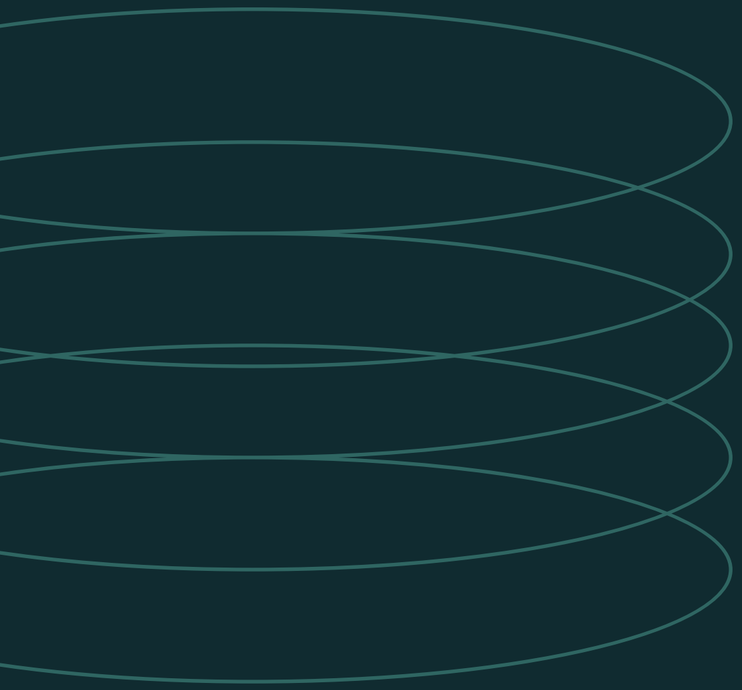
# A P2P application with Python

how to make a P2P network

# Key Discussion Points:

## Understand how this new network works

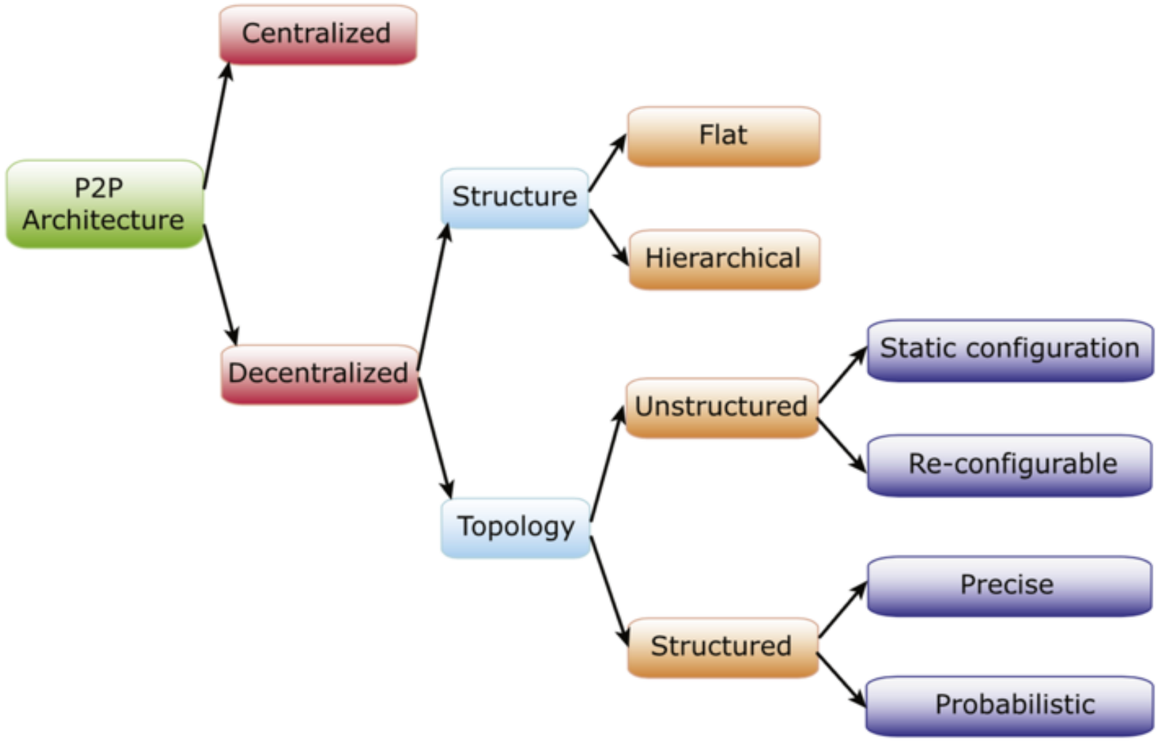
- What was the problem?
- Types of P2P networks
- Centralized vs. Decentralized model
- How to implement the code
- Provide solutions for network congestion time



# problem:

We intend to design a network in which each user is all our customers and server. This grid is to be used to multiply large matrices.

# P2P network
# Types of P2P networks



# Decentralized

- This architecture has more limitations than centralized architecture. In this architecture, there is no server to manage peers, and each peer with the information it has from a limited part of the network and its neighboring peers can exchange information with those parts. The most important issue in this architecture is its more limited resources compared to other models.

# Centralized

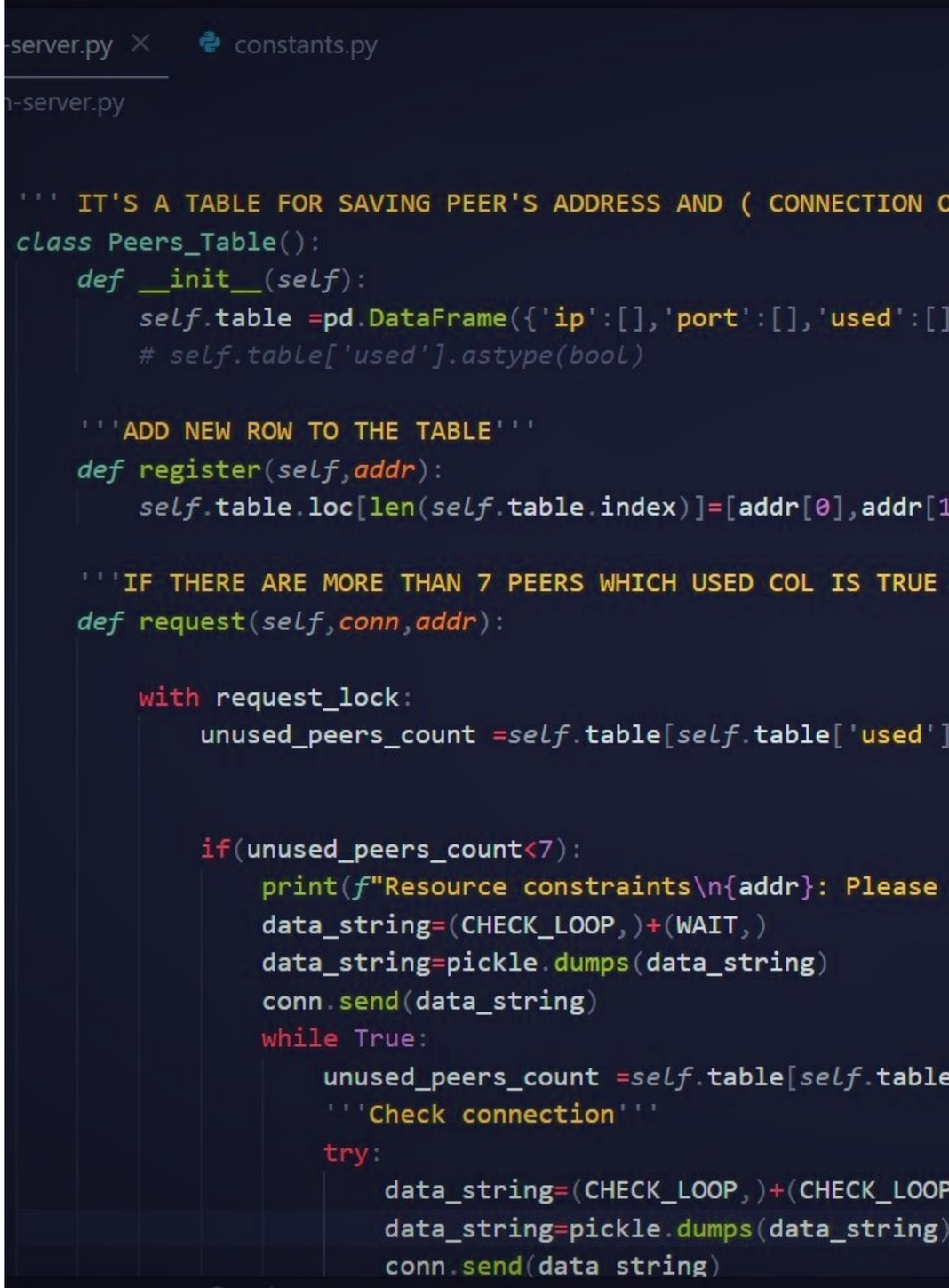
- In a P2P network with a centralized architecture, the main server stores the information of each peer in a table. This architecture is very similar to the client-server architecture, but with the difference that each peer can independently communicate with the list of other peers, it receives from the main server and exchange information with each other. It should be noted that this architecture is very scalable and the main server is responsible for managing peers and the number of threads created on each peer.

# How to implement

## Main server

To implement the requested network in the given problem, we designed the main server. When connecting to the peer, this main server uses the Data Frame provided by the pandas library, the information of that peer including IP number and port, the number of times allowed to send data simultaneously on each peer, and object connection in this table saves.

We have designed various queries for this main server through which we can request that the server send seven peer communication information to the requester or release the bound peers.



# How to implement

## Peer

- Each peer must be both a server and a client at the same time.
- It waits to receive information from the main server through the server_recving function, and it waits to receive information from other peers through the server_isListening function. Each peer can request the main server to send information to the other seven peers to communicate with them. By connecting each peer, a thread of the peer_handler function is created.
- The input_handler function executes user queries, which are:
- I need seven peers, release these peers, make matrices,
- multiply matrices
-
- The Strassen function is also responsible for breaking the desired matrix into smaller matrices and sending each piece of the matrix to a peer.

peer.py

```
peer.py
95  def server_isListening():
96      server_socket=socket.socket(socket.AF_INET, socket.SO
97      config=(server.ip,server.port,)
98      server_socket.bind(config)
99      print(f"[LISTENING] peer's server side is listening o
100     print('\n----------> \n')
101
102     server_socket.listen()
103
104     while True:
105         try:
106             conn,addr=server_socket.accept()
107             print(addr)
108             peer_handler_thread=threading.Thread(target=p
109             peer_handler_thread.start()
110             print(f"[Active connection] peer's server sid
111
112         except:
113             print(f"[LISTENING] peer's server side is not
114             server_socket.close()
115             break
116
117
118 def peer_handler(conn,addr):
119     import pickle
120     global limited_flag
121     # peers=None
122     print(f"[NEW CONNECTION] peer's client side '{addr}'
123     connected=True
124     while connected:
```

# Provide solutions for network congestion time

In this architecture, network congestion management is the responsibility of the main server, one of the limitations that can be considered for this network is that each peer can play the role of server a limited number of times, and if the peer requests a list from peer And all those peers are being used as much as the intended restriction, the requesting peer goes into Suspended state until the other peers are released.

# Provide solutions for network congestion time

However, this method alone does not work, because if there is a problem in releasing the requested peers, then the other requesting peer may enter a dull state, so through the round-robin algorithm, the amount of time that each peer can be bound. We specify that the algorithm for using the table on the main server is first to come first out, and when a request is made to the server, other queries are prevented from entering through a semaphore lock until that query is completed. Interference that may occur in the table or the Data Frame is prevented.

# Free Resources

https://medium.com

https://pandas.pydata.com

https://stackoverflow.com

https://realpython.com