Network Programming (Part 1)

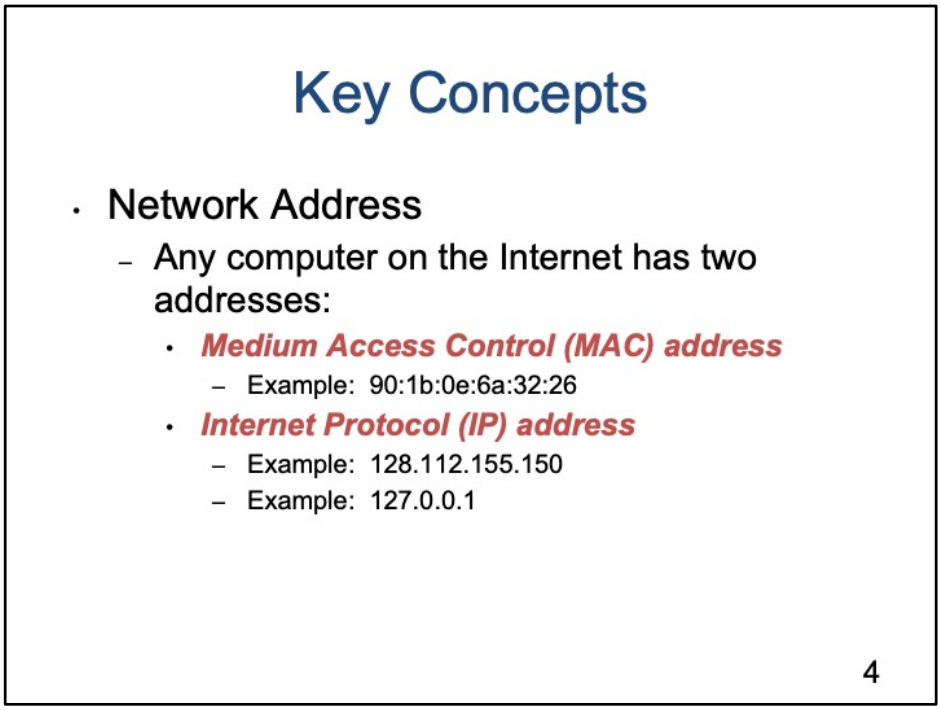
Robert M. Dondero, Ph.D.
Princeton University

Objectives

- We will cover:
  - Network programming key concepts
  - Client/server computing
  - Client/server computing in COS 333
  - Network programming in Python
    - How to compose a client
    - How to compose a server

Agenda

- **Key concepts**
- Client/server computing
- Client/server computing in COS 333
- Network programming



Key Concepts

Network Address

Any computer on the Internet has two addresses:

Medium Access Control (MAC) address

Used by network adapters, bridges, routers

Example: 90:1b:0e:6a:32:26

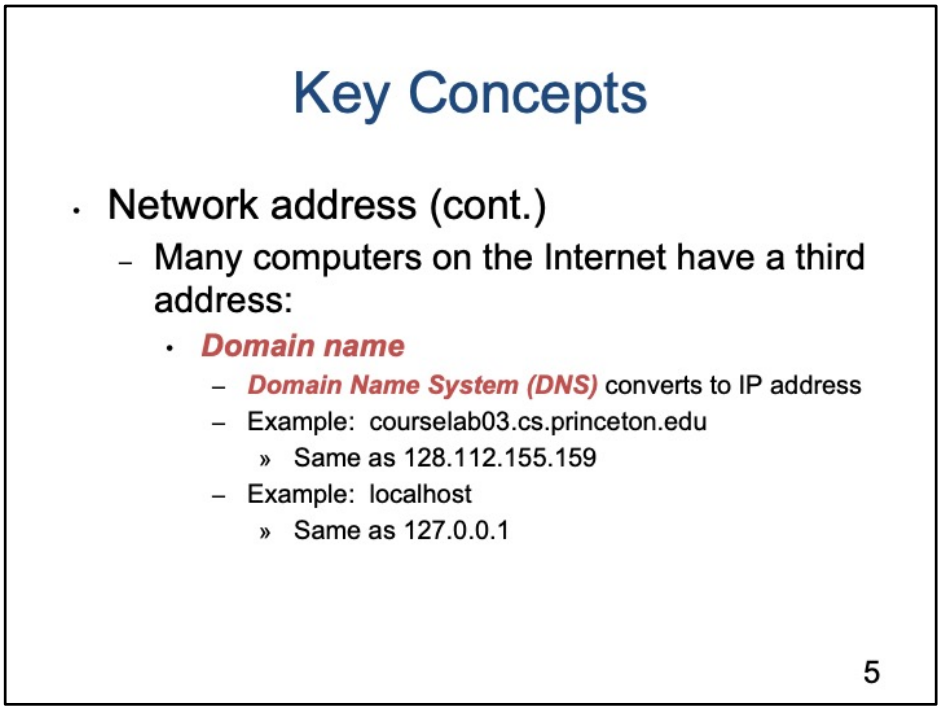
Internet Protocol (IP) address

Used by routers (and people)

32 bit integer, usually expressed in dotted-decimal form

Example: **128.112.155.150** (the IP address of courselab03)

Example: **127.0.0.1** (an alias for the IP address of the current computer)



Key Concepts

Network address (cont.)

Many computers on the Internet have a third address:

Domain name

Used by people

Domain Name System (DNS) converts to IP address

Example: **courselab03.cs.princeton.edu**

Same as **128.112.155.159**

Example: **localhost**

Same as **127.0.0.1**

Key Concepts

- ***Port***
  - A software abstraction
  - 16-bit integer (0 - 65535)
  - Any process that communicates over a network is associated with a specific port

Key Concepts

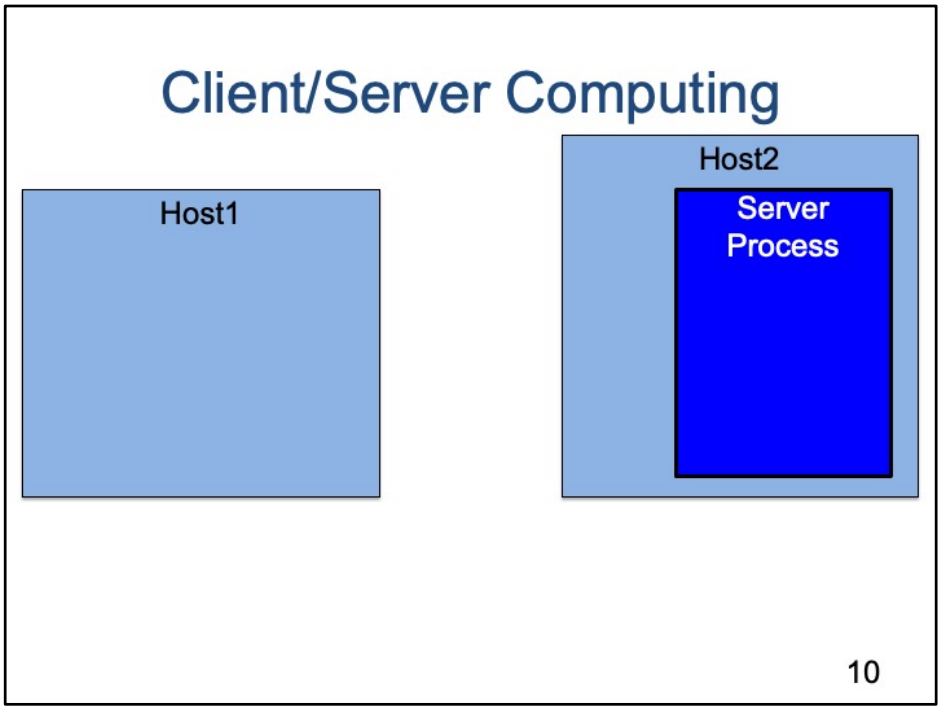
- ***Socket***
  - IP address + port
  - Dominant abstraction in network programming
  - First developed by Unix/C community
  - Adopted by Python, Java, many others
  - Used to implement…

Agenda

- Key concepts
- **Client/server computing**
- Client/server computing in COS 333
- Network programming

Client/Server Computing

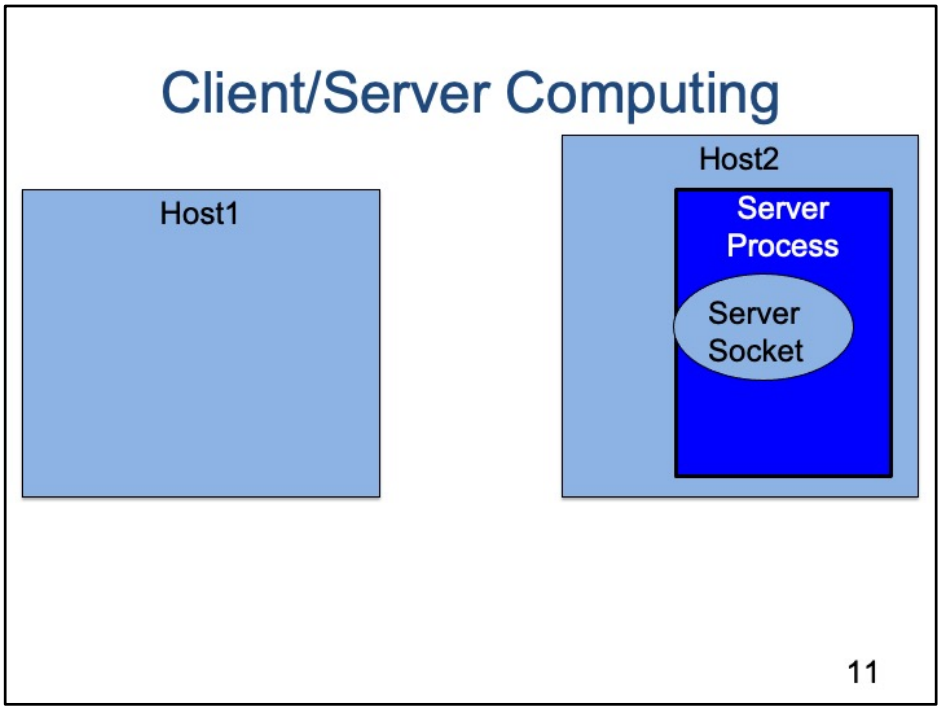
- ***Client/server computing***
  - ***Client*** process
    - Running on a host computer with IP address H1, at port P1
  - Communicates using a socket with…
  - ***Server*** process
    - Running on a host computer with IP address H2, at port P2



Client/Server Computing

[see slide]

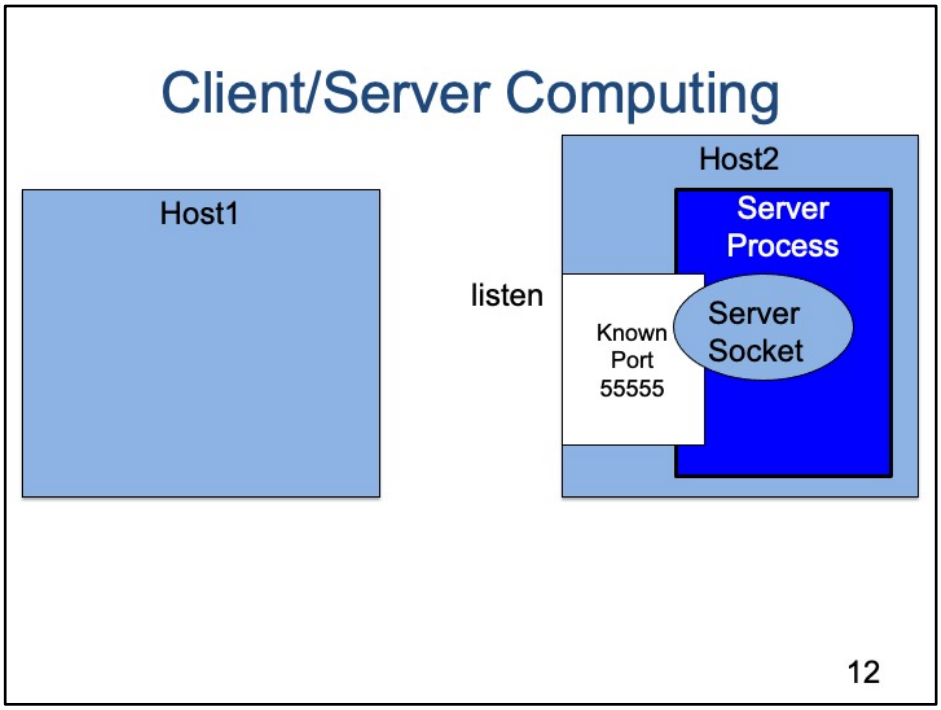
Run server process on host2



Client/Server Computing

[see slide]

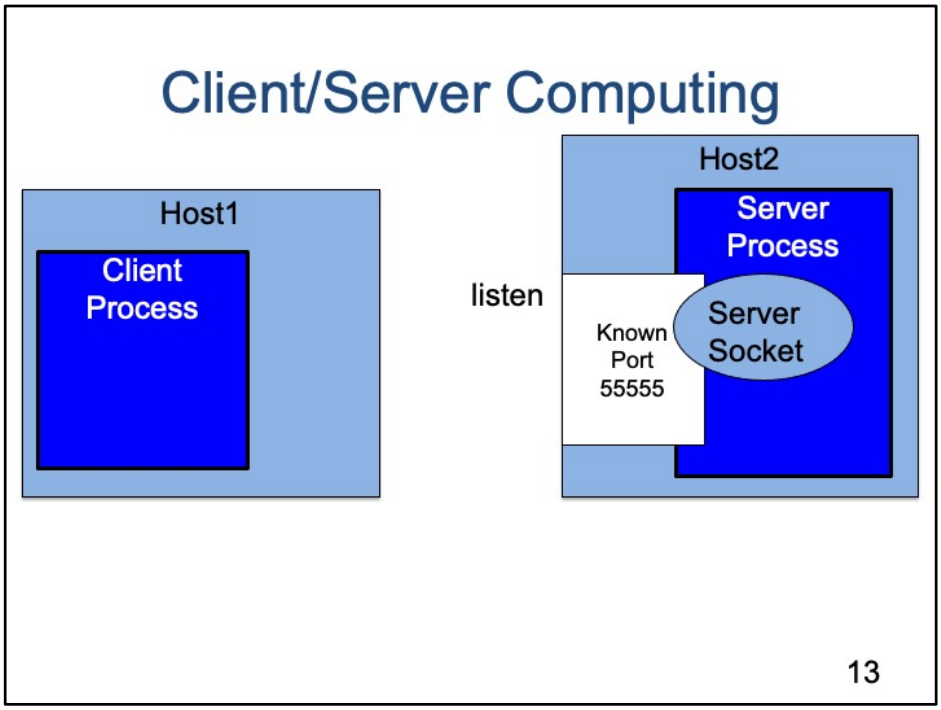
Server process creates a ServerSocket



Client/Server Computing

[see slide]

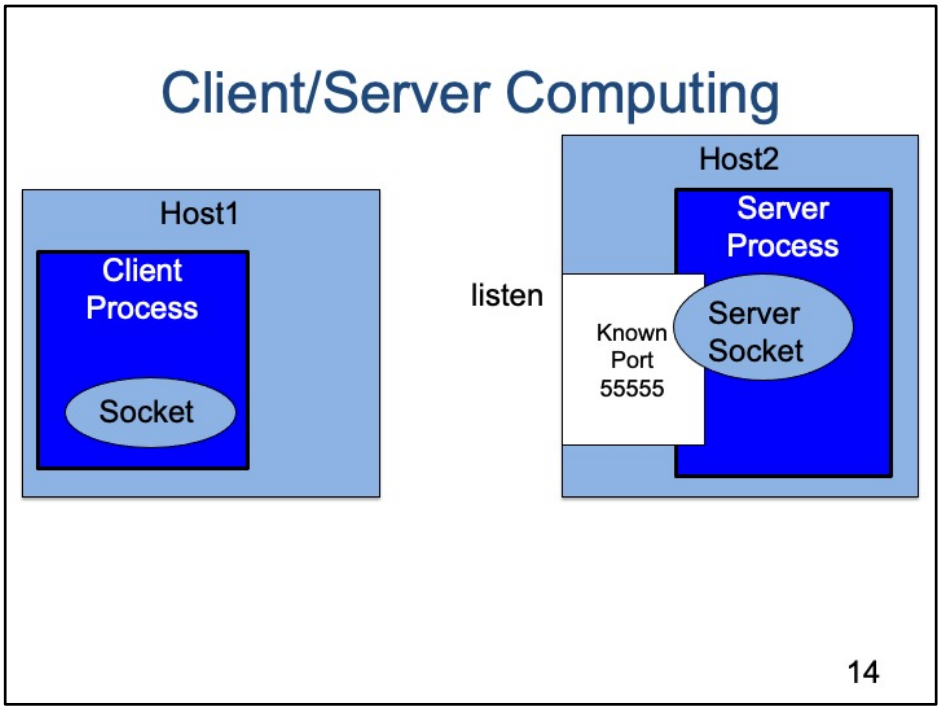
Server process commands ServerSocket to listen for connections on some *known* port (e.g., 55555)



Client/Server Computing

[see slide]

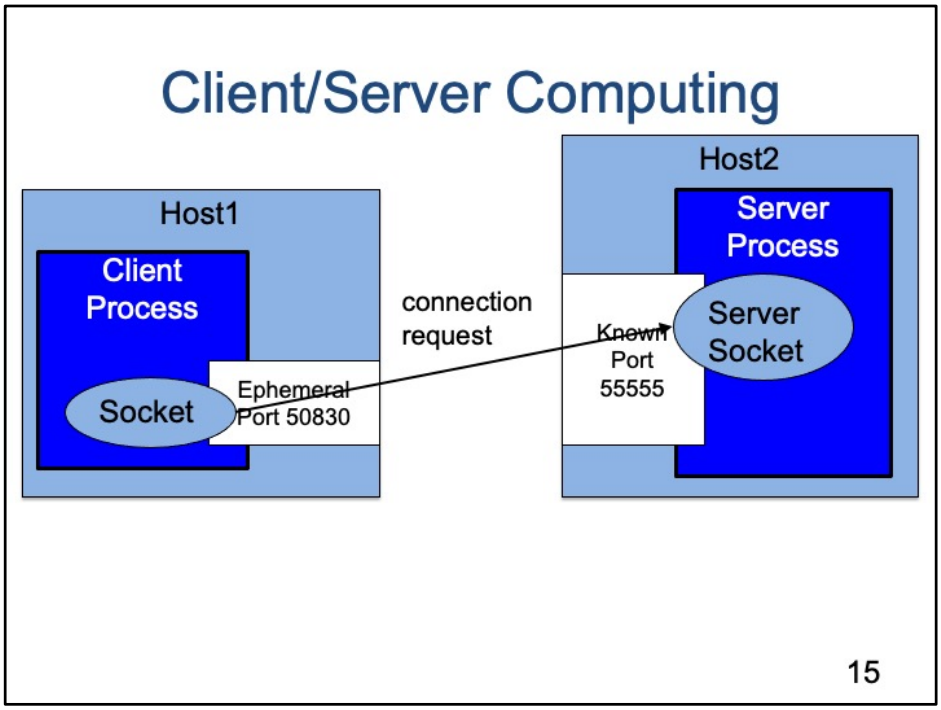
Run client process on host1



Client/Server Computing

[see slide]

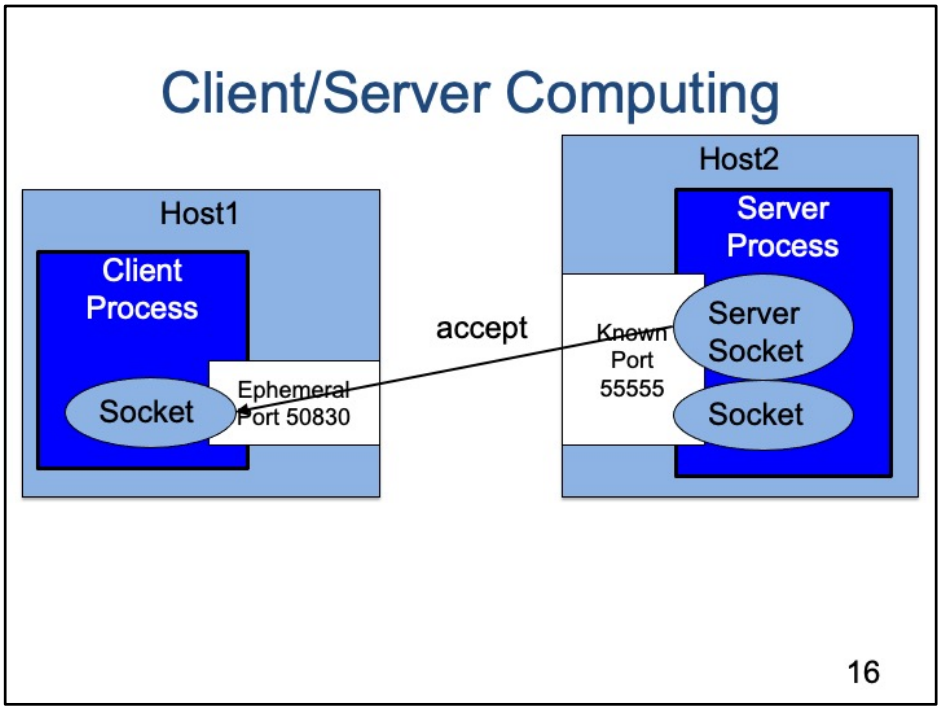
Client process creates Socket



Client/Server Computing

[see slide]

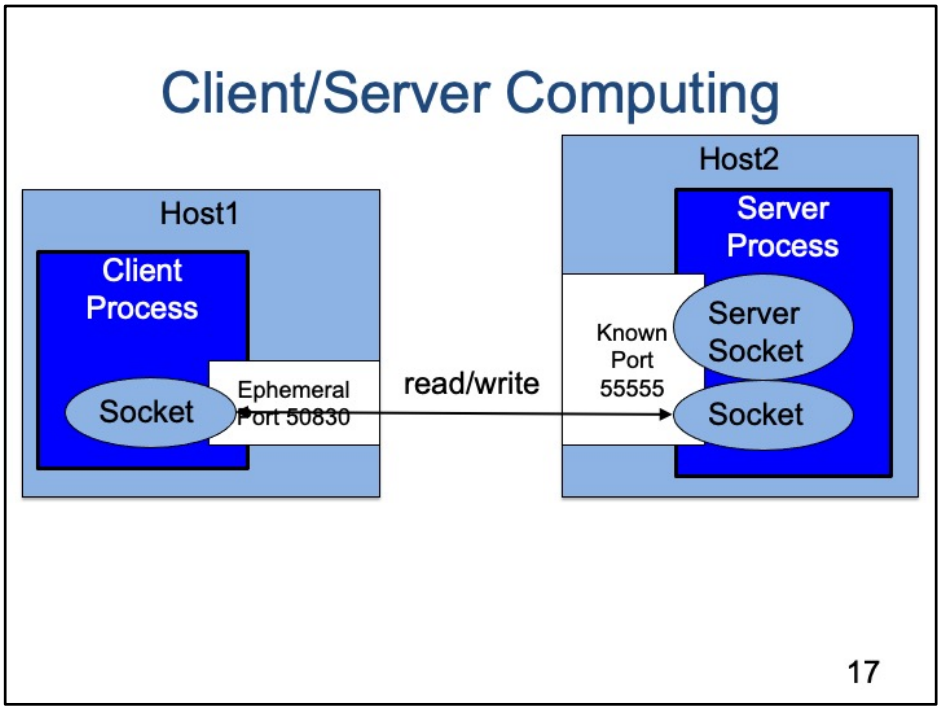
Client process commands Socket to connect with host2 at known port (e.g., 55555)
Client process's Socket chooses some *ephemeral* port (e.g., 50830)
Client process's socket communicates "connection request" to server process's ServerSocket at host2 port 55555



Client/Server Computing

[see slide]

Server process's ServerSocket accepts request
Server process's ServerSocket creates ordinary socket at *known* port
Server process's ServerSocket communicates "accept" to client process's Socket



Client/Server Computing

[see slide]

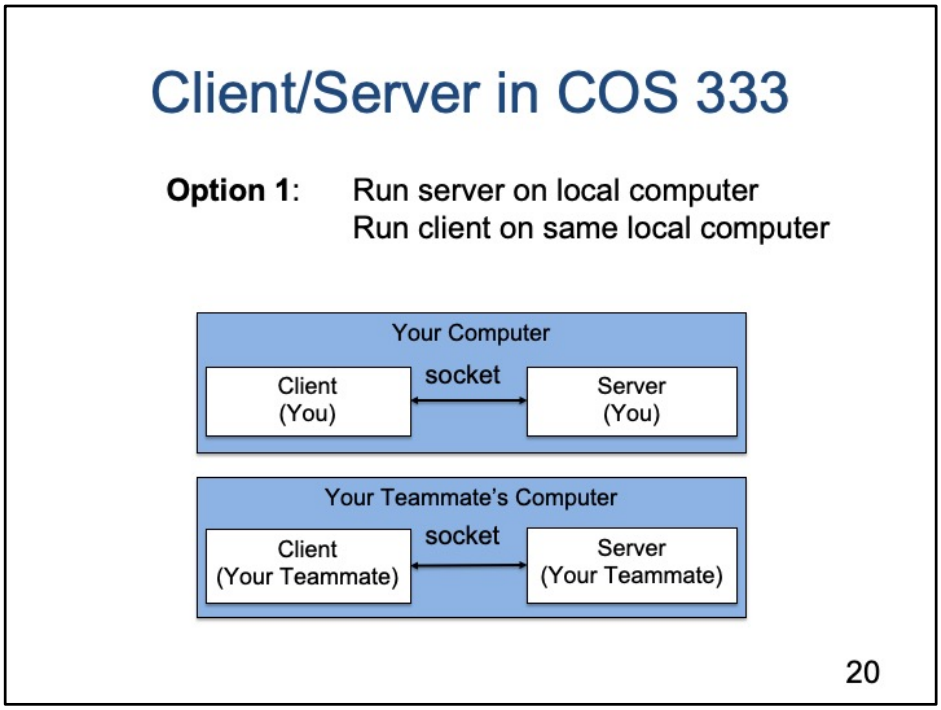
Client writes to its socket, and thereby to server
Server reads from its socket, and thereby from client
Server writes to its socket, and thereby to client
Client reads from its socket, and thereby from server

Agenda

- Key concepts
- Client/server computing
- **Client/server computing in COS 333**
- Network programming

Client/Server in COS 333

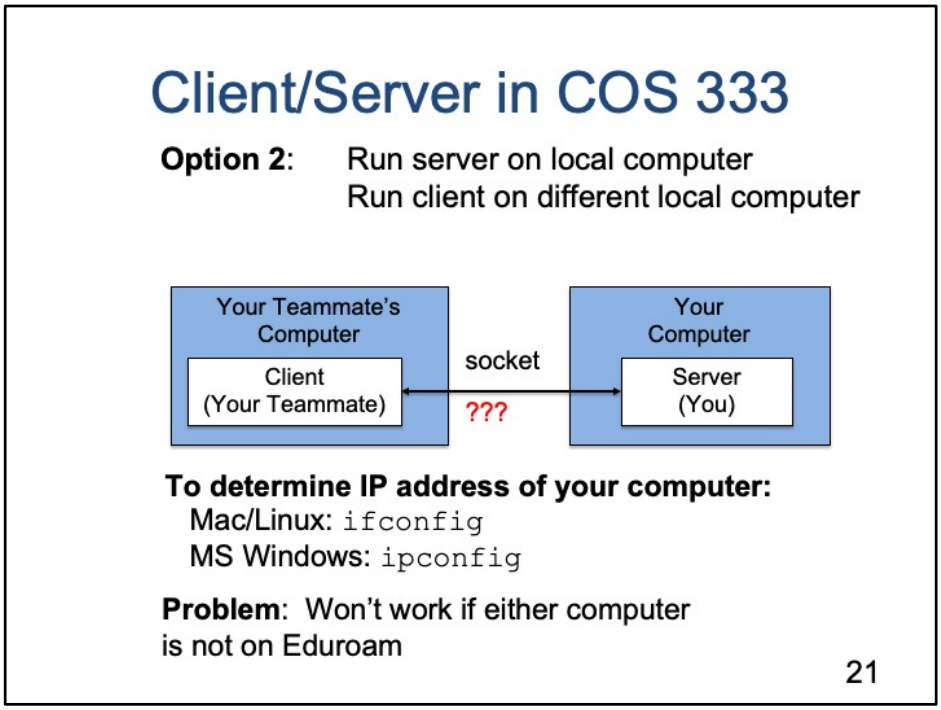
- In the context of Assignment 2...



Client/Server Computing in COS 333

[see slide]

You run server & client on your computer
Your teammate runs server & client on his/her computer



Client/Server Computing in COS 333

[see slide]

You run server on your computer
You tell your teammate:
IP addr of your computer
Use `ifconfig` (Mac/Linux) or `ipconfig` (MS Windows) to find out
Port at which server is listening
Your teammate runs client on his/her computer
Could switch roles

Problem:

Won't work if either computer is not on Eduroam

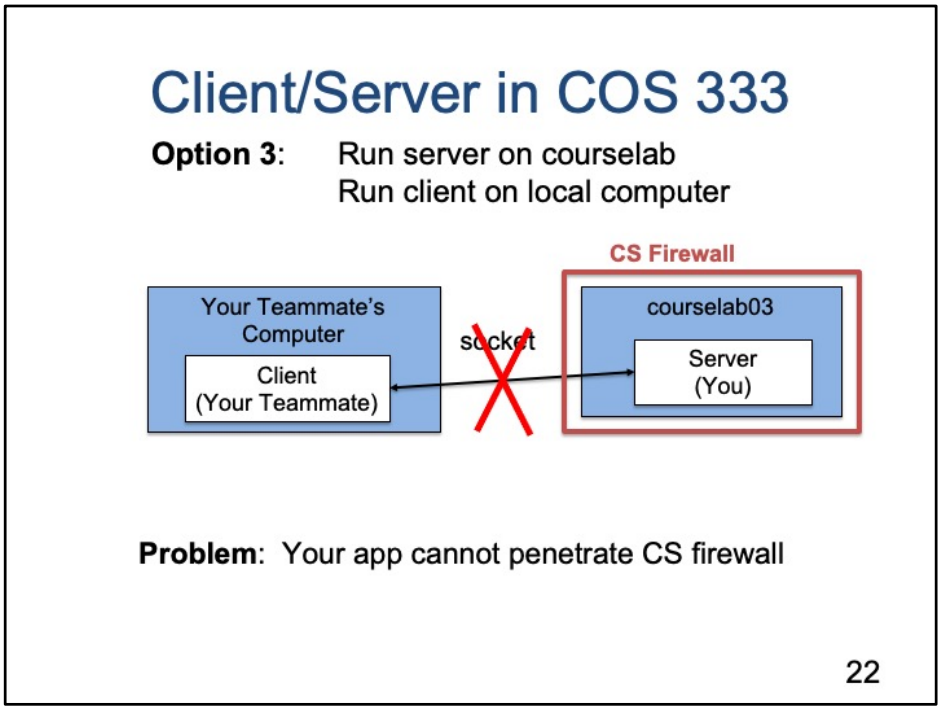
Eduroam uses **private IP addresses**

Trouble during recent remote lectures

Solution:

Reconfigure router(s) to use public IP addresses

(Not recommended)



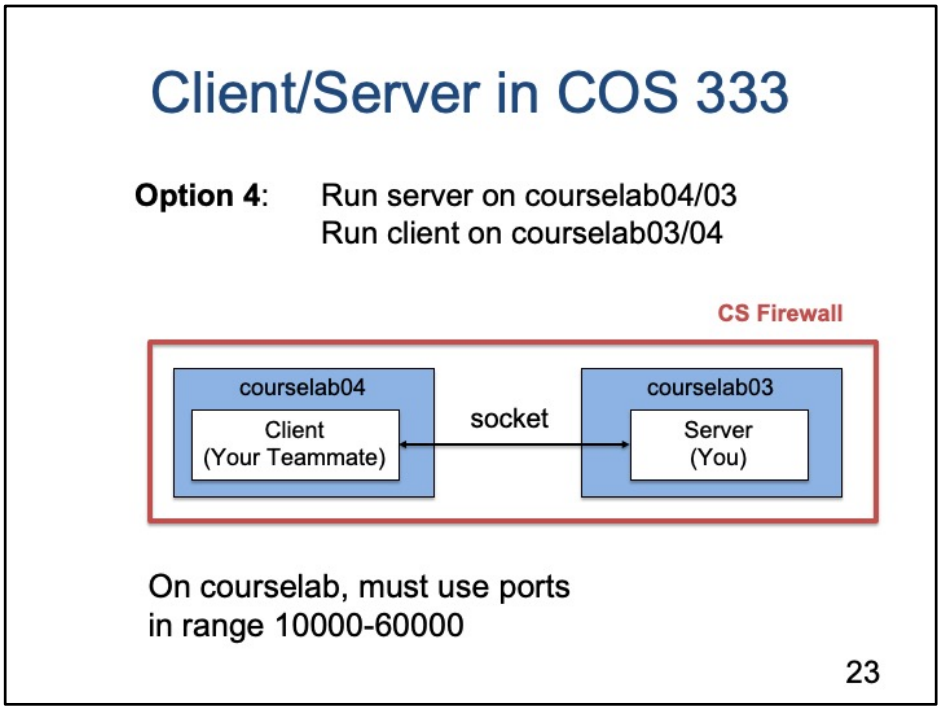
Client/Server Computing in COS 333

[see slide]

You run server on courselab03
You tell your teammate:
    Which computer (courselab03) is running server
    Port at which server is listening
Your teammate runs client on his/her computer
Could use courselab04; could switch roles

Problem: firewall
    Wikipedia: "In computing, a ***firewall*** is a network security system that monitors and controls incoming and outgoing network traffic based on predetermined security rules."
    Wikipedia: "A firewall typically establishes a barrier between a **trusted internal network** and **untrusted external network**, such as the Internet."
    Your app cannot penetrate CS firewall

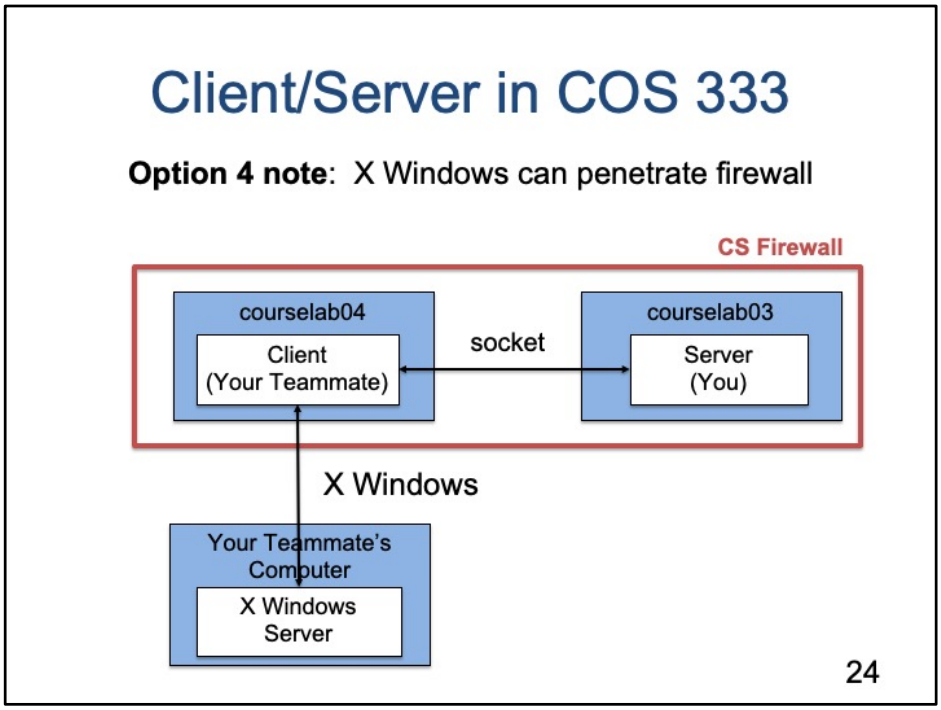
Could use **port forwarding**; beyond our scope; see me if you want



Client/Server Computing in COS 333

[see slide]

You run server on courselab03
You tell your teammate:
    Which computer (courselab03) is running server
    Port at which server is listening
        Important: use ports in range 10000-60000
Your teammate runs client on courselab04
Could switch computers; could switch roles



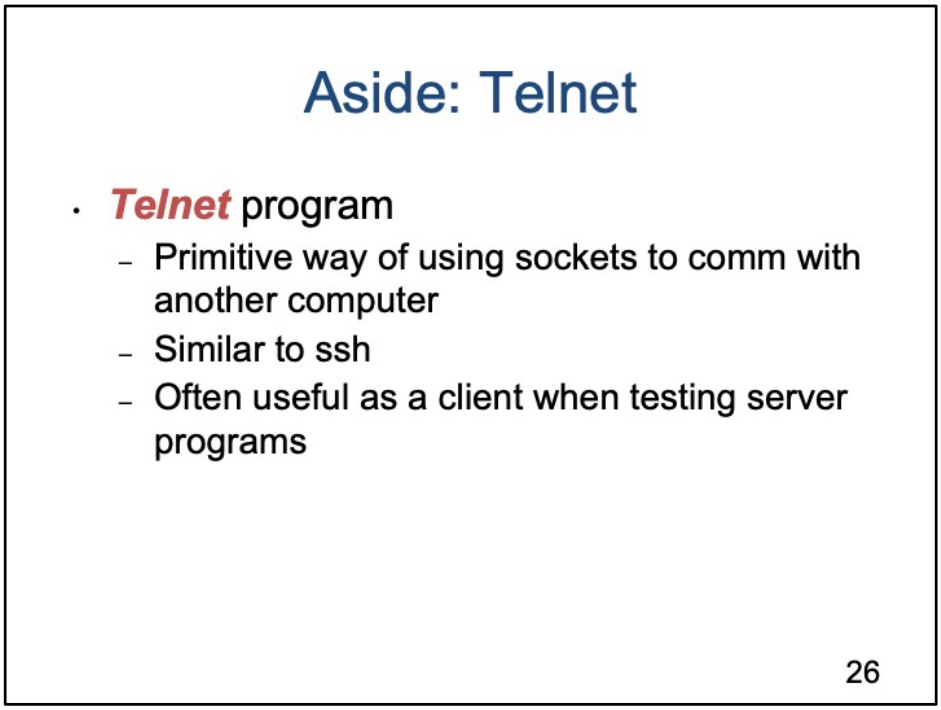
Client/Server Computing in COS 333

[see slide]

Will matter for Assignment 2

Client/Server in COS 333

- Suggestions:
  - Use **option 1** (client & server on your local computer) during development
  - If possible, use **option 2** (client & server on different local computers) to test network comm
- Requirement:
  - Use **option 4** (client & server on different courselab computers) to test network comm, and to make sure that both client & server run on courselab



Aside: Telnet

Telnet program

Run from a command line

Primitive way of using sockets to comm with another computer

Similar to **ssh**

But without data encryption

Often useful as a client when testing server programs

Aside: Telnet

- Installing telnet (Linux)
  - Already installed!

Aside: Telnet

- Installing telnet (Mac before High Sierra)
  - Already installed!
- Installing telnet (Mac starting with High Sierra)
  - Install Homebrew
    - `/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"`
  - Use Homebrew to install telnet
    - `brew install telnet`

Aside: Telnet

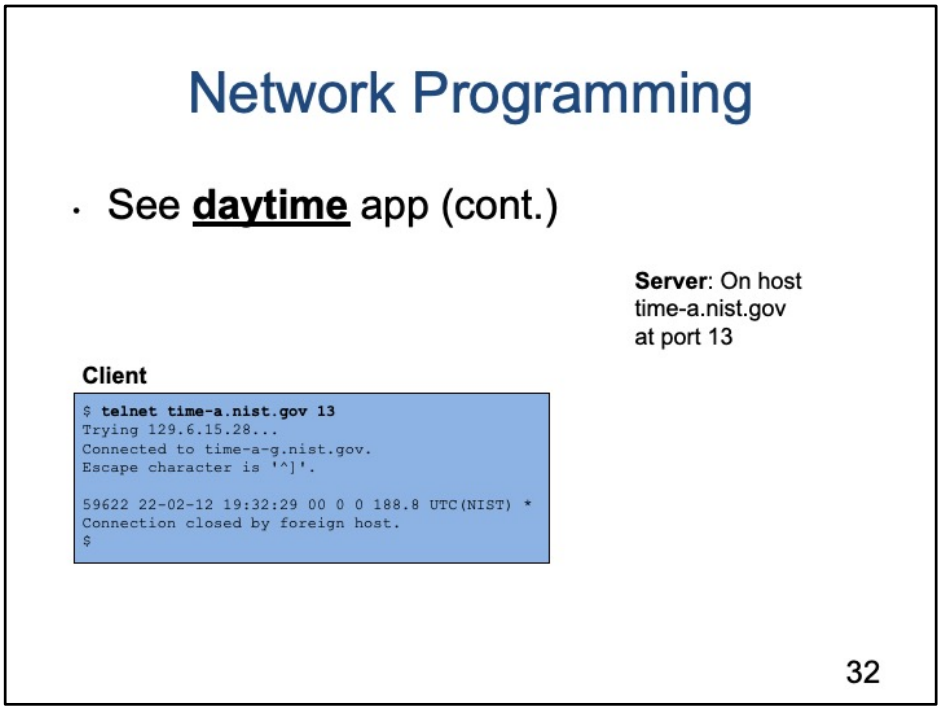
- Installing telnet (MS Windows)
  - Click Start
  - Select Control Panel
  - Click on Programs
  - Click on Programs and Features
  - Click on Turn Windows features on or off
  - Select the Telnet Client option
  - Click OK

Agenda

- Key concepts
- Client/server computing
- Client/server computing in COS 333
- **Network programming**

Network Programming

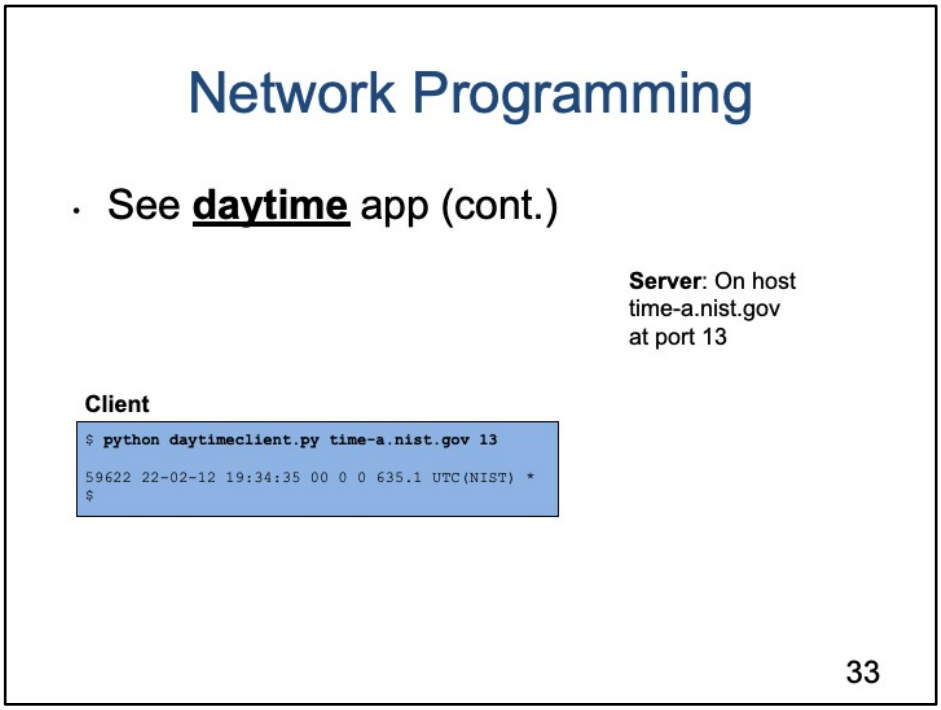
- See **daytime** app
  - The server's job:
    - Write current day and time to client
  - The client's job:
    - Read current day and time from server
    - Write current day and time to stdout



Network Programming

[see slide]

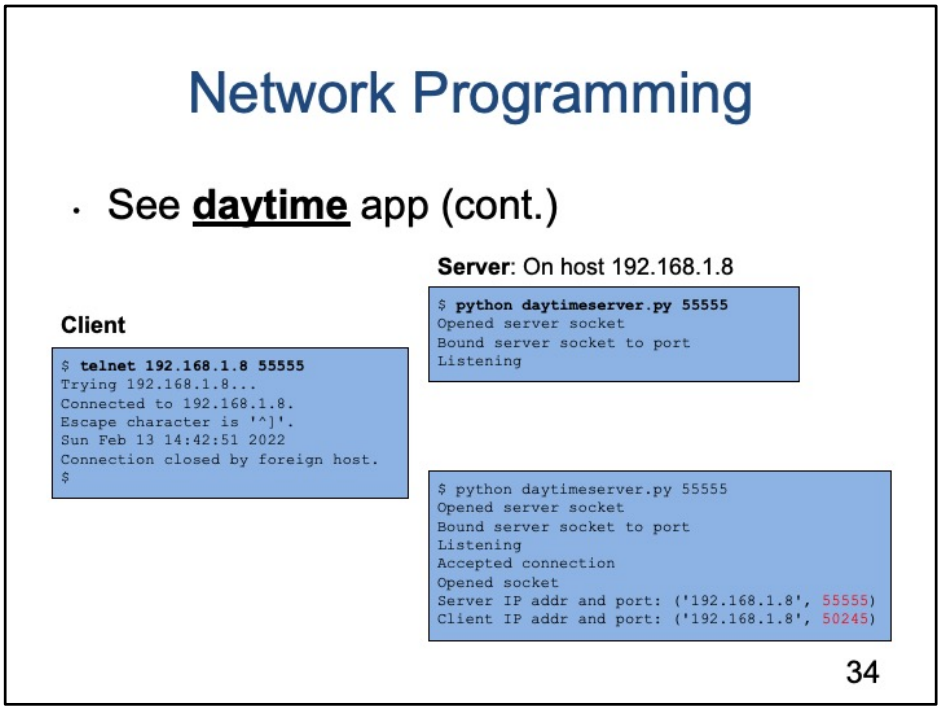
Client can be telnet
Server can be a third-party program
On host time-a.nist.gov at known port 13



Network Programming

[see slide]

Client can be a Python program that I compose
Server can be a third-party program



Network Programming

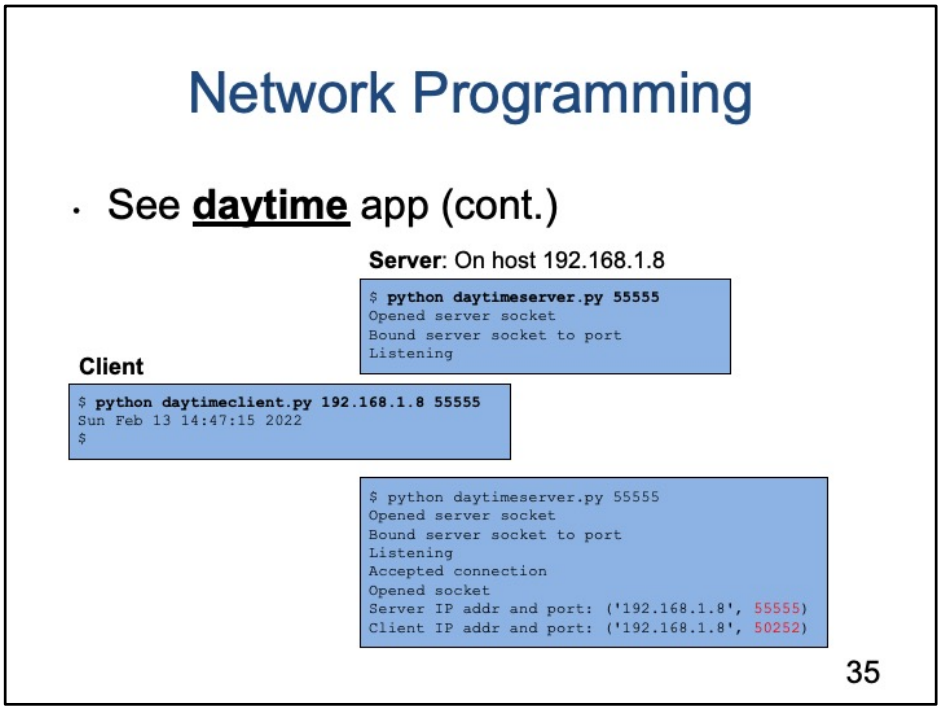
[see slide]

Client can be telnet
Server can be a Python program that I compose

Note server IP addr and port
Note client IP addr and port (client was running on same computer as server)

Note equivalent commands:

telnet 192.168.1.8 55555
telnet 127.0.0.1 55555
telnet localhost 55555



Network Programming

[see slide]

Client can be a Python program that I compose
Server can be a Python program that I compose

Network Programming

- Code structure

Baseline:

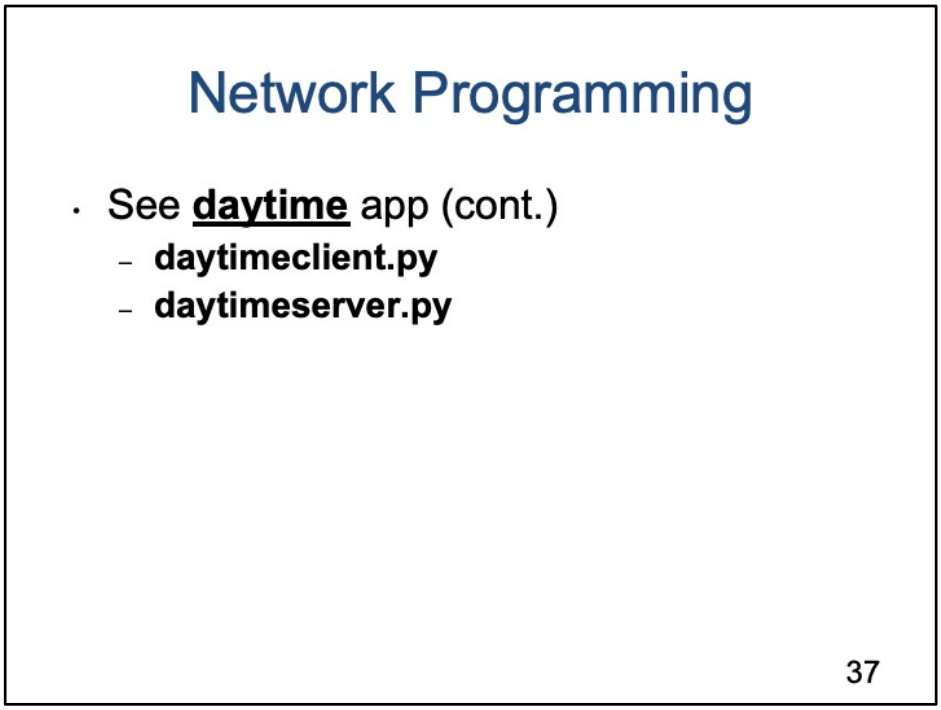
```
sock = socket(…)
…
…
sock.close()
```

Better:

```
sock = socket(…)
try:
    …
    …
finally:
    sock.close()
```

Better still:

```
with socket(…) as sock:
    …
    …
```



Network Programming

[see slide]

Code notes: daytimeclient.py

sock = socket()
Create a socket
sock.connect((host, port))
Connect the socket to the server process running on the specified host and listening at the specified port
flo = sock.makefile()
Wrap the socket with a flo that we can use to read from the socket
Implicit read via for statement
sock.close()
Performed implicitly by with statement

Code notes: daytimeserver.py

server_sock = socket()
Create a server socket
server_sock.setsockopt(...)
Unix-like systems: Allow immediate reuse of the specified port after this

process exits
MS Windows: Undesirable effect
Conditional execution

server_sock.bind((‘’, port))
Bind the port to the socket

server_sock.listen()
Start listening for connection requests

sock, client_addr = server_sock.accept()
Block until some client requests a connection

flo = sock.makefile(...)
Wrap the socket with a flo that we can use to write to the socket

flo.write()
Write characters to the socket

flo.flush()
Important; flush the cached characters into the socket

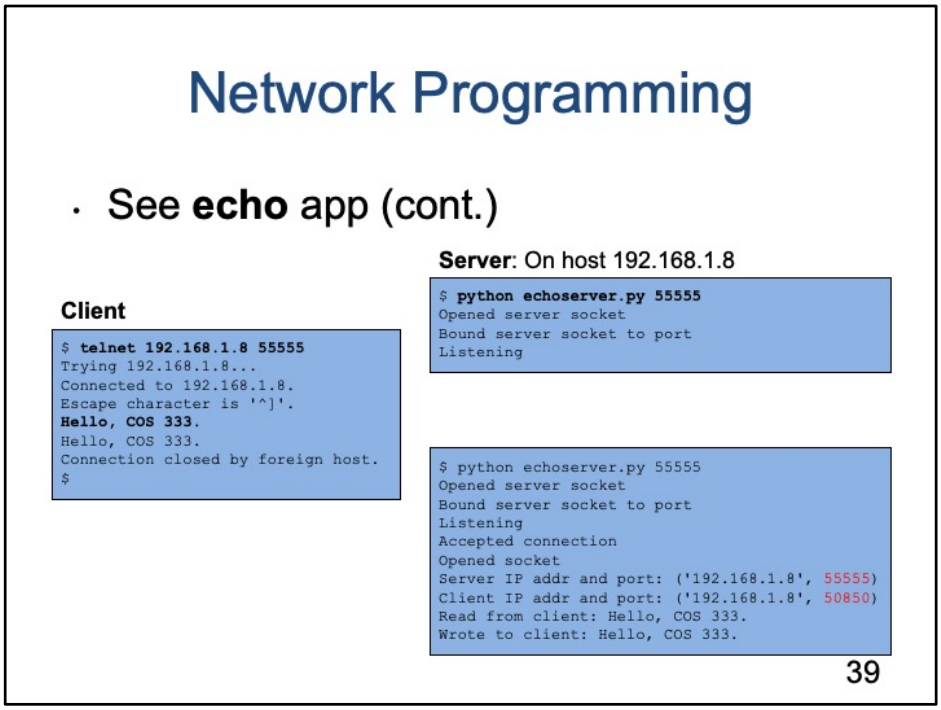
sock.close()
Performed implicitly by with statement

Loop infinitely (as servers often do)

Good idea to design server to write “log” messages

Network Programming

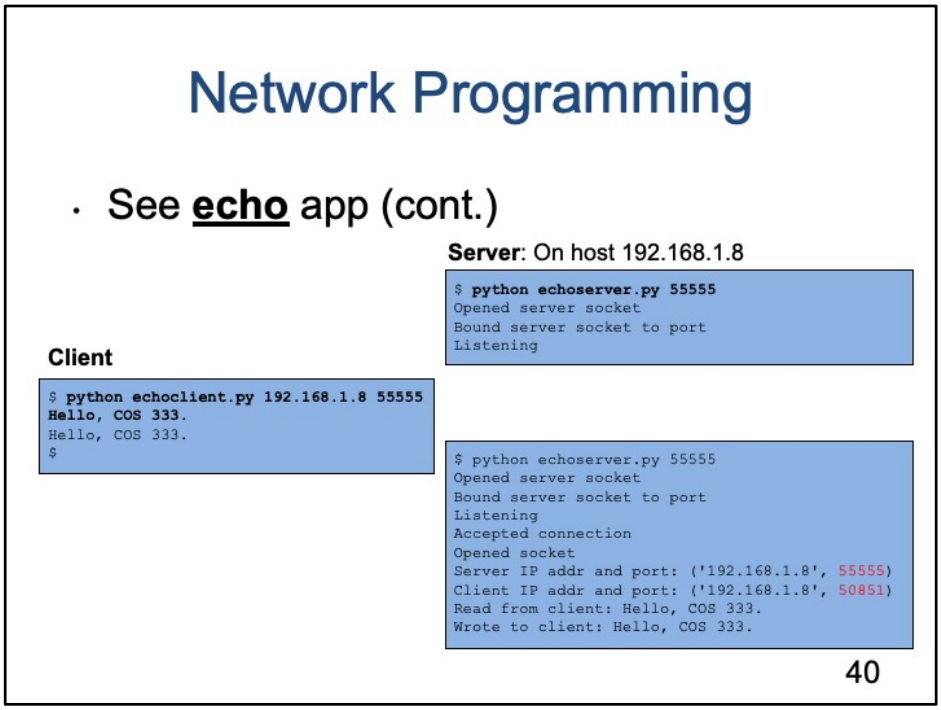
- See **echo** app
  - The client's job:
    - Read line from stdin, write line to server
  - The server's job:
    - Read line from client, write line to client
    - Loop
  - The client's job:
    - Read echoed line from server, write echoed line to stdout, exit



Network Programming

[see slide]

Client can be telnet
Server can be a Python program that I compose

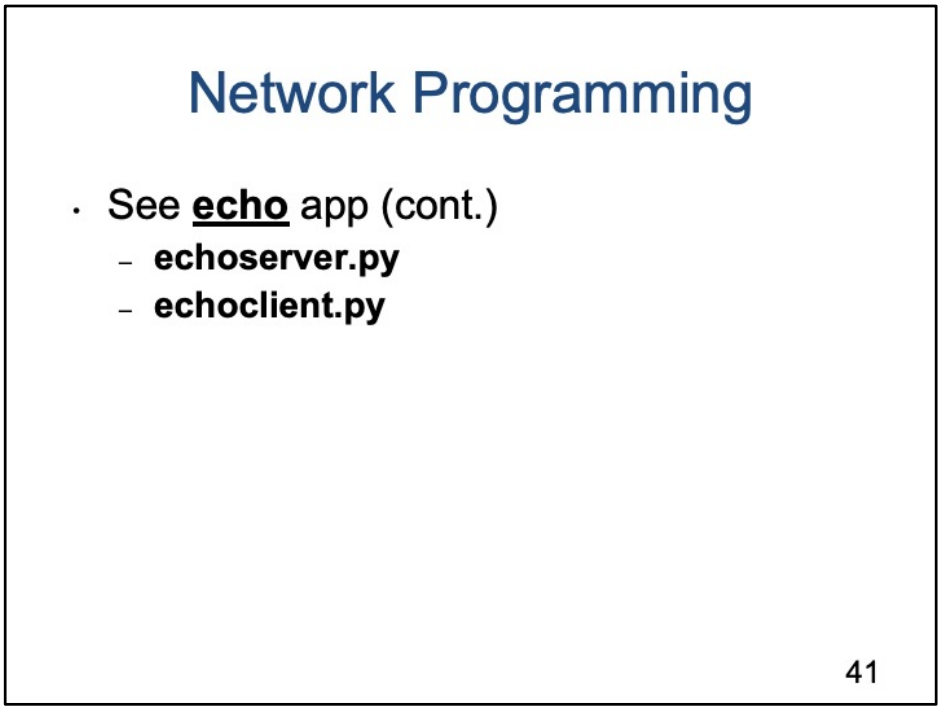


Network Programming

[see slide]

Client can be a Python program that I compose
Server can be a Python program that I compose

Server also could be a third-party program
But I can't find a publicly available echo server!



Network Programming

[see slide]

Code notes:

Sockets are bi-directional
Client writes to socket
Server reads from socket
Server writes to socket
Client reads from socket

Network Programming

- See **echo** app (cont.)
  - Note:
    - echoserver.py works with:
      - telnet
      - echoclient.py
      - An echo client written in Java, C, …
    - echoclient.py works with:
      - echoserver.py
      - An echo server written in Java, C, …

Summary

- We have covered:
  - Network programming key concepts
  - Client/server programming
  - Client/server programming in COS 333
  - Network programming in Python
    - How to compose a client
    - How to compose a server