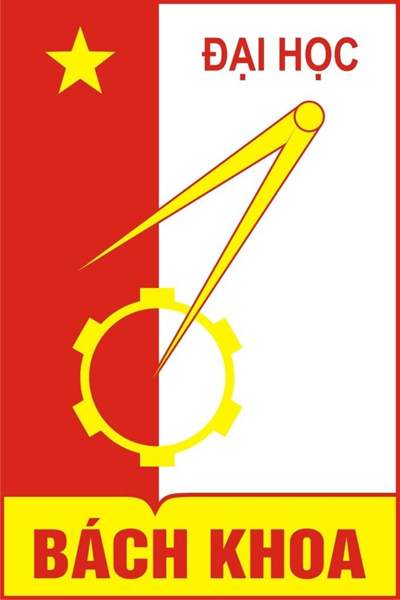
**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**KHOA CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

──────── \* ───────



**Network Programming Report**

***ĐỀ TÀI***: Xây dựng chương trình trò chơi Battleship qua mạng cục bộ

**Giảng viên** Trần Nguyên Ngọc

**Sinh viên thực hiện** Tạ Duy Hưng - 20101682

Trần Quang Duy - 20101258

**Lớp** ICT-k55

1. **Giới thiệu đề tài:**

Đề tài: Xây dựng chương trình trò chơi Battleship qua mạng cục bộ.

1. **Đặc tả yêu cầu:**
2. Client:

* Chương trình tương tác với người dùng qua giao diện menu lựa chọn.
* Cho phép người dùng đăng nhập, đăng ký tài khoản; xem danh sách những người chơi đang online cùng và tình trạng của họ.
* Cho phép người dùng mời hoặc chờ đợi có lời mời để chơi trò chơi Battleship giữa hai máy trong mạng.

1. Server:

* Kiểm tra danh sách người dùng từ cơ sở dữ liệu để tiến hành quá trình đăng ký, đăng nhập.
* Chuyển tiếp lời mời chơi giữa các người chơi và kết nối họ với nhau.
* Chuyển tiếp dữ liệu về nước cờ giữa các người chơi.
* Ghi dữ liệu log về ván chơi.
* Xử lý nghiệp vụ, thông báo cho người chơi còn lại nếu đối thủ đang chơi ngắt kết nối.

1. Game Logic:

* Bản đồ có kích thước 10x10.
* Điểm bắn được định nghĩa bằng vị trí hàng – cột
* Người chơi đặt tàu theo vị trí đầu tàu và hướng ( lên, xuống, trái, phải)
* Hai bên bắn qua lại cho đến khi một bên đã hỏng hết tàu.

1. **Phương hướng giải quyết:**

* Sử dụng selectserver để kết nối và xử lý nhiều dữ liệu gửi từ các client khác nhau đến server.
* Kết hợp với logic game từ game.c để tiến hành kiểm tra tính hợp lý trong quá trình thao tác trò chơi và quyết định kết quả trò chơi.

1. **Phân công công việc:**

* **Tạ Duy Hưng**: thiết kế Use Cases, Flowchart của chương trình và phần giao diện, logic, giao tiếp qua server của trò chơi Battleship.
* **Trần Quang Duy**: xây dựng các chức năng và giao diện tương tác giữa client và server, sử dụng selectserver để kết nối và xử lý với nhiều client cùng lúc.

1. **Cơ sở lý thuyết:**
   1. Khái niệm:

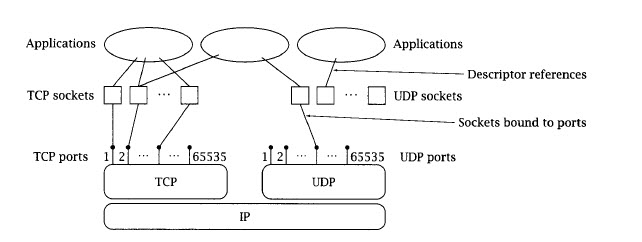
Socket là một khái niệm trừu tượng mà qua đó một ứng dụng có thể gửi và nhận dữ liệu, tương tự như việc ta thực hiện một cuộc điện thoại (có thể coi socket là những chiếc điện thoại). Một ứng dụng có thể sử dụng nhiều socket cùng một lúc.

Socket có nhiểu loại khác nhau tủy thuộc vào từng họ giao thức và stacks của họ giao thức khác nhau. Ngày nay giao thức TCP/IP là một loại giao thức phổ biến và được sử dụng rộng rãi trên Internet. Trong giao thức TCP/IP có hai loại socket chính là Stream Sockets và Datagram Sockets.

* + ***Stream Socket****: Dựa trên giao thức TCP (Tranmision Control Protocol) việc truyền dữ liệu chỉ thực hiên trên hai quá trình đã thiết lập kết nối. Giao thức này đảm bảo dữ liệu được truyền đến nơi nhận một cách đáng tin cậy, đúng thứ tự nhờ vào cơ chế quản lí luồng lưu thông trên mạng và chống tắc nghẽn.*
  + ***Datagam Socket****: Dựa trên giao thức UDP (User Datagram Protocol) việc truyền dữ liệu không yêu cầu có sự thiết lập kết nối giữa hai quá trình. Ngược lại giao thức TCP thì dữ liệu được truyền theo giao thức UDP không được tin cậy có thể không đúng trình tự và lặp lại. Tuy vậy nó không yêu cầu thiết lập kết nối không có những cơ chế phức tạp nên tốc độ nhanh.*

1. Cấu trúc địa chỉ Socket

Một Socket sử dụng giao thức TCP/IP được định nghĩa duy nhất bởi một địa chỉ Internet, một giao thức end-to-end (TCP hoặc UDP) và một số hiệu cổng (port number).

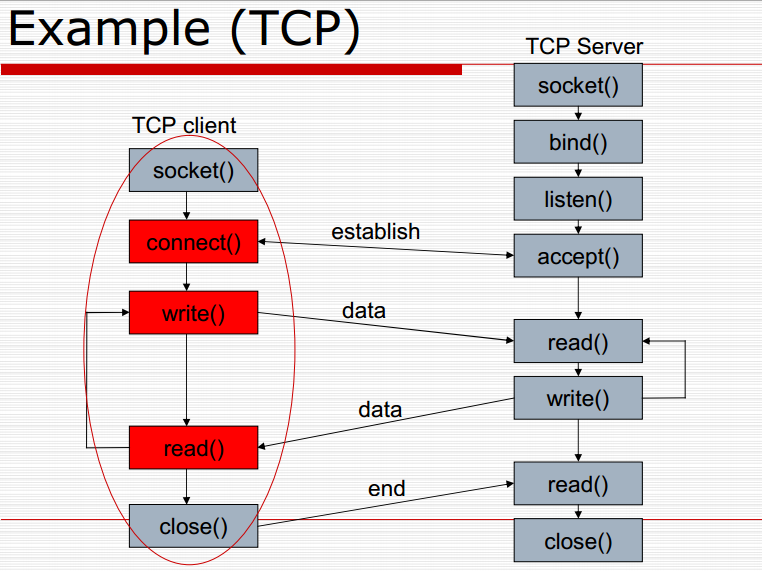


Sockets, protocols và ports

Hình trên miêu tả quan hệ logic giữa ứng dụng, socket, giao thức, và số hiệu cổng cho ứng dụng.

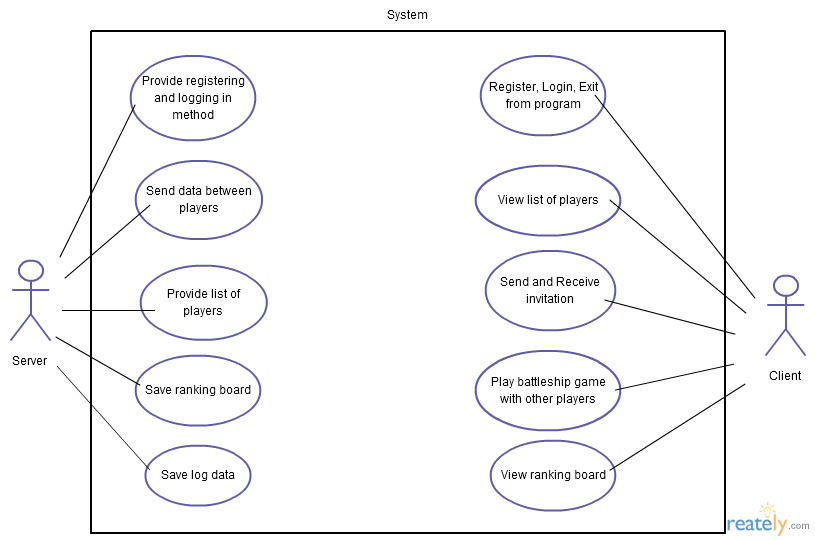
Hầu hết các hàm socket đều yêu cầu một con trỏ tới một cấu trúc địa chỉ socket như là một tham số. Mỗi họ giao thức hỗ trợ đều định nghĩa cấu trúc địa chỉ socket riêng của nó. Một cấu trúc địa chỉ socket là một cấu trúc chứa thông tin về một socket để tạo hoặc kết nối tới nó. Có hai loại cấu trúc địa chỉ socket cơ bản là IPv4 và IPv6.

1. Stream Socket (TCP)



* Socket: tạo một socket dựa trên domain, type, protocol. Trả về một Socket ID.
* Bind:đơn giản là gán một số hiệu cổng cho socket. (Vd: 192.168.1.1:80).
* Listen: thông báo cho hệ thống cho phép kết nối được thực hiện trên cổng đó (cổng đã được gán cho socket). Số kết nối ”treo” được tạo thành một hàng đợi để lắng nghe.
* Accept: server chấp nhận kết nối từ yêu cầu của một client nào đó. Có ba cách cơ bản để chấp nhận kết nối:
  + *Interating Server: chỉ một socket được mở tại một thời điểm (chỉ thực hiện được môt kết nối).*
  + *Forking server: Sau khi chấp nhận kết nối một tiến trình con được tạo ra (bằng cách gọi hàm fork()) để xử lí kết nối. Ta sẽ nói rõ ở phần sau.*
  + *Selecting server (concurrent Single Server): dùng select để đồng thời chờ trên tất cả các socket đang mở, và bắt đầu tiến trình khi có dữ liệu đến.*
* Connect: client yêu cầu kết nối tới server
* Send: Gửi một thông điệp tới socket
* Recv: Nhận dữ liệu từ một socket
* Close: Đóng Socket.

1. Các APIs hỗ trợ
2. *gethostname: trả vể tên của hệ thống*
3. *gethostbyname: lấy về một địa chỉ IP cho một hostname hoặc ngược lại.*
4. *htons, htonl, ntohs, ntohl: byte ordering*
5. *inet\_ntoa(), inet\_aton(), inet\_addr: biến đổi địa chỉ IP từ một chuỗi dots-and-number (vd: 192.168.1.1) sang một cấu trúc in\_addr và ngược lại.*
6. *inet\_ntop(), inet\_pton(): biến đổi địa chỉ IP từ dạng binary sang dạng human-readable.*
7. **Thiết kế chính của chương trình:**
8. Các cấu trúc và chức năng chính:



*Use Cases Diagram*

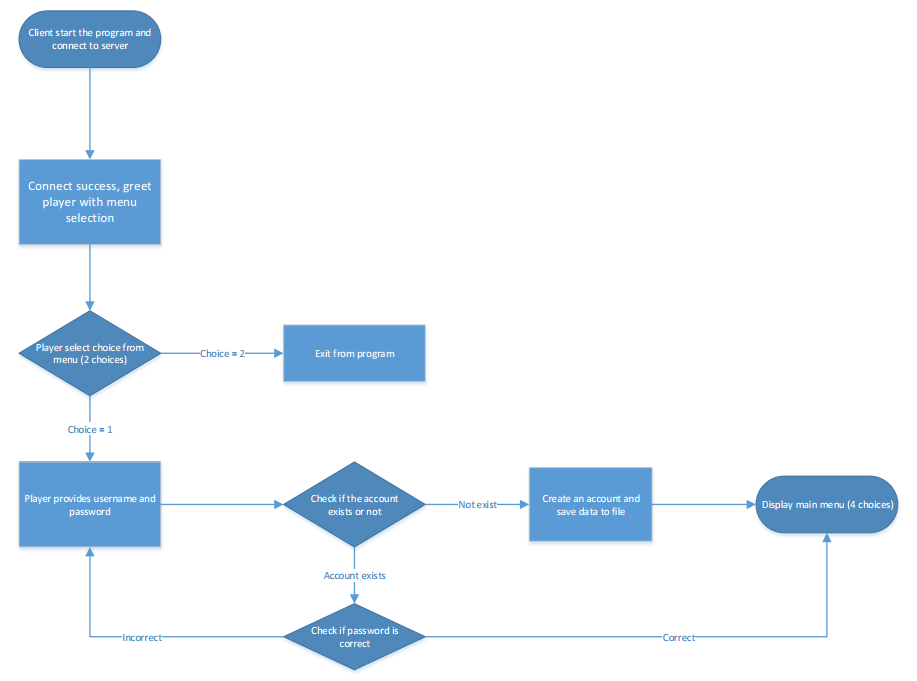
- Hình minh họa Use Cases tương ứng với các yêu cầu được đặc tả ở phần trên.

**

*Objects (Structs)*

* Hình minh họa thiết kế các objects được định nghĩa trong chương trình:
  + Player:
    - fd: giá trị socket của client đang đăng nhập player
    - win, lose: số ván thắng, thua
    - status: trạng thái (0: tự , 1: chờ lời mời, 2: đang chơi)
    - ip: ip của client đang đăng nhập player
  + Match:
    - Lưu thông tin hai người chơi trong trận đấu và log ghi lại trận đấu.
  + Coord:
    - x: tọa độ hàng
    - y: tọa độ cột
  + Ship: (không sử dụng)
    - head: điểm đặt tàu
    - direction: hướng tàu
    - length: độ dài tàu

1. Luồng xử lí chương trình:

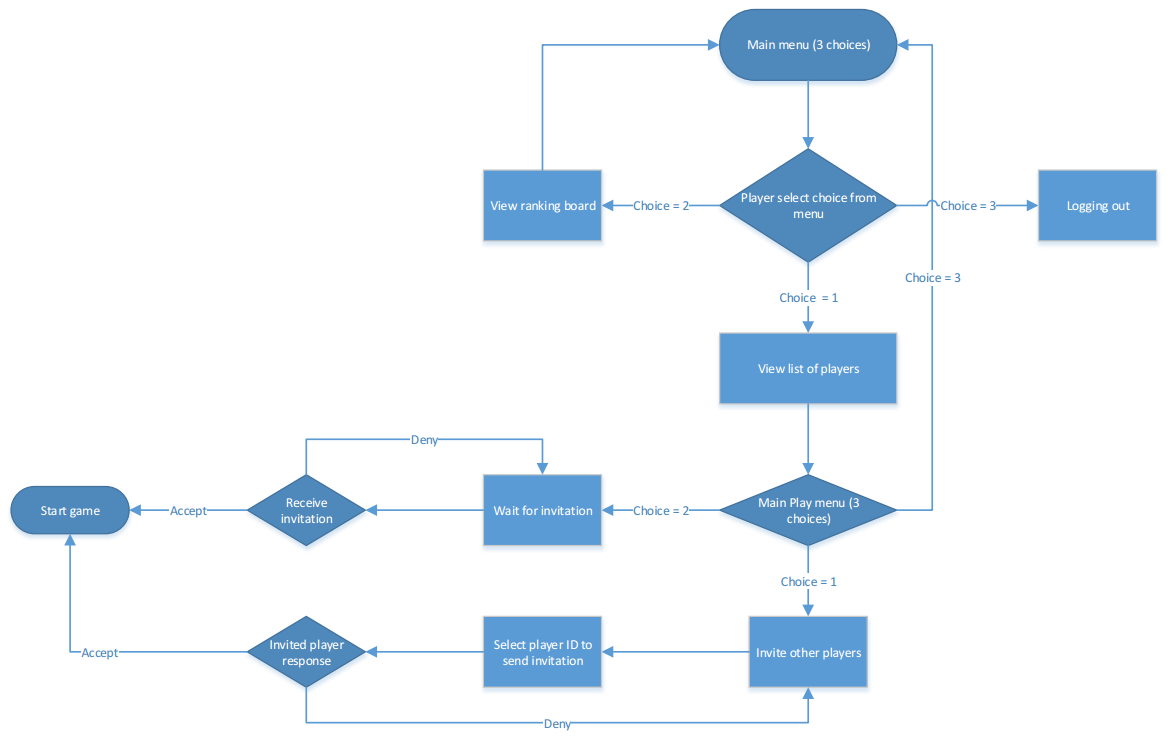


Khi người chơi khởi động chương trình, họ sẽ cần điền thông tin tài khoản (username và password để login vào hệ thống). Khi chứng thực được người dùng, chương trình sẽ chuyển sang màn hình main menu.

Người chơi sau khi đăng nhập được phép lựa chọn một số tùy chọn sau:

* 1. Xem danh sách người chơi
  2. Xem bảng xếp hang
  3. Đăng xuất

Với lựa chọn xem danh sách người chơi đang đăng nhập vào hệ thống trong thời điểm hiện tại, người chơi tiếp tục có thể tùy chọn mời một người chơi vào bàn hoặc đợi cho đến khi có ai đó mời mình vào chơi.



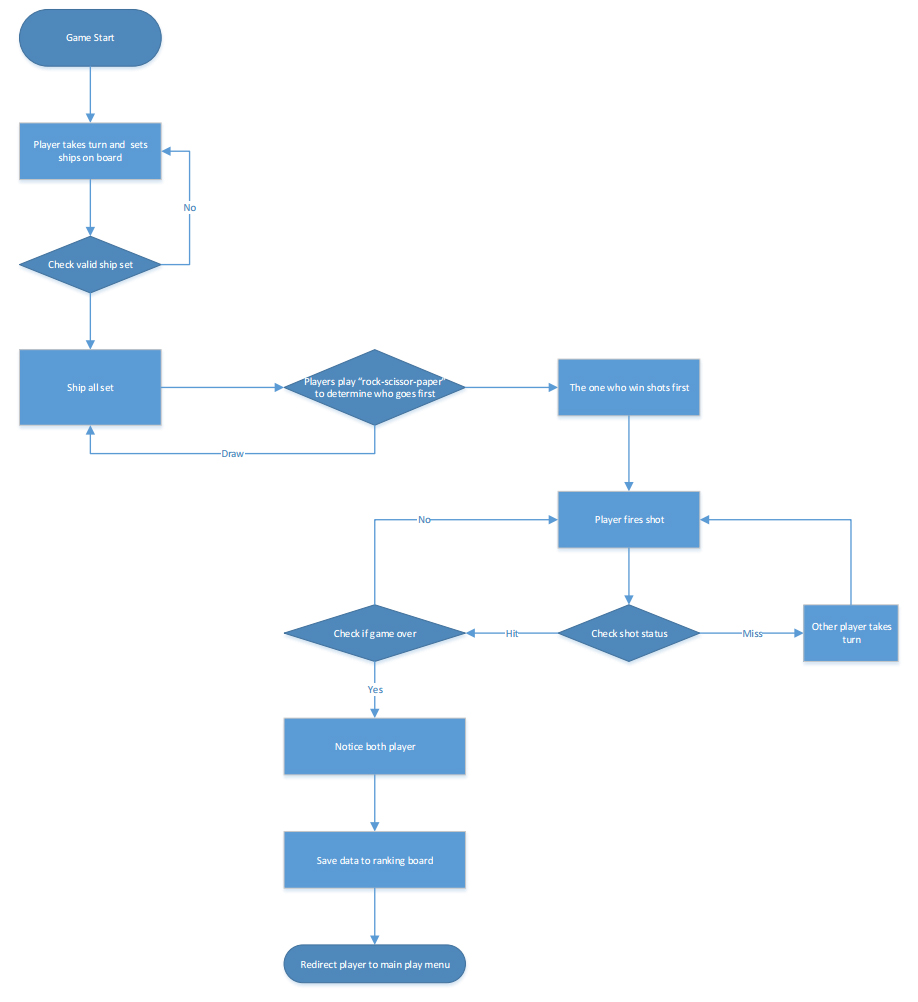
Khi một lời mời chơi được gửi đến người dùng, anh ta có thể quyết định đồng ý hoặc từ chối. Nếu như, chấp nhận lời mời, một thông điệp sẽ được gửi lên server và lúc này server sẽ xử lí thông điệp gửi lên, tạo màn chơi và kết nối 2 người chơi vào màn chơi.

Mọi xử lí logic game sau đó cũng sẽ được server game xử lí bằng các thông điệp lần lượt được gửi lên từ 2 client.

Người chơi có thể:

* Đặt vị trí tàu trên bản đồ theo tọa độ hàng – cột Đặt theo vị trí đầu tàu và chọn hướng ( lên, xuống, trái, phải)
* Hai bên bắn qua lại, điểm bắn được định nghĩa bằng vị trí hàng – cột.
* Kết quả phát bắn sẽ được trả về cho người bắn sau khi thực hiện phát bắn.
* Trò chơi kết thúc khi có một bên bị mất hết tàu.

Sau khi kết thúc một màn chơi, 2 người chơi được đưa về main menu và trở lại trạng thái rảnh rỗi. Họ có thể tiếp tục mời chơi, ngồi chờ hoặc đăng xuất ra khỏi hệ thống.



1. **Xây dựng chương trình:**
2. Giao tiếp client và server:

* Server sử dụng serverselect để tiếp nhận kết nối, dữ liệu và tình trạng kết nối với từng client.
* Tiếp nhận các message với nội dung được format trước và xử lý nghiệp vụ, đồng thời trả về dữ liệu cần thiết mà client yêu cầu.

1. Logic trò chơi:

* Sau khi hai client được kết nối với nhau, lúc này server chỉ chuyển trực tiếp message từ client này đến client kia với format định trước như sau:
  + Tọa độ hàng: message[0] với người đi trước và message[3] với người đi sau.
  + Tọa độ cột: emssage[2] với người đi trước và message[5] với người đi sau.
  + Kết quả phát bắn: message[1] với người đi trước và message[4] với người đi sau.
* Bắn cho đến khi một bên không còn tàu, bên không còn tàu sẽ gửi thông báo thua cuộc lên server và server trả về thông báo thắng cuộc cho người chơi còn lại.
* Đưa ra thông báo nếu có người chơi bị ngắt kết nối lúc đang chơi đối với người còn lại.

1. **Kết quả công việc:**
2. Các chức năng thực hiện hoàn chỉnh:

* Đăng nhập
* Xem danh sách người dùng đang online và tình trạng của họ
* Gửi, chờ và nhận lời mời giữa các người chơi.
* Đặt chỗ các tàu trên bàn cờ, kiểm tra tính hợp lý của việc đặt tàu.
* Hai người dùng trong một trận đấu có thể bắn qua bắn lại, cập nhật tình trạng bản đồ cho đến khi kết thúc trận đấu.
* Ghi dữ liệu trận đấu vào file log, cập nhật số trận thắng, thua của người chơi.

1. Các chức năng chưa kịp hoàn thiện:

* Đăng ký
* Xem bảng xếp hạng người chơi

1. Screen shots:



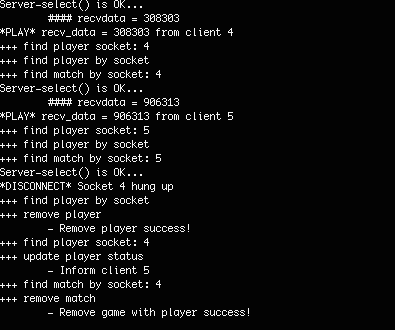
*Đặt tàu*

- Hình minh họa người chơi đặt tàu thứ nhất với độ dài là 2 với đầu tàu đặt ở tọa độ hàng 1 cột 1 hướng sang bên phải và bản đồ vị trí tàu sau khi được đặt.



*Menu*

- Hình minh họa quá trình chơi của hai player cho đến khi có một người bị disconnect khỏi server vì một lý do nào đó, màn hình trở về menu chính; bản đồ trò chơi với ‘X’ là điểm bắn trúng, ‘O’ là điểm bắn trượt, ‘\*’ là vị trí tàu và ‘.’ là biển hay vị trí không xác định.



*Server log*

- Hình minh họa các thông điệp gửi, nhận giữa server và client trên màn hình log của server với các tiến trình chính nằm giữa hai dấu ‘\*’, các hàm trong struct.c được gọi bắt đầu bằng dấu ‘+++’, dữ liệu truyền lên trong game từ client bắt đầu với ‘####’ và một số thông báo khác.