A diagram of a computer

Description automatically generated

# **Setup system**

## **I.1 Setup Manager Node.**

Requirement: Install 5GC Core and LISP

### I.1.1 Install 5GC core.

Following document on **https://free5gc.org/guide/3-install-free5gc/** :

1. Install Golang Version 1.18.10

* # remove go
* sudo rm -rf /usr/local/go
* dpkg -l | grep golang
* sudo apt-get remove golang\*
* # install go 1.18
* wget https://dl.google.com/go/go1.18.10.linux-amd64.tar.gz
* sudo tar -C /usr/local -zxvf go1.18.10.linux-amd64.tar.gz
* mkdir -p ~/go/{bin,pkg,src}
* # The following assume that your shell is bash:
* echo 'export GOPATH=$HOME/go' >> ~/.bashrc
* echo 'export GOROOT=/usr/local/go' >> ~/.bashrc
* echo 'export PATH=$PATH:$GOPATH/bin:$GOROOT/bin' >> ~/.bashrc
* echo 'export GO111MODULE=auto' >> ~/.bashrc
* source ~/.bashrc
* go version

1. Control-plane Supporting Packages (mongodb version 4.4)

* # Install mongodb 4.4 for control-plane
* sudo apt -y update
* wget -qO - https://www.mongodb.org/static/pgp/server-4.4.asc | sudo apt-key add -
* echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu $(lsb\_release -cs)/mongodb-org/4.4 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-4.4.list
* sudo apt-get update
* sudo apt-get install -y mongodb-org
* sudo systemctl start mongod
* sudo systemctl enable mongod
* # User-plane Supporting Packages
* sudo apt -y update
* sudo apt -y install git gcc g++ cmake autoconf libtool pkg-config libmnl-dev libyaml-dev

1. Clone the free5GC repository.

* # install control-plane
* cd ~
* git clone --recursive -b v3.3.0 -j `nproc` https://github.com/free5gc/free5gc.git

1. Compile network function services in free5gc

* cd free5gc
* make
* # Install User Plane Function (UPF)
* git clone -b v0.8.2 https://github.com/free5gc/gtp5g.git
* cd gtp5g
* make
* sudo make install
* cd ~/free5gc
* make upf

1. Install WebConsole

* # Note: 2GB or more of OS memory is recommended. WebConsole may be failed to build if memory is less then 1GB.
* sudo apt remove cmdtest yarn
* curl -sS https://dl.yarnpkg.com/debian/pubkey.gpg | sudo apt-key add -
* echo "deb https://dl.yarnpkg.com/debian/ stable main" | sudo tee /etc/apt/sources.list.d/yarn.list
* sudo apt update
* sudo apt install -y nodejs yarn
* cd ~/free5gc
* make webconsole

1. Change 5GC config

* cd ~/free5gc
* At run.sh located in free5gc folder, command two lines for disable UPF.



* At config/amfcfg.yaml, change AMF IP address:

A screenshot of a computer

Description automatically generated

* At config/smfcfg.yaml, change SMF config like below:

A screenshot of a computer

Description automatically generated

A close up of a computer screen

Description automatically generated

A screenshot of a computer code

Description automatically generated

A close-up of a message

Description automatically generated

1. Change SMF code for getting UP information

At /free5gc/NFs/smf/internal/context/datapath.go , add one more line (add code) and one more function Print()

A screen shot of a computer code

Description automatically generated

A black and white text

Description automatically generated

At /free5gc/NFs/smf/internal/sbi/producer/oam.go , update PDUSessionInfor and httpResponse as follows:

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

### I.1.2 Install LISP control plane.

Follow the <https://github.com/binhfdv/lispers-config> to set up configurations

|  |  |
| --- | --- |
|  |  |

In Manager node: install lisp-ms

In Worker 1 and Worker 3: install lisp-etr for each one

Replace content in files

- src/lisp.py from the original codes with that of and lisp\_v3.py

- src/lisp-etr.py from the original codes with that of lisp-etr-prometheus.py (remember to replace the Prometheus query in the codes with your own one in order to get the service resources)

- src/lisp-ms.py from the original codes with that of scr/lisp-ms.py

Notes: Ctrl + F to find term: “added code” to add api endpoints for dynamically changing lisp UI displaying values.

## **I.2 Setup Worker2** **Node.**

Requirement: Install **GNB** & **UE** following document on <https://github.com/aligungr/UERANSIM>

* cd ~
* git clone https://github.com/aligungr/UERANSIM
* sudo apt update
* sudo apt upgrade
* sudo apt install make
* sudo apt install gcc
* sudo apt install g++
* sudo apt install libsctp-dev lksctp-tools
* sudo apt install iproute2
* sudo snap install cmake –classic
* cd ~/UERANSIM
* make

### I.2.1 Run GNB with GNB config file:

A close-up of a computer screen

Description automatically generated

Change IP AMF

Chang **ngapIP** in GNB config file to IP which connects to AMF

Change **gtpIP** (10.100.200.3) in GNB config file to IP which connects to UPF

Then run GNB:

* ./build/nr-gnb -c config/free5gc-gnb.yaml

### I.2.2 Run UE with UE config file:

Change GNB IP in config/free5gc-ue.yaml

A close-up of blue text

Description automatically generated

Then run UE:

* ./build/nr-ue -c config/free5gc-ue.yaml

## **I.3 Setup Worker3/Worker1 Node.**

Requirement: Install K8S cluster with service running as Anycast IP address.

### I.3.1 Install K8S cluster version 1.22.

* sudo apt-get install kubeadm=1.22.0-00 kubelet=1.22.0-00 kubectl=1.22.0-00 -y
* sudo kubeadm init --pod-network-cidr=10.244.0.0/16
* mkdir -p $HOME/.kube
* sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
* sudo chown $(id -u):$(id -g) $HOME/.kube/config
* kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

### I.3.2 Install service-a running in K8S cluster.

Deploy service deployment by service.yaml file

A screenshot of a computer program

Description automatically generated

Deploy service by svc-deployment.yaml file. Cluster IP was setup with 10.96.10.10

A screenshot of a computer code

Description automatically generated

## **I.4 Setup Mininet Node.**

Following instructions of Onos+P4 tutorial command: <https://github.com/opennetworkinglab/ngsdn-tutorial>

A screenshot of a computer program

Description automatically generated

Clone P4-MUP-SRv6 repository and install required dependencies:

* cd P4-MUP-SRv6
* **make deps**

**Start P4-MUP-SRv6 project:**

* cd P4-MUP-SRv6
* **make restart**

Wait about 1 minute for warming up.

Inject network config file to Onos:

* make netcfg

Activate Onos application:

* make app-reload

Wait about 1 minute for warming up. Then go to Browser to check Onos system on <http://192.168.20.11:8181/onos/ui/login.html#/topo2>

* User: onos
* Password: rocks

Can monitor the network topology in real-time.

A network of connected lines and dots

Description automatically generated with medium confidence

Install “bridge-utils” packet for Mininet container:

* docker exec -it mininet sh
* echo "deb http://archive.debian.org/debian stretch main" > /etc/apt/sources.list && chmod 1777 /tmp && rm -rf /var/lib/apt/lists/\* && apt-get update && apt install bridge-utils

Then Ctrl+D to exit.

* **Connect worker2 to virtual Switch running inside Mininet container**

Then create a bridge for linking virtual ethernet of host(v\_eth1) to virtual ethernet of switch R1(v\_eth3) running inside Mininet:

* docker exec -it mininet brctl addbr mybridge
* ip link add v\_eth1 type veth peer name v\_eth2 && ifconfig v\_eth1 up
* ip link set v\_eth2 netns $(docker inspect --format='{{ .State.Pid }}' $(docker ps -aqf "name=mininet"))
* docker exec -it mininet ifconfig v\_eth2 up
* docker exec -it mininet brctl addif mybridge v\_eth2
* docker exec -it mininet brctl addif mybridge v\_eth3
* docker exec -it mininet ifconfig mybridge 10.100.200.19/24 up
* docker exec -it mininet ping 10.100.200.20 -I mybridge -c 2

Then create a bridge for linking virtual ethernet of host(v\_eth1) to physical ethernet of host(enxf8e43b4d17e5):

* ovs-vsctl del-br br0
* ovs-vsctl add-br br0
* ifconfig br0 up
* ovs-vsctl add-port br0 enxf8e43b4d17e5
* ovs-vsctl add-port br0 v\_eth1
* ifconfig enxf8e43b4d17e5 0
* ifconfig br0 10.100.200.4/24 up
* ping 10.100.200.20 -I br0 -c 2
* we can connect worker2(10.100.200.3) to virtual host running on Mininet (10.100.200.20).
* Test by “ping 10.100.200.20” from worker2
* **Connect worker3 to virtual Switch running inside Mininet container**

Then create a bridge for linking virtual ethernet of host(v2\_eth1) to virtual ethernet of switch R8(v2\_eth3) running inside Mininet:

* ip link add v2\_eth1 type veth peer name v2\_eth2 && ifconfig v2\_eth1 up
* ip link set v2\_eth2 netns $(docker inspect --format='{{ .State.Pid }}' $(docker ps -aqf "name=mininet"))
* docker exec -it mininet brctl addbr mybridge2
* docker exec -it mininet ifconfig v2\_eth2 up
* docker exec -it mininet brctl addif mybridge2 v2\_eth2
* docker exec -it mininet brctl addif mybridge2 v2\_eth3
* docker exec -it mininet ifconfig mybridge2 192.168.0.19/24 up
* docker exec -it mininet ping 192.168.0.20 -I mybridge2 -c 2

Then create a bridge for linking virtual ethernet of host(v2\_eth1) to physical ethernet of host(enxf8e43b26ac73):

* ovs-vsctl del-br br1
* ovs-vsctl add-br br1
* ifconfig br1 up
* ovs-vsctl add-port br1 enxf8e43b26ac73
* ovs-vsctl add-port br1 v2\_eth1
* ifconfig enxf8e43b26ac73 0
* ifconfig br1 192.168.0.4/24 up
* ping 192.168.0.20 -I br1 -c 2
* we can connect worker3(192.168.0.3) to virtual host running on Mininet (192.168.0.20).
* Test by “ping 192.168.0.20” from worker3
* **Connect worker1 to virtual Switch running inside Mininet container**

Then create a bridge for linking virtual ethernet of host(v3\_eth1) to virtual ethernet of switch R9(v3\_eth3) running inside Mininet:

* ip link add v3\_eth1 type veth peer name v3\_eth2 && ifconfig v3\_eth1 up
* ip link set v3\_eth2 netns $(docker inspect --format='{{ .State.Pid }}' $(docker ps -aqf "name=mininet"))
* docker exec -it mininet brctl addbr mybridge3
* docker exec -it mininet ifconfig v3\_eth2 up
* docker exec -it mininet brctl addif mybridge3 v3\_eth2
* docker exec -it mininet brctl addif mybridge3 v3\_eth3
* docker exec -it mininet ifconfig mybridge3 192.168.1.19/24 up
* docker exec -it mininet ping 192.168.1.20 -I mybridge3 -c 2

Then create a bridge for linking virtual ethernet of host(v3\_eth1) to physical ethernet of host(enxaaaaaaaaab7f):

* ovs-vsctl del-br br2
* ovs-vsctl add-br br2
* ifconfig br2 up
* ovs-vsctl add-port br2 enxaaaaaaaaab7f
* ovs-vsctl add-port br2 v3\_eth1
* ifconfig enxaaaaaaaaab7f 0
* ifconfig br2 192.168.1.4/24 up
* ping 192.168.1.20 -I br2 -c 2
* we can connect worker1(192.168.1.3) to virtual host running on Mininet (192.168.1.20).
* Test by “ping 192.168.1.20” from worker1

# **Running system**

* **Step 1: RUNNING 5G system**

On Worker3: start UPF

* cd NEW\_5GC/free5gc
* ./bin/upf

On Worker: start webconsole

* cd free5gc/webconsole
* ./bin/webconsole

On Worker: start 5GC core

* cd free5gc
* ./run.sh

On Worker2 node: start GNB

* cd NEW\_5GC/UERANSIM
* ./build/nr-gnb -c ./config/free5gc-gnb.yaml

On Worker2 node: start UE

* cd NEW\_5GC/UERANSIM
* ./build/nr-ue -c ./config/free5gc-ue.yaml
* **Step 2: RUNNING MININET NETWORK**

On Worker3: delete routing rules if have

* route del -net 10.60.0.0/24 gw 192.168.0.20

On Mininet: restart network

* cd P4-MUP-SRv6
* make restart
* docker exec -it mininet sh
* echo "deb http://archive.debian.org/debian stretch main" > /etc/apt/sources.list && chmod 1777 /tmp && rm -rf /var/lib/apt/lists/\* && apt-get update && apt install bridge-utils
* make netcfg
* make app-reload
* bash run.sh

Then on Worker2

* ping 10.100.200.20

On Worker3:

* ping 192.168.0.20

On Worker1:

* ping 192.168.1.20
* **Step3: Testing traffic from UE to UPF running in Worker3 and then terminate UPF.**

On Worker2:

* ping 192.168.0.3 -I uesimtun0 -i 0.3

On Worker3:

* stop UPF running by Ctrl+C
* **Step4: Running SRv6 MUP configuration**

On Worker3: add routing rules

* route add -net 10.60.0.0/24 gw 192.168.0.20

On Mininet:

* bash srv6\_mup.sh
* **Step5: Access service 10.96.10.10 running at worker3 from UE**

On Worker2:

* while sleep 1; do curl 10.96.10.10:5000 --interface uesimtun0; done
* **Step6: Steer UE traffic from worker3 to worker1**

On Mininet:

* bash util/onos-cmd route-gtp4d-insert device:r1 **10.96.10.10** 24 fcbb:bb00:01:: fcbb:bb00:6:7:9:fd44::

Notes:

* netplan config for internet connectivity:
* https://www.serverlab.ca/tutorials/linux/administration-linux/how-to-configure-networking-in-ubuntu-20-04-with-netplan/