming* 256 B

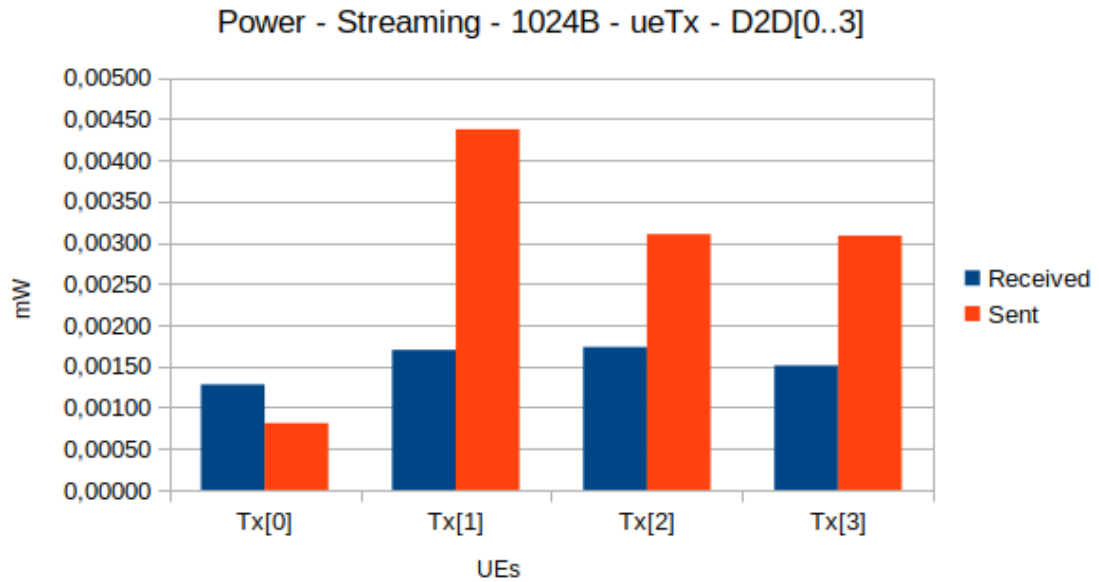


Figura C.29: Consumo energético dos UEs Tx no experimento streaming 256 B com 4 encaminhadores.

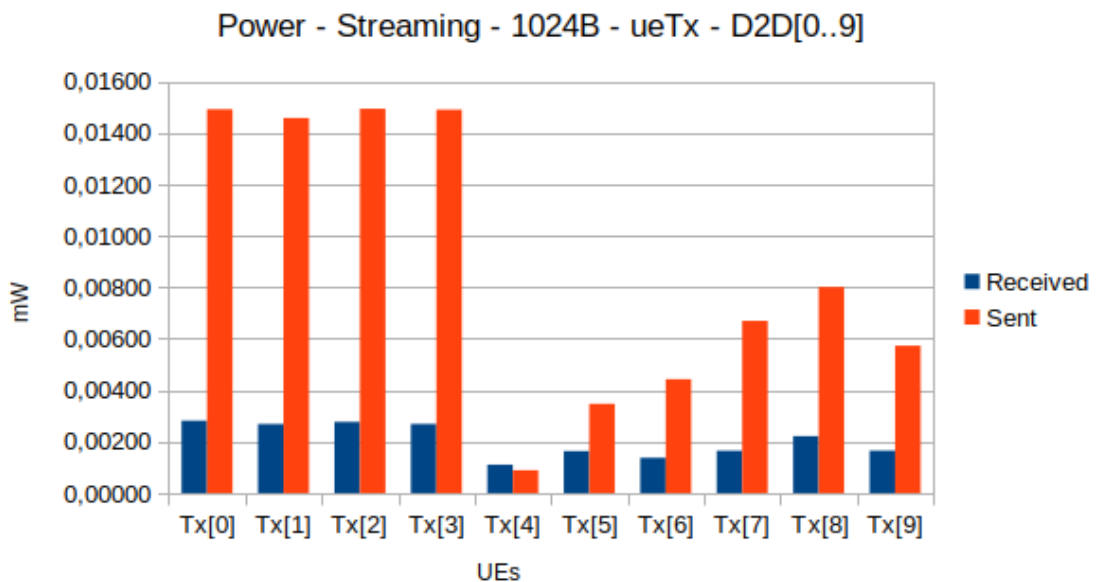


Figura C.30: Consumo energético dos UEs Tx no experimento streaming 256 B com 10 encaminhadores.

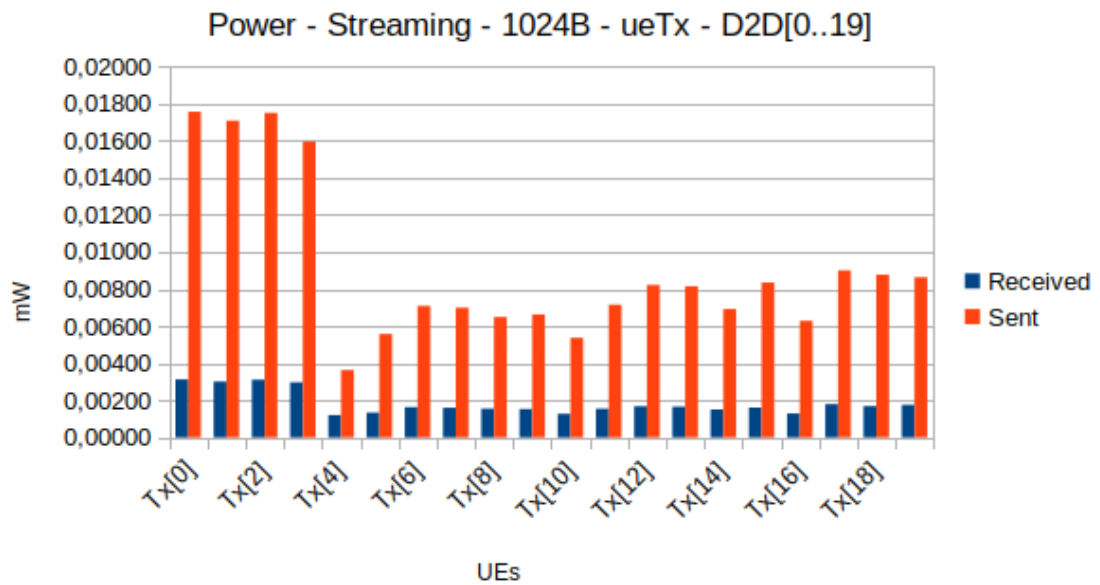


Figura C.31: Consumo energético dos UEs Tx no experimento streaming 256 B com 210 encaminhadores.

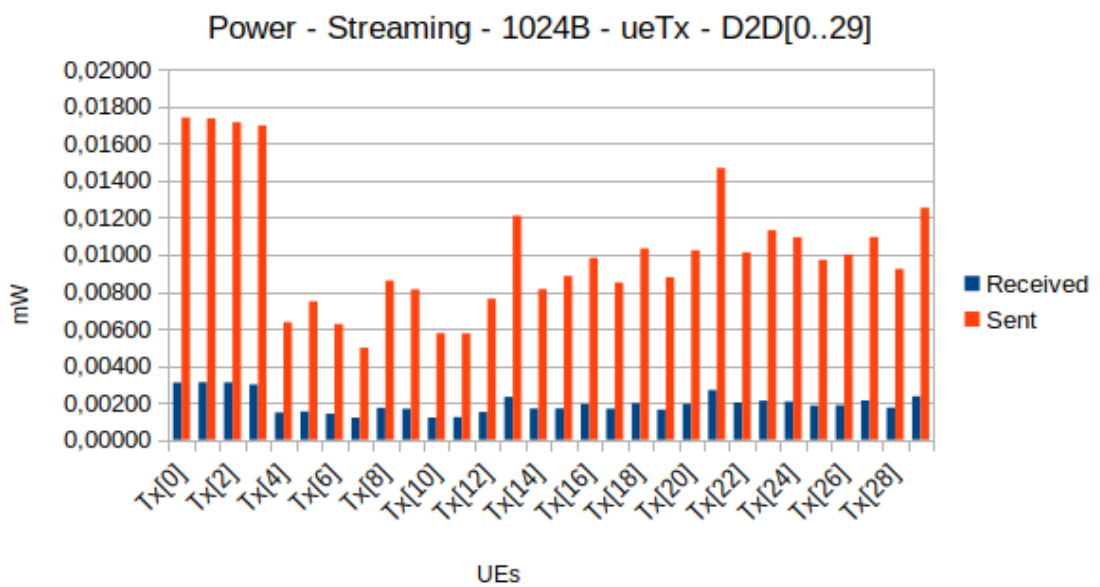


Figura C.32: Consumo energético dos UEs Tx no experimento streaming 256 B com 30 encaminhadores.

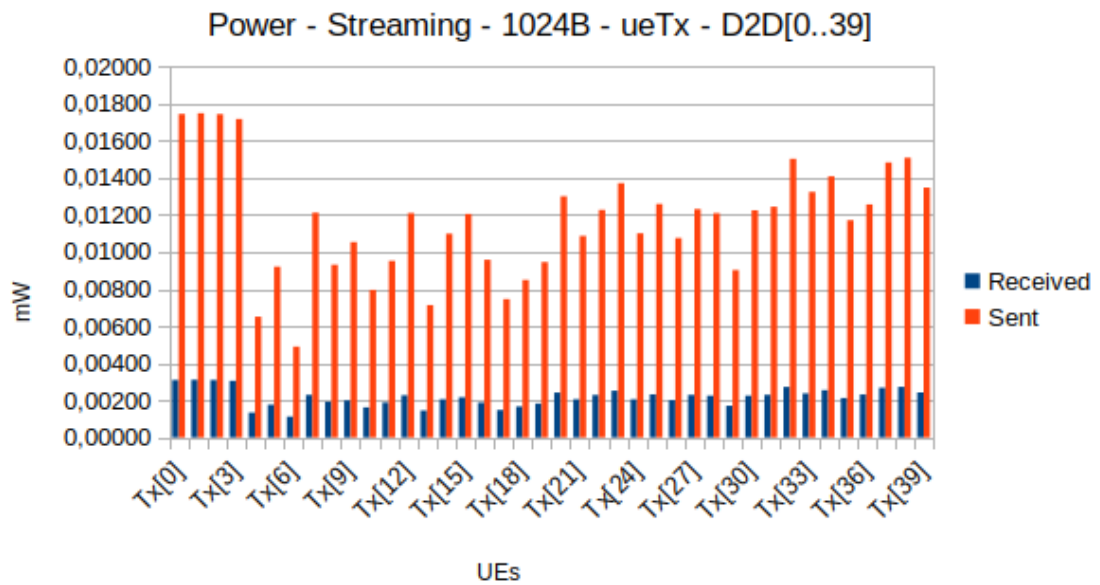


Figura C.33: Consumo energético dos UEs Tx no experimento streaming 256 B com 40 encaminhadores.

C.3.3 Consumo Energético dos UEs Rx no Experimento *Streaming* 256 B

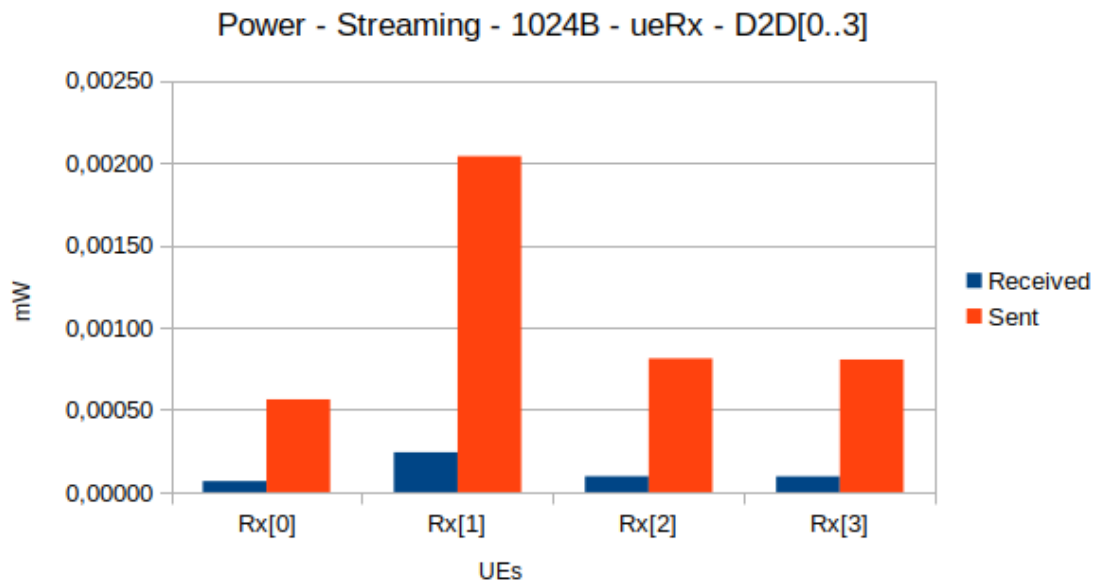


Figura C.34: Consumo energético dos UEs Rx no experimento streaming 256 B com 4 encaminhadores.

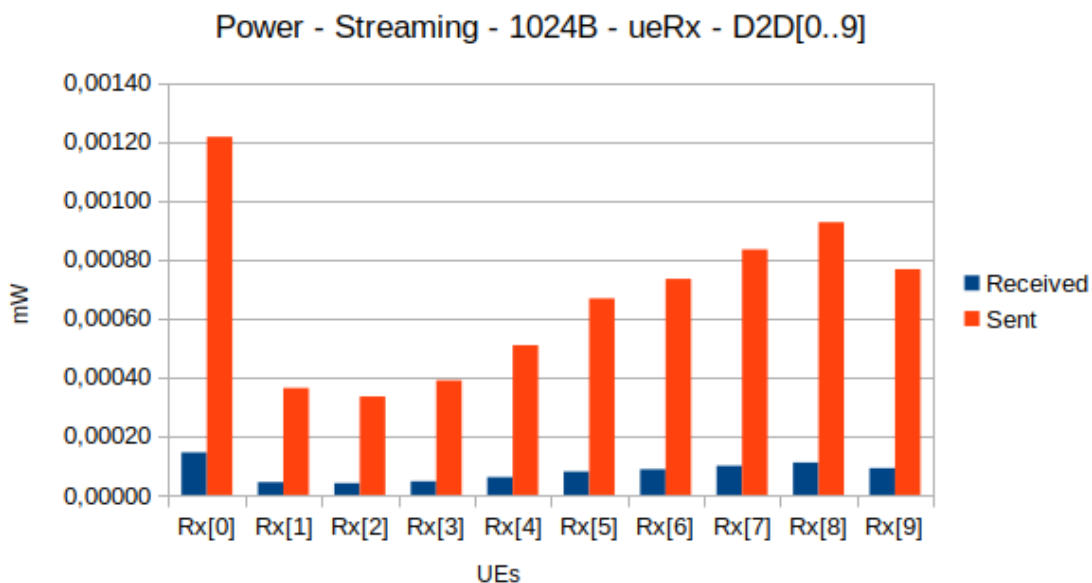


Figura C.35: Consumo energético dos UEs Rx no experimento streaming 256 B com 10 encaminhadores.

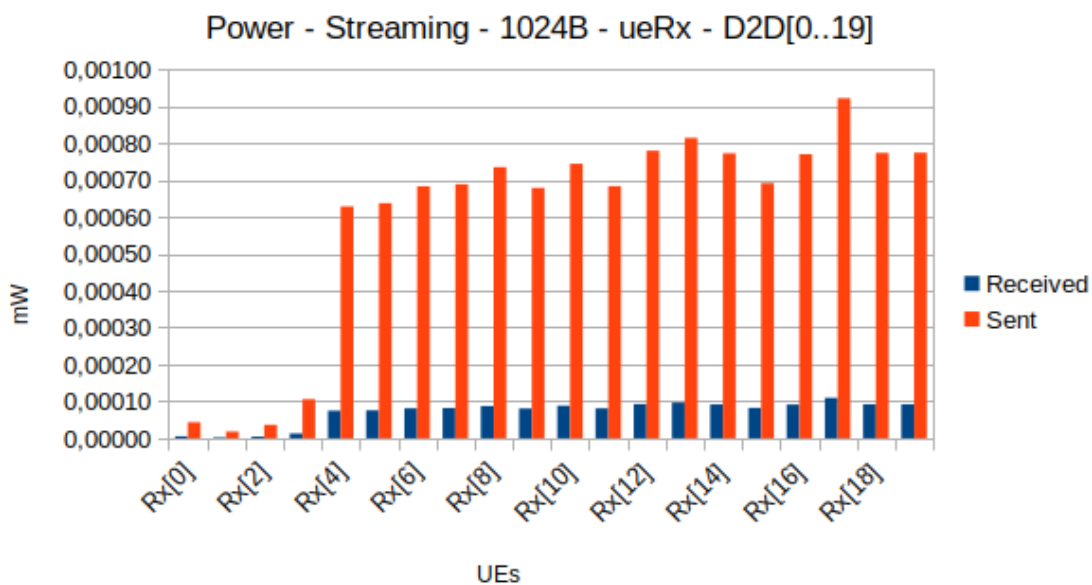


Figura C.36: Consumo energético dos UEs Rx no experimento streaming 256 B com 210 encaminhadores.

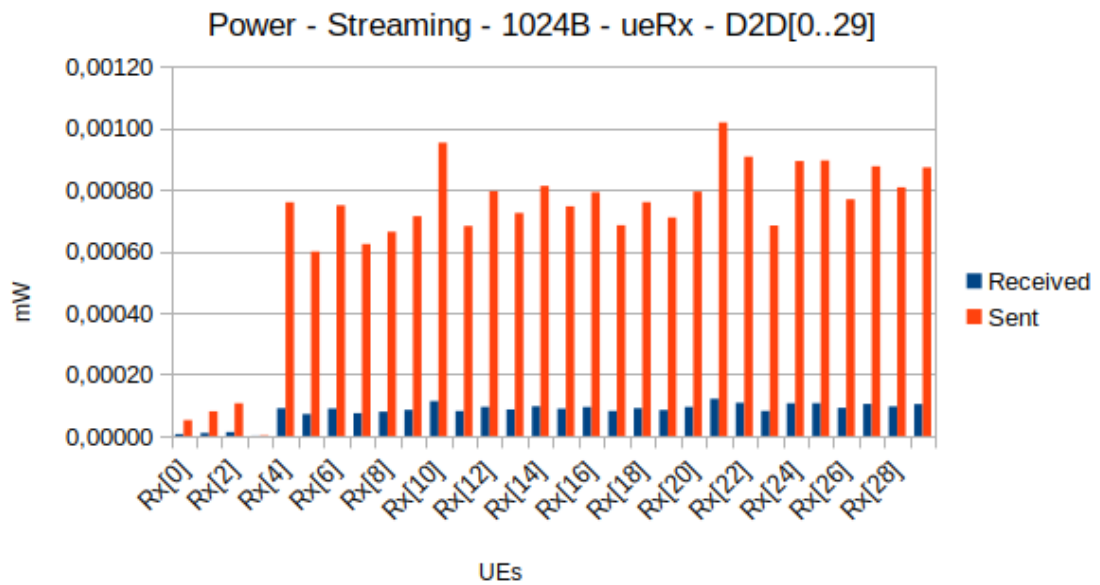


Figura C.37: Consumo energético dos UEs Rx no experimento streaming 256 B com 30 encaminhadores.

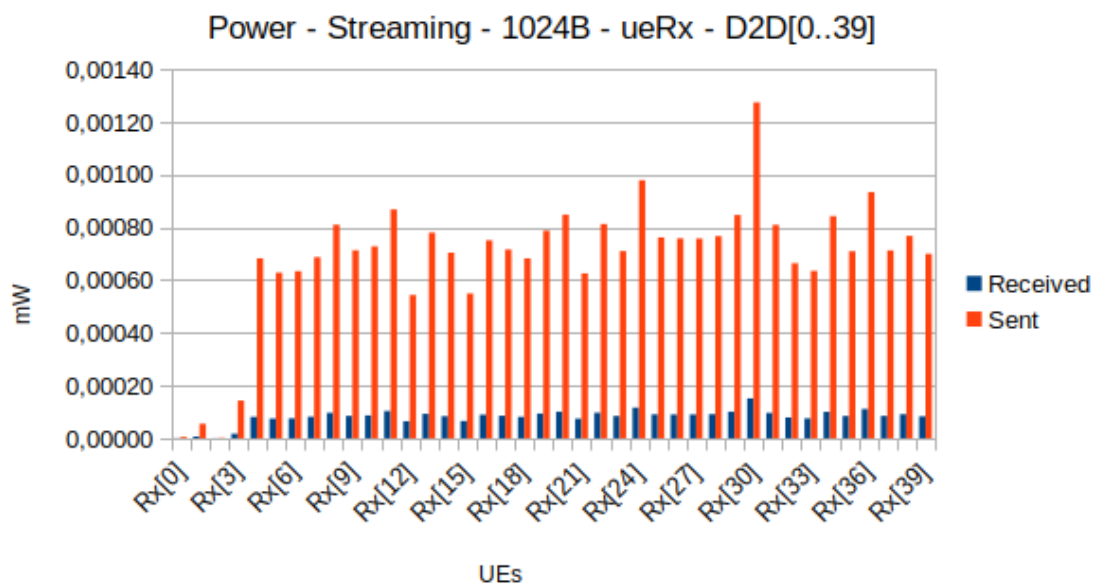


Figura C.38: Consumo energético dos UEs Rx no experimento streaming 256 B com 40 encaminhadores.

C.3.4 Consumo Energético Total da Rede no Experimento *Streaming* 256 B

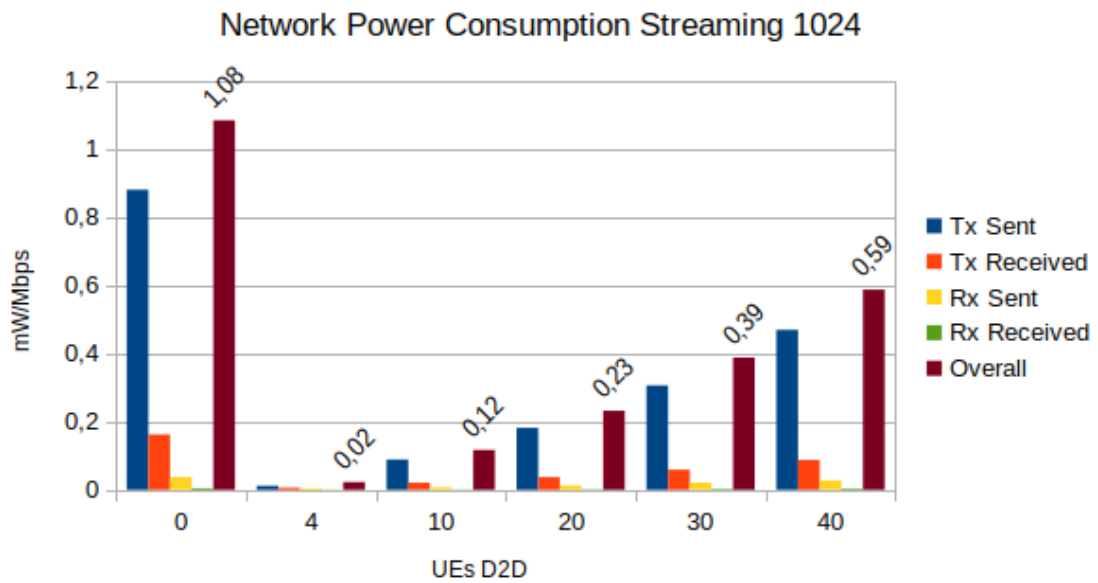


Figura C.39: Consumo energético total da rede e dos UEs Rx e Tx no experimento streaming 256 B para cada cenário.