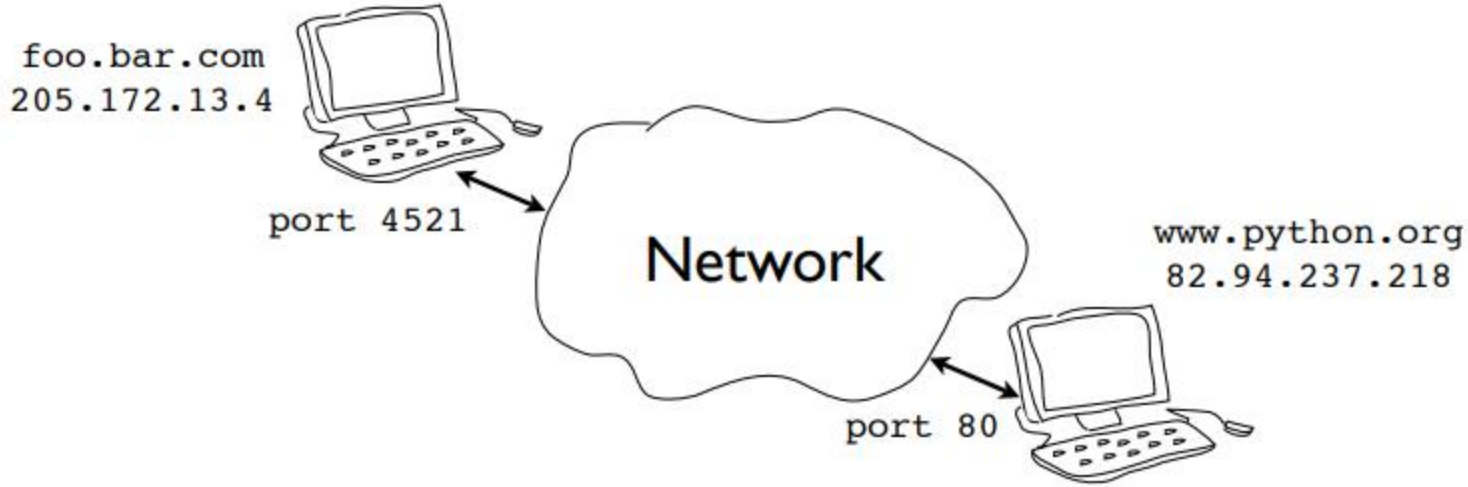
Tutorial 2: Python Socket Programming and SSL

Part of the content is borrowed from David M. Beazley's tutorial on python socket programming

Network Addressing

- Machines have a hostname and IP address
- Programs/services have port numbers



Standard Ports

- Ports for common services are preassigned

  21 FTP
  22 SSH
  23 Telnet
  25 SMTP (Mail)
  80 HTTP (Web)
  110 POP3 (Mail)
  443 HTTPS (web)

- Other port numbers may just be randomly assigned to programs by the operating system

Connections

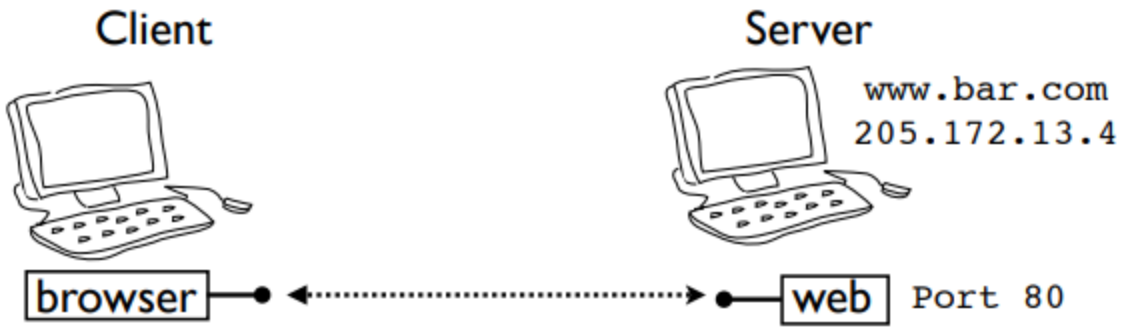
- Each endpoint of a network connection is always represented by a host and port #

- In Python you write it out as a tuple (host,port) ("www.python.org",80) ("205.172.13.4",443)

- In almost all of the network programs you'll write, you use this convention to specify a network address

Client/Server Concept

- Each endpoint is a running program
- Servers wait for incoming connections and provide a service (e.g., web, mail, etc.)
- Clients make connections to servers



Using Telnet or Netcat

- As a debugging aid, telnet or netcat can be used to directly communicate with many services

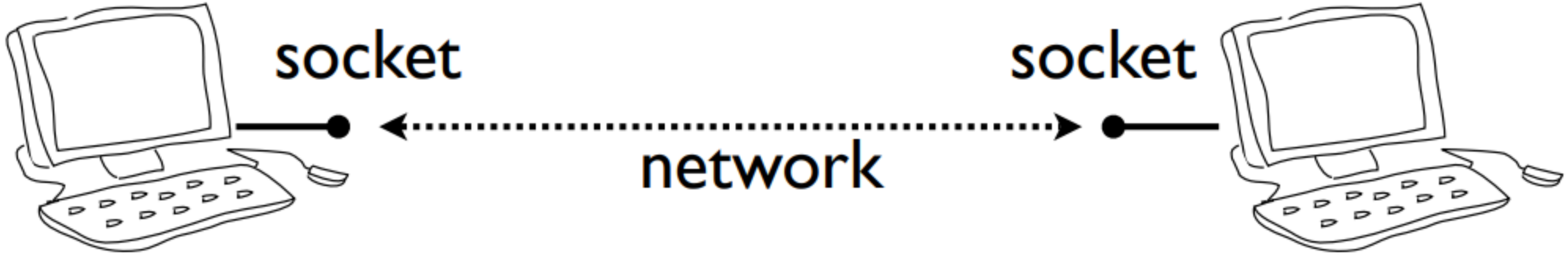
  telnet hostname portnum

- Example:

```
shell % telnet www.google.ca 80
Trying 2607:f8b0:400a:808::2003…
Connected to www.google.ca.
Escape character is '^]'.
GET /index.html HTTP/1.0

HTTP/1.1 200 OK
Date:  Fri, 15 Jan 2021 21:30:45 GMT
 …
```

Sockets

- Programming abstraction for network code
- Socket: A communication endpoint



Socket Basics

- To create a socket

```
import socket
s = socket.socket(addr_family, type)
```

- Address families

  socket.AF_INET Internet protocol (IPv4)
  socket.AF_INET6 Internet protocol (IPv6)

- Socket types

  socket.SOCK_STREAM Connection based stream (TCP)
  socket.SOCK_DGRAM Datagrams (UDP)

- Example

```
from socket import *
s = socket(AF_INET,SOCK_STREAM)
```

Using a Socket

- Creating a socket is only the first step
- Further use depends on application
- Server Listen for incoming connections
- Client Make an outgoing connection

TCP Client

- How to make an outgoing connection

```
from socket import *
s = socket(AF_INET,SOCK_STREAM)
s.connect(("www.google.ca",80)) # Connect
s.send("GET /index.html HTTP/1.0\n\n") # Send request
data = s.recv(10000) # Get response s.close()
```

- s.connect(addr) makes a connection
- Once connected, use send(),recv() to transmit and receive data
- close() shuts down the connection

Test your code

- Upload your code to linux.csc.uvic.ca
- ssh netlinkid@linux.csc.uvic.ca
- python3 smartclient.py www.google.ca

```
zhiming@DESKTOP-M225BQ0:/mnt/c/Users/Zhiming$ ssh zhiminghuang@linux.csc.uvic.ca
The authenticity of host 'linux.csc.uvic.ca (142.104.71.175)' can't be established.
ECDSA key fingerprint is SHA256:liBAris7BpeTxa0jJhx/fYa6Q6uwgnq/MdW8Y1XExNI.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'linux.csc.uvic.ca,142.104.71.175' (ECDSA) to the list of known hosts.
zhiminghuang@linux.csc.uvic.ca's password:

=====================================================================
Welcome to COSI Computing Services

   You are logged onto cluster linux.csc.uvic.ca, node linux204
   Ubuntu 20.04.3 LTS - (GNU/Linux 5.4.0-94-generic )

   If you need assistance:
    * Documentation: linux manual pages (man <command>)
    * Support: https://itsupport.csc.uvic.ca
    * Report Issues: https://radar.csc.uvic.ca
   All activity on this system is monitored and governed by University
   policy (http://www.csc.uvic.ca/courseinfo/policies/lab_usage.html) and
   https://www.uvic.ca/universitysecretary/assets/docs/policies/IM7200_6030_.pdf

   IT Support - Computer Science - University of Victoria
   itsupport@csc.uvic.ca
   policy:
   NEWS:
     * All Department computers authenticate with UVIC Netlink
     * Department servers have been upgraded to Ubuntu 20.04
     * For more info:
       https://itsupport.cs.uvic.ca/accounts/netlink/
```

SSL Wrapped Sockets

SSL Wrapped sockets

```
import socket
import ssl

#Create a default context
context = ssl.create_default_context()

#Wrap socket
conn = context.wrap_socket(socket.socket(socket.AF_INET),
server_hostname="www.google.ca")
```

Check HTTP2

- Before connection:
- 7. context.set_alpn_protocols(['http/1.1','h2'])

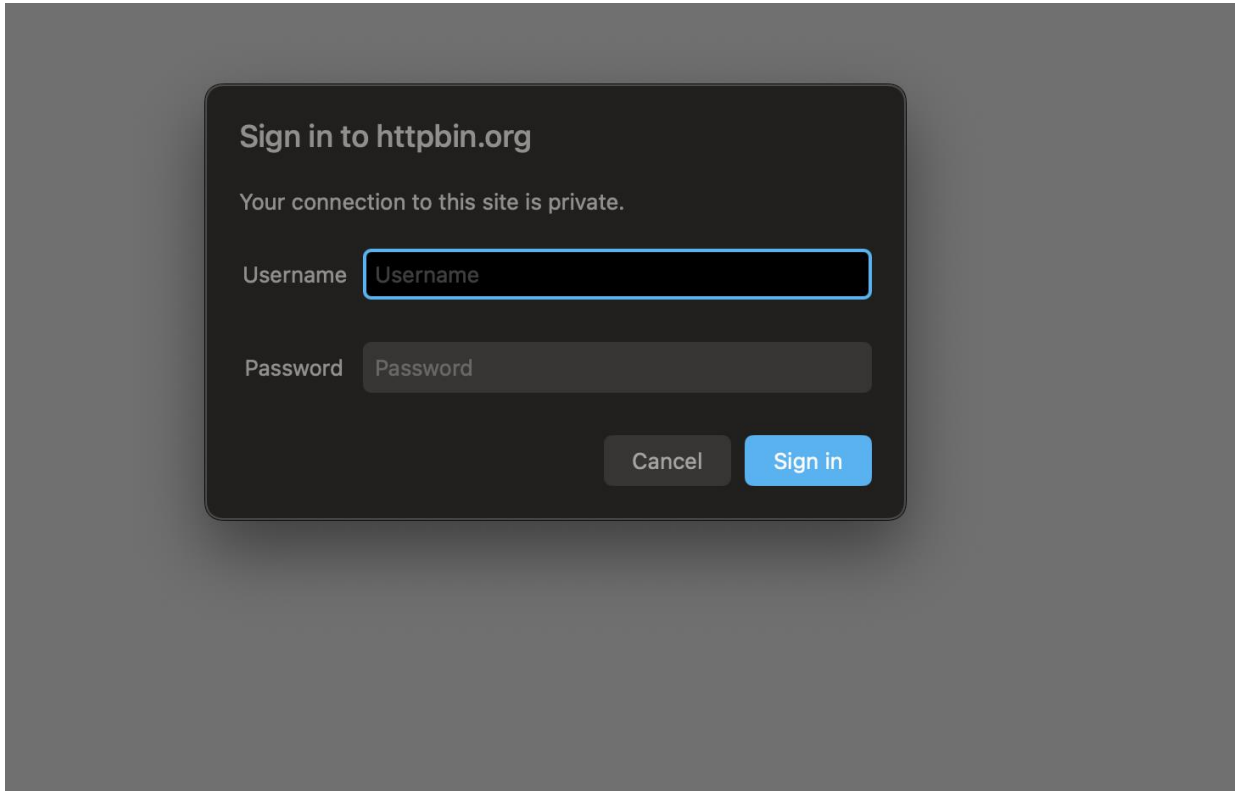
- After connection
- 8. Check if ‘h2’ in the list returned by conn.selected_alpn_protocol()

```
#Make a connection
conn.connect(("www.google.ca", 443))

#Send requests
conn.send(b"GET / HTTP/1.1\r\n\r\n")

#Receive response
print(conn.recv(1024))
```

Protected Website

- You may ask for any password-protected website for testing use.

- 1. Try http://httpbin.org/basic-auth/user/passwd



• 2. Deploy a password-protected website locally using our scripts

Check Brightspace for the file

A simple password-protected website running on your localhost

Note: This server is only for testing your program, you should not use the same http packege for this assignment

Usage: python passwordserver.py
Address: 127.0.0.1:8000
or localhost:8000

```python
from http.server import HTTPServer, SimpleHTTPRequestHandler
import base64

class BasicAuthHandler(SimpleHTTPRequestHandler):
    def do_HEAD(self):
        if not self.headers.get('Authorization'):
            self.send_response(401)
            self.send_header('WWW-Authenticate', 'Basic realm="Protected"')
            self.end_headers()
            return
        auth_type, credentials = self.headers.get('Authorization').split()
        user_pass = base64.b64decode(credentials).decode('utf-8')
        username, password = user_pass.split(':')
        if username == "admin" and password == "password":
            return super().do_HEAD()
        else:
            self.send_response(401)
            self.end_headers()

    def do_GET(self):
        self.do_HEAD()
        return super().do_GET()

server_address = ('', 8000)
httpd = HTTPServer(server_address, BasicAuthHandler)
print("Serving on port 8000 with basic auth (username: admin, password: password)")
httpd.serve_forever()
```