LỜI CẢM ƠN

Trước hết, chúng em xin gửi tới toàn thể các thầy, cô giáo trong Khoa Viễn Thông I, Học viện Công Nghệ Bưu Chính Viễn Thông lời cảm ơn chân thành nhất vì đã tạo điều kiện cho chúng em tham gia khóa học "Phát triển giao thức 4G/5G" của VHT.

Đặc biệt, chúng em xin bày tỏ lòng biết ơn sâu sắc tới anh chị chuyên gia trong VHT, người đã tận tâm giảng dạy và hướng dẫn chúng em trong suốt khóa học và hoàn thiện project.

Mặc dù bản thân đã rất cố gắng nhưng do thời gian, kiến thức và kinh nghiệm còn nhiều hạn chế, và cũng là lần đầu tiên được tiếp xúc các công nghệ mới nên bài làm của chúng em còn có nhiều thiếu sót trong việc trình bày, đánh giá và đề xuất ý kiến. Nhóm em rất mong nhận được sự góp ý từ phía anh chị và công ty để chúng em có thể rút kinh nghiệm và phát triển bản thân tốt hơn.

Em xin chân thành cảm ơn.

Sinh viên thực hiện

Nguyễn Thành Linh

Nguyễn Đức Kiên

Đặng Anh Lâm

1. Yêu cầu

- Xây dựng mô phỏng hệ thống 5G NR Paging dựa trên mô hình Client-Server.
- Có 3 tiến trình giao tiếp giữa AMF, gNodeB và UEs qua TCP socket:
  + AMF đóng vai trò là client gửi bản tin gồm 4 trường tới gNodeB.
  + gNodeB kiểm tra và lọc ra gói tin mong muốn rồi gửi xuống UE (client) theo đúng thời gian thức dậy của UE theo quy định.

2. Thiết kế hệ thống

➢ **Lưu ý :**

- Trong mô hình đề xuất, AMF và UEs là các clients và gNodeB là server. Do vậy, nhóm xem xét phần lập trình clients (AMF, Ues) thành 2 phần trong cùng file ***Client.c*** . Trong đó :
  + Cả AMF và UEs đều cần thiết lập kết nối tới gNodeB (server) qua TCP socket.
  + Đầu tiên, AMF gửi các bản tin tới gNodeB.
  + Sau đó, các UEs sẽ thực hiện gửi các bản tin thônng báo thức dậy tới gNodeB và thông báo khi nhận bản tin Taging thành công từ phía gNodeB.

- ***Source code của toàn project:***

https://github.com/linhnt31/VHT-Course-4-5G/tree/master/5G-Taging-Project/Source-code

a, Phía Server

- **Tạo socket với hàm**

```
socket (int family, int type, int protocol);
```

Trong đó, chúng ta thiết lập hệ thống sử dụng giao thức TCP nên họ giao thức sử dụng là: AF_INET; kiểu socket là SOCK_STREAM và kiểu giao thức để 0.

```
if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0){
    perror("socket failed");
    exit(EXIT_FAILURE);
}
```

- **Gán địa chỉ cho socket với hàm**

```
bind (int sockfd, const struct sockaddr *sockaddr, socklen_t addrlen);
```

```
if(setsockopt(server_fd,SOL_SOCKET,SO_REUSEADDR|SO_REUSEPORT, &opt, sizeof(opt))){
        perror("setsockopt");
        exit(EXIT_FAILURE);
}

address.sin_family = AF_INET;
address.sin_addr.s_addr = INADDR_ANY;
address.sin_port = htons( PORT );
if (bind(server_fd, (struct sockaddr *)&address, sizeof(address)){
    perror("bind failed");
    exit(EXIT_FAILURE);
}
```

- **Chỉ định socket lắng nghe kết nối :**

```
listen (int sockfd, int backlog);

if (listen(server_fd, 50) < 0){
    perror("listen");
    exit(EXIT_FAILURE);
}
```

Trong đó, soket vừa tạo là sockfd và số lượng tối đa kết nối backlog = 50.

- **Chờ/chấp nhận kết nối:**

```
accept (int sockfd, struct sockaddr *cliaddr, socklen_t *addrlen);

if((new_socket=accept(server_fd,(structsockaddr*)&address,(socklen_t*)&addrlen)) {
    perror("accept");
    exit(EXIT_FAILURE);
}
```

- **Quá trình nhận bản tin từ AMF**

Giả sử gNodeB nhận 5 bản tin, mỗi bản tin có các trường đều là trường đúng (với giá trị 100). Ngoại trừ, trường UE_ID được ***random*** nên ta sẽ tạo một hàng đợi để lưu và để chuyển tiếp bản tin đó đến đúng thời điểm UE thức dậy thì mới gửi trường đó tới UE:

```
// Số lượng gói tin nhận được từ AMF
int numberOfPackets = 5;

// Hàng đợi queue lưu các bản tin đến từ AMF
char queue[5][15];
int i = 0;
while(numberOfPackets--){
        printf("Message Type field from client: \n");
        valread = read( new_socket, buffer1, 1024);
        hello = &buffer1;
        printf("%s\n",buffer1 );
        send(new_socket, hello, strlen(hello), 0 );

        printf("UE ID from client: \n");
        valread = read( new_socket, buffer2, 1024);
        hello = &buffer2;
        printf("%s\n",buffer2 );
        send(new_socket, hello, strlen(hello), 0 );
        // Đẩy giá trị UE_ID tới hàng đợi queue
        strcpy(queue[i], buffer2);
        i++;
        printf("%s", queue[0]);

        printf("TAC from client: \n");
        valread = read( new_socket, buffer3, 1024);
        hello = &buffer3;
        printf("%s\n",buffer3 );
        send(new_socket, hello, strlen(hello), 0 );

        printf("CN Domain from client: \n");
        valread = read( new_socket, buffer4, 1024);
        hello = &buffer4;
        printf("%s\n",buffer4 );
        send(new_socket, hello, strlen(hello), 0 );
}
```

- **Quá trình kiểm tra bản tin đã nhận và thời điểm thức dậy để gửi bản tin đúng cho UE**

  Tiến hành kiểm tra các trường Message Type, TAC và CN Domain trong mỗi bản tin có phải trường đúng hay không (giá trị của trường 100). Nếu là trường đúng thì tiến hành kiểm tra xem thời điểm đó UE đã thức dậy hay chưa.

UE được kiểm tra thời điểm thức dậy bằng cách vào đúng thời điểm thức dậy của nó, client sẽ gửi cho server bản tin ***WakeUp*** với giá trị ***"True"*** báo hiệu rằng UE đã thức dậy, do đó khi server nhận được bản tin ***WakeUp*** với giá trị ***"True"*** sẽ lập tức gửi bản tin chứa trường UE_ID đang nằm trong ***hàng đợi queue*** sẵn ở bước trên cho Ues (clients)

```
int j = 0;
int check = 1;
while(check && j < 5){
        printf("Tin nhan UEs thuc day: \n");
        valread = read( new_socket, buffer5, 1024);
        hello = &buffer5;
        printf("%s\n",buffer5 );

        if(strcmp(buffer1, "100") == 0 && strcmp(buffer3, "100") == 0
&& strcmp(buffer4, "100") == 0 && strcmp(buffer5, "True") == 0){
            send(new_socket, &queue[j], strlen((&queue[j])), 0 );
            j++;
        }
}
```

b, Phía Client (AMF và UE)

- **Tạo socket với hàm:**

```
socket (int family, int type, int protocol);

if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
    printf("\n Socket creation error \n");
    return -1;
}
memset(&serv_addr, '0', sizeof(serv_addr));
serv_addr.sin_family = AF_INET;
serv_addr.sin_port = htons(PORT);
```

- **Connect tới địa chỉ server với hàm:**

```
connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr));

if (connect(sock, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0){
    printf("\nConnection Failed \n");
    return -1;
```

```
}
```

- **Tạo và gửi các bản tin random từ AMF :**

  Ở đây chỉ random trường UE_ID, mặc định các trường còn lại là trường đúng với giá trị là 100).

```
// AMF packets
struct Packets {
   char *messageType, *TAC, *CN_Domain, *WakeUp; // “100 default”
   int UE_ID;
};

// In main func
int numberOfPackets = 5;
while(numberOfPackets--){
        // Initilize fields
        p.messageType = "100";
        char *str[1000];
        int lower = 99, upper = 1000;
        p.UE_ID = rand() % (upper - lower + 1) + lower;
        sprintf(str, "%d", p.UE_ID);
        p.TAC = "100";
        p.CN_Domain = "100";

        /* AMF */
        send(sock , p.messageType , strlen(p.messageType) , 0 );
        valread = read( sock , buffer1, 1024);
        send(sock , str , strlen(str) , 0 );
        valread = read( sock , buffer2, 1024);
        send(sock , p.TAC , strlen(p.TAC) , 0 );
        valread = read( sock , buffer3, 1024);
        send(sock , p.CN_Domain , strlen(p.CN_Domain) , 0 );
        valread = read( sock , buffer4, 1024);
}
```

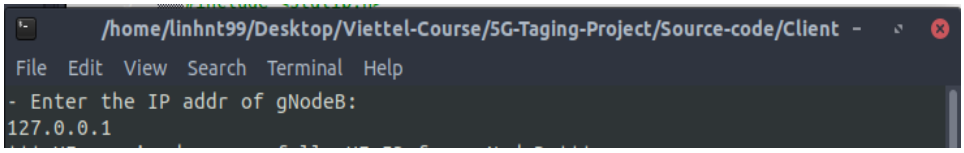
- **Quá trình gửi bản tin tới gNodeB từ UE để thông báo rằng UE đã thức dậy và có thể nhận bản tin Paging từ gNodeB**

  Giả sử thời gian bắt đầu được đếm từ t=0, sử dụng vòng while để đếm và Sleep(1) để sau mỗi 1 giây thì t sẽ tăng lên một đơn vị. Khi vào đúng thời điểm thức dậy, thì client sẽ gửi ***bản tin WakeUP với giá trị “True”*** cho server

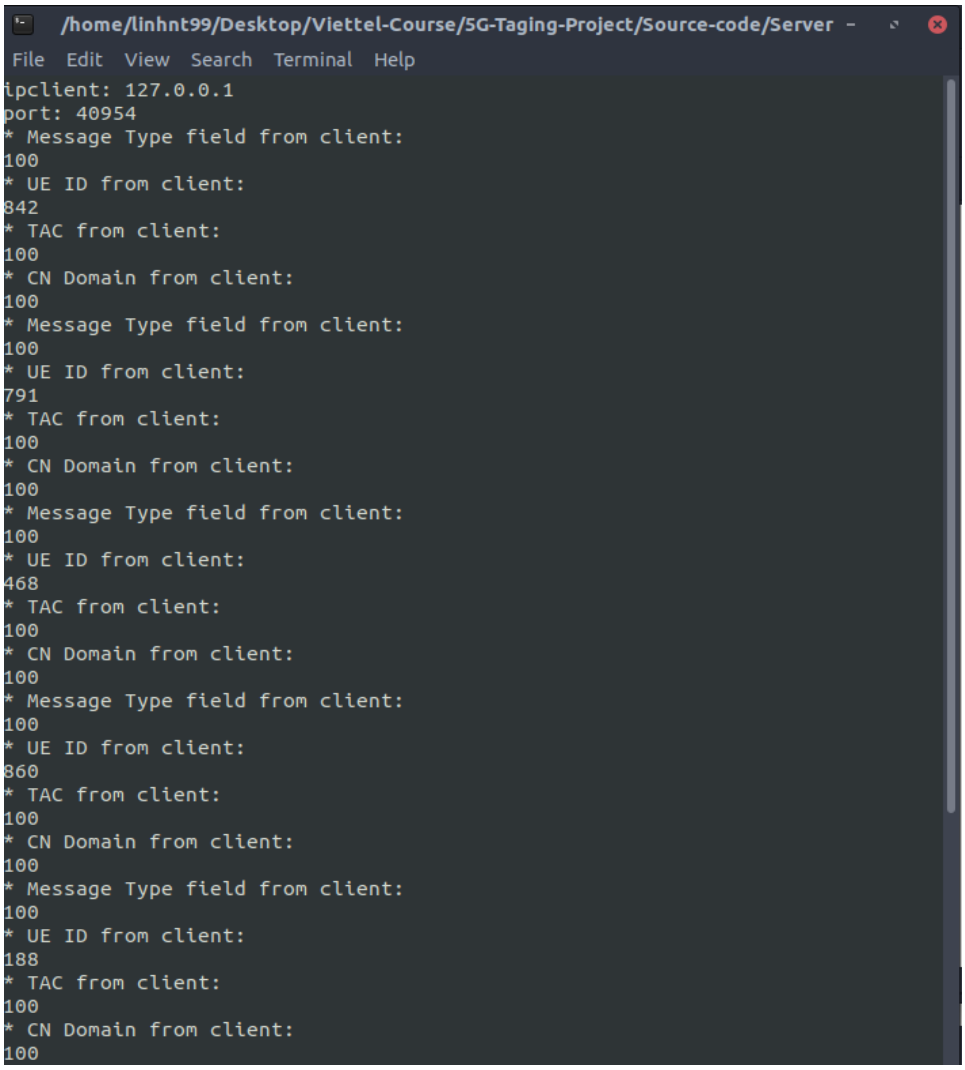
để báo hiệu UE đã thức dậy. Server nhận được bản tin báo hiệu đó liền gửi bản tin Paging về cho UE. UE đọc và in ra bản tin vừa nhận thành công.

```
int t = 0;
p.WakeUp = "True";
while(1){
  sleep(1);
  t += 1;
   if(t % 64 == 1){
      send(sock , p.WakeUp , strlen(p.WakeUp ) , 0 );
      printf("UE received successfully UE-ID from gNodeB: \n");
      valread = read( sock , buffer5, 1024);
      printf("%s\n",buffer5);
   }
}
```

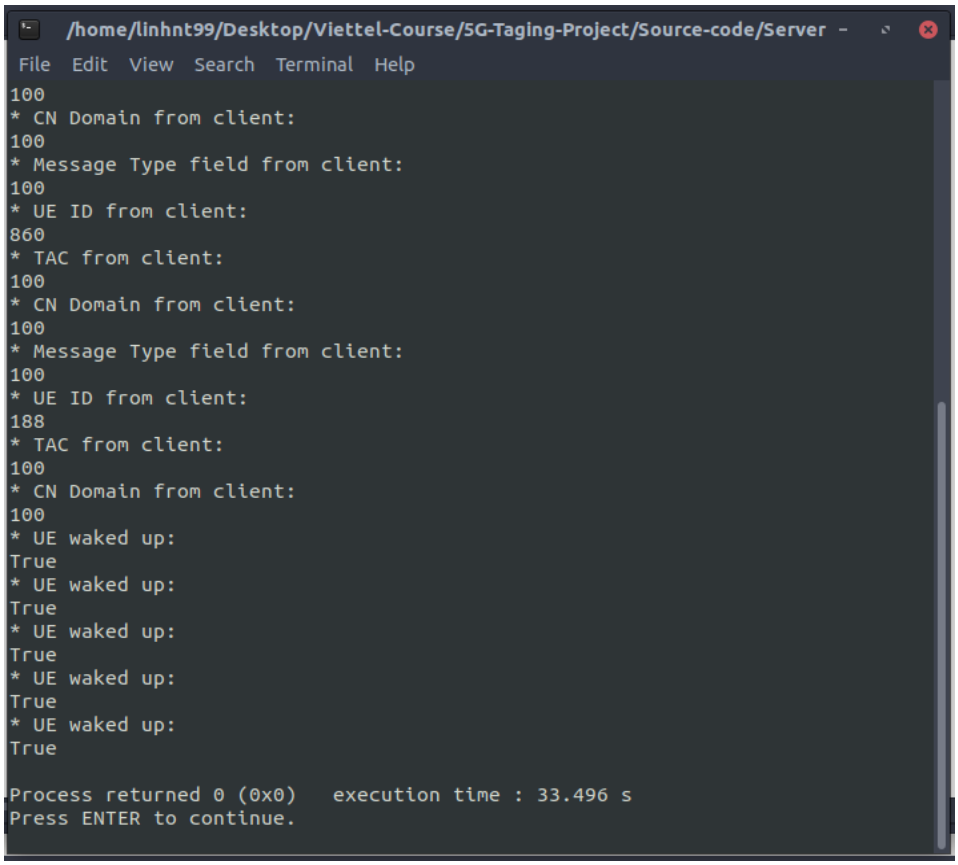
3. Kết quả mô phỏng



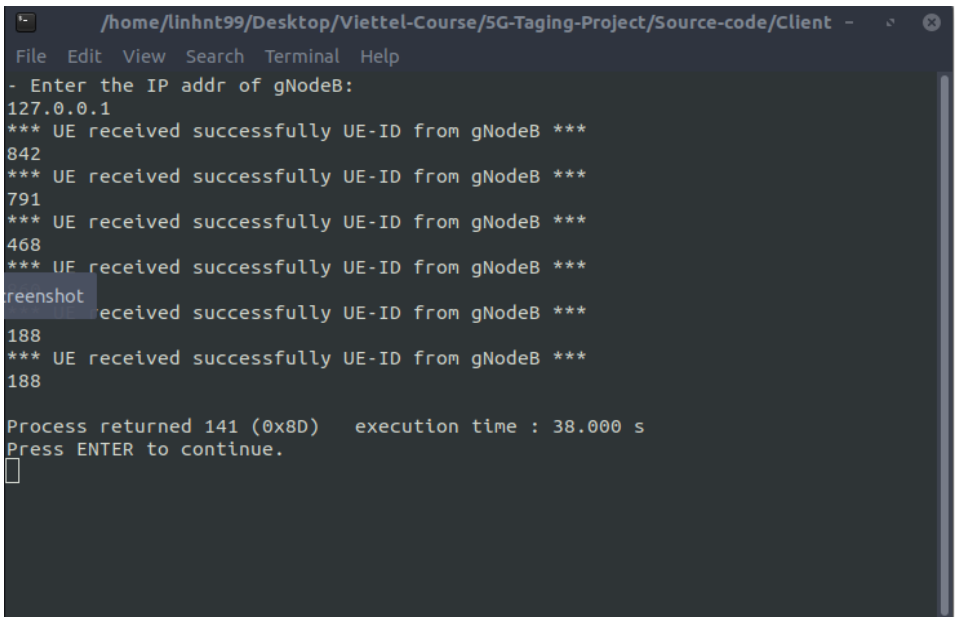
Hình 1. Kết nối AMF tới gNodeB



Hình 2. gNodeB kết nối thành công với AMF và nhận các bản tin từ phía AMF



Hình 3. gNodeB nhận được bản tin WakeUp từ phía UEs



Hình 4. UEs nhận thành công bản tin Paging từ gNodeB