

**BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ ĐÔNG Á
KHOA: CÔNG NGHỆ THÔNG TIN**



BÀI TẬP LỚN

HỌC PHẦN: LẬP TRÌNH MẠNG

ĐỀ SỐ 05: ỨNG DỤNG CHẤT NHÓM TCP SOCKET CONSOLE

Sinh viên thực hiện	Lớp	Khóa
Lê Quý Mùi	DCCNTT12.10.4	K12
Phạm Hồng Quân	DCCNTT12.10.4	K12
Nguyễn Văn Lâm	DCCNTT12.10.4	K12
Vì Văn Tuấn	DCCNTT12.10.4	K12

Bắc Ninh, năm 2023

**BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ ĐÔNG Á
KHOA: CÔNG NGHỆ THÔNG TIN**

BÀI TẬP LỚN

HỌC PHẦN: LẬP TRÌNH MẠNG

Nhóm: 4

ĐỀ SỐ 05: ỨNG DỤNG CHẤT NHÓM TCP SOCKET CONSOLE

STT	Sinh viên thực hiện	Mã sinh viên	Điểm bằng số	Điểm bằng chữ
1	Lê Quý Mùi	20211133		
2	Nguyễn Hồng Quân	20211180		
3	Nguyễn Văn Lâm	20211166		
4	Vi Văn Tuấn	20211138		

CÁN BỘ CHẤM 1

(Ký và ghi rõ họ tên)

CÁN BỘ CHẤM 2

(Ký và ghi rõ họ tên)

Mục Lục

MỞ ĐẦU.....	4
Chương I: Cơ Sở Lý Thuyết.....	5
1. Chức năng chính của 7 tầng trong mô hình OSI	5
2. Các tầng của TCP/IP	8
3. Socket	13
4. Các lớp socket	15
5. Multithread	17
5.1 Khái niệm	17
5.2 Vòng đời của Thread trong C#.....	17
5.3 Main Thread trong C#.....	17
5.4 Thuộc tính và Phương thức của lớp Thread trong C#.....	18
Chương II: Xây dựng ứng dụng Server/Client.....	23
1. Phần Server.....	23
2. Phần Client	27
3. Giao diện console	30
Server :.....	30
Client :	31
Chương III: Kết quả thực nghiệm chương trình.....	32
Giao diện client :	32
Giao diện server :.....	36
Kết luận	38
Ưu điểm:	38
Nhược điểm:	38
Phần tài liệu tham khảo	39

MỞ ĐẦU

Trong thời đại công nghệ thông tin phát triển như hiện nay, ứng dụng chat nhóm đã trở thành một phương tiện giao tiếp không thể thiếu trong cuộc sống cũng như trong các tổ chức, doanh nghiệp. Với sự phổ biến của mạng LAN, việc lập trình một ứng dụng chat nhóm cho mạng LAN là điều hết sức cần thiết.

Thông qua bài viết này, chúng ta sẽ tìm hiểu cách lập trình ứng dụng chat nhóm trong mạng LAN sử dụng giao thức TCP Socket chi tiết hơn. Đầu tiên, chúng ta sẽ cùng nhau thiết kế giao diện người dùng để người dùng có thể đăng nhập vào hệ thống và chọn phòng chat.

Sau đó, sử dụng TCP Socket để thiết lập kết nối đến máy chủ. Máy chủ sẽ giữ kết nối với tất cả các thành viên tham gia phòng chat và phát lại các tin nhắn đến tất cả các thành viên trong phòng chat.

Người dùng có thể đăng nhập vào phòng chat bằng cách chọn phòng chat từ danh sách và nhập tên đăng nhập. Sau khi đăng nhập thành công, người dùng có thể gửi và nhận tin nhắn từ các thành viên khác trong phòng chat. Các tin nhắn được gửi đến máy chủ và phát lại cho tất cả các thành viên trong phòng chat.

Khi người dùng muốn rời khỏi phòng chat, họ có thể chọn nút rời khỏi phòng chat để kết thúc kết nối của mình và không nhận được các tin nhắn mới. Khi người dùng muốn thoát khỏi ứng dụng, họ có thể đăng xuất khỏi tài khoản của mình để kết thúc kết nối và thông tin đăng nhập của họ sẽ bị xóa.

Để triển khai ứng dụng này, chúng ta có thể sử dụng các ngôn ngữ lập trình phổ biến như Java hoặc Python. Sự hỗ trợ của các công cụ phát triển như Eclipse, Visual Studio Code hay PyCharm giúp cho việc lập trình ứng dụng trở nên dễ dàng hơn.

Hy vọng rằng bài viết sẽ giúp các bạn hiểu rõ hơn về cách triển khai ứng dụng chat nhóm trong mạng LAN sử dụng giao thức TCP Socket và áp dụng kiến thức này vào thực tiễn trong cuộc sống và công việc của mình.

Chương I: Cơ Sở Lý Thuyết

1. Chức năng chính của 7 tầng trong mô hình OSI

- Mô hình OSI gồm 7 tầng và mỗi tầng có những chức năng chính khác nhau để giải quyết các vấn đề trong giao thức truyền thông. Chức năng chính cụ thể của 7 tầng trong mô hình OSI:

- **Physical Layer (tầng vật lý)**

Physical Layer là tầng thấp nhất trong mô hình 7 lớp OSI. Các thực thể tầng giao tiếp với nhau qua một đường truyền vật lý. Tầng vật lý xác định các thủ tục, chức năng về điện, quang, cơ để kích hoạt và duy trì các kết nối vật lý giữa các hệ thống mạng.

Cung cấp các cơ chế về hàm, điện, thủ tục,...nhằm kết nối các phần tử của mạng thành một hệ thống bằng các phương pháp vật lý. Đảm bảo các yêu cầu về chuyển mạch hoạt động, tạo ra các đường truyền thực cho các chuỗi bit thông tin.

Các chuẩn trong tầng vật lý là các chuẩn xác định giao diện người sử dụng cũng như môi trường mạng. Physical Layer gồm 2 loại: truyền dị bộ và truyền đồng bộ.

- **Data Link Layer (tầng liên kết dữ liệu)**

Tầng liên kết dữ liệu có chức năng chính là thực hiện thiết lập các liên kết, duy trì hay hủy bỏ liên kết dữ liệu. Kiểm soát lỗi và kiểm soát lưu lượng.

Thông tin được chia thành các khung, truyền các khung tuần tự và xử lý các thông điệp xác nhận được gửi về từ máy thu. Tháo gỡ các khung thành chuỗi bit thành các khung thông tin.

Do đường truyền vật lý có thể gây ra lỗi, vì thế tầng liên kết dữ liệu phải giải quyết vấn đề kiểm soát lỗi, kiểm soát lưu lượng, luồng, ngăn không để nút nguồn gây ngập dữ liệu cho bên thu có tốc độ thấp hơn.

Tầng con MAC điều khiển việc duy trì nhập đường truyền trong các mạng quảng bá.

- **Network Layer (tầng mạng)**

Trách nhiệm của lớp mạng Network là quyết định xem dữ liệu sẽ đến máy nhận như thế nào. Lớp này nắm các thành phần như việc xác định tuyến, địa chỉ và các giao thức logic.

Lớp mạng này tạo các đường logic được biết đến như các mạch ảo giữa máy nguồn và máy đích. Mạch ảo này cung cấp các gói dữ liệu riêng lẻ giúp chúng đến được đích của

chúng. Ngoài ra, lớp mạng cũng chịu trách nhiệm cho việc quản lý lỗi của chính nó, cho việc điều khiển xếp chuỗi và điều khiển việc tắc nghẽn.

Việc sắp xếp các gói rất cần thiết vì mỗi giao thức giới hạn kích thước tối đa của mỗi gói. Thường thì số lượng dữ liệu phải truyền đi vượt quá kích thước gói lớn nhất. Vì thế, dữ liệu được chia nhỏ thành nhiều gói nhỏ. Khi điều đó xảy ra, mỗi gói nhỏ này sẽ được lớp mạng gán vào một số thứ tự nhận dạng.

Lớp mạng sẽ kiểm tra số thứ tự nhận dạng của các gói khi dữ liệu này đến được máy tính người nhận và sử dụng chúng để sắp xếp các dữ liệu đúng như những gì mà chúng được chia từ phía người gửi lúc trước. Ngoài ra, còn có nhiệm vụ chỉ ra gói nào bị thiếu trong quá trình gửi.

Hiểu đơn giản thì nó giống như việc bạn gửi mail tài liệu có dung lượng lớn cho bạn của mình nhưng không có phong bì đủ lớn. Và để giải quyết vấn đề đó thì bạn cần chia nhỏ một số trang vào các phong bì nhỏ và sau đó dán nhãn các phong bì lại và điền số thứ tự để giúp bạn mình biết trình tự sắp xếp. Điều này cũng giống như những gì mà lớp mạng thực hiện.

- **Transport Layer (tầng vận chuyển)**

Tầng vận chuyển là tầng cao nhất có liên quan đến các giao thức trao đổi dữ liệu giữa các hệ thống mở và kiểm soát việc truyền dữ liệu từ nút tới nút. Thủ tục trong 3 tầng: vật lý, liên kết dữ liệu và mạng network chỉ phục vụ việc truyền dữ liệu giữa các tầng kề nhau ở từng hệ thống. Các thực thể trong một tầng hội thoại thương lượng với nhau trong quá trình truyền dữ liệu.

Transport Layer thực hiện việc chia các gói tin lớn thành các gói nhỏ hơn và đánh số các gói tin theo đúng số thứ tự trước khi được gửi đi. Giao thức tầng vận chuyển là tầng cuối cùng chịu trách nhiệm về mức độ an toàn trong truyền dữ liệu nên phụ thuộc nhiều vào bản chất của tầng mạng. Tầng vận chuyển cũng có thể thực hiện việc ghép kênh một vài liên kết vào cùng một để giảm giá thành.

- **Session Layer (tầng phiên)**

Session Layer cho phép người dùng trên máy tính khác nhau thiết lập, duy trì và đồng bộ phiên truyền thông giữa họ với nhau. Nghĩa là các tầng phiên thiết lập các giao dịch giữa các thực thể đầu cuối.

Dịch vụ phiên cung cấp liên kết giữa 2 đầu cuối sử dụng dịch vụ phiên nhằm trao đổi một cách đồng bộ dữ liệu và giải phóng liên kết khi kết thúc. Sử dụng thẻ bài để thực hiện truyền dữ liệu, đồng bộ, hủy bỏ liên kết trong các phương thức truyền đồng thời hay luân

phiên. Thiết lập các điểm đồng bộ hóa trong hội thoại để khi xảy ra sự cố có thể khôi phục hội thoại từ 1 điểm đồng bộ hóa đã thỏa thuận.

- **Presentation Layer (tầng trình bày)**

Tầng trình bày có chức năng giải quyết các vấn đề liên quan đến các ngữ nghĩa và cú pháp của thông tin được truyền đi. Biểu diễn thông tin làm việc phù hợp với thông tin người sử dụng và ngược lại. Thông thường các ứng dụng có thể được chạy trên hệ thống khác nhau nên việc biểu diễn thông tin các ứng dụng nguồn và ứng dụng đích có thể khác nhau.

Tầng trình bày có trách nhiệm chuyển đổi dữ liệu gửi đi trên mạng từ một loại biểu diễn này sang loại biểu diễn khác. Để đạt được điều đó thì tầng trình bày cung cấp một dạng biểu diễn truyền thông chung cho phép chuyển đổi sang bộ biểu diễn chung từ dạng biểu diễn cục bộ và ngược lại.

- **Application Layer (tầng ứng dụng)**

Application Layer là lớp trên cùng của mô hình OSI có chức năng chính là hỗ trợ ứng dụng và các tiến trình có liên quan đến người sử dụng cuối. Tại lớp này các vấn đề về chất lượng phục vụ, đối tác truyền thông, xác thực người dùng, quyền riêng tư hay bất kỳ ràng buộc nào về cú pháp dữ liệu đều sẽ được xem xét và xác định.

Tại tầng ứng dụng tất cả mọi thứ được cụ thể thành ứng dụng. Lớp này cung cấp các dịch vụ ứng dụng cho truyền email, file hay các dịch vụ phần mềm mạng khác. Một số ứng dụng nằm hoàn toàn trong cấp Application như dịch vụ Web, FTP, Telnet, còn kiến trúc ứng dụng phân tầng là một phần của lớp này.

Tuy nhiên, bạn cũng cần biết rằng lớp 7 này không ám chỉ đến các ứng dụng người dùng đang chạy mà chỉ cung cấp nền tảng làm việc ứng dụng đó chạy bên trên.

Để hiệu rõ hơn về ứng dụng này, chúng ta có thể giả dụ một người dùng nào đó muốn sử dụng Internet Explorer để mở một FTP session và truyền tải một file. Ở trường hợp này thì lớp ứng dụng sẽ định nghĩa một giao thức truyền tải. Người dùng cuối này vẫn phải sử dụng ứng dụng được thiết kế vì giao thức này không thể truy cập trực tiếp đến người dùng cuối để tương tác với giao thức truyền tải file. Trong trường hợp này, Internet Explorer sẽ làm ứng dụng đó.

TCP/IP hoặc Transmission Control Protocol/Internet Protocol (Giao thức điều khiển truyền vận/giao thức mạng) là một bộ các giao thức trao đổi thông tin được sử dụng để kết nối các thiết bị mạng trên Internet. TCP/IP có thể được sử dụng như là một giao thức trao đổi thông tin trong một mạng riêng ([intranet](#) hoặc extranet)

Toàn bộ bộ giao thức Internet - một tập hợp các quy tắc và thủ tục - thường được gọi là TCP/IP, mặc dù trong bộ cũng có các giao thức khác.

TCP/IP chỉ định cách dữ liệu được trao đổi qua Internet bằng cách cung cấp thông tin trao đổi đầu cuối nhằm mục đích xác định cách thức nó được chia thành các gói, được gắn địa chỉ, vận chuyển, định tuyến và nhận ở điểm đến. TCP/IP không yêu cầu quản lý nhiều và nó được thiết kế để khiến mạng đáng tin cậy hơn với khả năng phục hồi tự động.

Có hai giao thức mạng chính trong bộ giao thức mạng phục vụ các chức năng cụ thể.

- **TCP** xác định cách các ứng dụng tạo kênh giao tiếp trong mạng. Ngoài ra, nó cũng quản lý cách các tin được phân thành các gói nhỏ trước khi được chuyển qua Internet và được tập hợp lại theo đúng thứ tự tại địa chỉ đến.
- **IP** xác định cách gán địa chỉ và định tuyến từng gói để đảm bảo nó đến đúng nơi. Mỗi gateway trên mạng kiểm tra [địa chỉ IP](#) này để xác định nơi chuyển tiếp tin nhắn.
- TCP/IP sử dụng mô hình giao tiếp máy khách/máy chủ, trong đó người dùng hoặc thiết bị (máy khách) được một máy tính khác (máy chủ) cung cấp một dịch vụ (giống như gửi một trang web) trong mạng.
- Nói chung, bộ giao thức TCP/IP được phân loại là không có trạng thái, có nghĩa là mỗi yêu cầu của máy khách được xem là mới bởi vì nó không liên quan đến yêu cầu trước. Việc không có trạng thái này giúp giải phóng đường mạng, do đó chúng có thể được sử dụng liên tục.
- Tuy nhiên, tầng vận chuyển lại có trạng thái. Nó truyền một tin nhắn duy nhất và kết nối của nó vẫn giữ nguyên cho đến khi nhận được tất cả các gói trong tin nhắn và tập trung tại điểm đến.

Mô hình TCP/IP hơi khác so với [mô hình OSI](#) (Open Systems Interconnection - Mô hình kết nối các hệ thống mở) bảy lớp được thiết kế sau nó, nó xác định cách các ứng dụng giao tiếp trong một mạng.

2. Các tầng của TCP/IP

TCP/IP được chia thành bốn tầng, mỗi tầng bao gồm các giao thức cụ thể.

- **Tầng ứng dụng** cung cấp các ứng dụng với trao đổi dữ liệu được chuẩn hóa. Các giao thức của nó bao gồm Giao thức truyền tải siêu văn bản (HTTP), Giao thức truyền tập tin (File Transfer Protocol - FTP), Giao thức POP3, Giao thức

truyền tải thư tín đơn giản (Simple Mail Transfer Protocol - SMTP) và Giao thức quản lý mạng đơn giản (Simple Network Management Protocol - SNMP).

- **Tầng giao vận** chịu trách nhiệm duy trì liên lạc đầu cuối trên toàn mạng. TCP xử lý thông tin liên lạc giữa các máy chủ và cung cấp điều khiển luồng, ghép kênh và độ tin cậy. Các giao thức giao vận gồm giao thức TCP và giao thức UDP (User Datagram Protocol), đôi khi được sử dụng thay thế cho TCP với mục đích đặc biệt.
- **Tầng mạng**, còn được gọi là tầng Internet, có nhiệm vụ xử lý các gói và kết nối các mạng độc lập để vận chuyển các gói dữ liệu qua các ranh giới mạng. Các giao thức tầng mạng gồm IP và ICMP (Internet Control Message Protocol), được sử dụng để báo cáo lỗi.
- **Tầng vật lý** bao gồm các giao thức chỉ hoạt động trên một liên kết - thành phần mạng kết nối các nút hoặc các máy chủ trong mạng. Các giao thức trong lớp này bao gồm Ethernet cho mạng cục bộ (LAN) và Giao thức phân giải địa chỉ (Address Resolution Protocol - ARP).
- **Ưu điểm của TCP/IP**

TCP/IP không thuộc và chịu sự kiểm soát của bất kỳ công ty nào, do đó bộ giao thức mạng này có thể dễ dàng sửa đổi. Nó tương thích với tất cả các hệ điều hành, vì vậy có thể giao tiếp với các hệ thống khác. Ngoài ra, nó còn tương thích với tất cả các loại phần cứng máy tính và mạng.

TCP/IP có khả năng mở rộng cao và như một giao thức có thể định tuyến, nó có thể xác định đường dẫn hiệu quả nhất thông qua mạng.

– So Sánh 2 giao thức TCP và UDP.

- **UDP (User Datagram Protocol)** là một trong những giao thức cốt lõi của giao thức TCP/IP. Dùng UDP, chương trình trên mạng máy tính có thể gửi những dữ liệu ngắn được gọi là datagram tới máy khác. UDP không cung cấp sự tin cậy và thứ tự truyền nhận mà TCP làm; các gói dữ liệu có thể đến không đúng thứ tự hoặc bị mất mà không có thông báo. Tuy nhiên UDP nhanh và hiệu quả hơn đối với các mục tiêu như kích thước nhỏ và yêu cầu khắt khe về thời gian. Do bản chất không trạng thái của nó nên nó hữu dụng đối với việc trả lời các truy vấn nhỏ với số lượng lớn người yêu cầu.
- **TCP (Transmission Control Protocol – “Giao thức điều khiển truyền vận”)** là một trong các giao thức cốt lõi của bộ giao thức TCP/IP. Sử dụng TCP, các ứng dụng trên các máy chủ được nối mạng có thể tạo các “kết nối” với nhau, mà qua đó

chúng có thể trao đổi dữ liệu hoặc các gói tin. Giao thức này đảm bảo chuyển giao dữ liệu tới nơi nhận một cách đáng tin cậy và đúng thứ tự. TCP còn phân biệt giữa dữ liệu của nhiều ứng dụng (chẳng hạn, dịch vụ Web và dịch vụ thư điện tử) đồng thời chạy trên cùng một máy chủ.

– So sánh một cách đơn giản :

- **Giống nhau** : Đều là các giao thức mạng TCP/IP, đều có chức năng kết nối các máy lại với nhau, và có thể gửi dữ liệu cho nhau....

- **Khác nhau (cơ bản):**

Các header của TCP và UDP khác nhau ở kích thước (20 và 8 byte) nguyên nhân chủ yếu là do TCP phải hỗ trợ nhiều chức năng hữu ích hơn (như khả năng khôi phục lỗi). UDP dùng ít byte hơn cho phần header và yêu cầu xử lý từ host ít hơn.

TCP :

- Dùng cho mạng WAN
- Không cho phép mất gói tin
- Đảm bảo việc truyền dữ liệu
- Tốc độ truyền thấp hơn UDP

UDP:

- Dùng cho mạng LAN
- Cho phép mất dữ liệu
- Không đảm bảo.
- Tốc độ truyền cao, VoIP truyền tốt qua UDP

- TCP hoạt động theo hướng kết nối (connection-oriented), trước khi truyền dữ liệu giữa 2 máy, nó thiết lập một kết nối giữa 2 máy theo phương thức “bắt tay 3 bước (three-way-hand-shake)” bằng cách gửi gói tin ACK từ máy đích sang máy nhận, trong suốt quá trình truyền gói tin, máy gửi yêu cầu máy đích xác nhận đã nhận đủ các gói tin đã gửi, nếu có gói tin bị mất, máy đích sẽ yêu cầu máy gửi gửi lại, thường xuyên kiểm tra gói tin có bị lỗi hay ko, ngoài ra còn cho phép qui định số lượng gói tin được gửi trong một lần gửi (window-sizing), điều này đảm bảo máy nhận nhận được đầy đủ các gói tin mà máy gửi gửi đi → truyền dữ liệu chậm hơn UDP nhưng đáng tin cậy hơn UDP.

- UDP hoạt động theo hướng ko kết nối (connectionless), ko y/c thiết lập kết nối giữa 2 máy gửi và nhận, ko có sự đảm bảo gói tin khi truyền đi cũng như ko thông báo về việc mất gói tin, ko kiểm tra lỗi của gói tin.

→ truyền dữ liệu nhanh hơn UDP do cơ chế hoạt động có phần đơn giản hơn tuy nhiên lại ko đáng tin cậy bằng TCP.

- TCP và UDP là 02 Protocol hoạt động ở lớp thứ 04 (Transport Layer) của mô hình OSI và ở lớp thứ 02 (Transport Layer) mô hình TCP/IP (xin lưu ý: mô hình TCP/IP (***TCP/IP Model***) **khác hoàn toàn** với chồng giao thức TCP/IP (***TCP/IP Suite***))
- TCP là giao thức truyền tin cậy và phải bắt tay ba bước (*three-way-hand-shake*) nên khi Server không nhận được bất kỳ gói tin ACK từ Client gửi trả lời thì Server sẽ gửi lại gói tin đã thất lạc. Server sẽ gửi cho đến khi nhận được ACK của Client mới thôi => Điều này cũng là một nhân tố làm chậm và ngốn băng thông đường truyền.
- Ý chủ bài viết muốn nói lên TCP dành cho các ứng dụng secure. đối với các ứng dụng không cần secure thì dùng UDP là được, lợi hơn về performance. finding DC thì không cần secure, LDAP tất nhiên cần secure.
- Mặc dù tổng lượng lưu thông của UDP trên mạng thường chỉ vài phần trăm, nhưng có nhiều ứng dụng quan trọng dùng UDP, bao gồm DNS, SNMP, DHCP và RIP, dễ thấy hơn là điện thoại IP, xem truyền hình trực tiếp... Thực tế UDP dùng nhiều cho việc truyền tải Games Online. Như các bạn biết đó...việc nâng cấp sửa lỗi games các nhà quản trị Games và Bảo mật thường xuyên cập nhật bản vá lỗi của games, hoặc các thông tin dẫn truyền trong games đều dùng UDP rất nhiều. Nói TCP bảo mật hơn thì cũng đúng, nhưng UDP cũng rất bảo mật điều này phụ thuộc vào mức độ mã hóa của các Package trong liên kết truyền thông. Nghiên cứu thêm trong An Toàn Bảo Mật Thông Tin...
- Để biết máy tính có dùng UDP hay TCP thì bạn có thể thao khảo Sys Tools...những công cụ kiểm tra hệ thống và xem trên Router, check Port Connection....(Tất cả nằm trên anh Google...)
- “Tiền nào của nấy”. Sử dụng TCP có rất nhiều ưu điểm, nhưng ngược lại chi phí đắt. Khi mà ứng dụng không yêu cầu quá cao về chất lượng, có thể chấp nhận việc mất thông tin và trễ về thời gian (như điện thoại internet, media...) thì việc sử dụng UDP thay vì TCP sẽ tiết kiệm được nhiều chi phí.

- Ngoài ra:
 - + UDP có độ trễ nhỏ, do không có giai đoạn thiết lập đường truyền
 - + UDP nó đơn giản: Phía gửi và phía nhận không phải ghi nhớ trạng thái gửi/nhận.
 - + small segment header
 - + Không có cơ chế kiểm soát tắc nghẽn, do đó bên gửi có thể gửi dữ liệu với tốc độ tối đa
 - Các bạn có thể thấy UDP được sd nhiều trong : NTP, DNS, MDNS, BOOTP, DHCP, SysLog, NFS, ISAKMP, Cisco HSRP, Cisco MGCP,...
 - Công giao thức:
 - Port xác định duy nhất một quá trình (process) trên một máy trong mạng. Hay nói cách khác là cách mà phân biệt giữa các ứng dụng.
- VD: Khi máy bạn chạy nhiều ứng dụng mạng như Yahoo, Firefox, game online.... Ví dụ chương Yahoo sử dụng (port 5150 hay 5050) thì khi ai đó gửi tin nhắn đến cho bạn, lúc tin nhắn đến máy bạn nó sẽ dựa vào port để nhận biết đó là chương trình Yahoo (port 5150) chứ ko phải là chương trình khác. Sau đó thông tin sẽ đc xử lý và hiển thị tin nhắn lên.
- Một TCP/IP Socket gồm một địa chỉ IP kết hợp với một port? Xác định duy nhất một tiến trình (process) trên mạng. Hay nói cách khác Luồng thông tin trên mạng dựa vào IP là để xác định máy một máy trên mạng còn port xác định 1 tiến trình trên 1 máy.
- Địa chỉ IP, các địa chỉ IP dành riêng:
- ✓ Mỗi máy tính trên mạng sử dụng IPv4 được xác định bằng 1 địa chỉ IP duy nhất gồm 4 byte được biểu diễn bởi các số thập phân và chia thành 4 phần ngăn cách nhau bởi dấu “.”, ví dụ: 192.168.1.2
 - ✓ Mỗi phần này còn được biết đến là một octet có độ dài là 1 byte phạm vi dưới dạng thập phân là từ 0-255
 - ✓ Khi gói tin được truyền trên mạng, phần header của gói tin sẽ chứa địa chỉ của máy gửi và địa chỉ máy nhận. Máy nhận sẽ sử dụng IP của máy gửi để nhận biết máy nào gửi gói tin cho nó
 - ✓ Hiện nay IPv4 đang cạn kiệt và bắt đầu phổ biến dùng IPv6

- ✓ Con người rất khó nhớ địa chỉ IP dưới dạng các con số, do đó hệ thống tên miền Domain Name System(DNS) ra đời mục đích chuyển các địa chỉ IP sang dạng tên máy(hostname) cho dễ nhớ ví dụ: 208.201.239.101 chuyển thành www.oreilly.com
- ✓ Khi chương trình viết bằng Java sử dụng để truy cập mạng sẽ phải xử lý các địa chỉ được viết bằng số và các hostname tương ứng. Lớp Java.net.InetAddress sẽ cung cấp các phương thức để làm việc này
- ✓ Các dải địa chỉ bắt đầu 10.; 172.16 đến 172.31; 192.168 có thể dùng cho mạng cục bộ không thể truy cập trực tiếp vào internet
- ✓ Ngoài ra liên quan đến nội dung môn học một số địa chỉ đặc biệt thường được sử dụng là:
 - ✓ 127.0.0.1 còn gọi là các địa chỉ loopback cục bộ (local loopback address) hostname của chúng mặc định là “localhost”
 - ✓ 0.0.0.0/8 luôn chỉ đến máy gửi, nhưng có thể dùng như địa chỉ nguồn và hostname của chúng mặc định thường là “localhost”
 - ✓ 255.255.255.255 sử dụng làm địa chỉ quảng bá(broadcast address) gói tin gửi đến địa chỉ này được tất cả các máy trong mạng cục bộ nhận không gửi ra bên ngoài

3. Socket

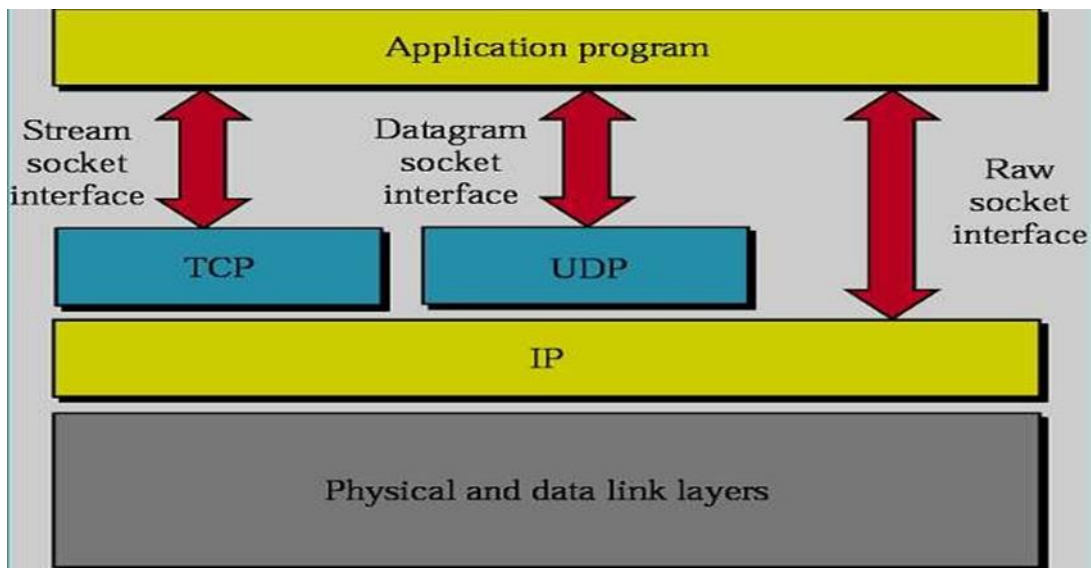
3.1 Khái niệm

Socket là một giao diện lập trình ứng dụng (API) mạng thông qua giao diện này chúng ta có thể lập trình nhiều khiếm việc truyền thông giữa hai máy sử dụng các giao thức mức thấp là TCP, UDP...

Socket là sự trừu tượng hoá ở mức cao, có thể tưởng tượng nó như là thiết bị truyền thông hai chiều gửi – nhận dữ liệu giữa hai máy tính với nhau.

3.2 Các loại socket

- Socket hướng kết nối (TCP Socket)
- Socket không hướng kết nối (UDP Socket)
- Raw Socket



Hình 1: Socket hướng kết nối

2.1.1 Đặc điểm của Socket hướng kết nối

- Có 1 đường kết nối ảo giữa 2 tiến trình
- Một trong 2 tiến trình phải đợi tiến trình kia yêu cầu kết nối.
- Có thể sử dụng để liên lạc theo mô hình Client Server
- Trong mô hình Client/Server thì Server lắng nghe và chấp nhận một yêu cầu kết nối
- Mỗi thông điệp gửi đi đều có xác nhận trở về
- Các gói tin chuyển đi tuần tự

2.1.2 Đặc điểm của Socket không hướng kết nối

- Hai tiến trình liên lạc với nhau không kết nối trực tiếp
- Thông điệp gửi đi phải kèm theo địa chỉ của người nhận
- Thông điệp có thể gửi nhiều lần
- Người gửi không chắc chắn thông điệp tới tay người nhận
- Thông điệp gửi sau có thể đến đích trước thông điệp gửi trước đó.

2.1.3 Số hiệu của cổng socket

- Để có thể thực hiện các cuộc giao tiếp, một trong hai quá trình phải công bố số hiệu cổng của socket mà mình sử dụng.
- Mỗi cổng giao tiếp thể hiện một địa chỉ xác định trong hệ thống. Khi quá trình được gán một số hiệu cổng, nó có thể nhận dữ liệu gửi đến cổng này từ các quá trình khác.
- Quá trình còn lại cũng yêu cầu tạo ra một sock

4. Các lớp socket

4.1 Lớp IPAddress

Một số Field cần chú ý:

- Any: Cung cấp một địa chỉ IP để chỉ ra rằng Server phải lắng nghe trên tất cả các Card mạng
- Broadcast: Cung cấp một địa chỉ IP quảng bá
- Loopback: Trả về một địa chỉ lặp
- AddressFamily: Trả về họ địa chỉ của IP hiện hành

Một số phương thức cần chú ý:

- Phương thức khởi tạo
 - IPAddress(Byte[])
 - IPAddress(Int64)
- IsLoopback: Cho biết địa chỉ có phải địa chỉ lặp không
- Parse: Chuyển IP dạng xấu về IP chuẩn
- ToString: Trả địa chỉ IP về dạng xấu
- TryParse: Kiểm tra IP ở dạng xấu có hợp lệ không?

4.2 Lớp IPEndPoint

Trong mạng, để hai trạm có thể trao đổi thông tin được với nhau thì chúng cần phải biết được địa chỉ (IP) của nhau và số hiệu cổng mà hai bên dùng để trao đổi thông tin. Lớp IPAddress mới chỉ cung cấp cho ta một vé là địa chỉ IP (IPAddress), còn thiếu vé thứ hai là số hiệu cổng (Port number). Như vậy, lớp IPEndPoint chính là lớp chứa đựng cả IPAddress và Port number.

Đối tượng IPEndPoint sẽ được dùng sau này để truyền trực tiếp cho các đối tượng UDP, TCP...

Hàm khởi tạo

- IPEndPoint(Int64, Int32): Tạo một đối tượng mới của lớp IPEndPoint, tham số truyền vào là địa chỉ IP (ở dạng số) và cổng sẽ dùng để giao tiếp.
- IPEndPoint(IPAddress, Int32): Tạo một đối tượng mới của lớp IPEndPoint, Tham số truyền vào là một địa chỉ IPAddress và số hiệu cổng dùng để giao tiếp.

Thuộc tính

- Address: Trả về hoặc thiết lập địa chỉ IP cho endpoint. (Trả về một đối tượng IPAddress)
- Addressfamily: Lấy về loại giao thức mà Endpoint này đang sử dụng.
- Port: Gets or sets số hiệu cổng của endpoint.

Phương thức

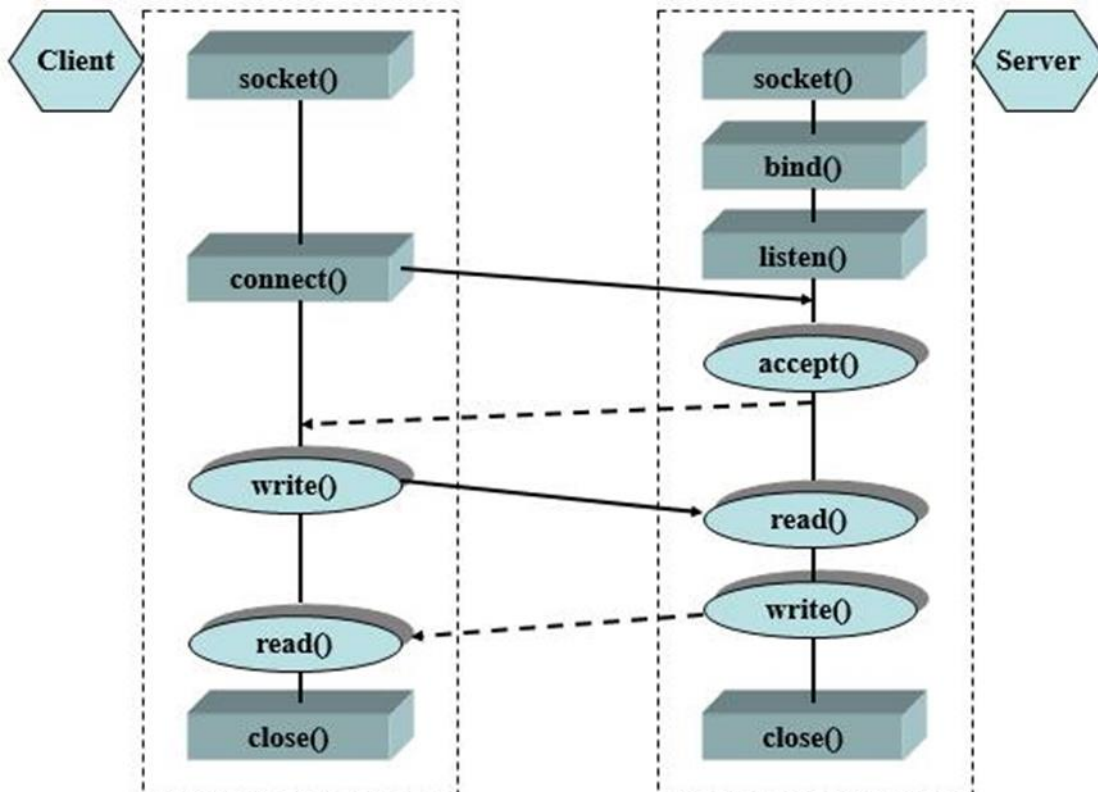
- Create: Tạo một endpoint từ một địa chỉ socket (socket address).
- ToString: trả về địa chỉ ip và số hiệu cổng theo khuôn dạng địa chỉ:

cổng, ví dụ: 192.168.1.1:8080

4.3 Lớp DNS

DNS (Domain Name Service) là một lớp giúp chúng ta trong việc phân giải tên miền (Domain Resolution) đơn giản. (Phân giải tên miền tức là : Đầu vào là Tên của máy trạm, ví dụ ServerCNTT thì đầu ra sẽ cho ta địa chỉ IP tương ứng của máy đó, ví dụ 192.168.3.8)

Ngoài ra lớp DNS còn có rất nhiều phương thức cho ta thêm thông tin về máy cục bộ như tên, địa chỉ,...



Hình 2: Lớp DNS

5. Multithread

5.1 Khái niệm

Một **thread** được định nghĩa như là một đường thực thi (execution path) của một chương trình. Mỗi Thread định nghĩa một dòng điều khiển duy nhất. Nếu application của bạn gồm các hoạt động phức tạp và tốn thời gian, thì nó thường là rất hữu ích khi thiết lập các execution path hoặc Thread, với mỗi Thread thực hiện một công việc cụ thể.

Các Thread là các **tiến trình nhẹ**. Một ví dụ phổ biến của sự sử dụng Thread là sự triển khai lập trình tương tranh (concurrent programming) bởi các hệ điều hành hiện đại. Sử dụng các Thread tiếp kiệm sự hao phí của CPU cycle và tăng hiệu quả của một application.

Tới chương này, chúng ta đã viết các chương trình mà một Thread đơn chạy như là một tiến trình đơn, đó là trình chạy thể hiện của application. Tuy nhiên, theo cách này, application có thể thực hiện một công việc tại một thời điểm. Để làm nó thực thi nhiều hơn một tác vụ tại một thời điểm, nó có thể được phân chia thành các Thread nhỏ hơn.

5.2 Vòng đời của Thread trong C#

Vòng đời của một Thread bắt đầu khi một đối tượng của lớp `System.Threading.Thread` được tạo và kết thúc khi Thread đó được kết thúc hoặc hoàn thành việc thực thi.

Dưới đây là các trạng thái đa dạng trong vòng đời của một Thread trong C#:

- **Unstarted State:** Nó là tình huống khi instance của Thread được tạo, nhưng phương thức `Start` chưa được gọi.
- **Ready State:** Nó là tình huống khi Thread đó sẵn sàng để chạy và đợi CPU cycle.
- **Not Runnable State:** Một Thread là không thể thực thi (not executable), khi:
 - Phương thức `Sleep` đã được gọi.
 - Phương thức `Wait` đã được gọi.
 - Bị ngăn chặn bởi hoạt động I/O.
- **Dead State:** Nó là tình huống khi Thread hoàn thành sự thực thi hoặc bị hủy bỏ.

5.3 Main Thread trong C#

Trong C#, lớp `System.Threading.Thread` được sử dụng để làm việc với các Thread. Nó cho phép tạo và truy cập các Thread riêng biệt trong một Multithreaded

Application. Thread đầu tiên để được thực thi trong một tiến trình được gọi là **Main** Thread trong C#.

Khi một chương trình C# bắt đầu thực thi, Main Thread được tự động tạo ra. Các Thread, được tạo bởi sử dụng lớp **Thread**, được gọi các Thread con của Main Thread. Bạn có thể truy cập một Thread bởi sử dụng thuộc tính **CurrentThread** của lớp Thread.

5.4 Thuộc tính và Phương thức của lớp Thread trong C#

Bảng dưới liệt kê một số **thuộc tính** được sử dụng phổ biến nhất của lớp **Thread** trong C#:

Thuộc tính	Mô tả
CurrentContext	Lấy ngữ cảnh (context) hiện tại mà trong đó Thread đang thực thi
CurrentCulture	Lấy hoặc thiết lập culture gồm language, date, time, currency, ... cho Thread hiện tại
CurrentPrinciple	Lấy hoặc thiết lập nguyên lý hiện tại của Thread
CurrentThread	Lấy Thread đang chạy hiện tại
CurrentUICulture	Lấy hoặc thiết lập culture hiện tại được sử dụng bởi Resource Manager để tìm kiếm cho Resource cụ thể tại runtime
ExecutionContext	Lấy một đối tượng ExecutionContext mà chứa thông tin về các context đa dạng của Thread hiện tại
IsAlive	Lấy một giá trị chỉ trạng thái thực thi của Thread hiện tại
IsBackground	Lấy hoặc thiết lập một giá trị chỉ rằng có hay không một Thread là Background Thread
IsThreadPoolThread	Lấy một giá trị chỉ rằng có hay không một Thread là của Managed Thread

	Pool
ManagedThreadId	Lấy một định danh duy nhất cho Managed Thread hiện tại
Name	Lấy hoặc thiết lập tên của Thread
Priority	Lấy hoặc thiết lập một giá trị chỉ quyền ưu tiên của một Thread
ThreadState	Lấy một giá trị chứa các trạng thái của Thread hiện tại

Bảng này liệt kê các **phương thức** được sử dụng phổ biến nhất của lớp **Thread** trong C#:

STT Phương thức	
1	public void Abort() Tạo một ThreadAbortException trong Thread mà trên đó nó được triệu hồi, để bắt đầu tiến trình kết thúc Thread đó. Gọi phương thức này thường kết thúc Thread
2	public static LocalDataStoreSlot AllocateDataSlot() Cấp phát một Unnamed Data Slot cho tất cả Thread. Để tăng hiệu suất, sử dụng các Field mà được đánh dấu với attribute là ThreadStaticAttribute để thay thế
3	public static LocalDataStoreSlot AllocateNamedDataSlot(string name) Cấp phát một Named Data Slot cho tất cả Thread. Để tăng hiệu suất, sử dụng các Field mà được đánh dấu với attribute là ThreadStaticAttribute để thay thế
4	public static void BeginCriticalRegion() Thông báo cho một host rằng sự thực thi là chuẩn bị đi vào một khu vực code, mà trong đó các ảnh hưởng của việc hủy bỏ một Thread hoặc các Exception không được xử lý có thể gây nguy hại tới các tác vụ khác trong miền ứng dụng

5	public static void BeginThreadAffinity() Thông báo cho một Host rằng Managed code là chuẩn bị thực thi các chỉ lệnh mà phụ thuộc vào tính đồng nhất của Physical operating system thread hiện tại
6	public static void EndCriticalRegion() Thông báo cho một host rằng sự thực thi là chuẩn bị đi vào một khu vực code, mà trong đó các ảnh hưởng của hủy bỏ một Thread hoặc các Exception không được xử lý bị hạn chế tới tác vụ hiện tại
7	public static void EndThreadAffinity() Thông báo cho một Host rằng Managed code đã kết thúc việc thực thi các chỉ lệnh mà phụ thuộc vào tính đồng nhất của Physical Operating System Thread hiện tại
8	public static void FreeNamedDataSlot(string name) Loại bỏ sự liên kết giữa một name và một slot, cho tất cả Thread trong tiến trình. Để tăng hiệu suất, sử dụng các Field mà được đánh dấu với attribute là ThreadStaticAttribute để thay thế
9	public static Object GetData(LocalDataStoreSlot slot) Thu hồi giá trị từ slot đã xác định trên Thread hiện tại, bên trong miền hiện tại của Thread hiện tại. Để tăng hiệu suất, sử dụng các Field mà được đánh dấu với attribute là ThreadStaticAttribute để thay thế
10	public static AppDomain GetDomain() Trả về miền hiện tại trong đó Thread đang chạy
11	public static AppDomain GetDomain() Trả về một định danh miền ứng dụng duy nhất
12	public static LocalDataStoreSlot GetNamedDataSlot(string name) Tìm kiếm một Named Data Slot. Để tăng hiệu suất, sử dụng các Field mà được đánh dấu với attribute là ThreadStaticAttribute để thay thế
13	public void Interrupt() Interrupt (ngắt) một Thread mà trong trạng thái WaitSleepJoin

14	public void Join() Chặn Thread đang gọi tới khi một Thread kết thúc, trong khi tiếp tục thực hiện COM và SendMessage Pumping. Phương thức này có các mẫu được nạp chồng khác nhau
15	public static void MemoryBarrier() Đồng bộ truy cập bộ nhớ như sau: Processor đang thực thi Thread hiện tại không thể sắp xếp lại các chỉ lệnh theo một cách dễ mà quyền truy cập bộ nhớ tới lời gọi đến MemoryBarrier thực thi sau khi các truy cập bộ nhớ mà theo sau lời gọi đó đến MemoryBarrier
16	public static void ResetAbort() Hủy một Abort được yêu cầu cho Thread hiện tại
17	public static void SetData(LocalDataStoreSlot slot, Object data) Thiết lập dữ liệu trong slot đã cho trên Thread đang chạy hiện tại, cho miền hiện tại của Thread đó. Để tăng hiệu suất, sử dụng các Field mà được đánh dấu với attribute là ThreadStaticAttribute để thay thế
18	public void Start() Bắt đầu một Thread
19	public static void Sleep(int millisecondsTimeout) Làm Thread dừng trong một khoảng thời gian
20	public static void SpinWait(int iterations) Làm một Thread đợi một khoảng thời gian đã được xác định trong tham số iterations
21	public static byte VolatileRead(ref byte address) public static double VolatileRead(ref double address) public static int VolatileRead(ref int address) public static Object VolatileRead(ref Object address) Đọc giá trị của một Field. Giá trị này là được viết mới nhất bởi bất kỳ Processor nào trong một máy tính, không quan tâm đến số lượng Processor hoặc trạng thái của Processor Cache. Phương thức

	này có các mẫu được nạp chồng khác nhau. Đó là các form ở trên
22	<p>public static void VolatileWrite(ref byte address, byte value)</p> <p>public static void VolatileWrite(ref double address, double value)</p> <p>public static void VolatileWrite(ref int address, int value)</p> <p>public static void VolatileWrite(ref Object address, Object value)</p> <p>Ghi một giá trị tới một Field ngay lập tức, để mà giá trị này là nhìn thấy cho tất cả Processor trong máy tính. Phương thức này có các mẫu được nạp chồng khác nhau. Đó là các form ở trên</p>
23	<p>public static bool Yield()</p> <p>Làm Thread đang gọi chuyển sự thực thi cho Thread khác mà đã sẵn sàng để chạy trên Processor hiện tại. Hệ điều hành chọn Thread để chuyển tới</p>

Chương II: Xây dựng ứng dụng Server/Client

1. Phần Server

– Code :

```
using System;
using System.Collections.Generic;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Threading;

namespace ServerApp
{
    class Server
    {
        private static TcpListener listener;
        private static bool isRunning;
        private static readonly object locker = new object();
        private static List<ConnectedClient> connectedClients = new
List<ConnectedClient>();

        static void Main(string[] args)
        {
            StartServer();
            Console.InputEncoding=Encoding.Unicode;
            Console.OutputEncoding=Encoding.Unicode;
        }

        private static void StartServer()
        {
            Console.WriteLine("Starting server...");

            listener = new TcpListener(IPAddress.Any, 8888);
            isRunning = true;
```

```

listener.Start();
Console.WriteLine("Server started on port 8888");

while (isRunning)
{
    TcpClient client = listener.AcceptTcpClient();

    Console.WriteLine("Client connected from {0}",
((EndPoint)client.Client.RemoteEndPoint).Address.ToString());

    ConnectedClient newClient = new ConnectedClient(client);
    connectedClients.Add(newClient);

    Thread clientThread = new Thread(new
ParameterizedThreadStart(HandleClient));
    clientThread.Start(newClient);
}
}

private static void HandleClient(object clientObject)
{
    ConnectedClient client = (ConnectedClient)clientObject;
    TcpClient tcpClient = client.TcpClient;

    while (true)
    {
        try
        {
            byte[] buffer = new byte[1024];
            int bytesRead = tcpClient.GetStream().Read(buffer, 0, buffer.Length);

            if (bytesRead == 0)
            {
                // client disconnected
                connectedClients.Remove(client);
                tcpClient.Close();
            }
        }
        catch { }
    }
}

```



```

        Console.WriteLine("Client {0} disconnected.", client.Username);
        break;
    }

    string message = Encoding.ASCII.GetString(buffer, 0, bytesRead);
    Console.WriteLine("[{0}] {1} {2}", DateTime.Now.ToString("HH:mm:ss"),
client.Username, message);

    // send message to all other clients
    lock (locker)
    {
        foreach (ConnectedClient otherClient in connectedClients)
        {
            if (otherClient != client)
            {
                byte[] messageBuffer =
Encoding.ASCII.GetBytes(string.Format("[{0}] {1} {2}",
DateTime.Now.ToString("HH:mm:ss"), client.Username, message));
                otherClient.TcpClient.GetStream().Write(messageBuffer, 0,
messageBuffer.Length);
            }
        }
    }
    catch (Exception ex)
    {
        Console.WriteLine("Error: {0}", ex.Message);
        break;
    }
}

class ConnectedClient
{
    public TcpClient TcpClient { get; set; }
    public string Username { get; set; }
}

```

```
public ConnectedClient(TcpClient tcpClient)
{
    TcpClient = tcpClient;
    Username = string.Format("Client", Guid.NewGuid().ToString().Substring(0,
4));
}
}
}
```

2. Phần Client

– Code :

```
using System;
using System.Net.Sockets;
using System.Text;
using System.Threading;

namespace ClientApp
{
    class Client
    {
        private static TcpClient client;
        private static bool isRunning;
        private static string username;
        static void Main(string[] args)
        {
            Console.InputEncoding = Encoding.Unicode;
            Console.OutputEncoding = Encoding.Unicode;
            Console.Write("Enter username: ");
            username = Console.ReadLine();

            Console.WriteLine("Connecting to server...");

            client = new TcpClient();
            client.Connect("localhost", 8888);
            isRunning = true;
            Console.WriteLine("Connected to server");
            Thread receiveThread = new Thread(new ThreadStart(ReceiveMessages));
            receiveThread.Start();
            Thread sendThread = new Thread(new ThreadStart(SendMessages));
            sendThread.Start();
        }
        private static void ReceiveMessages()
```

```

{
    while (isRunning)
    {
        try
        {
            byte[] buffer = new byte[1024];
            int bytesRead = client.GetStream().Read(buffer, 0, buffer.Length);

            if (bytesRead == 0)
            {
                // server disconnected
                Console.WriteLine("Server disconnected.");
                client.Close();
                isRunning = false;
                break;
            }

            string message = Encoding.ASCII.GetString(buffer, 0, bytesRead);
            Console.WriteLine(message);
        }
        catch (Exception ex)
        {
            Console.WriteLine("Error: {0}", ex.Message);
            break;
        }
    }
}

private static void SendMessages()
{
    while (isRunning)
    {
        try
        {
            string message = Console.ReadLine();

```

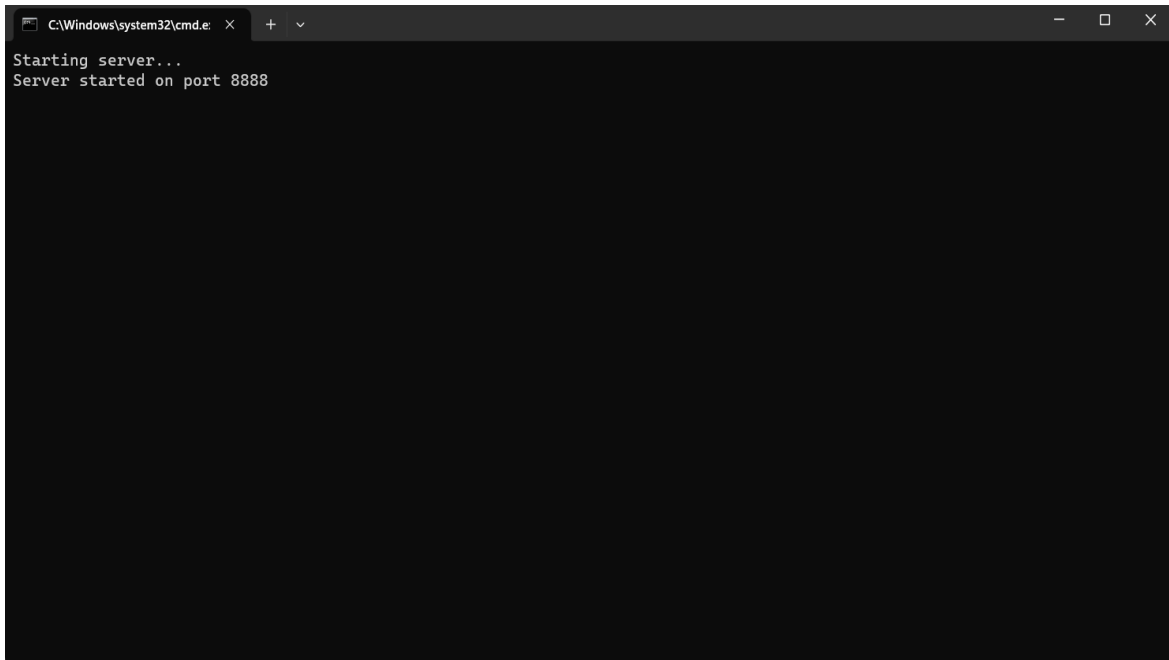
```

        byte[] buffer = Encoding.ASCII.GetBytes(string.Format("{0}: {1}",
username, message));
        client.GetStream().Write(buffer, 0, buffer.Length);
    }
    catch (Exception ex)
    {
        Console.WriteLine("Error: {0}", ex.Message);
        break;
    }
}
}
}
}

```

3. Giao diện console

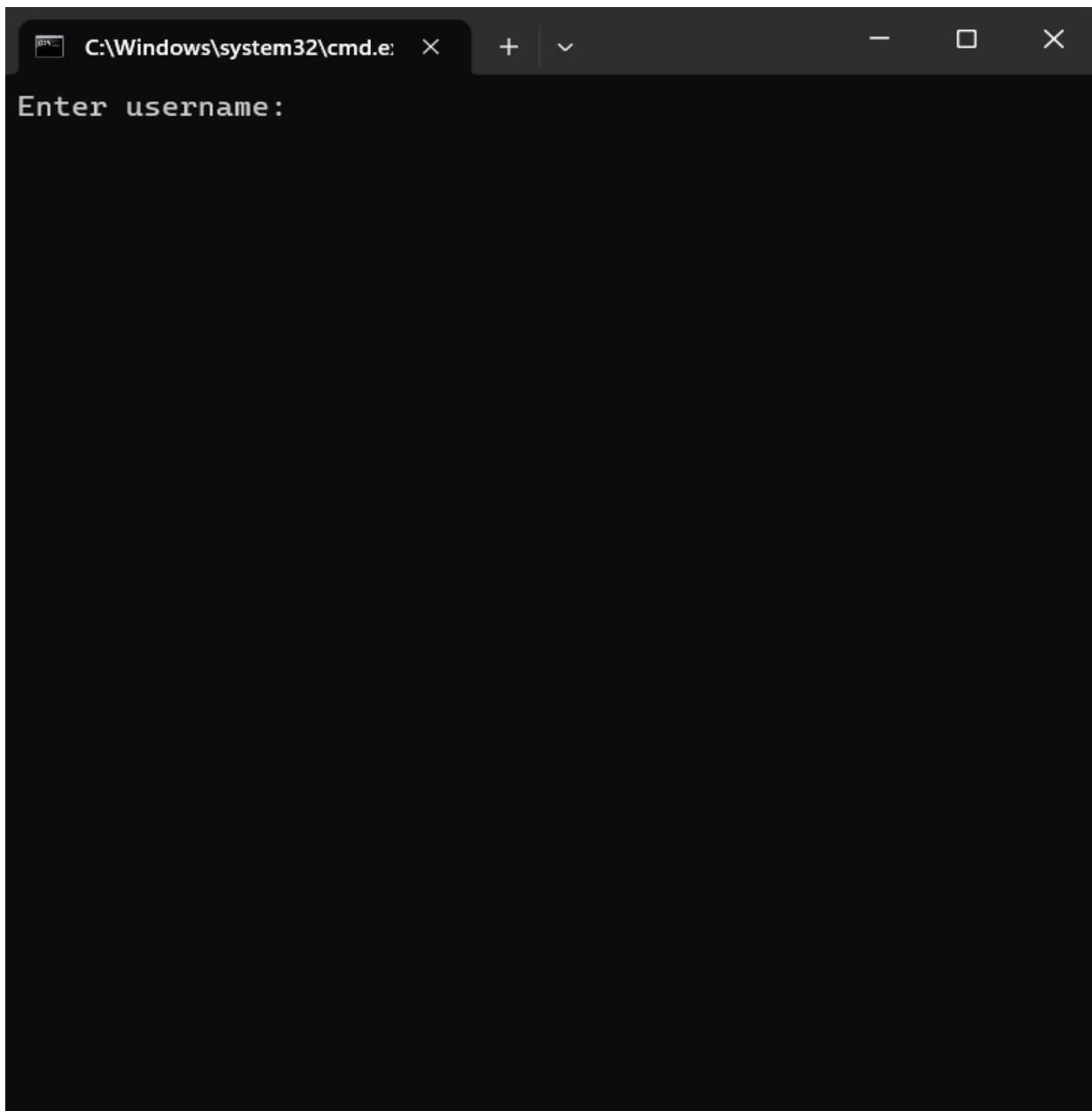
Server :



Hình 3: Giao diện console phía server

Khi chạy chương trình bên phía server thì giao diện console sẽ hiện ra và cùng lúc đó khởi động máy chủ và bắt đầu cho các client kết nối với máy chủ bằng cổng 888.

Client :

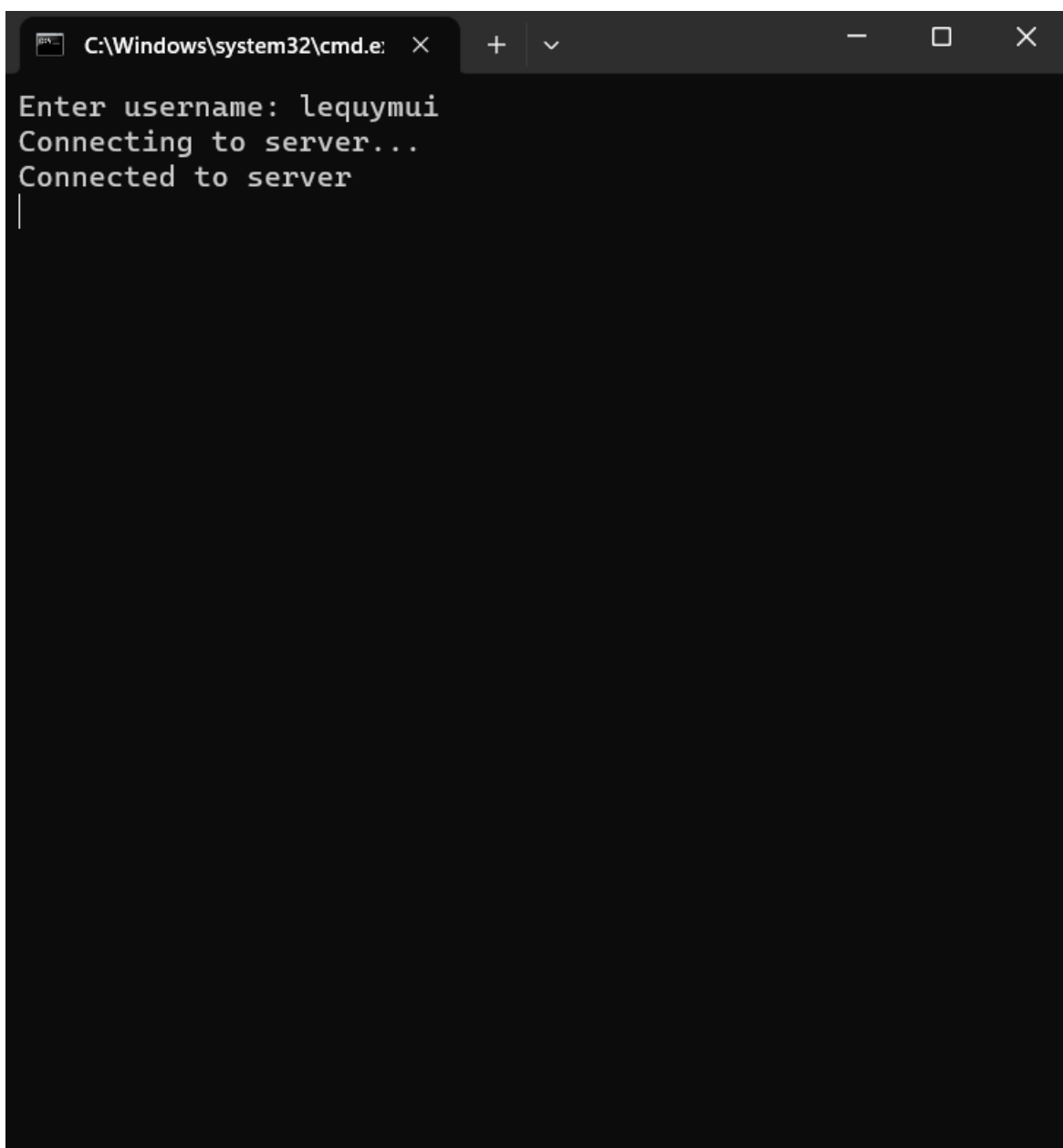


Hình 3: Giao diện console phía client

Khi chạy bên phía client thì sẽ hiển thị lên giao diện console client bằng đầu bằng việc nhập tên người dùng, rồi kết nối với server.

Chương III: Kết quả thực nghiệm chương trình.

Giao diện client :



```
C:\Windows\system32\cmd.e: x + v - □ X
Enter username: lequymui
Connecting to server...
Connected to server
|
```

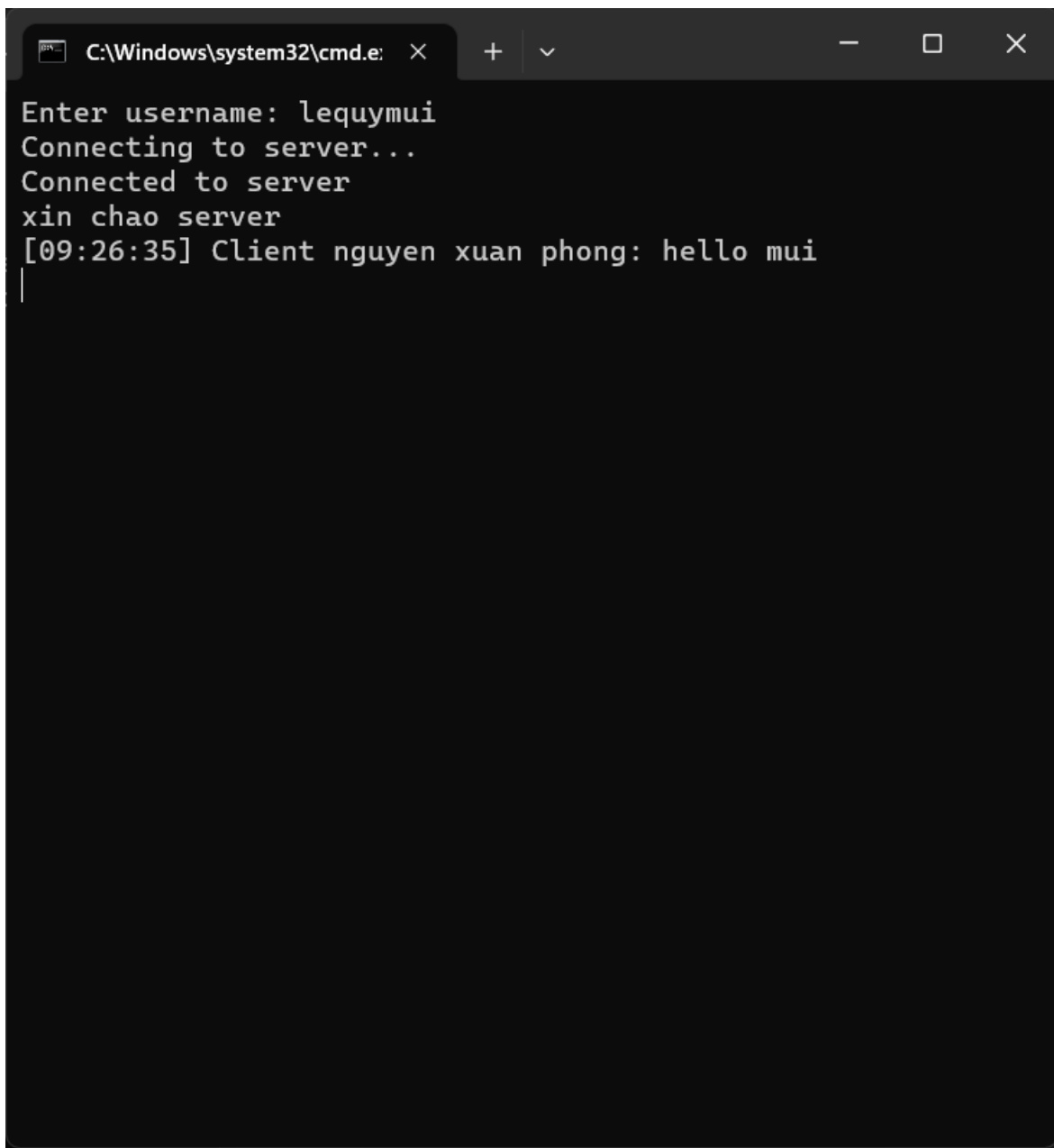
Hình 4: Giao diện console phía client



A screenshot of a Windows command prompt window. The title bar at the top shows the path 'C:\Windows\system32\cmd.e' followed by a close button and window control icons. The command prompt itself has a black background with white text. The text displayed is: 'Enter username: nguyen xuan phong', 'Connecting to server...', and 'Connected to server'.

```
C:\Windows\system32\cmd.e: x + v - □ X
Enter username: nguyen xuan phong
Connecting to server...
Connected to server
```

Hình 5: Giao diện console phía client



```
C:\Windows\system32\cmd.e  ×  +  v  _  □  ×  
Enter username: lequymui  
Connecting to server...  
Connected to server  
xin chao server  
[09:26:35] Client nguyen xuan phong: hello mui  
|
```

Hình 6: Giao diện console phía client

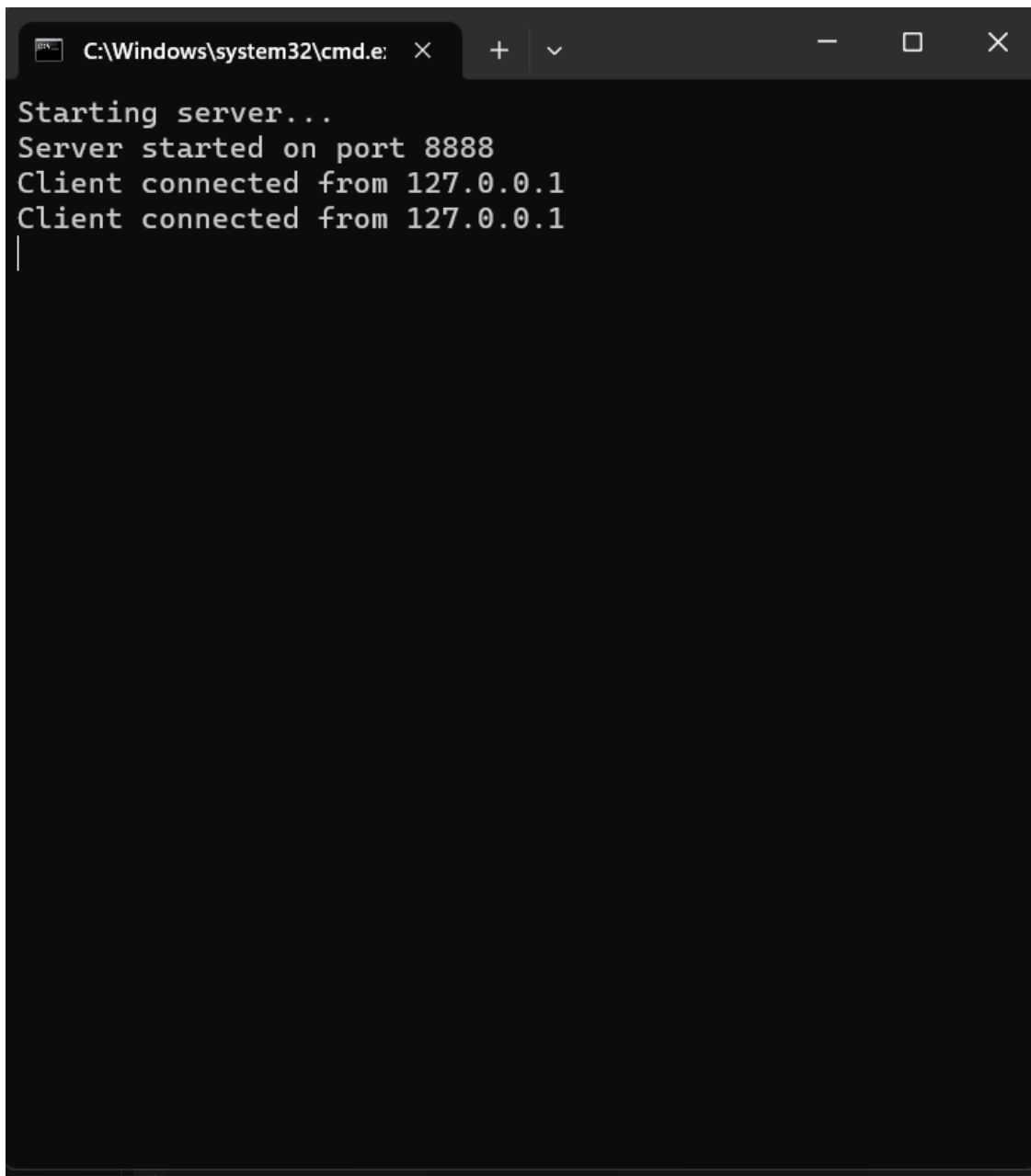


A screenshot of a Windows command prompt window. The title bar shows the path 'C:\Windows\system32' and standard window controls. The command prompt displays the following text: 'Enter username: nguyen xuan phong', 'Connecting to server...', 'Connected to server', '[09:26:22] Client lequymui: xin chao server', and 'hello mui'. A vertical cursor is visible on the line following 'hello mui'.

```
Enter username: nguyen xuan phong
Connecting to server...
Connected to server
[09:26:22] Client lequymui: xin chao server
hello mui
|
```

Hình 7: Giao diện console phía client

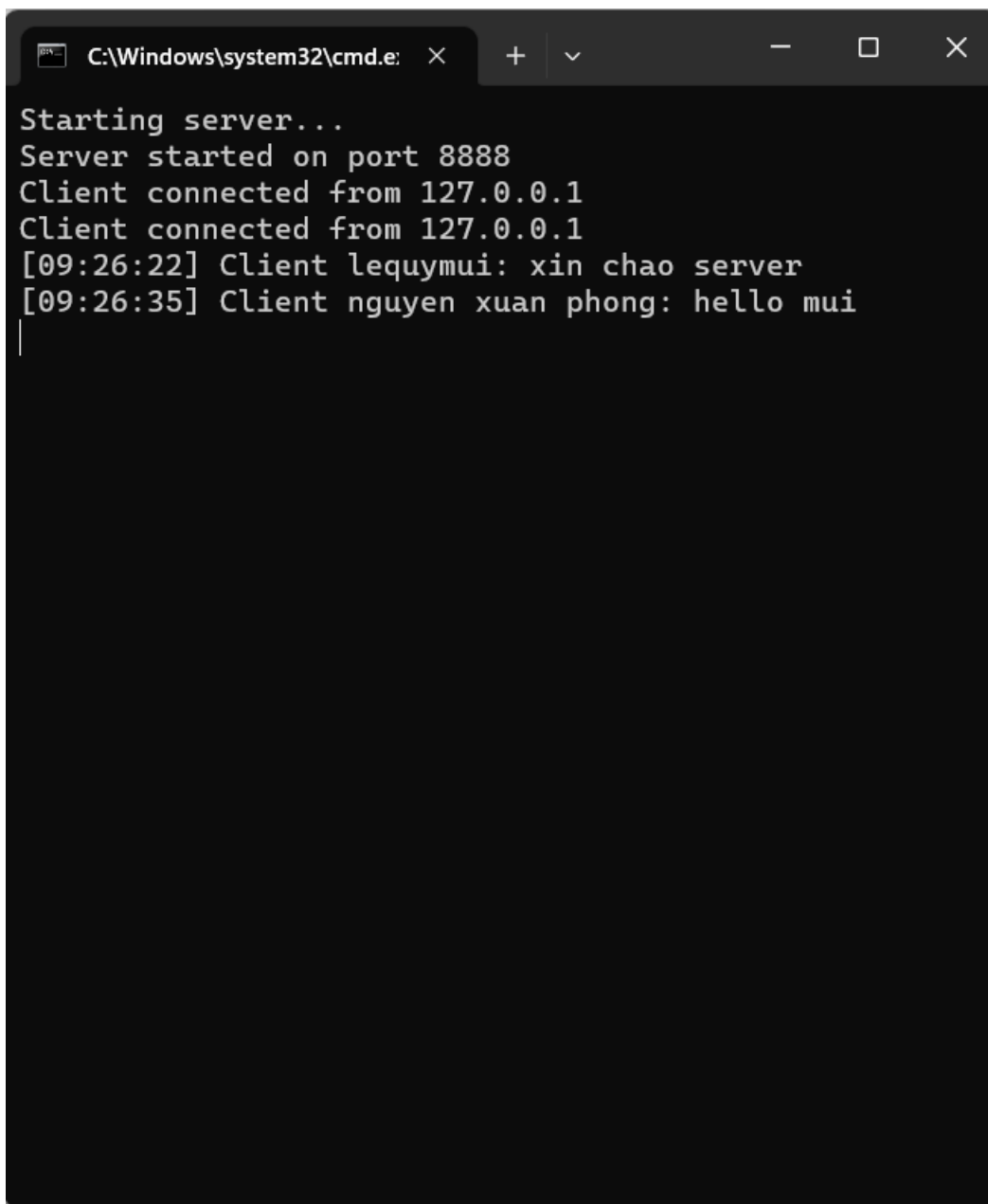
Giao diện server :



```
C:\Windows\system32\cmd.e: x + v - □ ×

Starting server...
Server started on port 8888
Client connected from 127.0.0.1
Client connected from 127.0.0.1
|
```

Hình 8: Giao diện console phía server

A screenshot of a Windows command prompt window. The title bar shows the path 'C:\Windows\system32\cmd.e' and standard window controls. The command prompt displays the following text:

```
Starting server...
Server started on port 8888
Client connected from 127.0.0.1
Client connected from 127.0.0.1
[09:26:22] Client lequymui: xin chao server
[09:26:35] Client nguyen xuan phong: hello mui
|
```

Hình 9: Giao diện console phía server

Kết luận

Ứng dụng chat nhóm TCP socket console là một ứng dụng cho phép nhiều người kết nối và trò chuyện với nhau thông qua giao thức TCP. Dưới đây là phân tích về ưu điểm và nhược điểm của ứng dụng này:

Ưu điểm:

Đơn giản và dễ triển khai: Ứng dụng chat nhóm TCP socket console có thiết kế đơn giản và dễ triển khai. Sử dụng TCP socket và giao thức TCP cho phép kết nối và truyền dữ liệu một cách đáng tin cậy.

Tính linh hoạt: Ứng dụng chat nhóm TCP socket console cho phép nhiều người kết nối và trò chuyện cùng nhau. Người dùng có thể gửi và nhận tin nhắn từ các thành viên khác trong nhóm một cách nhanh chóng và dễ dàng.

Hiệu suất cao: Với sử dụng giao thức TCP, ứng dụng chat nhóm TCP socket console có khả năng truyền dữ liệu một cách nhanh chóng và đáng tin cậy. Điều này đảm bảo rằng các tin nhắn sẽ được gửi và nhận một cách hiệu quả và không bị mất dữ liệu.

Bảo mật: Vì sử dụng giao thức TCP, ứng dụng chat nhóm TCP socket console có thể áp dụng các biện pháp bảo mật như mã hóa dữ liệu để đảm bảo an toàn thông tin trong quá trình truyền.

Nhược điểm:

Giao diện người dùng góc cạnh: Ứng dụng chat nhóm TCP socket console thường không cung cấp giao diện đồ họa và thường được thực hiện trong môi trường dòng lệnh. Điều này có thể làm cho trải nghiệm người dùng trở nên phức tạp và khó sử dụng đối với người dùng không quen thuộc với môi trường dòng lệnh.

Hạn chế trong khả năng tương tác: Do thiết kế dựa trên giao thức TCP, ứng dụng chat nhóm TCP socket console không thể hỗ trợ các tính năng tương tác phong phú như chia sẻ file, hình ảnh hoặc âm thanh trực tiếp.

Phụ thuộc vào kết nối Internet: Ứng dụng chat nhóm TCP socket console yêu cầu một kết nối Internet ổn định để hoạt động. Nếu kết nối bị gián đoạn hoặc không ổn định, việc truyền tải dữ liệu có thể gặp vấn đề.

Khả năng mở rộng hạn chế: Vì ứng dụng chat nhóm TCP socket console được triển khai dựa trên giao thức TCP socket, việc mở rộng ứng dụng để hỗ trợ số lượng người dùng lớn có thể gặp khó khăn và yêu cầu quản lý tài nguyên kỹ lưỡng.

Tóm lại, ứng dụng chat nhóm TCP socket console có những ưu điểm như đơn giản, linh hoạt, hiệu suất cao và bảo mật. Tuy nhiên, nó cũng có nhược điểm như giao diện người dùng góc cạnh, hạn chế trong khả năng tương tác, phụ thuộc vào kết nối Internet và khả năng mở rộng hạn chế.

Phản tài liệu tham khảo

<https://viettuts.vn/csharp/da-luong-trong-csharp>

<https://www.totolink.vn/article/149-mo-hinh-tcp-ip-la-gi-chuc-nang-cua-cac-tang-trong-mo-hinh-tcp-ip.html>

<https://fptcloud.com/socket-la-gi/>