CS2105 Live Class Lec3: Socket Programming

$_/|/|U_C h@NgRu!$

May 5, 2021

1 Overview of Lec2

1.1 HTTP connection

1.1.1 non-persistent HTTP

at most one object sent over TCP connection
GET /index.html HTTP/1.0\ r\ n

1.1.2 persistent HTTP

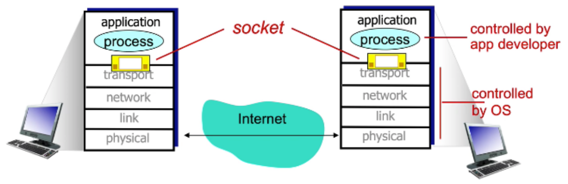
multiple objects can be sent over single TCP connection between client and server
GET /index.html HTTP/1.1\ r\ n

1.2 HTTP is stateless, but cookies is applied to keep states in Application layer

The id and state data

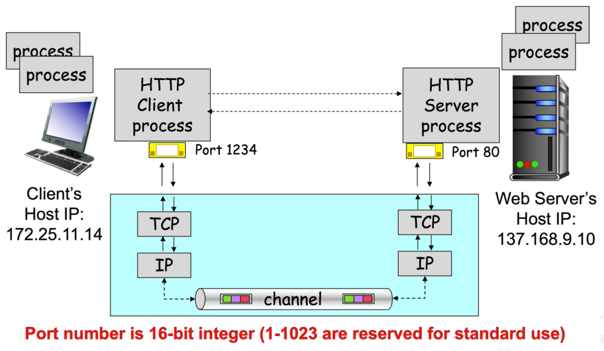
2 Socket Programming

2.1 where is the socket programming



The socket programming is locate at the interface between the application layer and the transportation layer
socket: door between application process and end-end=transportation protocol ; (the programs running in hosts are basically processes)
to identify the process to send: IP(machine)+port(process)
IMPORTANT: even a single process may have multiple socket, so message should be targeted not just process but to socket
With IP+port, one specific socket can be located

3 Port Number


process
process
HTTP Client process
Port 1234
Client's Host IP: 172.25.11.14
TCP
IP
channel
TCP
IP
HTTP Server process
Port 80
process
process
Web Server's Host IP: 137.168.9.10
Port number is 16-bit integer (1-1023 are reserved for standard use)

The HTTP server usually applied 80 for port number
The HTTP client's port number is random, assigned by the OS(but can also be determined by users with bind())

4 Two sockets type

4.1 UDP

unreliable **datagram**, connectionless
IMPORTANT: it's wrong to say "UDP connection"

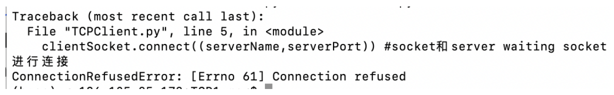
4.2 TCP

reliable **byte stream**, connection-oriented

4.2.1 TCP

server must exist before the client tries to connect it

If the client tries to connect an non-exist server, exception will be thrown



```
Traceback (most recent call last):
  File "TCPClient.py", line 5, in <module>
    clientSocket.connect((serverName,serverPort)) #socket和server waiting socket
进行连接
ConnectionRefusedError: [Errno 61] Connection refused
```

5 Application example in the demo

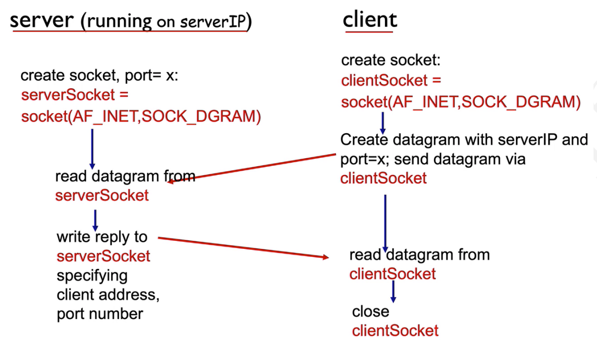
1. client reads a line of characters (data) from its keyboard and sends data to server

2. server receives the data and converts characters to uppercase

3.server sends modified data to client

4. client receives modified data and displays line on its screen

6 UDP



6.1 UDP server

```
from socket import *

serverSocket=socket(AF_INET,SOCK_DGRAM) #create a socket
#AF_INET means Internet
#SOCK_DGRAM means datagram of UDP
serverSocket.bind(('',12000)) #specifying the certain port number
#80 for web(Internet), 53 for DNS server;
#>1024 free styled;
#the IP port number is 16 bits=2 bytes(i.e. 0-65535)
#the '' here is a default parameter, the OS will attach the IP to it
print('Waiting for connections')
```

```
while True:
    message,address=serverSocket.recvfrom(2048)
    #read datagram from serverSocket
    #address is "the explicitly sender's IP+port"
    print(message,address)
    message=message.upper()
    serverSocket.sendto(message,address)
serverSocket.close()
```

6.2 UDP client

```
from socket import socket,SOCK_DGRAM,AF_INET
serverName= '10.0.2.15' #specify the IP address of server
#here host name is also ok(DNS)
serverPort=12000 #specify the server port(to target specified process)
clientSocket =socket(AF_INET, SOCK_DGRAM) #create socket
message=input('Input lowercase sentence: ').encode()
clientSocket.sendto(message, (serverName,serverPort)) #send the message to the server
modifiedMessage, addr=clientSocket.recvfrom(2048) #read data from client Socket
print(modifiedMessage,addr)
clientSocket.close()
```

6.3 characters of UDP

6.3.1 no "connection" between client and server

1. no handshake before sending data
2. sender explicitly attaches IP destination address and port # to **each packet** (in fact, IP address can also be the host name(DNS service))
3. receiver extracts **sender IP** address and port # from received packet

6.3.2 transmitted data may be lost or received out-of-order

The internet uses packet switching with packet switch, each packet switch has a buffer
If the sending side is much faster than the receiving capability, the receiving buffer will be full and the data will be lost
On the other hand, the sending buffer can also be full the data may also be lost because the sending buffer is full

6.3.3 application viewpoint

UDP provides unreliable transfer of groups of bytes(i.e.datagrams) between client and servers

7 TCP

7.1 step flow of TCP

7.1.1 client must contact server

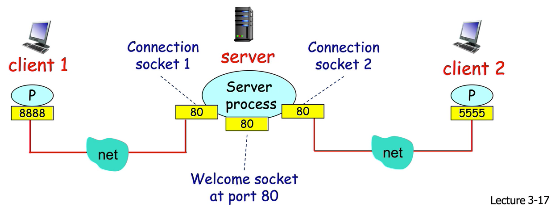
1. server runs first
2. server created socket that welcomes client's contact(the welcome socket)

7.1.2 client contacts server by

1. creating TCP socket, specifying IP address, port number of server process
2. when client creates socket: TCP client establishes connection to TCP server
3. TCP server creates **new socket with the same IP and port number** for server process to communicate with that particular client allows servers to talk with multiple clients source port numbers used to distinguish clients

7.2 view point

TCP provides reliable, in-order byte-stream transfer("pipe") between client and server



IMPORTANT
a. The server socket has only one port at the begining(the welcome socket)
IMPORTANT: there is no data through the welcome socket to the client socket; is only used to exchange control for messages and build connection
b. after the server receive the request throught welcome port, it will create another port to communicate with the client(IMPORTANT, the new port uses the same port number, but uses

the client's IP+host information as differentiator to differentiate the connection
c. the data will be exchanged through the connection

7.3 TCP server

```
from socket import *
serverPort=12000
#the IP port number is 16 bits=2 bytes(i.e. 0-65535)
#80 for web(Internet);
#>1024 free styled;
serverSocket=socket(AF_INET,SOCK_STREAM)
serverSocket.bind(('',serverPort))
serverSocket.listen(1)
print('The server is ready to receive')
while 1:
   connectionSocket, addr=serverSocket.accept()
   print(addr)
   sentence=connectionSocket.recv(1024) #note that the code is simplified(one server can
       only serve one client) and this socket is not welcome socket(it really response the
       client)
   print(sentence.decode())
   capitalizedSentence=sentence.upper()
   connectionSocket.send(capitalizedSentence)
   connectionSocket.close()
```

7.4 TCP client

```
from socket import *
serverName='127.0.0.1' #server ip address
#127.0.0.1 is localhost
serverPort=12000 #port
clientSocket=socket(AF_INET, SOCK_STREAM) #build socket
# af means address family;
#INET means internet;
#SOCK_STREAM means data stream way(for TCP)
clientSocket.connect((serverName,serverPort)) #socket and server waiting socket to connect
while 1:
   sentence =input("Input lowercase sentence:")
   if sentence=='closeSocket':
       clientSocket.close() #close socket
       break
   clientSocket.send(sentence.encode()) #send data to server through socket
```

```
#modifiedSentence=clientSocket.recv(1024) #store the data received in sentence
#print('From Server:',modifiedSentence.decode())
```