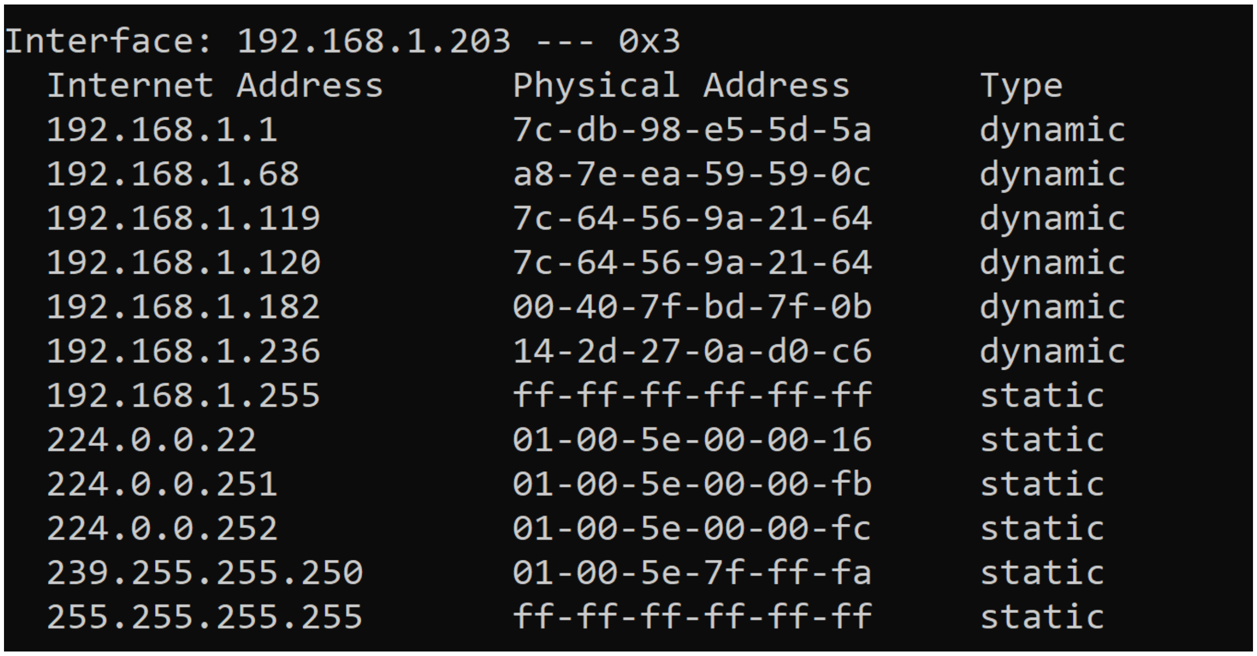
CECS 327 Report

We start by finding the local IP addresses of the devices in your network. This is done by using the following command lines:

1. ipconfig
2. arp -a



```
Interface: 192.168.1.203 --- 0x3
  Internet Address      Physical Address      Type
  192.168.1.1           7c-db-98-e5-5d-5a     dynamic
  192.168.1.68          a8-7e-ea-59-59-0c     dynamic
  192.168.1.119         7c-64-56-9a-21-64     dynamic
  192.168.1.120         7c-64-56-9a-21-64     dynamic
  192.168.1.182         00-40-7f-bd-7f-0b     dynamic
  192.168.1.236         14-2d-27-0a-d0-c6     dynamic
  192.168.1.255         ff-ff-ff-ff-ff-ff     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.251           01-00-5e-00-00-fb     static
  224.0.0.252           01-00-5e-00-00-fc     static
  239.255.255.250       01-00-5e-7f-ff-fa     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static
```

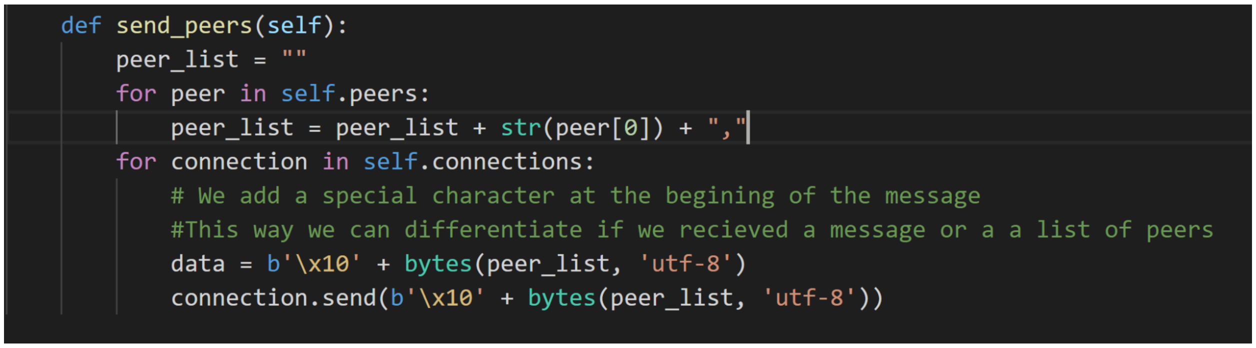
The network works as follows:

The **p2p.py** file is our main program. We keep looping over the list of peers (local ip addresses) and make all the peers clients and then make itself the server also.

In the **server.py** file, to create a connection, we define a socket and bind it to the host and a port number. We will use the same port number across the connections (12345). The server also has a list of peers to keep track of the peers that it is connected to.

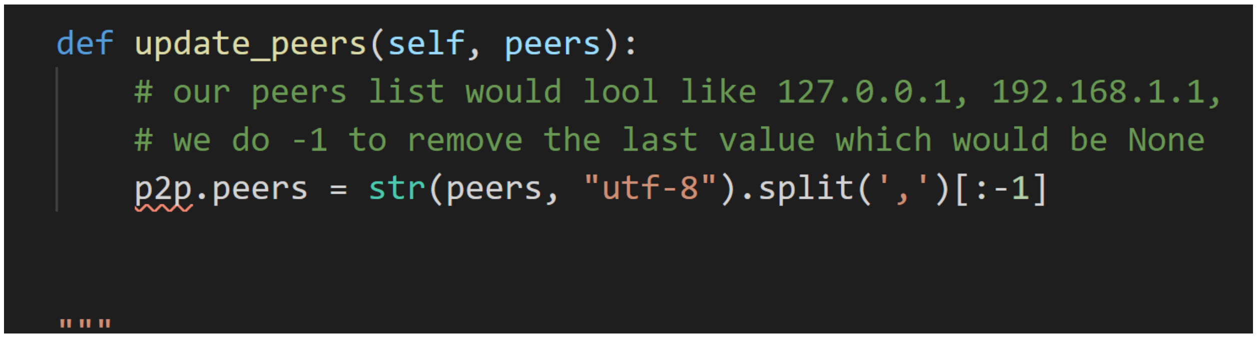
How to detect that a new peer is coming and update the list of peers ?

While the server is listening to connections, if a connection is accepted, the server will broadcast a message that contains a list of peers to its neighbors. We assume that in our network, once a peer joins the network, it will stay connected through the timeline of the program. Therefore, a coming connection always signals a new peer. The function send_peer takes care of this.


```
def send_peers(self):
    peer_list = ""
    for peer in self.peers:
        peer_list = peer_list + str(peer[0]) + ","
    for connection in self.connections:
        # We add a special character at the begining of the message
        #This way we can differentiate if we recieved a message or a a list of peers
        data = b'\x10' + bytes(peer_list, 'utf-8')
        connection.send(b'\x10' + bytes(peer_list, 'utf-8'))
```

To signal that this is a list-of-peer message, we will concatenate a special character at the beginning of our message before sending it to clients.

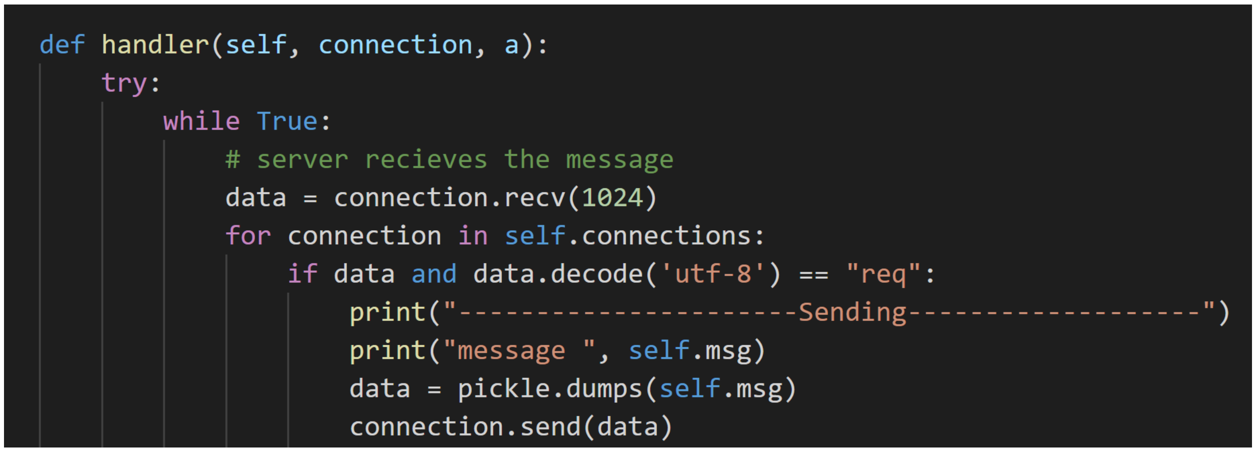
On the client side, once it receives a message that has a special character at the beginning, it will update its list of peers.


```
def update_peers(self, peers):
    # our peers list would lool like 127.0.0.1, 192.168.1.1,
    # we do -1 to remove the last value which would be None
    p2p.peers = str(peers, "utf-8").split(',')[:-1]


"""
```

By doing this, we make sure that each node is informed and updated the list of new coming peers properly.

Upon receiving the requesting message from the clients, the server will send over the files in a directory as a list of messages. The function handler takes care of this



```
def handler(self, connection, a):
    try:
        while True:
            # server recieves the message
            data = connection.recv(1024)
            for connection in self.connections:
                if data and data.decode('utf-8') == "req":
                    print("----------------------Sending-------------------")
                    print("message ", self.msg)
                    data = pickle.dumps(self.msg)
                    connection.send(data)
```

In **client.py**, the client class deals with setting up a connection to the socket so that it can send messages and receive messages from the server. The client sends a request message to the server, asking for files transfer. The receive_message function takes care of receiving the files from the server and saving them.

Unfortunately, our nodes, one is stuck in the server mode and the other is stuck in the client node. Therefore, currently the file transferring is one direction. We couldn’t figure out how to resolve this problem.