Ichlasul Hasanat

5025201091

Tugas 3 Pemrograman Jaringan A

1. Buatlah program client untuk Tugas 2 dengan mengimplementasikan ketentuan di tugas 2 dengan metode
   a. **Thread**
      - Buat file ‘client_thread.py’.
      - Import library yang dibutuhkan, terutama ‘threading’.

```
from socket import *
import socket
import threading
import logging
import time
from datetime import datetime
import sys
```

      - Buat fungsi ‘kirim_data’ sesuai dengan template yang telah diberikan pada ‘client1.py’ namun ubah port menjadi ‘45000’ sesuai dengan port server dan ubah message yang ingin di kirim.

```
def kirim_data():
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_address = ('localhost', 45000)
    logging.warning(f"membuka socket {server_address}")
    sock.connect(server_address)

    try:
        # Send data
        message = 'TIME\r\n'
        logging.warning(f"[CLIENT] mengirim {message}")
        sock.sendall(message.encode())
        # Look for the response
        amount_received = 0
        amount_expected = len(message)
        while amount_received < amount_expected:
            data = sock.recv(32)
            amount_received += len(data)
            logging.warning(f"[DITERIMA DARI SERVER] {data}")
    finally:
        logging.warning("closing")
        sock.close()
    return
```

      - Buat fungsi untuk membuat thread.

```
def create_thread():
    t = threading.Thread(target=kirim_data)
    t.start()
    t.join()
```

- Buat kode untuk menjalankan fungsi ‘create_thread’ pada fungsi main lalu buat while function untuk menjalankannya dalam loop dengan syarat waktu yang diberikan. (Disini dimasukkan 60 atau sama dengan 1 menit)

```
if __name__=="__main__":
    count = 0
    start = time.time()

    while time.time() - start < 60:
        create_thread()
        count += 1

    start_time = datetime.fromtimestamp(start)
    finish_time = datetime.fromtimestamp(time.time())
    logging.warning(f"Waktu mulai: {start_time}")
    logging.warning(f"Waktu selesai: {finish_time}")
    logging.warning(f"Jumlah threads dibuat: {count}")
```

b. Threadpool

- Buat file ‘threadpool.py’.
- Import library yang dibutuhkan, terutama ‘concurrent.futures.

```
from socket import *
import socket
import logging
import time
from datetime import datetime
import sys
from concurrent.futures import ThreadPoolExecutor
```

- Buat fungsi ‘kirim_data’ sesuai dengan template yang telah diberikan pada ‘client1.py’ namun ubah port menjadi ‘45000’ sesuai dengan port server dan ubah message yang ingin di kirim.

```
def kirim_data():
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_address = ('localhost', 45000)
    logging.warning(f"membuka socket {server_address}")
    sock.connect(server_address)

    try:
        # Send data
        message = 'TIME\r\n'
        logging.warning(f"[CLIENT] mengirim {message}")
        sock.sendall(message.encode())
        # Look for the response
        amount_received = 0
        amount_expected = len(message)
        while amount_received < amount_expected:
            data = sock.recv(32)
            amount_received += len(data)
            logging.warning(f"[DITERIMA DARI SERVER] {data}")
    finally:
        logging.warning("closing")
        sock.close()
    return
```

- Buat kode menggunakan fungsi ‘ThreadPoolExecutor()’. Lalu set variable ‘futures’ dan jalankan ‘executor.submit’ di dalam ‘futures.add()’ dalam while function dengan syarat waktu yang ditentukkan. (Disini dimasukkan 60 atau sama dengan 1 menit)

```
if __name__ == '__main__':
    with ThreadPoolExecutor() as executor:
        start = time.time()
        count = 0
        futures = set()

        while time.time() - start < 60:
            future = executor.submit(kirim_data)
            futures.add(future)
            completed_futures = {f for f in futures if f.done()}
            count += len(completed_futures)
            futures -= completed_futures

        for future in futures:
            future.result()

        start_time = datetime.fromtimestamp(start)
        finish_time = datetime.fromtimestamp(time.time())
        logging.warning(f"Waktu mulai: {start_time}")
        logging.warning(f"Waktu selesai: {finish_time}")
        logging.warning(f"Jumlah request terkirim: {count}")
```

c. Process

- Buat file ‘process.py’.
- Import library yang dibutuhkan, terutama ‘multiprocessing’.

```
from socket import *
import socket
import logging
from datetime import datetime
import time
import sys
from multiprocessing import Process
```

- Buat fungsi ‘kirim_data’ sesuai dengan template yang telah diberikan pada ‘client1.py’ namun ubah port menjadi ‘45000’ sesuai dengan port server dan ubah message yang ingin di kirim.

```
def kirim_data():
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_address = ('localhost', 45000)
    logging.warning(f"membuka socket {server_address}")
    sock.connect(server_address)

    try:
        # Send data
        message = 'TIME\r\n'
        logging.warning(f"[CLIENT] mengirim {message}")
        sock.sendall(message.encode())
        # Look for the response
        amount_received = 0
        amount_expected = len(message)
        while amount_received < amount_expected:
            data = sock.recv(32)
            amount_received += len(data)
            logging.warning(f"[DITERIMA DARI SERVER] {data}")
    finally:
        logging.warning("closing")
        sock.close()
    return
```

- Buat fungsi untuk membuat process.

```
def create_process():
    p = Process(target=kirim_data)
    p.start()
    p.join()
```

- Buat kode untuk menjalankan fungsi ‘create_process’ pada fungsi main lalu buat while function untuk menjalankannya dalam loop dengan syarat waktu yang diberikan. (Disini dimasukkan 60 atau sama dengan 1 menit)

```python
if __name__ == '__main__':
    start = time.time()
    count = 0

    while time.time() - start < 60:
        create_process()
        count += 1

    start_time = datetime.fromtimestamp(start)
    finish_time = datetime.fromtimestamp(time.time())
    logging.warning(f"Waktu mulai: {start_time}")
    logging.warning(f"Waktu selesai: {finish_time}")
    logging.warning(f"Jumlah process dibuat: {count}")
```

2. Jalankan server pada tugas 2, lalu, untuk setiap metode pada nomor 1, jalankan tugas 3 nomor 1 dan amatilah jumlah maksimum thread yang dapat dijalankan di lingkungan eksekusi anda.

    1. **Menjalankan client_thread.py**

    Waktu dijalankan : 1 menit (13:33:03 – 13:34:03)

    Output client
    Jumlah threads dibuat : 26693

```
WARNING:root:[DITERIMA DARI SERVER] b'JAM 13:34:03\r\n'
WARNING:root:closing
WARNING:root:Waktu mulai: 2023-06-08 13:33:03.300072
WARNING:root:Waktu selesai: 2023-06-08 13:34:03.300436
WARNING:root:Jumlah threads dibuat: 26693
```

    Output akhir server
    Jumlah response terkirim : 26693

```
WARNING:root:Jumlah response terkirim: 26692
WARNING:root:connection from ('127.0.0.1', 42198)
WARNING:root:[SERVER] menerima pesan TIME

WARNING:root:[SERVER] mengirim respon JAM 13:34:03

WARNING:root:Jumlah response terkirim: 26693
```

    2. **Menjalankan threadpool.py**

    Waktu dijalankan : 1 menit 16 detik (13:50:54 – 13:52:10)

    Output client
    Jumlah request terkirim: 1193

```
WARNING:root:[DITERIMA DARI SERVER] b'JAM 13:52:10\r\n'
WARNING:root:closing
WARNING:root:Waktu mulai: 2023-06-08 13:50:54.860917
WARNING:root:Waktu selesai: 2023-06-08 13:52:10.141633
WARNING:root:Jumlah request terkirim: 1193
```

Output akhir server
Jumlah response terkirim : 10274

```
WARNING:root:connection from ('127.0.0.1', 36088)
WARNING:root:[SERVER] menerima pesan TIME

WARNING:root:[SERVER] mengirim respon JAM 13:52:10

WARNING:root:Jumlah response terkirim: 10274
```

3. Menjalankan process.py

Waktu dijalankan : 1 menit (13:59:47 – 14:00:47)

Output client
Jumlah request terkirim: 11964

```
WARNING:root:[DITERIMA DARI SERVER] b'JAM 14:00:47\r\n'
WARNING:root:closing
WARNING:root:Waktu mulai: 2023-06-08 13:59:47.657226
WARNING:root:Waktu selesai: 2023-06-08 14:00:47.698352
WARNING:root:Jumlah process dibuat: 11964
```

Output akhir server
Jumlah response terkirim : 11964

```
WARNING:root:connection from ('127.0.0.1', 60988)
WARNING:root:[SERVER] menerima pesan TIME

WARNING:root:[SERVER] mengirim respon JAM 14:00:47

WARNING:root:Jumlah response terkirim: 11964
```

3. Jalankan di lab environment
    a. Tuliskan semua dalam satu file PDF dengan nama TUGAS3.PDF
        i. Link menuju source code anda di github (masing-masing harus punya repository di github)

            Link Repository : https://github.com/Ichlas02/Tugas3_Progjar_A/tree/main

        ii. Jalankan program client (tugas 3 nomor 2) dan server (pada tugas 2), capturelah interaksi antara client dan server. Berikan deskripsi tentang interaksi ini terutama dengan jumlah thread di masing-masing sisi (client / server)

Berdasarkan hasil yang didapat pada poin [2], setelah dijalankan selama kurang lebih 1 menit, dapat dirangkum ke dalam bentuk tabel sebagai berikut:

Metode Client	Request Terkirim Client	Response Terkirim Server	Request Gagal Diterima	Waktu dijalankan
Thread	26693	26693	0	60 detik
Threadpool	1193	10274	9081	76 detik
Process	11964	11964	0	60 detik

Berdasarkan rangkuman pada tabel diatas, dapat diambil beberapa kesimpulan yaitu:

- Thread dan Process mampu menghasilkan response dan request secara sempurna (100%).
- Threadpool hanya mampu mengirim request sekitar 12% dari response yang dikirim dari server.
- Waktu yang dijalankan pada metode threadpool lebih lama 16 detik dari waktu yang ditentukkan.