Research & Development the Protocol in 4G/5G

Practice on C programming

tuandn3@viettel.com.vn

Agenda

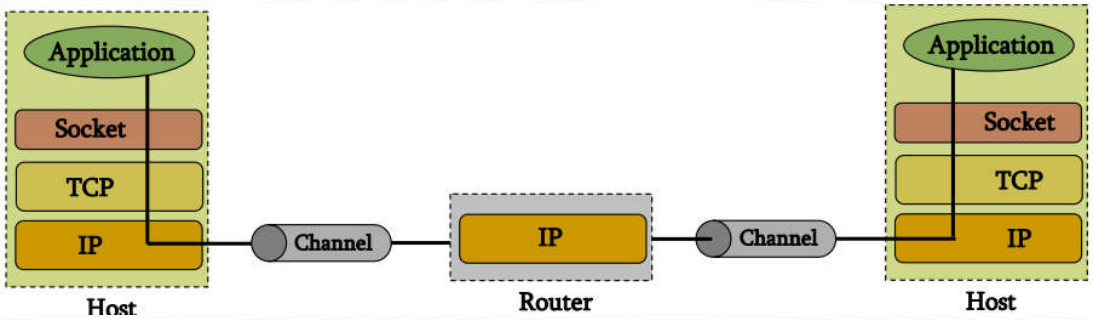
1. TCP/IP socket programming
   - Review
   - Other
2. TCP server
   - APIs Flow
   - Code & running
3. TCP client
   - APIs Flow
   - Code & running
4. Project 3 – 5G NR Paging by 3GPP
   - introduction
   - ref
5. Verify Project 3 – 5G NR Paging simulation
   - Requirement & system design (simple)
   - Code & Running
   - Verify: same team, different team,
6. Report Project 3
   - Requirement

1. Socket programming with TCP/IP

Socket

Introduction

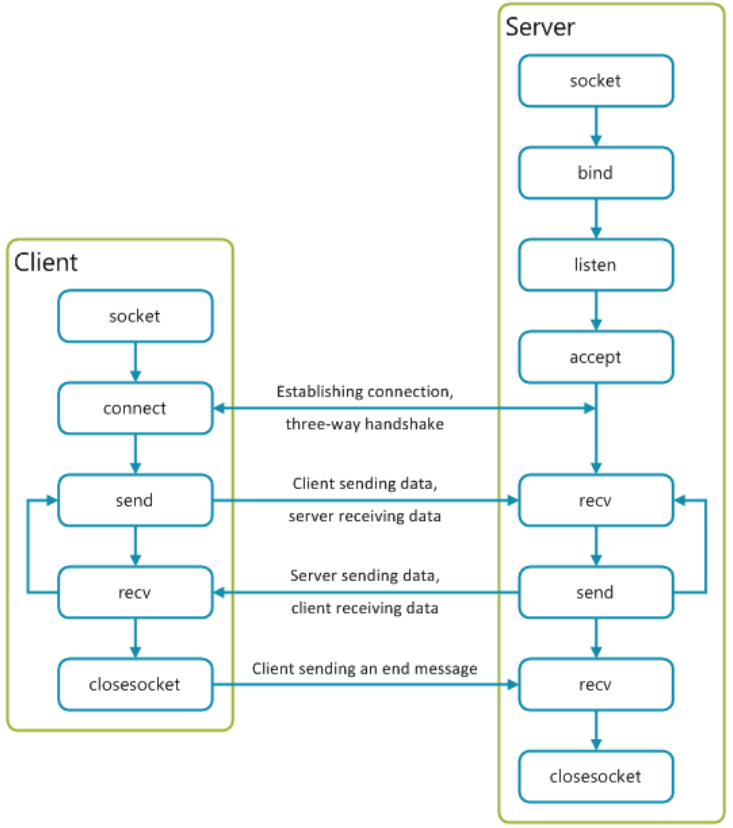
- Socket = IP address + Port
- Socket API or Berkeley socket was mostly written by University of California, Berkeley
- Socket API has used in many OS
- Application call this socket API to do the inter-process communication (same host or different node)
- Socket API provide some types of protocol sockets as TCP socket, UDP socket, SCTP socket



TCP Client – Server

Client-Server communication

- **Server**
  - passively waits for and responds to clients
  - **passive** socket
- **Client**
  - initiates the communication
  - must know the address and the port of the server
  - **active** socket



Flow Diagram for BSD Sockets Communication using TCP

Tutorial by Eleftherios Kosmas

socket

```
#include <sys/socket.h>

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

/* Returns: non-negative descriptor if OK, -1 on error */
```

- *socket()*: system call creates a new socket

- Family or Domain
  - AF_UNIX (same host)
  - AF_INET (via IPv4)
  - AF_INET6 (via IPv6)

- protocol
  - TCP use with SOCK_STREAM
  - UDP use with SOCK_DGRAM
  - SCTP

- Family or Domain
  - SOCK_STREAM : reliable, bidirectional, byte-stream
  - SOCK_DGRAM: unreliable, connectionless,

connect

```
#include <sys/socket.h>

int connect(int sockfd, const struct sockaddr *servaddr, socklen_t addrlen);

/* Returns: 0 if OK, -1 on error */
```

- Bind
  - The *sockfd*: file descriptor obtain from a previous call to *socket()*
  - The *myaddr*: a pointer to a structure specifying the address
  - The *addrlen*: specifies the size of the address structure.

- *socket()*: system call establishes a connection with another socket

bind

```
#include <sys/socket.h>

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

/* Returns: 0 if OK,-1 on error */
```

- *Bind()*: system call binds a socket to an address.

- Bind
  - The *sockfd*: file descriptor obtain from a previous call to *socket()*
  - The *myaddr*: a pointer to a structure specifying the address
  - The *addrlen*: specifies the size of the address structure.

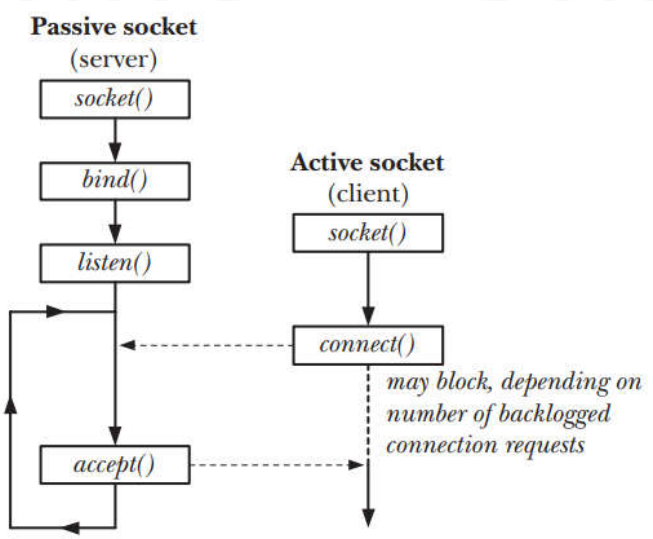
```
struct sockaddr {
    sa_family_t sa_family;          /* Address family (AF_* constant) */
    char        sa_data[14];        /* Socket address (size varies
                                       according to socket domain) */
};
```

listen

```
#include <sys/types.h>          /* See NOTES */
#include <sys/socket.h>

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

- listen
  - Return 0 on success, or -1 on error
  - The *sockfd*: file descriptor obtain from a previous call to *socket()*
  - The *backlog*: argument allow us to limit the number of such pending connection

- *listen()*: system call allow a stream socket to accept incoming connections from other sockets.



accept

```
#include <sys/socket.h>

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

/* Returns: non-negative descriptor if OK, -1 on error */
```

- *accept()*: system call accepts a connection from a peer application on a listening stream socket, and optionally returns the address of the peer socket.

- Bind
  - The *sockfd*: file descriptor obtain from a previous call to *socket()*
  - The *cliaddr*: a pointer to a structure specifying the client address
  - The *addrlen*: specifies the size of the address structure.

write or send

```
#include <unistd.h>

ssize_t write(int fd, const void *buf, size_t count);
```

- sockfd
- sockaddr
- addrlen

read or recv

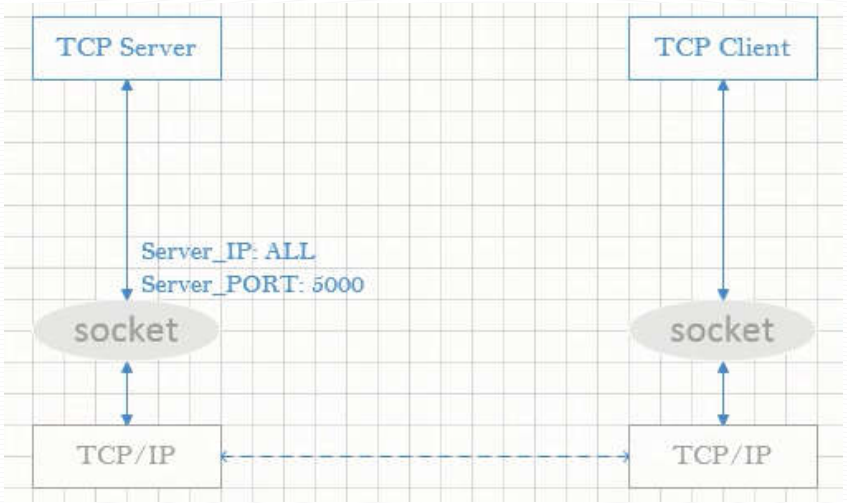
```
#include <unistd.h>

ssize_t read(int fd, void *buf, size_t count);
```

- fd
- buf
- count

2. TCP Server

TCPServer

- Steps:
  - Step 1: Open a socket
  - Step 2: Bind to a address and port
  - Step 3: Listen for incoming connections
  - Step 4: Accept connections
  - Step 5: Read/send



Step 1-3: socket, bind & listen

- AF_INET:
- SOCK_STREAM:
- 0 : TCP
- htonl
- INADDR_ANY
- htons

https://www.thegeekstuff.com/2011/12/c-socket-programming/

```
int main(int argc, char *argv[])
{
    int listenfd = 0, connfd = 0;
    struct sockaddr_in serv_addr;

    char sendBuff[1025];
    time_t ticks;

    listenfd = socket(AF_INET, SOCK_STREAM, 0);
    memset(&serv_addr, '0', sizeof(serv_addr));
    memset(sendBuff, '0', sizeof(sendBuff));

    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    serv_addr.sin_port = htons(5000);

    bind(listenfd, (struct sockaddr*)&serv_addr, sizeof(serv_addr));

    listen(listenfd, 10);
```

Step 4-5: accept & write

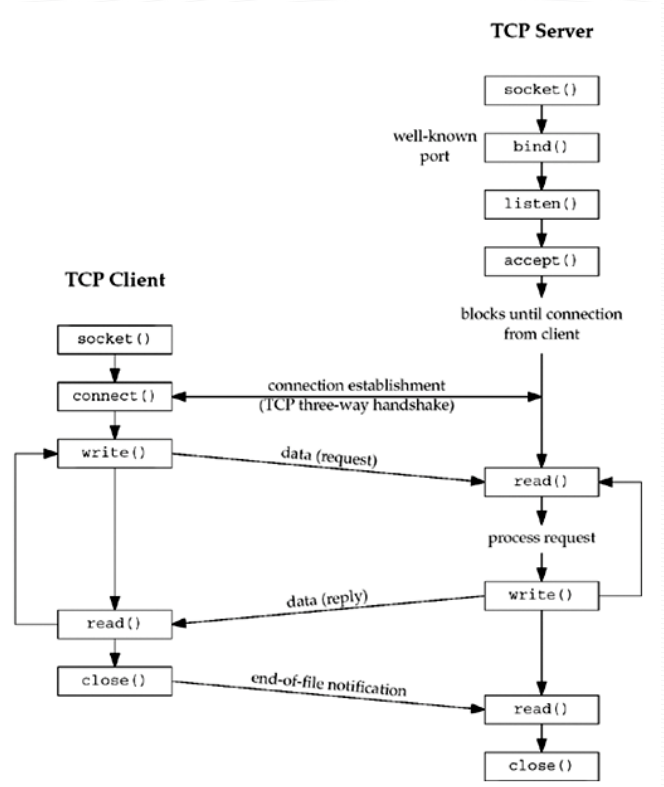
- connfd:
- sendBuff:
- close

```
    while(1)
    {
        connfd = accept(listenfd, (struct sockaddr*)NULL, NULL);

        ticks = time(NULL);
        snprintf(sendBuff, sizeof(sendBuff), "%.24s\r\n", ctime(&ticks));
        write(connfd, sendBuff, strlen(sendBuff));

        close(connfd);
        sleep(1);
     }
}
```

3. TCP Client

TCP Client: Flow

- Steps:
  - Step 1: Open a socket
  - Step 2: Input TCP Server address
  - Step 3: Connect to TCP server
  - Step 4: Read/send



Step 1: socket

- struct sockaddr_in:
- AF_INET
- SOCK_STREAM:
- 0 : TCP

https://www.thegeekstuff.com/2011/12/c-socket-programming/

```
int main(int argc, char *argv[])
{
    int sockfd = 0, n = 0;
    char recvBuff[1024];
    struct sockaddr_in serv_addr;

    if(argc != 2)
    {
        printf("\n Usage: %s <ip of server> \n",argv[0]);
        return 1;
    }

    memset(recvBuff, '0',sizeof(recvBuff));
    if((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
    {
        printf("\n Error : Could not create socket \n");
        return 1;
    }
```

Step 2-3: address & connect

- memset:
- AF_INET:
- htons
- Inet_pton
- sizeof

```
memset(&serv_addr, '0', sizeof(serv_addr));

serv_addr.sin_family = AF_INET;
serv_addr.sin_port = htons(5000);

if(inet_pton(AF_INET, argv[1], &serv_addr.sin_addr)<=0)
{
    printf("\n inet_pton error occured\n");
    return 1;
}

if( connect(sockfd, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0)
{
   printf("\n Error : Connect Failed \n");
   return 1;
}
```

https://www.thegeekstuff.com/2011/12/c-socket-programming/

Step 4: address & connect

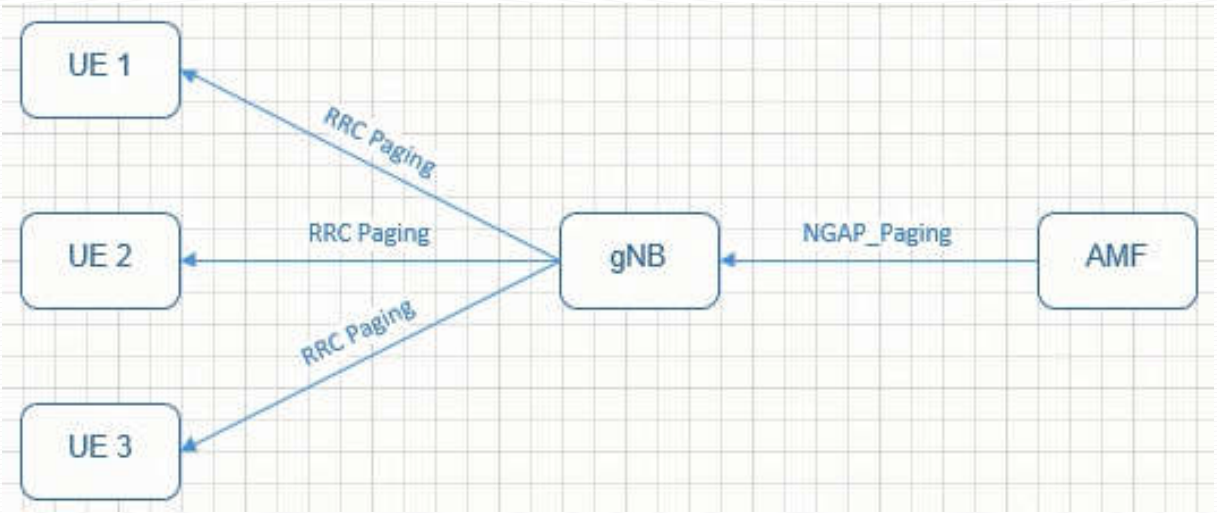
- read:
- fputs:
- recevBuff

```
    while ( (n = read(sockfd, recvBuff, sizeof(recvBuff)-1)) > 0)
    {
        recvBuff[n] = 0;
        if(fputs(recvBuff, stdout) == EOF)
        {
            printf("\n Error : Fputs error\n");
        }
    }

    if(n < 0)
    {
        printf("\n Read error \n");
    }

    return 0;
}
```

https://www.thegeekstuff.com/2011/12/c-socket-programming/

4. Project 3 – 5G NR Paging by 3GPP

Pre-study



- Process
  - AMF
  - gNB
  - UE
- IPC
  - TCP socket
- 3GPP Paging procedure
  - UE Paging procedure
  - RRC Paging procedure
  - NGAP paging procedure

5G NR Paging simulation

- Mission: Simulate the 5G NR Paging system
- Practices:
  - Task 1: gather requirement
  - Task 2: build the specs
  - Task 3: make the System Paging x
  - Task 4: make the Architecture Design
  - Task 5: make the Detailed Design
  - Task 6: Implementation & Verification

3GPP reference

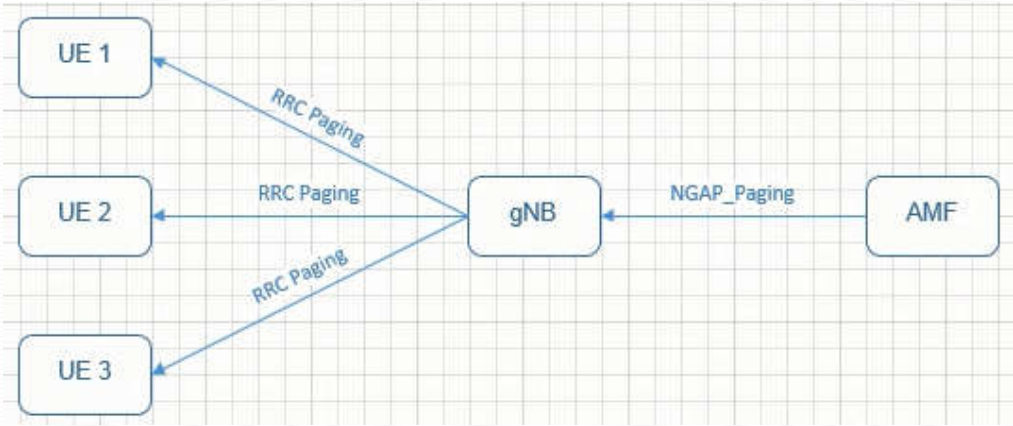
- https://www.3gpp.org/ftp/Specs/archive/38_series (release 15)
- 3GPP TS 36.413: "Evolved Universal Terrestrial Radio Access Network (E-UTRAN); S1 Application Protocol (S1AP)".
- 3GPP TS 38.304: "NR; User Equipment (UE) procedures in idle mode and in RRC inactive state".
- 3GPP TS 38.331: "NG-RAN; Radio Resource Control (RRC) Protocol Specification".
- 3GPP TS 38.401: "NG-RAN; Architecture description".
- 3GPP TS 38.410: "NG-RAN; NG general aspects and principles".

5. Verify Project 3 by Simulation

Project 3

Điều kiện/ Thông tin cho trước

- SFN bộ đếm thời gian tại UE và gNB từ [0 -1023]; 1 giây tăng 1 đơn vị
- UE thức dậy ở thời điểm SFN = m, m+ 64 * k (k = 0, 1, 2 ,..); m = 1
- Có 3 tiến trình: UE, gNB, AMF giao tiếp qua TCP socket
- gNB đóng vai trò TCP server.
- UE/AMF đóng vai trò TCP client.
- Bản tin NGAP_Paging và RRC Paging có 4 trường



Project 3

Định nghĩa bản tin giữa: UE, AMF, gNB

NgAP Paging message	
♣ Trường 1: Message_type	[4 byte]
♣ Trường 2: UE_ID	[4 byte]
♣ Trường 3: TAC	[4 byte]
♣ Trường 4: CN_Domain	[4 byte]

RRC Paging message	
♣ Trường 1: Message_type	[4 byte]
♣ Trường 2: UE_ID	[4 byte]
♣ Trường 3: TAC	[4 byte]
♣ Trường 4: CN_Domain	[4 byte]

Thông tin quy ước đúng

- ♣ Message_type giá trị 100 là bản tin Paging
- ♣ UE_ID giá trị 100 là bản tin Paging cho đúng UE tìm gọi
- ♣ TAC giá trị 100 là TAC đúng, Tracking Area cần Paging bản tin
- ♣ CN_Domain

  Giá trị 100 là Paging cho tìm gọi thoại // gọi điện thoại bình thường

  Giả tị 101 là Paging cho tìm gọi Data // gọi điện thoại qua app như Viber, Zalo

AMF & gNB

- B1: AMF gán giá trị đúng cho 4 trường trong bản tin NgAP_Paging và gửi cho gNB
  - Message_type = 100
  - UE_ID = 100
  - TAC = 100
  - CN_Domain = 100

- B2: gNB nhận được bản tin từ AMF; kiểm tra message_type
  - IF: Message_type == 100 sang B3;

- B3: gNB kiểm tra TAC
  - IF: TAC = 100 sang B4;

- B4: gNB kiểm tra CN_Domain
  - IF CN_Domain == 100 hoặc 101 sang B5

gNB

- B5: gNB xử lý lưu bản tin NgAP Paging vào Queue
  - ♣ Kiểm tra SFN_gNB hiện tại của gNB
  - ♣ So sánh khoảng thức dậy của UE: SFN_UE = m, m+ 64 * k (k = 0, 1, 2 ,,,); m = 1
  - ♣ Lưu và bộ nhớ queue_number;

    Ví dụ: UE thức dậy ở 1, 65, 129, 193, 257 …

    Nếu: SFN_gNB hiện tại của gNB là 20; lưu vào Queue_65

    Nếu: SFN_gNB hiện tại của gNB là 100; lưu vào Queue_129

    Nếu: SFN_gNB hiện tịa của gNB là 200; lưu và Queue_257

- B6; gNB kiểm tra thời điểm gửi Paging cho UE;
  - ♣ 1s SFN_gNB tăng 1 đơn vị
  - ♣ Kiểm tra SFN_gNB = queue_number, chuyển sang B7

gNB & UE

- B7; gNB gửi bản tin Paging cho UE;
  - ♣ Gán giá trị RRC Paging từ NgAP Paging trong queue_number
  - ♣ Gửi xuống UE

- B8; UE nhận được và kiểm tra
  - ♣ Message_type == 100;
  - ♣ UE_ID == 100
  - ♣ Hiển thị thông báo UE nhận bản tin Paging thành công

Run & Verify

- V1; same team
  - ♣ Verify đúng
  - ♣ Verify giá trị truyền vào NGAP_Paging khác đề bài
- V2; khác team
  - ♣ Verify đúng
  - ♣ Verify giá trị truyền vào NGAP_Paging khác đề bài
- V3; Phát triển bài toán
  - ♣ AMF sinh tự động bản tin Paging với giá trị truyền vào random, định kỳ gửi xuống gNB
  - ♣ ….

6. Report

Project 3 – 5G NR paging simulation

- Project 3
- Report: pdf, 3-5 pages + source code
  - Part 1: requirement
  - Part 2: system design
  - Part 3: Code & Result
- Team: 3 people
- Tính điểm Report sớm cho nhóm sinh viên xuất sắc