

MODUL 11 PEMROGRAMAN JARINGAN APLIKASI CHATTING MULTICAST (MULTITHREAD)



**NAMA : QORI EMALIA PUTRI M.
NIM : 1203210024
KELAS : IF-01-01**

**PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN BISNIS
INSTITUT TEKNOLOGI TELKOM SURABAYA
SEMESTER GENAP 2022/2023**

Modul 11

Aplikasi Chatting Multicast (Multithread)

I. Pengenalan Aplikasi Chatting Multicast (Multithread)

Hampir serupa dengan unicast, transmisi jaringan atau aliran data dengan sistem multicast dapat dilakukan dengan lebih dari dua perangkat sistem atau komputer. Multicast adalah transmisi jaringan one-to-many. Dengan menggunakan transmisi jaringan multicast dimaksudkan agar dapat mengirim data dan informasi kepada banyak komputer atau sistem tujuan, dengan tidak semua sebagai host. Multicast diperlukan dalam kondisi-kondisi tertentu, seperti ketika beberapa komputer dalam grup ingin menerima transmisi tersebut.

Salah satu contohnya adalah streaming video atau audio. Misalkan banyak komputer ingin terhubung ke komputer host dan menerima transmisi data video atau audio tersebut secara bersamaan. Jika transmisi dilakukan secara individu dan data informasi dikirim secara satu persatu, maka akan banyak aliran data dan waktu yang diperlukan. Sedangkan jika menggunakan transmisi broadcast, maka proses tidak lagi diperlukan sehingga komputer host dan komputer tujuan tidak lagi dapat berkomunikasi melalui jaringan tersebut.

Oleh karena itu, dengan menggunakan transmisi multicast maka video dan audio hanya dikirim sekali akan tetapi diterima oleh banyak sistem dan komputer. Transmisi multicast analoginya ketika pemberi undangan ingin memberikan informasi kepada teman sekolahnya. Pemberi sebagai host mengirimkan undangannya secara langsung di kelasnya sehingga informasi tersebut hanya dikirimkan satu kali dan diterima oleh semua teman dikelasnya. Melalui proses transmisi ini pemberi dapat mengetahui respon dari penerima.

II. Cara Kerja Aplikasi Chatting Multicast (Multithread)

Aplikasi chatting multicast dengan multithreading adalah aplikasi yang memungkinkan pengguna untuk mengirim pesan kepada sekelompok orang secara bersamaan, menggunakan konsep multithreading untuk memfasilitasi pengiriman pesan yang efisien. Dalam konteks aplikasi chatting multicast dengan multithreading, pengguna dapat membuat grup atau kanal dan mengirim pesan kepada semua anggota grup secara bersamaan. Aplikasi ini menggunakan multithreading, yaitu teknik yang memungkinkan pemrosesan pesan secara paralel atau simultan, untuk mengirim pesan kepada semua anggota grup dalam waktu yang singkat.

Aplikasi chatting multicast dengan multithreading bekerja dengan memanfaatkan konsep multicast dan multithreading dalam pengiriman pesan kepada sekelompok pengguna secara bersamaan. Berikut adalah gambaran umum tentang cara kerja aplikasi tersebut:

1. **Pembuatan grup atau kanal:** Pengguna dapat membuat grup atau kanal dan mengundang anggota-anggota lain untuk bergabung. Setiap grup memiliki identitas unik yang membedakan satu grup dari yang lain.
2. **Pengiriman pesan:** Ketika seorang pengguna mengirim pesan ke dalam grup, pesan tersebut dikirim ke server aplikasi.
3. **Multithreading:** Di sisi server, aplikasi menggunakan multithreading untuk mengelola

pengiriman pesan multicast secara paralel atau simultan. Setiap thread (benang) yang terbentuk akan bertanggung jawab mengirimkan pesan ke sekelompok pengguna dalam grup tersebut.

4. **Pengiriman pesan multicast:** Setiap thread akan mengirim pesan ke anggota grup yang dituju menggunakan protokol multicast. Protokol ini memungkinkan pengiriman pesan ke beberapa penerima secara bersamaan dalam satu operasi pengiriman.
5. **Penerimaan pesan:** Setiap anggota grup yang terhubung dengan aplikasi akan menerima pesan tersebut. Pesan tersebut akan ditampilkan dalam antarmuka aplikasi chat mereka secara real-time.
6. **Notifikasi:** Anggota grup yang terhubung ke aplikasi biasanya akan menerima notifikasi tentang adanya pesan baru dalam grup yang mereka ikuti. Notifikasi ini membantu pengguna tetap up-to-date dengan percakapan dalam grup.

Dengan menggunakan multithreading, aplikasi dapat mengirim pesan multicast dengan efisien dan cepat kepada sekelompok pengguna dalam grup. Teknik multithreading memungkinkan pengiriman pesan yang simultan, yang dapat meningkatkan responsivitas dan mengurangi waktu pengiriman pesan.

Multicast: Pengiriman data dari 1 pengirim ke beberapa penerima. untuk melakukan multicast dalam jaringan komputer, kita menggunakan Multicast Address dengan range 224.0.0.0 – 239.255.255.255

III. Pengenalan Aplikasi Chatting Broadcast (Multithread)

Broadcast merupakan transmisi jaringan one-to-all. Transmisi jaringan dengan sistem broadcast dapat dilakukan dengan banyak sekali penerima atau perangkat sistem tujuan. Akan tetapi sistem transmisi broadcast cenderung membuang resource. Dalam jaringan LAN, komputer asal sebagai host mengirimkan data informasi kepada sembarang komputer lain yang terhubung di dalam jaringan.

Transmisi jaringan broadcast memiliki kekurangan yaitu sistem asal yang menjadi host tidak akan mengetahui bagaimana respon dari sistem tujuan apakah data diterima dengan baik atau data terjadi corrupt. Contoh penggunaan broadcast adalah transmisi saluran data pada televisi. Transmisi jaringan broadcast memiliki analogi seperti seorang pemberi undangan yang ingin mengirimkan informasi kepada orang lain. Pemberi undangan sebagai host membagikan undangan dengan menyebarnya di jalan-jalan. Oleh karena itu, perangkat host tidak akan mengetahui bahwa informasi yang dikirimkannya telah sampai dan respon penerima pun tidak akan diketahui oleh perangkat sistem yang menjadi host.

IV. Cara Kerja Aplikasi Chatting Broadcast (Multithread)

Aplikasi chatting broadcast dengan multithreading adalah aplikasi yang memungkinkan pengguna untuk mengirim pesan kepada seluruh pengguna yang terhubung dalam jaringan atau grup secara bersamaan. Istilah "broadcast" merujuk pada pengiriman pesan kepada semua penerima yang ada. Dalam konteks aplikasi chatting broadcast dengan multithreading, setiap pesan yang dikirim oleh pengguna akan dikirim secara langsung kepada semua pengguna yang terhubung. Aplikasi ini menggunakan multithreading, yaitu teknik yang memungkinkan pemrosesan pesan secara paralel atau simultan, untuk mengirim pesan kepada semua penerima dalam waktu yang singkat.

Aplikasi chatting broadcast dengan multithreading bekerja dengan memanfaatkan konsep broadcast dan multithreading dalam pengiriman pesan kepada seluruh pengguna yang terhubung. Berikut adalah gambaran umum tentang cara kerja aplikasi tersebut:

1. **Pengiriman pesan broadcast:** Ketika seorang pengguna mengirim pesan broadcast, pesan tersebut dikirim ke server aplikasi.
2. **Multithreading:** Di sisi server, aplikasi menggunakan multithreading untuk mengelola pengiriman pesan broadcast secara paralel atau simultan. Setiap thread (benang) yang terbentuk akan bertanggung jawab mengirimkan pesan kepada semua pengguna yang terhubung dalam jaringan atau grup.
3. **Pengiriman pesan:** Setiap thread akan mengirim pesan ke pengguna-pengguna yang terhubung menggunakan protokol broadcast. Protokol ini memungkinkan pengiriman pesan kepada semua penerima secara bersamaan dalam satu operasi pengiriman.
4. **Penerimaan pesan:** Setiap pengguna yang terhubung ke aplikasi akan menerima pesan tersebut. Pesan akan ditampilkan dalam antarmuka aplikasi chat mereka secara real-time.
5. **Notifikasi:** Pengguna yang terhubung ke aplikasi biasanya akan menerima notifikasi tentang adanya pesan baru yang dikirim sebagai broadcast. Notifikasi ini membantu pengguna tetap up-to-date dengan pesan-pesan yang masuk.

Dengan menggunakan multithreading, aplikasi dapat mengirim pesan broadcast dengan efisien dan cepat kepada seluruh pengguna yang terhubung dalam jaringan atau grup. Teknik multithreading memungkinkan pengiriman pesan yang simultan, yang dapat meningkatkan responsivitas dan mengurangi waktu pengiriman pesan.

Broadcast: Pengiriman data dari 1 pengirim ke seluruh penerima. untuk melakukan broadcast ke seluruh penerima, kita menggunakan broadcast address yang merupakan IP terakhir dari sebuah network address, atau disebut juga dengan last IP address.

V. Pembuatan Simple Aplikasi Chatting Multicast (Multithread)

Berikut adalah contoh sederhana program Pembuatan Aplikasi Chatting Multicast (Multithread) menggunakan Python:

SENDER

```
# SENDER
import socket

group = '224.1.1.1'
port = 5004

# 2-hop restriction in network
ttl = 2

sock = socket.socket(socket.AF_INET,
                     socket.SOCK_DGRAM,
                     socket.IPPROTO_UDP)
sock.setsockopt(socket.IPPROTO_IP,
               socket.IP_MULTICAST_TTL,
               ttl)
```

```
while True:
    pesan = input("masukkan pesan :")
    sock.sendto(pesan.encode('utf-8'), (group, port))
```

RECEIVER 1

```
# RECEIVER
import socket
import struct

MCAST_GRP = '224.1.1.1'
MCAST_PORT = 5004

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM, socket.IPPROTO_UDP)
sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

sock.bind(('', MCAST_PORT))
mreq = struct.pack("4sl", socket.inet_aton(MCAST_GRP), socket.INADDR_ANY)

sock.setsockopt(socket.IPPROTO_IP, socket.IP_ADD_MEMBERSHIP, mreq)

while True:
    print(sock.recv(10240))
```

RECEIVER 2

```
# RECEIVER
import socket
import struct

MCAST_GRP = '224.1.1.1'
MCAST_PORT = 5004

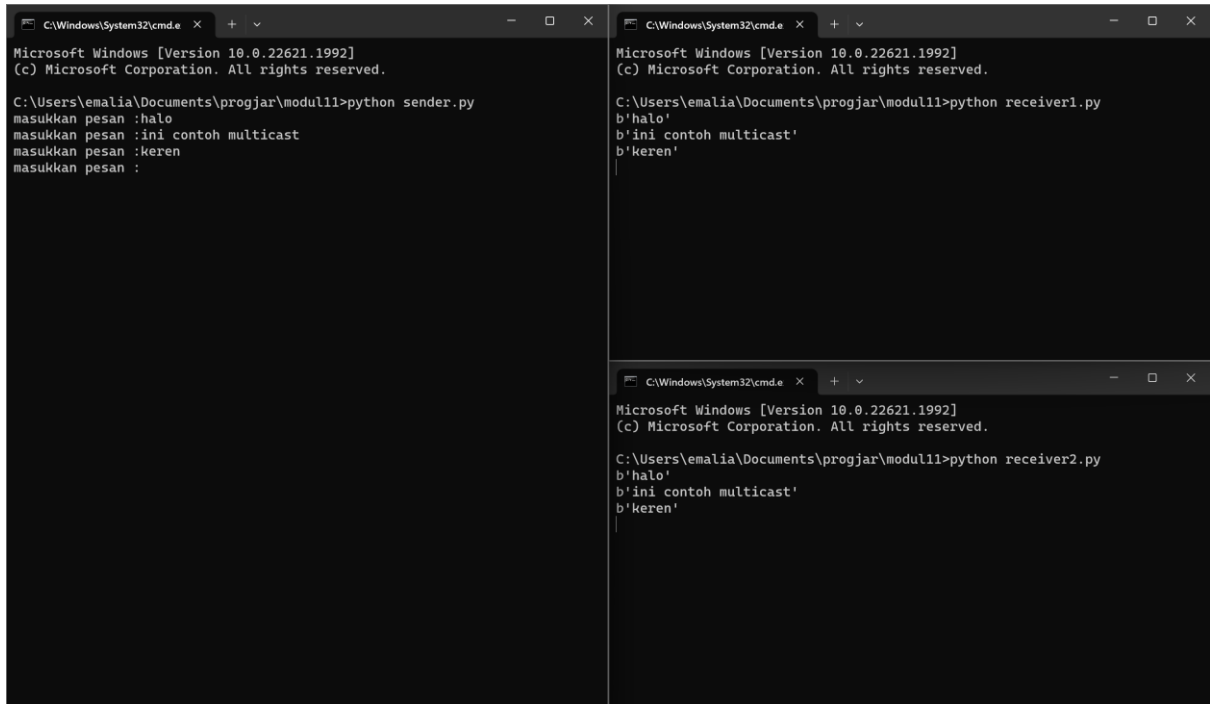
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM, socket.IPPROTO_UDP)
sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

sock.bind(('', MCAST_PORT))
mreq = struct.pack("4sl", socket.inet_aton(MCAST_GRP), socket.INADDR_ANY)

sock.setsockopt(socket.IPPROTO_IP, socket.IP_ADD_MEMBERSHIP, mreq)

while True:
    print(sock.recv(10240))
```

HASIL RUN



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.22621.1992]
(c) Microsoft Corporation. All rights reserved.

C:\Users\emalia\Documents\progjar\modul11>python sender.py
masukkan pesan :halo
masukkan pesan :ini contoh multicast
masukkan pesan :keren
masukkan pesan :
```

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.22621.1992]
(c) Microsoft Corporation. All rights reserved.

C:\Users\emalia\Documents\progjar\modul11>python receiver1.py
b'halo'
b'ini contoh multicast'
b'keren'
```

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.22621.1992]
(c) Microsoft Corporation. All rights reserved.

C:\Users\emalia\Documents\progjar\modul11>python receiver2.py
b'halo'
b'ini contoh multicast'
b'keren'
```

ANALISA HASIL PERCOBAAN

Program tersebut merupakan program untuk membuat sistem komunikasi multicast sederhana, di mana pengirim akan mengirim pesan ke grup multicast dan penerima-penerima akan menerima pesan yang dikirim oleh pengirim. Pada kode program di atas, TTL ditetapkan dengan nilai 2, artinya pesan hanya akan mencapai penerima dalam 2 hop (router) sebelum dihapus dari jaringan. Pertama, kita menjalankan server kemudian receiver1 dan receiver2. Pada program tersebut, yang bisa mengirim pesan hanyalah sender.

VI. Pembuatan Simple Aplikasi Chatting Broadcast (Multithread)

Berikut adalah contoh sederhana program Pembuatan Aplikasi Chatting Broadcast (Multithread) menggunakan Python:

SERVER

```
import socket

s = socket.socket() # Create a new socket object
host = socket.gethostname() # Get the hostname of the current machine
port = 8080 # Define the port number to listen on
s.bind((host, port)) # Bind the socket to the host and port
s.listen(2) # Listen for two connections (max number of queued
connections)
print("Waiting for connections...")

# Accept the first client's connection and get its socket object and
address
conn1, addr1 = s.accept()
```

```

print("Client_1 connected")
conn1.send("Selamat datang di Server...".encode()) # Send a welcome
message to the first client

# Accept the second client's connection and get its socket object and
address
conn2, addr2 = s.accept()
print("Client_2 connected")
conn2.send("Selamat datang di Server...".encode()) # Send a welcome
message to the second client

while True: # Start an infinite loop to keep the server running
    message = input("Masukan Pesan : ") # Prompt the server admin to enter
a message
    message = str(message).encode() # Convert the message to bytes (encode
it)
    conn1.send(message) # Send the message to the first client
    conn2.send(message) # Send the message to the second client
    print("Pesan Terkirim") # Print a message indicating that the message
has been sent

    recv_message = conn1.recv(1024) # Receive a message from the first
client (up to 1024 bytes)
    print("Client_1 : ", recv_message.decode()) # Print the received
message from the first client
    conn2.send(recv_message) # Send the received message to the second
client

    recv_message = conn2.recv(1024) # Receive a message from the second
client (up to 1024 bytes)
    print("Client_2 : ", recv_message.decode()) # Print the received
message from the second client
    conn1.send(recv_message) # Send the received message to the first
client

```

CLIENT 1

```

import socket

s = socket.socket() # Create a new socket object
host = socket.gethostname() # Get the hostname of the current machine
port = 8080 # Define the port number to connect to (must match the
server's port)
s.connect((host, port)) # Connect to the server using the specified host
and port
print("Menyambungkan ke Server") # Print a message indicating successful
connection

```

```

# Receive the welcome message from the server and decode it from bytes to a
string
message = s.recv(1024)
message = message.decode()
print("Pesan dari server : ", message) # Print the welcome message from
the server

while True: # Start an infinite loop to keep the client running
    # Receive a message from the server and decode it from bytes to a
    string
    message = s.recv(1024)
    message = message.decode()
    print("Server : ", message) # Print the received message from the
    server

    new_message = input("Masukkan Pesan : ") # Prompt the client to enter
    a new message
    new_message = str(new_message).encode() # Convert the new message to
    bytes (encode it)
    s.send(new_message) # Send the new message to the server
    print("Pesan Terkirim") # Print a message indicating that the new
    message has been sent

    # Receive the response from the server and decode it from bytes to a
    string
    message = s.recv(1024)
    message = message.decode()
    print("Client_2 : ", message) # Print the response from the server

```

CLIENT 2

```

import socket

s = socket.socket() # Create a new socket object
host = socket.gethostname() # Get the hostname of the current machine
port = 8080 # Define the port number to connect to (must match the
server's port)
s.connect((host, port)) # Connect to the server using the specified host
and port
print("Menyambungkan ke Server") # Print a message indicating successful
connection

# Receive the welcome message from the server and decode it from bytes to a
string
message = s.recv(1024)
message = message.decode()
print("Pesan dari server : ", message) # Print the welcome message from
the server

```



```

while True: # Start an infinite loop to keep the client running
    # Receive a message from the server and decode it from bytes to a
    string
    message = s.recv(1024)
    message = message.decode()
    print("Server : ", message) # Print the message received from the
    server

    # Receive a message from Client_1 and decode it from bytes to a string
    message = s.recv(1024)
    message = message.decode()
    print("Client_1 : ", message) # Print the message received from
    Client_1

    new_message = input("Masukkan Pesan : ") # Prompt the client to enter
    a new message
    new_message = str(new_message).encode() # Convert the new message to
    bytes (encode it)
    s.send(new_message) # Send the new message to the server
    print("Pesan Terkirim") # Print a message indicating that the new
    message has been sent

```

HASIL RUN

```

C:\Windows\System32\cmd.e x + v
Microsoft Windows [Version 10.0.22621.1992]
(c) Microsoft Corporation. All rights reserved.

C:\Users\emalia\Documents\progjar\modull1>python server.py
Waiting for connections...
Client_1 connected
Client_2 connected
Masukan Pesan : halo guys
Pesan Terkirim
Client_1 : woahh
Client_2 : ini client 2
Masukan Pesan : pesan kedua
Pesan Terkirim
Client_1 : client 1 di sinii
Client_2 : hmm
Masukan Pesan : |

C:\Windows\System32\cmd.e x + v
Microsoft Windows [Version 10.0.22621.1992]
(c) Microsoft Corporation. All rights reserved.

C:\Users\emalia\Documents\progjar\modull1>python client1.py
Menyambungkan ke Server
Pesan dari server : Selamat datang di Server...
Server : halo guys
Masukkan Pesan : woahh
Pesan Terkirim
Client_2 : ini client 2
Server : pesan kedua
Masukkan Pesan : client 1 di sinii
Pesan Terkirim
Client_2 : hmm

C:\Windows\System32\cmd.e x + v
Microsoft Windows [Version 10.0.22621.1992]
(c) Microsoft Corporation. All rights reserved.

C:\Users\emalia\Documents\progjar\modull1>python client2.py
Menyambungkan ke Server
Pesan dari server : Selamat datang di Server...
Server : halo guys
Client_1 : woahh
Masukkan Pesan : ini client 2
Pesan Terkirim
Server : pesan kedua
Client_1 : client 1 di sinii
Masukkan Pesan : hmm
Pesan Terkirim

```

ANALISA HASIL PERCOBAAN

Pertama, kita jalankan server, server akan mendengarkan dua koneksi, setelah menerima koneksi, server akan mengirim pesan selamat datang pada setiap client yang terhubung, kemudian server memasuki perulangan di mana ia menunggu masukan dari admin server (yang menjalankan program server) untuk mengirim pesan ke kedua client

yang terhubung. Setelah mengirim pesan, server menunggu untuk menerima respons dari kedua client dan meneruskan respons tersebut antara client-client tersebut.

Setelah itu kita jalankan client, client akan menerima pesan selamat datang dari server, kemudian client memasuki dalam perulangan di mana ia menunggu menerima pesan dari server dan client lainnya. client juga memungkinkan pengguna untuk memasukkan dan mengirimkan pesan yang ingin dikirimkan ke server.

Karena program tersebut merupakan komunikasi menggunakan broadcast, maka komunikasi dilakukan secara **paralel** (bergantian)

VII. Tugas dan Latihan

1. Buat laporan percobaan praktikum dan beri analisa hasil percobaan tadi yang sudah dibuat Aplikasi Chatting Multicast & Broadcast (Multithread).
2. Tugas tambahan, proses 4 koneksi dengan tambahkan client menjadi 4 client!

MODIFIKASI AGAR PROGRAM MENJADI 4 CLIENT

Server

```
import socket

s = socket.socket()
host = socket.gethostname()
port = 8080
s.bind((host, port))
s.listen(4) # kita ubah menjadi (4) karena akan terdapat 4 client

print("Waiting for connections...")

conn1, addr1 = s.accept()
print("Client_1 connected")
conn1.send("Selamat datang di Server".encode())

conn2, addr2 = s.accept()
print("Client_2 connected")
conn2.send("Selamat datang di Server".encode())

conn3, addr3 = s.accept()
print("Client_3 connected")
conn3.send("Selamat datang di Server".encode())

conn4, addr4 = s.accept()
print("Client_4 connected")
conn4.send("Selamat datang di Server...".encode())

while True:
    message = input("Masukan Pesan : ")
    message = str(message).encode()
    conn1.send(message)
    conn2.send(message)
```

```

conn3.send(message)
conn4.send(message)
print("Pesan Terkirim")

recv_message = conn1.recv(1024) #menerima pesan dari client
print("Client_1 : ", recv_message.decode())
conn2.send(recv_message)
conn3.send(recv_message)
conn4.send(recv_message)

recv_message = conn2.recv(1024)
print("Client_2 : ", recv_message.decode())
conn1.send(recv_message)
conn3.send(recv_message)
conn4.send(recv_message)

recv_message = conn3.recv(1024)
print("Client_3 : ", recv_message.decode())
conn1.send(recv_message)
conn2.send(recv_message)
conn4.send(recv_message)

recv_message = conn4.recv(1024)
print("Client_4 : ", recv_message.decode())
conn1.send(recv_message)
conn2.send(recv_message)
conn3.send(recv_message)

```

Client 1

```

import socket

s = socket.socket()
host = socket.gethostname()
port = 8080

s.connect((host, port))

print("Menyambungkan ke Server")

# Receive the welcome message from the server and decode it from bytes
to a string
message = s.recv(1024)
message = message.decode()
print("Pesan dari server : ", message) # Print the welcome message
from the server

while True: # Start an infinite loop to keep the client running

```

```

    # Receive a message from the server and decode it from bytes to a
    string
    message = s.recv(1024)
    message = message.decode()
    print("Server : ", message) # Print the message received from the
    server

    new_message = input("Masukkan Pesan : ") # Prompt the client to
    enter a new message
    new_message = str(new_message).encode() # Convert the new message
    to bytes (encode it)
    s.send(new_message) # Send the new message to the server
    print("Pesan Terkirim") # Print a message indicating that the new
    message has been sent

    # Receive a message from Client_2 and decode it from bytes to a
    string
    message = s.recv(1024)
    message = message.decode()
    print("Client_2 : ", message) # Print the message received from
    Client_2

    # Receive a message from Client_3 and decode it from bytes to a
    string
    message = s.recv(1024)
    message = message.decode()
    print("Client_3 : ", message) # Print the message received from
    Client_3

    # Receive a message from Client_4 and decode it from bytes to a
    string
    message = s.recv(1024)
    message = message.decode()
    print("Client_4 : ", message) # Print the message received from
    Client_4

```

Client 2

```

import socket

s = socket.socket()
host = socket.gethostname()
port = 8080

s.connect((host, port))

print("Menyambungkan ke Server")

```

```

# Receive the welcome message from the server and decode it from bytes
to a string
message = s.recv(1024)
message = message.decode()
print("Pesan dari server : ", message) # Print the welcome message
from the server

while True: # Start an infinite loop to keep the client running
    # Receive a message from the server and decode it from bytes to a
    string
    message = s.recv(1024)
    message = message.decode()
    print("Server : ", message) # Print the message received from the
    server

    # Receive a message from Client_1 and decode it from bytes to a
    string
    message = s.recv(1024)
    message = message.decode()
    print("Client_1 : ", message) # Print the message received from
    Client_2

    new_message = input("Masukkan Pesan : ") # Prompt the client to
    enter a new message
    new_message = str(new_message).encode() # Convert the new message
    to bytes (encode it)
    s.send(new_message) # Send the new message to the server
    print("Pesan Terkirim") # Print a message indicating that the new
    message has been sent

    # Receive a message from Client_3 and decode it from bytes to a
    string
    message = s.recv(1024)
    message = message.decode()
    print("Client_3 : ", message) # Print the message received from
    Client_3

    # Receive a message from Client_4 and decode it from bytes to a
    string
    message = s.recv(1024)
    message = message.decode()
    print("Client_4 : ", message) # Print the message received from
    Client_4

```

Client 3

```

import socket

s = socket.socket()

```

```

host = socket.gethostname()
port = 8080

s.connect((host, port))

print("Menyambungkan ke Server")

# Receive the welcome message from the server and decode it from bytes
to a string
message = s.recv(1024)
message = message.decode()
print("Pesan dari server : ", message) # Print the welcome message
from the server

while True: # Start an infinite loop to keep the client running
    # Receive a message from the server and decode it from bytes to a
    string
    message = s.recv(1024)
    message = message.decode()
    print("Server : ", message) # Print the message received from the
    server

    # Receive a message from Client_1 and decode it from bytes to a
    string
    message = s.recv(1024)
    message = message.decode()
    print("Client_1 : ", message) # Print the message received from
    Client_2

    # Receive a message from Client_2 and decode it from bytes to a
    string
    message = s.recv(1024)
    message = message.decode()
    print("Client_2 : ", message) # Print the message received from
    Client_3

    new_message = input("Masukkan Pesan : ") # Prompt the client to
    enter a new message
    new_message = str(new_message).encode() # Convert the new message
    to bytes (encode it)
    s.send(new_message) # Send the new message to the server
    print("Pesan Terkirim") # Print a message indicating that the new
    message has been sent

    # Receive a message from Client_4 and decode it from bytes to a
    string
    message = s.recv(1024)
    message = message.decode()

```

```
print("Client_4 : ", message) # Print the message received from  
Client_4
```

Client 4

```
import socket

s = socket.socket()
host = socket.gethostname()
port = 8080

s.connect((host, port))

print("Menyambungkan ke Server")

# Receive the welcome message from the server and decode it from bytes
to a string
message = s.recv(1024)
message = message.decode()
print("Pesan dari server : ", message) # Print the welcome message
from the server

while True: # Start an infinite loop to keep the client running
    # Receive a message from the server and decode it from bytes to a
    string
    message = s.recv(1024)
    message = message.decode()
    print("Server : ", message) # Print the message received from the
    server

    # Receive a message from Client_1 and decode it from bytes to a
    string
    message = s.recv(1024)
    message = message.decode()
    print("Client_1 : ", message) # Print the message received from
    Client_2

    # Receive a message from Client_2 and decode it from bytes to a
    string
    message = s.recv(1024)
    message = message.decode()
    print("Client_2 : ", message) # Print the message received from
    Client_3

    # Receive a message from Client_3 and decode it from bytes to a
    string
    message = s.recv(1024)
    message = message.decode()
    print("Client_3 : ", message) # Print the message received from
    Client_4
```



```

    new_message = input("Masukkan Pesan : ") # Prompt the client to
enter a new message
    new_message = str(new_message).encode() # Convert the new message
to bytes (encode it)
    s.send(new_message) # Send the new message to the server
    print("Pesan Terkirim") # Print a message indicating that the new
message has been sent

```

HASIL RUN SERVER

```

C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.22621.1992]
(c) Microsoft Corporation. All rights reserved.

C:\Users\emalia\Documents\progjar\modull1>python server.py
Waiting for connections...
Client_1 connected
Client_2 connected
Client_3 connected
Client_4 connected
Masukan Pesan : ini server
Pesan Terkirim
Client_1 : ini client1
Client_2 : ini client2
Client_3 : ini client3
Client_4 : ini client4
Masukan Pesan : absen lengkap
Pesan Terkirim

```

HASIL RUN CLIENT

<pre> C:\Windows\System32\cmd.exe Microsoft Windows [Version 10.0.22621.1992] (c) Microsoft Corporation. All rights reserved. C:\Users\emalia\Documents\progjar\modull1>python client1.py Menyambungkan ke Server Pesan dari server : Selamat datang di Server Server : ini server Masukkan Pesan : ini client1 Pesan Terkirim Client_2 : ini client2 Client_3 : ini client3 Client_4 : ini client4 Server : absen lengkap Masukkan Pesan : </pre>	<pre> C:\Windows\System32\cmd.exe Microsoft Windows [Version 10.0.22621.1992] (c) Microsoft Corporation. All rights reserved. C:\Users\emalia\Documents\progjar\modull1>python client2.py Menyambungkan ke Server Pesan dari server : Selamat datang di Server Server : ini server Client_1 : ini client1 Masukkan Pesan : ini client2 Pesan Terkirim Client_3 : ini client3 Client_4 : ini client4 Server : absen lengkap </pre>
<pre> C:\Windows\System32\cmd.exe Microsoft Windows [Version 10.0.22621.1992] (c) Microsoft Corporation. All rights reserved. C:\Users\emalia\Documents\progjar\modull1>python client3.py Menyambungkan ke Server Pesan dari server : Selamat datang di Server Server : ini server Client_1 : ini client1 Client_2 : ini client2 Masukkan Pesan : ini client3 Pesan Terkirim Client_4 : ini client4 Server : absen lengkap </pre>	<pre> C:\Windows\System32\cmd.exe Microsoft Windows [Version 10.0.22621.1992] (c) Microsoft Corporation. All rights reserved. C:\Users\emalia\Documents\progjar\modull1>python client4.py Menyambungkan ke Server Pesan dari server : Selamat datang di Server... Server : ini server Client_1 : ini client1 Client_2 : ini client2 Client_3 : ini client3 Masukkan Pesan : ini client4 Pesan Terkirim Server : absen lengkap </pre>