

Nama: Vincentzo Lantang

Nim :19210079

Landasan teori mengenai TCP(Transmission Control Protocol):

TCP (Transmission Control Protocol) adalah protokol jaringan yang beroperasi pada lapisan Transport Layer dalam model referensi OSI (Open Systems Interconnection). Protokol ini bertanggung jawab untuk memastikan pengiriman data yang andal, teratur, dan efisien antara dua perangkat dalam jaringan yang terhubung.

Beberapa konsep dasar dari TCP adalah:

Koneksi:

TCP menggunakan pendekatan koneksi yang diinisiasi oleh klien untuk memulai pertukaran data. Pertama, klien mengirimkan permintaan SYN ke server untuk memulai koneksi. Kemudian, server membalas permintaan tersebut dengan SYN-ACK. Akhirnya, klien membalas SYN, menyelesaikan tiga langkah koneksi (three-way handshake), dan koneksi ditetapkan.

Segmen:

TCP memecah data menjadi segmen yang terdiri dari header dan payload (data yang dikirim). Header TCP berisi informasi penting seperti nomor urut segmen, nomor ACK, ukuran window, dll. Segmen dikirim melalui jaringan, dan TCP menerima segmen dari protokol IP untuk diambil payloadnya.

Acknowledgement:

Setelah segmen diterima, penerima akan mengirim pesan ACK (acknowledgement) kembali ke pengirim untuk mengkonfirmasi bahwa segmen tersebut telah diterima. Pesan ACK dapat mengandung nomor urut segmen berikutnya yang diharapkan oleh penerima.

Retransmisi:

Jika pengirim tidak menerima pesan ACK dari penerima dalam waktu tertentu, pengirim akan mengirimkan segmen tersebut kembali. Hal ini dilakukan untuk memastikan pengiriman data yang andal dan teratur.

Flow Control:

TCP menggunakan window flow control untuk mengatur kecepatan pengiriman data. Pengirim akan memonitor ukuran window dari penerima dan mengirimkan data sesuai dengan ukuran window yang tersedia. Penerima akan mengatur window size sesuai dengan kemampuan menerimanya.

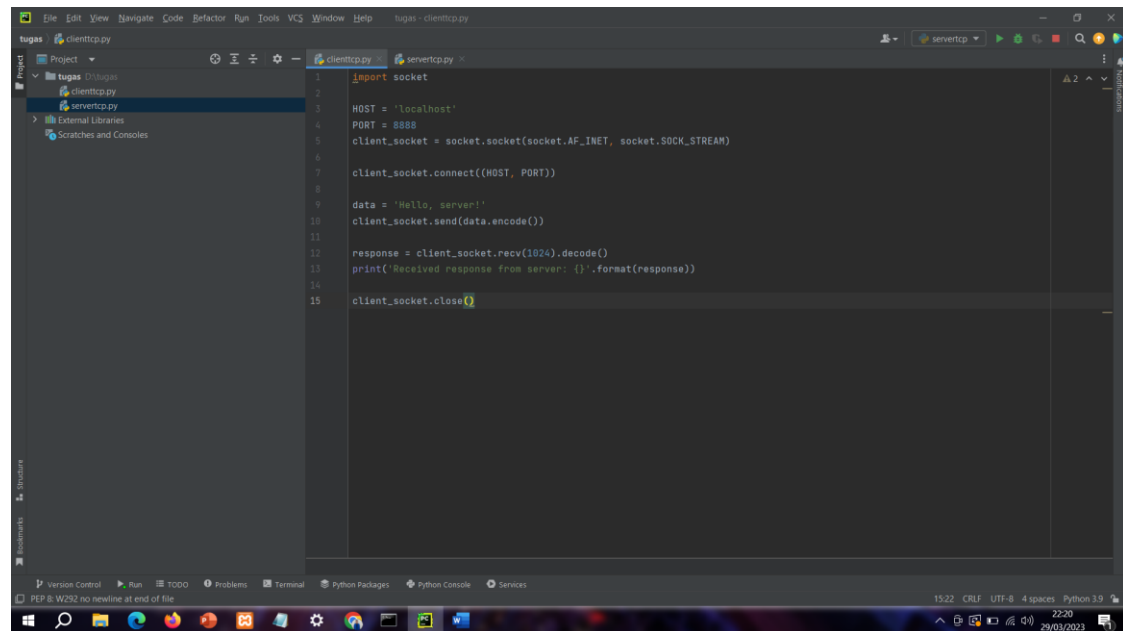
Congestion Control:

TCP juga memiliki mekanisme untuk mengatur laju pengiriman data agar tidak terlalu cepat dan menimbulkan kemacetan (congestion) pada jaringan. TCP akan memonitor keadaan jaringan dan menyesuaikan kecepatan pengiriman data agar sesuai dengan kondisi jaringan.

Dengan konsep-konsep dasar ini, TCP dapat memastikan pengiriman data yang andal, teratur, dan efisien antara dua perangkat dalam jaringan yang terhubung. Hal ini membuat TCP menjadi protokol yang banyak digunakan dalam aplikasi jaringan seperti web browsing, email, file transfer, dan lain sebagainya.

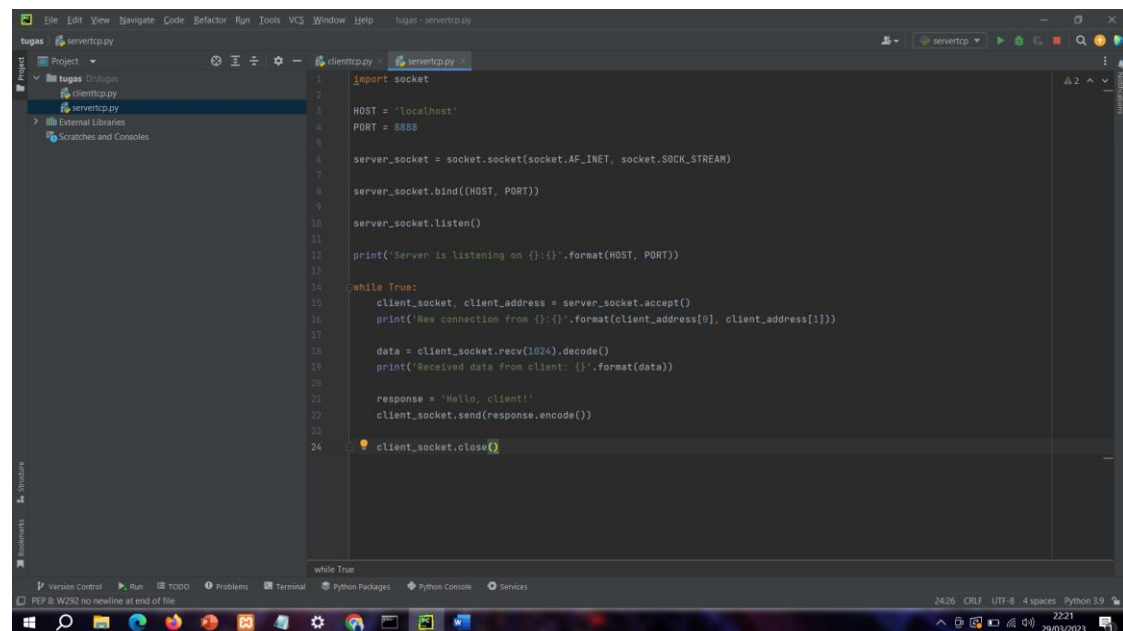
Hasil eksekusi program,screen-shot:

Server:



```
1 import socket
2
3 HOST = 'localhost'
4 PORT = 8888
5 server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
6
7 server_socket.connect((HOST, PORT))
8
9 data = 'Hello, server!'
10 client_socket.send(data.encode())
11
12 response = client_socket.recv(1024).decode()
13 print('Received response from server: {}'.format(response))
14
15 client_socket.close()
```

Client:



```
1 import socket
2
3 HOST = 'localhost'
4 PORT = 8888
5
6 server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
7
8 server_socket.bind((HOST, PORT))
9
10 server_socket.listen()
11
12 print('Server is listening on {}'.format(HOST, PORT))
13
14 while True:
15     client_socket, client_address = server_socket.accept()
16     print('New connection from {}'.format(client_address[0], client_address[1]))
17
18     data = client_socket.recv(1024).decode()
19     print('Received data from client: {}'.format(data))
20
21     response = 'Hello, client!'
22     client_socket.send(response.encode())
23
24     client_socket.close()
```

Penjelasan coding:

Kode di atas adalah implementasi sederhana dari komunikasi jaringan menggunakan protokol TCP antara server dan klien (client). Pada dasarnya, kode ini membuat program yang dapat menerima dan mengirim pesan antara server dan klien melalui jaringan.

Pada sisi server, `program_server()` menggunakan modul `socket` untuk membuat socket server TCP pada host dan port yang ditentukan. Kemudian `server_socket.bind()` digunakan untuk mengikat socket ke host dan port yang ditentukan, dan `server_socket.listen()` digunakan untuk mendengarkan koneksi masuk dari klien. Setelah koneksi berhasil terhubung, `server_socket.accept()` akan mengembalikan koneksi baru (`conn`) dan alamat klien (`address`). Loop `while` digunakan untuk terus menerima dan mengirim pesan dari klien ke server dan sebaliknya.

Pada sisi klien, `program_client()` juga menggunakan modul `socket` untuk membuat socket klien TCP pada host dan port yang ditentukan. Setelah socket klien berhasil terhubung ke server menggunakan `client_socket.connect()`, klien dapat mengirim pesan ke server menggunakan `client_socket.send()` dan menerima pesan dari server menggunakan `client_socket.recv()`. Klien juga menggunakan loop `while` untuk terus mengirim pesan ke server dan menerima pesan balasan dari server.

Secara keseluruhan, kode di atas adalah contoh sederhana penggunaan protokol TCP untuk membuat komunikasi jaringan antara server dan klien. Namun, kode ini dapat diperluas dan dikembangkan lebih lanjut untuk mengimplementasikan fitur-fitur lebih lanjut seperti enkripsi dan autentikasi untuk meningkatkan keamanan komunikasi.