

Mini Project

林彥廷 (M11102202)

鄭有宏 (M11102206)

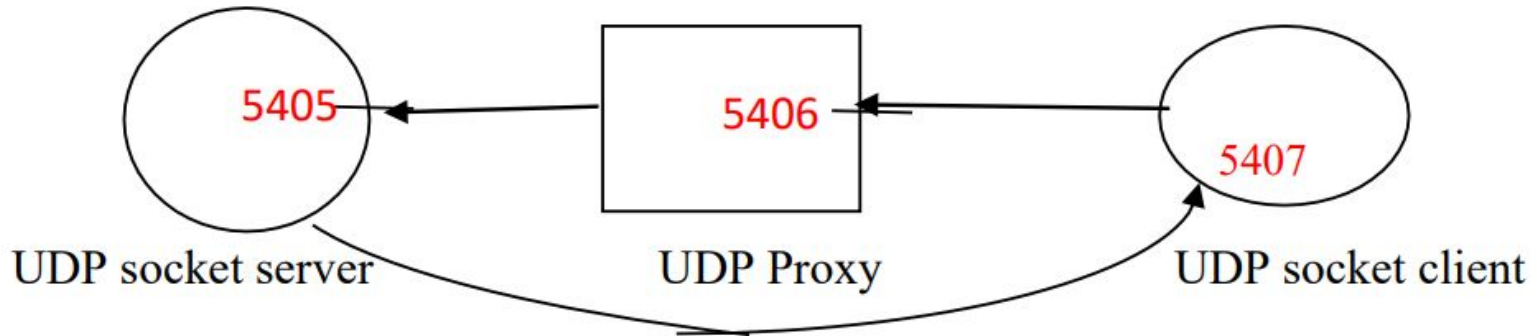
潘宣伊 (M11102266)

Contents

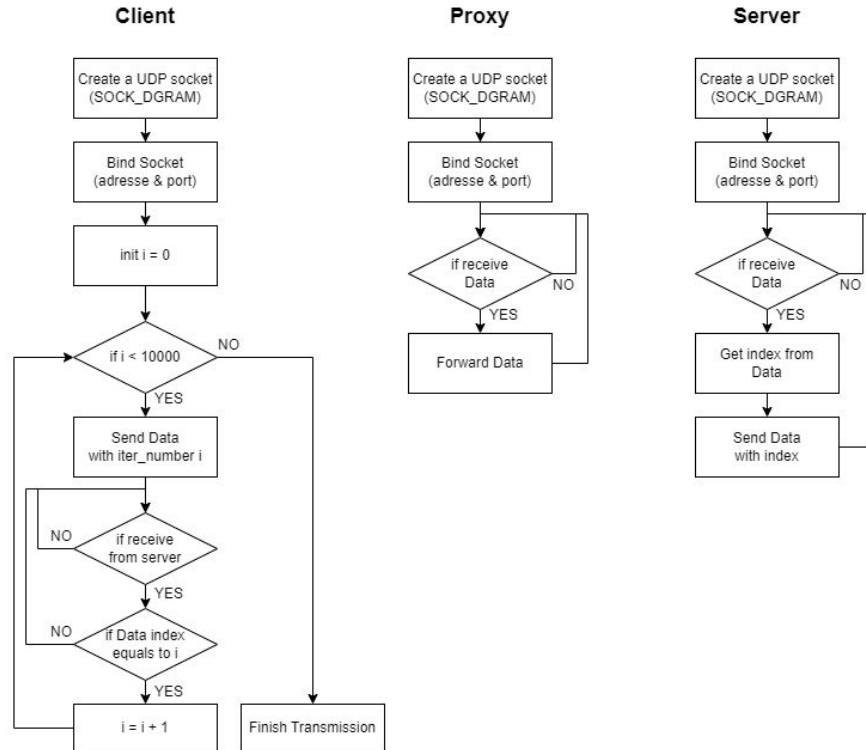
- Basic Connection Setup
- Loss & Delay Function
- Improvement : Loss / Delay Only
- Improvement : Loss & Delay
- Job Partition
- Q & A

Basic Connection Setup

The server listens to the packets, and the socket client send “Hello” to the server. When the socket server receives a “Hello”, it returns a “World” back.



Basic Connection Setup



Basic Connection Setup

This figure shows that the total time to send the message is 7.81 seconds.

```
udp_client.py - mini.project - Visual Studio Code

MINI PROJECT
  Q1
    udp_client.py
    udp_proxy.py
    udp_server.py
  Q2
    udp_client.py
    udp_proxy.py
    udp_server.py
  Q2_delay
    udp_client.py
    udp_proxy.py
    udp_server.py
  Q1_loss
    udp_client.py
    udp_proxy.py
    udp_server.py
  .gitignore
  README.md

Q1 > udp_client.py ~
1 import socket
2 import time
3
4 addr = '127.0.0.1'
5 port_self = 5407
6 port_destin = 5406
7
8 sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
9 sock.bind((addr, port_self))
10 # sock.settimeout(0.12) # at least larger than proxy delay
11
12 t_start = time.time()
13
14 for i in range(10000):
15     data = b'Hello %d' % (i+1)
16     sock.sendto(data, (addr, port_destin))
17     print("send", data, "to proxy")
18
19 while True:
```

```
Receive b'Hello 9992' from proxy
Send b'Hello 9992' to client
Receive b'Hello 9993' from proxy
Send b'Hello 9993' to client
Receive b'Hello 9994' from proxy
Send b'Hello 9994' to client
Receive b'Hello 9995' from proxy
Send b'Hello 9995' to client
Receive b'Hello 9996' from proxy
Send b'Hello 9996' to client
Receive b'Hello 9997' from proxy
Send b'Hello 9997' to client
Receive b'Hello 9998' from proxy
Send b'Hello 9998' to client
Receive b'Hello 9999' from proxy
Send b'Hello 9999' to client
Receive b'Hello 10000' from proxy
Send b'Hello 10000' to client

Receive b'Hello 9992' from client
Forward b'Hello 9992' to server
Receive b'Hello 9993' from client
Forward b'Hello 9993' to server
Receive b'Hello 9994' from client
Forward b'Hello 9994' to server
Receive b'Hello 9995' from client
Forward b'Hello 9995' to server
Receive b'Hello 9996' from client
Forward b'Hello 9996' to server
Receive b'Hello 9997' from client
Forward b'Hello 9997' to server
Receive b'Hello 9998' from client
Forward b'Hello 9998' to server
Receive b'Hello 9999' from client
Forward b'Hello 9999' to server
Receive b'Hello 10000' from client
Forward b'Hello 10000' to server

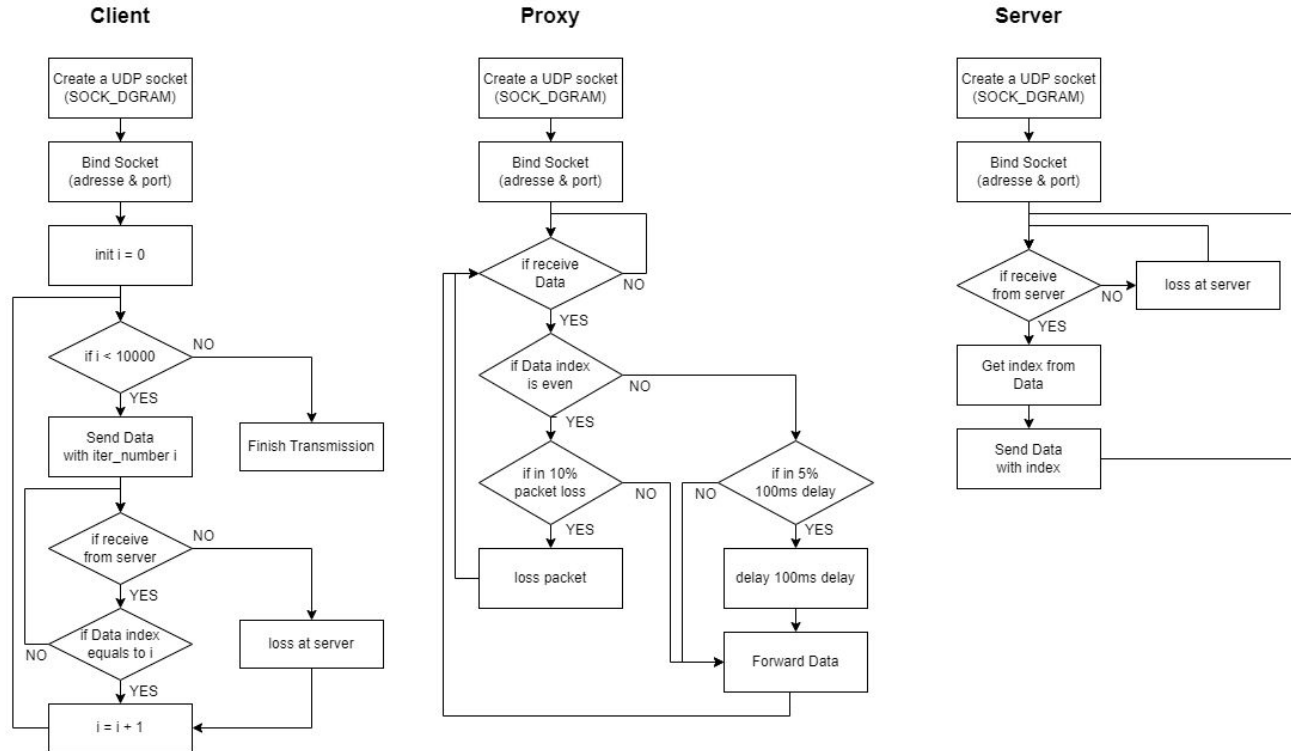
Receive b'world 9993' from server
Send b'Hello 9994' to proxy
Receive b'world 9995' from server
Send b'Hello 9996' to proxy
Receive b'world 9996' from server
Send b'Hello 9997' to proxy
Receive b'world 9997' from server
Send b'Hello 9998' to proxy
Receive b'world 9998' from server
Send b'Hello 9999' to proxy
Receive b'world 9999' from server
Send b'Hello 10000' to proxy
Receive b'world 10000' from server
Total running Time : 7.8108649233845215 seconds
Ps C:\Users\Ven-Ting Lin\OneDrive\文件\项目 - \
  無線通訊網路系統與物聯網應用\mini_projects\
```

Loss & Delay Function

When the UDP proxy receives the “Hello i ” message

- it drops each received packet with 10% probability if i is even number
- it delays 100ms the received packet with 5% probability before forwarding to the server if i is odd number

Loss & Delay Function



Loss & Delay Function

This figure shows that the total time to send the message is 35.96 seconds.

```
Q2 > udp_client.py > ...
1 import socket
2 import time
3
4 addr = '127.0.0.1'
5 port_self = 5487
6 port_destin = 5486
7
8 sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
9 sock.bind((addr, port_self))
10 t_start = time.time()
11
12 for i in range(10000):
13     data = b'Hello %d' % (i+1)
14     sock.sendto(data, (addr, port_destin))
15     print("Send", data, "to proxy")
16
17 while True:
18     # 32768 is max string length
19     data, address_server = sock.recvfrom(32768)
```

Receive b'loss' from proxy
Send b'loss' to client
Receive b'Hello 9992' from proxy
Send b'Hello 9993' to client
Receive b'Hello 9994' from proxy
Send b'Hello 9994' to client
Receive b'Hello 9995' from proxy
Send b'Hello 9995' to client
Receive b'loss' from proxy
Send b'loss' to client
Receive b'Hello 9997' from proxy
Send b'Hello 9997' to client
Receive b'Hello 9998' from proxy
Send b'Hello 9998' to client
Receive b'Hello 9999' from proxy
Send b'Hello 9999' to client
Receive b'Hello 10000' from proxy
Send b'Hello 10000' to client

Receive b'Hello 9992' from client
Forward nothing to server
Receive b'Hello 9993' from client
Forward b'Hello 9993' to server without delay
Receive b'Hello 9994' from client
Forward b'Hello 9994' to server without loss
Receive b'Hello 9995' from client
Forward b'Hello 9995' to server without delay
Receive b'Hello 9996' from client
Forward nothing to server
Receive b'Hello 9997' from client
Forward b'Hello 9997' to server without delay
Receive b'Hello 9998' from client
Forward b'Hello 9998' to server without loss
Receive b'Hello 9999' from client
Forward b'Hello 9999' to server without delay
Receive b'Hello 10000' from client
Forward b'Hello 10000' to server without loss

Receive b'world 9993' from server
Send b'Hello 9994' to proxy
Receive b'world 9994' from server
Send b'Hello 9995' to proxy
Receive b'world 9995' from server
Send b'Hello 9996' to proxy
Receive b'loss' from server
Send b'Hello 9997' to proxy
Receive b'world 9997' from server
Send b'Hello 9998' to proxy
Receive b'world 9998' from server
Send b'Hello 9999' to proxy
Receive b'world 9999' from server
Send b'Hello 10000' to proxy
Receive b'world 10000' from server
Total running time : 35.9558850918579 seconds

Improvement : Loss Only

If we consider only solving the issue of transmission caused by loss function.

In condition of : delay transmission problem still exists

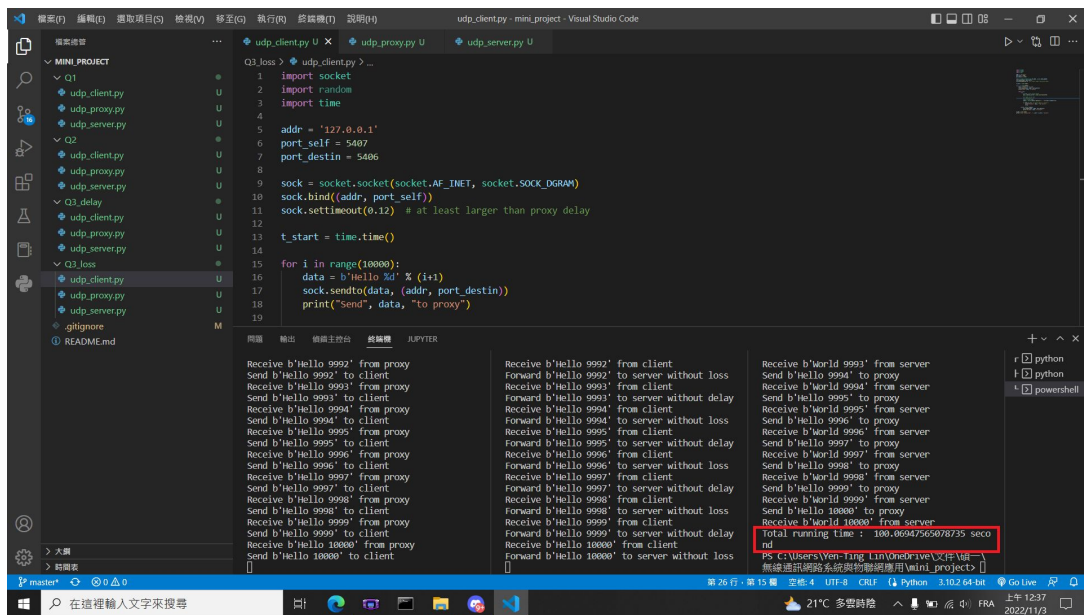
1 : Measuring the delay transmission time = T_{out} (\sim 120 ms)

2 : If waiting time $> T_{out}$, there exist transmission data loss

3 : Re-transmission until the server receives it

Improvement : Loss Only

This figure shows that the total time to send the message is 100.07 seconds.



```
Q3_loss > udp_client.py > ...
1 import socket
2 import random
3 import time
4
5 addr = '127.0.0.1'
6 port_self = 5407
7 port_destin = 5406
8
9 sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
10 sock.bind((addr, port_self))
11 sock.settimeout(0.12) # at least larger than proxy delay
12
13 t_start = time.time()
14
15 for i in range(10000):
16     data = b'hello %d' % (i+1)
17     sock.sendto(data, (addr, port_destin))
18     print("Send", data, "to proxy")
19
```

Receive b'hello 9992' from proxy
Send b'hello 9992' to client
Receive b'hello 9993' from proxy
Send b'hello 9993' to client
Receive b'hello 9994' from proxy
Send b'hello 9994' to client
Receive b'hello 9995' from proxy
Send b'hello 9995' to client
Receive b'hello 9996' from proxy
Send b'hello 9996' to client
Receive b'hello 9997' from proxy
Send b'hello 9997' to client
Receive b'hello 9998' from proxy
Send b'hello 9998' to client
Receive b'hello 9999' from proxy
Send b'hello 9999' to client
Receive b'hello 10000' from proxy
Send b'hello 10000' to client

Receive b'hello 9992' from client
Forward b'hello 9992' to server without loss
Receive b'hello 9993' from client
Forward b'hello 9993' to server without delay
Receive b'hello 9994' from client
Forward b'hello 9994' to server without loss
Receive b'hello 9995' from client
Forward b'hello 9995' to server without delay
Receive b'hello 9996' from client
Forward b'hello 9996' to server without loss
Receive b'hello 9997' from client
Forward b'hello 9997' to server without delay
Receive b'hello 9998' from client
Forward b'hello 9998' to server without loss
Receive b'hello 9999' from client
Forward b'hello 9999' to server without delay
Receive b'hello 10000' from client
Forward b'hello 10000' to server without loss

Receive b'world 9993' from server
Send b'hello 9994' to proxy
Receive b'world 9994' from server
Send b'hello 9995' to proxy
Receive b'world 9995' from server
Send b'hello 9996' to proxy
Receive b'world 9996' from server
Send b'hello 9997' to proxy
Receive b'world 9997' from server
Send b'hello 9998' to proxy
Receive b'world 9998' from server
Send b'hello 9999' to proxy
Receive b'world 9999' from server
Send b'hello 10000' to proxy
Receive b'world 10000' from server
Total running time: 100.0694756078735 seconds

Improvement : Delay Only

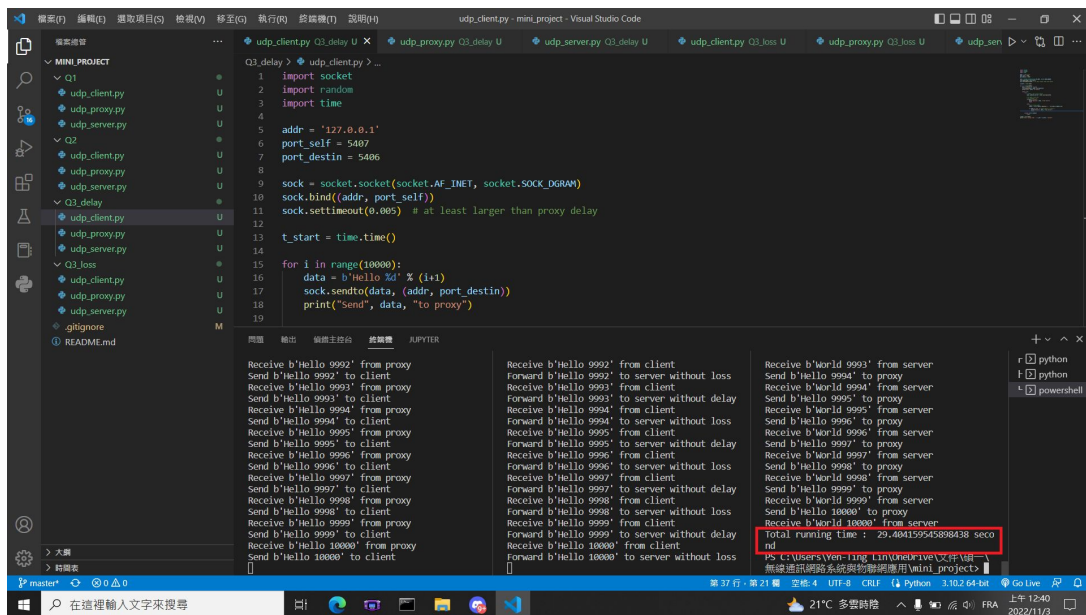
If we consider only solving the issue of transmission caused by delay function.

In condition of : loss transmission problem still exists

- 1 : We don't retransmit the loss message
- 2 : Server knows when is loss message, when is delay message
- 3 : When it is delay message, client send next message without knowing message from server

Improvement : Delay Only

This figure shows that the total time to send the message is 29.40 seconds.



```
Q3_delay > udp_client.py > ...
1 import socket
2 import random
3 import time
4
5 addr = '127.0.0.1'
6 port_self = 5407
7 port_destin = 5406
8
9 sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
10 sock.bind((addr, port_self))
11 sock.settimeout(0.005) # at least larger than proxy delay
12
13 t_start = time.time()
14
15 for i in range(10000):
16     data = b'hello %d' % (i+1)
17     sock.sendto(data, (addr, port_destin))
18     print("Send", data, "to proxy")
19
```

Receive b'hello 9992' from proxy
Send b'hello 9992' to client
Receive b'hello 9993' from proxy
Send b'hello 9993' to client
Receive b'hello 9994' from proxy
Send b'hello 9994' to client
Receive b'hello 9995' from proxy
Send b'hello 9995' to client
Receive b'hello 9996' from proxy
Send b'hello 9996' to client
Receive b'hello 9997' from proxy
Send b'hello 9997' to client
Receive b'hello 9998' from proxy
Send b'hello 9998' to client
Receive b'hello 9999' from proxy
Send b'hello 9999' to client
Receive b'hello 10000' from proxy
Send b'hello 10000' to client

Receive b'hello 9992' from client
Forward b'hello 9992' to server without loss
Receive b'hello 9993' from client
Forward b'hello 9993' to server without delay
Receive b'hello 9994' from client
Forward b'hello 9994' to server without loss
Receive b'hello 9995' from client
Forward b'hello 9995' to server without delay
Receive b'hello 9996' from client
Forward b'hello 9996' to server without loss
Receive b'hello 9997' from client
Forward b'hello 9997' to server without delay
Receive b'hello 9998' from client
Forward b'hello 9998' to server without loss
Receive b'hello 9999' from client
Forward b'hello 9999' to server without delay
Receive b'hello 10000' from client
Forward b'hello 10000' to server without loss

Receive b'world 9993' from server
Send b'hello 9994' to proxy
Receive b'world 9994' from server
Send b'hello 9995' to proxy
Receive b'world 9995' from server
Send b'hello 9996' to proxy
Receive b'world 9996' from server
Send b'hello 9997' to proxy
Receive b'world 9997' from server
Send b'hello 9998' to proxy
Receive b'world 9998' from server
Send b'hello 9999' to proxy
Receive b'world 9999' from server
Send b'hello 10000' to proxy
Receive b'world 10000' from server
Total running time : 29.40159545898438 seconds

Improvement : Loss & Delay

If we consider only solving the issue of transmission caused by loss function.

In condition of : delay transmission problem still exists

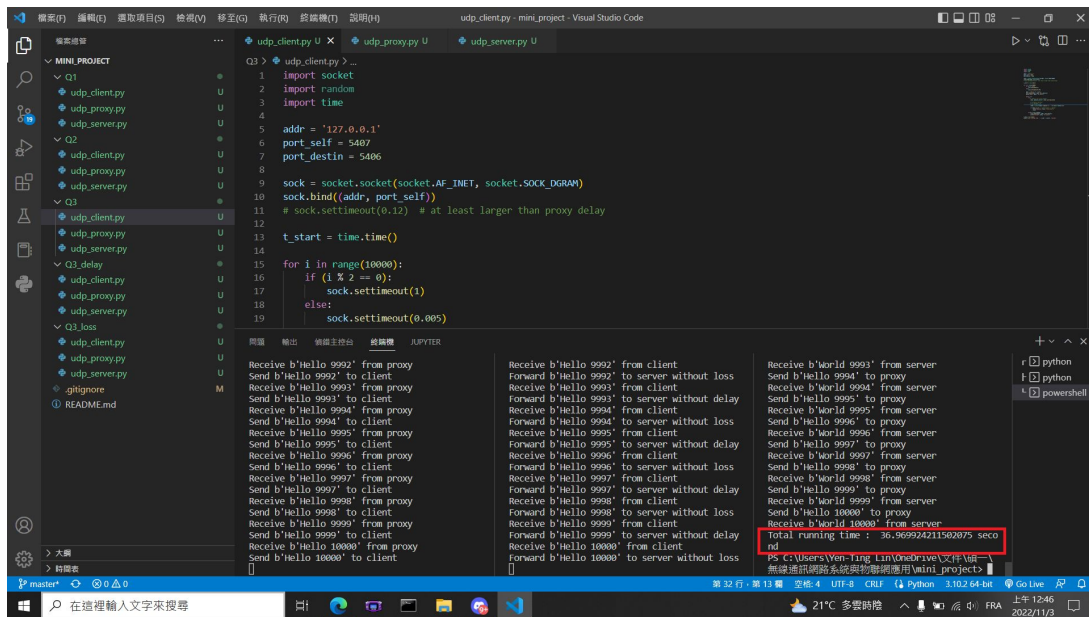
1 : T_{out} of even number is 5ms and timeout of odd number is 100ms

2 : This cannot be used to reduce the latency

3 : Make sure that only need to do one more time retransmission

Improvement : Loss & Delay

This figure shows that the total time to send the message is 36.97 seconds.



The screenshot displays a Visual Studio Code editor with a Python script named `udp_client.py` and its execution output in the terminal. The script is designed to send a series of messages through a proxy server to a destination server, measuring the total time taken.

```
Q3 > udp_client.py > ...
1 import socket
2 import random
3 import time
4
5 addr = '127.0.0.1'
6 port_self = 5407
7 port_destin = 5406
8
9 sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
10 sock.bind((addr, port_self))
11 # sock.settimeout(0.12) # at least larger than proxy delay
12
13 t_start = time.time()
14
15 for i in range(10000):
16     if (i % 2 == 0):
17         sock.settimeout(1)
18     else:
19         sock.settimeout(0.005)
```

The terminal output shows the sequence of messages being sent and received, along with the total running time:

```
Receive b'hello 9992' from proxy
Send b'hello 9992' to client
Receive b'hello 9993' from proxy
Send b'hello 9993' to client
Receive b'hello 9994' from proxy
Send b'hello 9994' to client
Receive b'hello 9995' from proxy
Send b'hello 9995' to client
Receive b'hello 9996' from proxy
Send b'hello 9996' to client
Receive b'hello 9997' from proxy
Send b'hello 9997' to client
Receive b'hello 9998' from proxy
Send b'hello 9998' to client
Receive b'hello 9999' from proxy
Send b'hello 9999' to client
Receive b'hello 10000' from proxy
Send b'hello 10000' to client
```

Total running time : 36.969924211502075 seconds

Job Partition

林彥廷	Q2 Implementation / Flow Chart / Solution Idea	30%
鄭有宏	Q3 Implementation / Writing Report / Solution Idea	40%
潘宣伊	Q1 Implementation / Writing Report / Solution Idea	30%



Q & A