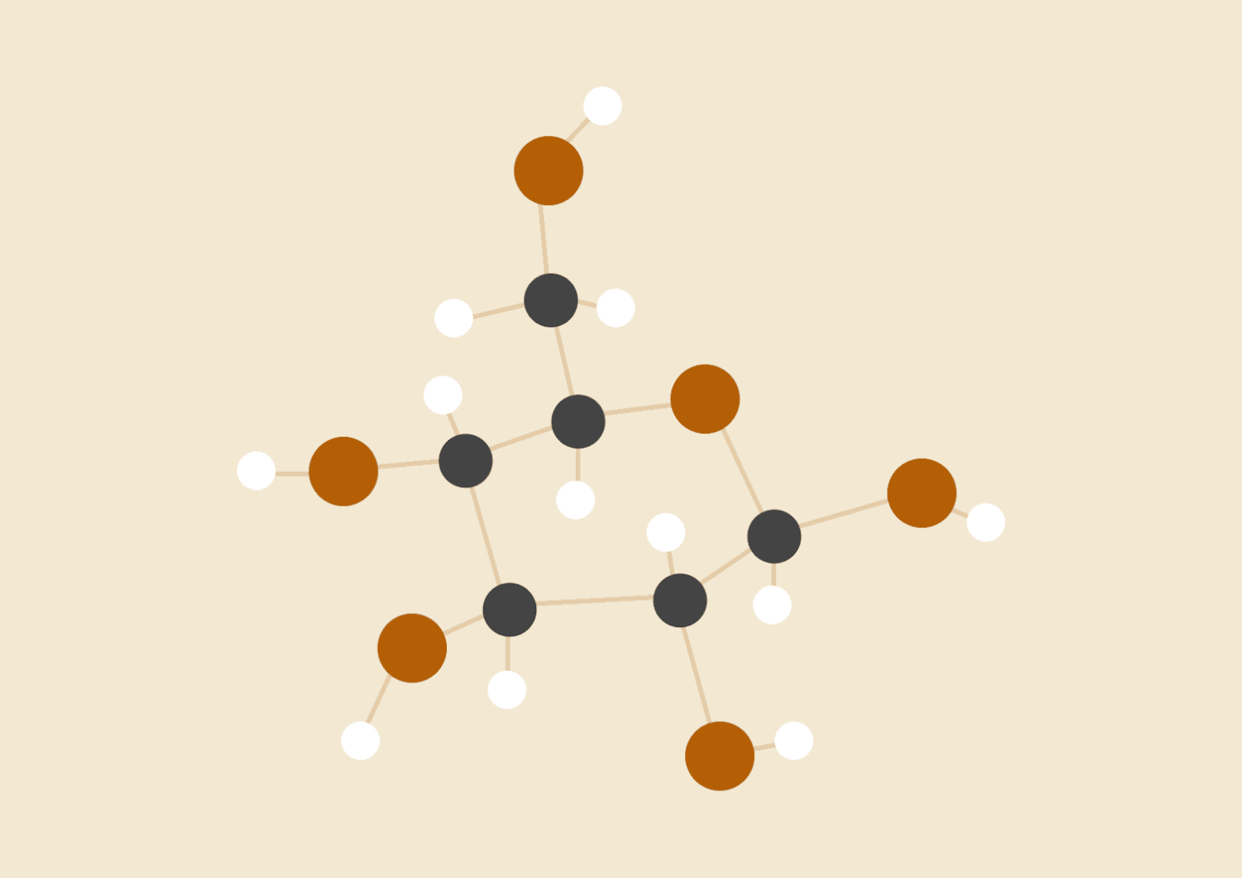
Computer Networks Lab 8



Alisetti Sai Vamsi

31/10/2021

111801002

1. Socket Programming

Host h1:

Host h1 sends a TCP message to h2 of the format “<**PORT> | <num1> | <num2>**” where PORT is the free port on which the TCP server is running and num1, num2 are the arguments passed to the compute.c program. The TCP server running on the free port receives the message of the squared number from h3.

1. **Program**: compute.c
2. **Compilation:** gcc compute.c -o compute
3. **Invocation**: ./compute.c <num1> <num2> <operation>
4. **Code Outline:**
    a. Initialization:
        i. Find a free port using the **find_free_port()** function and this is used as the TCP server port.
        ii. Construct the message to send from the arguments of the program in the following format -> “<TCPSERVERPORT> | <num1> | <num2>”. Example if the TCP Port number is 8000 and num1 and num2 are 10 and 24 respectively then the message will be “8000|10|24”.
    b. Create client socket to communicate with h2 via TCP
        i. Specify the IP address of the server and conditionally assign the PORT number of the server based on the operation provided in the arguments
        ii. Connect to the server using TCP connect.
    c. Create a TCP server
        i. Create server socket
        ii. Assign the socket to the specified port (TCP server port produced by find_free_port())
        iii. Set the socket mode to reusable to prevent socket binding error on consequent calls to the program
        iv. Bind the socket to the port
        v. Listen on the socket
        vi. Send the client message on the client socket and close the client socket -> We are sending it here to prevent the timing issue of h3 sending its message even before the tcp server starts.
        vii. Accept incoming connection (Blocking call)
        viii. Receive message from the client

Host h2:

Host h2 runs 4 TCP servers on different ports specified. On receiving the message from h1, it strips the IP address of h1 and parses the PORT, num1, num2 from the message. It then computes the operation and sends it to h3. It also sends an encoded message of the format “<PORT>:<IPaddress>” to h3 which specifies the port and ip address of h1 on which the tcp server is running.

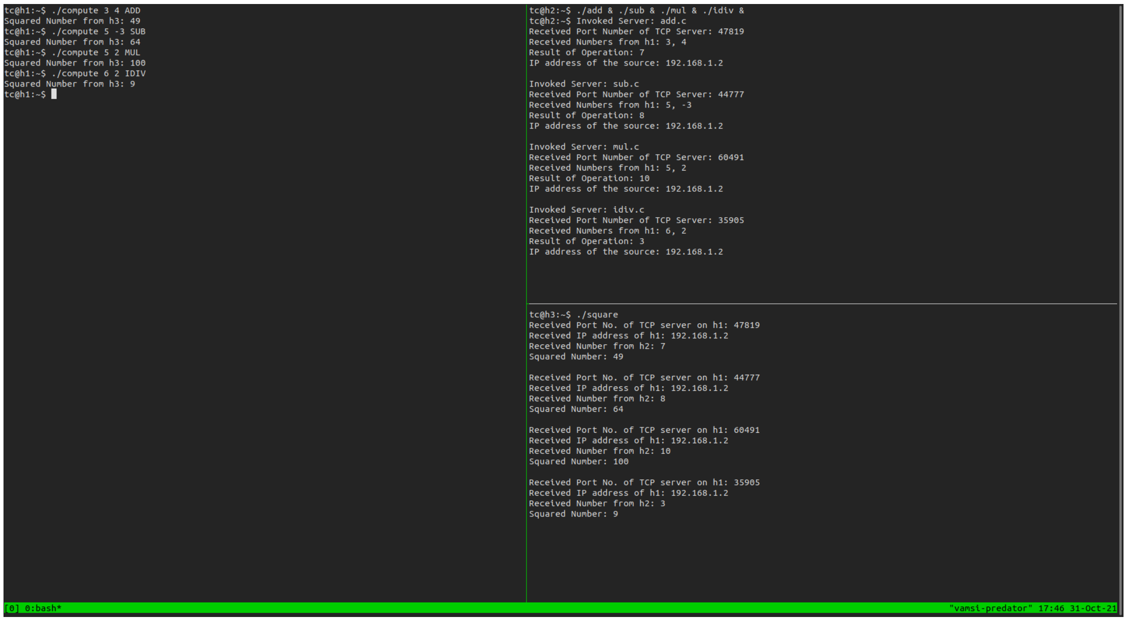
1. **Program**: add.c, sub.c, mul.c, idiv.c
2. **Compilation**:
   a. gcc add.c -o add
   b. gcc sub.c -o sub
   c. gcc mul.c -o mul
   d. gcc idiv.c -o idiv
3. **Invocation**: ./add & ./sub & ./mul & ./idiv &
4. **Code Outline**: (Outline is same for all the servers, the only difference is the operation performed)
   a. **Initialization**: UDP Server Port number between h2 and h3
   b. Create UDP Client
      i. Create socket
      ii. Assign the sever address
   c. Create a TCP server
      i. Create socket
      ii. Assign the socket to the specified port (TCP server port produced by find_free_port())
      iii. Set the socket mode to reusable to prevent socket binding error on consequent calls to the program
      iv. Bind the socket to the port
      v. Listen on the socket
      vi. Loop (Accept incoming connection (Blocking call))
         1. Receive message from the client
         2. Parse the message and get num1, num2, and port
         3. Get client ip from the client_addr structure in the accept function
         4. Perform the operation based on the program i.e (add numbers if add server, subtract numbers if sub server and so on ...)
         5. Encode client ip and port into a single string of the format “<PORT>:<IPaddress>”.
         6. UDP send the output of the operation
         7. UDP send the encoded string of ip and port.

Host h3:

Host h3 receives UDP messages from h2. It receives the final result of the operation and the port and ip of h1’s tcp server. h3 runs a UDP server running on the specified port. After receiving the message it computes the square and sends it to h1 via tcp send.

1. **Program**: square.c
2. **Compilation**: gcc square.c -o square -lm
3. **Invocation**: ./square
4. **Code Outline**:

```
a. Initialization: UDP server Port number
b. Creating a UDP server to listen to the messages from h2
   i.   Create socket
   ii.  Specify the port on which the server should run (8000 in this case)
   iii. Bind the socket to the servers port
   iv.  Loop (Listen to messages using the recvfrom function)
        1. Parse the message and find the number
        2. Compute the square
        3. Receive the ip and port number
        4. Parse the encoded message and extract port and ip
        5. Create a TCP Client
           a. Create Socket
           b. Specify the port and ip of the server
           c. connect to the server
           d. send the message
```



```
tc@h1:~$ ./compute 3 4 ADD
Squared Number from h3: 49
tc@h1:~$ ./compute 5 -3 SUB
Squared Number from h3: 64
tc@h1:~$ ./compute 5 2 MUL
Squared Number from h3: 100
tc@h1:~$ ./compute 6 2 IDIV
Squared Number from h3: 9
tc@h1:~$ 
```

```
tc@h2:~$ ./add & ./sub & ./mul & ./idiv &
tc@h2:~$ Invoked Server: add.c
Received Port Number of TCP Server: 47819
Received Numbers from h1: 3, 4
Result of Operation: 7
IP address of the source: 192.168.1.2

Invoked Server: sub.c
Received Port Number of TCP Server: 44777
Received Numbers from h1: 5, -3
Result of Operation: 8
IP address of the source: 192.168.1.2

Invoked Server: mul.c
Received Port Number of TCP Server: 60491
Received Numbers from h1: 5, 2
Result of Operation: 10
IP address of the source: 192.168.1.2

Invoked Server: idiv.c
Received Port Number of TCP Server: 35905
Received Numbers from h1: 6, 2
Result of Operation: 3
IP address of the source: 192.168.1.2
```

```
tc@h3:~$ ./square
Received Port No. of TCP server on h1: 47819
Received IP address of h1: 192.168.1.2
Received Number from h2: 7
Squared Number: 49

Received Port No. of TCP server on h1: 44777
Received IP address of h1: 192.168.1.2
Received Number from h2: 8
Squared Number: 64

Received Port No. of TCP server on h1: 60491
Received IP address of h1: 192.168.1.2
Received Number from h2: 10
Squared Number: 100

Received Port No. of TCP server on h1: 35905
Received IP address of h1: 192.168.1.2
Received Number from h2: 3
Squared Number: 9
```

```
[0] 0:bash*                                        "vamsi-predator" 17:46 31-Oct-21
```

2. Inetd Daemon

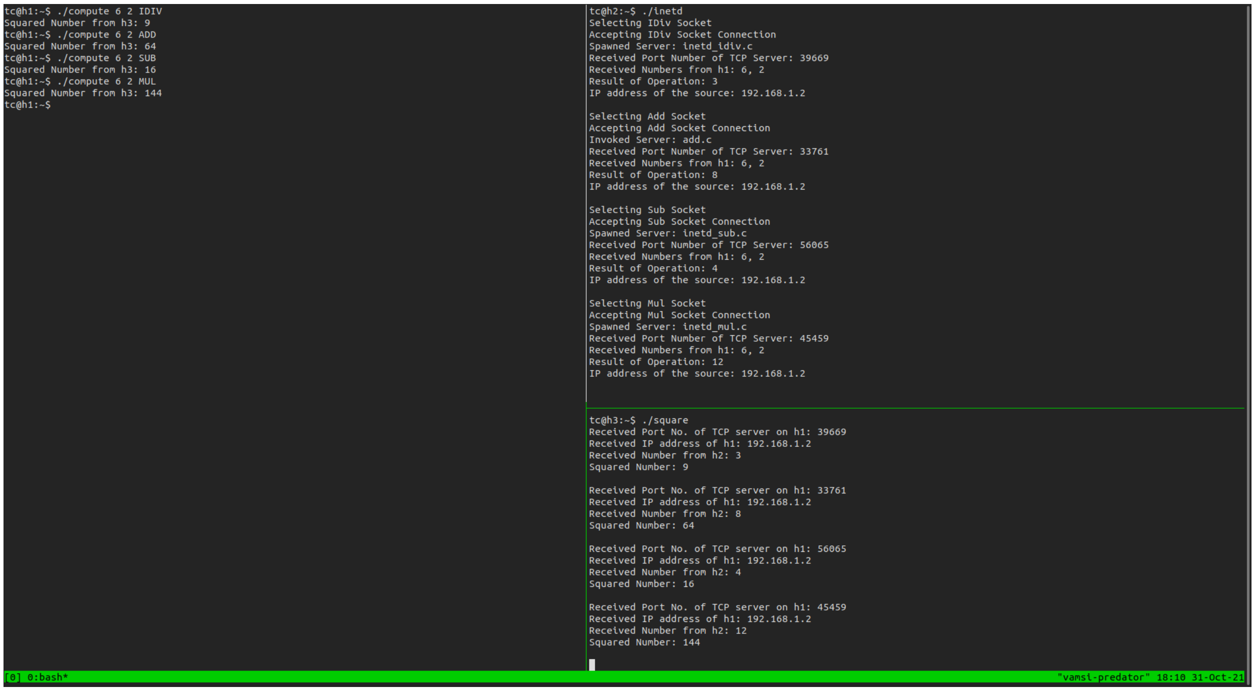
Here I have omitted the usage of a configuration that provides the port number to program mapping since it was not asked on moodle. Instead all of the ports and the program names are hard coded in the code.

1. **Program:** inetd.c
2. **Compilation**: gcc inetd.c -o inetd
3. **Invocation**: ./inetd
4. **Code Outline**:
    a. Initialization:
        i. Initialize all sockets and a file descriptor set.
    b. Create sockets
    c. Assign the server address to all the sockets
    d. Bind all sockets to the corresponding server address
    e. Listen on all sockets
    f. Set the file descriptor set with the socket descriptors
    g. Loop
        i. Reset the file descriptor set
        ii. Select the ready socket in this file descriptor set using select()
        iii. Use FD_ISSET() to find the ready socket after the select call
        iv. Once the ready socket is found
            1. Accept connection
            2. Fork a child process
                a. Make the child process overwrite stdin and stderr file descriptors with the connection socket obtained from accepting connection.
                b. Obtain the IP address of the client from client_addr from accept() function.
                c. Call execl() by providing path to the corresponding server program and pass client ip as an argument.
            3. Make the parent close the accepted connection

Note: Please make sure that before running the inetd.c daemon that add.c / sub.c / mul.c / idiv.c servers should not be running on h2. These should be killed manually using kill -9 <pid>.

Otherwise there will be socket creation error since they use the same ports.

1. **Program**: inetd_add.c, inetd_sub.c, inetd_mul.c, inetd_div.c
2. **Compilation**:
    a. gcc inetd_add.c -o inetd_add
    b. gcc inetd_sub.c -o inetd_sub
    c. gcc inetd_mul.c -o inetd_mul
    d. gcc inetd_idiv.c -o inetd_idiv
3. **Code Outline:** (Outline is same for all the servers, the only difference is the operation performed)
    a. Parse the arguments for the client IP address.
    b. Create UDP Client
        i. Create socket
        ii. Assign the sever address
    c. Receive TCP message
        i. Use read() function to listen on the socket since stdin is overwritten by the socket descriptor.
        ii. Parse the received message to get the port, num1, num2
        iii. Perform the operation based on the type of the server
        iv. Encode the port number and ip into a single string of the format “<PORT>:<IPaddress>”.
        v. UDP send the output of the operation
        vi. UDP send the encoded string of ip and port.



```
tc@h1:~$ ./compute 6 2 IDIV
Squared Number from h3: 9
tc@h1:~$ ./compute 6 2 ADD
Squared Number from h3: 64
tc@h1:~$ ./compute 6 2 SUB
Squared Number from h3: 16
tc@h1:~$ ./compute 6 2 MUL
Squared Number from h3: 144
tc@h1:~$
```

```
tc@h2:~$ ./inetd
Selecting IDiv Socket
Accepting IDiv Socket Connection
Spawned Server: inetd_idiv.c
Received Port Number of TCP Server: 39669
Received Numbers from h1: 6, 2
Result of Operation: 3
IP address of the source: 192.168.1.2

Selecting Add Socket
Accepting Add Socket Connection
Invoked Server: add.c
Received Port Number of TCP Server: 33761
Received Numbers from h1: 6, 2
Result of Operation: 8
IP address of the source: 192.168.1.2

Selecting Sub Socket
Accepting Sub Socket Connection
Spawned Server: inetd_sub.c
Received Port Number of TCP Server: 56065
Received Numbers from h1: 6, 2
Result of Operation: 4
IP address of the source: 192.168.1.2

Selecting Mul Socket
Accepting Mul Socket Connection
Spawned Server: inetd_mul.c
Received Port Number of TCP Server: 45459
Received Numbers from h1: 6, 2
Result of Operation: 12
IP address of the source: 192.168.1.2
```

```
tc@h3:~$ ./square
Received Port No. of TCP server on h1: 39669
Received IP address of h1: 192.168.1.2
Received Number from h2: 3
Squared Number: 9

Received Port No. of TCP server on h1: 33761
Received IP address of h1: 192.168.1.2
Received Number from h2: 8
Squared Number: 64

Received Port No. of TCP server on h1: 56065
Received IP address of h1: 192.168.1.2
Received Number from h2: 4
Squared Number: 16

Received Port No. of TCP server on h1: 45459
Received IP address of h1: 192.168.1.2
Received Number from h2: 12
Squared Number: 144
```