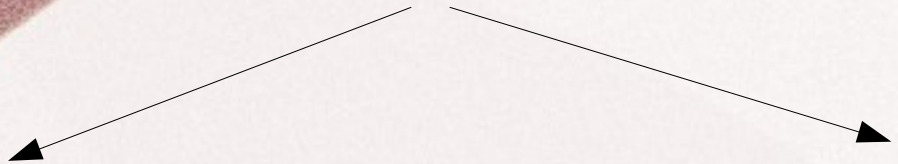
Lecture 23

- Section 16.3 Basic Network Services

16.2 Writing a Basic Client

models for network communication



Client-Server

server: sits passively, waits for a client to initiate contact

clients: connect to the server and send requests

Server software is different from client software

Peer-to-peer (P2P)

two or more connected machines have symmetric capabilities, and run the same software

peer-to-peer software is often implemented as a mixture of client and server software

Client programs we did

- program getting “presize” time gettime.py
- program downloading and save a web-page
  webclient.py easywebclient.py

16.3 Basic Network services

Writing a server is quite different from writing a client.

Server:

- listens on a particular port for incoming connections, then
- creates a socket to manage the connection, then
- follows the chosen *network protocol* for communication with the client

Python provides SocketServer module with some convenient tools

16.3 Basic Network services

TCP stands for *Transmission Control Protocol*

TCPServer class handles:
- all the details of listening on a specified port for incoming connections and
- creating dedicated socket for each connection

We will customize the precise interactions that will be taking place once a client has connected to the server.

Echo server

Let's write an echo server: it will be simply sending the message from the client back.

```
from socketserver import TCPServer,BaseRequestHandler

class EchoHandler(BaseRequestHandler):
   def handle(self):
      message = self.request.recv(1024).decode()
      self.request.send(message.encode())

# may need to customize localhost and
# port for your machine
echoServer = TCPServer( ('localhost', 9000),
EchoHandler())
print("Server is ready")
echoServer.serve_forever()
```

Download the file echoserver.py from our web-page, then run it.

Echo server - client

Run another Python IDE, and type in the following in the shell:

```
>>> from socket import socket
>>> echo = socket()
>>> echo.connect( ('localhost',9000) )
>>> message = “Hello, how are you?”
>>> echo.send(message.encode())


>>> echo.recv(1024)
```

Echo server - client

Run another Python IDE, and type in the following in the shell:

```
>>> from socket import socket
>>> echo = socket()
>>> echo.connect( ('localhost',9000) )
>>> message = “Hello, how are you?”
>>> echo.send(message.encode())


>>> echo.recv(1024)
```

try sending another message to the server and see what will you get as a response

- as soon as the `handle` method ends, the server automatically closes the connection

Echo server - client

We can also use existing network tools to connect to our echo server.

For example, if `telnet` is installed, we can do the following (after running `cmd` from *Windows start*):

```
> telnet localhost 9000
```

and we should see:

```
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
```

if we enter:

```
Hello, how are you?
```

we will get:

```
Hello, how are you?
Connection closed by foreign host.
```

Basic web server

Let's finally write a basic web server, which upon request (GET) from the client will be sending a file's content.

See two programs: webserver.py and webclientFile.py

In-class assignment

- Create a file myData.txt with some phrase and your name

  Example:

  I like to go to movie theater from time to time.

  Natalia

- Modify webserver.py to reflect your computer's IP address (replace “localhost” with your IP address).

  In Windows run tab type in cmd (command line) and press **Enter**.

  Then in the command line type ipconfig and press **Enter**,

  you will find your computer's IP address in the line with **Ipv4 address.**

- Modify webclientFile.py to get such a file from every student in our class. In order to do so you need to exchange your computer's IP addresses.

GOOD LUCK!