

Network Programming (Part 1)

Copyright © 2026 by
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: daytime example
- Network programming: echo example

Key Concepts

- Network Address
 - ***Medium Access Control (MAC) address***
 - Example: 90:1b:0e:6a:32:26
 - ***Internet Protocol (IP) address***
 - Example: 128.112.136.67
 - Example: 140.180.223.22
 - Example: 127.0.0.1

To determine the IP address of your computer:

Mac or Linux: `ifconfig`

MS Windows: `ipconfig`

Key Concepts

- Network address (cont.)
 - *Domain name*
 - *Domain Name System (DNS)* converts to IP address
 - Example: cs.princeton.edu
 - Same as 128.112.136.67
 - Example: princeton.edu
 - 140.180.223.22
 - Example: localhost
 - Same as 127.0.0.1

Key Concepts

- See **ipaddress.py**

```
$ python ipaddress.py cs.princeton.edu
Host name: cs.princeton.edu
IP address: 128.112.136.67
$ python ipaddress.py princeton.edu
Host name: princeton.edu
IP address: 140.180.223.22
$ python ipaddress.py localhost
Host name: localhost
IP address: 127.0.0.1
$
```

See also: `nslookup` program

Key Concepts

- *Port*
 - A software abstraction
 - 16-bit integer (0 - 65535)

Key Concepts

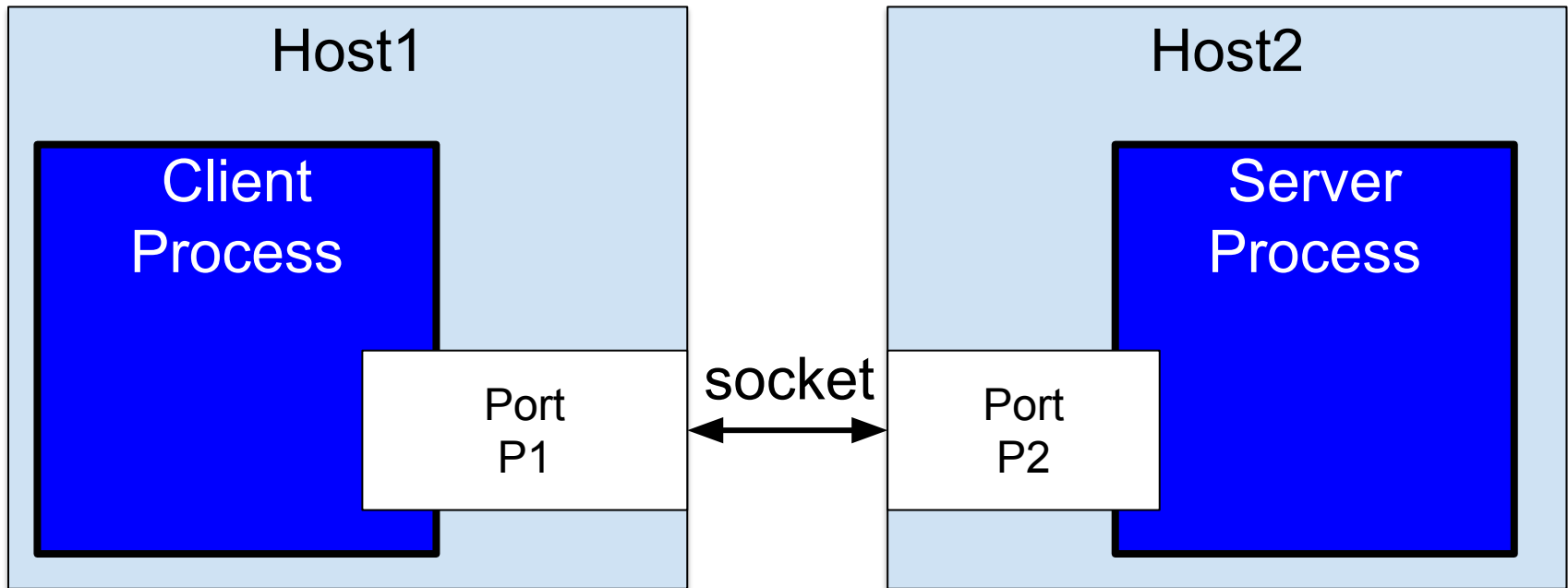
- ***Socket***
 - IP address + port
 - Used to implement...

Agenda

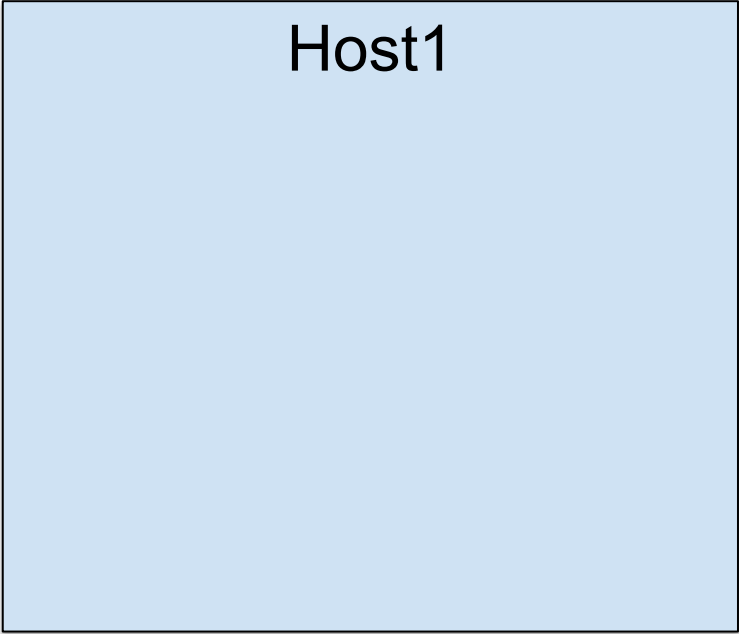
- Key concepts
- **Client/server computing**
- Client/server computing in COS 333
- Network programming: daytime example
- Network programming: echo example

Client/Server Computing

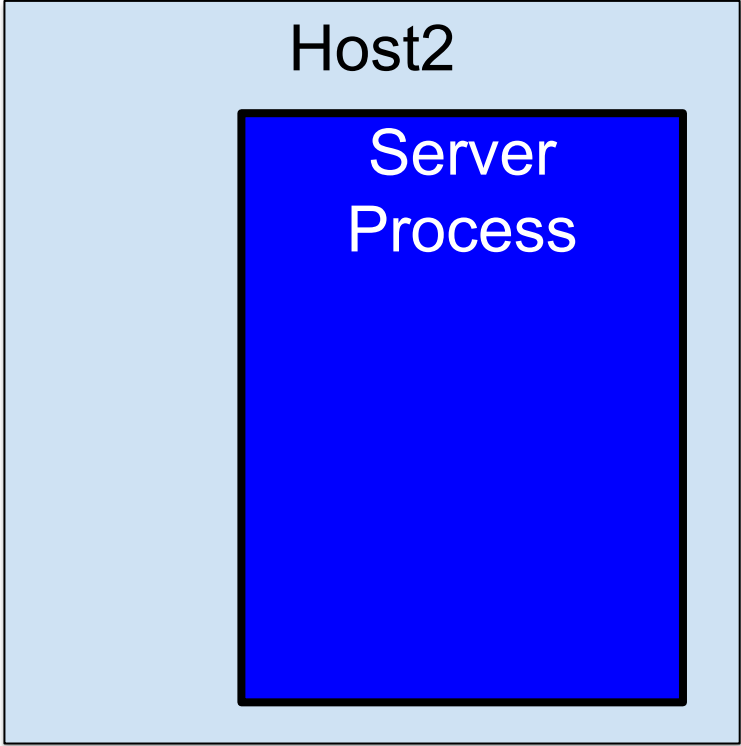
The big picture



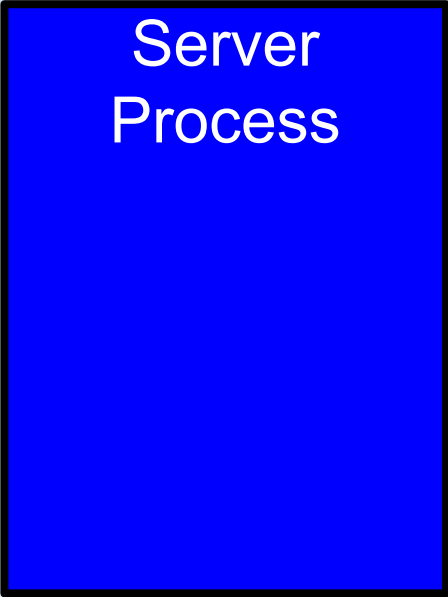
Client/Server Computing



Host1

A light blue square representing a host.

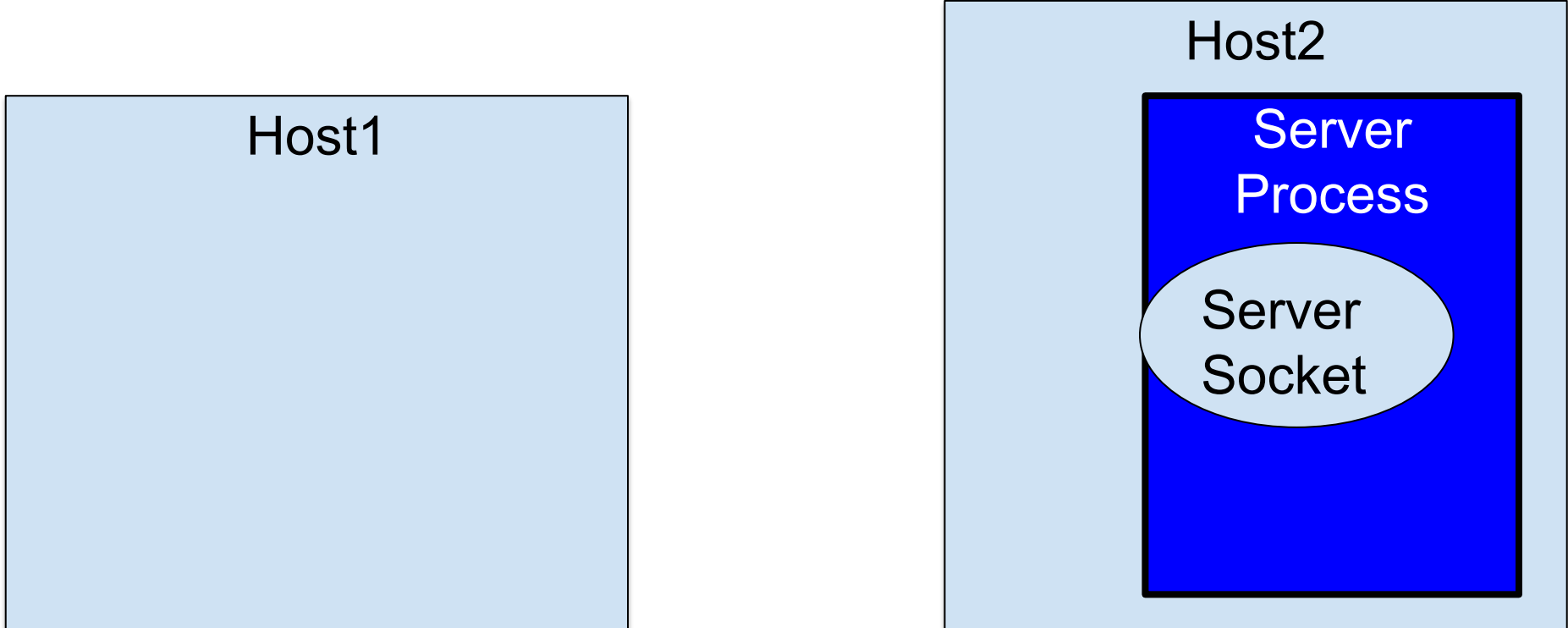
Host2

A light blue square representing a host.

Server
Process

A blue square representing a server process, nested within the Host2 square.

Client/Server Computing



The diagram illustrates a client-server architecture. On the left is a light blue square labeled 'Host1'. On the right is a larger light blue square labeled 'Host2'. Inside Host2 is a smaller blue square labeled 'Server Process'. Within the 'Server Process' is a light blue oval labeled 'Server Socket'.

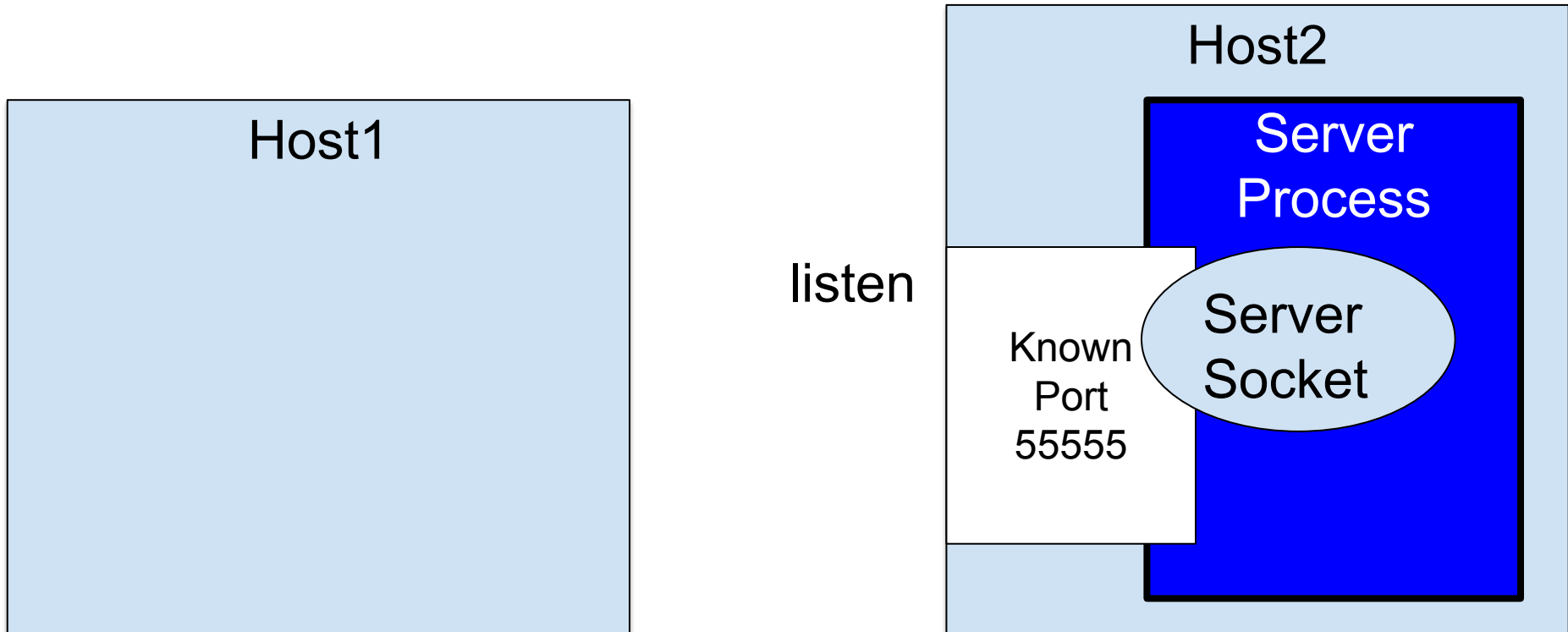
Host1

Host2

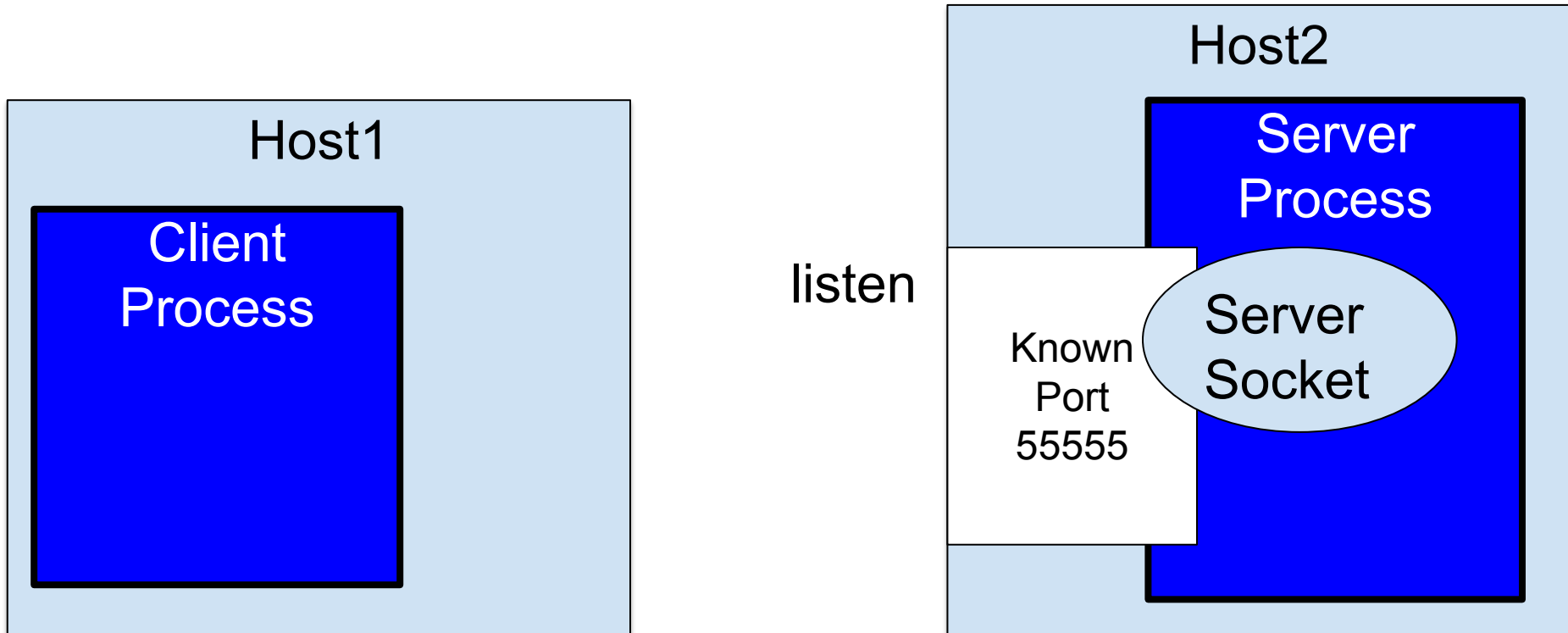
Server
Process

Server
Socket

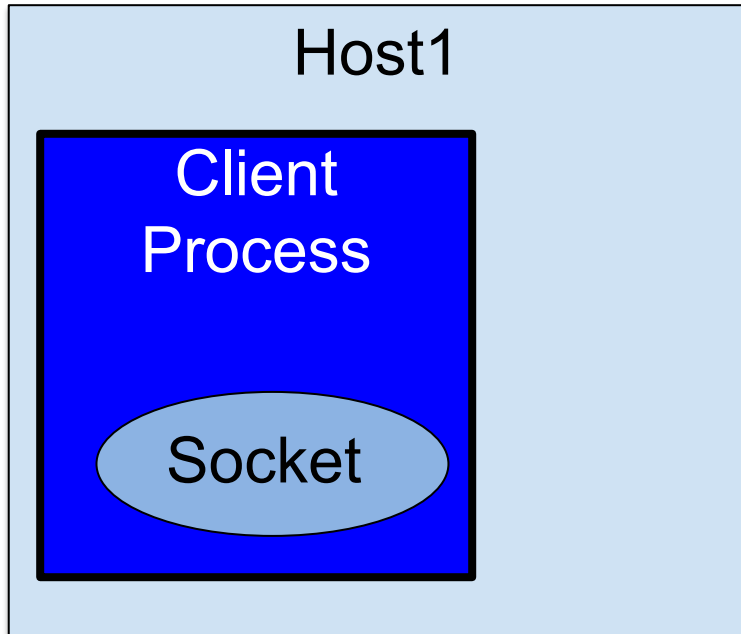
Client/Server Computing



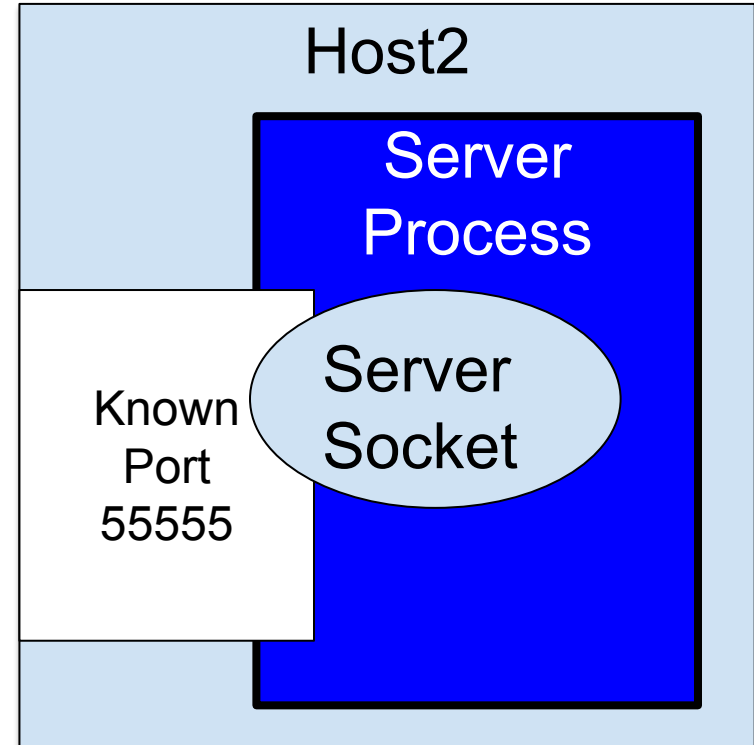
Client/Server Computing



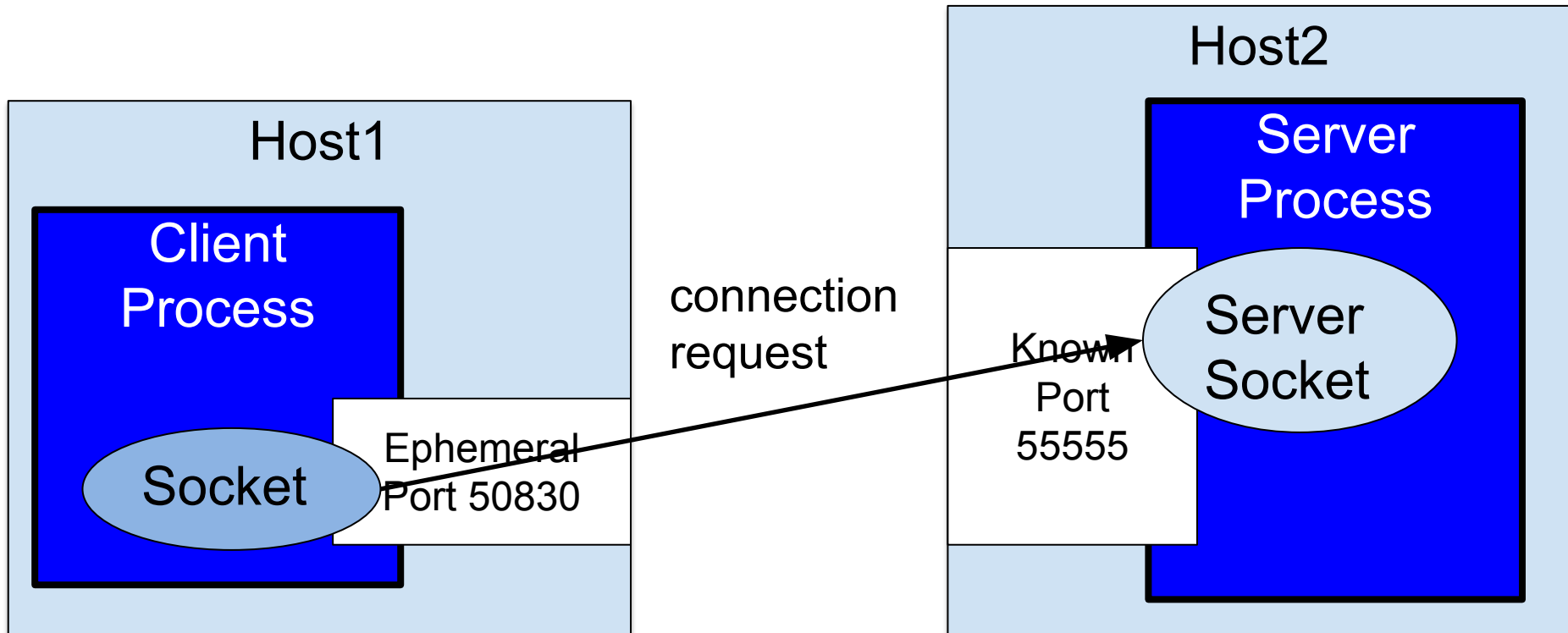
Client/Server Computing



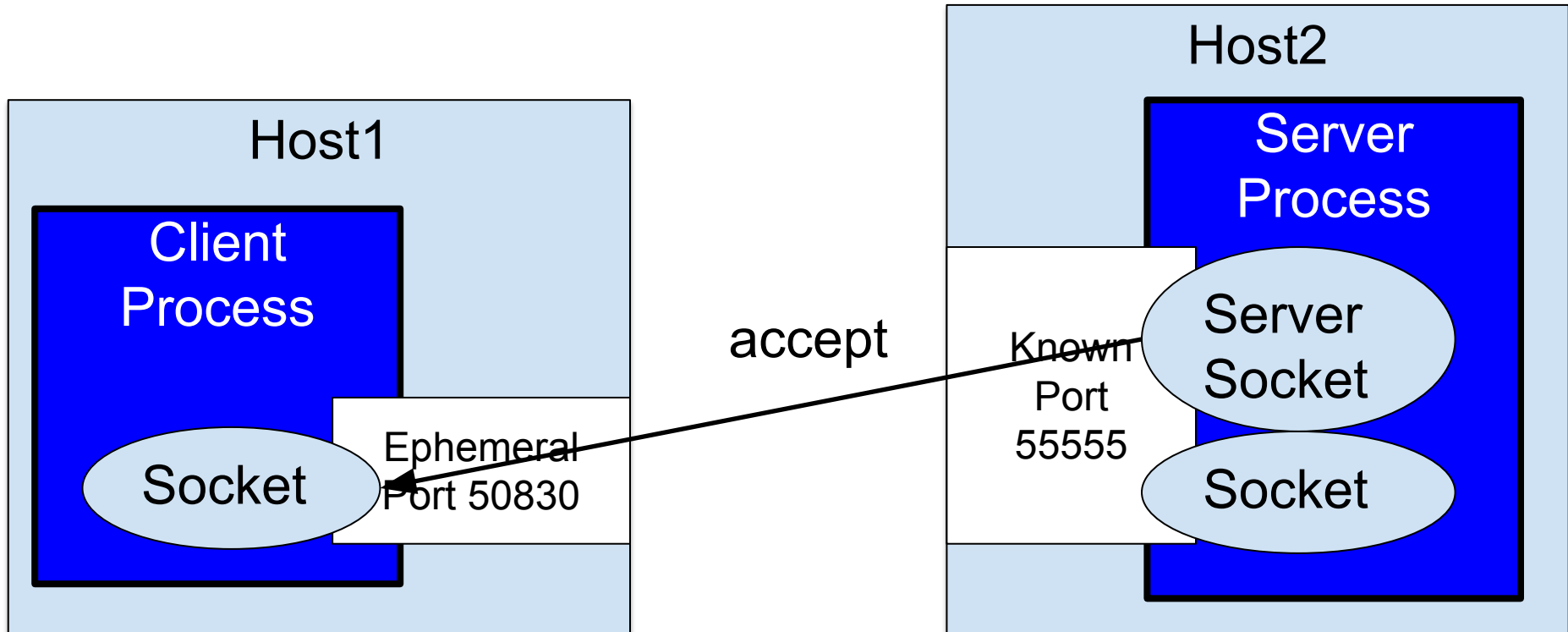
listen



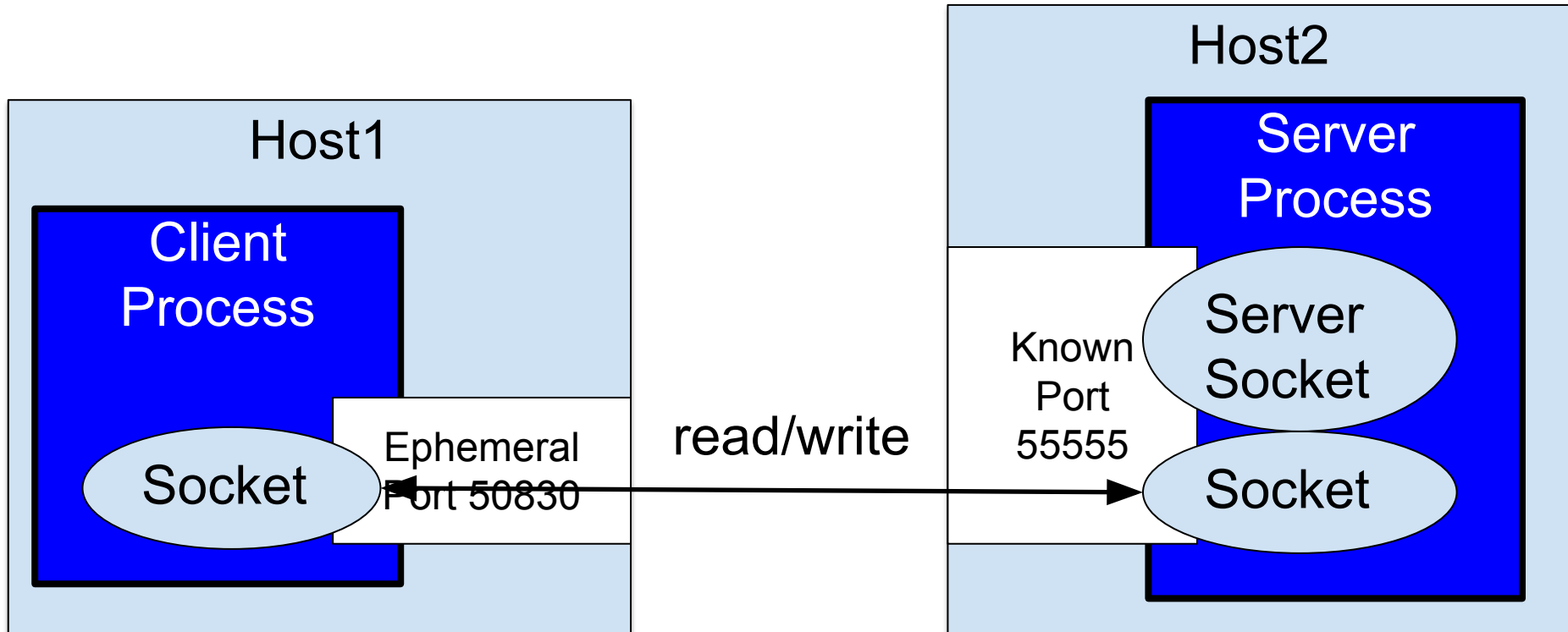
Client/Server Computing



Client/Server Computing



Client/Server Computing

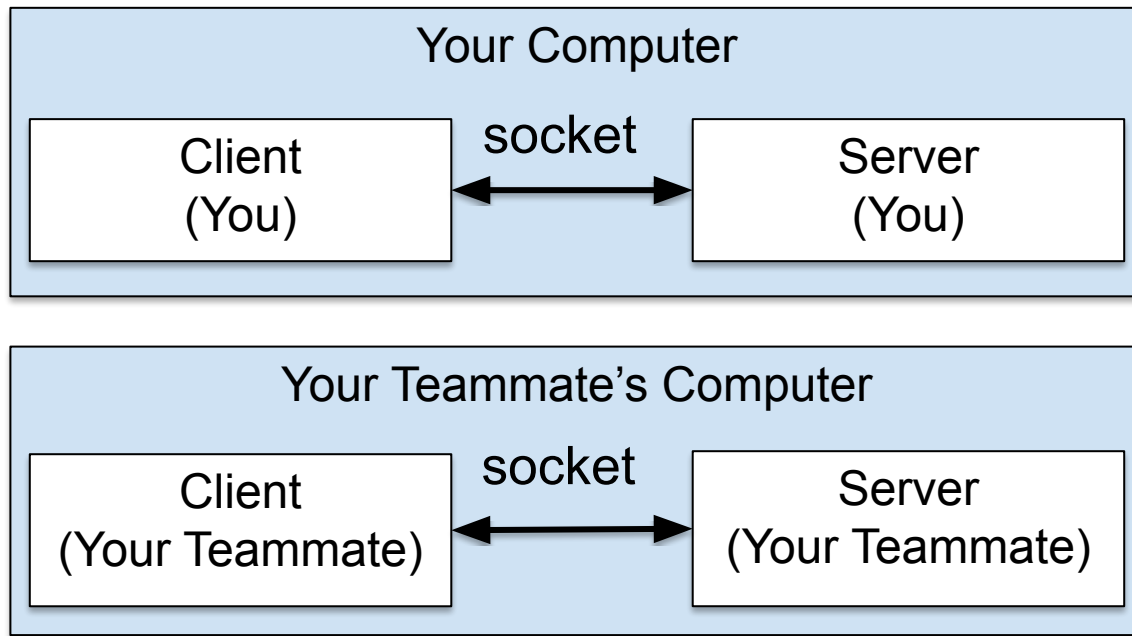


Agenda

- Key concepts
- Client/server computing
- **Client/server computing in COS 333**
- Network programming: daytime example
- Network programming: echo example

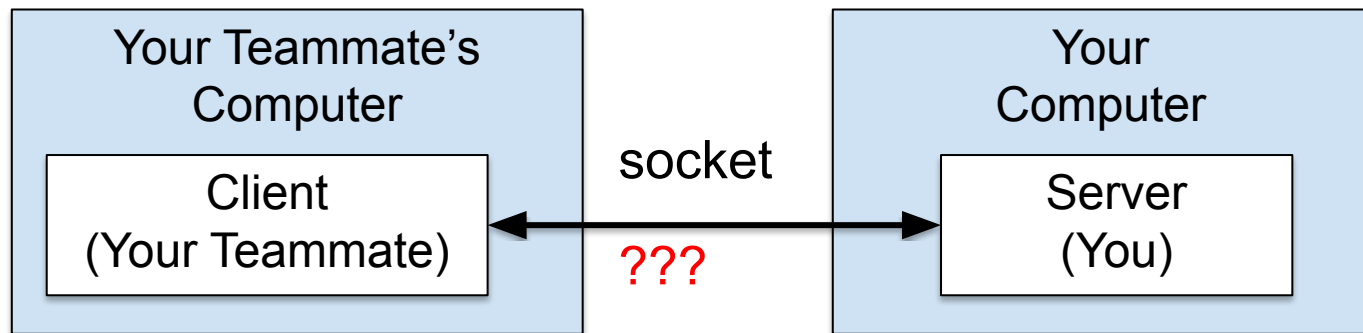
Client/Server in COS 333

Option 1: Run server on local computer
Run client on same local computer



Client/Server in COS 333

Option 2: Run server on local computer
Run client on different local computer



To determine IP address of your computer:

Mac/Linux: `ifconfig`

MS Windows: `ipconfig`

Won't work unless both computers are on Eduroam

Client/Server in COS 333

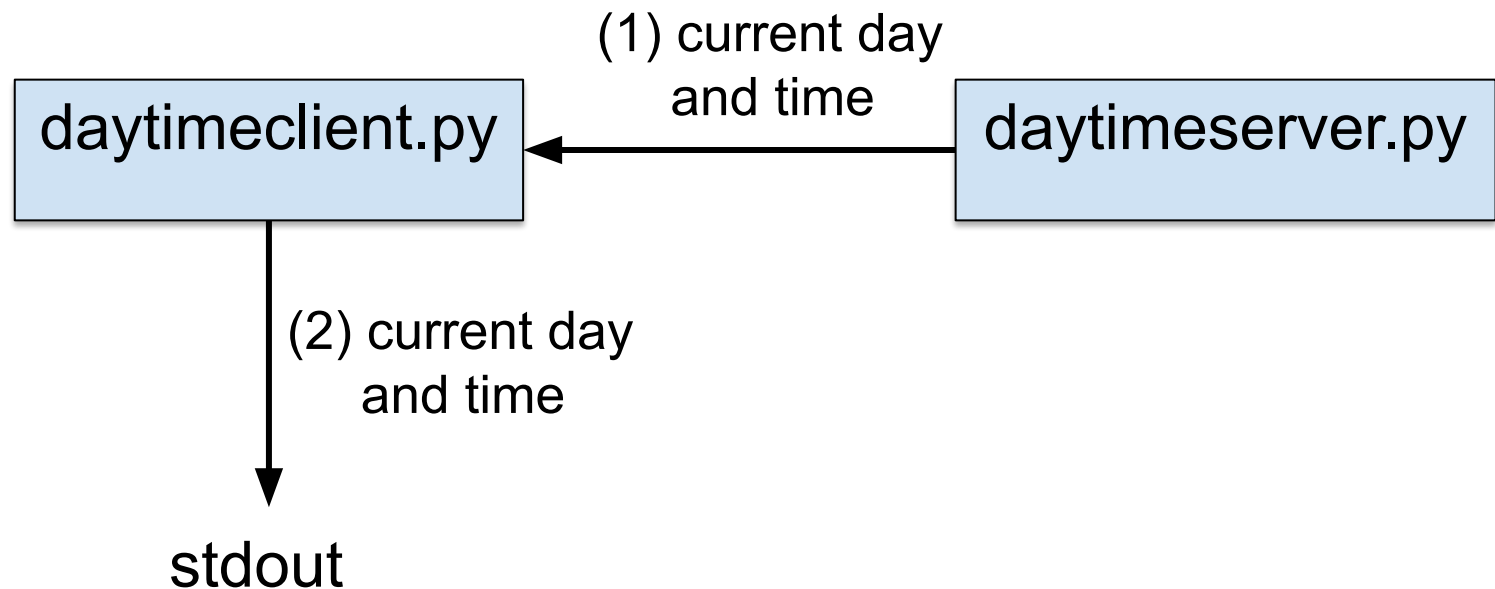
- Suggestions:
 - Use **option 1** during development
 - Use **option 2** to test network comm
 - Working alone =>
 - Use your computer and a COS 333 instructor's computer during office hours?

Agenda

- Key concepts
- Client/server computing
- Client/server computing in COS 333
- **Network programming: daytime example**
- Network programming: echo example

Network Programming: daytime

See daytime app



Network Programming: daytime

- See daytime app (cont.)

Server: On host 192.168.1.8

```
$ python server.py 55555 (1)
Opened server socket (1)
Bound server socket to port (1)
Listening (1)
Accepted connection (3)
Opened socket (3)
Server IP addr and port: ('192.168.1.8', 55555) (3)
Client IP addr and port: ('192.168.1.8', 50252) (3)
```

Client

```
$ python client.py 192.168.1.8 55555 (2)
Wed Feb 11 16:32:10 2026 (4)
$
```

Network Programming: daytime

- See daytime app (cont.)

Server: On host
time-a.nist.gov
at port 13

Client

```
$ python client.py time-a.nist.gov 13          (1)
                                                (2)
61082 26-02-11 21:30:43 00 0 0 34.3 UTC(NIST) * (2)
$
```

Network Programming: daytime

- See daytime app (cont.)
 - Code structure

Baseline:

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

Better:

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

Better still:

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

Network Programming: daytime

- See **daytime** app (cont.)
 - **client.py**
 - **server.py**

Network Programming: daytime

- See **daytimealt** app
 - **client.py**
 - **server.py**

Network Programming: daytime

- See daytimealt app (cont.)

Server: On host 192.168.1.8

```
$ python server.py 55555 (1)
Opened server socket (1)
Bound server socket to port (1)
Listening (1)
Accepted connection (3)
Opened socket (3)
Server IP addr and port: ('192.168.1.8', 55555) (3)
Client IP addr and port: ('192.168.1.8', 50252) (3)
```

Client

```
$ python client.py 192.168.1.8 55555 (2)
Wed Feb 11 16:32:10 2026 (4)
$
```

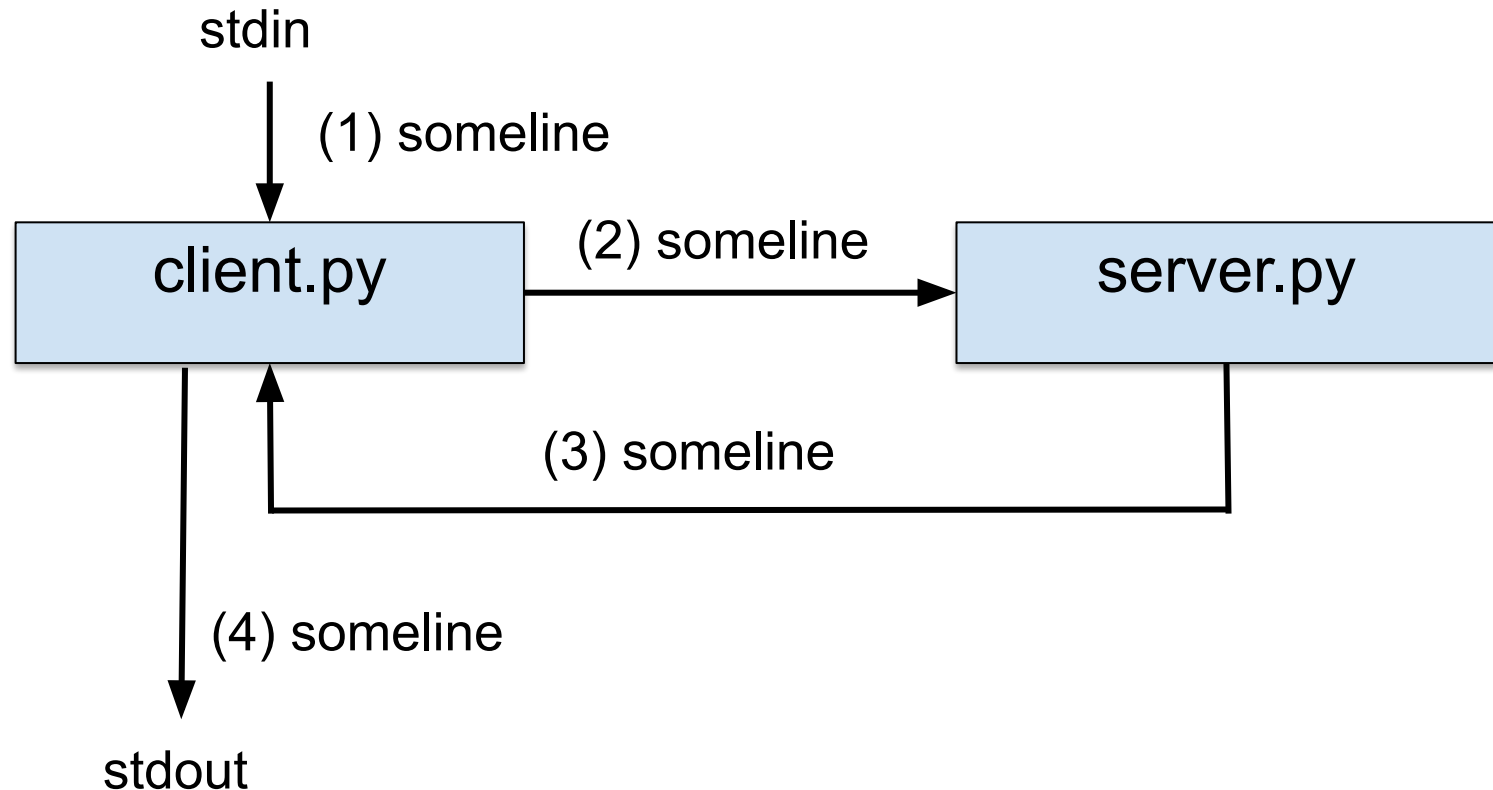
It works!

Agenda

- Key concepts
- Client/server computing
- Client/server computing in COS 333
- Network programming: daytime example
- **Network programming: echo example**

Network Programming: echo

See **echo** app



Network Programming: echo

- See echo app (cont.)

Server: On host 192.168.1.8

```
$ python server.py 55555 (1)
Opened server socket (1)
Bound server socket to port (1)
Listening (1)
Accepted connection (3)
Opened socket (3)
Server IP addr and port: ('192.168.1.8', 55555) (3)
Client IP addr and port: ('192.168.1.8', 50851) (3)
Read from client: Hello, COS 333. (3)
Wrote to client: Hello, COS 333. (3)
```

Client

```
$ python client.py 192.168.1.8 55555 (2)
Hello, COS 333. (2)
Hello, COS 333. (4)
$
```

Network Programming: echo

- See **echo** app (cont.)
 - **server.py**
 - **client.py**

Network Programming: echo

- See **echoaltbad** app
 - **server.py**
 - **client.py**

Network Programming: echo

- See echoaltbad app (cont.)

Server: On host 192.168.1.8

```
$ python server.py 55555 (1)
Opened server socket (1)
Bound server socket to port (1)
Listening (1)
Accepted connection (3)
Opened socket (3)
Server IP addr and port: ('192.168.1.8', 55555) (3)
Client IP addr and port: ('192.168.1.8', 50851) (3)
```

Client

```
$ python client.py 192.168.1.8 55555 (2)
Hello, COS 333. (2)
```

It fails!

With sockets, do line-oriented I/O

Network Programming: echo

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

Lecture Summary

- This lecture 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