

Only connect

Óðinn Karl Skúlason

First test basic network connectivity. Use the ping command

```
[onni2@onniArch ~]$ ping 130.208.246.98
PING 130.208.246.98 (130.208.246.98) 56(84) bytes of data:
64 bytes from 130.208.246.98: icmp_seq=1 ttl=58 time=6.17 ms
64 bytes from 130.208.246.98: icmp_seq=2 ttl=58 time=5.19 ms
64 bytes from 130.208.246.98: icmp_seq=3 ttl=58 time=6.20 ms
64 bytes from 130.208.246.98: icmp_seq=4 ttl=58 time=6.43 ms
64 bytes from 130.208.246.98: icmp_seq=5 ttl=58 time=5.96 ms
64 bytes from 130.208.246.98: icmp_seq=6 ttl=58 time=8.42 ms
64 bytes from 130.208.246.98: icmp_seq=7 ttl=58 time=4.23 ms
64 bytes from 130.208.246.98: icmp_seq=8 ttl=58 time=4.76 ms
64 bytes from 130.208.246.98: icmp_seq=9 ttl=58 time=11.8 ms
64 bytes from 130.208.246.98: icmp_seq=10 ttl=58 time=6.23 ms
64 bytes from 130.208.246.98: icmp_seq=11 ttl=58 time=5.52 ms
64 bytes from 130.208.246.98: icmp_seq=12 ttl=58 time=7.87 ms
```

Now check for open ports. Using the nmap command:

```
[onni2@onniArch ~]$ sudo nmap -sS -p 4000-4100 130.208.246.98
Starting Nmap 7.97 ( https://nmap.org ) at 2025-08-20 09:11 +0000
Nmap scan report for 130.208.246.98
Host is up (0.040s latency).
Not shown: 96 closed tcp ports (reset)
PORT      STATE SERVICE
4000/tcp  open  remoteanything
4001/tcp  open  newoak
4002/tcp  open  mlchat-proxy
4008/tcp  open  netcheque
4098/tcp  open  drmsfsd
Nmap done: 1 IP address (1 host up) scanned in 1.76 seconds
```

b) Sá too stafsetningarvillu eftir á

```
System information as of Thu Aug 21 01:22:44 PM UTC 2025
System load: 0.0 Processes: 203
Usage of /home: 0.3% of 186.89GB Users logged in: 4
Memory usage: 3% IPv4 address for enp6s18: 130.208.246.98
Swap usage: 0%

* Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
just raised the bar for easy, resilient and secure K8s cluster deployment.

https://ubuntu.com/engage/secure-kubernetes-at-the-edge

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

1 additional security update can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

*** System restart required ***
Last login: Thu Aug 21 13:20:32 2025 from 130.208.240.12
bdinns24@tsam:~$ ncat -ls 4050
Ncat: -l and -s are incompatible. Specify the address and port to bind to.
bdinns24@tsam:~$ ncat -ln 4050
hello
whaaaaa!
you are my best friend now
you twoo

Before we start programming, let's just check the status
to program with. For reasons that you will
that should ever be taken for granted. For
cybersecurity lab with IP address 130.208.246.98
(a) (2 points) First test basic network connectivity.
ping 130.208.246.98
Now check for open ports. Using the
nmap -sS 130.208.246.98
list all open TCP ports on the TSAM
of these two commands.
(b) (2 points) Now open two terminals
like you would a host to connect
ncat -ln 4050. In the other terminal
ncat 130.208.246.98 4050 and type
may use any port in the range 4000-4100
currently used by other people. )

We will use port 4000 as an example, but
[onni2@onniArch ~]$ ncat 130.208.246.98 4050
hello
whaaaaa! local computer, run the command
you are my best friend now return. (You
twoo. I will have shown you any that are
```

c)

```
sudo tcpdump -i wlo1 port 4050

[onni2@onniArch ~]$ sudo tcpdump -i wlo1 port 4050
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on wlo1, link-type EN10MB (Ethernet), snapshot length 262144 bytes
14:47:03.034263 IP onniArch.39342 > 130.208.246.98.cisco-wafs: Flags [P.], seq 910998246:910998289, ack 486235401, win 502, options [nop,nop,TS val 614278784 ecr 3829505988], length 43
14:47:03.038866 IP 130.208.246.98.cisco-wafs > onniArch.39342: Flags [.], ack 43, win 509, options [nop,nop,TS val 3829534568 ecr 614278784], length 0

server and provide a screen shot of the output

We will use port 4000 as an example, but
4000-4100 that did not get reported by nmap
to the TSAM server and run the command
al on your local computer, run the command
e a few characters followed by return. (You
4100, nmap will have shown you any that are

2 of 4

ncat 130.208.246.98 4050

[onni2@onniArch ~]$ ncat 130.208.246.98 4050
Remember to check for network connectivity
```

d) here is a screen shot of my local terminal sending the commands, then ssh tsam server sening it back using ncat -ln 4050 /bin/bash
And last is the tcpdump on port 4050 using sudo tcpdump -i wlo1 port 4050

```
[onni2@onniArch ~]$ ncat 130.208.246.98 4050
Remember to check for network connectivity
[onni2@onniArch ~]$ ncat 130.208.246.98 4050
SYS who
whoami
odinns24
hostname
tsam
```

```
odinns24@tsam:~$ ncat -ln 4050 /bin/bash
Ncat: Invalid port number "/bin/bash". QUITTING
odinns24@tsam:~$ ncat -ln 4050 -e /bin/bash
/bin/bash: line 1: SYS: command not found
```

```
14:51:36.095091 IP onniArch.35406 > 130.208.246.98.cisco-wafs: Flags [P.], seq 9:16, ack 1, win 502, options [nop,nop,TS val 614551844 ecr 3829788346], length 7
14:51:36.099517 IP 130.208.246.98.cisco-wafs > onniArch.35406: Flags [.], ack 16, win 509, options [nop,nop,TS val 3829807631 ecr 614551844], length 0
14:51:36.099519 IP 130.208.246.98.cisco-wafs > onniArch.35406: Flags [P.], seq 1:10, ack 16, win 509, options [nop,nop,TS val 3829807633 ecr 614551844], length 9
14:51:36.099583 IP onniArch.35406 > 130.208.246.98.cisco-wafs: Flags [.], ack 10, win 502, options [nop,nop,TS val 614551849 ecr 3829807633], length 0
14:51:44.016430 IP onniArch.35406 > 130.208.246.98.cisco-wafs: Flags [P.], seq 16:25, ack 10, win 502, options [nop,nop,TS val 614559766 ecr 3829807633], length 9
14:51:44.021422 IP 130.208.246.98.cisco-wafs > onniArch.35406: Flags [P.], seq 10:15, ack 25, win 509, options [nop,nop,TS val 3829815554 ecr 614559766], length 5
14:51:44.021463 IP onniArch.35406 > 130.208.246.98.cisco-wafs: Flags [.], ack 15, win 502, options [nop,nop,TS val 614559771 ecr 3829815554], length 0
```

2

a)first image is ncat, then tcpdump, then server

```
[onni2@onniArch ~]$ ncat 127.0.0.1 5000
^C
[onni2@onniArch ~]$ ncat 127.0.0.1 5000
SYS who
SYS eg
SYS ls
SYS etc

[onni2@onniArch ~]$ sudo tcpdump -AX -i lo host 127.0.0.1 and port 5000
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on lo, link-type EN10MB (Ethernet), snapshot length 262144 bytes
18:02:15.429961 IP localhost.52266 > localhost.complex-main: Flags [P.], seq 629649849:629649857, ack 565564899, win 512, options [nop,nop,TS val 399522944 ecr 399367053], length 8
 0x0000: 4500 003c c228 4000 4006 7a91 7f00 0001  E...@.z....
 0x0010: 7f00 0001 cc2a 1388 2587 b1b9 21b5 d5e3  ....%.
 0x0020: 8018 0200 fe2f 0000 0101 000a 17d0 3c80  ....@.....
 0x0030: 17cd dbd8 5359 5320 7768 6f0a  ....SYS:who.
18:02:15.429994 IP localhost.complex-main > localhost.52266: Flags [.], ack 8, win 512, options [nop,nop,TS val 399522944 ecr 399522944], length 0
 0x0000: 4500 0034 39ad 4000 4006 0315 7f00 0001  E..49.@.....
 0x0010: 7f00 0001 1388 cc2a 21b5 d5e3 2587 b1c1  ....*!...%.
 0x0020: 8018 0200 fe28 0000 0101 000a 17d0 3c80  ....(.....
 0x0030: 17d0 3c80  ....
18:02:42.845974 IP localhost.52266 > localhost.complex-main: Flags [P.], seq 8:15, ack 1, win 512, options [nop,nop,TS val 399550360 ecr 399522944], length 7
 0x0000: 4500 003b c229 4000 4006 7a91 7f00 0001  E...;).@.z....
 0x0010: 7f00 0001 cc2a 1388 2587 b1c1 21b5 d5e3  ....%.
 0x0020: 8018 0200 fe2f 0000 0101 000a 17d0 a798  ..../.....
 0x0030: 17d0 3c80 5359 5320 6567 0a  ....SYS:eg.
18:02:42.846003 IP localhost.complex-main > localhost.52266: Flags [.], ack 15, win 512, options [nop,nop,TS val 399550360 ecr 399550360], length 0
 0x0000: 4500 0034 39ae 4000 4006 0314 7f00 0001  E..49.@.....
 0x0010: 7f00 0001 1388 cc2a 21b5 d5e3 2587 b1c8  ....*!...%.
 0x0020: 8018 0200 fe28 0000 0101 000a 17d0 a798  ....(.....
 0x0030: 17d0 a798  ....
18:02:58.784762 IP localhost.52266 > localhost.complex-main: Flags [P.], seq 15:22, ack 1, win 512, options [nop,nop,TS val 399566299 ecr 399550360], length 7
 0x0000: 4500 003b c22a 4000 4006 7a90 7f00 0001  E...;*.@.z....
 0x0010: 7f00 0001 cc2a 1388 2587 b1c8 21b5 d5e3  ....%.
 0x0020: 8018 0200 fe2f 0000 0101 000a 17d0 e5db  ..../.....
 0x0030: 17d0 a798 5359 5320 6c73 0a  ....SYS:ls.
18:02:58.784794 IP localhost.complex-main > localhost.52266: Flags [.], ack 22, win 512, options [nop,nop,TS val 399566299 ecr 399566299], length 0
 0x0000: 4500 0034 39af 4000 4006 0313 7f00 0001  E..49.@.....
 0x0010: 7f00 0001 1388 cc2a 21b5 d5e3 2587 b1cf  ....*!...%.
 0x0020: 8018 0200 fe28 0000 0101 000a 17d0 e5db  ....(.....
 0x0030: 17d0 e5db  ....
18:03:16.258583 IP localhost.52266 > localhost.complex-main: Flags [P.], seq 22:30, ack 1, win 512, options [nop,nop,TS val 399583773 ecr 399566299], length 8
 0x0000: 4500 003c c22b 4000 4006 7a8e 7f00 0001  E...<*.@.z....
 0x0010: 7f00 0001 cc2a 1388 2587 b1cf 21b5 d5e3  ....%.
 0x0020: 8018 0200 fe30 0000 0101 000a 17d1 2a1d  ....@.....
 0x0030: 17d0 e5db 5359 5320 6574 630a  ....SYS:etc.
18:03:16.258612 IP localhost.complex-main > localhost.52266: Flags [.], ack 30, win 512, options [nop,nop,TS val 399583773 ecr 399583773], length 0
 0x0000: 4500 0034 39b0 4000 4006 0312 7f00 0001  E..49.@.....
 0x0010: 7f00 0001 1388 cc2a 21b5 d5e3 2587 b1d7  ....*!...%.
 0x0020: 8018 0200 fe28 0000 0101 000a 17d1 2a1d  ....(.....
 0x0030: 17d1 2a1d  ....

Usage: server <port>
[onni2@onniArch P1]$ ./server 5000
Listening on port: 5000
^C
[onni2@onniArch P1]$ ./server 5000
Listening on port: 5000
Client connected on server
Client closed connection: 4
^C
[onni2@onniArch P1]$ ./server 5000
Listening on port: 5000
Client connected on server
SYS who

onni2      tty1          Aug 14 16:56
onni2      pts/0             Aug 14 16:56 (:1)
onni2      pts/4             Aug 19 18:01
SYS eg

sh: line 1: eg: command not found
SYS ls

a.out server server.cpp
SYS etc

sh: line 1: etc: command not found
```

3. picture of server receiving, then tcpdump from that. Then last my client code. I added the while loop after for number 4.

```
[onni2@onniArch P1]$ ./server 5000
Listening on port: 5000
Client connected on server
SYS who
onni2    tty1          Aug 14 16:56
onni2    pts/0          Aug 14 16:56 (
onni2    pts/4          Aug 19 18:27
Client closed connection: 4
```

```
onni2@onniArch ~]$ sudo tcpdump -AX -i lo host 127.0.0.1 and port 5000
sudo! password for onni2:
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on lo, link-type EN10MB (Ethernet), snapshot length 262144 bytes
8:27:25.620242 IP localhost.37764 > localhost.complex-main: Flags [S], seq 2512123701, win 65495, options [mss 65495,sackOK,TS val 401033135 ecr 0,nop,wscale 7], length 0
0x0000: 4500 003c cb40 4000 4006 7179 7f00 0001 E..<.@.0.qy....
0x0010: 7f00 0001 9384 1388 95bb f735 0000 0000 .....5....
0x0020: a002 ffd7 fe30 0000 0204 ffd7 0402 080a .....0.....
0x0030: 17e7 47af 0000 0000 0103 0307 ..G.....
8:27:25.620278 IP localhost.complex-main > localhost.37764: Flags [S.], seq 2284338064, ack 2512123702, win 65483, options [mss 65495,sackOK,TS val 401033135 ecr 401033135,nop,wscale 7], length 0
0x0000: 4500 003c 0000 4000 4006 3c3a 7f00 0001 E..<.@.0.<.....
0x0010: 7f00 0001 1388 9384 8828 3b90 95bb f736 .....6....
0x0020: a012 ffd7 fe30 0000 0204 ffd7 0402 080a .....0.....
0x0030: 17e7 47af 17e7 47af 0103 0307 ..G.....
8:27:25.620308 IP localhost.37764 > localhost.complex-main: Flags [.], ack 1, win 512, options [nop,nop,TS val 401033135 ecr 401033135], length 0
0x0000: 4500 0034 cb41 4000 4006 7180 7f00 0001 E..4.A0.0.q.....
0x0010: 7f00 0001 9384 1388 95bb f736 8828 3b91 .....6.().
0x0020: 8010 0200 fe28 0000 0101 080a 17e7 47af .....G.....
0x0030: 17e7 47af ..G.....
8:27:32.666266 IP localhost.37764 > localhost.complex-main: Flags [P.], seq 1:8, ack 1, win 512, options [nop,nop,TS val 401040181 ecr 401033135], length 7
0x0000: 4500 003b cb42 4000 4006 7178 7f00 0001 E..;.80.0.qx....
0x0010: 7f00 0001 9384 1388 95bb f736 8828 3b91 .....6.().
0x0020: 8010 0200 fe2f 0000 0101 080a 17e7 6335 ...../.....c5
0x0030: 17e7 47af 5359 5320 7768 6f ..G.SYS.who
8:27:32.666302 IP localhost.complex-main > localhost.37764: Flags [.], ack 8, win 512, options [nop,nop,TS val 401040181 ecr 401040181], length 0
0x0000: 4500 0034 835a 4000 4006 b967 7f00 0001 E..4.Z0.0.g.....
0x0010: 7f00 0001 1388 9384 8828 3b91 95bb f73d .....(;)=
0x0020: 8010 0200 fe28 0000 0101 080a 17e7 6335 ...../.....c5
0x0030: 17e7 6335 .....c5
8:27:32.666339 IP localhost.37764 > localhost.complex-main: Flags [F.], seq 8, ack 1, win 512, options [nop,nop,TS val 401040181 ecr 401040181], length 0
0x0000: 4500 0034 cb43 4000 4006 717e 7f00 0001 E..4.C0.0.q~....
0x0010: 7f00 0001 9384 1388 95bb f73d 8828 3b91 .....;.
0x0020: 8011 0200 fe28 0000 0101 080a 17e7 6335 ...../.....c5
0x0030: 17e7 6335 .....c5
8:27:32.670725 IP localhost.complex-main > localhost.37764: Flags [F.], seq 1, ack 9, win 512, options [nop,nop,TS val 401040185 ecr 401040181], length 0
0x0000: 4500 0034 835b 4000 4006 b966 7f00 0001 E..4.[0..f....
0x0010: 7f00 0001 1388 9384 8828 3b91 95bb f73e .....(;)>
0x0020: 8011 0200 fe28 0000 0101 080a 17e7 6339 ...../.....c9
0x0030: 17e7 6335 .....c5
8:27:32.670774 IP localhost.37764 > localhost.complex-main: Flags [.], ack 2, win 512, options [nop,nop,TS val 401040185 ecr 401040185], length 0
0x0000: 4500 0034 cb44 4000 4006 717d 7f00 0001 E..4.D0.0.q....
0x0010: 7f00 0001 9384 1388 95bb f73e 8828 3b92 .....>().
0x0020: 8010 0200 fe28 0000 0101 080a 17e7 6339 ...../.....c9
0x0030: 17e7 6339 .....c9
```

```
[onni2@onniArch P1]$ tcpdump -AX -i lo host 127.0.0.1 and port 5000
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on lo, link-type EN10MB (Ethernet), snapshot length 262144 bytes
8:27:32.670725 IP localhost.37764 > localhost.complex-main: Flags [F.], seq 1, ack 9, win 512, options [nop,nop,TS val 401040185 ecr 401040181], length 0
0x0000: 4500 0034 835b 4000 4006 b966 7f00 0001 E..4.[0..f....
0x0010: 7f00 0001 1388 9384 8828 3b91 95bb f73e .....(;)>
0x0020: 8011 0200 fe28 0000 0101 080a 17e7 6339 ...../.....c9
0x0030: 17e7 6335 .....c5
```

G+ client.cpp > ...

```
1  #include <cstring>
2  #include <iostream>
3  #include <netinet/in.h>
4  #include <sys/socket.h>
5  #include <unistd.h>
6
7  int main()
8  {
9      const char* server_ip = "127.0.0.1";
10     const int server_port = 5000;
11     // creating socket
12     int clientSocket = socket(AF_INET, SOCK_STREAM, 0);
13     if (clientSocket < 0) {
14         std::cerr << "Error: could not create socket\n";
15         return 1;
16     }
17     // specifying address
18     sockaddr_in serverAddress;
19     serverAddress.sin_family = AF_INET;
20     serverAddress.sin_port = htons(5000);
21     serverAddress.sin_addr.s_addr = INADDR_ANY;
22
23     // sending connection request
24     connect(clientSocket, (struct sockaddr*)&serverAddress,
25            sizeof(serverAddress));
26
27     // sending data
28     while (true) {
29         std::string message;
30         std::cout << "Enter command: ";
31         std::getline(std::cin, message);
32         if (message == "exit") {
33             std::cout << "Exiting client.\n";
34             break;
35         }
36         send(clientSocket, message.c_str(), message.size(), 0);
37     }
38
39     // closing socket
40     close(clientSocket);
41
42     return 0;
43 }
```

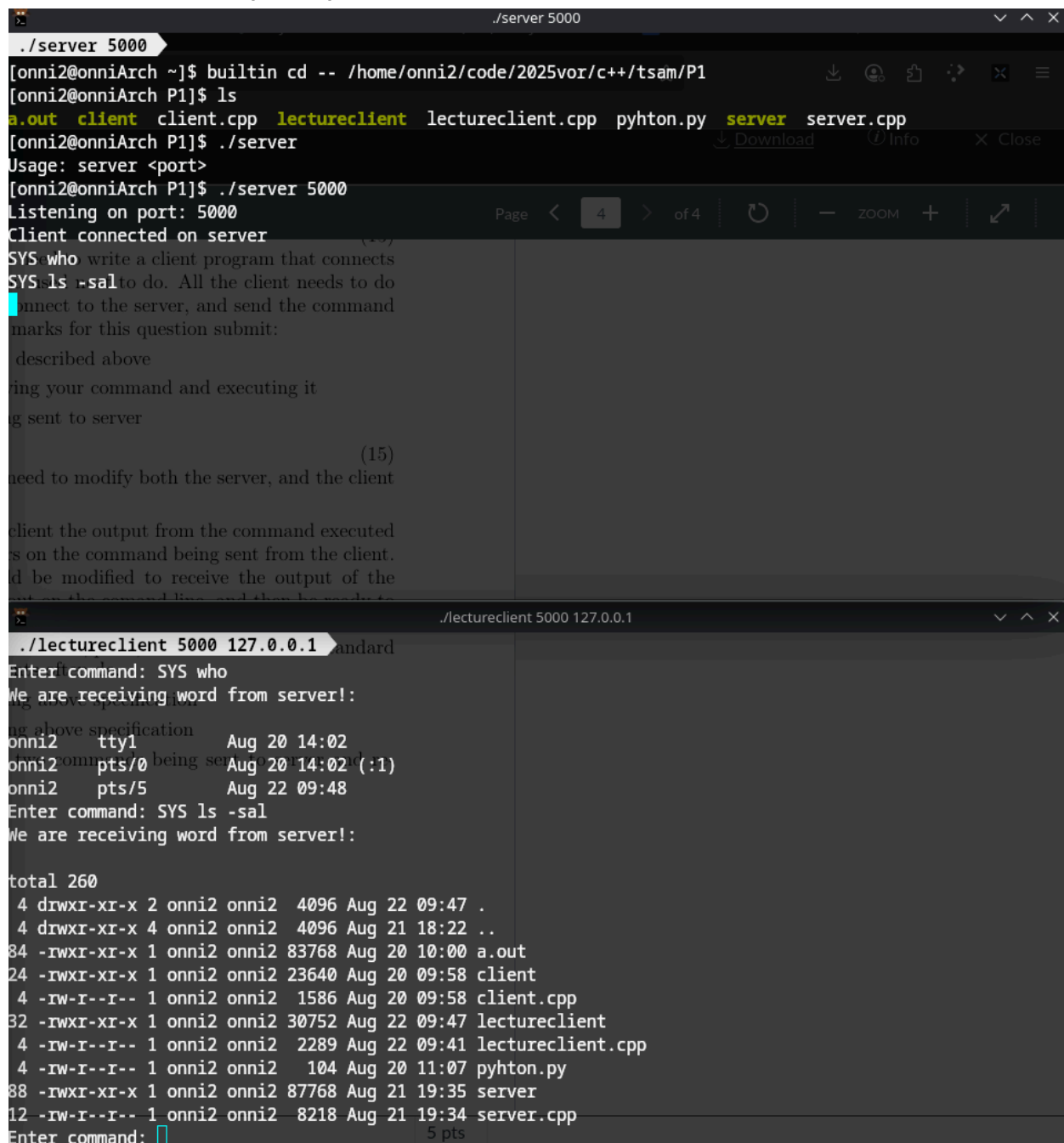
Updated code from teacher with not hardcoded ip

```
#include <sys/socket.h>
#include <iostream>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
int main(int argc, char *argv[]) {
    int sock;
    int port = 4000; //TODO: make this a command line argument
    char *ip_string = argv[1]; //TODO check for right number of arguments
    //create a socket
    if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        std::cerr << "Error: could not create socket\n";
        return 1;
    }
    //construct a destination address
    struct sockaddr_in server_addr;
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(port);
    if (inet_pton(AF_INET, ip_string, &server_addr.sin_addr) != 1) {
        std::cout << "ip_address is weird" << std::endl;
        exit(0);
    }
    if (connect(sock, (const sockaddr*)&server_addr, sizeof(server_addr)) < 0) {
        perror("connect failed");
        exit(0);
    };
    std::string message = "Hello, server!";

    int n = send(sock, message.c_str(), message.length(), 0);
    if (n < message.length()) {
        std::cerr << "Error: could not send complete message\n";
    }
    if (close(sock) < 0) {
        std::cerr << "Error: could not close socket\n";
        return 1;
    }
}
```


4. first pictures are of server and client talking and getting the output from server onto client screen

Below them are tcpdump from that transaction



```
./server 5000
[onni2@onniArch ~]$ builtin cd -- /home/onni2/code/2025vor/c++/tsam/P1
[onni2@onniArch P1]$ ls
a.out client client.cpp lectureclient lectureclient.cpp pyhton.py server server.cpp
[onni2@onniArch P1]$ ./server
Usage: server <port>
[onni2@onniArch P1]$ ./server 5000
Listening on port: 5000
Client connected on server
SYS who write a client program that connects
SYS ls -sal to do. All the client needs to do
connect to the server, and send the command
marks for this question submit:
described above
ing your command and executing it
g sent to server
(15)
need to modify both the server, and the client
client the output from the command executed
s on the command being sent from the client.
d be modified to receive the output of the
out on the server line, and then be ready to

./lectureclient 5000 127.0.0.1
Enter command: SYS who
We are receiving word from server!:
g above specification
onni2 tty1 Aug 20 14:02
onni2 pts/0 being se Aug 20 14:02(:1)
onni2 pts/5 Aug 22 09:48
Enter command: SYS ls -sal
We are receiving word from server!:
total 260
4 drwxr-xr-x 2 onni2 onni2 4096 Aug 22 09:47 .
4 drwxr-xr-x 4 onni2 onni2 4096 Aug 21 18:22 ..
84 -rwxr-xr-x 1 onni2 onni2 83768 Aug 20 10:00 a.out
24 -rwxr-xr-x 1 onni2 onni2 23640 Aug 20 09:58 client
4 -rw-r--r-- 1 onni2 onni2 1586 Aug 20 09:58 client.cpp
32 -rwxr-xr-x 1 onni2 onni2 30752 Aug 22 09:47 lectureclient
4 -rw-r--r-- 1 onni2 onni2 2289 Aug 22 09:41 lectureclient.cpp
4 -rw-r--r-- 1 onni2 onni2 104 Aug 20 11:07 pyhton.py
88 -rwxr-xr-x 1 onni2 onni2 87768 Aug 21 19:35 server
12 -rw-r--r-- 1 onni2 onni2 8218 Aug 21 19:34 server.cpp
Enter command: 5 pts
```



```

sudo tcpdump -AX -i lo host 127.0.0.1 and port 5000
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on lo, link-type EN10MB (Ethernet), snapshot length 262144 bytes
09:48:26.901627 IP localhost.49256 > localhost.complex-main: Flags [P.], seq 2041186841:2041186848, ack 3
71388503, win 512, options [nop,nop,TS val 457839803 ecr 457767997], length 7
  0x0000: 4500 003b fb51 4000 4006 4169 7f00 0001 E...Q@.Ai....
  0x0010: 7f00 0001 c068 1388 79aa 0a19 1622 f057 .....h..y....".W
  0x0020: 8018 0200 fe2f 0000 0101 080a 1b4a 14bb ...../.....J..
  0x0030: 1b48 fc3d 5359 5320 7768 6f Normal text .H.=SYS.who
09:48:26.901649 IP localhost.complex-main > localhost.49256: Flags [.] , ack 7, win 512, options [nop,nop,
TS val 457839803 ecr 457839803], length 0
  0x0000: 4500 0034 a8bd 4000 4006 9404 7f00 0001 E..4..@.....
  0x0010: 7f00 0001 1388 c068 1622 f057 79aa 0a20 .....h..".Wy...
  0x0020: 8010 0200 fe28 0000 0101 080a 1b4a 14bb .....(.....J...
  0x0030: 1b4a 14bb .J..
09:48:26.913928 IP localhost.complex-main > localhost.49256: Flags [P.], seq 1:111, ack 7, win 512, optio
ns [nop,nop,TS val 457839815 ecr 457839803], length 110
  0x0000: 4500 00a2 a8be 4000 4006 9395 7f00 0001 E.....@.....
  0x0010: 7f00 0001 1388 c068 1622 f057 79aa 0a20 .....h..".Wy...
  0x0020: 8018 0200 fe96 0000 0101 080a 1b4a 14c7 .....J...
  0x0030: 1b4a 14bb 6f6e 6e69 3220 2020 2074 7479 .J..onni2....tty
  0x0040: 3120 2020 2020 2020 2020 4175 6720 3230 1.....Aug.20
  0x0050: 2031 343a 3032 0a6f 6e6e 6932 2020 2020 .14:02.onni2...
  0x0060: 7074 732f 3020 2020 2020 2020 2041 7567 pts/0.....Aug
  0x0070: 2032 3020 3134 3a30 3220 283a 3129 0a6f .20.14:02.(.1).o
  0x0080: 6e6e 6932 2020 2020 7074 732f 3520 2020 nni2....pts/5...
  0x0090: 2020 2020 2041 7567 2032 3220 3039 3a34 .....Aug.22.09:4
  0x00a0: 380a 8.
09:48:26.913946 IP localhost.49256 > localhost.complex-main: Flags [.] , ack 111, win 512, options [nop,no
p,TS val 457839815 ecr 457839815], length 0
  0x0000: 4500 0034 fb52 4000 4006 416f 7f00 0001 E..4.R@.Ao...
  0x0010: 7f00 0001 c068 1388 79aa 0a20 1622 f0c5 .....h..y....".
  0x0020: 8010 0200 fe28 0000 0101 080a 1b4a 14c7 .....(.....J..
  0x0030: 1b4a 14c7 .J..
09:48:53.198702 IP localhost.49256 > localhost.complex-main: Flags [P.], seq 7:18, ack 111, win 512, opti
ons [nop,nop,TS val 457866100 ecr 457839815], length 11
  0x0000: 4500 003f fb53 4000 4006 4163 7f00 0001 E..?.S@.Ac....
  0x0010: 7f00 0001 c068 1388 79aa 0a20 1622 f0c5 .....h..y....".
  0x0020: 8018 0200 fe33 0000 0101 080a 1b4a 7b74 .....3.....J{t
  0x0030: 1b4a 14c7 5359 5320 6c73 202d 7361 6c .J..SYS.ls.-sal
09:48:53.203912 IP localhost.complex-main > localhost.49256: Flags [P.], seq 111:680, ack 18, win 512, op
tions [nop,nop,TS val 457866105 ecr 457866100], length 569
  0x0000: 4500 026d a8bf 4000 4006 91c9 7f00 0001 E..m..@.....
  0x0010: 7f00 0001 1388 c068 1622 f0c5 79aa 0a2b .....h..".y..+
  0x0020: 8018 0200 0062 0000 0101 080a 1b4a 7b79 .....b.....J{y
  0x0030: 1b4a 7b74 746f 7461 6c20 3236 300a 2034 .J{ttotal.260..4
  0x0040: 2064 7277 7872 2d78 722d 7820 3220 6f6e .drwxr-xr-x.2.on
  0x0050: 6e69 3220 6f6e 6e69 3220 2034 3039 3620 ni2.onni2..4096.

```

```
sudo tcpdump -AX -i lo host 127.0.0.1 and port 5000
[...]
```

Client code

```
#include <sys/socket.h>
#include <iostream>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <sstream>
#include <vector>

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

```

int sock;
char buffer[1025];
if (argc != 3){
    std::cout << "Usage: " << argv[0] << "<port> <ip>" << std::endl;
    return 1;
}
int port = atoi(argv[1]); //TODO: make this a command line argument
char *ip_string = argv[2]; //TODO check for right number of arguments
//create a socket
if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
    std::cerr << "Error: could not create socket\n";
    return 1;
}
//construct a destination address
struct sockaddr_in server_addr;
server_addr.sin_family = AF_INET;
server_addr.sin_port = htons(port);
if (inet_pton(AF_INET, ip_string, &server_addr.sin_addr) != 1){
    std::cout << "ip_address is weird" << std::endl;
    exit(0);
}
if (connect(sock, (const sockaddr*)&server_addr, sizeof(server_addr)) <
0) {
    perror("connect failed");
    exit(0);
};
std::string message;
while (!(message == "exit") ){
    std::cout << "Enter command: ";
    std::getline(std::cin, message);

    int n = send(sock, message.c_str(), message.length(), 0);
    if (n < message.length()) {
        std::cerr << "Error: could not send complete message\n";
    }
    int receive = recv(sock, buffer, sizeof(buffer), 0);
    if (receive < 0){
        std::cout << "ERROR" << std::endl;
    } else if (receive == 0) {

```

```

        std::cout << "Lost connection, trying to reconnect" <<
std::endl;
        if (connect(sock, (const sockaddr*)&server_addr,
sizeof(server_addr)) < 0) {
            perror("connect failed");
            exit(0);
        };
    } else {
        buffer[receive] = '\0';
        std::stringstream stream(buffer);
        std::string line;
        std::cout << "We are receiving word from server!:\n" <<
std::endl;
        while (std::getline(stream, line)) {
            std::cout << line << std::endl;
        }

    }
}
if (close(sock) < 0) {
    std::cerr << "Error: could not close socket\n";
    return 1;
}
}

```

Server code

```

//
// Simple server for TSAM-409 Assignment 1
//
// Compile: g++ -Wall -std=c++11 server.cpp
//
// Command line: ./server 5000
//
// Author: Jacky Mallett (jacky@ru.is)
//         Stephan Schiffel (stephans@ru.is)
//
#include <stdio.h>
#include <errno.h>

```

```

#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netinet/ip.h>
#include <netinet/tcp.h>
#include <netdb.h>
#include <arpa/inet.h>
#include <string.h>
#include <algorithm>
#include <map>
#include <vector>
//i added array
#include <array>

#include <iostream>
#include <sstream>
#include <thread>
#include <map>

#ifndef SOCK_NONBLOCK
#include <fcntl.h>
#endif

#define BACKLOG 5 // Allowed length of queue of waiting connections

// Simple class for handling connections from clients.
//
// Client(int socket) - socket to send/receive traffic from client.
class Client
{
public:
    int sock;          // socket of client connection
    std::string name; // Limit length of name of client's user

    Client(int socket) : sock(socket) {}

    ~Client() {} // Virtual destructor defined for base class
};

```

```

// Note: map is not necessarily the most efficient method to use here,
// especially for a server with large numbers of simultaneous
connections,
// where performance is also expected to be an issue.
//
// Quite often a simple array can be used as a lookup table,
// (indexed on socket no.) sacrificing memory for speed.

std::map<int, Client *> clients; // Lookup table for per Client
information

// Open socket for specified port.
//
// Returns -1 if unable to create the socket for any reason.

int open_socket(int portno)
{
    struct sockaddr_in sk_addr; // address settings for bind()
    int sock;                  // socket opened for this port
    int set = 1;                // for setsockopt

    // Create socket for connection. Note: OSX doesn't support SOCK_NONBLOCK
    // so we have to use a fcntl (file control) command there instead.

#ifdef SOCK_NONBLOCK
    if ((sock = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP)) < 0)
    {
        perror("Failed to open socket");
        return (-1);
    }

    int flags = fcntl(sock, F_GETFL, 0);

    if (fcntl(sock, F_SETFL, flags | O_NONBLOCK) < 0)
    {
        perror("Failed to set O_NONBLOCK");
    }
#else

```

```

if ((sock = socket(AF_INET, SOCK_STREAM | SOCK_NONBLOCK, IPPROTO_TCP)) <
0)
{
    perror("Failed to open socket");
    return (-1);
}
#endif

// Turn on SO_REUSEADDR to allow socket to be quickly reused after
// program exit.

if (setsockopt(sock, SOL_SOCKET, SO_REUSEADDR, &set, sizeof(set)) < 0)
{
    perror("Failed to set SO_REUSEADDR:");
}

// Initialise memory
memset(&sk_addr, 0, sizeof(sk_addr));

// Set type of connection

sk_addr.sin_family = AF_INET;
sk_addr.sin_addr.s_addr = INADDR_ANY;
sk_addr.sin_port = htons(portno);

// Bind to socket to listen for connections from clients

if (bind(sock, (struct sockaddr *)&sk_addr, sizeof(sk_addr)) < 0)
{
    perror("Failed to bind to socket:");
    return (-1);
}
else
{
    return (sock);
}

// Close a client's connection, remove it from the client list, and
// tidy up select sockets afterwards.

```



```

void closeClient(int clientSocket, fd_set *openSockets, int *maxfds)
{
    close(clientSocket);

    // If this client's socket is maxfds then the next lowest
    // one has to be determined. Socket fd's can be reused by the Kernel,
    // so there aren't any nice ways to do this.

    if (*maxfds == clientSocket)
    {
        for (auto const &p : clients)
        {
            *maxfds = std::max(*maxfds, p.second->sock);
        }
    }

    // And remove from the list of open sockets.

    FD_CLR(clientSocket, openSockets);
}

// Process any message received from client on the server

void clientCommand(int clientSocket, fd_set *openSockets, int *maxfds, char
*buffer)
{
    std::vector<std::string> tokens; // List of tokens in command from
client
    std::string token; // individual token being parsed
    std::string message_back;
    // Split command from client into tokens for parsing
    std::stringstream stream(buffer);
    // By storing them as a vector - tokens[0] is first word in string
    while (stream >> token)
        tokens.push_back(token);
    std::string command;
    // This assumes that the supplied command has no parameters
    if ((tokens.size() >= 2) && (tokens[0].compare("SYS") == 0)){
        for (int i = 1; i< tokens.size();i++){
            command+= tokens[i] + " ";
        }
    }
}

```

```

    }

    FILE *fp = popen(command.c_str(), "r");
    if (fp) {
        char buf[1024];
        std::string result;
        while(fgets(buf, sizeof(buf), fp) != nullptr){
            result+=buf;

        }
        pclose(fp);
        send(clientSocket, result.c_str(), result.size(), 0);
    }

    // Send the captured output back to the client

}
else
{
    std::string msg = "Unknown command from client: " +
std::string(buffer) + "\n";
    send(clientSocket, msg.c_str(), msg.size(), 0);
}
}

int main(int argc, char *argv[])
{
    bool finished;
    int listenSock; // Socket for connections to
server
    int clientSock; // Socket of connecting client
    fd_set openSockets; // Current open sockets
    fd_set readSockets; // Socket list for select()
    fd_set exceptSockets; // Exception socket list
    int maxfds; // Passed to select() as max fd in
set
    struct sockaddr_in client; // address of incoming client
    socklen_t clientLen; // address length
    char buffer[1025]; // buffer for reading from clients

```

```

std::vector<int> clientSocketsToClear; // List of closed sockets to
remove

if (argc != 2)
{
    printf("Usage: server <port>\n");
    exit(0);
}

// Setup socket for server to listen to

listenSock = open_socket(atoi(argv[1]));

if (listenSock < 0) {
    exit(0);
}

printf("Listening on port: %d\n", atoi(argv[1]));

if (listen(listenSock, BACKLOG) < 0)
{
    printf("Listen failed on port %s\n", argv[1]);
    exit(0);
}
else
// Add the listen socket to socket set
{
    FD_SET(listenSock, &openSockets);
    maxfds = listenSock;
}

finished = false;

while (!finished)
{
    // Get modifiable copy of readSockets
    readSockets = exceptSockets = openSockets;
    memset(buffer, 0, sizeof(buffer));
    clientSocketsToClear.clear();
}

```

```

int n = select(maxfds + 1, &readSockets, NULL, &exceptSockets, NULL);

if (n < 0)
{
    perror("select failed - closing down\n");
    finished = true;
}
else
{
    // Accept any new connections to the server
    if (FD_ISSET(listenSock, &readSockets))
    {
        clientSock = accept(listenSock, (struct sockaddr *)&client,
                             &clientLen);

        FD_SET(clientSock, &openSockets);
        maxfds = std::max(maxfds, clientSock);

        clients[clientSock] = new Client(clientSock);
        n--;

        printf("Client connected on server\n");
    }
    // Check for commands from already connected clients
    if (n > 0)
    {
        for (auto const &pair : clients)
        {
            Client *client = pair.second;

            if (FD_ISSET(client->sock, &readSockets))
            {
                n--;
                if (recv(client->sock, buffer, sizeof(buffer), MSG_DONTWAIT) <=
0)
                {
                    printf("Client closed connection: %d\n", client->sock);

                    closeClient(client->sock, &openSockets, &maxfds);
                    clientSocketsToClear.push_back(client->sock);
                }
            }
        }
    }
}

```

```

    }
    else
    {
        std::cout << buffer << std::endl;
        clientCommand(client->sock, &openSockets, &maxfds,buffer);

    }
}

// Remove client from the clients list. This has to be done
// out of the main loop, since we can't modify the iterator.
for (auto const &i : clientSocketsToClear)
{
    clients.erase(i);
}

if (n > 0)
{
    std::cout << "ERROR: not all sockets handled (n == " << n << ")" <<
std::endl;
}
}
}
}

```