

REPORT



과 목 :	네트워크프로그래밍
제출일자 :	2021. 11. 10
담당교수 :	황성호
학 과 :	컴퓨터공학과
학 번 :	201720970
이 름 :	권대한

1. TCPServer_FixedVariable.c의 소스코드

```
#define _WINSOCK_DEPRECATED_NO_WARNINGS

#pragma comment(lib, "ws2_32")

#include <stdio.h>
#include <stdlib.h>
#include <WinSock2.h>

#define SERVERPORT 9000
#define BUFSIZE 512

// 소켓 함수 오류 출력 후 종료
void err_quit(char* msg)
{
    LPVOID lpMsgBuf;
    FormatMessage(
        FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM,
        NULL,
        WSAGetLastError(),
        MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),
        (LPTSTR)&lpMsgBuf, 0, NULL);
    MessageBox(NULL, (LPCTSTR)lpMsgBuf, msg, MB_ICONERROR);
    LocalFree(lpMsgBuf);
    exit(1);
}

// 소켓 함수 오류 출력
void err_display(char* msg)
{
    LPVOID lpMsgBuf;
    FormatMessage(FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM,
        NULL,
        WSAGetLastError(),
        MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),
        (LPTSTR)&lpMsgBuf, 0, NULL);
    printf("[%s] %s", msg, (char*)lpMsgBuf);
    LocalFree(lpMsgBuf);
}
```

// 사용자 정의 데이터 수신 함수

```
int recvn(SOCKET s, char* buf, int len, int flags)
```

```
{
    int received;
    char* ptr = buf;
    int left = len;

    while (left > 0)
    {
        received = recv(s, ptr, left, flags);
        if (received == SOCKET_ERROR)
        {
            return SOCKET_ERROR;
        }
        else if (received == 0)
        {
            break;
        }
        left -= received;
        ptr += received;
    }

    return (len - left);
}
```

```
int main(int argc, char* argv[])
```

```
{
    int retval;
    //WinSock init
    WSADATA wsa;
    if (WSAStartup(MAKEWORD(2, 2), &wsa) != 0)
    {
        return 1;
    }
    //Socket()
    SOCKET listen_sock = socket(AF_INET, SOCK_STREAM, 0);
    if (listen_sock == INVALID_SOCKET)
    {
        err_quit("socket()");
    }
    //Bind()
```

```

SOCKADDR_IN serveraddr;
ZeroMemory(&serveraddr, sizeof(serveraddr));
serveraddr.sin_family = AF_INET;
serveraddr.sin_addr.s_addr = htonl(INADDR_ANY);
serveraddr.sin_port = htons(SERVERPORT);
retval = bind(listen_sock, (SOCKADDR*)&serveraddr, sizeof(serveraddr));
if (retval == SOCKET_ERROR)
{
    err_quit("bind()");
}
//Listen()
retval = listen(listen_sock, SOMAXCONN);
if (retval == SOCKET_ERROR)
{
    err_quit("listen()");
}
// 데이터 통신에 사용할 변수
SOCKET client_sock;
SOCKADDR_IN clientaddr;
int addrlen;
char buf[BUFSIZE + 1];
int len;
while (1)
{
    //accept()
    addrlen = sizeof(clientaddr);
    client_sock = accept(listen_sock, (SOCKADDR*)&clientaddr, &addrlen);
    if (client_sock == INVALID_SOCKET)
    {
        err_display("accept()");
        break;
    }
    // 접속한 클라이언트 정보 출력
    printf("\n [TCP 서버] 클라이언트 접속: IP 주소=%s, 포트 번호=%d\n",
inet_ntoa(clientaddr.sin_addr), ntohs(clientaddr.sin_port));

    //클라이언트와 데이터 통신
    while (1)
    {
        //데이터 받기
        retval = recvn(client_sock, (char *)&len, sizeof(int), 0);
    }
}

```

```

        if (retval == SOCKET_ERROR)
        {
            err_display("recv()");
            break;
        }
        else if (retval == 0)
        {
            break;
        }
        // 데이터 받기 (가변 길이)
        retval = recvn(client_sock, buf, len, 0);
        if (retval == SOCKET_ERROR)
        {
            err_display("recv()");
            break;
        }
        else if (retval == 0)
        {
            break;
        }
        //받은 데이터 출력
        buf[retval] = '\0';
        printf("[TCP/%s:%d] %s\n", inet_ntoa(clientaddr.sin_addr),
ntohs(clientaddr.sin_port), buf);
    }
    //closesocket()
    closesocket(client_sock);
    printf("[TCP 서버] 클라이언트 종료: IP 주소=%s, 포트 번호=%d\n",
inet_ntoa(clientaddr.sin_addr), ntohs(clientaddr.sin_port));

}

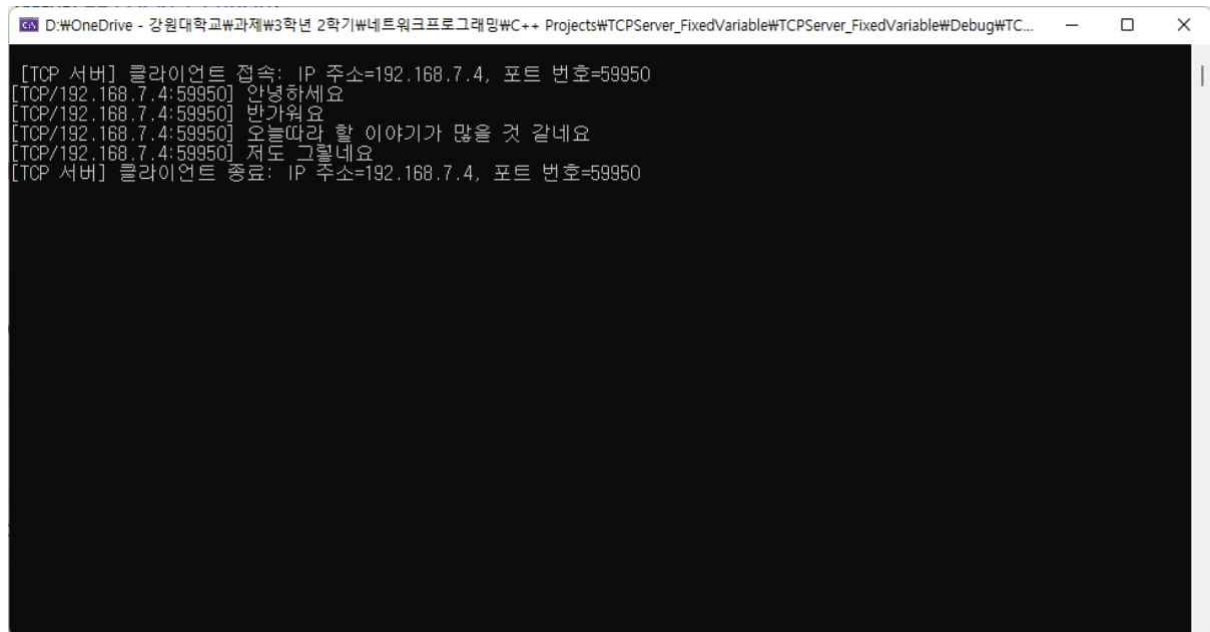
//closesocket()
closesocket(listen_sock);

//원속 종료
WSACleanup();
return 0;
}

```

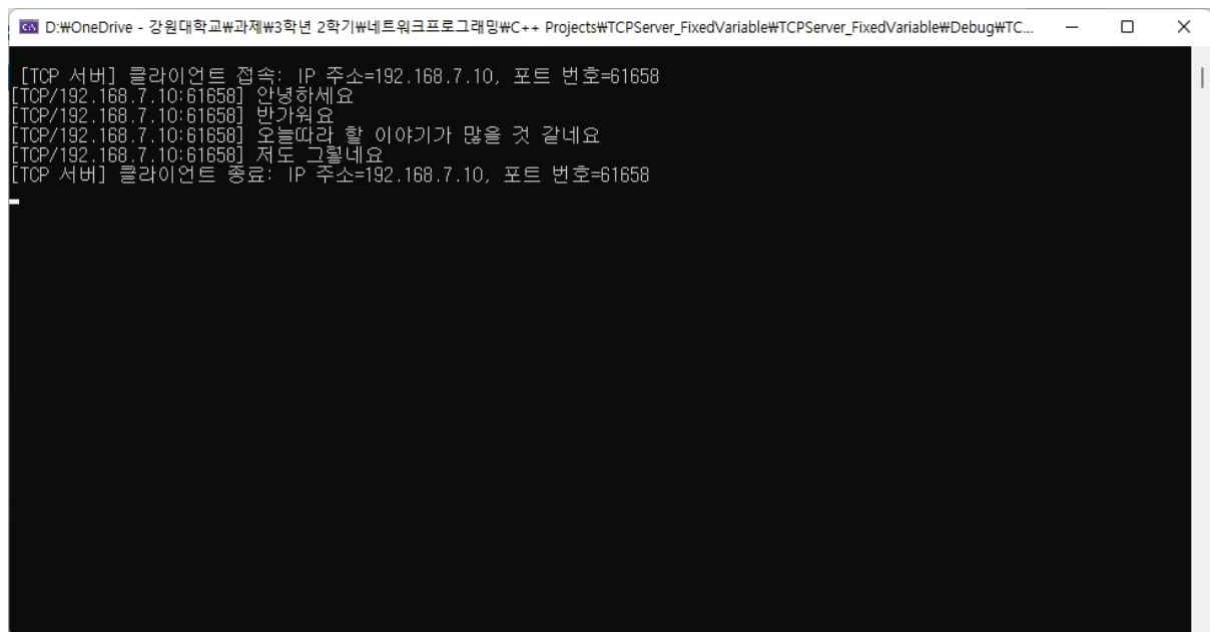
TCPServer_FixedVariable.c의 출력결과

Case 1. 192.168.7.4로부터 데이터를 수신했을 경우



```
D:\OneDrive - 강원대학교\학과\제3학년 2학기\네트워크프로그래밍\C++ Projects\TCPServer_FixedVariable\TCPServer_FixedVariable\Debug\TC...  
[TCP 서버] 클라이언트 접속: IP 주소=192.168.7.4, 포트 번호=59950  
[TCP/192.168.7.4:59950] 안녕하세요  
[TCP/192.168.7.4:59950] 반가워요  
[TCP/192.168.7.4:59950] 오늘따라 할 이야기가 많을 것 같네요  
[TCP/192.168.7.4:59950] 저도 그럴네요  
[TCP 서버] 클라이언트 종료: IP 주소=192.168.7.4, 포트 번호=59950
```

Case 2. 192.168.7.10로부터 데이터를 수신했을 경우



```
D:\OneDrive - 강원대학교\학과\제3학년 2학기\네트워크프로그래밍\C++ Projects\TCPServer_FixedVariable\TCPServer_FixedVariable\Debug\TC...  
[TCP 서버] 클라이언트 접속: IP 주소=192.168.7.10, 포트 번호=61658  
[TCP/192.168.7.10:61658] 안녕하세요  
[TCP/192.168.7.10:61658] 반가워요  
[TCP/192.168.7.10:61658] 오늘따라 할 이야기가 많을 것 같네요  
[TCP/192.168.7.10:61658] 저도 그럴네요  
[TCP 서버] 클라이언트 종료: IP 주소=192.168.7.10, 포트 번호=61658
```

2. TCPClient_FixedVariable.c의 소스코드

```
#pragma warning(disable : 4996)

#define _WINSOCK_DEPRECATED_NO_WARNINGS

#pragma comment(lib, "ws2_32")

#include <stdio.h>
#include <stdlib.h>
#include <WinSock2.h>

#define SERVERIP "192.168.7.4"
#define SERVERPORT 9000
#define BUFSIZE 50

// 소켓 함수 오류 출력 후 종료
void err_quit(char* msg)
{
    LPVOID lpMsgBuf;
    FormatMessage(
        FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM,
        NULL,
        WSAGetLastError(),
        MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),
        (LPTSTR)&lpMsgBuf, 0, NULL);
    MessageBox(NULL, (LPCTSTR)lpMsgBuf, msg, MB_ICONERROR);
    LocalFree(lpMsgBuf);
    exit(1);
}

// 소켓 함수 오류 출력
void err_display(char* msg)
{
    LPVOID lpMsgBuf;
    FormatMessage(FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM,
        NULL,
        WSAGetLastError(),
        MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),
        (LPTSTR)&lpMsgBuf, 0, NULL);
    printf("[%s] %s", msg, (char*)lpMsgBuf);
}
```

```

        LocalFree(lpMsgBuf);
    }

int main(int argc, char* argv[])
{
    int retval;

    //원속 초기화
    WSADATA wsa;
    if (WSAStartup(MAKEWORD(2, 2), &wsa) != 0)
    {
        return 1;
    }

    //Socket()
    SOCKET sock = socket(AF_INET, SOCK_STREAM, 0);
    if (sock == INVALID_SOCKET)
    {
        err_quit("socket()");
    }

    //Connect()
    SOCKADDR_IN serveraddr;
    ZeroMemory(&serveraddr, sizeof(serveraddr));
    serveraddr.sin_family = AF_INET;
    serveraddr.sin_addr.s_addr = inet_addr(SERVERIP);
    serveraddr.sin_port = htons(SERVERPORT);
    retval = connect(sock, (SOCKADDR*)&serveraddr, sizeof(serveraddr));
    if (retval == SOCKET_ERROR)
    {
        err_quit("connect()");
    }

    // 데이터 통신에 사용할 변수
    char buf[BUFSIZE];
    char* testdata[] = {
        "안녕하세요.",
        "반가워요.",
        "오늘따라 할 이야기가 많을 것 같네요.",
        "저도 그렇네요."
    };
};

```



```
int len;

// 서버와 데이터 통신
for (int i = 0; i < 4; i++)
{
    //데이터 입력(시뮬레이션)
    len = strlen(testdata[i]);
    strncpy(buf, testdata[i], len);

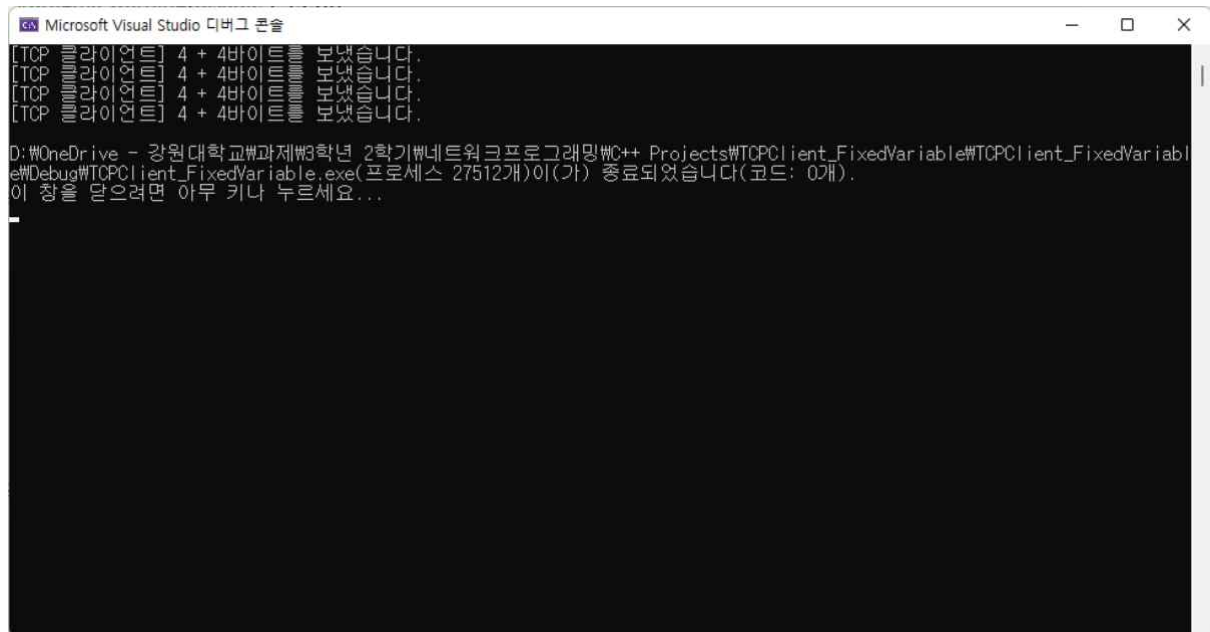
    //데이터 보내기
    retval = send(sock, (char *)&len, sizeof(int), 0);
    if (retval == SOCKET_ERROR)
    {
        err_display("send()");
        break;
    }
    printf("[TCP 클라이언트] %d + %d바이트를 보냈습니다.\n", sizeof(int), retval);
}

//Closesocket()
closesocket(sock);

//원속 종료
WSACleanup();
return 0;
}
```

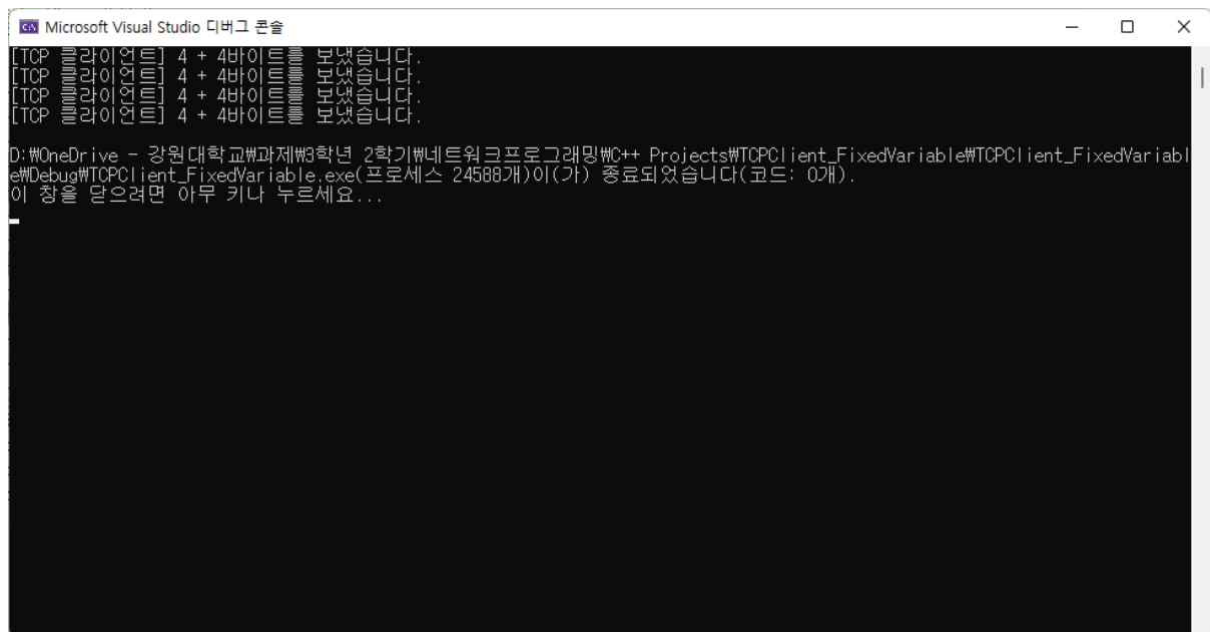
TCPClient_FixedVariable.c의 출력결과

Case 1. Server-192.168.7.4로 데이터를 송신하였을 경우



```
Microsoft Visual Studio 디버그 콘솔
[TCP 클라이언트] 4 + 4바이트 보냈습니다.
[TCP 클라이언트] 4 + 4바이트 보냈습니다.
[TCP 클라이언트] 4 + 4바이트 보냈습니다.
[TCP 클라이언트] 4 + 4바이트 보냈습니다.
D:\OneDrive - 강원대학교과제\3학년 2학기\네트워크 프로그래밍\C++ Projects\TCPClient_FixedVariable\Debug\TCPClient_FixedVariable.exe(프로세스 27512개)이(가) 종료되었습니다(코드: 0x0).
이 창을 닫으려면 아무 키나 누르세요...
```

Case 2. Server-192.168.7.10로 데이터를 송신하였을 경우



```
Microsoft Visual Studio 디버그 콘솔
[TCP 클라이언트] 4 + 4바이트 보냈습니다.
[TCP 클라이언트] 4 + 4바이트 보냈습니다.
[TCP 클라이언트] 4 + 4바이트 보냈습니다.
[TCP 클라이언트] 4 + 4바이트 보냈습니다.
D:\OneDrive - 강원대학교과제\3학년 2학기\네트워크 프로그래밍\C++ Projects\TCPClient_FixedVariable\Debug\TCPClient_FixedVariable.exe(프로세스 24568개)이(가) 종료되었습니다(코드: 0x0).
이 창을 닫으려면 아무 키나 누르세요...
```

3. PC에서의 고정 IP address 설정

인터넷 프로토콜 버전 4(TCP/IPv4) 속성

일반

네트워크가 IP 자동 설정 기능을 지원하면 IP 설정이 자동으로 할당되도록 할 수 있습니다. 지원하지 않으면, 네트워크 관리자에게 적절한 IP 설정값을 문의해야 합니다.

☐ 자동으로 IP 주소 받기(O)

☒ 다음 IP 주소 사용(S)

IP 주소(I): 192 . 168 . 7 . 1

서브넷 마스크(U): 255 . 255 . 255 . 0

기본 게이트웨이(D): . . .

☐ 자동으로 DNS 서버 주소 받기(B)

☒ 다음 DNS 서버 주소 사용(E)

기본 설정 DNS 서버(P): . . .

보조 DNS 서버(A): . . .

☐ 끝낼 때 설정 유효성 검사(L)

고급(V)...

확인 취소

4. ping 테스트 결과를 제출하시오.

Case 1. 192.168.7.4간 ICMP 통신 결과

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

새로운 기능 및 개선 사항에 대한 최신 PowerShell을 설치하세요! https://aka.ms/PSWindows

PS C:\Users\daeha> ping 192.168.7.4

Ping 192.168.7.4 32바이트 데이터 사용:
192.168.7.4의 응답: 바이트=32 시간<1ms TTL=128
192.168.7.4의 응답: 바이트=32 시간<1ms TTL=128
192.168.7.4의 응답: 바이트=32 시간<1ms TTL=128
192.168.7.4의 응답: 바이트=32 시간<1ms TTL=128

192.168.7.4에 대한 Ping 통계:
    패킷: 보냄 = 4, 받음 = 4, 손실 = 0 (0% 손실),
    왕복 시간(밀리초):
        최소 = 0ms, 최대 = 0ms, 평균 = 0ms
PS C:\Users\daeha> |
```

Case 2. 192.168.7.10간 ICMP 통신 결과

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

새로운 기능 및 개선 사항에 대한 최신 PowerShell을 설치하세요! https://aka.ms/PSWindows

PS C:\Users\daeha> ping 192.168.7.10

Ping 192.168.7.10 32바이트 데이터 사용:
192.168.7.10의 응답: 바이트=32 시간<1ms TTL=128
192.168.7.10의 응답: 바이트=32 시간<1ms TTL=128
192.168.7.10의 응답: 바이트=32 시간<1ms TTL=128
192.168.7.10의 응답: 바이트=32 시간<1ms TTL=128

192.168.7.10에 대한 Ping 통계:
    패킷: 보냄 = 4, 받음 = 4, 손실 = 0 (0% 손실),
    왕복 시간(밀리초):
        최소 = 0ms, 최대 = 0ms, 평균 = 0ms
PS C:\Users\daeha> |
```

5. Wireshark에서 캡처한 이미지를 제출하시오.

Server-192.168.7.4

161	39.065493	192.168.7.4	192.168.7.1	TCP	66 60014 → 9000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
162	39.065575	192.168.7.1	192.168.7.4	TCP	66 9000 → 60014 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM=1
163	39.066451	192.168.7.4	192.168.7.1	TCP	60 60014 → 9000 [ACK] Seq=1 Ack=1 Win=1051136 Len=0
164	39.066451	192.168.7.4	192.168.7.1	TCP	60 60014 → 9000 [PSH, ACK] Seq=1 Ack=1 Win=1051136 Len=4
165	39.066744	192.168.7.4	192.168.7.1	TCP	132 60014 → 9000 [FIN, PSH, ACK] Seq=5 Ack=1 Win=1051136 Len=78
166	39.066759	192.168.7.1	192.168.7.4	TCP	54 9000 → 60014 [ACK] Seq=1 Ack=84 Win=1049600 Len=0
167	39.068343	192.168.7.1	192.168.7.4	TCP	54 9000 → 60014 [FIN, ACK] Seq=1 Ack=84 Win=1049600 Len=0
168	39.068655	192.168.7.4	192.168.7.1	TCP	60 60014 → 9000 [ACK] Seq=84 Ack=2 Win=1051136 Len=0

Server-192.168.7.10

33	32.117639	192.168.7.10	192.168.7.1	TCP	66 61659 → 9000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
34	32.117731	192.168.7.1	192.168.7.10	TCP	66 9000 → 61659 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM=1
35	32.118282	192.168.7.10	192.168.7.1	TCP	60 61659 → 9000 [ACK] Seq=1 Ack=1 Win=1051136 Len=0
36	32.118282	192.168.7.10	192.168.7.1	TCP	60 61659 → 9000 [PSH, ACK] Seq=1 Ack=1 Win=1051136 Len=4
37	32.118797	192.168.7.10	192.168.7.1	TCP	132 61659 → 9000 [FIN, PSH, ACK] Seq=5 Ack=1 Win=1051136 Len=78
38	32.118821	192.168.7.1	192.168.7.10	TCP	54 9000 → 61659 [ACK] Seq=1 Ack=84 Win=1049600 Len=0
39	32.119239	192.168.7.1	192.168.7.10	TCP	54 9000 → 61659 [FIN, ACK] Seq=1 Ack=84 Win=1049600 Len=0
40	32.119537	192.168.7.10	192.168.7.1	TCP	60 61659 → 9000 [ACK] Seq=84 Ack=2 Win=1051136 Len=0

Client-192.168.7.4

898	438.575436	192.168.7.1	192.168.7.4	TCP	66 14258 → 9000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
899	438.576385	192.168.7.4	192.168.7.1	TCP	66 9000 → 14258 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM=1
900	438.576467	192.168.7.1	192.168.7.4	TCP	54 14258 → 9000 [ACK] Seq=1 Ack=1 Win=131328 Len=0
901	438.576497	192.168.7.1	192.168.7.4	TCP	58 14258 → 9000 [PSH, ACK] Seq=1 Ack=1 Win=131328 Len=4
902	438.576903	192.168.7.1	192.168.7.4	TCP	66 14258 → 9000 [FIN, PSH, ACK] Seq=5 Ack=1 Win=131328 Len=12
903	438.577485	192.168.7.4	192.168.7.1	TCP	60 9000 → 14258 [ACK] Seq=1 Ack=18 Win=131328 Len=0
904	438.579085	192.168.7.4	192.168.7.1	TCP	60 9000 → 14258 [FIN, ACK] Seq=1 Ack=18 Win=131328 Len=0
905	438.579185	192.168.7.1	192.168.7.4	TCP	54 14258 → 9000 [ACK] Seq=18 Ack=2 Win=131328 Len=0

Client-192.168.7.10

11	3.120556	192.168.7.1	192.168.7.10	TCP	66 14608 → 9000 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
12	3.121067	192.168.7.10	192.168.7.1	TCP	66 9000 → 14608 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM=1
13	3.121125	192.168.7.1	192.168.7.10	TCP	54 14608 → 9000 [ACK] Seq=1 Ack=1 Win=131328 Len=0
14	3.121171	192.168.7.1	192.168.7.10	TCP	58 14608 → 9000 [PSH, ACK] Seq=1 Ack=1 Win=131328 Len=4
15	3.121513	192.168.7.1	192.168.7.10	TCP	66 14608 → 9000 [FIN, PSH, ACK] Seq=5 Ack=1 Win=131328 Len=12
16	3.121742	192.168.7.10	192.168.7.1	TCP	60 9000 → 14608 [ACK] Seq=1 Ack=18 Win=1051136 Len=0
17	3.123307	192.168.7.10	192.168.7.1	TCP	60 9000 → 14608 [FIN, ACK] Seq=1 Ack=18 Win=1051136 Len=0
18	3.123369	192.168.7.1	192.168.7.10	TCP	54 14608 → 9000 [ACK] Seq=18 Ack=2 Win=131328 Len=0