



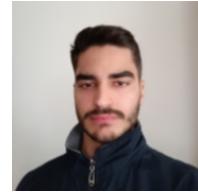
Universidade do Minho
Escola de Engenharia

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 de ser enfrentados para garantir que todos possam 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, 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.

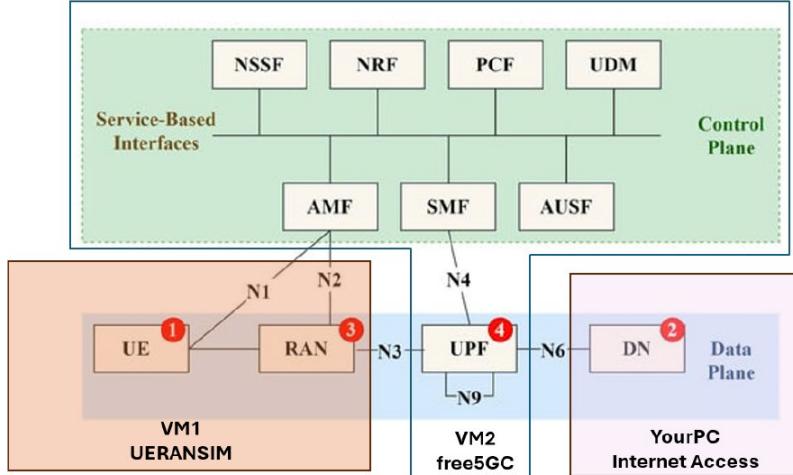


Fig. 1: Esquema da configuração utilizada

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*.

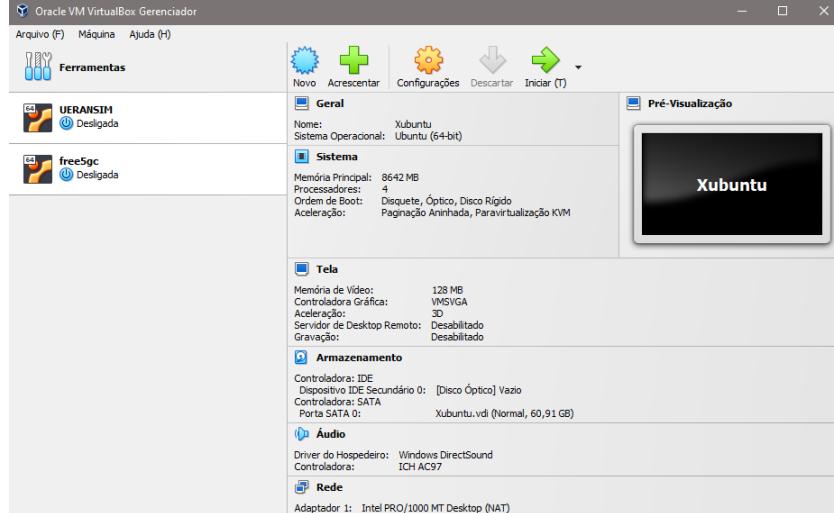


Fig. 2: VM's usadas

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*.

```
Terminal - ueransim@ueransim:~
```

Ficheiro	Editar	Ver	Terminal	Separadores	Ajuda
<pre>inet 192.168.56.102 netmask 255.255.255.0 broadcast 192.168.56.255 inet6 fe80::a00:27ff:fed: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 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</pre>					
ueransim@ueransim:~\$					

Fig. 3: Novo túnel criado

Observando a webconsole do free5gc, é possível ver que o UE criado e que está a ser executado na VM UERANSIM está conectado ao serviço.

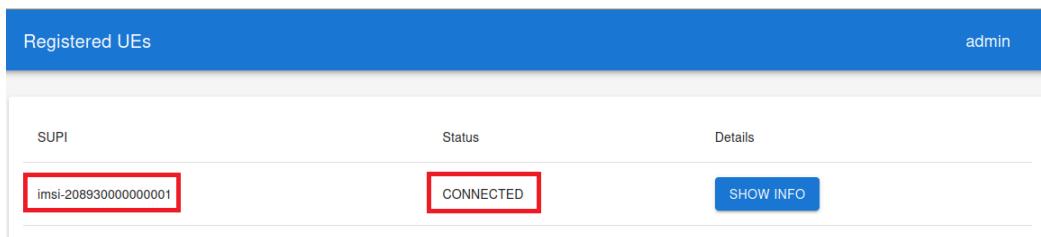


Fig. 4: Estado do UE

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.

```
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
```

```
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. 5: 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.

```
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:1d8f0:af8... ff02::2	216.58.209.78	ICMPv6	48	Router Solicitation
2	38.197192939	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	39.229363943	216.58.209.78	10.60.0.1	ICMP	84	Echo (ping) reply id=0x1516, seq=2/512, ttl=57 (request in 4)
6	40.193978282	10.60.0.1	216.58.209.78	ICMP	84	Echo (ping) request id=0x1516, seq=3/768, ttl=64 (reply in 7)
7	40.238828282	216.58.209.78	10.60.0.1	ICMP	84	Echo (ping) reply id=0x1516, seq=3/768, ttl=57 (request in 6)
8	41.209619854	10.60.0.1	216.58.209.78	ICMP	84	Echo (ping) request id=0x1516, seq=4/1024, ttl=64 (reply in 9)
9	41.229153978	216.58.209.78	10.60.0.1	ICMP	84	Echo (ping) reply id=0x1516, seq=4/1024, ttl=57 (request in 8)
10	42.2022219538	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. 6: 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) Operação de Registo:

Neste teste pretendemos estudar a comunicação entre o UE e o serviço free5gc durante o processo de registo. Primeiro procedemos à captura dos pacotes através do Wireshark na VM do free5gc, onde conseguimos obter com sucesso um pedaço de pacotes relacionados a esse processo.

43 10.248704794	127.0.0.1	127.0.0.1	SSL	391 Continuation Data
44 10.248727148	127.0.0.1	127.0.0.1	TCP	68 34816 - 27017 [ACK] Seq=125 Ack=
45 10.440966391	127.0.0.1	127.0.0.1	SSL	130 Continuation Data
46 10.441176788	127.0.0.1	127.0.0.1	SSL	391 Continuation Data
47 10.441189996	127.0.0.1	127.0.0.1	TCP	68 34850 - 27017 [ACK] Seq=125 Ack=
48 10.738427598	127.0.0.1	127.0.0.1	SSL	130 Continuation Data
49 10.738709949	127.0.0.1	127.0.0.1	SSL	391 Continuation Data
50 10.738722092	127.0.0.1	127.0.0.1	TCP	68 34854 - 27017 [ACK] Seq=125 Ack=
51 10.731655952	127.0.0.1	127.0.0.1	TCP	68 30009 - 34854 [ACK] Seq=1 Ack=
52 10.731409390	127.0.0.1	127.0.0.1	TCP	68 [TCP ACKed unseen segment] 300
53 10.751952544	127.0.0.1	127.0.0.1	TCP	68 34894 - 27017 [ACK] Seq=1 Ack=
54 10.751952982	127.0.0.1	127.0.0.2	TCP	68 [TCP Previous segment not car]
55 10.751956892	127.0.0.2	127.0.0.1	TCP	68 [TCP ACKed unseen segment] [TC]
56 10.751957664	127.0.0.1	127.0.0.1	TCP	68 [TCP ACKed unseen segment] 278
57 10.819680384	192.168.56.102	192.168.56.101	SCTP	84 INIT
58 10.819137344	192.168.56.101	192.168.56.102	SCTP	308 INIT_ACK
59 10.819472250	192.168.56.102	192.168.56.101	SCTP	288 COOKIE_ECHO
60 10.823342440	192.168.56.102	192.168.56.101	SCTP	52 NGAP
61 10.823342444	192.168.56.102	192.168.56.101	NGAP	136 NGSetupRequest
62 10.823595881	192.168.56.101	192.168.56.102	SCTP	64 SACK (Ack=0, Arwnd=106424)
63 10.823291581	192.168.56.101	192.168.56.102	NGAP	128 NGSetupResponse
64 10.823560718	192.168.56.102	192.168.56.101	SCTP	64 SACK (Ack=0, Arwnd=106443)
65 10.144213789	127.0.0.1	127.0.0.1	SSL	391 Continuation Data
66 11.144254665	127.0.0.1	127.0.0.1	TCP	68 34892 - 27017 [ACK] Seq=1 Ack=
67 11.161516768	127.0.0.1	127.0.0.1	SSL	130 Continuation Data
68 11.161699688	127.0.0.1	127.0.0.1	SSL	391 Continuation Data
69 11.161899844	127.0.0.1	127.0.0.1	TCP	68 34884 - 27017 [ACK] Seq=125 Ack=
70 11.162000100	127.0.0.1	127.0.0.1	TCP	68 34884 - 27017 [ACK] Seq=1 Ack=
71 11.162000100	127.0.0.1	127.0.0.1	TCP	68 [TCP ACKed unseen segment] 270
72 11.881978783	127.0.0.1	127.0.0.1	SSL	391 Continuation Data
73 11.881996583	127.0.0.1	127.0.0.1	TCP	68 34820 - 27017 [ACK] Seq=1 Ack=
74 12.105907192	127.0.0.1	127.0.0.1	SSL	391 Continuation Data
75 12.105922838	127.0.0.1	127.0.0.1	TCP	68 34836 - 27017 [ACK] Seq=1 Ack=
76 12.331181546	127.0.0.1	127.0.0.1	SSL	391 Continuation Data
77 12.331229106	127.0.0.1	127.0.0.1	TCP	68 47326 - 27017 [ACK] Seq=1 Ack=
78 12.715319477	127.0.0.1	127.0.0.1	SSL	391 Continuation Data
79 12.715319477	127.0.0.1	127.0.0.1	TCP	68 34854 - 27017 [ACK] Seq=1 Ack=
80 15.359914542	127.0.0.1	127.0.0.1	TCP	68 [TCP Keep-Alive ACK] 35622 - 8000
81 15.359926256	127.0.0.18	127.0.0.18	TCP	68 [TCP Keep-Alive ACK] 8000 - 35622
82 15.359942398	127.0.0.1	127.0.0.1	TCP	68 [TCP Dup ACK 1#1] 34854 - 2701
83 15.359953633	127.0.0.18	127.0.0.1	TCP	68 [TCP Keep-Alive ACK] 8000 - 35
84 15.359954247	127.0.0.1	127.0.0.18	TCP	68 [TCP Dup ACK 3#1] 35622 - 8000
85 15.359956827	127.0.0.1	127.0.0.1	TCP	68 [TCP Dup ACK 4#1] 27017 - 3485
86 19.536136046	127.0.0.1	127.0.0.1	SSL	130 Continuation Data
87 19.536449299	127.0.0.1	127.0.0.1	SSL	391 Continuation Data
88 19.536468103	127.0.0.1	127.0.0.1	TCP	68 34702 - 27017 [ACK] Seq=125 Ack=

Fig. 7: Registo do UE

Foram usadas oito mensagens para a operação de registo, onde o tamanho de cada pacote com SCTP variou entre 64 e 308 bytes e os de NGAP variam entre 120 e 136 bytes. A latência entre os pacotes 61 e 63, que correspondem às mensagens de NGSetupRequest e NGSetupResponse, foi de 0.0029573 segundos, ou seja, 2.9573 milissegundos, um intervalo relativamente curto.

frame.number == 61 frame.number == 63
No. Time Source Destination Protocol Length Info
61 0.000000000 192.168.56.102 192.168.56.101 NGAP 136 NGSetupRequest
63 0.002957337 192.168.56.101 192.168.56.102 NGAP 120 NGSetupResponse

Fig. 8: Latência no registo em NGAP

Analisando os pacotes, conseguimos perceber que os protocolos envolvidos neste processo incluem o SCTP e o NGAP. O SCTP (Stream Control Transmission Protocol) é um protocolo de transporte equiparável ao TCP e UDP, com foco em suporte a múltiplos fluxos, controlo de congestão adaptável e mecanismos de recuperação de falhas.

o NGAP (Next Generation Application Protocol) é o mais relevante para o caso de estudo, pois é um protocolo usado maioritariamente no contexto de redes móveis 5G. Sendo aplicado sob o SCTP, este protocolo é usado principalmente para estabelecer e controlar sessões de utilizador (UE) e recursos na rede 5G, permitindo comunicação entre a rede central (Core Network) e os nós de controle da rede de acesso (Access Network). Observando um dos dos pacotes capturados, mais especificamente o pacote de NGSetupRequest, conseguimos perceber que parte da informação contida na trama NGAP, consiste em informação relativa à RAN (Radio Access Network) utilizada.

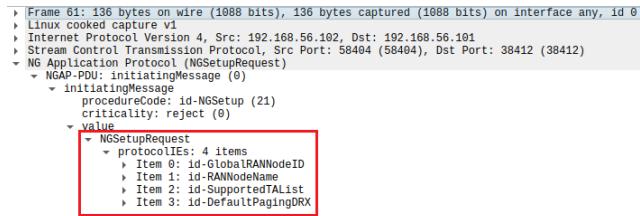


Fig. 9: Exemplo e pacote com NGAP

Também detetamos que após o processo de registo, em determinados intervalos de tempo, surgem pacotes intitulados HEARTBEAT. Estes pacotes servem para verificar a disponibilidade e a integridade de uma associação SCTP entre dois endpoints. O objetivo principal deste tipo de pacote é detetar falhas na conexão SCTP e garantir que a comunicação permaneça estável e confiável.

197	44.126814716	192.168.56.102	192.168.56.101	SCTP	108 HEARTBEAT
198	44.12687195	192.168.56.101	192.168.56.102	SCTP	108 HEARTBEAT_ACK

Fig. 10: Pacotes HEARTBEAT entre UE e free5gc

3) Teste de Download:

Neste teste pretendemos perceber melhor o que acontece em termos de comunicação entre um UE e o Free5GC quando utilizamos uma aplicação de download com ficheiros de diferentes tamanhos. Iremos dar maior relevância ao tipo de pacotes captados e ao intervalo de tempo e velocidade que cada transferência tem. Para o primeiro cenário, recorrendo ao website BuildSomeTech que disponibiliza ficheiros para testes, procedemos ao download de um ficheiro compactado com tamanho de 200 MB. Usamos a ferramenta *wget* onde especificamos que deveria ser usada a interface *uesimtun0* cujo endereço IP é 10.60.0.1 (ver figura 3). Optamos pelo *wget* pois este permite obter informação sobre as latências e velocidades das transferências.

```
wget --bind-address=10.60.0.1 http://212.183.159.230/200MB.zip
```

De seguida testamos a transferência do mesmo ficheiro sem recorrer a essa interface, ou seja, sem recorrer a uma rede 5G.

```
wget http://212.183.159.230/200MB.zip
```

Obtemos como resultado em 5G um tempo de 2 minutos e 26 segundos com uma velocidade de 2,42 MB/s. Usando apenas Wi-Fi, obtemos o ficheiro desejado após 3 minutos e 26 segundos a uma velocidade de 1,73 MB/s. Comparando os dois resultados, conseguimos perceber que a diferença de tempo na transferência entre os dois cenários ronda cerca de um minuto, com a transferência por 5G a ser mais rápida que a Wi-Fi tradicional. Também notamos que a velocidade de transferência em 5G é maior.

```
ueransim@ueransim:~$ wget --bind-address=10.60.0.1 http://212.183.159.230/200MB.zip
--2024-05-09 14:57:10 - http://212.183.159.230/200MB.zip
A ligar a 212.183.159.230:80... ligado.
Pedido HTTP enviado, a aguardar resposta... 200 OK
Tamanho: 209715200 (200M) [application/zip]
A gravar em: '200MB.zip'
200MB.zip          100%[=====] 200,00M  2,42MB/s  em 2m 26s
2024-05-09 14:59:36 (1,37 MB/s) - '200MB.zip' gravado [209715200/209715200]
```

```
ueransim@ueransim:~$ wget http://212.183.159.230/200MB.zip
--2024-05-09 14:59:55-- http://212.183.159.230/200MB.zip
A ligar a 212.183.159.230:80... ligado.
Pedido HTTP enviado, a aguardar resposta... 200 OK
Tamanho: 209715200 (200M) [application/zip]
A gravar em: '200MB.zip.1'

200MB.zip.1          100%[=====] 200,00M [1,73MB/s] em 3m 26s
2024-05-09 15:03:21 (994 KB/s) - '200MB.zip.1' gravado [209715200/209715200]
```

Fig. 11: Download de 200 MB

No entanto 200 MB é relativamente pequeno o que faz com que a diferença seja mínima. Vamos agora testar com um ficheiro comprimido de 1 GB de forma a tirar conclusões mais concretas. Usando novamente os comandos anteriores, apenas alteramos o link do ficheiro que queremos obter.

```
wget --bind-address=10.60.0.1 http://212.183.159.230/1GB.zip
```

```
wget http://212.183.159.230/1GB.zip
```

Os resultados obtidos na transferência por 5G foram de 10 minutos e 45 segundos de tempo e 1,50 MB/s de velocidade. Na transferência em Wi-Fi, obtemos um intervalo de tempo de 27 minutos e 23 segundos com uma velocidade de 1,57 MB/s. Fica claro que mesmo com um objeto muito maior, a rede 5G comporta-se muito melhor que a Wi-Fi tradicional.

```
ueransim@ueransim:~$ wget --bind-address=10.60.0.1 http://212.183.159.230/1GB.zip
--2024-05-09 20:54:10-- http://212.183.159.230/1GB.zip
A ligar a 212.183.159.230:80... ligado.
Pedido HTTP enviado, a aguardar resposta... 200 OK
Tamanho: 1073741824 (1,0G) [application/zip]
A gravar em: '1GB.zip.1'

1GB.zip.1          100%[=====] 1,00G [1,50MB/s] em 10m 45s
2024-05-09 21:04:55 (1,59 MB/s) - '1GB.zip.1' gravado [1073741824/1073741824]
```

```
ueransim@ueransim:~$ wget http://212.183.159.230/1GB.zip
--2024-05-09 15:17:48-- http://212.183.159.230/1GB.zip
A ligar a 212.183.159.230:80... ligado.
Pedido HTTP enviado, a aguardar resposta... 200 OK
Tamanho: 1073741824 (1,0G) [application/zip]
A gravar em: '1GB.zip'

1GB.zip          100%[=====] 1,00G [1,57MB/s] em 27m 23s
2024-05-09 15:45:12 (638 KB/s) - '1GB.zip' gravado [1073741824/1073741824]
```

Fig. 12: Download de 1 GB

Na imagem seguinte estão expostos os pacotes capturados durante a transferência usando o 5G.

2059	0.000052864	127.0.0.8	127.0.0.1	PFCP	329	PFCP Session Report Request
2060	0.000051371	127.0.0.8	127.0.0.1	PFCP	329	PFCP Session Report Request
2061	0.000051770	127.0.0.8	127.0.0.1	PFCP	329	PFCP Session Report Request
2062	0.000053697	127.0.0.8	127.0.0.1	PFCP	329	PFCP Session Report Request
2063	0.000051641	127.0.0.10	127.0.0.1	TCP	39	Syn - > 8890 [PSH, ACK] Seq=62 Ack=65 Win=33280 Len=17 Tsvl=3154688752 T
2064	0.000060836	127.0.0.10	127.0.0.10	TCP	68	44830 - 8890 [ACK] Seq=63 Ack=79 Win=33280 Len=0 Tsvl=2551167939 Tsecr=65
2065	0.000051024	127.0.0.8	127.0.0.1	PFCP	320	PFCP Session Report Request
2066	0.00016227	10.60.0.1	212.183.159.230	GTP <-->	100	60649 - 89 [ACK] Seq=139 Ack=273281 Win=65535 Len=0
2067	0.00000231	10.60.0.1	212.183.159.230	GTP <-->	100	60649 - 89 [ACK] Seq=139 Ack=281441 Win=65535 Len=0
2068	-0.000000231	10.60.0.1	212.183.159.230	TCP	56	60649 - 89 [ACK] Seq=139 Ack=273281 Win=65535 Len=0
2069	0.000035097	10.0.2.15	212.183.159.230	TCP	56	60649 - 89 [ACK] Seq=139 Ack=273281 Win=65535 Len=0
2070	-0.000034866	10.60.0.1	212.183.159.230	TCP	56	60649 - 89 [ACK] Seq=139 Ack=281441 Win=65535 Len=0
2071	0.0000048619	10.0.2.15	212.183.159.230	TCP	56	60649 - 89 [ACK] Seq=139 Ack=281441 Win=65535 Len=0
2072	0.000342513	10.60.0.1	212.183.159.230	GTP <-->	100	60649 - 89 [ACK] Seq=139 Ack=291441 Win=65535 Len=0
2073	0.0000066909	10.60.0.1	212.183.159.230	TCP	56	[TCP Dup ACK 2/2] 60649 - 89 [ACK] Seq=139 Ack=99301 Win=65535 Len=0
2074	0.0000066910	10.60.0.1	212.183.159.230	TCP	56	[TCP Dup ACK 2/2] 60649 - 89 [ACK] Seq=139 Ack=99301 Win=65535 Len=0
2075	0.000374197	127.0.0.8	127.0.0.1	PFCP	329	PFCP Session Report Request
2076	0.000420287	127.0.0.1	127.0.0.10	TCP	173	44830 - 8890 [PSH, ACK] Seq=65 Ack=79 Win=33280 Len=105 Tsvl=2551167940
2077	0.000025182	127.0.0.1	127.0.0.10	TCP	213	44830 - 8890 [PSH, ACK] Seq=170 Ack=79 Win=33280 Len=145 Tsvl=2551167940
2078	0.000031838	127.0.0.1	127.0.0.10	TCP	77	44830 - 8890 [PSH, ACK] Seq=315 Ack=79 Win=33280 Len=9 Tsvl=2551167940 T
2079	0.000034251	127.0.0.1	127.0.0.10	TCP	81	44830 - 8890 [PSH, ACK] Seq=324 Ack=79 Win=33280 Len=13 Tsvl=2551167941
2080	0.0000020502	127.0.0.1	127.0.0.10	TCP	68	44830 - 8890 [FIN, ACK] Seq=337 Ack=79 Win=33280 Len=9 Tsvl=2551167941 T
2081	0.0000076379	127.0.0.1	127.0.0.10	TCP	76	44840 - 8890 [SYN] Seq=0 Win=33280 Len=0 MSS=65495 SACK PERM Tsvl=2551167941
2082	0.0000078339	127.0.0.10	127.0.0.1	TCP	76	44840 - 8890 [SYN, ACK] Seq=0 Ack=1 Win=33280 Len=0 MSS=65495 SACK PERM T
2083	0.000007355	127.0.0.1	127.0.0.10	TCP	68	44840 - 8890 [ACK] Seq=1 Ack=1 Win=33280 Len=0 Tsvl=2551167941 Tsecr=315
2084	0.0000040198	127.0.0.1	127.0.0.10	TCP	132	44840 - 8890 [PSH, ACK] Seq=1 Ack=1 Win=33280 Len=64 Tsvl=2551167941 Tse

Fig. 13: Pacotes durante o download

4) Teste de Múltiplos UPF's:

O objetivo agora é de tornar o cenário um pouco mais complexo, aumentando o número de UPF's (User Plane Function) a executar de um para dois. Seguimos o seguinte tutorial https://github.com/s5uishida/free5gc_ueransim_sample_config?tab=readme-ov-file#changes_ue0.

Após varias tentativas, infelizmente o grupo não conseguiu executar esta tarefa dado a múltiplos erros possivelmente provenientes da configuração do free5gc C-Plane. Erros esses que infelizmente não conseguimos resolver. Na imagem 14 é possível ver os novos UE's que foram criados onde, imsi-208930000000002 deveria usar uma DNN *internet* associada ao UPF1 e imsi-208930000000003 a usar a DNN *ims* associada ao UPF2. Nas imagens 15 e 16 vemos as novas VM's criadas a partir da clonagem da free5gc original. UPF1 representa a free5gc U-Plane1 com IP estático 192.168.56.111 e UPF2 o free5gc U-Plane2 com IP estático 192.168.56.112. Os erros obtidos, pela observação feita, consistiam maioritariamente em erros a encontrar as novas DNN's (Domain Network Name) criadas bem como erros na autenticação dos novos UE's.

PLMN	UE ID	Delete	View
20893	imsi-208930000000001	<button>DELETE</button>	<button>VIEW</button>
20893	imsi-208930000000003	<button>DELETE</button>	<button>VIEW</button>
20893	imsi-208930000000002	<button>DELETE</button>	<button>VIEW</button>

Fig. 14: Novos UE's a criados

```

        }
    },
    DnnList: ([]factory.DnnList) (len=1 cap=1) {
        (*factory.DnnList) {
            Dnn: (string) (len=8) "internet",
            Cidr: (string) (len=12) "10.61.0.0/16",
            NatIfName: (string) (len=4) "eth0"
        }
    },
    Logger: (*factory.Logger)(0xc00043d740)({
        Enable: (bool) true,
        Level: (string) (len=4) "info",
        ReportCaller: (bool) false
    })
}
2024-05-09T22:42:44.394192668+01:00 [INFO][UPF][CFG] =====
2024-05-09T22:42:44.394197214+01:00 [INFO][UPF][Main] Log level is set to [info]
2024-05-09T22:42:44.394202403+01:00 [INFO][UPF][Main] Report Caller is set to [false]
2024-05-09T22:42:44.394215039+01:00 [INFO][UPF][Main] starting Gtp Forwarder [gtp5g]
2024-05-09T22:42:44.394220582+01:00 [INFO][UPF][Main] GTP Address: "192.168.56.111:2152"
2024-05-09T22:42:44.420687804+01:00 [INFO][UPF][BUFF] buff netlink server started
2024-05-09T22:42:44.420985388+01:00 [INFO][UPF][Perio] perio server started
2024-05-09T22:42:44.421037167+01:00 [INFO][UPF][Gtp5g] Forwarder started
2024-05-09T22:42:44.423290222+01:00 [INFO][UPF][PFCP][LAddr:192.168.56.111:8805] starting pfcp server
2024-05-09T22:42:44.423486124+01:00 [INFO][UPF][PFCP][LAddr:192.168.56.111:8805] pfcp server started
2024-05-09T22:42:44.423810160+01:00 [INFO][UPF][Main] UPF started
|

```

Fig. 15: UPF 1

```

        Name: (string) "",
        IfName: (string) "",
        MTU: (uint32) 0
    }
},
DnnList: ([]factory.DnnList) (len=1 cap=1) {
    (factory.DnnList) {
        Dnn: (string) (len=3) "ims",
        Cidr: (string) (len=12) "10.62.0.0/16",
        NatIfName: (string) ""
    }
},
Logger: (*factory.Logger)(0xc00043f740) {
    Enable: (bool) true,
    Level: (string) (len=4) "info",
    ReportCaller: (bool) false
}
)
2024-05-09T22:43:02.353404815+01:00 [INFO][UPF][CFG] =====
2024-05-09T22:43:02.353417221+01:00 [INFO][UPF][Main] Log level is set to [info]
2024-05-09T22:43:02.353422015+01:00 [INFO][UPF][Main] Report Caller is set to [false]
2024-05-09T22:43:02.354623368+01:00 [INFO][UPF][Main] starting Gtpu Forwarder [gtp5g]
2024-05-09T22:43:02.354710040+01:00 [INFO][UPF][Main] GTP Address: "192.168.56.112:2152"
2024-05-09T22:43:02.394675085+01:00 [INFO][UPF][BUFF] buff netlink server started
2024-05-09T22:43:02.394710424+01:00 [INFO][UPF][Perio] perio server started
2024-05-09T22:43:02.394722278+01:00 [INFO][UPF][Gtp5g] Forwarder started
2024-05-09T22:43:02.398573988+01:00 [INFO][UPF][PFCP][Addr:192.168.56.112:8805] starting pfc server
2024-05-09T22:43:02.398600933+01:00 [INFO][UPF][PFCP][Addr:192.168.56.112:8805] pfc server started
2024-05-09T22:43:02.398607484+01:00 [INFO][UPF][Main] UPF started

```

Fig. 16: UPF 2

III. CONCLUSÃO

Dado por terminado este trabalho, refletimos agora sobre o projeto desenvolvido e as conclusões retiradas. Relativamente à primeira parte, ou seja o processo de setup do projeto, consideramos que os tutoriais disponibilizados pelas equipas de desenvolvimento dos softwares Free5GC e UERANSIM, apesar de serem esteticamente bem estruturados, não são suficientemente claros, o que levou a uma atraso significativo no inicio da parte dois, de exploração, do projeto. Com isto queremos dizer que alguns passos cruciais para a instalação e arranque das ferramentas ou eram omitidos, ou então estavam num website completamente à parte, muitas vezes em páginas de fóruns de ajuda. Além disso, durante a instalação do Free5GC, dois elementos do grupo chegaram a um impasse no qual os processadores dos seus computadores não eram compatíveis com algumas ferramentas necessárias durante a instalação, uma vez que não continham as flags avx e avx2.

Relativamente à segunda parte do trabalho, de cariz mais exploratório, vale a pena referir que o grupo teve dificuldade em perceber que tipo de conclusões deveria retirar dos testes que foram feitos, o que acabou por impactar a qualidade final da segunda parte do relatório, que ficou um pouco a carecer de conteúdo. No entanto, dos testes que foram realizados, o grupo focou-se em retirar conclusões que estivessem relacionadas aos tamanhos de pacotes e latência de respostas do serviço Free5GC.

REFERENCES

- [1] <https://free5gc.org/>
- [2] <https://github.com/aligungr/UERANSIM>
- [3] <https://www.buildsome.tech/com/download-test-files/>