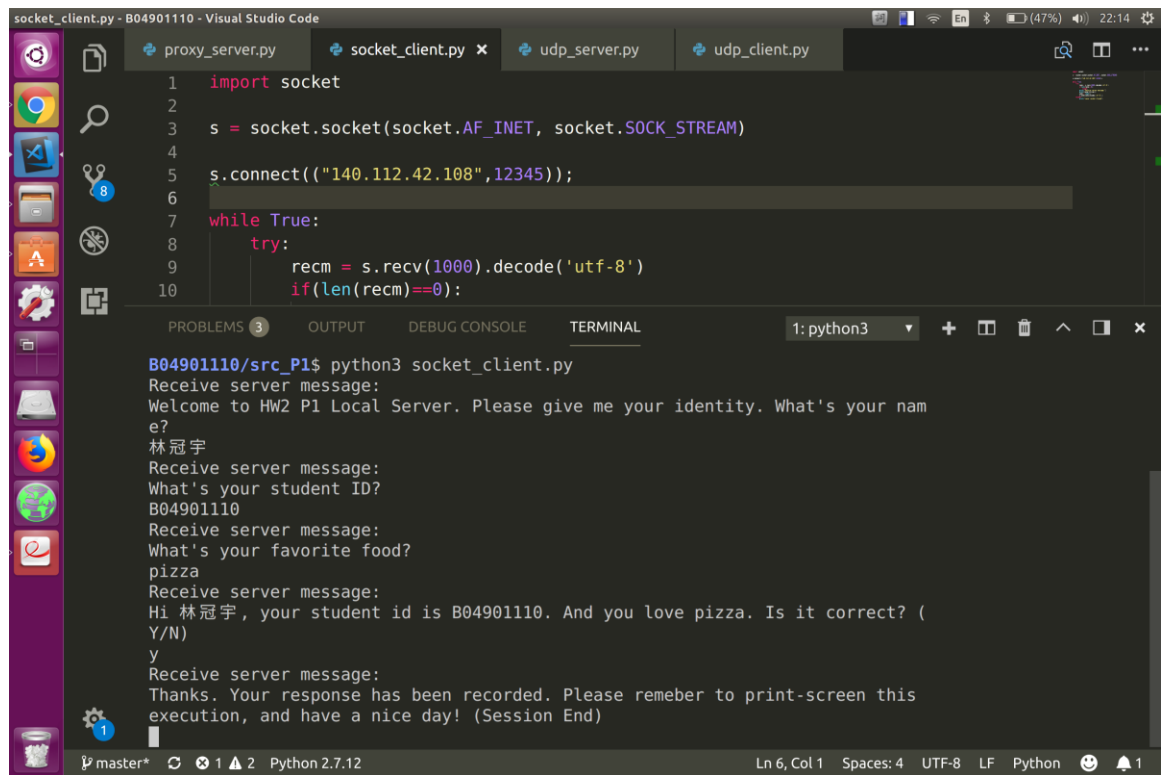


電腦網路導論 HW2

-B04901110 林冠宇

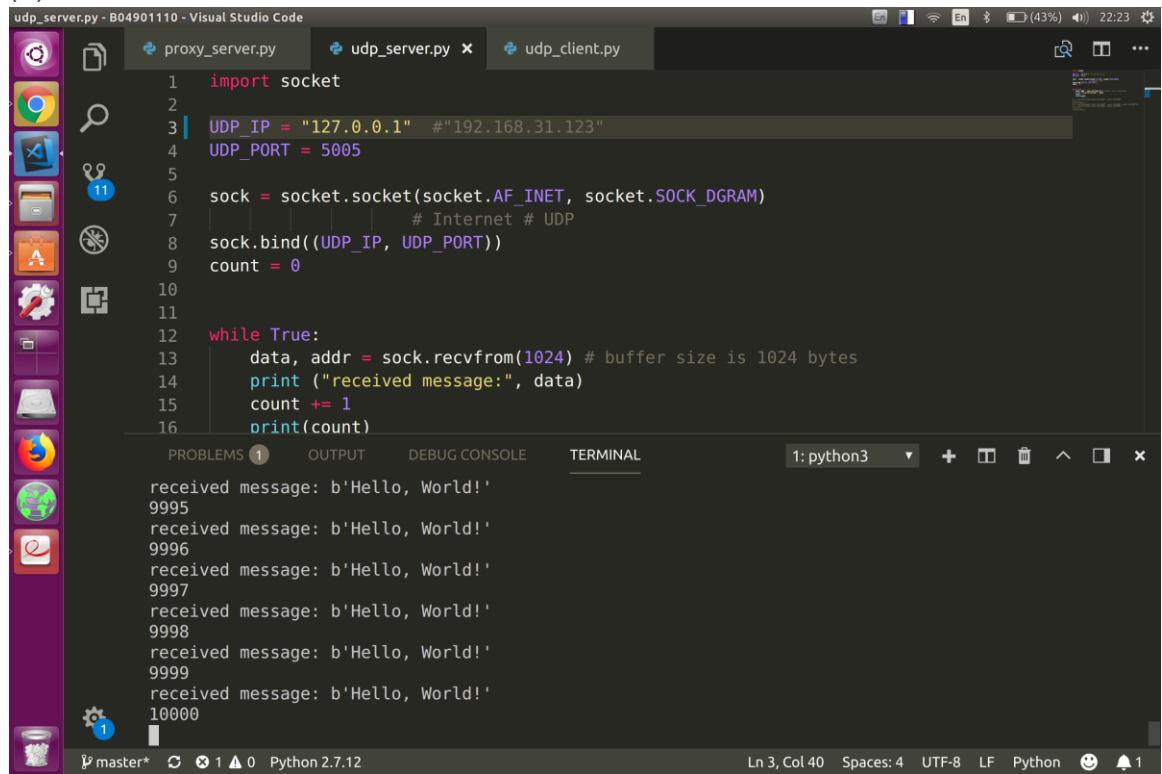
1. [Socket Programming - TCP]



The screenshot shows the Visual Studio Code editor with a file named `socket_client.py` open. The code is a Python script that connects to a server at `140.112.42.108` on port `12345` and receives messages. The terminal output shows the program's execution, including the server's prompts and the user's responses.

```
socket_client.py - B04901110 - Visual Studio Code
proxy_server.py socket_client.py x udp_server.py udp_client.py
1 import socket
2
3 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
4
5 s.connect(("140.112.42.108", 12345));
6
7 while True:
8     try:
9         recm = s.recv(1000).decode('utf-8')
10        if(len(recm)==0):
11
PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL 1: python3
B04901110/src_P1$ python3 socket_client.py
Receive server message:
Welcome to HW2 P1 Local Server. Please give me your identity. What's your nam
e?
林冠宇
Receive server message:
What's your student ID?
B04901110
Receive server message:
What's your favorite food?
pizza
Receive server message:
Hi 林冠宇, your student id is B04901110. And you love pizza. Is it correct? (
Y/N)
y
Receive server message:
Thanks. Your response has been recorded. Please remeber to print-screen this
execution, and have a nice day! (Session End)
```

2. (a) Run on same machine:



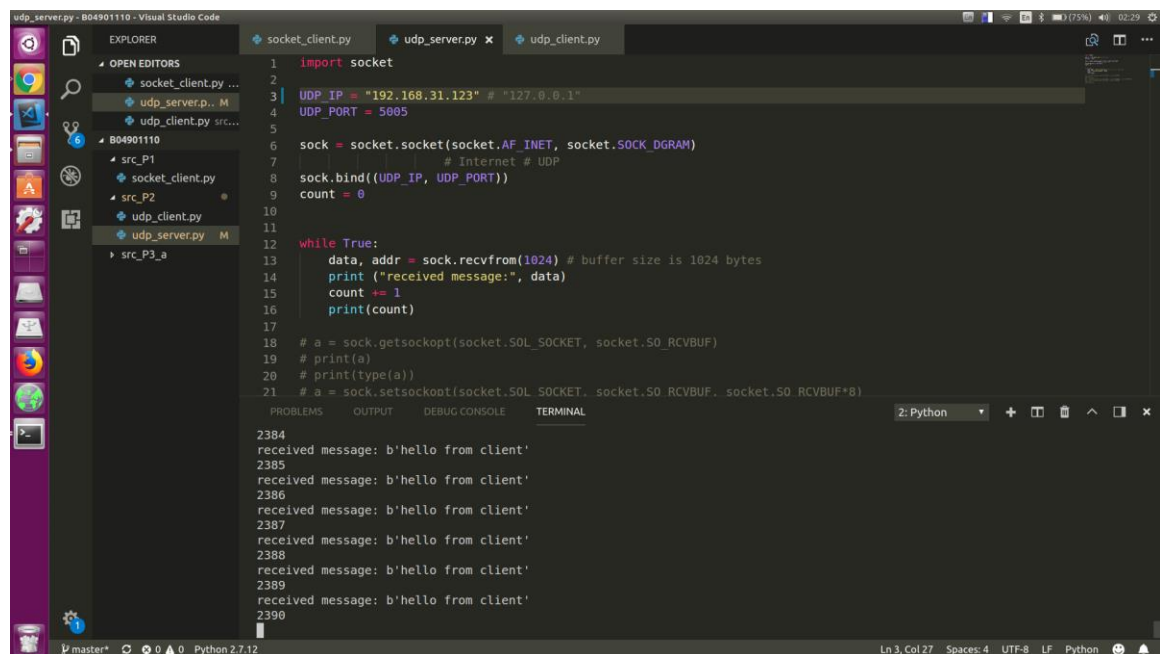
```
udp_server.py - B04901110 - Visual Studio Code
1 import socket
2
3 UDP_IP = "127.0.0.1" #"192.168.31.123"
4 UDP_PORT = 5005
5
6 sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
7 # Internet # UDP
8 sock.bind((UDP_IP, UDP_PORT))
9 count = 0
10
11
12 while True:
13     data, addr = sock.recvfrom(1024) # buffer size is 1024 bytes
14     print ("received message:", data)
15     count += 1
16     print(count)
```

received message: b'Hello, World!'
9995
received message: b'Hello, World!'
9996
received message: b'Hello, World!'
9997
received message: b'Hello, World!'
9998
received message: b'Hello, World!'
9999
received message: b'Hello, World!'
10000

how many packets will the server receive? 10000

And how many of them are lost? 0

(b) Run on different machine -1:

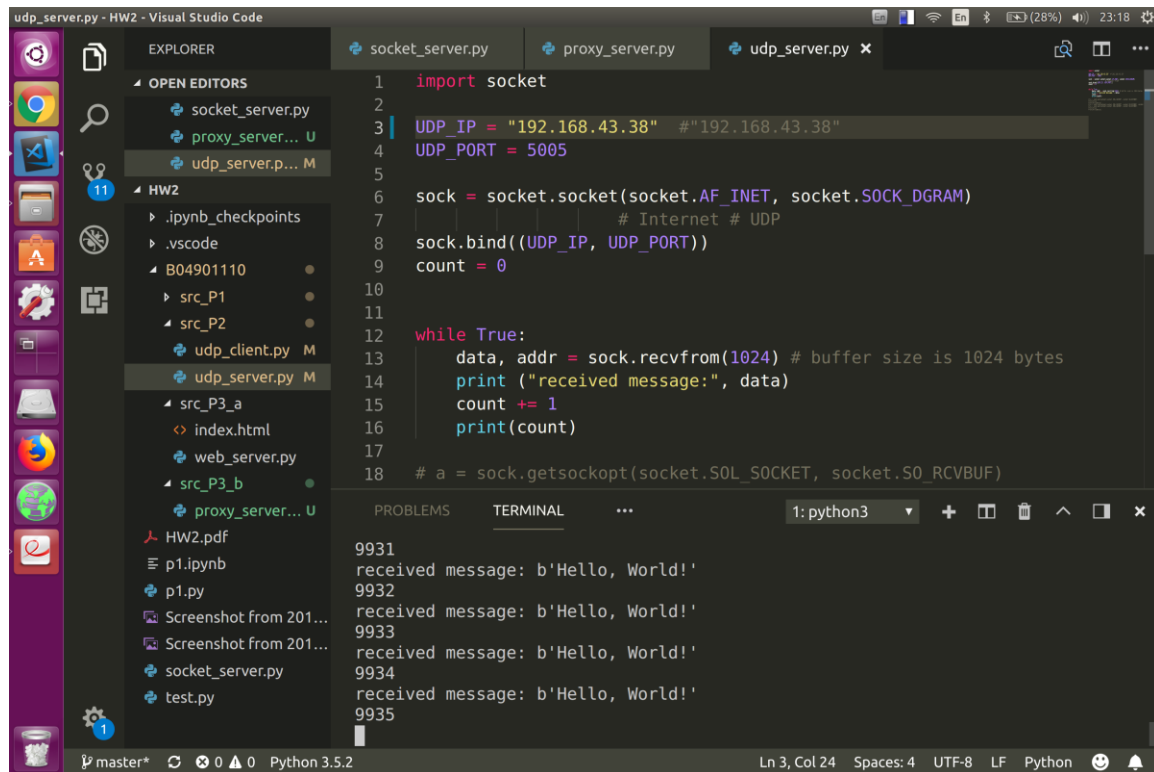


```
udp_server.py - B04901110 - Visual Studio Code
1 import socket
2
3 UDP_IP = "192.168.31.123" #"127.0.0.1"
4 UDP_PORT = 5005
5
6 sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
7 # Internet # UDP
8 sock.bind((UDP_IP, UDP_PORT))
9 count = 0
10
11
12 while True:
13     data, addr = sock.recvfrom(1024) # buffer size is 1024 bytes
14     print ("received message:", data)
15     count += 1
16     print(count)
```

2384
received message: b'hello from client'
2385
received message: b'hello from client'
2386
received message: b'hello from client'
2387
received message: b'hello from client'
2388
received message: b'hello from client'
2389
received message: b'hello from client'
2390

The loss rate is about 76% which is quite unacceptable, therefore I add `(time.sleep(0.0001))` after each packet sent, and below is the improved result.

(b)Run on different machine -2 (improved):



The screenshot shows the Visual Studio Code interface with a file explorer on the left, a code editor in the center, and a terminal at the bottom. The code editor displays the `udp_server.py` file, which is a Python script for a UDP server. The terminal shows the output of the script, which is receiving messages from a client.

```
udp_server.py - HW2 - Visual Studio Code
EXPLORER
  OPEN EDITORS
    socket_server.py
    proxy_server... U
    udp_server.p... M
  HW2
    .ipynb_checkpoints
    .vscode
    B04901110
      src_P1
      src_P2
      udp_client.py M
      udp_server.py M
      src_P3_a
      index.html
      web_server.py
      src_P3_b
      proxy_server... U
  HW2.pdf
  p1.ipynb
  p1.py
  Screenshot from 201...
  Screenshot from 201...
  socket_server.py
  test.py

1  import socket
2
3  UDP_IP = "192.168.43.38" #"192.168.43.38"
4  UDP_PORT = 5005
5
6  sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
7  sock.bind((UDP_IP, UDP_PORT))
8  count = 0
9
10
11
12  while True:
13      data, addr = sock.recvfrom(1024) # buffer size is 1024 bytes
14      print ("received message:", data)
15      count += 1
16      print(count)
17
18  # a = sock.getsockopt(socket.SOL_SOCKET, socket.SO_RCVBUF)
```

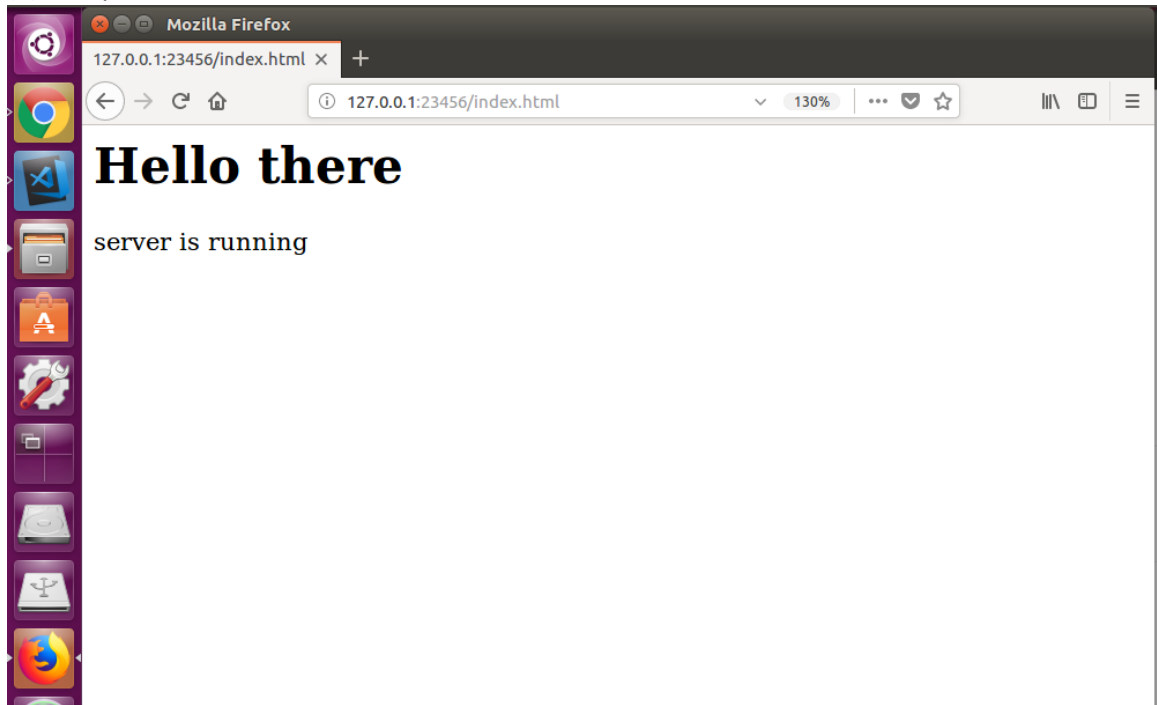
```
9931
received message: b'Hello, World!'
9932
received message: b'Hello, World!'
9933
received message: b'Hello, World!'
9934
received message: b'Hello, World!'
9935
```

1: python3

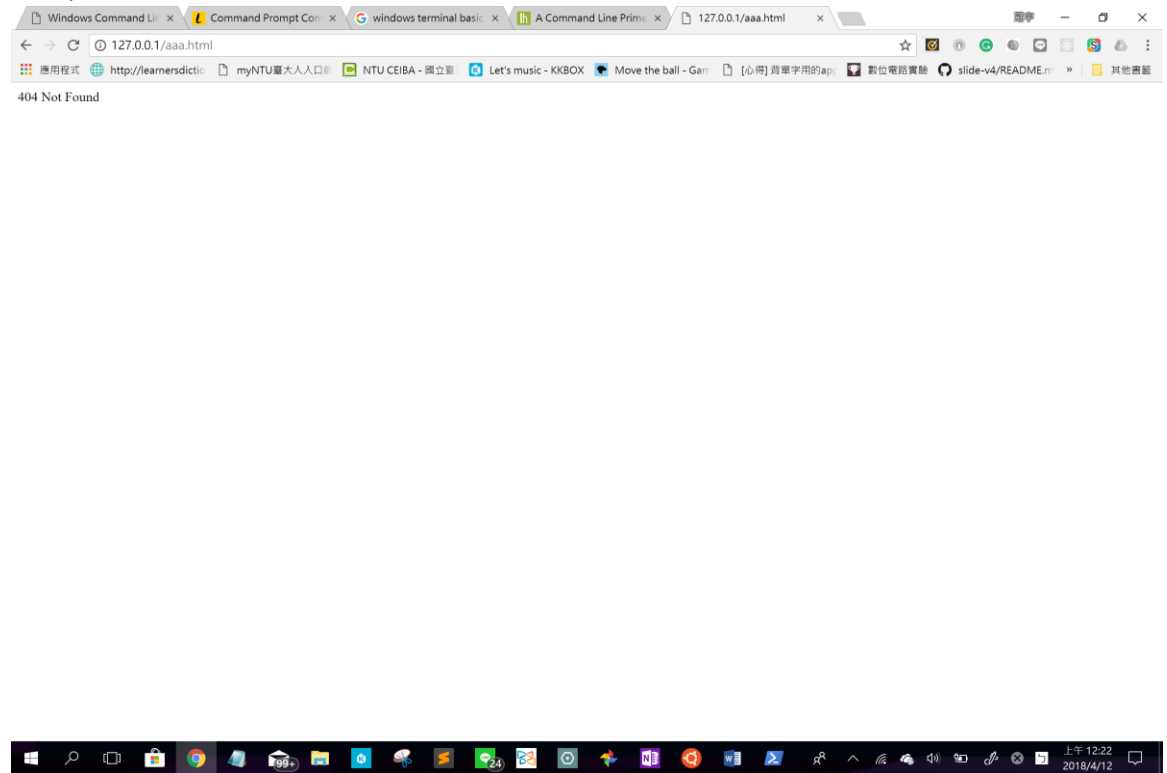
Ln 3, Col 24 Spaces: 4 UTF-8 LF Python

Which is much better than the previous experiment.

3. (a) if requested file exist



If request file is not in the server



(b) proxy succeeded in requesting file from the web server implemented in 3(a) after the first request from the client

And then send to client after the second request from the client

