CSCI315 – Operating Systems Design

Department of Computer Science
Bucknell University

Inter-Process Communications: Socket

This set of notes is based on notes from the textbook authors, as well as L. Felipe Perrone, Joshua Stough and other instructors.
Xiannong Meng, Fall 2021.

A portion of the materials are adopted from
http://www.cs.toronto.edu/~yganjali/resources/Course-Handouts/CSC458/H03--CSC458-Tutorial-I.pdf
And from *Computer Networking:A Top Down Approach* by Kurose and Ross

IPC: Socket

- In previous lectures and labs, we learned one way of process communications through Unix **pipes**.
  - **pipes** work with the processes that have relation, e.g., parent and child (one created the other).
- For independent processes that could reside on completely difference computers, we use **sockets**.
- A third popular IPC mechanism is **shared memory**, which we touched a bit when discussing OpenMP. We will not explore in details here.
- In the last lecture, we saw how sockets are used in network programming. Here we examine how sockets work.

Revisit the Client/Server Example

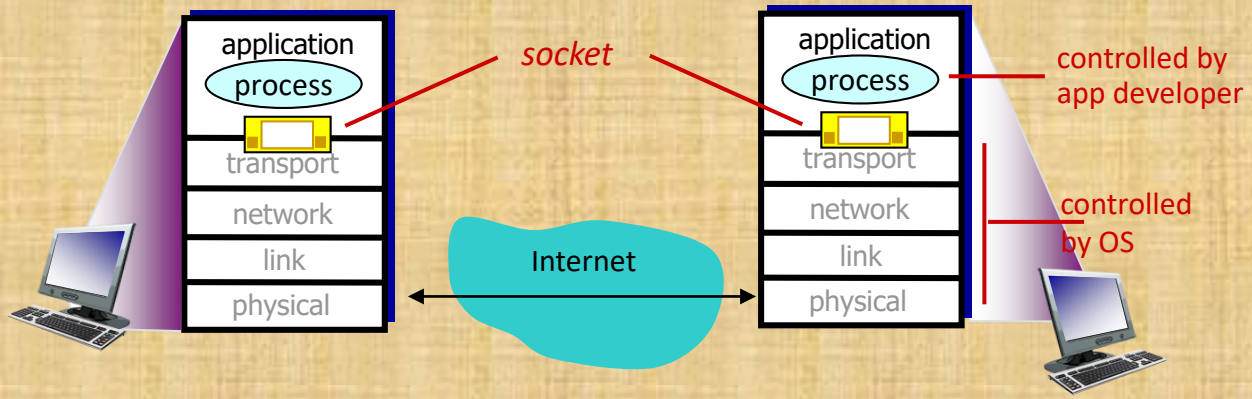
- Demonstrate how client and server work
- The server program provides two services
  - time()
  - is_prime()
- Copy programs from either source
  - On the web
    http://www.eg.bucknell.edu/~cs315/F2021/meng/code/time-prime/
  - On Linux file system

    cp ~cs315/public_html/F2021/meng/code/time-prime/* .

Socket programming

goal: learn how to build client/server applications that communicate using sockets

socket: interface between application process and end-end-transport protocol



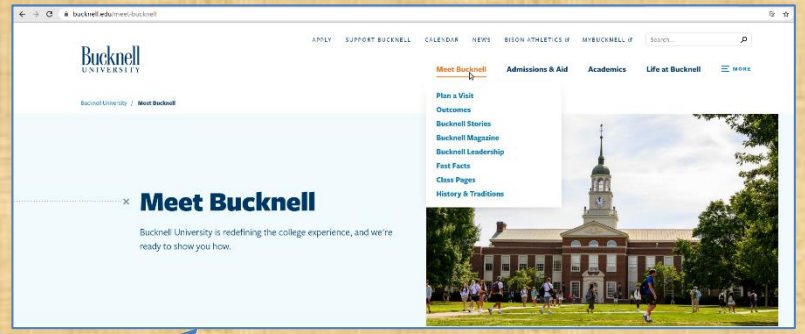
Two Types of Sockets

- There are two general types of sockets, one based on TCP and one based on UDP.
  - TCP socket is connection oriented, the two processes have to establish a connection before communication, similar to a **phone call**.
  - UDP socket is connection-less, a process can simply send a packet to another without connection, similar to a piece of **phone text**.
- We will study TCP socket only here. Detailed of others can be found in a computer networks course.

TCP and UDP

- TCP stands for Transmission Control Protocol
- UDP stands User Datagram Protocol
- Both are prominent communication protocols used in today's internet – TCP is more widely used
- **Protocol**: A set of conventions when the two parties communicate. E.g., dial the phone number first, say "hello", …, "bye", then hang up.

TCP in Action



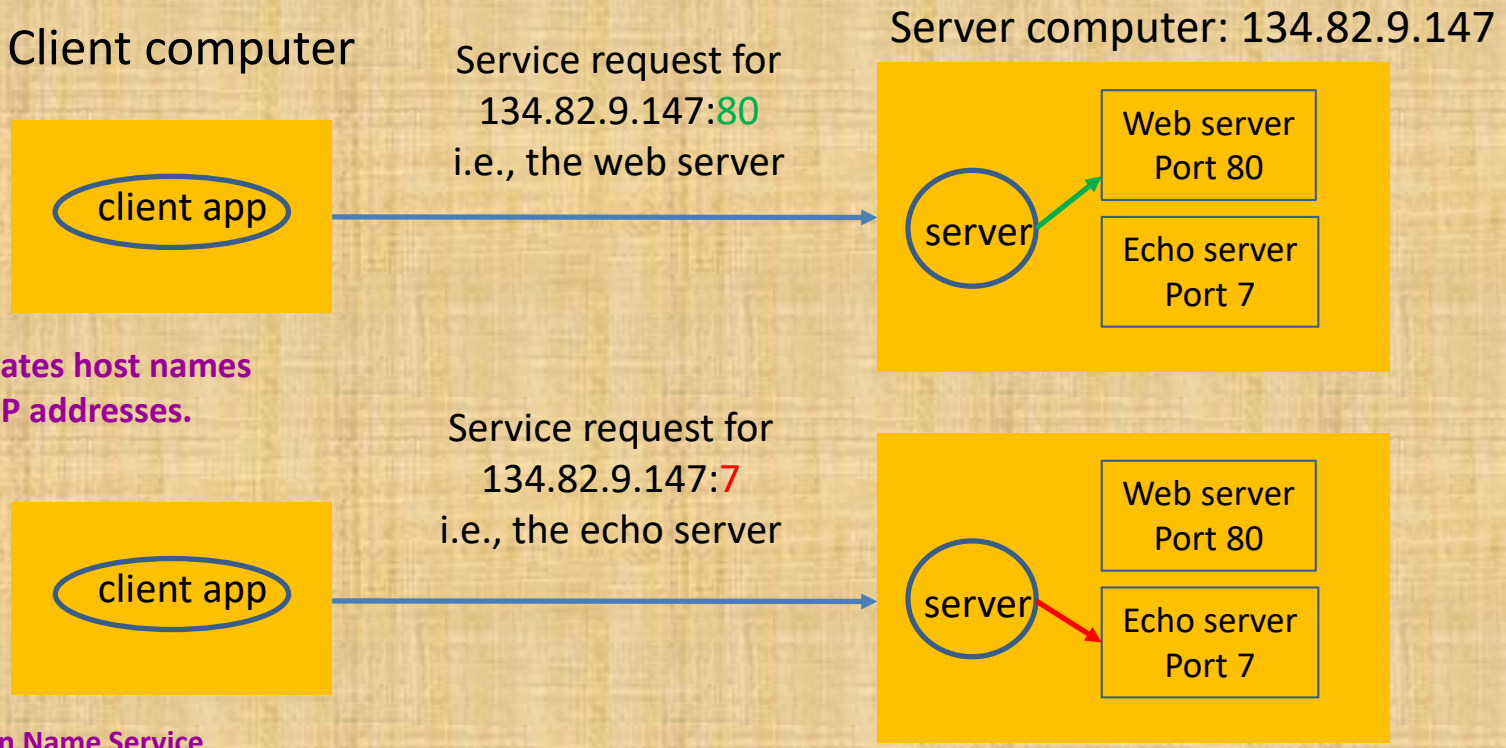
https://www.bucknell.edu/meet-bucknell

Typical Client Program

- **Create** a socket, specifying this socket uses **stream** communication, i.e., it is connection based.
- **Determine** the server name and a port number, e.g., www.bucknell.edu as server name and 80 as a port number for web servers.
  - One server computer can provide **many types of services**, a port number identifies exactly which service is requested.
  - How a port number is determined? By mutual agreement!
  - See a list of commonly used port numbers on this Wiki site

  https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers
- **Make** a connection request to the server.
- **Exchange** data with the server. Once accepted by the server, the communication is two-way.
- **Close** the connection.

Using Ports to Identify a Service



http://www.cs.toronto.edu/~yganjali/resources/Course-Handouts/CSC458/H03--CSC458-Tutorial-I.pdf

A Simple Web Server

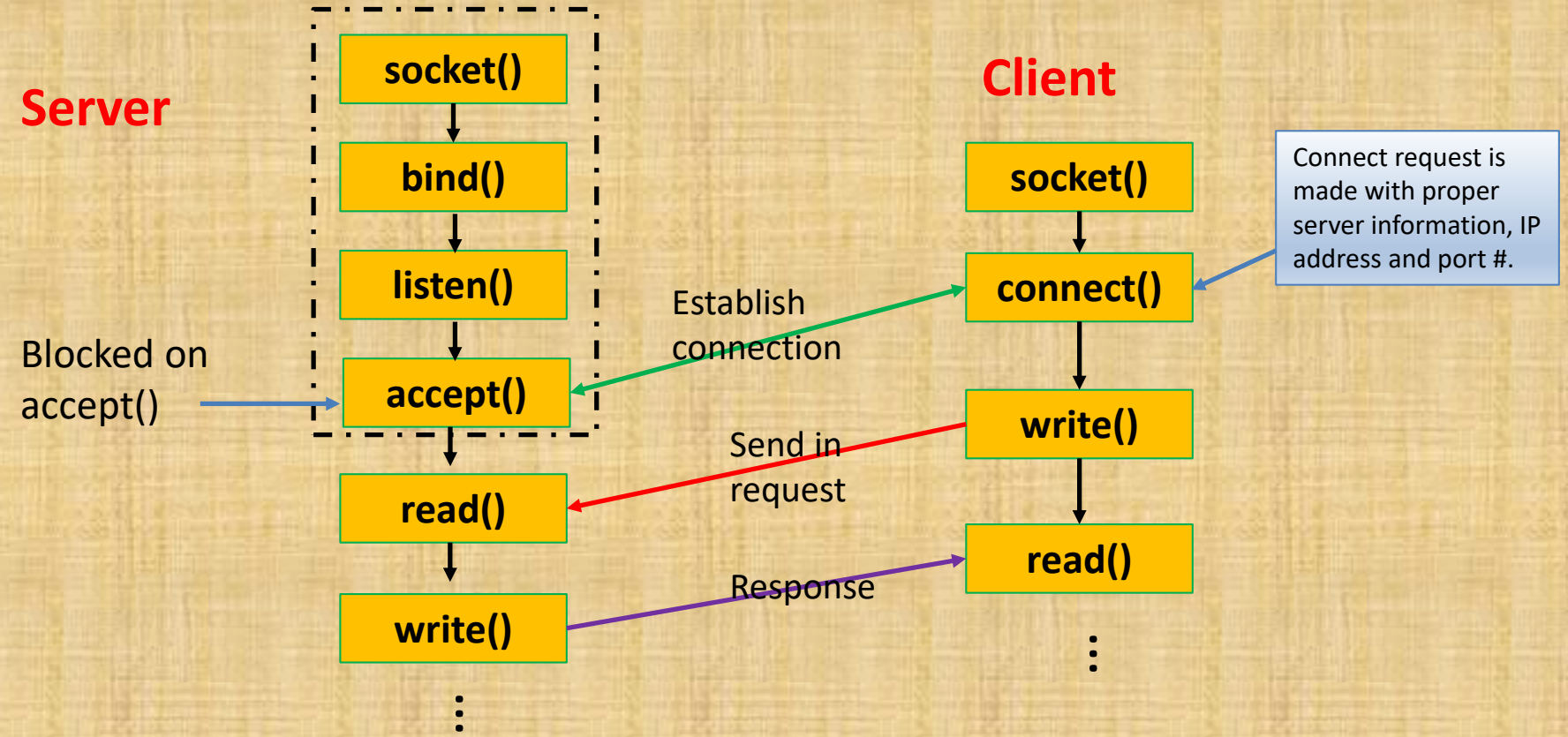
Try out a simple web server:
http://www.eg.bucknell.edu/~cs315/F2021/meng/code/web-client-server-c/

Or copy from Linux file system:
cp -r ~cs315/public_html/F2021/meng/code/web-client-server-c/ .

Typical Server Program

- **Create** a socket, specifying this socket uses **stream** communication, i.e., it is connection based.
- **Bind** the socket to local host and port for service.
- **Wait and accept** connection requests.
- **Exchange** data with the client. Once accepted by the server, the communication is two-way.
- **Close** the current connection.
- **Go** back to the **wait** state, and repeat.

Put All Together



Flow of Execution

- The previous slide shows the flow of execution in C language.
- You will practice it in the upcoming lab.
- Other programming languages that support socket API, such as Java, Python and many others, follow a similar pattern, though the syntax may vary.