

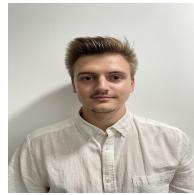
Universidade do Minho
Escola de Engenharia

Redes Fixas e Móveis

2023/2024

Trabalho Prático 3

André Alves :: pg53651; Renato Gomes :: pg54174; Afonso Marques :: pg53601



Abstract

A rápida evolução das redes móveis tem impulsionado a sociedade para uma era de conectividade sem precedentes. Com a implementação do 5G, a próxima geração de comunicações sem fio, espera-se uma revolução na forma como interagimos com o mundo digital. Esta nova tecnologia promete velocidades de transmissão de dados ultra rápidas, latência mínima e uma capacidade impressionante de conexão de dispositivos em massa, transformando setores inteiros, desde a saúde até a indústria automóvel.

Além disso, o 5G abre caminho para inovações como a Internet of Things (IoT), realidade aumentada e virtual, e veículos autónomos, inaugurando uma era de experiências digitais imersivas. No entanto, desafios relacionados à segurança cibernética, infraestrutura e inclusão digital precisam ser enfrentados para garantir que todos possam se beneficiar plenamente dessa nova era de conectividade móvel.

Index Terms

Core 5G, Redes Móveis.

I. INTRODUÇÃO

O seguinte trabalho pretende explorar o uso de um CORE 5G de código aberto. O free5GC é um projeto de código aberto que visa criar o núcleo de uma rede móvel de quinta geração (5G). O objetivo principal é implementar a rede central 5G (5GC) conforme definido pelo 3GPP Release 15 (R15) e além.

O projeto também tem como foco a utilização de um simulador tanto do User Equipment (UE) quanto da Radio Access Network (RAN), o UERANSIM, que simula dois elementos fundamentais de uma rede móvel: o Equipamento do Utilizador (UE) e a Rede de Acesso por Rádio (RAN), que inclui uma estação base 5G conhecida como gNodeB.

O seguinte esquema mostra uma configuração básica com duas máquinas virtuais, VM's, onde a VM1 representa tanto o UE quanto a RAN; e uma VM2 representa o Data Plane e o Control Plane de uma rede 5G incluindo as Network Functions (NFs) envolvidas.

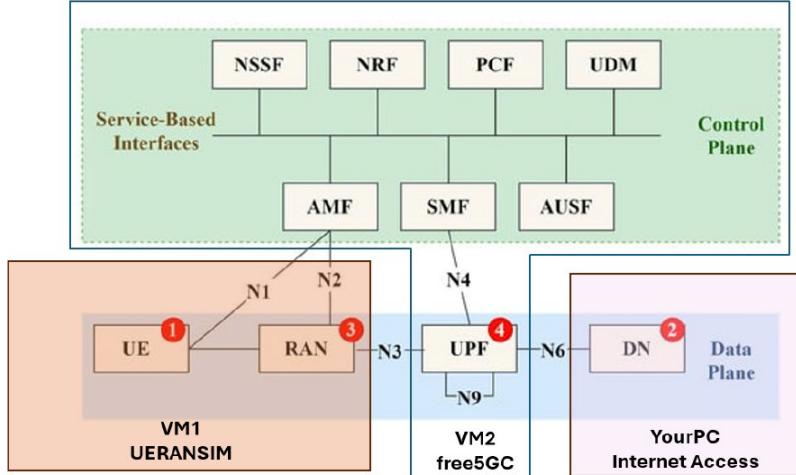


Fig. 1: Esquema da configuração utilizada

II. DESENVOLVIMENTO

A. Configurações Realizadas

1) Descrição das VM's:

De forma a realizar o trabalho foi necessário fazer a instalação e configuração de duas máquinas virtuais, denominadas *free5gc* e *UERANSIM*.

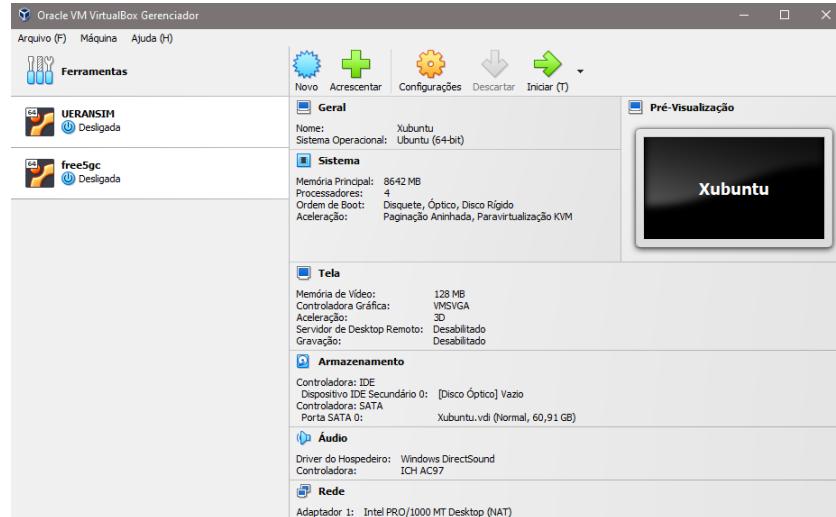


Fig. 2: VM's usadas

A primeira máquina virtual serve para alojar o CORE 5G, através da ferramenta *free5gc* disponível em código aberto no GitHub. Esta máquina tem um endereço IP estático, 192.168.56.101, e permite a criação de um núcleo de uma rede móvel de 5^a geração. Nesta máquina conseguimos aceder ao serviço WebUI do *free5gc*, onde podemos criar *subscribers*, que no contexto do Free5GC, refere-se a um assinante, ou seja, um utilizador que está registado na rede e possui uma identidade única dentro do sistema. Para o propósito deste trabalho criamos apenas um *subscriber*. Nesta máquina também foi necessário ativar o *routing* executando a seguinte sequência de comandos:

```
sudo sysctl -w net.ipv4.ip_forward=1
sudo iptables -t nat -A POSTROUTING -o enp0s3 -j MASQUERADE
```

```
sudo systemctl stop ufw
sudo iptables -I FORWARD 1 -j ACCEPT
```

Estes comandos são executados de forma automática através de um script bash criado pelo grupo para esse efeito, o routing.sh, que foi entregue em conjunto com este relatório.

A segunda máquina virtual aloja a ferramenta UERANSIM, que simula dois elementos de uma rede móvel: o Equipamento do Utilizador (UE) e a Rede de Acesso por Rádio (RAN). Esta máquina também tem um endereço IP estático, sendo este 192.168.56.102. Esta máquina executa dois processos, sendo eles o executável *nr-ue* e o *nr-gnb*, cada um com o seu ficheiro de configuração, onde indicamos quais os parâmetros necessários para estabelecer a comunicação entre as duas máquinas, sendo alguns deles o endereço da máquina free5gc e os parâmetros associados ao *subscriber* criado no serviço WebUI.

Concluindo o processo de configuração, partimos para a execução dos programas necessários para estabelecer a comunicação entre as máquinas, onde durante o processo é criado um túnel de comunicação denominado *uesimtun0*.

```
Ficheiro Editar Ver Terminal Separadores Ajuda
Terminal - ueransim@ueransim:~
inet 192.168.56.102 netmask 255.255.255.0 broadcast 192.168.56.255
inet6 fe80::a00:27ff:fedf:d5e7 prefixlen 64 scopeid 0x20<link>
ether 08:00:27:fd:d5:e7 txqueuelen 1000 (Ethernet)
RX packets 812 bytes 81140 (81.1 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 714 bytes 64799 (64.7 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
inet 127.0.0.1 netmask 255.0.0.0
inet6 ::1 prefixlen 128 scopeid 0x10<host>
loop txqueuelen 1000 (Local Loopback)
RX packets 6258 bytes 361879 (361.8 KB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 6258 bytes 361879 (361.8 KB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

uesimtun0: flags=369<UP,POINTOPOINT,NOTRAILERS,RUNNING,PROMISC> mtu 1400
inet 10.60.0.1 netmask 255.255.255.255 destination 10.60.0.1
inet6 fe80::76ac:901e:b5af:fe36 prefixlen 64 scopeid 0x20<link>
unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00 txqueuelen 500 (UNSPEC)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 7 bytes 448 (448.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

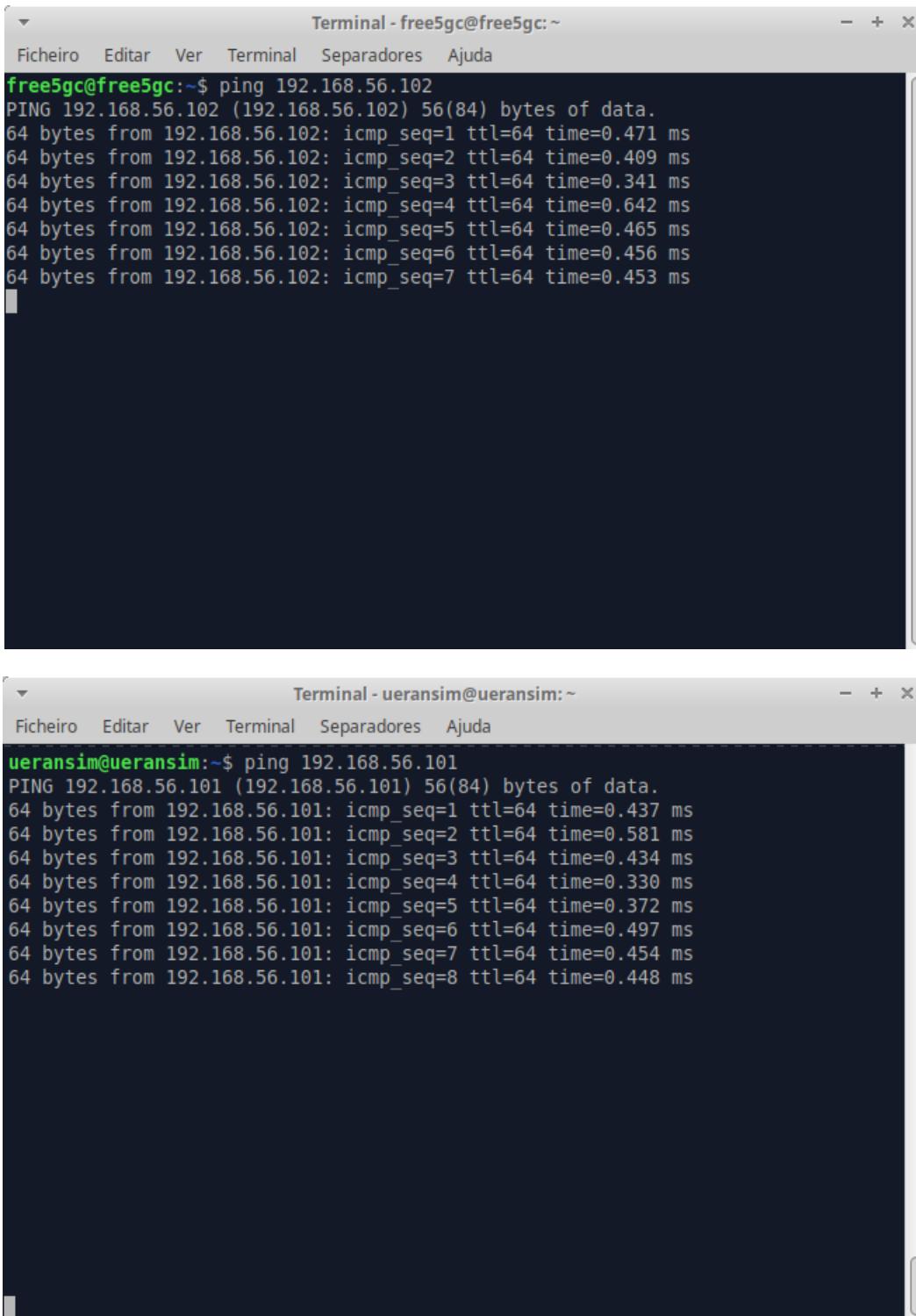
ueransim@ueransim:~$
```

Fig. 3: Novo túnel criado

B. Testes Realizados E Discussão

1) Testes de Ping:

O primeiro teste foi efetuado para verificar a conectividade entre as máquinas (free5gc e UERANSIM). Já o segundo teste serviu para verificar se a interface encontra-se em funcionamento.



The image displays two terminal windows side-by-side. Both windows have a dark background and light-colored text. The top window is titled "Terminal - free5gc@free5gc:~" and shows the command "ping 192.168.56.102" followed by its output. The bottom window is titled "Terminal - ueransim@ueransim:~" and shows the command "ping 192.168.56.101" followed by its output. Both outputs show multiple ICMP packets being sent to the specified IP address, with details like sequence number, TTL, and time taken.

```

Terminal - free5gc@free5gc:~
Ficheiro Editar Ver Terminal Separadores Ajuda
free5gc@free5gc:~$ ping 192.168.56.102
PING 192.168.56.102 (192.168.56.102) 56(84) bytes of data.
64 bytes from 192.168.56.102: icmp_seq=1 ttl=64 time=0.471 ms
64 bytes from 192.168.56.102: icmp_seq=2 ttl=64 time=0.409 ms
64 bytes from 192.168.56.102: icmp_seq=3 ttl=64 time=0.341 ms
64 bytes from 192.168.56.102: icmp_seq=4 ttl=64 time=0.642 ms
64 bytes from 192.168.56.102: icmp_seq=5 ttl=64 time=0.465 ms
64 bytes from 192.168.56.102: icmp_seq=6 ttl=64 time=0.456 ms
64 bytes from 192.168.56.102: icmp_seq=7 ttl=64 time=0.453 ms

Terminal - ueransim@ueransim:~
Ficheiro Editar Ver Terminal Separadores Ajuda
ueransim@ueransim:~$ ping 192.168.56.101
PING 192.168.56.101 (192.168.56.101) 56(84) bytes of data.
64 bytes from 192.168.56.101: icmp_seq=1 ttl=64 time=0.437 ms
64 bytes from 192.168.56.101: icmp_seq=2 ttl=64 time=0.581 ms
64 bytes from 192.168.56.101: icmp_seq=3 ttl=64 time=0.434 ms
64 bytes from 192.168.56.101: icmp_seq=4 ttl=64 time=0.330 ms
64 bytes from 192.168.56.101: icmp_seq=5 ttl=64 time=0.372 ms
64 bytes from 192.168.56.101: icmp_seq=6 ttl=64 time=0.497 ms
64 bytes from 192.168.56.101: icmp_seq=7 ttl=64 time=0.454 ms
64 bytes from 192.168.56.101: icmp_seq=8 ttl=64 time=0.448 ms

```

Fig. 4: Ping entre as duas VM's

A seguir testamos a nova interface *uesimtun0*, onde tentamos aceder à página principal do google através de um ping pela interface especificada. Pelas imagens em baixo, fica claro que o pedido foi respondido com sucesso e houve captura de pacotes.

```

Terminal - ueransim@ueransim:~
Ficheiro Editar Ver Terminal Separadores Ajuda
ueransim@ueransim:~$ ping -I uesimtun0 google.com -c 5
PING google.com (216.58.209.78) from 10.60.0.1 uesimtun0: 56(84) bytes of data.
64 bytes from google.com (216.58.209.78): icmp_seq=1 ttl=57 time=25.1 ms
64 bytes from google.com (216.58.209.78): icmp_seq=2 ttl=57 time=21.9 ms
64 bytes from google.com (216.58.209.78): icmp_seq=3 ttl=57 time=23.3 ms
64 bytes from google.com (216.58.209.78): icmp_seq=4 ttl=57 time=19.6 ms
64 bytes from google.com (216.58.209.78): icmp_seq=5 ttl=57 time=21.5 ms

--- google.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4005ms
rtt min/avg/max/mdev = 19.574/22.281/25.088/1.845 ms

```

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	fe80::cb0c:d8f0:af8...	ff02::2	ICMPv6	48	Router Solicitation
2	38.197102939	10.60.0.1	216.58.209.78	ICMP	84	Echo (ping) request id=0x1516, seq=1/256, ttl=64 (reply in 3)
3	38.222172473	216.58.209.78	10.60.0.1	ICMP	84	Echo (ping) reply id=0x1516, seq=1/256, ttl=57 (request in 2)
4	39.198482793	10.60.0.1	216.58.209.78	ICMP	84	Echo (ping) request id=0x1516, seq=2/512, ttl=64 (reply in 5)
5	39.220363043	216.58.209.78	10.60.0.1	ICMP	84	Echo (ping) reply id=0x1516, seq=2/512, ttl=57 (request in 4)
6	40.199070282	10.60.0.1	216.58.209.78	ICMP	84	Echo (ping) request id=0x1516, seq=3/768, ttl=64 (reply in 7)
7	40.222355871	216.58.209.78	10.60.0.1	ICMP	84	Echo (ping) reply id=0x1516, seq=3/768, ttl=57 (request in 6)
8	41.200610854	10.60.0.1	216.58.209.78	ICMP	84	Echo (ping) request id=0x1516, seq=4/1024, ttl=64 (reply in 9)
9	41.220163970	216.58.209.78	10.60.0.1	ICMP	84	Echo (ping) reply id=0x1516, seq=4/1024, ttl=57 (request in 8)
10	42.202219538	10.60.0.1	216.58.209.78	ICMP	84	Echo (ping) request id=0x1516, seq=5/1280, ttl=64 (reply in 11)
11	42.223687821	216.58.209.78	10.60.0.1	ICMP	84	Echo (ping) reply id=0x1516, seq=5/1280, ttl=57 (request in 10)

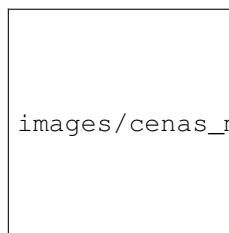
Fig. 5: Teste de ping em uesimtun0

Observando as figuras acima, conseguimos perceber que as comunicações estão estabelecidas e que podemos partir para testes mais complexos.

2) *Teste de ...:*

O objetivo agora é de propor uma nova solução de engenharia de tráfego em que o tráfego HTTP na porta 80 ou 8080 vá por um percurso e o tráfego UDP, nas portas 16384 a 3276, vá por outro alternativo.

Optamos por encaminhar o tráfego UDP pelo caminho LSP1 e o tráfego HTTP pelo caminho LSP2. Usamos o seguinte *route map*, denominado CUSTOMROUTE, para colocar essas restrições nas interfaces exteriores ao domínio MPLS, sendo elas GigabitEthernet0/0 do router n1 e GigabitEthernet0/2 do router n5.



images/cenas_mpls/route_map.png

Fig. 6: Informação sobre o route-map criado



Fig. 7: Interfaces exteriores configuradas

Começamos por instalar o iperf3 nos dois hosts. De seguida testamos um cenário onde n8 atua como um cliente e n10 atua como o servidor. Assim, tentamos chegar a n10 com um pacote TCP na porta 8080. O objetivo é comprovar que este pacote vai pelo caminho LSP2, tal como foi definido no route-map.

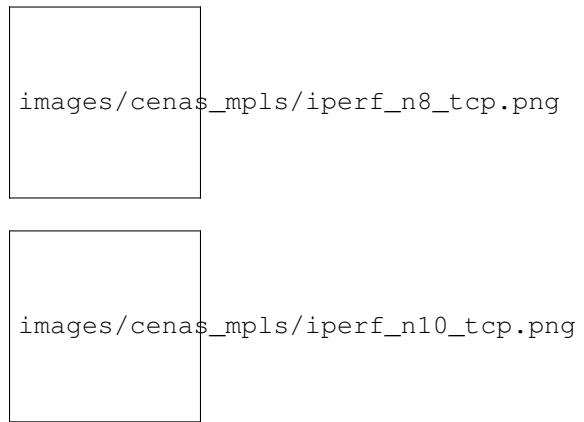


Fig. 8: Comando iperf nos hosts

Tendo duas instâncias de Wireshark a executar, uma na interface 1 do router n3 e outra na interface 1 do router n7, a sequência de pacotes TCP enviadas foi apenas capturada no router n7, como é possível verificar nas imagens em baixo:



Fig. 9: Pacotes capturados durante teste de iperf

Assim conseguimos concluir que a solução de engenharia de tráfego que propomos funciona dentro do esperado, respeitando as regras definidas no enunciado.

III. CONCLUSÃO

Por fim, consideramos que atingimos os objetivos propostos, e aprendemos outros conceitos, como configuração de routers com a sintaxe da CISCO. Infelizmente, despendemos demasiado tempo a pesquisar como implementar os conceitos aprendidos nas aulas teóricas, sendo que inicialmente foi difícil saber que comandos aplicar, e onde. Ainda assim, ficamos satisfeitos com o trabalho final.

REFERENCES

- [1] <https://free5gc.org/>
- [2] <https://github.com/aligungr/UERANSIM>