Networking Basics

ECE 454 / 751: Distributed Computing

Instructor: Dr. Wojciech Golab

wgolab@uwaterloo.ca

Slides are derived in part from A. S. Tanenbaum and M. Van Steen,
Distributed Systems: Principles and Paradigms, 2nd Edition, Pearson-Prentice Hall, 2006.

Learning objectives

Very quick review of:

- OSI model
- TCP, UDP, IP, ICMP

Introduction to:

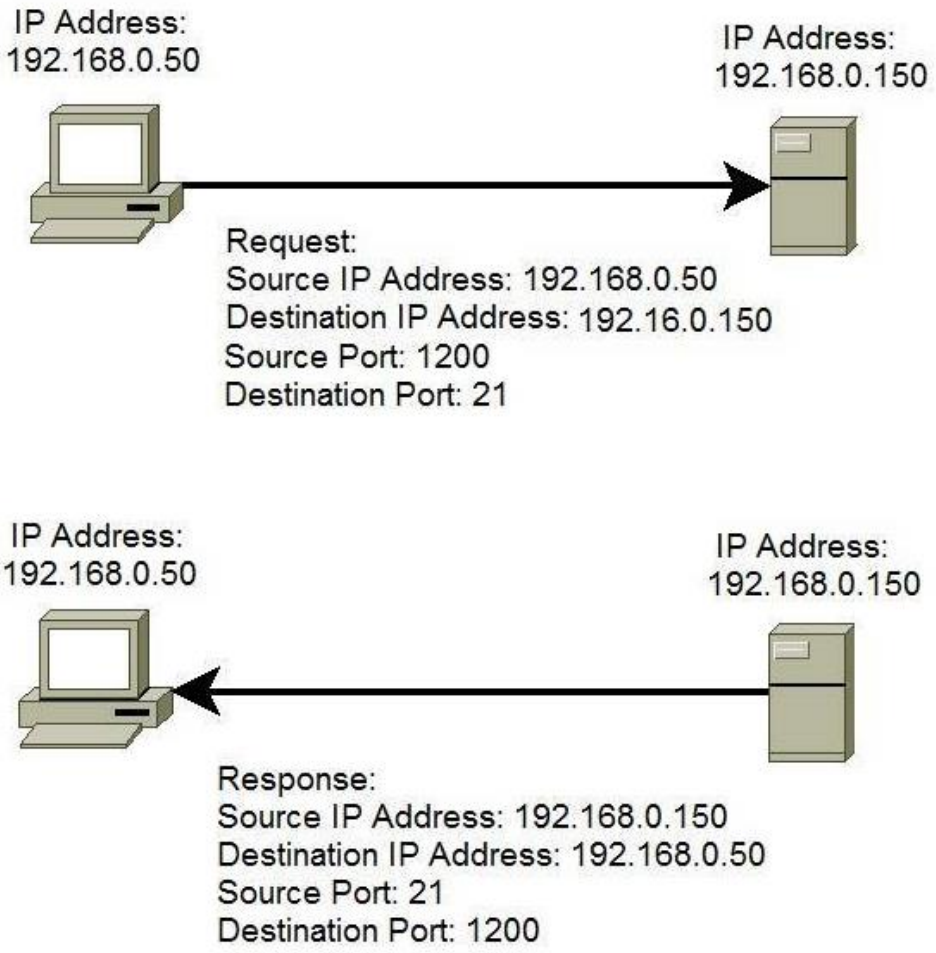
- network programming using sockets

Open Systems Interconnection (OSI) model

image source:
https://www.techcress.com/wp-content/uploads/2015/01/The-OSI-model-explained-in-simple-terms.jpg

TCP/IP addressing

- IP addresses identify hosts.
- TCP ports identify processes at the endpoints of a connection.
- UDP ports work similarly to TCP ports but UDP is connectionless.



Example using nc (netcat)

1. SSH into eceubuntu1 and run "nc -l 10000".
2. SSH into eceubuntu2 and run "nc eceubuntu1 10000".
3. On eceubuntu2 enter "Hello" into the terminal, and press ENTER.
4. On eceubuntu1 observe the output, then enter "Hi" into the terminal and press ENTER.
5. On eceubuntu2 observe the output.
6. Use Ctrl-C to terminate either process.
7. Try again with the server process shut down (i.e., skip step 1).

Hint 1: To avoid TCP port conflicts, try adding the last three digits of your student number to the port number.

Hint 2: Use "netstat -pnt | grep nc" to see the IP addresses and TCP port numbers of both endpoints while the connection is open.

ICMP (Internet Control Message Protocol)

- The ping utility uses the ICMP echo request (type 8) and echo reply (type 0) messages. It can be used to determine whether a host is alive.

  Example: log into eceterm and run "ping uwaterloo.ca"

- Type 3 ICMP messages are used to report errors, such as when a client sends a UDP packet to a port on which no service is listening. In contrast, unreachable TCP ports are dealt with using TCP RST packets.

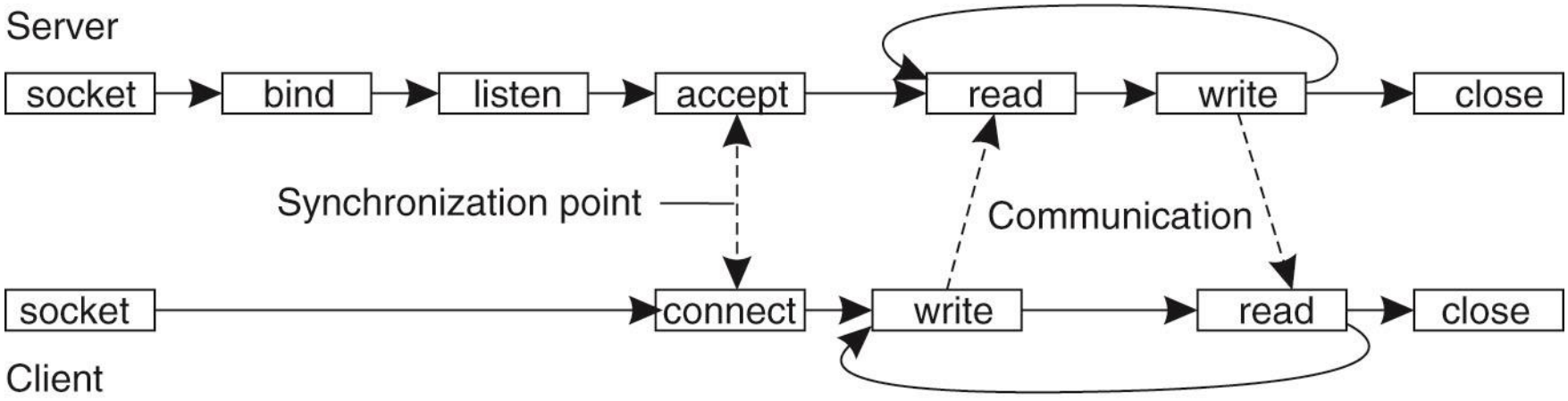
Berkeley sockets

Sockets provide an API for inter-process communication. The Berkeley sockets API evolved into a component of the Portable Operating System Interface (POSIX) specification.

Primitive	Meaning
Socket	Create a new communication end point
Bind	Attach a local address to a socket
Listen	Announce willingness to accept connections
Accept	Block caller until a connection request arrives
Connect	Actively attempt to establish a connection
Send	Send some data over the connection
Receive	Receive some data over the connection
Close	Release the connection

Connection-oriented communication using sockets

The server socket binds either to a specific network interface identified by an IP address, or to all the interfaces on the sever host. It always binds to exactly one TCP port.



Java example: client

```
public static void main(String args[]) throws Exception {
    Socket sock = new Socket(args[0], Integer.parseInt(args[1]));
    BufferedReader reader = new BufferedReader(
        new InputStreamReader(sock.getInputStream()));
    PrintWriter writer = new PrintWriter(sock.getOutputStream(), true);
    writer.println(args[2]);
    String line = reader.readLine();
    System.out.println("Got response: " + line);
    sock.close();
}
```

Java example: server

```
public static void main(String args[]) throws Exception {
    ServerSocket ssock = new ServerSocket(Integer.parseInt(args[0]));
    while (true) {
        try {
            Socket csock = ssock.accept();
            BufferedReader reader = new BufferedReader(
                    new InputStreamReader(csock.getInputStream()));
            PrintWriter writer = new PrintWriter(csock.getOutputStream(), true);
            writer.println(reader.readLine().toUpperCase());
            csock.close();
        } catch (Exception e) {
            …
```

Java example: food for thought

- How many sockets are created by the client? How many by the server?
- At what points in the code do the sockets bind, listen, accept, connect, read, write, and close?
- How are request and response messages delimited?
- Is the server able to process requests from multiple clients concurrently?
- Why is there no need to call writer.flush()?
- How portable is the code?

Java socket options

- **setReuseAddress:** controls SO_REUSEADDR option, which deals with the problem of binding a local address for which there is a prior TCP connection in the TIME_WAIT state (e.g., because a server process crashed unexpectedly and was restarted).
- **setSoTimeout:** determines the timeout period for a blocking read on the socket's InputStream.
- **setKeepAlive:** controls TCP "keep alive" packets
- **setTcpNoDelay:** controls TCP_NODELAY option, used to disable Nagle's algorithm when required to improve latency.
- **setSendBufferSize/setReceiveBufferSize:** controls the SO_SNDBUF and SO_RCVBUF options, which are hints for determining the size of the underlying network I/O buffers.