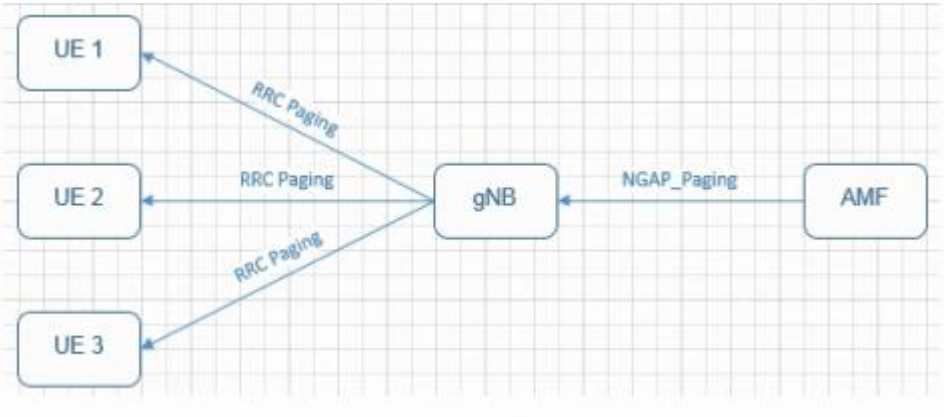
Họ và Tên: Phạm Hồng Hùng

MSSV: 20182558

Project 3 – 5G NR paging simulation

Part 1: Requirement

Để thực hiện project này trước tiên sinh viên phải có kiến thức cơ bản về Socket cũng như cách để lập trình Socket nhằm giải quyết bài toán mô phỏng trao đổi dữ liệu giữa các thực thể mạng. Sau khi đã nắm được cái kiến thức cơ bản về Socket sinh viên phải biết một số các khái niệm, thuật ngữ cơ bản trong mạng 5G (gNB, AMF, UE) để có cái nhìn tổng quan về đề project. Quan trọng nhất sinh viên cần phải hiểu Paging là gì? Tại sao phải sử dụng Paging?. Khi đã có được những hiểu biết về 5G và Paging sinh viên cần phải hình dung ra được quá trình gói tin Paging đi từ AMF tới gNB sau đó tới UE thông qua hình vẽ.



Hình 1: mô hình truyền bản tin Paging

Tiến hành xây dựng thuật toán và code:

Sinh viên cần xem yêu cầu của bài toán đặt ra là gì, từ những dữ kiện cho trước từ đó xây dựng 1 thuật toán phù hợp để thực hiện cũng như giải quyết các vấn đề của project.

Kiến thức cơ bản về lập trình C vô cùng quan trọng để lập trình Socket trên linux các kiến về Struct (mô tả cấu trúc gói tin), vòng lặp, biến, câu lệnh rẽ nhánh.

Part 2: System design

Hệ thống gồm có gNB (server), AMF (client) và các UE (client)

- AMF:
  - Khởi động sau gNB, có thể trước hoặc sau UE
  - gửi bản tin định kỳ xuống cho gNB sau mỗi khoảng thời gian nhất định
  - bản tin từ AMF có thể là bản tin hợp lệ hoặc không
  - trường UE_ID là Random
  - nhận thông báo từ gNB về bản tin AMF vừa gửi (hợp lệ hoặc không)

- gNB:
  - khởi động trước AMF, UE
  - Đếm thời gian ngay từ khi khởi động
  - với AMF:
    - nhận bản tin từ AMF
    - kiểm tra bản tin có hợp lệ hay không, trả về thông báo cho AMF
    - nếu giá bản tin hợp lệ thì lưu vào bộ nhớ để chờ gửi cho UE
  - với UE:
    - gNB khi khởi động sẽ bắt đầu đếm thời gian
    - khi UE lần đầu kết nối tới, gNB sẽ gửi SFN_gNB cho UE để đồng bộ
    - gNB nhận được các thông tin cần thiết của UE để tính toán thời gian thức dậy của UE
    - gửi toàn bộ các bản tin Paging trong bộ đệm cho UE khi UE thức dậy

- UE:
  - Khởi động sau gNB, có thể trước hoặc sau AMF
  - Quá trình kết nối đầu tiên nhận được SFN_gNB sau đó tính toán khoảng thời gian sleep() hợp lý thức dậy
  - Nhận toàn bộ bản tin Paging từ gNB
  - Kiểm tra bản tin Paging
  - Hiển thị thông báo nhận bản tin thành công

Part 3: Code & Result

3.1 Code

- Note: Phần code này em sử dụng Domain AF_UNIX thay cho AF_INET vì em nghĩ AF_INET khác biệt với AF_UNIX ở chỗ có thể kết nối với máy khác, mà mỗi bạn có 1 kiểu code khác nhau nên khả năng khi chạy chéo code khó mà chạy được nên em dùng AF_UNIX cho đơn giản.
- gNB.c :

```
#include <time.h>

#include <signal.h>

int main()

{

time_t begin = time(NULL);

int server_sockfd, client_sockfd;

int server_len, client_len;

struct sockaddr_un server_address;

struct sockaddr_un client_address;

struct mess {

        int message_type;

        int UE_ID;

        int TAC;

        int CN_Domain;

}NGAP;

int temp=0;

int temp1;

int ue_id_i = 3;

int Dong_bo = 1;

int chu_ky = 1;
```

```
int i = 0;
int SFN_gNB=0;
struct mess Queue[20];
/* 2. Loại bỏ các tên hay liên kết socket khác trước đó nếu có. Đồng thời thực hiện khởi tạo socket mới cho trình chủ */
unlink( "server_socket" );
server_sockfd = socket( AF_UNIX, SOCK_STREAM, 0 );
/* 3. Đặt tên cho socket của trình chủ */
server_address.sun_family = AF_UNIX;
strcpy( server_address.sun_path, "server_socket" );
server_len = sizeof( server_address );
/* 4. Ràng buộc tên với socket */
bind( server_sockfd, (struct sockaddr *)&server_address, server_len );
/* 5. Mở hàng đợi nhận kết nối - cho phép đặt hàng vào hàng đợi tối đa 5 kết nối */
listen( server_sockfd, 5 );
signal (SIGCHLD, SIG_IGN);
/* 6. Lặp vĩnh viễn để chờ và xử lý kết nối của trình khách */
//fork();
while ( 1 ) {
printf( "server waiting...\n" );
/* Chờ và chấp nhận kết nối */
client_sockfd = accept( server_sockfd, (struct sockaddr*)&client_address, &client_len );
time_t end = time(NULL);
SFN_gNB = end - begin;
if (SFN_gNB > 1024){
        SFN_gNB = SFN_gNB - 1024;
```

```
}
printf("The elapsed time is %d seconds\n", SFN_gNB);
/* Đọc dữ liệu do trình khách gửi đến */
        read(client_sockfd, &temp,1);
        temp1 = temp;
        if(temp1 == 1&&Dong_bo==1){
                printf("Dong_bo: %d\n", temp1);
                write(client_sockfd, &SFN_gNB,1);
                read(client_sockfd, &ue_id_i,1);
                read(client_sockfd, &chu_ky,1);
                printf("chu_ky: %d\n", chu_ky);
                printf("ue_id_i: %d\n", ue_id_i);
                Dong_bo = 0;
        }
        else if(temp1!=0){//1
        NGAP.message_type=temp;
        printf("%d\n", temp);
        read(client_sockfd, &temp,1);
        NGAP.UE_ID=temp;
        printf("%d\n", temp);
        read(client_sockfd, &temp,1);
        NGAP.TAC=temp;
        printf("%d\n", temp);
        read(client_sockfd, &temp,1);
        NGAP.CN_Domain=temp;
        printf("%d\n", temp);
```

```
        if (NGAP.message_type == 100){
                if(NGAP.TAC ==100){
                        if(NGAP.CN_Domain == 100 || NGAP.CN_Domain == 101){
                                temp = 0;
                                write( client_sockfd, &temp,1 );
                                Queue[i] =NGAP;
                                i = i+1;
                                printf("%d\n", i);
                        }
                else{
                temp = 1;
                write(client_sockfd, &temp,1 );
                }
        }
        else{
        temp = 1;
        write(client_sockfd, &temp,1 );
        }
}
else{
        temp = 1;
        write(client_sockfd, &temp,1 );
}
}//1
        else{
        if(((SFN_gNB-ue_id_i)%chu_ky)==0 && Dong_bo == 0){
```

```
                temp = i;
                write(client_sockfd, &temp,1);
                while ( i > 0){
                        temp = Queue[i-1].message_type;
                        write(client_sockfd, &temp,1);

                        temp = Queue[i-1].UE_ID;
                        write(client_sockfd, &temp,1);

                        temp =Queue[i-1].TAC;
                        write(client_sockfd, &temp,1);

                        temp =Queue[i-1].CN_Domain;
                        write(client_sockfd, &temp,1);
                        i = i -1 ;
                        printf("%d\n",i);
                }
        }
}
/* Đóng kết nối */
close( client_sockfd );
        }
}
```

- AMF.c :

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <unistd.h>
#include <time.h>
int main()
{
int sockfd; /* số mô tả socket – socket handle */
int len;
struct sockaddr_un address; /* structure quan trọng, chứa các thông tin về socket */
struct mess {
        int message_type;
        int UE_ID;
        int TAC;
        int CN_Domain;
}NGAP;
int result;
int temp; // bien tam
while(1){//*
sleep(3);
NGAP.message_type = 99+Get_Random(0,1);
NGAP.UE_ID=Get_Random(2,4);
NGAP.TAC =100;
```

```
NGAP.CN_Domain=101;
/* 2. Tạo socket cho trình khách. Lưu lại số mô tả socket */
sockfd = socket( AF_UNIX, SOCK_STREAM, 0 );
address.sun_family = AF_UNIX;
/* 3. Gán tên của socket trên máy chủ cần kết nối */
strcpy( address.sun_path, "server_socket" );
len = sizeof( address );
/* 4. Thực hiện kết nối */
result = connect( sockfd, (struct sockaddr*)&address, len );
if ( result == -1 ) {
perror( "Oops: client1 problem" );
exit( 1 );
}
/* 5. Sau khi socket kết nối, chúng ta có thể đọc ghi dữ liệu của socket tương tự đọc ghi trên file */
        temp = NGAP.message_type;
        write (sockfd, &temp,1);

        temp = NGAP.UE_ID;
        write (sockfd, &temp,1);

        temp =NGAP.TAC;
        write (sockfd, &temp,1);

        temp =NGAP.CN_Domain;
        write (sockfd, &temp,1);
```

```
        read ( sockfd, &temp, 1 );
        //printf("%d\n",temp);
if (temp == 1){

        printf("Ban tin khong hop le\n");

}
if(temp == 0){
        printf("Truyen thanh cong\n");
}
close( sockfd );
//exit( 0 );
}//*
}

int Get_Random(int min,int max){
   return min + (int)(rand()*(max-min+1.0)/(1.0+RAND_MAX));
}
```

- UE.c :

```
/* 1. Tạo các #include cần thiết để gọi hàm socket */
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <unistd.h>
#include <stdlib.h>
int main()
{
struct mess {
int message_type;
        int UE_ID;
        int TAC;
        int CN_Domain;
}RRC;
int ue_id_i =3;
int chu_ky = 17;
int SFN_UE = 0;
int Dong_bo = 1;
int temp =0;
int k = 0;
int i = 0;

while(1){
printf("SFN_UE: %d\n", SFN_UE);
```

```
if(SFN_UE > 1024){
        SFN_UE = SFN_UE - 1024;
        k = 0;
}
int sockfd=0; /* số mô tả socket – socket handle */
int len=0;
struct sockaddr_un address; /* structure quan trọng, chứa các thông tin về socket */
int result=0;
/* 2. Tạo socket cho trình khách. Lưu lại số mô tả socket */
sockfd = socket( AF_UNIX, SOCK_STREAM, 0 );
address.sun_family = AF_UNIX;
/* 3. Gán tên của socket trên máy chủ cần kết nối */
strcpy( address.sun_path, "server_socket" );
len = sizeof( address );
/* 4. Thực hiện kết nối */
result = connect( sockfd, (struct sockaddr*)&address, len );
if ( result == -1 ) {
perror( "Oops: client1 problem" );
exit( 1 );
}
/*5. Sau khi socket kết nối, chúng ta có thể đọc ghi dữ liệu của socket tương tự đọc ghi trên file */
        if(Dong_bo == 1){ // qua trinh dong bo thoi gian cua UE
                write (sockfd, &Dong_bo,1);
                printf("Dong Bo: %d\n", Dong_bo);
                read ( sockfd, &SFN_UE, 1 );
```

```
                printf("SFN_UE: %d\n", SFN_UE);
                write (sockfd, &ue_id_i,1);
                write (sockfd, &chu_ky,1);
                k = (int)((SFN_UE-ue_id_i)/chu_ky);
                printf("K: %d\n",k);
                temp = ue_id_i + (k+1)*chu_ky - SFN_UE; // tinh khoang thoi gian can phai
sleep de dong bo
                printf("temp: %d\n", temp);
                close( sockfd );
                sleep(temp);
                SFN_UE = ue_id_i + (k+1)*chu_ky;
                printf("SFN_UE: %d\n", SFN_UE);
        }
        else{
        write (sockfd, &Dong_bo,1);
        read(sockfd, &temp,1);
        i = temp;
        printf("i: %d\n",i);
        while ( i > 0){
        read(sockfd, &temp,1);
        RRC.message_type=temp;
        printf("RRC.message: %d\n", temp);
        read(sockfd, &temp,1);
        RRC.UE_ID=temp;
        printf("RRC.UE_ID: %d\n", temp);
        read(sockfd, &temp,1);
```

```
        RRC.TAC=temp;
        //printf("%d\n", temp);
        read(sockfd, &temp,1);
        RRC.CN_Domain=temp;
        //printf("%d\n", temp);
                printf("SFN_UE: %d\n", SFN_UE);
                i = i -1;
        if (RRC.message_type == 100 && RRC.UE_ID == ue_id_i){
        printf("Paging Thanh Cong\n");
}
        else {
        printf("Ban Tin Cua UE khac\n");
}
        }
        sleep(chu_ky);
        SFN_UE=SFN_UE+chu_ky;

        close( sockfd );
        }
        Dong_bo = 0;
        }
}
int Get_Random(int min,int max){
   return min + (int)(rand()*(max-min+1.0)/(1.0+RAND_MAX));
}
```

3.2 Result

Đồng bộ

```
server waiting...
The elapsed time is 21 seconds
Dong_bo: 1
chu_ky: 17
ue_id_i: 3
```

// có thể thấy thời điểm UE bắt đầu kết nối vào mạng là sau khi gNB khởi động 21s

//UE gửi cho gNB chu kỳ thức dậy (chu_ky)

//UE gửi cho gNB ID của mình (ue_id_i)

// nhờ ue_id_i và chu_ky gNB có thể tính được thời gian tiếp theo UE thức dậy

```
hung@hung-VirtualBox:~/Documents/Socket_programing/AF_UNIX$ ./UE
SFN_UE: 0
Dong Bo: 1
SFN_UE: 21
K: 1
temp: 16
SFN_UE: 37
```

//khi khởi động SFN_UE = 0;

// quá trình đồng bộ gNB gửi SFN_gNB cho UE, UE setup giá trị đó vào SFN_UE để đồng bộ thời gian với gNB

// từ SFN_UE, UE tính giá trị temp phù hợp để sleep, ở đây chu_ky = 17, ue_id_i = 3

Vậy thời gian thức dậy kế tiếp sẽ là: SFN_UE =SFN_gNB = 3 + (k+1)*17 = 3 + 34 = 37

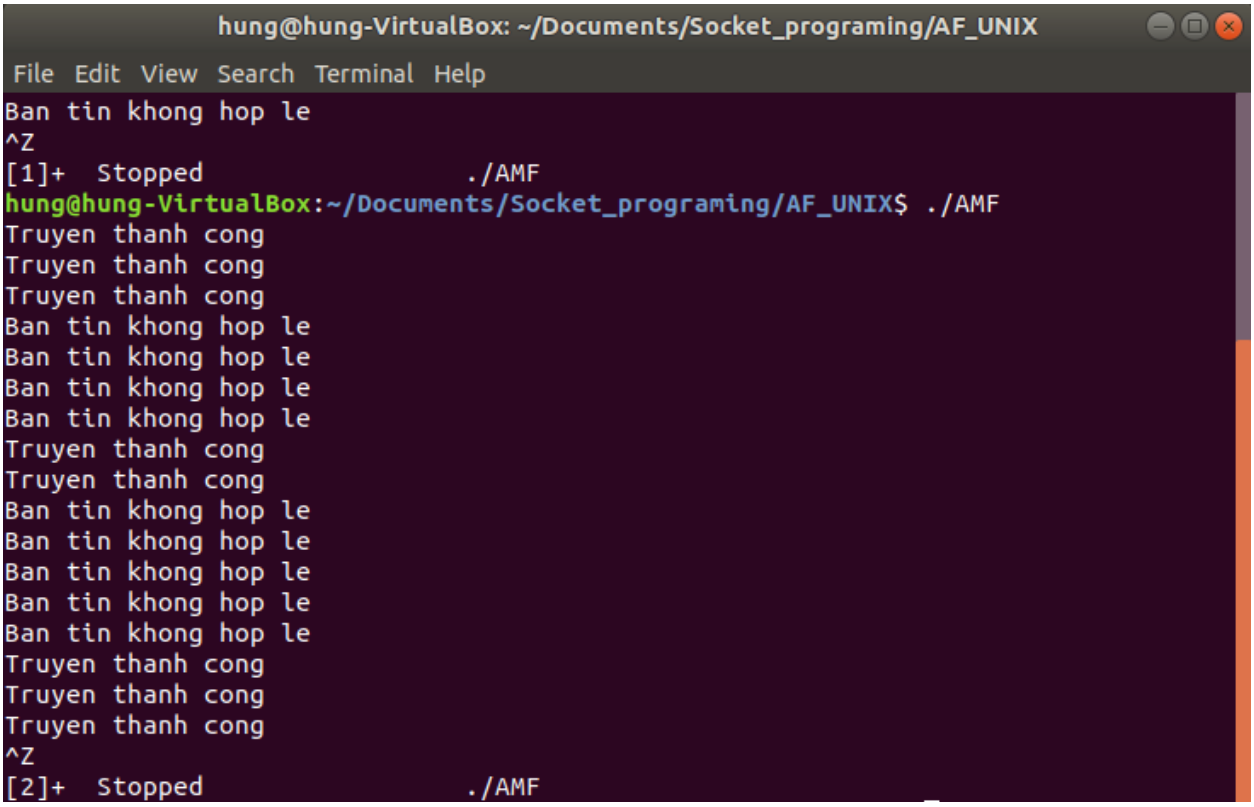
=> temp = 37 -21 = 16

Là quãng thời gian cần sleep cho tới lần thức dậy tiếp theo

// kết thúc quá trình đồng bộ

// sau đó UE chỉ cần thức dậy sau mỗi chu_ky (17s)

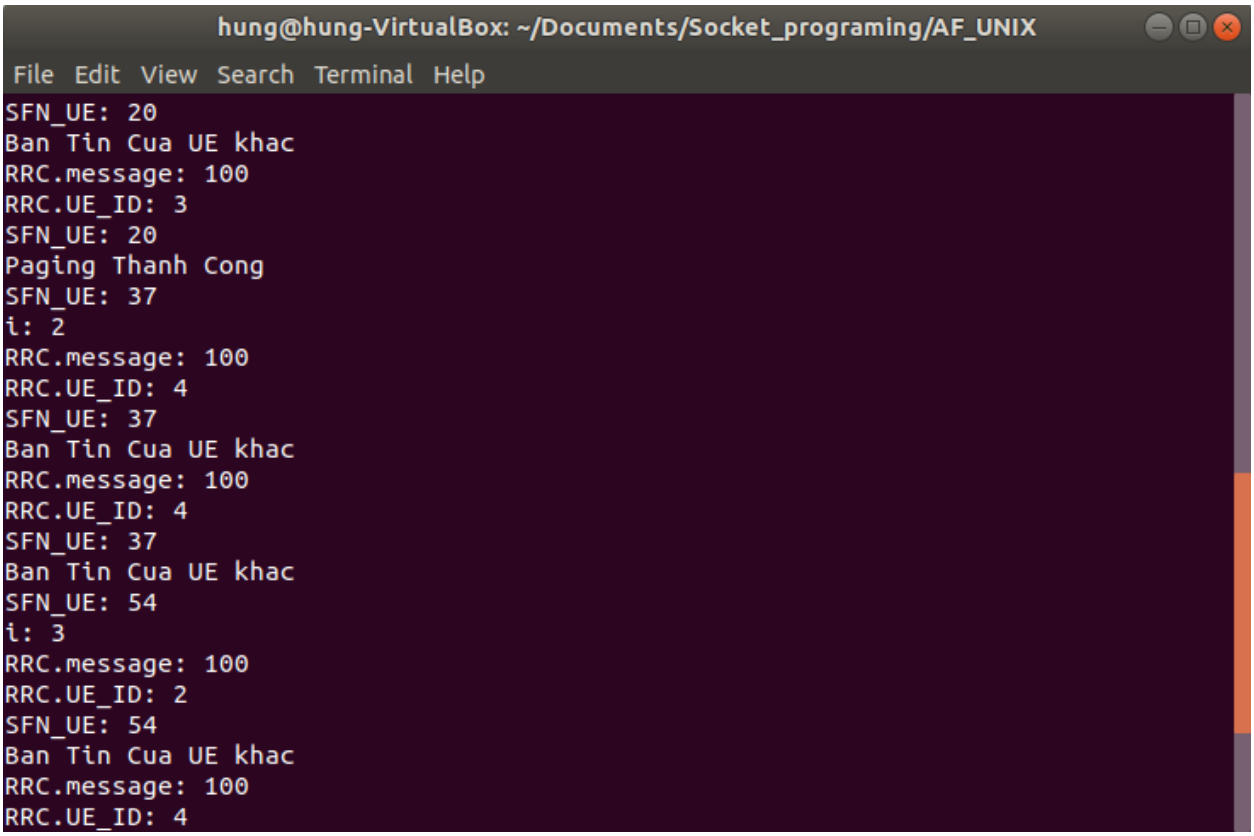
Cửa sổ của AMF.c



// AMF tạo mới và liên tục đẩy bản tin Paging tới gNB

// nhận phản hồi từ gNB

Cửa sổ của UE.c

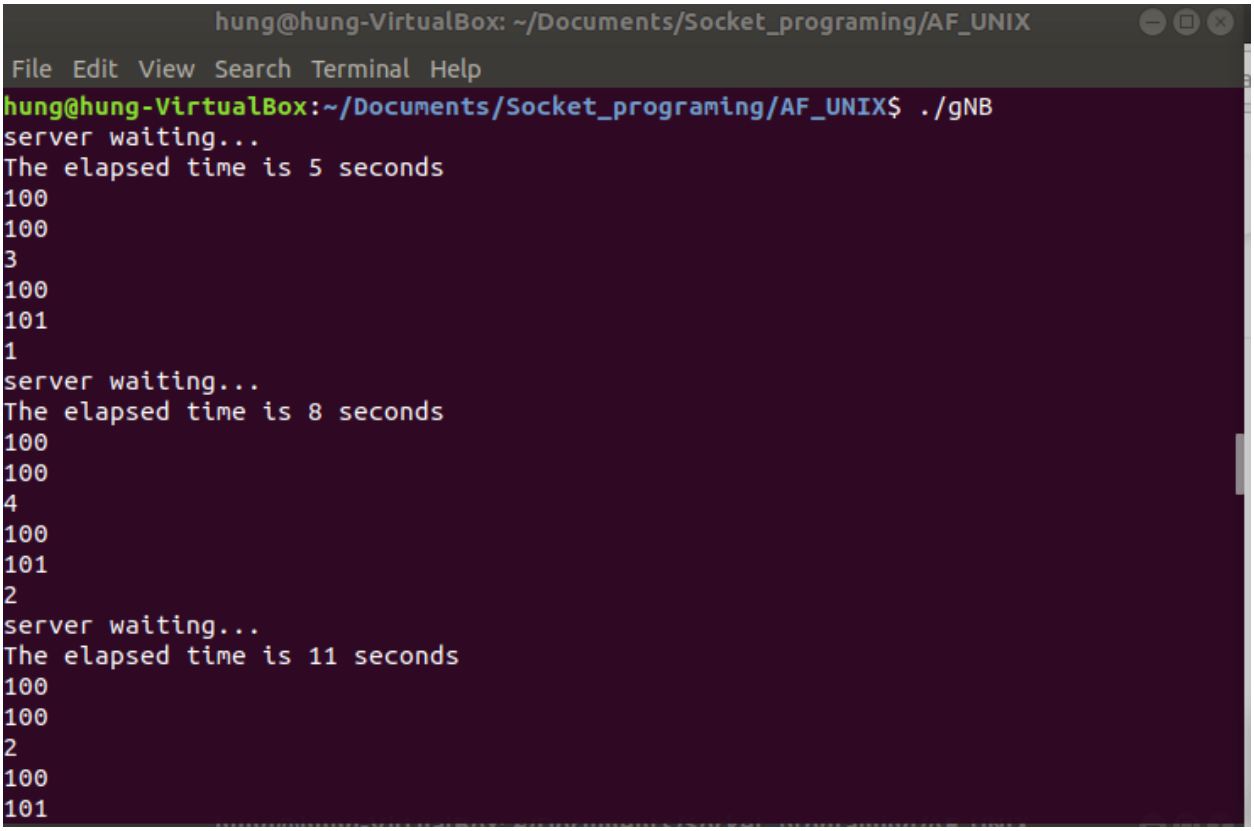


// setup UE_ID = 3, AMF gửi bản tin với trường UE_ID ngẫu nhiên từ hàm Get_Random(2,4)

// giá trị i ở đây là số các bản tin trong bộ nhớ của gNB tính tại thời điểm UE thức dậy

// khi UE thức dậy gNB sẽ đẩy toàn bộ các gói trong bộ nhớ cho UE xử lý và setup lại giá trị i (i = 0)

Cửa sổ của gNB.c



// cài đặt 1 bộ đếm thời gian thực tại UE cập nhật mỗi khi có kết nối thành công (sau khi chạy qua hàm accept)

// ở đây có thể thấy AMF đang gửi bản tin cho gNB với chu kỳ 3s

// thời điểm nhận được bản tin đầu tiên từ AMF là sau khi gNB được khởi động 5s