UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

RELATÓRIO DO PROJETO 3
ANÁLISE DE TRÁFEGO DE REDE NO WIRESHARK E NO MINET
DISCIPLINA MC833 - PROGRAMAÇÃO DE REDES DE COMPUTADORES

1º SEMESTRE DE 2024
TURMA A

PROFESSOR: EDMUNDO ROBERTO MAURO MADEIRA
ALUNO: DANIEL MENDES DOS SANTOS RA:214752
ALUNO: RAONITON ADRIANO DA SILVA RA:186291

1. Introdução

Este relatório tem por objetivo detalhar o processo de emular uma rede com hosts e switches, e a partir da criação da rede, fazer a transmissão de alguns pacotes, sendo então possível analisar a forma com que os dados são armazenados nos pacotes e as taxas de recebimento e transmissão.

A proposta do projeto era emular uma rede contendo 4 hosts e 1 switch, os pacotes fluíram de um host para o outro, passando através do switch. Essa rede é criada através do Mininet, que é um emulador de redes capaz de criar redes com switches, hosts, servidores, etc. Com a rede criada, utilizamos outro programa, o Wireshark, que é um programa de análise de pacotes, que é capaz de capturar os pacotes de forma muito detalhada. Após capturar os pacotes, os analisamos com auxílio de algoritmos em python.

2. Objetivos

Os principais objetivos:

a. Configurar o ambiente para a emulação da rede contendo 4 hosts e 1 switch

b. Capturar os packets transmitidos entre os hosts

c. Analisar através de um algoritmo em python detalhes presentes nos packets

d. E observar como são transmitidos e recebidos os dados, como também o fluxo e taxa de transmissão.

3. Como instalar os programas necessários

O sistema utilizado: *Ubuntu 23.10*

**Alguns sistemas podem ter mais dependências a serem instaladas, a seguir estão apenas os códigos para instalação padrão*

1. Wireshark

```
sudo apt install wireshark
sudo dpkg-reconfigure wireshark-common
sudo adduser $USER wireshark
/*Os comandos acima permitem que qualquer usuário capture pacotes*/
```

2. **Mininet**

```
sudo apt install mininet
```

3. **Xterm**

```
sudo apt-get install xterm
```

4. **Executando os programas**

Utilizamos a seguinte sequência de command lines:

Para fazer uma topologia de 4 hosts e 1 switch:

```
sudo mn --topo single,4
```

Para iniciar o Wireshark use o comando abaixo, que abrirá um novo terminal:

```
mininet > xterm s1
```

Nesse novo terminal do Wireshark executamos:

```
wireshark &
```

Por fim, voltamos ao terminal principal, no qual executamos:

```
mininet > h1 ping –c 500 h3
mininet > h2 ping –c 500 h4
```

5. **Script Python**

Python

```python
from scapy.all import rdpcap, IP, ICMP
from datetime import datetime

def extract_icmp_info(pcap_file):
    packets = rdpcap(pcap_file)
    icmp_packets = [pkt for pkt in packets if ICMP in pkt]

    if not icmp_packets:
        print(f"Nenhum ICMP packet encontrado no arquivo -> '{pcap_file}'")
        return

    src_ips = []
    dst_ips = []
    packet_times = []
    tuplasDestSrc = set()
    total_size = 0

    for pkt in icmp_packets:
        if IP in pkt and ICMP in pkt:
            #src_ips.append(pkt[IP].src)
            #dst_ips.append(pkt[IP].dst)
            tupla = (pkt[IP].src ,  pkt[IP].dst)
            if tupla not in tuplasDestSrc:
                tuplasDestSrc.add(tupla)
            packet_times.append(pkt.time)
            total_size += len(pkt)

    # Calculate statistics
    total_packets = len(icmp_packets)
    total_time = packet_times[-1] - packet_times[0] if total_packets > 1 else 0
    average_throughput = (total_size / total_time) if total_time > 0 else 0
    average_interval = (total_time / (total_packets - 1)) if total_packets > 1
else 0

    print("Analise de packets sob o protocolo ICMP")
    #print(f"Source IPs: {set(src_ips)}")
    #print(f"Destination IPs: {set(dst_ips)}")
    print(f"IP Src, IP Dest: {tuplasDestSrc}")
    print(f"Total ICMP Packets:                {total_packets}")
```

```
    print(f"Taxa de Transf. Media(Throughput): {average_throughput:.2f}
bytes/segundo")
    print(f"Intevalo Medio entre Packets:      {average_interval:.6f} segudos")

# Example usage
pcap_file = 'pacotes.pcapng'
extract_icmp_info(pcap_file)
```

- O script acima foi implementado para realizar a análise dos pacotes capturados com o Wireshark.
- Na parte superior do programa, realizamos as importações necessárias e na parte inferior passamos o nome do arquivo para a função que realizará o tratamento dos dados.

- A função **extract_icmp_info()** inicialmente lê os dados do arquivo passado e armazena em packets, feito isso, filtra os dados armazenando em icmp_packets somente os pacotes que estão sob o protocolo ICMP(Internet Control Message Protocol).

- Após isso, utilizando um laço for extraímos os IP origem e IP destino, contamos o número de packets, o tamanho total dos packets e salvamos o tempo total da transmissão.

- Com esses dados, calculamos:
  - $nº\ total\ de\ pacotes$
  - $Taxa\ de\ transmissão\ Média(Throughput) = \frac{Tamanho\ total}{Tempo\ total}$
  - $Intervalo\ médio\ entre\ packets = \frac{Tempo\ total}{nº\ total\ de\ packets - 1}$

- E no fim exibimos a análise:

```
raoniton@raoniton-Dell-G15:~/mc833/projeto3$ python3 analise.py
Analise de packets sob o protocolo ICMP
IP Src, IP Dest: {('10.0.0.3', '10.0.0.1'), ('10.0.0.4', '10.0.0.2'), ('10.0.0.1', '10.0.0.3'), ('10.0.0.2', '10.0.0.4')}
Total ICMP Packets:                4002
Taxa de Transf. Media(Throughput): 765.88 bytes/segundo
Intevalo Medio entre Packets:      0.127989 segudos
```

Figura 1. resultados terminal

```
Unset
$python3 analise.py
Analise de packets sob o protocolo ICMP
IP Src, IP Dest: {('10.0.0.3', '10.0.0.1'), ('10.0.0.4', '10.0.0.2'),
('10.0.0.1', '10.0.0.3'), ('10.0.0.2', '10.0.0.4')}
Total ICMP Packets:                 4002
Taxa de Transf. Media(Throughput): 765.88 bytes/segundo
Intevalo Medio entre Packets:      0.127989 segudos
```

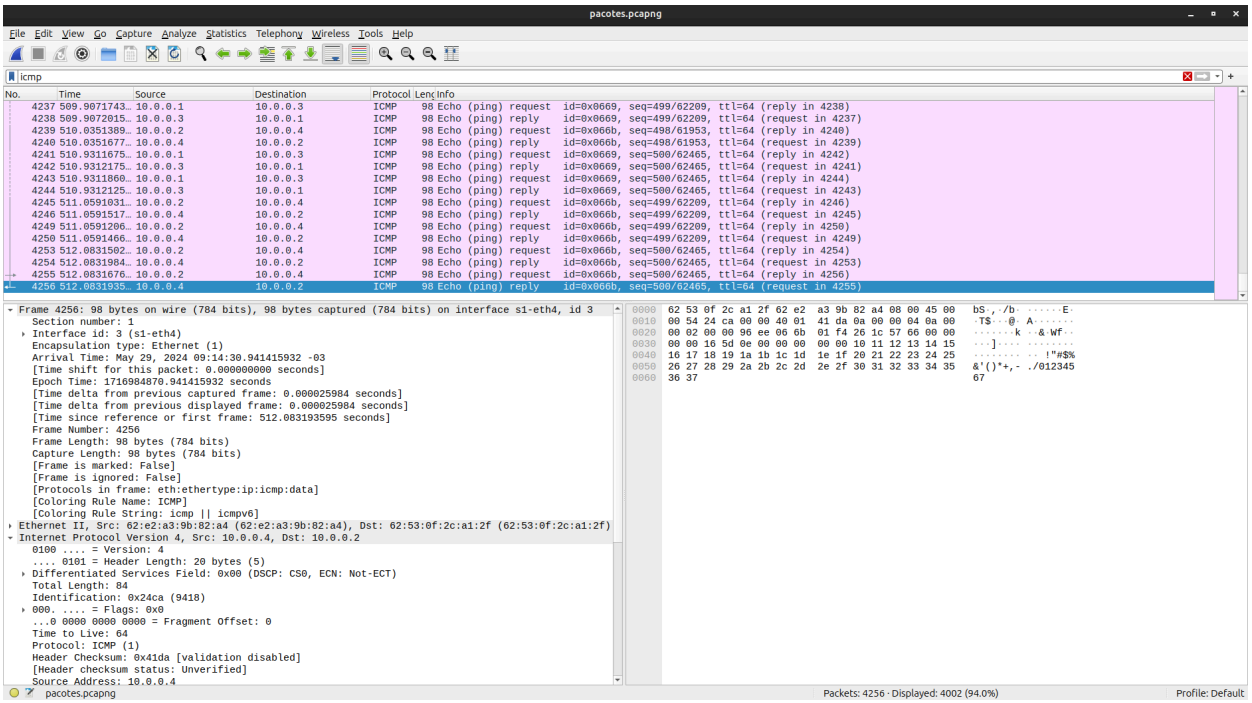

Figura 2.Resultados wireshark

6. Conclusão

Diante do observado, pode-se concluir que é uma taxa relativamente alta para um tempo relativamente baixo, isso se deve principalmente ao teste ser feito localmente, assim evitando a complexificação do trajeto. Além disso, para efeito de comparação, observou-se os resultados obtidos com o resultado de outros alunos e obteve-se o seguinte resultado:

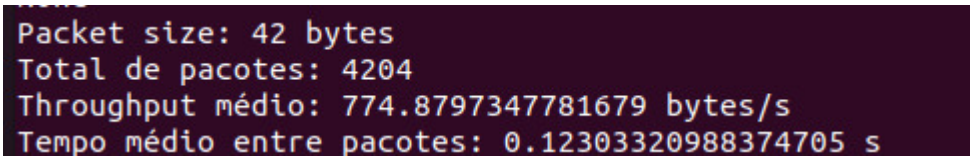

Figura 3.Resultados colegas de turma

Acima pode-se observar que os resultados são coerentes e seguem o mesmo padrão. Logo, acredita-se que a metodologia utilizada foi correta e que não houve complicações na utilização do software.

7. Referências

As principais referências:

a. *WIRESHARK. Wireshark GitLab.* Disponível em: https://gitlab.com/wireshark/wireshark/-/wikis/home. Acesso em: 12 jun. 2024.
b. *VIVA O LINUX. Servidor TCP/IP em C.* Disponível em: https://www.vivaolinux.com.br/script/Servidor-TCP-IP-em-C. Acesso em: 08 abr. 2024.
c. *MININET*. Disponível em: https://mininet.org/. Acesso em: 12 jun. 2024.
d. *GRINBERG*, Estefania. How to use Scapy: Python Networking. Free Code Camp. Disponível em: https://www.freecodecamp.org/news/how-to-use-scapy-python-networking/. Acesso em: 12 jun. 2024.

Bem como as aulas da disciplina de redes.