Relatório Trabalho 1 - Laboratório de redes de computadores

Deivid Santos, Henrique Andreata, João Vitor Pioner

1. Topologia

Foi escolhida uma topologia com 16 endereços de sub-rede, todos foram utilizados, ficando com CIDR 130.10.0.0/20. Existe um total de 6 roteadores, 3 hubs, 5 switches 14 hosts e um ponto de conexão simulando uma conexão externa com a internet.

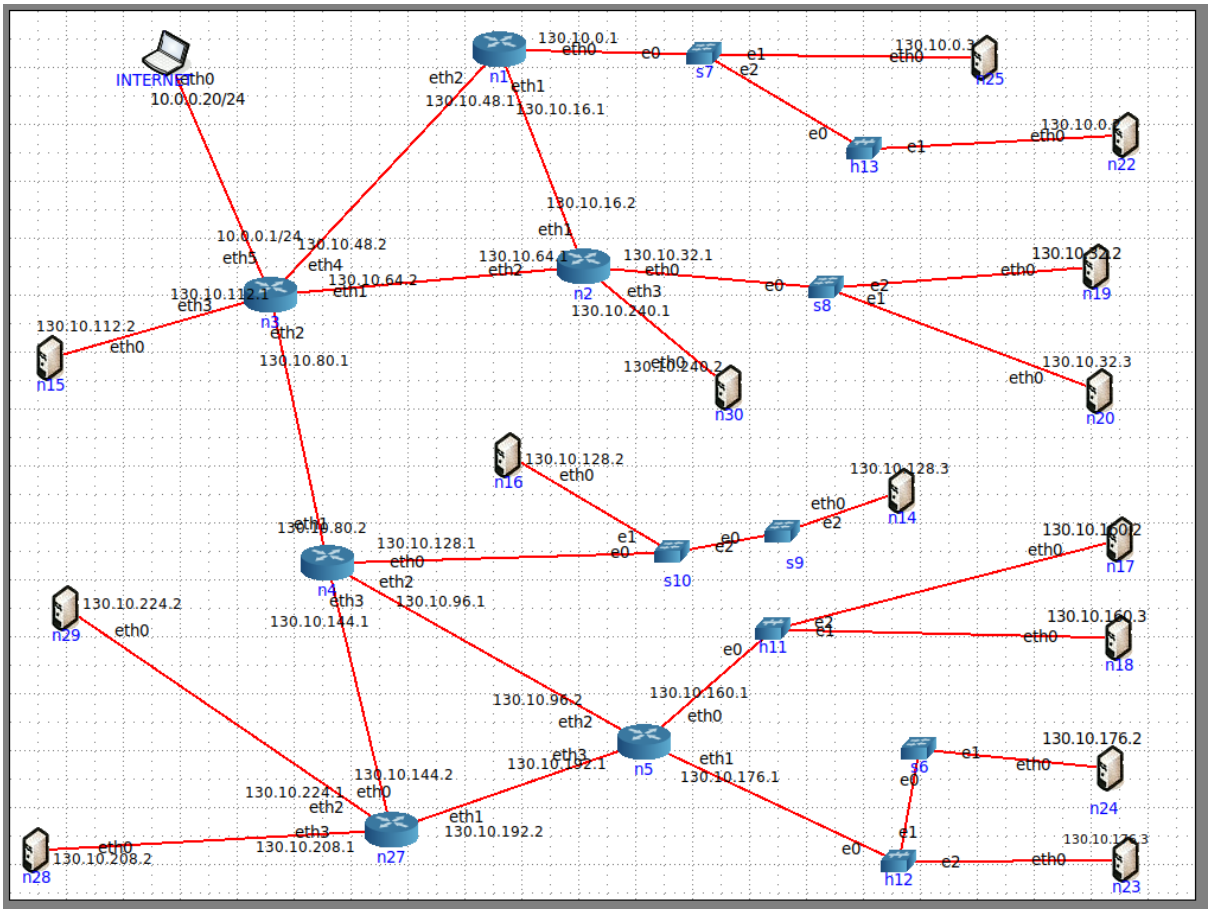

Figura 1 - Topologia

2. Ping

2.1. Nodo 25 -> Nodo 23

Na sequência de comandos a seguir nas figuras 2 e 3 é possível notar que a tabela arp está vazia no resultado do primeiro comando, depois é executado o comando de ping do nodo 25 para o nodo 23 e funciona corretamente, e então verificando a tabela arp novamente notamos que agora a tabela arp está preenchida com o endereço dos roteadores conectados em cada nodo.

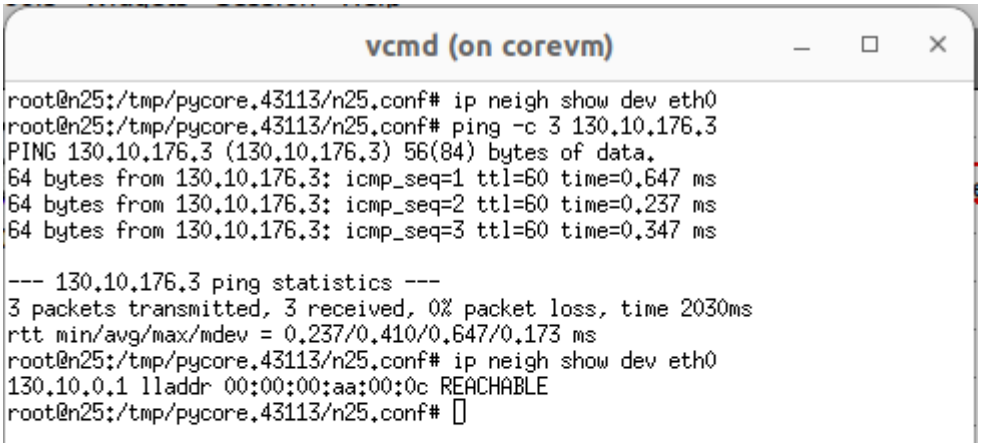

Figura 2 - Nodo origem do ping

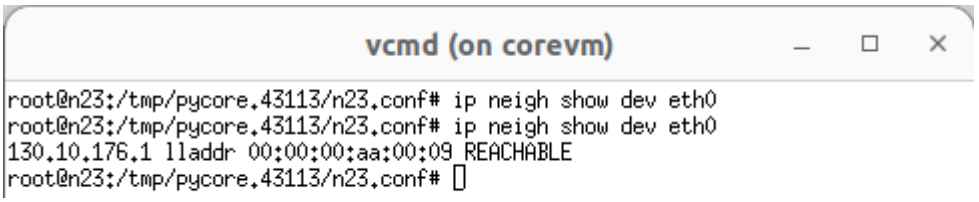

Figura 3 - nodo destino do ping

Nas figuras 4 e 5 estão os pacotes ARP e ICMP transferidos durante a execução do ping anterior, como a tabela desses nodos está vazia, os dois executam um broadcast para descobrir qual é o endereço MAC da máquina que o pacote será enviado, como o default do nodo 25 é o ip 130.10.0.1 ele tenta descobrir qual o Mac responsável por esse IP.

Depois são executados os comandos de envio do ping com o ICMP echo request e reply, as últimas linhas representam o retorno do ping, onde o roteador está tentando descobrir qual o Mac do IP 130.10.0.3, que é o nodo atual.

O cenário se repete parecido no nodo 23, porém com a ordem ao contrário, primeiro o roteador descobrindo o nodo por ARP e depois o nodo conhecendo o roteador.

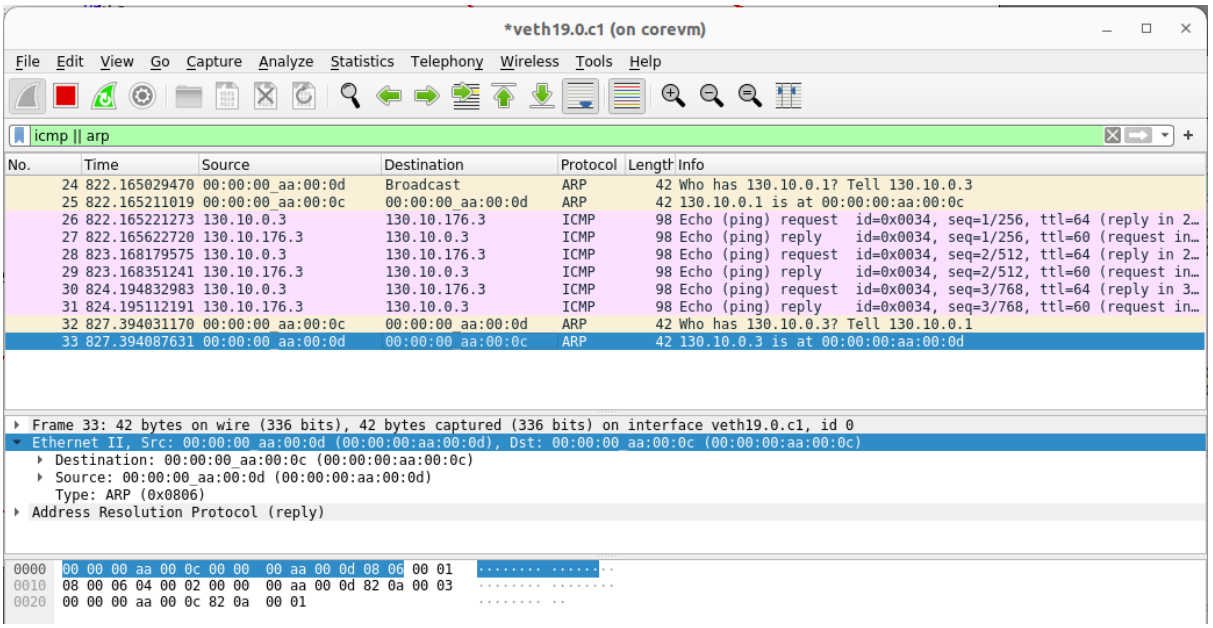

Figura 4 - Wireshark nodo origem

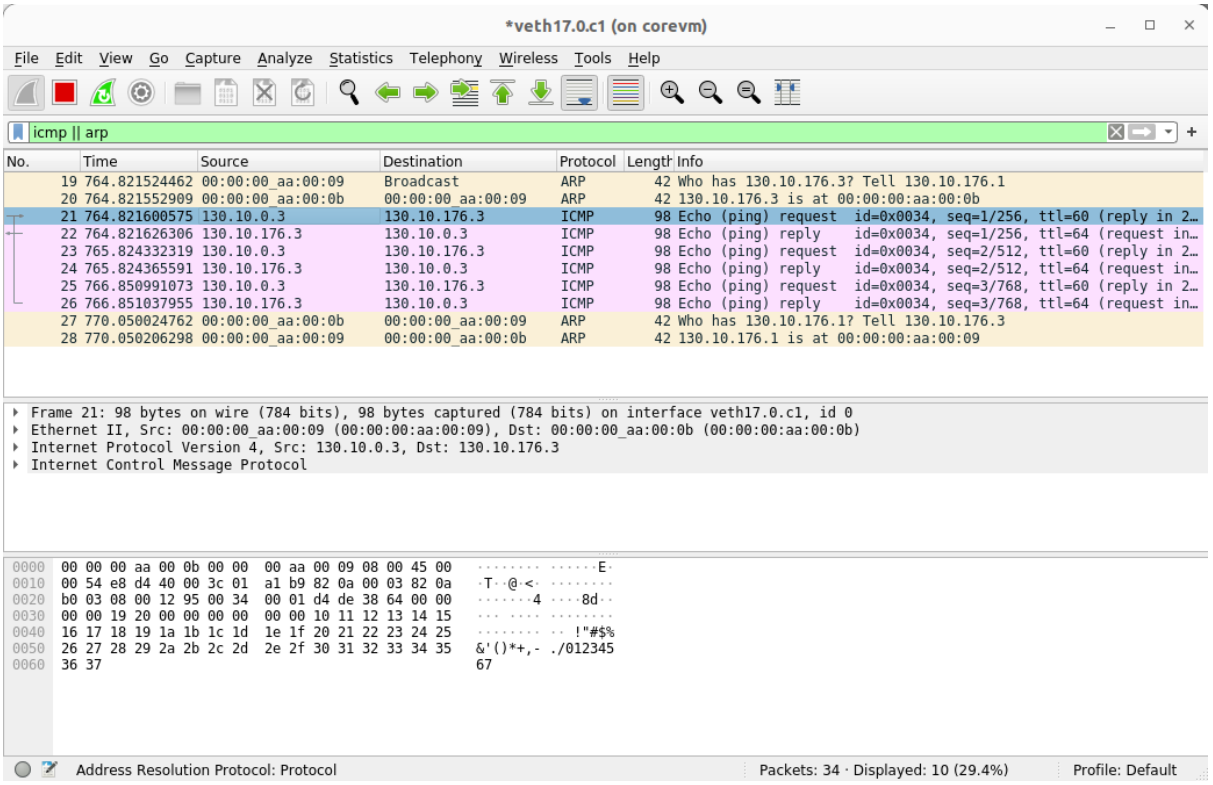

Figura 5 - Wireshark nodo destino

2.2. Nodo 15 -> Nodo 17

Para complementar o projeto, foi realizado um segundo ping do nodo 15 até o nodo 17 e o comportamento se manteve igual ao primeiro exemplo. Na imagem 6 abaixo, no lado esquerdo está o terminal e wireshark do nodo origem e na direita está o terminal e wireshark do nodo destino.

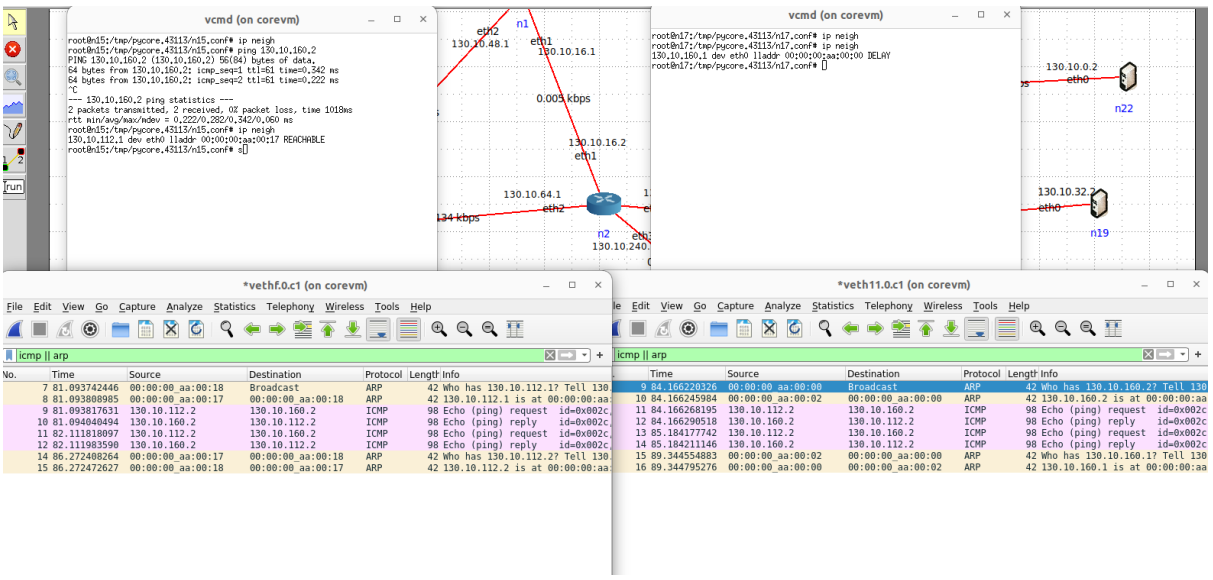

Figura 6 - Segundo ping

2.2.1. Flooding

Ao realizar o segundo ping do nodo 15 ao 17 utilizando o parâmetro de flooding conseguimos visualizar o caminho feito e notar o comportamento de um hub, que simplesmente espalha a request para todos os nodos conectados a ele, conforme é possível ver na figura 7.

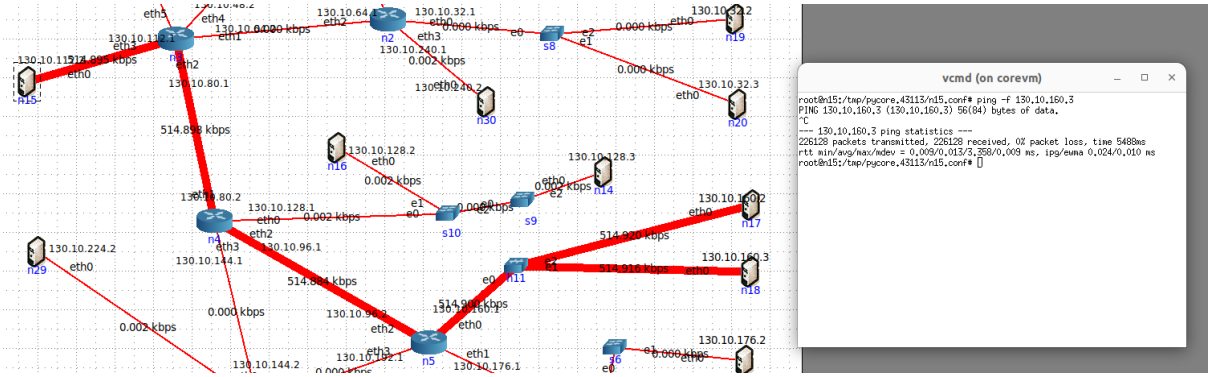

Figura 7 - Flooding

3. Traceroute

3.1. Nodo 29 -> nodo 20

Executando o comando traceroute do nodo 29 até o nodo 30 obtemos o resultado visto nas imagens 8 e 9, com 5 saltos de distância. Assim como no ping, inicialmente não temos nada na tabela arp e em seguida, após a execução do comando o roteador diretamente conectado foi adicionado a tabela arp do nodo.

```
vcmd (on corevm)
root@n29:/tmp/pycore.43113/n29.conf# ip neigh
root@n29:/tmp/pycore.43113/n29.conf# traceroute 130.10.240.2
traceroute to 130.10.240.2 (130.10.240.2), 30 hops max, 60 byte packets
 1  130.10.224.1 (130.10.224.1)  0.255 ms  0.035 ms  0.041 ms
 2  130.10.144.1 (130.10.144.1)  0.142 ms  0.060 ms  0.197 ms
 3  130.10.80.1 (130.10.80.1)  0.217 ms  0.078 ms  0.104 ms
 4  130.10.64.1 (130.10.64.1)  0.168 ms  0.115 ms  0.060 ms
 5  130.10.240.2 (130.10.240.2)  0.136 ms  0.074 ms  0.074 ms
root@n29:/tmp/pycore.43113/n29.conf# ip neigh
130.10.224.1 dev eth0 lladdr 00:00:00:aa:00:21 REACHABLE
root@n29:/tmp/pycore.43113/n29.conf#
```

Figura 8 - Nodo origem do traceroute

```
vcmd (on corevm)
root@n30:/tmp/pycore.43113/n30.conf# ip neigh
root@n30:/tmp/pycore.43113/n30.conf# ip neigh
130.10.240.1 dev eth0 lladdr 00:00:00:aa:00:25 DELAY
root@n30:/tmp/pycore.43113/n30.conf#
```

Figura 9 - Nodo destino do traceroute

Na figura 10 abaixo é possível notar o comportamento do traceroute em mais detalhes, onde temos vários timeouts devido ao comportamento onde é enviado primeiramente uma requisição com TTL 1 e assim incrementando a cada salto para obter a distância e todos os nodos do caminho.

Na figura 10 está destacado em azul na parte de baixo o time to live da primeira request e então obtemos um timeout, a partir disso, o TTL vai ser incrementado para 2 e uma request nova é feita para descobrir o próximo salto.

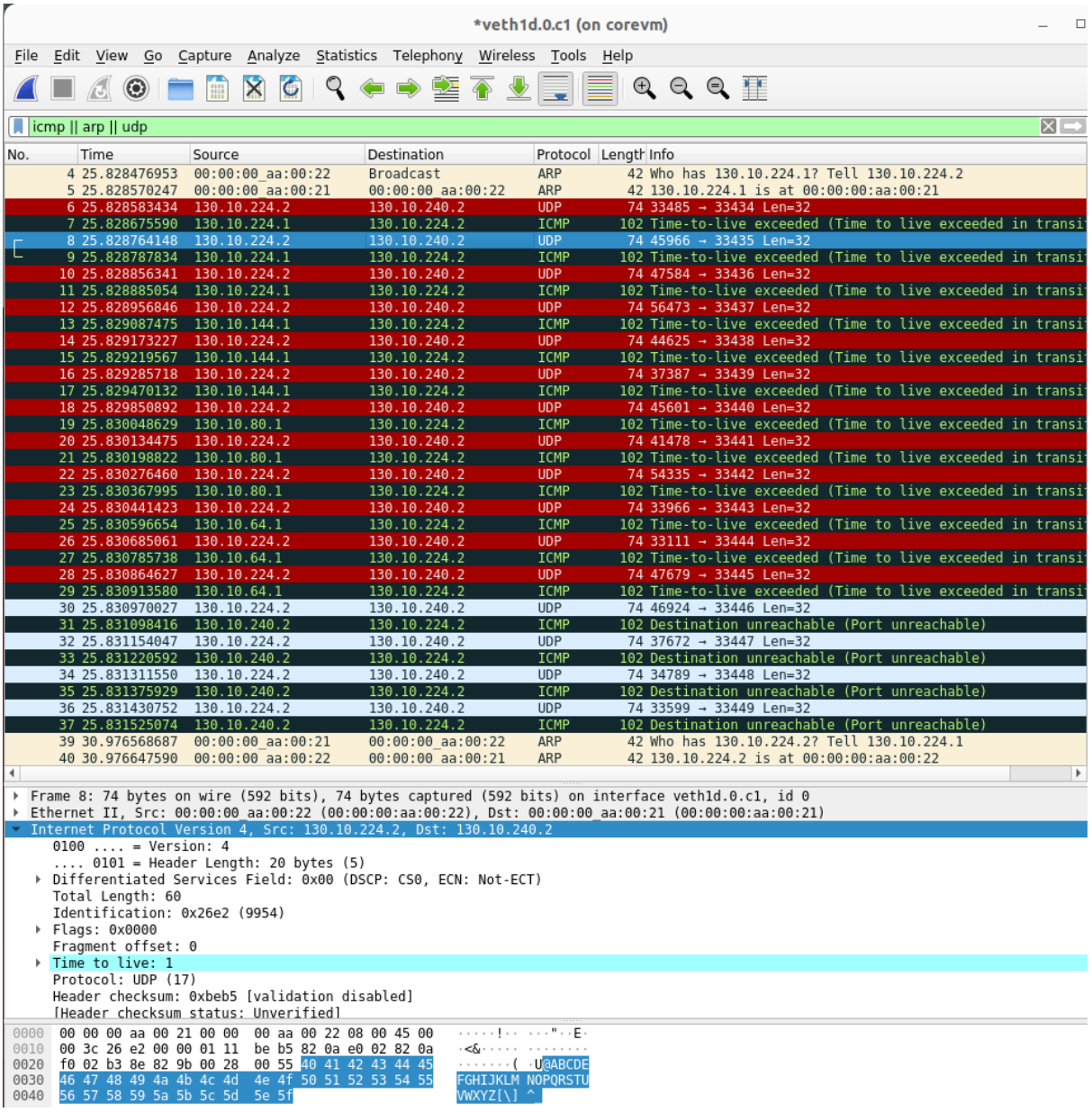

Figura 10 - Wireshark do nodo origem do traceroute

Na figura 11 abaixo está o TTL final da requisição, após os incrementos de cada tentativa de obter o resultado final, o ttl 6 chegou ao destino buscado e o comando foi finalizado.

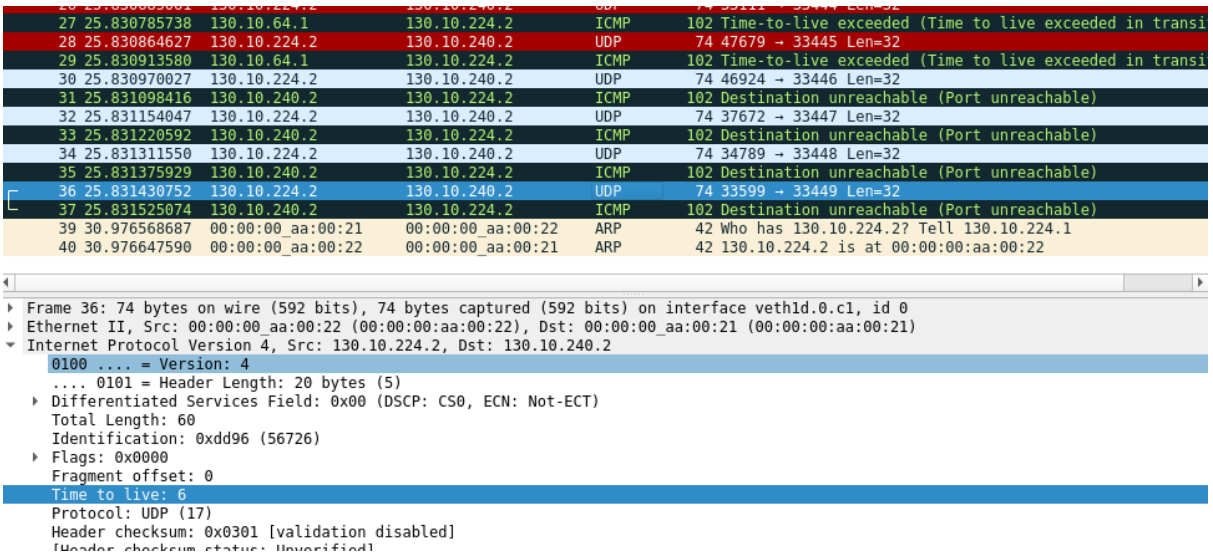

Figura 11 - Wireshark do nodo origem com ttl final.

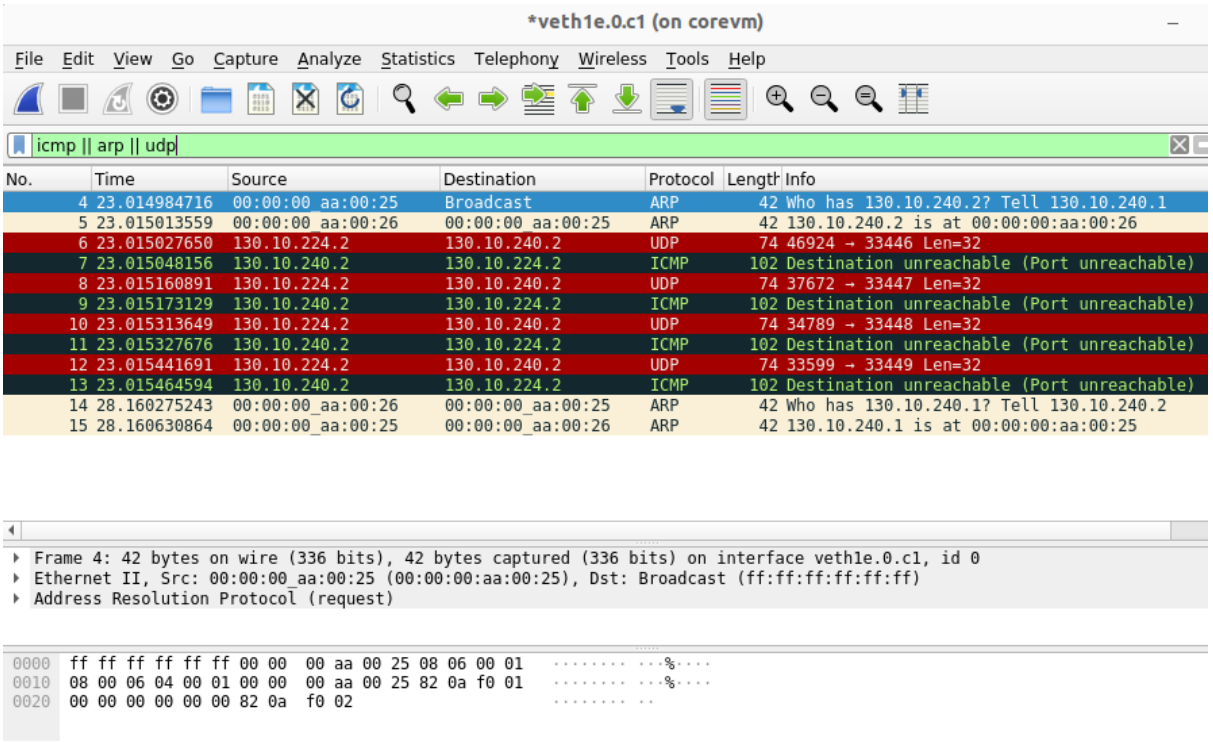

Figura 12 - Wireshark do nodo destino do traceroute.

Utilizando o comando ping com flooding conseguimos obter um resultado visual do caminho feito nesse traceroute.

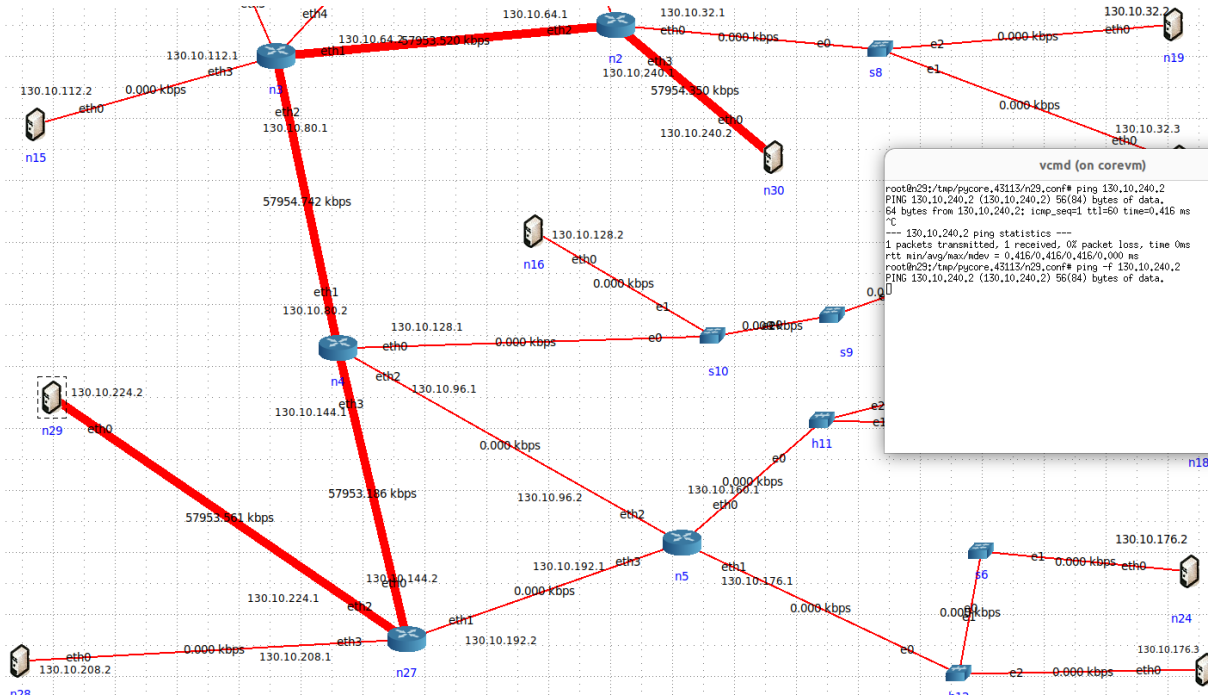

Figura 13 - Flooding do traceroute.

3.2. Nodo 14 -> nodo 28

No segundo traceroute do nodo 14 até o nodo 28 o comportamento se manteve igual ao primeiro exemplo. Na imagem 14 abaixo, no lado esquerdo está o terminal e wireshark do nodo origem e na direita está o terminal e wireshark do nodo destino.

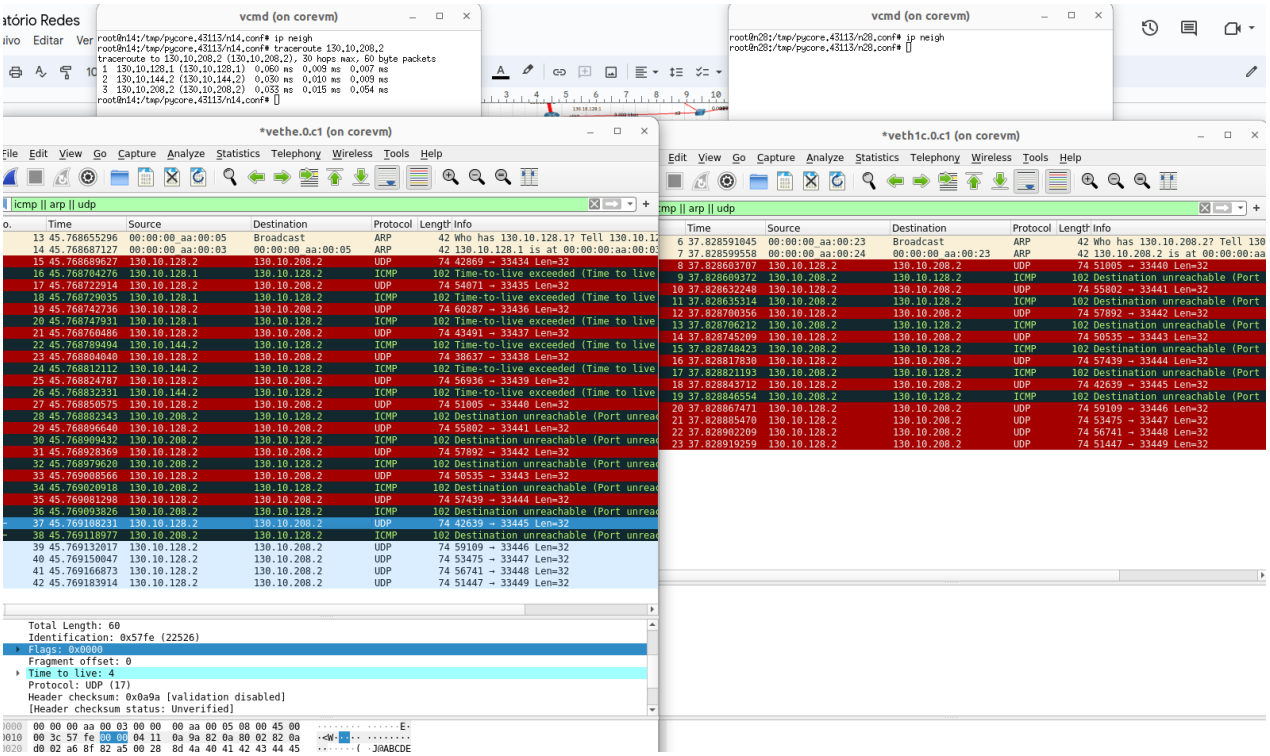

Figura 14 - Segundo traceroute.