# PEMROGRAMAN JARINGAN E

# TUGAS IMPLEMENTASI KASUS MENGGUNAKAN CONCURENCY

Fandi Wahyu Rusydi
05111840000108

Dosen Pengampu:
Royyana Muslim Ijtihadie, S.Kom.,M.Kom., Ph.D.

Departemen Teknik Informatika
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember (ITS)
Surabaya
2021

Praktikum

1. Buatlah program yang mengimplementasikan
   a. multi process
   b. multi thread
   c. multi process asynchronous
   d. multi thread asynchronous

   dengan menggunakan protokol transport UDP. kasus dapat didefinsikan sendiri.dan Buatlah arsitektur jaringan anda sendiri di simulator GNS3

2. buatlah laporan dalam bentuk PDF yang berisikan screenshot dari
   a. deskripsi kasus yang dibuat
   b. gambar arsitektur jaringan (dalam simulator GNS3)
   c. program yang dibuat (1-4)
   d. hasil outputnya

Studi Kasus :

Terdapat 1 client yaitu alpine-1 dan 2 server yaitu alpine-2 dan alpine-3, nantinya client akan men-download satu gambar lalu akan mem-broadcast atau mengirim file image tersebut ke kedua server dengan protokol transport UDP. Pengiriman atau broadcast dari client ke 2 server akan dilakukan dengan 4 berbeda yaitu multi process, multi thread, multi process asynchronous dan multi thread asynchronous sesuai perintah pada soal.

Github : https://github.com/Asmophel/Pemrograman_Jaringan_E/tree/Tugas-3-Implementasi-Kasus-Concurrency/progjar3/jawaban

Program yang digunakan :

1. **library.py**

64 lines (54 sloc) | 2.32 KB — Raw | Blame

```python
import logging
import socket
import requests
import os
import time
import datetime


def get_url_list():
    urls = dict()
    # urls['kompas']='https://asset.kompas.com/crops/qz_j7xyaZgGgboomdCEXsfbSpoc=/0x0:998x665/740x500/data/photo/2020/03/01/5e5b52f4db896.jpg'
    urls['its']='https://www.its.ac.id/wp-content/uploads/2017/10/logo-its-1.png'
    #urls['detik']='https://akcdn.detik.net.id/community/media/visual/2021/04/22/detikcom-ramadan-desktop-1.gif?d=1'
    # urls['file1']='https://file-examples-com.github.io/uploads/2018/04/file_example_MOV_480_700kB.mov'
    #urls['file2']='https://file-examples-com.github.io/uploads/2018/04/file_example_MOV_1280_1_4MB.mov'
    #urls['file3']='https://file-examples-com.github.io/uploads/2017/02/zip_2MB.zip'
    return urls


def download_bc_gambar(url=None,tuliskefile=False, target_ip=None, target_port=None):
    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)
    namafile = os.path.basename(url)
    # ekstensi = tipe[content_type]

    #download gambar
    if (tuliskefile):
        fp = open(f"{namafile}","wb")
        fp.write(ff.content)
        fp.close()

    #broadcast gambar
    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)

    print(f"broadcast file {namafile}")
    image_file = open(f"{namafile}","rb")
    image_bytes = image_file.read()
    sock.sendto(image_bytes, (target_ip, target_port))

    #waktu
    waktu_process = datetime.datetime.now() - waktu_awal
    waktu_akhir =datetime.datetime.now()
    logging.warning(f"writing {namafile} dalam waktu {waktu_process} {waktu_awal} s/d {waktu_akhir}")
    return waktu_process

# if __name__=='__main__':
#     #check fungsi
#     k = download_bc_gambar('https://asset.kompas.com/crops/qz_j7xyaZgGgboomdCEXsfbSpoc=/0x0:998x665/740x500/data/photo/2020/03/01/5e5b52f4db896.jpg')
#     print(k)
```

2. **udpserver_broadcast.py**

Asmophel codingan — Latest commit b02bd7e 3 minutes ago — History

1 contributor

29 lines (20 sloc) | 595 Bytes — Raw | Blame

```python
import socket
import threading

SERVER_IP = '192.168.122.90'
SERVER_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)

sock.bind(("", SERVER_PORT))

count = 1

while True:
    data, addr = sock.recvfrom(54272)
    #buffer size 1024 * 53
    print(addr)
    print("diterima ", data)
    print("dikirim oleh " , addr)

    received_image = 'img' + str(count) + ".jpg"
    count += 1
    file = open(received_image, 'wb')
    file.write(data)
    file.close()

```

3. **udpserver_broadcast2.py**

Asmophel codingan — Latest commit b02bd7e 6 minutes ago — History

1 contributor

29 lines (20 sloc) | 596 Bytes — Raw | Blame

```python
import socket
import threading

SERVER_IP = '192.168.122.132'
SERVER_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)

sock.bind(("", SERVER_PORT))

count = 1

while True:
    data, addr = sock.recvfrom(54272)
    #buffer size 1024 * 53
    print(addr)
    print("diterima ", data)
    print("dikirim oleh " , addr)

    received_image = 'img' + str(count) + ".jpg"
    count += 1
    file = open(received_image, 'wb')
    file.write(data)
    file.close()

```

4. **multi_process_async.py**

Asmophel codingan — Latest commit b02bd7e 7 minutes ago — History

1 contributor

35 lines (27 sloc) | 1.19 KB — Raw | Blame

```python
from library import download_bc_gambar, get_url_list
import time
import datetime
from multiprocessing import Process, Pool

TARGET_IP = '255.255.255.255'
TARGET_PORT = 5005


def download_bc_semua():
    texec = dict()
    urls = get_url_list()
    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 urls:
        print(f"mendownload {urls[k]}")
        #bagian ini merupakan bagian yang mengistruksikan eksekusi fungsi download dan broadcast gambar secara multiprocess
        texec[k] = task_pool.apply_async(func=download_bc_gambar, args=(urls[k],True,TARGET_IP, TARGET_PORT))

    #setelah menyelesaikan tugasnya, dikembalikan ke main process dengan mengambil hasilnya dengan get
    for k in urls:
        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 download_bc_gambar akan dijalankan secara multi process

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

5. **multi_process.py**

Asmophel codingan — Latest commit b02bd7e 7 minutes ago — History

1 contributor

27 lines (25 sloc) | 1016 Bytes — Raw | Blame

```python
from library import download_bc_gambar, get_url_list
import time
import datetime
from multiprocessing import Process

TARGET_IP = '255.255.255.255'
TARGET_PORT = 5005

def download_bc_semua():
    texec = dict()
    urls = get_url_list()
    catat_awal = datetime.datetime.now()
    for k in urls:
        print(f"mendownload {urls[k]}")
        waktu = time.time()
        #bagian ini merupakan bagian yang mengistruksikan eksekusi fungsi download dan broadcast gambar secara multiprocess
        texec[k] = Process(target=download_bc_gambar, args=(urls[k],True,TARGET_IP, TARGET_PORT))
        texec[k].start()
    #setelah menyelesaikan tugasnya, dikembalikan ke main process dengan join
    for k in urls:
        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 download_bc_gambar akan dijalankan secara multi process
if __name__=='__main__':
    download_bc_semua()
```

6. **multi_thread_async.py**

Asmophel codingan — Latest commit b02bd7e 8 minutes ago — History

1 contributor

36 lines (28 sloc) | 1.13 KB — Raw | Blame

```python
from library import download_bc_gambar,get_url_list
import time
import datetime
import concurrent.futures

TARGET_IP = '255.255.255.255'
TARGET_PORT = 5005


def download_bc_semua():
    texec = dict()
    urls = get_url_list()
    status_task = dict()
    task = concurrent.futures.ThreadPoolExecutor(max_workers=4)
    catat_awal = datetime.datetime.now()
    for k in urls:
        print(f"mendownload {urls[k]}")
        waktu = time.time()
        #bagian ini merupakan bagian yang mengistruksikan eksekusi fungsi download dan broadcast gambar secara multithread
        texec[k] = task.submit(download_bc_gambar, urls[k],True, TARGET_IP, TARGET_PORT)

    #setelah menyelesaikan tugasnya, dikembalikan ke main thread dengan memanggil result
    for k in urls:
        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 download_bc_gambar akan dijalankan secara multithreading

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

7. **multi_thread.py**

Asmophel codingan — Latest commit b02bd7e 9 minutes ago — History

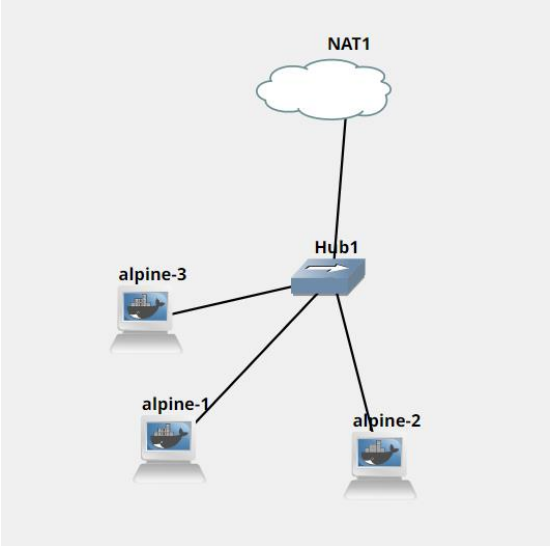
1 contributor

34 lines (25 sloc) | 997 Bytes — Raw | Blame

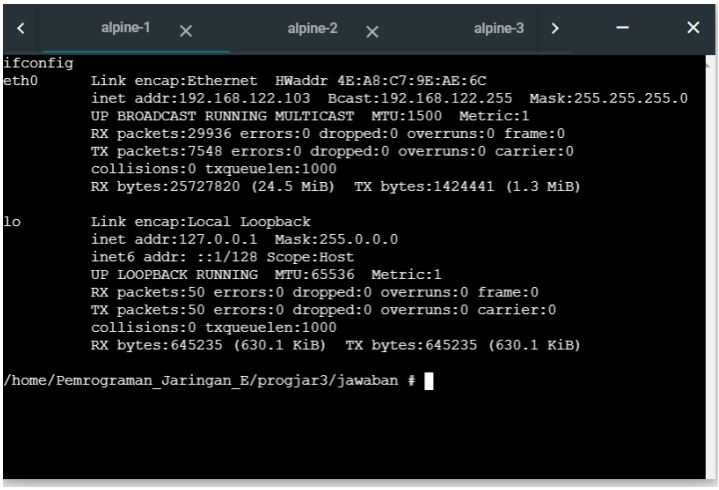
```python
from library import download_bc_gambar,get_url_list
import time
import datetime
import threading

TARGET_IP = '255.255.255.255'
TARGET_PORT = 5005


def download_bc_semua():
    texec = dict()
    urls = get_url_list()

    catat_awal = datetime.datetime.now()
    for k in urls:
        print(f"mendownload {urls[k]}")
        waktu = time.time()
        #bagian ini merupakan bagian yang mengistruksikan eksekusi fungsi download gambar secara multithread
        texec[k] = threading.Thread(target=download_bc_gambar, args=(urls[k],True,TARGET_IP, TARGET_PORT))
        texec[k].start()

    #setelah menyelesaikan tugasnya, dikembalikan ke main thread dengan join
    for k in urls:
        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 download_bc_gambar akan dijalankan secara multithreading

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

Dokumentasi :

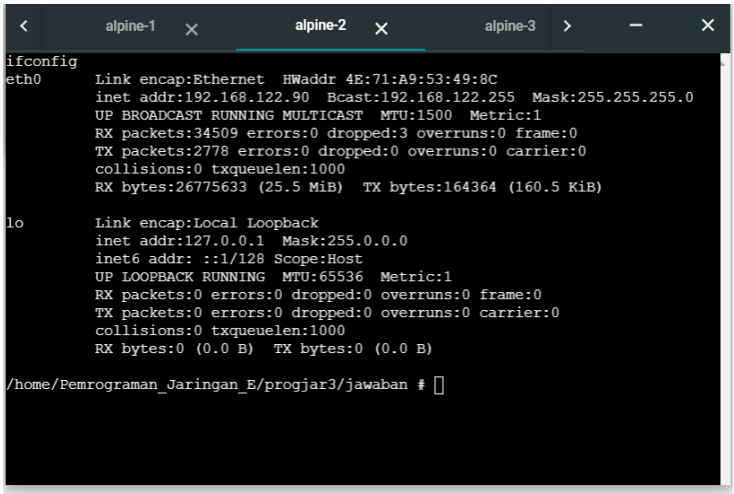
1. Arsitektur Jaringan, nantinya alpine-1 akan digunakan sebagai client, alpine-2 dan alpine-3 sebagai server



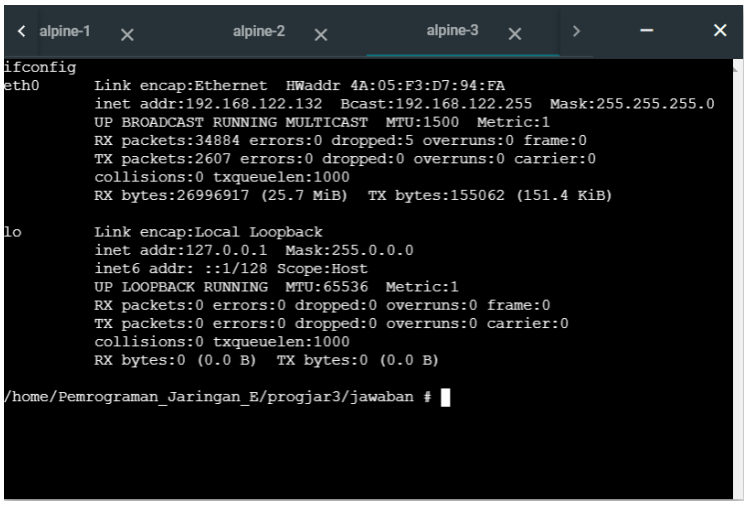

2. Cek ip pada client (alpine-1) dengan command **ifconfig**



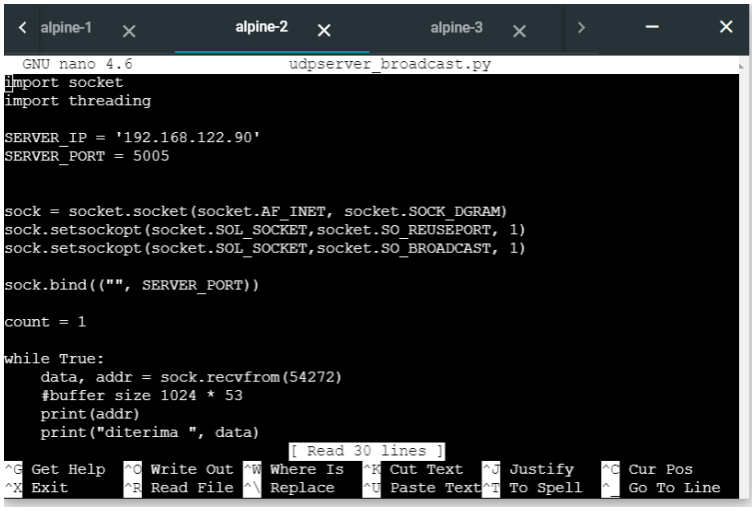

3. Cek ip pada server1 (alpine-2) dengan command **ifconfig**



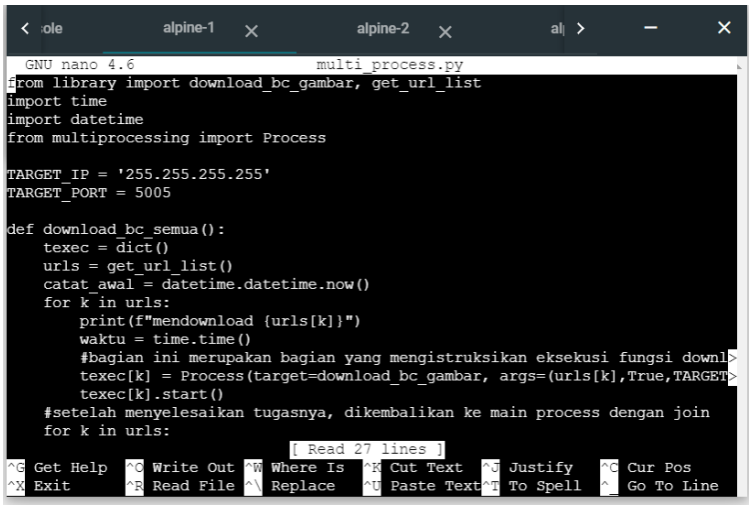
4. Cek ip pada server2 (alpine-3) dengan command **ifconfig**




5. Sesuaikan IP server pada masing-masing server sesuai alpine-2 dan alpine-3



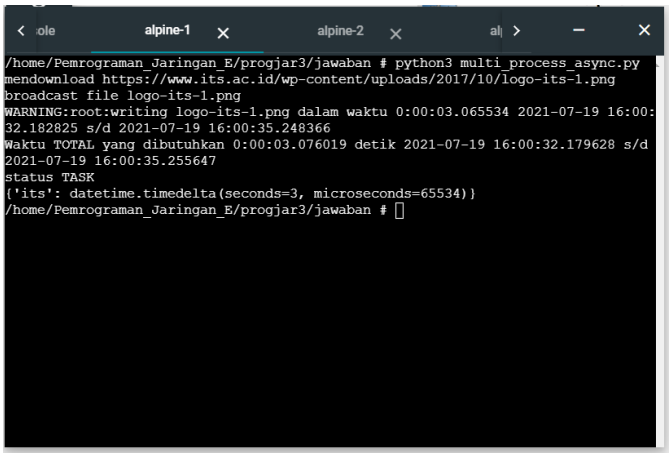

6. untuk IP client gunakan “255.255.255.255”, ubah coding pada program multi process, multi thread, multi process asynchronous dan multi thread asynchronous.



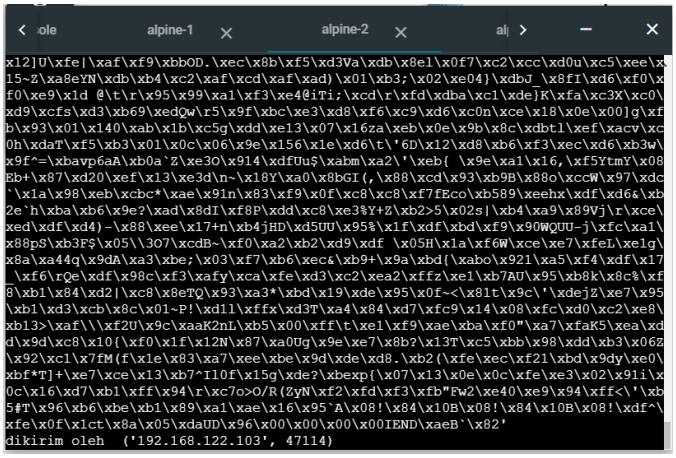
8. Selanjutnya run masing-masing server (alpine-2 dan alpine-3) dengan menjalankan program **udpserver_broadcast2.py.** Setelah itu jalankan ke 4 program tadi pada client secara bergantian.

Hasil Output **multi_process_async.py**

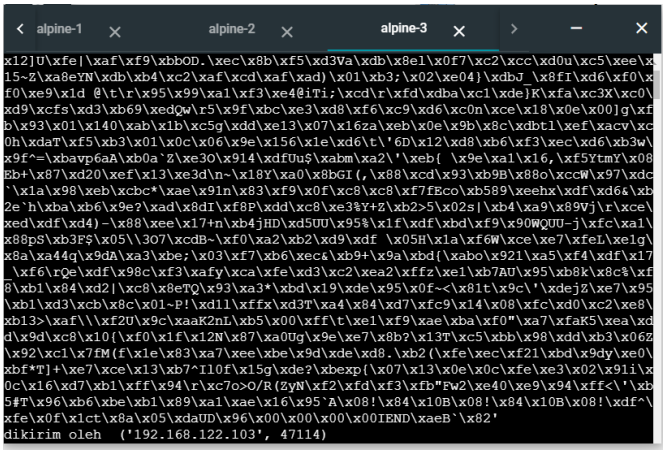
1. Client (alpine-1)




2. Server1 (alpine-2)

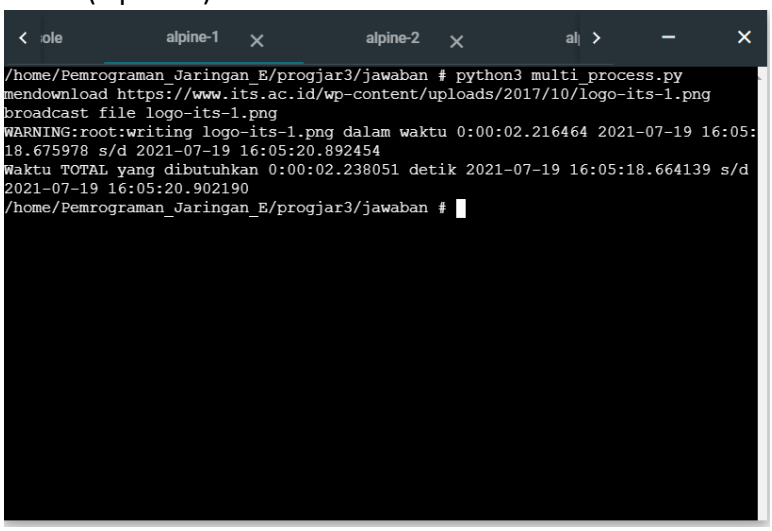



3. Server2 (alpine-3)

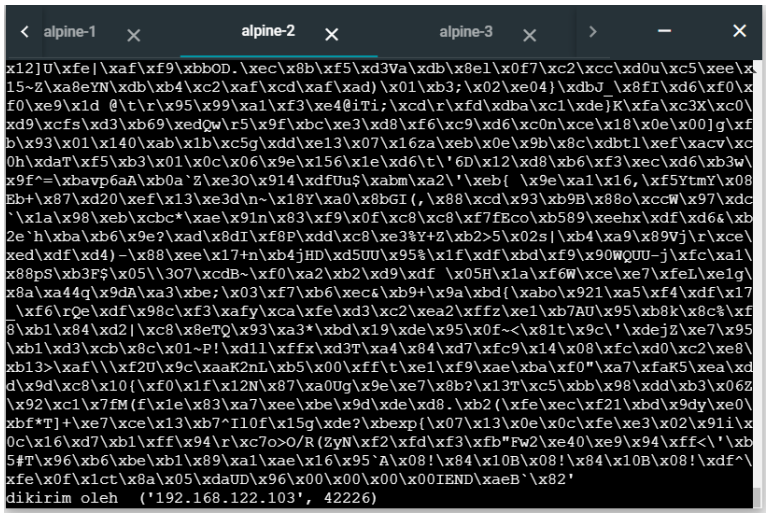


Hasil Output **multi_process.py**

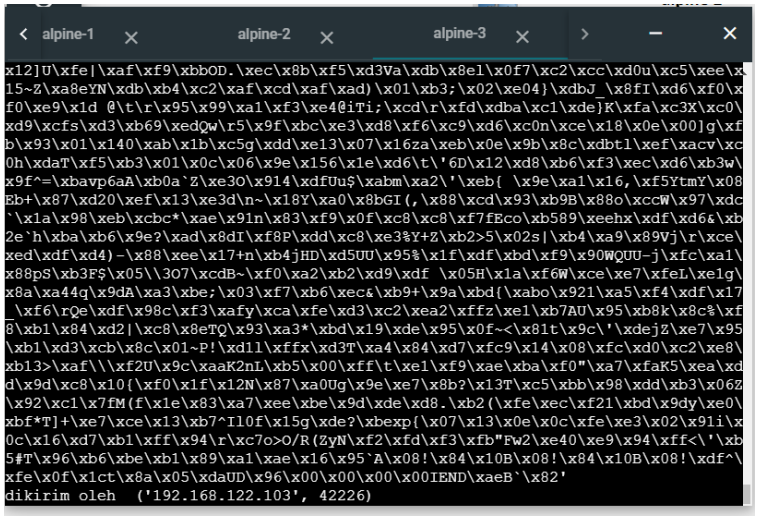
1. Client (alpine-1)




2. Server1 (alpine-2)

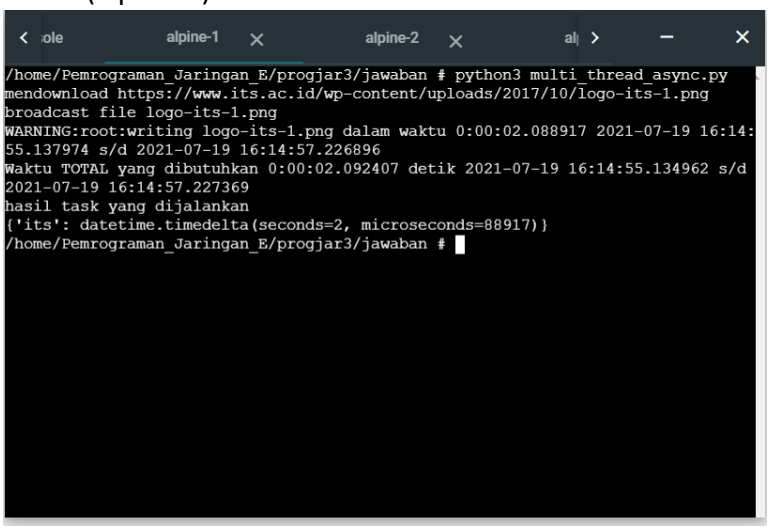



3. Server2 (alpine-3)

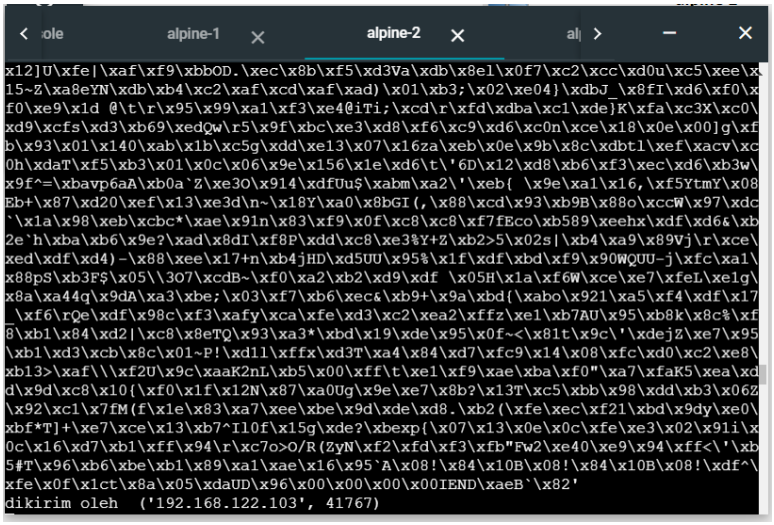


Hasil Output **multi_thread_async.py**

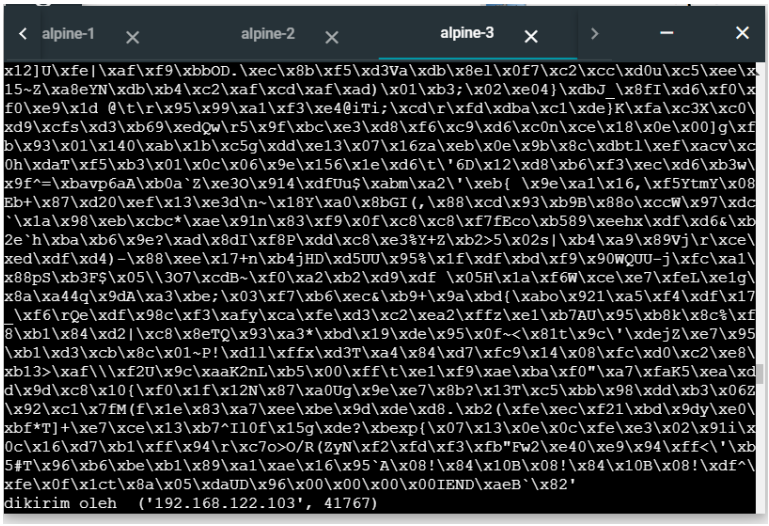
1. Client (alpine-1)




2. Server1 (alpine-2)

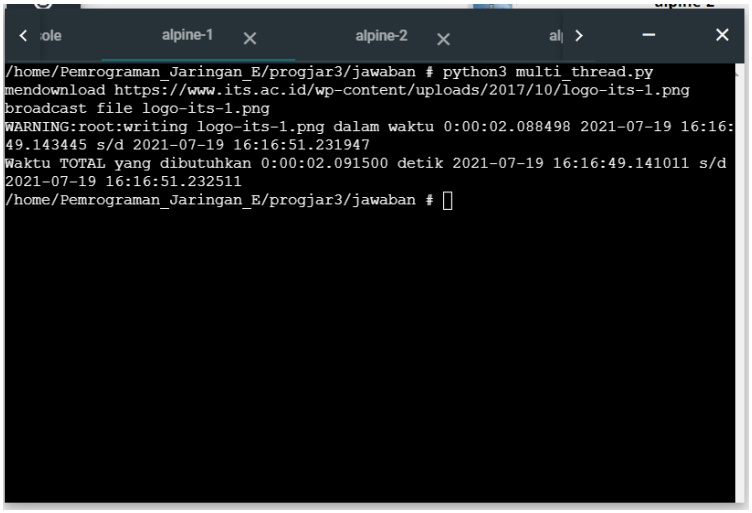



3. Server2 (alpine-3)

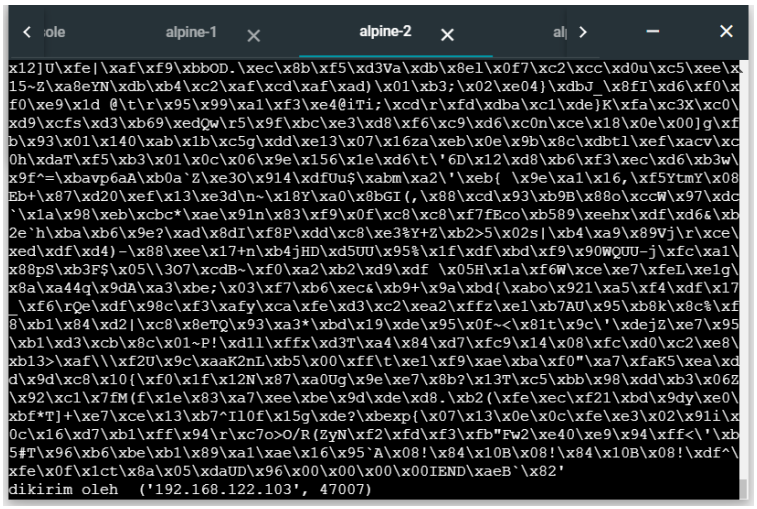


Hasil Output **multi_thread.py**

1. Client (alpine-1)




2. Server1 (alpine-2)




3. Server2 (alpine-3)