Dokumentasi Program

Fachrad Zauhar (50421429)

4IA10

1 Pendahuluan

Ada banyak hal yang ditawarkan dari internet untuk penggunanya. Meskipun tidak ada layanan Infrastruktur Komunikasi yang mendasarinya, Internet menyajikan tujuan umumnya berupa mekanisme komunikasi Aplikasi Internet dan Pemrograman Jaringan.

1.1 Pemrograman Jaringan

Kumpulan komputer yang saling berhubungan dan berkomunikasi antara satu sama lain disebut dengan Jaringan Komputer (Essy Malays Sari Bakti, 2023). Menurut (Fardian Anshori, 2019), Pemrograman Jaringan adalah praktik dalam mengelola, mengendalikan, dan mengoptimalkan fungsi jaringan menggunakan skrip atau bahasa pemrograman.

1.2 Socket Programming

Socket merupakan suatu abstraksi yang memungkinkan masuk ke dalam jaringan untuk berkomunikasi dengan aplikasi lain dalam jaringan yang sama. Jadi, *Socket Programming* adalah pemrograman dengan tujuan program-program yang ada bisa berkomunikasi satu sama lain pada jaringan yang sama.

2 Komponen Utama

1. Pengaturan Soket:
    - `PORT`: Port yang digunakan oleh server untuk mendengarkan koneksi masuk.
    - `SERVER`: Alamat IP server, ditentukan menggunakan `socket.gethostbyname(socket.gethostname())`.
    - `ADDR`: Tuple yang berisi alamat server dan port.
    - `CHUNKSIZE`: Ukuran potongan data yang dikirim atau diterima melalui jaringan.

- `SEPARATOR`: String yang digunakan untuk memisahkan nama file dan ukuran file dalam protokol komunikasi.

2. Fungsi Utama :

   - `start_server()`: Membuat soket server, mengikatnya ke alamat dan port yang ditentukan, dan mulai mendengarkan koneksi masuk. Ketika koneksi diterima, fungsi ini membuat thread baru untuk menangani klien tersebut dengan memanggil `handle_client()`.
   - `handle_client(conn, addr)`: Menangani komunikasi dengan klien. Menerima perintah dari klien ("SEND" atau "RECEIVE"). Jika perintah adalah "SEND", server menerima file dari klien dengan memanggil `receive_file()`. Jika perintah adalah "RECEIVE", server mengirim file yang diminta ke klien dengan memanggil `send_file()`. Jika file tidak ditemukan, server mengirim pesan kesalahan ke klien.
   - `send_file(conn, filename)`: Mengirim file ke klien. Mengirim nama file dan ukuran file terlebih dahulu, diikuti oleh data file dalam potongan-potongan.
   - `receive_file(conn)`: Menerima file dari klien. Menerima metadata file (nama dan ukuran) terlebih dahulu, kemudian menerima data file dalam potongan-potongan dan menulisnya ke file.

3. Eksekusi Utama :

   - Script memulai server dengan memanggil `start_server()`.

3 Alur Kerja

1. Menerima Koneksi

   - Server mendengarkan koneksi masuk pada port yang ditentukan. Ketika koneksi diterima, server membuat thread baru untuk menangani klien tersebut.

2. Menangani Perintah dari Client

   - Jika perintah adalah "SEND"
     - Server mengirim respons "OK" ke klien.
     - Server menerima nama file dari klien.
     - Server memanggil `receive_file()` untuk menerima file dari klien.
     - Server mencetak pesan bahwa file telah diterima.
   - Jika perintah adalah "RECEIVE"

- Server mengirim respons "OK" ke klien.
- Server menerima nama file dari klien.
- Jika file ada, server memanggil `send_file()` untuk mengirim file ke klien.
- Jika file tidak ditemukan, server mengirim pesan kesalahan ke klien.
- Server mencetak pesan bahwa file telah dikirim atau kesalahan jika file tidak ditemukan.

3. Mengirim dan Menerima File

   - Server menggunakan fungsi `send_file()` dan `receive_file()` untuk mengirim dan menerima file. Data file dikirim dan diterima dalam potongan-potongan untuk efisiensi.

4. Penanganan Masalah

   - Server mencetak kesalahan ke konsol jika terjadi masalah selama komunikasi dengan klien. Server menutup koneksi dengan klien setelah selesai menangani perintah atau jika terjadi kesalahan.

4 Kodingan

Listing 1: Kode fileserver.py

```
import socket
import threading
import os

PORT = 5050
SERVER = socket.gethostbyname(socket.gethostname())
ADDR = (SERVER, PORT)
CHUNKSIZE = 4096
SEPARATOR = "<SEPARATOR>"

def start_server():
    """
    Memulai server untuk menerima koneksi dan memproses file yang
    dikirim/diterima dari client.
    """
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind(ADDR)
    server.listen()
    print(f"[LISTENING] Server listening on {SERVER}:{PORT}")

    while True:
        conn, addr = server.accept()
        threading.Thread(target=handle_client, args=(conn, addr)).start
    ()
```

```
23
24  def handle_client(conn, addr):
25      """
26      Menangani koneksi client berdasarkan perintah yang dikirim (SEND
        atau RECEIVE).
27
28      Args:
29          conn (socket.socket): Objek koneksi socket.
30          addr (tuple): Alamat client.
31      """
32      print(f"[NEW CONNECTION] {addr}")
33      try:
34          cmd = conn.recv(1024).decode().strip()
35          if cmd == "SEND":
36              conn.send("OK".encode())
37              filename = conn.recv(1024).decode().strip()
38              receive_file(conn)
39              print(f"[RECEIVED] File '{filename}' dari {addr}")
40
41          elif cmd == "RECEIVE":
42              conn.send("OK".encode())
43              filename = conn.recv(1024).decode().strip()
44              if os.path.exists(filename):
45                  send_file(conn, filename)
46                  print(f"[SENT] File '{filename}' ke {addr}")
47              else:
48                  conn.send("ERROR: File not found".encode())
49                  conn.shutdown(socket.SHUT_WR)
50                  print(f"[ERROR] File '{filename}' tidak ditemukan")
51
52          else:
53              print(f"[INVALID CMD] {addr}: {cmd}")
54      except Exception as e:
55          print(f"[EXCEPTION] {addr}: {e}")
56      finally:
57          conn.close()
58          print(f"[DISCONNECTED] {addr}")
59
60  def send_file(conn, filename):
61      """
62      Mengirim file ke client.
63
64      Args:
65          conn (socket.socket): Objek koneksi socket.
66          filename (str): Nama file yang akan dikirim.
67      """
68      filesize = os.path.getsize(filename)
69      conn.send(f"{filename}{SEPARATOR}{filesize}".encode())
70      with open(filename, "rb") as f:
71          while chunk := f.read(CHUNKSIZE):
72              conn.sendall(chunk)
73
74  def receive_file(conn):
75      """
76      Menerima file dari client dan menyimpannya di server.
77
78      Args:
```

```
        conn (socket.socket): Objek koneksi socket.
    """
    meta = conn.recv(1024).decode()
    filename, filesize = meta.split(SEPARATOR)
    filename = os.path.basename(filename)
    filesize = int(filesize)
    with open(filename, "wb") as f:
        received = 0
        while received < filesize:
            chunk = conn.recv(CHUNKSIZE)
            if not chunk: break
            f.write(chunk)
            received += len(chunk)

if __name__ == "__main__":
    """
    Entry point dari program server.
    """
    start_server()
```

Listing 2: Kode fileclient.py

```
import socket
import os

PORT = 5050
SERVER = socket.gethostbyname(socket.gethostname())
ADDR = (SERVER, PORT)
CHUNKSIZE = 4096
SEPARATOR = "<SEPARATOR>"

def send_file_to_server(filename):
    """
    Mengirim file ke server.

    Args:
        filename (str): Nama file yang akan dikirim.
    """
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as client:
        client.connect(ADDR)
        client.send(b"SEND")
        if client.recv(1024).decode().strip() == "OK":
            client.send(filename.encode())
            send_file(client, filename)

def receive_file_from_server(filename):
    """
    Menerima file dari server.

    Args:
        filename (str): Nama file yang ingin diminta dari server.
    """
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as client:
        client.connect(ADDR)
        client.send(b"RECEIVE")
        if client.recv(1024).decode().strip() == "OK":
            client.send(filename.encode())
```

```
            response = client.recv(1024).decode()
            if response.startswith("ERROR"):
                print(response)
            else:
                filename, filesize = response.split(SEPARATOR)
                filename = os.path.basename(filename)
                filesize = int(filesize)
                with open(filename, "wb") as f:
                    received = 0
                    while received < filesize:
                        chunk = client.recv(CHUNKSIZE)
                        if not chunk: break
                        f.write(chunk)
                        received += len(chunk)
                print(f"File '{filename}' berhasil diterima.")

def send_file(conn, filename):
    """
    Mengirim file ke koneksi socket tertentu.

    Args:
        conn (socket.socket): Koneksi socket aktif.
        filename (str): Nama file yang akan dikirim.
    """
    filesize = os.path.getsize(filename)
    conn.send(f"{filename}{SEPARATOR}{filesize}".encode())
    with open(filename, "rb") as f:
        while chunk := f.read(CHUNKSIZE):
            conn.sendall(chunk)

if __name__ == "__main__":
    """
    Entry point dari program client. Menanyakan mode kirim/terima dan
    nama file.
    """
    mode = input("Ketik 1 untuk kirim, 2 untuk terima: ").strip()
    filename = input("Masukkan nama file: ").strip()
    if mode == "1":
        if os.path.exists(filename):
            send_file_to_server(filename)
        else:
            print("File tidak ditemukan.")
    elif mode == "2":
        receive_file_from_server(filename)
    else:
        print("Mode tidak valid.")
```