

**UNIVERSIDADE DO ESTADO DE SANTA CATARINA - UDESC
CENTRO DE CIÊNCIAS TECNOLÓGICAS - CCT
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO**

**FERNANDA MARIA DE SOUZA
NIKOLAS JENSEN
MATIAS GIULIANO GUTIERREZ BENITEZ
VINICIUS TAKEO FRIEDRICH KUWAKI**

**ANÁLISE DA CAPTURA DO TRÁFEGO DE REDES VIA
WIRESHARK E TCPDUMP: UM COMPARATIVO DO CONSUMO DE
DADOS NA PLATAFORMA DISCORD.**

**JOINVILLE
2020**

RESUMO

O COVID-19 tornou as comunicações online essenciais, devido ao isolamento social. Esse trabalho busca analisar uma das ferramentas de comunicação online que vêm sendo utilizada na comunidade durante esse período: o Discord. Utilizando das ferramentas Wireshark e TCPDump, foi realizado uma captura de pacotes em uma sessão online na plataforma Discord. De forma a comparar o consumo entre diferentes usuários em um mesmo servidor, realizou-se uma análise do tráfego de pacotes em diferentes usuários na mesma sala virtual. O trabalho conta também com um breve histórico das ferramentas utilizadas, suas obtenções e configurações. Ao final, os dados capturados foram analisados e algumas conclusões com relação a eles são mostradas. Uma comparação com um trabalho a respeito do Skype também foi realizada.

Palavras-chaves: Wireshark, TCPDump, Discord, Análise do Tráfego de Rede.

ABSTRACT

Due to COVID-19, online human communications become essential. This work seeks to analyze one of the most used tools by the community in this period: Discord. Using Wireshark and TCPDump, a packet capture was performed during an online session on Discord. An analysis of the packet traffic in different devices connected to the same virtual room was conducted, in order to compare the behaviour of the packets in different users. Also, a brief history of the tools was made, as well as your settings and gettings. In addition, some conclusion towards the capture packets were presented. Finally, a comparison between Discord and a related work using Skype has been done.

Key-words: Wireshark, TCPDump, Discord, Network Traffic Analysis.

LISTA DE ILUSTRAÇÕES

Figura 1 – Instalação Discord Windows - 1. Fonte: Próprios Autores.	11
Figura 2 – Instalação Discord Windows - 2. Fonte: Próprios Autores.	12
Figura 3 – Instalação Discord Linux - 1. Fonte: Próprios Autores.	12
Figura 4 – Instalação Discord Linux - 2. Fonte: Próprios Autores.	12
Figura 5 – Instalação Discord macOS. Fonte: Próprios Autores.	13
Figura 6 – Instalação Skype Windows - 1. Fonte: Próprios Autores.	14
Figura 7 – Instalação Skype Windows - 2. Fonte: Próprios Autores.	15
Figura 8 – Instalação Skype no Linux. Fonte: Próprios Autores.	15
Figura 9 – Instalação Skype no macOS. Fonte: Próprios Autores.	16
Figura 10 – Estrutura de um Packet Sniffer. Fonte: Adaptado de (OLUDELE et al., 2015).	17
Figura 11 – Página Principal do Wireshark	18
Figura 12 – Gerald Combs, criador do Wireshark. Fonte: SoldierX.	19
Figura 13 – Terminal - primeiro comando. Fonte: Próprios Autores.	21
Figura 14 – Terminal - segundo comando. Fonte: Próprios Autores.	21
Figura 15 – Terminal - terceiro comando. Fonte: Próprios Autores.	22
Figura 16 – Terminal - quarto comando. Fonte: Próprios Autores.	22
Figura 17 – Configuração máquina virtual. Fonte: Próprios Autores.	23
Figura 18 – Configuração máquina virtual. Fonte: Próprios Autores.	23
Figura 19 – Configuração máquina virtual. Fonte: Próprios Autores.	23
Figura 20 – Seleção das ferramentas a serem instaladas. Fonte: Próprios Autores	24
Figura 21 – Senha sendo requisitada ao executar o Wireshark como superusuário. Fonte: Próprios Autores.	25
Figura 22 – Instalador do Wireshark no MacOS. Fonte: Próprios autores.	26
Figura 23 – Execução do passo 1. Nela é possível notar que o TCPDump não estava instalado: "comando não encontrado". Fonte: Próprios Autores.	28
Figura 24 – Exceção dos comandos 3 e 5 da esquerda para a direita. Fonte: Próprios Autores.	29
Figura 25 – Possível ubicações de filtros no <i>Wireshark</i> . Fonte: Próprios Autores.	31
Figura 26 – Na figura a cima podemos encontrar um exemplo de filtro com sintaxe errada e na figura a embaixo um exemplo com a estrutura correta de filtragem. Fonte: Próprios Autores.	32
Figura 27 – Caminho para começar captura de pacotes no Wireshark. Fonte: Próprios Autores.	33
Figura 28 – Monitoramento de captura de protocolos. Fonte: Próprios Autores.	33

Figura 29 – Na figura a esquerda está representada a execução do comando sudo su, após digitada a senha, a figura da direita apresenta o ter- minal no modo super usuário. Fonte: Próprios Autores.	35
Figura 30 – Capturando pacotes com filtros. Fonte: Próprios Autores.	38
Figura 31 – Abrindo um arquivo .pcap utilizando o TCPDump. Fonte: Próprios Autores.	38
Figura 32 – Mudanças no tráfego na Internet analisadas pela Cloudflare. Fonte: (GRAHAM-CUMMING, 2020)	42
Figura 33 – Arquitetura do Experimento. Adaptado de: (KASSIM et al., 2017) . . .	44
Figura 34 – Fluxograma do experimento. Fonte: Adaptado de: (KASSIM et al., 2017)	45
Figura 35 – Gráfico I/O no <i>Wireshark</i> . Fonte: (KASSIM et al., 2017).	46
Figura 36 – Impactos na qualidade da internet no Brasil durante a pandemia de COVID-19, análise realizada entre 31/01/2020 e 19/07/2020. Fonte: (BR, 2020).	47
Figura 37 – Gráfico de Barras para o tamanho dos pacotes.	49
Figura 38 – Gráfico de Barras para o tamanho dos pacotes.	49
Figura 39 – Gráficos RTT - Da esquerda para direita capturas de 5 e 30 minutos lado a lado. De cima para baixo linha a linha, dispositivos 1, 2 e 3. Fonte: Próprios autores.	51
Figura 40 – Gráfico I/O - Dispositivo 1 para capturas de 5 minutos. Fonte: Pró- prios Autores.	53
Figura 41 – Gráfico I/O - Dispositivo 1 para capturas de 30 minutos. Fonte: Pró- prios Autores.	54
Figura 42 – Gráfico I/O - Dispositivo 2 para capturas de 5 minutos. Fonte: Pró- prios Autores.	55
Figura 43 – Gráfico I/O - Dispositivo 2 para capturas de 30 minutos. Fonte: Pró- prios Autores.	56
Figura 44 – Gráfico I/O - Dispositivo 3 para capturas de 5 minutos. Fonte: Pró- prios Autores.	57
Figura 45 – Gráfico I/O - Dispositivo 3 para capturas de 30 minutos. Fonte: Pró- prios Autores.	58
Figura 46 – Gráficos Throughtput - Da esquerda para direita capturas de 5 e 30 minutos lado a lado. De cima para baixo linha a linha, dispositivos 1, 2 e 3. Fonte: Próprios autores.	59
Figura 47 – Fonte: Próprios autores.	59

LISTA DE TABELAS

Tabela 1 – Características sobre diferentes Packet Sniffers. Fonte: Próprios Autores.	18
Tabela 2 – Tipos de análise de rede suportados em várias plataformas. Fonte: Adaptado de (WIRESHARK, 2020a).	20
Tabela 3 – Tabela com as especificações dos dispositivos utilizados pelos autores. Fonte: Próprios Autores.	34
Tabela 4 – Largura de Banda Total para chamadas de voz e vídeo no Skype. Fonte: Adaptado de: (KASSIM et al., 2017).	46
Tabela 5 – Quantidade de pacotes perdidos durante chamadas de voz e vídeo no Skype. Fonte: Adaptado de: (KASSIM et al., 2017).	46
Tabela 6 – Tráfego em RTT. Fonte: (KASSIM et al., 2017).	47
Tabela 7 – Bytes in Flight. Fonte: Próprios Autores.	48
Tabela 8 – Retransmissão. Fonte: Próprios Autores.	49
Tabela 9 – Perda de Pacotes. Fonte: Próprios Autores.	49

LISTA DE ABREVIATURAS E SIGLAS

OV	Organização Virtual
CC	<i>Cloud Computing</i>
OC	Oportunidade de Colaboração
AHP	<i>Analytic Hierarchy Process</i>
DEA	<i>Data Envelopment Analysis</i>

SUMÁRIO

1	INTRODUÇÃO	10
2	DISCORD	11
2.1	HISTÓRICO	11
2.2	INSTALAÇÃO	11
2.2.1	Windows	11
2.2.2	Linux	12
2.2.3	Mac	12
3	SKYPE	14
3.1	HISTÓRICO	14
3.2	INSTALAÇÃO	14
3.2.1	Windows	14
3.2.2	Linux	15
3.2.3	Mac	15
4	MANUAL DE INSTALAÇÃO	17
4.1	WIRESHARK/TSHARK	17
4.1.1	Definição da Ferramenta	17
4.1.2	Histórico da Ferramenta	18
4.1.3	Requisitos Básicos	19
4.1.3.1	Windows	19
4.1.3.2	Linux	20
4.1.3.3	Mac OS	20
4.1.4	Obtenção de Software	20
4.1.4.1	Windows	21
4.1.4.2	Linux	21
4.1.4.3	MacOS	22
4.1.5	Configuração	23
4.1.5.1	Windows	24
4.1.5.2	Linux	25
4.1.5.3	MacOS	25
4.1.6	Q & A	26
4.2	TCPDUMP	27
4.2.1	Histórico da Ferramenta	27

4.2.2	Requisitos Básicos	28
4.2.3	Instalação e Configuração	28
4.2.4	Q & A	29
5	MANUAL DE UTILIZAÇÃO	30
5.1	WIRESHARK	30
5.1.1	Parâmetros configuráveis	30
5.1.2	Inicialização	32
5.1.3	Monitorando a execução	32
5.1.4	Coletando os resultados	34
5.1.5	Q&A	34
5.2	TCPDUMP	35
5.2.1	Parâmetros configuráveis	35
5.2.1.1	Parâmetros de Filtros	35
5.2.1.2	Parâmetros de Saída	37
5.2.2	Inicializando a captura	37
5.2.3	Monitorando a execução	38
5.2.4	Coletando os resultados	38
5.2.5	Interpretando e analisando os resultados	39
5.2.6	Q&A	39
6	ANALISANDO AS CAPTURAS	40
6.1	Interpretação da Saída	40
6.2	Q&A	41
7	CASOS DE ESTUDO	42
7.1	REFERENCIADO	43
7.1.1	Objetivo	43
7.1.2	Coleta de Dados	43
7.1.3	Análise	44
7.1.4	Conclusões	46
7.2	COMPARATIVO DO TRÁFEGO DE DADOS	47
7.2.1	Objetivo	48
7.2.2	Coleta de Dados	48
7.2.3	Análise	48
7.2.4	Comparação entre capturas do Wireshark e o TCPDump	51
7.2.5	Conclusões	52
8	CONSIDERAÇÕES FINAIS	60

1 INTRODUÇÃO

O COVID-19 trouxe ao mundo uma nova forma de interagir socialmente. A forma de comunicar-se e muitos dos hábitos aos quais estávamos acostumados, precisaram ser repensados. Com grande partes dos países restringindo suas populações a períodos de isolamento social, foi necessário que as atividades humanas, pelo menos muitas delas, precisassem ser realizadas virtualmente. Surge então a necessidade de se utilizar ferramentas que proporcionem formas eficazes de comunicação. Nesse sentido, esse trabalho se propõe a analisar o uso da ferramenta Discord como meio de comunicação, realizando um comparativo com relação a quantidade de pacotes em um período de tempo transmitidos entre os participantes de um servidor da plataforma.

O trabalho está dividido nas seguintes seções: as seções dois, três e quatro apresentam o histórico e formas de obtenção das ferramentas utilizadas no comparativo e da ferramenta usada em um caso comentado retirado da literatura. São elas: Discord, Skype, Wireshark e TCPDump. Nessas seções são apresentadas algumas definições, requisitos e como realizar a instalação nas principais plataformas: Windows, Linux e Mac OS. Já as seções cinco e seis, abordam a forma de utilização e interpretação, respectivamente, das ferramentas de captura de pacotes: Wireshark e TCPDump. O trabalho não focou na forma de uso das ferramentas Discord e Skype, visto que objetivava a captura e análise dos dados e não o uso da ferramenta de comunicação em si. Em seguida, a seção sete discute os casos de estudo: um caso de estudo do Skype encontrado na literatura e em seguida o comparativo da plataforma Discord, realizado pelos autores. Por fim a seção oito apresenta as considerações finais obtidas no trabalho.

2 DISCORD

Como mencionado em seções anteriores, o COVID-19 gerou transformações na sociedade. Transformações estas que o Discord conseguiu reverter em partes. Nada substitui as reuniões cara-a-cara de grupos sociais. Porém em tempos de pandemia, o Discord tem se mostrado um grande aliado nas rodas de amigos, sejam estas para atividades de lazer ou até mesmo para atividades mais formais, como reuniões ou aulas online.

2.1 HISTÓRICO

A ferramenta surgiu da paixão dos criadores Jason Citron e Stan Vishnevskiy por jogos online. Desde pequeno os criadores buscavam uma ferramenta eficaz para a comunicação durante partidas online, mas a época, nenhuma estava disponível. Logo, em 2015 a ferramenta foi criada e disponibilizada ao mundo. Facilitando conversas sejam elas por vídeo, voz ou texto. De acordo com o site oficial, atualmente a plataforma conta com mais de 100 milhões de usuários ativos por mês, conversando por mais de 4 horas diárias.

2.2 INSTALAÇÃO

Nas próximas subseções serão abordadas as formas de instalação nos principais sistemas operacionais.

2.2.1 Windows

A instalação no Windows é simplificada, visto que possui interface gráfica. Primeiramente deve-se entrar em <<https://discord.com/download>>. Seguindo os passos descritos pelas Figuras 1 2, o Discord será instalado com sucesso.



Figura 1 – Instalação Discord Windows - 1. Fonte: Prórios Autores.

Após baixado, apenas basta executar o .exe.

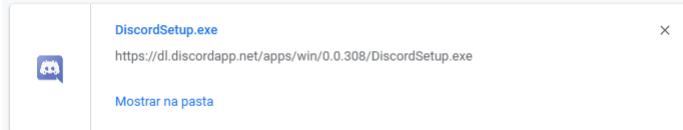


Figura 2 – Instalação Discord Windows - 2. Fonte: Próprios Autores.

2.2.2 Linux

Para instalar o Discord no Linux basta apenas baixá-lo no Gerenciador de Aplicativos conforme Figuras 3 e 4.

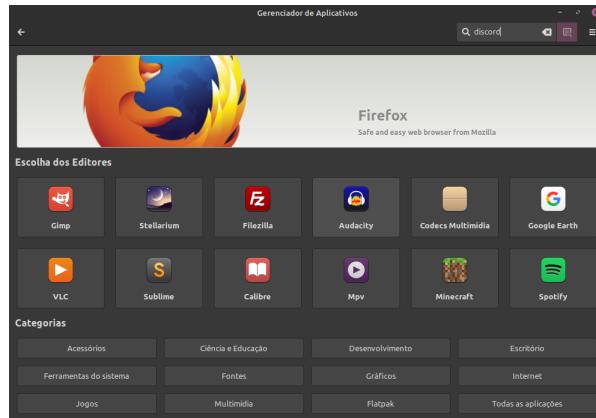


Figura 3 – Instalação Discord Linux - 1. Fonte: Próprios Autores.

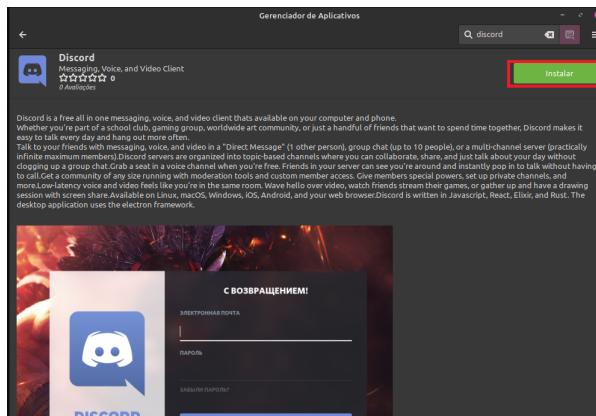


Figura 4 – Instalação Discord Linux - 2. Fonte: Próprios Autores.

2.2.3 Mac

A instalação de *Discord* no macOS é bem similar a do *Windows*, basta com entrar no site <<https://discord.com/download>> e apertar o botão “Baixar para Mac” como mostra na figura 5. Logo depois de apertar o botão, começará a instalação do aplicativo.



Figura 5 – Instalação Discord macOS. Fonte: Próprios Autores.

3 SKYPE

VoIP é um serviço de telefonia global que conecta usuários, onde um usuário A realiza uma chamada a um usuário B, todos os dispositivos registrados na conta B começam a tocar e enviar informações ao usuário A com relação ao endereço de IP e os serviços disponíveis a partir de B (NICOLETTI; BERNASCHI, 2019). O Skype, assim como o Discord discutido na seção anterior, é uma dentre várias ferramentas que utilizam esse serviço para realizar comunicações entre usuários.

3.1 HISTÓRICO

Lançado em 2003 e criado por Janus Friis e Niklas Zennstrom, é um software de comunicação tal como o Discord. A seguir serão apresentadas as formas de instalação nos principais sistemas operacionais.

3.2 INSTALAÇÃO

Para cada sistema operacional, serão discutido a seguir as formas de obtenção. Basta acessar o link: <<https://www.skype.com/pt-br/get-skype/>>. Nas subseções a seguir irão mostrar os passos em diferentes sistemas.

3.2.1 Windows

A instalação do *Skype* no *Windows* é feita pelo link apresentado anteriormente e demonstrada pelas Figuras 6 e 7.

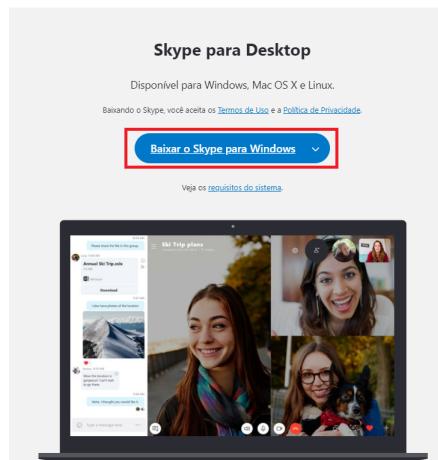


Figura 6 – Instalação Skype Windows - 1. Fonte: Próprios Autores.

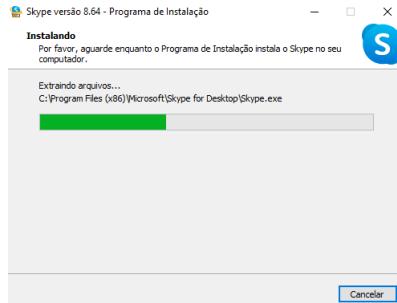


Figura 7 – Instalação Skype Windows - 2. Fonte: Próprios Autores.

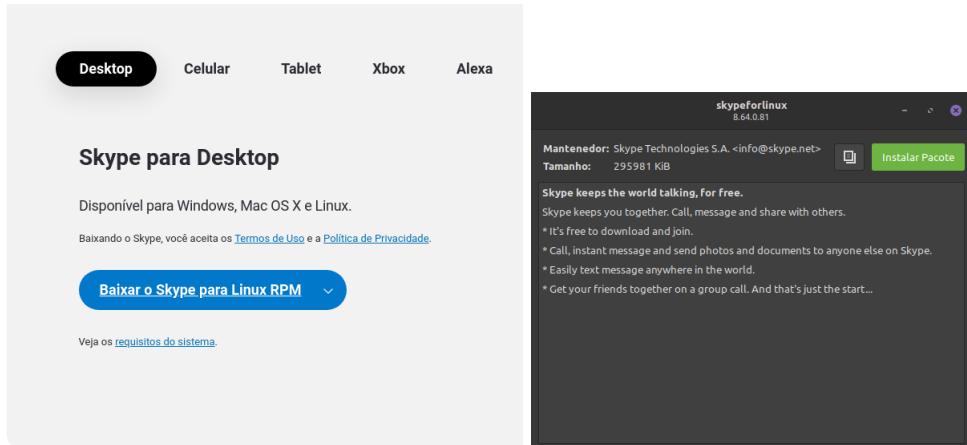


Figura 8 – Instalação Skype no Linux. Fonte: Próprios Autores.

3.2.2 Linux

Acessando o link discutido anteriormente, basta realizar o download do instalador de acordo com a sua distro. Na figura foi utilizado uma distro Linux Mint, logo optou-se pelo arquivo .deb. Na figura 8 é possível ver a opção de instalação, e o instalador .deb aberto para a instalação do pacote.

3.2.3 Mac

Para instalar o aplicativo Skype no macOS é necessário acessar o link anteriormente apresentado, depois apertar o botão “Get skype for Mac” como ilustrado na figura 9. Logo após isso começará a instalação.

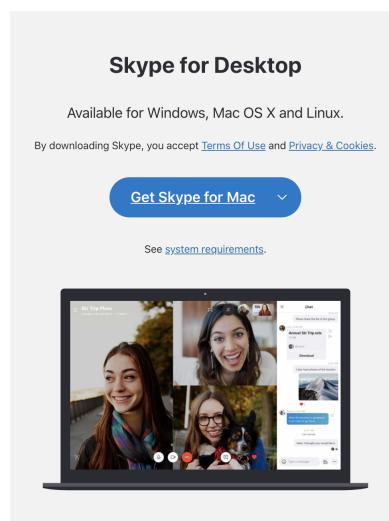


Figura 9 – Instalação Skype no macOS. Fonte: Próprios Autores.

4 MANUAL DE INSTALAÇÃO

Packet sniffers são programas executados em dispositivos conectados a uma rede, que recebem passivamente todos os quadros transmitidos a camada de enlace através do adaptador de rede (ASRODIA; PATEL,), conforme Figura 10. A seguir serão discutidas formas de obtenção, configuração e uso de dois dos *packet sniffers* mais comuns: *Wireshark/TShark* e *TCPDump*.

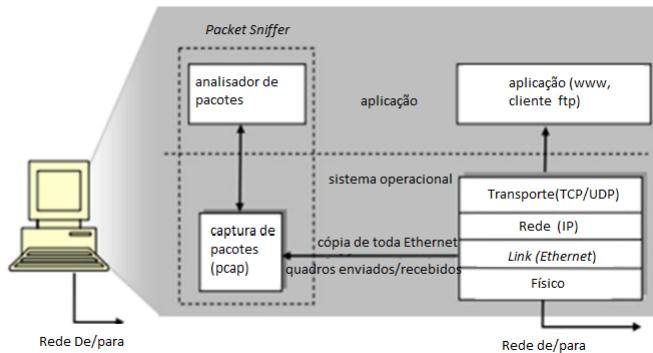


Figura 10 – Estrutura de um Packet Sniffer. Fonte: Adaptado de (OLUDELE et al., 2015).

4.1 WIRESHARK/TSHARK

Wireshark, assim como mencionado em outras seções, é um *packet sniffer*. Nas subseções seguintes serão discutidas a fundo o que é a ferramenta, seu histórico, requisitos e como obtê-la.

4.1.1 Definição da Ferramenta

Wireshark se trata de um analisador de tráfego de redes multiplataforma, com software de código aberto gratuito disponível em <wireshark.org>. Por se tratar de uma ferramenta de rede passiva, não transmite *bytes* para a rede, nem é o destinatário de *bytes* que outros computadores enviaram. O software atua captando e decodificando os quadros que estão sendo transmitidos em uma rede, apresentando detalhadamente cada pacote de dados capturado, organizando-os por tipo de protocolo. Por isso, é uma ferramenta ideal para solução de problemas de rede, otimização, segurança (análise forense da rede) e análise de aplicações (CHAPPELL; COMBS, 2013).

Na Figura 11, tem-se a página principal do Wireshark após a captura de pacotes da placa de rede *Wifi*.

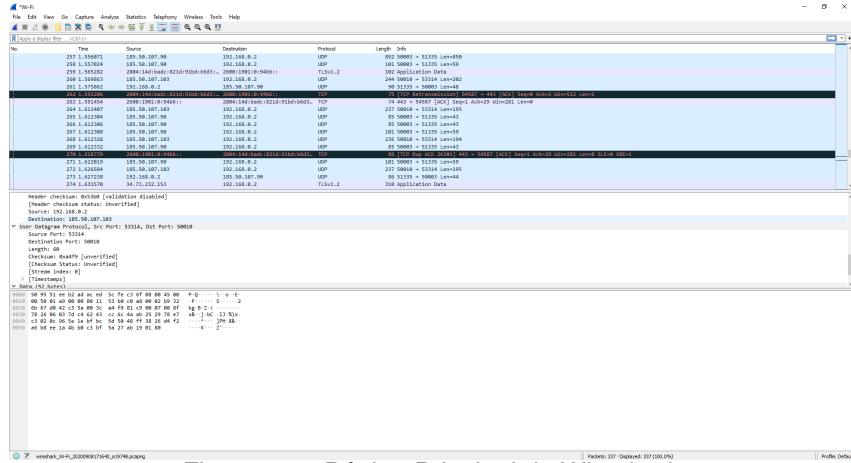


Figura 11 – Página Principal do Wireshark

Alguns benefícios do *Wireshark* que o tornam diferentes de outros analisadores de tráfego, são:

- Suporte a diversos protocolos: O software traz suporte a mais de 1000 protocolos, principalmente por ser desenvolvido no modelo de software livre, com suporte a novos protocolos adicionados com o lançamento de novas atualizações.
- Facilidade de Uso: A interface do *Wireshark* é facilmente entendível por possuir interface GUI, em comparação com qualquer ferramente de detecção de pacotes como o *TCPDump*, que é baseado em linhas de comando.
- Suporte a todos os sistemas operacionais: A ferramenta suporta todos os sistemas operacionais modernos, como Windows, *Linux-based*, e OS X.
- Custo: Como é uma ferramenta *open-source* e lançada sobre a licença GNU, *Wireshark* é totalmente gratuito tanto para âmbitos comerciais ou pessoais.

Outras características sobre diferenças entre Packets Sniffers estão presentes na Tabela 1.

Packet Sniffers	Custo	Espaço Necessário	Compatibilidade	Comentários
Wireshark	De graça	15 MB	Windows, Linux, Solaris	Facilidade na captura e filtragem Interpreta, decodifica e apresenta centenas de protocolos
TCPDump	De graça	1 MB	Windows e Linux	Não é completamente compatível com alguns pacotes IPv6 Precisa de outras ferramentas para analisar dados
Snoop	De graça	Vem com SO	Solaris	Mesmas funcionalidades do TCPdump
Kismet	De graça	1 MB	Mac, Linux, Solaris	Baixa funcionalidade de importar e exportar informações Alta flexibilidade
Etherapeek	895\$/ano	10 GB e 512 RAM	Windows	Regras flexíveis Interface GUI amigável e sistema de alertas.

Tabela 1 – Características sobre diferentes Packet Sniffers. Fonte: Prórios Autores.

4.1.2 Histórico da Ferramenta

A história do mais popular analisador de redes do mundo(LYON, 2019) começou em 1997, quando softwares do mesmo tipo possuíam preços entre \$5,000 a

\$20,000 dólares (CHAPPELL, 2012). Gerald Combs, Figura 12, foi um dos cientistas que sentiu as restrições orçamentárias das caras ferramentas comerciais, e com isso, decidiu criar o seu próprio programa de análise de redes. Originalmente, na primeira versão do analisador lançada em 14 de Julho de 1998, o nome adotado foi *Ethereal* (versão 0.2.0).

Os primeiros pesquisadores que contribuiram a pesquisa de Combs foram Gilbert Ramirez, Guy Harris e Richard Sharpe. Quando Gerald começou a trabalhar com a *CACE Technologies* em Junho de 2006, questões de propriedade de marca registrada do nome *Ethereal* forçaram a mudança de nome, para então ser batizado como *Wireshark*. Em 2008, foi lançada a versão 1.0 do *Wireshark*, sendo esta versão a primeira considerada completa, com os recursos mínimos implementados. Após, em 2015, a liberação do *Wireshark* 2.0 ao público trouxe uma nova *interface* ao usuário.

Atualmente, o *Wireshark* é mantido por uma comunidade ativa de desenvolvedores por todo o mundo, possuindo mais de 600 autores descritos em seu site oficial(WIRESHARK, 2020b).



Figura 12 – Gerald Combs, criador do Wireshark. Fonte: SoldierX.

4.1.3 Requisitos Básicos

A quantidade de recursos que o *Wireshark* necessita depende do ambiente e do tamanho do arquivo de captura analisado. Os requisitos básicos a seguir devem ser adequados para arquivos de captura com tamanho médio de não mais do que algumas centenas de MB (WIRESHARK, 2020a). Arquivos de captura maiores exigirão mais memória e espaço de disco

4.1.3.1 Windows

O *Wireshark* deve oferecer suporte a qualquer versão do *Windows* que ainda esteja dentro de seu tempo de vida de suporte estendido (WIRESHARK, 2020a). Os requisitos básicos para funcionamento da ferramenta são:

- *Universal C Runtime*.

- Qualquer sistema operacional moderno com processador de 64-bit AMD64/x86-64 ou 32-bit x86.
- 500 MB de memória RAM. Arquivos de captura maiores requerem mais RAM.
- 500 MB de espaço disponível em disco. Os arquivos de captura requerem espaço em disco adicional.
- Qualquer exibição moderna. Recomenda-se uma resolução de 1280 x 1024 ou superior. O *Wireshark* fará uso de resoluções HiDPI ou Retina.
- Tipos de rede suportados em várias plataformas, conforme Tabela 2:

	AIX	FreeBSD	HP-UX	Irix	Linux	macOS	NetBSD	OpenBSD	Solaris	Tru64 UNIX	Windows
Interfaces físicas											
ATM	desconhecido	desconhecido	desconhecido	desconhecido	sim	não	desconhecido	desconhecido	sim	desconhecido	desconhecido
Bluetooth	não	não	não	não	sim	não	não	não	não	não	não
CiscoHDLC	desconhecido	sim	desconhecido	desconhecido	sim	desconhecido	sim	sim	desconhecido	desconhecido	desconhecido
Ethernet	sim	sim	sim	sim	sim	sim	sim	sim	sim	sim	sim
FDDI	desconhecido	desconhecido	desconhecido	desconhecido	sim	não	desconhecido	desconhecido	sim	desconhecido	desconhecido
FrameRelay	desconhecido	desconhecido	não	não	sim	não	desconhecido	desconhecido	não	não	não
IrDA	não	não	não	não	sim	não	não	não	não	não	não
PPP2	desconhecido	desconhecido	desconhecido	desconhecido	sim	sim	desconhecido	desconhecido	não	desconhecido	sim
TokenRing	sim	sim	desconhecido	não	sim	não	sim	sim	sim	desconhecido	sim
USB	não	não	não	não	sim	não	não	não	não	não	não
WLAN4	desconhecido	sim	desconhecido	desconhecido	sim	sim	sim	sim	desconhecido	desconhecido	sim
Interfaces virtuais											
Loopback	desconhecido	sim	não	desconhecido	sim	sim	sim	sim	não	sim	não
VLAN Tags	sim	sim	sim	desconhecido	sim	sim	sim	sim	sim	sim	sim

Tabela 2 – Tipos de análise de rede suportados em várias plataformas. Fonte: Adaptado de (WIRESHARK, 2020a).

4.1.3.2 Linux

O *Wireshark* funciona na maioria dos sistemas operacionais *UNIX* e plataformas *UNIX-like* incluindo Linux e variantes do BSD (*Berkeley Software Distribution*). Os requisitos do sistema são os mesmos das especificações do *Windows*.

4.1.3.3 Mac OS

O *Wireshark* é suportado apenas para versões superiores ao Mac OS 10.12. As versões do MacOS compatíveis dependem de bibliotecas de terceiros e dos requisitos da *Apple*.

4.1.4 Obtenção de Software

O *Wireshark* está disponível para as plataformas Windows, Linux e MacOS. Nas subseções que se seguem, serão abordados em detalhes como obter o software em cada uma delas.

4.1.4.1 Windows

Para *Windows* a instalação possui interface para o usuário, que pode ser encontrado no endereço <<https://www.wireshark.org/download.html>>. Neste podem ser encontrados instaladores para computadores 32-bits e 64-bits, ou até para aplicações portáteis *Windows*, i.e., *Windows PortableApps* 32-bits. Os requisitos mínimos do *Wireshark* para *Windows* incluem a biblioteca universal da linguagem C, qualquer processador x86, no mínimo 500 *megabytes* de memória RAM (um arquivo de captura maior necessita de mais memória) e 500 *megabytes* livres em disco. Além disso, ainda precisa do *Npcap* instalado no computador para realizar a captura dos pacotes.

4.1.4.2 Linux

A instalação em distribuições Linux é feita por meio de linhas de comando. Primeiramente, o usuário deve abrir o terminal por meio do atalho Ctrl + Alt + T. Após, deverá digitar os seguintes comandos:

1. sudo add-apt-repository ppa:wireshark-dev/stable

```
nanda@nanda-VirtualBox: ~
Arquivo Editar Ver Pesquisar Terminal Ajuda
nanda@nanda-VirtualBox:~$ sudo add-apt-repository ppa:wireshark-dev/stable
[sudo] senha para nanda:
Você está prestes a adicionar o seguinte PPA:
Latest stable Wireshark releases back-ported from Debian package versions.

Back-porting script is available at https://github.com/rbalint/pkg-wireshark-ubuntu-ppa

From Ubuntu 16.04 you also need to enable "universe" repository, see:
http://askubuntu.com/questions/148638/how-do-i-enable-the-universe-repository

The packaging repository for Debian and Ubuntu is at: https://salsa.debian.org/
debian/wireshark
Mais informações: https://launchpad.net/~wireshark-dev/+archive/ubuntu/stable
Pressione Enter para continuar ou Ctrl+C para cancelar
```

Figura 13 – Terminal - primeiro comando. Fonte: Próprios Autores.

2. sudo apt-get update

```
nanda@nanda-VirtualBox: ~
Arquivo Editar Ver Pesquisar Terminal Ajuda
Back-porting script is available at https://github.com/rbalint/pkg-wireshark-ubuntu-ppa

From Ubuntu 16.04 you also need to enable "universe" repository, see:
http://askubuntu.com/questions/148638/how-do-i-enable-the-universe-repository

The packaging repository for Debian and Ubuntu is at: https://salsa.debian.org/
debian/wireshark
Mais informações: https://launchpad.net/~wireshark-dev/+archive/ubuntu/stable
Pressione Enter para continuar ou Ctrl+C para cancelar

Executing: /tmp/apt-key-gpghome.1zeihlgyu/gpg.1.sh --keyserver hkp://keyserver.ubuntu.com:443 --recv-keys A2E402B85A4B70CD78D8A39D9875551314EC0F0
gpg: chave 0875551314EC0F0: chave pública "Launchpad PPA for Wireshark Developers" importada
gpg: Número total processado: 1
gpg:           importados: 1
nanda@nanda-VirtualBox:~$ sudo apt-get update
Ign:1 http://packages.linuxmint.com ulyana InRelease
Atingido:2 http://ppa.launchpad.net/git-core/ppa/ubuntu focal InRelease
Atingido:3 http://archive.canonical.com/ubuntu focal InRelease
Atingido:4 http://packages.linuxmint.com ulyana Release
Atingido:5 http://archive.ubuntu.com/ubuntu focal InRelease
Atingido:6 http://packages.microsoft.com/repos/vscode stable InRelease
Atingido:7 http://security.ubuntu.com/ubuntu focal-security InRelease
```

Figura 14 – Terminal - segundo comando. Fonte: Próprios Autores.

3. sudo apt-get install wireshark

```
nanda@nanda-VirtualBox: ~
Arquivo Editar Ver Pesquisar Terminal Ajuda
Obter:14 http://ppa.launchpad.net/wireshark-dev/stable/ubuntu focal/main Transla
tion-en [1.704 B]
Baixados 30.0 kB em 3s (9.103 B/s)
Lendo listas de pacotes... Pronto
nanda@nanda-VirtualBox:~$ sudo apt-get install wireshark
Lendo listas de pacotes... Pronto
Construindo árvore de dependências
Lendo informação de estado... Pronto
The following additional packages will be installed:
libqt5multimedia5 libqt5multimedia5-plugins libqt5multimediasettings5
libqt5multimediawidgets5 libqt5opengl5 libqt5printsupport5 libsmi2l0
libspandsp2 libwireshark-data libwireshark13 libwiretap10 libwsutil11
wireshark-common wireshark-qt
Pacotes sugeridos:
snmp-mibs-downloader geoupdate geoip-database-extra libjs-leaflet
libjs-leaflet.markercluster wireshark-doc
Os NOVOS pacotes a seguir serão instalados:
libqt5multimedia5 libqt5multimedia5-plugins libqt5multimediasettings5
libqt5multimediawidgets5 libqt5opengl5 libqt5printsupport5 libsmi2l0
libspandsp2 libwireshark-data libwireshark13 libwiretap10 libwsutil11
wireshark wireshark-common wireshark-qt
0 pacotes atualizados, 15 pacotes novos instalados, 0 a serem removidos e 61 não
atualizados.
É preciso baixar 22,8 MB de arquivos.
Depois desta operação, 119 MB adicionais de espaço em disco serão usados.
Você quer continuar? [S/n] [
```

Figura 15 – Terminal - terceiro comando. Fonte: Próprios Autores.

Ao final da execução o *software Wireshark* estará instalado com sucesso. O *Wireshark* deve ser acessado por meio de um super usuário, Para entrar como *root*, basta digitar o comando *sudo su* no terminal e a senha do usuário conectado. Após, para inicialização do *software*, basta digitar *wireshark* já no *root*. A Figura 16 demonstra a ação necessária.

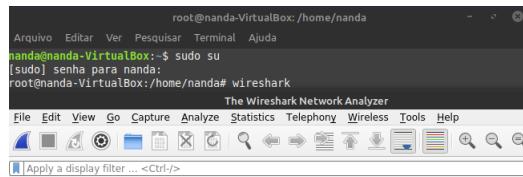


Figura 16 – Terminal - quarto comando. Fonte: Próprios Autores.

No caso de máquinas virtuais (VM) com Linux, o usuário deve configurar no seu gerenciador de máquinas virtuais para VM possuir acesso direto à rede física do *host*, inclusive obtendo um endereço IP do servidor DHCP físico, se existir. Logo, isso permitirá um contato direto da VM com a rede real por meio do modo *bridge*. Na Figura 17, com o gerenciador *Oracle VM Virtual Box*, pode-se ajustar essa configuração seguindo os passos descritos pelas Figuras 18 e 19:

4.1.4.3 MacOS

Para instalar o *Wireshark* no *MacOS* pode acessar o seguinte endereço <<https://www.wireshark.org/download.html>>. A única opção disponível para instalação é a de 64-bits com o nome "*macOS Intel 64-bit .dmg*". Após fazer clique na opção já começara o *download*.

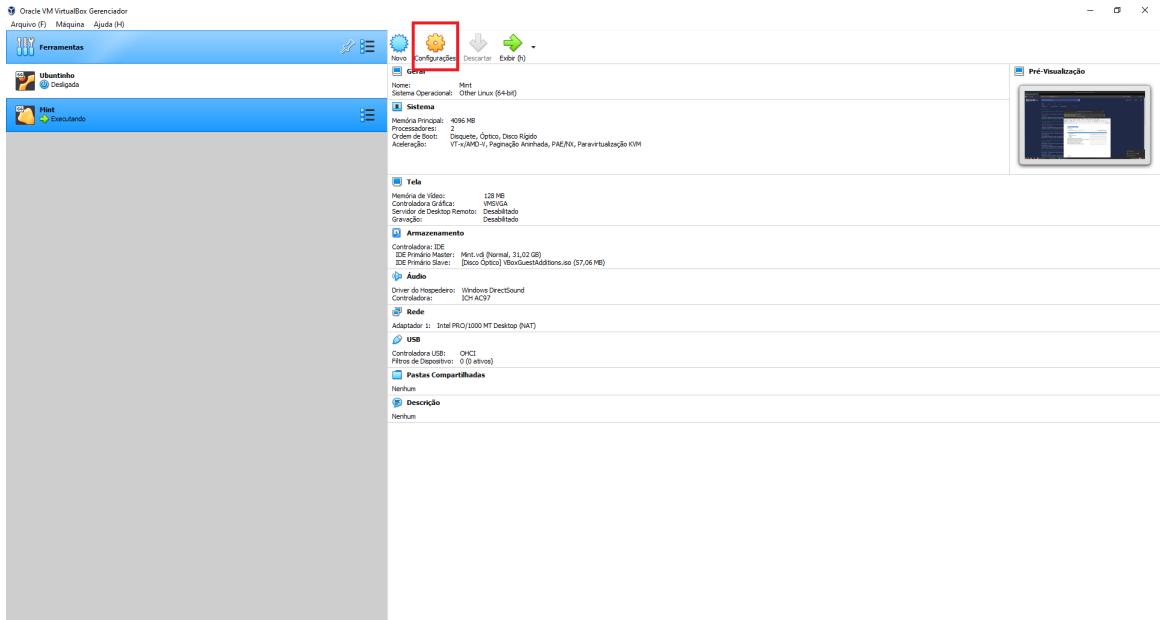


Figura 17 – Configuração máquina virtual. Fonte: Próprios Autores.

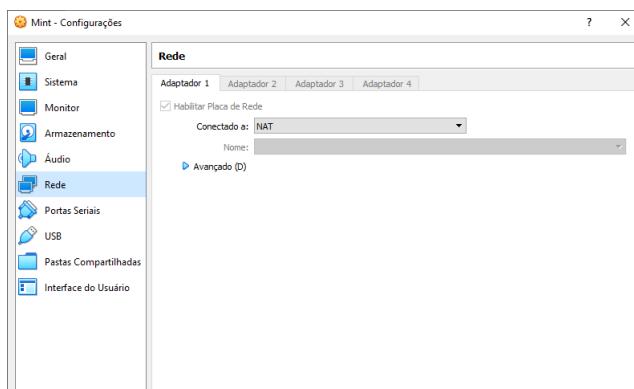


Figura 18 – Configuração máquina virtual. Fonte: Próprios Autores.

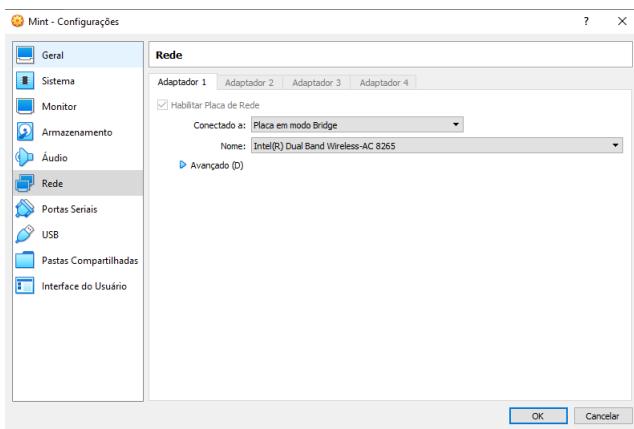


Figura 19 – Configuração máquina virtual. Fonte: Próprios Autores.

4.1.5 Configuração

Nas subseções consequentes, serão apresentadas as configurações da ferramenta, assim como feito anteriormente, o mesmo processo será repetido no sistemas operacionais Linux, Windows e MacOS.

4.1.5.1 Windows

O instalador do *Wireshark* inclui o *Npcap* na instalação, o qual é necessário para a captura de pacotes. O instalador oferece componentes a serem instalados os

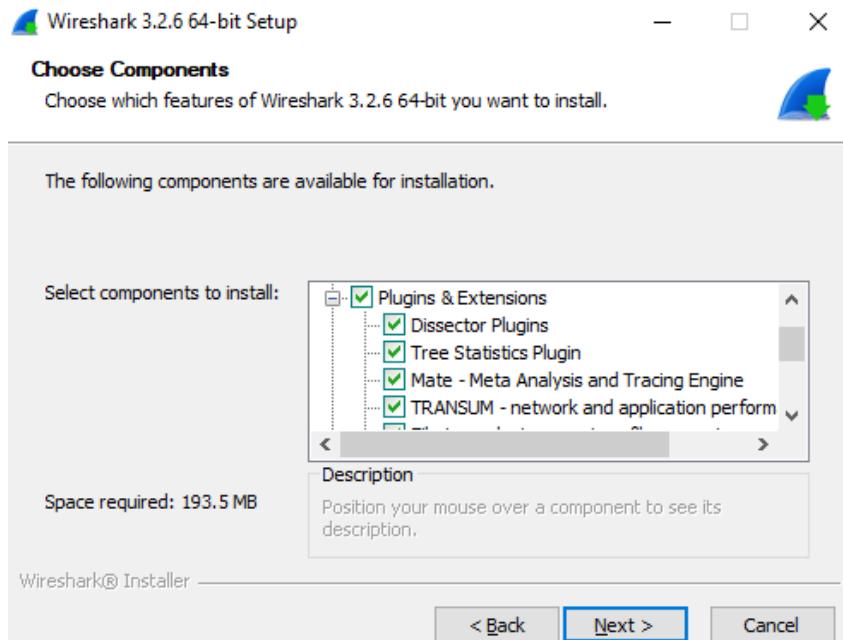


Figura 20 – Seleção das ferramentas a serem instaladas. Fonte: Próprios Autores

quais são:

- TShark: Um analisador de pacotes de rede.
- Extensões para o *Wireshark*:
 - *Plugins* de dissecação: Adicionais para dissecação.
 - *Trees statistic plugins*: Extensão para estatísticas.
 - Meta-análise e motor de restreaming (MATE): analisa os *frames* capturados e especifica como diferentes *frames* são relacionados entre si.
 - *SNMP MIBs*: O Protocolo Simples de Gerência de Redes (SNMP) é um protocolo padrão (GOMES, 2017) e pode ser descrito com uma Base de Informações de Gerenciamento (MIB), com árvore hierárquica por tipo de informação.
- Ferramentas: Adicionais para linha de comando para utilizar em arquivos de captação.
 - *Editcap*: Lê um arquivo de captura e escreve alguns ou todos os pacotes em outro arquivo.

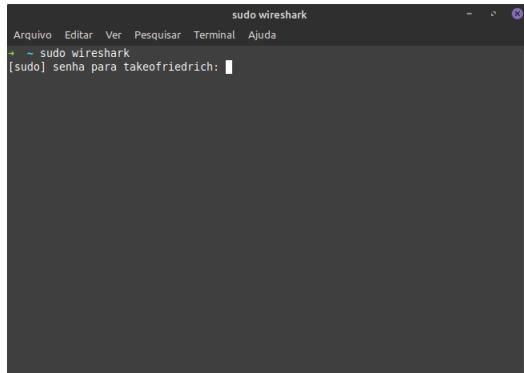


Figura 21 – Senha sendo requisitada ao executar o Wireshark como superusuário. Fonte: Próprios Autores.

- *Text2Pcap*: Lê em *ASCII hex dump* e escreve em um arquivo de captura *pcap*.
- *Reordercap*: Reordena um arquivo de captura pela data.
- *Mergecap*: Combina múltiplos arquivos de captura salvos num único arquivo de saída.
- *Capinfos*: Fornece informações em arquivos de captura.
- *Rawshark*: Filtro *raw* de capturas.
- Guia do usuário: Instalação local do guia do usuário, caso não esteja instalado o botão de "Ajuda" irá redirecionar para uma página *web*.

4.1.5.2 Linux

Ao utilizar o Wireshark no Linux, é possível utilizá-lo usando dois modos: normal e superusuário. No primeiro você tem acesso a recursos limitados, por exemplo: não é possível realizar captura de pacotes estando nesse modo. Para isso é necessário abrir a ferramenta no modo superusuário, utilizando o comando **sudo wireshark**. Na figura 21 é possível ver a execução do comando.

4.1.5.3 MacOS

Logo de instalar o *Wireshark* e abrir o instalador você vai ver a seguinte janela, como ilustrada na figura 22:

Como todo novo aplicativo instalado no *MacOS* você deve arrastar o símbolo do *Wireshark* para a pasta *Applications*. Além disso você tem a opção de instalar o pacote de nome "*ChmodBPF.pkg*". O pacote mencionado faz algumas mudanças nas permissões do sistema para que o pacote possa capturar pacotes de rede.

A outra opção disponível é adicionar o *Wireshark* ao *system path*. Isto permite que o *TShark* seja utilizado em qualquer pasta do *Finder*.

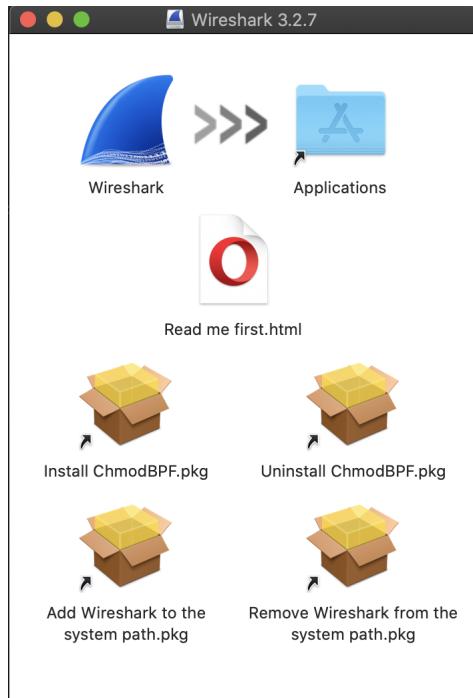


Figura 22 – Instalador do Wireshark no MacOS. Fonte: Próprios autores.

4.1.6 Q & A

Como todo *software*, a comunidade possui perguntas a serem respondidas sobre tal, algumas vezes respondido por usuários mais experientes ou até mesmo pelos desenvolvedores. Então, foram selecionadas as perguntas mais realizadas:

- Quais protocolos o *Wireshark* suporta?

Segundo (WIRESHARK...), centenas de protocolos são suportados, mas os principais: HTTP, TCP, UDP, SMTP e DNS.

- Quais dispositivos podem ser usados para capturar pacotes?

O *Wireshark* pode ler dados de Ethernet, Token-Ring, FDDI, serial (PPP e SLIP) (se o sistema operacional permitir o *Wireshark*), 802.11 wireless LAN (se o sistema operacional permitir o *Wireshark*), conexões ATM (se o sistema operacional permitir o *Wireshark*), e qualquer dispositivo suportado no Linux por versões recentes do *libcap*.

- O *Wireshark* suporta interfaces virtuais?

Sim, sendo estas:

- *Loopback*: Captura o tráfego de uma máquina para ela mesma, incluindo o *localhost*.
- *Pipes*: É possível utilizar os *pipes* do UNIX para capturar pacotes de outras aplicações.

- **VLAN:** Captura o tráfego de uma LAN virtual (VLAN), incluindo VLAN *tags*.
 - **WinPcapRemote:** Captura remota do *WinPcap*, que auxilia a placa de rede operar em modo híbrido.
 - Porque o *Wireshark* é útil?
- Segundo uma resposta de um usuário: "É uma ótima ferramenta de aprendizado, tanto para amadores, iniciantes, intermediários e avançados na segurança da informação. É muito interessante para entender o tráfego de rede, como as coisas acontecem. Eu utilizei para entender ataques de segurança e cenários deste tipo.", (WARRAG, 2016).
- Qual a diferença entre *Tcpdump* e *Wireshark*?

"*Tcpdump* é uma ferramenta CLI (Interface por linha de comando), que pode ser executado remotamente em uma sessão SSH (Secure socket shell), aceita uma diversidade grande de filtros e é possível mostrar o dados correndo na rede. É muito poderoso.

Wireshark é uma ferramenta com interface gráfica onde é possível capturar pacotes e olhar para estes. É possível isolar *streams* assim como a conversação durante uma sessão TCP específica.", segundo (MALLET, 2017).

4.2 TCPDUMP

TCPDump é um analisador de pacotes executado em linhas de comando (AS-RODIA; PATEL, 2012). Tal ferramenta, disponibiliza uma série de recursos ao utilizador, para representar, de forma mais "clara" aos olhos humanos, pacotes capturados na rede. Adiante serão apresentadas o funcionamento da ferramenta, como configurá-la em um ambiente Linux e as ferramentas utilizada pelos autores para realizar o experimento. O TCPDump também possui uma versão para Windows, chamada WinDump, porém, esse estudo utilizará apenas a versão Linux da ferramenta.

4.2.1 Histórico da Ferramenta

De acordo com Suri e Batra (2010) o TCPDump foi desenvolvido pelo grupo *Network Research Group* (NRG) da *Information and Computing Sciences Division* (ICSD) no *Lawrence Berkeley National Laboratory* (LBNL) e atualmente é mantido por www.tcpdump.org. No link mencionado anteriormente é possível acessar a documentação oficial, além de outros tutoriais a respeito da ferramenta.

4.2.2 Requisitos Básicos

Assim como mencionado anteriormente, o foco do trabalho é a utilizar a ferramenta em ambientes Linux, sendo assim o único requisito básico é um dispositivo equipado com uma sistema operacional Linux ou uma máquina virtual com Linux.

4.2.3 Instalação e Configuração

Para realizar a instalação em seu dispositivo Linux, basta seguir os passos a seguir:

1. Primeiro precisamos verificar se o TCPDump já não encontra-se instalado, para isso, execute o comando **sudo tcpdump**. Caso seja exibida uma mensagem do tipo: *command not found: tcpdump*, execute o passo 2.

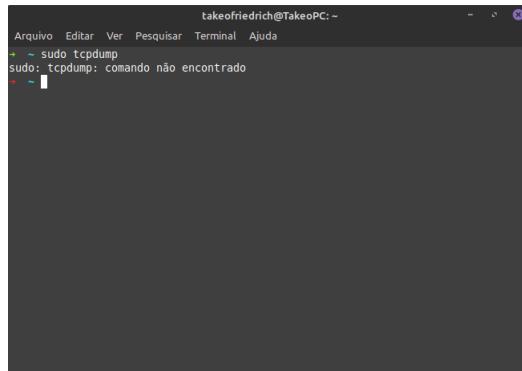


Figura 23 – Execução do passo 1. Nela é possível notar que o TCPDump não estava instalado: "comando não encontrado". Fonte: Próprios Autores.

2. Abra uma janela de terminal e execute o comando **sudo apt-get update** para atualizar seu repositório com as versões mais atuais dos pacotes;
3. Em seguida, execute o comando **sudo apt-get install tcpdump**.
4. Alguns pré-requisitos serão solicitados a serem instalados, para que o TCPDump funcione apropriadamente. Digite **Y** e pressione **ENTER** quando requerido.
5. Após finalizada a instalação, execute o comando **tcpdump -h**.

Na figura 24 à direita, é possível ver a versão da ferramenta e um resumo dos comandos, ao executar o passo 5. Uma explicação mais detalhada pode ser encontrada em (CHAPPELL; COMBS, 2013).

```

takeofriedrich@TakeoPC: ~
Arquivo Editar Ver Pesquisar Terminal Ajuda
qml-module-org-kde-kirigami2 qml-module-org-kde-kquickcontrols
qml-module-org-kde-kquickcontrolsaddons qml-module-org-kde-newstuff
qml-module-org-kde-people qml-module-org-kde-purpose
qml-module-org-kde-runnermodel qml-module-qtgraphicaleffects
qml-module-qtqml-models2 qml-module-qtquick-controls
qml-module-qtquick-controls2 qml-module-qtquick-dialogs
qml-module-qtquick-layouts qml-module-qtquick-privatewidgets
qml-module-qtquick-templates2 qml-module-qtquick-window2 qml-module-qtquick2
qml-module-ubuntu-onlineaccounts signon-plugin-oauth2 sonnet-plugins sshfs
Utilize 'sudo apt autoremove' para os remover.
Os NOVOS pacotes a seguir serão instalados:
tcpdump
0 pacotes atualizados, 1 pacotes novos instalados, 0 a serem removidos e 197 não
atualizados.
É preciso baixar 0 B/370 kB de arquivos.
Depois desta operação, 1.113 kB adicionais de espaço em disco serão usados.
A seleccionar pacote anteriormente não seleccionado tcpdump.
(Lendo banco de dados ... 393324 ficheiros e directórios actualmente instalados.
A preparar para desempacotar .../tcpdump_4.9.3-4_amd64.deb ...
A descompactar tcpdump (4.9.3-4) ...
Configurando tcpdump (4.9.3-4) ...
A processar 'triggers' para man-db (2.9.1-1) ...

takeofriedrich@TakeoPC: ~
Arquivo Editar Ver Pesquisar Terminal Ajuda
tcpdump -h
tcpdump version 4.9.3
libpcap version 1.9.1 (with TPACKET_V3)
OpenSSL 1.1.1f 31 Mar 2020
Usage: tcpdump [-A|B|C|D|E|F|G|H|I|J|L|N|O|P|Q|S|U|V|X|] [ -B size ] [ -c count ]
[ -C file size ] [ -E algo:secret ] [ -F file ] [ -G seconds ]
[ -I interface ] [ -J timestamptype ] [ -M secret ] [ -N number ]
[ -O inout|inout ]
[ -R file ] [ -S snaplen ] [ -T time-stamp-precision precision ]
[ --immediate-mode ] [ -T type ] [ --version ] [ -V file ]
[ -W filecount ] [ -Y datalinktype ] [ -Z postrotate
-command ]
[ -Z user ] [ expression ]

```

Figura 24 – Exceção dos comandos 3 e 5 da esquerda para a direita. Fonte: Próprios Autores.

4.2.4 Q & A

- O *TCPDump* só captura protocolos TCP?

Não, igual ao *Wireshark* o *TCPDump* captura todo tipo de protocolos como por exemplo: TCP, UDP, SMTP entre outros.

- Existe o *TCPDump* para o sistema operativo *Windows*?

Existe uma versão de *TCPDump* para *Windows* chamada *WinDump*. Ela é totalmente compatível com o primeiro mencionado e captura os pacotes utilizando o *WinPcap*.

- O *TCPDump* é indicado para principiantes?

Normalmente o *TCPDump* é indicado para pessoas que já tem algum tipo experiência com algum aplicativo de captura de pacotes de rede devido a que não possui uma interface gráfica, isto faz o que a utilização da ferramenta seja menos amigável com o usuário. Mas igualmente de poderosa.

- Qual a diferença entre *TCPDump* e *Wireshark*?

"*Tcpdump* é uma ferramenta CLI (Interface por linha de comando), que pode ser executado remotamente em uma sessão SSH (Secure socket shell), aceita uma diversidade grande de filtros e é possível mostrar o dados correndo na rede. É muito poderoso.

Wireshark é uma ferramenta com interface gráfica onde é possível capturar pacotes e olhar para estes. É possível isolar *streams* assim como a conversação durante uma sessão TCP específica.", segundo (MALLET, 2017).

5 MANUAL DE UTILIZAÇÃO

Após as ferramentas terem sido instaladas, será discutido um pouco como utilizá-las. Assim como mencionado anteriormente, Discord e Skype ficarão de fora dessa discussão pois o foco do trabalho não é o uso das funcionalidades da ferramenta. Como o Wireshark e TCPDump serão utilizados para a parte da análise, o foco da discussão será nelas.

5.1 WIRESHARK

Como já foi explicado na seção 4.1 o *Wireshark* é um analisador de tráfego de redes multiplataforma, ele é utilizado para capturar os protocolos que entram e saem de uma rede. Em esta seção vamos falar sobre a utilização da ferramenta depois de sua configuração inicial.

Em primeiro lugar serão apresentados os principais parâmetros para filtrar redes, logo após como começar uma captura de pacotes e por último como monitorar esta captura além de como salvá-la.

5.1.1 Parâmetros configuráveis

A principal configuração a ser feita pre-captura de pacotes é a configuração de filtros, estas são inseridas em dois campos de texto diferentes como indicados na figura 25. A seta 1 indica o campo de texto para inserir filtros que vão afetar só visualmente a captura, ou seja, o *Wireshark* vai capturar todos os pacotes mas vai mostrar aqueles que você já indicou no espaço mencionado. Por outro lado a seta 2 mostra o campo de texto onde são inseridos os filtros de captura, estes filtros afetam diretamente na leitura de protocolos, isto quer dizer que não serão capturados protocolos que não sejam especificados anteriormente no filtro.

Agora que já mostramos onde podem ser inseridos os filtros vamos analisar quais são estes filtros.

Segundo o (WIRESHARK, 2020a) existem mais de 251000 possíveis campos em 3000 protocolos diferentes na versão atual do *Wireshark* (3.2.7), já que é praticamente impossível abranger todas as possibilidades diferentes vamos apresentar os principais filtros em tópicos.

- Filtro por protocolo padrão: o nome dos protocolos devem escrever-se em minúscula.

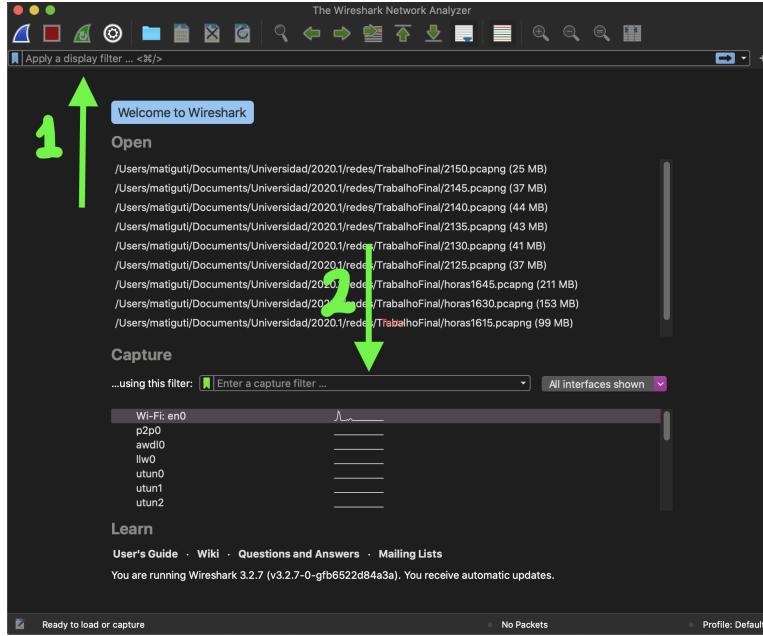


Figura 25 – Possível ubicações de filtros no Wireshark. Fonte: Próprios Autores.

- smtp: filtra o protocolo SMTP.
 - ldap: filtra o protocolo ldap
 - ssl: filtro para protocolos SSL y TLS, pode-se usar com HTTPS e TLS no caso de *e-mail*.
 - http: filtro de navegação WEB.
 - dns: filtro para obter consultas DNS.
 - icmp: filtra o protocolo ICMP.
- Filtro com símbolos lógicos: Os protocolos apresentados no item anterior podem ser concatenados com simbolos lógicos para gerar novos filtros. Como por exemplo:
 - not smtp: Exclui o protocolo smtp.
 - smtp or icmp: Filtra tanto o protocolo SMTP como o protocolo ICMP.
 - apresentaremos mais exemplos e casos mais complexos nos seguintes *items*.
 - Mais opções de filtro por protocolo: Dentro de cada protocolo podemos achar dezenas de opções de filtragem, para acessar estas opções adicionamos um ponto logo de escrever o nome do protocolo que queremos filtrar. Para entender melhor vamos ver vários exemplos de combinações embaixo.
 - ip.addr == 172.20.10.5: filtra só os protocolos que possuam o endereço 172.20.10.5.

- `tcp.port == 25`: Mostra só os protocolos que entram através da porta 25 (SMTP).
- `ip.addr == 172.20.10.5 and ip.addr == 192.168.0.5`: filtra o tráfego de rede entre o servidor 172.20.10.5 e o servidor 192.168.0.5.
- `smtp.req.command eq RCPT and smtp.req.parameter contains "a@domi.com"`: filtra um e-mail onde o destinatário é "a@domi.com".

Como deve ter percebido o limite de filtros depende praticamente da sua imaginação, obviamente sempre que siga a lógica de redes. Na figura 26 podemos observar um detalhe na interface gráfica que ajuda muito na hora filtrar protocolos, o campo de texto muda de cor para vermelho caso o filtro que você tenha escrito esteja errado ou para verde caso esteja correto e seja possível realizar dito filtro.

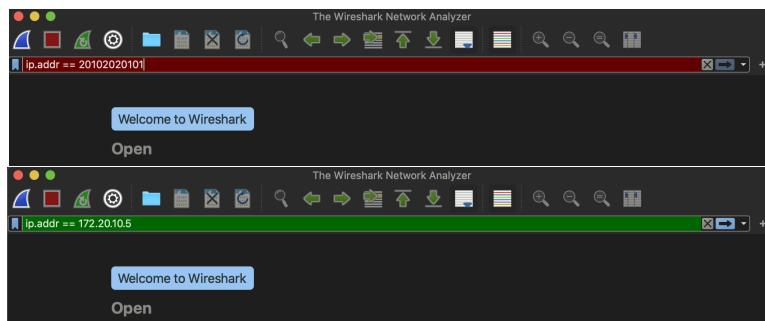


Figura 26 – Na figura a cima podemos encontrar um exemplo de filtro com sintaxe errada e na figura a embaixo um exemplo com a estrutura correta de filtragem. Fonte: Próprios Autores.

5.1.2 Inicialização

Para inicializar uma captura de protocolos você deve em primeiro lugar escolher a interface, podemos observar na figura 27 as opções, vamos fazer um exemplo seguindo o caso estudado em este trabalho.

Escolhemos a opção *Wi-Fi* indicada na seta 1, depois apertamos o símbolo azul do *Wireshark* indicado pela seta 2. A seguir você começará a captura sem filtros, conseguindo ver todos os protocolos de rede capturados ao vivo. A seção a seguir apresenta o monitoramento logo de ter começado a execução.

5.1.3 Monitorando a execução

Uma vez iniciada a captura vamos encontrar a janela da figura 28.

Fizemos a separação por cores da estrutura de monitoramento. No retângulo amarelo podemos encontrar vários botões que interagem diretamente com a leitura de protocolos, dentro das opções encontramos por exemplo o segundo símbolo, o quadrado vermelho, que serve para parar a captura, os outros botões servem para

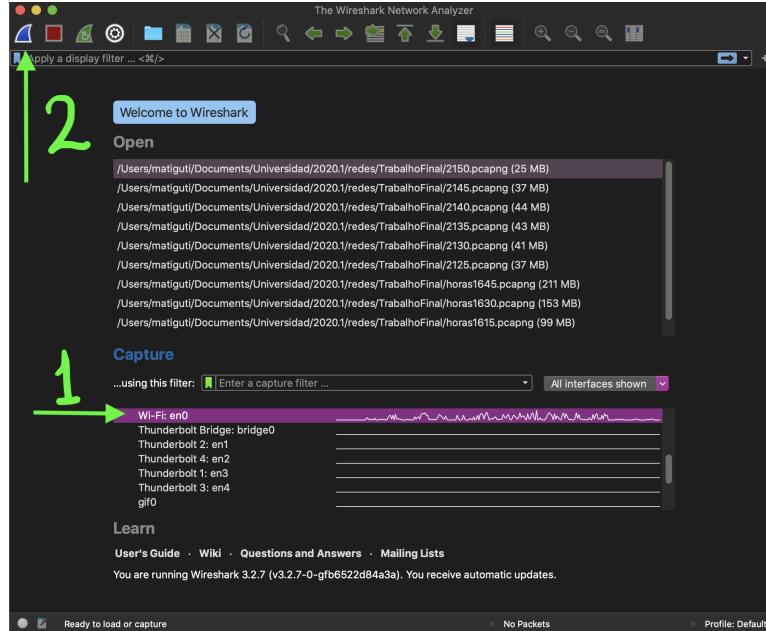


Figura 27 – Caminho para começar captura de pacotes no Wireshark. Fonte: Próprios Autores.

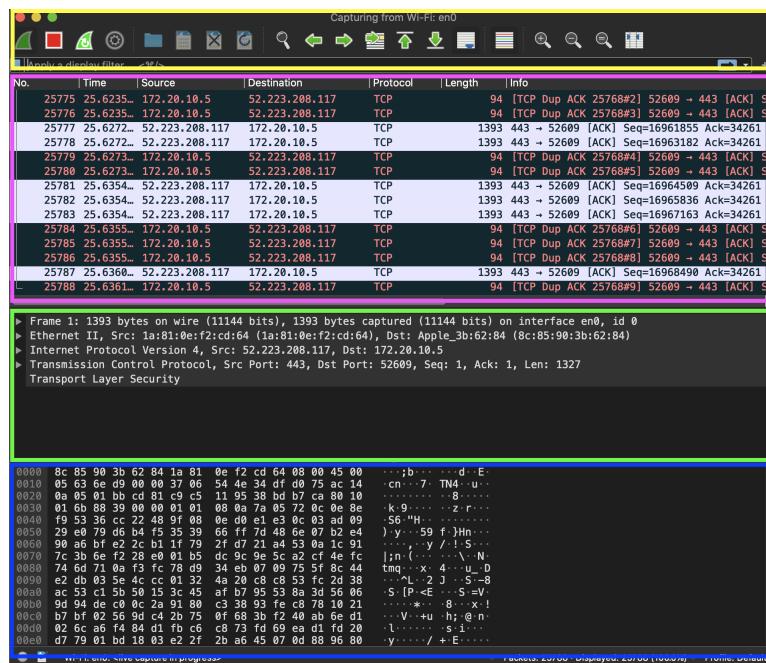


Figura 28 – Monitoramento de captura de protocolos. Fonte: Próprios Autores.

organizar melhor o seguimento das entradas. No retângulo rosa achamos todos os pacotes chegando um por um, eles são divididos por colunas que servem para detectar e organizar melhor cada pacote. Caso selecionemos um pacote dentro do retângulo rosa, vamos observar as informações detalhadas de cada nível do pacote dentro da área demarcada pelo retângulo verde. Por ultimo, dentro do retângulo azul achamos as informações dentro dos pacotes em hexadecimal.

5.1.4 Coletando os resultados

Seguindo os passos apresentados nas seções anteriores coletamos um total de duas capturas de cinco minutos e uma de trinta em três dispositivos diferentes. Os dispositivos utilizados são os apresentados na tabela 3

	Localização	Sistema Operacional	Processador	Placa de Rede	Taxa de Download	Taxa de Upload
Dispositivo 1	Joinville, SC - Brasil	Linux Mint 20	Intel i3 9100F 3.6 GHz	Realtek RTL8111/8168/8411	14 Mbps	1.3 Mbps
Dispositivo 2	Assunção, Paraguai	MacOS Catalina 10.15.5	Intel i7 7700HQ 2.8 GHz	AirPort Extreme (0x14E4, 0x173)	10 Mbps	34 Mbps
Dispositivo 3	Florianópolis, SC - Brasil	Windows 10 Home 1909	Intel i7 7700HQ 2.8 GHz	Intel(R) Dual Band Wireless-AC 8265.	135 Mbps	15.63 Mbps

Tabela 3 – Tabela com as especificações dos dispositivos utilizados pelos autores. Fonte: Próprios Autores.

As capturas foram realizadas sem filtros aplicados previamente, ou seja, foram capturados todos os pacotes possíveis. Ao terminar o tempo pre-definido de captura todos eles foram salvos em arquivos .pcap e analisados na ferramenta *Wireshark*. Estes dados serão utilizados nas seções seguintes para fazer uma análise do consumo do aplicativo Discord.

5.1.5 Q&A

- É possível filtrar durante uma captura de pacotes?

Sim, só precisa escrever seu filtro desejado no campo de texto chamado "Display Filter".

- O *Wireshark* consegue abrir capturas feitas em outros aplicativos?

Sim, um exemplo claro seria o *Wireshark* abrindo arquivos .pcap gerados pelo TCPDump.

- O *Wireshark* consegue filtrar IPv6?

Sim, é possível só precisa digitar ipv6 dentro do campo de filtros. Vamos analisar um exemplo:

ipv6.addr == fe80::f61f:c2ff:fe58:7dcb

Nesse caso estamos filtrando pelo IP fe80::f61f:c2ff:fe58:7dcb claramente escrito em hexadecimal.

5.2 TCPDUMP

Assim como mencionado nas seções anteriores, o TCPDump será utilizado para capturar os pacotes transitando em uma rede. Serão apresentados a seguir parâmetros que podem ser utilizados durante a captura e formas de filtrar os pacotes que estão sendo capturados. Os comandos utilizados são expressões booleanas, que quando um pacote está de acordo com ela, ele é “separado” dos demais. Tais comandos representam filtros. Além dos filtros também é possível configurar a forma em que eles são dispostos, o local onde serão despejados, dentre outros. A seguir apresentaremos tais formas.

5.2.1 Parâmetros configuráveis

Os parâmetros configuráveis serão divididos em dois: os para realização de filtros e os para geração de saída. Nas subseções que se seguem, eles serão apresentados.

5.2.1.1 Parâmetros de Filtros

Para utilizar os parâmetros de filtros que serão discutidos aqui, é necessário estar no modo **super usuário**. Para tal, utilize o comando o **sudo su** e após utilize o comando **tcpdump COMANDO**, ou utilize os comandos na forma **sudo tcpdump COMANDO**. Em ambas será necessário digitar a senha de superusuário.

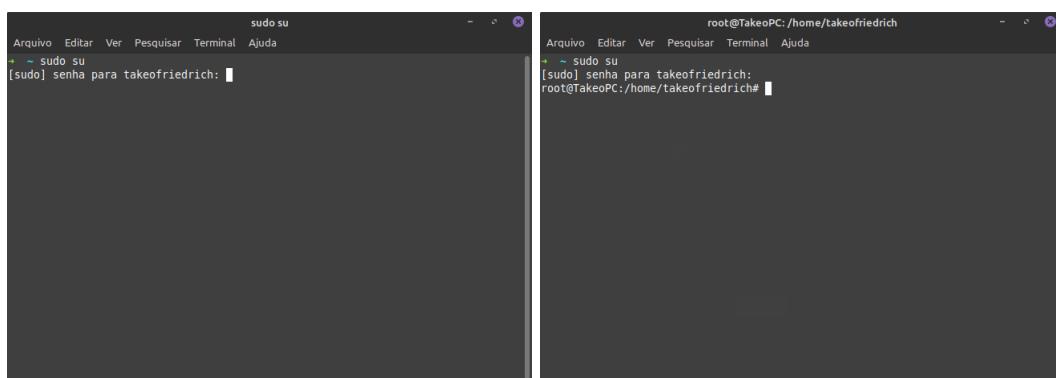


Figura 29 – Na figura a esquerda está representada a execução do comando **sudo su**, após digitada a senha, a figura da direita apresenta o terminal no modo super usuário. Fonte: Próprios Autores.

Agora, serão descritos alguns comandos, também chamados de qualificadores, que podem ser utilizados nas expressões booleanas utilizadas pelo TCPDump. Como são expressões booleanas, eles podem ser agrupados utilizando os operadores **or** e **and**. Caso queira realizar operações com prioridade nos operadores, é necessário utilizar o caractere de escape contra-barra, por exemplo: **tcpdump x and \y or z**, onde x,y e z são qualificadores. Por default, se nenhum qualificador for incluído

(**sudo tcpdump**), todos os pacotes serão jogados na saída padrão, no caso, o próprio terminal.

Os qualificadores podem ser divididos em três: de tipos, de direção e de protocolo. Os qualificadores de tipo dizem qual tipo de identificador de nome ou número se refere. Por exemplo: **port 20** (filtra os pacotes recebidos na porta 20). Já os direcionais, especificam, assim como o próprio nome, a direção. Por exemplo: **src 192.168.0.10** (pacotes emitidos a partir da fonte 192.168.0.10). E por último, os de protocolo, especificam quais protocolos serão filtrados. Por exemplo: **tcp** (pacotes TCP).

Veja a seguir algumas primitivas apresentadas na documentação oficial:

- **host host**: filtra os pacotes que são enviados ou recebidos pelo host. O campo em negrito pode ser substituído por **src** (apenas os pacotes recebidos de) e **dst** (apenas os pacotes enviados por).
- **ether host ehost**: pacotes em que o ehost é a origem ou o destino ethernet. O ether pode ser substituído por scr e dst assim como no comando anterior.
- **gateway host**: pacotes em que o host é um gateway.
- **net net**: pacotes em que a origem ou destino IPv4/v6 possui net como número de rede.
- **net net mask netmask**: semelhante ao comando anterior, só que incluido a máscara de rede.
- **net net/len**: pacotes provindo de redes com o tamanho de máscara dos endereços correspondente ao valor net/len. Pode ser utilizando em conjunto com o src e dst ao inicio para incluir especificações de origem e destino.
- **dst port port**: pacotes de uma porta port recebidos. O dst também pode ser substituído por src.
- **less length**: pacotes que possuem tamanho menores que lenght. less pode ser substituído por greater (maiores que o tamanho).
- **ip broadcast**: pacotes que são de broadcast.
- **ip multicast**: pacotes que são de multicast.
- **tcp,udp,icmp**: pacotes que são de um desses protocolos.

5.2.1.2 Parâmetros de Saída

Para visualizar os dados, o TCPDump disponibiliza alguns comandos específicos. Alguns deles serão citados a seguir:

- -A exibe cada pacote in ASCII.
- -d exibe os pacotes de forma legível a humanos.
- -dd exibe os pacotes como fragmento de código em C.
- -ddd exibe os pacotes como números na base 10.
- -F arquivo: utiliza o arquivo como input para a expressão booleana.
- -h: exibe os comandos.
- –version: exibe a versão.
- -n: não converte os endereços para nomes.
- -r arquivo: lê os pacotes de um arquivo.
- -v: quando os pacotes são redirecionados para um arquivo, algumas informações ainda são exibidas no console.
- -w arquivo: armazena os pacotes no arquivo.

5.2.2 Inicializando a captura

Algumas opções de filtros foram apresentadas anteriormente. Nessa seção serão apresentadas os filtros que serão utilizados no caso de estudo desse trabalho. Para que os dados possam ser visualizados no Wireshark, gerou-se arquivos de saída .pcap, assim os arquivos podem ser aberto na ferramenta, possibilitando o uso dos recursos de análise de dados do próprio Wireshark. Os autores utilizaram o comando **ip addr**, como é possível ver na Figura 30 no quadro a para obter o endereço IPv4.

Após, utilizou se o filtro **sudo tcpdump host ENDERECO_IP -w NOME_ARQUIVO.pcap -v**, onde o **ENDERECO_IP** é o endereço IP e o **NOME_ARQUIVO** é o nome do arquivo gerado de saída. O parâmetro **-v** foi utilizado para exibir no terminal a quantidade pacotes capturados durante a geração do arquivo.

```

takeofriedrich@TakeoPC: ~
Arquivo Editar Ver Pesquisar Terminal Ajuda
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
t qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lif forever preferred_lif forever
    inet6 ::1/128 scope host
        valid_lif forever preferred_lif forever
2: enp5s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether b4:2e:99:cc:f1:75 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.17/24 brd 192.168.0.255 scope global dynamic noprefixroute enp5s0
        valid_lif 1874sec preferred_lif 1874sec
        inet6 2b04:14c:2b7:9884::2/128 scope global dynamic noprefixroute
            valid_lif 54077sec preferred_lif 39677sec
            inet6 2b04:14c:2b7:9884:7d83:9ded:94fb:8809/64 scope global temporary dynamic
                valid_lif 86387sec preferred_lif 71987sec
                inet6 fe80::e185:faf2:6105:bffa/64 scope link noprefixroute
                    valid_lif forever preferred_lif forever
* - [  ] 

```

```

Arquivo Editar Ver Pesquisar Terminal Ajuda
+ ~ sudo tcpdump host 192.168.0.17 -w teste.pcap -v
tcpdump: listening on enp5s0, link-type EN10MB (Ethernet), capture size 262144 bytes
Got 50

```

Figura 30 – Capturando pacotes com filtros. Fonte: Próprios Autores.

5.2.3 Monitorando a execução

Como comentado anteriormente, o parâmetro **-v** gera uma saída no terminal enquanto as informações referentes aos pacotes são disponibilizados em um arquivo externo utilizando o parâmetro **-w**. Na figura 30 a direita é possível ver a quantidade de pacotes capturados a cada momento. O resultado gerado pode ser aberto tanto no próprio TCPDump como no Wireshark.

```

takeofriedrich@TakeoPC: ~
Arquivo Editar Ver Pesquisar Terminal Ajuda
win 11791, options [nop,nop,sack 1 {279128:279202}], length 0
00:29:21.630410 IP 162.159.128.232.https > TakeoPC.47692: Flags [.], seq 276208:277668, ack 11814, options [nop,nop,sack 1 {279128:279202}], length 0
00:29:21.630447 IP TakeoPC.47692 > 162.159.128.232.https: Flags [.], ack 277668, win 11814, options [nop,nop,sack 1 {279128:279202}], length 0
00:29:21.648300 IP 162.159.128.232.https > TakeoPC.47692: Flags [.], seq 277668:279128, ack 338, win 70, length 1460
00:29:21.648336 IP TakeoPC.47692 > 162.159.128.232.https: Flags [.], ack 279202, win 11836, length 0
00:29:21.828705 IP 162.159.136.234.https > TakeoPC.35110: Flags [P.], seq 496:552, ack 54, win 184, length 56
00:29:21.828727 IP TakeoPC.35110 > 162.159.136.234.https: Flags [.], ack 552, win 6638, length 0
00:29:22.294008 IP TakeoPC.56512 > 153.232.73.34.bc.googleusercontent.com.https: Flags [.], ack 1038618756, win 1146, options [nop,nop,TS val 3845013384 ecr 308882734], length 0
00:29:22.316039 IP TakeoPC.48162 > 216-21-12-105.customer.totaluptime.net.https: Flags [.], ack 4267608573, win 62780, length 0
00:29:22.432458 IP 153.232.73.34.bc.googleusercontent.com.https > TakeoPC.56512: Flags [.], ack 1, win 2059, options [nop,nop,TS val 3088921868 ecr 3044923214], length 0
00:29:22.462064 IP 216-21-12-105.customer.totaluptime.net.https > TakeoPC.48162: Flags [.], ack 1, win 65535, length 0

```

Figura 31 – Abrindo um arquivo .pcap utilizando o TCPDump. Fonte: Próprios Autores.

Para abrir o resultado no próprio terminal utilizando o TCPDump, basta utilizar o comando **sudo tcpdump -r NOME_ARQUIVO**, veja na figura 31.

5.2.4 Colegando os resultados

Os resultados foram obtidos a partir de três dispositivos diferentes, as especificações de cada um estão na apresentados na tabela 3

Foram realizadas duas capturas de cinco minutos e uma de trinta em três dispositivos diferentes. Estas serão utilizadas como caso de estudo e comentadas a respeitos nas sessões seguintes.

5.2.5 Interpretando e analisando os resultados

A partir dos resultados obtidos com o comando **`sudo tcpdump -r NOME_ARQUIVO`**, uma linha da saída do arquivo foi retirada e é apresentada a seguir:

```
00:28:55.601019 IP 162.159.133.233.https > TakeoPC.35210: Flags [.], ack 1, win 131,
length 0
```

A partir dela é possível retirar algumas informações pertinentes ao pacote. Por exemplo, a primeira coluna representa o horário da captura do pacote **00:28:55.601019**. A seguir é possível perceber que o pacote é transmitido de um emissor a um destinatário, e essa informação está separada por um símbolo de **>**. No caso do exemplo o emissor é **IP 162.159.133.233.https** e o destinatário é: **TakeoPC.35210**. O campo flags do pacote está vazio, pois: **Flags [.]**. O pacote é um pacote de *Acknowledgment*, pois seu campo ACK possui um 1. Sua janela de transmissão é 131. Isso pode ser visto em **win 131** e seu tamanho é zero: **length 0**.

5.2.6 Q&A

- É possível ver quais são as interfaces de rede que consegue-se realizar capturas?

Sim, com o argumento `--list-interfaces`, o qual retornará os nomes de cada interface de rede.

- Como monitorar interfaces de rede específicas pelo tcpdump, i.e., `eth0`, `eth0`, etc?

Para monitorar uma interface específica é possível utilizar-se do argumento `-i`, e após isso o nome da interface que deseja realizar a captura.

- Como capturar uma os pacotes obtidos numa porta específica, ou então num protocolo específico?

Segundo (GITE, 2008), primeiro deve se informar qual interface de rede será utilizada para a captura, e.g., `eth1`, e, após a interface, entre aspas simples identificar a porta a ser monitorada, i.e., "port 80", o comando para monitorar a porta 80 (HTTP), seria "`tcpdump -i eth1 'port 80'`". Também é possível dar um intervalo de portas a serem analisadas, e.g., "`tcpdump portrange 21-23`".

- Como salvar as capturas em arquivos?

Com o argumento `-w`"e o nome do arquivo, com isso os dados da captura, i.e., pacotes, protocolos, endereços, etc, serão escritos no arquivo de saída.

6 ANALISANDO AS CAPTURAS

Após os dados terem sido coletados, é necessário entender um pouco a respeito da disposição dos dados em uma saída. As seções seguintes discutirão isso.

6.1 INTERPRETAÇÃO DA SAÍDA

Os computadores conectados a rede são identificados com um *Internet Protocol* (IP), o qual, deve ser único para ele, o qual um pacote enviado da Noruega, por exemplo, deve saber qual o destinatário dele. O destinatário e a fonte possuem IP únicos para serem identificados e receberem os dados enviados. Sendo assim, possuímos sete colunas de dados de saída no Wireshark, as quais serão explicadas na lista a seguir:

- Número: é o número do pacote sendo analisado numa ordem crescente, de acordo como o pacote é recebido.
- Tempo: tempo em que o pacote foi recebido, iniciando numa contagem de 0 de quando a captura é iniciada.
- Fonte: IP da máquina a qual envia um pacote para outra.
- Destino: IP da máquina destino, a qual recebe o pacote enviado.
- Protocolo: Protocolo utilizado na comunicação entre máquinas, i.e., *Hypertext Transfer Protocol* (HTTP), *User Datagram Protocol* (UDP), *Transmission Control Protocol* (TCP), etc.
- Tamanho: representa o tamanho do pacote enviado, onde possui os dados e as informações referentes ao protocolo e às máquinas utilizadas.
- Informações: esta contém os dados contidos nos pacotes enviados, porém, possuem uma janela de transmissão, uma vez que não é possível enviar um arquivo inteiro de uma vez só. Com isso, quando tem-se "50001 -> 61590", é possível afirmar que estamos enviando do *frame* 50001 ao 61590.

Em relação aos dados obtidos, é muito clara a ampla utilização do UDP como protocolo principal para a comunicação, em média, 97% dos pacotes estudados utilizam o protocolo. O UDP é mais rápido que o TCP, porém, o custo dele vem em forma de que os dados não chegam na ordem que são enviados, ou seja, a integridade dele

pode ser comprometida se depender da ordem. Porém, com uma boa aplicação é possível organizar os pacotes de maneira que a vantagem da velocidade se torne mais atraente.

6.2 Q&A

- Como ver os dados dos tamanhos dos pacotes? Após realizar uma captura é possível clicar em: "Statistics -> Packet Lengths", que então será dado os dados acerca dos tamanhos dos pacotes, i.e, número de ocorrências, média, entre outros.
- Como ver quantos pacotes foram enviados ou recebidos em um segundo? Para isso é necessário gerar um gráfico de entrada e saída, para isso é necessário clicar em: "Statistics -> I/O Graph".
- Como gerar um gráfico mostrando a Latência *Round Trip Time*? Será necessário inserir o filtro: "tcp.flags.syn ==1", e então clicar nas seguintes opções: "Statistics -> TCP Stream Graphs -> Round Trip Time".

7 CASOS DE ESTUDO

Com o uso da Internet atualmente se tornando cada vez mais amplo, e o aumento da largura de banda usada sendo aprimorado constantemente (KASSIM et al., 2017), a indústria da tecnologia desenvolveu e aprimorou nos últimos anos ferramentas para que usuários fiquem conectados com as pessoas ao seu redor. Respectivamente, dois dos aplicativos multimídia mais famosos do momento que suportam chamadas de voz e vídeo, são o *Skype* (40 milhões de usuários ativos por mês (MEHDI, 2020)) e o *Discord* (100 milhões de usuários de usuários ativos por mês (NELLY, 2020)).

Além do natural aumento do uso da Internet com cada vez mais usuários conectados, desde o primeiro surto da COVID-19 entre o final de 2019 e atualmente, a pandemia trouxe mudanças na quantidade de acessos a diversas plataformas. Com restrições limitando a mobilidade dos indivíduos para sair de casa, o trabalho e o ensino remoto cresceram juntamente com opções de lazer online: como jogos, comunicação via voz/vídeo e *streaming* (FAVALE et al., 2020).

De acordo com a Cloudflare, empresa consolidada no ramo de servidores para a Internet, com a Figura 32 pode-se analisar as mudanças no tráfego na Internet durante o período de pandemia:

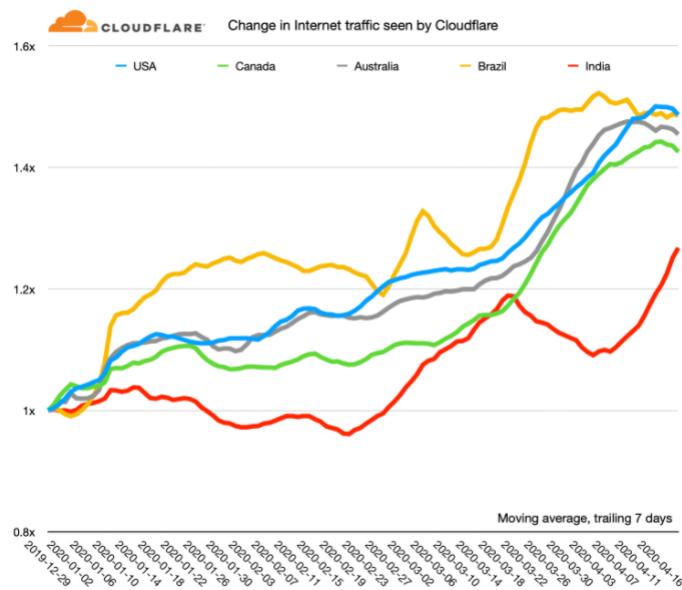


Figura 32 – Mudanças no tráfego na Internet analisadas pela Cloudflare. Fonte: (GRAHAM-CUMMING, 2020)

Tendo em vista isso, com o aumento gradual do número de usuários nessas plataformas, a necessidade de quantizar as informações sobre o consumo de dados

em ambientes domésticos é crescente, visando estudar as consequências do uso de determinados softwares da categoria VOIP como o *Skype* e *Discord* em redes 3G/4G e *WiFi*. Nesse sentido, partimos do pressuposto de analisar um artigo referenciado sobre o *Skype* e outro, conforme não foram achados trabalhos ou estudos desse tema, sobre o *Discord*.

Portanto, nas Seções 7.1 e 7.2, serão exploradas as análises de consumo de dados e características sobre o tráfego da rede dos softwares VOIP *Skype* e *Discord* com os *Packet sniffers Wireshark* e *TCPDump*.

7.1 REFERENCIADO

Nesse trabalho também buscou-se um artigo relacionado na literatura. Na próxima subseção será discutido o artigo escolhido.

7.1.1 Objetivo

O artigo (KASSIM et al., 2017) apresenta uma análise, baseada em chamadas de vídeo e voz, do tráfego de rede da aplicação *Skype* em uma Rede UniFi doméstica. Seu principal objetivo foi detectar se o uso de largura de banda (*bandwidth*) oferecida por uma rede doméstica UniFi é totalmente utilizada por seus usuários à medida que pagam por isso. A Figura 33 mostra a arquitetura da rede para o experimento:

As métricas utilizadas para a análise das características do tráfego de rede foram:

- O uso da largura de banda;
- *Round trip time* (RTT) - Latência;
- Perda de pacotes;
- Análise do protocolo do *Skype*;

Por meio disso, com as métricas estabelecidas, o método envolveu estabelecer chamadas de voz e vídeo em diferentes horários usando o *Skype* na rede UniFi enquanto um usuário captura o tráfego por meio do *Packet Sniffer Wireshark*. Os dados foram coletados e analisados usando o software MATLAB, além de comparações entre diferentes protocolos usados durante a captura.

7.1.2 Coleta de Dados

A Figura 34 mostra o fluxograma do sistema para a pesquisa. O Wireshark começa a captura uma vez que o usuário 2 atende a ligação. As chamadas foram

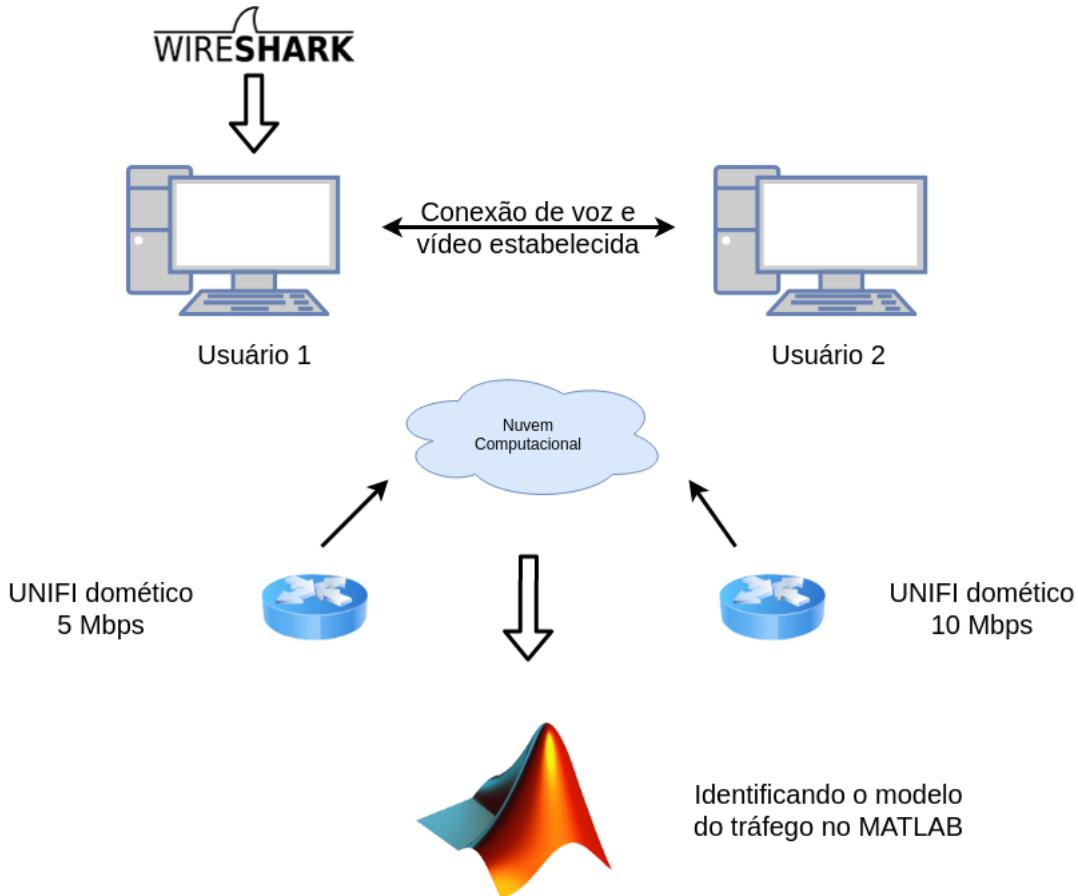


Figura 33 – Arquitetura do Experimento. Adaptado de: (KASSIM et al., 2017)

setadas para 1, 10 e 30 minutos, simultaneamente com a captura pelo *Packet Sniffer*. Após a coleta de dados, a latência e a quantidade de pacotes perdidos é determinada por meio do *Wireshark*, além da geração do gráfico I/O apresentando o uso de largura de banda de protocolos usados no tráfego capturado.

7.1.3 Análise

Conforme explicitado na seção 7.1.2, foi feita a geração de um gráfico I/O apresentando a largura de banda dos protocolos usados no tráfego capturado. Logo, a Figura 35 expõe o mesmo, usando um filtro para capturar protocolos UDP (linhas em vermelho) e TCP (linhas em azul).

A largura de banda usada foi organizada por meio da tabela 4 de acordo com o tempo de duração da chamada e horários de pico em mesmas redes ou diferentes.

De acordo com a Figura acima, em vídeo chamadas o Skype utilizou cerca de 75% da largura de banda oferecida, enquanto para chamadas de duração entre 1 a 10 minutos, menos de 5% do total. Com isso, prova-se que grande parte da internet não é utilizada. Além disso, com os resultados extraídos foi possível analisar que chamadas de vídeo precisam 4x mais largura de banda do que chamadas de áudio.

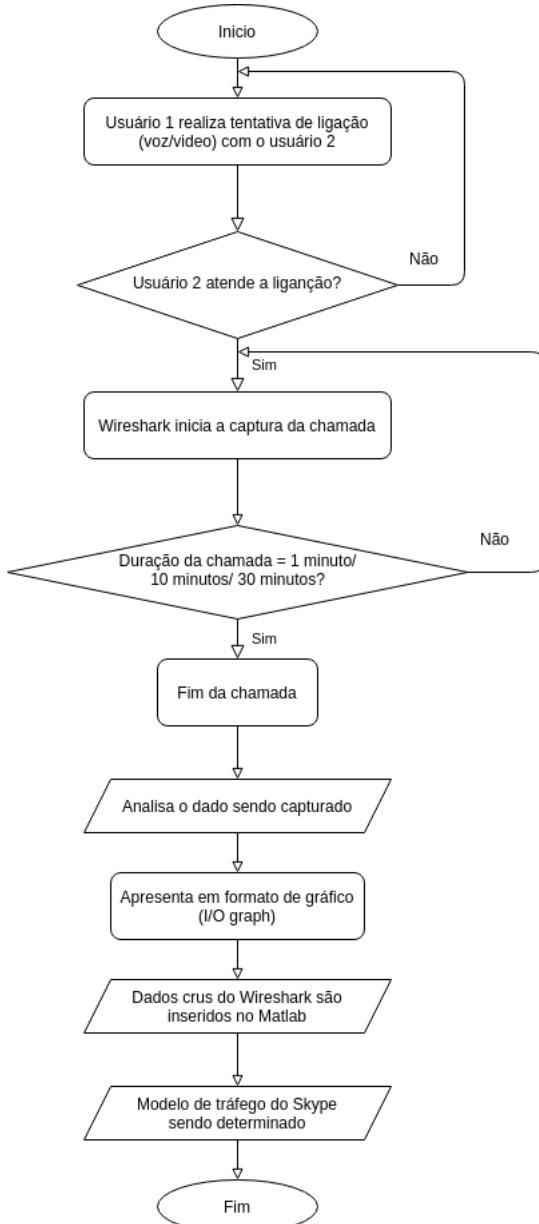


Figura 34 – Fluxograma do experimento. Fonte: Adaptado de: (KASSIM et al., 2017)

Para análise quantitativa da perda de pacotes, outra tabela foi gerada conforme a Tabela 5. A porcentagem de pacotes perdidos é cerca de 0.3% do tráfego original de dados, o que representa uma baixa perda de pacotes.

Para determinar a latência (RTT) do tráfego, foi usado no artigo o filtro **tcp.flags.syn=1**. Essa medida é referente entre o tempo gasto para dois sinal serem enviados e o tempo de confirmação de sinal recebido. Chamadas VoIP podem tolerar atrasos ponta a ponta de até 150ms em uma única direção (Chandran; Lingam, 2015). A Figura 6 mostra o valor geral do RTT, em segundos.

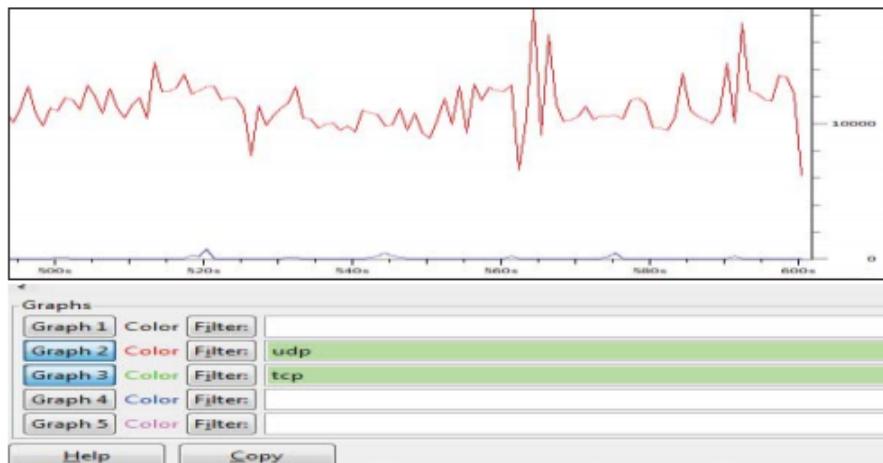


Figura 35 – Gráfico I/O no Wireshark. Fonte: (KASSIM et al., 2017).

Minutos	Mesma Rede		Rede Diferente	
	Hora com Pico	Hora Sem-Pico	Hora com Pico	Hora Sem-Pico
1 m Voz	965.963	768.204	955.091	3.651.753
10 m Voz	5.152.728	8.345.655	9.619.294	11.486.743
30 m Voz	6.554.357	54.309.322	26.333.824	13.547.012
1 m Vídeo	8.725.198	4.208.836	8.313.583	653.401
10 m Vídeo	91.323.711	29.867.780	66.189.705	58.269.214
30 m Vídeo	199.000.000	863.000.000	181.000.000	201.000.000

Tabela 4 – Largura de Banda Total para chamadas de voz e vídeo no Skype. Fonte: Adaptado de: (KASSIM et al., 2017).

Tipo de	Duração em	Mesma Rede		Rede Diferente	
		Hora com Pico	Hora Sem-Pico	Hora com Pico	Hora Sem-Pico
Aplicação Vídeo	Minutos	0	0	3	0
	10	0	0	18	0
	30	3	125	0	0
Voz	1	0	0	2	2
	10	0	0	28	1
	30	16	743	75	0

Tabela 5 – Quantidade de pacotes perdidos durante chamadas de voz e vídeo no Skype. Fonte: Adaptado de: (KASSIM et al., 2017).

7.1.4 Conclusões

Ao final do trabalho, a análise provou que o Skype usa os protocolos UDP e TCP, porém usando em maioria o protocolo UDP para chamadas de voz e vídeo.

Tipo de	Duração em	Mesma Rede		Rede Diferente	
		Hora com Pico	Hora Sem-Pico	Hora com Pico	Hora Sem-Pico
Aplicação Vídeo	Minutos	0,038	-	-	-
	10	0,264	0,011	0,216	0,395
	30	0,199	0,003	0,098	0,007
Voz	1	0,038	-	-	0,231
	10	0,0436	0,018	0,204	0,913
	30	0,005	0,003	0,041	0,045

Tabela 6 – Tráfego em RTT. Fonte: (KASSIM et al., 2017).

Essa configuração é explicada pela razão dessas chamadas não precisarem da verificação de transmissões confiáveis, pois são em tempo real e devem possuir menos sobrecarga. Além disso, baseado no tráfego coletado em rede *UniFi* de banda larga, é identificado que o *Skype* não usa muita largura de banda, cerca de 10% do oferecido.

Portanto, com os resultados finais observados, o artigo conclui uma análise de tráfego multimídia do software *Skype* na rede *Home UniFi* que concentra-se no tráfego de chamadas de voz e vídeo.

7.2 COMPARATIVO DO TRÁFEGO DE DADOS

Assim como mencionado anteriormente, o COVID-19 trouxe um novo contexto as relações sociais. Eventos que antes eram realizados presencialmente, tiveram que se adaptar a um novo contexto. Com isso, várias plataformas de comunicação tiveram aumento no número de usuários. Na figura 36 é possível ver na linha laranja, um pico na perda de pacotes dias após a primeira morte em São Paulo devido ao COVID-19. Pico este influenciado pela sobre carga de usuários na rede.

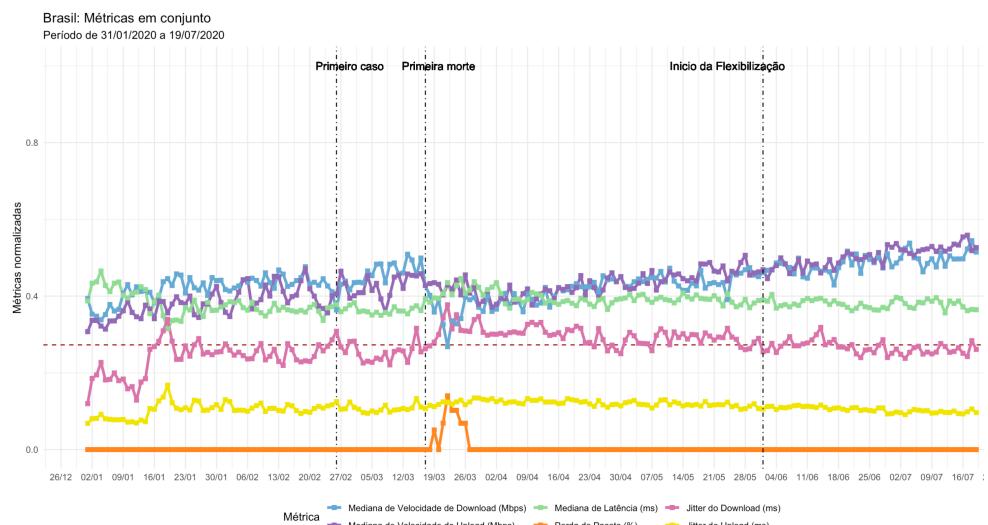


Figura 36 – Impactos na qualidade da internet no Brasil durante a pandemia de COVID-19, análise realizada entre 31/01/2020 e 19/07/2020. Fonte: (BR, 2020).

Nesse contexto, uma das ferramentas disponíveis como substituta as comunicações não-virtuais é o Discord, foco desse trabalho.

7.2.1 Objetivo

Com a plataforma Discord, buscou-se capturar os pacotes transmitidos durante uma sessão. A partir dos dados coletados comparou-se os resultados com o trabalho de (KASSIM et al., 2017), mencionado anteriormente. Levando em consideração fatores como a banda-larga de cada usuário e o tipo de rede, gráficos comparativos foram gerados.

7.2.2 Coleta de Dados

Os dados foram realizados por três computadores com as características descritas na tabela 3, de maneira que foram realizadas duas capturas de cinco minutos, e uma de trinta minutos.

7.2.3 Análise

Como visto na seção 6 e em 7.1, o protocolo mais utilizado para a comunicação via *Voice over Internet Protocol* (VoIP), é o UDP tendo que em média, dos arquivos de captura gerados pela equipe que realizou o trabalho, 97% dos pacotes analisados utilizaram UDP.

		Bytes in Flight (% do total da captura e número de pacotes)		
Duração em minutos	Versão da Captura	Dispositivo 1	Dispositivo 2	Dispositivo 3
5	Captura 1	0.2 % - 257	0.7% -773	0.3 % - 486
	Captura 2	0.2% - 190	0.5% - 614	0.2 % - 306
30	Captura 1	0.3% - 1547	0.5% - 3300	0.4% - 3029

Tabela 7 – Bytes in Flight. Fonte: Próprios Autores.

Bytes in flight representam pacotes que não receberam o *acknowledged*, ou seja, foram enviados os dados, porém, o emissor não sabe se eles realmente foram recebidos pelo receptor. Com isso, no próximo envio, serão enviados menos dados, porque a janela de transmissão não pode ser atualizada. Nos dados coletados pela equipe foi visto que a configuração do dispositivo 1 foi a que registrou um menor números de *Bytes in flight*, o que pode ser justificado por uma conexão melhor com o servidor buscado. Numa captura realizada por 30 minutos pode se notar que a diferença é de quase metade, quando relacionado aos outros analisados.

A retransmissão é realizada quando ocorre uma falta de pacote, que é solicitada ao emissor, o qual reenvia ele. Esta retransmissão acaba tendo um impacto no tempo total de execução, e tende a ocorrer menos com uma boa conexão entre fonte e destino, uma vez que caso o pacote não chegue, ou chegue danificado, incom-

Duração em minutos	Versão da Captura	Retransmissão (% do total da captura e número de pacotes)		
		Dispositivo 1	Dispositivo 2	Dispositivo 3
5	Captura 1	0 % - 9	0% - 31	0% - 14
	Captura 2	0% - 6	0% - 15	0% - 5
30	Captura 1	0% - 59	0% - 172	0% - 5

Tabela 8 – Retransmissão. Fonte: Próprios Autores.

pleto deverá ser reenviado. O pior dado, novamente, é encontrado na rede 4G, a qual apresenta o pior desempenho quando comparada às outras.

Duração em minutos	Versão da Captura	Perda de pacotes (% do total da captura e número de pacotes)		
		Dispositivo 1	Dispositivo 2	Dispositivo 3
5	Captura 1	0% - 12	0% - 14	0% - 0
	Captura 2	0% - 3	0% - 1	0% - 1
30	Captura 1	0% - 16	0% - 11	0% - 7

Tabela 9 – Perda de Pacotes. Fonte: Próprios Autores.

De acordo com a Tabela 9, a perda de pacotes foi baixa em todas as redes analisadas, sendo próximas a zero, porém, a que registrou um pior desempenho foi nesse quesito foi o dispositivo 1, a qual registrou o maior número nas capturas realizadas.

Tamanho dos pacotes para capturas de 5 minutos:

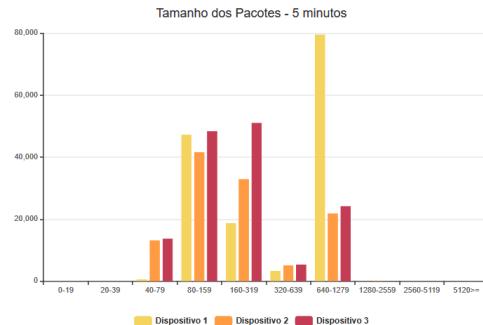


Figura 37 – Gráfico de Barras para o tamanho dos pacotes.

Tamanho dos pacotes para capturas de 30 minutos:

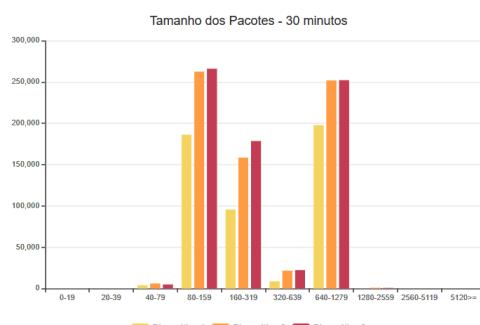


Figura 38 – Gráfico de Barras para o tamanho dos pacotes.

O dispositivo 1 é quem apresenta uma média de tamanho de pacote maior, sendo esta 549,97, o que pode desencadear em perdas de pacote, uma vez que podem causar congestionamento na rede, pois, caso ocorra um erro, mais pacotes deverão ser reenviados. Como o caso estudado foi utilizado VoIP, então é muito comum ocorrerem erros de transmissão, devido ao volume de pacotes, então deveriam ser utilizados tamanhos menores. (Korhonen; Wang, 2005) afirma que grandes pacotes evitam o *overhead*, porém, pequenos pacotes irão gerar mais segurança e integridade para os dados transmitidos.

Latência de *Round-trip Time* é descrita como o tempo que um pacote leva do seu destino até seu destinatário, e esta pode ser afetada por diversos fatores, i.e, demora de: propagação, processamento, fila e de codificação (CALLSTATS, 2018). Sendo assim, é afetada por diversos fatores, mas é um dos pontos principais quando o dado a ser analisado é a velocidade da rede monitorada. Acerca dos gráficos de RTT dispostos, conforme Figuras 39, é possível notar uma certa semelhança entre a captura de 5 minutos e de 30 minutos do dispositivo 2. Os outros dispositivos monitorados, diferentemente do dispositivo 2 apresentaram diferenças notáveis ao suas capturas serem comparadas

Para a geração de gráficos I/O, foram utilizados filtros apenas para capturar pacotes TCP e UDP. A partir dos mesmos, conforme Figuras 40, 41, 42, 43, 44, 45, conclui-se que em todos os casos o número de pacotes UDP foi superior aos TCP. A razão para esta característica se deve a comunicação em tempo real promovida pelo *Discord*: os dados necessitam chegar o mais rápido possível sem a sobrecarga dos mesmos, priorizando a rapidez em relação à verificação da integridade dos pacotes. O gráfico de I/O nos mostra a vazão de pacotes de determinada rede, uma vez que é dado o número de pacotes por segundo, sendo assim, quanto maiores se manterem os picos, mais transferências ocorrem.

Visto as saídas de I/O, é possível notar que a rede do dispositivo 1 se encontra instável, uma vez sofre diversos picos e vales. O dispositivo 2 é mais controlado, sofre alguns picos, porém, acaba mantendo-se numa mesma linha. O dispositivo 3 é o que mais apresenta uniformidade na taxa de I/O, pode-se dizer que é o mais estável no momento em que foi monitorado.

Acerca do *Throughput*, exposto na imagem 46, é possível verificar que o dispositivo 1 possui uma vazão de *bits* por segundo maior, quando comparada às outras, isso se dá devido a sua maior velocidade. A rede 4G do dispositivo 2 mostra dificuldade em manter uma grande taxa de vazão, enquanto a rede do dispositivo 3 apresenta uma vazão inconstante, atingindo picos.

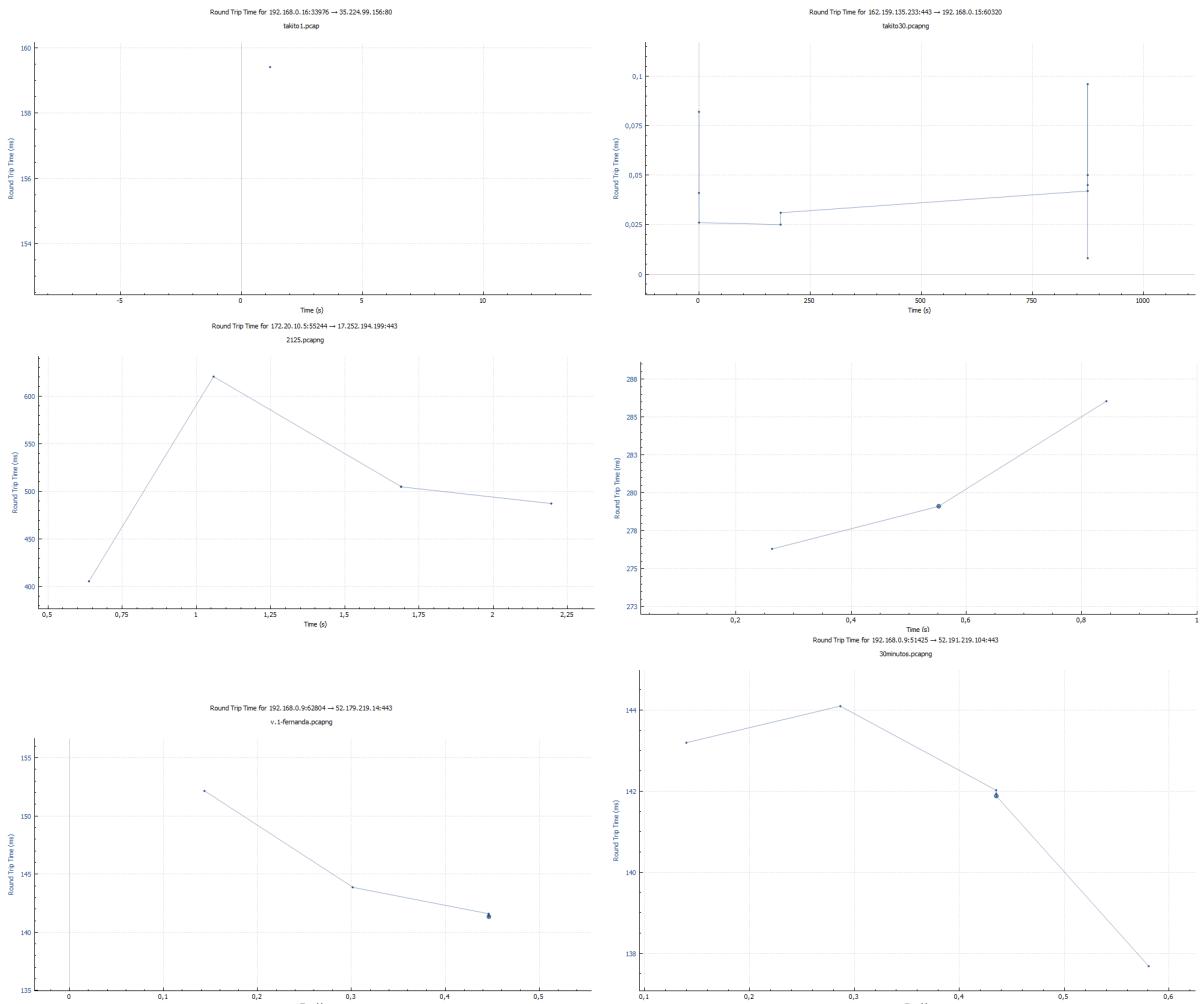


Figura 39 – Gráficos RTT - Da esquerda para direita capturas de 5 e 30 minutos lado a lado. De cima para baixo linha a linha, dispositivos 1, 2 e 3. Fonte: Próprios autores.

7.2.4 Comparação entre capturas do Wireshark e o TCPDump

Comparando os resultados do Wireshark e o TCPDump podemos notar que não existe uma diferença grande com respeito a perda de pacotes, mas sim, existe uma peculiaridade quando observamos os resultados do total de pacotes recebidos.

Em teoria tanto o Wireshark como o TCPDump trabalham da mesma maneira capturando os protocolos, mas por fatores que podem ter sido externos achamos que para o dispositivo numero 1 existe uma discrepância grande nos resultados como pode ser visto na figura 47. Nenhuma explicação factível e baseada em dados foi encontrada, de qualquer maneira podemos levar em consideração os seguintes fatores:

- Para o dispositivo 1, houve uma grande diferença entre as duas ferramentas, entretanto nenhuma explicação com bases nos dados coletados foi encontrada.
- Para o 2, não houve tanta diferença no uso das duas ferramentas;

- Para o 3, houve uma diferença considerável, entretanto a captura foi realizada em uma máquina virtual;

7.2.5 Conclusões

É possível notar a diferença em cada uma das redes analisadas, mas não somente isso, também somadas ao seu local de origem. A rede 4G do dispositivo 2 foi a que teve dificuldades em relação à transmissão de pacotes mesmo possuindo a mediana entre os tamanhos de pacotes. Outro fato interessante a ser notado é a grande diferença, entre o dispositivo 1 com o resto, quando consideramos o *Bytes in flight*, de maneira que representa metade dos números analisados nos outros dispositivos monitorados. Para tornar algo mais relacionado ao que foi feito na seção 7.1, este trabalho utilizou capturas realizadas da maneira descrita na subseção 7.2.2, onde se assemelha ao trabalho (KASSIM et al., 2017). Tanto Discord como Skype obtiveram resultados semelhantes com relação a perda de pacotes. No artigo (KASSIM et al., 2017) a perda de pacote para horários sem pico e usuários em redes diferentes para vídeo em trinta minutos esteve próximo de zero. Resultado este também concluído nesse estudo, como é possível ver na tabela 9, a perda de pacotes para trinta minutos nos três cenários se aproximou de zero.

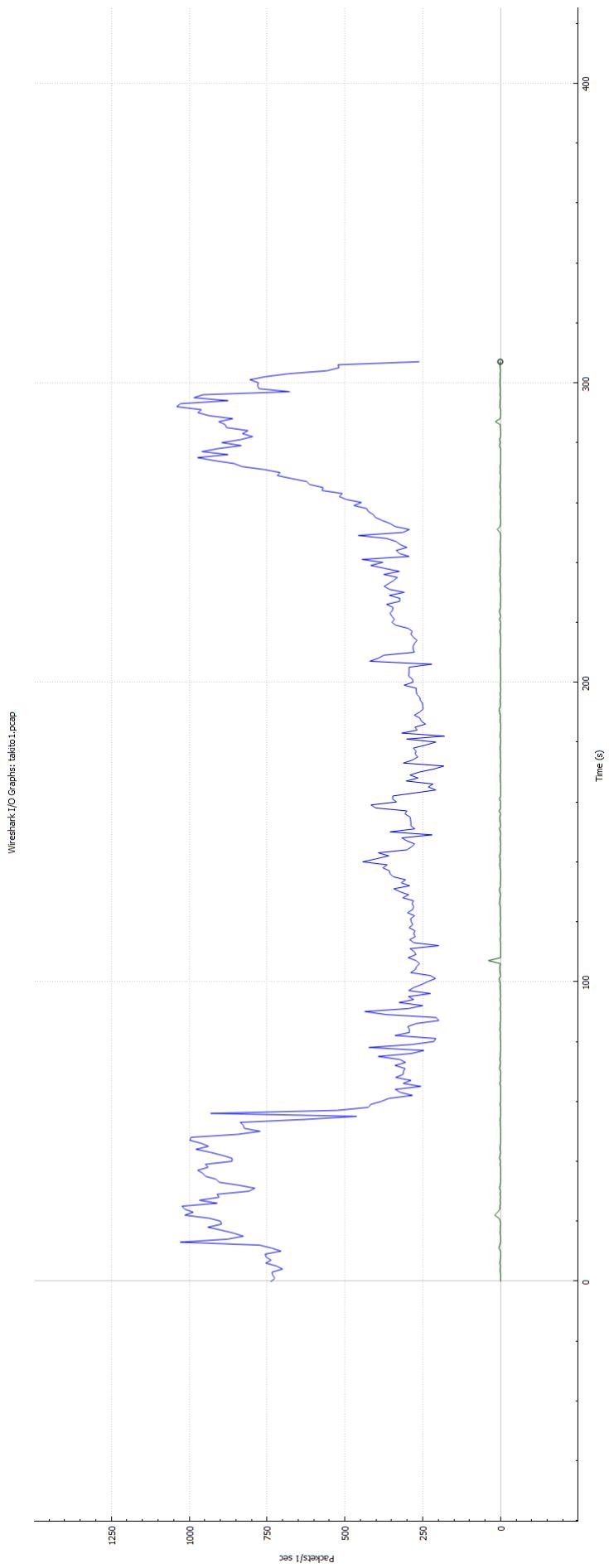


Figura 40 – Gráfico I/O - Dispositivo 1 para capturas de 5 minutos. Fonte: Próprios Autores.

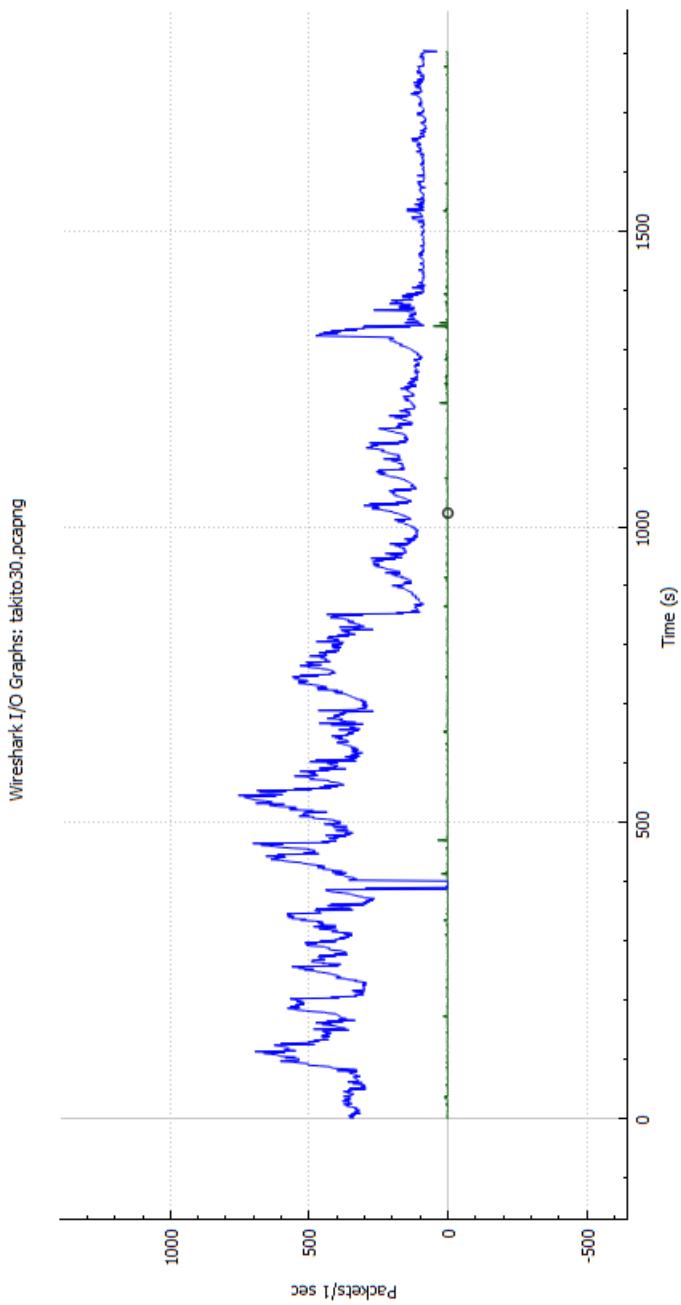


Figura 41 – Gráfico I/O - Dispositivo 1 para capturas de 30 minutos. Fonte: Próprios Autores.

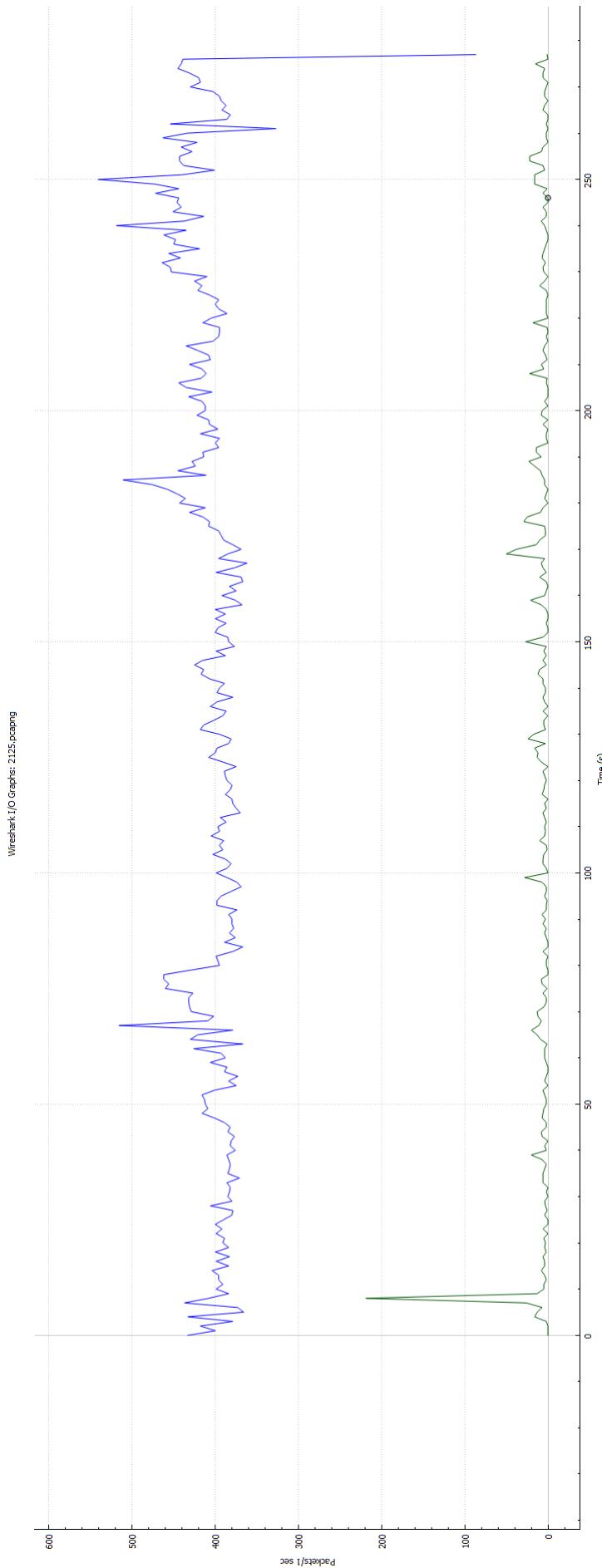


Figura 42 – Gráfico I/O - Dispositivo 2 para capturas de 5 minutos. Fonte: Próprios Autores.

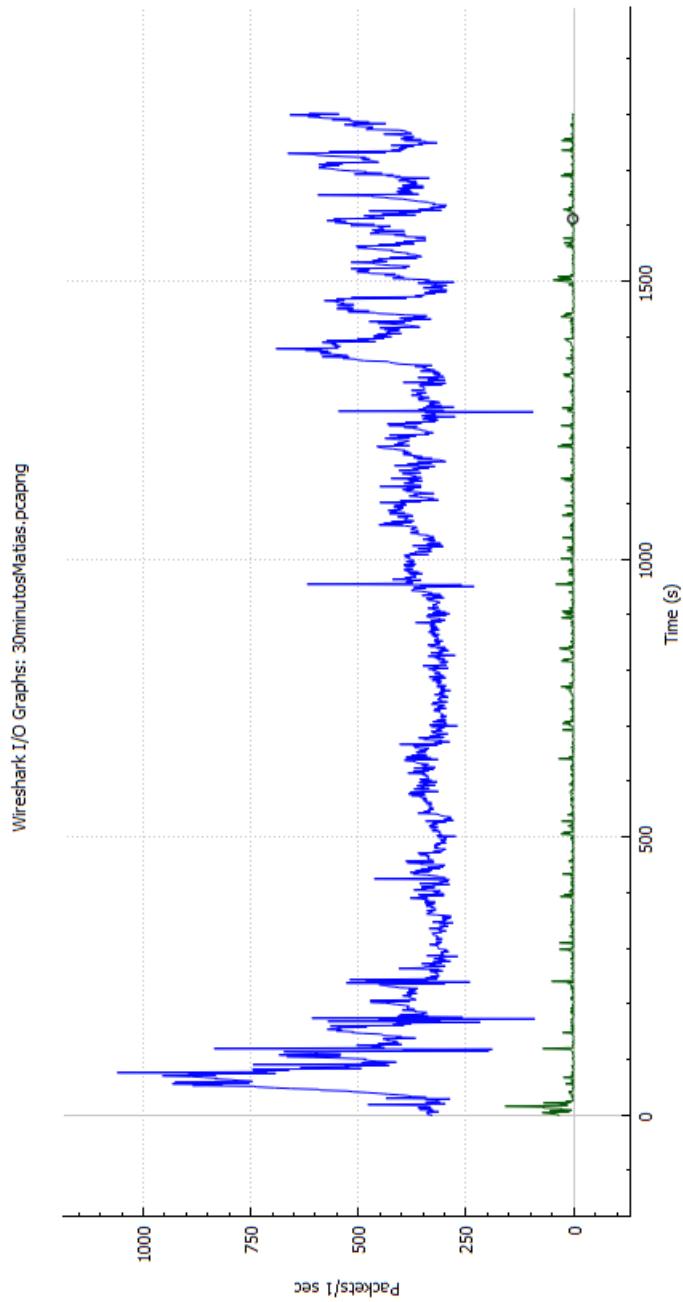


Figura 43 – Gráfico I/O - Dispositivo 2 para capturas de 30 minutos. Fonte: Próprios Autores.

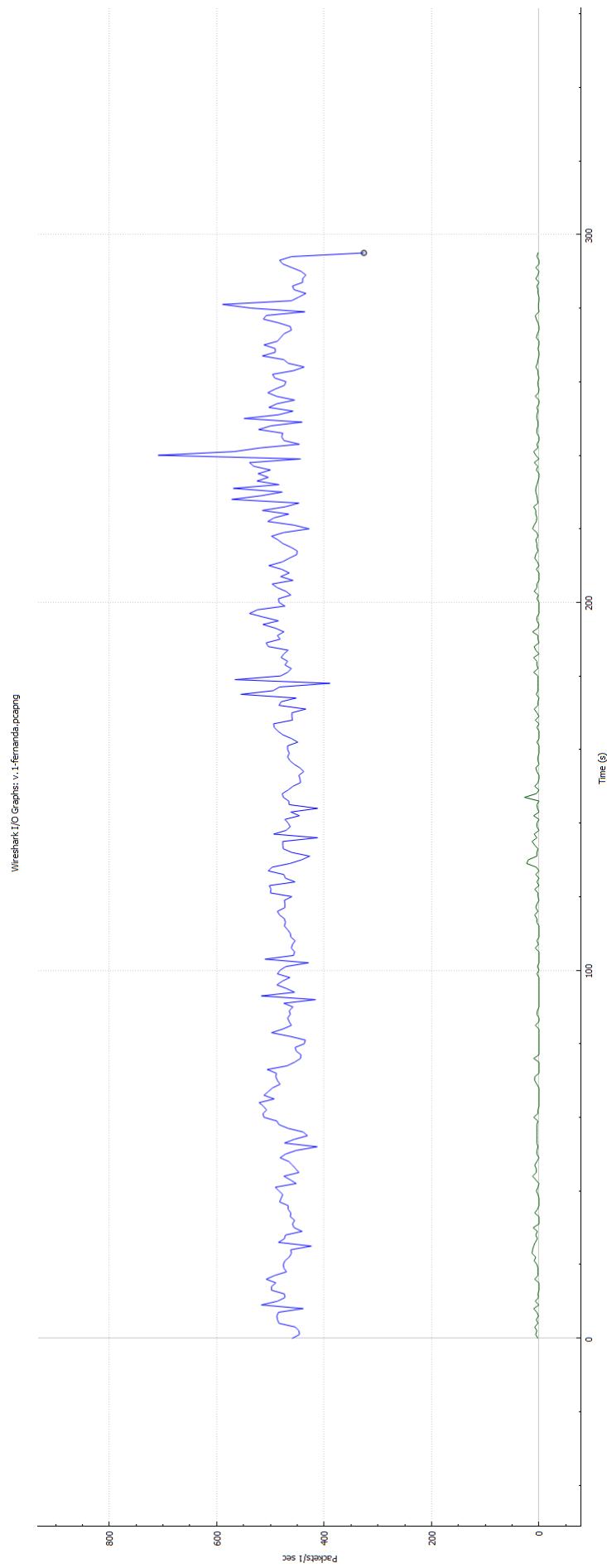


Figura 44 – Gráfico I/O - Dispositivo 3 para capturas de 5 minutos. Fonte: Próprios Autores.

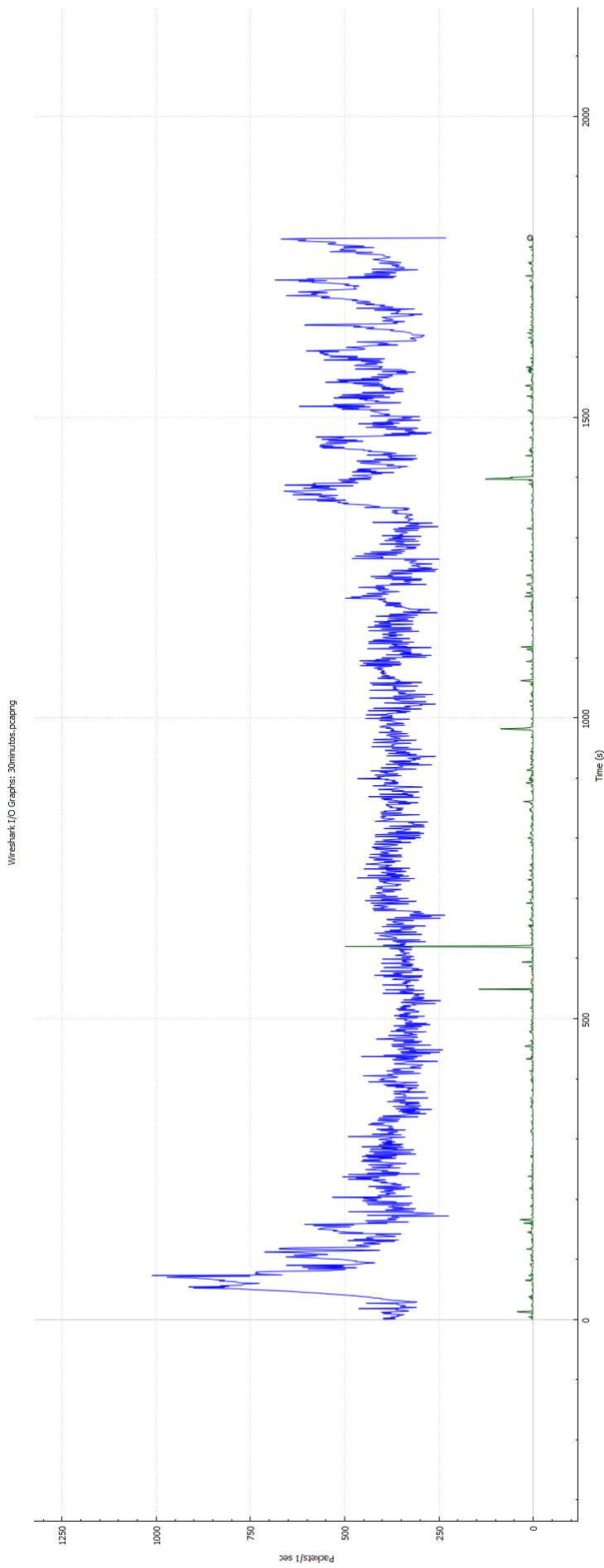


Figura 45 – Gráfico I/O - Dispositivo 3 para capturas de 30 minutos. Fonte: Próprios Autores.

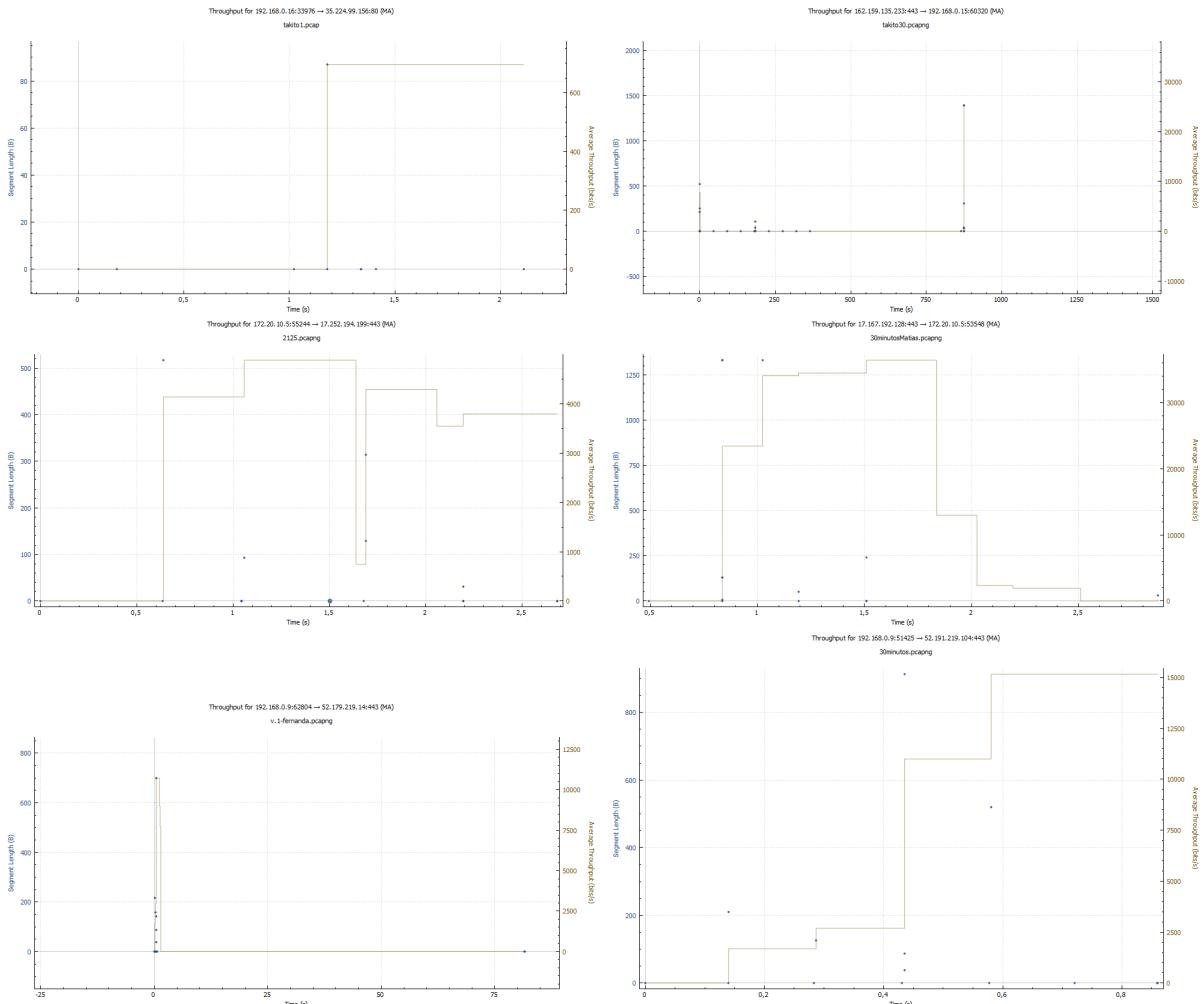


Figura 46 – Gráficos Throughput - Da esquerda para direita capturas de 5 e 30 minutos lado a lado. De cima para baixo linha a linha, dispositivos 1, 2 e 3. Fonte: Próprios autores.

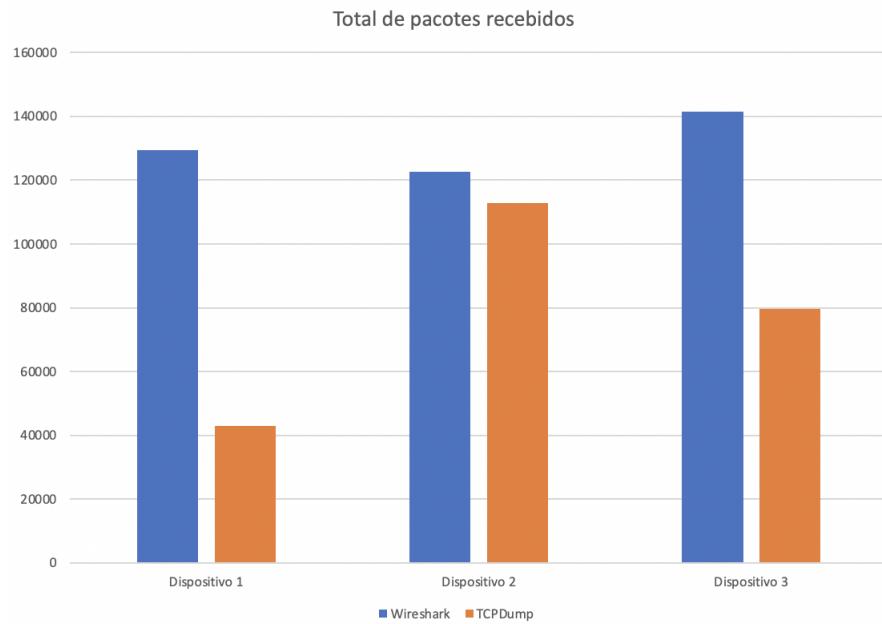


Figura 47 – Fonte: Próprios autores.

8 CONSIDERAÇÕES FINAIS

Desde a sua criação até os dias de hoje, o *Discord* tem se provado uma ferramenta muito eficiente para o que ela se propôs a fazer. Em tempos de isolamento social, tanto o *Skype* como o *Discord* foram grandes aliados das pessoas. Após realizar uma procura na literatura sobre artigos que mencionassem a plataforma *Discord*, nenhum fora encontrado. Devido a isso, decidiu-se realizar um comparativo entre a plataforma *Skype*, que já a anos está consolidada na comunidade, com o *Discord*, uma plataforma que vem ganhando mercado nos últimos anos. A pesquisa utilizou do *Wireshark* e *TCPDump* para analisar o tráfego de pacotes em uma sessão no *Discord*. Após coletados os resultados, verificou-se que em uma das análises o *Discord* e o *Skype* se aproximam de zero pacotes perdidos por segundo. Além disso, após a coleta de dados verificou-se que para um dos dispositivos, a diferença da captura entre as duas ferramentas foi mínima. Já para outro, ela foi maior. O fato do dispositivo 2 ter utilizado uma rede 4G, localizada fora do país, apresentar desempenho semelhante com redes *WiFi* de dentro do Brasil, pode apresentar o *Discord* como uma boa aplicação, uma vez que trata diferentes caminhos de roteamento com desempenho semelhante.

REFERÊNCIAS

- ASRODIA, P.; PATEL, H. **Network Traffic Analysis Using Packet Sniffer.**
- ASRODIA, P.; PATEL, H. Analysis of various packet sniffing tools for network monitoring and analysis. **International Journal of Electrical, Electronics and Computer Engineering**, Citeseer, v. 1, n. 1, p. 55–58, 2012.
- BR, N. de Informação e Coordenação do P. **COVID-19 IMPACTOS NA QUALIDADE DA INTERNET NO BRASIL**. 2020. Disponível em: <<https://www.cepro.br/assets/publicacoes/pdf/2020.07.13-relatorio-semanal.pdf>>.
- CALLSTATS. **What is Round-trip Time and How Does it Relate to Network Latency?** 2018. Disponível em: <<https://www.callstats.io/blog/what-is-round-trip-time-and-how-does-it-relate-to-network-latency>>.
- Chandran, P.; Lingam, C. Performance evaluation of voice transmission in wi-fi networks using r-factor. In: **2015 International Conference on Information Processing (ICIP)**. [S.I.: s.n.], 2015. p. 481–484.
- CHAPPELL, L. **Wireshark Network Analysis (Second Edition): The Official Wireshark Certified Network Analyst Study Guide**. Chapell University, 2012. (Wireshark Solutions). ISBN 9781893939943. Disponível em: <https://books.google.com.br/books/about/Wireshark_Network_Analysis.html?id=x4iaLgEACAAJ&redir_esc=y>.
- CHAPPELL, L.; COMBS, G. **Wireshark 101: Essential Skills for Network Analysis**. Protocol Analysis Institute, Chapell University, 2013. (Wireshark Solutions). ISBN 9781893939721. Disponível em: <<https://books.google.com.br/books?id=FQEZmAEACAAJ>>.
- FAVALE, T. et al. Campus traffic and e-learning during covid-19 pandemic. **Computer Networks**, v. 176, p. 107290, 2020. ISSN 1389-1286. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S1389128620306046>>.
- GITE, V. **TCPDump: Capture and Record Specific Protocols / Port**. 2008. Disponível em: <<https://www.cyberciti.biz/faq/tcpdump-capture-record-protocols-port/>>.
- GOMES, P. C. T. **ENTENDA O QUE É O PROTOCOLO SNMP E SUA IMPORTÂNCIA NO MONITORAMENTO**. 2017. Disponível em: <<https://www.opservices.com.br/snmp/>>.
- GRAHAM-CUMMING, J. **Internet performance during the COVID-19 emergency**. 2020. Disponível em: <<https://blog.cloudflare.com/recent-trends-in-internet-traffic/>>.
- KASSIM, M. et al. Skype multimedia application traffic analysis on home unifi network. In: . [S.I.: s.n.], 2017. p. 184–189.
- Korhonen, J.; Wang, Y. Effect of packet size on loss rate and delay in wireless links. In: **IEEE Wireless Communications and Networking Conference, 2005**. [S.I.: s.n.], 2005. v. 3, p. 1608–1613 Vol. 3.

LYON, G. **Top 125 Network Security Tools**. 2019. Disponível em: <<https://sectools.org/>>.

MALLET, Q. **What is the difference between Tcpdump and Wireshark?** 2017. Disponível em: <<https://www.quora.com/What-is-the-difference-between-Tcpdump-and-Wireshark>>.

MEHDI, Y. **Introducing the new Microsoft 365 Personal and Family subscriptions**. 2020. Disponível em: <<https://www.microsoft.com/en-us/microsoft-365/blog/2020/03/30/introducing-new-microsoft-365-personal-family-subscriptions>>.

NELLY. **Discord Transparency Report: Jan — June 2020**. 2020. Disponível em: <<https://blog.discord.com/discord-transparency-report-jan-june-2020-2ef4a3ee346d>>.

NICOLETTI, M.; BERNASCHI, M. Forensic analysis of microsoft skype for business. **Digital Investigation**, Elsevier, v. 29, p. 159–179, 2019.

OLUDELE, A. et al. Packet sniffer – a comparative characteristic evaluation study. In: . [S.I.: s.n.], 2015. p. 091–100.

SURI, S.; BATRA, V. Comparative study of network monitoring tools. **International Journal of Innovative Technology and Exploring Engineering (IJITEE)**, Citeseer, v. 1, n. 3, p. 63–65, 2010.

WARRAG, T. **Why is Wireshark useful?** 2016. Disponível em: <<https://www.quora.com/Why-is-Wireshark-useful>>.

WIRESHARK. 2020. Disponível em: <<https://1.as.dl.wireshark.org/docs/user-guide.pdf>>.

WIRESHARK. **Awards and Accolades**. 2020. Disponível em: <<https://www.wireshark.org/about.html#authors>>.

WIRESHARK FAQ. Disponível em: <<https://ask.wireshark.org/questions/>>.