Nama : Excel Deo Cornelius

NRP : 05111840000117

Soal

Buatlah program yang mengimplementasikan
1. Multi process
2. Multi thread
3. Multi process asynchronous
4. Multi thread asynchronous

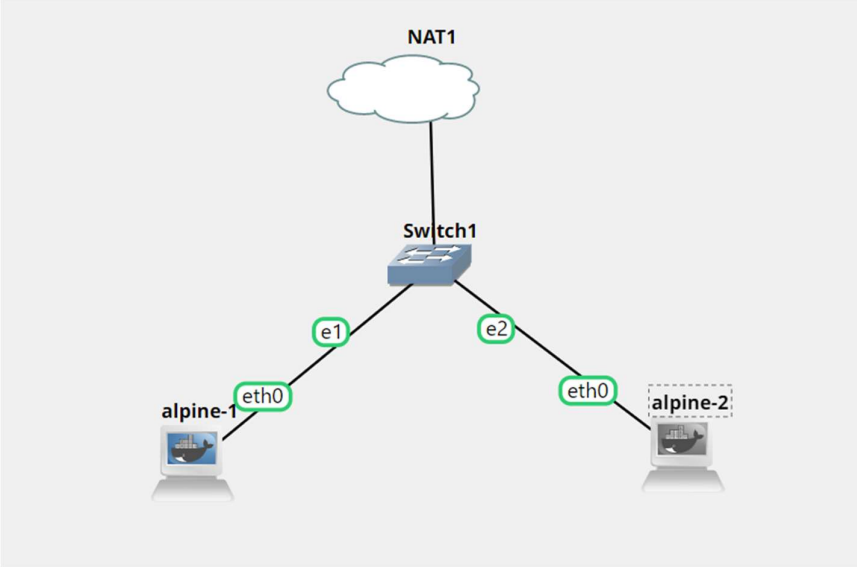
dengan menggunakan protokol transport UDP. Kasus dapat didefinsikan sendiri dan buatlah arsitektur jaringan anda sendiri di simulator GNS3.
Buatlah laporan dalam bentuk PDF yang berisikan screenshot dari
1. Deskripsi kasus yang dibuat
2. Gambar arsitektur jaringan (dalam simulator GNS3)
3. Program yang dibuat (1-4)
4. Hasil outputnya

Jawaban

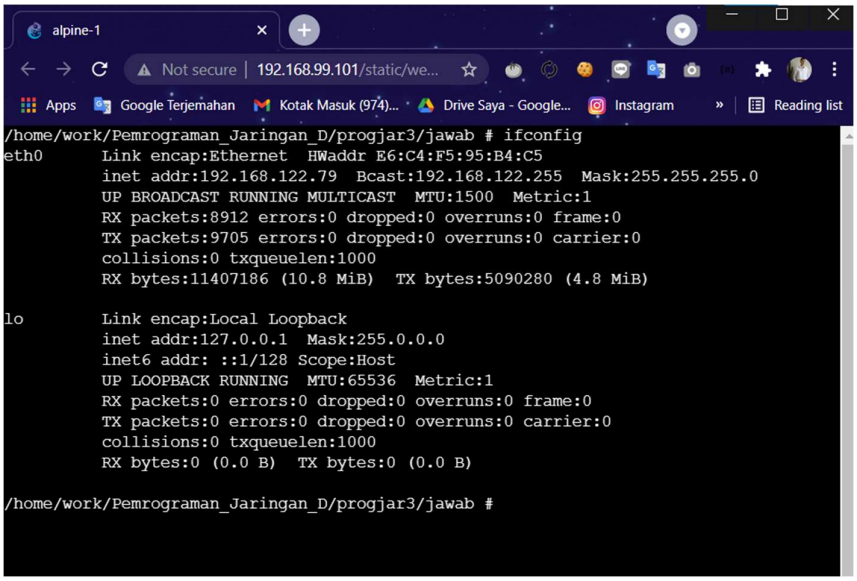
1. Kasus

Terdapat 1 server dan 1 client atau lebih. Client akan meminta request untuk mengirimkan file gambar secara concurrency antara lain : multi process, multi thread, multi process asynchronous, dan multi thread asynchronous. kemudian server akan mengirimkan request tersebut dan mengirimkan ke seluruh client.

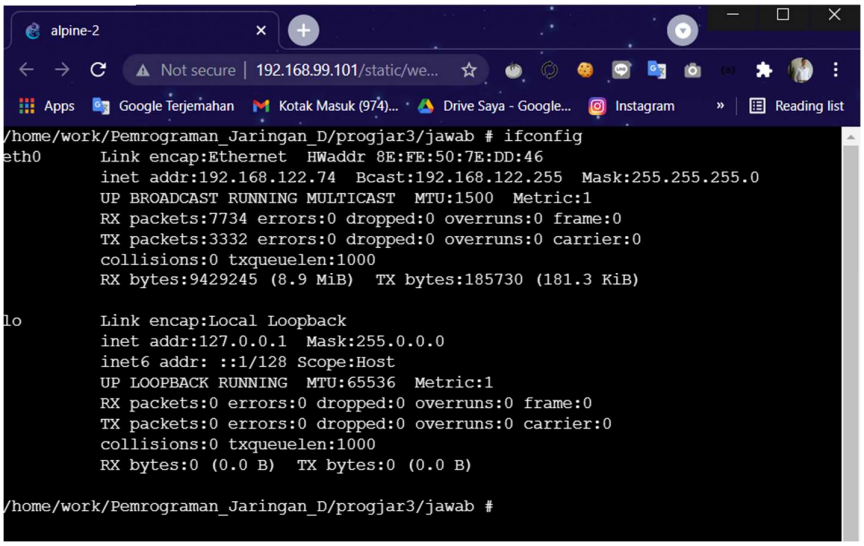
2. Topologi



Konfigurasi
alpine-1: client



alpine-2: server



3. Program yang dibuat

dalam program yang saya buat saya menggabungkan semua concurrency dalam 1 file client.py

New Paste

```
from library import download_gambar, get_url_list
import time
import socket
import logging
import datetime
import threading
import concurrent.futures
from multiprocessing import Process, Pool
import os
import sys

#target IP kirim_sync
TARGET_IP = "192.168.122.255" #Bcast = Broadcast Address
TARGET_PORT = 5005

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.setsockopt(socket.SOL_SOCKET,socket.SO_REUSEPORT, 1)
sock.setsockopt(socket.SOL_SOCKET,socket.SO_BROADCAST, 1)

def single_thread():
    urls = get_url_list()

    catat = datetime.datetime.now()
    for k in urls:
        print(f"mendownload {urls[k]}")
        waktu_proses = download_gambar(urls[k])
        print(f"completed {waktu_proses} detik")
    selesai = datetime.datetime.now() - catat
    print(f"Waktu TOTAL yang dibutuhkan {selesai} detik")

def kirim_multi_process_sync(daftar=None):
    if (daftar is None):
        return False
    f = open(daftar,"rb")
    l = f.read(1024)
    while (1):
        if(sock.sendto(l, (TARGET_IP, TARGET_PORT))):
                l = f.read(1024)
    f.close()

def multi_process_sync():
    texec = dict()
    daftar = ['testing1.png', 'testing2.jpeg']

    catat_awal = datetime.datetime.now()
    for k in range(len(daftar)):
        print(f"mengirim {daftar[k]}")
        texec[k] = Process(target=kirim_multi_process_sync, args=(daftar[k],))
        texec[k].start()
    for k in range(len(daftar)):
        texec[k].join()

    catat_akhir = datetime.datetime.now()
    selesai = catat_akhir - catat_awal
    print(f"Waktu TOTAL yang dibutuhkan {selesai} detik {catat_awal} s/d {catat_akhir}")


def kirim_multi_process_async(daftar=None):
    if (daftar is None):
        return False
    f = open(daftar,"rb")
    l = f.read(1024)
    while (1):
        if(sock.sendto(l, (TARGET_IP, TARGET_PORT))):
                l = f.read(1024)
    f.close()

def multi_process_async():
    texec = dict()
    daftar = ['testing1.png', 'testing2.jpeg']
    status_task = dict()
    task_pool = Pool(processes=20)
    catat_awal = datetime.datetime.now()
    for k in range(len(daftar)):
        print(f"mengirim {daftar[k]}")
        texec[k] = task_pool.apply_async(func=kirim_multi_process_async, args=(daftar[k],))
    for k in range(len(daftar)):
        status_task[k]=texec[k].get(timeout=10)

    catat_akhir = datetime.datetime.now()
    selesai = catat_akhir - catat_awal
    print(f"Waktu TOTAL yang dibutuhkan {selesai} detik {catat_awal} s/d {catat_akhir}")
    print("status TASK")
    print(status_task)

def kirim_multi_thread_sync(daftar=None):
    if (daftar is None):
        return False
    f = open(daftar,"rb")
    l = f.read(1024)
    while (1):
        if(sock.sendto(l, (TARGET_IP, TARGET_PORT))):
                l = f.read(1024)
    f.close()

def multi_thread_sync():
    texec = dict()
    daftar = ['testing1.png', 'testing2.jpeg']

    catat_awal = datetime.datetime.now()
    for k in range(len(daftar)):
        print(f"mengirim {daftar[k]}")
        texec[k] = threading.Thread(target=kirim_multi_thread_sync, args=(daftar[k],))
        texec[k].start()
    for k in range(len(daftar)):
        texec[k].join()

    catat_akhir = datetime.datetime.now()
    selesai = catat_akhir - catat_awal
    print(f"Waktu TOTAL yang dibutuhkan {selesai} detik {catat_awal} s/d {catat_akhir}")
```

```
def kirim_multi_thread_async(daftar=None):
    if (daftar is None):
        return False
    f = open(daftar,"rb")
    l = f.read(1024)
    while (l):
        if(sock.sendto(l, (TARGET_IP, TARGET_PORT))):
                l = f.read(1024)
    f.close()

def multi_thread_async():
    texec = dict()
    daftar = ['testing1.png', 'testing2.jpeg']
    status_task = dict()
    task = concurrent.futures.ThreadPoolExecutor(max_workers=4)

    catat_awal = datetime.datetime.now()
    for k in range(len(daftar)):
        print(f"mendownload {daftar[k]}")
        waktu = time.time()
        texec[k] = task.submit(kirim_multi_thread_async, daftar[k])
    for k in range(len(daftar)):
        status_task[k]=texec[k].result()

    catat_akhir = datetime.datetime.now()
    selesai = catat_akhir - catat_awal
    print(f"Waktu TOTAL yang dibutuhkan {selesai} detik {catat_awal} s/d {catat_akhir}")
    print("hasil task yang dijalankan")
    print(status_task)


def menu():
   'Menu pada user'
   while True:
      time.sleep(0.5)
      print("\n=========== Masukkan Perintah ===========")
      print("Ketik '1' untuk melakukan Single Thread")
      print("Ketik '2' untuk melakukan Multi Process Sync")
      print("Ketik '3' untuk melakukan Multi Process Async")
      print("Ketik '4' untuk melakukan Multi thread Sync")
      print("Ketik '5' untuk melakukan Multi thread Async")
      print("Ketik 'keluar' untuk menutup aplikasi\n")
      time.sleep(0.5)
      command = input("Perintah > ")
      if command == "1":
         single_thread()
      elif command == "2":
         multi_process_sync()
      elif command == "3":
         multi_process_async()
      elif command == "4":
         multi_thread_async()
      elif command == "5":
         multi_thread_sync()
      elif command == "keluar":
         closeApp()
      else:
         print("Perintah > Tidak ada perintah")
         continue

def closeApp():
   "Menutup aplikasi"
   time.sleep(0.5)
   os.system('cls' if os.name == 'nt' else 'clear')
   print("Aplikasi akan ditutup. Menutup koneksi dengan server.")
   clientSocket.close() # Menutup socket
   sys.exit() # Keluar menuju ke sistem

# Menampilkan menu
menu()
```

Berikut adalah file library.py yang di gunakan untuk single thread

```
import logging
import requests
import os
import time
import datetime

def get_url_list():
    urls = dict()
    urls['suzy']='https://asset-a.grid.id/crop/0x0:0x0/945x630/photo/2021/05/04/cover-foto-bae-suzy-cr-instagra-20210504065517.jpg'
    urls['naruto']='http://2.bp.blogspot.com/-BBr9_JumuTY/UMwx0J5OQwI/AAAAAAAAAdI/Yo4UY2MiQkE/s1600/animasi-naruto-gif.gif'
    urls['jihyo']='https://cdn.idntimes.com/content-images/community/2019/12/c015051e080acf9b7db402ccfa97cc011-
d9be772ef78e358209975d786cd3c417_600x400.jpg'
    urls['video1']='https://filesamples.com/samples/video/mov/sample_960x540.mov'
    urls['video2']='https://filesamples.com/samples/video/mov/sample_640x360.mov'
    return urls

def download_gambar(url=None,tuliskefile=False):
    waktu_awal = datetime.datetime.now()
    if (url is None):
        return False
    ff = requests.get(url)
    tipe = dict()
    tipe['image/png']='png'
    tipe['image/jpg']='jpg'
    tipe['image/gif']='gif'
    tipe['image/jpeg']='jpg'
    tipe['application/zip']='jpg'
    tipe['video/quicktime']='mov'
    time.sleep(2) #untuk simulasi, diberi tambahan delay 2 detik

    content_type = ff.headers['Content-Type']
    logging.warning(content_type)
    if (content_type in list(tipe.keys())):
        namafile = os.path.basename(url)
        ekstensi = tipe[content_type]
        if (tuliskefile):
            fp = open(f"{namafile}.{ekstensi}","wb")
            fp.write(ff.content)
            fp.close()
        waktu_process = datetime.datetime.now() - waktu_awal
        waktu_akhir =datetime.datetime.now()
        logging.warning(f"writing {namafile}.{ekstensi} dalam waktu {waktu_process} {waktu_awal} s/d {waktu_akhir}")
        return waktu_process
    else:
        return False
```

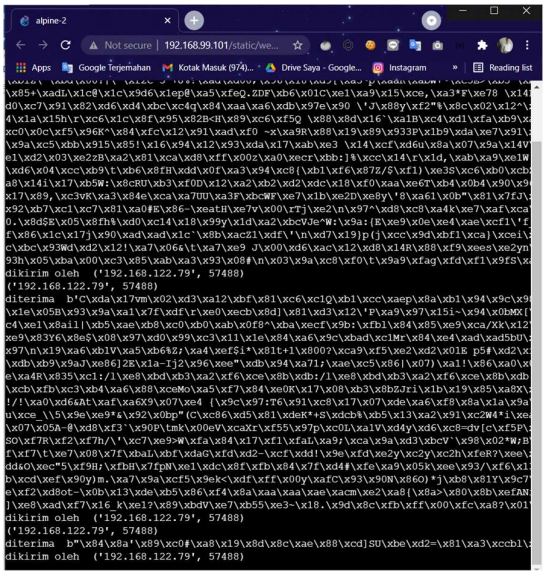
Berikut adalah codingan untuk server.py

```
import socket

SERVER_IP = '192.168.122.255'
SERVER_PORT = 5005

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.bind(("", SERVER_PORT))

while True:
    data, addr = sock.recvfrom(1024)        You, an hour ago • memperbaiki
    print(addr)
    print("diterima ", data)
    print("dikirim oleh ", addr)
```

4. Hasil output

1. Multi process Sync

- server

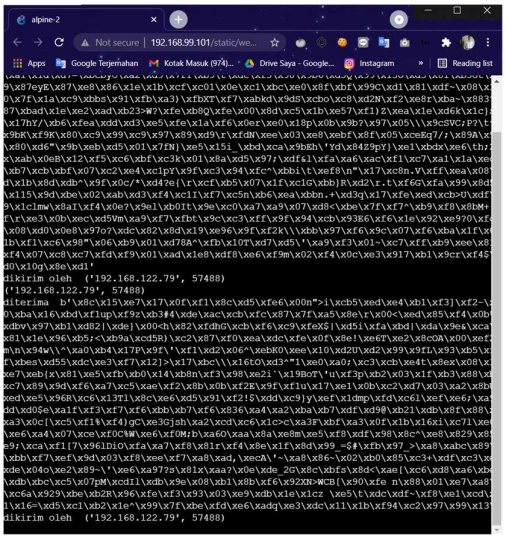


- client

```
============ Masukkan Perintah ============
Ketik '1' untuk melakukan Single Thread
Ketik '2' untuk melakukan Multi Process Sync
Ketik '3' untuk melakukan Multi Process Async
Ketik '4' untuk melakukan Multi thread Sync
Ketik '5' untuk melakukan Multi thread Async
Ketik 'keluar' untuk menutup aplikasi

Perintah > 2
mengirim testing1.png
mengirim testing2.jpeg
Waktu TOTAL yang dibutuhkan 0:00:00.020658 detik 2021-07-01 18:23:14.526510 s/d 2021-
```

2. Multi process asynchronous

- server

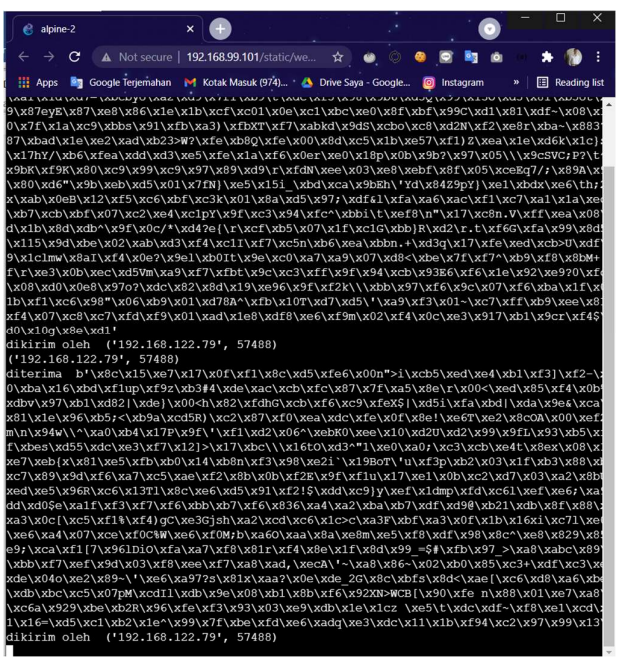


- client

```
=========== Masukkan Perintah ===========
Ketik '1' untuk melakukan Single Thread
Ketik '2' untuk melakukan Multi Process Sync
Ketik '3' untuk melakukan Multi Process Async
Ketik '4' untuk melakukan Multi thread Sync
Ketik '5' untuk melakukan Multi thread Async
Ketik 'keluar' untuk menutup aplikasi

Perintah > 3
mengirim testing1.png
mengirim testing2.jpeg
Waktu TOTAL yang dibutuhkan 0:00:00.037572 detik 2021-07-01 18:24:08.750630 s/d 2021-
status TASK
{0: None, 1: None}
```

3. Multi thread Sync

- server

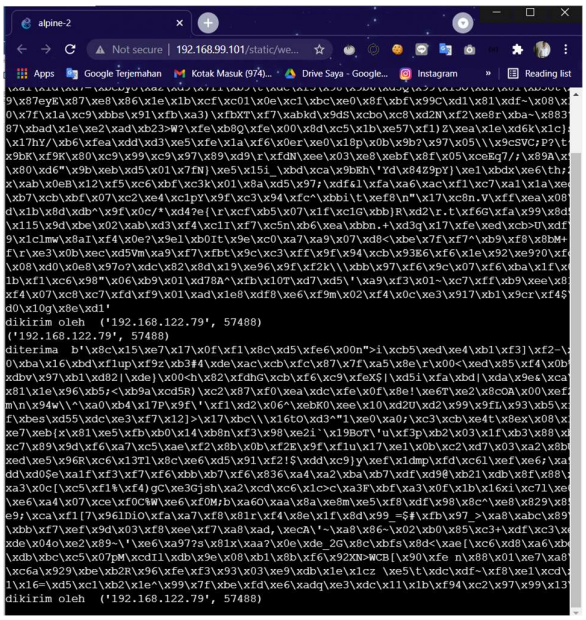


- client

```
=========== Masukkan Perintah ===========
Ketik '1' untuk melakukan Single Thread
Ketik '2' untuk melakukan Multi Process Sync
Ketik '3' untuk melakukan Multi Process Async
Ketik '4' untuk melakukan Multi thread Sync
Ketik '5' untuk melakukan Multi thread Async
Ketik 'keluar' untuk menutup aplikasi

Perintah > 4
mendownload testing1.png
mendownload testing2.jpeg
Waktu TOTAL yang dibutuhkan 0:00:00.006667 detik 2021-07-01 18:24:57.449679 s/d 2021-
hasil task yang dijalankan
{0: None, 1: None}
```

4. Multi thread asynchronous

- server



- client

```
=========== Masukkan Perintah ===========
Ketik '1' untuk melakukan Single Thread
Ketik '2' untuk melakukan Multi Process Sync
Ketik '3' untuk melakukan Multi Process Async
Ketik '4' untuk melakukan Multi thread Sync
Ketik '5' untuk melakukan Multi thread Async
Ketik 'keluar' untuk menutup aplikasi

Perintah > 5
mengirim testing1.png
mengirim testing2.jpeg
Waktu TOTAL yang dibutuhkan 0:00:00.006264 detik 2021-07-01 18:25:35.052459 s/d 2021-
```