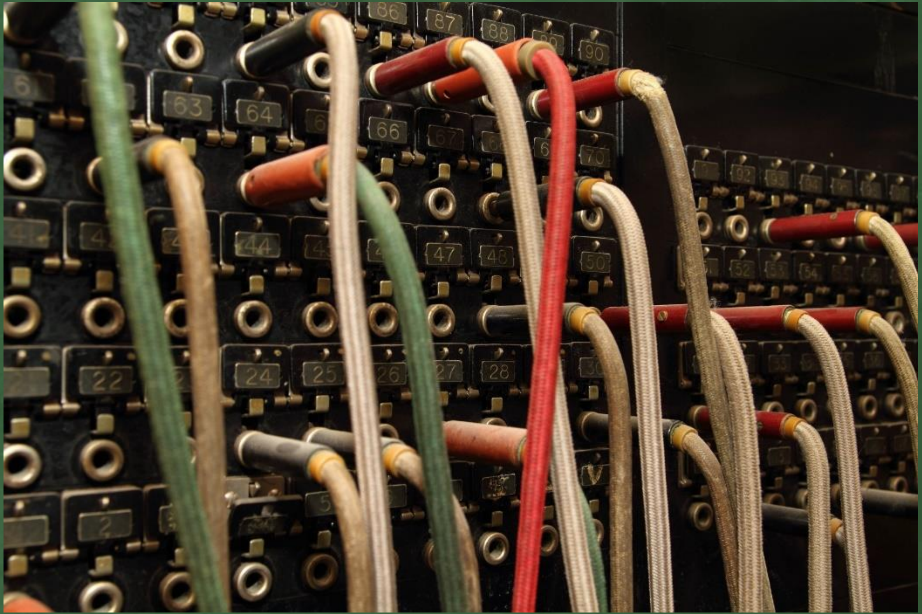
TCP

- TCP Basics
- Server Sockets
- Send and Recv
- Utility Functions



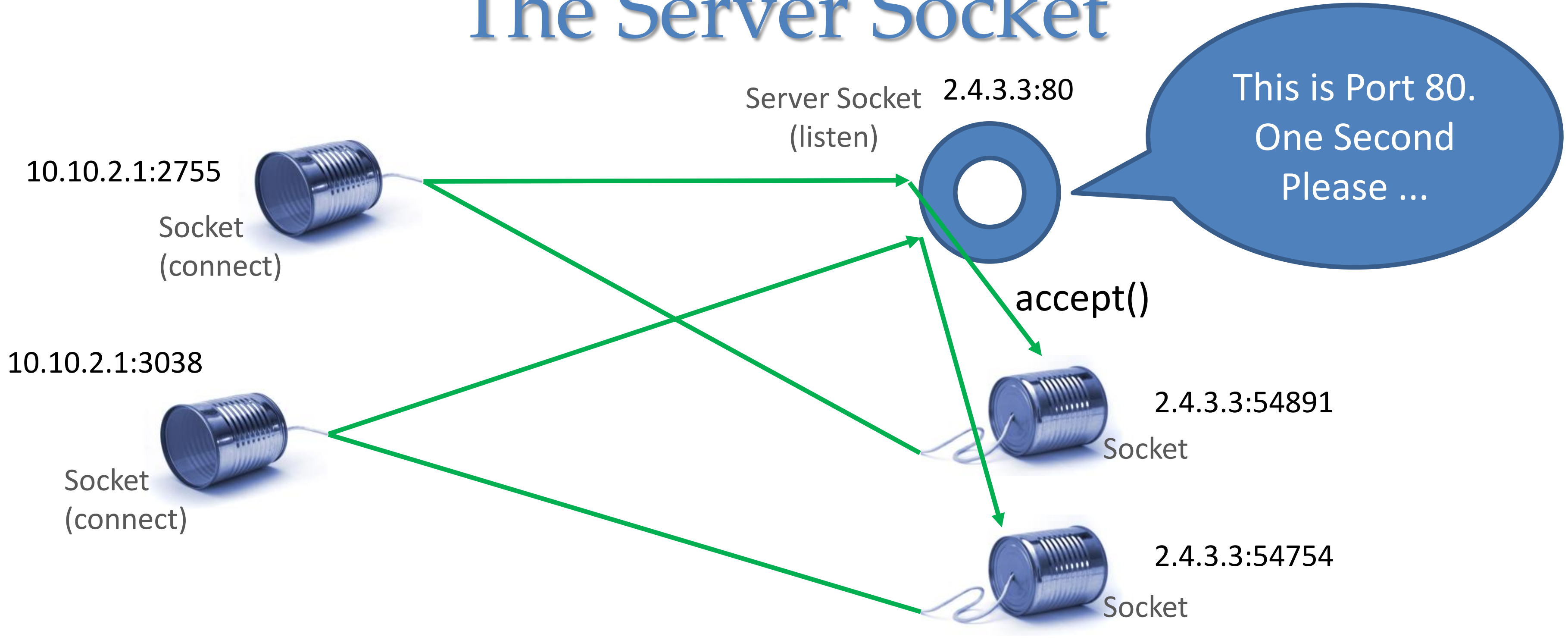
Advanced Python Programming

See Also

https://wiki.python.org/moin/TcpCommunication

https://docs.python.org/2/howto/sockets.html

The Server Socket



- The OS buffers the connections as they come in
- You must call “accept” to take an incoming connection
- RemoteAddress/Port, LocalAddress/Port will be unique

TCP With Python

```python
import socket

s = socket.socket(socket.AF_INET,
                  socket.SOCK_STREAM)

s.bind( ("", 8888) )
s.listen(1)

conn, addr = s.accept()
print('Connection address:', addr)

mes = conn.recv(1024)
print(":"+mes.decode()+":")
```

```python
import socket

s = socket.socket(socket.AF_INET,
                  socket.SOCK_STREAM)

s.connect( ("localhost", 8888))

message = b"HelloWorld"

s.send(message)
```

```
Connection address: ('127.0.0.1', 56367)
:HelloWorld:
```

TCP With Python

```python
import socket

s = socket.socket(socket.AF_INET,
                  socket.SOCK_STREAM)

s.bind( ("", 8888) )
s.listen(1)

conn, addr = s.accept()
print('Connection address:', addr)

mes = conn.recv(1024)
print(":"+mes.decode()+":")
```

```python
import socket
import time

s = socket.socket(socket.AF_INET,
                  socket.SOCK_STREAM)

s.connect( ("localhost", 8888))

s.send(b"Hello")

time.sleep(5) # Wait 5 seconds

s.send(b"World")
```

```
Connection address: ('127.0.0.1', 56367)
:HelloWorld:
:Hello:
```

TCP With Python

```python
import socket

s = socket.socket(socket.AF_INET,
                  socket.SOCK_STREAM)

s.bind( ("", 8888) )
s.listen(1)

conn, addr = s.accept()
print('Connection address:', addr)

mes = conn.recv(1024)
print(":"+mes.decode()+":")
```

```python
import socket
import time

s = socket.socket(socket.AF_INET,
                  socket.SOCK_STREAM)

s.connect( ("localhost", 8888))

time.sleep(5) # Wait 5 seconds

message = b"HelloWorld"
 # s.send(message)
```

```
Connection address: ('127.0.0.1', 56367)
::
```

recv and send

```
mes = conn.recv(1024)
```

- Waits for at least one byte
- Returns no more than N bytes
- Returns an empty string if connection has been closed

```
n = s.send(message)
```

- Sends at least one byte
- Returns number of bytes sent
- Returns 0 if the connection has been closed

Send and Recv Helpers

```
def send_all_bytes(sock,msg):
    totalsent = 0
    while totalsent <len(msg):
        sent = sock.send(msg[totalsent:])
        if sent == 0:
            raise RuntimeError("socket connection broken")
        totalsent = totalsent + sent

def receive_all_bytes(sock,num):
    chunks = []
    bytes_recd = 0
    while bytes_recd < num:
        chunk = sock.recv(num - bytes_recd)
        if not chunk:
            raise RuntimeError("socket connection broken")
        chunks.append(chunk)
        bytes_recd = bytes_recd + len(chunk)
    return b''.join(chunks)
```

You must know how many bytes are coming (good for binary data)

Self Sizing Text Messages

XML

```
<line>
    <point x="5" y="6"/>
    <point x="2" y="0"/>
    <color>Dark Blue</color>
</line>
```

JSON

```
{
  "firstPoint" : {
      "x" : 5,
      "y" : 6
  },
  "secondPoint" : {
      "x" : 2,
      "y" : 0
  },
  "color" : "Dark Blue"
}
```

HTTP (web)

```
POST /cgi-bin/process.cgi HTTP/1.1 ↵
User-Agent: Mozilla/4.0 ↵
Host: www.tutorialspoint.com ↵
Content-Type: text/xml; charset=utf-8 ↵
Content-Length: 60 ↵
Accept-Language: en-us ↵
Accept-Encoding: gzip, deflate ↵
Connection: Keep-Alive ↵
↵
first=Zara&last=Ali ↵
↵
```

String Helpers

```
def send_string(sock,msg):
    slen = str(len(msg.encode()))
    while len(slen)<8:
        slen = "0" + slen
    send_all_bytes(sock,slen.encode())
    send_all_bytes(sock,msg.encode())

def read_string(sock):
    slen = read_all_bytes(sock,8)
    return read_all_bytes(sock,int(slen))
```

Tinkering

- Type in the helper functions from this lesson.
- Make a new server that returns the average of a list of numbers using TCP. You can reuse parts of your code from the UDP lesson.
- Hint: you can use the string helpers we made to send pickled objects.