3주차_TCP/IP_소켓프로그래밍

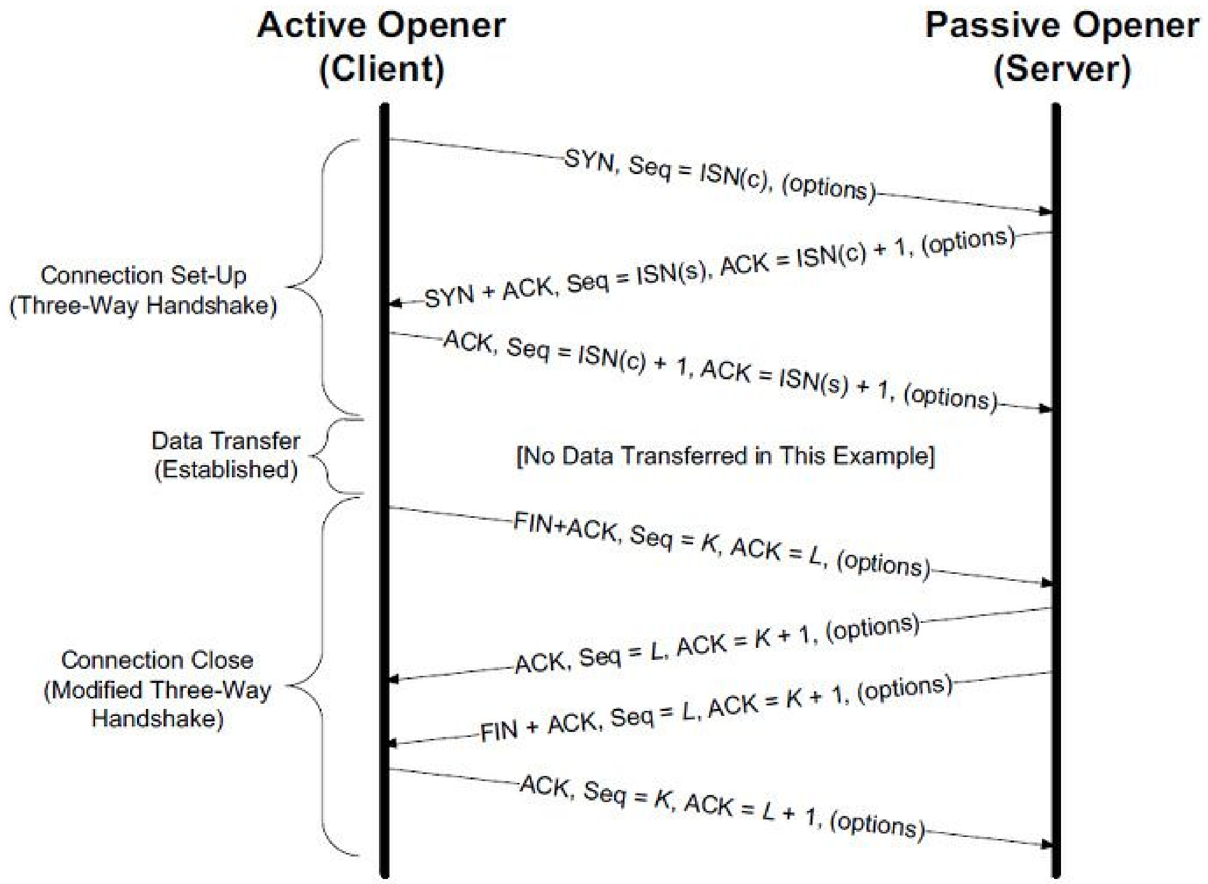
데이터 네트워크연구실
이현호
lee075@cs-cnu.org

Goals

- TCP 통신 이해하기
- UDP와 TCP 소켓의 차이

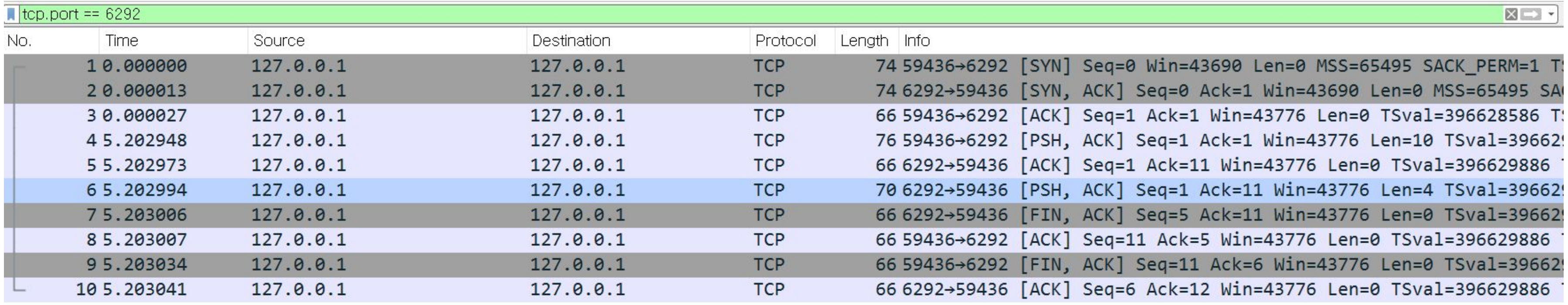
TCP 통신 과정

- 연결 맺기
- 데이터 전송
- 연결끊기


Active Opener (Client)
Passive Opener (Server)
Connection Set-Up (Three-Way Handshake)
SYN, Seq = ISN(c), (options)
SYN + ACK, Seq = ISN(s), ACK = ISN(c) + 1, (options)
ACK, Seq = ISN(c) + 1, ACK = ISN(s) + 1, (options)
Data Transfer (Established)
[No Data Transferred in This Example]
Connection Close (Modified Three-Way Handshake)
FIN+ACK, Seq = K, ACK = L, (options)
ACK, Seq = L, ACK = K + 1, (options)
FIN + ACK, Seq = L, ACK = K + 1, (options)
ACK, Seq = K, ACK = L + 1, (options)

Wire Shark

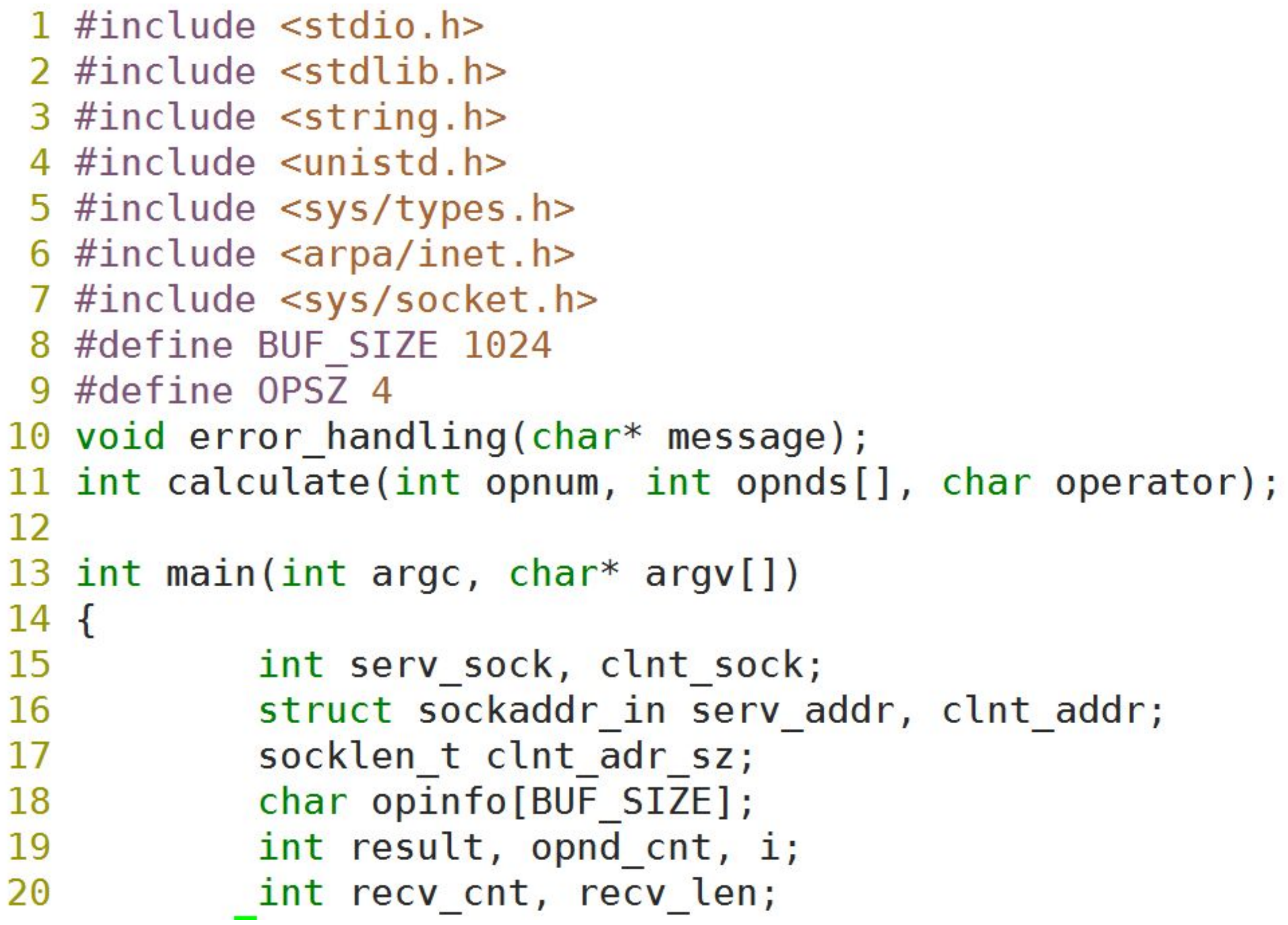
- tcpdump -w tcp.pcap -i lo



```
tcp.port == 6292

No.  Time      Source     Destination  Protocol  Length  Info
  1  0.000000  127.0.0.1  127.0.0.1    TCP       74 59436→6292 [SYN] Seq=0 Win=43690 Len=0 MSS=65495 SACK_PERM=1 T
  2  0.000013  127.0.0.1  127.0.0.1    TCP       74 6292→59436 [SYN, ACK] Seq=0 Ack=1 Win=43690 Len=0 MSS=65495 SA
  3  0.000027  127.0.0.1  127.0.0.1    TCP       66 59436→6292 [ACK] Seq=1 Ack=1 Win=43776 Len=0 TSval=396628586 T
  4  5.202948  127.0.0.1  127.0.0.1    TCP       76 59436→6292 [PSH, ACK] Seq=1 Ack=1 Win=43776 Len=10 TSval=39662
  5  5.202973  127.0.0.1  127.0.0.1    TCP       66 6292→59436 [ACK] Seq=1 Ack=11 Win=43776 Len=0 TSval=396629886
  6  5.202994  127.0.0.1  127.0.0.1    TCP       70 6292→59436 [PSH, ACK] Seq=1 Ack=11 Win=43776 Len=4 TSval=39662
  7  5.203006  127.0.0.1  127.0.0.1    TCP       66 6292→59436 [FIN, ACK] Seq=5 Ack=11 Win=43776 Len=0 TSval=39662
  8  5.203007  127.0.0.1  127.0.0.1    TCP       66 59436→6292 [ACK] Seq=11 Ack=5 Win=43776 Len=0 TSval=396629886
  9  5.203034  127.0.0.1  127.0.0.1    TCP       66 59436→6292 [FIN, ACK] Seq=11 Ack=6 Win=43776 Len=0 TSval=39662
 10  5.203041  127.0.0.1  127.0.0.1    TCP       66 6292→59436 [ACK] Seq=6 Ack=12 Win=43776 Len=0 TSval=396629886
```

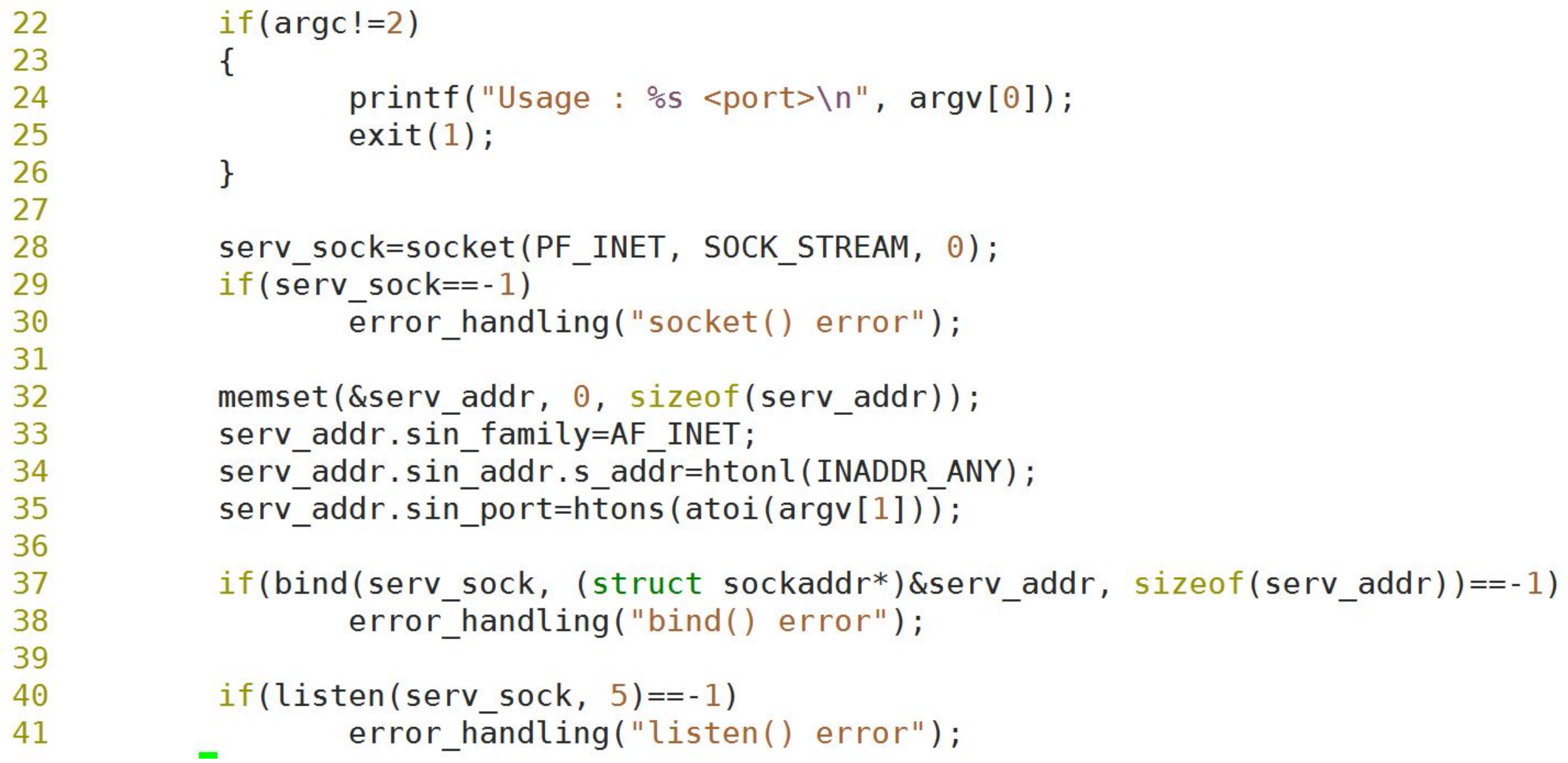
Server

Server



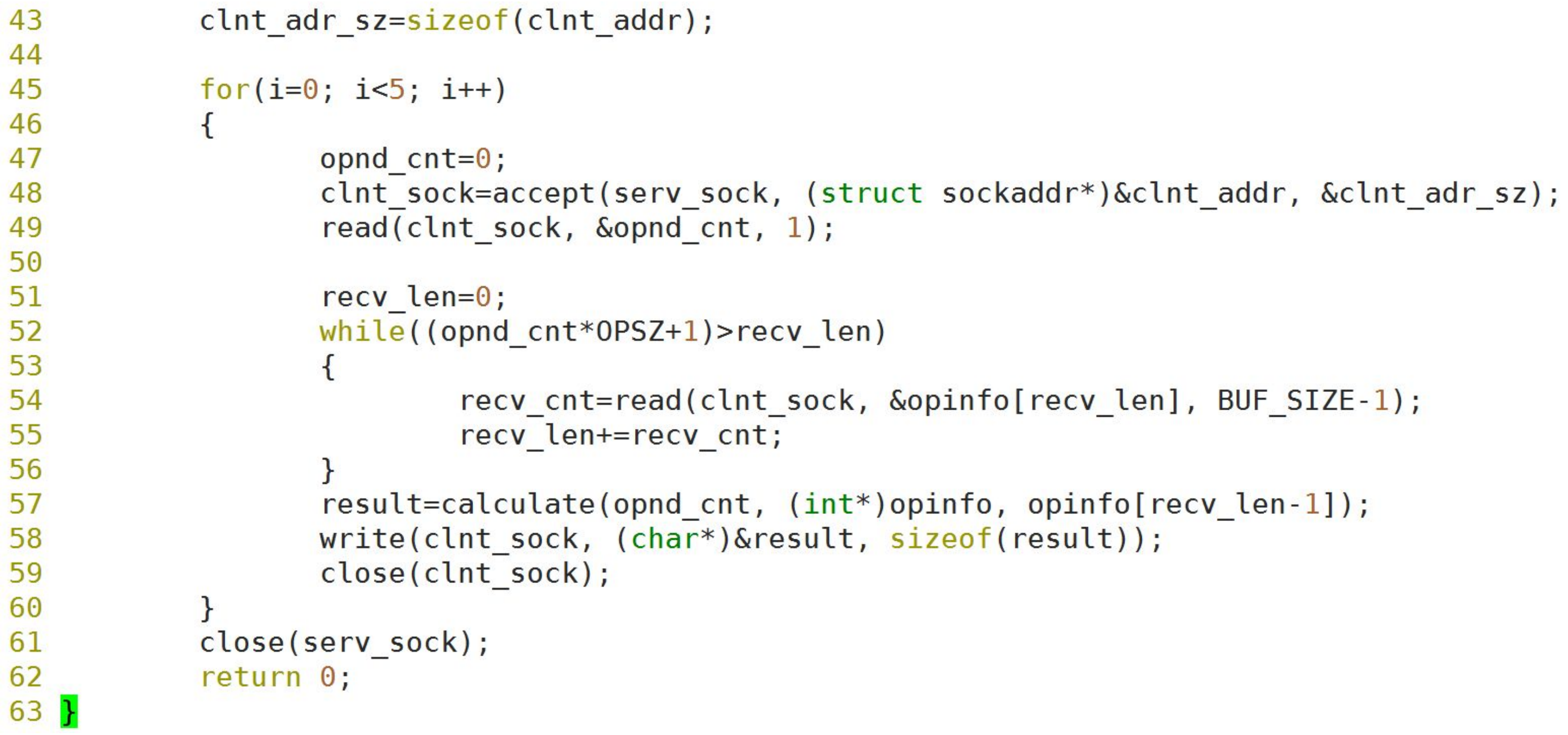
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#define BUF_SIZE 1024
#define OPSZ 4
void error_handling(char* message);
int calculate(int opnum, int opnds[], char operator);

int main(int argc, char* argv[])
{
        int serv_sock, clnt_sock;
        struct sockaddr_in serv_addr, clnt_addr;
        socklen_t clnt_adr_sz;
        char opinfo[BUF_SIZE];
        int result, opnd_cnt, i;
        int recv_cnt, recv_len;
```

Server



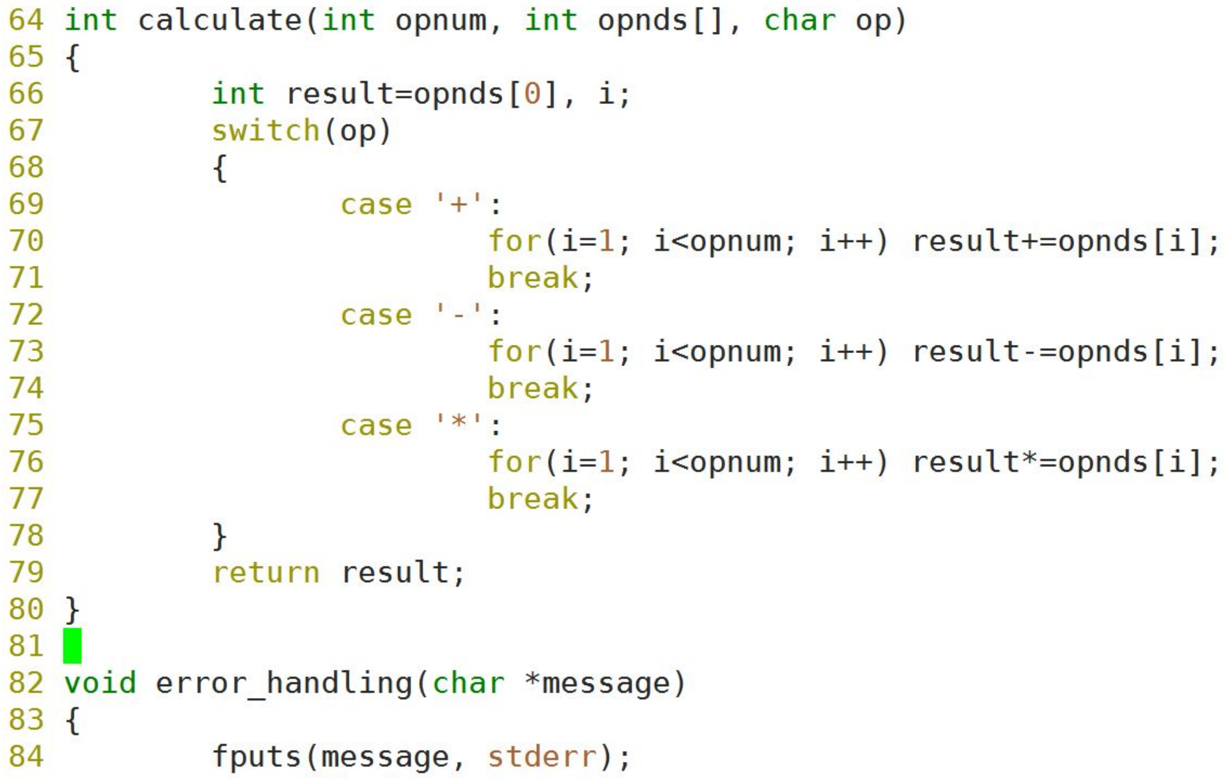
```
        if(argc!=2)
        {
                printf("Usage : %s <port>\n", argv[0]);
                exit(1);
        }

        serv_sock=socket(PF_INET, SOCK_STREAM, 0);
        if(serv_sock==-1)
                error_handling("socket() error");

        memset(&serv_addr, 0, sizeof(serv_addr));
        serv_addr.sin_family=AF_INET;
        serv_addr.sin_addr.s_addr=htonl(INADDR_ANY);
        serv_addr.sin_port=htons(atoi(argv[1]));

        if(bind(serv_sock, (struct sockaddr*)&serv_addr, sizeof(serv_addr))==-1)
                error_handling("bind() error");

        if(listen(serv_sock, 5)==-1)
                error_handling("listen() error");
```

Server



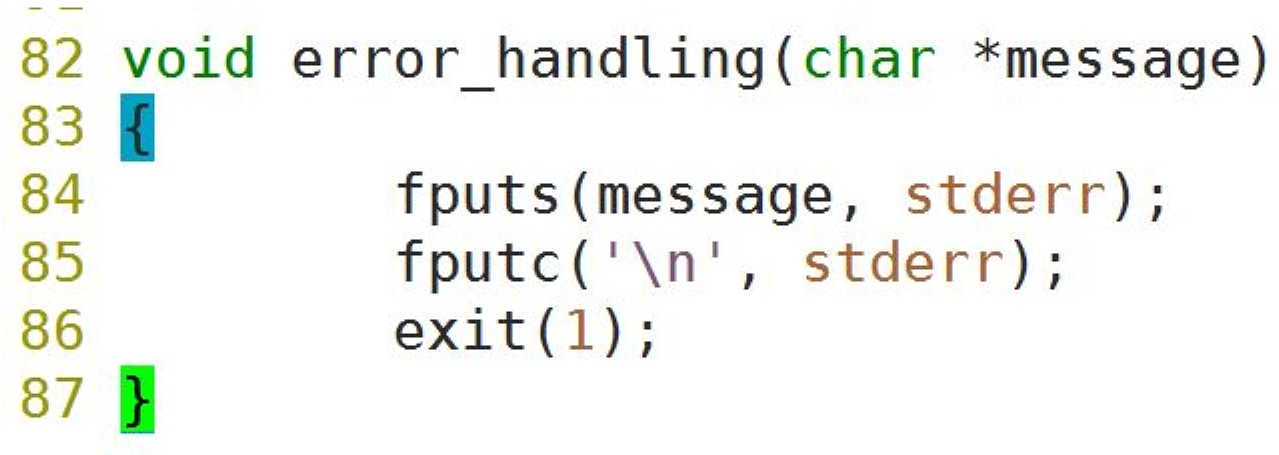
```
        clnt_adr_sz=sizeof(clnt_addr);

        for(i=0; i<5; i++)
        {
                opnd_cnt=0;
                clnt_sock=accept(serv_sock, (struct sockaddr*)&clnt_addr, &clnt_adr_sz);
                read(clnt_sock, &opnd_cnt, 1);

                recv_len=0;
                while((opnd_cnt*OPSZ+1)>recv_len)
                {
                        recv_cnt=read(clnt_sock, &opinfo[recv_len], BUF_SIZE-1);
                        recv_len+=recv_cnt;
                }
                result=calculate(opnd_cnt, (int*)opinfo, opinfo[recv_len-1]);
                write(clnt_sock, (char*)&result, sizeof(result));
                close(clnt_sock);
        }
        close(serv_sock);
        return 0;
}
```

Server



```c
int calculate(int opnum, int opnds[], char op)
{
        int result=opnds[0], i;
        switch(op)
        {
               case '+':
                       for(i=1; i<opnum; i++) result+=opnds[i];
                       break;
               case '-':
                       for(i=1; i<opnum; i++) result-=opnds[i];
                       break;
               case '*':
                       for(i=1; i<opnum; i++) result*=opnds[i];
                       break;
        }
        return result;
}

void error_handling(char *message)
{
        fputs(message, stderr);
```

Server

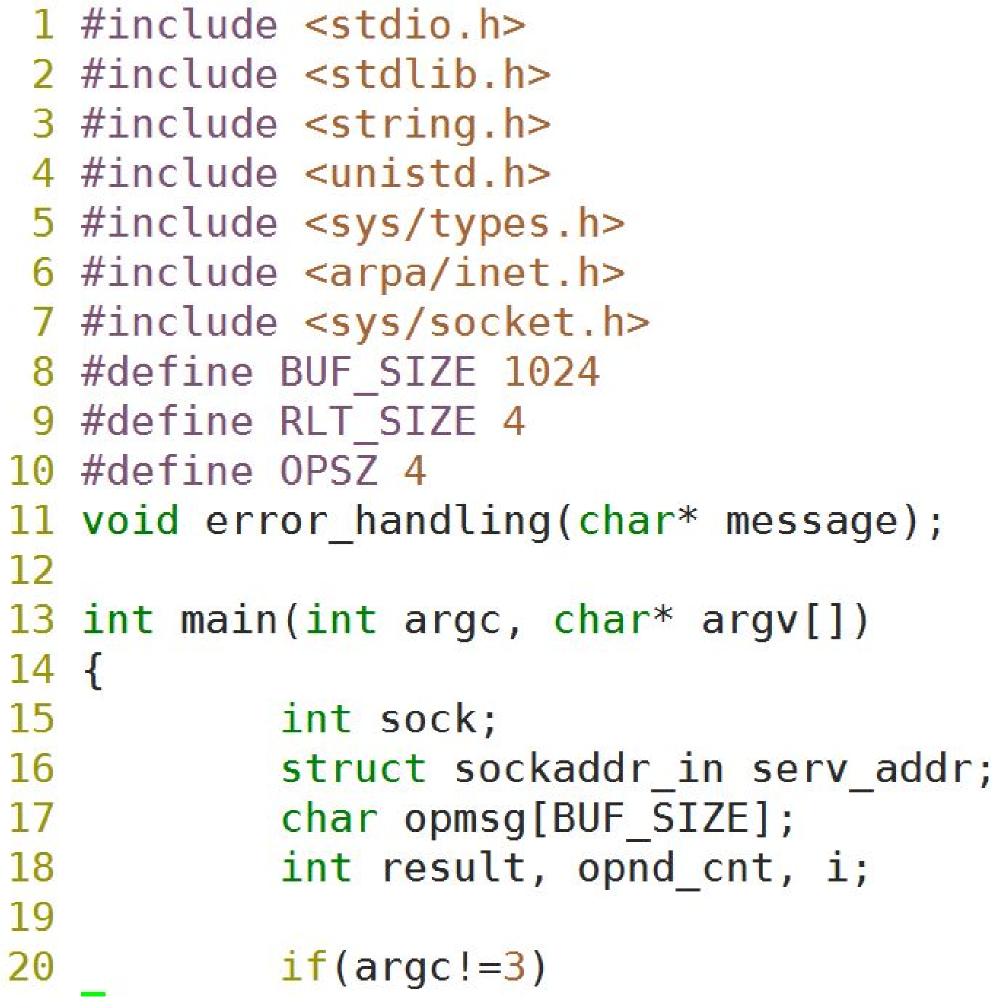


```
void error_handling(char *message)
{
                fputs(message, stderr);
                fputc('\n', stderr);
                exit(1);
}
```

Client

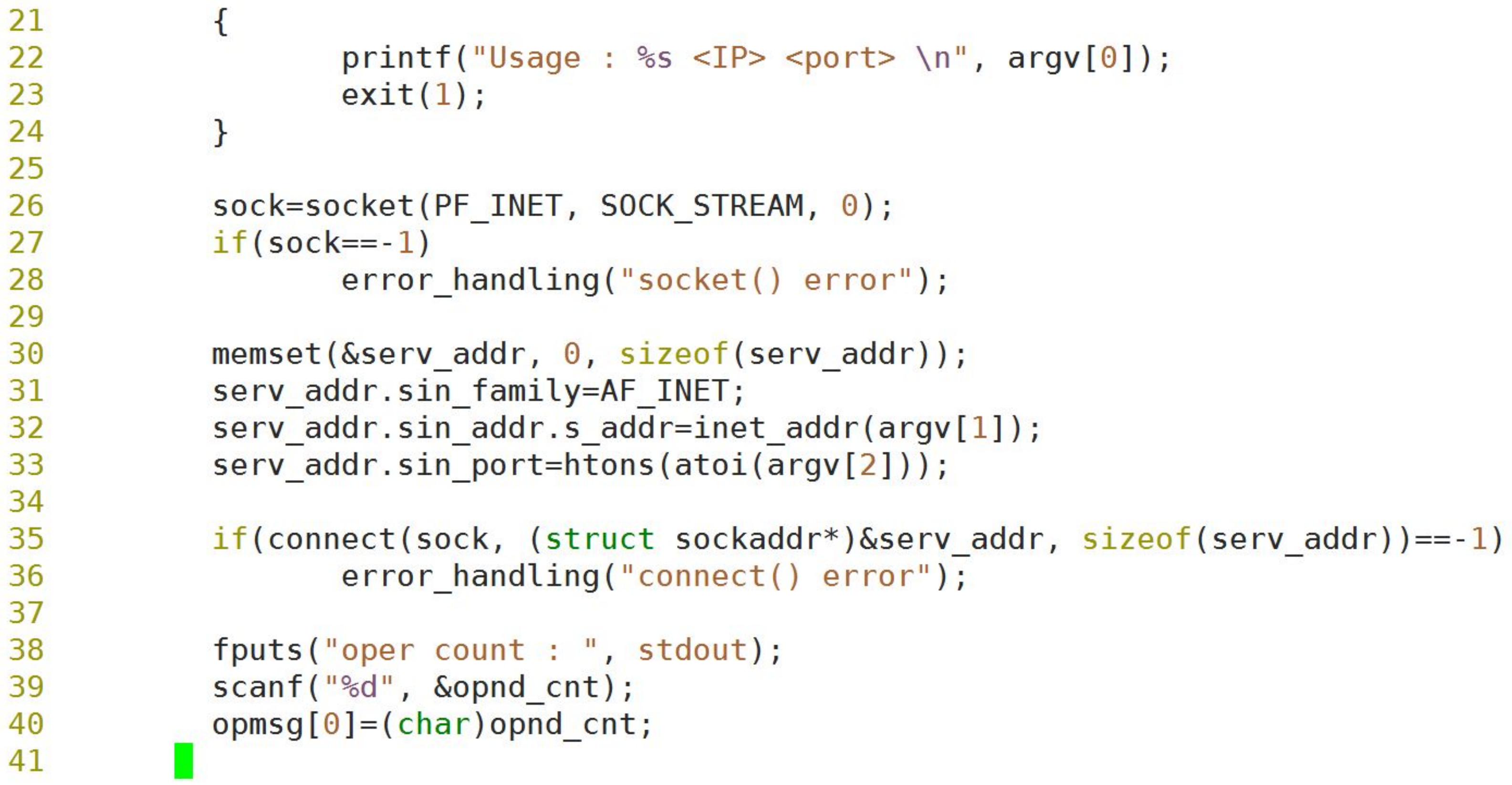
Client

-



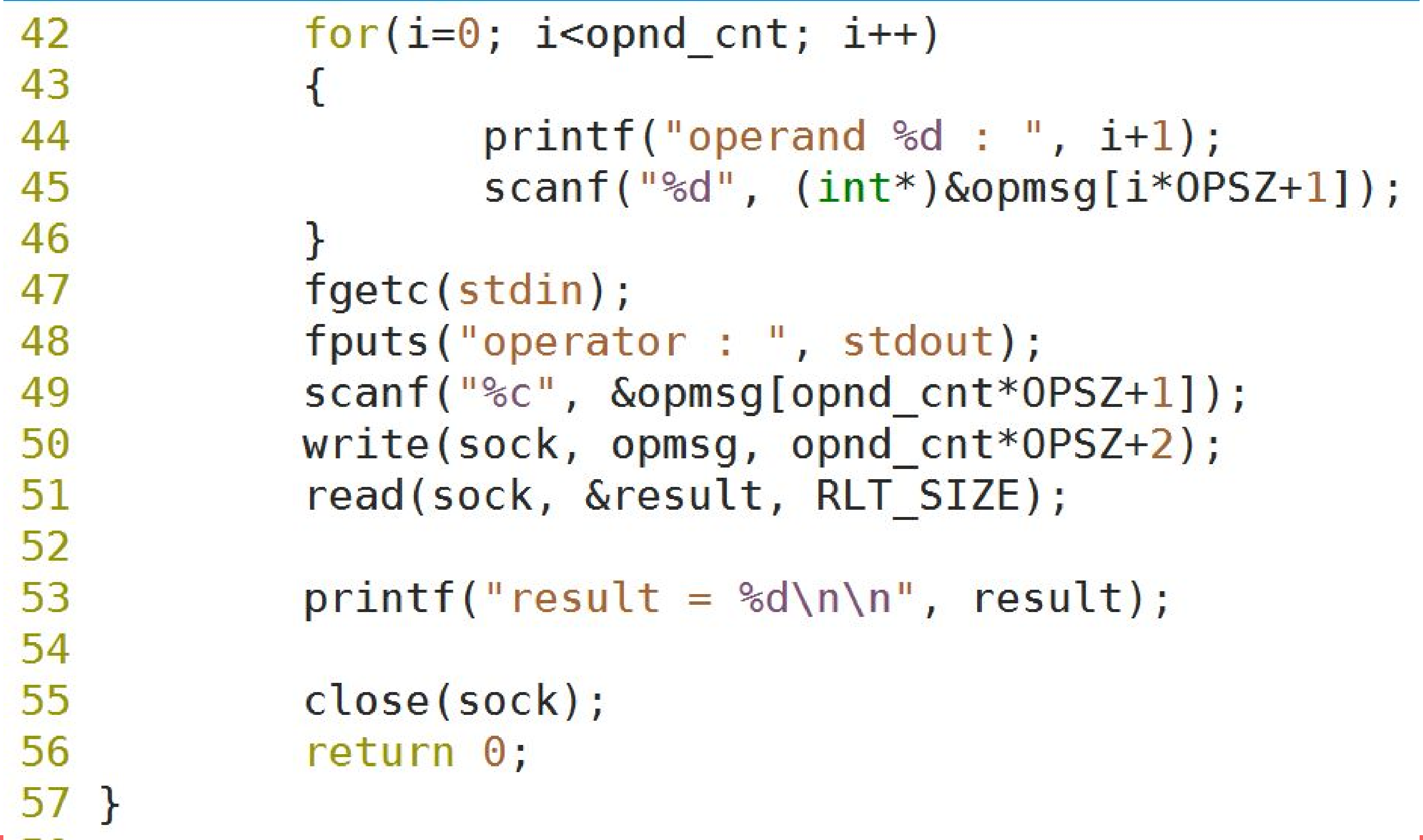
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#define BUF_SIZE 1024
#define RLT_SIZE 4
#define OPSZ 4
void error_handling(char* message);

int main(int argc, char* argv[])
{
        int sock;
        struct sockaddr_in serv_addr;
        char opmsg[BUF_SIZE];
        int result, opnd_cnt, i;

        if(argc!=3)
```

Client



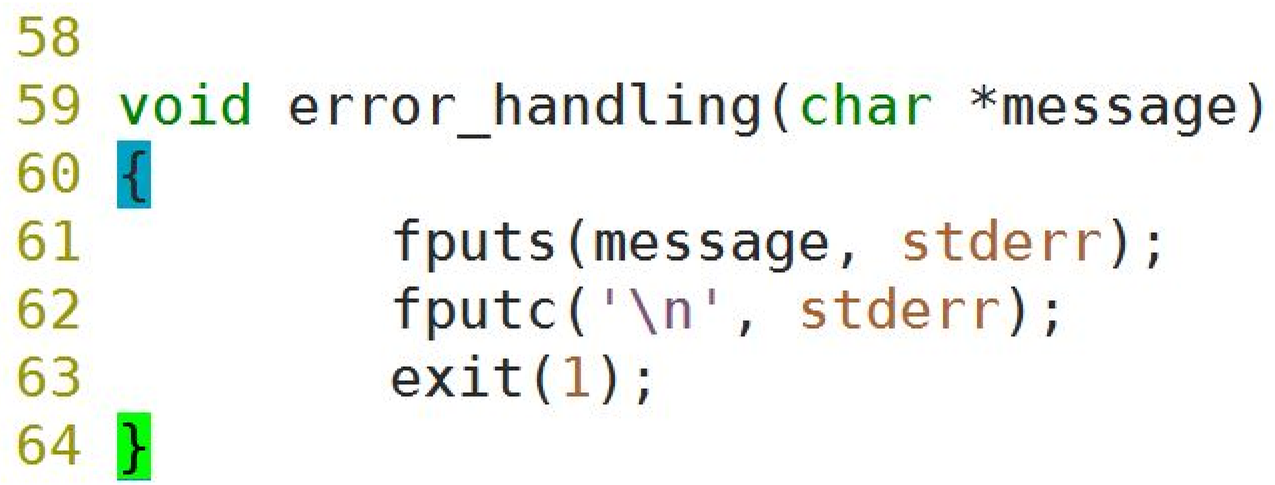
```
	{
		printf("Usage : %s <IP> <port> \n", argv[0]);
		exit(1);
	}

	sock=socket(PF_INET, SOCK_STREAM, 0);
	if(sock==-1)
		error_handling("socket() error");

	memset(&serv_addr, 0, sizeof(serv_addr));
	serv_addr.sin_family=AF_INET;
	serv_addr.sin_addr.s_addr=inet_addr(argv[1]);
	serv_addr.sin_port=htons(atoi(argv[2]));

	if(connect(sock, (struct sockaddr*)&serv_addr, sizeof(serv_addr))==-1)
		error_handling("connect() error");

	fputs("oper count : ", stdout);
	scanf("%d", &opnd_cnt);
	opmsg[0]=(char)opnd_cnt;

```

Client



```
	for(i=0; i<opnd_cnt; i++)
	{
		printf("operand %d : ", i+1);
		scanf("%d", (int*)&opmsg[i*OPSZ+1]);
	}
	fgetc(stdin);
	fputs("operator : ", stdout);
	scanf("%c", &opmsg[opnd_cnt*OPSZ+1]);
	write(sock, opmsg, opnd_cnt*OPSZ+2);
	read(sock, &result, RLT_SIZE);

	printf("result = %d\n\n", result);

	close(sock);
	return 0;
}
```

Client

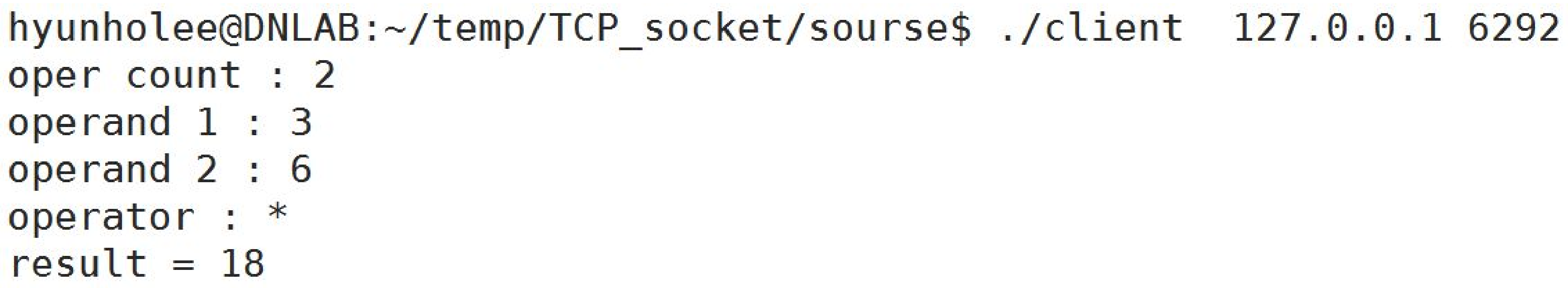


```c

void error_handling(char *message)
{
	fputs(message, stderr);
	fputc('\n', stderr);
	exit(1);
}
```

Result

Result

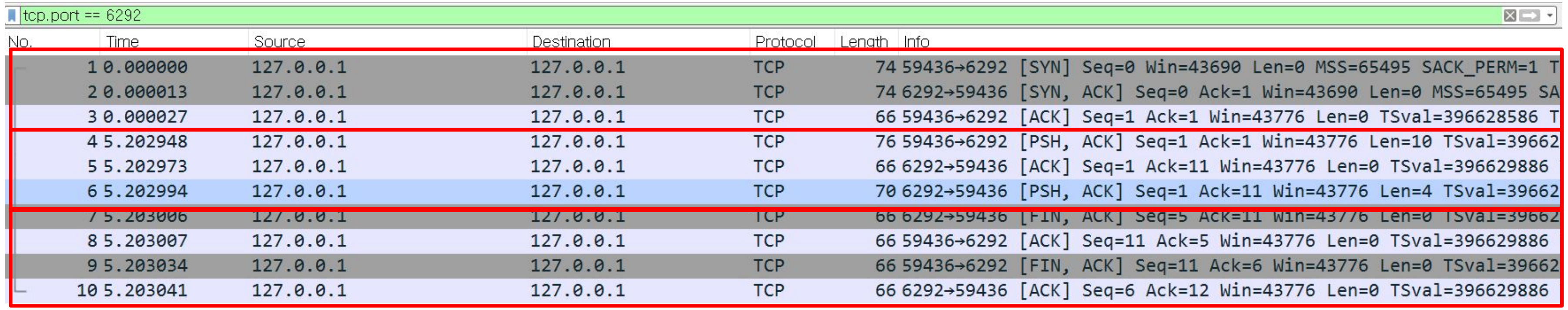
-



```
hyunholee@DNLAB:~/temp/TCP_socket/sourse$ ./client  127.0.0.1 6292
oper count : 2
operand 1 : 3
operand 2 : 6
operator : *
result = 18
```

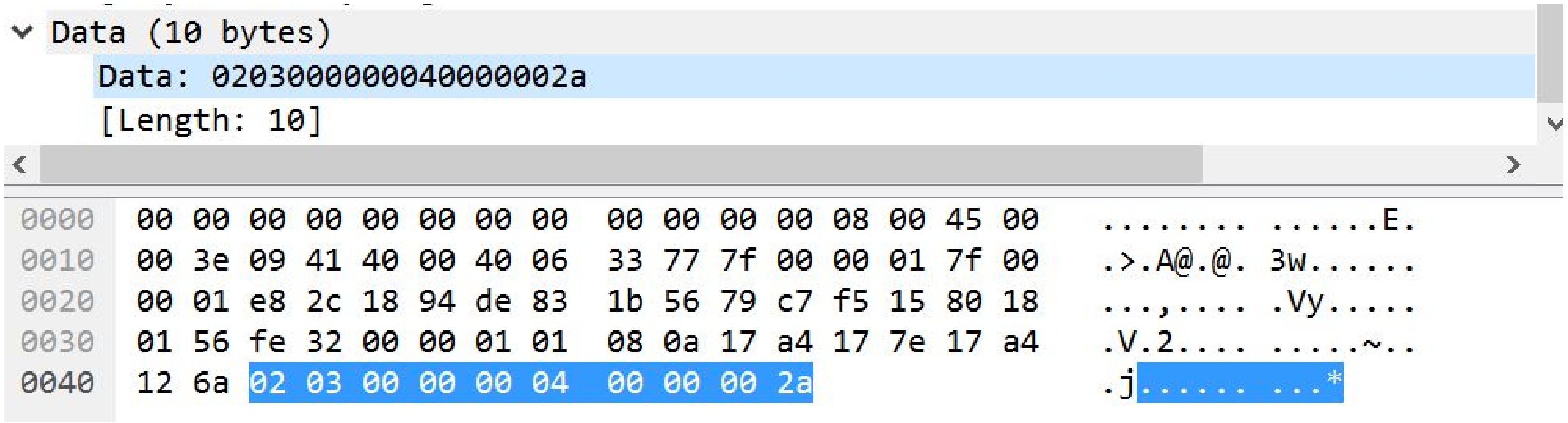
Wire Shark

- 구조 파악하기



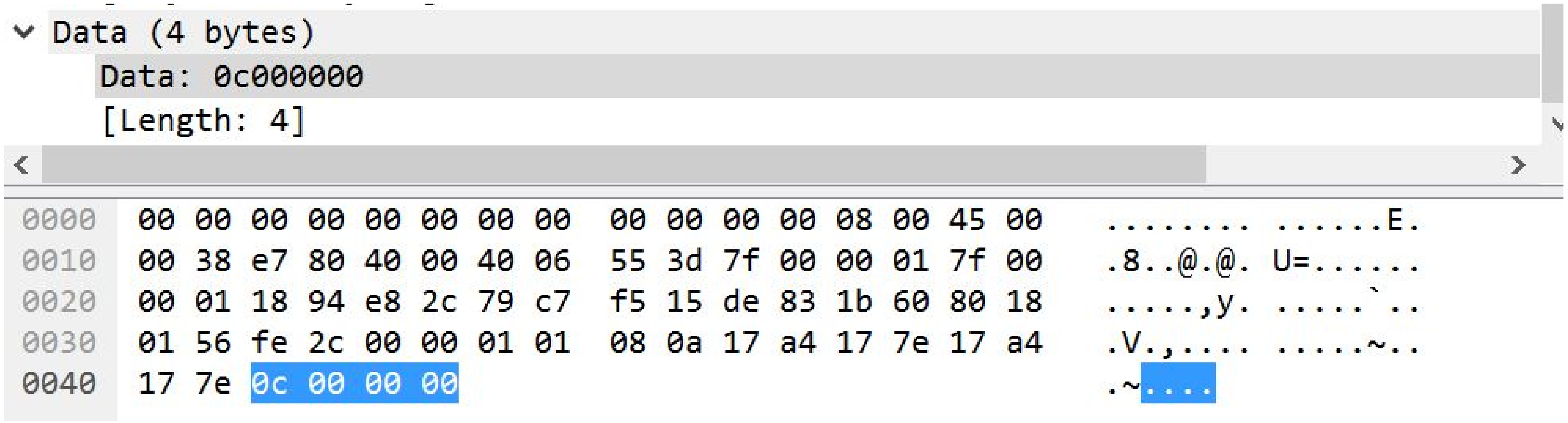
```
tcp.port == 6292

No.   Time       Source      Destination  Protocol  Length  Info
    1 0.000000   127.0.0.1   127.0.0.1    TCP         74 59436→6292 [SYN] Seq=0 Win=43690 Len=0 MSS=65495 SACK_PERM=1 T
    2 0.000013   127.0.0.1   127.0.0.1    TCP         74 6292→59436 [SYN, ACK] Seq=0 Ack=1 Win=43690 Len=0 MSS=65495 SA
    3 0.000027   127.0.0.1   127.0.0.1    TCP         66 59436→6292 [ACK] Seq=1 Ack=1 Win=43776 Len=0 TSval=396628586 T
    4 5.202948   127.0.0.1   127.0.0.1    TCP         76 59436→6292 [PSH, ACK] Seq=1 Ack=1 Win=43776 Len=10 TSval=39662
    5 5.202973   127.0.0.1   127.0.0.1    TCP         66 6292→59436 [ACK] Seq=1 Ack=11 Win=43776 Len=0 TSval=396629886
    6 5.202994   127.0.0.1   127.0.0.1    TCP         70 6292→59436 [PSH, ACK] Seq=1 Ack=11 Win=43776 Len=4 TSval=39662
    7 5.203006   127.0.0.1   127.0.0.1    TCP         66 6292→59436 [FIN, ACK] Seq=5 Ack=11 Win=43776 Len=0 TSval=39662
    8 5.203007   127.0.0.1   127.0.0.1    TCP         66 59436→6292 [ACK] Seq=11 Ack=5 Win=43776 Len=0 TSval=396629886
    9 5.203034   127.0.0.1   127.0.0.1    TCP         66 59436→6292 [FIN, ACK] Seq=11 Ack=6 Win=43776 Len=0 TSval=39662
   10 5.203041   127.0.0.1   127.0.0.1    TCP         66 6292→59436 [ACK] Seq=6 Ack=12 Win=43776 Len=0 TSval=396629886
```

Result



```
Data (10 bytes)
    Data: 0203000000040000002a
    [Length: 10]

0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ........ ......E.
0010  00 3e 09 41 40 00 40 06  33 77 7f 00 00 01 7f 00   .>.A@.@. 3w......
0020  00 01 e8 2c 18 94 de 83  1b 56 79 c7 f5 15 80 18   ...,.... .Vy.....
0030  01 56 fe 32 00 00 01 01  08 0a 17 a4 17 7e 17 a4   .V.2.... .....~..
0040  12 6a 02 03 00 00 00 04  00 00 00 2a               .j...... ...*
```

Result



```
Data (4 bytes)
    Data: 0c000000
    [Length: 4]

0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ........ ......E.
0010  00 38 e7 80 40 00 40 06  55 3d 7f 00 00 01 7f 00   .8..@.@. U=......
0020  00 01 18 94 e8 2c 79 c7  f5 15 de 83 1b 60 80 18   .....,y. .....`..
0030  01 56 fe 2c 00 00 01 01  08 0a 17 a4 17 7e 17 a4   .V.,.... .....~..
0040  17 7e 0c 00 00 00                                  .~....
```

Assignment

과제

1. TCP 통신 주석달기

과제 제출

- 과제 제출 기한:
    - 실습 하루 전 18시
- e-learing 페이지에 제출
- 보고서 제목 : NW_학번_이름_실습번호.pdf
- 추가 첨부파일 : NW_학번_이름_실습번호.zip
    - 추가 첨부파일은 본인이 작성한 파일로 제한합니다