TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

Viện Công nghệ thông tin và Truyền thông

Thiết kế giao thức tầng ứng dụng

Học phần: Lập trình mạng

Nhóm sinh viên:

Nguyễn Bá Ngọc – 20173287

Nguyễn Đức Quyết – 20177021

Lê Đình Khánh – 20173194

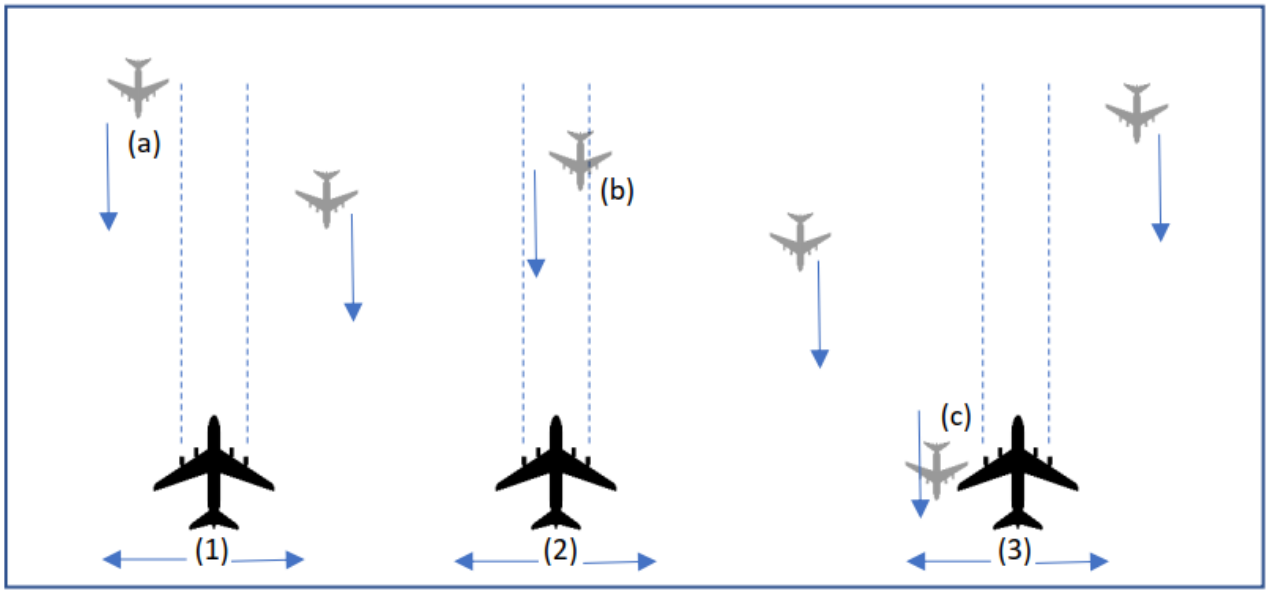
Giảng viên hướng dẫn: TS. Phạm Huy Hoàng

Hà Nội, ngày 12 tháng 06 năm 2020

Mục lục

I. Tổng quan hệ thống

1. Mô tả



Hệ thống game máy bay chạy trên mạng theo mô hình client-server:

1. Server đợi nghe kết nối từ client để join vào game.

2. Nhiều client kết nối đến server, login, yêu cầu join game & start (sau khi đã có đủ người join).

3. Trong một bài (level), các client điều khiển máy bay của mình (số 1,2,3 trong hình vẽ) để di chuyển sang trái, phải và bắn đạn. Các máy bay địch di chuyển từ xa về gần (từ phía trên màn hình xuống dưới). Bài game level càng cao thì tốc độ máy bay địch càng lớn.

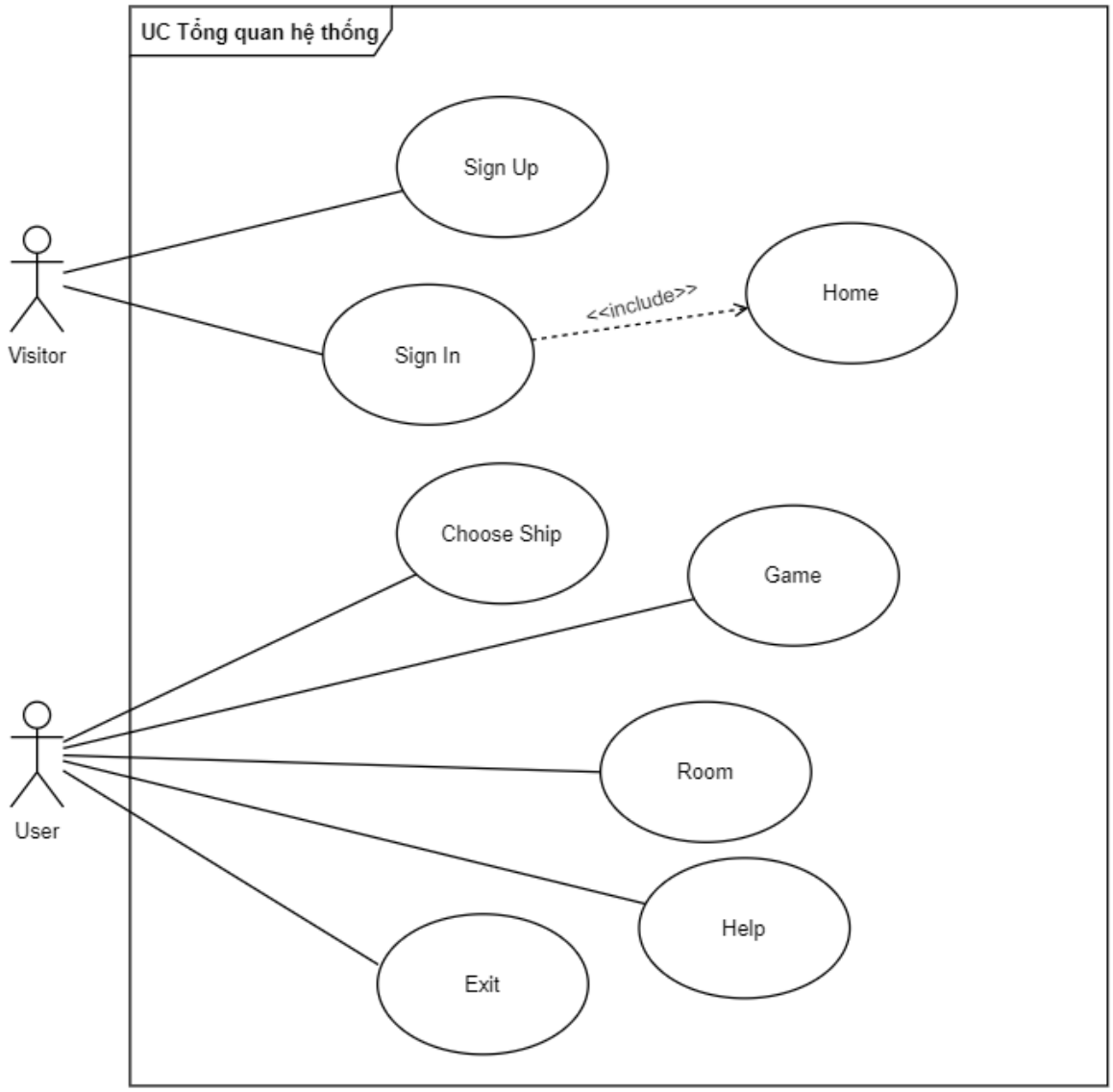
4. Màn hình của mỗi client hiển thị đầy đủ các máy bay quân ta (là những người join vào game) và các máy bay quân địch (được tạo ra ngẫu nhiên). Yêu cầu các màn hình client luôn được cập nhật thông tin từ server để hiển thị giống nhau và thể hiển giống nhau vị trí & di chuyển của các máy bay ta & địch.

5. Máy bay quân ta có thể bắn đạn vào máy bay địch. Trường hợp trúng đạn (điểm (b) trên hình), máy bay địch bị xóa khỏi màn hình của tất cả client.

6. Máy bay quân ta có thể bị máy bay địch đâm vào (điểm (c) trên hình). Khi đó người điều khiển máy bay quân ta nếu hết khiên thì sẽ bị thua và out khỏi game. Các người chơi còn lại vẫn tiếp tục.

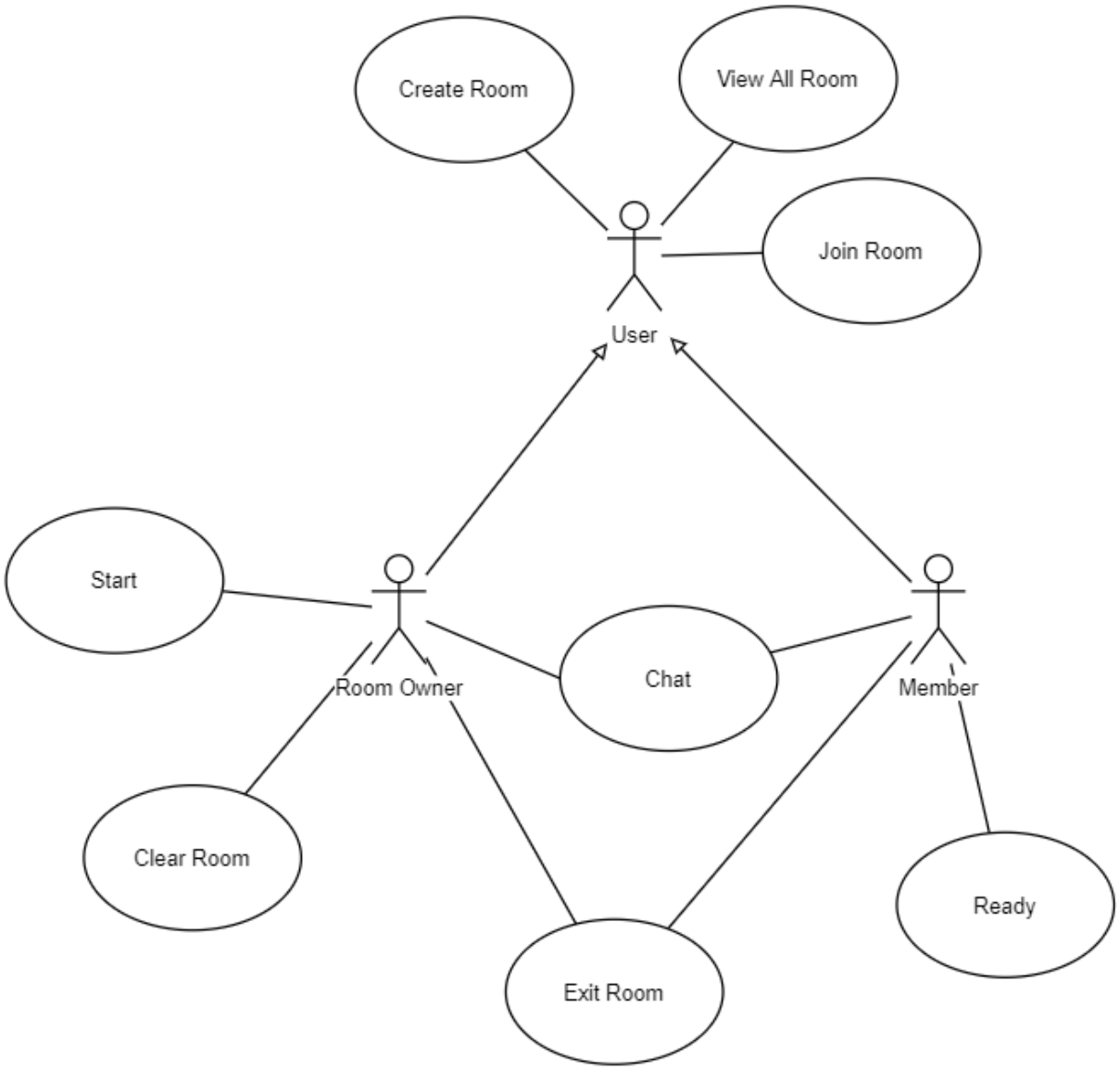
2. Biểu đồ usecase

2.1. Biểu đồ usecase tổng quan hệ thống

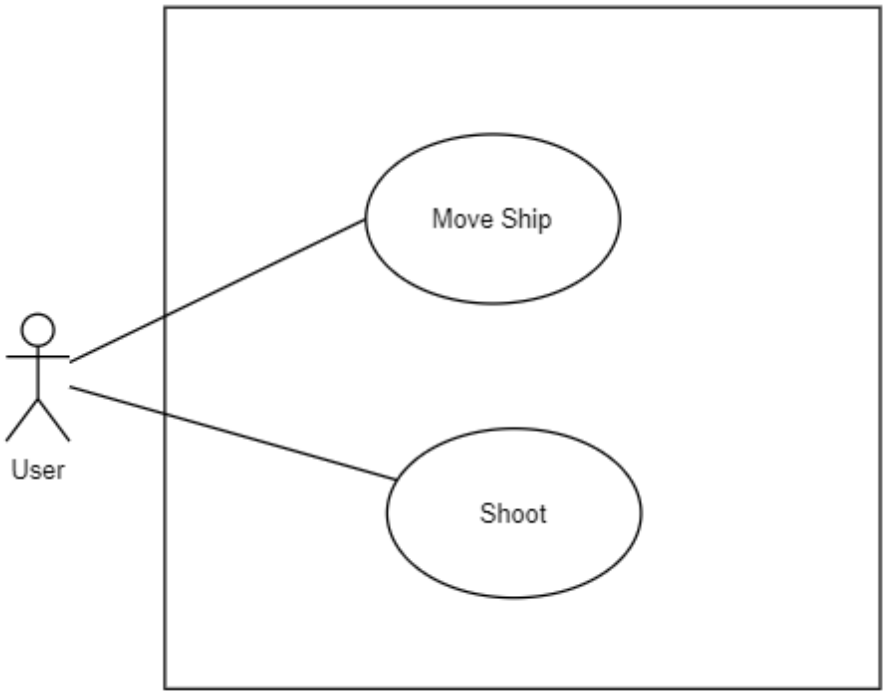


2.2. Các biểu đồ usecase phân rã

2.2.1. Đặc tả usecase “Room”



2.2.2. Đặc tả usecase “Game”



II. Thiết kế giao thức tầng ứng dụng

1. Mô tả một số mã

1.1. Status code

Code	Mô tả
0	Success
1	Fail

1.2. Error code

Code	Mô tả
400	Server lỗi
401	Lỗi đăng nhập
402	Lỗi đăng ký

403	Lỗi tạo phòng
404	Lỗi tham gia phòng
405	Lỗi rời phòng
406	Lỗi xóa phòng

1.3. Request code

Code	Mô tả
11	Đăng nhập
12	Đăng ký
20	Xem tất cả phòng hiện có
21	Tạo phòng
22	Vào phòng
23	Bắt đầu game
24	Chat room
25	Sẵn sàng
26	Rời phòng
27	Xóa phòng
41	Thao tác người chơi (di chuyển, bắn)
42	Người chơi bị hạ
43	Địch bị hạ
60	Thiết lập kết nối udp

1.4. Response Code từ Server

Code	Mô tả
69	Server lỗi
81	Người mới vào phòng

82	Thao tác người chơi (di chuyển, bắn)
83	Thiết lập kết nối udp
84	Gửi các thành phần game tới người chơi (địch,…)
85	Cập nhật điểm của người chơi
86	Level mới
87	Chiến thắng
88	Thất bại
89	Người chơi bị hạ
110	Đăng nhập
120	Đăng ký
200	Xem tất cả phòng
210	Tạo phòng
220	Vào phòng
230	Bắt đầu
240	Chat
250	Sẵn sàng
260	Rời phòng
270	Xóa phòng
280	Đặt chủ phòng mới
290	Thông báo người chơi rời phòng

2. Chi tiết thiết kế

Dữ liệu truyền được xây dựng là một đối tượng JSON, trước khi truyền sẽ được chuyển thành String. Bên nhận thực hiển chuyển ngược lại từ String thành JSON để xử lý.

2.1. Đăng nhập

Request:

```
{
        "req_code": 11,
        "username": <username>,
        "password": <password>
```

}

Response:

- Success:

{
"tcp_code": 110,
"status": 1,
"user_id": <userId>
}

- Failed

{
"tcp_code": 110,
"status": 0,
"error_code": 401,
"message": "Tài khoản hoặc mật khẩu không chính xác"
}

{
"tcp_code": 110,
"status": 0,
"error_code": 401,
"message": "Tài khoản đang được sử dụng"
}

2.2. Đăng ký

Request:

{
"req_code": 12,
"username": <username>,
"password": <password>
}

Response:

- Success

{
"tcp_code": 120,
"status": 1
}

- Failed

{
"tcp_code": 120,

"status": 0,
"error_code": 402,
"message": "Tài khoản đã tồn tại"
}

2.3. Xem tất cả phòng

Request:
{
"req_code": 20
}

Response:
{
"tcp_code": 200,
"status": 1,
"list_room": [<list room information>]
}

Note: Mỗi phần tử của list_room là một JSONArray có dạng sau:

[<roomID>, <roomName>, <capacity>, <isRunning>, <hasPassword>]

2.4. Tạo phòng

Request:
{
"req_code": 21,
"room_name"": <roomName>,
"room_owner": <roomOwner>,
"size": <roomSize>,
"ship": <shipName>,
"room_pass": <roomPass> **(có hoặc không)**
}

Response:
- Success

{
"tcp_code": 210,
"status": 1,

"room": {
"room_id": <id>,
"room_name": <roomName>,
"has_pass": <0 or 1>,
"room_size": <roomSize>
}
}

- Failed

{
"tcp_code": 210,
"status": 0,
"error_code": 403,
"message": "Phòng đã tồn tại. Hãy chọn tên khác"
}

2.5. Vào phòng

Request:

{
"req_code"; 22,
"room_name": <roomName>,
"player_name": <playerName>,
"ship": <shipName>,
"room_pass": <roomPass> **(có hoặc không)**
}

Response:

- Success

{
"tcp_code": 220,
"status": 1,
"room": {
"room_id": <id>,
"room_name": <roomName>,
"has_pass": <0 or 1>,
"room_size": <roomSize>,
"owner_id": <ownerId>,
"team": [<list informations of players>]
}
}

Note: player's information

{
"name": <playerName>,
"player_id": <playerId>,
"ship": <shipName>
}

- Failed

{
"tcp_code": 220,
"status": 0,
"error_code": 404,
"message": <one of the messages below>
}

Note: messages

- *"Phòng không tồn tại"*
- *"Mật khẩu phòng không chính xác"*
- *"Phòng yêu cầu mật khẩu"*
- *"Phòng đã đầy"*
- *"Đang trong trận đấu"*

2.6. Bắt đầu game

Mô tả: Chủ phòng gửi gói tin yêu cầu start game, server khi nhận được gói tin sẽ thực hiện gửi gói tin TCP yêu cầu các Client (các thành viên trong phòng) thiết lập kết nốt UDP.

Client request:

{
"req_code": 23
}

Server response:

- Success

{
"tcp_code": 230,
"status": 1
}

- Failed

{
"tcp_code",
"status": 0
}

Yêu cầu thiết lập kết nối:

{
"tcp_code": 83
}

2.7. Rời phòng

Request:

{
"req_code": 26,
"player_id": <playerId>
}

Response:

- Success:

{
"tcp_code": 260,
"status": 1
}

Notify to all members

{
"tcp_code": 290,
"player_id": <playerId>
}

New owner

{
"tcp_code": 280,
"owner_id": <ownerId>
}

- Failed:

{
"tcp_code": 260,
"status": 0,
"error_code": 405,
"message": "Đang trong trận đấu"
}

2.8. Xóa phòng

Mô tả: Chủ phòng gửi request đến server thông báo xóa phòng, nếu đang trong game thì sẽ nhận được response failed, ngược lại server gửi response xóa phòng thành công đến tất cả các người chơi trong phòng.

Request:

```
{
        "req_code": 27,
        "player_id": <playerId>
}
```

Response:

- Status

```
{
        "tcp_code": 270,
        "status": 1
}
```

- Failed

```
{
        "tcp_code": 270,
        "status": 0,
        "error_code": 406,
        "message": "Đang trong trận đấu"
}
```

2.9. Chat

Mô tả: người chơi gửi bản tin request có chứa tin nhắn, server thực hiện gửi tin nhắn đó đến các người chơi còn lại trong phòng

Request:

```
{
        "req_code": 24,
        "sender": <senderName>,
        "data": [<senderName>, <message>]
}
```

The server sends a response to all members

```
{
        "tcp_code": 240,
        "data": [<senderName>, <message>]
}
```

2.10. Người chơi bị hạ

Mô tả: khi người chơi bị hạ, sẽ gửi gói tin thông báo tới server, server gửi gói tin thông báo tới các người chơi khác (gửi id của người chơi vừa bị hạ đến những người chơi còn lại)

Request:

```
{
        "req_code": 42,
        "player_id": <playerId>
}
```

The server notifies to members

- Success

```
{
        "tcp_code": 89,
        "player_id": <playerId>
}
```

2.11. Địch bị hạ

Mô tả: Khi người chơi bắn hạ quân địch sẽ gửi gói tin tới server (gồm id của người chơi và của địch vừa bị hạ), server thực hiện kiểm tra và nếu đúng là người chơi này bắn hạ địch thì gửi về điểm của người chơi.

Request:

```
{
        "req_code": 43,
        "enemy_id": <enemyId>,
        "killer_id": <killerId>
}
```

Response:

```
{
        "tcp_code": 85,
        "score": <yourScore>
}
```

2.12. Thao tác người chơi (di chuyển: trái, phải và bắn)

Mô tả: các thao tác (di chuyển, bắn) của người chơi sẽ được gửi đến server thông qua giao thức UDP. Server thực hiện chuyển tiếp thông tin về thao tác này đến các người chơi còn lại
Gói tin gửi đi gồm các trường dữ liệu sau:

- thông tin về người chơi: shipId, vị trí, thao tác (trái, phải, bắn)
- id người chơi
- id của phòng

Request:

```
{
        "req_code": 41,
        "player_id": playerId,
        "room_id": roomId,
        "data": [<shipId>, <layoutX>, <action>]
}
```

Response:

```
{
        "udp_code": 82,
        "payload": data
}
```