

# Socket programming with Python

Socket là cánh cổng ngăn cách giữa Application Layer với Transport Layer.

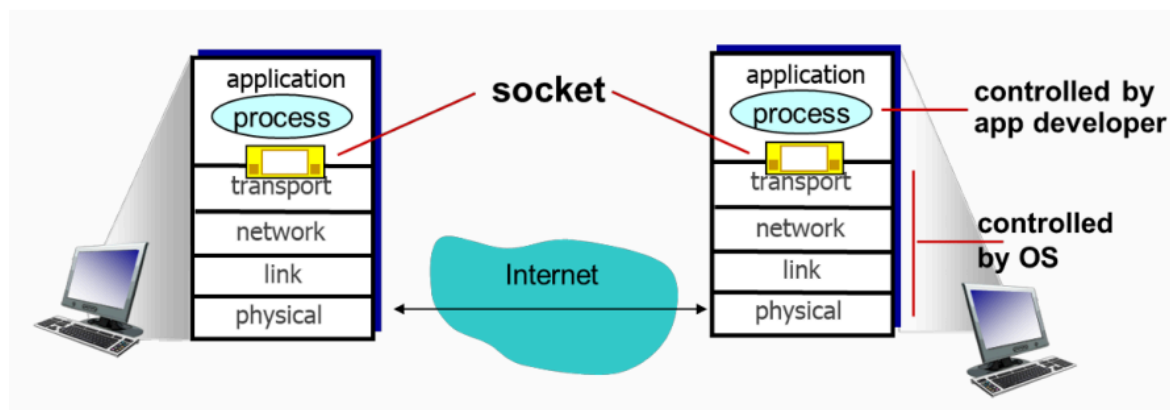


Figure 1: Mô hình socket

Thư viện socket của Python hỗ trợ một số API sau để thiết lập kết nối socket TCP/UDP:

- `socket()`
- `bind()`
- `listen()`
- `accept()`
- `connect()`
- `connect_ex()`
- `send()`
- `recv()`
- `close()`

Thư viện socket của Python hỗ trợ cả TCP socket lẫn UDP socket. Trong project này, nhóm dự định sử dụng TCP socket để hiện thực dự án.

Flow của một TCP socket connection có thể được thể hiện như hình dưới đây:

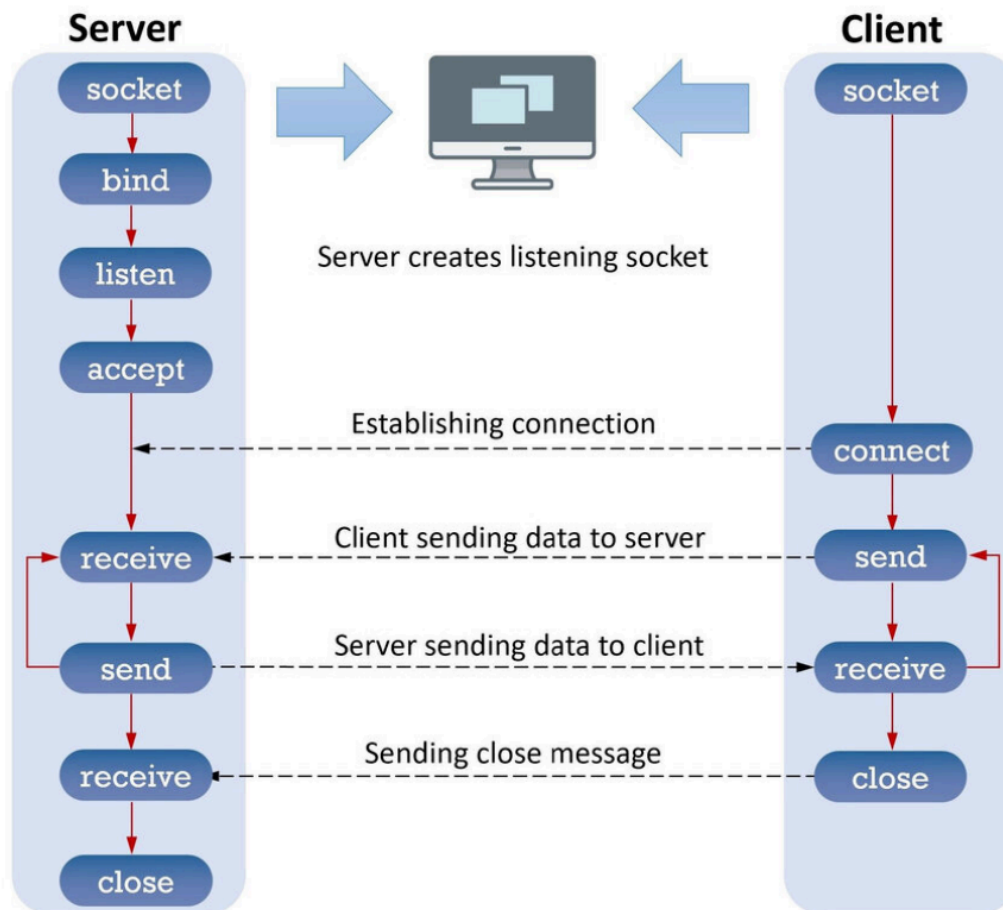


Figure 2: Quy trình một kết nối socket để gửi nhận dữ liệu (Nguồn ResearchGate)

# Application design

## Architecture design

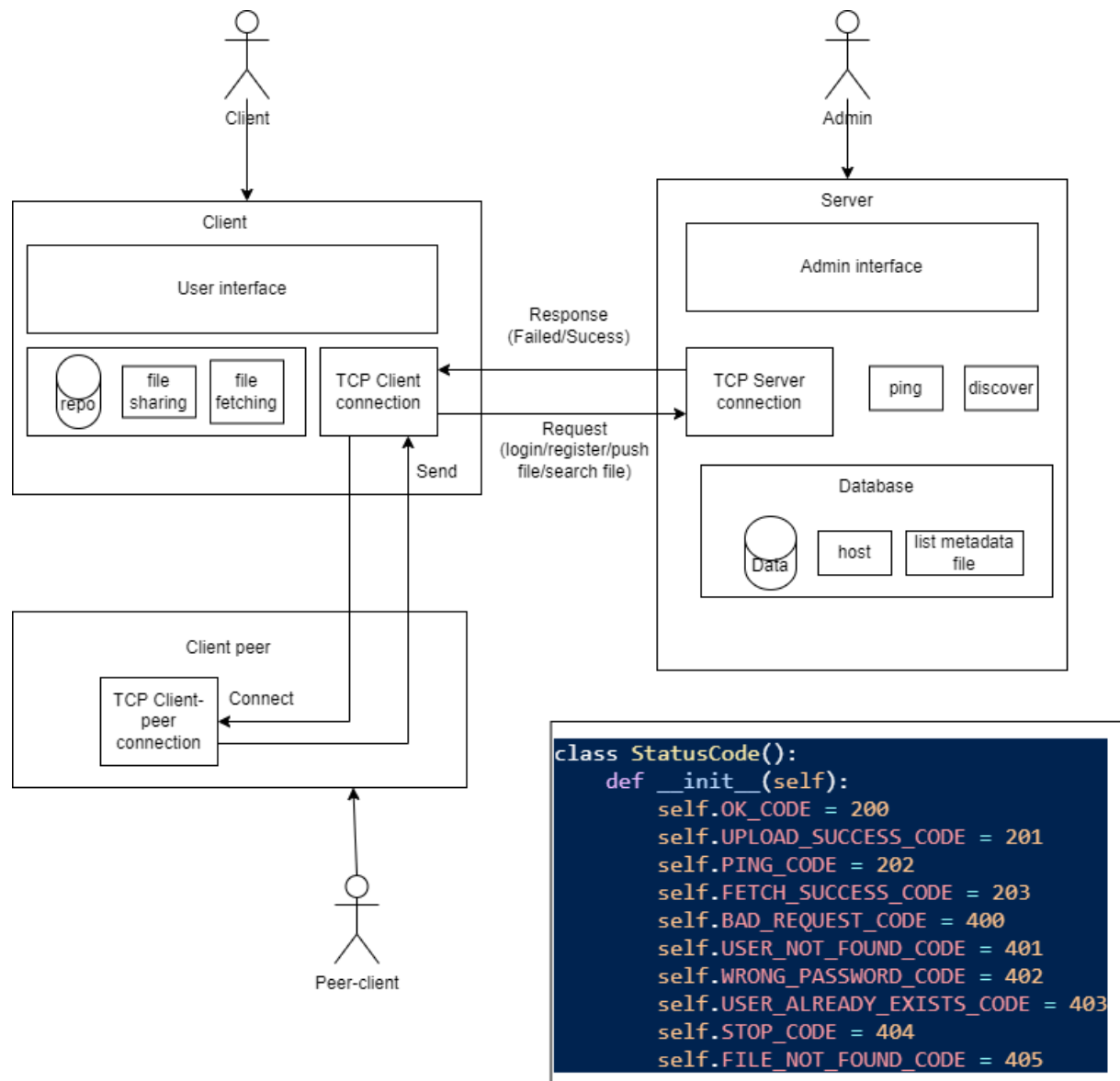


Figure 3: Kiến trúc tổng quan của hệ thống

## Flow protocol design

### Register

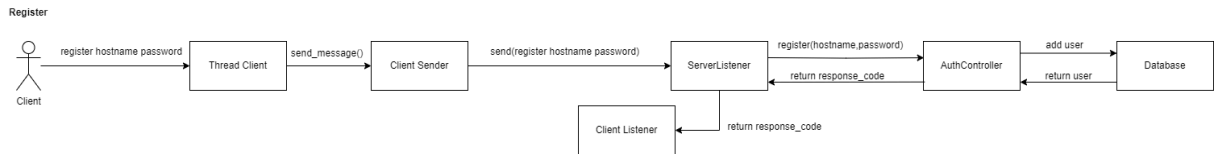


Figure 4: Sơ đồ luồng của chức năng đăng ký

### Login

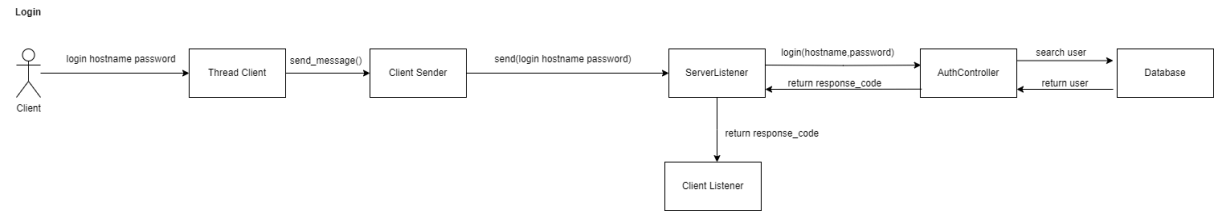


Figure 5: Sơ đồ luồng của chức năng đăng nhập

### Ping

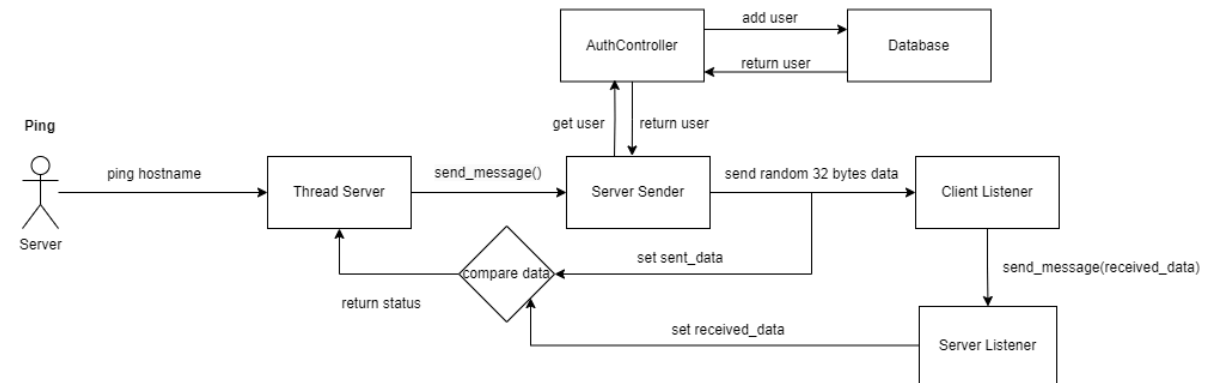


Figure 6: Sơ đồ luồng của chức năng ping

### Discover

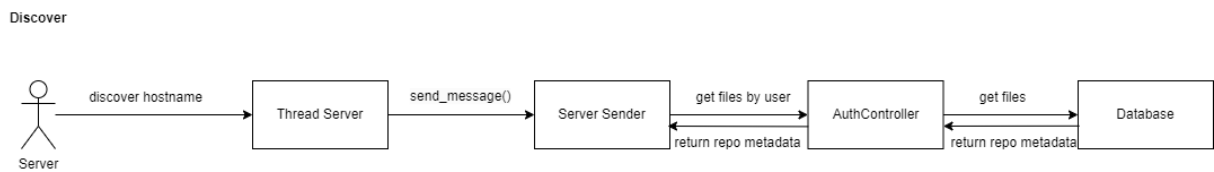


Figure 7: Sơ đồ luồng của chức năng discover

## Publish

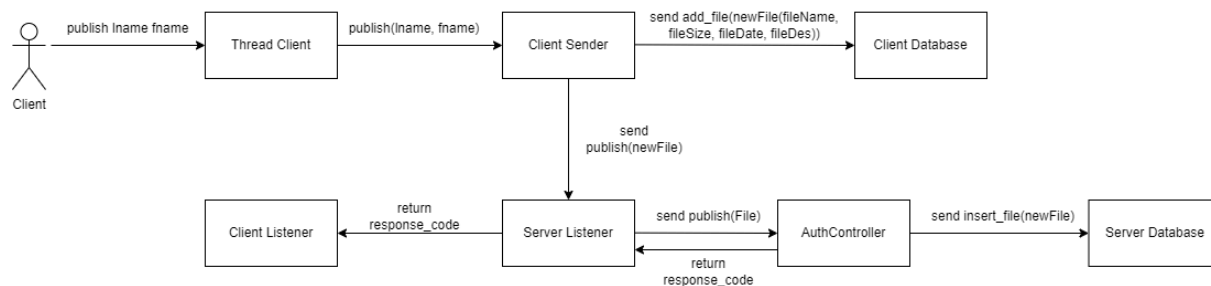


Figure 8: Sơ đồ luồng của chức năng publish

## Fetch

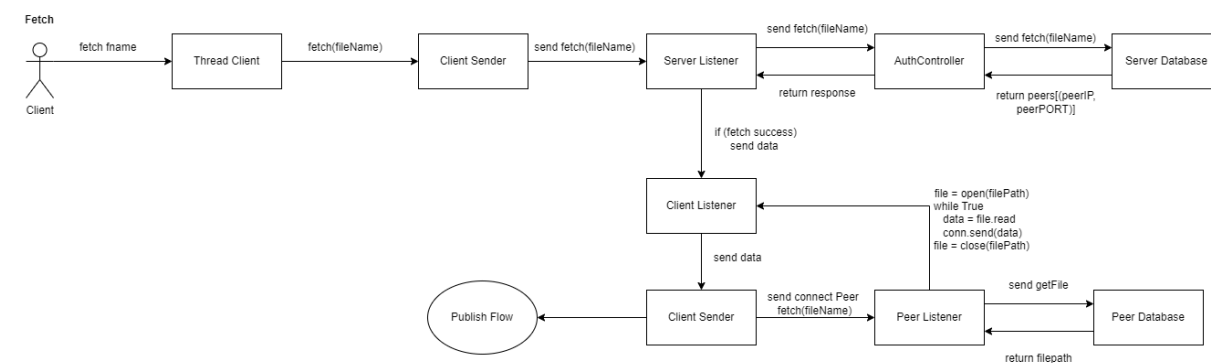


Figure 9: Sơ đồ luồng của chức năng fetch

## Activity diagram

### Register

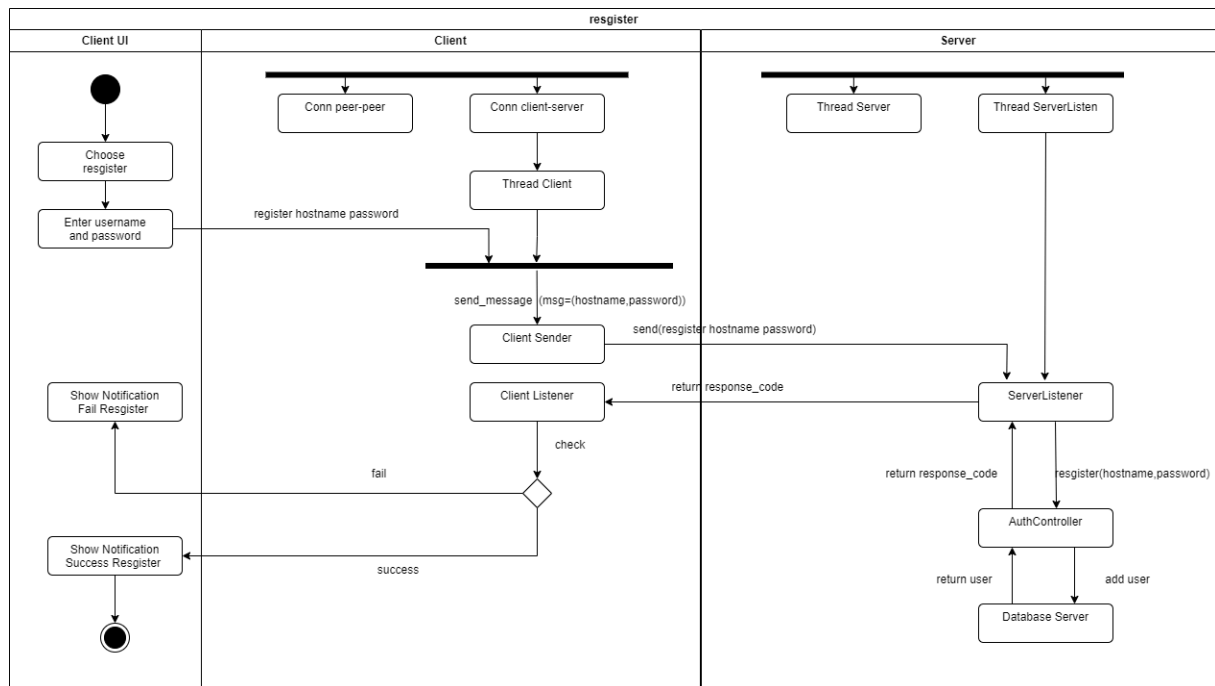


Figure 10: Sơ đồ hoạt động của chức năng đăng ký

### Login

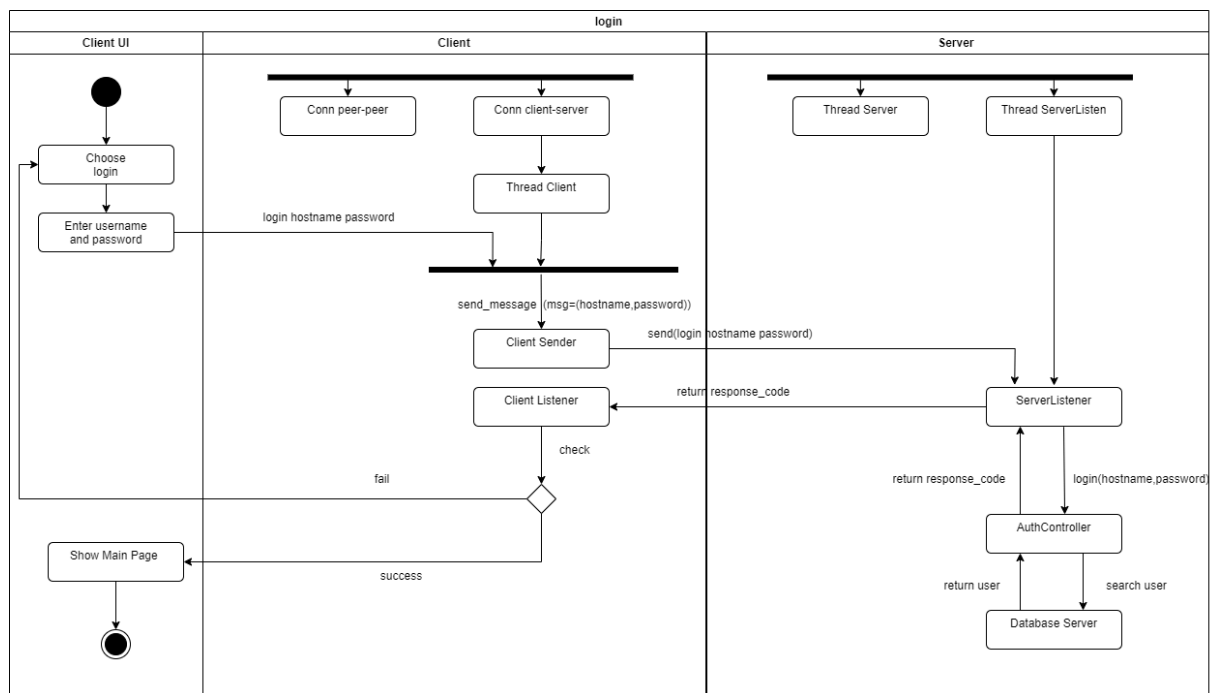


Figure 11: Sơ đồ hoạt động của chức năng đăng nhập

### Ping

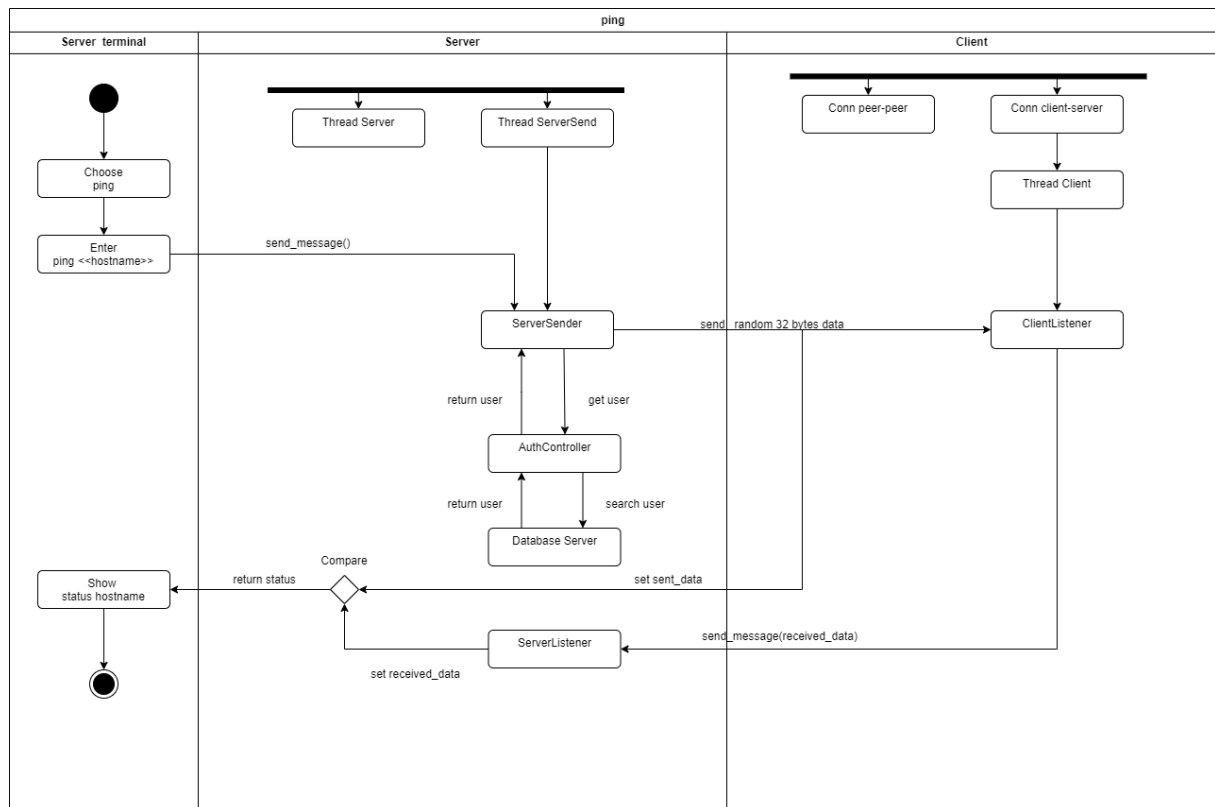


Figure 12: Sơ đồ hoạt động của chức năng ping

## Discover

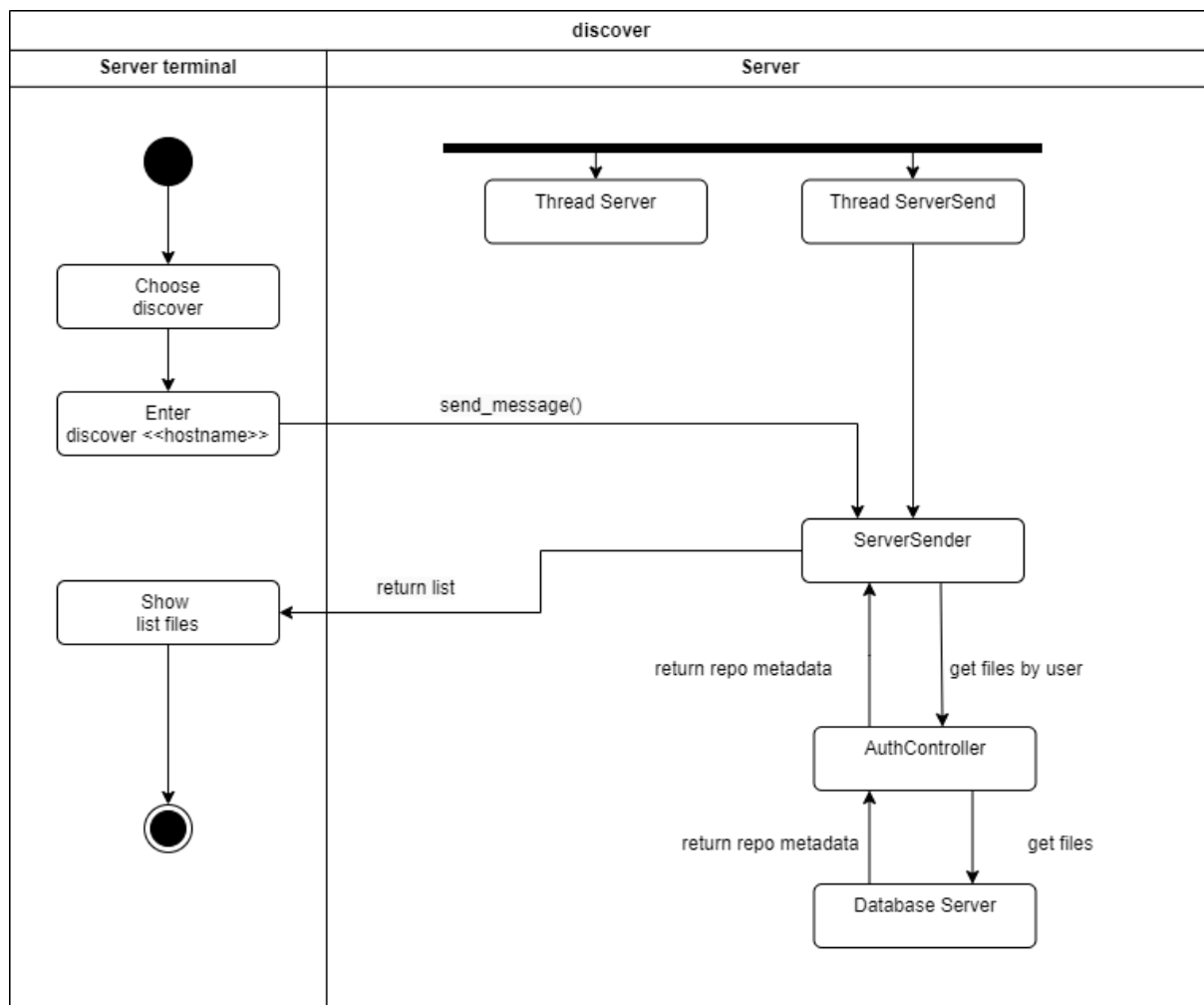


Figure 13: Sơ đồ hoạt động của chức năng discover



## Publish

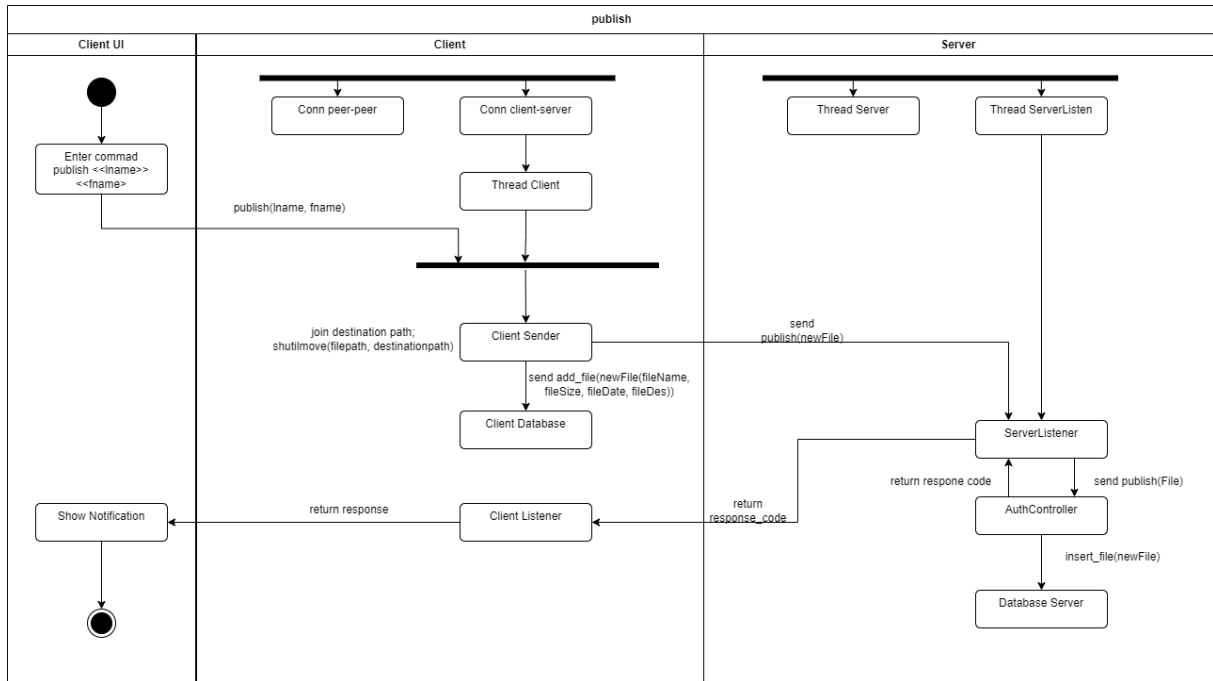


Figure 14: Sơ đồ hoạt động của chức năng publish

## Fetch

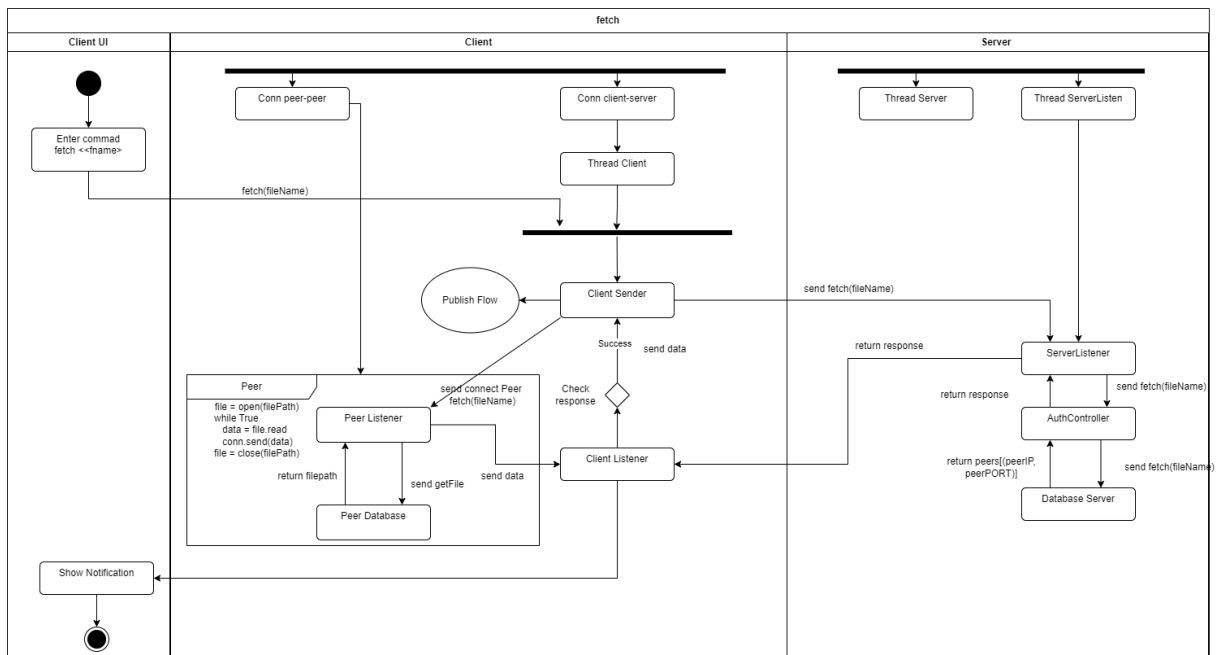


Figure 15: Sơ đồ hoạt động của chức năng fetch

## Status Code

Ngoài ra, nhóm còn định nghĩa thêm một số status code để phục vụ cho việc trao đổi thông tin giữa server đến client và ngược lại.

Status Code	Mô tả
200	Đăng nhập thành công
201	Đăng ký thành công
202	Publish thành công
203	Server muốn ping client
204	Fetch thành công
205	Download file thành công
400	Yêu cầu không hợp lệ
401	Không tìm thấy tài khoản
402	Sai mật khẩu
403	Tài khoản đã đăng nhập
404	Tài khoản đã tồn tại
405	Client thoát khỏi server
406	File không tồn tại

Table 1: Các status code được định nghĩa

```
client.py ClientListener.py
44 > init__(self, client):...
47 > t_fetch_peers(self):...
49 > art(self):...
52 sten(self):
53 y:
54 while self.running:
55     response_code = self.client.recv(3).decode('utf-8')
56     if response_code: # Connection closed by the server
57         print(f"Received response code: {response_code}")
58         if response_code == self.statuscode.LOGIN_SUCCESS():
59             print("login successful.")
60             self.success = True
61             self.notifications.append((datetime.now().strftime("%H
62 elif response_code == self.statuscode.REGISTER_SUCCESS():
63             print("Registration successful.")
64             self.notifications.append((datetime.now().strftime("%H
65 elif response_code == self.statuscode.USER_NOT_FOUND():
66             print("User not found.")
67             self.notifications.append((datetime.now().strftime("%H
68 elif response_code == self.statuscode.WRONG_PASSWORD():
69             print("Wrong password.")
70             self.notifications.append((datetime.now().strftime("%H
71 elif response_code == self.statuscode.USER_ALREADY_ONLINE():
72             print("User already online.")
73             self.notifications.append((datetime.now().strftime("%H
74 elif response_code == self.statuscode.USER_ALREADY_EXISTS():
75             print("User already exists.")
76             self.notifications.append((datetime.now().strftime("%H
77 elif response_code == self.statuscode.UPLOAD_SUCCESS():
78             print("File uploaded successfully.")
79             self.notifications.append((datetime.now().strftime("%H
80 elif response_code == self.statuscode.PING():
81             data = self.receive_message(self.client)
82             self.notifications.append((datetime.now().strftime("%H
83             message = f"ping {data}"
84             length = len(message)
85             packed_length = struct.pack("II", length)
86             print(f"Sending message: {message}")
87             self.notifications.append((datetime.now().strftime("%H

server.py AuthController.py
5 class AuthController():
6     def __init__(self):
7         self.database = Database()
8         self.response_code = StatusCode()
9         self.user = None
10
11 def login(self, hostname, password, ipAddress, peerPort):
12     print(f"Login request: {hostname} {password} {ipAddress} {pe
13     self.user = self.database.get_user(hostname)
14     print(self.user)
15     if (not self.user):
16         return self.response_code.USER_NOT_FOUND()
17     elif (self.user[2] != password):
18         return self.response_code.WRONG_PASSWORD()
19     elif (self.user[5] == 1):
20         return self.response_code.USER_ALREADY_ONLINE()
21     else:
22         self.database.update_user(hostname, ipAddress, peerPort)
23         return self.response_code.LOGIN_SUCCESS()
24
25 def register(self, hostname, password, ipAddress, peerPort):
26     print(f"Register request: {hostname} {password} {ipAddress}
27     if (self.database.get_user(hostname) != None):
28         return self.response_code.USER_ALREADY_EXISTS()
29     else:
30         self.database.add_user(hostname, password, ipAddress, pe
31         self.user = self.database.get_user(hostname)
32         return self.response_code.REGISTER_SUCCESS()
33
34 def publish(self, file):
35     if self.user:
36         print(file)
37         self.database.insert_file(self.user[0], file[0], file[1], f
38         return self.response_code.UPLOAD_SUCCESS()
39     else:
40         return self.response_code.BAD_REQUEST()
41
42 def fetch(self, file_name):
43     data = self.database.fetch(file_name)
44     print(data)
45     if len(data) == 0:
```

Figure 16: Trao đổi status code giữa client và server

