**UMKC**

**Dwarkamoye Mohanty**

**16281824**

**Network Architecture 1**

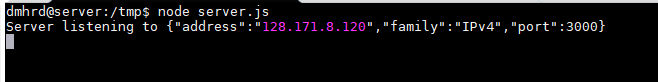
**2019 Spring CSEE 5110 Individual Project1**

**Part 2**

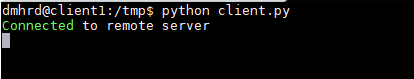
**Part 2:**

(a)

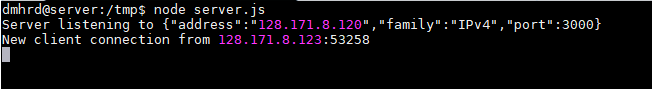
Starting the server:



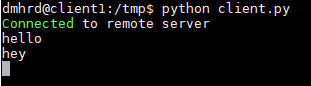
Starting the client:



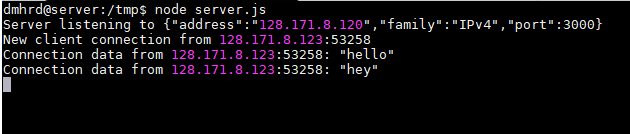
Server shows the client connected:



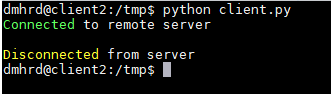
Client sends message to server:



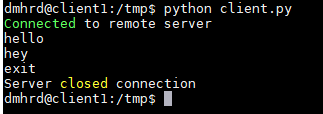
Server **displays the messages** received:

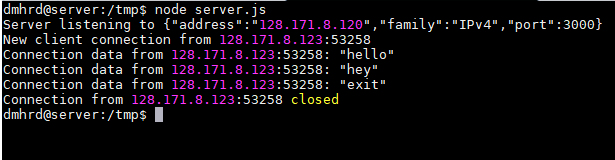


Second client tries to connect to server and gets disconnected as server allows only **one connection**:



Client sends **“exit” message, which results in end** of both client and server program:





Code for Server:

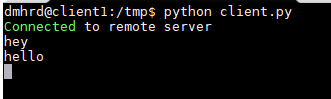
var net = require('net');  
var server = net.createServer();  
server.on('connection', handleConnection);  
server.listen(3000,'128.171.8.120', function() {  
 console.log('Server listening to %j', server.address());  
});  
server.maxConnections = 1;  
function handleConnection(conn) {   
 var remoteAddr = conn.remoteAddress + ':' + conn.remotePort;  
 console.log('New client connection from %s', remoteAddr);  
 conn.setEncoding('utf8');  
 conn.on('data', onConnData);  
 conn.once('close', onConnClose);  
 conn.on('error', onConnError);  
 function onConnData(data) {  
 console.log('Connection data from %s: %j', remoteAddr, data);  
 if (data == "exit")  
 {  
 conn.write(data);  
 server.close();  
 }  
 }  
 function onConnClose() {  
 console.log('Connection from %s closed', remoteAddr);  
 }  
 function onConnError(error) {  
 console.log('Error %s connection: %s', remoteAddr, error.message);  
 }  
}

Code for Client:

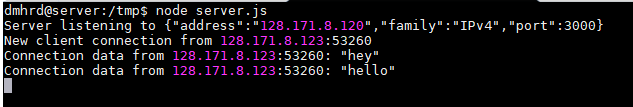
import sys  
import select  
import socket  
  
hostname = '128.171.8.120'  
buffer\_size = 1024  
portnumber = 3000  
  
  
def client():  
 soc = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
  
 try:  
 soc.connect((hostname, portnumber))  
 except:  
 print 'Connection not available'  
 sys.exit()  
  
 print 'Connected to remote server'  
  
 while True:  
 socketList = [sys.stdin, soc]  
 read, write, error = select.select(socketList, [], [])  
  
 for sock in read:  
 if sock == soc:   
 dataRecv = sock.recv(buffer\_size)  
 if not dataRecv:  
 print '\nDisconnected from server'  
 sys.exit()  
 elif dataRecv == "exit":  
 print ("Server closed connection")  
 sys.exit()  
 else:  
 print(dataRecv)  
 else:  
 # user typed a message  
 msg = raw\_input()  
 soc.send(msg.encode("utf8"))  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 sys.exit(client())

(b)

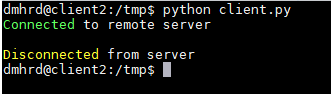
Client starts communicating with server:



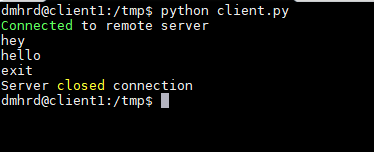
Server displays all:



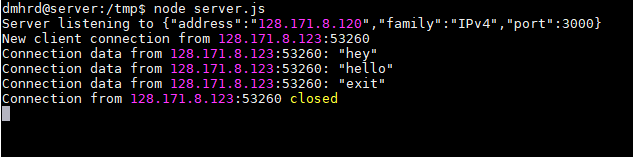
Second client tries to connect to server and gets disconnected as server allows only **one connection**:



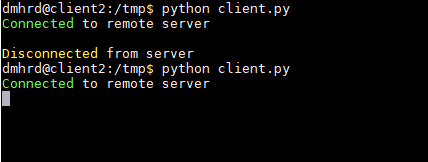
On typing “exit” client gets closed:



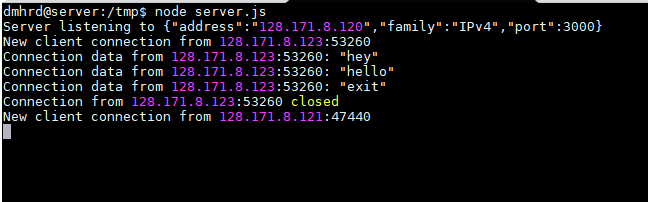
But server is still open for connections:



Now when second client tries to connect it gets connected:



Server shows the new client connection:



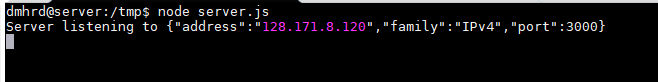
Code for Server:

var net = require('net');  
var server = net.createServer();  
server.on('connection', handleConnection);  
server.listen(3000,'128.171.8.120', function() {  
 console.log('Server listening to %j', server.address());  
});  
server.maxConnections = 1;  
function handleConnection(conn) {   
 var remoteAddr = conn.remoteAddress + ':' + conn.remotePort;  
 console.log('New client connection from %s', remoteAddr);  
 conn.setEncoding('utf8');  
 conn.on('data', onConnData);  
 conn.once('close', onConnClose);  
 conn.on('error', onConnError);  
 function onConnData(data) {  
 console.log('Connection data from %s: %j', remoteAddr, data);  
 if (data == "exit")  
 {  
 conn.write(data);  
 }  
 }  
 function onConnClose() {  
 console.log('Connection from %s closed', remoteAddr);  
 }  
 function onConnError(error) {  
 console.log('Error %s connection: %s', remoteAddr, error.message);  
 }  
}

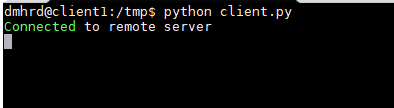
**No change in client code. (Same as (a))**

(c)

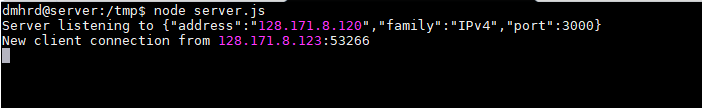
Started the server:



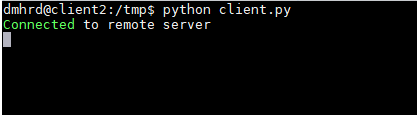
Starting the first client:



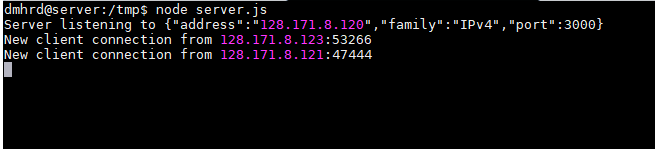
Server listens to first client:



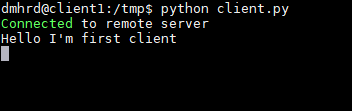
Starting second client:



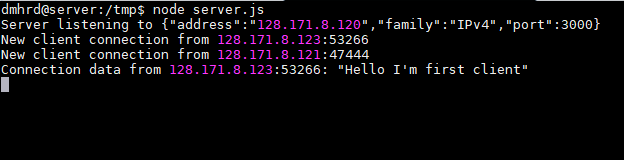
Server listens to second connection:



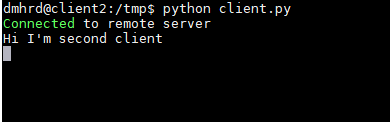
First client sending message:



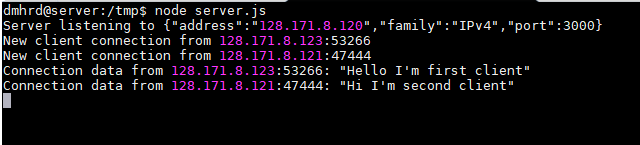
Server displaying the message from first client:



Second client sending the message:

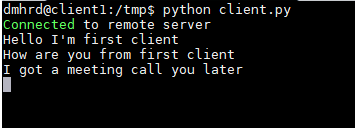


Server displaying message from second client:

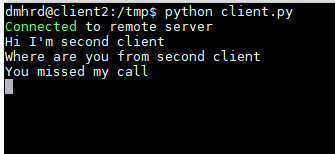


Some more messages from both the clients in random order:

Client 1-🡪

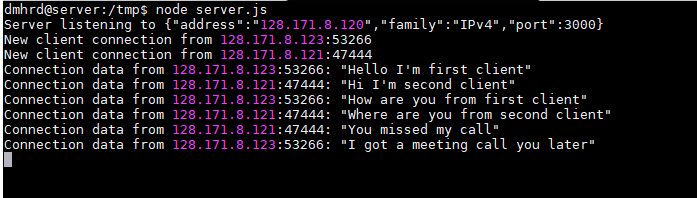


Client 2-🡪



Server displays everything it receives from all the clients in the same order:

(Messages can be distinguished from the IP address of both the clients)



Code for Server:

var net = require('net');  
var sockets = [];  
var server = net.createServer(function(socket){  
sockets.push(socket);  
});

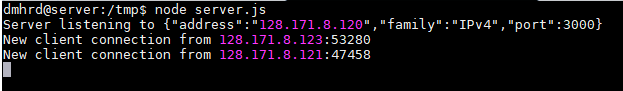
server.on('connection', handleConnection);  
server.listen(3000,'128.171.8.120', function() {  
 console.log('Server listening to %j', server.address());  
});  
function handleConnection(conn) {   
 var remoteAddr = conn.remoteAddress + ':' + conn.remotePort;  
 console.log('New client connection from %s', remoteAddress);  
 conn.setEncoding('utf8');  
 conn.on('data', onConnData);  
 conn.once('close', onConnClose);  
 conn.on('error', onConnError);  
 function onConnData(data) {  
 console.log('Connection data from %s: %j', remoteAddr, data);  
 if (data == "exit")  
 {  
 conn.write(data);  
 }  
 }  
 function onConnClose() {  
 console.log('Connection from %s closed', remoteAddr);  
 }  
 function onConnError(error) {  
 console.log('Error %s connection: %s', remoteAddr, error.message);  
 }  
}

Code for Client:

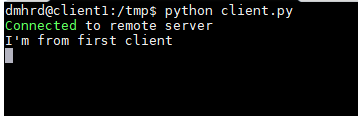
import sys  
import select  
import socket  
  
hostname = '128.171.8.120'  
buffer\_size = 1024  
portnumber = 3000  
  
  
def client():  
 soc = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
  
 try:  
 soc.connect((hostname, portnumber))  
 except:  
 print 'Connection not available'  
 sys.exit()  
  
 print 'Connected to remote server'  
  
 while True:  
 socketList = [sys.stdin, soc]  
 read, write, error = select.select(socketList, [], [])  
  
 for sock in read:  
 if sock == soc:   
 dataRecv = sock.recv(buffer\_size)  
 if not dataRecv:  
 print '\nDisconnected from server'  
 sys.exit()  
 elif dataRecv == "exit":  
 print ("Server closed connection")  
 sys.exit()  
 else:  
 print(dataRecv)  
 else:  
 # user typed a message  
 msg = raw\_input()  
 soc.send(msg.encode("utf8"))  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 sys.exit(client())

(d)

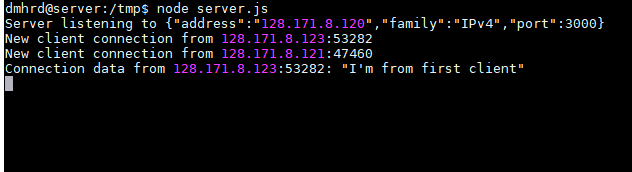
Server listening to both the client:



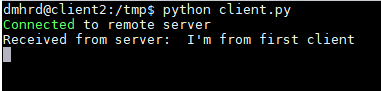
First client sends message:



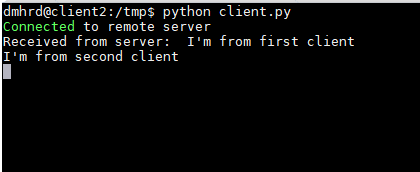
Server displays the message:



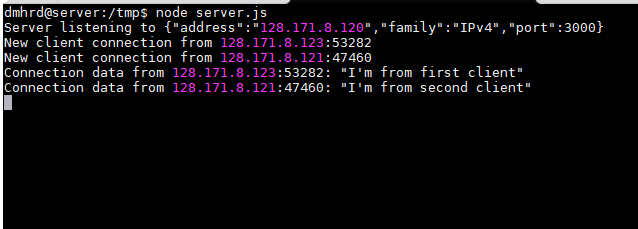
Server echoes the received message to another client:



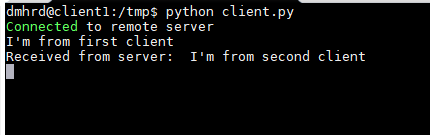
Now second client sends a message:



Server displays the message received:



Server echoes to the other client:



Code for Client:

import sys  
import select  
import socket  
  
hostname = '128.171.8.120'  
buffer\_size = 1024  
portnumber = 3000  
  
  
def client():  
 soc = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
  
 try:  
 soc.connect((hostname, portnumber))  
 except:  
 print 'Connection not available'  
 sys.exit()  
  
 print 'Connected to remote server'  
  
 while True:  
 socketList = [sys.stdin, soc]  
 read, write, error = select.select(socketList, [], [])  
  
 for sock in read:  
 if sock == soc:   
 dataRecv = sock.recv(buffer\_size)  
 if not dataRecv:  
 print '\nDisconnected from server'  
 sys.exit()  
 elif dataRecv == "exit":  
 print ("Server closed connection")  
 sys.exit()  
 else:  
 print("Received from server: " + dataRecv)  
 else:  
 # user typed a message  
 msg = raw\_input()  
 soc.send(msg.encode("utf8"))  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 sys.exit(client())

Code for Server:

var net = require('net');  
var sockets = [];  
var server = net.createServer(function(socket){  
sockets.push(socket);  
});  
server.on('connection', handleConnection);  
server.listen(3000,'128.171.8.120', function() {  
 console.log('Server listening to %j', server.address());  
});  
function handleConnection(conn) {   
 var remoteAddr = conn.remoteAddress + ':' + conn.remotePort;  
 console.log('New client connection from %s', remoteAddr);  
 conn.setEncoding('utf8');  
 conn.on('data', onConnData);  
 conn.once('close', onConnClose);  
 conn.on('error', onConnError);  
 function onConnData(data) {  
 console.log('Connection data from %s: %j', remoteAddr, data);  
 for (var i = 0; i < sockets.length; i++){  
 if (sockets[i] === conn) continue;  
 sockets[i].write(data.toString());  
 }  
 }  
 function onConnClose() {  
 console.log('Connection from %s closed', remoteAddr);  
 }  
 function onConnError(error) {  
 console.log('Error %s connection: %s', remoteAddr, error.message);  
 }  
}

(e)

Both server and client are implemented using two different languages.

Server is implemented using NodeJS. Client is implemented using Python.

GitHub Link:

<https://github.com/Dwarkamoye/Network-Architecture-1-5110-0001/tree/master/Phase-2>