**Progjar Implementasi Kasus menggunakan concurrency**

Nama : Rimas Muhammad Irfansyah

NRP : 05111740000185

# 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)

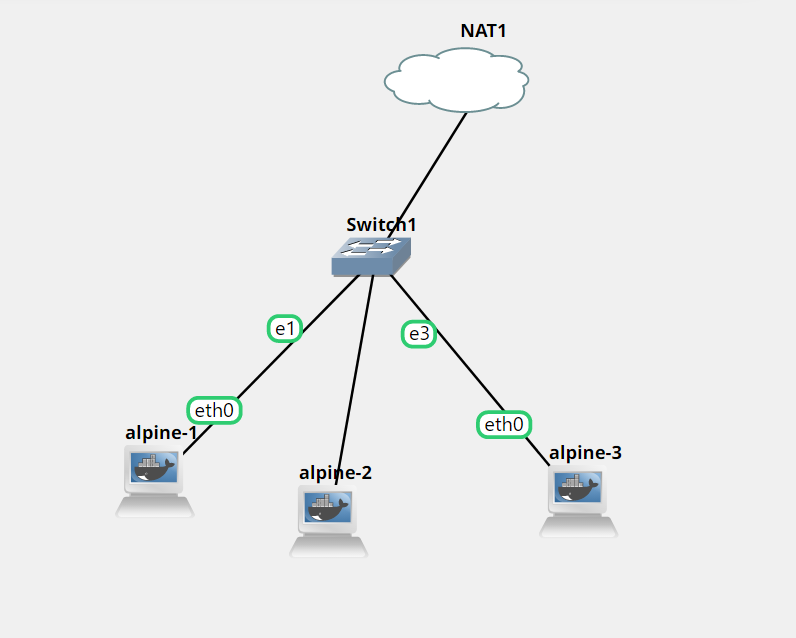
4. Hasil outputnya

# Jawaban:

## Kasus

Dari arsitektur akan terdapat 2 server dan 1 client. Client akan melakukan pengiriman 2 buah file gambar ke 2 server berbeda dengan menggunakan transport UDP. Pengiriman file gambar dari client ke 2 server tersebut dilakukan dengan 4 cara yang berbeda yaitu multi process, multi thread, multi process asynchronous, dan multi thread asynchronous.

## Topologi



Alpine-1 dan alpine-2 sebagai server.

Alpine-3 sebagai client.

## Program yang dibuat

Multi\_process.py

# from library import download\_gambar, get\_url\_list  
import time  
import datetime  
import socket  
from multiprocessing import Process  
  
  
TARGET\_IP = "0.0.0.0"  
TARGET\_PORT = 1000  
  
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 kirim\_gambar(image=None):  
 if (image is None):  
 return False  
  
 #buka gambar dalam format binary  
 f = open(image, "rb")  
 l = f.read(1024)  
 while (l):  
 if(sock.sendto(l, (TARGET\_IP, TARGET\_PORT))):  
 l = f.read(1024)  
 f.close()  
  
  
def kirim\_semua():  
 texec = dict()  
 images = ['1.png', '2.png']  
 catat\_awal = datetime.datetime.now()  
 for k in range(len(images)):  
 print(f"mengirim {images[k]}")  
 waktu = time.time()  
  
 #bagian ini merupakan bagian yang mengistruksikan eksekusi fungsi kirim gambar secara multiprocess  
 texec[k] = Process(target=kirim\_gambar, args=(images[k],))  
 texec[k].start()  
  
 #setelah menyelesaikan tugasnya, dikembalikan ke main process dengan join  
 for k in range(len(images)):  
 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}")  
  
  
#fungsi kirim\_gambar akan dijalankan secara multi process  
if \_\_name\_\_=='\_\_main\_\_':  
 kirim\_semua()

Multi\_process\_async.py

# from library import download\_gambar, get\_url\_list  
import time  
import datetime  
import socket  
from multiprocessing import Process, Pool  
  
  
TARGET\_IP = "0.0.0.0"  
TARGET\_PORT = 1000  
  
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 kirim\_gambar(image=None):  
 if (image is None):  
 return False  
  
 #buka gambar dalam format binary  
 f = open(image, "rb")  
 l = f.read(1024)  
 while (l):  
 if(sock.sendto(l, (TARGET\_IP, TARGET\_PORT))):  
 l = f.read(1024)  
 f.close()  
  
  
def kirim\_semua():  
 texec = dict()  
 images = ['1.png', '2.png']  
 status\_task = dict()  
 task\_pool = Pool(processes=20) #2 task yang dapat dikerjakan secara simultan, dapat diset sesuai jumlah core  
 catat\_awal = datetime.datetime.now()  
 for k in range(len(images)):  
 print(f"mengirim {images[k]}")  
  
 #bagian ini merupakan bagian yang mengistruksikan eksekusi fungsi kirim gambar secara multiprocess  
 texec[k] = task\_pool.apply\_async(func=kirim\_gambar, args=(images[k],))  
  
 #setelah menyelesaikan tugasnya, dikembalikan ke main process dengan mengambil hasilnya dengan get  
 for k in range(len(images)):  
 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)  
  
  
#fungsi kirim\_gambar akan dijalankan secara multi process  
if \_\_name\_\_=='\_\_main\_\_':  
 kirim\_semua()

Multi\_thread.py

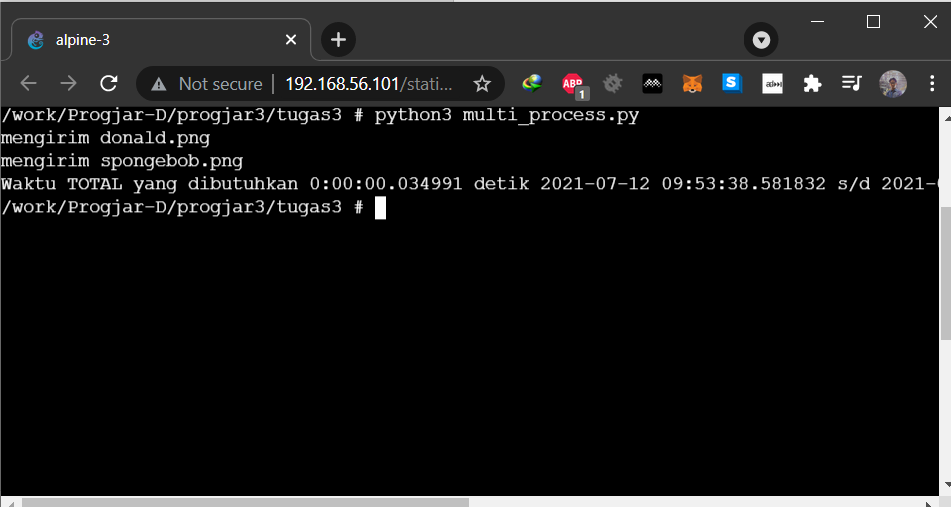
# from library import download\_gambar,get\_url\_list  
import time  
import datetime  
import threading  
import socket  
  
TARGET\_IP = "0.0.0.0"  
TARGET\_PORT = 1000  
  
sock = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)  
sock.setsockopt(socket.SOL\_SOCKET, socket.SO\_REUSEPORT, 1)  
sock.setsockopt(socket.SOL\_SOCKET, socket.SO\_BROADCAST, 1)  
  
  
def kirim\_gambar(image=None):  
 if (image is None):  
 return False  
  
 # buka gambar dalam format binary  
 f = open(image, "rb")  
 l = f.read(1024)  
 while (l):  
 if (sock.sendto(l, (TARGET\_IP, TARGET\_PORT))):  
 l = f.read(1024)  
 f.close()  
  
  
def kirim\_semua():  
 texec = dict()  
 images = ['1.png', '2.png']  
  
 catat\_awal = datetime.datetime.now()  
 for k in range(len(images)):  
 print(f"mengirim {images[k]}")  
 waktu = time.time()  
  
 # bagian ini merupakan bagian yang mengistruksikan eksekusi fungsi kirim gambar secara multithread  
 texec[k] = threading.Thread(target=kirim\_gambar, args=(images[k],))  
 texec[k].start()  
  
 # setelah menyelesaikan tugasnya, dikembalikan ke main thread dengan join  
 for k in range(len(images)):  
 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}")  
  
  
# fungsi kirim\_gambar akan dijalankan secara multithreading  
if \_\_name\_\_ == '\_\_main\_\_':  
 kirim\_semua()

Multi\_thread\_async.py

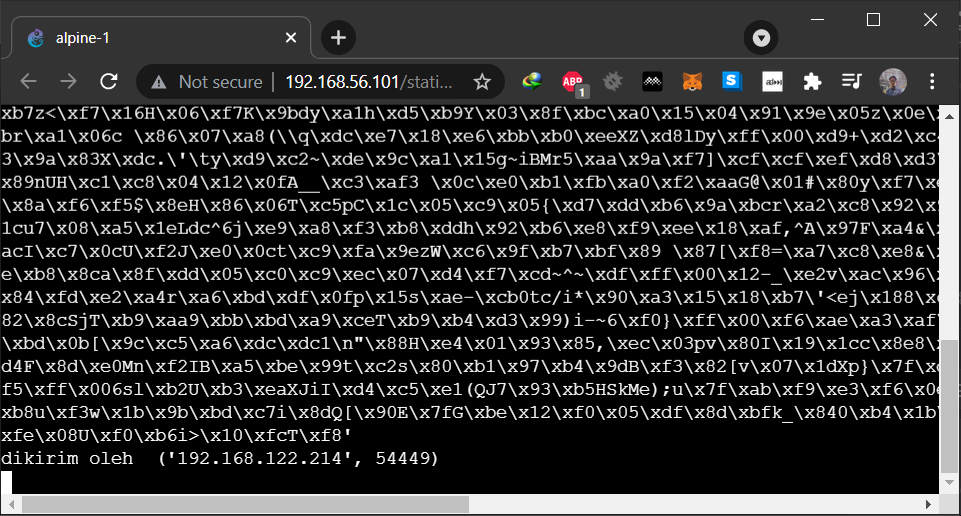
# from library import download\_gambar,get\_url\_list  
import time  
import datetime  
import concurrent.futures  
import socket  
  
  
TARGET\_IP = "0.0.0.0"  
TARGET\_PORT = 1000  
  
sock = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)  
sock.setsockopt(socket.SOL\_SOCKET, socket.SO\_REUSEPORT, 1)  
sock.setsockopt(socket.SOL\_SOCKET, socket.SO\_BROADCAST, 1)  
  
  
def kirim\_gambar(image=None):  
 if (image is None):  
 return False  
  
 #buka gambar dalam format binary  
 f = open(image, "rb")  
 l = f.read(1024)  
 while (l):  
 if(sock.sendto(l, (TARGET\_IP, TARGET\_PORT))):  
 l = f.read(1024)  
 f.close()  
  
  
def kirim\_semua():  
 texec = dict()  
 images = ['1.png', '2.png']  
 status\_task = dict()  
 task = concurrent.futures.ThreadPoolExecutor(max\_workers=4)  
 catat\_awal = datetime.datetime.now()  
 for k in range(len(images)):  
 print(f"mengirim {images[k]}")  
 waktu = time.time()  
  
 #bagian ini merupakan bagian yang mengistruksikan eksekusi fungsi kirim gambar secara multithread  
 texec[k] = task.submit(kirim\_gambar, images[k])  
  
 #setelah menyelesaikan tugasnya, dikembalikan ke main thread dengan memanggil result  
 for k in range(len(images)):  
 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)  
  
  
#fungsi kirim\_gambar akan dijalankan secara multithreading  
if \_\_name\_\_=='\_\_main\_\_':  
 kirim\_semua()

## Hasil Output

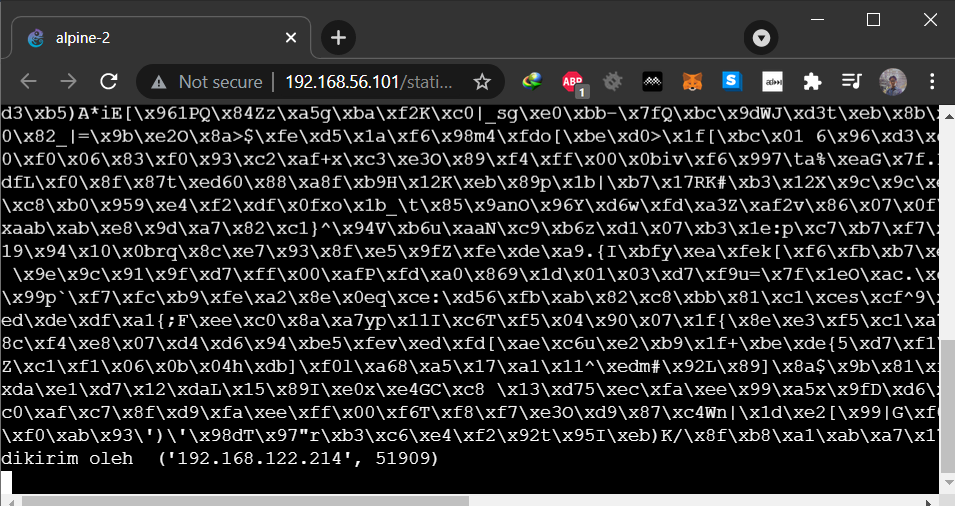
Client Multi Process



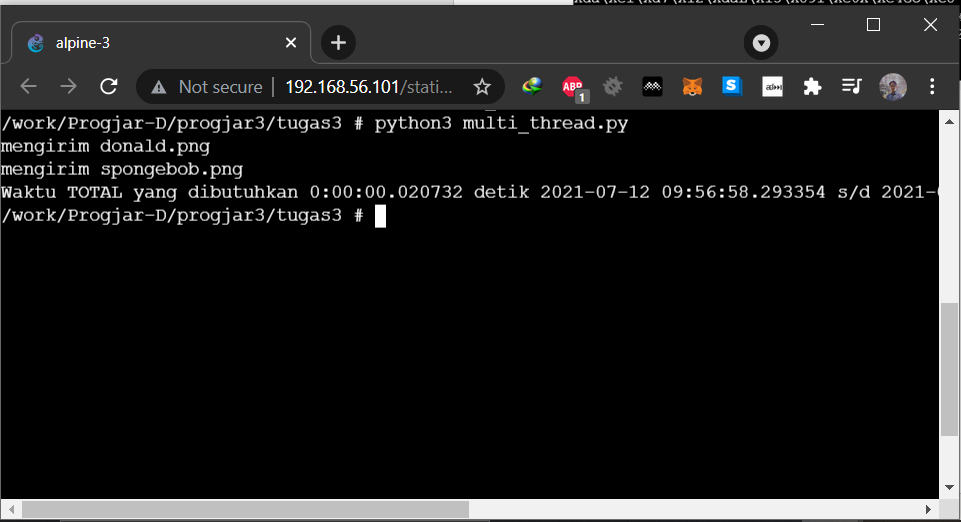
Server 1 Multi Process



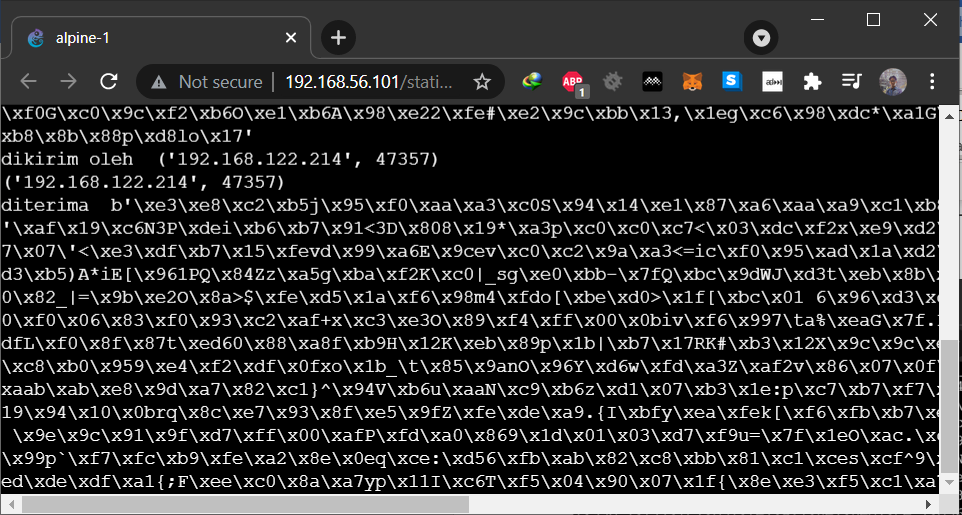
Server 2 Multi Process



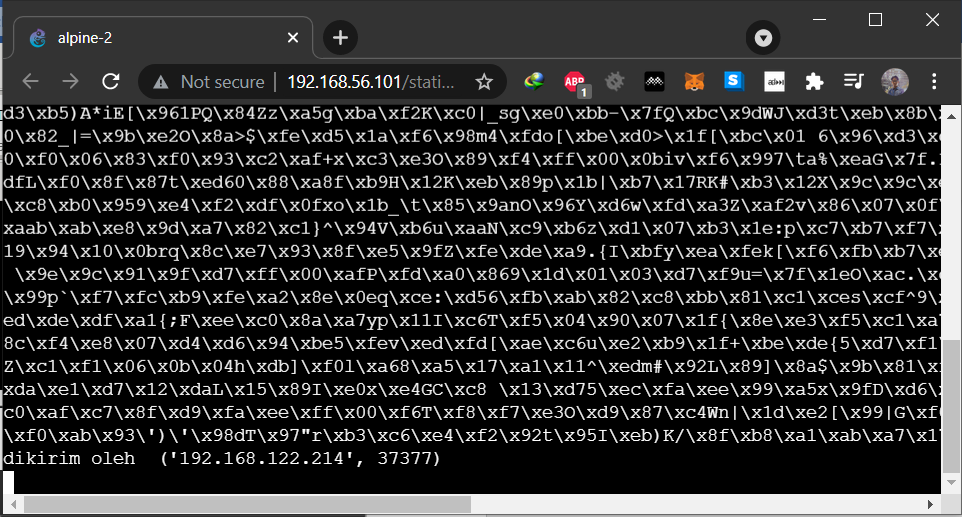
Client Multi Thread



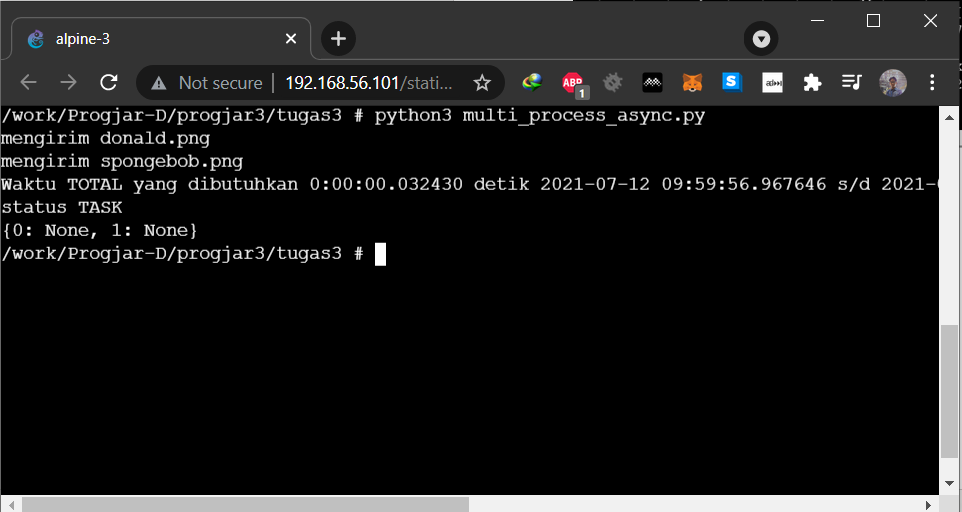
Server 1 Multi Thread



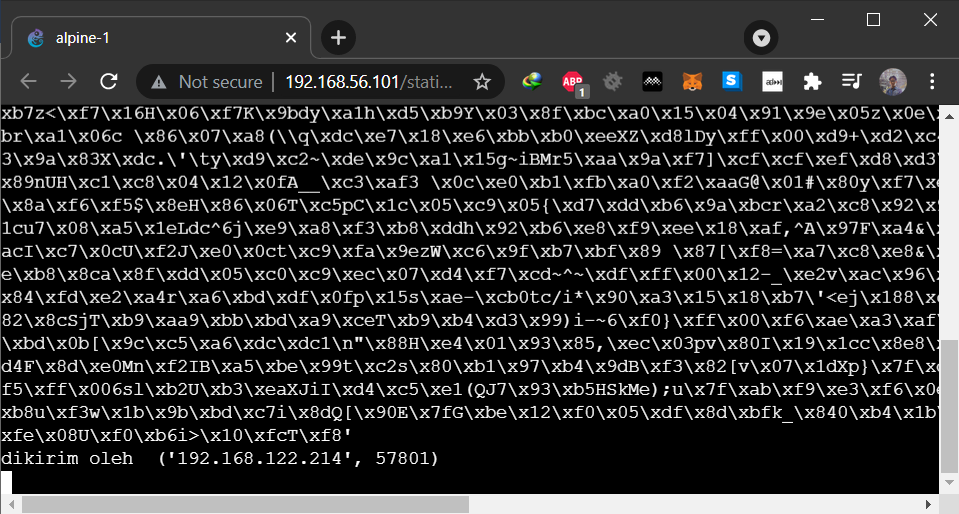
Server 2 Multi Thread



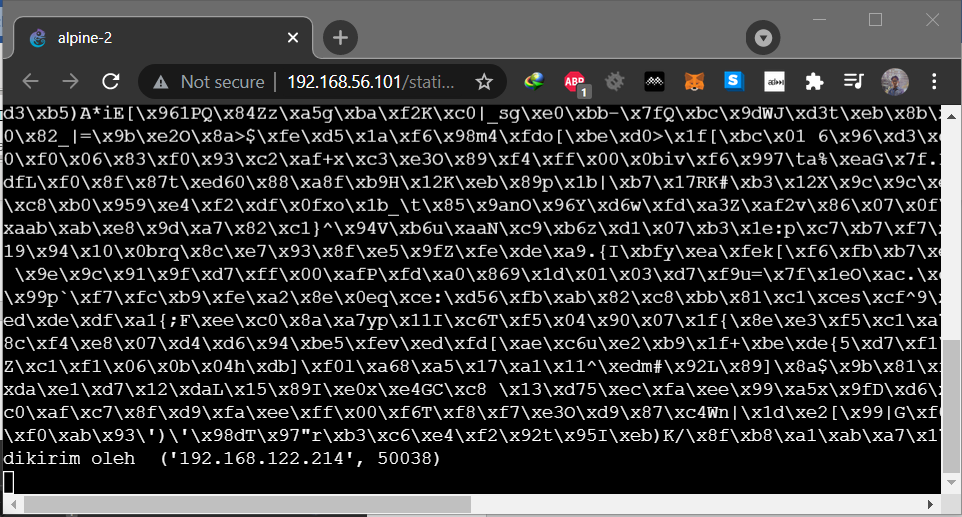
Client Multi-process Async



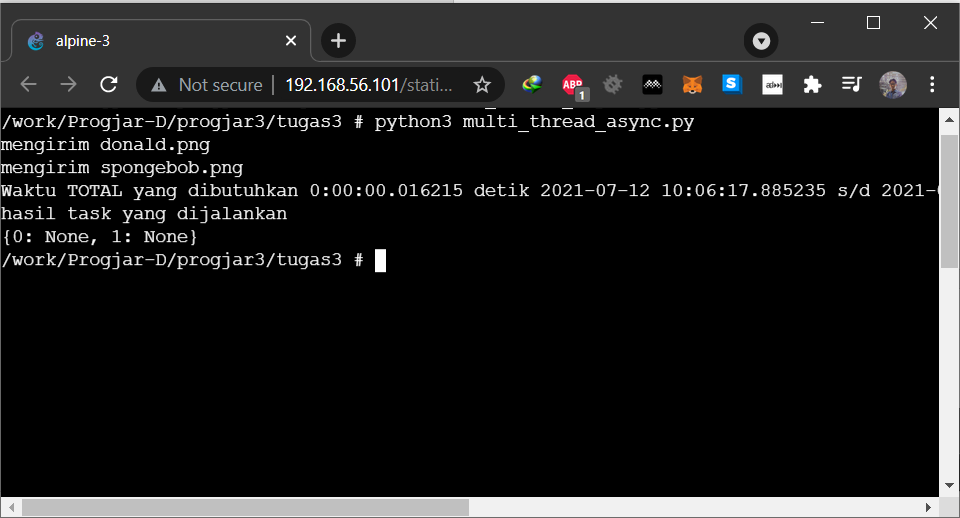
Server 1 Multi-process Async



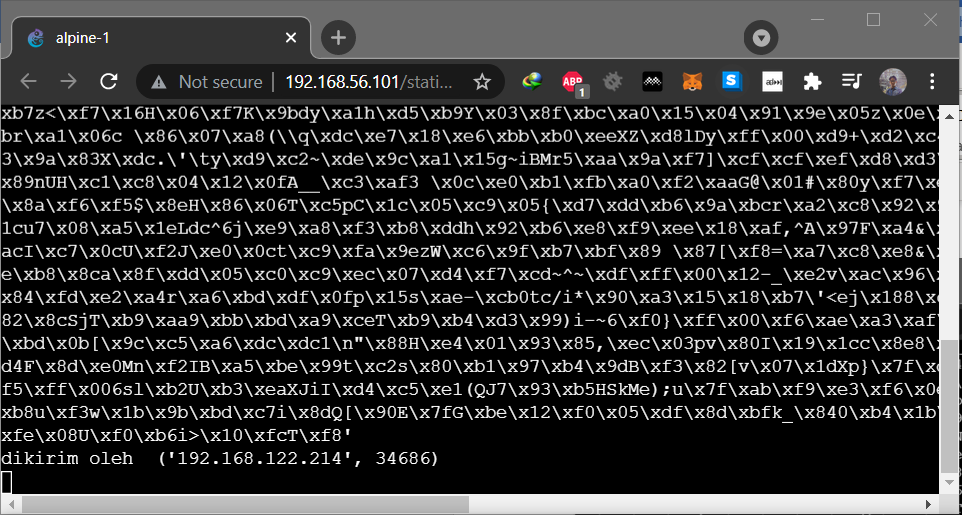
Server 2 Multi-process Async



Client Multi-thread Async



Server 1 Multi-thread Async



Server 2 Multi-thread Async

