LAPORAN PRAKTIKUM JARINGAN KOMPUTER

SOCKET PROGRAMMING: TCP ECHO CLIENT/SERVER

(PYTHON)

DAMAR GALIH AJI PRADANA (3122521001)

3 D3 TEKNIK INFORMATIKA

PSDKU LAMONGAN

POLITEKNIK ELEKTRONIKA NEGERI SURABAYA

SURABAYA

2020

SOCKET PROGRAMMING: TCP ECHO CLIENT/SERVER

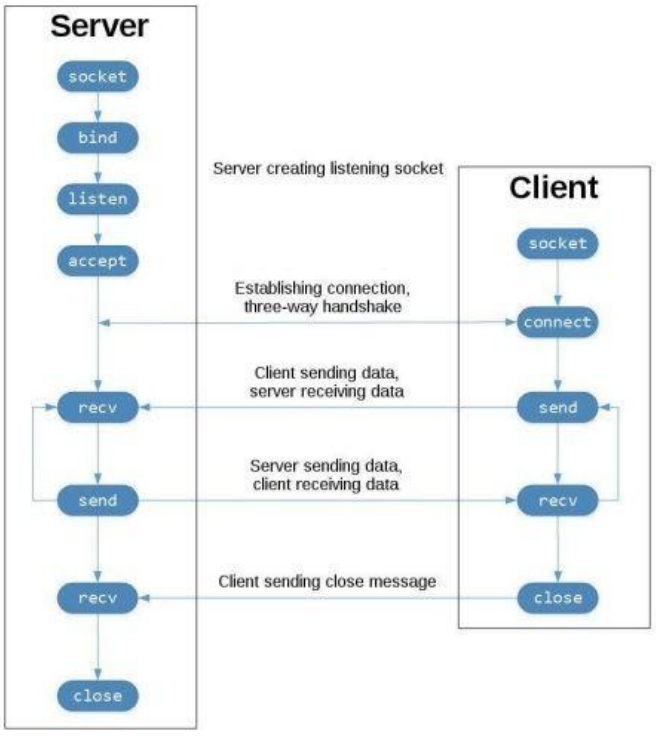
(PYTHON)

A. TUJUAN

1. Mahasiswa dapat menjelaskan konsep jaringan berbasis client-server
2. Mahasiswa dapat menjelaskan konsep pemrograman socket berbasis TCP
3. Mahasiswa dapat menjelaskan cara kerja program TCP echo client/server

B. PENDAHULUAN

Transmission Control Protocol (TCP) merupakan protokol pada transport layer berbasis connection-oriented. TCP menjamin realibitas pengiriman data. Selain itu, TCP memiliki mekanisme three-way handshake sebelum client-server dapat saling mengirimkan data. Pada gambar 1 berikut menunjukkan alur komunikasi client-server mengggunakan TCP.



Gambar 1. TCP Socket

Pada gambar 1 menunjukkan bahwa di sisi server terdapat beberapa socket API yang digunakan agar dapat berkomunikasi pada jaringan. Socket API pada TCP server yang bertugas untuk membuat listening socket yaitu: socket(), bind(), listen(), accept(). Proses listening socket merupakan saat dimana server menunggu koneksi dari client. Jadi, saat client menghubungkan ke server dengan connect(), hal ini mengawali proses connection establishment. Kemudian server akan menerima koneksi dari client melalui accept(), dan mekanisme three-way handshake selesai. Selanjutnya, client dan server dapat saling berkomunikasi atau bertukar data menggunakan send() dan recv(). Pada praktikum kali ini, kita akan mempelajari socket programming, dimana program yang akan dibuat, memungkinkan client dan server berkomunikasi. Program echo server dan echo client menggunakan protokol TCP. Server akan membalas atau mengulang kembali data yang dikirim oleh client.

C. PROSEDUR PERCOBAAN

1. Buatlah program *server*.py

```
import socket
import sys
import argparse

host = 'localhost'
data_payload = 1024  # Changed payload size
backlog = 5
def echo_server(port):
    try:
        server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        server_address = (host, port)
        print("Starting up echo server on %s port %s" % server_address)

        server_socket.bind(server_address)
        server_socket.listen(backlog)

        while True:
            print("Waiting to receive message from client")
            client, address = server_socket.accept()
            print("Connected to client at %s" % str(address))
            data = client.recv(data_payload)

            if data:
                print("Data: %s" % data.decode())  # Decode bytes to string
                client.send(data)
                print("Sent %d bytes back to %s" % (len(data), address))
            client.close()

    except KeyboardInterrupt:
        print("Server is shutting down.")
    except PermissionError as e:
        print("Permission error: %s" % str(e))
    except AddressInUseError as e:
        print("Address in use error: %s" % str(e))
    except Exception as e:
        print("Error: %s" % str(e))
    finally:
        server_socket.close()

if __name__ == '__main__':
    parser = argparse.ArgumentParser(description='Socket Server Example')
    parser.add_argument('--port', action="store", dest="port", type=int, required=True)
    given_args = parser.parse_args()
    port = given_args.port
    echo_server(port)
```

2. Buatlah program *client*.py

```
import socket
import sys
import argparse

host = 'localhost'
data_payload = 1024  # Changed payload size

def echo_client(port):
    """
    Fungsi untuk mengirim pesan ke server dan menerima pesan balasan dari server.

    Args:
    - port (int): nomor port yang akan digunakan untuk menghubungi server.

    Cara menggunakan:
    Panggil fungsi echo_client dengan memberikan nomor port yang akan digunakan sebagai argumen.
    """

    try:
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        server_address = (host, port)
        print("Menghubungi %s pada port %s" % server_address)
```

```
        sock.connect(server_address)

        message = "Test message. This will be echoed"
        print("Mengirim pesan: %s" % message)
        sock.sendall(message.encode('utf-8'))
        amount_received = 0
        amount_expected = len(message)

        while amount_received < amount_expected:
            data = sock.recv(data_payload)
            amount_received += len(data)
            if not data:
                break
            print("Menerima: %s" % data.decode('utf-8'))

    except socket.error as e:
        print("Terjadi kesalahan socket: %s" % str(e))
    except Exception as e:
        print("Terjadi kesalahan lain: %s" % str(e))
    finally:
        print("Menutup koneksi ke server")
        sock.close()

if __name__ == '__main__':
    parser = argparse.ArgumentParser(description='Socket Client Example')
    parser.add_argument('--port', action="store", dest="port", type=int, required=True)
    given_args = parser.parse_args()
    port = given_args.port
    echo_client(port)
```

3. Jalankan program server.py terlebih dulu. Tampilkan hasil pengujian anda. (Catatan : port tersebut adalah port server, dimana client nantinya akan menggunakan port tersebut agar terhubung ke server, gunakan non privileged ports yaitu port > 8080)

```
user@D4MAR-G4LIH MINGW64 /d/Pyhton/Socket
$ python Server.py --port 8080
```

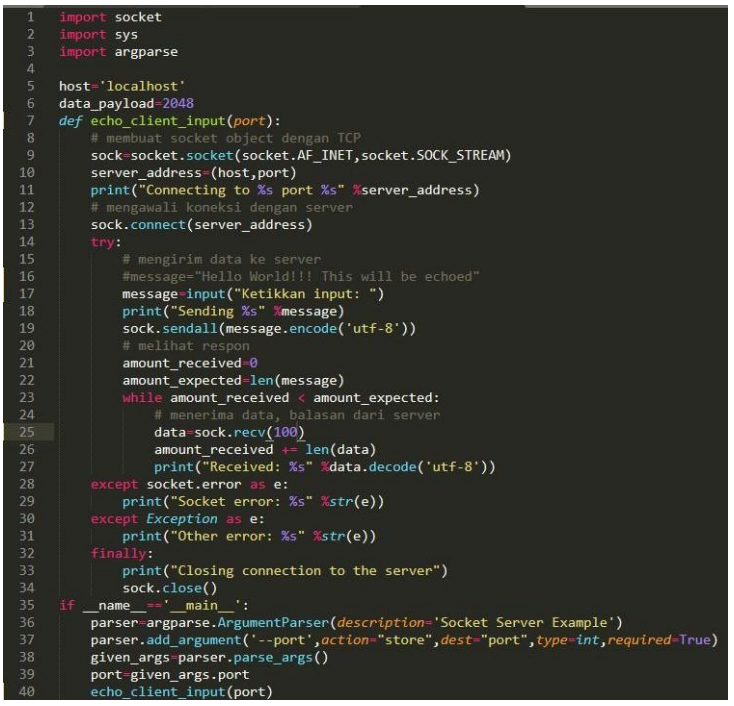
4. Jalankan program client.py secara terpisah terminal atau command prompt dari program echo server. Tampilkan hasil pengujian anda.

```
PS D:\Pyhton\Socket> py Client.py --port=8080
Connecting to localhost port 8080
Sending Test message. This will be echoed
Received: Test message. This will be echoed
Closing connection to the server
PS D:\Pyhton\Socket>
```

Sedangkan pada server akan tampil seperti berikut:

Beberapa saat kemudian akan tampil tulisan seperti berikut

5. Pada pengujian di prosedur 3 dan 4 menggunakan localhost, anda dapat juga menggunakan host yang berbeda dengan menyesuaikan IP address tujuan.
6. Buatlah program input client/server, dimana anda bisa memberikan inputan melalui program client kemudian server akan membalas dengan mengirimkan ulang ke client apapun yang anda kirimkan sebelumnya. Program client dapat membaca inputan dari keyboard dan anda dapat mengirimkan data ke server beberapa kali selama koneksi dari client belum diakhiri.



```python
import socket
import sys
import argparse

host='localhost'
data_payload=2048
def echo_client_input(port):
    # membuat socket object dengan TCP
    sock=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
    server_address=(host,port)
    print("Connecting to %s port %s" %server_address)
    # mengawali koneksi dengan server
    sock.connect(server_address)
    try:
        # mengirim data ke server
        #message="Hello World!!! This will be echoed"
        message=input("Ketikkan input: ")
        print("Sending %s" %message)
        sock.sendall(message.encode('utf-8'))
        # melihat respon
        amount_received=0
        amount_expected=len(message)
        while amount_received < amount_expected:
            # menerima data, balasan dari server
            data=sock.recv(100)
            amount_received += len(data)
            print("Received: %s" %data.decode('utf-8'))
    except socket.error as e:
        print("Socket error: %s" %str(e))
    except Exception as e:
        print("Other error: %s" %str(e))
    finally:
        print("Closing connection to the server")
        sock.close()
if __name__=='__main__':
    parser=argparse.ArgumentParser(description='Socket Server Example')
    parser.add_argument('--port',action="store",dest="port",type=int,required=True)
    given_args=parser.parse_args()
    port=given_args.port
    echo_client_input(port)
```

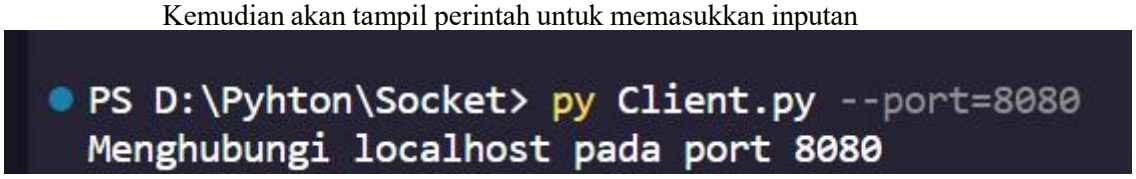
7. Lampirkan program echo client/server yang anda buat untuk prosedur no. 6 pada laporan anda. Tampilkan hasil setelah anda menjalankan program yang anda buat tersebut.

PROGRAM

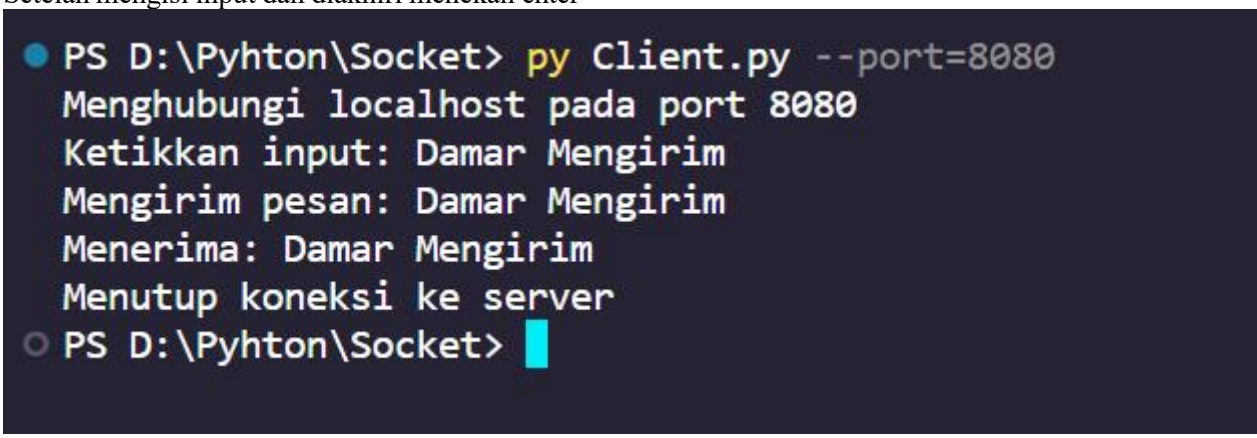
```python
import socket
import argparse

host = 'localhost'
data_payload = 1024
def echo_client(port):
    try:
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        server_address = (host, port)
        print("Menghubungi %s pada port %s" % server_address)
        sock.connect(server_address)
        message = input("Ketikkan input: ")
        print("Mengirim pesan: %s" % message)
        sock.sendall(message.encode('utf-8'))
        amount_received = 0
        amount_expected = len(message)
        while amount_received < amount_expected:
            data = sock.recv(data_payload)
            amount_received += len(data)
            if not data:
                break
            print("Menerima: %s" % data.decode('utf-8'))
    except socket.error as e:
        print("Terjadi kesalahan socket: %s" % str(e))
    except Exception as e:
        print("Terjadi kesalahan lain: %s" % str(e))
    finally:
        print("Menutup koneksi ke server")
        sock.close()
if __name__ == '__main__':
    parser = argparse.ArgumentParser(description='Socket Client Example')
    parser.add_argument('--port', action="store", dest="port", type=int, required=True)
    given_args = parser.parse_args()
    port = given_args.port
    echo_client(port)
```

HASIL

Kemudian akan tampil perintah untuk memasukkan inputan



Setelah mengisi input dan diakhiri menekan enter



8. Analisa program echo client/server pada prosedur 1 dan 2 serta pada tugas di prosedur. Jelaskan bagaimana penerapan mekanisme TCP pada socket programming dan perintah-perintah yang digunakan pada program socket yang telah dibuat.

ANALISA

Jaringan client server didefinisikan sebagai suatu arsitektur jaringan komputer dimana perangkat client melakukan proses meminta data, dan server yang memiliki tugas untuk memberikan respon berupa data terhadap request tersebut. Perangkat client biasanya berupa perangkat komputer dengan aplikasi software jaringan yang telah terinstal guna untuk meminta dan menerima data melalui jaringan. Salah satu contoh aplikasi software yang

paling sering digunakan untuk meminta dan menerima data pada jaringan ialah web browser, dimana user dapat melakukan request untuk sebuah halaman web, melalui aplikasi web browser (persis seperti yang anda lakukan saat ini) Perangkat lain yang dapat pula dikategorikan sebagai client ialah perangkat mobile seperti smartphone atau tablet. Server merupakan sebuah komputer yang dirancang khusus untuk melayani client dengan memproses request yang telah diterima dari client lalu kemudian mengirimkan kembali respon data kepada client melalui jaringan. Server menyimpan informasi dan data yang kompleks yang mungkin dibutuhkan client, oleh karena itu biasanya server terdiri dari komputer dengan performa yang tinggi baik dari segi pemrosesan maupun dari segi memori, hal tersebut agar server mampu melayani request dari banyak client secara bersamaan.

Socket programming adalah pemrograman yang menggunakan socket. Socket ini semacam terowongan/tunnel yang bisa dipakai untuk komunikasi/pertukaran arah secara bolak-balik. Dengan socket programming, komunikasi dapat terjalin antara bahasa pemrograman yang berbeda, antara tingkatan user yang berbeda, bahkan antar komputer yang berbeda atau gabungan ketiganya.

Secara garis besar langkah–langkah yang dilakukan pada client adalah sebagai berikut dimulai dengan membuka koneksi client ke server, yang di dalamnya adalah membuat socket dengan perintah socket(), kemudian melakukan pengalamatan ke server selanjutnya menghubungi server dengan connect(). Setelah itu melakukan komunikasi (mengirim dan menerima data), dengan menggunakan perintah recv() dan sendall(). Dan terakhir menutup hubungan dengan perintah close(). Untuk menampilkan hasil pada terminal menggunakan perintah echo().

Sama halnya dengan langkah – langkah yang dilakukan pada client, untuk membuat server yang harus

dilakukan antara lain dimuali dengan membuat socket dengan perintah socket(). Selanjutnya mengikatkan socket kepada sebuah alamat network dengan perintah bind(). Kemudian menyiapkan socket untuk menerima koneksi yang masuk dengan perintah listen(). Untuk menerima koneksi yang masuk ke server dengan perintah accept()Melakukan komunikasi (mengirim dan menerima data), dengan menggunakan perintah recv() dan sendall( ). Dan untuk menampilkan hasil pada terminal menggunakan perintah echo().

Jaringan komputer dalam melakukan pengiriman data bisa diilustrasikan sebagai pengiriman surat. Supaya surat bisa terkirim secara benar maka alamat pengirim dan penerima harus tertulis dengan jelas dan lengkap. Begitu juga dalam koneksi di socket, diperlukan variable yang dipakai untuk menyimpan address client dan server. Address ini akan dipakai pada waktu melakukan connect(), bind() dan accept().

Proses dari kerja aplikasi yang menggunakan TCP adalah untuk bisa melakukan koneksi client server, program server harus berjalan terlebih dahulu. Di sisi server disediakan sebuah socket, yang disebut welcoming socket yang fungsinya untuk mendeteksi adanya permintaan koneksi dari sisi client. Di sisi client terdapat client socket. Jika ingin menghubungi server, maka melalui client socket-nya, client membuat inisialisai koneksi ke welcoming socket milik server, dengan mode three-way handshake. Setelah welcoming socket menerima inisialisasi koneksi dari client socket, aplikasi server akan membuat connection socket di sisi server. Dengan connection socket ini, client socket dan connection socket berinteraksi satu sama lain untuk mengirim dan menerima data. Client membaca data yang dikirim oleh server dari client socket-nya. Kemudian menampilkan data tersebut di

monitor. TCP mengharuskan terjadinya koneksi terlebih dahulu, kemudian mengirimkan paket-paket data secara berurutan, penerima juga dijamin akan menerima data dengan urutan yang benar, dimulai dari data pertama yang dikirimkan hingga data terakhir. TCP dapat menangani data yang hilang, rusak, terpecah, ataupun terduplikasi.

KESIMPULAN

Berdasarkan praktikum dan analisa dapat ditarik kesimpulan sebagai berikut :

1. Socket programming adalah pemrograman yang menggunakan socket untuk komunikasi atau pertukaran arah secara bolak-balik.
2. Jaringan client server merupakan suatu arsitektur jaringan komputer dimana perangkat client melakukan proses meminta data, dan server yang memiliki tugas untuk memberikan respon berupa data terhadap request tersebut.
3. TCP memerlukan pertukaran data dua arah yang valid

TUGAS PERTANYAAN

1. Jelaskan beberapa keunggulan protokol TCP
   a Bisa digunakan untuk menghubungkan meisn-mesin dengan perangkat lunak, serta bersifat *open protocol standart*
   b Memiliki tingkat konsistensi yang tinggi, sehingga banyak penggunanya
   c TCP/IP berkembang dengan menggunakan standar protokol terbuka, maksudnya TCP/IP ini tersedia secara luas. Efek dari standar yang baru itu membuat semua orang bisa mendevelop software mereka sendiri sehinga dapat berkomunikasi dengan protokol ini. Hal ini tentu membuat pemakaian TCP/IP meluas dengan sangat cepat, terutama dari sisi pengadopsian oleh berbagai OS dan aplikasi jaringan.
   d TCP/IP tidak bergantung terhadap hardware (tidak terikat pada jenis perangkat

keras khusus) atau OS tertentu, hal ini membuat TCP/IP available untuk menyatukan bermacam macam jaringan. Contoh: Ethernet, dial-up, dll.

e TCP/IP menggunakan teknik pengalamatan yang unik dalam skala besar, bisa dibilang Global. Sehingga memungkinkan komputer dapat mengidentidifikasikan secara unik komputer yang lain didalam seluruh jaringan, walau sebesar Worlwide Internet pun bisa diidentifikasi. Setiap komputer yang tersambung dengan jaringan TCP/IP (Internet) akan memiliki address yang hanya dimiliki olehnya.

2. Jelaskan server socket method dan client socket method pada socket programming mengggunakan Python

Socket adalah mekanisme komunikasi untuk pertukaran data antar aplikasi yang terdapat di dalam sebuah mesin maupun beda mesin dan pertukaran ini terjadi pada sebuah jaringan komputer. Socket biasa digunakan untuk

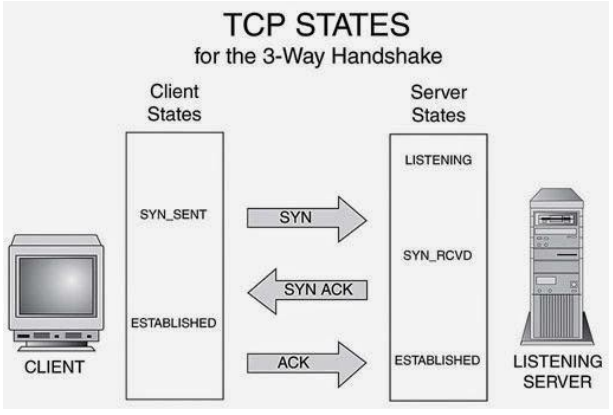
pemrograman berbasis client – server yang dapat menggunakan socket TCP/IP atau socket UDP.

a. Socket Server: digunakan untuk menangani seluruh request yang dikirimkan dari socket client.
b. Socket Client: sebuah penghubung antara dua host, yang dapat dibangun dengan tujuh dasar operasi.1

Terdapat beberapa method, yaitu:

- **socket()** = method yang digunakan untuk menkonfigurasi koneksi, parameter **AF_INET** merujuk ke alamat ipv4 dan **SOCK_STREAM** menandakan bahwa koneksi berorientasi pada protokol TCP.
- **setsockopt()** = method yang digunakan untuk mengontrol socket behaviour.
- **bind()** = method yang memungkinkan Anda untuk melampirkan objek ke suatu fungsi.
- **listen()** = method yang digunakan untuk mendapatkan koneksi masuk dari client.
- **accept()** = method yang digunakan untuk menerima koneksi dari client.
- **send()** = method yang digunakan untuk mengirimkan pesan ke socket lain (client).
- **recv()** = digunakan untuk menerima pesan dari soket, dan dapat digunakan untuk menerima data pada soket yang berorientasi koneksi atau tidak.

3. Jelaskan yang anda pahami mengenai mekanisme three-way handshake.



Three-way-handshake yang juga dikenal sebagai jabat tangan TCP adalah metode yang digunakan dalam jaringan TCP / IP untuk membuat koneksi antara host / client dan server lokal. Ini adalah metode tiga langkah yang membutuhkan baik klien dan server untuk bertukar SYN dan ACK (pengakuan) paket sebelum komunikasi data aktual dimulai. Metode ini dirancang sehingga ketika dua komputer mencoba memulai komunikasi akan

dilakukan negosiasi parameter jaringan TCP terlebih dahulu sehingga kedua komputer dapat memulai dan bernegosiasi terpisah TCP koneksi socket pada waktu yang sama.