

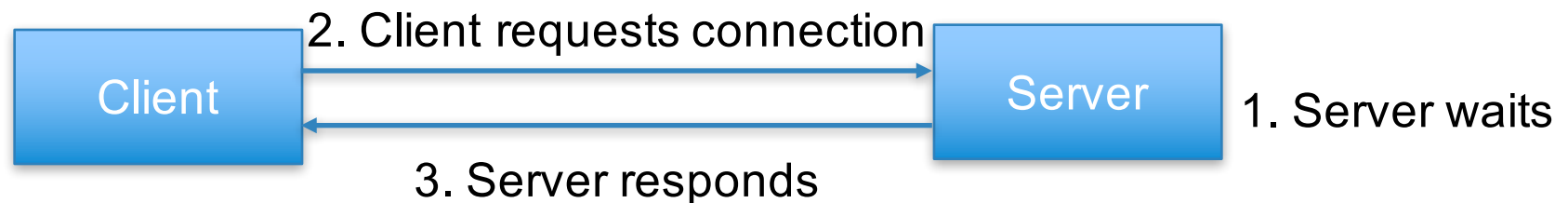
PIC 16 Winter 2019

Networking

Based on the slides of Matt Haberland

Client/Server

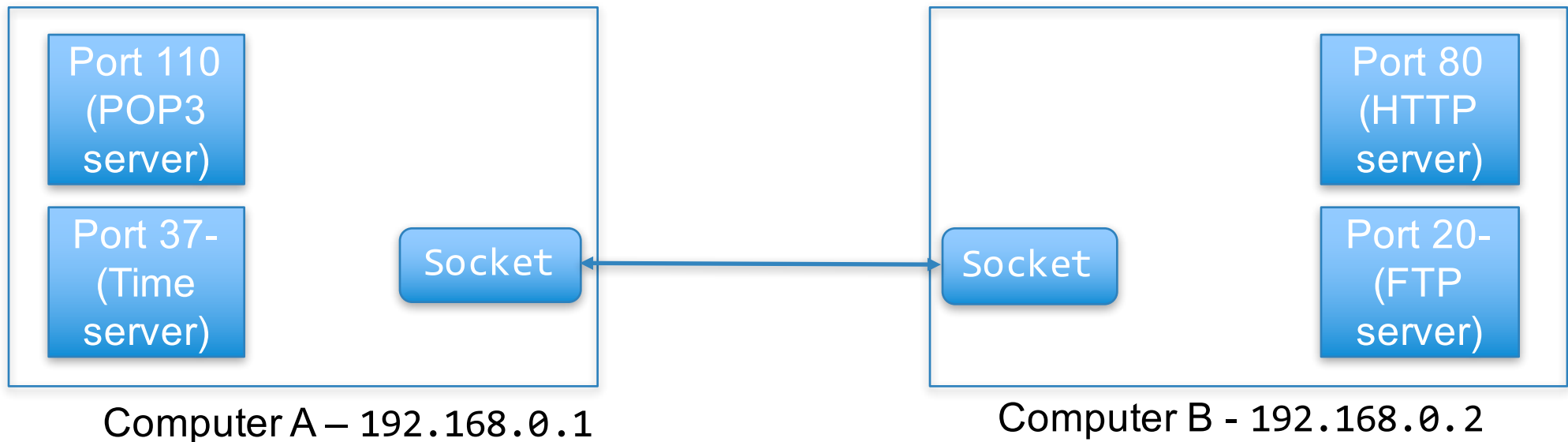
- What is a client/server?
 - Client – the device that initiates the connection
 - Server – the device that accepts the connection
- In Transmission Control Protocol (TCP) networking, one computer requests a connection with another computer.
- We assign the names client/server accordingly:
 - Server waits for connection requests
 - A client requests a connection with a server
 - The server responds



- Once a connection is established, either client or server can send and receive data

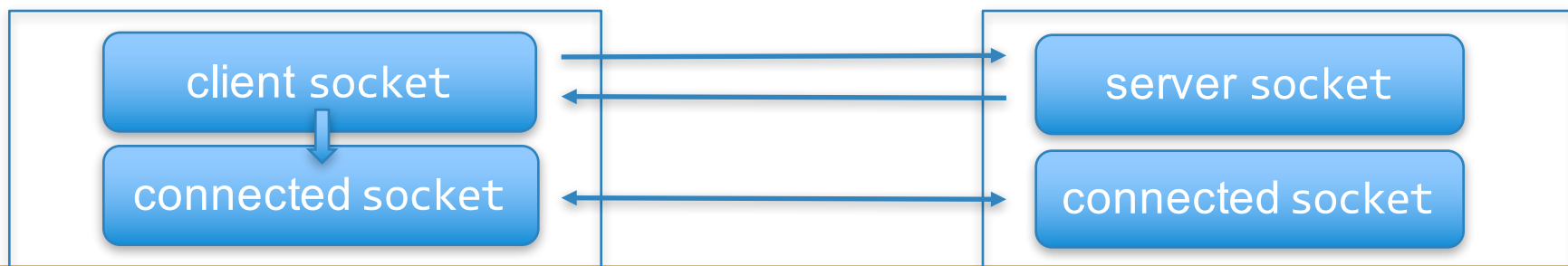
Terminology

- An IP address is a number like 127.0.0.1 that serves as a unique identifier for a computer on a network
- A socket is an endpoint of two-way communication between computers on a network
- A port number identifies which application on a computer gets the data received by a socket



Sockets

- While there is only one socket class, make a distinction between the states a socket can be in:
 - A *server* socket
 - waits for connection requests from *client* sockets
 - creates a new *connected* socket when it accepts
 - A *client* socket
 - requests a connection with a *server* socket
 - turns into a *connected* socket when connection is accepted
 - A *connected* socket
 - can send and receive data



Establishing a connection

1. Server makes a *server* socket, binds to a “port”, and listens

```
s = socket.socket()  
s.bind(("", port))  
s.listen(5)
```

A socket, like something
that gets plugged into

A port is just a number 0-65535
0-1023 are reserved

“max connections”, according to documentation, but even 0 works

2. Client makes a *client* socket to connect to the server socket

```
s = socket.socket()  
s.connect(("localhost", port))
```

The port number on which
the server is listening

The IP address of the server. “127.0.0.1” or “localhost” means
the same computer as the client (“this computer”).

3. The *server* makes a regular socket once the client connects

```
s, addr = s.accept()
```

connected socket
information about client

accept method waits until it receives a connection
request, then returns connected socket to
communicate on. Connection is complete;
connected sockets are used to communicate.

Communicating with connected sockets

- Once the connection is established a connected socket can:
 - `send(str_to_send)` ← A string
 - `recv(no_of_bytes)` ← maximum # of bytes to receive at once
 - What type do you think this method returns?
 - The string sent by the other connected socket, of course
 - `close()` ← terminates the connection

listen first, then connect

- Can a server socket accept without first listening?
 - Nope. Error in server program.
- What do you think happens when a client socket tries to connect when no server socket is listening and accepting?
 - Error in client program.
- What do you think happens when a connected socket sends a message without the other connected socket ready to recv?
 - Nothing. The message will get sent, but not received.
- What do you think happens when connected socket is ready to recv but no message is sent?
 - By default, the socket blocks (waits) indefinitely. You can set a timeout so that program execution can continue.

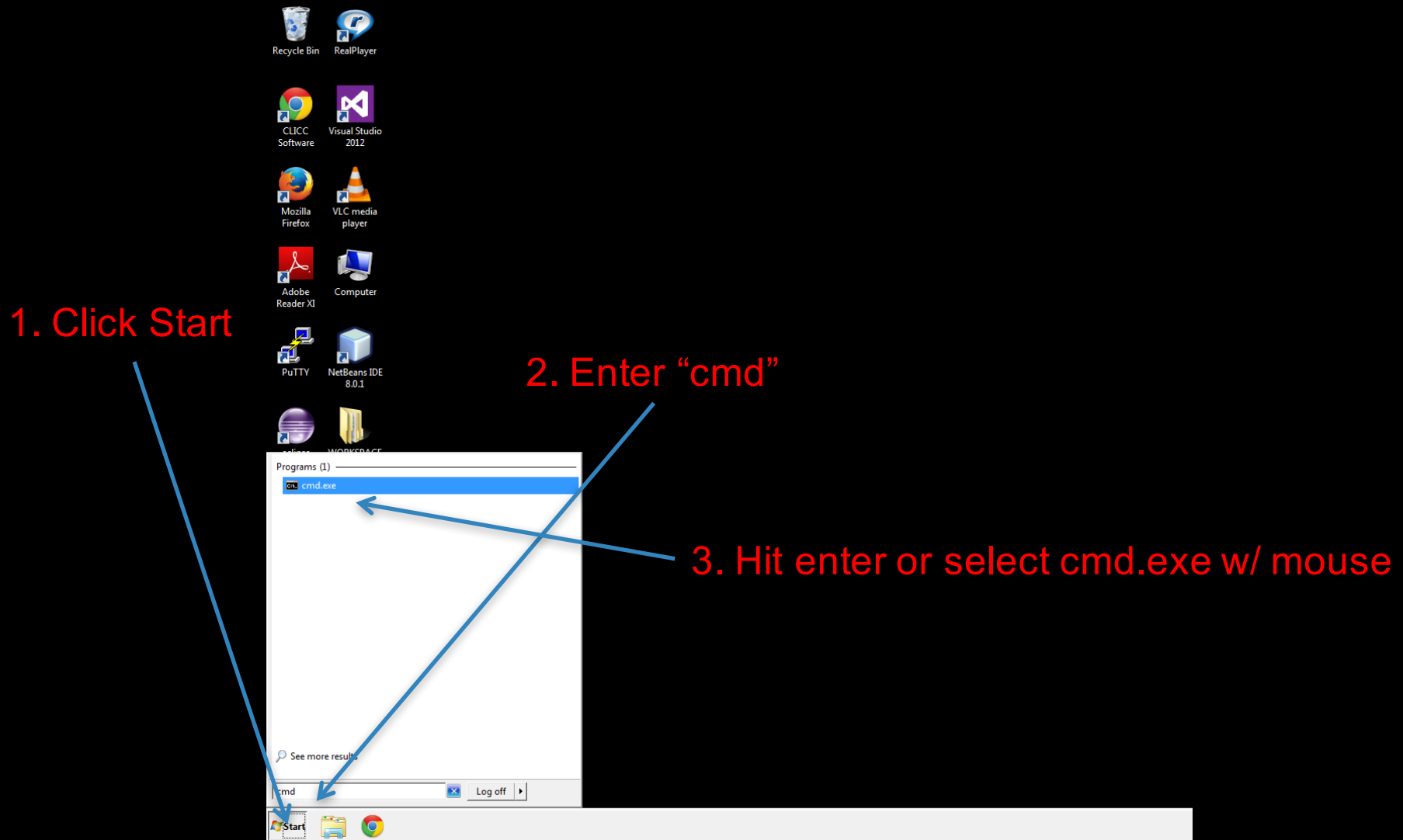
Hello World!

Today we're going to create a "Hello World!" program in Python, except that "Hello World!" will print to your *partner's* computer's console rather than your own!

The 0th step is to log on to the PIC lab computers using your usual Bruin Online username and password. If these do not work for you, please discuss with the PIC lab attendant. If the problem cannot be resolved, please work with a (potentially new) friend.

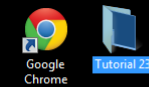
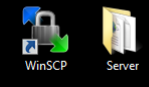
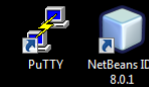
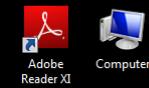
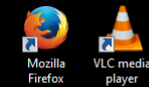
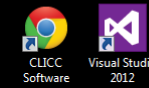
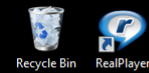
The following slides provide a graphical tutorial.

Step 1: Open Command Prompt



The command prompt is a handy way to invoke operating system commands.

Step 2: Get your IP address



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\haberland>ipconfig

Windows IP Configuration

Ethernet adapter Local Area Connection:

    Connection-specific DNS Suffix  . : 
    Link-local IPv6 Address . . . . . : fe80::b553:1bd:9813:2c94::12
    IPv4 Address. . . . . : 128.97.12.120
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 128.97.12.254

Tunnel adapter 6TO4 Adapter:

    Connection-specific DNS Suffix  . : 
    IPv6 Address. . . . . : 2002::8061:c78::8061:c78
    Default Gateway . . . . . : 

Tunnel adapter Local Area Connection* 12:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . : 

Tunnel adapter isatap.{CE74A3945-774D-4A40-AEBB-6279DFA554B0}:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . : 

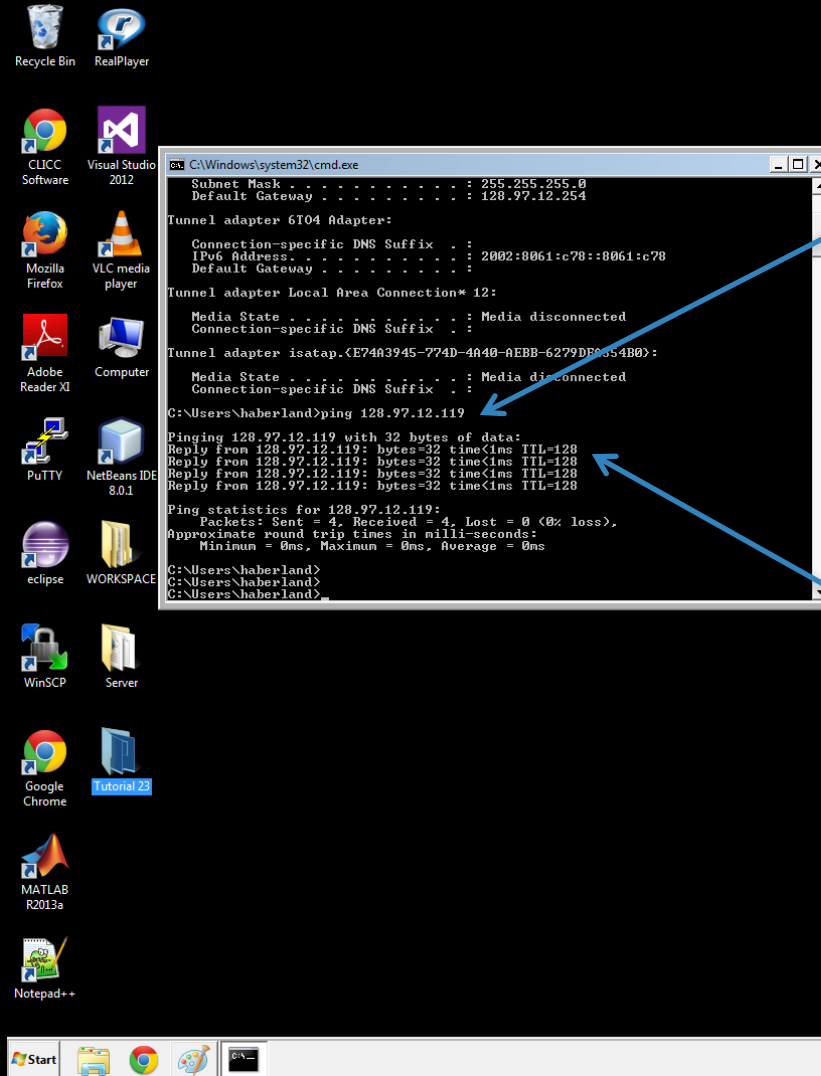
C:\Users\haberland>
```

1. Type “ipconfig” at the prompt and hit enter

2. Note the IPv4 address, the unique number assigned to your computer. Other computers can use this number, like a telephone number, to talk to your computer.

`ipconfig` is a Windows command for viewing and setting internet protocol settings

Step 3: Ping your partner



```
C:\Windows\system32\cmd.exe
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 128.97.12.254

Tunnel adapter 6T04 Adapter:
Connection-specific DNS Suffix . : 
IPv6 Address. . . . . : 2002:0061:c78::0061:c78
Default Gateway . . . . . : 

Tunnel adapter Local Area Connection* 12:
Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . : 
Tunnel adapter isatap.{CE74A3945-774D-4A40-AEBB-6279DE0554B0}:
Media State . . . . . : Media disconnected
Connection-specific DNS Suffix . : 

C:\Users\haberland>ping 128.97.12.119

Pinging 128.97.12.119 with 32 bytes of data:
Reply from 128.97.12.119: bytes=32 time<1ms TTL=128
Reply from 128.97.12.119: bytes=32 time<1ms TTL=128
Reply from 128.97.12.119: bytes=32 time<1ms TTL=128
Reply from 128.97.12.119: bytes=32 time<1ms TTL=128

Ping statistics for 128.97.12.119:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\Users\haberland>
C:\Users\haberland>
```

1. Type “ping ” followed by your partner’s IP address and hit enter.

2. You should get replies quickly! This indicates that your computer can talk to your partner’s computer (at least on the port used by the ping application. A firewall might block *other* ports, but at least your computer can “see” the other one)

Ping is a command to test whether your computer can talk to another

Step 5: Write a Server

- One of your computers will need to run a server program. The server needs to:
 - Create a server socket and bind it (any) unreserved port
 - Prepare the server socket to listen
 - Print “Waiting for connection...,” to the console
 - accept any incoming connection request, returning a connected socket (and “the address bound to the socket on the other end of the connection”)
 - Print “Connection request accepted!” to the console
 - send a message, “Hello World!”
 - Print “Message sent!” to the console
 - Close the socket
- Refer to the readings only if absolutely necessary! Try to figure it out from the names of functions above and socket documentation.

Step 6: Write a Client

- The other computer will need to run a client program. The client needs to:
 - Create a client socket
 - Print “Connecting to server...” to the console.
 - Have the socket connect to the other computer at its IPv4 address, and specifically, the port associated with the server program.
 - Print “Waiting for reply...” to the console.
 - Have the socket recv the message from the server program and print it to the console
 - Close the socket.
- Again, please refer to the readings only if absolutely necessary!