درس : شبکه های کامپیوتری

استاد: صادقیان

دانشجو: مهدی منصوری خواه

شماره دانشجویی: 9931056

گزارش پروژه

معماری peer to

به بیان ساده، یک شبکه همتا به همتا (P2P) هنگامی تشکیل می‌شود که دو یا چند دستگاه که معمولا کامپیوتر هستند، به یکدیگر متصل بوده و منابع و داده‌های مختلف را با هم به اشتراک می‌گذارند.
به عبارت دیگر می‌توان گفت همتا به همتا (P2P) یا Peer-to-Peer یک مدل ارتباطاتی و اشتراک‌گذاری فایل غیرمتمرکز است که برخلاف مدل کلاینت/سرور که در آن کلاینت تقاضای سرویس می‌کند و سرور تقاضا را انجام می‌دهد، به هر کاربر یا نود اجازه می‌دهد که هم به عنوان کلاینت و هم به عنوان سرور عمل کند.

حال به توضیح کد می پردازیم:
همانطور که در دستور کار گفته شد ما یک سرویس مدیریت آدرس همتاها داریم که وظیفه دارد اطلاعات هر peer رو در خود(redis) ذخیره کند و اطلاعات لازم برای وصل شدن هر peer به peer دیگر را به کاربر مورد نظر بدهد.
پیاده سازی آن به صورت زیر است:

```python
redis_host = 'localhost'
redis_port = 6379

redis_query = redis.StrictRedis(host=redis_host, port=redis_port, decode_responses=True)


mahdimnkh81
class EchoHandeler(BaseHTTPRequestHandler):
    mahdimnkh81
    def do_Post(self):
        print("Post")
        if self.path.endswith("/info"):
            content_length = int(self.headers['Content-type'])
            post_data = self.rfile.read(content_length)
            x = json.loads(post_data.decode())
            id = list(x.keys())[0]
            port = list(x.keys())[1]
            hostname = list(x.keys())[2]
            ip_client = x[id]
            port_client = x[port]
            hostname_client = x[hostname]
            redis_query.rpush(id, ip_client, port_client, hostname_client)
            redis_query.rpush('ids', id)
```

ابتدا redis را بالا می آوریم(روی پورت 6379) سپس یک متد post تعریف می کنیم که اطلاعات peer را دریافت میکند(اعم از هدر و body) سپس id,port,hostname آن را در redis ذخیره می کنیم(ارسال مشخصات همتا)

در اینجا یک متد get تعریف کردیم میاد لیست تمام peer های موجود را از redis میگیرد و به کاربر پیامی می فرستد که میخوای به کدام peer متصل شی. همچنین اگر کاربر Request مناسب نفرستد با پیام 404 مواجه می شود.

```python
def do_Post(self):
    if self.path.endswith("/choseClient"):
        print("/choseClient")
        content_length = int(self.headers['Content-type'])
        post_data = self.rfile.read(content_length)
        x = json.loads(post_data.decode())
        id = list(x.keys())[0]
        id_client = x[id]
        info_client = redis_query.lrange(id_client, 0, -1)
        print(info_client)
        try:
            # send code 200 response
            self.send_response(200, message="The information required to connect to the system")

            # send header first
            # self.send_header(ids, 'text-html')
            # self.send_response(ids)
            json_info_client = json.dumps(info_client)
            self.send_header('infoClient', json_info_client)
            self.end_headers()

        except IOError:
            self.send_error(404, 'file not found')
```

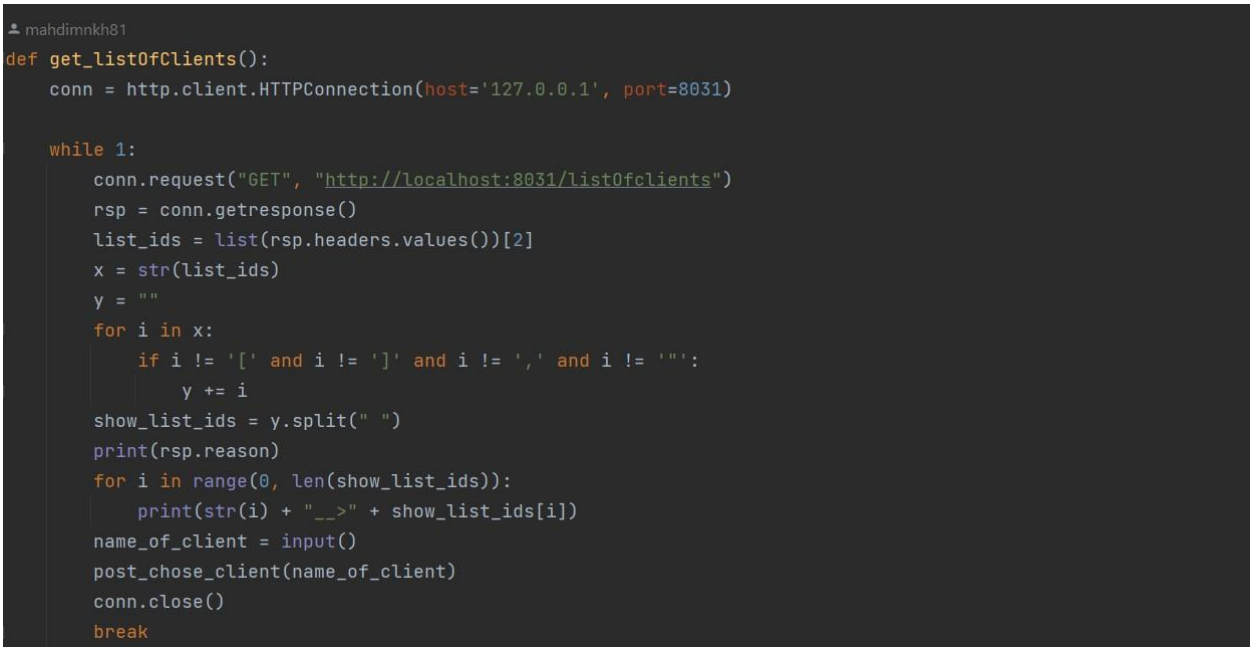
در اینجا peer،کاریری را که انتخاب کرده است را با متد post میفرستد و STUN ، اطلاعات مورد نیاز برای وصل شدن را برای peer میفرستد.کد خیلی واضح است اما برای توضیحش می توان گفت ابتدا id طرف رو از درخواست کاربر پیدا میکنیم. سپس در redis به دنبال اطلاعات آن کاربر میگیریم و در آخر اطلاعات را ارسال می کنیم.
لازم به ذکر است تمام اتصالات به صورت http بوده است و اطلاعات به صورت json ردوبدل می شود.

حال به سراغ پیاده سازی peer می رویم:

هر کار(task) در یک متد جداگانه نوشته شده است تا خوانایی کد بیشتر شود.

```
mahdimnkh81
def post_information():
    con = http.client.HTTPConnection(host='127.0.0.1', port=8031)
    id = input("ENTER YOUR NAME!!")
    ids = r.lrange('ids', 0, -1)
    flage = True
    while (flage):
        if id in ids:
            print("This name has already been chosen")
            print("Choose another name")
            id = input("ENTER YOUR NAME!!")
        else:
            flage = False
    body = {id: ip_address, "port": PORT, "hostname": hostname}
    json_body = json.dumps(body)
    headers = {'Content-type': f'{len(json_body)}'}
    con.request("Post", "http://localhost:8031/info", body=json_body, headers=headers)
    con.close()
```

در اینجا peer، اطلاعات خود را به سمت سرور می فرستد این اطلاعات از قبیل نام کاربری،پورت،hostname می باشد و همچنین چک میکند که کاربر نام تکراری وارد نکند(با یک for در redis)



در اینجا کاربر با استفاده از متد get درخواست خود را می فرستد ولیستی از تمام نام های کاربران دریافت می کند و در آخر یکی از آن ها را انتخاب می کند.(اون for به این علت است که اطلاعات رو به صورت قابل فهم برای کاربر بفرستد)

```
mahdimnkh81
def post_chose_client(id):
    conn = http.client.HTTPConnection(host='127.0.0.1', port=8031)
    body = {"name": id}
    json_body = json.dumps(body)
    headers = {'Content-type': f'{len(json_body)}'}
    conn.request("Post", "http://localhost:8031/choseClient", body=json_body, headers=headers)
    rsp = conn.getresponse()
    info = list(rsp.headers.values())[2]
    x = str(info)
    y = ""
    for i in x:
        if i != '[' and i != ']' and i != ',' and i != '"':
            y += i
    show_list_info = y.split(" ")
    print(rsp.reason)
    print("ipAddress" + "__>" + show_list_info[0])
    print("port" + "__>" + show_list_info[1])
    print("hostname" + "__>" + show_list_info[2])
    connect_to_client(show_list_info[2], show_list_info[1], show_list_info[0])
    conn.close()
```

کابر نام peer انتخابی خود در body با متد post به STUN می فرستد و اطلاعات مورد نیاز برای اتصال، را دریافت میکند.

```
mahdimnkh81 *
def connect_to_client(hostname, port, ipaddress):
    question = input("Do you want to give or receive services?")
    if question == "give":
        p = input("Do you want give text or image?")
        if p == "text":
            server_TCP_text()
        else:
            server_UDP_image()
    else:
        result = input("Do you want send text or image?")
        if result == 'text':
          client_TCP_text(hostname, port, ipaddress)
        else:
          client_UDP_image(hostname, port, ipaddress)
```

دراینجا از کاربر سوال می شود که میخوای سرویس بدی یا سرویس بگیری و نوع سرویس رو مشخص کن.

```
mahdimnkh81 *
def server_TCP_text():
    new *
    def handle_client(client_socket):
        with client_socket as sock:
            time.sleep(10)
            request = sock.recv(1024)
            # print(f'[*] Received: {request.decode("utf-8")}')
            sock.send(b'ACK')
            sock.close()

    host = socket.gethostname()
    port = 5000
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind((host, port))
    server.listen(2)
    print(f'[*] Listening on {host}:{port}')

    while True:
        client, address = server.accept()

        print(f'[*] Accepted connection from {address[0]}:{address[1]}')
        client_handler = threading.Thread(target=handle_client, args=(client,))
        client_handler.start()
```

این متد برای برای دریافت text می باشد که از نوع tcp می باشد و اگر اطلاعات به صورت صحیح و کامل دریافت شده باشد ACK میفرستد برای مشخص شدن این موضوع یک time.sleep(10) گذاشتم که در ارائه حضوری بیشتر توضیح می دم.(بقیه خط ها نحو وصل شذن را نشان می دهد)

```python
def server_UDP_image():
    localIP = "127.0.0.1"
    localPort = 20051
    bufferSize = 16
    msgFromServer = "ACK"
    bytesToSend = str.encode(msgFromServer)
    UDPServerSocket = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)
    UDPServerSocket.bind((localIP, localPort))
    print("UDP server up and listening")
    recv = bytearray()
    x = []
    bytesAddressPair = UDPServerSocket.recvfrom(1024)
    address = bytesAddressPair[1]
    while recv != b'finish':
        recv = UDPServerSocket.recv(1024)
        array_recv = list(map(int, recv))
        x.append(array_recv)
        UDPServerSocket.sendto(bytesToSend, address)

    two_dim = []
    three_dim = []
    z = []
    c = 0
    for i in x:
        for j in i:
            if j != 0:
                z.append(j)
        # print(z)
```

```python
        two_dim.append(z)
        z = []
        c += 1
        if c == bufferSize:
            three_dim.append(two_dim)
            two_dim = []
            c = 0
    # print(three_dim)
    produce_image = np.array(three_dim, dtype=np.uint8)
    # print(type(final))
    pilImage = Image.fromarray(produce_image)
    pilImage.show()
```

این متد برای دریافت عکس می باشد و تا زمانی که finish دریافت نکند،منتظر دریافت packet است. درارسال با تعداد صفر ها در آخر آرایه مشخص میکنم که بسته چندم دریافت شده و ترتیب چک می شود(در ارائه حضوری بیشتر توضیح می دم.) و در آخر عکس را نمایش می دهیم.

```
mahdimnkh81 *
def client_TCP_text(hostname, port, ipaddress):
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect((hostname, int(port)))
    text = input("Enter your text")
    print("send...")
    s.send(text.encode("utf-8"))
    flag = ''
    while flag != "ACK":
        ready = select.select([s], [], [], 2)
        if ready[0]:
            response = s.recv(4096)
            flag = response.decode('utf-8')
            print(flag)
        else:
            print("timeout")
            print("send again...")
            s.send(text.encode("utf-8"))

    s.close()
```

در اینجا text به صورت اتصال tcp فرستاده می شود و تا زمانی که ACK دریافت نکند، timeout می شود و پیام خود را می فرستد.

```
def client_UDP_image(hostname, port, ipaddress):
    msgFromClient = input("Enter image")
    bytesToSend = str.encode(msgFromClient)
    serverAddressPort = ("127.0.0.1", int(port))
    bufferSize = 1024
    s = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)
    s.sendto(bytesToSend, serverAddressPort)
    array_image = array(msgFromClient)
    for i in array_image:
        k = 0
        for j in i:
            bytes_of_image = bytes(j) + bytes(k)
            k += 1
            s.sendto(bytes_of_image, serverAddressPort)
            msgFromServer = s.recvfrom(bufferSize)
            print(msgFromServer[0])
    s.sendto(bytes("finish".encode("utf-8")), serverAddressPort)
```

در اینجا عکس به صورت اتصال UDP فرستاده می شود، ابتدا عکس به ماتریس تبدیل می شود سپس به بایت و همچنین شماره بسته بهش چسبیده می شود و فرستاده می شود و زمانی که تموم شد پیام finish خود را می فرستد.

موفق باشید😊