



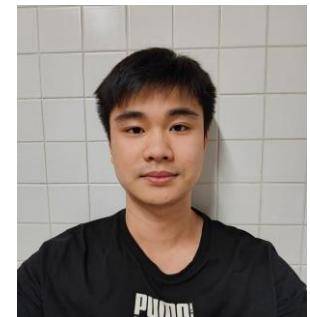
Licenciatura em Engenharia e Gestão de Sistemas de Informação

Universidade do Minho

Introdução às Redes de Computadores

2024/2025

Grupo 6 PL3



Afonso Leite
A108552
@uminho.pt

Rui Mendes
A108720
@uminho.pt

Zelong Ma
A108448
@uminho.pt



Índice

Introdução	1
1-Emulação de LANs Ethernet.....	1
Construção da Topologia	1
Atribuição de Endereços IP e Configuração	2
Testes de Conectividade e Captura de Tráfego.....	2
Captura com Wireshark.....	3
2- Interligação de LANs e redes IP	3
Tabela de Endereçamento das Redes LAN (Gama 10.6.0.0/19).....	4
Ligações entre Routers (Gama 192.168.0.0/24).....	4
Tabela de Encaminhamento das Redes LAN	6
Teste de Conectividade	10
Adição de mais dispositivos na LAN E utilizando a rede 192.168.100.0/26.....	12
3-DHCP–Dynamic Host Configuration Protocol.....	13
Processo de Atribuição de IP via DHCP	14
4-Uso das camadas de rede e transporte por parte das aplicações	14
5-Interligação via Network Address Translation.....	17
Relatório de Análise Técnica - Conexão FTP, HTTP e NAT	18

Índice de Ilustrações

Figura 1-Topologia emulada no CORE.....	1
Figura 2-Resultado de ping no Wireshark	2
Figura 3-Resultado de ping no Wireshark com filtro	3
Figura 4-Ping para analise no Wireshark	3
Figura 5-Ativação zebra	9
Figura 6- Router RA	9
Figura 7- Router RB	9
Figura 8- Router RC	9
Figura 9- Router RD	9
Figura 10- Router RE	9
Figura 11- Router R1	9
Figura 12 Router R2	10
Figura 13- Router R3	10
Figura 14- Router R4	10
Figura 15-Conectividade entre RA-RB1.....	10
Figura 16-Conectividade entre RA-RC.....	10
Figura 17-Conectividade entre RA-RB2	10
Figura 18-Conectividade entre RA-RE.....	10
Figura 19-Conectividade entre RA-RD	11
Figura 20-Conectividade entre RB1-RC	11
Figura 21-Conectividade entre RB1-RB2	11
Figura 22-Conectividade entre RB1-RD	11
Figura 23-Conectividade entre RB1-RE	11
Figura 24-Conectividade entre RB2-RC	11
Figura 25-Conectividade entre RB2-RD	11
Figura 26-Conectividade entre RB2-RE	11
Figura 27-Conectividade entre RC-RD	12
Figura 28-Conectividade entre RC-RE	12
Figura 29-Conectividade entre RD-RE	12
Figura 30-Nova rede na LanE	12
Figura 31-Topologia final 2	13
Figura 32-Ativação DHTP	13
Figura 33-Configuração DHCP	14
Figura 34-Leitura DHCP no Wireshark	14
Figura 35-Ativação http	15
Figura 36-Testar conectividade http	15
Figura 37-Wireshark http	15
Figura 38-Formulário de login HTML.....	16
Figura 39-Links2 execução de formulário	16
Figura 40-Captura HTTP POST do formulário de login.....	16
Figura 41-Configuração Firewall da NAT	17
Figura 42-Topologia de NAT e FTP	17
Figura 43-Conexão FTP;HTTP;NAT	18
Figura 44-Fluxo de comunicação wireshark NAT	18



Introdução

O presente trabalho prático tem como principal finalidade o desenvolvimento de competências na área das redes de computadores, através da conceção e implementação de um projeto baseado nas tecnologias Ethernet e TCP/IP.

Recorreu-se ao emulador de redes CORE (Common Open Research Emulator) para planear, configurar e testar uma rede de computadores funcional, aplicando os conhecimentos adquiridos nas aulas teóricas e teórico-práticas da unidade curricular.

Complementarmente, foi utilizada a ferramenta Wireshark para realizar a captura e análise de tráfego, permitindo observar o comportamento dos principais protocolos de comunicação nas diferentes camadas da pilha TCP/IP.

Este trabalho permitiu consolidar os conhecimentos técnicos adquiridos, promovendo também o desenvolvimento da capacidade de diagnóstico, resolução de problemas e trabalho colaborativo, num contexto de simulação próximo da realidade.

1-Emulação de LANs Ethernet

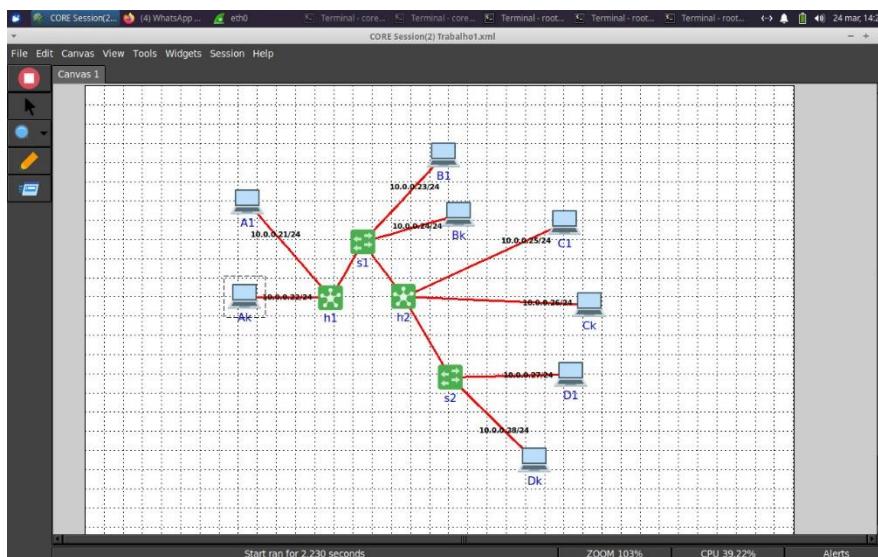


Figura 1-Topologia emulada no CORE

Neste exercício foi construída, no emulador CORE, uma rede local (LAN) com o objetivo de estudar o comportamento de dispositivos de camada 2 (Hubs e Switches), utilizando ferramentas de diagnóstico como o ping e o Wireshark para analisar os protocolos ARP e ICMP.

Construção da Topologia

A topologia emulada segue o modelo proposto no enunciado e inclui dois Hubs (H1 e H2) e dois Switches (S1 e S2), interligando um total de oito PCs, com pares de nós ligados a cada Hub ou Switch. Os PCs estão nomeados de forma a evidenciar a associação a cada segmento da rede:

Ao Hub H1 estão ligados os nós A1, Ak e B1, Bk, via Switch S1.

Ao Hub H2 estão ligados os nós C1, Ck e D1, Dk, via Switch S2.

Todos os nós têm endereços IP do mesmo espaço de endereçamento (ex.: 10.0.0.X/24), assegurando que pertencem à mesma rede e que existe conectividade IP entre eles.



Atribuição de Endereços IP e Configuração

Os endereços IP foram atribuídos automaticamente pelo CORE. Verificou-se que todos os terminais pertencem à mesma sub-rede, permitindo comunicação sem necessidade de gateways ou routers.

Comportamento dos Switches: Os Switches aprendem os endereços MAC com base no tráfego recebido e constroem uma tabela de encaminhamento. Quando recebem um ARP Request (que é um broadcast), este é transmitido a todas as portas. No entanto, respostas ARP ou pacotes ICMP unicast são enviados apenas à porta correta. Isso reduz significativamente o tráfego na rede e aumenta a eficiência.

Testes de Conectividade e Captura de Tráfego

Foram realizados testes de conectividade entre os vários nós da rede utilizando o comando ping. Verificou-se que os pacotes ICMP são corretamente enviados e recebidos entre todos os nós da rede, demonstrando a correta configuração IP e funcionamento da camada de ligação de dados (camada 2).

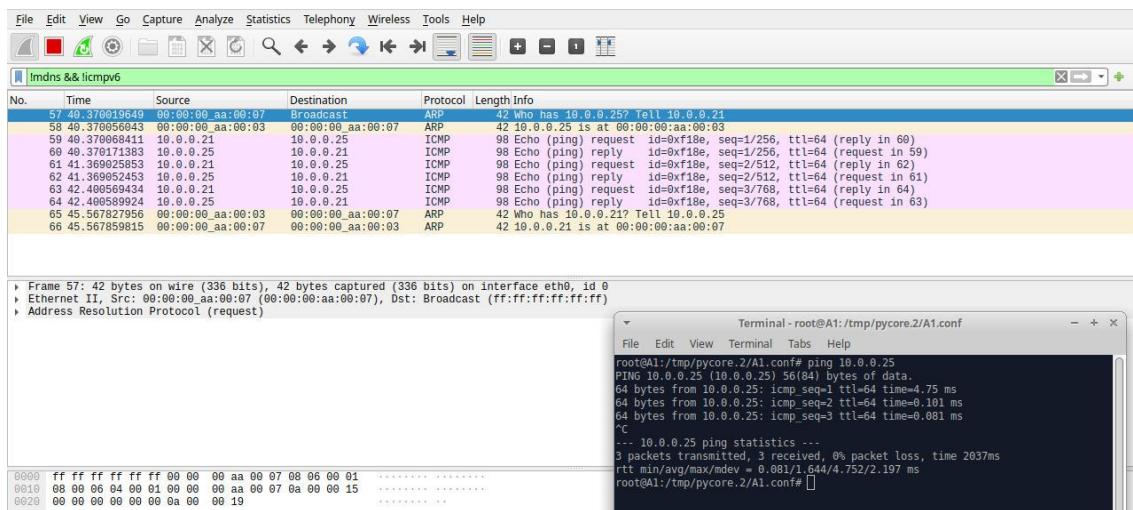


Figura 2-Resultado de ping no Wireshark

Durante a análise do tráfego de rede, foi possível observar uma troca de mensagens ARP entre dois dispositivos da rede local. Inicialmente, o host com endereço IP 10.0.0.21 envia uma solicitação ARP (**ARP Request**) a perguntar "Who has 10.0.0.25? Tell 10.0.0.21", ou seja, está a tentar descobrir o endereço MAC que corresponde ao IP 10.0.0.25. Em seguida, o host com IP 10.0.0.25 responde com uma mensagem ARP (**ARP Reply**) a informar que "10.0.0.25 is at 00:00:00:aa:00:03", revelando o seu endereço físico. Posteriormente, ocorreu uma nova solicitação ARP originada no host 10.0.0.25, agora a questionar "Who has 10.0.0.21? Tell 10.0.0.25", invertendo os papéis da consulta. O host 10.0.0.21 responde "10.0.0.21 is at 00:00:00:aa:00:07". Essa troca de ARP é fundamental para que os dispositivos possam associar os endereços IP aos endereços MAC correspondentes, permitindo a comunicação correta em uma rede local baseada em Ethernet.

Na imagem seguinte entramos no terminal do PC B1 e fizemos ping para o PC CK (ping 10.0.0.28). Em seguida, aplicamos o filtro !mdns && ! icmpv6 no Wireshark, para ignorarmos os pacotes que não nos interessam e ficamos à escuta a partir do PC D1. Este foi o nosso resultado:

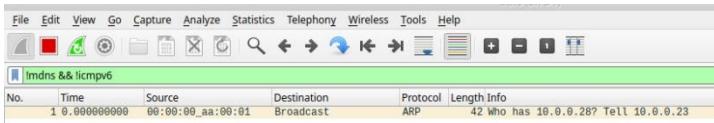


Figura 3-Resultado de ping no Wireshark com filtro

```

root@B1:/tmp/pycore.2/B1.conf# ping 10.0.0.28
PING 10.0.0.28 (10.0.0.28) 56(84) bytes of data.
64 bytes from 10.0.0.28: icmp seq=1 ttl=64 time=1.31 ms
64 bytes from 10.0.0.28: icmp seq=2 ttl=64 time=0.084 ms
64 bytes from 10.0.0.28: icmp seq=3 ttl=64 time=0.101 ms
64 bytes from 10.0.0.28: icmp seq=4 ttl=64 time=0.072 ms
...
. 10.0.0.28 ping statistics ...
4 packets transmitted, 4 received, 0% packet loss, time 3041ms
rtt min/avg/max/mdev = 0.072/0.392/1.315/0.531 ms
root@B1:/tmp/pycore.2/B1.conf#

```

Figura 4-Ping para analise no Wireshark

Captura com Wireshark

As capturas de tráfego com o Wireshark permitiram analisar os seguintes comportamentos:

Os Hubs, por não terem qualquer tipo de inteligência ou tabela de endereços, replicam todos os pacotes recebidos para todas as suas portas. Assim, ao enviar um ping ou um ARP Request de um PC ligado a um Hub, todos os restantes dispositivos ligados a esse mesmo Hub recebem esses pacotes, mesmo que não sejam os destinatários.

Usando os Switches, a captura de tráfego mostrou que os pacotes foram encaminhados apenas para o destinatário correto, reduzindo o tráfego desnecessário. Apenas o primeiro pacote ARP foi transmitido para toda a rede, permitindo a resolução de endereços MAC/IP. Após isso, os Switches aprenderam os endereços e encaminharam os pacotes diretamente.

2- Interligação de LANs e redes IP

Rede	Cálculo do número de endereços a atribuir a cada rede	M = 19 Resultado
A	$2^{32-(19+1)}-3$	4093
B	$2^{32-(19+2)}+2^{32-(M+4)}-6$	2554
C	$2^{32-(19+3)}-3$	1021
D	$2^{32-(19+5)}-3$	253
E	$2^{32-(19+5)}-3$	253



Tabela de Endereçamento das Redes LAN (Gama 10.6.0.0/19)

	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1
	/19	/20	/21	22/	/23	/24	/25	/26	/27	/28	/29	/30	/31	/32
A	0	0	H	H	H	H	H	H	H	H	H	H	H	H
B1	0	1	0	H	H	H	H	H	H	H	H	H	H	H
C	0	1	1	0	H	H	H	H	H	H	H	H	H	H
B2	0	1	1	1	0	H	H	H	H	H	H	H	H	H
E	0	1	1	1	1	0	H	H	H	H	H	H	H	H
D	0	1	1	1	1	1	H	H	H	H	H	H	H	H

H - host

Rede	Número de Endereços	Endereço de Rede	Endereço de Difusão	Máscara	Gama de Endereços
A	4096	10.6.0.0	10.6.15.255	/20	10.6.0.1 - 10.6.15.254
B1	2048	10.6.16.0	10.6.23.255	/21	10.6.16.1 - 10.6.23.254
C	1024	10.6.24.0	10.6.27.255	/22	10.6.24.1 - 10.6.27.254
B2	512	10.6.28.0	10.6.29.255	/23	10.6.28.1 - 10.6.29.254
E	256	10.6.30.0	10.6.30.255	/24	10.6.30.1 - 10.6.30.254
D	256	10.6.31.0	10.6.31.255	/24	10.6.31.1 - 10.6.31.254

Ligações entre Routers (Gama 192.168.0.0/24)

Para o encaminhamento, optamos por seguir a métrica do menor número de saltos. Algumas rotas podem ter mais que um caminho com o mesmo número de saltos. Colocamos na tabela identificado com um * aqueles que são os caminhos alternativos. Esta redundância contribui para uma maior resiliência da rede, permitindo contornar falhas e manter a conectividade através de rotas alternativas.



	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1
	/25	/26	/27	/28	/29	/30	/31	/32
RA-R1	0	0	0	0	0	0	H	H
RB-R2	0	0	0	0	0	1	H	H
R1-R2	0	0	0	0	1	0	H	H
R2-R3	0	0	0	0	1	1	H	H
R2-R4	0	0	0	1	0	0	H	H
R4-R3	0	0	0	1	0	1	H	H
R4-RD	0	0	0	1	1	1	H	H
R1-R3	0	0	1	0	0	0	H	H
R1-RC	0	0	1	0	0	1	H	H
R3-RC	0	0	1	0	1	1	H	H
R3-RE	0	0	1	1	1	1	H	H
RC-RE	0	1	0	0	0	0	H	H

	Rede	Endereço de Rede	Endereço de Difusão	Máscara	Gama de Endereço
1	RA-R1	192.168.0.0	192.168.0.3	/30	192.168.0.1 - 192.168.0.2
2	RB-R2	192.168.0.4	192.168.0.7	/30	192.168.0.5 - 192.168.0.6
3	R1-R2	192.168.0.8	192.168.0.11	/30	192.168.0.9 - 192.168.0.10
4	R2-R3	192.168.0.12	192.168.0.15	/30	192.168.0.13 - 192.168.0.14
5	R2-R4	192.168.0.16	192.168.0.19	/30	192.168.0.17 - 192.168.0.18
6	R4-R3	192.168.0.20	192.168.0.23	/30	192.168.0.21 - 192.168.0.22
7	R4-RD	192.168.0.24	192.168.0.27	/30	192.168.0.25 - 192.168.0.26



8	R1-R3	192.168.0.28	192.168.0.31	/30	192.168.0.29 - 192.168.0.30
9	R1-RC	192.168.0.32	192.168.0.35	/30	192.168.0.33 - 192.168.0.34
10	R3-RC	192.168.0.36	192.168.0.39	/30	192.168.0.37 - 192.168.0.38
11	R3-RE	192.168.0.40	192.168.0.43	/30	192.168.0.41 - 192.168.0.42
12	RC-RE	192.168.0.44	192.168.0.47	/30	192.168.0.45 - 192.168.0.46
13	RD-RE	192.168.0.48	192.168.0.51	/30	192.168.0.49 - 192.168.0.50

Tabela de Encaminhamento das Redes LAN

LAN A	Rede Destino	Máscara	Interface de Saída	Próximo Nó
A	10.6.0.0	/20	10.6.0.1	-----
B1	10.6.16.0	/21	192.168.0.1	192.168.0.2
C	10.6.24.0	/22	192.168.0.1	192.168.0.2
B2	10.6.28.0	/23	192.168.0.1	192.168.0.2
D	10.6.31.0	/24	192.168.0.1	192.168.0.2
E	10.6.30.0	/24	192.168.0.1	192.168.0.2

LAN B	Rede Destino	Máscara	Interface de Saída	Próximo Nó
A	10.6.0.0	/20	192.168.0.5	192.168.0.6
B1	10.6.16.0	/21	10.6.16.1	-----
C	10.6.24.0	/22	192.168.0.5	192.168.0.6
B2	10.6.28.0	/23	10.6.28.1	-----
D	10.6.31.0	/24	192.168.0.5	192.168.0.6
E	10.6.30.0	/24	192.168.0.5	192.168.0.6

LAN C	Rede Destino	Máscara	Interface de Saída	Próximo Nó
A	10.6.0.0	/20	192.168.0.34	192.168.0.33
B1	10.6.16.0	/21	192.168.0.34	192.168.0.33
B1*	10.6.28.0	/23	192.168.0.34	192.168.0.33



C	10.6.24.0	/22	10.6.24.1	-----
B2	10.6.28.0	/23	192.168.0.38	192.168.0.37
B2*	10.6.28.0	/23	192.168.0.38	192.168.0.37
D	10.6.31.0	/24	192.168.0.45	192.168.0.46
E	10.6.30.0	/24	192.168.0.45	192.168.0.46

LAN D	Rede Destino	Máscara	Interface de Saída	Próximo Nó
A	10.6.0.0	/20	192.168.0.26	192.168.0.25
A*	10.6.0.0	/20	192.168.0.49	192.168.0.50
B1	10.6.16.0	/21	192.168.0.26	192.168.0.25
C	10.6.24.0	/22	192.168.0.49	192.168.0.50
B2	10.6.28.0	/23	192.168.0.26	192.168.0.25
D	10.6.31.0	/24	10.6.31.1	-----
E	10.6.30.0	/24	192.168.0.49	192.168.0.50

LAN E	Rede Destino	Máscara	Interface de Saída	Próximo Nó
A	10.6.0.0	/20	192.168.0.42	192.168.0.41
A*	10.6.0.0	/20	192.168.0.46	192.168.0.45
B1	10.6.16.0	/21	192.168.0.42	192.168.0.41
C	10.6.24.0	/22	192.168.0.46	192.168.0.45
B2	10.6.28.0	/23	192.168.0.42	192.168.0.41
D	10.6.31.0	/24	192.168.0.50	192.168.0.49
E	10.6.30.0	/24	10.6.30.1	-----

LAN R1	Rede Destino	Máscara	Interface de Saída	Próximo Nó
A	10.6.0.0	/20	192.168.0.2	192.168.0.1
B1	10.6.16.0	/21	192.168.0.9	192.168.0.10
C	10.6.24.0	/22	192.168.0.33	192.168.0.34
B2	10.6.28.0	/23	192.168.0.9	192.168.0.10



D	10.6.31.0	/24	192.168.0.29	192.168.0.30
E	10.6.30.0	/24	192.168.0.29	192.168.0.30

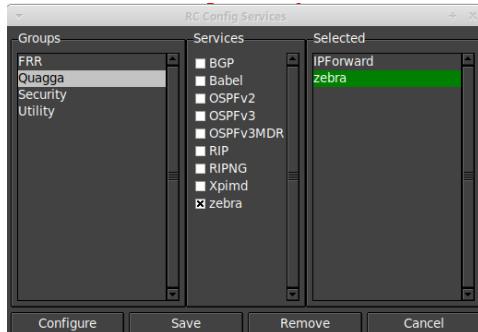
LAN R2	Rede Destino	Máscara	Interface de Saída	Próximo Nô
A	10.6.0.0	/20	192.168.0.10	192.168.0.9
B1	10.6.16.0	/21	192.168.0.6	192.168.0.5
C	10.6.24.0	/22	192.168.0.10	192.168.0.9
B2	10.6.28.0	/23	192.168.0.6	192.168.0.5
C*	10.6.24.0	/22	192.168.0.13	192.168.0.14
D	10.6.31.0	/24	192.168.0.17	192.168.0.18
E	10.6.30.0	/24	192.168.0.13	192.168.0.14

LAN R3	Rede Destino	Máscara	Interface de Saída	Próximo Nô
A	10.6.0.0	/20	192.168.0.30	192.168.0.29
B1	10.6.16.0	/21	192.168.0.14	192.168.0.13
C	10.6.24.0	/22	192.168.0.37	192.168.0.38
B2	10.6.28.0	/23	192.168.0.14	192.168.0.13
D	10.6.29.0	/24	192.168.0.22	192.168.0.21
D*	10.6.31.0	/24	192.168.0.41	192.168.0.42
E	10.6.30.0	/24	192.168.0.41	192.168.0.42

LAN R4	Rede Destino	Máscara	Interface de Saída	Próximo Nô
A	10.6.0.0	/20	192.168.0.18	192.168.0.17
A*	10.6.0.0	/20	192.168.0.21	192.168.0.22
B1	10.6.16.0	/21	192.168.0.18	192.168.0.17
C	10.6.24.0	/22	192.168.0.21	192.168.0.22
B2	10.6.28.0	/23	192.168.0.18	192.168.0.17
D	10.6.31.0	/24	192.168.0.25	192.168.0.26
E	10.6.30.0	/24	192.168.0.21	192.168.0.22



E*	10.6.30.0	/24	192.168.0.25	192.168.0.26
----	-----------	-----	--------------	--------------



Desativado o encaminhamento dinâmico.

Nesta etapa, foi desativado o encaminhamento dinâmico por meio da desativação dos protocolos OSPFv2 (para IPv4) e OSPFv3 (para IPv6) em todos os routers da topologia. Essa medida teve como objetivo substituir o roteamento dinâmico por rotas estáticas, conforme o esquema de encaminhamento definido anteriormente.

Figura 5-Ativação zebra

Nas imagens abaixo encontram-se as configurações zebra de todos os routers para o encaminhamento estático.

```
zebra Config Service
Directories/Files Startup/Shutdown Validation
Directories and templates
Directories /usr/local/etc/quagga
Files /usr/local/etc/quagga/Quagga.conf
Rendered Template
interface eth1
ip address 10.6.0.1/20
!
interface eth2
ip address 192.168.0.1/30
!
ip route 10.6.16.0/21 192.168.0.2
ip route 10.6.28.0/23 192.168.0.2
ip route 10.6.24.0/22 192.168.0.2
ip route 10.6.31.0/24 192.168.0.2
ip route 10.6.30.0/24 192.168.0.2
```

Figura 6- Router RA

```
zebra Config Service
Directories/Files Startup/Shutdown Validation
Directories and templates
Directories /usr/local/etc/quagga
Files /usr/local/etc/quagga/Quagga.conf
Rendered Template
interface eth1
ip address 192.168.0.5/30
!
interface eth2
ip address 10.6.16.1/21
!
interface eth3
ip address 10.6.28.1/23
!
ip route 10.6.0.0/20 192.168.0.6
ip route 10.6.24.0/22 192.168.0.6
ip route 10.6.31.0/24 192.168.0.6
ip route 10.6.30.0/24 192.168.0.6
```

Figura 7- Router RB

```
zebra Config Service
Directories/Files Startup/Shutdown Validation
Directories and templates that will be rendered
Directories /usr/local/etc/quagga
Files /usr/local/etc/quagga/Quagga.conf
Rendered Template
interface eth0
ip address 192.168.0.34/30
!
interface eth1
ip address 192.168.0.45/30
!
interface eth2
ip address 192.168.0.38/30
!
interface eth3
ip address 10.6.24.1/22
!
ip route 10.6.0.0/20 192.168.0.33
ip route 10.6.16.0/21 192.168.0.33
ip route 10.6.28.0/23 192.168.0.37
ip route 10.6.31.0/24 192.168.0.46
ip route 10.6.30.0/24 192.168.0.46
```

Figura 8- Router RC

```
zebra Config Service
Directories/Files Startup/Shutdown Validation
Directories and templates that will be rendered
Directories /usr/local/etc/quagga
Files /usr/local/etc/quagga/Quagga.conf
Rendered Template
interface eth0
ip address 192.168.0.26/30
!
interface eth1
ip address 192.168.0.49/30
!
interface eth2
ip address 10.6.31.1/24
!
ip route 10.6.0.0/20 192.168.0.25
ip route 10.6.16.0/21 192.168.0.25
ip route 10.6.28.0/23 192.168.0.25
ip route 10.6.24.0/22 192.168.0.50
ip route 10.6.30.0/24 192.168.0.50
```

Figura 9- Router RD

```
zebra Config Service
Directories/Files Startup/Shutdown Validation
Directories and templates that will be rendered
Directories /usr/local/etc/quagga
Files /usr/local/etc/quagga/Quagga.conf
Rendered Template
interface eth0
ip address 192.168.0.42/30
!
interface eth1
ip address 192.168.0.50/30
!
interface eth2
ip address 192.168.0.46/30
!
interface eth3
ip address 10.6.30.1/24
!
ip route 10.6.0.0/20 192.168.0.41
ip route 10.6.16.0/21 192.168.0.41
ip route 10.6.28.0/23 192.168.0.41
ip route 10.6.24.0/22 192.168.0.45
ip route 10.6.31.0/24 192.168.0.49
```

Figura 10- Router RE

```
zebra Config Service
Directories/Files Startup/Shutdown Validation
Directories and templates that will be rendered
Directories /usr/local/etc/quagga
Files /usr/local/etc/quagga/Quagga.conf
Rendered Template
interface eth0
ip address 192.168.0.9/30
!
interface eth1
ip address 192.168.0.33/30
!
interface eth2
ip address 192.168.0.2/30
!
interface eth3
ip address 192.168.0.29/30
!
ip route 10.6.0.0/20 192.168.0.1
ip route 10.6.16.0/21 192.168.0.10
ip route 10.6.28.0/23 192.168.0.10
ip route 10.6.24.0/22 192.168.0.34
ip route 10.6.31.0/24 192.168.0.36
ip route 10.6.30.0/24 192.168.0.36
```

Figura 11- Router R1



```

zebra
Directories/Files Startup/Shutdown Validation
Directories and template
Directories /usr/local/etc/quagga
Files /usr/local/etc/quagga/Quagga.conf

Rendered Template
interface eth0
ip address 192.168.0.13/30
!
interface eth2
ip address 192.168.0.6/30
!
interface eth3
ip address 192.168.0.10/30
!
interface eth4
ip address 192.168.0.17/30
!
ip route 10.6.0.0/20 192.168.0.9
ip route 10.6.16.0/21 192.168.0.5
ip route 10.6.28.0/23 192.168.0.5
ip route 10.6.24.0/22 192.168.0.9
ip route 10.6.30.0/24 192.168.0.14
ip route 10.6.31.0/24 192.168.0.18

```

Figura 12 Router R2

```

zebra
Directories/Files Startup/Shutdown Validation
Directories and template
Directories /usr/local/etc/quagga
Files /usr/local/etc/quagga/Quagga.conf

Rendered Template
interface eth0
ip address 192.168.0.22/30
!
interface eth1
ip address 192.168.0.14/30
!
interface eth2
ip address 192.168.0.41/30
!
interface eth3
ip address 192.168.0.37/30
!
interface eth4
ip address 192.168.0.30/30
!
ip route 10.6.0.0/20 192.168.0.17
ip route 10.6.16.0/21 192.168.0.17
ip route 10.6.28.0/23 192.168.0.17
ip route 10.6.24.0/22 192.168.0.22
ip route 10.6.31.0/24 192.168.0.22
ip route 10.6.30.0/24 192.168.0.42

```

Figura 13- Router R3

```

zebra
Directories/Files Startup/Shutdown Validation
Directories and template
Directories /usr/local/etc/quagga
Files /usr/local/etc/quagga/Quagga.conf

Rendered Template
interface eth0
ip address 192.168.0.21/30
!
interface eth2
ip address 192.168.0.25/30
!
interface eth3
ip address 192.168.0.18/30
!
ip route 10.6.0.0/20 192.168.0.17
ip route 10.6.16.0/21 192.168.0.17
ip route 10.6.28.0/23 192.168.0.17
ip route 10.6.24.0/22 192.168.0.22
ip route 10.6.31.0/24 192.168.0.22
ip route 10.6.30.0/24 192.168.0.42

```

Figura 14- Router R4

Teste de Conectividade

Para testar a conectividade entre as diferentes redes, utilizamos os comandos ping e traceroute. No Wireshark para capturar e analisar o tráfego de rede no pc para o qual o ping é enviado.

```

Terminal - root@n15:/tmp/pycore.1/n15.conf
File Edit View Terminal Tabs Help
root@n15:/tmp/pycore.1/n15.conf# ping 10.6.16.3
PING 10.6.16.3 (10.6.16.3) 56(84) bytes of data.
64 bytes from 10.6.16.3: icmp_seq=1 ttl=60 time=18.0 ms
64 bytes from 10.6.16.3: icmp_seq=2 ttl=60 time=0.000 ms
^C
--- 10.6.16.3 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1010ms
rtt min/avg/max/mdev = 0.000/8.996/17.992/8.996 ms
root@n15:/tmp/pycore.1/n15.conf# traceroute 10.6.16.3
traceroute to 10.6.16.3 (10.6.16.3), 30 hops max, 60 byte packets
  1  10.6.0.1 (10.6.0.1)  1.999 ms  0.000 ms  0.000 ms
  2  192.168.0.2 (192.168.0.2)  0.000 ms  0.000 ms  0.000 ms
  3  192.168.0.10 (192.168.0.10)  2.000 ms  0.000 ms  0.000 ms
  4  192.168.0.5 (192.168.0.5)  0.999 ms  0.000 ms  0.000 ms
  5  10.6.16.3 (10.6.16.3)  0.000 ms  0.000 ms  0.000 ms
root@n15:/tmp/pycore.1/n15.conf#

```

Figura 15-Conectividade entre RA-RB1

```

Terminal - root@n15:/tmp/pycore.1/n15.conf
File Edit View Terminal Tabs Help
root@n15:/tmp/pycore.1/n15.conf# ping 10.6.24.3
PING 10.6.24.3 (10.6.24.3) 56(84) bytes of data.
64 bytes from 10.6.24.3: icmp_seq=1 ttl=61 time=16.3 ms
64 bytes from 10.6.24.3: icmp_seq=2 ttl=61 time=0.086 ms
^C
--- 10.6.24.3 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 0.086/8.203/16.321/8.117 ms
root@n15:/tmp/pycore.1/n15.conf# traceroute 10.6.24.3
traceroute to 10.6.24.3 (10.6.24.3), 30 hops max, 60 byte packets
  1  10.6.0.1 (10.6.0.1)  3.468 ms  2.644 ms  2.633 ms
  2  192.168.0.2 (192.168.0.2)  2.623 ms  2.540 ms  2.531 ms
  3  192.168.0.34 (192.168.0.34)  2.522 ms  1.532 ms  1.517 ms
  4  10.6.24.3 (10.6.24.3)  1.507 ms  1.365 ms  1.293 ms
root@n15:/tmp/pycore.1/n15.conf#

```

Figura 16-Conectividade entre RA-RC

```

Terminal - root@n15:/tmp/pycore.1/n15.conf
File Edit View Terminal Tabs Help
root@n15:/tmp/pycore.1/n15.conf# ping 10.6.28.1
PING 10.6.28.1 (10.6.28.1) 56(84) bytes of data.
64 bytes from 10.6.28.1: icmp_seq=1 ttl=61 time=6.00 ms
64 bytes from 10.6.28.1: icmp_seq=2 ttl=61 time=3.97 ms
^C
--- 10.6.28.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1005ms
rtt min/avg/max/mdev = 3.966/4.985/6.004/1.019 ms
root@n15:/tmp/pycore.1/n15.conf# traceroute 10.6.28.1
traceroute to 10.6.28.1 (10.6.28.1), 30 hops max, 60 byte packets
  1  10.6.0.1 (10.6.0.1)  1.795 ms  0.544 ms  0.552 ms
  2  192.168.0.2 (192.168.0.2)  0.545 ms  0.367 ms  0.355 ms
  3  192.168.0.10 (192.168.0.10)  0.345 ms  0.222 ms  0.211 ms
  4  10.6.28.1 (10.6.28.1)  0.181 ms  0.128 ms  0.114 ms
root@n15:/tmp/pycore.1/n15.conf#

```

Figura 17--Conectividade entre RA-RB2

```

Terminal - root@n15:/tmp/pycore.1/n15.conf
File Edit View Terminal Tabs Help
root@n15:/tmp/pycore.1/n15.conf# ping 10.6.30.2
PING 10.6.30.2 (10.6.30.2) 56(84) bytes of data.
64 bytes from 10.6.30.2: icmp_seq=1 ttl=60 time=7.85 ms
64 bytes from 10.6.30.2: icmp_seq=2 ttl=60 time=0.091 ms
^C
--- 10.6.30.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1003ms
rtt min/avg/max/mdev = 0.091/3.971/7.851/3.880 ms
root@n15:/tmp/pycore.1/n15.conf# traceroute 10.6.30.2
traceroute to 10.6.30.2 (10.6.30.2), 30 hops max, 60 byte packets
  1  10.6.0.1 (10.6.0.1)  0.999 ms  0.000 ms  0.000 ms
  2  192.168.0.2 (192.168.0.2)  0.000 ms  0.000 ms  0.000 ms
  3  192.168.0.30 (192.168.0.30)  1.000 ms  0.000 ms  0.000 ms
  4  192.168.0.42 (192.168.0.42)  0.000 ms  0.000 ms  0.000 ms
  5  10.6.30.2 (10.6.30.2)  0.000 ms  0.000 ms  0.000 ms
root@n15:/tmp/pycore.1/n15.conf#

```

Figura 18--Conectividade entre RA-RE



```
Terminal - root@n15:/tmp/pycore.1/n15.conf
File Edit View Terminal Tabs Help
root@n15:/tmp/pycore.1/n15.conf# ping 10.6.31.2
PING 10.6.31.2 (10.6.31.2) 56(84) bytes of data.
64 bytes from 10.6.31.2: icmp_seq=1 ttl=59 time=11.1 ms
64 bytes from 10.6.31.2: icmp_seq=2 ttl=59 time=2.40 ms
```
--- 10.6.31.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 2.397/6.753/11.110/4.356 ms
root@n15:/tmp/pycore.1/n15.conf# traceroute 10.6.31.2
traceroute to 10.6.31.2 (10.6.31.2), 30 hops max, 60 byte packets
1 10.6.0.1 (10.6.0.1) 3.875 ms 3.886 ms 3.899 ms
2 192.168.0.2 (192.168.0.2) 3.892 ms 3.893 ms 3.962 ms
3 192.168.0.30 (192.168.0.30) 3.964 ms 3.966 ms 3.968 ms
4 192.168.0.18 (192.168.0.18) 3.970 ms 3.972 ms 3.974 ms
5 192.168.0.26 (192.168.0.26) 3.977 ms 0.450 ms 0.409 ms
6 10.6.31.2 (10.6.31.2) 0.316 ms 0.086 ms 0.178 ms
root@n15:/tmp/pycore.1/n15.conf#
```

Figura 19-Conectividade entre RA-RD

```
Terminal - root@n18:/tmp/pycore.1/n18.conf
File Edit View Terminal Tabs Help
root@n18:/tmp/pycore.1/n18.conf# ping 10.6.24.2
PING 10.6.24.2 (10.6.24.2) 56(84) bytes of data.
64 bytes from 10.6.24.2: icmp_seq=1 ttl=60 time=0.879 ms
64 bytes from 10.6.24.2: icmp_seq=2 ttl=60 time=0.163 ms
```
--- 10.6.24.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1005ms
rtt min/avg/max/mdev = 0.163/0.521/0.879/0.358 ms
root@n18:/tmp/pycore.1/n18.conf# traceroute 10.6.24.2
traceroute to 10.6.24.2 (10.6.24.2), 30 hops max, 60 byte packets
1 10.6.16.1 (10.6.16.1) 4.179 ms 4.093 ms 4.079 ms
2 192.168.0.6 (192.168.0.6) 4.867 ms 3.942 ms 3.926 ms
3 192.168.0.9 (192.168.0.9) 3.911 ms 3.835 ms 3.817 ms
4 192.168.0.34 (192.168.0.34) 3.580 ms 3.412 ms 3.397 ms
5 10.6.24.2 (10.6.24.2) 3.382 ms 1.001 ms 0.000 ms
root@n18:/tmp/pycore.1/n18.conf#
```

Figura 20-Conectividade entre RB1-RC

```
Terminal - root@n18:/tmp/pycore.1/n18.conf
File Edit View Terminal Tabs Help
root@n18:/tmp/pycore.1/n18.conf# ping 10.6.28.3
PING 10.6.28.3 (10.6.28.3) 56(84) bytes of data.
64 bytes from 10.6.28.3: icmp_seq=1 ttl=63 time=14.8 ms
64 bytes from 10.6.28.3: icmp_seq=2 ttl=63 time=3.33 ms
```
--- 10.6.28.3 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 3.329/9.075/14.821/5.746 ms
root@n18:/tmp/pycore.1/n18.conf# traceroute 10.6.28.3
traceroute to 10.6.28.3 (10.6.28.3), 30 hops max, 60 byte packets
1 10.6.16.1 (10.6.16.1) 2.546 ms 1.244 ms 1.197 ms
2 10.6.28.3 (10.6.28.3) 0.438 ms 0.320 ms 0.306 ms
root@n18:/tmp/pycore.1/n18.conf#
```

Figura 21-Conectividade entre RB1-RB2

```
Terminal - root@n18:/tmp/pycore.1/n18.conf
File Edit View Terminal Tabs Help
root@n18:/tmp/pycore.1/n18.conf# ping 10.6.31.2
PING 10.6.31.2 (10.6.31.2) 56(84) bytes of data.
64 bytes from 10.6.31.2: icmp_seq=1 ttl=60 time=0.32 ms
64 bytes from 10.6.31.2: icmp_seq=2 ttl=60 time=0.289 ms
```
--- 10.6.31.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1009ms
rtt min/avg/max/mdev = 0.289/0.802/1.315/0.513 ms
root@n18:/tmp/pycore.1/n18.conf# traceroute 10.6.31.2
traceroute to 10.6.31.2 (10.6.31.2), 30 hops max, 60 byte packets
1 10.6.16.1 (10.6.16.1) 2.008 ms 2.008 ms 2.006 ms
2 192.168.0.6 (192.168.0.6) 2.008 ms 2.008 ms 2.006 ms
3 192.168.0.18 (192.168.0.18) 2.008 ms 2.008 ms 2.006 ms
4 192.168.0.26 (192.168.0.26) 2.006 ms 2.006 ms 2.006 ms
5 10.6.31.2 (10.6.31.2) 2.006 ms 0.000 ms 0.000 ms
root@n18:/tmp/pycore.1/n18.conf#
```

Figura 22-Conectividade entre RB1-RD

```
Terminal - root@n18:/tmp/pycore.1/n18.conf
File Edit View Terminal Tabs Help
root@n18:/tmp/pycore.1/n18.conf# ping 10.6.30.3
PING 10.6.30.3 (10.6.30.3) 56(84) bytes of data.
64 bytes from 10.6.30.3: icmp_seq=1 ttl=60 time=3.95 ms
64 bytes from 10.6.30.3: icmp_seq=2 ttl=60 time=0.129 ms
64 bytes from 10.6.30.3: icmp_seq=3 ttl=60 time=0.093 ms
```
--- 10.6.30.3 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2073ms
rtt min/avg/max/mdev = 0.093/1.387/3.945/1.808 ms
root@n18:/tmp/pycore.1/n18.conf# traceroute 10.6.30.3
traceroute to 10.6.30.3 (10.6.30.3), 30 hops max, 60 byte packets
1 10.6.16.1 (10.6.16.1) 10.227 ms 10.239 ms 10.241 ms
2 192.168.0.6 (192.168.0.6) 10.243 ms 10.249 ms 10.267 ms
3 192.168.0.14 (192.168.0.14) 10.270 ms 10.273 ms 10.275 ms
4 192.168.0.42 (192.168.0.42) 10.277 ms 3.110 ms 2.951 ms
5 10.6.30.3 (10.6.30.3) 2.935 ms 0.668 ms 0.637 ms
root@n18:/tmp/pycore.1/n18.conf#
```

Figura 23-Conectividade entre RB1-RE

```
Terminal - root@n26:/tmp/pycore.1/n26.conf
File Edit View Terminal Tabs Help
root@n26:/tmp/pycore.1/n26.conf# ping 10.6.31.2
PING 10.6.31.2 (10.6.31.2) 56(84) bytes of data.
64 bytes from 10.6.31.2: icmp_seq=1 ttl=60 time=0.000 ms
64 bytes from 10.6.31.2: icmp_seq=2 ttl=60 time=0.000 ms
```
--- 10.6.31.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1068ms
rtt min/avg/max/mdev = 0.000/0.000/0.000/0.000 ms
root@n26:/tmp/pycore.1/n26.conf# traceroute 10.6.31.2
traceroute to 10.6.31.2 (10.6.31.2), 30 hops max, 60 byte packets
1 10.6.28.1 (10.6.28.1) 0.000 ms 0.000 ms 0.000 ms
2 192.168.0.6 (192.168.0.6) 0.000 ms 0.000 ms 0.000 ms
3 192.168.0.18 (192.168.0.18) 0.000 ms 0.000 ms 0.000 ms
4 192.168.0.26 (192.168.0.26) 0.000 ms 2.000 ms 2.000 ms
5 10.6.31.2 (10.6.31.2) 2.000 ms 2.000 ms 2.000 ms
root@n26:/tmp/pycore.1/n26.conf#
```

Figura 25-Conectividade entre RB2-RD

```
Terminal - root@n26:/tmp/pycore.1/n26.conf
File Edit View Terminal Tabs Help
root@n26:/tmp/pycore.1/n26.conf# ping 10.6.30.3
PING 10.6.30.3 (10.6.30.3) 56(84) bytes of data.
64 bytes from 10.6.30.3: icmp_seq=1 ttl=60 time=0.594 ms
64 bytes from 10.6.30.3: icmp_seq=2 ttl=60 time=0.095 ms
```
--- 10.6.30.3 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1038ms
rtt min/avg/max/mdev = 0.095/0.344/0.594/0.249 ms
root@n26:/tmp/pycore.1/n26.conf# traceroute 10.6.30.3
traceroute to 10.6.30.3 (10.6.30.3), 30 hops max, 60 byte packets
1 10.6.28.1 (10.6.28.1) 1.021 ms 0.012 ms 0.004 ms
2 192.168.0.6 (192.168.0.6) 0.015 ms 0.005 ms 0.005 ms
3 192.168.0.14 (192.168.0.14) 0.017 ms 0.006 ms 0.007 ms
4 192.168.0.42 (192.168.0.42) 0.123 ms 0.010 ms 0.008 ms
5 10.6.30.3 (10.6.30.3) 0.051 ms 0.009 ms 0.010 ms
root@n26:/tmp/pycore.1/n26.conf#
```

Figura 26-Conectividade entre RB2-RE



```
Terminal - root@n19:/tmp/pycore.1/n19.conf
File Edit View Terminal Tabs Help
root@n19:/tmp/pycore.1/n19.conf# ping 10.6.31.2
PING 10.6.31.2 (10.6.31.2) 56(84) bytes of data.
64 bytes from 10.6.31.2: icmp_seq=1 ttl=61 time=5.41 ms
64 bytes from 10.6.31.2: icmp_seq=2 ttl=61 time=2.00 ms
```
--- 10.6.31.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 2.000/3.704/5.408/1.704 ms
root@n19:/tmp/pycore.1/n19.conf# traceroute 10.6.31.2
traceroute to 10.6.31.2 (10.6.31.2), 30 hops max, 60 byte packets
1 10.6.24.1 (10.6.24.1) 4.153 ms 0.584 ms 0.555 ms
2 192.168.0.46 (192.168.0.46) 0.546 ms 0.510 ms 0.501 ms
3 192.168.0.49 (192.168.0.49) 0.493 ms 0.312 ms 0.142 ms
4 10.6.31.2 (10.6.31.2) 0.130 ms 0.107 ms 0.096 ms
root@n19:/tmp/pycore.1/n19.conf#
```

Figura 27-Conectividade entre RC-RD

```
Terminal - root@n19:/tmp/pycore.1/n19.conf
File Edit View Terminal Tabs Help
root@n19:/tmp/pycore.1/n19.conf# ping 10.6.30.2
PING 10.6.30.2 (10.6.30.2) 56(84) bytes of data.
64 bytes from 10.6.30.2: icmp_seq=1 ttl=62 time=0.263 ms
64 bytes from 10.6.30.2: icmp_seq=2 ttl=62 time=0.091 ms
```
--- 10.6.30.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1040ms
rtt min/avg/max/mdev = 0.091/0.177/0.263/0.086 ms
root@n19:/tmp/pycore.1/n19.conf# traceroute 10.6.30.2
traceroute to 10.6.30.2 (10.6.30.2), 30 hops max, 60 byte packets
1 10.6.24.1 (10.6.24.1) 0.693 ms 0.658 ms 0.623 ms
2 192.168.0.46 (192.168.0.46) 0.614 ms 0.595 ms 0.587 ms
3 10.6.30.2 (10.6.30.2) 0.578 ms 0.241 ms 0.222 ms
root@n19:/tmp/pycore.1/n19.conf#
```

Figura 28-Conectividade entre RC-RE

```
Terminal - root@n23:/tmp/pycore.1/n23.conf
File Edit View Terminal Tabs Help
root@n23:/tmp/pycore.1/n23.conf# ping 10.6.30.3
PING 10.6.30.3 (10.6.30.3) 56(84) bytes of data.
64 bytes from 10.6.30.3: icmp_seq=1 ttl=62 time=1.22 ms
64 bytes from 10.6.30.3: icmp_seq=2 ttl=62 time=0.078 ms
64 bytes from 10.6.30.3: icmp_seq=3 ttl=62 time=0.081 ms
```
--- 10.6.30.3 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2009ms
rtt min/avg/max/mdev = 0.078/0.459/1.220/0.537 ms
root@n23:/tmp/pycore.1/n23.conf# traceroute 10.6.30.3
traceroute to 10.6.30.3 (10.6.30.3), 30 hops max, 60 byte packets
1 10.6.31.1 (10.6.31.1) 3.421 ms 3.416 ms 3.407 ms
2 192.168.0.50 (192.168.0.50) 3.398 ms 2.893 ms 2.668 ms
3 10.6.30.3 (10.6.30.3) 2.640 ms 2.597 ms 2.580 ms
root@n23:/tmp/pycore.1/n23.conf#
```

Figura 29-Conectividade entre RD-RE

Adição de mais dispositivos na LAN E utilizando a rede 192.168.100.0/26

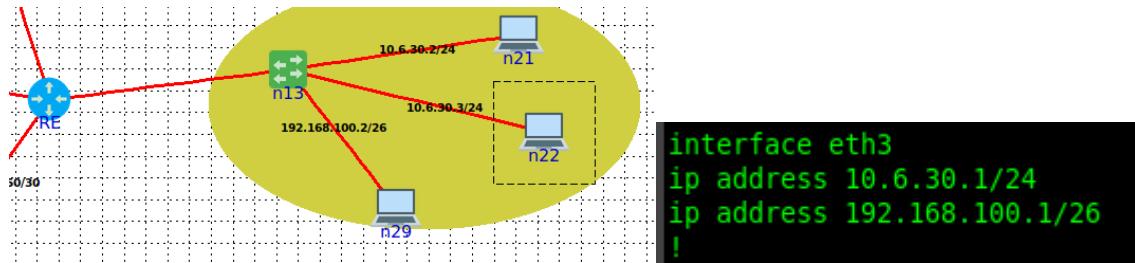


Figura 30-Nova rede na LanE

Para permitir que uma interface de rede pertença simultaneamente a duas redes IP distintas na mesma LAN física, utilizou-se a configuração do Zebra, que permite a atribuição de múltiplos endereços IP à mesma interface. Para que haja comunicação entre dispositivos de redes diferentes na mesma LAN, é necessário um router com os endereços configurados em ambas as redes. Antes de aplicar a configuração no Zebra, foi necessário remover o IP previamente atribuído pelo sistema, para evitar conflitos ou sobreposição, garantindo que a nova configuração seja corretamente aplicada.

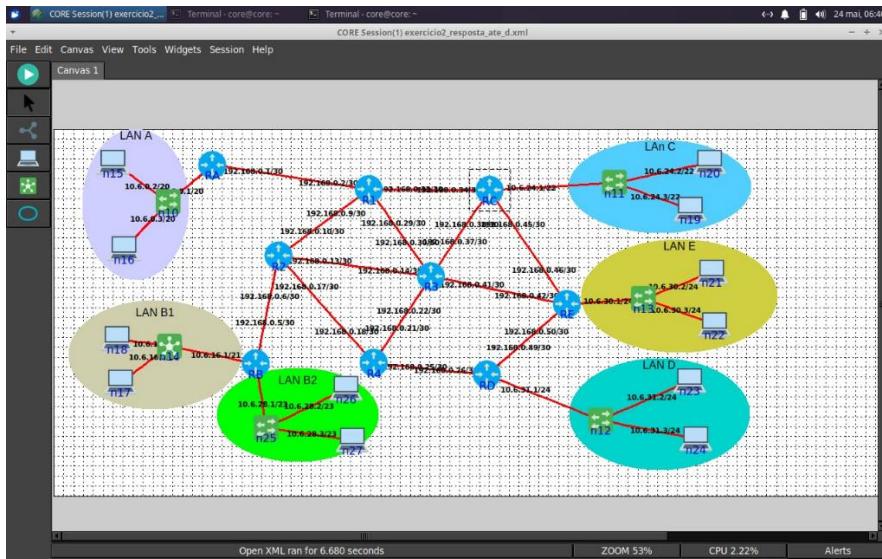


Figura 31-Topologia final 2

3-DHCP–Dynamic Host Configuration Protocol

O DHCP é um protocolo que atribui dinamicamente um IP a cada cliente da mesma rede. Este protocolo facilita a ligação de um dispositivo a uma nova rede, dado que, sendo um dispositivo novo, não é necessário configurar um IP estático, aumentando assim a simplicidade da utilização.

Configuração:

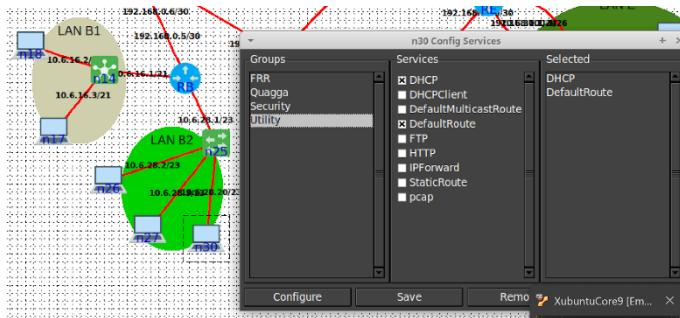


Figura 32-Ativação DHCP

Em primeiro lugar, foi feita a instalação a partir do comando “`sudo apt-get install isc-dhcp-server`” e de seguida a configuração dos sistemas nos terminais.

A máquina n30 foi configurada como servidor DHCP.

As restantes máquinas foram definidas como clientes DHCP, recebendo os IPs dinamicamente. Também foi feita a configuração de range para o endereço de gama do B1 definido no exercício anterior.



```
# auto-generated by DHCP service (utility.py)
# NOTE: move these option lines into the desired pool { } block(s) below
#option domain-name "test.com";
#option domain-name-servers 10.0.0.1;
#option routers 10.0.0.1;

log-facility local6;

default-lease-time 600;
max-lease-time 7200;

ddns-update-style none;

subnet 10.6.28.0 netmask 255.255.254.0 {
    pool {
        range 10.6.28.1 10.6.29.254;
        default-lease-time 600;
        option routers 10.6.28.20;
    }
}
```

Figura 33-Configuração DHCP

Processo de Atribuição de IP via DHCP

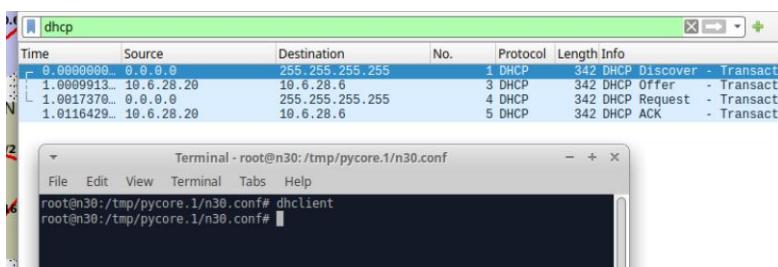


Figura 34-Leitura DHCP no Wireshark

Utilizando o filtro “dhcp” para uma melhor visualização, podemos observar as quatro fases de dhcp, Discover - Offer - request - Ack.

O processo de atribuição dinâmica de endereços IP através do protocolo DHCP inicia-se quando um cliente envia uma mensagem DHCP Discover em broadcast para a rede local. Nesta fase inicial, como o cliente ainda não possui um endereço IP configurado, o campo de endereço de origem aparece como 0.0.0.0. Esta mensagem é recebida pelo servidor DHCP configurado na rede.

Na fase seguinte, o servidor DHCP responde com uma mensagem DHCP Offer, também enviada em broadcast. Esta mensagem contém a proposta de configuração de rede para o cliente (10.6.28.6). Ao receber a proposta, o cliente DHCP envia então uma mensagem DHCP Request, também em broadcast, indicando ao servidor que aceita os parâmetros propostos e solicita formalmente a atribuição desse endereço IP. Esta etapa permite que outros servidores DHCP eventualmente presentes na rede tomem conhecimento da seleção feita pelo cliente.

Finalmente, o servidor DHCP responde com uma mensagem DHCP ACK (acknowledgement), confirmando a atribuição do endereço IP e dos demais parâmetros ao cliente. Só após a receção desta mensagem final é que o cliente passa a utilizar a configuração de rede atribuída, completando assim o processo de leasing DHCP. Todo este processo ocorre automaticamente sem necessidade de intervenção manual, demonstrando a eficiência do protocolo DHCP na gestão dinâmica de endereços numa rede.

4-Uso das camadas de rede e transporte por parte das aplicações

Um servidor HTTP (HyperText Transfer Protocol) é um serviço de rede que permite disponibilizar conteúdos web (como páginas HTML, imagens ou ficheiros) através do protocolo HTTP. Quando é instalado num computador (ou nó) dentro de uma rede simulada no CORE, esse servidor



permite que os restantes nós da mesma rede acedam a esses conteúdos utilizando um navegador web ou comandos como curl ou wget.

Ou seja, instalar um servidor HTTP consiste em configurar um dos PCs virtuais da rede (isto é, um nó) para funcionar como servidor web. Esse nó irá então responder aos pedidos enviados por outros nós que pretendam aceder ao conteúdo web que ele disponibiliza.

Configuração:

Em primeiro lugar, foi feita a instalação a partir do comando “ sudo apt install apache2 ” e o “ sudo apt install links2 ”

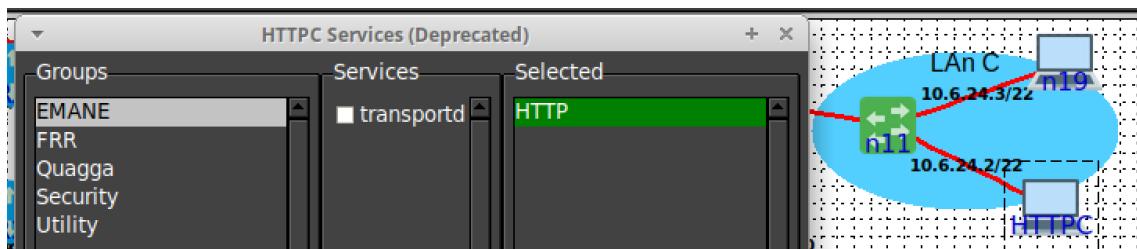


Figura 35-Ativação http

Adotamos um dos PCs como servidor HTTP, foi renomeado para HTTPC para uma melhor visualização. Para testar a conectividade usar um pc da rede D, usando comando wget 10.6.24.2.

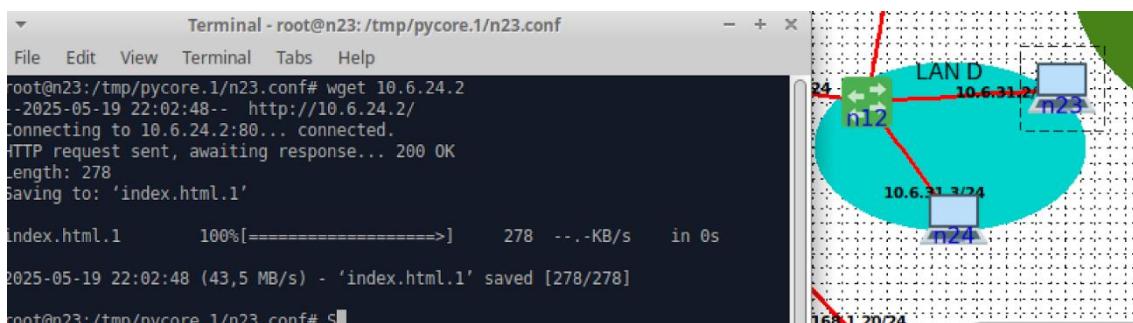


Figura 36-Testar conectividade http

10.6.31.2	10.6.24.2	13 TCP	74 49926 → 80 [SYN] Seq=0 Win=64240 Len: 60
10.6.24.2	10.6.31.2	14 TCP	74 80 → 49926 [SYN, ACK] Seq=0 Ack=1 Win: 64240
10.6.31.2	10.6.24.2	15 TCP	66 49926 → 80 [ACK] Seq=1 Ack=1 Win=64240
10.6.31.2	10.6.24.2	16 HTTP	190 GET / HTTP/1.1
10.6.24.2	10.6.31.2	17 TCP	66 80 → 49926 [ACK] Seq=1 Ack=125 Win: 64240
10.6.24.2	10.6.31.2	18 HTTP	604 HTTP/1.1 200 OK
10.6.31.2	10.6.24.2	19 TCP	66 49926 → 80 [ACK] Seq=125 Ack=539 Win: 64240
10.6.31.2	10.6.24.2	20 TCP	66 49926 → 80 [FIN, ACK] Seq=125 Ack=539 Win: 64240
10.6.24.2	10.6.31.2	21 TCP	66 80 → 49926 [FIN, ACK] Seq=539 Ack=126 Win: 64240
10.6.31.2	10.6.24.2	22 TCP	66 49926 → 80 [ACK] Seq=126 Ack=540 Win: 64240
00:00:00_aa:00:19	00:00:00_aa:00:16	23 ARP	42 Who has 10.6.24.1? Tell 10.6.24.2
00:00:00_aa:00:16	00:00:00_aa:00:19	24 ARP	42 Who has 10.6.24.2? Tell 10.6.24.1
00:00:00_aa:00:19	00:00:00_aa:00:16	25 ARP	42 10.6.24.2 is at 00:00:00:aa:00:19
00:00:00_aa:00:16	00:00:00_aa:00:19	26 ARP	42 10.6.24.1 is at 00:00:00:aa:00:16

Figura 37-Wireshark http

Na captura feita por wireshark, podemos verificar uma comunicação HTTP entre um cliente (10.6.31.2) e um servidor (10.6.24.2). Primeiro, ocorre o three-way handshake para estabelecer a conexão TCP: o cliente envia um SYN, o servidor responde com SYN-ACK e o cliente confirma com o ACK. Após isso, o cliente faz uma requisição HTTP GET(wget) no pacote 16. O servidor responde com HTTP/1.1 200 OK no pacote 18, indicando assim sucesso no carregamento da página. Finalmente, a conexão é encerrada com a troca de pacotes FIN e ACK entre o cliente e o servidor.



Foi criado um formulário simples em HTML contendo dois campos: username e password. E o conteúdo do mesmo é:

```

<html><body>
<h1>Formulário de login</h1>
<form method="POST">
<label>Username:</label> <input type="text" name="username" /><br/><br/>
<label>Password:</label> <input type="password" name="password" /><br/><br/>
<input type="submit" value="Entrar" />
</form>
</body></html>

```

Figura 38-Formulário de login HTML

Usamos o comando “links2” e acedemos à página Web. Preenchemos o URL e seguidamente somos direcionados para o formulário de login, onde preenchemos os campos de username e da password e submetemos o formulário. Esta ação cria um pedido HTTP POST, visível na captura de tráfego realizada com o Wireshark.

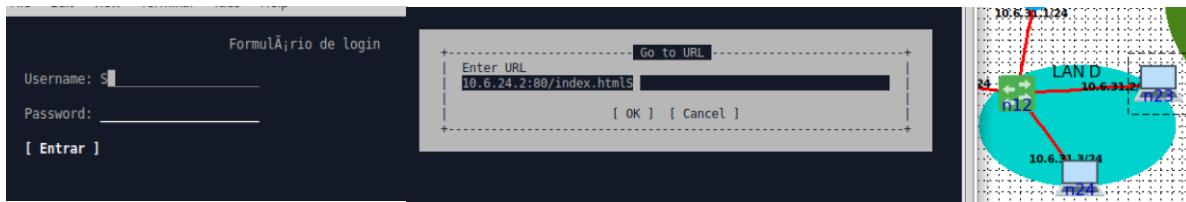


Figura 39-Links2 execução de formulário

A imagem abaixo mostra os pacotes capturados durante o processo de envio do formulário. É possível identificar:

- Estabelecimento da ligação TCP através do handshake em três fases: SYN, SYN-ACK e ACK;
- Submissão do formulário através de um pedido HTTP POST com os dados username e password;
- Resposta do servidor com um HTTP 200 OK, indicando o sucesso na receção e processamento dos dados;
- Encerramento da ligação TCP com a troca de pacotes FIN e ACK entre o cliente e o servidor.

Time	Source	Destination	No.	Protocol	Length	Info
0.000000000	10.6.24.2	10.6.31.2	1	TCP	74	37506 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 TStamp=2320294439 TSectr=0 WS=128
0.000015800	10.6.24.2	10.6.31.2	2	TCP	74 80 → 37506 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK_PERM=1 TStamp=2333591145 TSectr=23...	
0.000055400	10.6.31.2	10.6.24.2	3	TCP	66 37506 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TStamp=2320294439 TSectr=233591145	
0.000118600	10.6.31.2	10.6.24.2	4	HTTP	819 POST /index.html HTTP/1.1 (application/x-www-form-urlencoded)	
0.000132300	10.6.24.2	10.6.31.2	5	TCP	66 80 → 37506 [ACK] Seq=1 Ack=754 Win=64512 Len=0 TStamp=233591145 TSectr=2320294439	
0.000504900	10.6.24.2	10.6.31.2	6	HTTP	684 HTTP/1.1 200 OK	
0.000510000	10.6.24.2	10.6.31.2	7	TCP	66 37506 → 80 [ACK] Seq=754 Ack=530 Win=63744 Len=0 TStamp=233591145 TSectr=233591145	
5.003023900	10.6.24.2	10.6.31.2	8	TCP	66 80 → 37506 [FIN, ACK] Seq=530 Ack=754 Win=64512 Len=0 TStamp=2333591148 TSectr=2320294439 TSectr=2333591148	
5.044705300	10.6.31.2	10.6.24.2	9	TCP	66 37506 → 80 [ACK] Seq=754 Ack=540 Win=63744 Len=0 TStamp=2320294483 TSectr=2333591148	
28.020472000	10.6.31.2	10.6.24.2	10	TCP	66 37506 → 80 [FIN, ACK] Seq=754 Ack=548 Win=63744 Len=0 TStamp=2320314459 TSectr=2333596148	
28.020488000	10.6.24.2	10.6.31.2	11	TCP	66 80 → 37506 [ACK] Seq=540 Ack=755 Win=64512 Len=0 TStamp=2333611165 TSectr=2320314459	

Figura 40-Captura HTTP POST do formulário de login



Para demonstrar o funcionamento de uma comunicação HTTP através do protocolo TCP, foi realizada uma captura de pacotes com o Wireshark. A análise mostra todas as fases essenciais da ligação entre cliente e servidor, desde o estabelecimento da sessão TCP até à troca de dados e ao encerramento da ligação.

O processo inicia-se quando o cliente, com o endereço IP 10.6.31.2, envia uma mensagem TCP com a flag SYN ativa para o servidor 10.6.24.2, utilizando a porta de origem 37566 e a porta de destino 80 (utilizada por defeito para HTTP). Esta mensagem indica a intenção de iniciar uma ligação. Em resposta, o servidor envia um segmento TCP com as flags SYN e ACK, confirmando a receção do pedido e aceitando estabelecer a ligação. Por fim, o cliente responde com um segmento ACK, completando assim o processo de three-way handshake que estabelece a sessão TCP entre as duas máquinas.

Imediatamente após a ligação estar estabelecida, o cliente envia um pedido HTTP do tipo GET, solicitando o ficheiro "/index.html". O servidor responde com uma mensagem HTTP contendo o código 200 OK, indicando que o pedido foi bem-sucedido, e envia o conteúdo da página solicitada. Durante esta fase, são trocadas várias mensagens de confirmação (ACK), assegurando que os dados são corretamente recebidos por ambas as partes.

Uma vez concluída a transferência dos dados, o cliente inicia o processo de encerramento da ligação TCP ao enviar uma mensagem com a flag FIN, sinalizando que não irá enviar mais dados. O servidor responde com uma mensagem FIN, ACK, aceitando terminar a comunicação. Por fim, o cliente envia um último ACK, concludo assim a sessão.

Todo este processo decorre de forma automática, garantindo assim uma comunicação fiável e estruturada entre o cliente e o servidor. A análise desta captura evidencia claramente o funcionamento conjunto dos protocolos TCP e HTTP na troca de informação na Internet.

5-Interligação via Network Address Translation

O Network Address Translation (NAT) é uma técnica utilizada em redes de computadores que permite a tradução de endereços IP privados (internos) para um endereço IP público (externo).

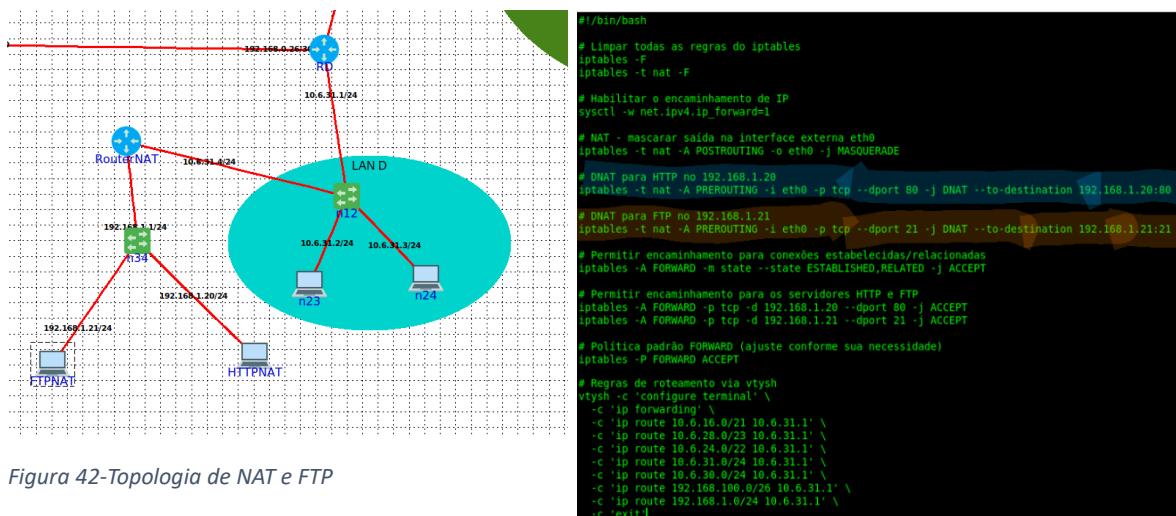


Figura 42-Topologia de NAT e FTP

Redireciona tráfego externo para o servidor HTTP interno

Redireciona tráfego externo para o servidor FTP interno (vsftpd)

Figura 41-Configuração Firewall da NAT

Como se pode observar na imagem incluída anteriormente, a topologia da rede foi expandida com a criação de uma rede local adicional usando a gama de endereços 192.168.1.0/24. Esta nova rede está ligada ao Router



A instalação do FTP foi feita com o comando “sudo apt install vsftpd” de seguida, a configuração do NAT (Network Address Translation) foi efetuada diretamente no router RouterNAT, utilizando iptables responsável pela criação de regras de firewall. O RouterNAT, seu ip público é 10.6.31.4, possui duas interfaces: uma ligada à rede privada com o endereço IP 192.168.1.1/24 e outra ligada às redes públicas com o endereço IP 10.6.31.4/24. Esta estrutura permite que o router funcione como intermediário entre as duas redes, aplicando as devidas regras de tradução de endereços e encaminhamento de tráfego.

Para corrigir o erro: "500 OOPS: vsftpd: refusing to run with writable root inside chroot()" usamos “chmod 755 var/run/vsftpd/empty” e “chmod 755 var/ftp” estes ,definem permissões de acesso que permitem ao servidor FTP funcionar corretamente, evitando erros de permissão durante conexões.

Relatório de Análise Técnica- Conexão FTP, HTTP e NAT

0.00090000	10.6.28.20	192.168.1.21	1 TCP	66 39732 - 21 [FIN, ACK] Seq=1 Ack=1 Win=0 TStamp=2602710429 TSectr=1360387381
0.00058240	192.168.1.21	10.6.28.20	2 TCP	66 21 - 39732 [ACK] Seq=1 Ack=2 Win=510 Len=0 TStamp=1360395454 TSectr=2602710429
0.00088816	192.168.1.21	10.6.28.20	3 TCP	66 22 - 39732 [FIN, ACK] Seq=1 Ack=2 Win=510 Len=0 TStamp=1360395455 TSectr=2602710429
0.00058240	192.168.1.21	10.6.28.20	4 TCP	66 39732 - 21 [ACK] Seq=1 Ack=38 Win=65536 Len=0 TStamp=2602710433 TSectr=1360399088
0.00058061	10.6.28.20	192.168.1.21	5 TCP	74 36552 - 21 [ACK] Seq=1 Ack=38 Win=65536 Len=0 MSS=1460 SACK PERM=1 TStamp=2602710409 TSectr=0 WS=4
3.6315181	192.168.1.21	10.6.28.20	6 TCP	74 21 - 36552 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SACK PERM=1 TStamp=1360399885 TSectr=26
3.6316233	10.6.28.20	192.168.1.21	7 TCP	66 36552 - 21 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TStamp=2602714860 TSectr=1360399085
3.6345093	192.168.1.21	10.6.28.20	8 FTP	103 Response: 220 Welcome to the CORE FTP service
3.6346272	10.6.28.20	192.168.1.21	9 TCP	66 36552 - 21 [ACK] Seq=1 Ack=38 Win=65536 Len=0 TStamp=2602714063 TSectr=1360399088
11.434234	10.6.28.20	192.168.1.21	10 FTP	77 Request: USER core
11.434314	192.168.1.21	10.6.28.20	11 TCP	66 21 - 36552 [ACK] Seq=38 Ack=12 Win=65280 Len=0 TStamp=1360406688 TSectr=2602721863
11.434545	192.168.1.21	10.6.28.20	12 FTP	180 Response: 331 Please specify the password
11.434549	10.6.28.20	192.168.1.21	13 TCP	66 36552 - 21 [ACK] Seq=12 Ack=1 Win=65536 Len=0 TStamp=2602721864 TSectr=1360406688
13.193502	10.6.28.20	192.168.1.21	14 FTP	77 Request: PASS core
13.233917	192.168.1.21	10.6.28.20	15 TCP	66 21 - 36552 [ACK] Seq=72 Ack=23 Win=65280 Len=0 TStamp=1360408688 TSectr=2602723622
13.398778	192.168.1.21	10.6.28.20	16 FTP	89 Response: 230 Login successful.
13.398925	10.6.28.20	192.168.1.21	17 TCP	66 36552 - 21 [ACK] Seq=23 Ack=95 Win=65536 Len=0 TStamp=2602723730 TSectr=1360408755
13.301186	10.6.28.20	192.168.1.21	18 TCP	72 Request: SYST
13.301193	192.168.1.21	10.6.28.20	19 TCP	66 21 - 36552 [ACK] Seq=95 Ack=29 Win=65280 Len=0 TStamp=1360408755 TSectr=2602723730
13.301314	192.168.1.21	10.6.28.20	20 TCP	85 Response: 213 UNIX Type: L8
13.301421	192.168.1.21	10.6.28.20	21 TCP	72 Request: FEAT
13.301444	192.168.1.21	10.6.28.20	22 TCP	91 Response: 211 Features:
13.301444	192.168.1.21	10.6.28.20	23 TCP	73 Response: EPRT
13.301484	192.168.1.21	10.6.28.20	24 FTP	73 Response: EPSV
13.301484	192.168.1.21	10.6.28.20	25 TCP	73 Response: MDTM
13.301503	192.168.1.21	10.6.28.20	26 FTP	73 Response: PASV
13.301522	192.168.1.21	10.6.28.20	27 TCP	88 Response: REST STREAM
13.301541	192.168.1.21	10.6.28.20	28 FTP	73 Response: SIZE
13.301559	192.168.1.21	10.6.28.20	29 FTP	73 Response: Lvfs
13.301779	192.168.1.21	10.6.28.20	30 TCP	75 Response: 221 End
13.301623	10.6.28.20	192.168.1.21	31 TCP	66 36552 - 21 [ACK] Seq=35 Ack=194 Win=65536 Len=0 TStamp=2602723730 TSectr=1360406755

Figura 43-Conexão FTP,HTTP,NAT

Foi realizada com sucesso uma conexão FTP entre o cliente 10.0.26.20 (rede interna) e o servidor 10.6.31.4 (rede NAT). A análise da Wireshark revelou detalhes importantes sobre a comunicação. O cliente na rede interna (10.0.26.20) estabeleceu comunicação com o servidor FTP após NAT (192.168.1.21), demonstrando o correto funcionamento da tradução de endereços de rede. A captura mostrou claramente o handshake TCP inicial (SYN, SYN-ACK, ACK) na porta 21 (controlo FTP), seguido pela troca de mensagens de autenticação. O tráfego subsequente incluiu comandos adicionais como SYST (para identificação do sistema) e negociação de recursos (FEAT, EPSV), com respostas adequadas do servidor.

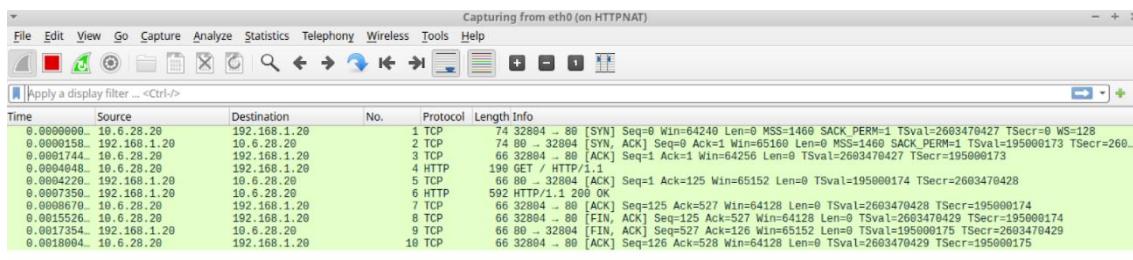


Figura 44-Fluxo de comunicação wireshark NAT

A análise simultânea no Wireshark capturou todo o fluxo de comunicação, A captura demonstra claramente o funcionamento adequado do NAT, com a tradução correta de endereços entre as redes interna (10.6.28.20) e externa (192.168.1.20), permitindo a comunicação bidirecional sem erros.