計算機網路概論 Final Project

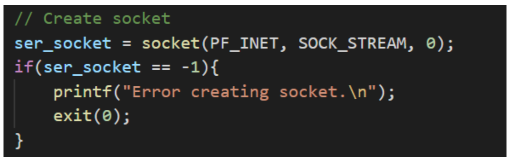
學號: 109062211　　　　　　　　　　　　姓名: 張惇媛

1　實作過程 (使用 Linux 寫法)

1.1　server 端

1.1.1　Create

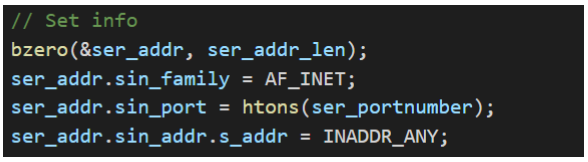
建立 TCP socket，若回傳值等於 -1，代表建立發生錯誤，印出 error report，並關閉程式。



```c
// Create socket
ser_socket = socket(PF_INET, SOCK_STREAM, 0);
if(ser_socket == -1){
    printf("Error creating socket.\n");
    exit(0);
}
```

1.1.2　Set

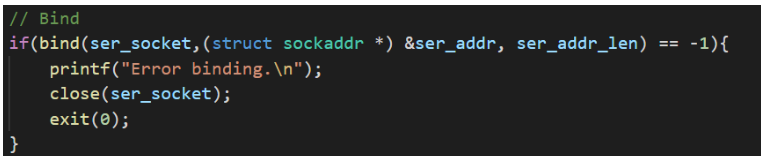
設定 socket 的信息。



```c
// Set info
bzero(&ser_addr, ser_addr_len);
ser_addr.sin_family = AF_INET;
ser_addr.sin_port = htons(ser_portnumber);
ser_addr.sin_addr.s_addr = INADDR_ANY;
```

1.1.3　Bind

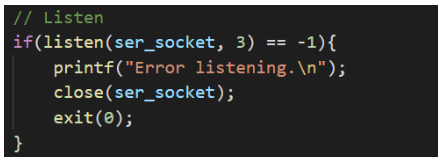
把設定的 address 綁在 socket 身上，若回傳值等於 -1，代表綁定發生錯誤，印出 error report，並關閉 TCP socket、程式。



```c
// Bind
if(bind(ser_socket,(struct sockaddr *) &ser_addr, ser_addr_len) == -1){
    printf("Error binding.\n");
    close(ser_socket);
    exit(0);
}
```

1.1.4　Listen

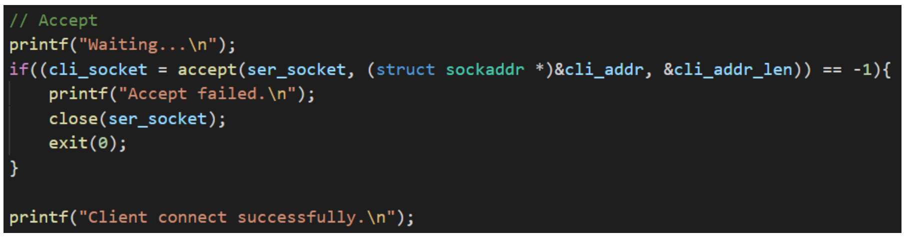
等待請求，若回傳值等於 -1，代表等待發生錯誤，印出 error report，並關閉 TCP socket、程式。



```c
// Listen
if(listen(ser_socket, 3) == -1){
    printf("Error listening.\n");
    close(ser_socket);
    exit(0);
}
```

1.1.5　Accpet

接受請求，若回傳值等於 -1，代表接受發生錯誤，印出 error report，並關閉 TCP socket、程式。成功，則代表 client 端已連線到 server 端，可以開始互相溝通。

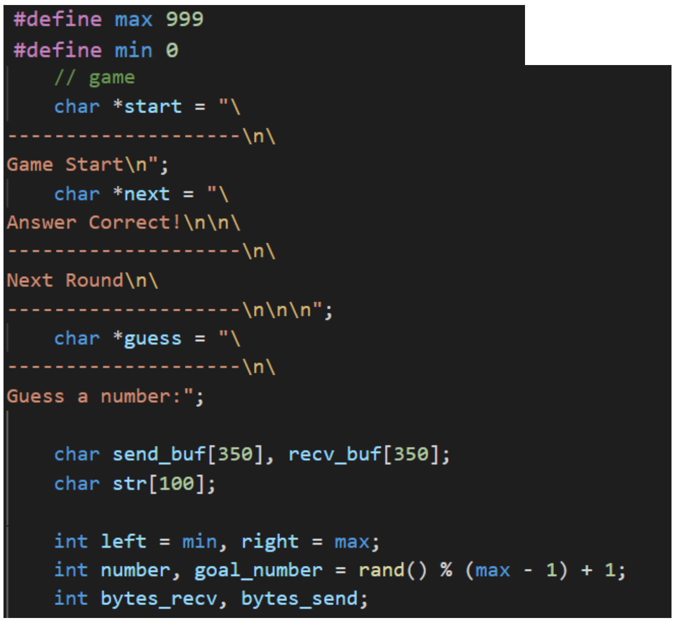


```c
// Accept
printf("Waiting...\n");
if((cli_socket = accept(ser_socket, (struct sockaddr *)&cli_addr, &cli_addr_len)) == -1){
    printf("Accept failed.\n");
    close(ser_socket);
    exit(0);
}

printf("Client connect successfully.\n");
```

1.1.6　Communicate

1.1.6.1　定義變數

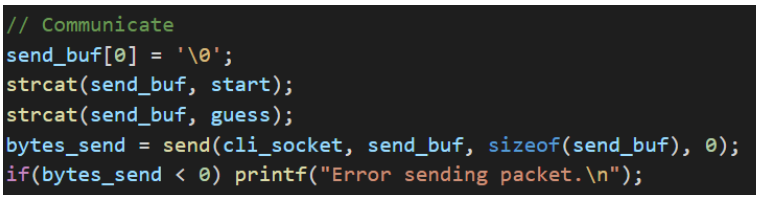
定義字串、整數以及產生一個亂數 goal_number 並初始 left & right 的數值。



```
#define max 999
#define min 0
    // game
    char *start = "\
--------------------\n\
Game Start\n";
    char *next = "\
Answer Correct!\n\n\
--------------------\n\
Next Round\n\
--------------------\n\n\n";
    char *guess = "\
--------------------\n\
Guess a number:";

    char send_buf[350], recv_buf[350];
    char str[100];

    int left = min, right = max;
    int number, goal_number = rand() % (max - 1) + 1;
    int bytes_recv, bytes_send;
```

1.1.6.2 Game Start

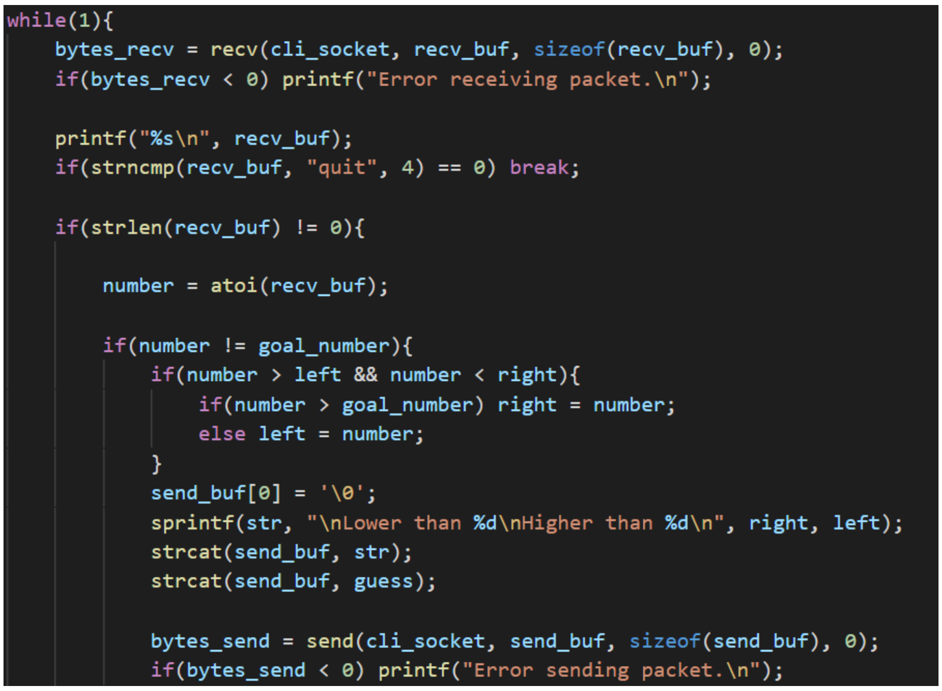
一開始啟動遊戲，傳送 start 和 guess 的訊息給 client 端，若傳送失敗，印出 error report。



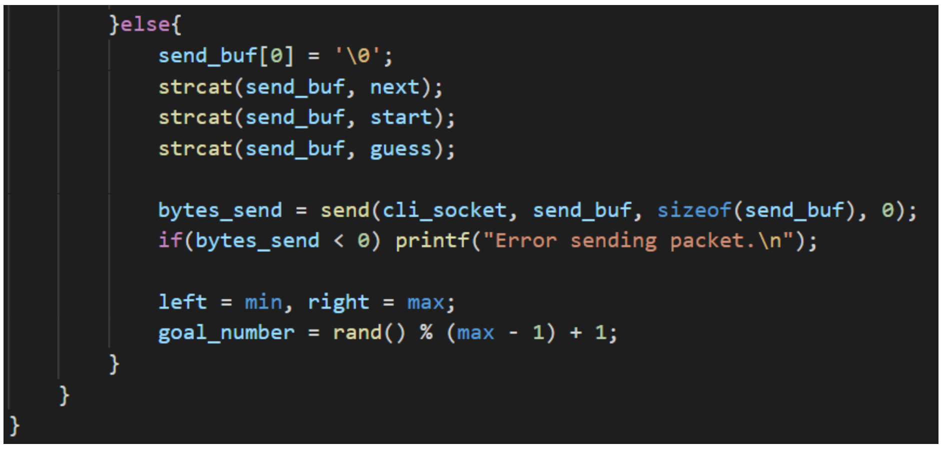
```
// Communicate
send_buf[0] = '\0';
strcat(send_buf, start);
strcat(send_buf, guess);
bytes_send = send(cli_socket, send_buf, sizeof(send_buf), 0);
if(bytes_send < 0) printf("Error sending packet.\n");
```

1.1.6.3 Recv & Send

建立迴圈，接收 client 端的訊息，並印出。若接收到"quit"，則跳出迴圈。接著，判斷 recv_buf 的長度，若不等於 0，則將 recv_buf 的訊息轉換成 number 數字，if number != goal_number，代表沒猜中數字，因此判斷邊界(left 和 right)的更新，再將 lower & higher 和 guess 的訊息傳送給 client 端；else 代表猜中數字，將 next、start 和 guess 的訊息傳送給 client 端，並將 left & right 初始，重新產生亂數，開啟下一輪遊戲。若傳送失敗，印出 error report。



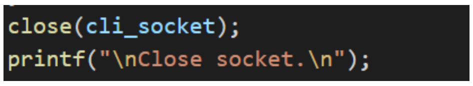
```
while(1){
    bytes_recv = recv(cli_socket, recv_buf, sizeof(recv_buf), 0);
    if(bytes_recv < 0) printf("Error receiving packet.\n");

    printf("%s\n", recv_buf);
    if(strncmp(recv_buf, "quit", 4) == 0) break;

    if(strlen(recv_buf) != 0){

        number = atoi(recv_buf);

        if(number != goal_number){
            if(number > left && number < right){
                if(number > goal_number) right = number;
                else left = number;
            }
            send_buf[0] = '\0';
            sprintf(str, "\nLower than %d\nHigher than %d\n", right, left);
            strcat(send_buf, str);
            strcat(send_buf, guess);

            bytes_send = send(cli_socket, send_buf, sizeof(send_buf), 0);
            if(bytes_send < 0) printf("Error sending packet.\n");
```



```
        }else{
            send_buf[0] = '\0';
            strcat(send_buf, next);
            strcat(send_buf, start);
            strcat(send_buf, guess);

            bytes_send = send(cli_socket, send_buf, sizeof(send_buf), 0);
            if(bytes_send < 0) printf("Error sending packet.\n");

            left = min, right = max;
            goal_number = rand() % (max - 1) + 1;
        }
    }
}
```

1.1.7 Close

關閉 cli_socket。

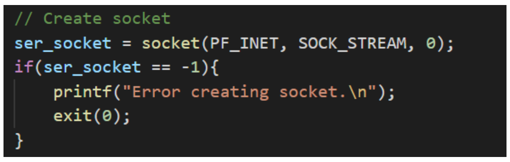


```
close(cli_socket);
printf("\nClose socket.\n");
```

1.2 client 端

1.2.1 Create

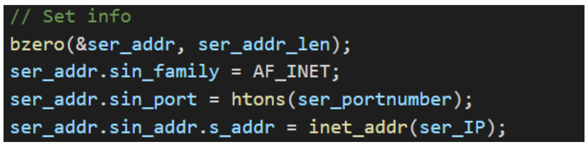
建立 TCP socket，若回傳值等於 -1，代表建立發生錯誤，印出 error report，並關閉程式。



```
// Create socket
ser_socket = socket(PF_INET, SOCK_STREAM, 0);
if(ser_socket == -1){
    printf("Error creating socket.\n");
    exit(0);
}
```

1.2.2 Set

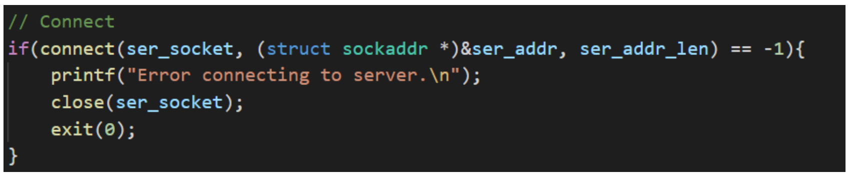
設定 socket 的信息。



```
// Set info
bzero(&ser_addr, ser_addr_len);
ser_addr.sin_family = AF_INET;
ser_addr.sin_port = htons(ser_portnumber);
ser_addr.sin_addr.s_addr = inet_addr(ser_IP);
```

1.2.3 Connect

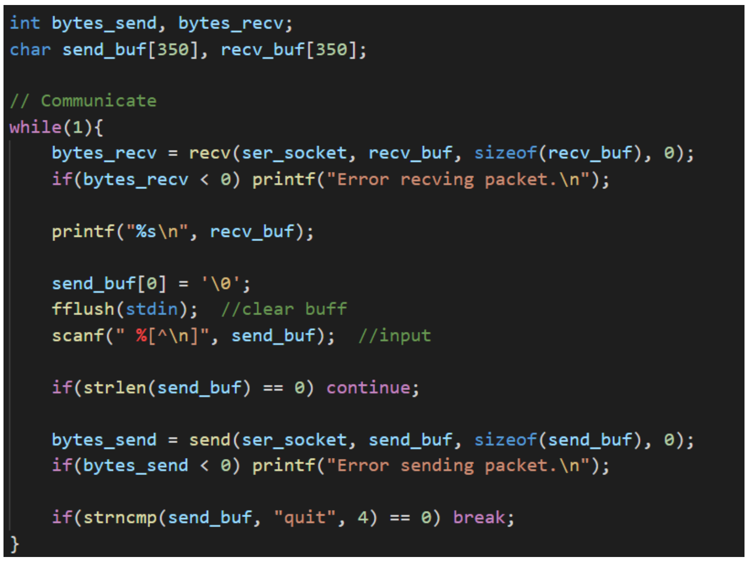
建立連線，若回傳值等於 -1，代表連線發生錯誤，印出 error report，並關閉 TCP socket、程式。



```
// Connect
if(connect(ser_socket, (struct sockaddr *)&ser_addr, ser_addr_len) == -1){
    printf("Error connecting to server.\n");
    close(ser_socket);
    exit(0);
}
```

1.2.4 Recv & Send

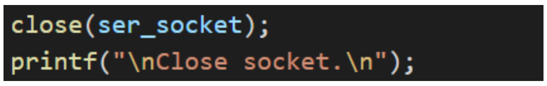
定義變數，接著，建立迴圈，接收 server 端的訊息，並印出。清空輸入緩衝區，為了確保不影響後面的資料讀取，讀取輸入的字串，if send_buf 的長度不等於 0，則將 send_buf 傳送給 server 端。若 send_buf 的訊息為 ”quit”，則跳出迴圈。



```c
int bytes_send, bytes_recv;
char send_buf[350], recv_buf[350];

// Communicate
while(1){
    bytes_recv = recv(ser_socket, recv_buf, sizeof(recv_buf), 0);
    if(bytes_recv < 0) printf("Error recving packet.\n");

    printf("%s\n", recv_buf);

    send_buf[0] = '\0';
    fflush(stdin);  //clear buff
    scanf(" %[^\n]", send_buf);  //input

    if(strlen(send_buf) == 0) continue;

    bytes_send = send(ser_socket, send_buf, sizeof(send_buf), 0);
    if(bytes_send < 0) printf("Error sending packet.\n");

    if(strncmp(send_buf, "quit", 4) == 0) break;
}
```

1.2.5 Close

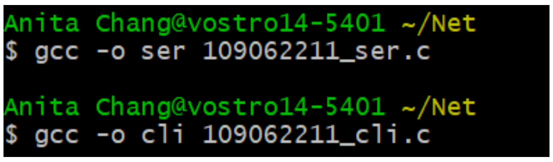
關閉 ser_socket。



```c
close(ser_socket);
printf("\nClose socket.\n");
```

2 執行結果 (使用 Cygwin64 Terminal 執行)

2.1 分別編譯 109062211_ser.c & 109062211_cli.c 兩個檔案，得到 ser.exe & cli.exe



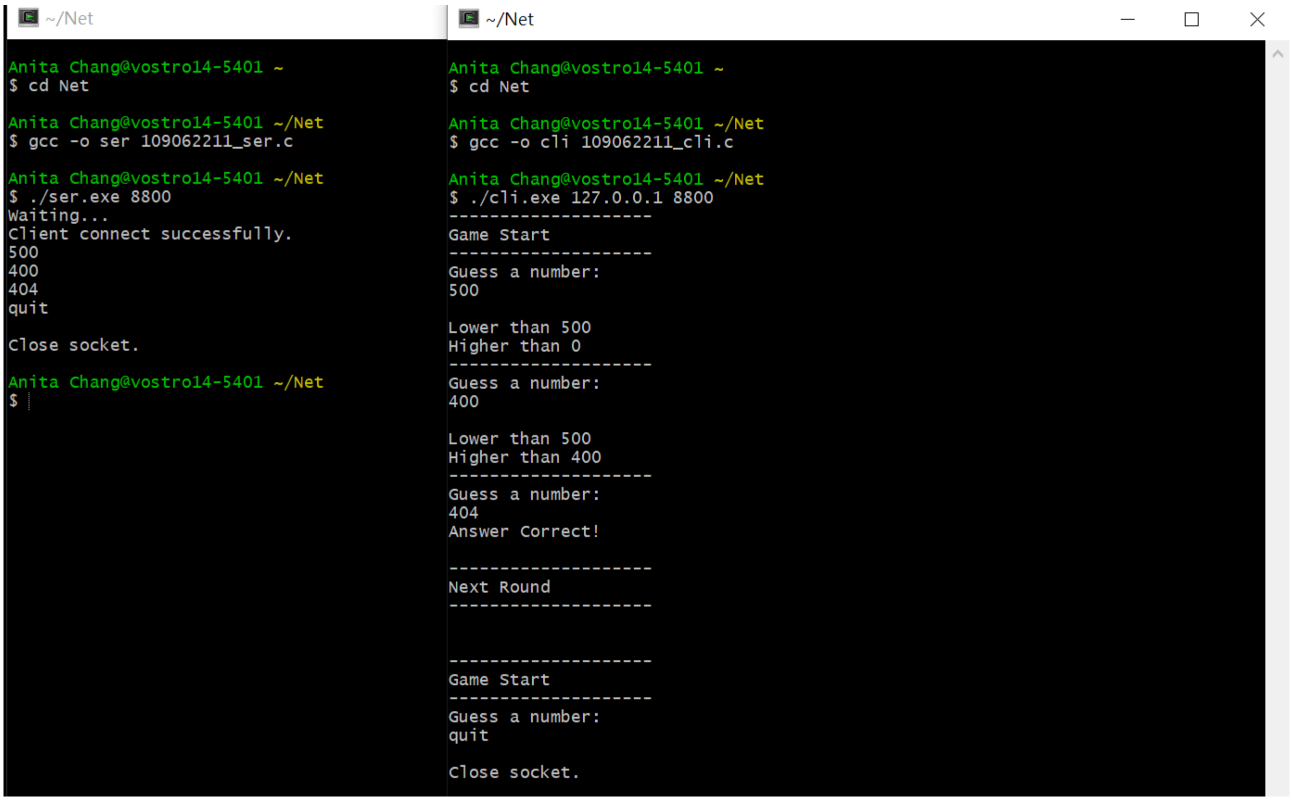
```
Anita Chang@vostro14-5401 ~/Net
$ gcc -o ser 109062211_ser.c

Anita Chang@vostro14-5401 ~/Net
$ gcc -o cli 109062211_cli.c
```

2.2 執行 server 端： ./ser.exe 8800(port_number)

2.3 執行 client 端： ./cli.exe 127.0.0.1(server IP) 8800(port_number)

2.4 Game start

在 client 端印出訊息，並且輸入數字，傳送訊息給 server 端。server 端，印出數字，並傳送訊息給 client 端。server 和 client 持續互相溝通，直到輸入 ”quit” 結束程式。



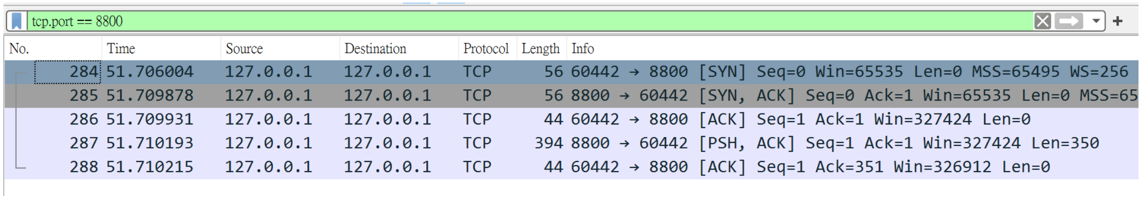
```
Anita Chang@vostro14-5401 ~
$ cd Net

Anita Chang@vostro14-5401 ~/Net
$ gcc -o ser 109062211_ser.c

Anita Chang@vostro14-5401 ~/Net
$ ./ser.exe 8800
Waiting...
Client connect successfully.
500
400
404
quit

Close socket.

Anita Chang@vostro14-5401 ~/Net
$
```

```
Anita Chang@vostro14-5401 ~
$ cd Net

Anita Chang@vostro14-5401 ~/Net
$ gcc -o cli 109062211_cli.c

Anita Chang@vostro14-5401 ~/Net
$ ./cli.exe 127.0.0.1 8800
--------------------
Game Start
--------------------
Guess a number:
500

Lower than 500
Higher than 0
--------------------
Guess a number:
400

Lower than 500
Higher than 400
--------------------
Guess a number:
404
Answer Correct!

--------------------
Next Round
--------------------


--------------------
Game Start
--------------------
Guess a number:
quit

Close socket.
```

3 Wireshark (使用 Windows 系統執行)

3.1 TCP handshaking 封包

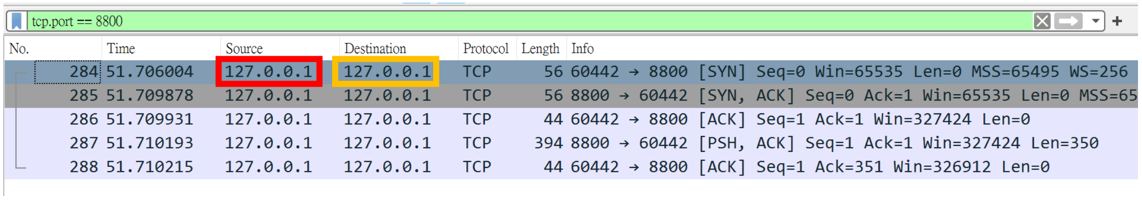
Ans : No. 284. 285. 286 三個封包



```
tcp.port == 8800
No.   Time       Source     Destination Protocol Length Info
  284 51.706004  127.0.0.1  127.0.0.1   TCP       56 60442 → 8800 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256
  285 51.709878  127.0.0.1  127.0.0.1   TCP       56 8800 → 60442 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65
  286 51.709931  127.0.0.1  127.0.0.1   TCP       44 60442 → 8800 [ACK] Seq=1 Ack=1 Win=327424 Len=0
  287 51.710193  127.0.0.1  127.0.0.1   TCP      394 8800 → 60442 [PSH, ACK] Seq=1 Ack=1 Win=327424 Len=350
  288 51.710215  127.0.0.1  127.0.0.1   TCP       44 60442 → 8800 [ACK] Seq=1 Ack=351 Win=326912 Len=0
```

3.2 IP address

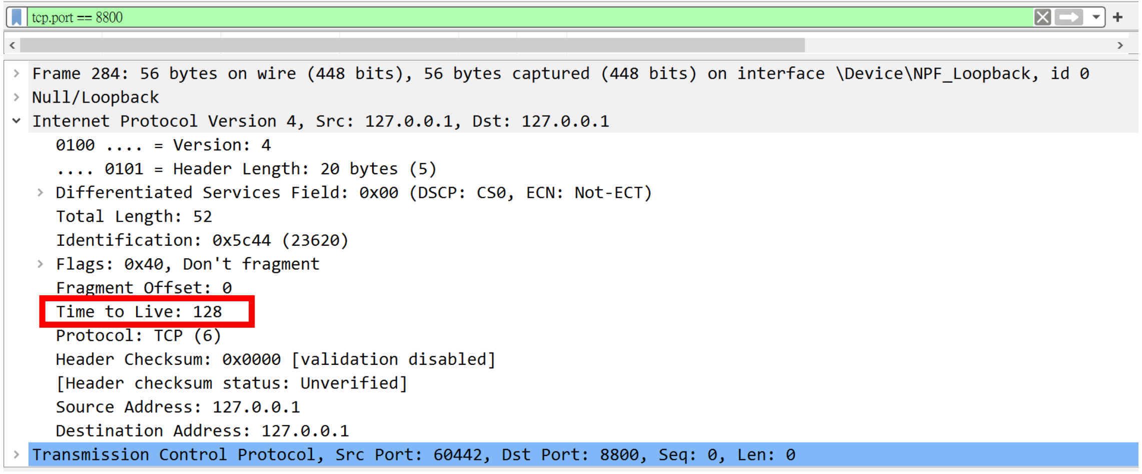
Ans : server IP address = 127.0.0.1 (黃框標示)，client IP address = 127.0.0.1 (紅框標示)



```
tcp.port == 8800
No.   Time       Source     Destination Protocol Length Info
  284 51.706004  127.0.0.1  127.0.0.1   TCP       56 60442 → 8800 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256
  285 51.709878  127.0.0.1  127.0.0.1   TCP       56 8800 → 60442 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65
  286 51.709931  127.0.0.1  127.0.0.1   TCP       44 60442 → 8800 [ACK] Seq=1 Ack=1 Win=327424 Len=0
  287 51.710193  127.0.0.1  127.0.0.1   TCP      394 8800 → 60442 [PSH, ACK] Seq=1 Ack=1 Win=327424 Len=350
  288 51.710215  127.0.0.1  127.0.0.1   TCP       44 60442 → 8800 [ACK] Seq=1 Ack=351 Win=326912 Len=0
```

3.3 router

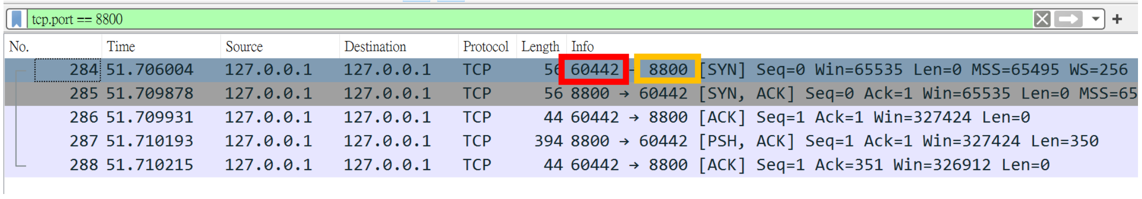
Ans : 0，因為 router 數 = 128 (windows:128、linux:64) – TTL (TTL 值隨經過 router 遞減)



```
tcp.port == 8800
> Frame 284: 56 bytes on wire (448 bits), 56 bytes captured (448 bits) on interface \Device\NPF_Loopback, id 0
> Null/Loopback
v Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 52
    Identification: 0x5c44 (23620)
  > Flags: 0x40, Don't fragment
    Fragment Offset: 0
    Time to Live: 128
    Protocol: TCP (6)
    Header Checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 127.0.0.1
    Destination Address: 127.0.0.1
> Transmission Control Protocol, Src Port: 60442, Dst Port: 8800, Seq: 0, Len: 0
```

3.4 port

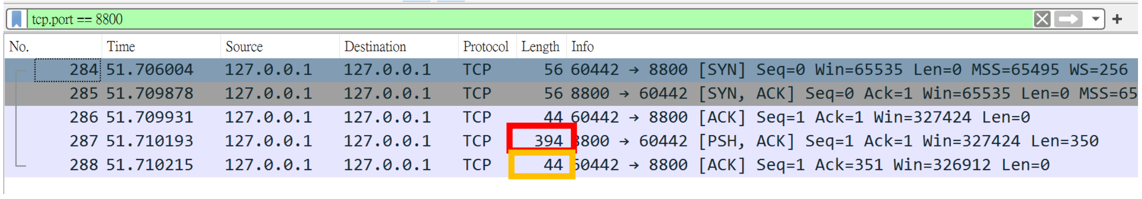
Ans : server IP address = 8800 (黃框標示)，client IP address = 60442 (紅框標示)



```
tcp.port == 8800
No.   Time       Source     Destination Protocol Length Info
  284 51.706004  127.0.0.1  127.0.0.1   TCP       56 60442 → 8800 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256
  285 51.709878  127.0.0.1  127.0.0.1   TCP       56 8800 → 60442 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65
  286 51.709931  127.0.0.1  127.0.0.1   TCP       44 60442 → 8800 [ACK] Seq=1 Ack=1 Win=327424 Len=0
  287 51.710193  127.0.0.1  127.0.0.1   TCP      394 8800 → 60442 [PSH, ACK] Seq=1 Ack=1 Win=327424 Len=350
  288 51.710215  127.0.0.1  127.0.0.1   TCP       44 60442 → 8800 [ACK] Seq=1 Ack=351 Win=326912 Len=0
```

3.5 封包 data 大小

Ans : server to client = 394 bytes (紅框標示)，client to server = 44 bytes (黃框標示)



```
tcp.port == 8800
No.   Time       Source     Destination Protocol Length Info
  284 51.706004  127.0.0.1  127.0.0.1   TCP       56 60442 → 8800 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256
  285 51.709878  127.0.0.1  127.0.0.1   TCP       56 8800 → 60442 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=65
  286 51.709931  127.0.0.1  127.0.0.1   TCP       44 60442 → 8800 [ACK] Seq=1 Ack=1 Win=327424 Len=0
  287 51.710193  127.0.0.1  127.0.0.1   TCP      394 8800 → 60442 [PSH, ACK] Seq=1 Ack=1 Win=327424 Len=350
  288 51.710215  127.0.0.1  127.0.0.1   TCP       44 60442 → 8800 [ACK] Seq=1 Ack=351 Win=326912 Len=0
```

4 學到的東西與遇到的困難

4.1 學習 socket programming

第一次撰寫 socket programming 的程式，原本想說因為不熟悉所以寫起來會很難，但看完 lab_tutorial 之後，發現按照 flow chart 寫下來，其實 socket programming 蠻容易。