ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH

Mạng máy tính

BÀI TẬP LỚN 1

DEVELOP A NETWORK APPLICATION

Giảng viên hướng dẫn: **Vũ Thành Tài**
Sinh viên thực hiện: **Nhóm 10**

TP. HỒ CHÍ MINH, THÁNG 11 NĂM 2024

Danh sách thành viên & Mức độ đóng góp

STT	Họ và Tên	MSSV	Đóng góp	Ghi chú
1	Nguyễn Huỳnh Hải Đăng	2210737	100%	
2	Lê Đức Nghĩa	221	100%	
3	Nguyễn Đức Hoài Nam	2211862	100%	
4	Võ Ngọc Tú	2213857	100%	

Danh mục bảng

Mục lục

1. Thiết kế cho phía Server

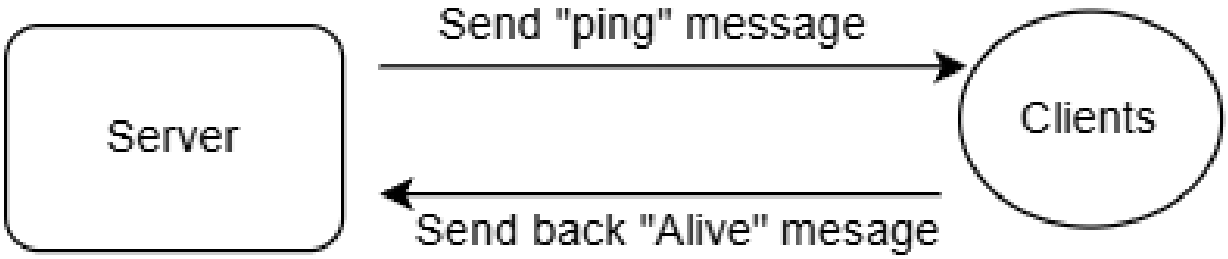
1.1. Thiết kế lệnh ping

1.1.1 Mục đích

Máy chủ sử dụng lệnh ping để kiểm tra xem các peer (client) trong mạng có đang "hoạt động" hay không.

1.1.2 Vấn đề và luồng thực thi

Trong lệnh ping, cần giải quyết vấn đề sau: Khi máy chủ gửi yêu cầu ping đến một client cụ thể dựa trên tên máy (hostname), client đó cần gửi một thông báo xác nhận lại cho máy chủ để báo hiệu rằng nó vẫn đang hoạt động.



Hình 1: Luồng thực thi lệnh ping.

- Máy chủ gửi yêu cầu ping đến client để xác nhận client đó có đang hoạt động hay không.
- Client nhận được yêu cầu từ máy chủ và phản hồi lại nhằm xác nhận rằng nó đang hoạt động.

1.1.3 Thiết kế

Yêu cầu từ máy chủ đến client sẽ bao gồm tên máy (hostname) để đảm bảo gửi lệnh chính xác đến client cần kiểm tra. Nội dung của message ping là chuỗi "ping". Sau khi nhận được yêu cầu ping từ máy chủ, client sẽ gửi lại một message phản hồi cho máy chủ với nội dung là "Alive", nhằm thông báo rằng lệnh ping đã thành công và client có tên hostname đang hoạt động.

Vd: ping 127.0.0.1

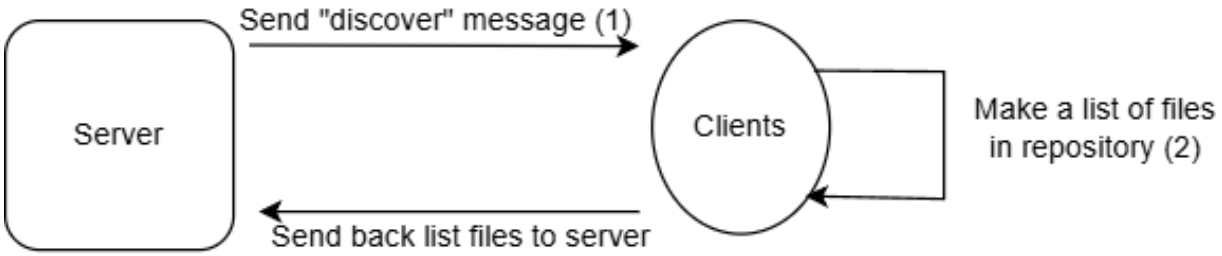
1.2. Thiết kế lệnh discover

1.2.1 Mục đích

Lệnh discover có mục tiêu thông báo cho máy chủ về các tệp mà client có thể sẵn sàng chia sẻ sau khi đã sử dụng lệnh publish.

1.2.2 Vấn đề và luồng thực thi

Với lệnh discover, cần xử lý như sau: Khi client nhận được yêu cầu discover từ máy chủ, client sẽ gửi lại danh sách các tệp mà nó đã sẵn sàng chia sẻ và đã được publish trước đó.



Hình 2: Luồng thực thi lệnh discover.

- (1): Máy chủ gửi lệnh discover đến client cần nhận thông tin.
- (2): Client nhận được yêu cầu discover và tạo ra một danh sách các tệp có trong kho lưu trữ.
- (3): Client gửi danh sách này lại cho máy chủ.

1.2.3 Thiết kế

Yêu cầu từ máy chủ đến client sẽ bao gồm hostname để gửi chính xác lệnh đến client mong muốn. Khi nhận được lệnh `discover`, client sẽ tạo một danh sách các tệp dưới dạng JSON như sau:

```
{"files": [filename]}
```

Sau khi danh sách được tạo thành công, client sẽ gửi nó lại cho máy chủ. Máy chủ khi nhận được dữ liệu JSON này sẽ phân tích và hiển thị ra màn hình.

```
> Files received from client:
> file_name_1
> file_name_2
```

2. Thiết kế cho phía Client

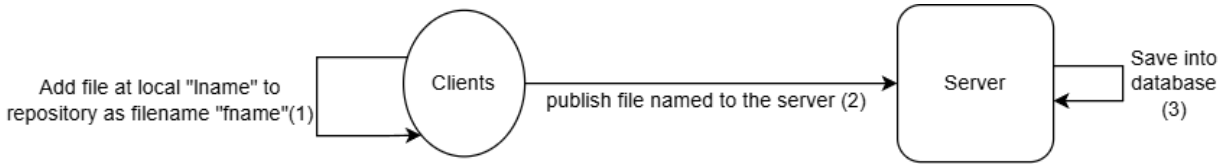
2.1. Thiết kế lệnh publish

2.1.1 Mục đích

Lệnh publish được dùng để thông báo cho máy chủ rằng client (host hiện tại) có một tệp tên là fname có thể được chia sẻ với các peer khác trong mạng.

2.1.2 Vấn đề và luồng thực thi

Để thực hiện lệnh publish, cần xử lý như sau: Khi máy chủ nhận được yêu cầu publish, nó phải lấy địa chỉ IP và cổng (port) của client đã gửi yêu cầu. Đây là địa chỉ mà máy chủ sử dụng để lắng nghe từ các peer khác, bao gồm client đã gửi yêu cầu publish.



Hình 3: Luồng thực thi lệnh publish.

- (1)Client thêm đường dẫn của tệp nội bộ muốn chia sẻ (đường dẫn này là lname) vào kho lưu trữ của mình, sử dụng fname làm key.
- (2)Client gửi thông tin của tệp này đến máy chủ để báo rằng nó có một tệp tên là fname sẵn sàng chia sẻ với các peer khác trong mạng.
- (3)Máy chủ nhận thông tin từ client về tệp fname và cập nhật vào cơ sở dữ liệu của mình.

2.1.3 Thiết kế

Kho lưu trữ của client ở bước (1) là một bảng ánh xạ (dictionary) với các cặp key-value; trong đó, "key" là tên của tệp fname và "value" là đường dẫn lname đến tệp

```
{'BTL': 'D:\\BTL_MMT\\BTL.txt'}
```

Yêu cầu publish mà client gửi đến máy chủ tập trung ở bước (2) sẽ có định dạng JSON, bao gồm 2 trường dữ liệu:
"publish": tên tệp fname mà client muốn chia sẻ với máy chủ.
"seeding_port": cổng mà client sử dụng để gửi yêu cầu.

```
{
  "publish": fname,
  "seeding_port": PEER_SRV_PORT
}
```

Cơ sở dữ liệu của máy chủ ở bước (3) là một dictionary với các cặp key-value; "key" là tên của tệp fname, và "value" là danh sách các địa chỉ (IP, port) của các peer chứa tệp này.

```
{
  [[192.168.11.11, 45555], [192.168.11.12, 45556]]
}
```

2.2. Thiết kế lệnh fetch

2.2.1 Mục đích

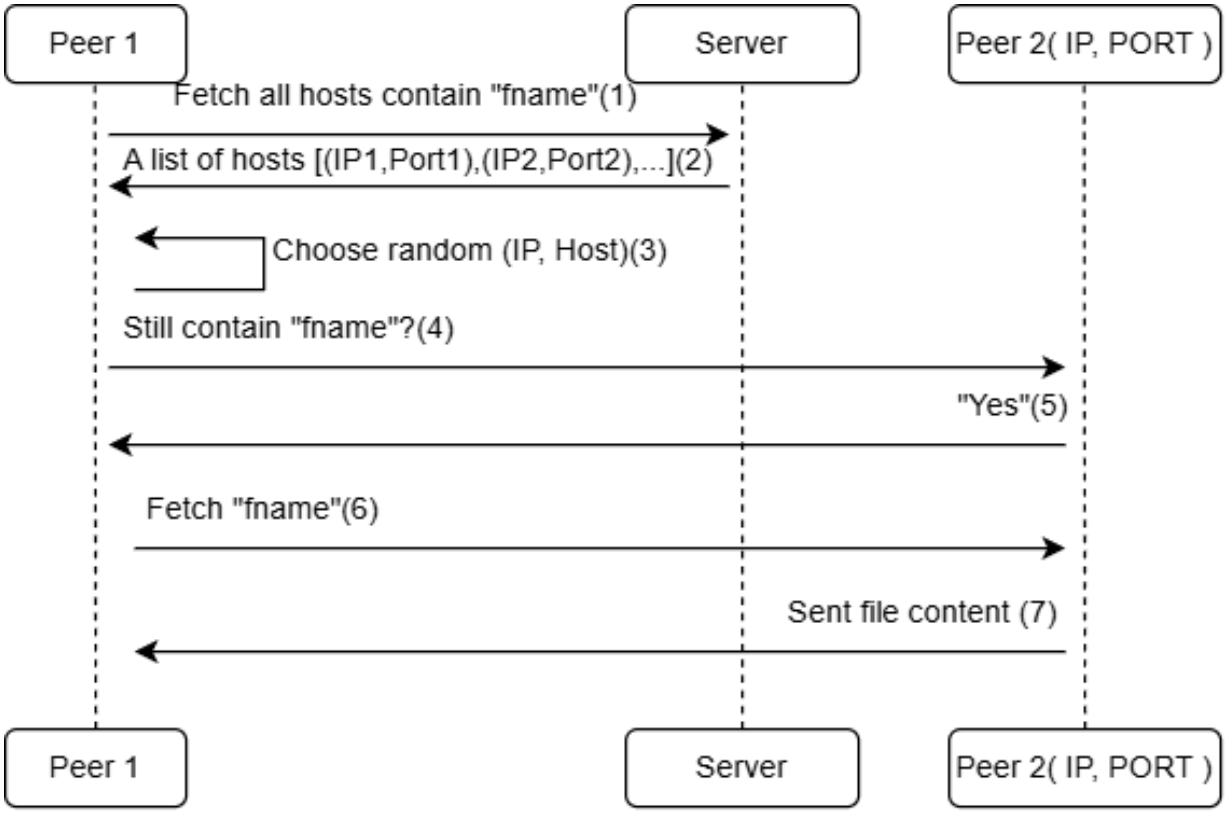
Lệnh fetch được thiết kế để cho phép một peer tải về một tệp từ các peer khác trong mạng. Khi thực hiện lệnh này, peer cần lấy danh sách các địa chỉ (IP, port) của các peer có chứa tệp mong muốn, từ đó có thể chọn một peer để tải tệp.

2.2.2 Vấn đề và luồng thực thi

Để thực hiện lệnh fetch, chúng ta cần giải quyết các vấn đề sau:

1. Khi một peer muốn tải một tệp, peer đó cần nhận được danh sách các địa chỉ (IP, port) của các peer đang giữ tệp được yêu cầu. Địa chỉ (IP, port) này là nơi một peer lắng nghe các yêu cầu từ các peer khác.

2. Sau khi peer đã nhận được danh sách các peer chứa tệp và chọn được một peer cụ thể, cần có cơ chế xác minh rằng peer được chọn có thực sự giữ tệp đó hay không.



Hình 4: Luồng thực thi lệnh fetch.

- (1), (2): Peer 1 thực hiện lệnh fetch fname, tiến trình này sẽ gửi một yêu cầu đến máy chủ để lấy thông tin về các host có chứa tệp fname
- (3): Peer 1 lựa chọn một peer từ danh sách các peer nhận được từ máy chủ tập trung. Lựa chọn này được thực hiện theo chính sách chọn ngẫu nhiên (Random selection policy).
- (4), (5): Peer 1 thực hiện việc xác minh xem tệp fname có thực sự tồn tại tại Peer 2 hay không. Peer 2 sẽ gửi kết quả xác thực về cho Peer 1.
- (6), (7): Nếu xác thực thành công, Peer 1 gửi yêu cầu tải tệp fname tới Peer 2 và Peer 2 sẽ gửi nội dung của tệp đó cho Peer 1.

2.2.3 Thiết kế

Yêu cầu (Request) mà Peer 1 gửi đến máy chủ tập trung ở bước (1) có định dạng JSON với hai trường dữ liệu:
"op": thao tác mà client muốn máy chủ thực hiện, trong trường hợp này là yêu cầu lấy danh sách host có chứa tệp.
"fname": tên tệp mà peer muốn tìm kiếm các host có chứa nó.

```
{
  "op": "fetch",
  "fname": fname
}
```

Máy chủ sẽ trả về phản hồi ở bước (2) là danh sách các cặp (host, port). Ví dụ phản hồi tương ứng với ví dụ trong lệnh publish trước đó:

```
{
  [[192.168.11.11, 45555], [192.168.11.12, 45556]]
}
```

Sau khi chọn một peer để thực hiện việc tải tệp, Peer 1 sẽ gửi một yêu cầu đến peer đã chọn để kiểm tra xem tệp fname có tồn tại tại Peer 2 hay không. Yêu cầu này có định dạng như sau:

```
{
  "op": "check",
  "fname": fname
}
```

Phản hồi từ Peer 2 sẽ là:
Nếu tệp tồn tại, Peer 2 trả về "EXIST"
Nếu tệp không tồn tại, Peer 2 trả về "NOT EXIST"
Sau khi Peer 2 xác nhận rằng nó có tệp fname và Peer 1 biết điều này, Peer 1 sẽ gửi một yêu cầu để tải tệp fname từ Peer 2 với định dạng như sau:

```
{
  "op": "fetch",
  "fname": fname
}
```

Tài liệu tham khảo