ICN Programming Assignment Report 1

Student ID: b10201054

April 1, 2025

1 TCP transmission

1.1 socket_client

First, we look at socket_client.py. In the Figure 1.(a), we set up the HOST and PORT of client then build a socket. In the Figure 2.(b), we first receive the msg from server, since the msg is encoded by utf-8, we need to decode it to get the msg. Then, we send the question we want to ask with encoding by utf-8. Last, we receive the msg from server again to get the ans. The if block is to send the response to server prompt.

```
# TODO Start
if select_server == 'TA':
    HOST, PORT = '140.112.42.104', 7777
else:
    HOST, PORT = '127.0.0.1', 2077
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((HOST, PORT))
# TODO End
```

(a) set HOST, PORT

```
if line == "Y" or line == "N":
    # If the line is "Y" or "N", treat it as a response to a server pr
    response = line
    log_message(logFile, "Response to server prompt: " + response)

    # Send the response to the server
    # TODO Start
    s.send(response.encode('utf-8'))
    # TODO End
else:
    # If not "Y" or "N," assume it's a mathematical expression
    question = line

    # Receive the server's message without color
    # TODO Start
    server_message = s.recv(1024).decode('utf-8')
    # TODO End

    # Log the server's message
    log_message(logFile, "Received the message from server: ", RESET)
    log_message(logFile, server_message, RED)

    # Send the question to the server
    # TODO Start
    s.send(question.encode('utf-8'))
    # TODO End
    log_message(logFile, "Question: " + question)

    # Receive and log the answer from the server
    # TODO Start
    ans = s.recv(1024).decode('utf-8')      You, 3 分鐘前 • first commi
    # TODO End

    log_message(logFile, "Get the answer from server: ", RESET)
    log_message(logFile, ans, RED)
```

(b) send msg to server

Figure 1: Socket_client.py

1.2 socket_server

The settings of HOST, PORT, socket are same as client part, the unique step is to listen the channel in Figure 2.(b). Listen function is to tell server how much request it

can handle simultaneously. Accept function is to get the client's socket and its address. In the Figure 2.(c), send and receive are same as client part.

```python
# 1. Create a socket
# 2. Bind the socket to the address
# TODO Start
serverSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
serverSocket.bind((HOST, PORT))
# TODO End
```

(a) set HOST, PORT, socket

```python
while True:
    # Listen to a new request with the socket
    # TODO Start
    serverSocket.listen(1)
    # TODO End

    now = datetime.now()
    print("The Server is running..")
    logFile.write(now.strftime("%H:%M:%S ") + "The Server is running
    logFile.flush()

    # Accept a new request and admit the connection
    # TODO Start
    client, address = serverSocket.accept()
    # TODO End

    client.settimeout(15)
    print(str(address) + " connected")
    now = datetime.now()
    logFile.write(now.strftime("%H:%M:%S ") + "connected " + str(addr
    logFile.flush()
```

(b) accept the link from client

```python
try:
    while True:
        client.send(b"Please input a question for calculation")

        # Recieve the data from the client
         (variable) question: str
        question = client.recv(1024).decode('utf-8')
        # TODO End

        now = datetime.now()      You, 5 分鐘前 • first commit
        logFile.write(now.strftime("%H:%M:%S ") + question + '\n'
        logFile.flush()

        # TODO: Call the calculate_expression function here
        ans = calculate_expression(question)

        # Ask if the client want to terminate the process
        message = f"{ans}\nDo you wish to continue? (Y/N)"

        # Send the answer back to the client
        # TODO Start
        client.send(message.encode('utf-8'))
        ans = client.recv(1024).decode('utf-8')
        # TODO End

        # Terminate the process or continue
        if ans.lower() ≠ 'y':
            break
```

(c) send and receive

```
Received the message from server:
Please input a question for calculation
Question: 1+1
Get the answer from server:
2.0
Do you wish to continue? (Y/N)
Response to server prompt: Y
Received the message from server:
Please input a question for calculation
Question: 2-4
Get the answer from server:
-2.0
Do you wish to continue? (Y/N)
Response to server prompt: Y
Received the message from server:
Please input a question for calculation
Question: 3*5
Get the answer from server:
15.0
Do you wish to continue? (Y/N)
Response to server prompt: Y
Received the message from server:
Please input a question for calculation
Question: 4/2
Get the answer from server:
2.0
Do you wish to continue? (Y/N)
Response to server prompt: N
```

(d) Final result at client side

Figure 2: Socket_server.py

2 HTTP server

2.1 Code

In Figure 3.(a)and(b), we do the same thing as socket_server to establish a socket. In Figure 3.(c), we split the msg to get the filename the client wants. In Figure 3.(d), if we can open the file the client asks, we send http response 200 OK, then use sendall to send the whole msg. Otherwise, we send http response 404 to tell the client we didn't find the file you want.

```python
# Server setup
# Specify the IP address and port number (Use "127.0.0.1" for loc
# TODO Start
HOST, PORT = '127.0.0.1', 2077
# TODO end


# 1. Create a socket
# 2. Bind the socket to the address
# TODO Start
serverSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
serverSocket.bind((HOST, PORT))
# TODO End

# Listen for incoming connections (maximum of 1 connection in the
# TODO Start
serverSocket.listen(1)
# TODO End
```

(a) set HOST, PORT

```python
# Start an infinite loop to handle incoming client requests
while True:
    print('Ready to serve...')

    # Accept an incoming connection and get the client's address
    # TODO Start
    connectionSocket, address = serverSocket.accept()
    # TODO End
    print(str(address) + " connected")

    try:
        # Receive and decode the client's request
        # TODO Start
        message = connectionSocket.recv(1024).decode()
        # TODO End

        # If the message is empty, set it to a default value
        if message == "":
```

(b) accept the link from client

```python
    # Receive and decode the client's request
    # TODO Start
    message = connectionSocket.recv(1024).decode()
    # TODO End

    # If the message is empty, set it to a default value
    if message == "":
        message = "/ /"

    # Print the client's request message
    print(f"client's request message: \n {message}")

    # Extract the filename from the client's request
    # TODO Start
    temp = message.split(' ')
    filename = temp[1][1:]
    http_ver = temp[2]
    # TODO End
```

(c) receive

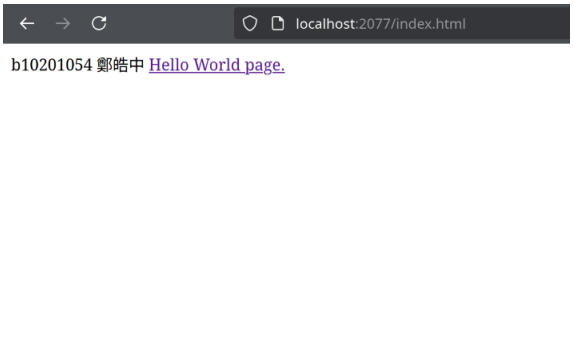
```python
    # 1. Send an HTTP response header to the client
    # 2. Send the content of the requested file to the client line
    # 3. Close the connection to the client
    # TODO Start
    response = http_ver + ' 200 OK\r\n\r\n'
    content = ''.join(outputdata)
    connectionSocket.sendall((response + content).encode('utf-8'))
    connectionSocket.close()
    # TODO End

except IOError:
    # If the requested file is not found, send a 404 Not Found res
    # TODO Start
    response = http_ver + ' 404 NOT FOUND\r\n\r\n'
    content = '404 NOT FOUND'
    connectionSocket.sendall((response + content).encode('utf-8'))
    connectionSocket.close()
    # TODO End
```

(d) send HTTP response

Figure 3: web_server.py

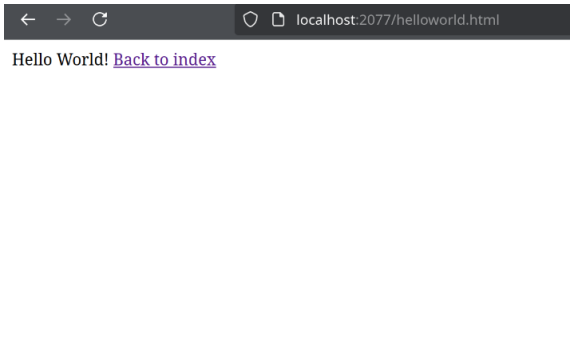
2.2 Result

Here's the result of 2. 3. 4. in Figure 4.



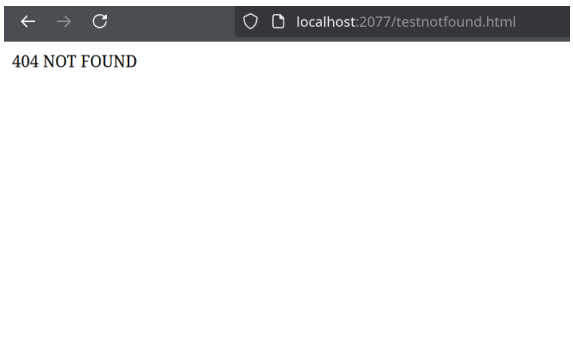
(a) Access index.html (b) Access index.html

Figure 4: Access index.html



(a) Access helloworld.html (b) Access helloworld.html

Figure 5: Access helloworld.html



(a) Access testnotfound.html (b) Access testnotfound.html

Figure 6: Access testnotfound.html

3 Proxy Server

3.1 Code

In the Figure 7.(a) and (b), it is same to setup socket and link. In the Figure 7.(c), if we had already visited the page, it would be saved in cache, so that we can visit the page faster than asking web server again. In the Figure 7.(d) and (e), if we hadn't visited the page yet, the proxy server would ask web server to get the data, then save the data in cache. If the web server HTTP response is 404, then don't cache anything since there is no data at web server.

```python
# Set the server IP address and port
# TODO Start
HOST, PORT = '127.0.0.1', 9999
# TODO end

# Create a server socket, bind it to the specified IP and port, and start li
# TODO Start
serverSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
serverSocket.bind((HOST, PORT))
serverSocket.listen(1)
# TODO end

while True:
    print('Ready to serve...')
    # Accept an incoming connection and get the client's address
    # TODO Start
    client_socket, client_address = serverSocket.accept()
    # TODO end

    print('Received a connection from:', client_address)
```

(a) set HOST, PORT

```python
    try:
        # Receive and parse the client's request
        # TODO Start
        request = client_socket.recv(1024).decode('utf-8')
        # TODO end
        print(request)

        # Extract the requested filename from the HTTP reque
        if request == "":
            request = "/ /"
        filename = request.split()[1].partition("/")[2]
        print(filename)
        file_path = "/" + filename
        print(file_path)
```

(b) accept the link from client

```python
file_exist = "false"
try:
    with open(file_path[1:], "rb") as cache_file:
        output_data = cache_file.read()
    file_exist = "true"

    # ProxyServer finds a cache hit and generates a response
    # Send the file data to the client
    # TODO Start
    response = b'HTTP/1.1 200 OK\r\n\r\n'
    content = output_data
    client_socket.sendall(response + content)
    # TODO End
    print('Read from cache')
```

(c) send and receive

```python
except FileNotFoundError:
    if file_exist == "false":
        # # Establish a connection to the web server
        # TODO Start
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s_HOST, s_PORT = '127.0.0.1', 2077
        # TODO End

        try:
            print("Trying to connect to the web server")
            # Connect the socket to the web server port
            # TODO Start
            s.connect((s_HOST, s_PORT))
            # TODO End
            print("Connected successfully")

            # Send HTTP GET request to the web server
            # TODO Start
            s.send(request.encode('utf-8'))
            # TODO End
            print("Sent the request to the web server successfully")
```

(d) Final result at client side

```python
            # TODO Start
            response = s.recv(1024).decode('utf-8')
            temp = response.split('\r\n\r\n', 1)      You, 53 分鐘前 • fir
            http_msg = temp[0]
            content = temp[1]
            client_socket.sendall(response.encode('utf-8'))
            if '404' not in http_msg:
                with open(file_path[1:], "w") as f:
                    f.write(content)
            # TODO End

            # If not, cache only valid files and send the valid response
            # TODO Start
            # TODO End

        except:
            # Handle errors related to connection issues or invalid reque
            print("Illegal request")

        finally:
            # Ensure the connection to the web server is closed properly
            # TODO Start
            s.close()
```

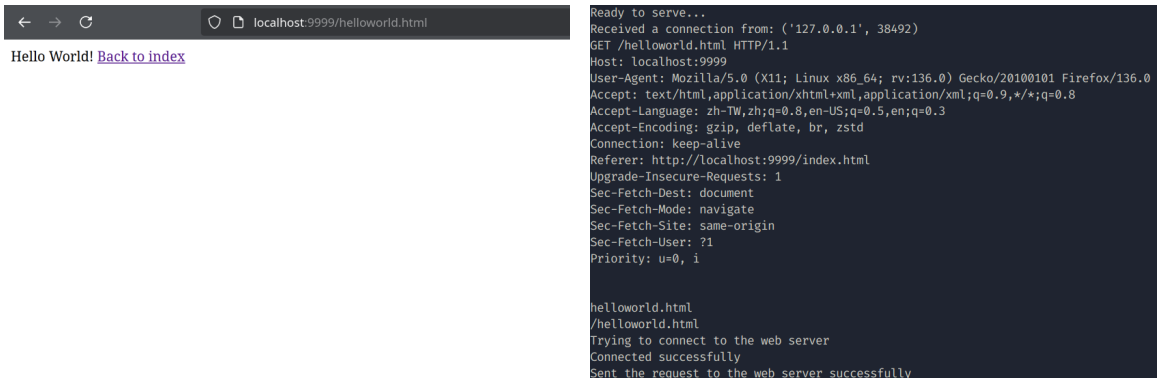
(e) Final result at client side

Figure 7: proxy_server.py

3.2 Result

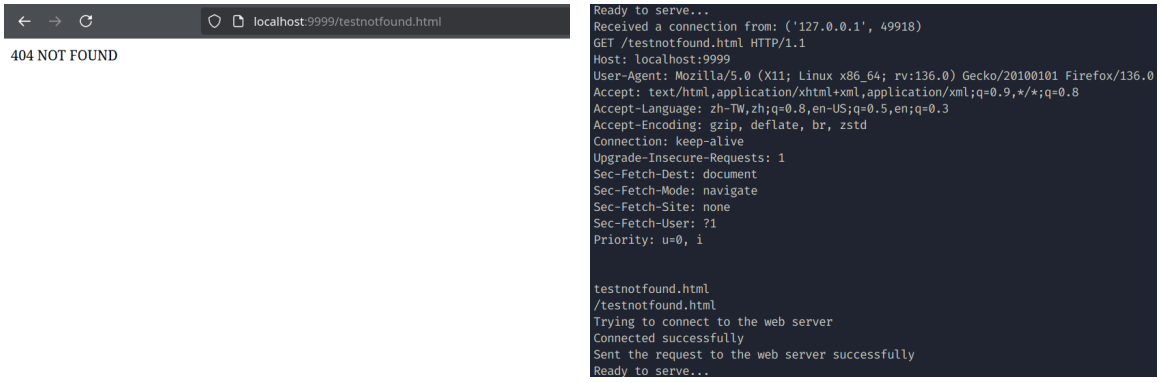
(a) Access index.html (b) Access index.html (c) Access index.html

Figure 8: Access index.html



(a) Access helloworld.html (b) Access helloworld.html

Figure 9: Access helloworld.html



(a) Access testnotfound.html (b) Access testnotfound.html

Figure 10: Access testnotfound.html