**LAB 3**

III.1. Xây dựng chương trình UDP Client – Server đơn giản

***1) Như hình trên port 2747 ở đâu ra***

Port này là **port ngẫu nhiên (ephemeral port)** được hệ điều hành tự động cấp cho client khi client tạo socket để gửi dữ liệu. Nó không được lập trình viên chỉ định.

***2) Có phải lúc nào client cũng mở port 2747 để kết nối với Server không?***

Không. Mỗi lần chương trình client chạy, hệ điều hành có thể cấp **một port khác nhau**, miễn là port đó đang rảnh và không trùng với port đang dùng.

***3) Tại sao khi lập trình mạng dùng giao thức UDP thì client phải gởi câu chào lên server  
trước?***

Vì UDP **không có kết nối** như TCP, server không biết client nào đang tồn tại nếu client chưa gửi dữ liệu. Khi client gửi gói đầu tiên (“câu chào”), server mới biết địa chỉ và port của client để phản hồi lại.

III.3. Cải tiến chương trình UDP client-server để có thể gởi và nhận dữ  
liệu liên tụ

***1) Khi chạy chương trình với đoạn code gởi nhận dữ liệu như trên, lúc chưa nhập dữ liệu cho client để gởi lên thì không xảy ra lỗi nhưng khi nhập dữ liệu để gởi lên server sẽ xảy ra lỗi, vì sao lại xảy ra lỗi này?***

Vì biến remote (chứa địa chỉ server) **chưa được khởi tạo hoặc truyền sai kiểu tham chiếu** khi gọi SendTo hoặc ReceiveFrom, khiến chương trình không biết gửi dữ liệu đến đâu.

***2) Khi server chưa bật thì chương trình trên có bị lỗi không? Tại sao?***

Không bị lỗi nghiêm trọng, vì UDP chỉ gửi gói tin đi mà **không kiểm tra xem server có nhận hay không**. Tuy nhiên, client sẽ không nhận được phản hồi nào.

***3) Khi đang chạy chương trình tắt client thì chương trình trên có bị lỗi không? Tại sao?***

Server có thể bị lỗi hoặc hiển thị ngoại lệ nếu vẫn cố đọc/gửi dữ liệu đến client đã ngắt kết nối, vì **địa chỉ client không còn hợp lệ**.

III.4. Sử dụng phương thức Connect ở client để thiết lập kết nối trước với server

***1) Khi chạy chương trình mà server chưa được bật thì có hiện tượng gì xảy ra? Tại sao lại có hiện tượng này?***

Gọi Connect() sẽ khiến socket UDP **thử thiết lập kết nối logic**. Nếu server chưa bật, khi client gửi dữ liệu sẽ báo lỗi **“SocketException: No connection could be made”** vì không có ai lắng nghe ở đầu kia.

***2) Khi đang chạy tắt server thì chương trình trên có bị lỗi không? Tại sao?***

Có thể bị lỗi SocketException do client **gửi đến địa chỉ không còn phản hồi** (server đã tắt). UDP không kiểm tra đường truyền, nên lỗi thường chỉ xuất hiện khi gọi Receive.

***3) Khi đang chạy chương trình tắt client thì chương trình trên có bị lỗi không? Tại sao?***

Server sẽ không bị lỗi, chỉ là **không nhận thêm gói tin nào từ client** nữa. UDP không cần đóng kết nối.

III.5. Kiểm tra khả năng phân biệt biên thông điệp của giao thức UDP

***1) Tại sao khi lập trình bằng giao thức UDP các thông điệp được phân biệt với nhau?***

Vì UDP **làm việc theo từng gói tin (datagram)**, mỗi lần gửi là một gói độc lập, có độ dài và ranh giới riêng. Hệ thống mạng đảm bảo mỗi gói nhận được sẽ trọn vẹn, không bị nối dính như TCP.

III.6. Ngăn cản mất dữ liệu khi lập trình mạng sử dụng giao thức UDP

***1) Khi gởi dữ liệu với kích thước lớn hơn 10 byte thì có lỗi xảy ra không? Tại sao?***

Có. Vì bộ đệm nhận (buff = new byte[10]) quá nhỏ, dữ liệu bị cắt hoặc mất phần dư. UDP không có cơ chế tự chia nhỏ và ghép lại gói dữ liệu lớn, nên dữ liệu vượt quá kích thước bộ đệm sẽ bị mất.

***2) Khi dùng phương pháp này các dữ liệu bị mất do kích thước bộ đệm ban đầu nhỏ có lấy lại được không?***

Không hoàn toàn. Phương pháp chỉ giúp **phát hiện lỗi và gửi lại dữ liệu**, chứ không thể khôi phục phần đã mất nếu gói tin không được gửi lại đúng cách hoặc bị lỗi trong quá trình truyền.

III.7. Ngăn cản mất gói tin khi lập trình mạng sử dụng giao thức UDP

***3) Sử dụng phương pháp này có ngăn cản triệt để mất dữ liệu không? Nếu không hãy đề  
xuất phương pháp khác***

Không triệt để. Vì nếu mất gói tin nhiều lần liên tiếp, client sẽ vẫn không nhận được phản hồi sau số lần retry giới hạn.  
Giải pháp khác:

* **Thêm cơ chế xác nhận (ACK)** giữa client–server.
* **Đánh số thứ tự gói tin** để biết gói nào bị mất và chỉ gửi lại gói đó.
* Hoặc sử dụng **TCP** – giao thức có sẵn cơ chế kiểm soát lỗi, đảm bảo truyền dữ liệu tin cậy.