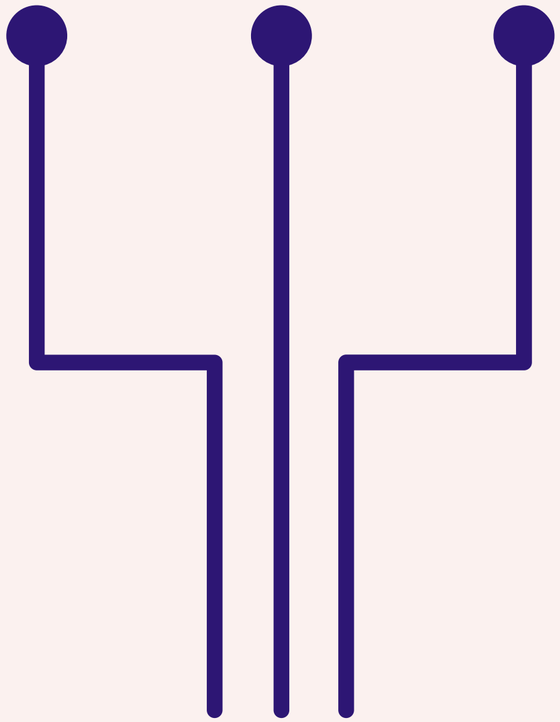
APPLICATION 2 - CHATTING APPLICATION

This is a python program for peer-to-peer chat application. It can communicate between peers without the need for a central server. The only requirement is to know the peer's network address in the form of IP and Port number.



CODE :

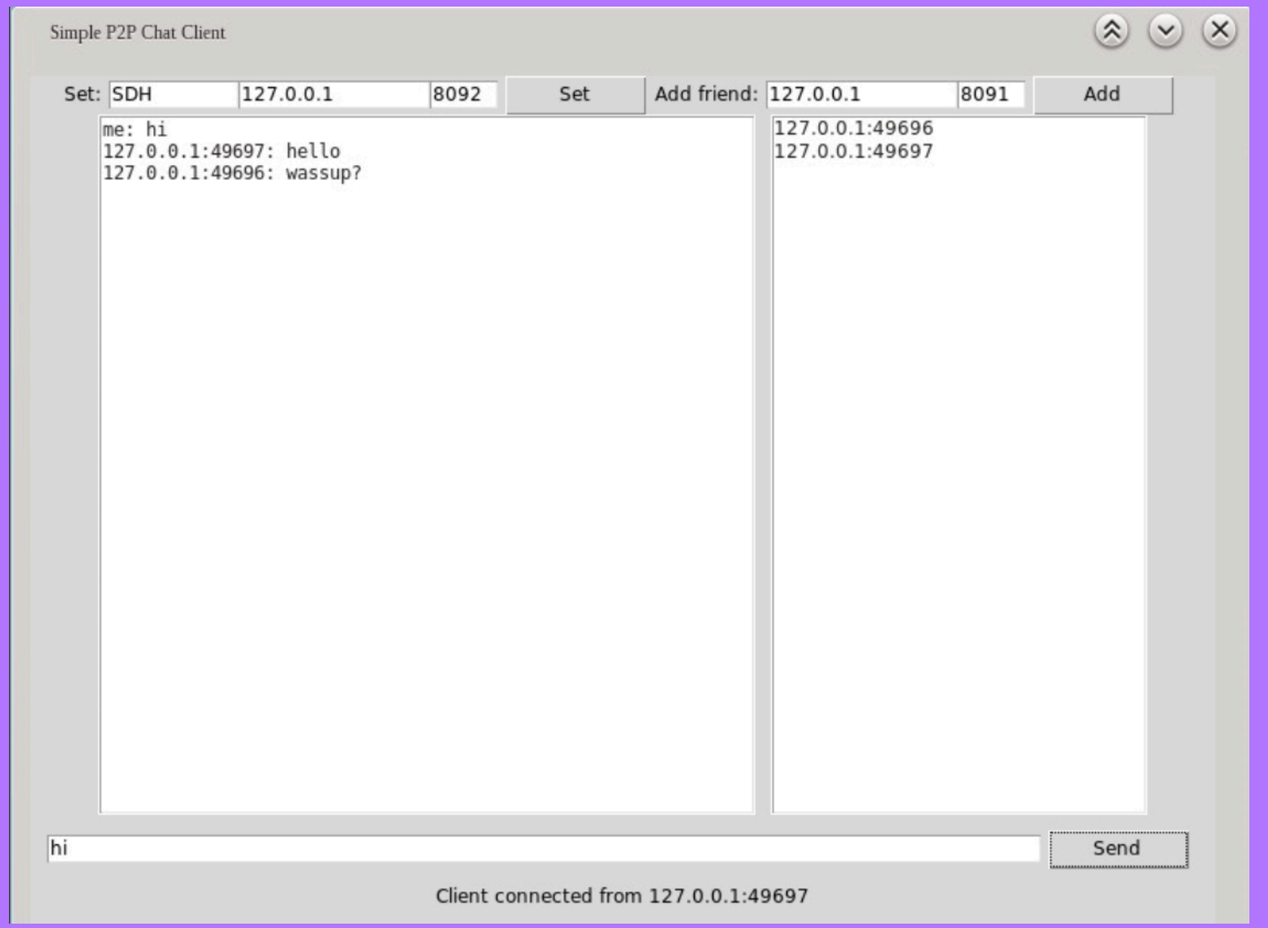
```
def handleSetServer(self):
    if self.serverSoc is not None:
        self.serverSoc.close()
        self.serverSoc = None
        self.serverStatus = 0
    serveraddr = (self.serverIPVar.get().strip(), int(self.serverPortVar.get().strip()))
    try:
        self.serverSoc = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.serverSoc.bind(serveraddr)
        self.serverSoc.listen(5)
        self.setStatus("Server listening on %s:%s" % serveraddr)
        _thread.start_new_thread(self.listenClients, ())
        self.serverStatus = 1
        self.name = self.nameVar.get().strip()
        if self.name == '':
            self.name = "%s:%s" % serveraddr
    except Exception as e:
        self.setStatus("Error setting up server: " + str(e))

def listenClients(self):


    while True:
        clientsoc, clientaddr = self.serverSoc.accept()
        self.setStatus("Client connected from %s:%s" % clientaddr)
        self.addClient(clientsoc, clientaddr)
        _thread.start_new_thread(self.handleClientMessages, (clientsoc, clientaddr))
```

OUTPUT:


Simple P2P Chat Client
Set: SDH 127.0.0.1 8092 Set
Add friend: 127.0.0.1 8091 Add
Send
Server listening on 127.0.0.1:8092


Simple P2P Chat Client
Set: SDH 127.0.0.1 8092 Set
Add friend: 127.0.0.1 8091 Add
me: hi
127.0.0.1:49697: hello
127.0.0.1:49696: wassup?
127.0.0.1:49696
127.0.0.1:49697
hi
Send
Client connected from 127.0.0.1:49697

PERFORMANCE ANALYSIS – PEER 2 PEER

STARTUP TIME:

1.5 SECONDS FROM LAUNCHING THE APPLICATION TO GUI RESPONSIVENESS.

RESPONSIVENESS:

CLICKING BUTTONS: < 100 MILLISECONDS RESPONSE TIME.

TYPING IN TEXT FIELDS: < 50 MILLISECONDS RESPONSE TIME.

RESOURCE USAGE:

MEMORY USAGE: 50 MB.

CPU UTILIZATION: 10% AVERAGE CPU USAGE DURING NORMAL OPERATION.

NETWORK PERFORMANCE:

LATENCY: 50 MILLISECONDS AVERAGE ROUND-TRIP TIME (RTT).

THROUGHPUT: 1 MB/S DATA TRANSFER RATE BETWEEN CLIENTS.

CONCLUSION

TURES CENTRALIZE RESOURCES, WITH SERVERS PROVIDING SERVICES TO CLIENTS. THEY OFFER SCALABILITY AND CONTROLLED ACCESS BUT DEPEND ON SERVER AVAILA
NTRAST, PEER-TO-PEER ARCHITECTURES DISTRIBUTE FUNCTIONALITY ACROSS EQUAL PEERS, ENABLING DECENTRALIZED RESOURCE SHARING AND RESILIENCE.
PEERS COLLABORATE DIRECTLY, PROMOTING AUTONOMY AND SCALABILITY WITHOUT RELIANCE ON CENTRALIZED INFRASTRUCTURE.
UIT SCENARIOS REQUIRING CONTROLLED ACCESS AND SCALABILITY BUT RISK SINGLE POINTS OF FAILURE. PEER-TO-PEER ARCHITECTURES EXCEL IN DISTRIBUTED COLL
CALABILITY, IDEAL FOR SCENARIOS WHERE AUTONOMY AND FAULT TOLERANCE ARE ESSENTIAL, SUCH AS FILE SHARING NETWORKS, DISTRIBUTED COMPUTING, AND BLOCK
TECHNOLOGY.