Document

“Protocol Stack Integration

with core network”

Table of Contents

[1 Introduction 4](#_Toc54186500)

[1.1 System setup 4](#_Toc54186501)

[1.1.1 Interface Overview 5](#_Toc54186502)

[1.1.2 Interface Setup ( JSON file – All values has to be in Decimal format ) 6](#_Toc54186503)

[2 Prerequisites and Installation Steps ( Ubuntu 16.04 ) 7](#_Toc54186504)

[2.1 JSON 7](#_Toc54186505)

[2.2 ROHC 8](#_Toc54186506)

[2.3 Nettle 8](#_Toc54186507)

[2.4 Libexplain 8](#_Toc54186508)

[2.5 DPDK 9](#_Toc54186509)

[2.6 ASN1 9](#_Toc54186510)

[2.7 SCTP 9](#_Toc54186511)

[2.8 AESNI and IPSEC 9](#_Toc54186512)

[2.9 Exporting Environment Variables 10](#_Toc54186513)

[2.10 Loading Hugepages 10](#_Toc54186514)

[3 Running setup ( Ubuntu 16.04 ) 11](#_Toc54186515)

[3.1 gNB setup 11](#_Toc54186516)

[3.2 UE setup 11](#_Toc54186517)

Table of Figures

[Figure 1 System setup overview 4](#_Toc54186518)

[Figure 2 Control Plane Interface details 5](#_Toc54186519)

[Figure 3 User Plan Interface details 5](#_Toc54186520)

# Introduction

This document covers details of integration done between Protocol stack and core network. This document will cover details of setup, interfaces and commands.

## System setup

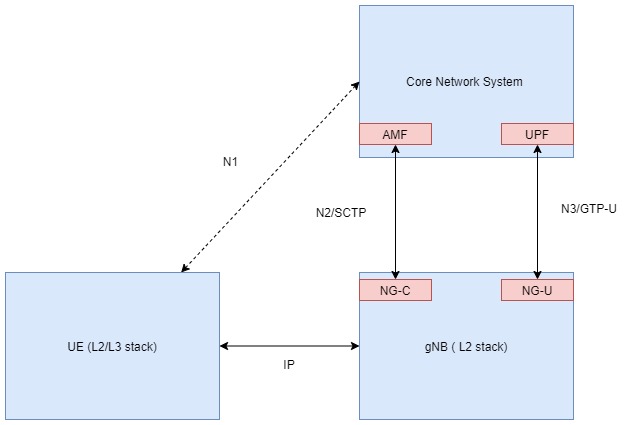


Figure 1 System setup overview

Setup consist of 3 different systems running

* Core Network
* gNB protocol stack
* UE protocol stack

### Interface Overview

Core Network <-> gNB interface

Control Plane

* N2 Interface
* Uses SCTP protocol for communication
* Core Network entity involved for communication: NG-C at AMF
* gNB entity involved of communication: NG-C at gNB

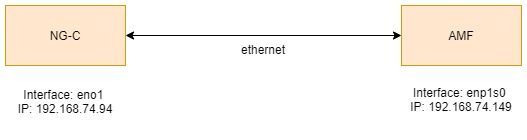


Figure 2 Control Plane Interface details

User Plane

* N1 Interface
* Uses GTP-U protocol for communication
* Core Network entity involved for communication: UPF
* gNB entity involved for communication: NG-U interface

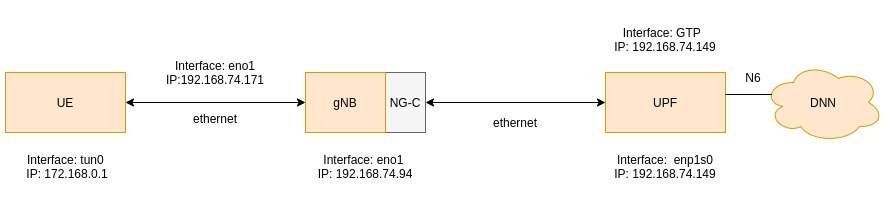


Figure 3 User Plan Interface details

### Interface Setup ( JSON file – All values has to be in Decimal format )

**gNB Side:**

For Socket Connection “wnSock”, Need to update “src” which is own system MAC addr in decimal format and “dst” which is destination system MAC addr to be provide in JSON file

Need to provide own(gNB System) ip Addr “ip”

**UE Side:**

For Socket Connection “wnSock”, Need to update “src” which is own system MAC addr in decimal format and “dst” which is destination system MAC addr to be provide in JSON file

# Prerequisites and Installation Steps ( Ubuntu 16.04 )

For installing all the prerequisites except AESNI and IPSEC, use setupInitial.sh file

Command to run : sh setupInitial.sh

## JSON

#Installing Pre-requisites:

sudo apt install git

sudo apt install autoconf automake libtool

sudo apt install valgrind # optional

sudo apt install cmake

#Build instructions:

git clone https://github.com/json-c/json-c.git

mkdir json-c-build

cd json-c-build

cmake ../json-c

sudo apt-get install libjson-c-dev

#To build and run the test programs:

cd json-c-build

make test

make USE\_VALGRIND=0 test # optionally skip using valgrind

## ROHC

git clone https://git.cryptomilk.org/projects/cmocka.git ~/rohc\_ext\_lib/cmocka

cd ~/rohc\_ext\_lib/cmocka

mkdir build

cd build

cmake -DCMAKE\_INSTALL\_PREFIX=/usr -DCMAKE\_BUILD\_TYPE=Debug ~/rohc\_ext\_lib/cmocka

make

sudo make install

sudo apt install libpcap-dev gawk mawk( Optional, depends on system)

wget -O ~/rohc\_ext\_lib/rohc-2.3.1.tar.xz https://rohc-lib.org/download/rohc-2.3.x/2.3.1/rohc-2.3.1.tar.xz

cd ~/rohc\_ext\_lib/

tar -xvf rohc-2.3.1.tar.xz

cd rohc-2.3.1/

./autogen.sh --enable-static

make

sudo make install

## Nettle

sudo apt-get install -y nettle-dev

## Libexplain

sudo apt-get update

sudo apt-get install libexplain-dev

## DPDK

wget http://fast.dpdk.org/rel/dpdk-20.02.1.tar.xz

tar -xf dpdk-20.02.1.tar.xz

cd dpdk-stable-20.02.1

make defconfig

make install T=x86\_64-native-linuxapp-gcc

## ASN1

sudo apt-get install asn1c

## SCTP

sudo apt-get install -y libsctp-dev

## AESNI and IPSEC

Download nasm compiler from https://www.nasm.us/

* Download nasm-2.14.03rc2 version.
* mkdir nasm
* Unzip the zip file to nasm folder
* Goto nasm folder
* Vim configure
* Press ESC
* :set fileformat=unix
* :wq
* ./configure
* vim ./include/nasmlib.h
* add line of code : #include <time.h>
* make
* make install

Download intel-ipsec file by https://github.com/01org/intel-ipsec-mb/archive/v0.52.zip and unzip it

cd intel-ipsec-mb-0.52

make

sudo make install

Export path to IPSEC Library:

• example "export AESNI\_MULTI\_BUFFER\_LIB\_PATH ='/home/suraj/Documents/

• Workspace/dpdk-18.05.1/intel-ipsec-mb' ".Path will be different.

• Also you can alternatively keep export path in /home/user/.bashrc file

Set CONFIG\_RTE\_LIBRTE\_PMD\_AESNI\_MB=y in config/common\_base in dpdk install directory and then follow below commands.

* make
* sudo make install T=x86\_64-native-linuxapp-gcc

## Exporting Environment Variables

export RTE\_SDK= <path to DPDK folder installed>

Ex: export RTE\_SDK=/home/venkatrahul/Downloads/dpdk-20.02.1/dpdk-stable-20.02.1/

export RTE\_TARGET=x86\_64-native-linuxapp-gcc

## Loading Hugepages

sudo su

# echo 4096 > /sys/kernel/mm/hugepages/hugepages-2048kB/nr\_hugepages

# exit

# Running setup ( Ubuntu 16.04 )

Order of running setup is for this integration is as follow

* Starting gNB
* Staring UE

## gNB setup

1. sudo chmod 777 gnbapp
2. sudo ./gnbapp <interface name> 0

Ex: sudo ./gnbapp eno1 0

* 1. 0 if we don’t want to connect with GTP using socket
  2. To enable logs use -- -<flags>

All the configurations will be taken from JSON file ( Values available in JSON are Decimal format)

## UE setup

1. Sudo chmod 777 ueapp
2. sudo ./ueapp <interface name> 0

Ex: sudo ./ueapp eno1 0

* 1. 0 if you don’t want to enable any additional logs
  2. To enable logs use -- -<flags>

1. sudo ifconfig tun00 172.168.0.1 netmask 255.255.255.0 up
2. sudo ip route add 192.168.74.149 via 172.168.0.1 dev tun00

All the configurations will be taken from JSON file ( Values available in JSON are Decimal format )

**NOTE:**   
1. Currently UE wont support packet filters. Core network need to configure QFI2 only to check data path.

2. Need to give CTRL+C to terminate ue/gnb( When terminating gNB, it will Seg fault. Ignore it). In order re-run the ue/gnb, need to terminate 1st and then run.