**DOKUMENTASI PROYEK AKHIR MATA KULIAH PEMROGRAMAN JARINGAN**



Disusun oleh :

|  |  |
| --- | --- |
| Irfan Maulana Akbar | 175150207111036 |
| Yohanna Fransiska Aladina | 175150201111027 |

PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN TEKNIK INFORMATIKA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA

2020

**DAFTAR ISI**

[BAB 1 PENDAHULUAN 3](#_Toc40202437)

[**1.1** **Latar Belakang** 3](#_Toc40202438)

[**1.2** **Alur Komunikasi** 3](#_Toc40202439)

[**1.3** **Representasi Data** 4](#_Toc40202440)

[**1.4** **Kelebihan dan Kekurangan** 4](#_Toc40202441)

[BAB 2 IMPLEMENTASI 5](#_Toc40202442)

[**Source Code** 5](#_Toc40202443)

[BAB 3 PENGUJIAN SISTEM 8](#_Toc40202444)

[BAB 4 DAFTAR REFERENSI 10](#_Toc40202445)

# **BAB 1 PENDAHULUAN**

## **1.1 Latar Belakang**

*Peer-to-peer* (P2P) merupakan protokol yang berada di layer aplikasi pada *OSI Reference* *Model.* Protokol ini memungkinkan setiap komputer untuk memiliki kemampuan dan tanggung jawab yang setara untuk memungkinkan pertukaran *resource* dan *services* secara langsung di antara *peers* (Miller, Michael., 2001). P2P menerapkan konsep desentralisasi, di mana setiap *peer* dapat berkomunikasi secara langsung ke *peer* lainnya yang berada di dalam satu jaringan tanpa perlu ditangani oleh server. Hal ini berbeda dengan arsitektur *client/*server tradisional, di mana satu atau lebih komputer telah ditetapkan untuk memberikan layanan kepada yang membutuhkannya. Aplikasi *Peer-to-Peer* (P2P) telah banyak digunakan untuk *chatting*, bertukar video, music, game, dan file data berukuran besar lainnya melalui internet. *Traffic* pada P2P menempati *bandwidth* jaringan yang signifikan dan sangat mempengaruhi pola penggunaan internet (Yu, Zhen., Zheng, Xue-feng., Wang, Shao-jie. & Wen, Chao., 2008 dan Lee, HyunYong., Nakao, Akihiro. & Kim, JongWon., 2010.)

Python adalah bahasa pemrograman yang *interpreted*, berorientasi objek, dan bahasa tingkat tinggi dengan semantik yang dinamis serta manajemen memori otomatis (Python Software Foundation., 2018). Contoh dari fitur relevan yang membuat Python semakin menarik untuk digunakan adalah bersifat *free*, *open-source*, dapat dijalankan di berbagai tipe sistem operasi, mempunyai banyak *library*, menggunakan sintaks yang *simple*, dan Python menyediakan bagian interaktif karena *programmer* dapat mengetik perintah di teks editor mereka secara interaktif yang akhirnya dapat dieksekusi dengan segera (Milano, Federico., 2013 dan Millman, K.Jarrod. & Aivazis, Michael., 2011).

Oleh karena itu, disini kami membuat implementasi metode socket, mendefinisikan alur komunikasi, dan merepresentasikan datanya untuk sebuah studi kasus *chatting* menggunakan protokol P2P yang dibuat dari awal dengan menggunakan Python socket yaitu dengan berbasis *thread* dan menambahkan satu server yang berfungsi sebagai *tracker* untuk melakukan pelacakan *peer* mana yang sedang aktif (terkoneksi).

## **1.2 Alur Komunikasi**

Alur komunikasi dari Implementasi Chatting Menggunakan Protokol P2P adalah :

1. Terdapat beberapa *peer* yang terhubung dan membuat koneksi
2. Beberapa *peer* tersebut *cepat-cepatan*sehingga yang pertama kali terkoneksi seolah-olah menjadi server dan akan melakukan *listing* terhadap *peer* lain yang telah terkoneksi (*peer* ini seolah-olah menjadi *client*)
3. *Peer* lain yang seolah-olah menjadi *client* saling mengirimkan pesan (dalam hal ini adalah *chatting*)
4. Jika *peer* yang bertindak sebelumnya seolah-olah menjadi server melakukan *disconnect*, maka *peer* lain yang seolah-olah menjadi *client* akan menggantikan tugasnya untuk seolah-olah menjadi server dan melakukan *listing* terhadap *peer* lain yang masih terkoneksi
5. *Peer* lain yang seolah-olah menjadi *client* dan masih terkoneksi bisa melakukan *chatting*
6. Begitu seterusnya.

## **1.3 Representasi Data**

Data yang berada pada Implementasi Chatting Menggunakan Protokol P2P adalah data berupa berupa *bytes* berjenis UTF-8 yang merupakan pengkodean karakter 8-bit *unicode* sehingga mampu mengkodekan hingga 1.112.0664 poin kode karakter.

## **1.4 Kelebihan dan Kekurangan**

Kelebihan dari Implementasi Chatting Menggunakan Protokol P2P adalah :

1. *Peer* langsung bisa mejadikan dirinya untuk bertindak seolah-olah sebagai server atau *client.*
2. Tidak perlu menyediakan server tambahan karena *client* juga bisa bertindak sebagai server.

Sedangkan kekurangannya adalah :

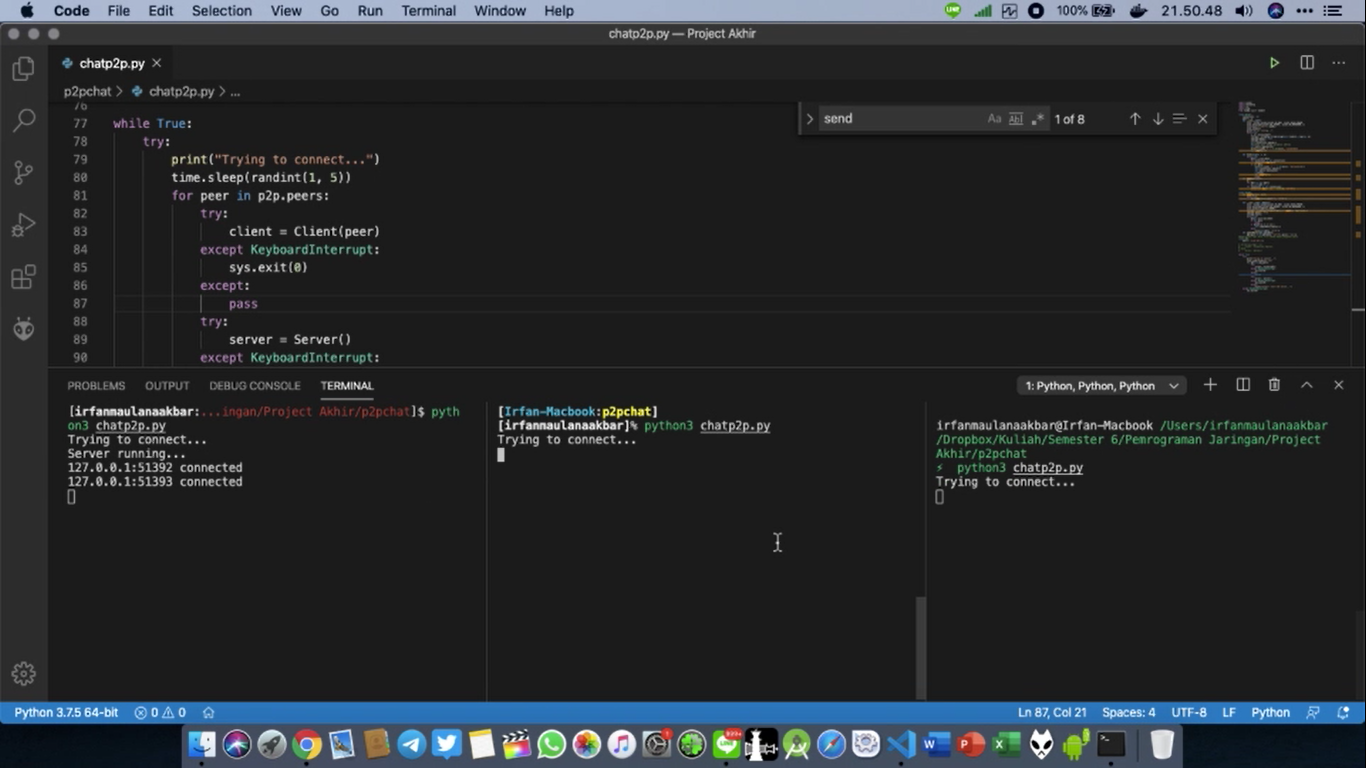
1. Antar *peer* baru bisa melakukan *chatting* ketika *peer* yang bertindak seolah-olah menjadi *client* sudah menemukan *peer* lain yang juga bertindak seolah-olah bertindak sebagai *client*. Jika tidak ditemukan pasangan *peer* lainnya, maka salah satu *peer* tersebut akan terus menunggu *peer* lainnya hingga ada.

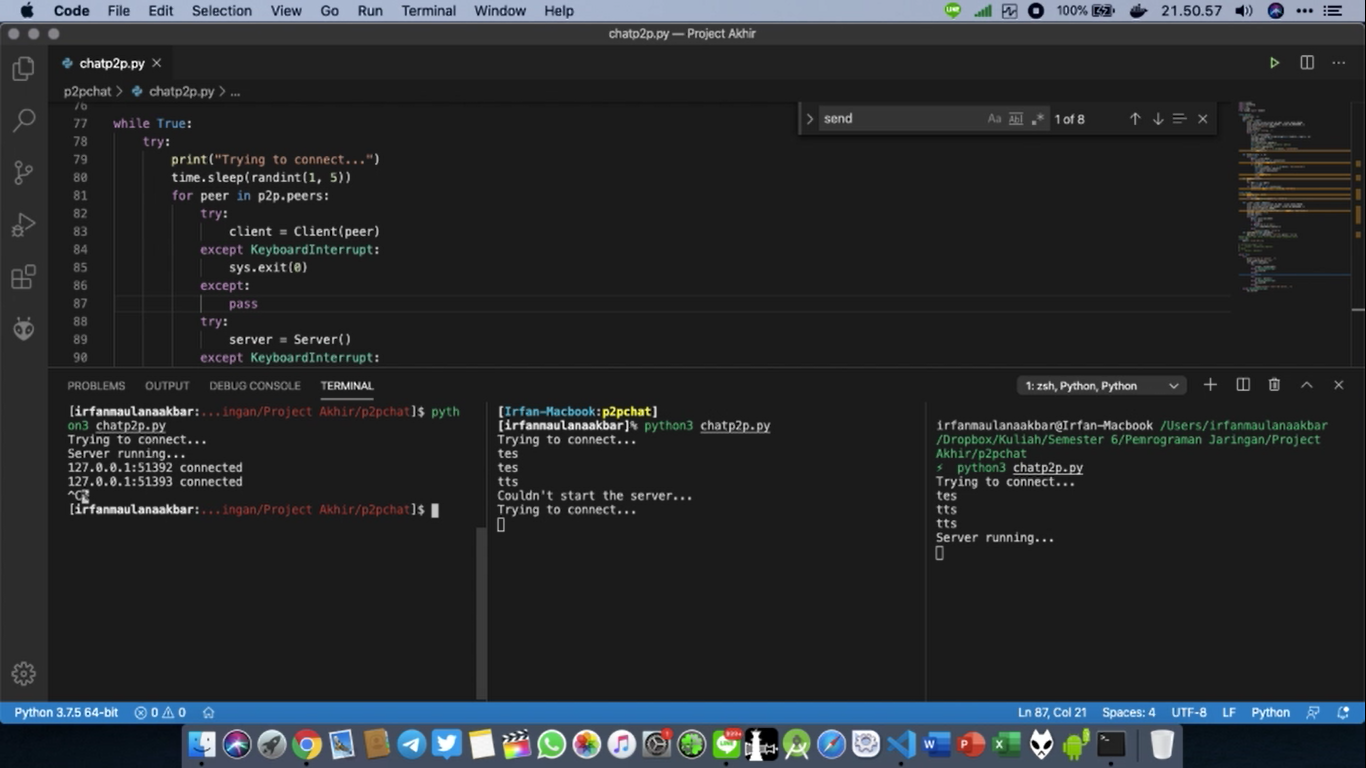
# **BAB 2 IMPLEMENTASI**

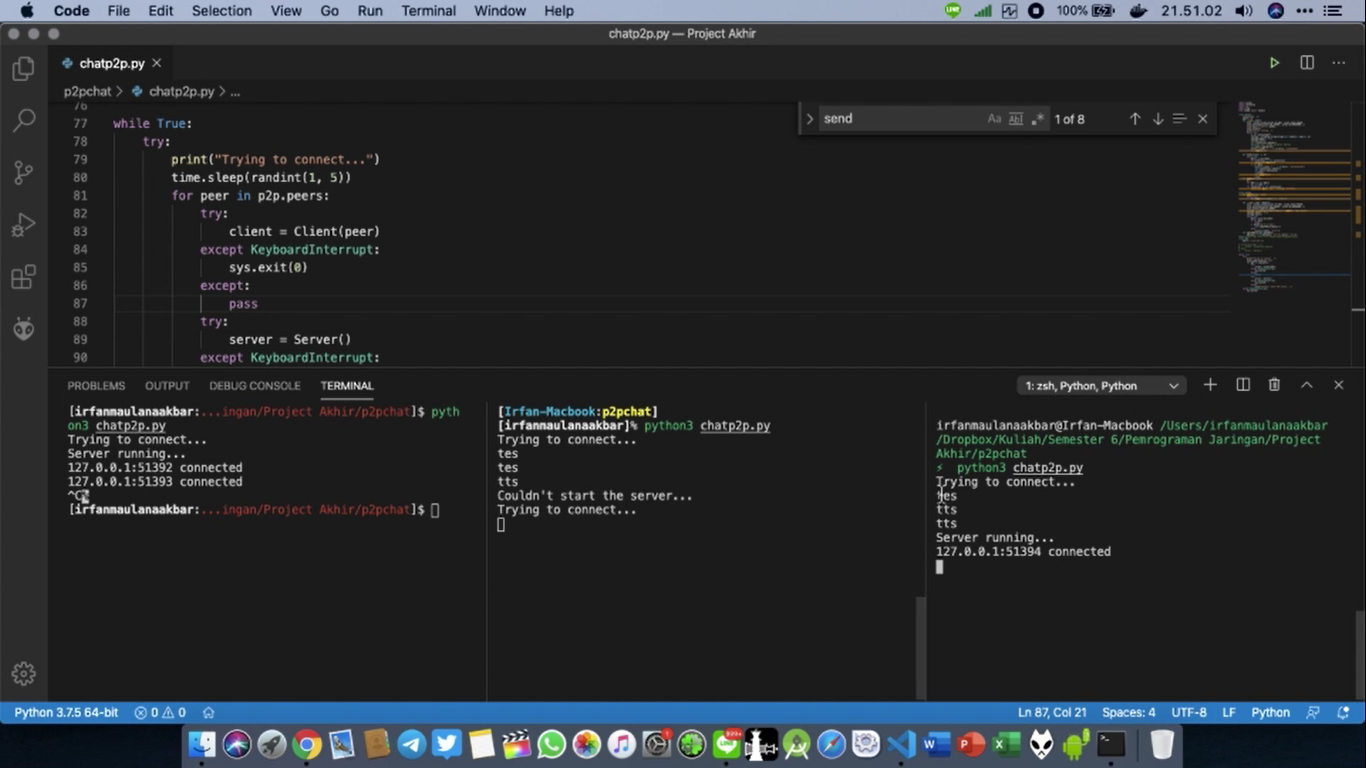
## **Source Code**

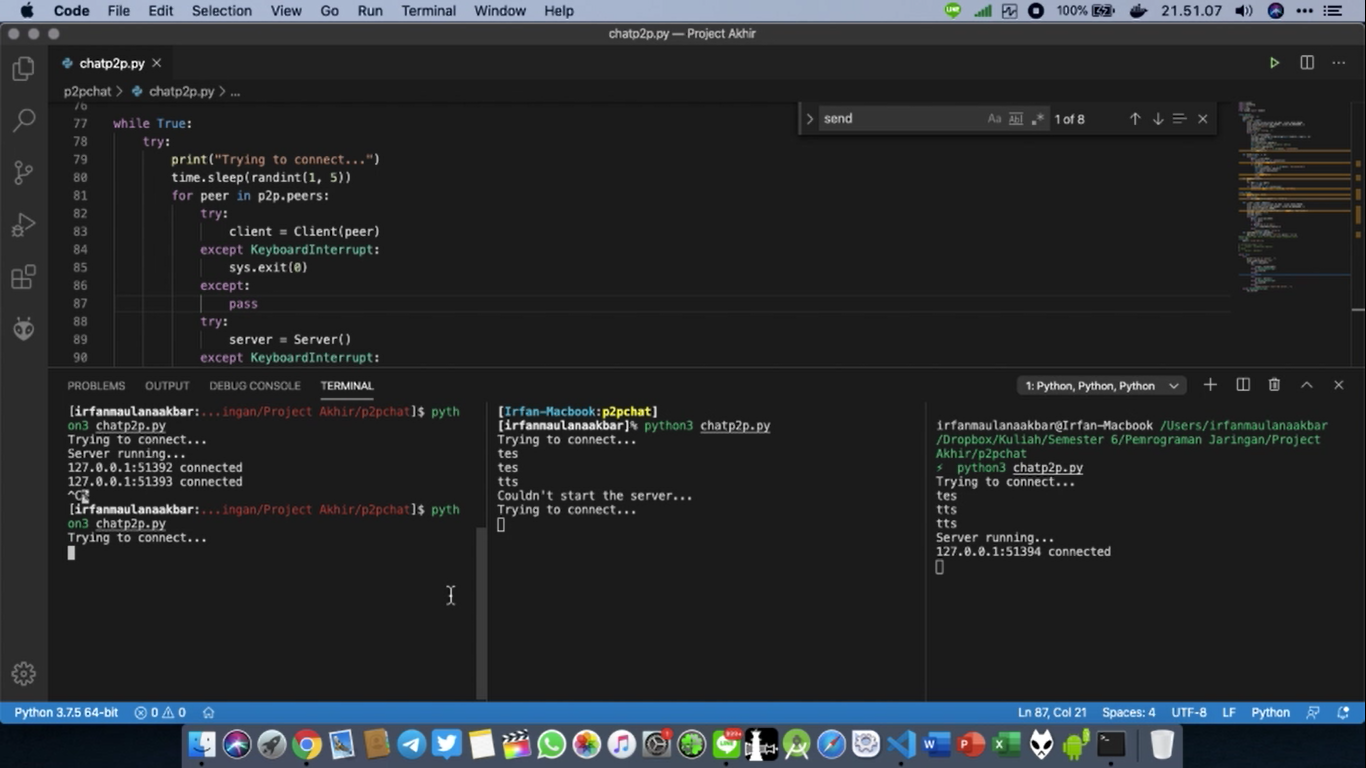
|  |  |
| --- | --- |
| chatp2p.py | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95 | import socket  import threading  import sys  import time  from random import randint  class Server:      connections = []      peers = []      def \_\_init\_\_(self):          sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)          sock.setsockopt(socket.SOL\_SOCKET, socket.SO\_REUSEADDR, 1)          sock.bind(("0.0.0.0", 10000))          sock.listen(1)          print("Server running...")          while True:              c, a = sock.accept()              cThread = threading.Thread(target=self.handler, args=(c, a))              cThread.daemon = True              cThread.start()              self.connections.append(c)              #menambah peers dengan parameter address              self.peers.append(a[0])              print(str(a[0]) + ':' + str(a[1]), "connected")              self.sendPeers()      def handler(self, c, a):          while True:              data = c.recv(1024)              for connection in self.connections:                  connection.send(data)              if not data:                  print(str(a[0]) + ':' + str(a[1]), "disconnected")                  self.connections.remove(c)                  self.peers.remove(a[0])                  c.close()                  self.sendPeers()                  break      def sendPeers(self):          p = ""          for peer in self.peers:              p = p + peer + ","          for connection in self.connections:              connection.send(b'\x11' + bytes(p, "utf-8"))  class Client:      def sendMsg(self, sock):          while True:              sock.send(bytes(input(""), 'utf-8'))      def \_\_init\_\_(self, address):          sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)          sock.setsockopt(socket.SOL\_SOCKET, socket.SO\_REUSEADDR, 1)          sock.connect((address,10000))          iThread = threading.Thread(target=self.sendMsg, args=(sock,))          iThread.daemon = True          iThread.start()          while True:              data = sock.recv(1024)              if not data:                  break              if data[0:1] == b'\x11':                  self.updatePeers(data[1:])              else:                  print(str(data, 'utf-8'))      def updatePeers(self,peerData):          p2p.peers = str(peerData, "utf-8").split(",")[:-1]  #kelas dibuat agar while true dibawah dapat mengakses peers  class p2p:      peers = ['127.0.0.1']  # if(len(sys.argv) > 1):  #     client = Client(sys.argv[1])  # else:  #     server = Server()  while True:      try:          print("Trying to connect...")          time.sleep(randint(1, 5))          for peer in p2p.peers:              try:                  client = Client(peer)              except KeyboardInterrupt:                  sys.exit(0)              except:                  pass              try:                  server = Server()              except KeyboardInterrupt:                  sys.exit(0)              except:                  print("Couldn't start the server...")      except KeyboardInterrupt:          sys.exit(0) |

# **BAB 3 PENGUJIAN SISTEM**









# **BAB 4 DAFTAR REFERENSI**

Miller, Michael., 2001. *Discovering P2P*. California: Sybex Inc.

Yu, Zhen., Zheng, Xue-feng., Wang, Shao-jie. & Wen, Chao., 2008. *A Mechanism for P2P Traffic* Control. Beijing: IEEE.

Lee, HyunYong., Nakao, Akihiro. & Kim, JongWon., 2010. Traffic control through bilateral cooperation between network operators and peers in P2P networks. Tokyo: IEEE.

Python Software Foundation., 2018. *What is Python? Executive summary*. [Online]. Tersedia di <<https://www.python.org/doc/essays/blurb>> [Diakses 10 Mei 2020].

Milano, Federico., 2013. A python-based software tool for power system analysis. Ireland: IEEE.

Millman, K.Jarrod. & Aivazis, Michael., 2011. *Python for Scientists and Engineers*. California: IEEE.