Nama : Algof Kristian Zega
NRP : 5025231235
Kelas : Pemrograman Jaringan D
Link GitHub : https://github.com/algof/tugas-ets-pemrograman-jaringan

Tugas ETS

1. Dari hasil modifikasi program (https://github.com/rm77/progjar/tree/master/progjar4a) pada TUGAS 3
2. Rubahlah model pemrosesan concurrency yang ada, dari multithreading menjadi
   a. Multihreading menggunakan pool
      Source code

```
import time
import sys
from socket import *
import socket
import threading
import logging
from concurrent.futures import ThreadPoolExecutor
from file_protocol import  FileProtocol
fp = FileProtocol()

def handle_client(connection, address):
    logging.warning(f"Handling client {address}")
    with connection:
        while True:
            data = b''
            while not data.endswith(b'\r\n'):
                part = connection.recv(1)
                if not part:
                    break
                data += part
            if data:
                try:
                    d = data.decode().strip()
                    hasil = fp.proses_string(d)
                    hasil = hasil + "\r\n\r\n"
                    connection.sendall(hasil.encode())
                except Exception as e:
                    logging.error(f"Error while handling client {address}: {e}")
                    break
            else:
                break
    logging.warning(f"Client {address} disconnected")
```

```python
class Server(threading.Thread):
    def __init__(self, ipaddress='0.0.0.0', port=8889, max_workers=10):
        self.ipinfo = (ipaddress, port)
        self.my_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.my_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        self.executor = ThreadPoolExecutor(max_workers=max_workers)
        threading.Thread.__init__(self)

    def run(self):
        logging.warning(f"Server running on {self.ipinfo}")
        self.my_socket.bind(self.ipinfo)
        self.my_socket.listen(5)
        try:
            while True:
                try:
                    conn, addr = self.my_socket.accept()
                    logging.warning(f"Connection from {addr}")
                    self.executor.submit(handle_client, conn, addr)
                except OSError:
                    break
        except KeyboardInterrupt:
            logging.warning("Server stopped by user (KeyboardInterrupt)")
        finally:
            logging.warning("Shutting down thread pool and closing socket...")
            self.my_socket.close()
            self.executor.shutdown(wait=True)

def main():
    svr = Server(ipaddress='0.0.0.0', port=46666, max_workers=5)
    svr.start()
    svr.join()

if __name__ == "__main__":
    main()
```

Modifikasi yang dilakukan tidak terlalu banyak, dari yang sebelumnya menggunakan library import threading diganti menggunakan from concurrent.futures import ThreadPoolExecutor.

Library threading memungkinkan kita untuk membuat thread sebanyak yang kita butuhkan, namun kendalanya adalah overhead tinggi jika terlalu banyak tugas yang dikerjakan, tidak efisien untuk skenario tugas kecil namun sering, dan tidak skalabel dalam jangka panjang.

Library thread pool memungkinkan kita untuk mendefinisikan berapa banyak thread yang ingin kita buat (disebut workers), sehingga jumlah thread tidak akan berubah sepanjang program berjalan. Tugas-tugas dikirim ke “antrian tugas” dan dikerjakan oleh thread yang sedang luang. Thread tidak dimatikan setelah selesai, melainkan digunakan ulangan. Keuntungannya adalah lebih efisien dalam penggunaan resources, skalabel untuk banyak tugas kecil, dan cocok untuk server atau aplikasi paralel yang memproses banyak pertanyaan.

b. Multiprocessing menggunakan pool
Source code

```
from socket import *
import socket
import logging
from multiprocessing import Pool
from file_protocol import FileProtocol
fp = FileProtocol()

def proses_data(d):
    return fp.proses_string(d)

def handle_client(connection, address, pool):
    logging.warning(f"Handling client {address}")
    with connection:
        while True:
            data = b''
            while not data.endswith(b'\r\n'):
                part = connection.recv(1)
                if not part:
                    break
                data += part

            if data:
                try:
                    d = data.decode().strip()
                    result = pool.apply(proses_data, args=(d,))
                    response = result + "\r\n\r\n"
                    connection.sendall(response.encode())
                except Exception as e:
                    logging.error(f"Error handling client {address}: {e}")
                    break
            else:
                break
    logging.warning(f"Client {address} disconnected")

def run_server(ip='0.0.0.0', port=6666, max_workers=4):
```

```
    pool = Pool(processes=max_workers)

    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    server_socket.bind((ip, port))
    server_socket.listen(5)
    logging.warning(f"Server running on {(ip, port)}")

    try:
        while True:
            conn, addr = server_socket.accept()
            logging.warning(f"Connection from {addr}")
            handle_client(conn, addr, pool)
    except KeyboardInterrupt:
        logging.warning("Server stopped by user")
    finally:
        server_socket.close()
        pool.close()
        pool.join()

if __name__ == '__main__':
    run_server()
```

Modifikasi yang dilakukan disini juga tidak terlalu banyak, hanya mengganti konsep multithreading menjadi multiprocessing. Keuntungan menggunakan multiprocessing dibandingkan multithreading adalah performa yang lebih tinggi karena memerlukan alokasi memori dan switching yang lebih mahal, multiprocess juga lebih stabil karena setiap process terisolasi sehingga jika satu process crash, tidak memengaruhi process lain.

Multithreading juga punya keuntungan yaitu lebih ringan karena berbagi sumber daya dan tidak perlu duplikasi memori, lebih cepat dalam segi eksekusi dan komunikasi, namun rentan terhadap crash antar thread.

3. Modifikasilah program client untuk melakukan
   Source code

```
import socket
import json
import base64
import logging

def send_command(command_str=""):
    global server_address
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.connect(server_address)
```

```
    logging.warning(f"connecting to {server_address}")
    try:
        logging.warning(f"sending message ")
        sock.sendall((command_str + "\r\n").encode())
        data_received=""
        while True:
            data = sock.recv(16)
            if data:
                data_received += data.decode()
                if "\r\n\r\n" in data_received:
                    break
            else:
                break
        hasil = json.loads(data_received)
        logging.warning("data received from server:")
        return hasil
    except:
        logging.warning("error during data receiving")
        return False


def remote_list():
    command_str=f"LIST"
    hasil = send_command(command_str)
    if (hasil['status']=='OK'):
        print("daftar file : ")
        for nmfile in hasil['data']:
            print(f"- {nmfile}")
        return True
    else:
        print("Gagal")
        return False

def remote_get(filename=""):
    command_str=f"GET {filename}"
    hasil = send_command(command_str)
    if (hasil['status']=='OK'):
        namafile= hasil['data_namafile']
        isifile = base64.b64decode(hasil['data_file'])
        fp = open(namafile,'wb+')
        fp.write(isifile)
        fp.close()
        print(hasil)
        return True
    else:
```

```
        print("Gagal")
        return False

def remote_delete(filename=""):
    command_str=f"DELETE {filename}"
    hasil = send_command(command_str)
    if(hasil['status'] == 'OK'):
        print(f"file '{filename}' berhasil di delete")
        print(hasil['message'])
    else:
        print("Gagal")

def remote_upload(filename="", isifile=""):
    command_str=f"UPLOAD {filename} {isifile}"
    hasil = send_command(command_str)
    if hasil and hasil.get('status') == 'OK':
        print(hasil['data'])
    else:
        print("Gagal")

def remote_download(filename=""):
    command_str=f"DOWNLOAD {filename}"
    hasil = send_command(command_str)
    if (hasil['status']=='OK'):
        namafile= hasil['data_namafile']
        isifile = base64.b64decode(hasil['data_file'])
        fp = open(namafile,'wb+')
        fp.write(isifile)
        fp.close()
        print(hasil)
        print(f"{namafile} berhasil di download ke direktori lokal anda")
        return True
    else:
        print("Gagal")
        return False

if __name__=='__main__':
    server_address=('172.16.16.101', 46666)
    remote_list()
    # remote_get('hello_2.txt')
    # remote_delete('hello_2.txt')
    # remote_upload("hello_4.txt", "SGVsbG8sIFdvcmxkIQ==")
    remote_download("hello.txt")
```

Modifikasi jika dibandingkan dengan template github ada beberapa penambahan fungsi yaitu upload dan delete yang dikerjakan di tugas 3 dan download yang dikerjakan di tugas ETS. File ini menguji program `file_server.py` apakah dapat bekerja sesuai dengan yang diinginkan atau tidak.

a. Download file

Source code program server

```
def download(self,params=[]):
    try:
        filename = params[0]
        if (filename == ''):
            return None
        fp = open(f"{filename}",'rb')
        isifile = base64.b64encode(fp.read()).decode()
        return dict(status='OK',data_namafile=filename,data_file=isifile)
    except Exception as e:
        return dict(status='ERROR',data=str(e))
```

Source code program client

```
def remote_download(filename=""):
    command_str=f"DOWNLOAD {filename}"
    hasil = send_command(command_str)
    if (hasil['status']=='OK'):
        namafile= hasil['data_namafile']
        isifile = base64.b64decode(hasil['data_file'])
        fp = open(namafile,'wb+')
        fp.write(isifile)
        fp.close()
        print(hasil)
        print(f"{namafile} berhasil di download ke direktori lokal anda")
        return True
    else:
        print("Gagal")
        return False
```

Jadi untuk download disini, dari pihak client akan mengirimkan nama file yang ingin di download dari direktori server. Client bisa mencari tahu list file yang ada pada direktori server dengen command “LIST”.

Jika nama file valid dan file berwujud di direktori server, maka server akan membaca file dan mengirimkan datanya dalam bentuk encoding base64. Pihak client akan menerima data dan mendecoding base64 kembali ke wujud asalnya, seperti .txt, .jpg, dan lainnya.

Di pihak client akan menulis file baru untuk memasukan data, hasil dapat dilihat di direktori tempat client menjalankan program, akan ada file baru yang tercipta atau ter-download.

b. Upload file

Source code server

```
def remote_upload(filename="", isifile=""):
    command_str=f"UPLOAD {filename} {isifile}"
    hasil = send_command(command_str)
    if hasil and hasil.get('status') == 'OK':
        print(hasil['data'])
    else:
        print("Gagal")
```

Source code client

```
def upload(self,params=[]):
    try:
        filename = params[0]
        content = params[1]
        file_bytes = base64.b64decode(content)

        if filename.endswith(".txt"):
            with open('hasil.txt', 'w', encoding='utf-8') as f:
                f.write(file_bytes.decode('utf-8'))
        else:
            with open(filename, 'wb+') as file_pointer:
                file_pointer.write(file_bytes)
        return dict(status='OK',data=f"{filename} has been uploaded")
    except Exception as e:
        return dict(status='ERROR',data=str(e))
```

Jadi, untuk upload disini, pihak client akan mengirimkan nama untuk file yang disimpan dan isi file yang akan disimpan dalam bentuk base64. Setelah itu isi file akan di decoding oleh server untuk ditulis ke dalam file dan disimpan. Lalu akan ada return value untuk memberikan logging terkait hasil kepada client.

c. List file

Source code server

```
def list(self,params=[]):
    try:
        filelist = glob('*.*')
        return dict(status='OK',data=filelist)
    except Exception as e:
        return dict(status='ERROR',data=str(e))
```

Source code client

```
def remote_list():
    command_str=f"LIST"
```

```
        hasil = send_command(command_str)
        if (hasil['status']=='OK'):
            print("daftar file : ")
            for nmfile in hasil['data']:
                print(f"- {nmfile}")
            return True
        else:
            print("Gagal")
            return False
```

Jadi, untuk list disini, client akan mengirimkan command “LIST” dan server akan melakukan list dengan fungsi glob dengan parameter “*.*” artinya selama ada sebuah titik dalam namanya akan disimpan ke list (folder tidak memiliki tanda titik di dalam namanya).

4. Lakukan stress test pada program server tersebut dengan cara membuat client agar melakukan proses pada nomor 3 secara concurrent dengan menggunakan multithreading pool dan multiprocessing pool

   Kombinasi stress test
   - Operasi download, upload
   - Volume file 10 MB, 50 MB, 100 MB
   - Jumlah client worker pool 1, 5, 50
   - Jumlah server worker pool 1, 5, 50

   Untuk setiap kombinasi tersebut catatlah
   A. Waktu total per client melakukan proses upload/download (dalam seconds)
   B. Throughput per client (dalam bytes per second, total bytes yang sukses diproses per second)
   C. Jumlah worker client yang sukses dan gagal (jika sukses semua, maka gagal = 0)
   D. Jumlah worker server yang sukses dan gagal (jika sukses semua, maka gagal = 0)

   Source code:

```
import socket
import json
import base64
import logging
import threading
import concurrent.futures
import time

def send_command(command_str=""):
    global server_address
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.connect(server_address)
    logging.warning(f"connecting to {server_address}")
    try:
        logging.warning(f"sending message ")
```

```
        sock.sendall((command_str + "\r\n").encode())
        data_received=""
        while True:
            data = sock.recv(16)
            if data:
                data_received += data.decode()
                if "\r\n\r\n" in data_received:
                    break
            else:
                break
        hasil = json.loads(data_received)
        logging.warning("data received from server:")
        return hasil
    except:
        logging.warning("error during data receiving")
        return False


def remote_list():
    command_str=f"LIST"
    hasil = send_command(command_str)
    if (hasil['status']=='OK'):
        print("daftar file : ")
        for nmfile in hasil['data']:
            print(f"- {nmfile}")
        return True
    else:
        print("Gagal")
        return False

def remote_get(filename=""):
    command_str=f"GET {filename}"
    hasil = send_command(command_str)
    if (hasil['status']=='OK'):
        namafile= hasil['data_namafile']
        isifile = base64.b64decode(hasil['data_file'])
        fp = open(namafile,'wb+')
        fp.write(isifile)
        fp.close()
        print(hasil)
        return True
    else:
        print("Gagal")
        return False
```

```
def remote_delete(filename=""):
    command_str=f"DELETE {filename}"
    hasil = send_command(command_str)
    if(hasil['status'] == 'OK'):
        print(f"file '{filename}' berhasil di delete")
        print(hasil['message'])
    else:
        print("Gagal")

def remote_upload(filename="", isifile=""):
    command_str=f"UPLOAD {filename} {isifile}"
    hasil = send_command(command_str)
    if hasil and hasil.get('status') == 'OK':
        print(hasil['data'])
    else:
        print("Gagal")

def remote_download(filename="", dest_file=""):
    command_str=f"DOWNLOAD {filename}"
    hasil = send_command(command_str)
    if (hasil['status']=='OK'):
        namafile= hasil['data_namafile']
        isifile = base64.b64decode(hasil['data_file'])
        fp = open(dest_file,'wb+')
        fp.write(isifile)
        fp.close()
        print(hasil)
        print(f"{namafile} berhasil di download ke direktori lokal anda")
        return True
    else:
        print("Gagal")
        return False

def thread_function(index):
    command = "remote_download"
    print(f"[Thread-{index}] Starting {command}")
    start = time.time()
    remote_download("donalbebek.jpg", "donalbebekdownload.jpg")
    # remote_list()
    # isi_file = get_binary_from_file("file_10mb.txt") # works 1 worker
    # isi_file = get_binary_from_file("file_50mb.txt") # works 1 worker
    # isi_file = get_binary_from_file("file_100mb.txt") # works 1 worker
    # remote_upload("stress_test_10mb.txt", isi_file)
    end = time.time()
    print(f"[Thread-{index}] Finished {command}")
```

```
    print(f"Waktu eksekusi: {end - start} detik")

def get_binary_from_file(nama_file=""):
    with open(f'{nama_file}', 'r', encoding='utf-8') as file:
        text_data = file.read()

    text_bytes = text_data.encode('utf-8')

    encoded_bytes = base64.b64encode(text_bytes)

    return encoded_bytes

if __name__ == '__main__':
    server_address = ('172.16.16.101', 46666)
    num_of_workers = 1

    start_time = time.time()

    with concurrent.futures.ThreadPoolExecutor(max_workers=num_of_workers) as
executor:

        futures = [executor.submit(thread_function, i) for i in
range(num_of_workers)]

        for future in concurrent.futures.as_completed(futures):
            future.result()

    end_time = time.time()
    print(f"\nSemua thread selesai. Total waktu eksekusi: {end_time -
start_time:.2f} detik")
```

Terdapat fungsi bernama `thread_function` yang digunakan untuk mendefinisikan tugas yang akan dilakukan oleh setiap thread yang dibuat. Misalnya ingin melakukan stress test terkait upload, jalankan remote upload beserta prasyaratnya, seperti parameter yang perlu dikirim.

Saya juga menambahkan pencatatan waktu dan logging untuk menandakan state suatu thread. Misalnya ingin melakukan stress test terkait download, jalankan remote download, atau apapun fungsi remote yang lain yang ingin dilakukan.

5. Hasil stress test, harus direkap ke sebuah tabel yang barisnya adalah total kombinasi dari nomor 4. Total baris kombinasi = 2 x 3 x 3 x 3 = 81 baris, dengan kolom
   a. Nomor
   b. Operasi
   c. Volume
   d. Jumlah client worker pool

e. Jumlah server worker pool
f. Waktu total per client
g. Throughput per client
h. Jumlah worker client yang sukses dan gagal
i. Jumlah worker server yang sukses dan gagal

Hasil stress test:

Nomor	Operasi	Volume	Jumlah client worker pool	Jumlah server worker pool	Waktu total per client	Ukuran buffer	Throughput per client (dalam MegaByte per second)	Jumlah worker client yang sukses dan gagal	Jumlah worker server yang sukses dan gagal
1	Download	10MB	1	1	0.46 detik	buffer klien 50MB, buffer server 50MB	0.01241927471 MBps	Sukses : 1, Gagal : 0	Sukses : 1, Gagal : 0
2	Download	10MB	1	5	0.44 detik	buffer klien 50MB, buffer server 50MB	0.01228546507 MBps	Sukses : 1, Gagal : 0	Sukses : 5, Gagal : 0
3	Download	10MB	1	50	0.44 detik	buffer klien 50MB, buffer server 50MB	0.0123342584 MBps	Sukses : 1, Gagal : 0	Sukses : 50, Gagal : 0
4	Download	10MB	5	1	1.96 detik	buffer klien 50MB, buffer server 50MB	5.102040816 MBps	Sukses : 5, Gagal : 0	Sukses : 1, Gagal : 0
5	Download	10MB	5	5	1.92 detik	buffer klien 50MB, buffer server 50MB	5.208333333 MBps	Sukses : 5, Gagal : 0	Sukses : 5, Gagal : 0
6	Download	10MB	5	50	1,97 detik	buffer klien 50MB, buffer server 50MB	5.076142132 MBps	Sukses : 5, Gagal : 0	Sukses : 50, Gagal : 0
7	Download	10MB	50	1	19.56 detik	buffer klien 50MB, buffer server 50MB	0.5112474438 MBps	Sukses : 50, Gagal : 0	Sukses : 1, Gagal : 0
8	Download	10MB	50	5	16.56 detik	buffer klien 50MB, buffer server 50MB	0.6038647343 MBps	Sukses : 50, Gagal : 0	Sukses : 5, Gagal : 0
9	Download	10MB	50	50	17.62 detik	buffer klien 50MB, buffer server 50MB	0.5675368899 MBps	Sukses : 50, Gagal : 0	Sukses : 50, Gagal : 0
10	Download	50MB	1	1	7.80 detik	buffer klien 50MB, buffer server 50MB	6.41025641 MBps	Sukses : 1, Gagal : 0	Sukses : 1, Gagal : 0
11	Download	50MB	1	5	7.77 detik	buffer klien 50MB, buffer server 50MB	6.435006435 MBps	Sukses : 1, Gagal : 0	Sukses : 5, Gagal : 0
12	Download	50MB	1	50	7.78 detik	buffer klien 50MB, buffer server 50MB	6.426735219 MBps	Sukses : 1, Gagal : 0	Sukses : 50, Gagal : 0
13	Download	50MB	5	1	38.11 detik	buffer klien 50MB, buffer server 50MB	1.311991603 MBps	Sukses : 5, Gagal : 0	Sukses : 1, Gagal : 0
14	Download	50MB	5	5	38.49 detik	buffer klien 50MB, buffer server 50MB	1.299038711 MBps	Sukses : 5, Gagal : 0	Sukses : 5, Gagal : 0
15	Download	50MB	5	50	37.46 detik	buffer klien 50MB, buffer server 50MB	1.334757074 MBps	Sukses : 5, Gagal : 0	Sukses : 50, Gagal : 0
16	Download	50MB	50	1	393.09 detik	buffer klien 50MB, buffer server 50MB	0.1271973339 MBps	Sukses : 50, Gagal : 0	Sukses : 1, Gagal : 0
17	Download	50MB	50	5	399.34 detik	buffer klien 50MB, buffer server 50MB	0.02504131817 MBps	Sukses : 50, Gagal : 0	Sukses : 5, Gagal : 0
18	Download	50MB	50	50		buffer klien 4MB, buffer server 4MB			
19	Download	100MB	1	1					
20	Download	100MB	1	5					
21	Download	100MB	1	50					
22	Download	100MB	5	1					
23	Download	100MB	5	5					
24	Download	100MB	5	50					
25	Download	100MB	50	1					
26	Download	100MB	50	5					
27	Download	100MB	50	50	impossible untuk device saya	buffer klien 16 byte, buffer server 4MB	impossible untuk device saya	impossible untuk device saya	impossible untuk device saya
28	Upload	10MB	1	1	4.91 detik	buffer klien 16 byte, buffer server 4MB	2.036659878 MBps	Sukses : 1, Gagal : 0	Sukses : 1, Gagal : 0
29	Upload	10MB	1	5	4.87 detik	buffer klien 16 byte, buffer server 4MB	2.05338809 MBps	Sukses : 1, Gagal : 0	Sukses : 5, Gagal : 0
30	Upload	10MB	1	50	4.80 detik	buffer klien 16 byte, buffer server 4MB	2.083333333 MBps	Sukses : 1, Gagal : 0	Sukses : 50, Gagal : 0
31	Upload	10MB	5	1	31.86 detik	buffer klien 16 byte, buffer server 4MB	0.3138731952 MBps	Sukses : 5, Gagal : 0	Sukses : 1, Gagal : 0
32	Upload	10MB	5	5	22.49 detik	buffer klien 16 byte, buffer server 4MB	0.4446420631 MBps	Sukses : 5, Gagal : 0	Sukses : 5, Gagal : 0
33	Upload	10MB	5	50	22.89 detik	buffer klien 16 byte, buffer server 4MB	0.4368719965 MBps	Sukses : 5, Gagal : 0	Sukses : 50, Gagal : 0
34	Upload	10MB	50	1	235.26 detik	buffer klien 16 byte, buffer server 4MB	0.04250616339 MBps	Sukses : 50, Gagal : 0	Sukses : 1, Gagal : 0
35	Upload	10MB	50	5	223.32 detik	buffer klien 16 byte, buffer server 4MB	0.04477879276 MBps	Sukses : 50, Gagal : 0	Sukses : 5, Gagal : 0
36	Upload	10MB	50	50	impossible untuk device saya	buffer klien 16 byte, buffer server 4MB	impossible untuk device saya	Sukses : 0, Gagal : 50	Sukses : 0, Gagal : 50
37	Upload	50MB	1	1	83.47 detik	buffer klien 16 byte, buffer server 4MB	0.59901761 MBps	Sukses : 1, Gagal : 0	Sukses : 1, Gagal : 0
38	Upload	50MB	1	5	138.42 detik	buffer klien 16 byte, buffer server 4MB	0.361219477 MBps	Sukses : 1, Gagal : 0	Sukses : 5, Gagal : 0
39	Upload	50MB	1	50	137.83 detik	buffer klien 16 byte, buffer server 4MB	0.3627657259 MBps	Sukses : 1, Gagal : 0	Sukses : 50, Gagal : 0
40	Upload	50MB	5	1	693.52 detik	buffer klien 16 byte, buffer server 4MB	0.07209597416 MBps	Sukses : 5, Gagal : 0	Sukses : 1, Gagal : 0
41	Upload	50MB	5	5	935.62 detik	buffer klien 16 byte, buffer server 4MB	0.05344049935 MBps	Sukses : 5, Gagal : 0	Sukses : 5, Gagal : 0
42	Upload	50MB	5	50	894.42 detik	buffer klien 16 byte, buffer server 4MB	0.05590214888 MBps	Sukses : 5, Gagal : 0	Sukses : 50, Gagal : 0
43	Upload	50MB	50	1	impossible untuk device saya	buffer klien 16 byte, buffer server 4MB	impossible untuk device saya	Sukses : 0, Gagal : 50	Sukses : 0, Gagal : 1
44	Upload	50MB	50	5	impossible untuk device saya	buffer klien 16 byte, buffer server 4MB	impossible untuk device saya	Sukses : 0, Gagal : 50	Sukses : 0, Gagal : 5
45	Upload	50MB	50	50	impossible untuk device saya	buffer klien 16 byte, buffer server 4MB	impossible untuk device saya	Sukses : 0, Gagal : 50	Sukses : 0, Gagal : 50
46	Upload	100MB	1	1	331.09 detik	buffer klien 16 byte, buffer server 4MB	0.30203267 MBps	Sukses : 1, Gagal : 0	Sukses : 1, Gagal : 0
47	Upload	100MB	1	5	755.29 detik	buffer klien 16 byte, buffer server 4MB	0.1323994757 MBps	Sukses : 1, Gagal : 0	Sukses : 5, Gagal : 0
48	Upload	100MB	1	50	855.31 detik	buffer klien 16 byte, buffer server 4MB	0.1169166735 MBps	Sukses : 1, Gagal : 0	Sukses : 50, Gagal : 0
49	Upload	100MB	5	1	3993.92 detik	buffer klien 16 byte, buffer server 4MB	0.02503805785 MBps	Sukses : 5, Gagal : 0	Sukses : 1, Gagal : 0
50	Upload	100MB	5	5	4578.90 detik	buffer klien 16 byte, buffer server 4MB	0.02183930638 MBps	Sukses : 5, Gagal : 0	Sukses : 5, Gagal : 0
51	Upload	100MB	5	50	4420.91 detik	buffer klien 16 byte, buffer server 4MB	0.02261977738 MBps	Sukses : 5, Gagal : 0	Sukses : 50, Gagal : 0
52	Upload	100MB	50	1	impossible untuk device saya	buffer klien 16 byte, buffer server 4MB	impossible untuk device saya	Sukses : 0, Gagal : 50	Sukses : 0, Gagal : 1
53	Upload	100MB	50	5	impossible untuk device saya	buffer klien 16 byte, buffer server 4MB	impossible untuk device saya	Sukses : 0, Gagal : 50	Sukses : 0, Gagal : 5
54	Upload	100MB	50	50	impossible untuk device saya	buffer klien 16 byte, buffer server 4MB	impossible untuk device saya	Sukses : 0, Gagal : 50	Sukses : 0, Gagal : 50

Mohon maaf, karena saya baru mengerjakan mendekati deadline, saya tidak tahu bahwa stress test membutuhkan waktu yang sangat lama, oleh karena itu saya hanya sempat mencoba beberapa yang cepat yaitu yang ukuran filenya ringan dan jumlah worker client sedikit.

Instruksi submission

- Semua dimasukkan dalam satu file PDF
- Dalam file PDF ini harus berisikan
  - Link menuju ke repository di github
  - Konfigurasi, arsitektur stress test
  - Capture screenshot dari pengerjaan
  - Tabel kombinasi percobaan stress test
  - Semua harus dijelaskan (konfigurasi, arsitektur,capture, tabel dan gambar yang lain)