

ĐẠI HỌC QUỐC GIA TP HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA



MẠNG MÁY TÍNH (CO3094)

Bài tập lớn 1

FILE SHARING APPLICATION

GVHD:	La Quốc Nhật Huân	
Sinh viên:	Phạm Anh Dũng	2110101
	Phạm Anh Kiệt	2110304
Lớp:	L02	

TP.Hồ Chí Minh, Tháng 11 năm 2023



MỤC LỤC

1	PHASE 1	3
1.1	Lý thuyết cần chuẩn bị	3
1.1.1	Lý thuyết về Protocol	3
1.1.2	Mô hình TCP/IP	3
1.1.3	Mô hình Client-Server và mô hình Peer-To-Peer(P2P)	4
1.2	Các hàm chức năng và giao thức	6
1.2.1	Các hàm chức năng	6
1.2.2	Các giao thức	6
2	PHASE 2	7
2.1	Function Description	7
2.1.1	Client.py	7
2.1.2	Server.py	8
2.2	Class diagram	9
2.3	User Manual	9
3	Source code:	13



DANH SÁCH THÀNH VIÊN

STT	Họ tên	MSSV
1	Phạm Anh Dũng	2110101
2	Phạm Anh Kiệt	2110304

1 PHASE 1

1.1 Lý thuyết cần chuẩn bị

1.1.1 Lý thuyết về Protocol

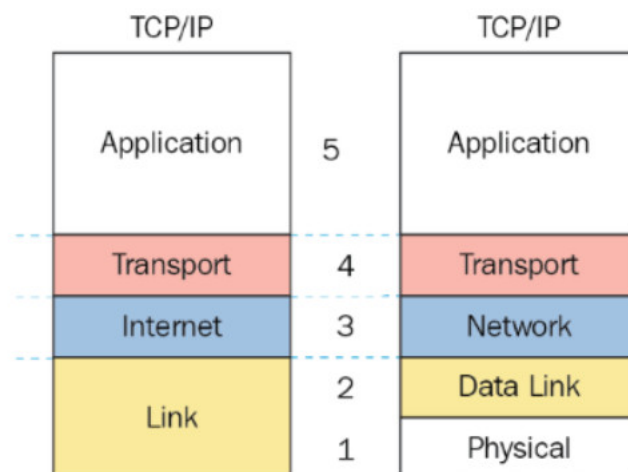
- Giao thức (protocol) là một bộ quy tắc và hướng dẫn để định dạng, truyền và giải mã dữ liệu giữa các thiết bị máy tính trên mạng. Giao thức xác định cách các thiết bị kết nối với nhau, cách chúng chia sẻ thông tin và cách chúng xử lý lỗi lúc truyền dữ liệu.
- Giao thức được sử dụng trong tất cả các loại mạng, bao gồm mạng cục bộ (LAN), mạng diện rộng (WAN) và cả Internet. Chúng là nền tảng cho mọi hoạt động truyền gửi dữ liệu trên mạng, từ truy cập website đến gửi mail hay gọi điện video.
- Một số giao thức phổ biến
 - Giao thức mạng (network protocol):
 1. IP (Internet Protocol)
 2. TCP (Transmission Control Protocol)
 3. UDP (User Datagram Protocol)
 - Giao thức ứng dụng (application protocol):
 1. HTTP (Hypertext Transfer Protocol)
 2. SMTP (Simple Mail Transfer Protocol)
 3. FTP (File Transfer Protocol)

Trong Bài Tập Lớn 1 này, nhóm sẽ sử dụng mô hình TCP/IP là chủ yếu.

1.1.2 Mô hình TCP/IP

Mô hình TCP/IP (Transmission Control Protocol/Internet Protocol) là một bộ giao thức chuẩn được sử dụng để quản lý việc truyền thông tin trên Internet. Nó bao gồm hai giao thức chính: TCP (Transmission Control Protocol) và IP (Internet Protocol).

Một mô hình TCP/IP tiêu chuẩn bao gồm 4 lớp được chồng lên nhau, bắt đầu từ tầng thấp nhất là Tầng vật lý (Physical) → Tầng mạng (Network) → Tầng giao vận (Transport) và cuối cùng là Tầng ứng dụng (Application) (Hình bên trái). Tuy nhiên, một số ý kiến lại cho rằng mô hình TCP/IP là 5 tầng, tức các tầng 4 đến 2 đều được giữ nguyên, nhưng tầng Datalink sẽ được tách riêng và là tầng nằm trên so với tầng vật lý. (Hình bên phải)



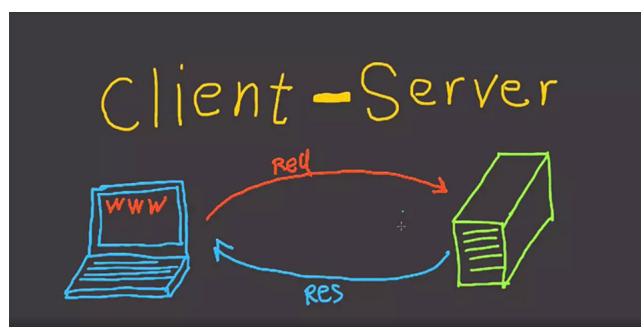
Hình 1: Các thành phần của mô hình TCP/IP

Trong bài tập lớn này, chúng ta cùng hiện thực lại tầng ứng dụng (Application).

1.1.3 Mô hình Client-Server và mô hình Peer-To-Peer(P2P)

1. Mô hình Client-Server:

Client-Server là mô hình mạng máy tính gồm có 2 thành phần chính đó là máy khách (Client) và máy chủ (Server). Server chính là nơi giúp lưu trữ tài nguyên cũng như cài đặt các chương trình dịch vụ theo đúng như yêu cầu của Client. Ngược lại, Client bao gồm máy tính cũng như các loại thiết bị điện tử nói chung sẽ tiến hành gửi yêu cầu truy cập hay khai thác thông tin đến cho Server. Hay nói đơn giản thì Client là các thiết bị gửi yêu cầu đến Server để thực hiện một việc gì đó (Request) và Server là nơi tiếp nhận và xử lý các yêu cầu mà Client gửi đến rồi phản hồi lại với Client (Response).

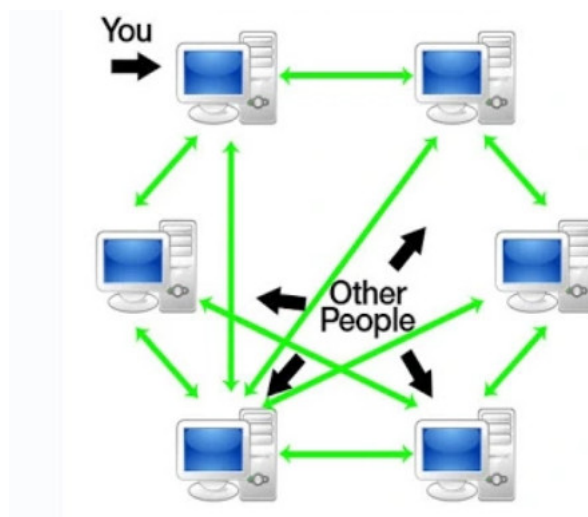


Hình 2: Mô hình Client-Server

2. Mô hình Peer-To-Peer:

Peer-to-peer (P2P) là một kiến trúc mạng trong đó các thiết bị (hoặc “Peer”) kết nối và trao đổi thông tin trực tiếp với nhau mà không thông qua một trung tâm điều khiển hay

máy chủ trung gian như mô hình Client-Server. Trong mô hình này, các thiết bị được coi là ngang hàng với nhau và có khả năng chia sẻ tài nguyên và dữ liệu với nhau. Trong mạng P2P, mỗi Peer có khả năng đóng vai trò là cả người gửi và người nhận thông tin. Điều này cho phép các thiết bị trực tiếp kết nối với nhau để trao đổi dữ liệu, chia sẻ tài nguyên như tệp tin, ứng dụng, hoặc dịch vụ mà không cần một máy chủ trung gian quản lý.



Hình 3: Mô hình Peer-To-Peer

3. Một số điểm khác nhau giữa mô hình Client-Server và mô hình Peer-To-Peer:

Điểm so sánh	Client-Server	Peer-To-Peer
Vai trò, phân quyền	Phân chia vai trò rõ ràng giữa Client và Server	Trong cùng một mạng thì tất cả các Peer đều ngang hàng
Quản trị mạng	Cần có người quản trị mạng	Không cần người quản trị mạng
Phần cứng, phần mềm cần thiết	Cần máy chủ, phần cứng và hệ điều hành	Cần khá ít phần cứng, có thể không cần hệ điều hành và máy chủ
Chi phí cài đặt	Cao	Thấp
Quản lý tài nguyên	Tài nguyên được tập trung trên máy chủ	Tài nguyên được chia sẻ trực tiếp giữa các thiết bị
Hiệu suất và mở rộng	Có thể quản lý số lượng lớn Client và tối ưu hóa hiệu suất của Server	Hiệu suất có thể giảm đi khi số lượng thiết bị tăng lên, do cần tiêu tốn thêm tài nguyên để xử lý các yêu cầu từ các peer khác.
Bảo mật	Một máy chủ trung tâm có thể tập trung vào việc triển khai các biện pháp bảo mật, giúp kiểm soát và bảo vệ tài nguyên.	Cần phải có các biện pháp bảo mật phân tán cho mỗi thiết bị, do không có một điểm trung tâm quản lý.

1.2 Các hàm chức năng và giao thức

1.2.1 Các hàm chức năng

Có 2 loại thông điệp mà các socket trao đổi với nhau trong ứng dụng:

- **Yêu cầu(Request)**: là các yêu cầu từ máy người dùng gửi đến với mong muốn nhận được kết quả của yêu cầu. phía nhận có thể là máy của người dùng khác hoặc server.
- **Phản hồi(Response)**: Là phản hồi của máy nhận với yêu cầu của máy gửi; phản hồi có thể được gửi bởi Server hoặc Peer khác trong mạng

Các hàm chức năng:

- **Client**:
 - **Publish files**: tiến hành gửi thông tin chi tiết của file lên Server chứ không gửi dữ liệu của file lên Server.
 - **Fetch files**: fetch một bản copy của file mà người dùng yêu cầu. Client gửi yêu cầu lên Server và Server gửi danh sách các hostname đã publish file kèm theo thông tin, Client chọn kết nối peer to peer với một trong các hostname để fetch bản copy của file cần tải về.
- **Server**:
 - **Discover hostname**: Server tiến hành kiểm tra các file hiện có trong kho chứa của client có tên là "host name".
 - **Ping hostname**: Server tiến hành kiểm tra xem client có tên là "host name" có đang trực tuyến không.

1.2.2 Các giao thức

- **Client-to-Server protocol**:
 - **Input hostname**: client kết nối với Server dưới tên "hostname" của mình.
Nếu hostname chưa tồn tại:
 - * Request: REGISTER hostname files
 - * Response: REGNếu hostname đã tồn tại, và không client nào đang sử dụng hostname đó:
 - * Request: REGISTER hostname files
 - * Response: LOGNếu hostname đã tồn tại nhưng đang có client khác sử dụng
 - * Request: REGISTER hostname files
 - * Response: FAIL
 - **Publish(fname)**: Publish thông tin file trên local của client, tên file lưu trên server sẽ trùng với tên file trên local.

- * Request: PUBLISH fname
 - * Response: OK
- **Fetch(fname)**: Yêu cầu server cung cấp thông tin địa chỉ của các hostname đã publish file có tên fname
 - * Request: FETCH fname
 - * Response: LIST_OF_SOURCES
- Client-to-Client protocol:
 - **Request(filename)**: Client yêu cầu fetch file từ client khác
 - * Request: Request fname
 - * Response: FILE_DATA
- Server-to-Client protocol:
 - **Ping(hostname)**: Tiến hành ping tới địa chỉ của hostname - Địa chỉ của hostname đang kết nối với server
 - * Request: PING hostname
 - * Response: PONG
 - Địa chỉ của hostname đang offline
 - * Request: PING hostname
 - * Response: Request timeout
- TCP/IP protocol stack:
 - **ICP**: TCP sẽ được sử dụng để cung cấp kết nối đáng tin cậy cho tầng transport của ứng dụng. Điều này là cần thiết để đảm bảo rằng các file sẽ được truyền một cách chính xác và đầy đủ.
 - **IP**: IP được sử dụng để định tuyến đường đi cho các gói dữ liệu(packet) giữa các Client và Server.

2 PHASE 2

2.1 Function Description

2.1.1 Client.py

- **Class variable**
 - server_address : Lưu thông tin địa chỉ của server
 - client_socket: Client socket dùng để kết nối và giao tiếp với socket của server
 - p2p_socket: Socket dùng để nhận kết nối với các client khác
- **__init__(self, server_address)**: Phương thức để khởi tạo cho class Client, kết nối socket client với socket của server và khởi tạo các biến của class
- **send_command(self)**: Quản lý việc nhập command-shell interpreter từ user, các command gồm có: fetch, publish, quit.

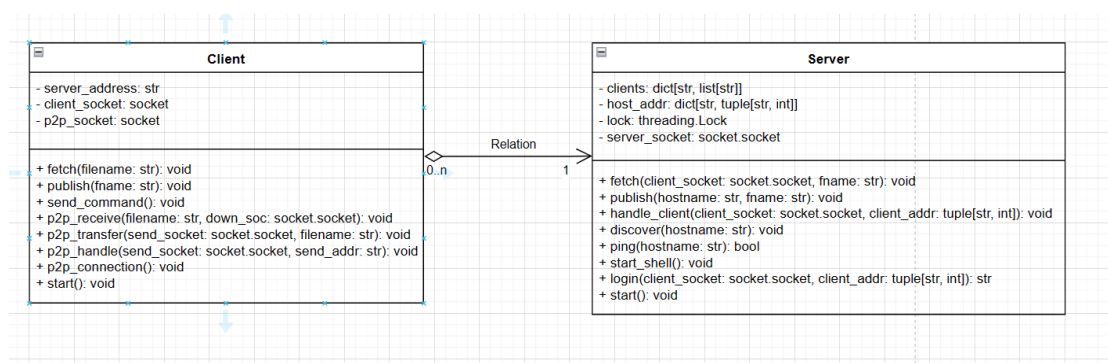
- **publish(self, fname)**: Phương thức để nhận response từ server sau khi gửi yêu cầu publish file
- **fetch(self, filename)**: Phương thức nhận response từ server là một danh sách các source đã publish file cần fetch. Sau đó sẽ khởi tạo socket **down_soc**, user sẽ điền address và port từ danh sách đó và dùng **down_soc** để kết nối trực tiếp đến **p2p_socket** của client khác. Sau khi đã kết nối, gọi phương thức **p2p_receive** để bắt đầu thực hiện fetch file. User nếu kết nối thất bại có lựa chọn kết nối với những hostname khác.
- **p2p_receive(self, filename, down_soc)**: Phương thức dùng để nhận file từ hostname khác, việc nhận file sẽ theo từng chunk với size là 131072 bytes, kết thúc khi nhận được dữ liệu "<END>". Sau khi nhận xong sẽ write và user sẽ có file xuất hiện trên local.
- **p2p_transfer(self, send_socket, filename)**: Phương thức dùng để chuyển file tới hostname khác, dữ liệu kết thúc là "<END>" để nhận biết end-point data của file.
- **p2p_handle(self, send_socket, send_addr)**: Dùng để quản lý việc nhận request từ các client khác, ngoài ra cũng nhận tín hiệu ping từ server.
- **p2p_connection(self)**: Quản lý nhận kết nối socket từ các client khác, thực hiện multi-threading để connect nhiều client cùng lúc.
- **start(self)**: Dùng để chạy phương thức **send_command(self)** và **p2p_connection(self)**. Ngoài ra còn ghi nhận tên hostname từ user để liên kết với địa chỉ và port đang chạy.

2.1.2 Server.py

- **Class variable**
 - **clients** : Lưu danh sách các hostname đã đăng kí cùng với file được publish
 - **host_addr**: Lưu danh sách các địa chỉ và port socket tương ứng với các hostname
 - **lock**: Dùng để quản lý các khu vực lock giữa các threading
 - **server_socket**: socket của server
- **fetch(self, client_socket, fname)**: Phương thức để response với request "fetch" từ client với nội dung là danh sách địa chỉ của các hostname đã publish fname với server. Sử dụng json để gửi mảng danh sách tới socket của client.
- **publish(self, hostname, fname)**: Phương thức dùng để thực hiện lưu các file được publish bởi hostname vào mảng **client** tương ứng.
- **handle_client(self, client_socket, client_addr)**: Phương thức dùng để quản lý các request từ client và gọi hàm tương ứng để response. Đồng thời sẽ ghi nhận tên hostname của client đang kết nối qua hàm **login**. Sử dụng biến **lock** của class để khoá các vùng xử lý các mảng giữa các threading.
- **login(self, client_socket, client_addr)**: Phương thức dùng để xử lý hostname user nhập từ client. Phương thức sẽ liên kết địa chỉ với hostname, ngoài ra nếu là một hostname chưa đăng kí, server sẽ khởi tạo mảng **client**. Nếu hostname đã được đăng kí, phương thức sẽ gọi hàm **ping** để kiểm tra có client nào đang chạy dưới tên hostname này không, và sẽ đồng ý nếu không có client nào đang chạy.

- **start_shell(self)**: Dùng để quản lý các command discover, ping của server và gọi hàm tương ứng.
- **ping(self, hostname)**: Phương thức dùng để thực hiện kiểm tra tình trạng kết nối của hostname. Phương thức tạo ra một socket tạm **temp_socket** để kết nối tới địa chỉ và port của hostname được lưu trong mảng **host_addr** và trả về các kết quả tương ứng.
- **discover(self, hostname)**: Xuất các file đã được publish trong mảng **clients** của hostname tương ứng.
- **start(self)**: Phương thức dùng để chạy hàm **start_shell** để bắt đầu nhận command và hàm **handle_client**, được xử lý multithreading để kết nối nhiều client cùng lúc

2.2 Class diagram



2.3 User Manual

Ta cài đặt các thư viện cần thiết: socket, os, threading, tqdm và json.

Trước tiên ta khởi động Server bằng cách chạy Server.py. Ta nhập command line sau:

```
1 python Server.py
```

```
HP@JonSnow MINGW64 ~/Documents/Desktop/Mang may tinh/As1/file-sharing app
$ python Server.py
Server listening for incoming connections...
```

Sau đó ta chạy Client bằng câu lệnh:

```
1 python Client.py
```

```
HP@JonSnow MINGW64 ~/Documents/Desktop/Mang may tinh/As1/file-sharing app
$ python Client.py
Connected to File-sharing server
>Input your hostname:
```

Ta nhập tên hostname muốn đăng kí với server, server sẽ không đồng ý hostname nếu có một Client khác đang sử dụng

```
HP@JonSnow MINGW64 ~/Documents/Desktop/Mang may tinh/As1/file-sharing app
$ python Client.py
Connected to File-sharing server
>Input your hostname: dung
Sign in Successfully
To start, type 'publish fname' or 'fetch fname' or 'quit'
>
```

```
HP@JonSnow MINGW64 ~/Documents/Desktop/Mang may tinh/As1/file-sharing app
$ python Client.py
Connected to File-sharing server
>Input your hostname: dung
Hostname is already running, please retry !
>Input your hostname: |
```

Ta nhập command line dưới đây để publish một file có tên "ref1.pdf" với server:

```
1 publish ref1.pdf
```

```
HP@JonSnow MINGW64 ~/Documents/Desktop/Mang may tinh/As1/file-sharing app
$ python Client.py
Connected to File-sharing server
>Input your hostname: dung
Sign in Successfully
To start, type 'publish fname' or 'fetch fname' or 'quit'
> publish ref1.pdf
Publish success file ref1.pdf
>
```

Ta nhập command line dưới đây để fetch một file có tên "ref1.pdf", server sẽ gửi danh sách những hostname đã publish file "ref1.pdf":







```
1 fetch ref1.pdf
```

```
MINGW64:/c/Users/HP/Documents/Desktop/Mang may tinh/As1/file-sharing...
HP@JonSnow MINGW64 ~/Documents/Desktop/Mang may tinh/As1/file-sharing app
$ python Client.py
Connected to File-sharing server
>Input your hostname: ryan
Sign in Successfully
To start, type 'publish fname' or 'fetch fname' or 'quit'
> fetch ref1.pdf
List of hostname address that published the file:
Hostname: dung, Address: 127.0.0.1, Port: 11332
Choose the address and port of the hostname you want to fetch form
>Address: .....
```

Ta nhập address và port từ danh sách trên để tải về file "ref1.pdf" từ hostname đó

```
MINGW64:/c/Users/HP/Documents/Desktop/Mang may tinh/As1/file-sharing...
HP@JonSnow MINGW64 ~/Documents/Desktop/Mang may tinh/As1/file-sharing app
$ python Client.py
Connected to File-sharing server
>Input your hostname: ryan
Sign in Successfully
To start, type 'publish fname' or 'fetch fname' or 'quit'
> fetch ref1.pdf
List of hostname address that published the file:
Hostname: dung, Address: 127.0.0.1, Port: 11332
Choose the address and port of the hostname you want to fetch form
>Address: 127.0.0.1
>Port: 11332
Downloading ref1.pdf...
0%|          | 0.00/1.50M [00:00<?, ?B/s]
Download ref1.pdf completed
1.70MB [00:00, 84.7MB/s]
> |
```

Sau khi tải xong, file "ref1.pdf" sẽ xuất hiện dưới tên "received_ref1.pdf" trên local

	Client.py	11/12/2023 2:32 AM	Python File	6 KB
	py	11/9/2023 3:21 PM	File	0 KB
	received_ref1.pdf	11/12/2023 2:51 AM	Microsoft Edge PD...	1,465 KB
	ref1.pdf	11/2/2023 8:46 PM	Microsoft Edge PD...	1,465 KB
	Server.py	11/12/2023 2:32 AM	Python File	5 KB
	Test.7z	10/20/2023 10:26 PM	WinRAR archive	120,824 KB

Nhập quit để ngắt kết nối với server

```

MINGW64:/c/Users/HP/Documents/Desktop/Mang may tinh/As1/file-sharing...
HP@JonSnow MINGW64 ~/Documents/Desktop/Mang may tinh/As1/file-sharing app
$ python Client.py
Connected to File-sharing server
>Input your hostname: ryan
Sign in Successfully
To start, type 'publish fname' or 'fetch fname' or 'quit'
> fetch ref1.pdf
List of hostname address that published the file:
Hostname: dung, Address: 127.0.0.1, Port: 11332
Choose the address and port of the hostname you want to fetch form
>Address: 127.0.0.1
>Port: 11332
Downloading ref1.pdf...
0%|          | 0.00/1.50M [00:00<?, ?B/s]
Download ref1.pdf completed
1.70MB [00:00, 84.7MB/s]
> fetch ref2.pdf
None hostname published that file you requested
> quit

HP@JonSnow MINGW64 ~/Documents/Desktop/Mang may tinh/As1/file-sharing app
$

```

Về phía của server, ta sẽ nhập command line sau để liệt kê những file đã được publish bởi hostname "dung":

```
1 discover dung
```

```

MINGW64:/c/Users/HP/Documents/Desktop/Mang may tinh/As1/file-sharing...
HP@JonSnow MINGW64 ~/Documents/Desktop/Mang may tinh/As1/file-sharing app
$ python Server.py
Server listening for incoming connections...
Connected to ('127.0.0.1', 11361)
Connected to ('127.0.0.1', 11362)
Success publish ref1.pdf from dung
discover dung
Files published by dung:
ref1.pdf
Success discover

```

Ngoài ra ta có thể ping tới hostname xem tình trạng kết nối của hostname đó:

```

ping dung
dung is active
Disconnected to dung
dung is not active
Connected to ('127.0.0.1', 11368)
dung is active

```



3 Source code:

Link github: <https://github.com/adung1211/as1-file-sharing>

References

- [1] R. Elmasri & S.B. Navathe, Addison-Wesley. (2017). *Fundamentals of Database Systems*, 7th Edition.