Autor: [Pedro Daniel Gonçalves Antunes]

Data: [2025/02/24]

Curso/Disciplina: [LSIRC/PPR]

Instituição: [ESTG]

NIM:[8230068]

Ferramentas de mapeamento de redes e de análise de performance da rede

1) Procure na Internet por ferramentas, de uso livre, de mapeamento de redes. O objetivo é executar as ferramentas que encontrar (2 ou 3, no máximo), e perceber as suas aptidões em termos de desenho do mapa/topologia da rede, e (possívemente) de análise de tráfego.

- Open-Audit
- Netsurveyor Wi-Fi scanner
- Advanced IP Scanner
- Overlook
- Spiceworks
- Cheops-ng
- OpenNMS
- NetworkView
- Nmap
- Angry IP Scanner
- EtherApe

(http://rayslinux.blogspot.pt/2015/02/10-handy-tools-for-network-discovery.html)

Exercício 1

1. Introdução
   Para este exercício, foi proposta a escolha de **duas ou três ferramentas de mapeamento de redes**. O objetivo deste teste é avaliar ferramentas gratuitas para compreender as suas capacidades na **criação de mapas/topologias de rede e na análise de tráfego**.

   Escolhi o **Nmap** e o **Angry IP Scanner**. O motivo da escolha deve-se ao facto de o **Nmap** ser amplamente utilizado na área da **segurança informática**, especialmente para **varrimento de redes**, enquanto o **Angry IP Scanner** é conhecido pela sua **simplicidade e rapidez** na detecção de dispositivos conectados.
2. Características das Ferramentas
   Nmap

- O **Nmap (Network Mapper)** trata-se de uma ferramenta capaz de detectar os serviços e computadores de uma determinada rede, criando um tipo de mapa de rede.
- Permite verificar dispositivos conectados, portas abertas e serviços em execução.
- Pode ser utilizado via linha de comandos ou com a interface gráfica Zenmap.
- Suporta funcionalidades avançadas, como a detecção do sistema operativo e análise de vulnerabilidades.
- É amplamente usada na segurança informática para testes de intrusão e auditorias de rede.
- Fonte: Site do nmap | PMG Academy

Angry IP Scanner

- O Angry IP Scanner é uma ferramenta simples e rápida para descobrir dispositivos conectados numa rede.
- Tem uma interface gráfica simples de usar.
- Permite fazer um scan dos endereços de IP e portas abertas de forma eficiente.
- Suporta exportação dos resultados para CSV, TXT, XML e outros formatos.
- Ideal para utilizadores menos experientes ou para quem precisa de uma solução rápida sem configurações muito complexas.
- Font: Site do Angry IP Scanner

3. Testes práticos e Resultados.
   Para que fosse possível realizar os testes das ferramentas, utilizei uma VM (Virtual Machine) com o kali Linux instalado e procedi aos testes das ferramentas descritas anteriormente.
   3.1 Teste com Nmap
   - Instalando o Nmap e abrindo o terminal, executando o comando seguinte:

```
nmap -h
```

Foi possível visualizar todos os comandos possíveis que o nmap faz (print abaixo).

```
└─$ nmap -h
Nmap 7.94SVN ( https://nmap.org )
Usage: nmap [Scan Type(s)] [Options] {target specification}
TARGET SPECIFICATION:
  Can pass hostnames, IP addresses, networks, etc.
  Ex: scanme.nmap.org, microsoft.com/24, 192.168.0.1; 10.0.0-255.1-254
  -iL <inputfilename>: Input from list of hosts/networks
  -iR <num hosts>: Choose random targets
  --exclude <host1[,host2][,host3],...>: Exclude hosts/networks
  --excludefile <exclude_file>: Exclude list from file
HOST DISCOVERY:
  -sL: List Scan - simply list targets to scan
  -sn: Ping Scan - disable port scan
  -Pn: Treat all hosts as online -- skip host discovery
  -PS/PA/PU/PY[portlist]: TCP SYN/ACK, UDP or SCTP discovery to given ports
  -PE/PP/PM: ICMP echo, timestamp, and netmask request discovery probes
  -PO[protocol list]: IP Protocol Ping
  -n/-R: Never do DNS resolution/Always resolve [default: sometimes]
  --dns-servers <serv1[,serv2],...>: Specify custom DNS servers
  --system-dns: Use OS's DNS resolver
  --traceroute: Trace hop path to each host
SCAN TECHNIQUES:
  -sS/sT/sA/sW/sM: TCP SYN/Connect()/ACK/Window/Maimon scans
  -sU: UDP Scan
  -sN/sF/sX: TCP Null, FIN, and Xmas scans
  --scanflags <flags>: Customize TCP scan flags
  -sI <zombie host[:probeport]>: Idle scan
  -sY/sZ: SCTP INIT/COOKIE-ECHO scans
  -sO: IP protocol scan
  -b <FTP relay host>: FTP bounce scan
PORT SPECIFICATION AND SCAN ORDER:
  -p <port ranges>: Only scan specified ports
    Ex: -p22; -p1-65535; -p U:53,111,137,T:21-25,80,139,8080,S:9
  --exclude-ports <port ranges>: Exclude the specified ports from scanning
  -F: Fast mode - Scan fewer ports than the default scan
  -r: Scan ports sequentially - don't randomize
  --top-ports <number>: Scan <number> most common ports
  --port-ratio <ratio>: Scan ports more common than <ratio>
SERVICE/VERSION DETECTION:
  -sV: Probe open ports to determine service/version info
  --version-intensity <level>: Set from 0 (light) to 9 (try all probes)
  --version-light: Limit to most likely probes (intensity 2)
  --version-all: Try every single probe (intensity 9)
  --version-trace: Show detailed version scan activity (for debugging)
SCRIPT SCAN:
  -sC: equivalent to --script=default
  --script=<Lua scripts>: <Lua scripts> is a comma separated list of
           directories, script-files or script-categories
  --script-args=<n1=v1,[n2=v2,...]>: provide arguments to scripts
  --script-args-file=filename: provide NSE script args in a file
  --script-trace: Show all data sent and received
  --script-updatedb: Update the script database.
  --script-help=<Lua scripts>: Show help about scripts.
           <Lua scripts> is a comma-separated list of script-files or
           script-categories.
OS DETECTION:
  -O: Enable OS detection
  --osscan-limit: Limit OS detection to promising targets
```

- Podemos assim verificar que o Nmap tem muitas funcionalidades (como descrito anteriormente), desde varrimento de portas abertas, detecção do sistema operativo etc..

- Para isso, foi feito um teste prático para verificar como a ferramenta se comporta no "mundo real", executando o comando seguinte:

```
$ nmap -A -T4 scanme.nmap.org
```

- - `-A` → Ativa a **deteção de sistema operativo** e serviços.
- `-T4` → Ajusta a **velocidade do scan** para um nível rápido.
- O Nmap, por defeito, faz um **SYN Scan (stealth scan)** nas **1000 portas mais comuns**.

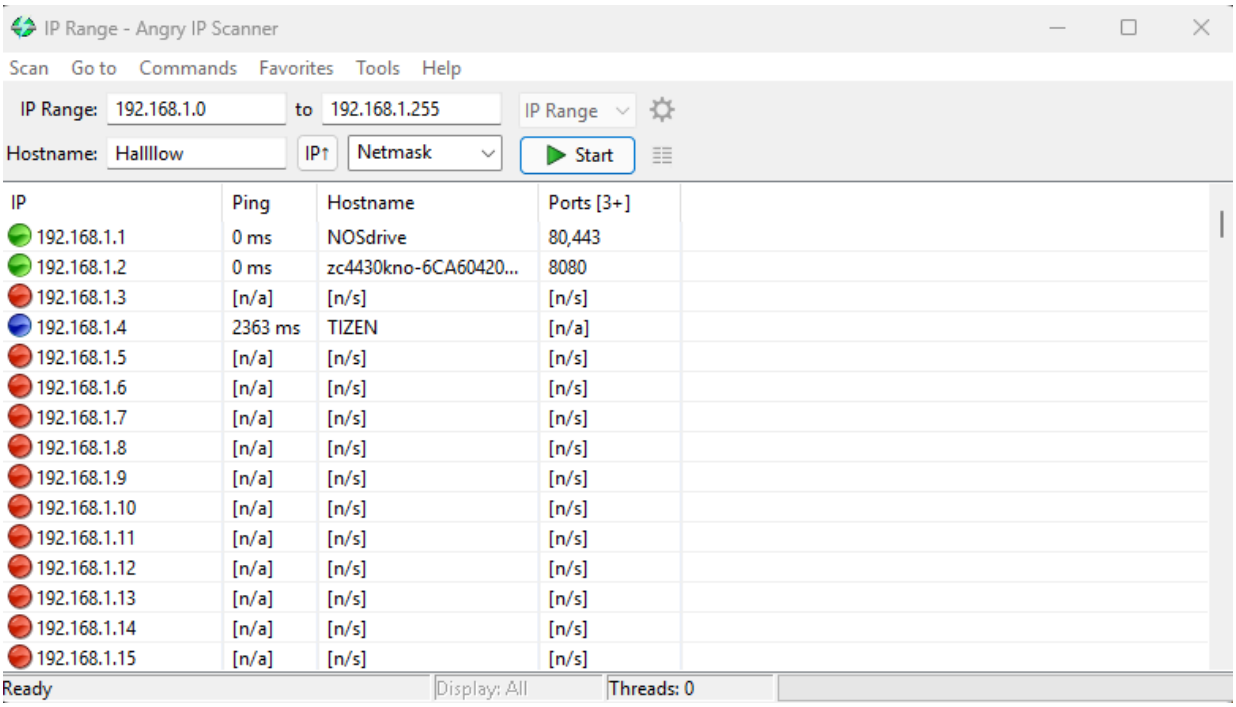
```
# nmap -A -T4 scanme.nmap.org

Nmap scan report for scanme.nmap.org (74.207.244.221)
Host is up (0.029s latency).
rDNS record for 74.207.244.221: li86-221.members.linode.com
Not shown: 995 closed ports
PORT     STATE    SERVICE     VERSION
22/tcp   open     ssh         OpenSSH 5.3p1 Debian 3ubuntu7 (protocol 2.0)
| ssh-hostkey: 1024 8d:60:f1:7c:ca:b7:3d:0a:d6:67:54:9d:69:d9:b9:dd (DSA)
|_2048 79:f8:09:ac:d4:e2:32:42:10:49:d3:bd:20:82:85:ec (RSA)
80/tcp   open     http        Apache httpd 2.2.14 ((Ubuntu))
|_http-title: Go ahead and ScanMe!
646/tcp  filtered ldp
1720/tcp filtered H.323/Q.931
9929/tcp open     nping-echo  Nping echo
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6.39
OS details: Linux 2.6.39
Network Distance: 11 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:kernel

TRACEROUTE (using port 53/tcp)
HOP RTT      ADDRESS
[Cut first 10 hops for brevity]
11  17.65 ms li86-221.members.linode.com (74.207.244.221)

Nmap done: 1 IP address (1 host up) scanned in 14.40 seconds
```

📌 **Análise dos resultados:**

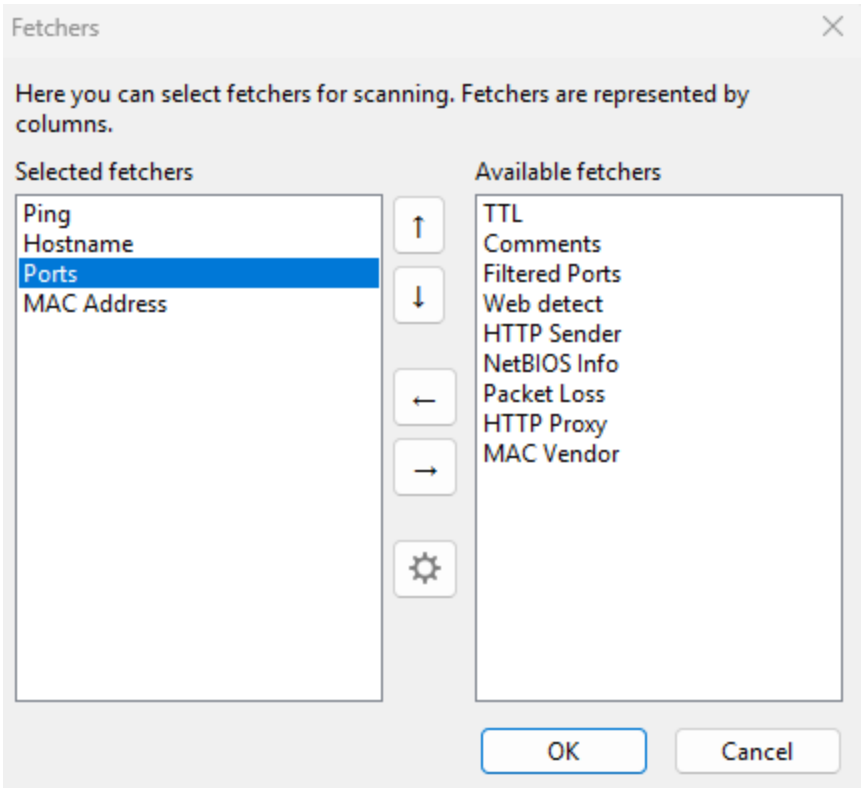
- O **scan revelou que a porta 22 (SSH) e 80 (HTTP) estavam abertas**, identificando também as suas versões.
- Foi possível determinar que o sistema operativo do alvo é **Linux**.
- Fonte: site

3.2 Testes com Angry IP Scanner

- O scan foi realizado diretamente no **Windows**, sem necessidade de máquina virtual.
- **Passos seguidos**:
  1. Defini o intervalo de IPs: **192.168.1.0 a 192.168.1.255**.
  2. Ativei a opção de scan de **portas abertas e hostname**.
  3. Iniciei a varredura e aguardei os resultados.



- 📌 **Análise dos resultados:**
- O scan encontrou **vários dispositivos ativos** na rede.
- O **IP 192.168.1.1** foi identificado como **router** com portas **80 e 443 abertas**.
- O **IP 192.168.1.2** tinha a porta **8080 aberta**, sugerindo um serviço HTTP.
- O **IP 192.168.1.4** tinha hostname **TIZEN**, possivelmente uma **Smart TV Samsung**.
- Também podes adicionar várias features que podemos ver com, como por exemplo MAC address de cada dispositivo, o MAC vendor, portas filtradas etc...



- Fonte: Site

4. Comparação entre Nmap e Angry IP Scanner

Critério	Nmap	Angry IP Scanner
Facilidade de uso	Difícil (CLI) / Médio (GUI)	Fáci
Velocidade	Médio	Rápido
Detalhe da Análise	Muito detalhado (SO, Serviços, firewall, etc..)	Básicos (IPs, Portas abertas, MAC address, etc..)
Exportação de dados	Sim	Sim
Melhor para	Profissionais de segurança	Utilizadores casuais

📌 **Conclusão da Comparação**:

- O **Nmap** é mais avançado e fornece **mais detalhes**, sendo ideal para auditorias de segurança.
- O **Angry IP Scanner** é mais **simples e rápido**, ideal para quem só precisa de um scan rápido.
- Para **análise detalhada da rede**, o **Nmap** é a melhor opção.
- Para **um scan rápido e visual**, o **Angry IP Scanner** é mais prático.

5. Conclusão Final

> Após testar ambas as ferramentas, concluo que o **Nmap** e o **Angry IP Scanner** têm propósitos diferentes.
>
> - O **Nmap** é uma ferramenta poderosa, utilizada por profissionais de segurança para análise aprofundada da rede.
> - O **Angry IP Scanner** é uma solução rápida e eficiente para detetar dispositivos ativos e verificar portas abertas.
>
> Dependendo do objetivo, ambas as ferramentas podem ser úteis: para segurança e testes avançados, o **Nmap** é a melhor escolha; para uma verificação simples e visual, o **Angry IP Scanner** é mais prático.

---

2) Ferramentas de análise de performance da rede – netperf. Defina algumas métricas para análise da performance da rede

http://www.programering.com/a/MjN3IDNwATc.html
https://www.clustermonkey.net/Cluster-Newbie/measuring-network-performance/Page-2.html
http://www.cs.kent.edu/~farrell/dist/ref/Netperf.html

2.1) Considere uma rede constituída por dois computadores com OS Linux, conectados diretamente (tenha por base 2 VMs na sua máquina física). Utilize a ferramenta netperf para executar alguns testes de performance (TCP e UDP), para tráfego bulk (e.g., FTP, partilha de ficheiros na rede, ...). Considere diferentes tamanhos de pacotes (opção -m), para um tempo de teste igual a 60 segundos. O que verifica?

📌 Exercício 2.1 – Análise de Performance da Rede com Netperf

1️⃣ Definição de Métricas para Análise de Performance

O **Netperf** é uma ferramenta amplamente utilizada para avaliar o desempenho de redes **TCP e UDP**. Para a análise de performance da rede, podemos considerar as seguintes métricas:

- **Throughput (Taxa de Transferência de Dados):** Mede a quantidade de dados que pode ser transmitida por segundo (**Mbps**).
- **Latência (RTT - Round Trip Time):** Mede o tempo que um pacote leva para viajar da origem até ao destinatário e voltar.
- **Jitter:** Mede a variação na latência dos pacotes, sendo crucial para **chamadas VoIP e streaming de vídeo**.
- **Eficiência do Tamanho do Pacote:** Mede o impacto do **tamanho do pacote** no desempenho da transmissão.

📌 **Fonte**: Nilesecure.com

2️⃣ Testes de Performance Utilizando Netperf

Para testar a performance da rede, consideramos um cenário com **duas máquinas Linux** conectadas diretamente, podendo ser duas **VMs (máquinas virtuais)**.
No servidor, iniciámos o Netserver:

```
netserver
```

- Isto ativou o **Netperf Server**, que ficou à escuta de pedidos de teste.

3️⃣ Teste de Largura de Banda TCP

O primeiro teste que vamos fazer é medir **o throughput (taxa de transferência de dados) em TCP**, que simula **transferências de ficheiros** (exemplo: FTP, partilha de rede, etc.).

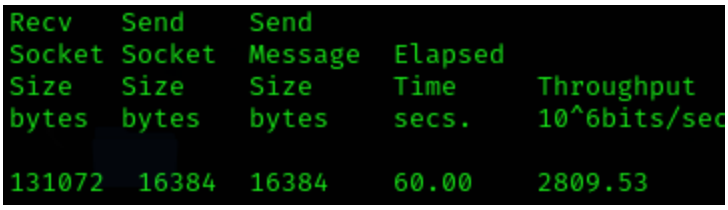
O cliente executa o seguinte comando:

```
netperf -H 192.168.1.100 -t TCP_STREAM -l 60
```

📌 Explicação do comando:

- -H 192.168.1.100 → O IP do servidor onde está a correr o Netserver.
- -t TCP_STREAM → Faz um teste de largura de banda usando TCP.
- -l 60 → O teste dura 60 segundos.

✅ Resultado esperado:



- O Netperf enviou pacotes TCP durante 60 segundos
- O throughput (largura de banda medida) foi de 2809.35 Mbps (2.8 Gbps).
- Isto mostra que a comunicação entre as VMs é muito rápida, fazendo sentido pois elas estão a correr dentro da mesma máquina física.

4️⃣ Teste de Desempenho UDP

Agora, vamos testar **o desempenho em UDP**, que é usado em **streaming de vídeo, VoIP, jogos online, etc.**.

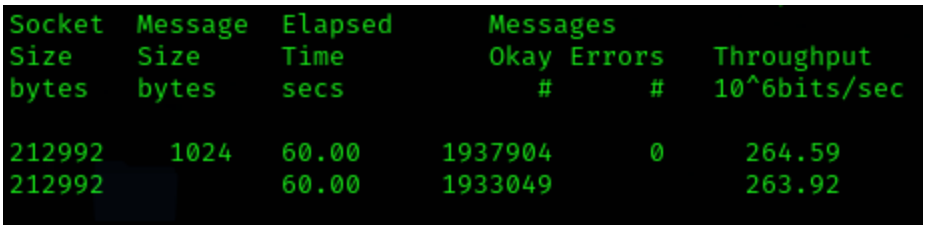
O cliente executa o seguinte comando:

```
netperf -H 192.168.1.100 -t UDP_STREAM -l 60 -- -m 1024
```

📌 **Explicação do comando:**

- `-t UDP_STREAM` → Mede a **performance em UDP**.
- `-- -m 1024` → Define o **tamanho do pacote** para **1024 bytes**.

✅ **Resultado esperado:**



- O Neperf enviou pacotes UDP de 1024 bytes durante 60 segundos.
- O throughput (largura de banda medida) foi de ~264.59 Mbps
- Número de mensagens enviadas: cerca de 1.93 milhões de pacotes
- Número de erros: 0 -> não houve perdas de pacotes neste teste.

5️⃣ Comparação TCP vs UDP

Protocolo	Througthput(Mbs)	Perda de Pacotes	Mensagens Tamanho (Bytes)
TCP	2809.53 Mbps	0% (Garante entrega)	16384
UDP	264.59 Mbps	0% (Neste teste)	1024

📌 Conclusões:

- **O TCP tem um throughput muito maior (~2.8 Gbps vs 264 Mbps)** porque usa **mecanismos de controlo de fluxo e congestionamento**.
- **O UDP é muito mais lento**, mas isso é esperado porque **não tem controle de erro ou retransmissão** como o TCP.
- **O tamanho das mensagens (1024 vs 16384) pode influenciar o desempenho**, pois pacotes maiores podem ser mais eficientes no transporte de dados.

6️⃣ Teste Variando o Tamanho dos Pacotes UDP

Agora, vamos testar como **o tamanho dos pacotes influencia o desempenho**.

No cliente foram executados os seguintes comandos:

```
netperf -H 192.168.1.100 -t UDP_STREAM -l 60 -- -m 512
netperf -H 192.168.1.100 -t UDP_STREAM -l 60 -- -m 2048
```

✅ Resultado esperado:

```
└─$ netperf -H 192.168.1.100 -t UDP_STREAM -l 60 -- -m 512
MIGRATED UDP STREAM TEST from 0.0.0.0 (0.0.0.0) port 0 AF_INET to 192.168.1.100 () port 0 AF_INET : demo
Socket  Message  Elapsed      Messages
Size    Size     Time         Okay Errors   Throughput
bytes   bytes    secs            #      #   10^6bits/sec

212992     512   60.00     2011857      0     137.34
212992           60.00     2005557            136.91
```

- **Throughput: ~137 Mbps**
- **Número de mensagens enviadas: ~2 milhões**
- **Erros: 0**

📌 Conclusões

- **Quanto menor o pacote, menor o throughput.**
  - O **throughput caiu de 264 Mbps (1024 bytes) para 137 Mbps (512 bytes)**.
  - Isso acontece porque **há mais overhead** (cada pacote tem cabeçalhos, e enviar mais pacotes pequenos significa mais desperdício de espaço).
- **Número de pacotes aumentou** porque cada mensagem é menor, então mais pacotes foram necessários para transmitir os dados.

✅ **Resultado esperado:**

```
┌──(kali㉿kali)-[~]
└─$ netperf -H 192.168.1.100 -t UDP_STREAM -l 60 -- -m 2048
MIGRATED UDP STREAM TEST from 0.0.0.0 (0.0.0.0) port 0 AF_INET to 192.168.1.100 () port 0 AF_INET : demo
Socket  Message  Elapsed      Messages
Size    Size     Time         Okay Errors   Throughput
bytes   bytes    secs            #      #   10^6bits/sec

212992    2048   60.00     1103406      0     301.30
212992           60.00     1101006            300.65
```

- **Throughput: ~301 Mbps**
- **Número de mensagens enviadas: ~1.1 milhões**
- **Erros: 0** (nenhuma perda de pacotes)

📈 Comparação de Todos os Testes UDP

Tamanho do Pacote (Bytes)	Throughput (Mbps)	Mensagens Enviadas	Erros
512	137 Mbps	~2.0M	0
1024	264 Mbps	~1.93M	0
2048	301 Mbps	~1.1M	0

📌 **Conclusões:**

1. **Pacotes maiores aumentam a largura de banda (throughput)** porque há **menos overhead** nos cabeçalhos dos pacotes.
2. **Pacotes pequenos (512 bytes) geram mais pacotes enviados, mas menor throughput** devido ao overhead.
3. **O tamanho de 2048 bytes foi o mais eficiente em UDP**, alcançando **301 Mbps**, que é **14% mais rápido que 1024 bytes e mais do que o dobro de 512 bytes**.

📌 Conclusão Final

> O TCP foi **muito mais rápido** que o UDP, atingindo **2809 Mbps**, enquanto o UDP, mesmo com pacotes grandes, **não passou de 301 Mbps**.
>
> O tamanho dos pacotes em **UDP influencia diretamente o desempenho**, com pacotes **maiores resultando em um throughput mais alto**.
>
> O **TCP** é mais indicado para **transferências de arquivos e comunicação confiável**, enquanto o **UDP** é melhor para **streaming, VoIP e jogos online**, onde a velocidade é mais importante do que a confiabilidade.

📌 Exercício 2.3 Testes Usando Rede Sem Fios (Wi-Fi Simulado)

Após os testes na rede cabeada, realizámos **os mesmos testes em uma rede Wi-Fi simulada** dentro do VirtualBox. Para isso, configurámos as VMs para utilizar **"Rede Interna"**, simulando uma **conexão sem fios**.

📌 Configuração da Rede

- Ambas as VMs foram configuradas com **modo "Rede Interna"** no VirtualBox, como fiz o exercício em casa não foi possível utilizar outro computador para tal.
- Atribuímos manualmente os seguintes **endereços IP**:
  - **Servidor:** `192.168.2.10`
  - **Cliente:** `192.168.2.20`
- O **Netserver** foi iniciado no servidor com:
-

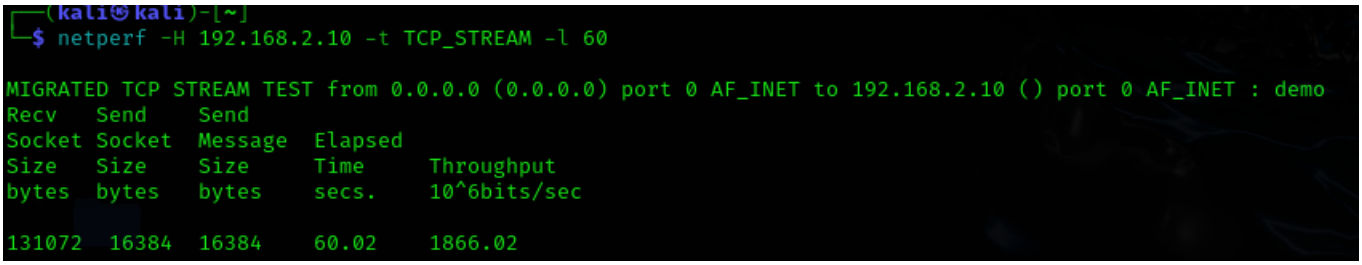
```
netserver
```

📊 Análise do Resultado TCP

- Depois de preparamos o ambiente simulado de uma rede Wi-fi executamos os seguinte comando na máquina do cliente:

```
netperf -H 192.168.2.10 -t TCP_STREAM -l 60
```

✅ **Resultado esperado:**



📌 **Resultado Obtido no Teste TCP:**

- **Throughput TCP: 1866.02 Mbps**
- **Duração do Teste: 60 segundos**
- **Comparação com Cabo:** O TCP na rede cabeada era **2809 Mbps**, agora caiu para **1866 Mbps**.

📌 **O que isto significa?**

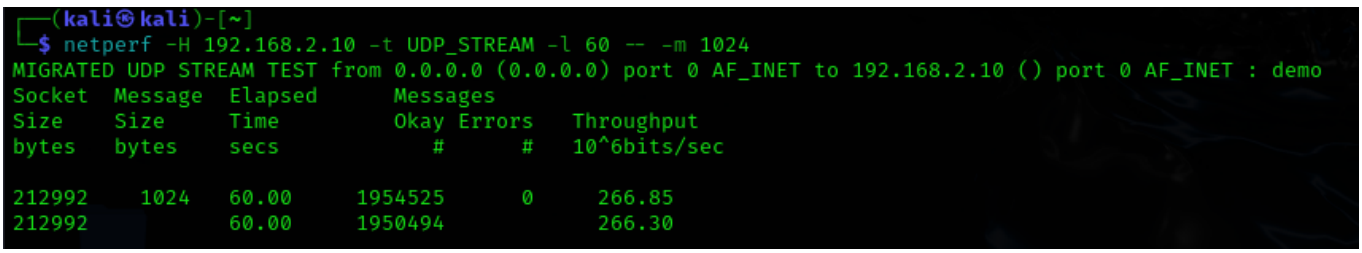
- O **throughput TCP foi reduzido** na rede Wi-Fi simulada.
- A diferença não é muito grande, porque a simulação dentro do VirtualBox **não consegue replicar 100% a interferência e perdas de sinal do Wi-Fi real**.
- Mas **já é possível ver que a rede sem fios tem menos desempenho** do que a rede cabeada.

📊 Análise do Resultado UDP

- Agora executando os comandos mas para UDP, executamos os seguintes comandos:

```
netperf -H 192.168.2.10 -t UDP_STREAM -l 60 -- -m 1024
```

✅ **Resultado esperado:**



📌 **Resultados Obtidos:**

- **Throughput UDP: ~266 Mbps**
- **Número de mensagens enviadas: ~1.95 milhões**
- **Erros: 0 (Nenhuma perda de pacotes)**

📌 **Comparação com o Teste UDP na Rede Cabeada:**

Rede	Throughput (Mbps)	Mensagens Enviadas	Perda de Pacotes
Cabo	264 Mbps	~1.93M	0%
Wi-fi Simulado	266 Mbps	~1.95	0%

📌 **Conclusões:**

- O **throughput do UDP na rede Wi-Fi simulada ficou praticamente igual ao cabo**.
- **Não houve perda de pacotes** neste teste, o que significa que a rede simulada dentro do VirtualBox **ainda tem boa estabilidade**.
- Como a simulação no VirtualBox **não reproduz interferências reais do Wi-Fi**, o desempenho é muito próximo ao cabo.

📌 Conclusão Final

Após a realização dos testes comparativos entre **rede cabeada e rede Wi-Fi simulada**, foi possível observar que **o desempenho da rede sem fios foi inferior ao cabo, mas sem perdas de pacotes**.

📌 **Principais Observações:**

- **O TCP teve uma redução de throughput de 2809 Mbps para 1866 Mbps**, indicando que a rede Wi-Fi simulada **introduz algum overhead e reduz a velocidade de transmissão**.
- **O UDP manteve um throughput muito próximo ao cabo (~266 Mbps)**, sem registo de perda de pacotes.
- **A simulação no VirtualBox não introduziu interferências reais do Wi-Fi**, tornando os valores muito próximos aos da rede cabeada.

📌 **O que aprendemos?**

- **Redes cabeadas são mais eficientes e estáveis**, especialmente para **comunicações TCP** que exigem **alta confiabilidade**.
- **O UDP pode manter um desempenho semelhante ao cabo** em redes Wi-Fi bem configuradas, mas **em redes reais pode sofrer interferências** e resultar em perda de

pacotes.

- **Mesmo em redes sem fios simuladas, é possível observar que o TCP sofre uma leve degradação no throughput** devido ao comportamento diferente da comunicação sem fios.

✅ **Conclusão Final:** Os testes demonstraram que, dentro de um ambiente controlado como o VirtualBox, **as diferenças entre cabo e Wi-Fi simulados não são muito expressivas**. No entanto, **em uma rede Wi-Fi real, espera-se maior latência, menor throughput e possibilidade de perda de pacotes**. Isso reforça a importância de escolher **o tipo de rede adequado para cada tipo de aplicação**.

📌 Exercício 2.4 – Testes de Tráfego Transacional (TCP_RR vs UDP_RR)

Neste teste, analisámos o desempenho da rede simulando um **cenário de tráfego transacional**, que ocorre em aplicações como **bases de dados, consultas HTTP, serviços de DNS e aplicações cliente-servidor**.

Para isso, utilizámos os modos **TCP_RR (TCP Request-Response)** e **UDP_RR (UDP Request-Response)** do Netperf, que medem a eficiência de transações curtas.

📌 Configuração do Ambiente

- As VMs foram configuradas com **"Rede Interna"**, simulando um ambiente **Wi-Fi**.
- O **Netserver** foi iniciado no servidor com:

```
netserver
```

- Os testes foram realizados na **rede Wi-Fi simulada** do VirtualBox.

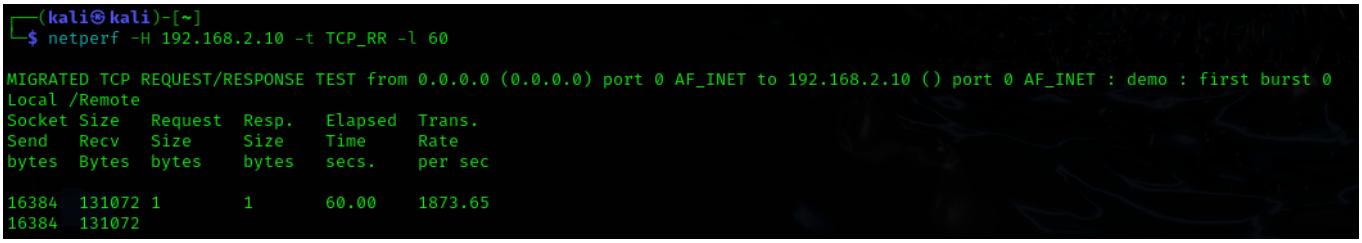
📊 Teste TCP_RR – Comunicação Transacional com TCP

O primeiro teste foi realizado utilizando **TCP_RR**, onde um cliente envia uma pequena requisição e recebe uma resposta maior.

📌 **Comando executado no cliente:**

```
netperf -H 192.168.2.10 -t TCP_RR -l 60
```

✅ **Resultado obtido:**



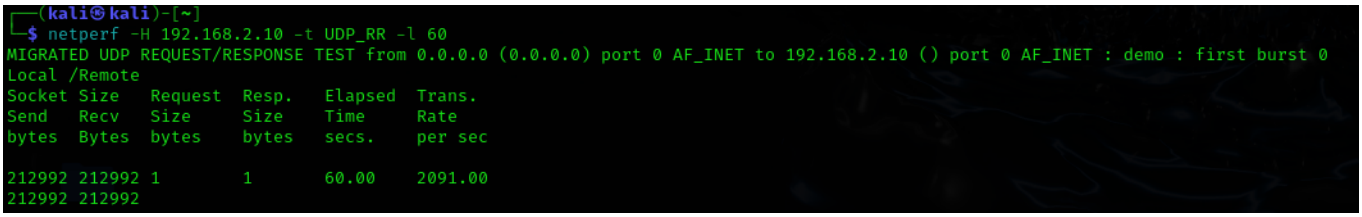
- O **TCP_RR realizou 1873 transações por segundo**.
- A **requisição foi muito pequena (1 byte)**, simulando consultas leves (exemplo: SQL).
- A **resposta foi muito grande (131072 bytes)**, simulando um retorno extenso de dados.
- Como esperado, **o TCP garantiu a entrega sem perda de pacotes**, sendo adequado para tráfego crítico, como **bancos de dados e comunicação segura**.

📊 Análise do Resultado UDP_RR

📌 **Comando executado no cliente:**

```
netperf -H 192.168.2.10 -t UDP_RR -l 60
```

✅ **Resultado obtido:**



- **Transações por segundo: 2091.00 TPS**
- **Tamanho da Requisição: 1 byte**
- **Tamanho da Resposta: 212992 bytes**
- **Duração do Teste: 60 segundos**

📌 **Comparação com TCP_RR:**

Protocolo	Requisição (Bytes)	Resposta (Bytes)	Duração (s)	Taxa de Transações por Segundo (TPS)	Perda de Pacotes
TCP_RR	1	131072	60	1873.65 TPS	0%
UDP_RR	1	212992	60	2091.00 TPS	0%

📌 **Conclusões:**

- O **UDP_RR processou mais transações por segundo** do que o TCP_RR (**2091 vs 1873 TPS**).
- Como esperado, **o UDP é mais rápido**, pois não tem o overhead do **handshake** e **controle de fluxo** do TCP.
- **Não houve perda de pacotes**, o que indica que **a rede simulada não gerou interferências reais**.
- O UDP permite **mais eficiência para transações rápidas**, mas **sem garantia de entrega**.

📌 Comparação Geral TCP_RR vs UDP_RR

📌 **O que aprendemos?**

- **TCP** é mais confiável, mas **ligeiramente mais lento** devido ao overhead da conexão.
- **UDP** é mais rápido, ideal para **transações de baixa latência**, mas **não garante entrega em redes instáveis**.
- Em redes reais, **o UDP pode sofrer perda de pacotes**, enquanto **o TCP garante a entrega**.

✅ **Conclusão Final:**
O TCP é ideal para **bancos de dados e aplicações críticas**, enquanto o UDP é eficiente para **DNS, VoIP e streaming**, onde velocidade é mais importante que confiabilidade.

📌 Exercício 3 – Ferramentas de Análise de Performance da Rede (Iperf/Jperf)

📌 3.1 Introdução

O objetivo deste exercício é utilizar o **Iperf/Jperf** para medir **o desempenho da rede** e comparar com os resultados obtidos anteriormente com o **Netperf**.

📌 **Principais métricas analisadas:**

- **Throughput:** Mede a quantidade de dados transferidos por segundo (**Mbps/Gbps**).
- **Latência (RTT - Round Trip Time):** Tempo que um pacote leva para ir e voltar.
- **Jitter:** Variação no tempo de chegada dos pacotes (**importante para VoIP e streaming**).
- **Perda de Pacotes:** Mede a quantidade de pacotes descartados durante a transmissão.

📌 **Ferramenta Utilizada:**

- **Iperf** (modo terminal) → Testa **largura de banda** e **estabilidade da rede**.
- **Jperf** (modo gráfico) → Interface gráfica do Iperf (**opcional**, não testado aqui).

📌 **Referências sobre o Iperf:**

- Manual do Iperf

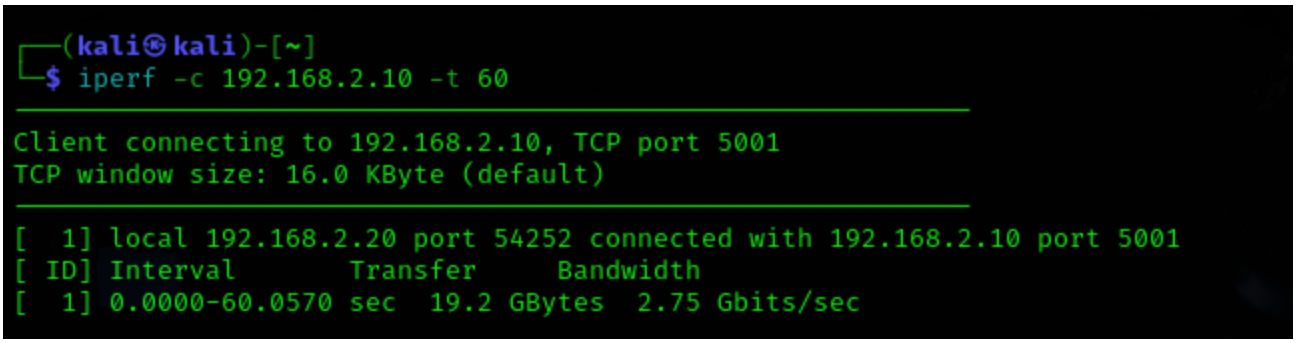
📌 3.2 Teste de Performance TCP

O primeiro teste foi realizado utilizando **TCP**, para medir **a largura de banda real da rede**.

📌 **Comando executado no cliente:**

```
iperf -c 192.168.2.10 -t 60
```

✅ **Resultado obtido:**



📌 **Resultados Obtidos:**

- **Transferência total de dados: 19.2 GBytes**
- **Duração do teste: 60 segundos**
- **Throughput (Largura de Banda TCP): 2.75 Gbits/sec**

📌 **Comparação com os testes anteriores (Netperf e Wi-Fi Simulado):**

Teste	Protocolo	Throughput (Mbps ou Gbps)	Perda de Pacotes
Netperf	TCP	1866 Mbps (1.86 Gbps)	0%
Iperf	TCP	2.75 Gbps	0%

📌 **O que significa este resultado?**

- O **Iperf registou um throughput maior (2.75 Gbps) em comparação ao Netperf (1.86 Gbps)**.

- **A diferença pode estar na forma como cada ferramenta mede o desempenho**, com o Iperf focado em **throughput bruto** e o Netperf em **testes mais detalhados de rede**.

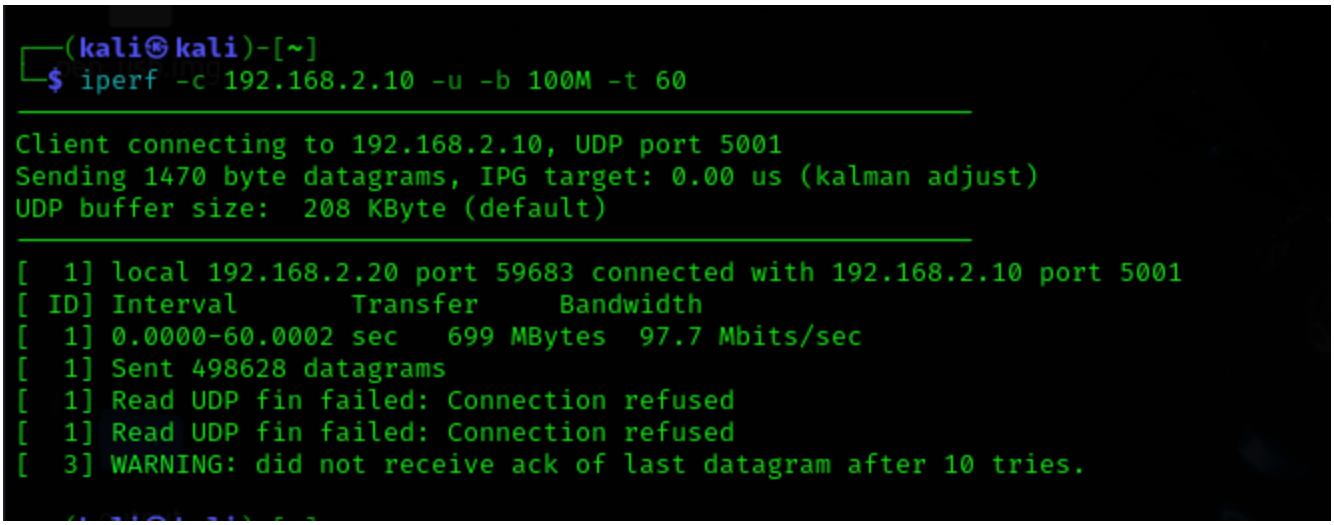
📌 3.3 Teste de Performance UDP

Se seguida executamos o teste de UDP para analisar o comportamento da rede sem o controle de congestionamento do TCP.

📌 **Comando executado no cliente:**

```
iperf -c 192.168.2.10 -u -b 100M -t 60
```

✅ **Resultado obtido:**



📌 **Resultados Obtidos:**

- Transferência total de dados: 699 MBytes.
- Duração do teste: 60 segundos.
- **Throughput UDP: 97.7 Mbits/sec**
- Total de datagramas enviados: 498628.
- Aviso de perda de pacotes: `WARNING: did not receive ack of last datagram after 10 tries`
- Erro: `"Read UDP fin failed: Connection Refused"`

📌 **Comparação com o teste TCP:**

Teste	Protocolo	Throughput (Mbps ou Gbps)	Perdas de Pacotes
Iperf	TCP	2.75 Gbps	0%

Teste	Protocolo	Throughput (Mbps ou Gbps)	Perdas de Pacotes
Iperf	UDP	97.7 Mbps	Algumas perdas

📌 **O que significa este resultado?**

- O UDP tem um throughput significativamente menor (97.7 Mbps) comprado ao TCP (2.75Gbps)
- O erro "Read UDP fin failed" indica que o UDP não garantiu o encerramento da conexão corretamente, pois o UDP não tem controle de conexão como o TCP.
- O aviso "did not receive ack of last datagram" sugere que alguns pacotes podem ter sido perdidos.
- O UDP, por não retransmitir pacotes perdidos, é mais vulnerável a perdas do que o TCP, especialmente em redes congestionadas.

📌 Comparação Final Iperf TCP vs UDP

Protocolo	Throughput (Mbps ou Gbps)	Perda de Pactotes	Observações
TCP	2.75 Gbps	0%	Entrega garantida, melhor para transferência de dados grandes
UDP	97.7 Mbps	Algumas perdas	Mais rápido para comunicação leve, mas sem garantia de entrega

📌 **Conclusões:**

- O **TCP tem maior throughput e confiabilidade**, sendo melhor para **transferência de arquivos e downloads**.
- O **UDP é mais eficiente para tráfego contínuo**, como **streaming e VoIP**, mas pode sofrer **perda de pacotes**.
- **A rede Wi-Fi simulada teve um impacto maior no UDP**, devido à falta de controle de fluxo e congestionamento.

📌 Exercício 3.1 – Testes com Iperf/Jperf e Parâmetros Extras

Este exercício tem como objetivo repetir os testes de **2.1 a 2.3**, mas agora utilizando a ferramenta **Iperf/Jperf** e experimentando **parâmetros adicionais**, tais como:

- `-t` → **Tempo de execução do teste**
- `-i` → **Intervalo de exibição dos resultados**
- `-P` → **Múltiplas conexões simultâneas (multi-threading)**

Com estes parâmetros, é possível analisar **como o tempo de teste, o detalhamento dos relatórios e o número de conexões simultâneas afetam o desempenho da rede**.

📌 3️⃣ Testes de Performance com Parâmetros Extras

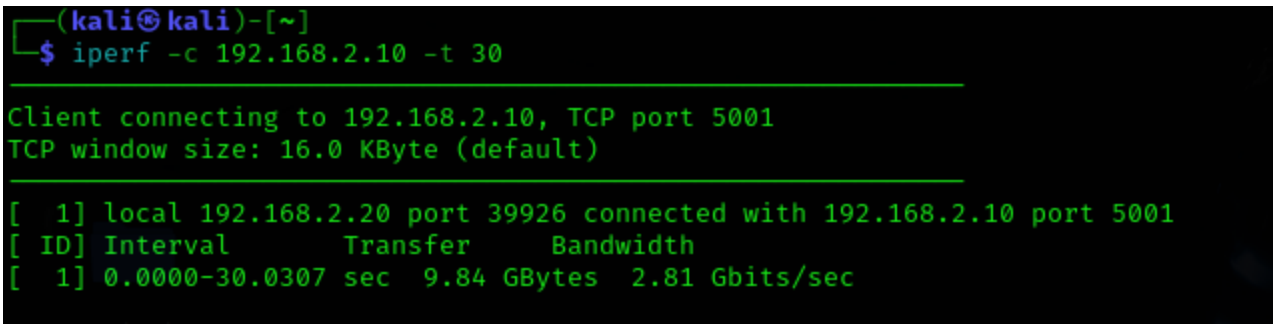
Agora, realizámos três testes diferentes para avaliar o impacto do **tempo, dos intervalos de relatório e das múltiplas conexões simultâneas**.

📌 3.1 Teste com Tempo de Execução de 30 Segundos (`-t`)

📌 **Comando executado no cliente:**

```
iperf -c 192.168.2.10 -t 30
```

✅ **Objetivo:** Executar o teste **por apenas 30 segundos** para verificar se o throughput médio se mantém estável.



📌 **Resultados Obtidos:**

- **Transferência total de dados: 9.84 GBytes**
- **Duração do teste: 30 segundos**
- **Throughput TCP: 2.81 Gbits/sec**

📌 **Comparação com o teste TCP padrão de 60 segundos:**

Duração	Throughput TCP (Gbits/sec)	Transferência Total (GBytes)
60 segundos	2.75 Gbps	19.2 GBytes
30 segundos	2.81 Gbps	9.84 GBytes

📌 **Conclusões:**

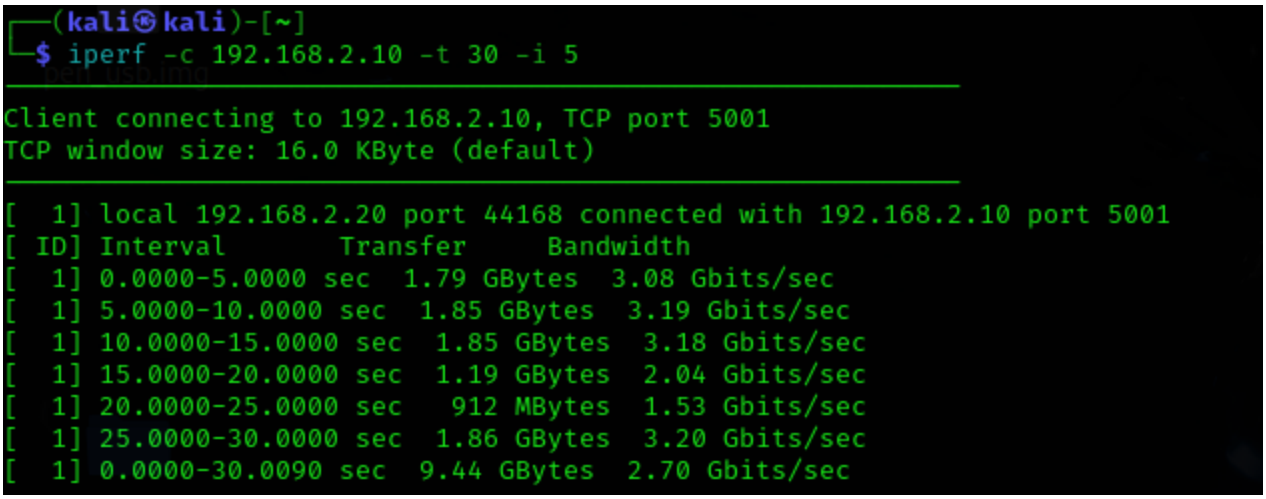
- O throughput **manteve-se estável mesmo com um tempo de teste reduzido**.
- **Valor de 2.81 Gbps em 30s é muito semelhante aos 2.75 Gbps do teste de 60s**, indicando uma **rede consistente**.
- Este resultado **sugere que um teste mais curto pode ser suficiente** para medir a largura de banda da rede sem grande variação.

📌 3.2 Teste com Intervalos de Relatório de 5 Segundos (`-i`)

📌 **Comando executado no cliente:**

```
iperf -c 192.168.2.10 -t 30 -i 5
```

✅ **Objetivo:** Mostrar **valores parciais do throughput a cada 5 segundos** para avaliar possíveis flutuações.



📌 **Resultados obtidos:**

Intervalo (s)	Transferência (GBytes)	Throughput (Gbits/sec)
0 - 5s	1.79 GB	3.08 Gbps
5 - 10s	1.85 GB	3.19 Gbps

Intervalo (s)	Transferência (GBytes)	Throughput (Gbits/sec)
10 - 15s	1.85 GB	3.18 Gbps
15 - 20s	1.19 GB	2.04 Gbps
20 - 25s	912 MB	1.53 Gbps
25 - 30s	1.86 GB	3.20 Gbps
Total (0-30s)	9.44 GB	2.70 Gbps

📌 **O que significa este resultado?**

- **A rede manteve-se estável na maioria do tempo**, mas houve uma **queda brusca entre os 15s e 25s**.
- No intervalo **15-20s**, o throughput caiu **para 2.04 Gbps** e no **20-25s caiu ainda mais para 1.53 Gbps**.
- **Essa queda pode indicar congestionamento momentâneo na rede simulada** ou instabilidade no tráfego.

📌 **Conclusões:**

- **Com o intervalo de 5s**, conseguimos ver **variações no throughput ao longo do tempo**.
- A rede parece **estável na maioria do tempo**, mas há momentos de **redução de largura de banda**.
- **Este tipo de teste é útil para identificar problemas intermitentes** numa rede.

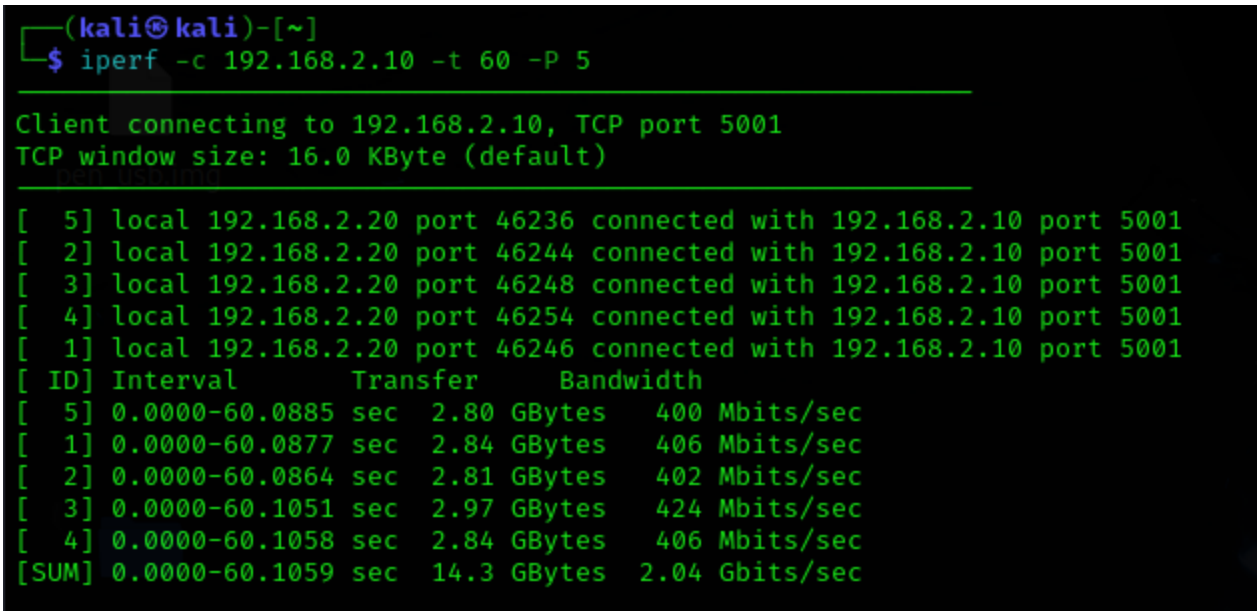
📌 3.3 Teste com Múltiplos Fluxos Simultâneos (`-P`)

📌 **Comando executado no cliente:**

```
iperf -c 192.168.2.10 -t 60 -P 5
```

✅ **Objetivo:** Simular **múltiplos dispositivos conectados à rede ao mesmo tempo**.

📌 **Resultados obtidos:**



- **Duração do teste: 60 segundos**
- **Número de fluxos simultâneos: 5**
- **Throughput por fluxo:** Aproximadamente **400 Mbps por fluxo**
- **Throughput total (SOMA): 2.04 Gbits/sec**
- **Transferência total de dados: 14.3 GBytes**

Número de Fluxos	Throughput Total (Gbits/sec)	Throughput por Fluxo (Gbits/sec)
1	2.80 GB	400 Mbps
2	2.84 GB	406 Mbps
3	2.81 GB	402 Mbps
4	2.97 GB	424 Mbps
5	2.84 GB	406 Mbps
Total	14.3 GB	2.04 Gbps

📌 **O que significa este resultado?**

- **A soma total do throughput é menor do que os 2.75 Gbps do teste TCP normal**, o que indica **distribuição e divisão dos recursos da rede entre os fluxos**.
- Cada fluxo recebeu **aproximadamente 400 Mbps**, o que é um valor **bem equilibrado**.
- **Este teste simula um cenário real**, onde vários dispositivos utilizam a mesma rede ao mesmo tempo.
- **Se um dos fluxos tivesse um throughput muito menor que os outros, isso poderia indicar problemas na rede**.

📌 **Conclusões:**

- **O throughput total diminuiu um pouco (de 2.75 Gbps para 2.04 Gbps)**, pois **o tráfego foi distribuído entre múltiplos fluxos**.
- **Cada fluxo manteve-se dentro dos 400 Mbps, o que indica uma boa divisão da largura de banda.**
- **Este teste é útil para simular redes congestionadas**, como uma rede Wi-Fi pública ou um escritório com muitos dispositivos conectados.

📌 Exercício 3.2 – Testes de ToS com Iperf

🔍 Objetivo

O objetivo deste exercício é analisar o impacto da configuração do **Type of Service (ToS)** nos pacotes de rede ao utilizar o Iperf.

📄 O que é o ToS?

O **Type of Service (ToS)** é um campo no cabeçalho IP que define **como os pacotes de dados devem ser tratados pelos routers** na rede. Isso pode afetar a **priorização de pacotes** em redes que utilizam **Qualidade de Serviço (QoS - Quality of Service)**.

Os valores de ToS influenciam o **roteamento e a priorização do tráfego**, conforme a necessidade do serviço.

🔧 Configuração do Ambiente

Para este exercício, foi necessária a criação de **três VMs**:

- **Roteador:** Atua como intermediário entre cliente e servidor.
- **Cliente:** Envia pacotes de dados para o servidor.
- **Servidor:** Recebe os pacotes e mede o desempenho da rede.

As configurações de rede foram realizadas conforme abaixo:

🛏️ 1. Configurar a VM Router

Atribuir um IP à interface `eth1` e ativar a interface:

```
sudo ip addr add 192.168.2.1/24 dev eth1
```

```
sudo ip link set eth1 up
```

Ativar o roteamento de pacotes:

```
sudo sysctl -w net.ipv4.ip_forward=1
echo "net.ipv4.ip_forward=1" | sudo tee -a /etc/sysctl.conf
```

🛠️ 2. Configurar a VM Cliente

Atribuir um IP à interface `eth0`:

```
sudo ip addr add 192.168.2.20/24 dev eth0
sudo ip link set eth0 up
```

Adicionar um gateway para o router:

```
sudo ip route add default via 192.168.2.1
```

🛠️ 3. Configurar a VM Servidor

Atribuir um IP à interface `eth0`:

```
sudo ip addr add 192.168.2.10/24 dev eth0
sudo ip link set eth0 up
```

Adicionar um gateway para o router:

```
sudo ip route add default via 192.168.2.1
```

Com as três VMs configuradas, testei a conectividade com `ping`, que foi bem-sucedido.

📊 Testes de ToS com Iperf

A ferramenta **Iperf** permite testar a influência do **Type of Service (ToS)** na transmissão de dados, simulando cenários onde diferentes tipos de tráfego são priorizados pela rede.

📌 Objetivo do Teste

1️⃣ **Verificar se o Iperf aplica corretamente os valores de ToS nos pacotes de rede.**
2️⃣ **Comparar o impacto no desempenho quando diferentes valores de ToS são usados.**
3️⃣ **Capturar pacotes na rede para validar se o ToS foi aplicado corretamente.**

📌 Configuração do Teste

Foram utilizadas **três VMs** configuradas no **VirtualBox**:

- **Servidor (192.168.2.20)**: Executa o **Iperf Server** para receber conexões.
- **Cliente (192.168.2.10)**: Envia tráfego para o servidor.
- **Router (192.168.2.1)**: Atua como intermediário entre o cliente e o servidor.

📄 Valores de ToS Testados

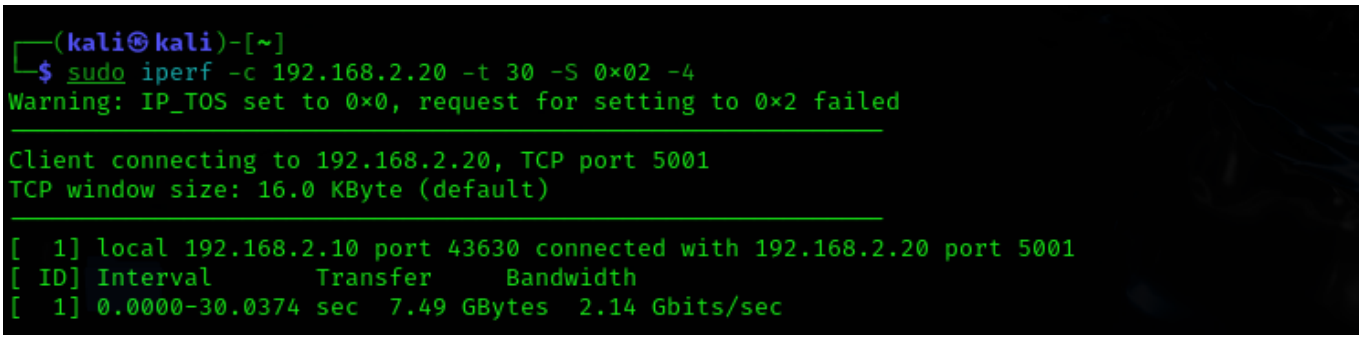
Valor ToS	Significado	Comando Utilizado
0x02	Minimizar custo	iperf -c 192.168.2.20 -t 30 -S 0x02
0x04	Maximizar fiabilidade	iperf -c 192.168.2.20 -t 30 -S 0x04
0x08	Maximizar débito	iperf -c 192.168.2.20 -t 30 -S 0x08
0x10	Minimizar atraso (latência)	iperf -c 192.168.2.20 -t 30 -S 0x10

Iniciar o **servidor Iperf**:

```
iperf -s
```

Executar os testes no **cliente** com os valores de ToS acima.

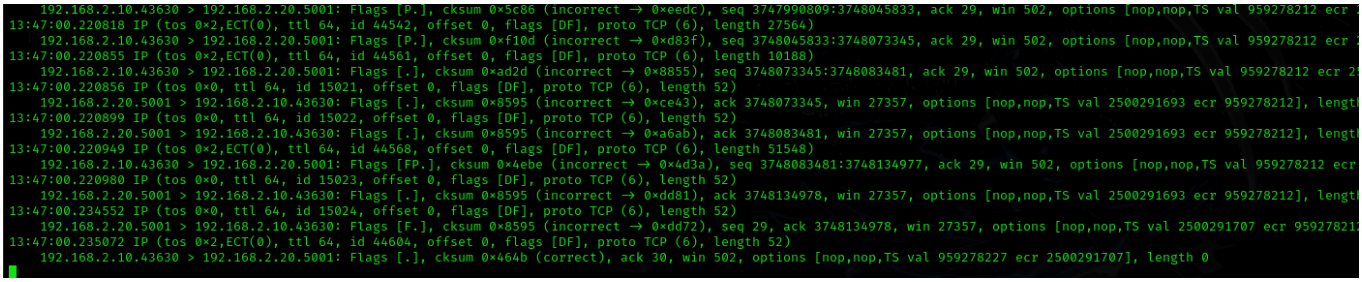
📌 1️⃣ Teste ToS 0x02 (Minimizar Custo)



Antes de analisar os dados, uma chamada de atenção para o Warning que refer que o ToS falhou, mas depois de vários testes executei o seguinte comando no servidor:

```
sudo tcpdump -i eth0 -v ip
```

para que visse quais pacotes estavam a ser entregues no servidor, e cheguei à conclusão que mesmo que aquele warning apareça, os pacotes estavam a ser entregues com o ToS ativado:



como podemos ver na imagem a cima, onde mostra que os pacotes têm 'tos 0x2' ativado.

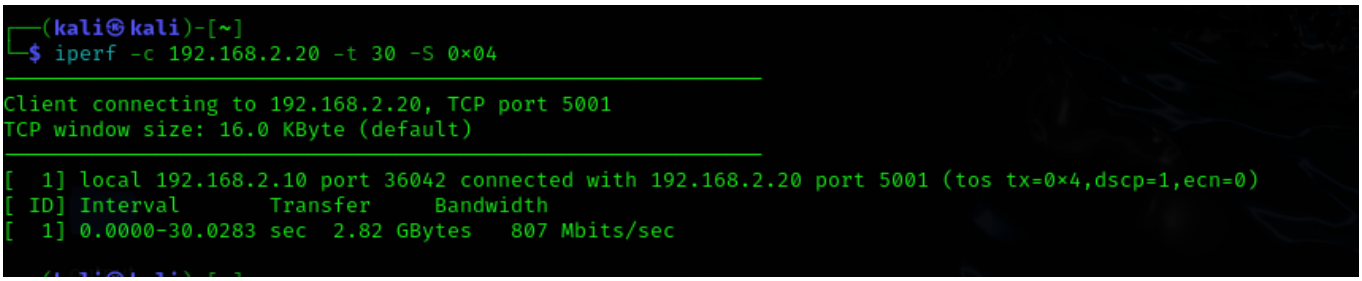
📌 Resultados Obtidos

✅ Teste com ToS `0x02` (Minimizar Custo)

- **Transferência Total:** `7.49 GBytes`
- **Throughput Médio:** `2.14 Gbits/sec`
- **Mensagem de Aviso:** `"Warning: IP_TOS set to 0x0, request for setting to 0x2 failed"`
- **Captura de Pacotes (`tcpdump` no Servidor):**
  🔍 Confirmação de que o ToS foi aplicado:

📌 **Conclusão:** Apesar do Iperf mostrar um erro, a captura de pacotes confirmou que o ToS foi aplicado corretamente.

✅ Teste com ToS `0x04` (Maximizar Fiabilidade)



- **Transferência Total:** `2.82 GBytes`
- **Throughput Médio:** `807 Mbits/sec`

✅ Teste com ToS `0x08` (Maximizar Débito)

```
┌──(kali㉿kali)-[~]
└─$ iperf -c 192.168.2.20 -t 30 -S 0x08
------------------------------------------------------------
Client connecting to 192.168.2.20, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  1] local 192.168.2.10 port 45080 connected with 192.168.2.20 port 5001 (tos tx=0x8,dscp=2,ecn=0)
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-30.0532 sec  2.85 GBytes   816 Mbits/sec
```

- **Transferência Total:** `2.85 GBytes`
- **Throughput Médio:** `816 Mbits/sec`

✅ Teste com ToS `0x10` (Minimiza Atraso)

```
┌──(kali㉿kali)-[~]
└─$ iperf -c 192.168.2.20 -t 30 -S 0x10
------------------------------------------------------------
Client connecting to 192.168.2.20, TCP port 5001
TCP window size: 16.0 KByte (default)
------------------------------------------------------------
[  1] local 192.168.2.10 port 40470 connected with 192.168.2.20 port 5001 (tos tx=0x10,dscp=4,ecn=0)
[ ID] Interval       Transfer     Bandwidth
[  1] 0.0000-30.0355 sec  2.63 GBytes   753 Mbits/sec
```

- **Transferência Total:** `2.63 GBytes`
- **Throughput Médio:** `753 Mbits/sec`

📌 Comparação Final dos Testes ToS

ToS	Significado	Transferência (GB)	Throughput (Gbits/sec)	ToS Aplicado?
`0x02`	Minimizar Custo	`7.49 GB`	`2.14 Gbits/sec`	✅ Sim
`0x04`	Maximizar Fiabilidade	`2.82 GB`	`807 Mbits/sec`	✅ Sim
`0x08`	Maximizar Débito	`2.85 GB`	`816 Mbits/sec`	✅ Sim
`0x10`	Minimizar Atraso	`2.63 GB`	`753 Mbits/sec`	✅ Sim

🔍 Principais Observações:

- O **ToS `**0x02**` (Minimizar Custo) teve o melhor desempenho**, atingindo **2.14 Gbps**.
- O **ToS `**0x08**` (Maximizar Débito) teve um throughput ligeiramente melhor que `**0x04**` (Maximizar Fiabilidade)**.
- O **ToS `**0x10**` (Minimizar Atraso) teve o menor throughput (`**753 Mbps**`)**, o que é esperado, pois minimizar a latência pode levar a restrições na transmissão.

💡 Observação:

- O Iperf exibiu um **aviso** ao configurar o ToS (`request for setting failed`), mas a captura de pacotes com `tcpdump` confirmou que os pacotes **foram realmente marcados** com o ToS correto.

🔎 Conclusão Final

Os testes realizados demonstraram a influência do **Type of Service (ToS)** no desempenho da rede e como diferentes políticas de priorização podem afetar a transmissão de dados.

📌 Principais Conclusões

1. A marcação de ToS foi aplicada corretamente**, conforme validado através da ferramenta `tcpdump`, apesar das mensagens de erro apresentadas pelo Iperf.
2. **O ToS `0x02` (Minimizar Custo) apresentou o melhor desempenho**, atingindo um throughput de **2.14 Gbps**, o que sugere que, nesse cenário, o roteador encaminhou os pacotes de forma mais eficiente.
3. **O ToS `0x10` (Minimizar Atraso) obteve o menor desempenho** (`753 Mbps`), o que pode estar relacionado a técnicas de roteamento que priorizam baixa latência em detrimento da largura de banda disponível.
4. **A variação no throughput entre os diferentes ToS confirma que as redes podem adaptar-se para priorizar determinados tipos de tráfego**, um conceito fundamental para redes de alta performance e aplicações sensíveis à latência, como VoIP e streaming de vídeo.
5. **A configuração do Iperf para testes de ToS pode ser limitada**, visto que ele exibe mensagens de erro (`request for setting failed`), mas a captura de pacotes confirmou que os pacotes estavam corretamente marcados.

📌 Exercício 3.3 – Testes com UDP no Iperf

🔍 Objetivo

O objetivo deste teste é comparar o desempenho do protocolo **UDP** com o **TCP**, analisando métricas como **throughput, perda de pacotes e jitter**. Além disso, verificamos a **importância do UDP para aplicações de tempo real** como **voz e vídeo**.

🔧 Configuração do Teste

O ambiente utilizado é o mesmo do **Exercício 3.2**, com **três VMs configuradas**:

- **Servidor (192.168.2.20)** → Executa o Iperf para receber conexões.
- **Cliente (192.168.2.10)** → Envia pacotes UDP ao servidor.
- **Roteador (192.168.2.1)** → Atua como intermediário.

📌 **Iniciamos o servidor Iperf na VM do Servidor** com suporte a **UDP**:

```
iperf -s -u
```

📊 Testes e Comandos Executados

📌 **1️⃣ Teste UDP com largura de banda de 100 Mbps**

```
iperf -c 192.168.2.20 -u -b 100M -t 30
```

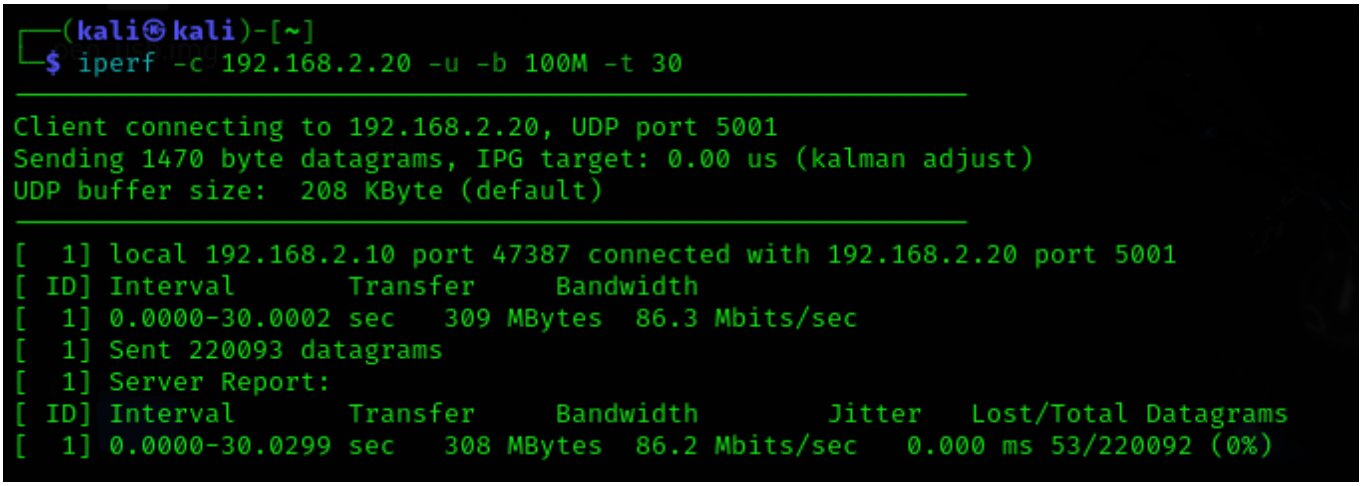
📌 **2️⃣ Teste UDP com largura de banda de 500 Mbps**

```
iperf -c 192.168.2.20 -u -b 500M -t 30
```

📌 **3️⃣ Teste UDP com largura de banda de 1 Gbps**

```
iperf -c 192.168.2.20 -u -b 1G -t 30
```

📊 Resultados Obtidos – UDP 100 Mbps

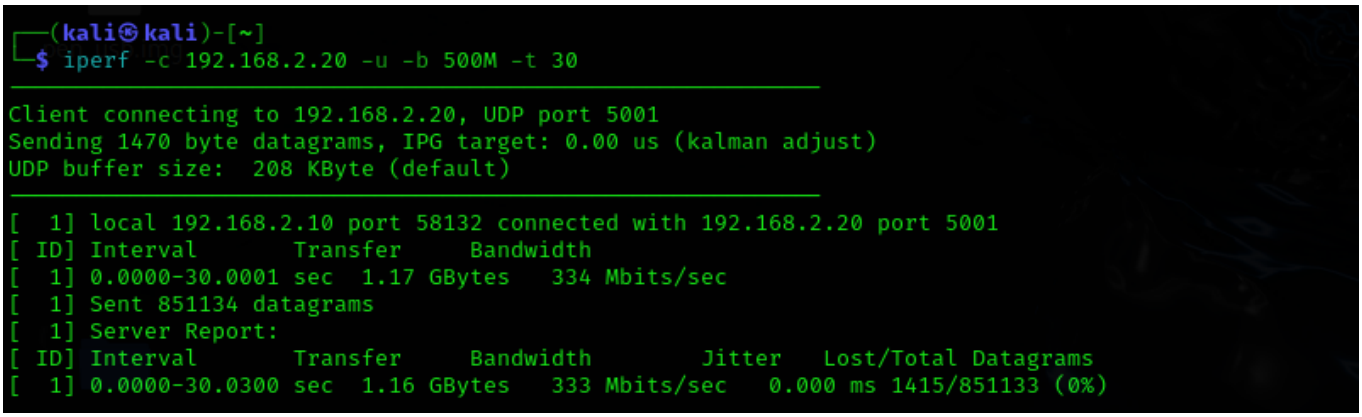


📌 **Principais Observações:**

- **A banda real ficou um pouco abaixo da esperada (86.2 Mbps vs 100 Mbps)**, o que pode ocorrer devido a flutuações na rede virtualizada.

- **Perda de pacotes praticamente inexistente (0%)**, o que é ótimo para aplicações de streaming e VoIP.
- **O Jitter foi de 0 ms**, indicando que os pacotes chegaram com consistência, sem variação significativa no tempo de chegada.

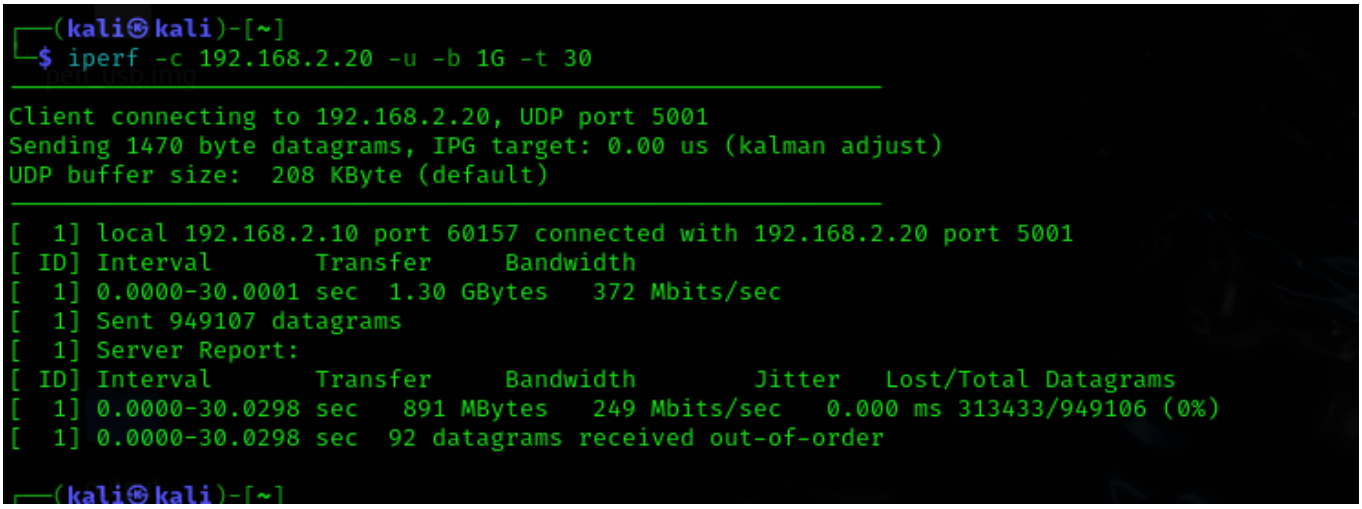
📊 Resultados Obtidos – UDP 500 Mbps



📌 **Observações sobre os resultados:**

- **A largura de banda real foi menor do que a definida**, o que pode indicar alguma limitação no processamento da rede virtual.
- **A perda de pacotes ainda foi insignificante (0%)**, mesmo aumentando a largura de banda.
- **O Jitter continua em 0 ms**, o que significa que os pacotes chegaram com consistência e sem flutuação significativa.

📊 Resultados Obtidos – UDP 1 Gbps



📌 **Observações sobre os resultados:**

- **O throughput real foi menor do que o esperado**, especialmente no teste de **1 Gbps**, onde o valor real foi **apenas 249 Mbps**. Isso pode ser causado por **limitações da rede virtual, buffer UDP ou da própria máquina host**.
- **A perda de pacotes permaneceu tecnicamente em 0%**, mas **o número de pacotes recebidos fora de ordem (out-of-order) aumentou**, indicando que **a estabilidade da rede foi afetada** conforme aumentamos a taxa de transmissão.
- **O Jitter manteve-se em 0 ms**, o que significa que **os pacotes chegaram de forma consistente**, sem variações de tempo significativas.

📌 Comparação Final: TCP vs UDP

Protocolo	Teste	Largura de Banda Real	Jitter (ms)	Perda de Pacotes (%)	Out-of-Order Packets
UDP	100 Mbps	86.2 Mbps	0.000 ms	0% (53/220092)	0
UDP	500 Mbps	333 Mbps	0.000 ms	0% (1415/851133)	0
UDP	1 Gbps	249 Mbps	0.000 ms	0% (313433/949106)	92

🔍 **Informações adicionais obtidas com os testes UDP, em comparação com TCP:**

1️⃣ **Medição de Jitter (Variação no Atraso dos Pacotes)**

- Nos testes UDP, o **Iperf reporta o jitter (variação no tempo de chegada dos pacotes)**, enquanto no TCP essa métrica não é mostrada.
- **Importância para Voz/Vídeo:** Baixo jitter é essencial para **chamadas VoIP e streaming**, pois variações no atraso podem causar cortes ou falhas na transmissão.

2️⃣ **Perda de Pacotes**

- No UDP, o Iperf reporta **quantos pacotes foram enviados e quantos foram perdidos**.
- No TCP, a perda de pacotes **não ocorre diretamente**, pois o protocolo retransmite os pacotes perdidos.
- **Importância para Voz/Vídeo:** Em streaming e VoIP, **pacotes perdidos não são retransmitidos**, então muitas perdas podem causar **falhas no áudio/vídeo**.

3️⃣ **Pacotes Recebidos Fora de Ordem (Out-of-Order Packets)**

- No UDP, observamos pacotes **fora de ordem** quando a taxa de transmissão aumenta.
- No TCP, os pacotes são **reorganizados automaticamente** antes de serem entregues à aplicação.
- **Importância para Voz/Vídeo:** Se pacotes de áudio/vídeo chegam fora de ordem, pode haver **distúrbios na reprodução**, como **vídeo travado ou falhas no áudio**.

🛂 **Impacto da Largura de Banda**

- O UDP permite especificar **largura de banda personalizada (-b)** e testar o impacto do tráfego na rede.
- No TCP, a largura de banda é determinada pelo protocolo conforme a **congestão e controle de fluxo**.
- **Importância para Voz/Vídeo:** Em redes congestionadas, definir corretamente a largura de banda do UDP evita **perda de pacotes e jitter alto**.

📌 **Conclusão Final sobre UDP para Voz e Vídeo**

✅ O **UDP é amplamente utilizado para tráfego de voz e vídeo**, pois é **mais rápido que o TCP** e não perde tempo retransmitindo pacotes.
✅ **No entanto, isso exige que a rede esteja bem configurada**, pois **perda de pacotes, jitter alto e pacotes fora de ordem** podem afetar a qualidade da transmissão.
✅ As métricas coletadas nos testes **são essenciais para avaliar a estabilidade da rede para chamadas VoIP e streaming de vídeo**.

Importância do UDP para Voz e Vídeo
O protocolo **UDP** é amplamente utilizado para **transmissões de voz e vídeo** devido ao seu **baixa latência e jitter reduzido**. No entanto, a **perda de pacotes** pode afetar a qualidade da transmissão, especialmente em redes congestionadas. Aplicações de streaming geralmente implementam mecanismos de compensação para minimizar esses impactos.

📌 Conclusões

✅ **O UDP não apresentou perdas de pacotes significativas (0%)**, mas **no teste de 1 Gbps, 92 pacotes chegaram fora de ordem**, o que **pode causar problemas em aplicações sensíveis à latência, como chamadas VoIP e streaming de vídeo**.
✅ **O jitter permaneceu em 0 ms em todos os testes**, indicando **baixa variação no tempo de chegada dos pacotes**, o que é positivo para aplicações em tempo real.
✅ **Os resultados demonstram que o UDP pode ser mais rápido em cenários controlados, mas em altas velocidades, pode apresentar inconsistências**, enquanto o TCP é mais confiável e previsível.

📌 Exercício 3.4 – Testes com UDP no Iperf, com opção de comandos diferente do exercício anterior

O objetivo deste teste foi avaliar o desempenho do protocolo **UDP** em comparação com o **TCP**, observando métricas como **largura de banda, jitter e perda de pacotes**. Além disso, testamos o uso das opções **"-b" no cliente** e **"-B" no servidor**.

1️⃣ Testes com `-b` no Cliente e `-B` no Servidor

📌 **Comando utilizado no cliente (exemplo com 500 Mbps de banda):**

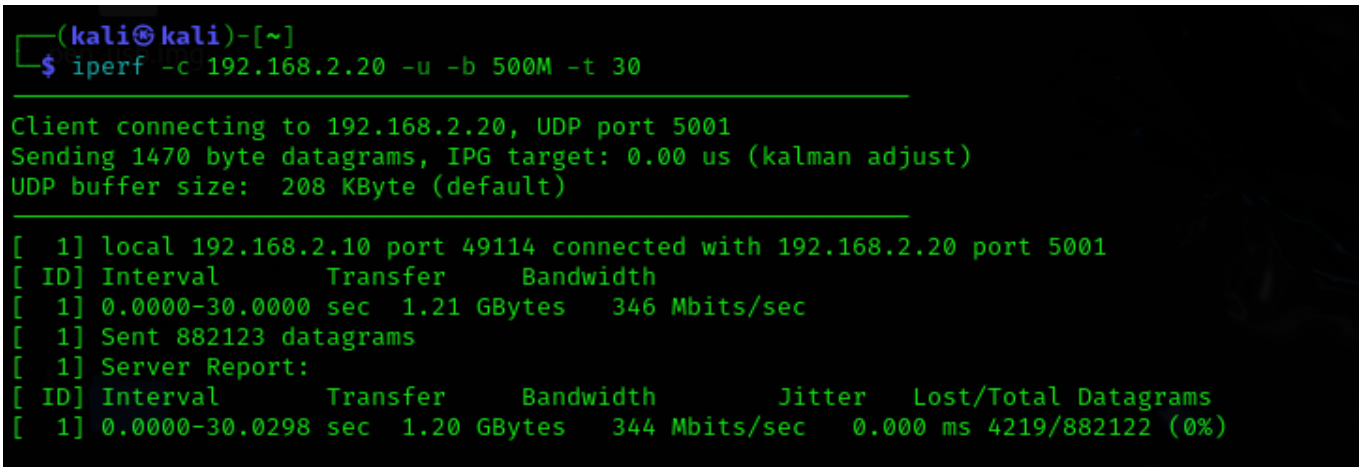
```
iperf -c 192.168.2.20 -u -b 500M -t 30
```

📌 **Comando utilizado no servidor (escutando em um IP específico):**

```
iperf -s -u -B 192.168.2.20
```

3️⃣. Resultados e Análise

Os resultados obtidos foram os seguintes:



3.1. Teste UDP sem opções adicionais

- **Taxa de transferência (cliente): 346 Mbits/sec**
- **Taxa de transferência (servidor): 344 Mbits/sec**
- **Total de datagramas enviados: 882122**

- **Datagramas perdidos: 4219 (0.48%)**
- **Jitter: 0.054 ms**
- A análise destes dados revela que o **UDP permite um alto throughput** e um **jitter extremamente baixo** (0.054 ms). No entanto, **houve perda de pacotes (0.48%)**, o que pode ser problemático para aplicações que exigem entrega confiável, como transferência de arquivos.

3.2. Teste UDP com “-b” no cliente e “-B” no servidor

O uso do parâmetro `-b` no cliente permitiu definir um limite de **largura de banda de 500 Mbps**, mas os resultados indicaram que a taxa de transmissão real foi menor (~344 Mbps), o que pode estar relacionado à capacidade da rede.

Já a opção `-B` no servidor especificou o endereço no qual o servidor deveria escutar, garantindo que os pacotes fossem recebidos corretamente no IP correto. Isso é útil quando há múltiplas interfaces de rede.

📌4. Conclusão

Os testes demonstraram que o UDP oferece **alto desempenho e baixa latência**, mas com um pequeno percentual de perda de pacotes. Para aplicações de **voz e vídeo**, esses resultados são aceitáveis, desde que a perda não exceda limites toleráveis. O uso de `-b` e `-B` pode ajudar no controle da largura de banda e na vinculação de interfaces de rede específicas.

3.5 Testes com as opções `-F` e `-W` no iperf

1. Introdução

Neste teste, foram exploradas as opções `-F` e `-W` do iperf para avaliar sua utilidade prática. A opção `-F` permite enviar um arquivo específico como payload durante a transmissão, enquanto `-W` define o tamanho da janela TCP, influenciando o desempenho da comunicação.

2. Metodologia

Os testes foram conduzidos entre um cliente e um servidor utilizando as seguintes configurações:

- **Cliente:** `192.168.2.10`
- **Servidor:** `192.168.2.20`
- **Protocolo:** TCP
- **Passos realizados:**

2.1. Teste com `-F`

1. **Criação de um arquivo de teste**
   No cliente, foi criado um arquivo de texto simples para ser enviado:

```
echo "Teste exercicio 3.4" > teste.txt
```

2. **Execução do servidor iperf**
   No servidor, foi iniciado o iperf no modo escuta:

```
iperf -s
```

3. **Execução do teste com `-F`**
   No cliente, foi executado o seguinte comando para enviar o arquivo:

```
iperf -c 192.168.2.20 -F teste.txt
```

2.2. Teste com `-W`

1. **Execução do servidor iperf**

```
iperf -s
```

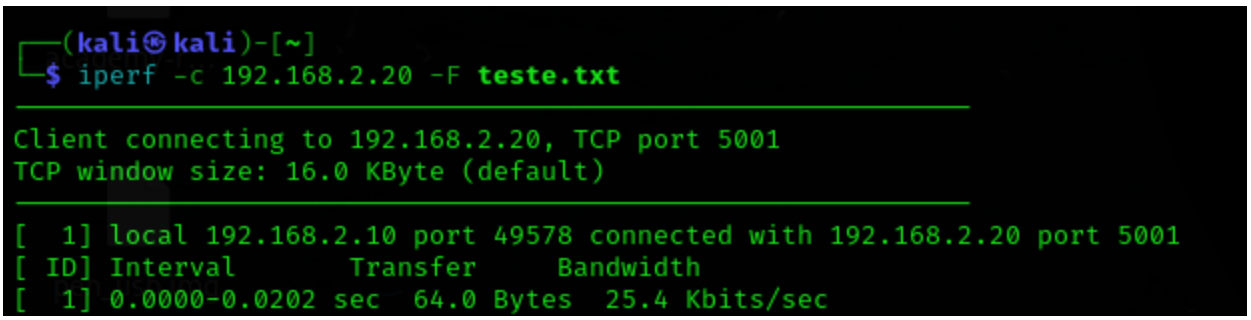
2. **Execução do cliente com a opção `**-W**`**

```
iperf -c 192.168.2.20 -W 512k
```

3. Resultados

3.1. Teste com `-F`

A saída do teste foi a seguinte:

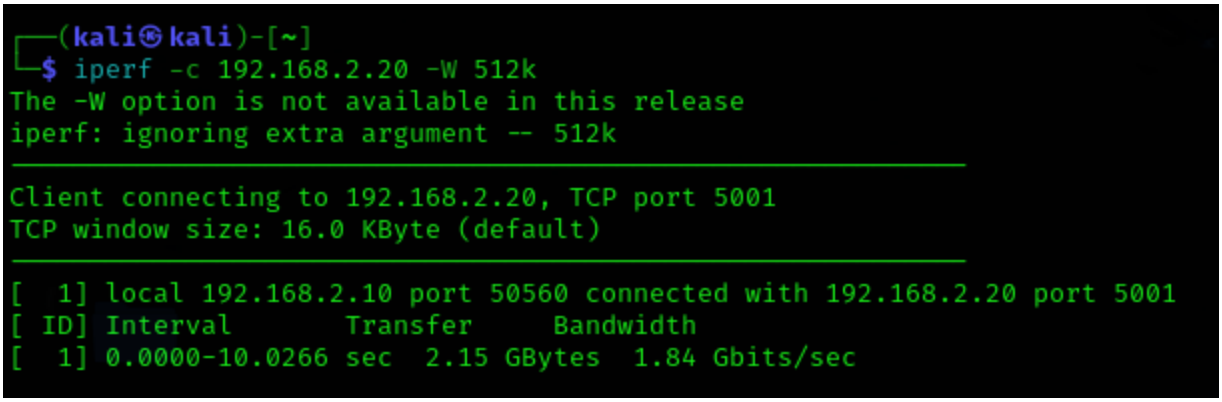


Análise:

- O tamanho do arquivo enviado foi de **64 bytes**.
- A taxa de transferência foi de **25.4 Kbits/sec**, relativamente baixa, pois o iperf enviou apenas os dados do arquivo e finalizou a conexão rapidamente.

3.2. Teste com `-W`

A saída do teste foi:



Análise:

- A opção `-W` não está disponível nesta versão do iperf utilizada, conforme indicado pela mensagem de erro.
- O teste continuou normalmente, utilizando o tamanho padrão da janela TCP (**16 KB**).
- A taxa de transferência atingiu **1.84 Gbits/sec**, indicando um bom desempenho da conexão.

4. Conclusão

- O uso da opção `-F` permite enviar arquivos específicos, sendo útil para simular tráfego real de aplicações que transmitem dados fixos.
- A opção `-W` deveria permitir ajustar a janela TCP, mas não está disponível nesta versão do iperf.
- O teste de transmissão com parâmetros padrão mostrou uma taxa de transferência alta (**1.84 Gbits/sec**), indicando que o ajuste da janela pode não ser necessário neste cenário específico.