



Educação, Pesquisa
e Inovação em Rede

Manual de Instalação de *Core 5G*

Instalando e Configurando Instância do Projeto free5GC

Novembro de 2024

Coordenador do CT-GId

Emerson Ribeiro de Mello, Dr. (IFSC)

Assistente Técnica do CT-GId

Shirlei Aparecida de Chaves, Ma. (IFSC)

Coordenador da RNP para o CT-GId

Clayton Reis da Silva, Me. (RNP)

Diretor Adjunto de e-Ciência e Ciberinfraestrutura Avançada

Leandro Neumann Ciuffo, Me. (RNP)

Diretora de Pesquisa e Desenvolvimento

Iara Machado, Ma. (RNP)

Autores

Leonardo Azalim de Oliveira (UFJF)
Antônio Marcos da Silva Junior (UFJF)
Mariana Siano Pinto (UFJF)
Edelberto Franco Silva, Dr. (UFJF)



SUMÁRIO

Lista de abreviaturas e siglas	3
1 Introdução	4
1.1 Breve Histórico	4
1.2 Versões do Documento	4
2 Configuração do Ambiente	5
2.1 Requisitos para Criação das Máquinas Virtuais	5
2.2 Descrição do Ambiente Virtual Testado	5
2.2.1 <i>Hardware</i> e Sistema Operacional (SO)	5
2.2.2 Aplicações e <i>Software</i>	6
2.3 Ambiente Virtual	7
2.4 Configuração dos Pré-requisitos na Máquina Servidora	7
2.4.1 Etapas iniciais	7
2.4.2 Suporte à Go Lang	7
2.4.3 Banco de Dados	8
2.4.4 Configurar os pré-requisitos do <i>User Plane Function</i> (UPF)	9
2.4.5 Configurar o Control Plane	9
2.5 Instalar o UPF na Máquina Servidora	10
2.5.1 Instalar o WebConsole	10
2.6 Executando o free5GC	10
2.6.1 Execução do <i>5G Core</i> (5GC)	10
2.6.2 Execução do Webconsole	10
2.7 Instalação do UERANSIM na Máquina Cliente	11
2.8 Criação de um UE virtual no free5GC	11
2.8.1 Adição de um novo UE	11
2.9 Conexão do UERANSIM com o free5GC	12
2.9.1 Reaplicar Configurações na Máquina do free5GC após <i>Reboot</i>	14
2.10 Teste de Conexão do UE virtual com o Free5GC	14

Lista de abreviaturas e siglas

5G quinta geração de redes móveis.

5GC *5G Core*.

AMF *Access and Mobility Management Function*.

EAP *Extensible Authentication Protocol*.

Eduroam *Education roaming*.

GNB *gNodeB*.

GTP *GPRS Tunnelling Protocol*.

LAN *Local Area Network*.

LTS *Long-Term Support*.

LVM *Logical Volume Manager*.

NF Função de Rede (*Network Function*).

PGId Programa de Gestão de Identidade.

RNP Rede Nacional de Pesquisa.

SO Sistema Operacional.

UE *User Equipment*.

UPF *User Plane Function*.

Lista de Figuras

1	Exemplo de saída para verificar a presença da instrução AVX na CPU.	8
2	Lista de interfaces disponíveis após conexão do UE com o 5GC.	15
3	Exemplo de saída para o teste de conexão do UE com a internet.	15

Lista de Tabelas

1	Mapeamento das versões do manual.	4
2	Mapeamento das Funções de Rede (NFs, <i>Network Function</i>) do 5GC e respectivas versões utilizadas.	6
3	Listagem das versões dos demais <i>softwares</i> utilizados.	6

1 Introdução

O presente documento foi produzido no âmbito do projeto intitulado “Eduroam e 5G: autenticação integrada via redes móveis e Wi-Fi no core 5G” que faz parte do grupo de projetos de pesquisa em andamento na Rede Nacional de Pesquisa (RNP) em 2024.

O documento detalha os aspectos técnicos do ambiente de testes que foi construído utilizando um estilo de escrita semelhante ao de um guia. Um de seus principais objetivos é melhorar a reprodutibilidade do ambiente utilizado e dos resultados obtidos durante o estudo que foi conduzido.

Este manual assume que o leitor, além de alguma familiaridade com os componentes do 5GC, possui experiência com redes de computadores e com a administração de sistemas Linux, e que os sistemas operacionais utilizados no ambiente são iguais aos mencionados na Subseção 2.2, ou que ao menos sejam do grupo de sistemas “Debian-based”¹.

1.1 Breve Histórico

Este manual foi originalmente criado durante a execução da proposta intitulada “Estudo e Avaliação de Métodos de Autenticação EAP na Infraestrutura de Redes de Telecomunicação 5G” selecionada pelo Programa de Gestão de Identidade (PGId) 2023. Na época de sua concepção, o material técnico relacionado ao projeto free5GC em português era extremamente escasso. Além de trazer as instruções técnicas, o manual também continha comentários extras que possibilitassem a reprodução dos resultados alcançados durante a execução da proposta. A versão anterior do manual e os demais artefatos produzidos em 2023 podem ser acessados no repositório espelho no GitHub².

1.2 Versões do Documento

A Tabela 1 contém as principais modificações realizadas em cada uma das versões deste manual.

Tabela 1: Mapeamento das versões do manual.

Versão	Modificado por	Data Referência	Comentários
1.0.0	Leonardo A. de Oliveira	11/23	Primeira versão pública
1.1.0	Leonardo A. de Oliveira Antônio M. da Silva Jr.	11/24	Atualiza as versões dos softwares testados Revisa a sintaxe dos comandos listados Revisa a grafia do texto do manual Remove temporariamente a seção do N3IWF

¹https://en.wikipedia.org/wiki/List_of_Linux_distributions#Debian-based

²<https://github.com/oliveiraleo/RNP-PGId-2023>

2 Configuração do Ambiente

Esta seção contém as instruções para construção do ambiente de testes juntamente com outras informações gerais.

2.1 Requisitos para Criação das Máquinas Virtuais

O TrueNAS³ foi escolhido como software de virtualização para instalação das máquinas virtuais por conta de sua maior flexibilidade e controle quando comparado com soluções como o Oracle VM VirtualBox⁴ e o VMWare Workstation⁵.

Os softwares citados, bem como os demais disponíveis no mercado podem ser utilizados para a construção do ambiente, pois a forma como as máquinas são executadas é independente do ponto de vista do escopo coberto por este manual.

O principal requisito que deve ser observado durante a escolha da solução de virtualização é a possibilidade de acesso direto à rede do sistema hospedeiro (que geralmente é realizado via *bridge*⁶).

2.2 Descrição do Ambiente Virtual Testado

Esta subseção detalha os requisitos de *software* para correta implantação do ambiente de testes.

2.2.1 Hardware e Sistema Operacional (SO)

- Máquina do 5GC (requisitos mínimos)
 - 2GB RAM
 - 1 *core* de CPU
 - 20GB de HDD
 - Sistema Operacional (SO) Linux Ubuntu Server 20.04.6 *Long-Term Support* (LTS)
- Máquina do 5GC (requisitos recomendados)⁷
 - 4GB RAM ou mais
 - 2 *cores* de CPU ou mais
 - 160GB de HDD/SSD
 - SO Linux Ubuntu Server 20.04.6 LTS

A versão do Ubuntu Server foi escolhida em virtude da presença do kernel Linux versão 5.4.x, que é uma dependência do módulo de *GPRS Tunnelling Protocol* (GTP)⁸ que deve estar instalado no SO em que o 5GC será executado.

- Máquina do UERANSIM (requisitos mínimos)
 - 1GB RAM
 - 1 *core* de CPU

³<https://www.truenas.com/>

⁴<https://www.virtualbox.org/>

⁵<https://www.vmware.com/br/products/workstation-pro.html>

⁶https://www.virtualbox.org/manual/ch06.html#network_bridged

⁷Caso o UPF e as demais NFs sejam instanciados na mesma máquina que o 5GC, utilizar esta configuração

⁸https://en.wikipedia.org/wiki/GPRS_Tunnelling_Protocol

- 10GB de HDD
- Sistema Operacional (SO) Linux Ubuntu Server 20.04.6 LTS
- Máquina do UERANSIM (requisitos recomendados)
 - 2GB RAM
 - 2 *cores* de CPU
 - 20GB de HDD
 - SO Linux Ubuntu Server 20.04.6 LTS ou 22.04.3 LTS

O Ubuntu Linux que foi selecionado como requisito para esta outra máquina virtual é igual a da VM do 5GC somente por que torna mais fácil a criação da mesma a partir de uma clonagem de máquinas, mas é possível, neste caso, utilizar também a versão 22.04.

2.2.2 Aplicações e Software

Por conta da complexidade do ambiente, e com o objetivo de documentar os demais *softwares* e componentes utilizados em sua construção, as Tabelas 2 e 3 apresentam uma listagem dos *softwares* e das respectivas versões testadas.

Tabela 2: Mapeamento das NFs do 5GC e respectivas versões utilizadas.

NF	<i>Commit</i>	Versão
AMF	299aeb99d8a2cf3350ad9abfb3f2abb9586cf003	1.2.5
AUSF	842217c4784b193f48e92a218fcee2ac903d944f	1.2.3
CHF	0f1a6470b2c17a426fc127482c8d10c7ec7a3326	1.0.3
N3IWF	9fe155e2de2dc6e6fc8dcc18a47b9b3b6138a3b3	1.2.4
NRF	4d85a024f3637365b22113657b55caf812231f48	1.2.5
NSSF	a73f10ff6d7176d5bbae7315ad5f23136b56a143	1.2.3
PCF	bc1afea63efcc9d53e4cc3c5a8b68e8529bcf820	1.2.5
SMF	ed09202fd1615cfae758faf245e7ece9709a0170	1.2.5
UDM	3f284476f2b70a6f4f1330f2c59f7bdee17e0ec9	1.2.3
UDR	ee6e0c866c4ebc0ff5050c94cf6843ac41a49205	1.2.4
UPF	922281ba58b6ca42d73be218671c8feae3658420	1.2.3

Tabela 3: Listagem das versões dos demais *softwares* utilizados.

<i>Software</i>	Versão
MongoDB	3.6.8
Go	1.21.8
GTP-U	v0.8.10
UERANSIM	v3.2.6
CMake	3.16.3

2.3 Ambiente Virtual

Para correta execução do ambiente de testes, serão necessárias ao menos 2 máquinas virtuais. Como sugerido na documentação do free5GC, é possível realizar a configuração das máquinas virtuais a partir de uma máquina “base”.

De acordo com a documentação⁹, durante o processo de instalação do SO dessa máquina, é recomendado habilitar o servidor OpenSSH para simplificar o acesso remoto ao sistema e *não* habilitar o *Logical Volume Manager (LVM)*¹⁰ no disco da máquina para facilitar futuras modificações no tamanho e disposição de suas partições.

Após terminar de instalar uma máquina virtual¹¹ com Ubuntu Server, criar uma cópia desta máquina e seguir as instruções contidas na Subseção 2.4.

2.4 Configuração dos Pré-requisitos na Máquina Servidora

A presente subseção e a Subseção 2.5 contém as instruções relacionadas aos pré-requisitos para construção do ambiente de testes e, idealmente, devem ser executadas somente uma vez.

2.4.1 Etapas iniciais

- Atualizar os pacotes do sistema

```
1 sudo apt update && sudo apt upgrade -y
```

- Verificar a versão do kernel

```
1 uname -r
```

NOTA: Por conta do *patch* aplicado pelo GTP necessário para o 5GC funcionar é necessário instalar um kernel da versão 5.4.x.

2.4.2 Suporte à Go Lang

- Verificar a versão instalada

```
1 go version
```

Saída exemplo:

```
1 go version go1.18.10 linux/amd64
```

- Instalar o Go 1.21.8

Caso uma outra versão de Go já tenha sido instalada no passado, executar:

```
1 # this assumes your current version of Go is in the default location:
2 sudo rm -rf /usr/local/go
3 wget https://dl.google.com/go/go1.21.8.linux-amd64.tar.gz
4 sudo tar -C /usr/local -zxvf go1.21.8.linux-amd64.tar.gz
```

⁹<https://free5gc.org/guide/1-vm-en/#3-create-a-ubuntu-server-vm>

¹⁰[https://en.wikipedia.org/wiki/Logical_Volume_Manager_\(Linux\)](https://en.wikipedia.org/wiki/Logical_Volume_Manager_(Linux))

¹¹<https://free5gc.org/guide/1-vm-en/>

Se não, basta executar esses outros comandos abaixo:

```
1 wget https://dl.google.com/go/go1.21.8.linux-amd64.tar.gz
2 sudo tar -C /usr/local -zxvf go1.21.8.linux-amd64.tar.gz
3 # The following assumes that your shell is bash:
4 echo 'export PATH=$PATH:/usr/local/go/bin' >> ~/.bashrc
5 source ~/.bashrc
```

2.4.3 Banco de Dados

- Se for usar o mongoDB 5.x ou superior, verificar se as instruções AVX são suportadas pela CPU onde a máquina está rodando, pois estas são requisito para correta execução

```
1 grep --color avx /proc/cpuinfo
```

Um exemplo de saída para o comando acima pode ser visto na Figura 1.

```
[~]$ grep --color avx /proc/cpuinfo
flags       : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr
sse sse2 ss ht tm pbe syscall nx rdtscp lm constant_tsc arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc cp
uid aperfmperf pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic po
pcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm cpuid_fault epb pti ssbd ibrs ibpb stibp tpr_shadow flexpr
iority ept vpid fsgsbase smep erms xsaveopt dtherm ida arat pln pts vnmi md_clear flush_l1d
flags       : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr
sse sse2 ss ht tm pbe syscall nx rdtscp lm constant_tsc arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc cp
uid aperfmperf pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic po
pcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm cpuid_fault epb pti ssbd ibrs ibpb stibp tpr_shadow flexpr
iority ept vpid fsgsbase smep erms xsaveopt dtherm ida arat pln pts vnmi md_clear flush_l1d
flags       : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr
sse sse2 ss ht tm pbe syscall nx rdtscp lm constant_tsc arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc cp
uid aperfmperf pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic po
pcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm cpuid_fault epb pti ssbd ibrs ibpb stibp tpr_shadow flexpr
iority ept vpid fsgsbase smep erms xsaveopt dtherm ida arat pln pts vnmi md_clear flush_l1d
flags       : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr
sse sse2 ss ht tm pbe syscall nx rdtscp lm constant_tsc arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc cp
uid aperfmperf pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic po
pcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm cpuid_fault epb pti ssbd ibrs ibpb stibp tpr_shadow flexpr
iority ept vpid fsgsbase smep erms xsaveopt dtherm ida arat pln pts vnmi md_clear flush_l1d
flags       : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr
sse sse2 ss ht tm pbe syscall nx rdtscp lm constant_tsc arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc cp
uid aperfmperf pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic po
pcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm cpuid_fault epb pti ssbd ibrs ibpb stibp tpr_shadow flexpr
iority ept vpid fsgsbase smep erms xsaveopt dtherm ida arat pln pts vnmi md_clear flush_l1d
flags       : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr
```

Figura 1: Exemplo de saída para verificar a presença da instrução AVX na CPU.

Se houver suporte, é possível instalar o mongoDB 5.x, senão usar a versão 4.4.x. O comando abaixo instala por padrão no Ubuntu 20.04 a versão 3.x

```
1 sudo apt install mongodb wget git
```

- Iniciar o serviço do banco de dados

```
1 sudo systemctl start mongodb
```

- Verificar se o serviço do banco de dados está ativo

```
1 systemctl status mongodb
```

NOTA: O comando acima deve retornar "active (running)"

- [Opcional] Para iniciar o serviço do banco de dados quando o SO iniciar

```
1 sudo systemctl enable mongod
```

2.4.4 Configurar os pré-requisitos do UPF

- Instalar os pré-requisitos para executar o UPF

```
1 sudo apt -y install git gcc g++ cmake autoconf libtool pkg-config libmnl-dev libyaml-dev
```

- Habilitar o encaminhamento de pacotes e configurar o *firewall* do servidor

```
1 sudo sysctl -w net.ipv4.ip_forward=1
2 sudo iptables -t nat -A POSTROUTING -o <dn_interface> -j MASQUERADE
3 sudo iptables -A FORWARD -p tcp -m tcp --tcp-flags SYN,RST SYN -j TCPMSS --set-mss 1400
4 sudo systemctl stop ufw
5 sudo systemctl disable ufw
```

No caso do Ubuntu Server 20.04, os comandos ficariam da seguinte forma:

```
1 sudo sysctl -w net.ipv4.ip_forward=1
2 sudo iptables -t nat -A POSTROUTING -o enp0s3 -j MASQUERADE
3 sudo iptables -A FORWARD -p tcp -m tcp --tcp-flags SYN,RST SYN -j TCPMSS --set-mss 1400
4 sudo systemctl stop ufw
5 sudo systemctl disable ufw
```

2.4.5 Configurar o Control Plane

- Clonar o repositório

```
1 git clone --recursive -b v3.4.3 -j `nproc` https://github.com/free5gc/free5gc.git
2 cd free5gc
```

NOTA: A versão mais recente disponível no momento de escrita era a 3.4.3, verificar se há uma nova versão no repositório oficial¹²

- Compilar as NFs

Para compilar todas as NFs:

```
1 cd ~/free5gc
2 make
```

Para compilar uma NF específica:

```
1 cd ~/free5gc
2 make <NF>
```

OBS: Substituir <NF> pelo nome em caixa baixa do módulo a ser compilado (ex: upf)

¹²<https://github.com/free5gc/free5gc>

2.5 Instalar o UPF na Máquina Servidora

Começar instalando o módulo personalizado do kernel:

```
1 cd ~ #retorna para a home do usuario
2 git clone -b v0.8.10 https://github.com/free5gc/gtp5g.git
3 cd gtp5g
4 make
5 sudo make install
```

NOTA: A versão mais recente disponível no momento de escrita era a 0.8.10, verificar se há uma nova versão no repositório oficial¹³

OBS: Neste tutorial será mantida a configuração padrão, mas se necessário, personalize o UPF como desejado. O arquivo de configuração invocado pelo *script* `run.sh` está localizado em `free5gc/config/upfcfg.yaml`

2.5.1 Instalar o WebConsole

Começar instalando as dependências:

```
1 curl -fsSL https://deb.nodesource.com/setup_20.x | sudo -E bash -
2 sudo apt update
3 sudo apt install -y nodejs
4 sudo corepack enable # setup yarn automatically
```

Compilar o webconsole¹⁴ usando:

```
1 cd ~/free5gc
2 make webconsole
```

Para executar o Webconsole, olhar a Subseção 2.6.2.

2.6 Executando o free5GC

2.6.1 Execução do 5GC

Para instanciar o 5GC free5GC¹⁵

```
1 cd ~/free5gc/
2 ./run.sh
```

2.6.2 Execução do Webconsole

Para que o módulo funcione corretamente, seus pré-requisitos devem ter sido previamente instalados de acordo com as instruções da Subseção 2.5.1 e a instância do 5GC deve estar sendo executada de acordo com as instruções contidas na Subseção 2.6.1.

- Executar o servidor do módulo Webconsole

Na máquina onde está o *core*:

¹³<https://github.com/free5gc/gtp5g>

¹⁴Mesmo ao executar o comando *make* para compilar todas as NF, o Webconsole não é compilado

¹⁵Caso seja necessário, dar permissão de execução para o *script* usando o comando `chmod +x run.sh`

```
1 cd ~/free5gc/webconsole
2 ./bin/webconsole
```

Após o servidor estar em execução, a seguinte mensagem deve ser exibida no terminal:

```
[GIN-debug] Listening and serving HTTP on :5000
```

2.7 Instalação do UERANSIM na Máquina Cliente

Aproveitando a máquina virtual que foi criada a partir das instruções da Subseção 2.3, criar um clone dela¹⁶ e então executar

```
1 sudo apt update && sudo apt upgrade -y
```

- Instalar as dependências

```
1 sudo apt install -y make g++ libsctp-dev lksctp-tools iproute2
2 sudo snap install cmake --classic
```

- Clonar o repositório

```
1 cd ~
2 git clone https://github.com/aligungr/UERANSIM.git
3 cd UERANSIM
```

E avançar a um *commit* específico

```
1 git checkout 2134f6b
```

Caso ainda não tenha sido feito, executar na máquina do 5GC as instruções para instalação do Webconsole que se encontram na Subseção 2.5.1. Esse console será útil para a configuração do *User Equipment* (UE) virtual.

- Compilar o UERANSIM

```
1 cd ~/UERANSIM
2 make
```

NOTA: O processo de compilação demorou em torno de 5 minutos na máquina com a configuração descrita como recomendada que foi utilizada nos testes

O processo deve finalizar com a seguinte mensagem: "UERANSIM successfully built."

2.8 Criação de um UE virtual no free5GC

Caso ainda não esteja, executar o Webconsole de acordo com as instruções da Subseção 2.6.2.

2.8.1 Adição de um novo UE

- Para adicionar um novo UE no core

Abriu um *browser* e navegar até o endereço `http://<IP>:5000`, onde IP é o endereço de IP da VM do *core*.

Exemplo: `http://192.168.56.101:5000`

¹⁶Lembrar de alterar o *hostname* e o IP desta nova máquina para não conflitar com a máquina do 5GC

As credenciais de acesso ao Webconsole são `admin:free5gc`

Na página inicial, navegar pela barra lateral esquerda até a opção “Subscribers”. Clicar nela e depois, na nova página que abrir, no botão “Create”. No *pop-up* que abrir, haverá diversos parâmetros de configuração. Rolar a tela até atingir o parâmetro “Operator Code Type” e alterá-lo para “OP”.

Neste tutorial os demais parâmetros ficarão com seus valores padrão. Clicar no botão “Create” para salvar os dados do novo UE.

NOTA: Neste ponto já é possível sair do *browser* e fechar o processo do servidor do Webconsole usando `Ctrl+C`

2.9 Conexão do UERANSIM com o free5GC

Na máquina do *core*, alterar os IPs que se encontram dentro de 3 arquivos de configuração mudando dos IPs de *loopback* para o da *Local Area Network* (LAN). Inicialmente, editar o arquivo `amfcfg.yaml`.

```
1 cd ~/free5gc
2 nano config/amfcfg.yaml
```

Na linha

```
1 ngapIpList: # the IP list of N2 interfaces on this AMF
2   - 127.0.0.18
```

Alterar para

```
1 ngapIpList: # the IP list of N2 interfaces on this AMF
2   - <IP>
```

Caso o IP da máquina do 5GC fosse `172.16.0.114`, a linha ficaria da seguinte forma:

```
1 ngapIpList: # the IP list of N2 interfaces on this AMF
2   - 172.16.0.114
```

O próximo arquivo a ser alterado é `smfcfg.yaml`

```
1 nano config/smfcfg.yaml
```

Navegar pelas seções `userplaneInformation` / `upNodes` / `UPF` / `interfaces` / `endpoints` de dentro do arquivo até encontrar:

```
1 interfaces: # Interface list for this UPF
2   - interfaceType: N3 # the type of the interface (N3 or N9)
3     endpoints: # the IP address of this N3/N9 interface on this UPF
4       - 127.0.0.8
```

E alterar o IP. Considerando o exemplo, o arquivo ficaria da seguinte forma:

```
1 interfaces: # Interface list for this UPF
2   - interfaceType: N3 # the type of the interface (N3 or N9)
3     endpoints: # the IP address of this N3/N9 interface on this UPF
4       - 172.16.0.114
```

Por fim, no arquivo `upfcfg.yaml`

```
1 nano config/upfcfg.yaml
```

Alterar o IP em:

```
1 gtpu:
2   forwarder: gtp5g
3   # The IP list of the N3/N9 interfaces on this UPF
4   # If there are multiple connection, set addr to 0.0.0.0 or list all the addresses
5   ifList:
6     - addr: 127.0.0.8
7     type: N3
```

Para, de acordo com o ambiente de exemplo:

```
1 gtpu:
2   forwarder: gtp5g
3   # The IP list of the N3/N9 interfaces on this UPF
4   # If there are multiple connection, set addr to 0.0.0.0 or list all the addresses
5   ifList:
6     - addr: 172.16.0.114
7     type: N3
```

Agora, na máquina do UERANSIM, devemos apontar onde o *core* se encontra na rede.

Existem dois arquivos de configuração no UERANSIM que são relacionados com o free5GC:

- ~/UERANSIM/config/free5gc-gnb.yaml
- ~/UERANSIM/config/free5gc-ue.yaml

Configurar tanto o IP da LAN no UERANSIM, quanto apontar qual o IP da *Access and Mobility Management Function (AMF)* do 5GC

```
1 nano ~/UERANSIM/config/free5gc-gnb.yaml
```

Nas linhas que possuem:

```
1 ngapIp: 127.0.0.1 # gNB's local IP address for N2 Interface (Usually same with local IP)
2 gtpIp: 127.0.0.1 # gNB's local IP address for N3 Interface (Usually same with local IP)
3
4 # List of AMF address information
5 amfConfigs:
6   - address: 127.0.0.1
```

Colocar nas duas primeiras linhas o IP da máquina do UERANSIM, já na outra o IP da máquina do 5GC. Abaixo segue um exemplo com os IPs fictícios:

```
1 ngapIp: 172.16.0.118 # gNB's local IP address for N2 Interface (Usually same with local IP)
2 gtpIp: 172.16.0.118 # gNB's local IP address for N3 Interface (Usually same with local IP)
3
4 # List of AMF address information
5 amfConfigs:
6   - address: 172.16.0.114
```

Abrir o arquivo do UE:

```
1 nano ~/UERANSIM/config/free5gc-ue.yaml
```

Alterar o parâmetro:

```
1 # This value specifies the OP type and it can be either 'OP' or 'OPC'
2 opType: 'OPC'
```

Para:

```
1 # This value specifies the OP type and it can be either 'OP' or 'OPC'
2 opType: 'OP'
```

E checar se os demais parâmetros do arquivo conferem com aqueles que aparecem no Webconsole¹⁷. Então verificar se é necessário executar os comandos da subseção 2.9.1.

2.9.1 RePLICAR Configurações na Máquina do free5GC após *Reboot*

Toda vez que a máquina do 5GC reiniciar, executar os comandos de configuração do *firewall* e habilitar o encaminhamento de pacotes no kernel

```
1 sudo sysctl -w net.ipv4.ip_forward=1
2 sudo iptables -t nat -A POSTROUTING -o enp0s4 -j MASQUERADE
3 sudo iptables -I FORWARD 1 -j ACCEPT
```

Lembrar manter o serviço de banco de dados em execução por meio do comando abaixo

```
1 sudo systemctl start mongod
```

2.10 Teste de Conexão do UE virtual com o Free5GC

Na máquina do 5GC, instanciar o *core*:

```
1 cd ~/free5gc
2 ./run.sh
```

Agora serão necessárias 3 conexões SSH (ou 3 abas de terminal) na máquina do UERANSIM:
Na primeira, instanciar um *gNodeB* (gNB):

```
1 cd ~/UERANSIM
2 build/nr-gnb -c config/free5gc-gnb.yaml
```

Na segunda, instanciar um UE:

```
1 cd ~/UERANSIM
2 sudo build/nr-ue -c config/free5gc-ue.yaml # for multiple-UEs, use -n and -t for number and delay
```

Na terceira, verificar se a interface de conexão foi criada com sucesso:

```
1 ip addr show
```

O comando acima deve retornar algo como o que é mostrado no exemplo da Figura 2. Note que a interface *uesimtun0* é aquela que foi criada pelo UE. O próximo passo é testar a conexão do UE com a internet usando, por exemplo, o comando abaixo:

```
1 ping -I uesimtun0 google.com
```

Uma saída de exemplo pode ser vista na Figura 3.

¹⁷Caso o UE tenha sido criado com a configuração padrão, provavelmente os parâmetros já se encontram corretos

```
netlab@ueransim:~$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s4: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:a0:98:34:4f:4a brd ff:ff:ff:ff:ff:ff
    inet 172.16.0.118/16 metric 100 brd 172.16.255.255 scope global dynamic enp0s4
        valid_lft 5426sec preferred_lft 5426sec
    inet6 fe80::2a0:98ff:fe34:4f4a/64 scope link
        valid_lft forever preferred_lft forever
3: uesimtun0: <POINTOPOINT,PROMISC,NOTRAILERS,UP,LOWER_UP> mtu 1400 qdisc fq_codel state UNKNOWN group default qlen 500
    link/none
    inet 10.60.0.1/32 scope global uesimtun0
        valid_lft forever preferred_lft forever
    inet6 fe80::2c5f:7ba1:123b:7ac/64 scope link stable-privacy
        valid_lft forever preferred_lft forever
```

Figura 2: Lista de interfaces disponíveis após conexão do UE com o 5GC.

```
3: uesimtun0: <POINTOPOINT,PROMISC,NOTRAILERS,UP,LOWER_UP> mtu 1400 qdisc fq_codel state UNKNOWN group default qlen 500
    link/none
    inet 10.60.0.1/32 scope global uesimtun0
        valid_lft forever preferred_lft forever
    inet6 fe80::2c5f:7ba1:123b:7ac/64 scope link stable-privacy
        valid_lft forever preferred_lft forever
netlab@ueransim:~$ ping -I uesimtun0 google.com
PING google.com (142.251.132.238) from 10.60.0.1 uesimtun0: 56(84) bytes of data:
.
64 bytes from gru14s46-in-f14.1e100.net (142.251.132.238): icmp_seq=1 ttl=50 time=25.6 ms
64 bytes from gru14s46-in-f14.1e100.net (142.251.132.238): icmp_seq=2 ttl=50 time=25.5 ms
64 bytes from gru14s46-in-f14.1e100.net (142.251.132.238): icmp_seq=3 ttl=50 time=26.8 ms
64 bytes from gru14s46-in-f14.1e100.net (142.251.132.238): icmp_seq=4 ttl=50 time=25.7 ms
64 bytes from gru14s46-in-f14.1e100.net (142.251.132.238): icmp_seq=5 ttl=50 time=25.7 ms
^C
--- google.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 6634ms
rtt min/avg/max/mdev = 25.535/25.891/26.842/0.480 ms
```

Figura 3: Exemplo de saída para o teste de conexão do UE com a internet.

