**分散式系統  
Lab: Networking**

請務必填寫學號系級姓名，以免成績登錄錯誤。

學號系級姓名:

**112753207 資碩計一 張詠軒**

請依問題與提示在指定區域回答問題，並依規定時間內上傳至moodle。

第一部份 UDP

1. 請依據課程投影片中的範例，寫作符合下列要求的「UDP Client與Server」，其中，Server 請以Node.js寫作，Client請以Python寫作。
2. Server的需求:
   * 以Node.js寫作
   * 傾聽本地端(127.0.0.1)的port 20213，接收Client傳來的訊息
   * 在console中印出所接收到的訊息
   * 將收到的訊息的最前面加上「XXXX:」再送回給Client，其中，XXXX是Client的port number。(client的port number請由node.js的API取得)

<https://nodejs.org/docs/latest/api/dgram.html#dgram_event_message>

提示: 使用server.send方法要嵌入在on message的call back function中，才會在收到訊息之後執行。

* + 印出後立即關閉連線(提示: server.close方法要嵌入在server.send方法中)

1. Client的需求:
   * 以Python寫作
   * 連接到本地端的UDP Server，port為20213
   * 送出訊息: 「This is a test from python client」，記得要先將訊息轉為binary: b”This is…..”
   * 在console中印出所接收到的回應訊息 (記得用message.decode(‘utf-8’)將binary轉回文字，正常的話應該會印出xxxxx:This is a test from python client，而不是b’xxxxx:This is a test form python client)
   * 收到後關閉連線

請將Server的程式碼則在下面:

答:

//使用 'dgram'模組建立一個UDP Socket

//使用 createSocket() 方法，並傳入'udp4'參數來建立一個IPv4的Socket

const dgram = require('node:dgram');

const server = dgram.createSocket('udp4');

//處理錯誤

//印出錯誤訊息，然後關閉伺服器

server.on('error', (err) => {

    console.error(`server error:\n${err.stack}`);

    server.close();

});

//處理與回應訊息

//設定message事件的處理程序

//當收到客戶端發送的訊息時，會印出該訊息、客戶端的IP位址以及port number

server.on('message', (msg, rinfo) => {

    console.log(`server got: ${msg} from ${rinfo.address}:${rinfo.port}`);

    //const responseMessage = JSON.parse(`${rinfo.port}:${msg.toString}`);

    const responseMessage = Buffer.from(`${rinfo.port}:${msg}`);

    server.send(responseMessage, rinfo.port, rinfo.address, (err) => {

        server.close();

    });

});

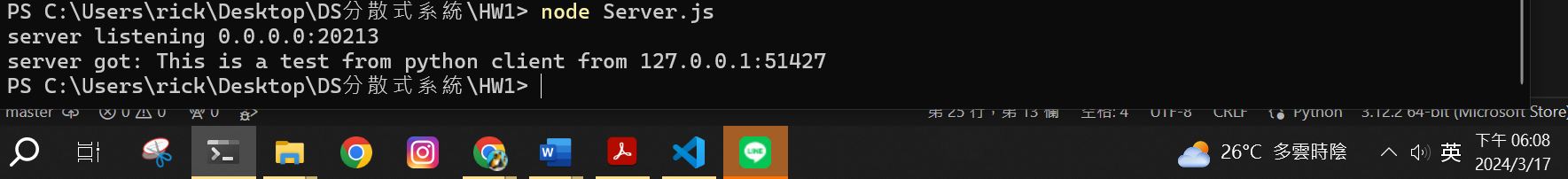
server.on('listening', () => {

  const address = server.address();

  console.log(`server listening ${address.address}:${address.port}`);

});

server.bind(("127.0.0.1", 20213));



請將Client的程式碼貼在下面:

答:

import socket

server\_address = '127.0.0.1'

server\_port = 20213

message = b"This is a test from python client"

sock = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

#UDP是無連結協定，因此connect()方法不是必須的，但可將socket「連接」到特定的遠端位址，即可使用send()和recv()而不是sendto()和recvfrom()，使程式碼簡潔。

#並不意味著建立了一個真正的連接，因為UDP本質上仍然是無連接的。

#對於TCP是必要的，TCP是面向連接通訊，要確保客戶端與伺服器的穩定連結，通過連結發送和接收數據。

sock.connect(("127.0.0.1", 20213))

#傳送資料

print(f'Client sending: {message.decode("utf-8")}')

sock.send(message)

#接收資料

#解碼與列印:接收到的資料被解碼為UTF-8字串並列印出來

#recv(bufsize)，bufsize:一次接收最大字數值，可自己設定

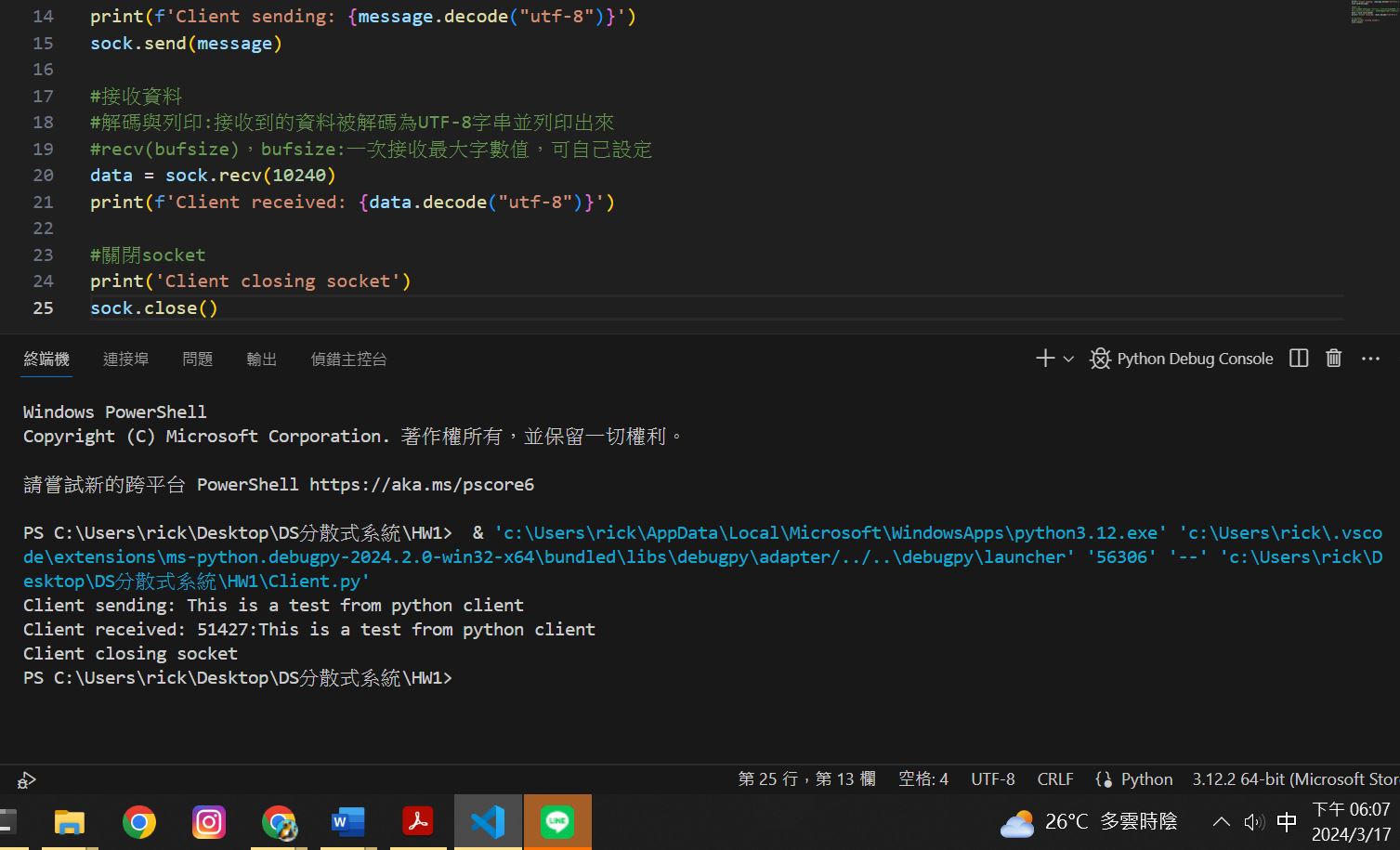
data = sock.recv(10240)

print(f'Client received: {data.decode("utf-8")}')

#關閉socket

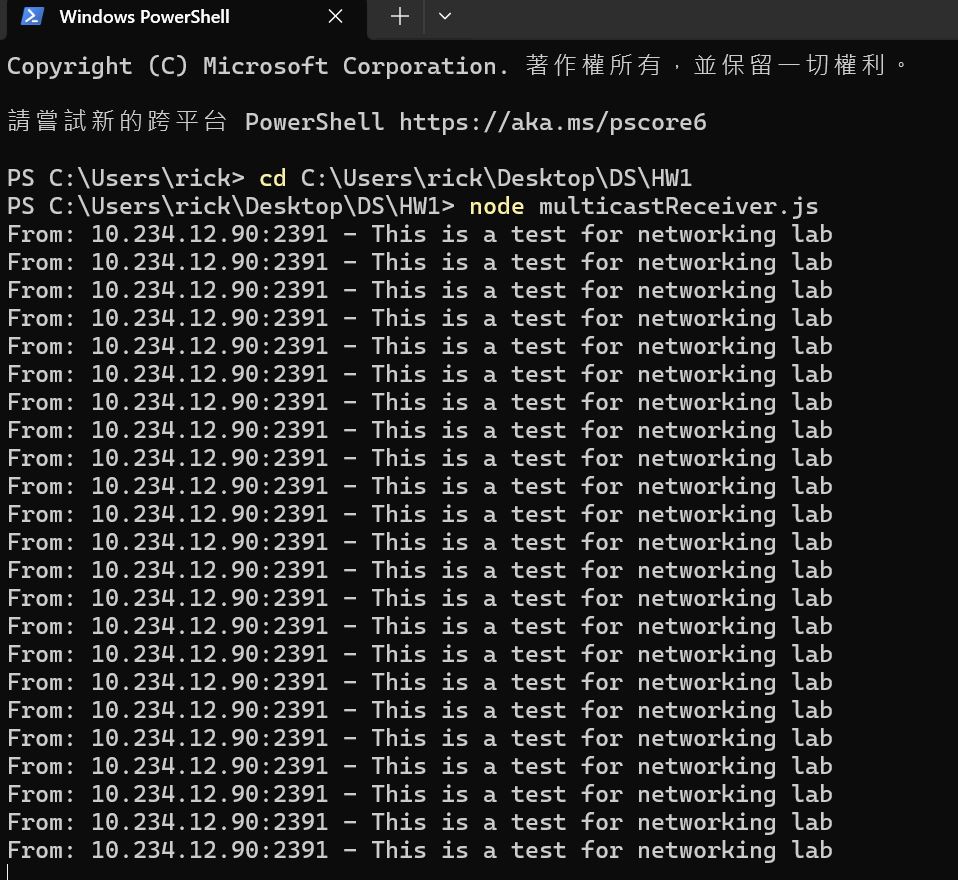
print('Client closing socket')

sock.close()



第二部份 群播

1. 執行multicastReceiver.js與multicaseSender.js，此時，multicastSender會每5秒送一個封包到群播位址，並被multicastReceiver所接收

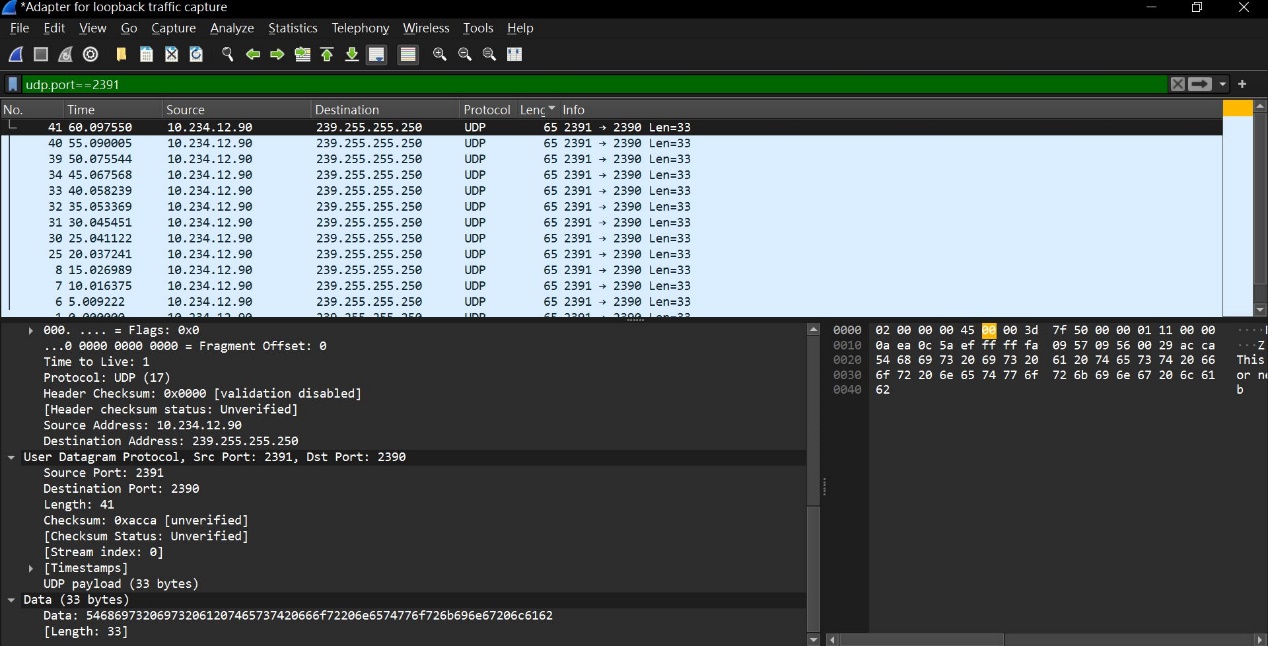


1. 請問它送到那一個群播位址(不含port)? 所傳送的內容為何?

答: **239.255.255.250, This is a test for networking lab**

1. 使用wireshark，設定filter為udp.port==2391，抓取multicastSender送出的封包並觀察它UDP的Length欄位值為多少? UDP Payload的大小(值)為多少?

答: **41 bytes, 33 bytes**



1. UDP Header中的Length欄位代表什麼意思? 為什麼它會比UDP Payload的大小多8 bytes?

答: **Length欄位代表UDP封包的長度，包括UDP header和UDP Payload的長度。因為在計算UDP封包的長度時，需要把UDP header的長度也算進去，而UDP header的長度固定為8 bytes，其中包含Source port、Destination port、Length、Checksum欄位，各佔2 bytes。**