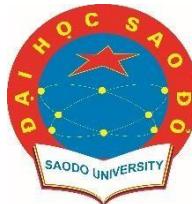


BỘ CÔNG THƯƠNG
TRƯỜNG ĐẠI HỌC SAO ĐỎ



ĐỒ ÁN TỐT NGHIỆP

Ngành: Công nghệ thông tin

TÊN ĐỀ TÀI: LẬP TRÌNH ỨNG DỤNG CLIENT – SERVER
LẤY SỐ THỨ TỰ KHÁM BỆNH

Họ và tên sinh viên: Nguyễn Văn Chiến

Lớp, khoá: DK10-CNTT

Giảng viên hướng dẫn: Ths. Phạm Thị Hướng

HẢI DƯƠNG – NĂM 2023

LỜI CAM ĐOAN

Chúng tôi xin cam đoan các kết quả đưa ra trong đồ án tốt nghiệp này là các kết quả thu được trong quá trình nghiên cứu, thực nghiệm của tôi dưới sự hướng dẫn của Ths Phạm Thị Hường, không sao chép bất kỳ kết quả nghiên cứu nào của các tác giả khác.

Nội dung nghiên cứu có tham khảo và sử dụng một số thông tin, tài liệu từ các nguồn tài liệu đã được liệt kê trong danh mục các tài liệu tham khảo.

Nếu sai tôi xin chịu mọi hình thức kỷ luật theo quy định.

Hải Dương, ngày 25 tháng 11 năm 2023

Sinh viên thực hiện

Nguyễn Văn Chiến

LỜI CẢM ƠN

Để hoàn thành đồ án, trước hết em xin gửi lời cảm ơn chân thành và sâu sắc đến cô Phạm Thị Hường, người đã tận tình hướng dẫn em trong suốt quá trình nghiên cứu vừa qua. Trong thời gian được cô hướng dẫn, em không những tiếp thu thêm nhiều kiến thức bổ ích mà còn học tập được tinh thần, thái độ làm việc nghiêm túc, hiệu quả. Đây là điều rất cần thiết cho em trong quá trình học tập và làm việc sau này.

Em chân thành cảm ơn các thầy cô trong khoa Công nghệ thông tin đã tâm huyết dạy dỗ, truyền đạt những kiến thức quý báu cho em trong suốt những năm học đại học. Những kiến thức đó không chỉ giúp em hoàn thành đồ án tốt nghiệp này mà còn là thứ hành trang quý báu để em có thể tự tin trên quá trình theo đuổi thành công của mình.

Em xin cảm ơn gia đình, bạn bè, người thân và đặc biệt là giáo viên chủ nhiệm và tập thể lớp DK10-CNTT đã hỗ trợ em hết mình trong những năm tháng sinh viên.

Xin trân trọng cảm ơn!

Hải Dương, ngày 25 tháng 11 năm 2023

Sinh viên thực hiện

Nguyễn Văn Chiến

MỤC LỤC

	Trang
LỜI CAM ĐOAN	i
MỤC LỤC	iii
DANH MỤC CÁC HÌNH VẼ	vi
MỞ ĐẦU	1
CHƯƠNG 1. CƠ SỞ LÝ THUYẾT	3
1.1. Một số mô hình mạng	3
1.1.1. Mô hình OSI	3
1.1.2. Mô hình TCP/IP	4
1.2. Cổng giao thức	5
1.2.1. Khái niệm	5
1.2.2. Một số cổng và giao thức thông dụng	6
1.3. Địa chỉ IP, mặt nạ	7
1.4. Một số giao thức mạng	8
1.4.2. Giao thức TCP	8
1.4.3. Giao thức UDP	10
1.5. Mô hình Client – Server	11
1.6. Lập trình Socket	12
1.6.1. Giới thiệu về socket	12
1.6.2. Lập trình với giao thức TCP	21
1.6.3. Lập trình với giao thức UDP	23
1.7. Kết luận chương 1	25
CHƯƠNG 2. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG	26
2.1. Bài toán lấy số tự động	26
2.2. Khảo sát và đánh giá hiện trạng	28
2.2.1. Phân tích về đối tượng xử lý	28
2.2.2. Phân tích về chức năng nghiệp vụ	30
2.2.3. Phân tích yêu cầu phi chức năng	30
2.3. Phân tích phần cứng và chức năng phần mềm	30
2.3.1. Phân tích chung	30

2.3.2. Phân tích sơ bộ phần cứng thiết bị	32
2.3.3. Phân tích chức năng phần mềm	35
2.3.4. Xây dựng logic chức năng	36
2.4. Kết luận chương 2	38
CHƯƠNG 3. XÂY DỰNG ỨNG DỤNG CLIENT – SERVER LẤY SỐ KHÁM BỆNH TỰ ĐỘNG	39
3.1. Lựa chọn công cụ	39
3.2. Xây dựng logic chức năng	40
3.2.1. Kiến trúc mô hình	40
3.2.2. Ứng dụng phía Server	41
3.2.3. Ứng dụng phía Client	45
3.3. Kết quả thực nghiệm và đánh giá	49
3.3.1. Kết quả thực nghiệm	49
3.3.2. Đánh giá dựa trên kết quả thực nghiệm	65
3.4. Kết luận chương 3	66
KẾT LUẬN	67
TÀI LIỆU THAM KHẢO	68

DANH MỤC CÁC BẢNG

	Trang
Bảng 1.1. So sánh giữa TCP/IP và OSI	5
Bảng 1.2. Một số địa chỉ cổng	6
Bảng 1.3. Các tham số của AddressFamily	16
Bảng 1.4. Kiểu Socket	18
Bảng 1.5. Kiểu giao thức	19
Bảng 2.1. Phần cứng thiết bị	32
Bảng 2.2. Chức năng của đối tượng trong hệ thống	34
Bảng 2.3. Chức năng lấy số thứ tự	35
Bảng 2.4. Chức năng xử lý số thứ tự	35
Bảng 3.1. Các chức năng chính phía Server	40
Bảng 3.2. Các chức năng chính phía Client	44

DANH MỤC CÁC HÌNH VẼ

	Trang
Hình 1.1. Mô hình OSI	3
Hình 1.2. Mô hình TCP/IP	4
Hình 1.3. Các dải địa chỉ cổng	6
Hình 1.4. Cổng truy nhập dịch vụ TCP	10
Hình 1.5. Hoạt động của mô hình Client – Server	11
Hình 1.6. Mô hình Client – Server	12
Hình 1.7. Đường đi của dữ liệu từ ứng dụng qua Socket tới mạng	13
Hình 1.8. Quan hệ tiến trình, Socket, dịch vụ TCP trên thiết bị đầu cuối	13
Hình 1.9. Minh họa Socket	14
Hình 1.10. Stream Socket	15
Hình 1.11. Datagram Socket	16
Hình 1.12. Mô hình lập trình Socket hướng kết nối	21
Hình 1.13. Mô hình lập trình Socket phi kết nối	23
Hình 2.1. Quy trình lấy và xử lý số thứ tự tự động	28
Hình 2.2. Quy trình thực hiện của khách hàng (lấy số)	29
Hình 2.3. Quy trình thực hiện của nhân viên (xử lý số)	29
Hình 2.4. Sơ đồ phần cứng mô hình Client – Server và ánh xạ vào ứng dụng	32
Hình 2.5. Mô hình 2 máy tính kết nối mạng Internet	32
Hình 2.6. Mô hình nhiều máy tính kết nối mạng Internet	33
Hình 2.7. Mô hình nhiều máy tính kết nối mạng Internet theo mô hình Client – Server	33
Hình 2.8. Mô hình 2 máy tính kết nối mạng LAN	34
Hình 2.9. Mô hình nhiều máy tính kết nối mạng LAN	34
Hình 2.10. Mô hình nhiều máy tính kết nối mạng LAN theo mô hình Client – Server	35
Hình 2.11. Sơ đồ chức năng lấy số thứ tự khám bệnh	37
Hình 2.12. Sơ đồ chức năng xử lý số thứ tự khám bệnh	37
Hình 3.1. Kiến trúc Server	40
Hình 3.2. Kiến trúc Client	40
Hình 3.3. Cài đặt chương trình Server	49

Hình 3.4. Cài đặt chương trình Server bước 2	50
Hình 3.5. Cài đặt chương trình Server bước 3	50
Hình 3.6. Cài đặt chương trình Server bước 4	50
Hình 3.7. Cài đặt chương trình Server bước 5	50
Hình 3.8. Cài đặt chương trình Server bước 6	51
Hình 3.9. Cài đặt chương trình Client bước 1	51
Hình 3.10. Cài đặt chương trình Client bước 2	51
Hình 3.11. Cài đặt chương trình Client bước 3	52
Hình 3.12. Cài đặt chương trình Client bước 4	52
Hình 3.13. Giao diện khởi động chương trình	52
Hình 3.14. Giao diện kết quả lấy số thứ tự tự động	53
Hình 3.15. Giao diện khởi động của Client 1	53
Hình 3.16. Giao diện khởi động của Client 1	54
Hình 3.17. Giao diện Client 1 xử lý số 01	54
Hình 3.18. Giao diện gọi khách hàng 01 đến cửa, quầy, phòng tương ứng	55
Hình 3.19. Giao diện Client 2 xử lý số 02	55
Hình 3.20. Giao diện Client 2 xử lý số 03	55
Hình 3.21. Giao diện gọi khách hàng 02 đến cửa, quầy, phòng tương ứng	56
Hình 3.22. Giao diện gọi khách hàng 03 đến cửa, quầy, phòng tương ứng	56
Hình 3.23. Giao diện Client 1 xử lý số 04	56
Hình 3.24. Giao diện gọi khách hàng 04 đến cửa, quầy, phòng tương ứng	57
Hình 3.25. Giao diện khởi động Client 1	57
Hình 3.26. Giao diện khởi động Client 2	57
Hình 3.27. Giao diện Client 1 xử lý số 01	58
Hình 3.28. Giao diện gọi khách hàng 01 đến cửa, quầy, phòng tương ứng	58
Hình 3.29. Giao diện Client 2 xử lý số 02	58
Hình 3.30. Giao diện gọi khách hàng 02 đến cửa, quầy, phòng tương ứng	59
Hình 3.31. Giao diện Client 2 xử lý số 03	59
Hình 3.32. Giao diện gọi khách hàng 03 đến cửa, quầy, phòng tương ứng	59
Hình 3.33. Giao diện Client 1 xử lý số 04	60
Hình 3.34. Giao diện gọi khách hàng 04 đến cửa, quầy, phòng tương ứng	60
Hình 3.35. Cấu hình địa chỉ IP cho (a) Server, (b) Client 1, (c) Client 2	60

Hình 3.36. Kiểm tra các máy thông mạng	61
Hình 3.37. Giao diện khởi động Server	61
Hình 3.38. Giao diện Client 1 xử lý số 01	60
Hình 3.39. Giao diện Client 1 xử lý số 02	62
Hình 3.40. Giao diện Client 2 xử lý số 03	62
Hình 3.41. Giao diện Client 2 xử lý số 04	62
Hình 3.42. Giao diện Client 2 xử lý số 05	62
Hình 3.43. Giao diện Client 2 xử lý số 06	63
Hình 3.44. Giao diện gọi khách hàng 01 đến cửa, quầy, phòng tương ứng	63
Hình 3.45. Giao diện gọi khách hàng 02 đến cửa, quầy, phòng tương ứng	63
Hình 3.46. Giao diện gọi khách hàng 03 đến cửa, quầy, phòng tương ứng	64
Hình 3.47. Giao diện gọi khách hàng 04 đến cửa, quầy, phòng tương ứng	64
Hình 3.48. Giao diện gọi khách hàng 05 đến cửa, quầy, phòng tương ứng	64
Hình 3.49. Giao diện gọi khách hàng 06 đến cửa, quầy, phòng tương ứng	64
Hình 3.50. Biểu đồ thể hiện tốc độ phản hồi	65
Hình 3.51. Biểu đồ thể hiện tốc độ phát âm thanh	65

MỞ ĐẦU

1. Tính cấp thiết của đề tài

Hàng ngày, tại các đơn vị hành chính công, cơ sở y tế hay các ngân hàng thương mại,... có nhiều khách hàng đến sử dụng dịch vụ. Khi số lượng khách hàng đông thì việc bảo đảm thứ tự xử lý theo nguyên tắc khách hàng nào đến trước sẽ được phục vụ trước sẽ gặp khó khăn, hiện tượng chen lấn xô đẩy có thể xảy ra do việc điều tiết thủ công. Tại một số đơn vị có trang bị máy in số thứ tự cho khách hàng khi đến giao dịch. Tuy nhiên, hệ thống thường có phần cứng phức tạp hoặc giá thành cao. Việc ứng dụng công nghệ thông tin vào xây dựng hệ thống xử lý yêu cầu của khách hàng lấy số thứ tự tự động thể hiện tính công bằng cho mỗi khách hàng khi tham gia dịch vụ, việc hướng đến một văn hóa phục vụ khách hàng văn minh, lịch sự là yêu cầu tất yếu của xã hội hiện đại. Hệ thống được đề xuất sử dụng phần mềm và tài nguyên mạng LAN để thực hiện tốt các tác vụ yêu cầu của khách hàng, giúp hạn chế tối đa phần cứng, mang lại hoạt động ổn định, dễ lắp đặt, bảo trì và hiệu chỉnh, tạo ra sự công bằng và minh bạch trong việc sắp xếp khách hàng khi sử dụng dịch vụ, đồng thời giúp tiết kiệm thời gian cho công tác quản lý. Từ ý nghĩa đó, nhóm chúng em nghiên cứu, triển khai đề tài “Lập trình ứng dụng Client – Server lấy số thứ tự khám bệnh”.

2. Mục tiêu nghiên cứu

Xây dựng được hệ thống lấy số thứ tự tự động gồm các thành phần sau:

- Phần mềm lấy số thứ tự tự động của khách hàng.
- Phần mềm xử lý các yêu cầu của khách hàng theo thứ tự tự động.
- Mô hình lập trình Client – Server giao tiếp các yêu cầu xử lý.
- Thiết kế mô hình phần cứng giả định hiển thị trên màn hình các số được hệ thống xử lý tự động theo thứ tự yêu cầu của khách hàng.
- Hệ thống bảo đảm các chức năng cơ bản, có giao diện thẩm mỹ, hoạt động ổn định thời gian thực.

3. Đối tượng nghiên cứu

- Mô hình Client – Server và các giao thức giao tiếp.
- Bài toán lấy số khám bệnh tại cơ sở y tế.

4. Phạm vi nghiên cứu

Đồ án nghiên cứu, xây dựng ứng dụng lấy số thứ tự tự động theo mô hình Client – Server trên mạng LAN và Internet cho bệnh viện.

5. Phương pháp nghiên cứu

Phương pháp nghiên cứu tài liệu: Nghiên cứu các tạp chí khoa học, đồ án và tài liệu chuyên ngành.

Phương pháp thực nghiệm: Thiết kế, xây dựng, chạy thử nghiệm ứng dụng và đánh giá kết quả.

6. Ý nghĩa khoa học và thực tiễn của đồ án

Ý nghĩa khoa học: Đồ án đề xuất phương pháp lấy số thứ tự tự động theo mô hình Client – Server trên mạng LAN và Internet.

Ý nghĩa thực tiễn: Xây dựng ứng dụng lấy số thứ tự tự động theo mô hình Client – Server trên mạng LAN và Internet.

7. Kết cấu của đồ án

Đồ án trình bày gồm 3 chương:

Chương 1. Cơ sở lý thuyết: Trình bày khái quát về mô hình OSI, TCP/IP, kiến trúc mô hình Client – Server và các giao thức truyền thông trong lập trình Socket với các giao thức TCP, UDP.

Chương 2. Phân tích và thiết kế hệ thống: Trình bày cách thiết kế ứng dụng .

Chương 3. Xây dựng ứng dụng Client – Server lấy số khám bệnh tự động: Trình bày cách triển khai ứng dụng và đánh giá kết quả thực nghiệm.

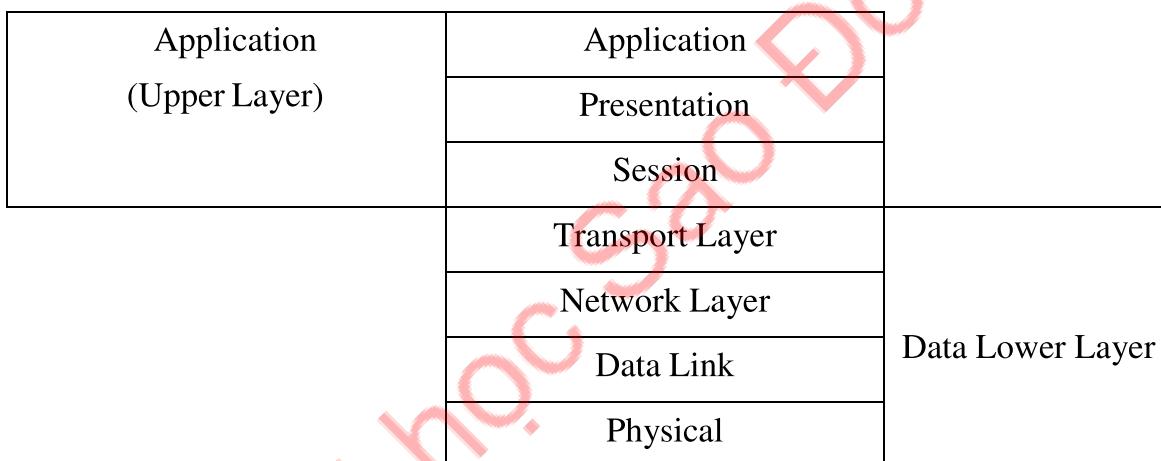
CHƯƠNG 1. CƠ SỞ LÝ THUYẾT

1.1. Một số mô hình mạng

1.1.1. Mô hình OSI

Mô hình OSI (Open Systems Interconnection) hay còn được gọi là “mô hình tham chiếu 7 tầng OSI”. Mục đích chính của chúng là giúp người sử dụng dễ dàng hơn về cơ chế truyền tin giữa các máy tính với nhau. Mô hình OSI bao gồm 7 tầng, mỗi tầng đều có đặc tính là chỉ sử dụng chức năng của tầng dưới nó, đồng thời chúng cũng chỉ cho phép tầng trên sử dụng các chức năng của mình. Mô hình OSI thực chất là chia nhỏ các hoạt động phức tạp của mạng thành các phần công việc đơn giản, dễ hình dung hơn.

Hình 1.1 mô tả bảy tầng OSI, 4 tầng đầu định nghĩa cách thức cho đầu cuối thiết lập kết nối với nhau để trao đổi dữ liệu. 3 tầng trên dùng để phát triển các ứng dụng để đầu cuối kết nối với nhau và người dùng [1].



Hình 1.1. Mô hình OSI

Trong mô hình này, 3 tầng trên cùng của mô hình OSI thường được gọi là các lớp ứng dụng hay còn gọi là các tầng cao. Các tầng này thường liên quan tới giao tiếp với người dùng, định dạng của dữ liệu và phương thức truy nhập các ứng dụng đó.

Chức năng của từng tầng:

- + Tầng 1: Tầng vật lý (Physical Layer) có chức năng chính là điều khiển việc truyền tải các bit trên đường truyền vật lý. Chúng định nghĩa các tín hiệu điện, trạng thái đường truyền, phương pháp mã hóa dữ liệu.

- + Tầng 2: Tầng liên kết dữ liệu (Data-Link Layer) đảm bảo truyền tải các khung dữ liệu (Frame) giữa hai máy tính có đường truyền vật lý nối trực tiếp với nhau. Ngoài ra nó còn cài đặt cơ chế phát hiện và xử lý lỗi dữ liệu nhận.

- + Tầng 3: Tầng mạng (Network Layer) đảm nhiệm việc truyền các gói tin (packet) giữa hai máy tính bất kỳ trong mạng máy tính.

+ Tầng 4: Tầng vận chuyển (Transport Layer) vai trò của chúng là phân nhỏ các gói tin có kích thước lớn khi gửi và tập hợp chúng khi nhận, quá trình phân nhỏ khi gửi và nhận đảm bảo tính toàn vẹn cho dữ liệu (không bị mất mát, không lặp và đúng thứ tự).

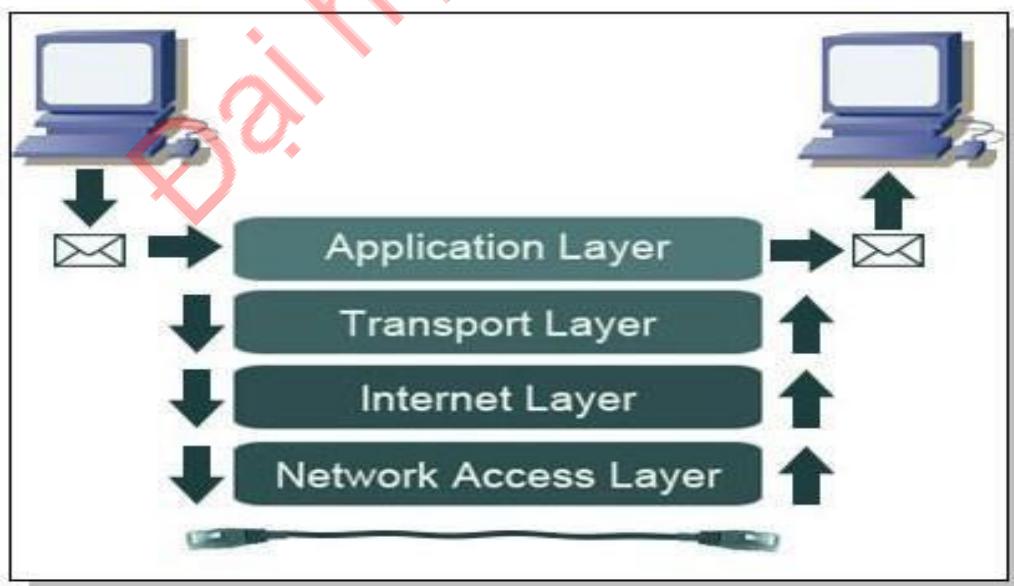
+ Tầng 5: Tầng giao dịch (Session) quản lý phiên làm việc giữa các người sử dụng chính là việc mà chúng làm. Tầng mạng này cung cấp cơ chế nhận biết tên và chức năng bảo mật thông tin qua mạng máy tính.

+ Tầng 6: Tầng trình bày (Presentation Layer) đảm bảo các máy tính có kiểu dạng dữ liệu khác nhau vẫn có thể trao đổi thông tin cho nhau. Thường thì các máy tính sẽ thống nhất với nhau về một kiểu định dạng dữ liệu trung gian để trao đổi thông tin giữa các máy tính. Trong quá trình truyền dữ liệu, tầng trình bày bên máy gửi có nhiệm vụ dịch dữ liệu từ định dạng riêng sang định dạng chung và quá trình ngược lại trên tầng trình bày bên máy nhận.

+ Tầng 7: Tầng ứng dụng (Application Layer) là tầng cung cấp các ứng dụng truy xuất đến các dịch vụ mạng như Web Browser, Mail User Agent... hoặc các Program cung cấp các dịch vụ mạng như Web Server, FTP Server, Mail Server...

1.1.2. Mô hình TCP/IP

Nếu OSI được hình thành mang tính chất dùng cho học tập nghiên cứu nhiều hơn là triển khai thực tế, thì TCP/IP lại khác hoàn toàn. Chính trên chiếc máy tính chúng ta đang sử dụng hàng ngày cũng dùng các giao thức TCP/IPv4 hoặc TCP/IPv6. Bộ giao thức TCP/IP được đặt tên theo hai giao thức chính của nó là TCP (Giao thức điều khiển giao vận) và IP (Giao thức liên mạng).



Hình 1.2. Mô hình TCP/IP

Chức năng các tầng:

+ Tầng 1: Tầng truy cập (Network Access Layer) có thể coi là một tầng riêng biệt hoặc cũng có thể tách nó thành 2 tầng vật lý và tầng liên kết dữ liệu như trong

mô hình OSI. Nó được sử dụng để truyền gói tin từ tầng mạng đến các Host trong mạng. Các thiết bị vật lý như: Switch, cáp mạng, card mạng HBA-Host Bus Adapter là các thành phần truy cập.

+ Tầng 2: Tầng mạng (Internet Layer) trên mô hình TCP/IP có vai trò chính là giải quyết vấn đề dẫn đến các gói tin đi qua các mạng để đến đúng đích.

+ Tầng 3: Tầng vận chuyển (Transport Layer) đảm nhiệm việc phân nhỏ các gói tin có kích thước lớn khi gửi và tập hợp lại khi nhận, tính toàn vẹn cho dữ liệu (không lỗi, không mất, đúng thứ tự) là yếu tố được đảm bảo. Nếu để ý thì bạn sẽ thấy chức năng của tầng vận chuyển ở giao thức TCP/IP cũng giống với tầng vận chuyển của mô hình OSI.

+ Tầng 4: Tầng ứng dụng (Application Layer) là nơi các chương trình mạng như Web Browser, Mail User Agent làm việc để liên lạc giữa các node mạng. Do mô hình TCP/IP không có tầng nào nằm giữa các tầng ứng dụng và tầng vận chuyển, nên tầng Application của TCP/IP bao gồm các giao thức hoạt động như tầng trình diễn và giao dịch trong OSI [2].

Bảng 1.1. So sánh giữa TCP/IP và OSI

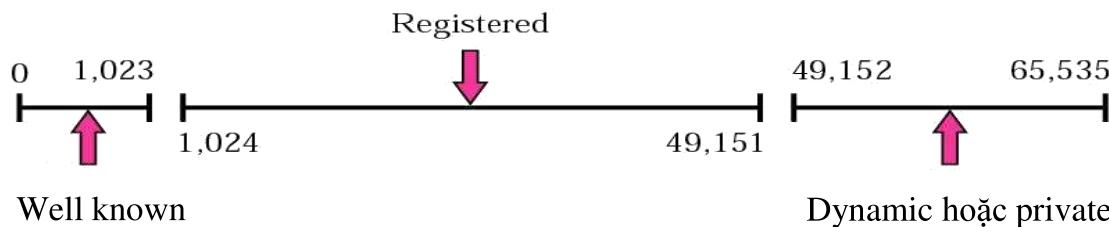
TCP/ IP	OSI
<ul style="list-style-type: none"> - Là giao thức điều khiển truyền vận. - Được phát triển dựa trên các điểm hướng tới mô hình Internet. - Có 4 tầng. - Đáng tin cậy hơn OSI. - Không có ranh giới nghiêm ngặt. - Trong tầng ứng dụng, TCP/IP sử dụng cả tầng phiên và tầng trình diễn. - Tiếp cận theo chiều ngang. - Phát triển giao thức trước sau đó mới phát triển mô hình. - Hỗ trợ truyền thông không kết nối trong tầng mạng. - Phụ thuộc vào giao thức. 	<ul style="list-style-type: none"> - Là kết nối hệ thống mở. - Có 7 tầng. - Có ranh giới chặt chẽ giữa các tầng. - Sử dụng tầng phiên và tầng trình diễn khác nhau. - Tiếp cận theo chiều dọc. - Phát triển mô hình trước sau đó mới phát triển giao thức. - Hỗ trợ kết nối không dây và kết nối định tuyến trong tầng mạng. - Giao thức độc lập.

1.2. Cổng giao thức

1.2.1. Khái niệm

Đa số các hệ điều hành mạng hiện nay đều đa nhiệm nên cho phép nhiều tiến trình truyền thông chạy đồng thời trên cùng một máy tính và đều chung một địa chỉ IP. Chính vì như vậy, 2 tiến trình trên 2 máy tính muốn truyền thông với nhau mà chỉ sử

dụng địa chỉ IP là chưa thể thực hiện được. Để phân biệt các tiến trình chạy trên cùng một máy tính đồng thời, người ta gán cho mỗi tiến trình một nhãn duy nhất để phân biệt các tiến trình với nhau. Trong kỹ thuật mạng máy tính, người ta sử dụng một số nguyên 16 bit để làm nhãn và nó được gọi là số hiệu cổng hoặc địa chỉ cổng (port). Địa chỉ cổng này được sử dụng và được quản lý bởi tầng giao vận và nó có giá trị từ 0 đến 65535, được chia làm 3 dải:



Hình 1.3. Các dải địa chỉ cổng

Dải địa chỉ từ 0 đến 1023: Dải này dùng cho hệ thống, người sử dụng không nên dùng. Các địa chỉ cổng trong dải này thường được gán mặc định cho các giao thức truyền thông phổ biến như: FTP: 21, Telnet: 23, SMTP: 25, POP3: 110, HTTP:80.

Dải địa chỉ từ 1024 đến 49151: Dải địa chỉ cổng này người sử dụng được phép dùng, nhưng phải đăng ký để tránh trùng lặp. Dải địa chỉ từ 49152 đến 65535: Đây là dải địa chỉ động hoặc dùng riêng. Người sử dụng dùng địa chỉ trong dải này không phải đăng ký và cũng không phải chịu trách nhiệm khi xảy ra xung đột địa chỉ.

1.2.2. Một số cổng và giao thức thông dụng

Bảng 1.2. Một số địa chỉ cổng

Cổng	Giao thức	Mô tả
7	Echo	Phản hồi Datagram nhận được trả lại nơi gửi.
9	Discard	Loại bỏ mọi Datagram nhận được.
13	Daytime	Trả về ngày và giờ.
19	Chargen	Trả về một chuỗi ký tự.
20	FTP, Data	Phía Server FTP (Kết nối dữ liệu).
21	FTP, Control	Phía Server FTP (Kết nối điều khiển).
23	Telnet	Mạng đầu cuối.
25	SMTP	Giao thức gửi thư Internet.
53	DNS	Giao thức DNS.
67	BOOTP	Giao thức Bootstrap.
79	Finger	Finger.

Cổng	Giao thức	Mô tả
80	HTTP	Giao thức truyền siêu văn bản.
111	RPC	Giao thức gọi thủ tục từ xa.
110	POP3	Giao thức truy cập Email.
143	IMAP4	Giao thức truy cập Email.

1.3. Địa chỉ IP, mặt nạ

Địa chỉ IP

Hai phiên bản địa chỉ IP thông dụng là IPv4 và IPv6. Hiện thế giới cũng nhu Việt Nam đang chuyển sang sử dụng IPv6.

Mặt nạ (mask)

Mặt nạ là một giá trị hằng (một số nhị phân 32 bit) cho phép phân tách địa chỉ mạng từ địa chỉ IP. Cụ thể khi cho bất kỳ một địa chỉ IP nào trong khối địa chỉ, bằng cách thực hiện phép toán AND mực bit, mặt nạ sẽ giữ nguyên phần netid và xóa toàn bộ các bit phần hostid về giá trị 0, tức là trả về địa chỉ đầu khối địa chỉ đó. Mặt nạ của một mạng con có thể là mặt nạ có chiều dài cố định hoặc biến đổi. Các mặt nạ mặc định của các lớp địa chỉ A, B, C tương ứng là: 255.0.0.0, 255.255.0.0, 255.255.255.0. Trong kỹ thuật chia một mạng thành nhiều mạng con (subnet), hoặc để tạo thành siêu mạng (supernet) đối với lớp C, người ta phải tìm được mặt nạ mạng và định danh cho các mạng đó bằng cách mượn một số bit phần hostid (subnet) hoặc phần netid (supernet). Mặt nạ có vai trò quan trọng trong việc định tuyến cho một gói tin đi đến đúng đích [3].

Một số địa chỉ IP đặc biệt:

Địa chỉ mạng: Netid là định danh của mạng, các bit hostid đều bằng 0.

Địa chỉ Broadcast trực tiếp: Là địa chỉ đích, có phần netid của mạng, các bit phần hostid đều có giá trị 1.

Địa chỉ Broadcast hạn chế: Là địa chỉ đích và có tất cả các bit phần netid và hostid đều có giá trị 1. Gói tin có địa chỉ này sẽ bị chặn bởi các router.

Địa chỉ this host on this network: Có tất cả các bit netid và hostid đều bằng 0. Địa chỉ này là địa chỉ nguồn được máy trạm sử dụng tại thời điểm Bootstrap để truyền thông khi nó biết địa chỉ IP của nó.

Địa chỉ máy trạm cụ thể trong một mạng: Có tất cả các bit netid bằng 0 và phần hostid là địa chỉ host cụ thể trong mạng.

Địa chỉ Loopback: Địa chỉ này có byte đầu tiên là 127, còn các byte còn lại có thể có giá trị bất kỳ: 127.X.Y.Z. Địa chỉ này được dùng để chạy thử các chương trình ứng dụng mạng trên cùng một máy, nhất là khi không có mạng. Địa chỉ Loopback là

địa chỉ đích, khi địa chỉ này được sử dụng, gói tin sẽ không bao giờ truyền ra khỏi máy. Địa chỉ Loopback tiêu biểu là 127.0.0.1 hoặc có thể dùng chuỗi “localhost”.

Địa chỉ riêng: Một số khối địa chỉ trong các lớp được quy định chỉ sử dụng cho mạng riêng (mạng cục bộ) mà không được phép sử dụng trên mạng Internet. Khi các gói tin truyền thông trên mạng Internet, các router và switch trên mạng xương sống Internet được cấu hình loại bỏ gói tin sử dụng các địa chỉ trong các khối địa chỉ riêng này. Người ta phân địa chỉ IP ra làm 5 lớp:

- A: Các địa chỉ IP có octet đầu tiên nằm trong khoảng từ 1-126.
- B: Các địa chỉ IP có octet đầu tiên nằm trong khoảng từ 128-191.
- C: Các địa chỉ IP có octet đầu tiên nằm trong khoảng từ 192-223.
- D: Các địa chỉ IP có octet đầu tiên nằm trong khoảng từ 224-239.
- E: Các địa chỉ IP có octet đầu tiên nằm trong khoảng từ 240-255.

Trong thực tế, chỉ có các địa chỉ lớp A, B, C là được dùng để cài đặt cho các nút mạng, địa chỉ lớp D được dùng trong một vài ứng dụng dạng truyền thông đa phương tiện, như chuyển tải luồng video trong mạng. Riêng lớp E vẫn còn nằm trong phòng thí nghiệm và dự phòng.

Ngoài ra người ta còn sử dụng các địa chỉ không theo lớp mà cho các khối địa chỉ có chiều dài biến đổi, các địa chỉ này có dạng CIDR: a.b.c.d/n.

1.4. Một số giao thức mạng

1.4.1. Tổng quan

Đơn giản, giao thức mạng như một bộ quy tắc và chúng phải tuân theo những quy tắc bắt buộc đó. Nó giống như những tiêu chuẩn và chính sách chính thức và gộp lại, tạo nên những quy tắc đó. Các giao thức mạng này nhằm thực hiện những hành động, chính sách và giải quyết vấn đề từ đầu đến cuối giúp quá trình giao tiếp mạng hoặc dữ liệu diễn ra kịp thời.

Giao thức mạng bao gồm trong đó có sự liên kết giữa máy tính, router, máy chủ và các thiết bị hỗ trợ mạng khác khi chúng muốn giao tiếp với nhau. Để đảm bảo quá trình giao tiếp dữ liệu/mạng diễn ra suôn sẻ thì các giao thức mạng luôn phải được xác nhận và cài đặt bởi người gửi và người nhận.

1.4.2. Giao thức TCP

Giao thức truyền thông là tập các qui tắc, qui ước mà mọi thực thể tham gia truyền thông phải tuân theo để mạng có thể hoạt động tốt. Hai máy tính nối mạng muốn truyền thông với nhau phải cài đặt và sử dụng cùng một giao thức. Dựa vào phương thức hoạt động, người ta có thể chia giao thức truyền thông thành 2 loại: Giao thức hướng kết nối và giao thức hướng không kết nối.

Giao thức hoạt động theo hướng có kết nối là loại giao thức truyền thông sử dụng kết nối ảo để truyền thông. Đặc điểm của loại giao thức này là truyền thông theo

kiểu điểm - điểm, dữ liệu truyền qua mạng là một dòng các byte liên tục truyền từ nơi gửi tới nơi nhận, mỗi byte có một chỉ số xác định.

Quá trình truyền thông được thực hiện thông qua 3 giai đoạn:

- Thiết lập kết nối.
- Truyền dữ liệu kèm theo cơ chế kiểm soát chật chẽ.
- Huỷ bỏ kết nối.

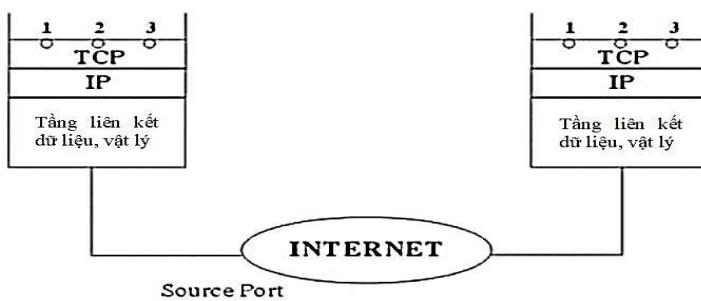
Giao thức tiêu biểu là giao thức TCP (Transmission Control Protocol) là một giao thức lõi chạy ở tầng giao vận, cung cấp các dịch vụ truyền dữ liệu theo dòng, tin cậy và được sử dụng bởi hầu hết các ứng dụng hiện nay. TCP chạy bên trên IP và chạy bên dưới ứng dụng.

Giao thức IP cung cấp cơ chế truyền dữ liệu là các gói tin giữa các máy với nhau, nhưng không có sự đảm bảo về trật tự, mất mát thông tin. Trái lại, TCP cung cấp dịch vụ truyền dữ liệu chính xác, theo dòng, và đúng trật tự ~~giữa~~ các ứng dụng trên các máy khác nhau. Ngoài ra TCP còn kiểm soát tốc độ truyền, chống nghẽn mạng,...

TCP thực hiện chia dữ liệu từ tầng ứng dụng thành các đoạn, mỗi đoạn kích thước thường không vượt quá kích thước của gói tin IP. TCP thêm các thông tin điều khiển vào phần đầu đoạn và chuyển xuống tầng dưới để gửi đi. Dữ liệu của các ứng dụng trên cùng một máy tính được phân biệt thông qua trường Port (16 bit) trong header của TCP. Nếu một ứng dụng muốn nhận thông tin từ mạng, nó sẽ đăng ký một cổng với hệ điều hành, và TCP sẽ chuyển dữ liệu tới ứng dụng đó.

TCP dán nhãn các gói tin theo dạng đánh số. TCP cũng sẽ đảm bảo rằng dữ liệu tới đích trong một thời hạn xác định (một khoảng thời gian vài trăm mili giây được gọi là thời gian chờ) và tuân theo một số quy định kỹ thuật khác. Với mỗi gói tin nhận được, thiết bị gửi sẽ được thông báo thông qua một gói được gọi là xác nhận. Sau khi hết thời gian chờ, không nhận được xác nhận, nguồn gửi sẽ gửi đi một bản sao của gói tin bị mất hoặc bị hoãn. Các gói tin không theo trình tự cũng sẽ không được xác nhận. Nhờ vậy, tất cả các gói dữ liệu sẽ luôn được tập hợp theo thứ tự, không có sơ hở, trong một khoảng thời gian chờ xác định và chấp nhận được.

Địa chỉ TCP: Trong khi IP có một cơ chế hoàn chỉnh để định gán địa chỉ được gọi là địa chỉ IP, TCP lại không có hệ thống địa chỉ phức tạp như vậy. Đúng hơn là TCP không cần đến hệ thống này. TCP chỉ sử dụng các số được cung cấp bởi thiết bị mà nó đang chạy trên đó để xác định nơi nhận và truyền gói tin ở đâu, cho dịch vụ nào. Các số này được gọi là các port. Nói cách khác, một tiến trình ứng dụng trong một máy tính truy nhập vào các dịch vụ của giao thức TCP thông qua một cổng (port) của TCP. Số hiệu cổng của TCP được thể hiện qua 2 byte.



Hình 1.4. Cổng truy nhập dịch vụ TCP

Một cổng TCP kết hợp với địa chỉ IP tạo thành một đầu nối TCP/IP 9(Socket) duy nhất trong liên mạng. Dịch vụ được cung cấp nhờ một liên kết logic giữa một cặp đầu nối TCP/IP. Một đầu nối TCP/IP có thể tham gia nhiều liên kết với các đầu nối TCP/IP ở xa khác nhau. Trước khi truyền dữ liệu giữa hai trạm cần phải thiết lập một liên kết TCP giữa chúng và khi không còn nhu cầu truyền dữ liệu thì liên kết đó sẽ được giải phóng. Ví dụ, các trình duyệt web thường sử dụng cổng 80 cho TCP, cổng 25 dùng cho email,... Số port thường đi kèm với địa chỉ IP để chỉ 1 dịch vụ, ví dụ: 192.168.66.5:80. Ứng dụng của giao thức TCP: TCP/IP được sử dụng để kết nối thông tin trong Internet. Có rất nhiều dịch vụ ở tầng ứng dụng sử dụng TCP, ví dụ dịch vụ web chạy ở cổng 80, FTP ở cổng 21, SMTP cổng 25, POP3 cổng 110, IMAP cổng 143,... Người dùng có thể thực hiện cung cấp thông tin từ xa, gửi mail, truyền file, ảnh hoặc phân phối web trên mạng Internet. Phương thức này cũng cho phép truy cập máy chủ từ xa, thay đổi trạng thái thông tin truyền trong môi trường Internet. Với giao thức TCP/IP, người dùng có thể dễ dàng thay đổi cách biểu thị thông tin thông qua các giao thức cơ bản hoặc giao thức ở mỗi lớp khi thông tin được truyền qua. Nhờ đó, thao tác truyền thông tin sẽ chính xác và hiệu quả hơn.

1.4.3. Giao thức UDP

Kiểu giao thức này khi thực hiện truyền thông không cần kết nối ảo để truyền dữ liệu. Giao thức kiểu này có đặc điểm sau:

- Truyền thông theo kiểu điểm - đa điểm.
- Quá trình truyền thông chỉ có một giai đoạn duy nhất là truyền dữ liệu, không có giai đoạn thiết lập kết nối cũng như huỷ bỏ kết nối.
- Dữ liệu truyền được tổ chức thành các tin gói tin độc lập, trong mỗi gói dữ liệu có chứa địa chỉ nơi nhận.

Giao thức tiêu biểu loại này là giao thức UDP (User Datagram Protocol) cũng là một giao thức lõi trong bộ TCP/IP. UDP cung cấp cơ chế truyền dữ liệu giữa các ứng dụng trên các máy khác nhau. UDP thực hiện chia nhỏ dòng dữ liệu ở tầng ứng dụng thành các đơn vị gọi là datagram và chuyển xuống tầng mạng. UDP không đảm bảo thứ tự của các datagram, cũng như cơ chế phát hiện sự mất mát lõi hoặc trùng datagram. 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. UDP thích hợp với những ứng dụng cần tính thời gian thực cao, có thể sai sót như thoại, video...

UDP cũng sử dụng một số 16 bit trong header gọi là cổng để phân biệt giữa các ứng dụng. Cấu trúc UDP header đơn giản hơn TCP.

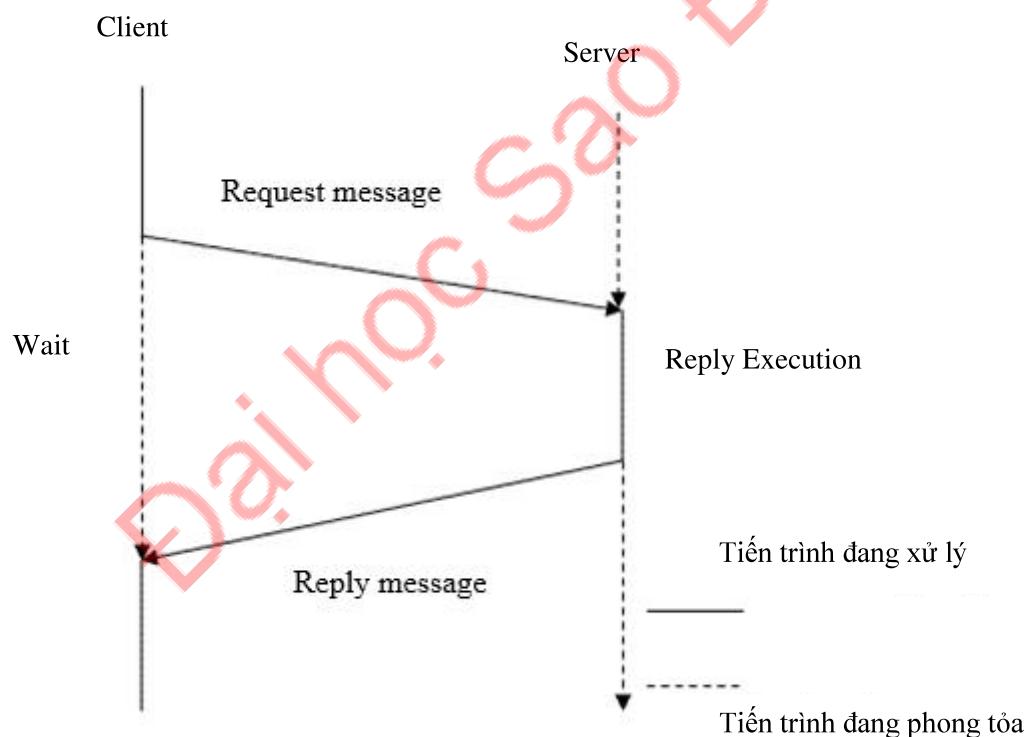
Một vài dịch vụ chạy trên UDP: Phân giải tên miền (DNS:53), RSTP, MMS...

Ngoài ra còn có một số giao thức truyền thông Internet phổ biến:

- Giao thức tầng Internet: IP, ARP, RARP, ICMP, IGMP.
- Giao thức tầng giao vận: TCP, UDP.
- Giao thức dịch vụ: Telnet, FTP, TFTP, SMTP, POP3, IMAP4, DNS, HTTP,...

1.5. Mô hình Client – Server

Chương trình ứng dụng mạng tổ chức theo mô hình Client – Server được sử dụng phổ biến trong thực tế. Chương trình ứng dụng mạng theo mô hình này gồm có 2 phần: Phần Server (phục vụ) và phần Client (máy khách).

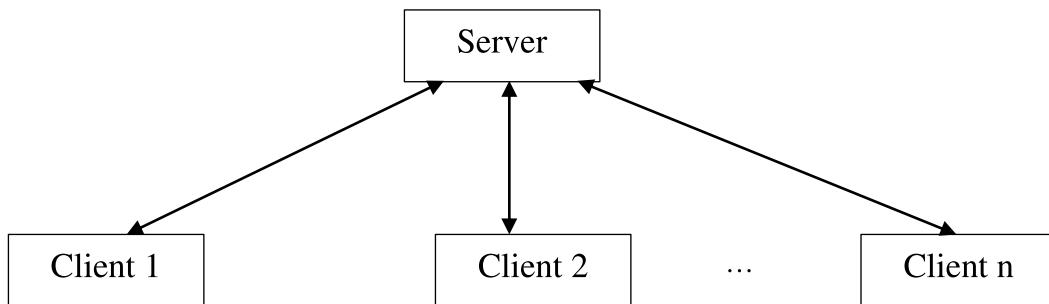


Hình 1.5. Hoạt động mô hình Client – Server

Chế độ blocking (nghẽn): Khi Client hay Server phát ra lệnh send gửi thông điệp, quá trình Client hay Server tương ứng sẽ bị treo cho đến khi phía nhận phát lệnh receive tiếp nhận thông điệp. Tương tự khi gọi lệnh receive mà chưa có send cũng sẽ vào trạng thái treo cho đến khi có lệnh send từ xa gửi thông điệp đến.

Chế độ non blocking: Khi phát ra lệnh truyền thông, quá trình Client hay Server vẫn tiếp tục được thực thi mà không bị treo.

Một chương trình Server có thể phục vụ nhiều chương trình Client đồng thời hoặc tuần tự (kiểu lặp).



Hình 1.6. Mô hình Client – Server

Chương trình Client: Client là một chương trình chạy trên máy cục bộ mà đưa ra yêu cầu dịch vụ đối với Server. Chương trình Client có thời gian chạy hữu hạn. Nó được khởi đầu bởi người sử dụng hoặc một chương trình ứng dụng khác và kết thúc khi dịch vụ đã thực hiện hoàn thành. Sau khi khởi tạo, Client thực hiện mở một kênh truyền thông sử dụng địa chỉ IP của máy trạm từ xa và địa chỉ cổng (nhãn) đã biết rõ của chương trình Server cụ thể chạy trên máy tính từ xa đó. Cách mở đó của Client được gọi là mở tích cực (active open). Sau khi kênh truyền thông được mở Client sẽ gửi yêu cầu tới Server và nhận đáp ứng trả về từ Server.

Chương trình Server: Chương trình này có đặc điểm là có thời gian chạy vô tận và chỉ dừng chạy bởi người sử dụng hoặc tắt máy tính. Chương trình này sau khi khởi tạo, nó sẽ thực hiện mở thụ động (passive open) và được đặt ở trạng thái “nghe” chờ tín hiệu gửi tới từ Client nếu có, nó sẽ nhận yêu cầu gửi tới từ Client, thực hiện xử lý và đáp ứng yêu cầu đó.

Các chức năng trong một chương trình ứng dụng gồm:

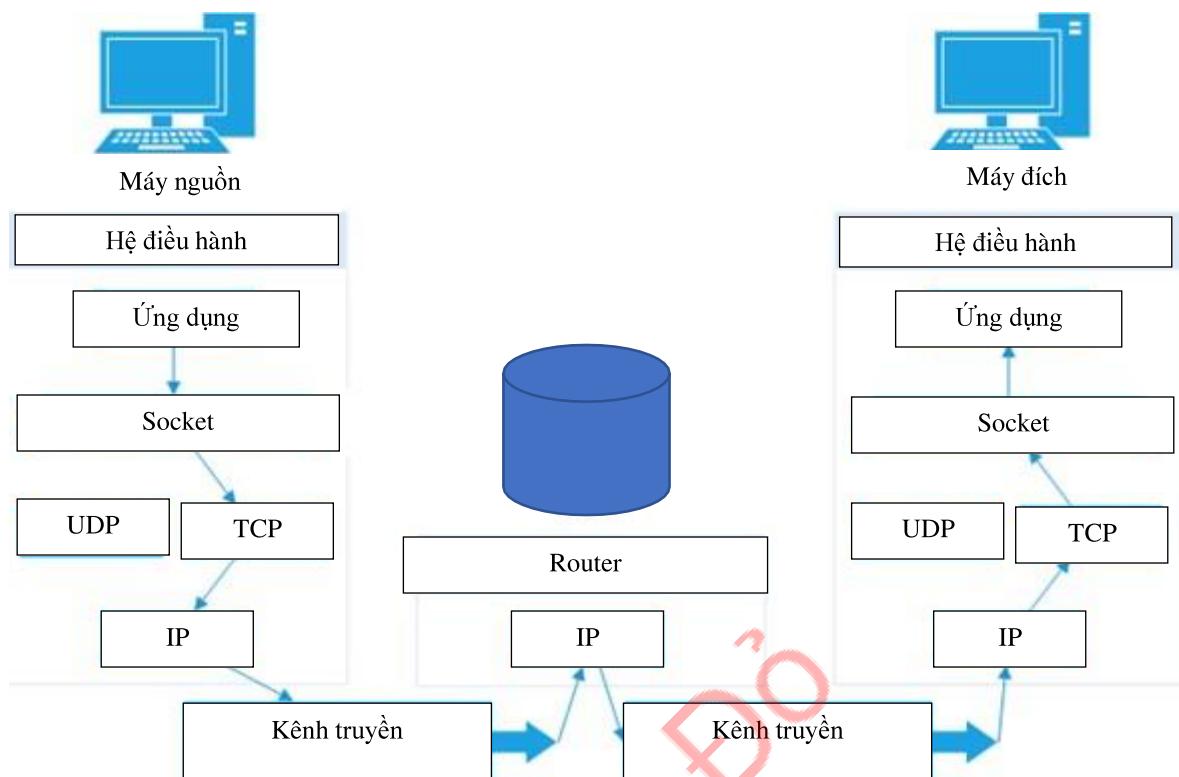
- Giao diện nhập liệu.
- Tính toán, xử lý thông tin theo qui định.
- Truy vấn và lưu trữ thông tin.

1.6. Lập trình Socket

1.6.1. Giới thiệu về socket

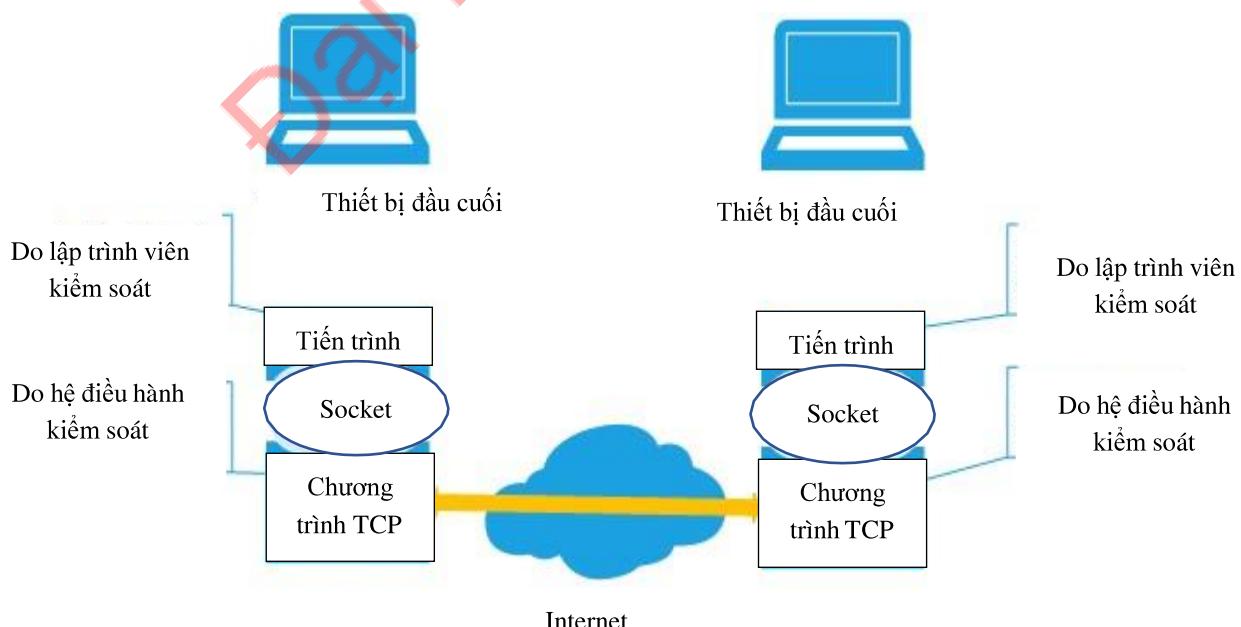
1.6.1.1. Định nghĩa

Khi nhìn nhận từ khía cạnh truyền thông (đường đi của thông tin), Socket có thể được hình dung như là một cánh cửa ngăn cách giữa chương trình ứng dụng (thuộc toàn quyền của người lập trình) và ngăn xếp giao thức mạng (thuộc quyền quản lý của hệ điều hành). Dữ liệu do chương trình tạo ra đi xuyên qua cánh cửa này để đến thế giới mạng, nơi nó sẽ được truyền tới nơi cần đến. Ở chiều ngược lại, dữ liệu từ mạng có thể đi xuyên qua cánh cửa này để tới được chương trình, nơi nó sẽ được xử lý.



Hình 1.7. Đường đi của dữ liệu từ ứng dụng qua Socket tới mạng

Nhìn từ ứng dụng, Socket là giao diện giữa ứng dụng và dịch vụ tầng giao vận trên mỗi máy. Trong mô hình mạng TCP/IP, Socket có thể xem như giao diện giữa tầng ứng dụng và tầng giao vận. Người phát triển ứng dụng có toàn quyền kiểm soát phía tầng ứng dụng của Socket nhưng không thể kiểm soát phía tầng giao vận của Socket. Ở phía tầng giao vận, người phát triển ứng dụng chỉ có thể lựa chọn giao thức của tầng này và điều chỉnh một vài tham số (kích thước tối đa của bộ nhớ đệm, kích thước tối đa của các segment dữ liệu) [1].



Hình 1.8. Quan hệ tiến trình, Socket, dịch vụ TCP trên thiết bị đầu cuối

Trong hai cách nhìn nhận trên, Socket đóng vai trò điểm đầu và điểm cuối của quá trình truyền thông mạng, cũng như phân tách giữa tiến trình (chương trình ứng dụng) và dịch vụ vận chuyển của mạng.

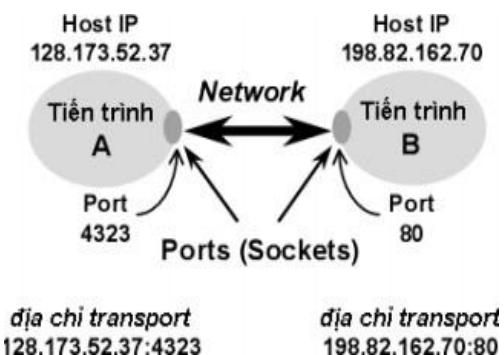
Đối với người lập trình ứng dụng, Socket có thể được hình dung là một giao diện lập trình ứng dụng (API) để gọi tới các chương trình con của hệ điều hành. Tương tự như các chương trình hệ thống khác, TCP hay UDP (và cả IP) cũng cung cấp API để người lập trình có thể sử dụng được các chương trình này. Socket API đầu tiên được xây dựng bởi đại học Berkeley cho hệ điều hành BSD nên thường được gọi là *BSD Socket* hay *Berkeley Socket*. Sau đó Microsoft tham khảo và tạo ra các Socket API dành cho hệ điều hành Windows, gọi là *Windows Socket*, hay thường gọi tắt là *Winsock*.

Socket là giao diện lập trình ứng dụng mạng được dùng để truyền và nhận dữ liệu trên mạng. Giữa hai chương trình chạy trên mạng cần có một liên kết giao tiếp hai chiều để kết nối 2 tiến trình trò chuyện với nhau. Điểm cuối (endpoint) của liên kết này được gọi là Socket. Về bản chất, hoạt động của Socket là kết nối Client với máy chủ bằng TCP/IP hay UDP để thực hiện việc truyền tải và nhận dữ liệu qua môi trường Internet. Do đó, Socket chỉ có thể hoạt động được khi có đủ các thông tin thông số IP, số hiệu Port của 2 ứng dụng muốn trao đổi dữ liệu.

Ngoài ra, 2 ứng dụng muốn truyền tải dữ liệu cho nhau cần phải đáp ứng điều kiện là chúng cùng nằm trên một máy tính hoặc cũng có thể ở 2 máy khác nhau. Đối với trường hợp cùng nằm trên một máy thì số hiệu cổng bắt buộc phải khác nhau.

$$\text{Socket} = \text{Địa chỉ IP} + \text{Số hiệu cổng}.$$

Chức năng chính của Socket là để duy trì kết nối giữa Client với Server để ứng dụng có thể hoạt động thời gian thực. Chẳng hạn như một ứng dụng nhắn tin, Socket sẽ được sử dụng để lắng nghe mỗi khi có tin nhắn mới để cập nhật lên giao diện. Nếu không sử dụng socket, ta vẫn có một cách khác để cập nhật dữ liệu đó là liên tục gửi/nhận các Request/Response trong một khoảng thời gian nhất định, ví dụ cứ sau mỗi 2s ta lại cho ứng dụng kiểm tra API xem dữ liệu có gì thay đổi không, nếu có thì cập nhật vào ứng dụng. Socket giải quyết vấn đề này giúp tiết kiệm tài nguyên cho cả Client và Server.



Hình 1.9. Minh họa Socket

Socket có 4 loại khác nhau là Stream Socket, Datagram Socket, Websocket và Unix socket.

Stream Socket:

Đây là một Socket hướng kết nối hoạt động qua giao thức TCP. Có nghĩa là nó chỉ hoạt động khi Server và Client đã kết nối thành công với nhau.

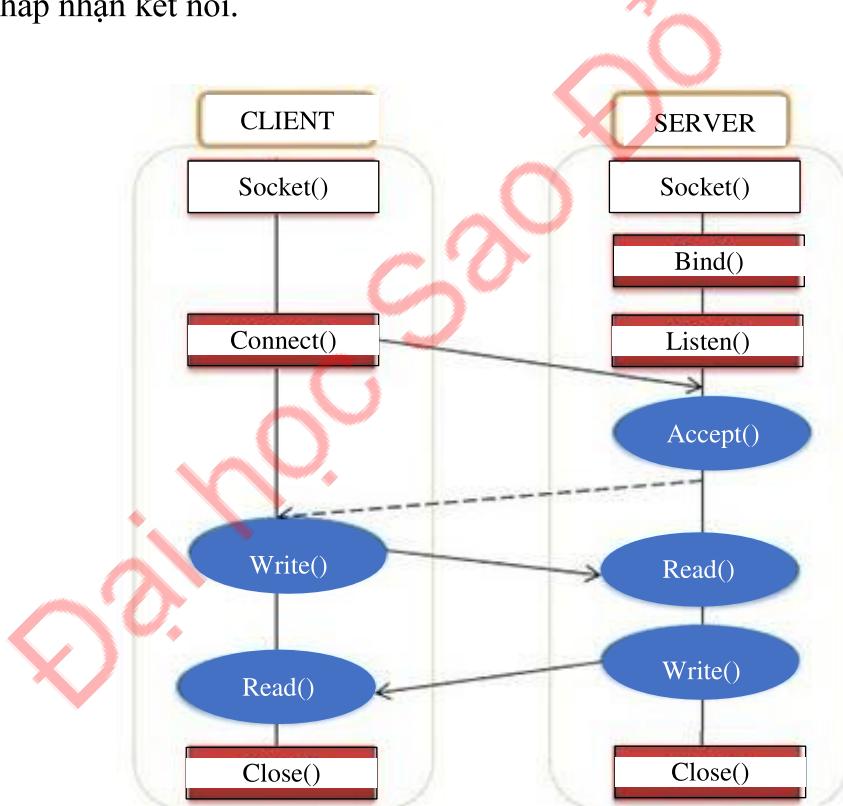
Ưu điểm:

- Đảm bảo truyền dữ liệu đến đúng đối tượng, đúng thứ tự một cách nhanh chóng.

- Khi thông điệp được gửi đi thì hệ thống luôn có xác nhận nhằm thông báo về tình trạng của tiến trình truyền tải cho người dùng.

Nhược điểm:

Vì chỉ có một địa chỉ IP giữa máy chủ và máy nhận nên bắt buộc một máy phải chờ máy kia chấp nhận kết nối.



Hình 1.10. Stream Socket

Datagram Socket:

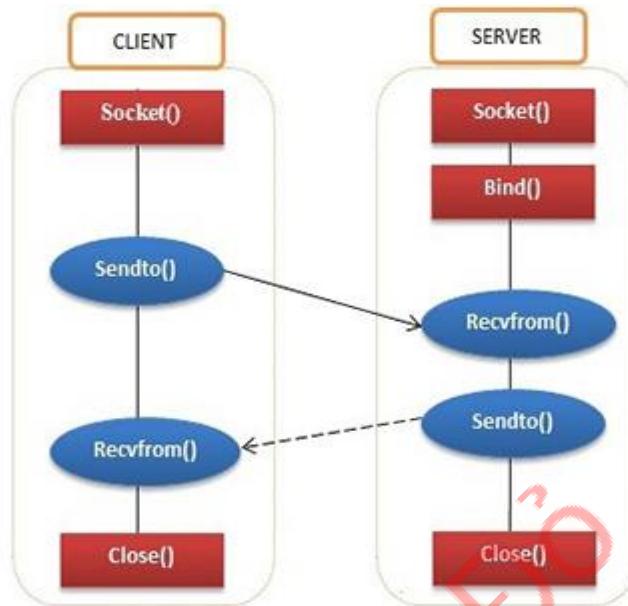
Ngược lại với Stream Socket, Datagram Socket là Socket không hướng kết nối và chúng hoạt động qua giao thức UDP (viết tắt của User Datagram Protocol). Vì thế, Socket này có thể hoạt động dù không có bất kỳ sự kết nối nào của 2 máy với nhau.

Ưu điểm:

- Quá trình kết nối, truyền dữ liệu vô cùng đơn giản.
- Việc truyền tải thông tin dữ liệu diễn ra nhanh chóng.

- Nhược điểm:

Thông tin truyền tải có thể bị lặp hoặc truyền sai thứ tự, do đó thông tin nhận được có độ tin cậy chưa cao.



Hình 1.11. Datagram Socket

Web Socket:

Đây là một Socket có chức năng hỗ trợ việc kết nối trong môi trường Internet giữa Client với Server để chúng được diễn ra nhanh chóng, tăng hiệu quả hơn bằng cách dùng Socket TCP. Web Socket được dùng cho ứng dụng Web và tất cả các ứng dụng cần trao đổi thông tin qua môi trường Internet.

Ưu điểm:

- Tốc độ truyền thông tin giữa Server và Client diễn ra nhanh chóng.
- Người dùng dễ dàng phát hiện lỗi để có thể xử lý chúng.
- Cách sử dụng đơn giản và không cần cài thêm các phần mềm khác để hỗ trợ.
- Người dùng không cần kết hợp thêm các phương pháp kết nối khác trong quá trình sử dụng.

Nhược điểm:

- Hạn chế trong việc tương thích với trình duyệt, bởi Web Socket chưa hỗ trợ được ở tất cả các trình duyệt đang có mặt trên thị trường hiện nay.
- Web Socket chưa hoàn toàn hỗ trợ các dịch vụ yêu cầu về phạm vi.

Unix Socket:

Xét về tốc độ kết nối thì Unix Socket cũng giống như Web Socket, giúp tăng tốc độ kết nối và quá trình truyền dữ liệu được diễn ra nhanh chóng, nhẹ và hiệu quả.

Unix Socket được xem như một điểm giao tiếp, thúc đẩy tiến trình trao đổi thông tin giữa các ứng dụng khác nhau trên cùng một máy tính. Theo đó, toàn bộ hoạt động Unix Socket được thực hiện ở nhân của hệ điều hành. Chính nhờ điều này mà dữ liệu giữa các ứng dụng được truyền tải nhanh chóng hơn rất nhiều.

Ngoài ra, nhờ khả năng tránh được routing hay các bước kiểm tra nên với sự hỗ trợ Unix Socket, việc truyền tải của dữ liệu càng được dễ dàng và đảm bảo hơn.

Ưu điểm:

- Quá trình truyền tải nhanh chóng, nhẹ nhàng và chính xác.
- Tốc độ truy cập vào MySQL tăng đến 30 – 50%.
- Giảm độ trễ thời gian từ 60ms xuống còn 5ms.

Nhược điểm:

- Đối với các ứng dụng không cùng trên một máy chủ thì Unix Socket không thể kết nối được.

- Vấn đề phân quyền của các tập tin đôi khi xảy ra lỗi.

1.6.1.2. Các chế độ giao tiếp

Socket hướng kết nối (TCP Socket)

Dữ liệu truyền theo liên kết TCP là một chuỗi byte liên tục, loại Socket này còn có một tên gọi khác là *Socket hướng dòng* (Stream Socket). Các đặc điểm là:

- 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 từ Client.
- Mỗi thông điệp gửi đều có xác nhận trả về.
- Các gói tin chuyển đi tuần tự.

Socket không hướng kết nối (UDP Socket)

Do UDP không tạo liên kết mà truyền dữ liệu theo các gói độc lập, UDP Socket còn được gọi là *Socket phi liên kết* hay *dgram Socket*. Các đặc điểm là:

- 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.

Sau khi một Socket được tạo ra nó phải được gắn vào một địa chỉ mạng và một cổng trên hệ thống cục bộ hay ở xa. Một khi Socket đã được gắn vào các địa chỉ mạng và cổng, nó có thể được dùng để gửi và nhận dữ liệu trong mạng. Trong .Net Framework, lớp Socket hỗ trợ cho việc lập trình Socket. Phương thức tạo lập như sau:

Socket (AddressFamily, SocketType, ProtocolType)

AddressFamily: Họ địa chỉ được dùng, tham số này có thể có các giá trị sau:

Bảng 1.3. Các tham số của AddressFamily

Giá trị	Ý nghĩa
AppleTalk	Địa chỉ AppleTalk.
Atm	Native ATM services address.
Banyan	Địa chỉ Banyan.
Ccitt	Địa chỉ cho giao thức CCITT, như X25.
Chaos	Địa chỉ cho giao thức MIT CHAOS.
Cluster	Địa chỉ cho các sản phẩm cluster của Microsoft.
DataKit	Địa chỉ cho giao thức Datakit.
DataLink	Địa chỉ của giao thức tầng data - link.
DecNet	Địa chỉ DECnet.
Ecma	Địa chỉ ECMA.
FireFox	Địa chỉ FireFox.
HyperChannel	Địa chỉ NSC Hyperchannel.
Ieee12844	Địa chỉ workgroup IEEE 1284.4.
ImpLink	Địa chỉ ARPANET IMP.
InterNetwork	Địa chỉ IP version 4.
InterNetworkV6	Địa chỉ IP version 6.
Ipx	Địa chỉ IPX hoặc SPX.
Irda	Địa chỉ IrDA.
Iso	Địa chỉ cho giao thức ISO.
Max	Địa chỉ MAX.

Giá trị	Ý nghĩa
NetBios	Địa chỉ NetBios.
NetworkDesigners	Địa chỉ Network Designers.
Osi	Địa chỉ cho giao thức ISO.
Pup	Địa chỉ cho giao thức PUP.
Sna	Địa chỉ IBM SNA.
Unix	Địa chỉ Unix.
Unknown	Chưa biết họ địa chỉ.
Unspecified	Chưa chỉ ra họ địa chỉ.
VoiceView	Địa chỉ VoiceView.

+ **SocketType:** Kiểu Socket, tham số này có thể có các giao thức sau:

Bảng 1.4. Kiểu Socket

Kiểu	Mô tả
Dgram	Được sử dụng trong các giao thức phi kết nối, không tin tưởng. Thông điệp có thể bị mất, bị trùng lặp hoặc có thể đến sai thứ tự. Dgram sử dụng giao thức UDP và họ địa chỉ InterNetwork.
Raw	Được sử dụng trong các giao thức cấp thấp như Internet Control Message Protocol (ICMP) và Internet Group Management Protocol (IGMP). Ứng dụng phải cung cấp IP header khi gửi. Khi nhận sẽ nhận được IP header và các tùy chọn tương ứng.
Seqpacket	Cung cấp hướng kết nối và truyền 2 chiều các dòng byte một cách tin cậy. Seqpacket không trùng lặp dữ liệu và bảo vệ biên dữ liệu. Socket kiểu Seqpacket truyền thông với 1 máy đơn và yêu cầu kết nối trước khi truyền dữ liệu.
	Được sử dụng trong các giao thức phi kết nối, hướng thông điệp, truyền thông điệp tin cậy, và biên của thông điệp được bảo vệ RDM

Kiểu	Mô tả
RDM	(Reliably Delivered Messages) thông điệp đến không bị trùng lặp và đúng thứ tự. Hơn nữa, thiết bị nhận được nếu thông điệp bị mất. Nếu khởi tạo Socket dùng RDM, ta không cần yêu cầu kết nối tới host ở xa trước khi gửi và nhận dữ liệu.
Stream	Được sử dụng trong các giao thức hướng kết nối, không bị trùng lặp dữ liệu, không bảo vệ biên dữ liệu. Socket kiểu Stream chỉ truyền thông với một máy đơn và yêu cầu kết nối trước khi truyền dữ liệu. Stream dùng giao thức Transmission Control Protocol (TCP) và họ địa chỉ InterNetwork.
Unknown	Chưa biết kiểu Socket.

+ **ProtocolType:** Kiểu giao thức, tham số này có thể có các giá trị sau:

Bảng 1.5. Kiểu giao thức

ProtocolType	Mô tả
Ggp	Gateway To Gateway Protocol.
Icmp	Internet Control Message Protocol.
IcmpV6	Internet Control Message Protocol IPv6.
Idp	Internet Datagram Protocol.
Igmp	Internet Group Management Protocol.
IP	Internet Protocol.
IPSecAuthenticationHeader	IPv6 Authentication.
IPSecEncapsulatingSecurityPayload	IPv6 Encapsulating Security Payload header.
IPv4	Internet Protocol version 4.
IPv6	Internet Protocol version 6 (IPv6).
Ipx	Internet Packet Exchange Protocol.
ND	Net Disk Protocol (unofficial).

ProtocolType	Mô tả
Raw	Raw IP Packet Protocol.
Spx	Sequenced Packet Exchange Protocol.
SpxII	Sequenced Packet Exchange version 2 Protocol.
Tcp	Transmission Control Protocol.
Udp	User Datagram Protocol.
Unknown	Giao thức chưa biết.
Unspecified	Giao thức chưa được chỉ ra.

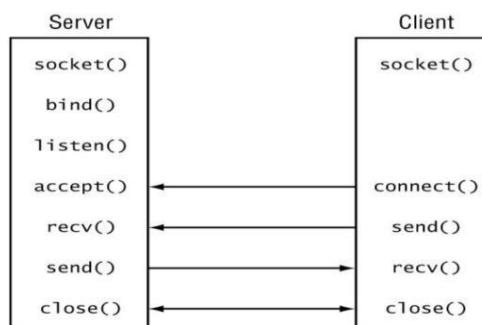
Ví dụ phương thức tạo lập của lớp Socket:

Socket sk = Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

1.6.2. Lập trình với giao thức TCP

1.6.2.1. Mô hình lập trình

Khi viết ứng dụng và có yêu cầu tương tác với một ứng dụng khác, chúng ta thường dựa vào mô hình Client – Server). Ứng dụng chủ (trình chủ hay Server): Ứng dụng có khả năng phục vụ hoặc cung cấp thông tin nào đó. Ứng dụng khách (trình khách hay Client): Sử dụng để gửi yêu cầu đến trình chủ. Trước khi yêu cầu một dịch vụ của trình chủ thực hiện điều gì đó, trình khách phải có khả năng kết nối được với trình chủ. Kết nối giữa trình khách và trình chủ tương tự như việc cắm phích điện vào ổ cắm điện. Trình khách thường được coi như phích cắm điện, còn trình chủ được xem như ổ cắm điện, một ổ cắm có thể cắm vào đó nhiều phích điện khác nhau cũng như một máy chủ có thể kết nối và phục vụ cho rất nhiều máy khách. Nếu kết nối Socket thành công thì trình khách và trình chủ có thể thực hiện các yêu cầu về trao đổi dữ liệu với nhau. Để lập trình Socket hướng kết nối ta phải thực hiện một loạt các thao tác giữa Client và Server như trong mô hình sau:



Hình 1.12. Mô hình lập trình Socket hướng kết nối

1.6.2.2. Lập trình phía Server

Trước hết ứng dụng chủ mở một Socket. Đây chỉ là quá trình hệ điều hành phân bổ tài nguyên để chuẩn bị kết nối còn gọi là tạo “ô cắm” Socket cho trình chủ Server. Để ứng dụng khách biết đến ô cắm Socket của trình chủ, phải đặt cho Socket trình chủ một tên. Nếu trên máy cục bộ và dựa vào hệ thống tập tin của Linux, ta có thể đặt tên cho Socket như là một tên tập tin (với đầy đủ đường dẫn). Đối với giao tiếp mạng thông qua giao thức TCP/IP tên của Socket được thay thế bằng khái niệm cổng. Cổng là một số nguyên 2 bytes thay thế cho tên tập tin. Nếu trình khách và trình chủ nằm trên hai máy khác nhau, giao thức TCP/IP còn yêu cầu xác định thêm địa chỉ IP để kết nối đến máy chủ ở xa.

Sau khi đã chỉ định tên hoặc số hiệu cổng cho Socket, cần gọi bind() để ràng buộc hay đặt tên chính thức cho Socket của trình chủ. Tiếp đến, trình chủ sẽ gọi hàm listen() để tạo hàm lắng nghe các kết nối từ trình khách đưa đến. Nếu có yêu cầu kết nối từ trình khách, trình chủ gọi hàm accept() để tiếp nhận yêu cầu của trình khách. Hàm accept() sẽ tạo một Socket vô danh khác, cắm kết nối của trình khách vào Socket vô danh này và thực hiện quá trình chuyển dữ liệu trao đổi giữa khách chủ. Socket được đặt tên trước đó vẫn tiếp tục hoạt động để chờ nhận yêu cầu từ trình khách khác. Mọi giao tiếp đọc ghi thông qua Socket cũng đơn giản như việc dùng lệnh read/write để đọc ghi trên tập tin. Nếu tập tin dựa vào số mô tả để đọc ghi trên một tập tin xác định thì Socket cũng dựa vào số mô tả để xác định Socket cần đọc ghi cho hàm read/write.

Đầu tiên Server sẽ tạo một Socket, Socket này sẽ được gắn vào một địa chỉ IP và một cổng cục bộ, hàm để thực hiện việc này là hàm Bind(). Hàm này cần một danh đối số là một IPEndPoint cục bộ:

```
IPEndPoint ipep = new IPEndPoint(IPAddress.Any, 5000);
```

```
Socket server = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
```

```
server.Bind(ipep);
```

Bởi vì Server thường chấp nhận kết nối trên chính địa chỉ IP và cổng riêng của nó nên ta dùng IPAddress.Any để chấp nhận kết nối trên bất kỳ card mạng nào. Địa chỉ IP ta dùng ở đây là địa chỉ IP version 4 và kiểu giao thức là TCP nên AddressFamily là InterNetwork và SocketType là Stream.

Sau khi Socket đã được gắn kết vào một địa chỉ và một port, Server phải sẵn sàng chấp nhận kết nối từ Client. Việc này được thực hiện nhờ vào hàm Listen(). Hàm Listen() có một đối số, đó chính là số Client tối đa mà nó lắng nghe.

```
server.Listen(10);
```

Tiếp theo Server dùng hàm Accept() để chấp nhận kết nối từ Client:

```
Socket client = server.Accept();
```

Hàm Accept() này sẽ dùng Server lại và chờ cho đến khi nào có Client kết nối đến nó sẽ trả về một Socket khác, Socket này được dùng để trao đổi dữ liệu với Client. Khi đã chấp nhận kết nối với Client thì Server có thể gửi và nhận dữ liệu với Client thông qua phương thức Send() và Receive().

1.6.2.3. Lập trình phía Client

Lập trình Socket hướng kết nối phía Client đơn giản hơn phía Server. Client cũng phải gắn kết một địa chỉ của một Socket đã được tạo ra nhưng sử dụng phương thức Connect() chứ không sử dụng phương thức Bind() giống như phía Server. Phương thức Connect() yêu cầu một IPEndPoint của Server mà Client cần kết nối đến.

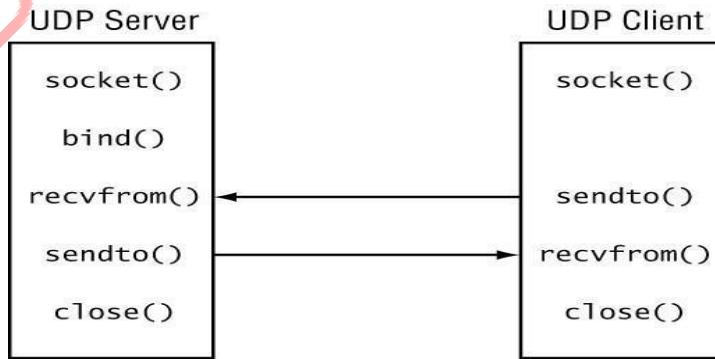
```
IPEndPoint ipep = new IPEndPoint(IPAddress.Parse("127.0.0.1"), 5000);
Socket server = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
ProtocolType.Tcp);
```

1.6.3. Lập trình với giao thức UDP

Các Socket phi kết nối cho phép gửi các thông điệp mà không cần phải thiết lập kết nối trước. Một phương thức đọc sẽ đọc toàn bộ thông điệp được gửi bởi một phương thức gửi, điều này làm tránh được các rắc rối, phức tạp với biên dữ liệu. Tuy nhiên, giao thức phi kết nối UDP không đảm bảo dữ liệu được truyền tới đích. Nhiều yếu tố như mạng bận, mạng bị đứt nửa chừng có thể ngăn cản các gói tin được truyền tới đích.

Nếu một thiết bị chờ dữ liệu từ một thiết bị ở xa, nó phải được gán một địa chỉ và cổng cục bộ, dùng hàm Bind() để gán. Một khi đã thực hiện xong, thiết bị có thể dùng Socket để gửi dữ liệu ra ngoài hay nhận dữ liệu từ Socket.

Bởi vì thiết bị Client không tạo ra kết nối đến một địa chỉ Server cụ thể do đó phương thức Connect() không cần dùng trong chương trình UDP Client. Mô hình bên dưới mô tả các bước lập trình Socket phi kết nối:



Hình 1.13. Mô hình lập trình Socket phi kết nối

Khi kết nối không được thành lập thì phương thức Send() và Receive() không được dùng bởi vì trong hai phương thức trên đều không chỉ ra địa chỉ đích của dữ liệu. Thay vào đó, Socket phi kết nối cung cấp hai phương thức để thực hiện việc này là SendTo() và ReceiveFrom().

Lập trình phía Server:

UDP là một giao thức phi kết nối do đó các lập trình viên chỉ phải làm hai việc để tạo ra một ứng dụng Server gửi và nhận dữ liệu đó là tạo ra Socket và kết nối Socket đến một IPEndPoint cụ bô.

```
IPEndPoint ipep = new IPPEndPoint(IPAddress.Any, 5000);
```

```
Socket newsock = new Socket(AddressFamily.InterNetwork,  
SocketType.Dgram, ProtocolType.Udp);  
newsock.Bind(ipep);
```

Để thực hiện truyền thông phi kết nối, chúng ta phải chỉ ra SocketType là Dgram và ProtocolType là UDP. Sau khi thực hiện xong hai bước trên, Socket có thể được dùng hoặc để chấp nhận các gói tin UDP đến trên IPEndPoint hoặc gửi các gói tin UDP đến các thiết bị nhận khác trên mạng.

Phương thức SendTo() dùng để gửi dữ liệu, phương thức này chỉ ra dữ liệu để gửi và IPEndPoint của thiết bị nhận. Có nhiều quá tải hàm của phương thức SendTo() có thể được dùng tùy vào yêu cầu cụ thể.

```
SendTo(byte[] data, EndPoint Remote)
```

Phương thức trên gửi một mảng dữ liệu đến mộtEndPoint được chỉ ra bởi Remote. Một quá tải hàm khác phức tạp hơn của phương thức SendTo():

```
SendTo(byte[] data, SocketFlags Flags, EndPoint Remote)
```

Phương thức này cho phép thêm cờ SocketFlag, nó chỉ ra các tùy chọn UDP được sử dụng. Để chỉ ra số byte được gửi từ mảng byte ta sử dụng quá tải hàm sau của phương thức SendTo():

```
SendTo(byte[] data, int Offset, int Size, SocketFlags Flags, EndPoint Remote)
```

Phương thức ReceiveFrom() có dùng định dạng với phương thức SendTo(), chỉ có một điểm khác biệt sau ở cách EndPoint được khai báo. Phương thức ReceiveFrom() đơn giản được định nghĩa như sau:

```
ReceiveFrom(byte[] data, ref EndPoint Remote)
```

Cũng như thông thường, tham số thứ nhất là một mảng byte được định nghĩa để nhận dữ liệu, tham số thứ hai là tham chiếu của đối tượng EndPoint. Tham chiếu này tham chiếu đến vị trí bộ nhớ nơi biến được lưu trữ. Phương thức ReceiveFrom() sẽ đặt thông tinEndPoint từ thiết bị ở xa vào vùng bộ nhớ của đối tượng EndPoint tham chiếu đến. Bằng việc sử dụng đối số thứ hai là tham chiếu ta sẽ lấy được địa chỉ IP và port của máy ở xa.

Lập trình phía Client:

Bởi vì Client không cần chờ trên một cổng UDP định sẵn nên nó cũng chẳng cần dùng phương thức Bind(), thay vì vậy nó sẽ lấy một cổng ngẫu nhiên trên hệ thống khi dữ liệu được gửi và nó giữ cổng này để nhận dữ liệu trả về.

Sử dụng phương thức Connect() trong chương trình UDP Client:

Các phương thức UDP được thiết kế để cho phép các lập trình viên gửi các gói tin đến bất kỳ máy nào trên mạng bất cứ lúc nào. Bởi vì giao thức UDP không yêu cầu kết nối trước khi gửi dữ liệu nên phải chỉ ra địa chỉ của máy nhận trong phương thức SendTo() và phương thức ReceiveFrom(). Nếu chương trình của chúng ta chỉ cần gửi và nhận dữ liệu từ một máy, chúng ta có thể dùng phương thức Connect().

Sau khi UDP socket được tạo ra, chúng ta có thể dùng phương thức Connect() giống như trong chương trình TCP để chỉ ra udp Server ở xa. Sau khi dùng phương thức Connect() xong ta có thể dùng phương thức Send() và Receive() để truyền tải dữ liệu giữa các thiết bị với nhau.

1.7. Kết luận chương 1

Chương 1 đã cung cấp một cái nhìn tổng quan về mô hình OSI (Open Systems Interconnection) và mô hình TCP/IP, hai khung làm việc quan trọng trong lĩnh vực mạng máy tính. Mô hình OSI phân chia quá trình truyền thông thành 7 lớp, từ lớp vật lý đến lớp ứng dụng, trong khi mô hình TCP/IP được sử dụng rộng rãi và chỉ có 4 lớp: lớp vật lý, lớp liên kết dữ liệu, lớp mạng và lớp ứng dụng. Giao thức TCP (Transmission Control Protocol) và UDP (User Datagram Protocol) là hai giao thức truyền thông quan trọng được sử dụng trong mô hình TCP/IP. TCP cung cấp một kết nối đáng tin cậy, đảm bảo việc truyền tải dữ liệu một cách có thứ tự và không bị mất. Ngược lại, UDP là một giao thức không kết nối, không đảm bảo tính đáng tin cậy, nhưng nó mang lại hiệu suất cao và thích hợp cho các ứng dụng cần truyền dữ liệu nhanh chóng như truyền tải âm thanh và video. Chương 1 cũng trình bày về lập trình Socket là một phần quan trọng trong việc phát triển ứng dụng mạng sử dụng TCP và UDP. Sockets cung cấp một cơ chế giao tiếp giữa các tiến trình trên mạng, cho phép truyền và nhận dữ liệu qua mạng. Việc lập trình socket TCP đòi hỏi xây dựng một kết nối đáng tin cậy giữa máy chủ và máy khách, trong khi lập trình socket UDP thì không yêu cầu kết nối trước, và dữ liệu được truyền mà không cần xác nhận.

CHƯƠNG 2. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

2.1. Bài toán lấy số tự động

Ngày nay nhiều cơ quan và tổ chức sử dụng hệ thống lấy số tự động để cải thiện quy trình phục vụ và trải nghiệm của khách hàng. Bài toán xử lý số tự động là việc tự động hóa quy trình cấp và quản lý số đối với khách hàng trong môi trường cơ quan, tổ chức hoặc các điểm dịch vụ khác. Bài toán được mô tả như sau:

Hệ thống cung cấp cơ chế cho khách hàng quy trình để tự lấy số một cách tự động thông qua máy lấy số. Máy lấy số là một thiết bị được đặt tại cơ quan hành chính công hoặc bất kỳ địa điểm nào mà yêu cầu lấy số tự động. Người dùng đến máy và sử dụng giao diện người dùng để chọn dịch vụ hoặc lĩnh vực cụ thể mà họ muốn sử dụng. Sau đó, máy lấy số cấp một số tự động cho khách hàng. Máy lấy số gửi thông tin về số được cấp và các thông tin liên quan khác lên máy tính để xử lý. Thông tin này có thể được chuyển qua mạng hoặc qua một kết nối.

Hệ thống quản lý hàng đợi để đảm bảo rằng các số tự động được xử lý theo thứ tự đến lượt. Các số tự động được cấp cho khách hàng được sắp xếp thành một hàng đợi trên màn hình hoặc trên một khu vực hiển thị khác. Hàng đợi này có thể bao gồm thông tin như số hiện tại và thời gian dự kiến cho đến khi đến lượt của khách hàng.

Các ưu điểm của hệ thống lấy số tự động như sau:

Tiết kiệm thời gian cho khách hàng: Khách hàng không cần phải đứng chờ đợi trong hàng dài. Họ có thể lấy số một cách nhanh chóng và tiện lợi, giảm thiểu thời gian mà họ phải dành cho việc đợi lượt.

Quy trình phục vụ linh hoạt: Hệ thống lấy số tự động giúp tổ chức quy trình phục vụ một cách linh hoạt hơn. Nhân viên có thể dễ dàng quản lý số tự động và điều chỉnh dịch vụ dựa trên nhu cầu thực tế.

Hiệu suất làm việc cao: Các cơ quan sử dụng hệ thống lấy số tự động thường có hiệu suất làm việc cao hơn. Việc tự động hóa quy trình lấy số giúp giảm áp lực công việc cho nhân viên và tối ưu hóa sử dụng nguồn nhân lực.

Cải thiện trải nghiệm người dùng: Khách hàng được trải nghiệm dịch vụ một cách thuận tiện hơn và thoải mái hơn. Họ có thể chủ động lấy số và quản lý thời gian của mình mà không cần phải chờ đợi.

Đồng bộ hóa thông tin: Hệ thống lấy số thường được kết nối với máy tính và hệ thống quản lý thông tin, giúp đồng bộ hóa thông tin giữa các bước trong quy trình phục vụ làm giảm nguy cơ sai sót và tăng cường chính xác thông tin.

Đối với bệnh viện, lấy số tự động để sử dụng dịch vụ khám chữa bệnh của khách hàng và xử lý số một cách tự động được thiết kế để giúp quản lý lưu lượng người đến, giảm thời gian chờ đợi và tăng hiệu suất trong quá trình phục vụ. Khách

hàng là người lấy số tại hệ thống, dữ liệu số thứ tự khách hàng được truyền về các máy xử lý, máy xử lý có nhiệm vụ đọc số và thông báo thông tin liên quan đến khách hàng. Quy trình thực hiện gồm các bước sau:

Bước 1: Khách hàng lấy số

- Khách hàng sử dụng máy lấy số để lấy số thứ tự cho lượt khám chữa bệnh của mình tại bệnh viện.
- Hệ thống ghi lại số thứ tự của khách hàng.

Bước 2: Truyền dữ liệu về máy xử lý

Dữ liệu số thứ tự được truyền về máy xử lý qua kết nối mạng hoặc giao thức liên kết.

Bước 3: Máy xử lý đọc số

- Máy xử lý nhận dữ liệu số thứ tự từ hệ thống.
- Thực hiện xử lý để đảm bảo tính chính xác của dữ liệu.

Bước 4: Hiển thị số và thông báo

- Máy xử lý hiển thị số thứ tự của khách hàng trên màn hình hoặc bảng hiển thị.
- Phát âm thanh hoặc thông báo trực tuyến thông báo đến khách hàng về số đang được phục vụ.

Bước 5: Gửi thông tin về hệ thống chính

- Máy xử lý cập nhật trạng thái số thứ tự đã được phục vụ và gửi thông tin về hệ thống chính.
- Hệ thống chính có thể sử dụng thông tin này để quản lý lưu lượng và cung cấp thông báo đến các khu vực khác nếu cần.

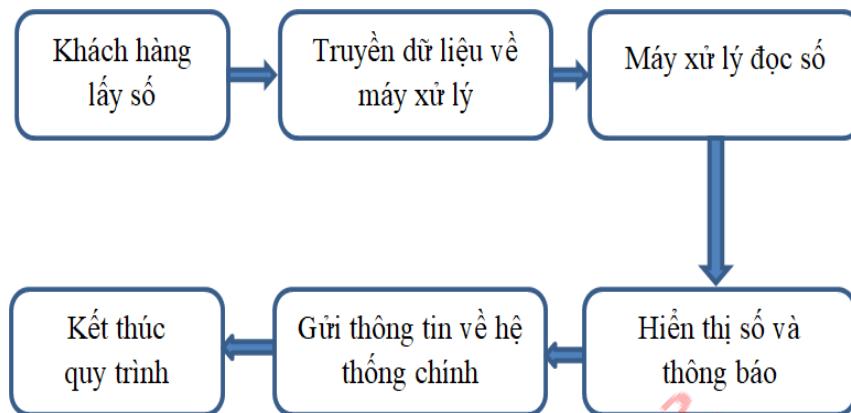
Bước 6: Kết thúc quy trình

Khi khách hàng đã được phục vụ, quy trình kết thúc và chuẩn bị cho khách hàng tiếp theo.

Thông thường, với tình trạng khách hàng là đối tượng khẩn cấp cần phục vụ như cấp cứu, yêu cầu này sẽ được ưu tiên phục vụ ngoài không cần lấy số tự động theo thứ tự. Tuy nhiên, một số hệ thống lấy số tự động có cơ chế ưu tiên hoặc lựa chọn dịch vụ cho khách hàng sử dụng để tự động hóa hầu hết quy trình. Hệ thống lấy số tự động dạng này giúp định rõ mức độ ưu tiên của từng trường hợp dựa trên triệu chứng và mức độ nghiêm trọng. Khi đó, bằng cách sử dụng hệ thống lấy số tự động, nhân viên y tế có thể ước lượng mức độ cần thiết của sự can thiệp dựa trên độ ưu tiên. Các trường hợp nghiêm trọng sẽ được ưu tiên cao hơn để nhận được sự chăm sóc ngay lập tức.

Dựa trên đánh giá, bệnh nhân sẽ được phân loại vào các mức độ khẩn cấp khác nhau, từ nhẹ đến nặng.

Quy trình lấy số thông thường được thể hiện trong sơ đồ sau:



Hình 2.1. Quy trình lấy và xử lý số thứ tự tự động

2.2. Khảo sát và đánh giá hiện trạng

2.2.1. Phân tích về đối tượng xử lý

Ứng dụng lấy số thứ tự khám bệnh gồm các đối tượng sau:

Khách hàng: Là người đến bệnh viện và sử dụng dịch vụ khám chữa bệnh, họ thường sẽ thực hiện các bước sau:

Bước 1: Lấy số đánh dấu

Khách hàng sẽ sử dụng máy bấm số để lấy số đánh dấu. Máy sẽ cung cấp các nút hoặc màn hình cảm ứng để khách hàng có thể lấy số đánh dấu cho thứ tự được phục vụ. Mỗi số thường đi kèm với thông tin về loại dịch vụ hoặc phòng mà họ cần. Máy in một tờ vé có số hoặc tạo một số điện tử để gửi đến khách hàng. Một máy in hoặc màn hình hiển thị sẽ cung cấp số cho khách hàng. Máy tạo số sẽ kết nối với các máy xử lý khác giúp các máy hoạt động đồng bộ với toàn bộ hệ thống và theo dõi trạng thái của dịch vụ.

Bước 2: Theo dõi màn hình hiển thị

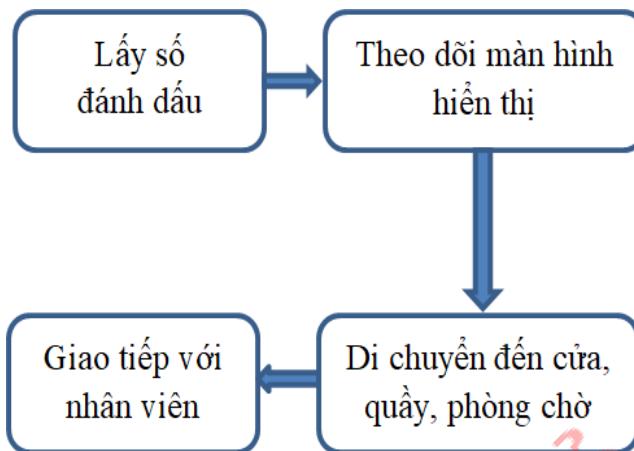
Có một màn hình hiển thị số gọi tiếp theo hoặc thông báo vị trí nơi khách hàng cần đến. Khách hàng sẽ theo dõi màn hình để biết khi nào đến lượt của họ.

Bước 3: Di chuyển đến quầy, cửa, phòng chờ

Khi số của họ được gọi, khách hàng sẽ di chuyển đến phòng hoặc khu vực được chỉ định để nhận dịch vụ hoặc chờ đợi.

Bước 4: Giao tiếp với nhân viên

Tại phòng, cửa hoặc quầy, khách hàng sẽ gặp gỡ nhân viên y tế hoặc nhân viên đang phục vụ. Họ sẽ trình bày vấn đề hoặc mục đích của mình để được hỗ trợ.



Hình 2.2. Quy trình thực hiện của khách hàng (lấy số)

Nhân viên: Là người làm việc tại bệnh viện và xử lý các yêu cầu về khám chữa bệnh của khách hàng. Nhân viên tại quầy xử lý số, hay còn gọi là "phòng xử lý số", có trách nhiệm quản lý quá trình gọi số của bệnh nhân tại bệnh viện, họ thường sẽ thực hiện các bước sau:

Bước 1: Thông báo số được mời xử lý

Nhân viên sẽ làm việc với một máy tính ở khu vực xử lý gọi số trong hệ thống bấm số tự động, đầu tiên họ thông báo khách hàng nào được phục vụ (theo thứ tự số bấm trong hàng đợi).

Bước 2: Xử lý dịch vụ mà khách hàng yêu cầu

Nhân viên xử lý dịch vụ mà khách hàng yêu cầu và thông báo thông tin liên quan đến khách hàng và xác nhận tình trạng xử lý xong yêu cầu của khách hàng.



Hình 2.3. Quy trình thực hiện của nhân viên (xử lý số)

2.2.2. Phân tích về chức năng nghiệp vụ

Hệ thống xử lý số thứ tự khám bệnh gồm 2 phần chính gọi là hai khu vực: Khu vực lấy số thứ tự tự động (dành cho khách hàng) và khu vực xử lý yêu cầu của khách hàng theo số thứ tự (dành cho nhân viên tại các cửa, quầy, phòng).

- Lấy số thứ tự tự động: Cho phép khách hàng bấm lấy số thứ tự, có thể bao gồm hiển thị thông báo lên màn hình LED, phát trên loa gọi khách hàng hoặc có thể thực hiện trên hệ thống khác gọi là hệ thống chung tại khu vực chờ đợi.

- Xử lý yêu cầu của khách hàng theo số thứ tự: Cho phép nhân viên làm việc tại các quầy, cửa, phòng xử lý hiển thị số. Máy sẽ hiển thị số của khách hàng đang được phục vụ lên màn hình hoặc bảng hiển thị tại khu vực xử lý số (khu vực 1) hoặc khu vực chung. Máy cũng có thể phát âm thanh thông báo, thông báo số của khách hàng để họ biết đến lượt của mình. Thông báo có thể được thực hiện thông qua loa hoặc tai nghe đặt sẵn tại hệ thống chung tại khu vực chờ đợi.

2.2.3. Phân tích yêu cầu phi chức năng

- Bộ cục ứng dụng khoa học và phù hợp với thực tế.
- Có tính mở để nâng cấp, phát triển khi cần thiết.
- Ứng dụng phải đảm bảo thời gian phản hồi nhanh chóng khi người dùng yêu cầu lấy số thứ tự khám bệnh. Không gặp tình trạng gián đoạn hoặc trễ hẹn trong quá trình cập nhật số thứ tự mới.
- Ứng dụng cần có khả năng mở rộng để xử lý một lượng lớn người dùng đồng thời, đặc biệt là trong các khu vực có nhiều người sử dụng.
- Bảo vệ thông tin cá nhân của người dùng và đảm bảo rằng số thứ tự khám bệnh chỉ có thể được truy cập bởi những người có quyền. Ứng dụng phải có cơ chế xác thực mạnh mẽ để đảm bảo rằng thông tin được truy cập là chính xác và an toàn.
- Cung cấp chức năng thông báo để thông báo người dùng khi số thứ tự của họ đã sắp đến.
- Giao diện người dùng được thiết kế sao cho dễ sử dụng và dễ hiểu, ngay cả đối với người dùng không chuyên nghiệp về công nghệ.
- Cho phép quản trị viên điều chỉnh và cấu hình các thông số như thời gian chờ đợi tối đa, giờ làm việc, tên máy xử lý và các thiết lập khác để đáp ứng nhu cầu cụ thể của từng cơ sở y tế.

2.3. Phân tích phần cứng và chức năng phần mềm

2.3.1. Phân tích chung

Hệ thống bao gồm một Server và hai Client làm nhiệm vụ tạo số tự động cho bệnh nhân và xử lý hàng đợi tại hai quầy khác nhau (với hệ thống có nhiều quầy thì tương tự như hệ thống 2 quầy). Hệ thống này giúp tự động hóa quá trình tạo số cho

bệnh nhân và quản lý hiệu quả hàng đợi, đồng thời thông báo và xóa bệnh nhân sau khi được xử lý để đảm bảo tính nhất quán và hiệu suất trong quá trình phục vụ. Chi tiết về cấu trúc và hoạt động của hệ thống được mô tả như sau:

Server:

Mô tả chung:

- Chịu trách nhiệm tạo số tự động cho bệnh nhân và quản lý hàng đợi.
- Cung cấp API hoặc giao diện để cho phép Client tương tác với hệ thống.

Chức năng:

- Tạo số tự động cho bệnh nhân khi có yêu cầu từ Client.
- Quản lý hàng đợi, bao gồm thêm bệnh nhân vào hàng đợi và xóa bệnh nhân khi đã được xử lý.

Client 1 và Client 2:

Mô tả chung: Hai Client đặt ở hai quầy khác nhau, đại diện cho hai điểm phục vụ trong hệ thống.

Chức năng:

- Gửi yêu cầu tạo số tự động cho bệnh nhân đến Server.
- Kiểm tra hàng đợi để xem liệu có bệnh nhân nào cần xử lý không.
- Nếu có bệnh nhân trong hàng đợi, xử lý bệnh nhân và thông báo cho Client còn lại.
- Thông báo cho bệnh nhân vừa xử lý xong và xóa bệnh nhân khỏi hàng đợi.

Tương tác:

- Khi Client 1 xử lý một bệnh nhân, nó thông báo cho Client 2 để tránh xử lý trùng lặp.
- Khi một bệnh nhân được xử lý xong, cả hai Client đều thông báo cho bệnh nhân và xóa khỏi hàng đợi.

Luồng công việc chính:

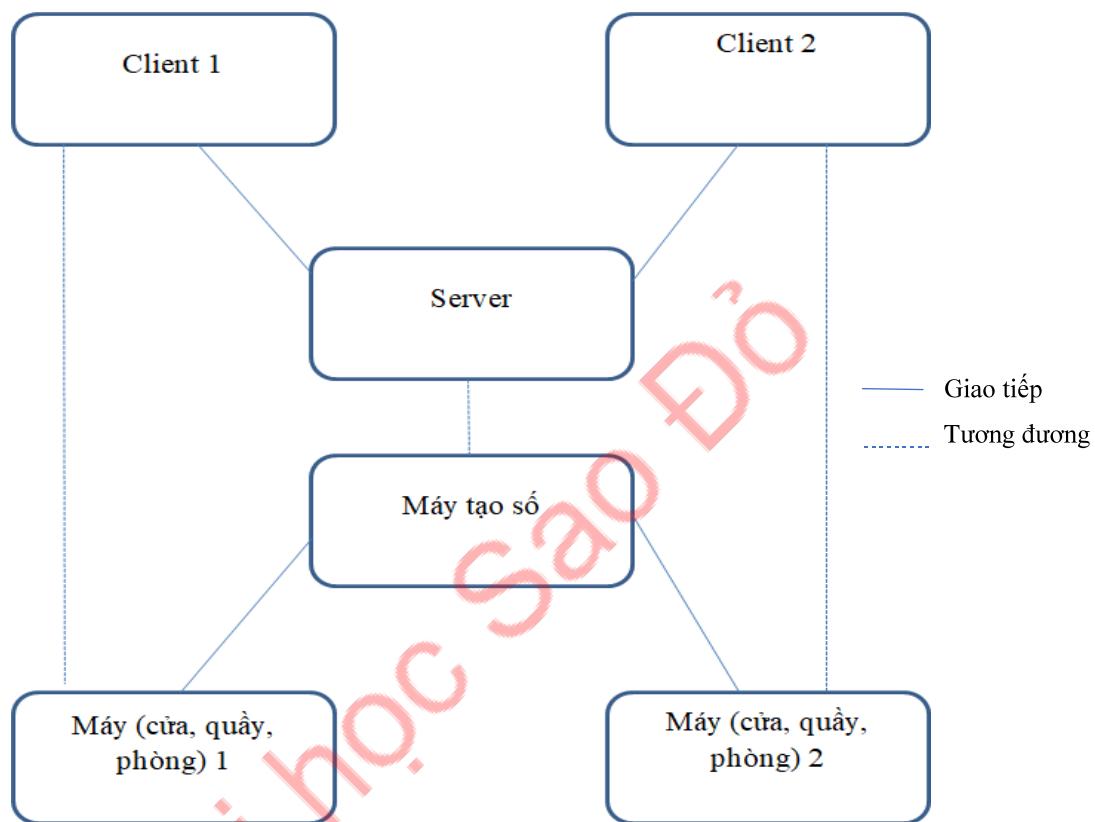
- Yêu cầu tạo số tự động: Client 1 và Client 2 gửi yêu cầu tạo số tự động đến Server.
- Tạo số và thêm vào hàng đợi: Server tạo số tự động cho bệnh nhân và thêm vào hàng đợi.
- Kiểm tra và xử lý hàng đợi: Cả hai Client kiểm tra hàng đợi để xem liệu có bệnh nhân nào cần xử lý không.
- Xử lý bệnh nhân:
 - + Nếu có bệnh nhân trong hàng đợi, Client thực hiện xử lý.
 - + Thông báo cho Client còn lại để tránh xử lý trùng lặp.

- Thông báo và xóa khỏi hàng đợi:

- + Sau khi xử lý xong, cả hai Client đều thông báo cho bệnh nhân.
- + Xóa bệnh nhân đã được xử lý khỏi hàng đợi.

2.3.2. Phân tích sơ bộ phần cứng thiết bị

Sơ đồ phân cứng thiết bị như sau:



Hình 2.4. Sơ đồ phân cứng mô hình Client – Server và ánh xạ vào ứng dụng

Hệ thống có thể thực thi ở 2 trường hợp: Trên mạng Internet và mạng LAN tại bệnh viện.

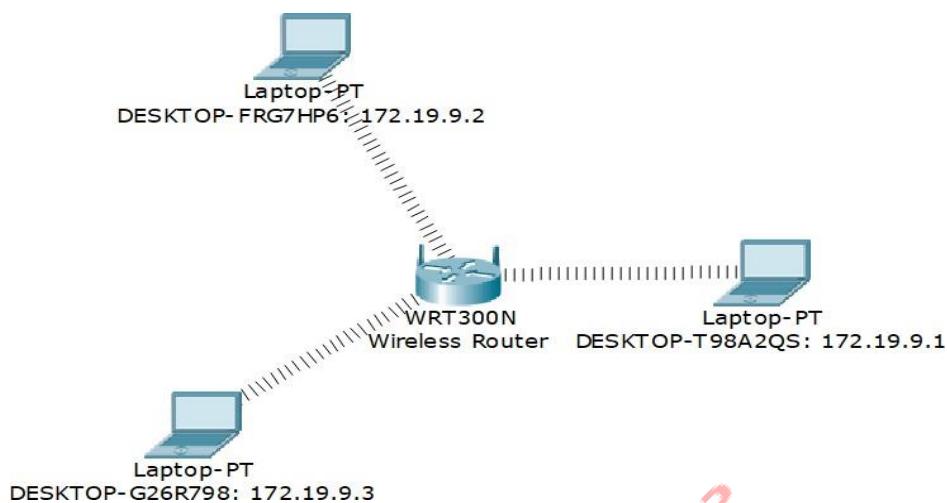
Trường hợp 1: Sơ đồ hệ thống triển khai trên mạng Internet

- Mô hình triển khai hai máy tính kết nối với nhau bằng mạng Internet, địa chỉ IP tĩnh hoặc động cho 2 PC, sử dụng lệnh Ping để kiểm tra kết nối, khảo sát gói tin ICMP giữa hai máy tính bằng Packet Tracer như sau:



Hình 2.5. Mô hình 2 máy tính kết nối mạng Internet

Mô hình triển khai nhiều máy tính nối với nhau bằng mạng Internet, đặt IP tĩnh cho các máy tính, kiểm tra kết nối giữa các máy tính bằng Packet Tracer như sau:



Hình 2.6. Mô hình nhiều máy tính kết nối mạng Internet

Mô hình triển khai nhiều máy tính nối với nhau theo mô hình Client – Server trong cùng mạng bằng mạng Internet (DESKTOPT98A2QS là Server), đặt IP tĩnh cho các máy tính, kiểm tra kết nối giữa các máy tính bằng Packet Tracer như sau:



Hình 2.7. Mô hình nhiều máy tính kết nối mạng Internet theo mô hình Client – Server

Trong trường hợp địa chỉ IP động, cần biết được địa chỉ IP của máy Server để thực hiện kết nối Client vào hệ thống.

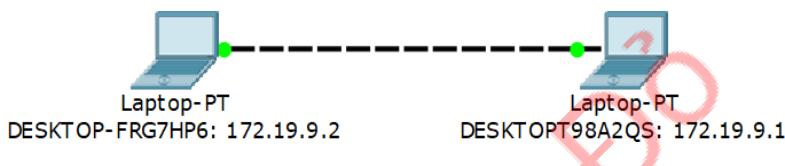
Trường hợp 2: Mô hình Client – Server trên mạng LAN

Bảng 2.1. Phần cứng thiết bị

Mô hình Client – Server	Ứng dụng lấy số thứ tự tự động
Server	Host name: DESKTOP-T98A2QS IP: 172.19.9.1

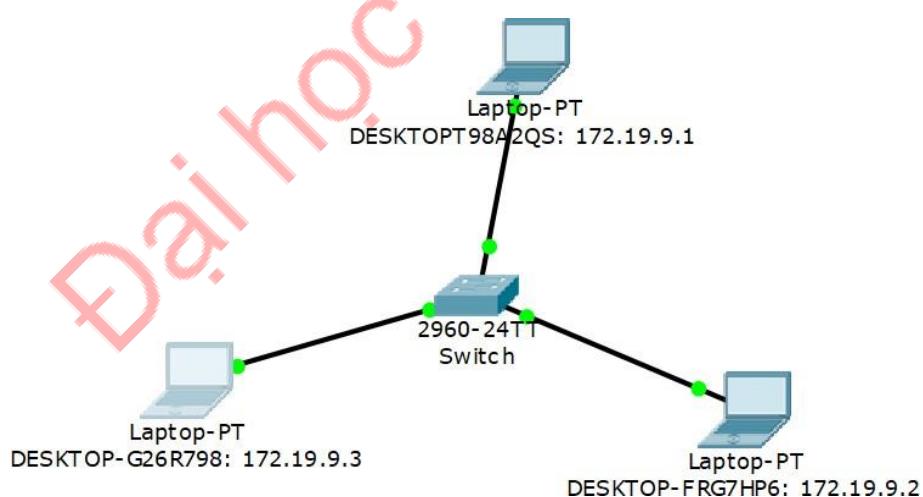
Mô hình Client – Server	Ứng dụng lấy số thứ tự tự động
Client 1	Host name: DESKTOP-FRG7HP6 IP: 172.19.9.2
Client 2	Host name: DESKTOP-G26R798 IP: 172.19.9.3

- Mô hình triển khai hai máy tính kết nối với nhau bằng mạng LAN, đặt IP tĩnh cho 2 PC, sử dụng lệnh Ping để kiểm tra kết nối, khảo sát gói tin ICMP giữa hai máy tính bằng Packet Tracer như sau:



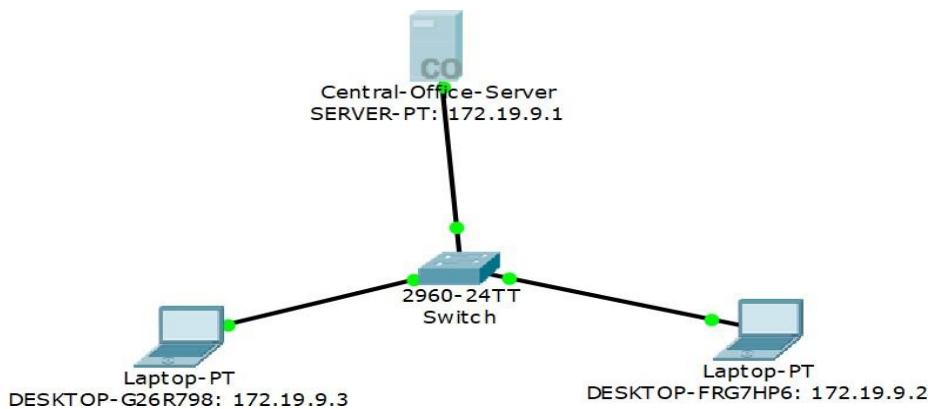
Hình 2.8. Mô hình 2 máy tính kết nối mạng LAN

Mô hình triển khai nhiều máy tính kết nối với nhau bằng mạng Internet, đặt IP tĩnh cho các máy tính, kiểm tra kết nối giữa các máy tính bằng Packet Tracer như sau:



Hình 2.9. Mô hình nhiều máy tính kết nối mạng LAN

Mô hình triển khai nhiều máy tính kết nối với nhau theo mô hình Client – Server trong cùng mạng bằng mạng Internet (DESKTOPT98A2QS là Server), đặt IP tĩnh cho các máy tính, kiểm tra kết nối giữa các máy tính bằng Packet Tracer như sau:



Hình 2.10. Mô hình nhiều máy tính kết nối mạng LAN theo mô hình Client – Server

2.3.3. Phân tích chức năng phần mềm

Trên cơ sở mô tả các yêu cầu chức năng, ta xác định các chức năng cơ bản của ứng dụng lấy số thứ tự khám bệnh đối với từng đối tượng như sau:

Bảng 2.2. Chức năng của đối tượng trong hệ thống

Chức năng	Đối tượng	Thao tác
Lấy số thứ tự	Khách hàng	<ul style="list-style-type: none"> - Nhấn nút lấy số. - Gửi yêu cầu đến hệ thống: Số thứ tự được đưa vào hàng đợi tại các quầy xử lý của nhân viên.
	Hệ thống	<ul style="list-style-type: none"> - Gửi yêu cầu là số thứ tự của khách hàng đến máy của nhân viên xử lý. - Chuyển tiếp hệ thống.
Xử lý số thứ tự	Nhân viên	<ul style="list-style-type: none"> - Xử lý số (xóa khỏi hàng đợi). - Gửi yêu cầu hệ thống.
	Hệ thống	<ul style="list-style-type: none"> - Thông báo đến khách hàng: Hiển thị số thứ tự và thời gian xử lý số lên quầy. - Thông báo khách hàng nào, xử lý tại quầy nào đến hệ thống chờ (hệ thống chung) để hiển thị trên màn hình LED hoặc loa thông báo.

2.3.4. Xây dựng logic chức năng

Hệ thống gồm hai chức năng chính: Lấy số và xử lý số. Logic chức năng được thể hiện trong bảng sau:

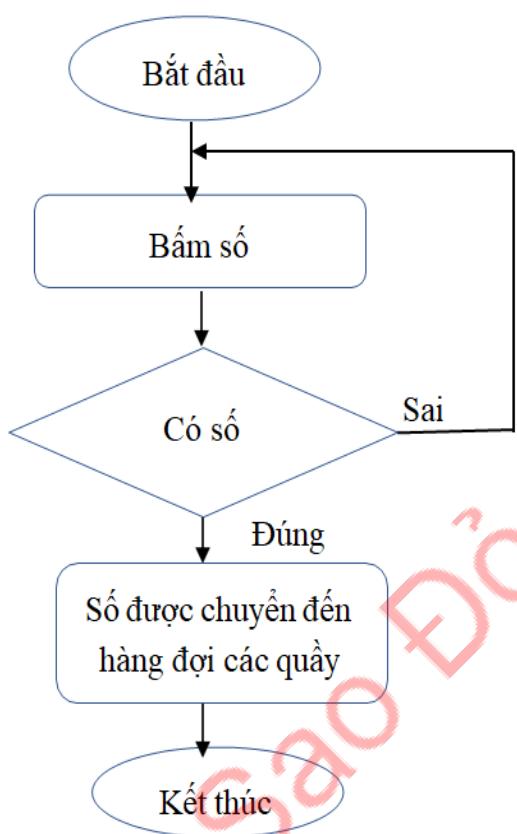
Bảng 2.3. Chức năng lấy số thứ tự

Mô tả	Khách hàng lấy số thứ tự khám bệnh
Đầu vào	Số thứ tự (được sinh tự động theo chế độ tự tăng, khách hàng không được nhập số tùy ý)
Quá trình xử lý	<ul style="list-style-type: none"> - Khách hàng bấm lấy số thứ tự. - Số thứ tự được gửi đồng bộ đến hàng đợi tại các quầy.
Kiểm tra hợp lệ	<ul style="list-style-type: none"> - Các quầy đã sẵn sàng (đã kết nối với máy lấy số) - Khách hàng có bấm lấy số.
Đầu ra	Hiển thị số thứ tự mà khách hàng đã lấy (phục vụ cho việc chờ đến phiên xử lý).

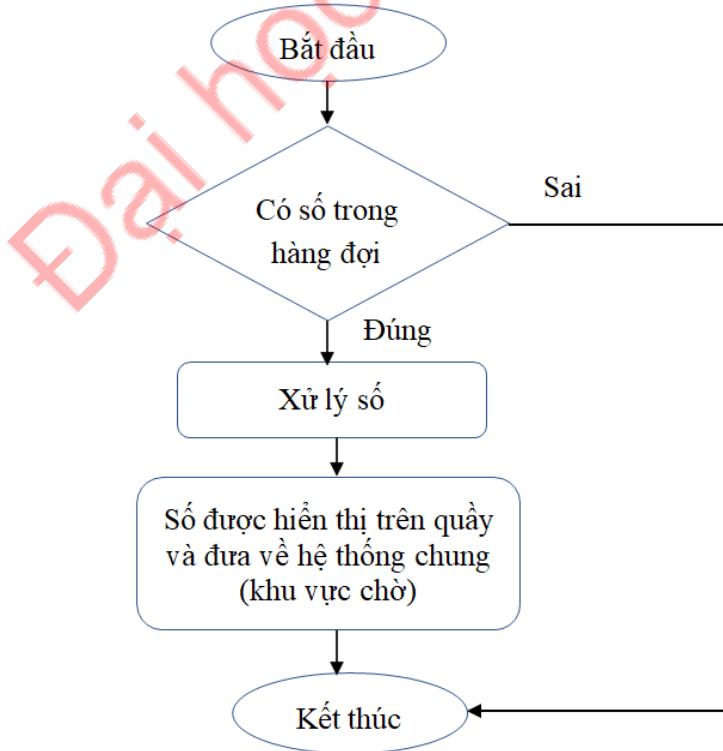
Bảng 2.4. Chức năng xử lý số thứ tự

Mô tả	Xử lý số thứ tự theo yêu cầu của khách hàng trong hàng đợi.
Đầu vào	Số thứ tự của khách hàng đã lấy số.
Quá trình xử lý	<ul style="list-style-type: none"> - Bấm xử lý. - Số đầu tiên trong hàng đợi (khách hàng có thời gian chờ lâu nhất) được xử lý trước. - Hiển thị số được xử lý lên màn hình tại quầy. - Số thứ tự đã xử lý được xóa khỏi hàng đợi ở tất cả các quầy. - Thông báo (hiển thị màn hình LED hoặc loa) đến máy lấy số của khách hàng hoặc khu vực chờ thông tin quầy nào xử lý khách hàng nào.
Kiểm tra hợp lệ	Hàng đợi không rỗng, tồn tại khách hàng cần xử lý.
Đầu ra	Hiển thị kết quả.

Sơ đồ thuật toán các chức năng lấy số và xử lý số thứ tự như sau:



Hình 2.11. Sơ đồ chức năng lấy số thứ tự khám bệnh



Hình 2.12. Sơ đồ chức năng xử lý số thứ tự khám bệnh

2.4. Kết luận chương 2

Chương 2 của đồ án tập trung vào bài toán lấy số tự động trong bệnh viện, cung cấp cái nhìn tổng quan về các yếu tố quan trọng cần được xem xét và thiết kế để đảm bảo chức năng, hiệu suất và tính ổn định của hệ thống. Cụ thể, chương này tiến hành một số phân tích chi tiết về đối tượng xử lý, yêu cầu chức năng và phi chức năng như giao diện, độ tin cậy, hiệu suất và an toàn của hệ thống, cân nhắc về khả năng tương tác với người dùng và đảm bảo tính thân thiện, dễ sử dụng, cũng như phân tích về phần cứng và chức năng phần mềm để đảm bảo yêu cầu tối thiểu của hệ thống lấy số.

Đại học Sao Đỏ

CHƯƠNG 3. XÂY DỰNG ỨNG DỤNG CLIENT – SERVER LẤY SỐ KHÁM BỆNH TỰ ĐỘNG

3.1. Lựa chọn công cụ

Lập trình ứng dụng mạng là một công việc nhằm phát triển các ứng dụng doanh nghiệp hoạt động trên mạng. Theo đó, người lập trình cần viết ra một chương trình mạng để chạy những chương trình trên các máy tính khác nhau, giúp các nút mạng ấy chia sẻ dữ liệu, truyền tin an toàn, hiệu quả. Bản chất của lập trình mạng chính là sử dụng các công cụ lập trình để tạo ra một chương trình giúp các máy tính giao tiếp với nhau. Các công cụ bao gồm ngôn ngữ lập trình; kiến thức về phân tích thiết kế hệ thống, kiến thức hệ thống mạng, mô hình xây dựng chương trình ứng dụng mạng, kiến thức về cơ sở dữ liệu, ...

Hầu hết các ngôn ngữ lập trình đều có thể sử dụng để lập trình mạng, tuy nhiên việc lập trình mạng còn phụ thuộc vào các thư viện và môi trường lập trình có hỗ trợ hay không. Có thể liệt kê các ngôn ngữ lập trình sử dụng để lập trình mạng như sau:

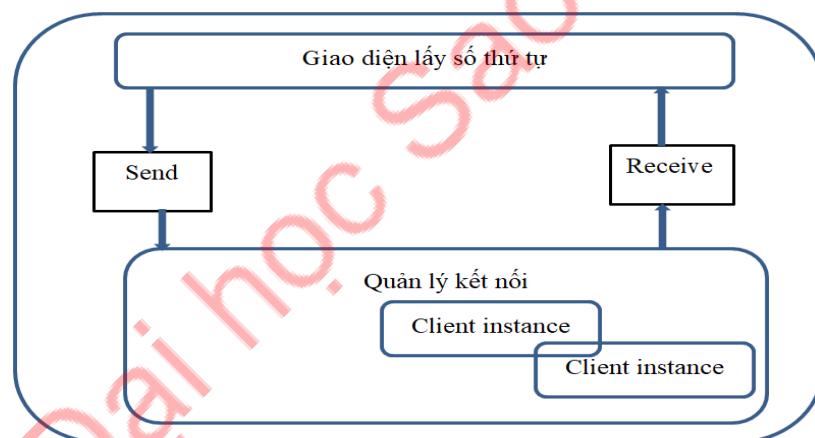
- C/C++: Ngôn ngữ lập trình rất mạnh và phổ biến, dùng để viết mọi loại ứng dụng trong đó có ứng dụng mạng.
- Java: Ngôn ngữ lập trình khá thông dụng và hỗ trợ trên nhiều môi trường, trong đó có thể viết ứng dụng chạy trên điện thoại di động.
- C#: Ngôn ngữ lập trình cũng rất mạnh và dễ sử dụng, chỉ hỗ trợ trên hệ điều hành Windows của Microsoft.
- Python, Perl, PHP...: Các ngôn ngữ thông dịch, sử dụng để viết nhanh các tiện ích nhỏ một cách nhanh chóng, trong đó có ứng dụng mạng.
- Việc lập trình mạng phụ thuộc rất nhiều vào các thư viện hỗ trợ đến từ hệ thống. Tùy thuộc vào nền tảng phát triển ứng dụng mà có thể sử dụng các thư viện khác nhau. Có thể liệt kê một vài thư viện hỗ trợ lập trình mạng như sau:
 - Winsock: Thư viện liên kết động của Microsoft, được phân phối cùng hệ điều hành Windows. Winsock cung cấp khá nhiều API để phát triển ứng dụng mạng. Winsock có thể sử dụng cùng bất kỳ ngôn ngữ lập trình nào, bộ đôi C/C++ với Winsock đem lại hiệu năng cao nhất, nhưng khó sử dụng.
 - Thư viện System.Net trong .NET framework: Thư viện cung cấp nhiều API để sử dụng để xây dựng ứng dụng mạng. Để sử dụng thư viện này, người ta thường dùng C#, việc phát triển ứng dụng mạng khá dễ dàng.
 - Thư viện MFC Socket: Thư viện đi kèm bộ phát triển Visual Studio C++.

Ứng dụng được xây dựng bằng Visual Studio là một IDE (Integrated Development Environment) mạnh mẽ và linh hoạt, hỗ trợ nhiều ngôn ngữ lập trình, cung cấp các công cụ debug và quản lý dự án hiệu quả. Đồng thời chọn ngôn ngữ C# và mô hình Client – Server với TCP Socket để xây dựng ứng dụng Server tạo số tự

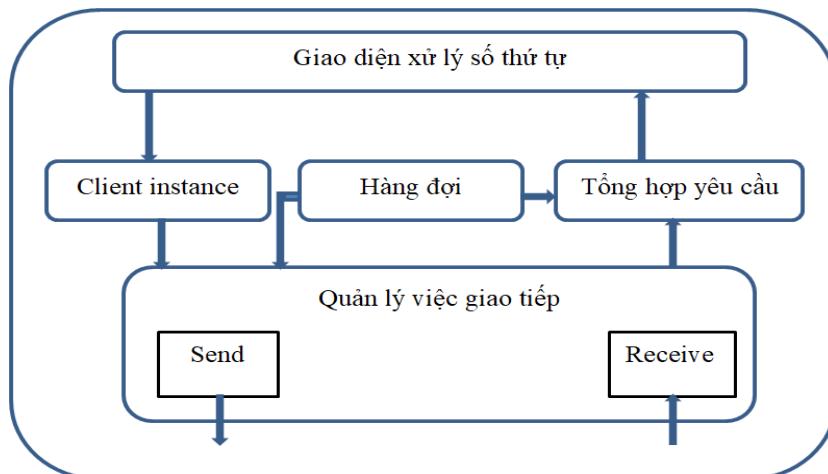
động và các Client xử lý xóa sổ, gửi thông tin số đã xóa cho các Client và thông báo đến Server qua loa. Ngôn ngữ C# tích hợp chặt chẽ với .NET Framework cung cấp nhiều thư viện mạnh mẽ và các tính năng đa dạng giúp phát triển ứng dụng linh hoạt. Hơn nữa, mô hình Client – Server và TCP Socket quản lý giao tiếp giữa các thành phần của hệ thống, chia thành Client (điều khiển xóa sổ) và Server (tạo số tự động) thiết lập kết nối ổn định, đảm bảo truyền tải dữ liệu đáng tin cậy giữa Client và Server mà không bị gián đoạn hoặc mất mát kết nối. TCP cũng cung cấp cơ chế phục hồi lỗi tích hợp, đảm bảo rằng dữ liệu được chuyển đến đúng theo đúng thứ tự mà nó được gửi, điều này là quan trọng trong các ứng dụng yêu cầu sự liên tục và tuân thủ thứ tự. Ngoài ra, TCP hỗ trợ quản lý hiệu quả đa luồng, cho phép nhiều luồng dữ liệu chạy đồng thời mà không gây xung đột hoặc hiện tượng đối ứng. TCP có thể được kết hợp với các lớp bảo mật như TLS/SSL để đảm bảo an toàn trong quá trình truyền tải dữ liệu qua mạng. TCP hỗ trợ nhiều kết nối đồng thời giữa các Client và Server, cho phép ứng dụng mở rộng để phục vụ nhiều người dùng. Trong đồ án này sử dụng C#.Net.

3.2. Xây dựng logic chức năng

3.2.1. Kiến trúc mô hình



Hình 3.1. Kiến trúc Server



Hình 3.2. Kiến trúc Client

3.2.2. Ứng dụng phía Server

Các chức năng chính phía Server được thể hiện trong bảng sau:

Bảng 3.1. Các chức năng chính phía Server

Phương thức và sự kiện	Mô tả
Connect()	Tạo một Server TCP, lắng nghe các kết nối từ Client và tạo các luồng riêng biệt để xử lý dữ liệu từ mỗi kết nối.
Send()	Gửi dữ liệu từ Server đến Client thông qua kết nối Socket đã được thiết lập sau khi thông điệp được mã hóa thành mảng byte.
Receive()	Nhận dữ liệu từ một Client thông qua kết nối Socket
Serialize()	Đóng gói (packing) hoặc mã hóa (encoding), chuyển đổi chuỗi thành một mảng byte trước khi gửi đi.
Deserialize()	Giải mã một mảng byte thành một đối tượng chuỗi.
AddMessenger()	Thêm các số thứ tự vào danh sách (màn hình) cho khách hàng quan sát. Định dạng số thứ tự dạng 01 đến 100 hoặc 0001 đến 1000 tùy thuộc số lượng khách hàng truy cập sử dụng lấy số.
lollipopBtnSend_Click()	Sự kiện nhấn lấy số tự động, tạo một tin nhắn với định dạng số và gửi đến tất cả các Client trong danh sách Client. Sau đó, hiển thị thông điệp trên giao diện người dùng.
NotifyClientsToDelete()	Thông báo cho tất cả các Client khác về yêu cầu xóa, gửi yêu cầu xóa đến tất cả Client (trừ Client gửi yêu cầu), sau đó cập nhật giao diện người dùng để hiển thị thông điệp xóa. Mục đích sử dụng để đồng bộ hóa thông tin giữa các Client kết nối với Server.
AddClientNotification()	Thêm một thông báo vào màn hình LED và sau chuyển đổi văn bản thông báo thành giọng nói để phát ra loa.

Trong đó các hàm xử lý phía Server thực hiện như sau:

Hàm Connect():

Bước 1: Khởi tạo các đối tượng

ListClient: Danh sách để lưu trữ các đối tượng Socket đại diện cho các kết nối từ các Client.

IP: Đối tượng IPEndPoint xác định địa chỉ IP và cổng mà Server sẽ lắng nghe.

Server: Đối tượng Socket đại diện cho Server.

Bước 2: Gắn địa chỉ và cổng cho Server

Server được gắn với địa chỉ IP và cổng mà nó lắng nghe: Server.Bind(IP).

Bước 3: Lắng nghe kết nối

Server được đặt ở trạng thái lắng nghe và sẽ chấp nhận các kết nối từ Client với số lượng kết nối tối đa được chấp nhận đồng thời. Trong trường hợp này đặt 100 kết nối đồng thời: **Server.Listen(100)**

Bước 4: Tạo luồng lắng nghe các kết nối đến từ Client.

Thread listen = new Thread(...)

Bước 5: Chấp nhận kết nối mới

Chờ đến khi có một kết nối mới và tạo một đối tượng Socket đại diện cho kết nối đó: (Socket client = Server.Accept())

Bước 6: Thêm kết nối vào danh sách

Thêm đối tượng Socket của Client mới vào danh sách Client để theo dõi các kết nối đang hoạt động: (ListClient.Add(client))

Bước 7: Bắt đầu luồng xử lý kết nối dữ liệu từ Client: receive.Start(client)

Hàm Send():

Bước 1: Kiểm tra kết nối Server

Kiểm tra xem đối tượng Client đã được thiết lập trước khi gửi dữ liệu.

Bước 2: Gửi dữ liệu

Server.Send(...): Gửi dữ liệu đã được serialize từ Server đến Client thông qua kết nối Socket. Hàm Send của đối tượng Socket này được sử dụng để thực hiện việc này. Lưu ý rằng dữ liệu gửi đi thường được đóng gói thành các gói tin và phải được chuyển đổi (mã hóa dữ liệu) thành dạng dữ liệu có thể được truyền đi thông qua mạng (ví dụ biến đối tượng thành một mảng byte).

Hàm Receive():**Bước 1: Khởi tạo đối tượng nhận**

Do đối tượng gửi đi là các gói tin đã mã hóa sang dạng mảng byte nên khi nhận cần có mảng byte chứa gói tin này.

Khởi tạo một mảng byte để chứa dữ liệu nhận được từ Client. Trong trường hợp này, kích thước của mảng là 1024 * 5000 byte:

```
byte[] data = new byte[1024 * 5000];
```

Bước 2: Nhận dữ liệu từ Client

Sử dụng phương thức Receive của đối tượng Socket để nhận dữ liệu từ Client và lưu trữ số lượng byte nhận được vào biến bytesRead:

```
int bytesRead = clientSocket.Receive(data);
```

Bước 3: Giải mã dữ liệu từ mảng byte thành chuỗi

```
string message = (string)Deserialize(data);
```

Bước 4: Thông báo cho các Client khác về yêu cầu xóa

```
NotifyClientsToDelete();
```

Bước 5: Cập nhật kết quả

Lặp qua danh sách các Client.

```
foreach (Socket item in ListClient);
```

Kiểm tra và gửi thông điệp đã nhận được đến tất cả các client khác (trừ Client gửi thông điệp này). Nếu đang chạy trong một luồng không phải luồng chính (InvokeRequired), thì sử dụng Invoke để thực hiện một hành động trên luồng chính. Sau đó, gọi hàm AddClientNotification để thêm thông điệp vào giao diện người dùng trên bảng LED (listBox2) và phát ra loa.

Hàm Serialize():**Bước 1: Tạo đối tượng để chứa dữ liệu đã được đóng gói**

Tạo một đối tượng MemoryStream, một luồng mà dữ liệu có thể được ghi vào và đọc ra từ bộ nhớ (memory).

```
MemoryStream stream = new MemoryStream();
```

Bước 2: chuyển đổi tượng thành dạng dữ liệu nhị phân

```
BinaryFormatter formatter = new BinaryFormatter();
```

Để serialize đối tượng (obj) và ghi kết quả vào MemoryStream (stream).

Bước 3: Chuyển thành mảng byte

```
return stream.ToArray();
```

Chuyển đổi nội dung của MemoryStream thành một mảng byte bằng cách sử dụng phương thức `ToArray()`. Mảng byte này là kết quả của quá trình serialize và nó có thể được sử dụng để truyền dữ liệu qua mạng hoặc lưu trữ dữ liệu.

Hàm Deserialize():**Bước 1: Tạo luồng dữ liệu có thể được đọc ra từ bộ nhớ**

Đối tượng `MemoryStream` này sẽ được sử dụng để chứa dữ muôn giải mã:

```
MemoryStream stream = new MemoryStream(data);
```

Bước 2: Tạo đối tượng để thực hiện quá trình serialize và deserialize dữ liệu.

Tạo một đối tượng `BinaryFormatter`, một đối tượng trong .NET Framework được sử dụng để thực hiện quá trình serialize và deserialize dữ liệu:

```
BinaryFormatter format = new BinaryFormatter();
```

Bước 3: Chuyển đổi dữ liệu từ dạng nhị phân thành một đối tượng tương ứng

```
return format.Deserialize(stream);
```

Sự kiện Click lollipopBtnSend()**Bước 1: Khởi tạo gói tin (thông báo) cần gửi**

Vì số được lấy tăng dần, do đó cần khởi tạo biến đếm tăng sau mỗi lần nhấn nút.

Bước 2: Định dạng thông điệp

Các số được lấy theo thứ tự tự tăng từ 1, do đó sử dụng biến đếm số lượng và chuyển dạng thông báo "00X" với số <10 và "0XX" (với X là giá trị của biến đếm) để đảm bảo chuỗi là một chuỗi có ba chữ số, với các chữ số đầu tiên có thể là "001", ...

Bước 3: Gửi dữ liệu đến Client

```
foreach (Socket client in ListClient) ;
```

```
Send(client, message);
```

Bước 4: Hiển thị số lên màn hình

```
Hiển thị thông điệp lên màn hình: AddMesseger(message);
```

Hàm Receive():**Bước 1: Nhận dữ liệu**

Dữ liệu được nhận từ clientSocket và lưu vào mảng byte data. Số byte đã nhận được được trả về bởi clientSocket.Receive().

Bước 2: Chuyển đổi dữ liệu

Dữ liệu nhận được sau đó được chuyển đổi thành chuỗi bằng cách sử dụng phương thức Deserialize().

Bước 3: Kiểm tra loại thông báo

Kiểm tra xem thông báo có chứa chuỗi mà Client yêu cầu không. Nếu có, thông báo xóa tới tất cả các Client.

Bước 4: Xử lý thông báo

Thông báo được gửi đến tất cả các Client khác thông qua vòng lặp foreach. Nếu có yêu cầu Invoke, thì Invoke được sử dụng để thực hiện thao tác trên luồng chính của giao diện người dùng. Invoke được sử dụng để thực hiện một hoặc nhiều thao tác trên luồng chính (main thread) của ứng dụng. Trong môi trường đa luồng (multithreading), chỉ có luồng chính mới có quyền truy cập và thay đổi trực tiếp giao diện người dùng. Điều này có thể tạo ra vấn đề khi bạn muốn thay đổi giao diện từ một luồng khác, ví dụ như khi nhận dữ liệu từ mạng hoặc xử lý các sự kiện không đồng bộ.

Hàm AddClientNotification():**Bước 1: Thêm thông báo lên màn hình**

Sử dụng phương thức Items.Add() để hiển thị lên màn hình LED.

```
listBox2.Items.Add($"{notification}");
```

`\${notification}` là cú pháp nội suy chuỗi trong C#, là một cách ngắn gọn để nối chuỗi.

Bước 2: Chuyển văn bản thành giọng nói

```
SpeechSynthesizer sp = new SpeechSynthesizer();
```

SpeechSynthesizer đọc thông báo bằng cách sử dụng phương thức Speak().

3.2.3. Ứng dụng phía Client

Các chức năng chính phía Client được thể hiện trong bảng sau:

Bảng 3.2. Các chức năng chính phía Client

Phương thức và sự kiện	Mô tả
Connect()	Kết nối tới một Server thông qua một địa chỉ IP và cổng đã được xác định.

Phương thức và sự kiện	Mô tả
Send()	Gửi dữ liệu từ Client đến Server thông qua kết nối Socket đã được thiết lập sau khi thông điệp được mã hóa thành mảng byte.
Receive()	Liên tục nhận dữ liệu từ Server, giải mã và xử lý thông điệp nhận được. Nếu thông điệp là yêu cầu xóa số ("mời khách số"), Server sẽ thông báo cho các Client khác. Nếu không, nó sẽ hiển thị thông điệp trên giao diện người dùng.
Serialize()	Quá trình đóng gói (packing) hoặc mã hóa (encoding), chuyển đổi chuỗi thành một mảng byte trước khi gửi đi.
Deserialize()	Giải mã một mảng byte thành một đối tượng chuỗi.
AddMesseger()	Thêm các số thứ tự vào danh sách (màn hình) chờ. Các số thứ tự này do Server (máy lấy số của khách hàng gửi đến).
DNS.GetHostName()	Tự động lấy tên máy tính trong trường hợp không cấu hình tên cụ thể.
lollipopBtnSend_Click	Xử lý sự kiện khi nhân viên click nút để xử lý số thứ tự, lấy và xử lý phần tử đầu tiên từ hàng đợi, sau đó gửi yêu cầu xóa đến Server, xóa phần tử đã xử lý khỏi hàng đợi và cập nhật giá trị số thứ tự và thời gian bắt đầu xử lý lên cho khách hàng quan sát.
SendDeleteRequestToServer()	Tạo một thông điệp yêu cầu xóa dưới dạng chuỗi và sau đó gửi đến Server thông qua một kết nối Socket (Client) nhằm thông báo cho Server về yêu cầu xóa một số cụ thể từ nhân viên các quầy, cửa, phòng.
NotifyOtherClientsToDelete()	Thông báo cho các Client khác về yêu cầu xóa số cụ thể. Sau đó, cập nhật danh sách bằng cách xóa số đã bị xóa khỏi hàng đợi.

Hàm Connect:

Bước 1: Khởi tạo đối tượng IPEndPoint để xác định địa chỉ IP và cổng của Server

```
IP = new IPEndPoint(IPAddress.Parse(IPaddress), 7421);
```

Địa chỉ IP được chuyển đổi từ một chuỗi thành một đối tượng IPAddress thông qua IPAddress.Parse. Trong trường hợp này, cổng là 7421.

Bước 2: Khởi tạo đối tượng Socket để thiết lập kết nối với Server:

```
Client = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
```

Đối tượng này được thiết lập để sử dụng giao thức TCP và truyền dữ liệu theo kiểu luồng (stream).

Bước 3: Kết nối tới Server thông qua địa chỉ IP và cổng đã xác định.

```
Client.Connect(IP);
```

Bước 4: Khởi tạo một luồng mới để lắng nghe dữ liệu từ Server

```
Thread listen = new Thread(Receive);
```

Thiết lập luồng listen là luồng nền, nghĩa là nó sẽ tự động kết thúc khi chương trình chính (hoặc luồng chính) kết thúc.

```
listen.IsBackground = true;
```

Bước 5: Lắng nghe dữ liệu từ Server

```
listen.Start();
```

Khởi động luồng listen, bắt đầu quá trình.

Sự kiện xử lý số lollipopBtnSend_Click():

Bước 1: Lấy phần tử đầu tiên trong hàng đợi và chuyển đổi thành chuỗi

```
string selectedMessage = listBox1.Items[0].ToString();
```

Bước 2: Gửi yêu cầu xóa đến Server để hiện lên LED và phát ra loa

```
SendDeleteRequestToServer(selectedMessage);
```

Hàm SendDeleteRequestToServer và truyền vào yêu cầu (selectedMessage).

Bước 3: Xử lý số (xóa phần tử đầu tiên khỏi hàng đợi)

```
listBox1.Items.RemoveAt(0);
```

Bước 4: Hiển thị số và thời gian xử lý của số thứ tự được xử lý cho khách hàng tại quầy, cửa, phòng thực hiện xử lý.

```
txtSoXL.Text = selectedMessage;  
txtTG.Text = DateTime.Now.ToString();
```

Hàm SendDeleteRequestToServer():

Bước 1: Tạo thông điệp chứa số thứ tự được xử lý và Client là người (cửa, quầy, phòng) xử lý số đó.

```
string deleteRequest = $"{{User}} mời khách số: {{deletedNumber}}";
```

Tạo một chuỗi deleteRequest bằng cách kết hợp giá trị của biến User với chuỗi " mời khách số: " và số đã bị xóa (deletedNumber). Điều này tạo ra một thông điệp có dạng "Tên người dùng mời khách số: X", trong đó X là số đã bị xóa.

Bước 2: Gửi yêu cầu xóa đến Server

```
Client.Send(Serialize(deleteRequest));
```

Sử dụng đối tượng Socket (Client) để gửi yêu cầu xóa (deleteRequest) đến Server. Hàm Serialize() được sử dụng để chuyển đổi chuỗi thành một mảng byte trước khi gửi đi.

Hàm Receive():

Bước 1: Tạo mảng byte để chứa dữ liệu nhận được từ Server

```
byte[] data = new byte[1024 * 5000];
```

Bước 2: Nhận dữ liệu từ Server và lưu trữ vào mảng byte

```
Client.Receive(data);
```

Sử dụng phương thức Receive của đối tượng Socket (Client) để nhận dữ liệu từ Server và lưu trữ vào mảng byte data.

Bước 3: Giải mã dữ liệu từ mảng byte thành một chuỗi

```
string s = (string)Deserialize(data);
```

Bước 4: Thực hiện trên luồng chính (đối với các thay đổi trên giao diện người dùng) thông báo cho các Client khác về yêu cầu xóa số.

Kiểm tra xem chuỗi s có chứa "mời khách số" không. Nếu đúng, thực hiện một hành động.

```
if (s.Contains("mời khách số"))  
    Invoke(new Action(() => NotifyOtherClientsToDelete(s)));
```

Nếu chuỗi s chứa "mời khách số", sử dụng Invoke để thực hiện hàm NotifyOtherClientsToDelete trên luồng chính (đối với các thay đổi trên giao diện người dùng). Hàm này thông báo cho các Client khác về yêu cầu xóa số.

Hàm NotifyOtherClientsToDelete():

Bước 1: Tách chuỗi thành một mảng các phần tử

```
string[] parts = deleteRequest.Split(new[] { "mời khách số: " },  
StringSplitOptions.RemoveEmptyEntries);
```

Sử dụng phương thức Split để tách chuỗi deleteRequest thành một mảng các phần tử, với "mời khách số:" là dấu phân cách. Phương thức này sẽ trả về một mảng các chuỗi con trong chuỗi deleteRequest.

Nếu chuỗi deleteRequest có dạng "Tên người dùng mời khách số: X", thì mảng parts sẽ có độ dài là 2, với parts[0] là phần "Tên người dùng", và parts[1] là phần "X" (số đã bị xóa).

Gán giá trị của phần tử thứ hai trong mảng parts (phần "X") vào biến deletedNumber.

```
string deletedNumber = parts[1];
```

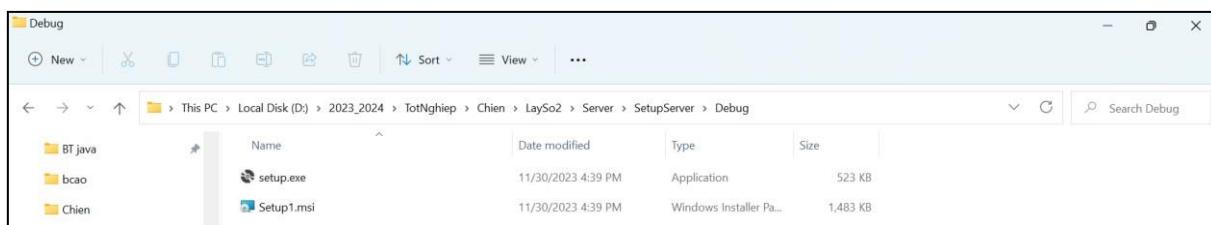
Bước 2: Cập nhật giao diện người dùng và hiển thị rằng đã bị xóa lên màn hình tại quầy, cửa, phòng xử lý số thứ tự.

listBox1.Items.Remove(deletedNumber); Xóa phần tử có giá trị là deletedNumber khỏi danh sách listBox1.

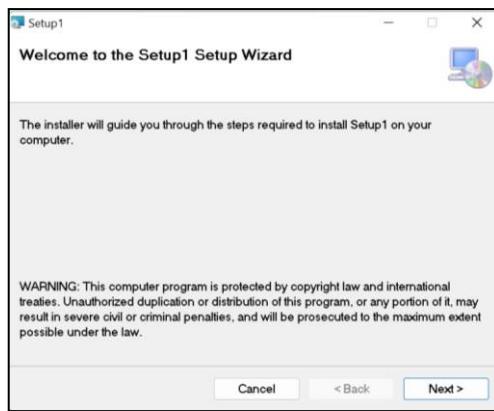
3.3. Kết quả thực nghiệm và đánh giá

3.3.1. Kết quả thực nghiệm

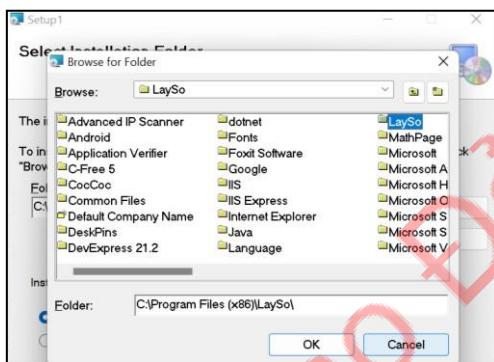
Ứng dụng gồm 2 chương trình chạy trên các máy tính khác nhau tương ứng chức lấy số dành cho khách hàng là bệnh nhân cần lấy số thứ tự và chức năng xử lý số thứ tự đồng bộ trên các máy của nhân viên xử lý số thứ tự đó. Ứng dụng đã được cài đặt thành công và chạy khá ổn định trên mạng LAN, Internet wifi gồm một Server đại diện cho máy lấy số của bệnh nhân tại địa chỉ 172.19.9.1, một Client số 1 có địa chỉ 172.19.9.2 đại diện cho một cửa, quầy, phòng xử lý số tự động mà bệnh nhân đã lấy tại máy lấy số tự động, một Client số 2 có địa chỉ 172.19.9.3 đại diện cho một cửa, quầy, phòng xử lý số tự động khác mà bệnh nhân đã lấy tại máy lấy số tự động. Chương trình phía Server được thiết kế với giao diện đơn giản, phù hợp với mọi đối tượng khách hàng là bệnh nhân.



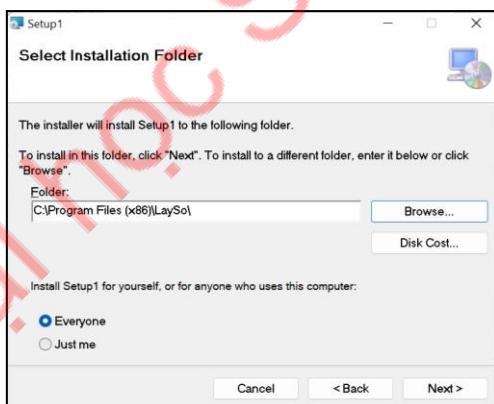
Hình 3.3. Cài đặt chương trình Server



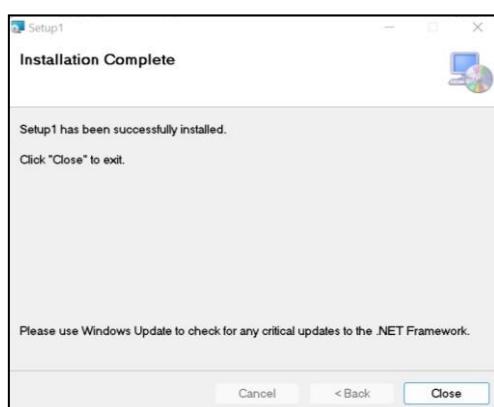
Hình 3.4. Cài đặt chương trình Server bước 2



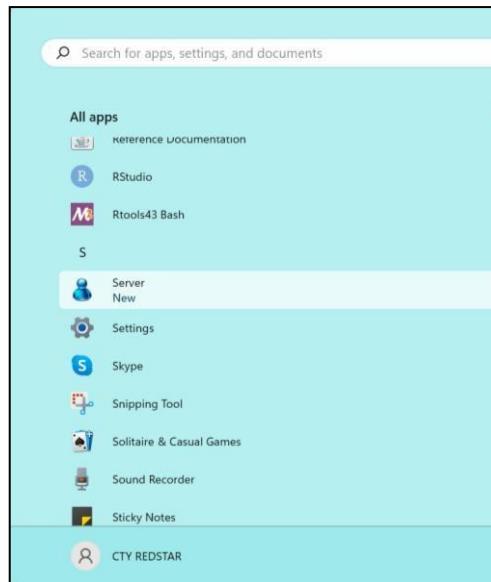
Hình 3.5. Cài đặt chương trình Server bước 3



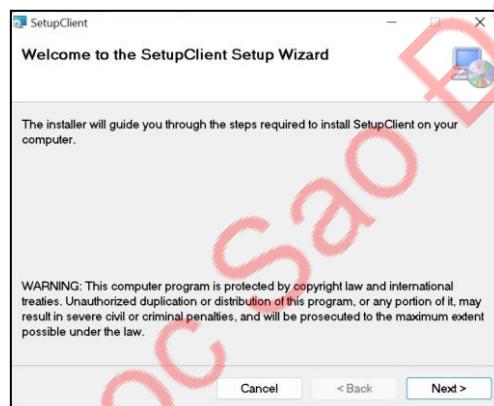
Hình 3.6. Cài đặt chương trình Server bước 4



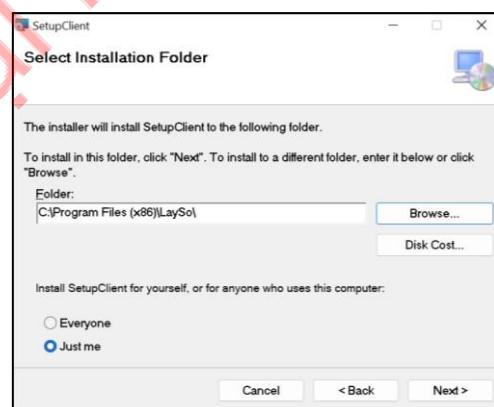
Hình 3.7. Cài đặt chương trình Server bước 5



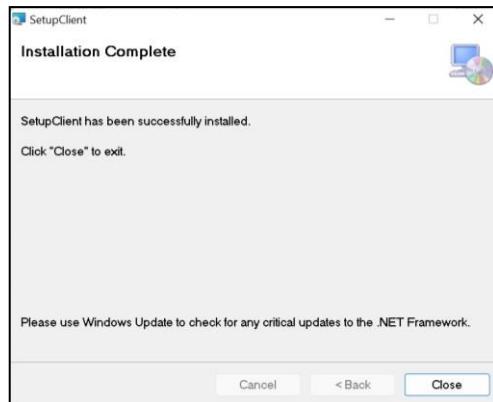
Hình 3.8. Cài đặt chương trình Server bước 6



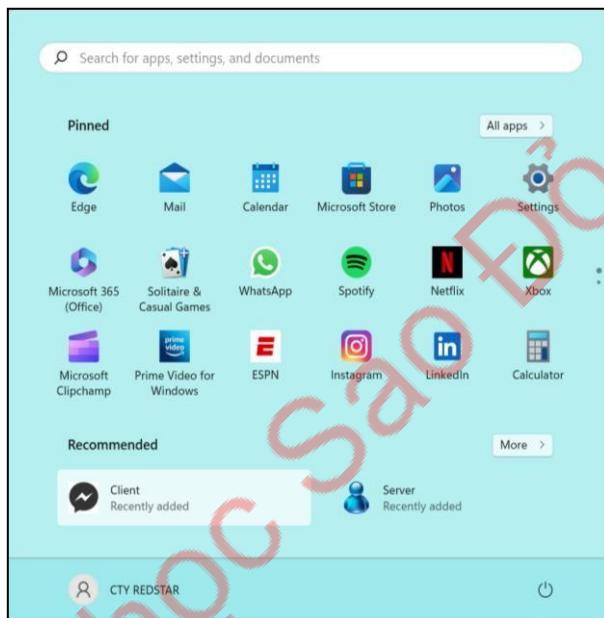
Hình 3.9. Cài đặt chương trình Client bước 1



Hình 3.10. Cài đặt chương trình Client bước 2



Hình 3.11. Cài đặt chương trình Client bước 3



Hình 3.12. Cài đặt chương trình Client bước 4

Trường hợp thực thi trên mạng Internet wifi, giao diện chính bắt đầu khởi động như sau:



Hình 3.13. Giao diện khởi động chương trình

Khi có khách hàng đến yêu cầu lấy số cần chọn chức năng lấy số thông qua hình thức bấm nút nhấn. Khi đó, số thứ tự được sinh ra tự động tương ứng lần lượt số từ thấp đến cao cho lần bấm số trước hoặc sau. Giao diện kết quả lấy số như sau:



Hình 3.14. Giao diện kết quả lấy số thứ tự tự động

Sau khi khách hàng lấy số tự động tại máy lấy số Server, các số này được truyền đồng bộ về các máy Client để xử lý tại quầy của nhân viên. Giao diện Client dễ sử dụng, hiển thị chức năng gọi số khách hàng thông qua hình thức bấm nút của nhân viên. Ngoài ra, giao diện hiển thị số hiện tại đang được xử lý lên cho khách hàng nhận biết cùng thời gian thực xử lý. Trường hợp 1, các máy Client được lập trình lấy tên quầy, cửa, phòng là tên máy tính của hệ thống (User = Dns.GetHostName()) với mục đích Client có thể tự động lấy tên máy tính thực thi khi triển khai trên các máy tính nhau.



Hình 3.15. Giao diện khởi động của Client 1



Hình 3.16. Giao diện khởi động của Client 2

Tại giao diện xử lý số tự động, nhân viên thực hiện gọi khách hàng thông qua nhấn nút “Tiếp theo” thì số thứ tự đầu tiên (lần nhấn lấy số sớm nhất) trong hàng đợi ở tất cả các máy Client được xử lý trước (tức là số thứ tự này đồng thời không còn hiển thị trên các hàng đợi của các máy xử lý khác), đồng thời hiển thị thông tin này trên mỗi cửa để khách hàng quan sát.



Hình 3.17. Giao diện Client 1 xử lý số 01

Lúc này, tại giao diện máy lấy số Server hiển thị thông tin quầy nào xử lý số thứ tự nào đồng thời phát ra loa cho khách hàng theo dõi. Trong giao diện này là kết quả máy (quầy, cửa, phòng) có tên theo HostName là DESKTOP-FGR7HP6 xử lý khách hàng số 01:



Hình 3.18. Giao diện gọi khách hàng 01 đến cửa, quầy, phòng tương ứng

Quá trình này được lặp lại ở cả các máy lấy số Server, máy xử lý Client. Trong giao diện này, Client 2 xử lý hai số liên tiếp 02, 03 và thông tin cũng hiển thị như ở Client 1:



Hình 3.19. Giao diện Client 2 xử lý số 02



Hình 3.20. Giao diện Client 2 xử lý số 03

Thông tin xử lý một lần nữa lại được truyền về máy Server để hiển thị trên màn hình LED hoặc loa.

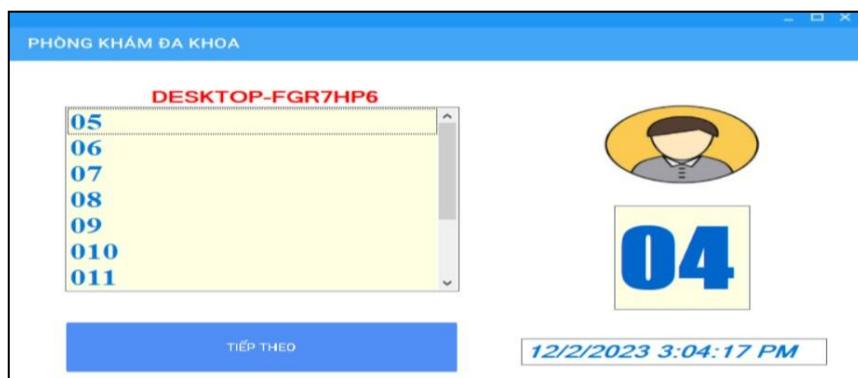


Hình 3.21. Giao diện gọi khách hàng 02 đến cửa, quầy, phòng tương ứng



Hình 3.22. Giao diện gọi khách hàng 03 đến cửa, quầy, phòng tương ứng

Tương tự, Client 1 tiếp tục xử lý khách hàng số 04, số này được loại ra khỏi hàng đợi của tất cả các máy Client tại các quầy và hiển thị thông báo lên Server cho khách hàng.



Hình 3.23. Giao diện Client 1 xử lý số 04



Hình 3.24. Giao diện gọi khách hàng 04 đến cửa, quầy, phòng tương ứng

Trường hợp 2, trên mỗi máy xử lý tại các quầy, cửa, phòng của Client được cấu hình sẵn tên từng quầy, cửa, phòng (ví dụ cửa số 1, số 2; phòng số 1, số 2) thì thông tin này được đồng bộ trên các máy tương tự trường hợp 1 nhưng thay vì mặc định tên máy thì nhận diện tên từng quầy, cửa, phòng đó.



Hình 3.25. Giao diện khởi động Client 1



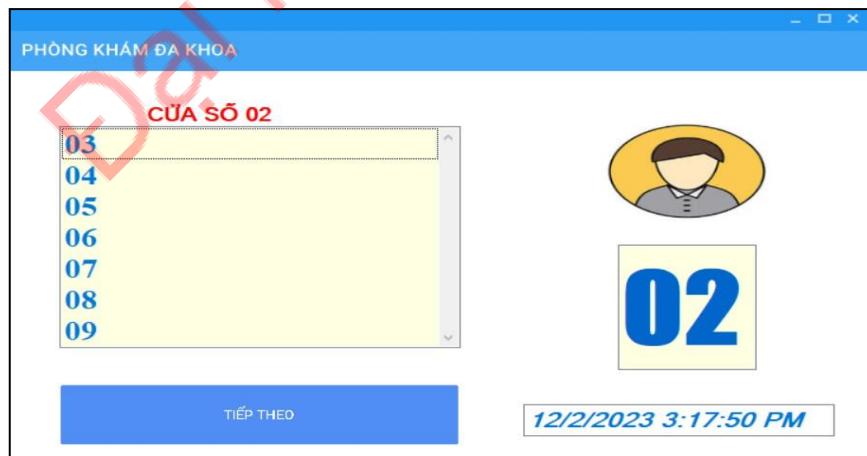
Hình 3.26. Giao diện khởi động Client 2



Hình 3.27. Giao diện Client 1 xử lý số 01



Hình 3.28. Giao diện gọi khách hàng 01 đến cửa, quầy, phòng tương ứng



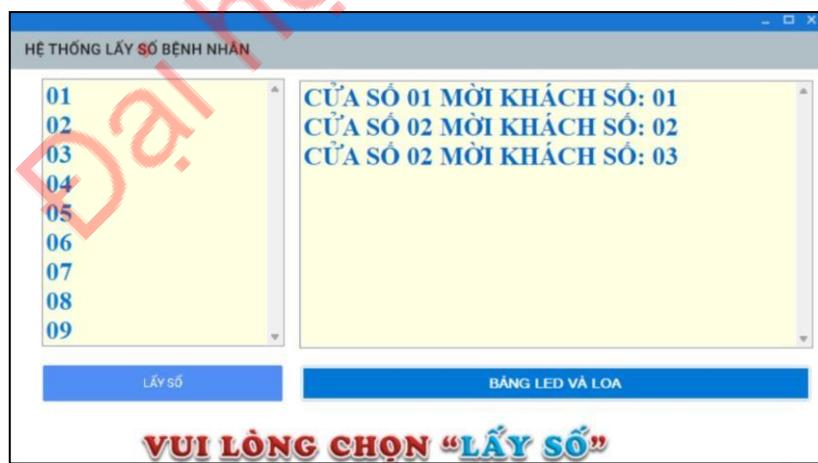
Hình 3.29. Giao diện Client 2 xử lý số 02



Hình 3.30. Giao diện gọi khách hàng 02 đến cửa, quầy, phòng tương ứng



Hình 3.31. Giao diện Client 2 xử lý số 03



Hình 3.32. Giao diện gọi khách hàng 03 đến cửa, quầy, phòng tương ứng

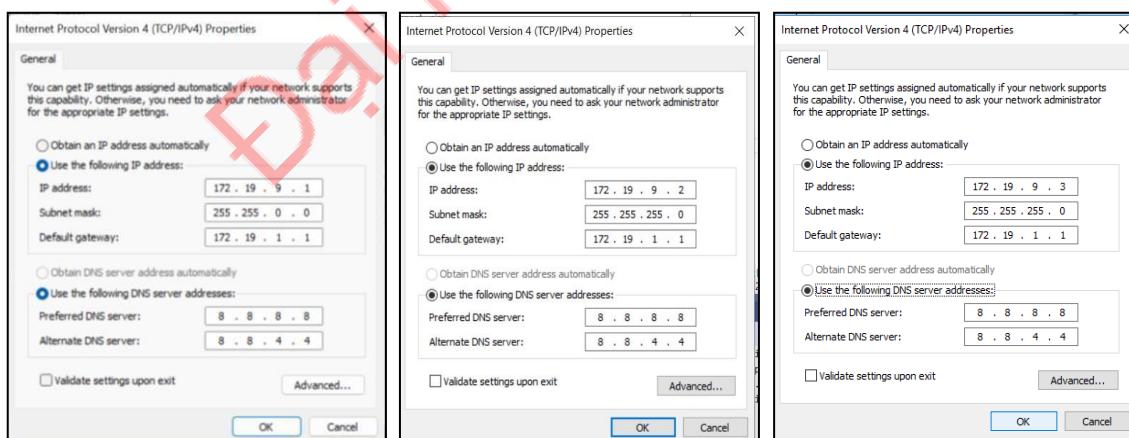


Hình 3.33. Giao diện Client 1 xử lý số 04

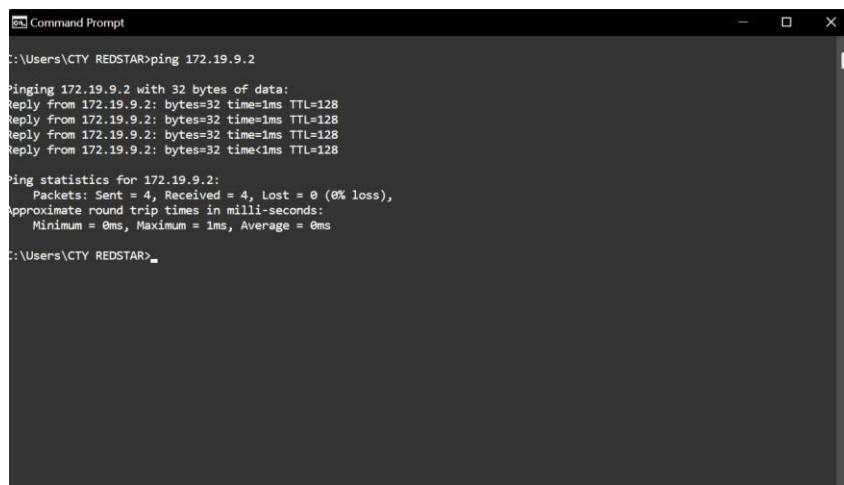


Hình 3.34. Giao diện gọi khách hàng 04 đến cửa, quầy, phòng tương ứng

Trường hợp thực thi trên mạng LAN với địa chỉ của các máy như sau:



Hình 3.35. Cấu hình địa chỉ IP cho (a) Server, (b) Client 1, (c) Client 2



Hình 3.36. Kiểm tra các máy thông mạng



Hình 3.37. Giao diện khởi động Server



Hình 3.38. Giao diện Client 1 xử lý số 01



Hình 3.39. Giao diện Client 1 xử lý số 02



Hình 3.40. Giao diện Client 2 xử lý số 03



Hình 3.41. Giao diện Client 2 xử lý số 04



Hình 3.42. Giao diện Client 2 xử lý số 05



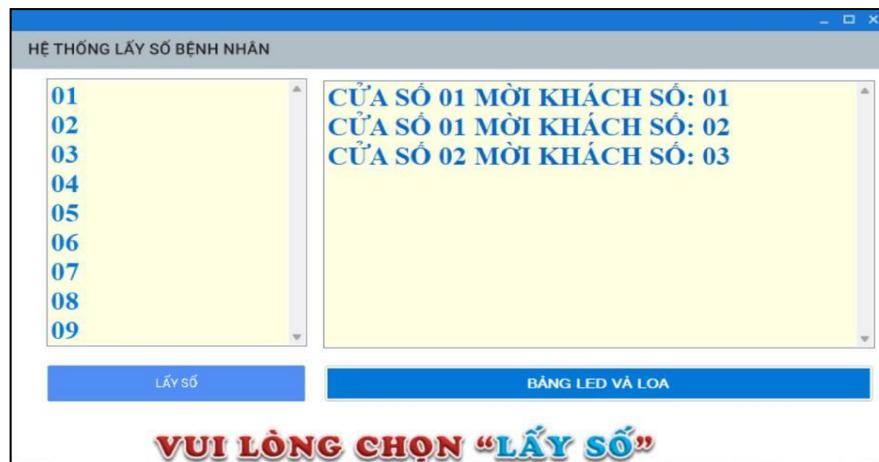
Hình 3.43. Giao diện Client 2 xử lý số 06



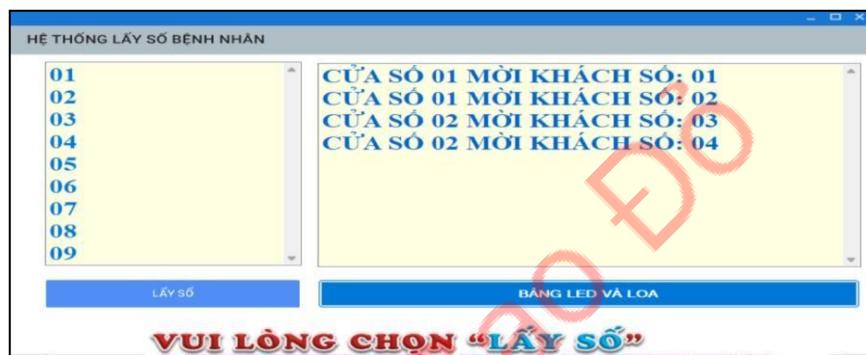
Hình 3.44. Giao diện gọi khách hàng 01 đến cửa, quầy, phòng tương ứng



Hình 3.45. Giao diện gọi khách hàng 02 đến cửa, quầy, phòng tương ứng



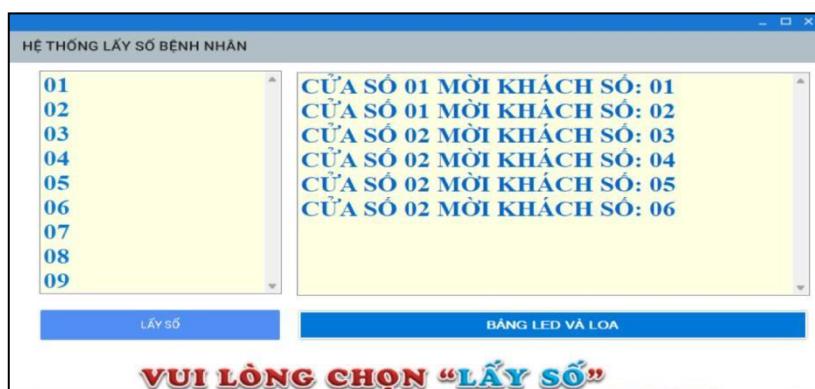
Hình 3.46. Giao diện gọi khách hàng 03 đến cửa, quầy, phòng tương ứng



Hình 3.47. Giao diện gọi khách hàng 04 đến cửa, quầy, phòng tương ứng



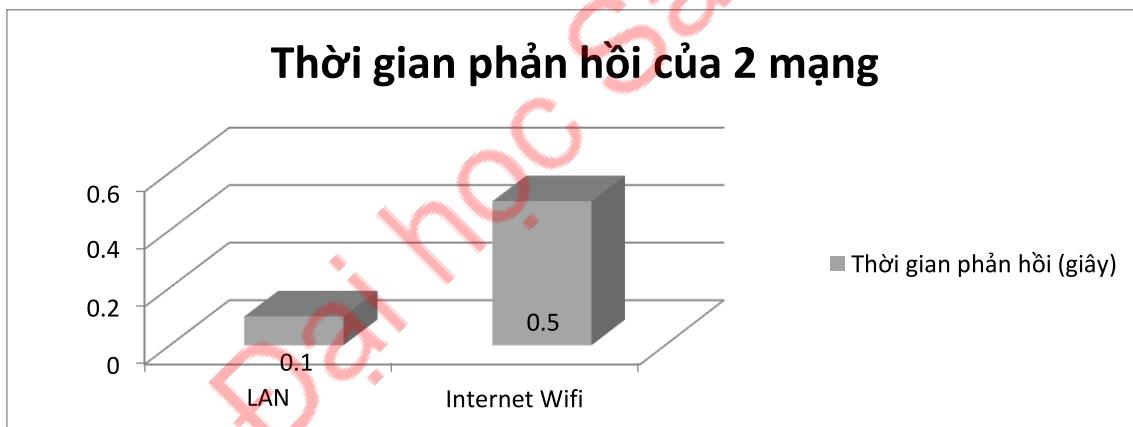
Hình 3.48. Giao diện gọi khách hàng 05 đến cửa, quầy, phòng tương ứng



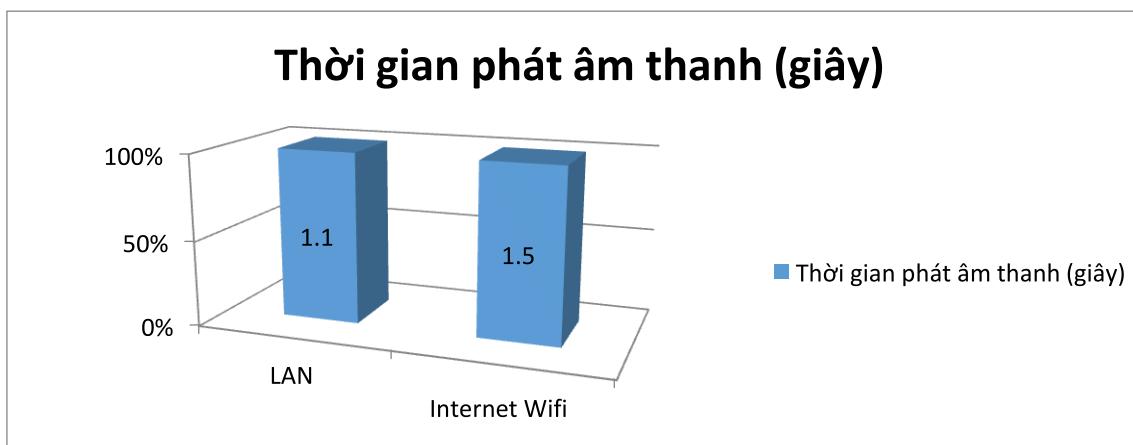
Hình 3.49. Giao diện gọi khách hàng 06 đến cửa, quầy, phòng tương ứng

3.3.2. Đánh giá dựa trên kết quả thực nghiệm

Hệ thống được xây dựng theo mô hình Client – Server nên có thể chịu tải được với hàng nghìn user cùng vào mạng tại cùng một thời điểm (được kiểm nghiệm bằng tool tự sinh các Client khoảng 1500 client cùng xử lý một lúc) và đánh giá tình trạng sử dụng tài nguyên hệ thống trên Windows Task Manager về phần trăm CPU, bộ nhớ sử dụng, lượt thao tác Handle và luồng thực thi Thread, mặc dù trong thực tế bệnh viện, ngân hàng, cơ quan hành chính công thì số lượng quầy, cửa, phòng để xử lý một thời điểm lên đến hàng nghìn ít khi xảy ra. Tốc độ phản hồi từ phía Server về các Client và ngược lại trong cả trường hợp mạng Internet và LAN đều đảm bảo, mặc dù Server để thực nghiệm trong trường hợp này là Laptop bình thường, không phải Server chuyên dụng. Các phản hồi từ các máy Server và Client là tức thì (0.1 giây cho mỗi thao tác) ngoại trừ thao tác đọc số trên loa do chờ thời gian đọc xong câu thông báo (1 giây) tùy thuộc độ dài ngắn lời gọi. Trong thực nghiệm gồm: “Tên quầy, cửa, phòng” + “Mời khách số” + “Số thứ tự”. Với mạng Internet wifi thì ít nhiều ảnh hưởng bởi chất lượng đường truyền và lượng người truy cập. Tuy nhiên, tại một thời điểm chỉ có một khách hàng vào bấm số nên tình trạng nghẽn là không xảy ra. Với quầy, cửa, phòng của nhân viên xử lý, tại một thời điểm xử lý một khách hàng theo nguyên tắc lần lượt của hàng đợi là vào trước xử lý trước nên hiện tượng tắc nghẽn trên cũng không bao giờ xảy ra.



Hình 3.50. Biểu đồ thể hiện tốc độ phản hồi



Hình 3.51. Biểu đồ thể hiện tốc độ phát âm thanh

3.4. Kết luận chương 3

Chương 3 trình bày việc lựa chọn công cụ C#.Net, mô hình Client – Server TCP Socket, xây dựng kiến trúc mô hình tổng thể, ứng dụng phía Server và phía Client cũng như cách thức triển khai, đánh giá trên mạng LAN và Internet về thời gian truyền nhận, hiệu suất và khả năng chịu tải của ứng dụng. Việc sử dụng C#.Net đảm bảo tính linh hoạt, khả năng tích hợp và hỗ trợ đầy đủ từ cộng đồng phát triển. Mô hình Client – Server TCP Socket được chọn vì khả năng truyền thông tin đồng thời, đồng bộ, không bị mất tin và truyền có thứ tự do cơ chế truyền lại gói tin đã mất giúp tối ưu hóa quá trình giao tiếp giữa các thành phần của hệ thống.

Quá trình xây dựng kiến trúc mô hình tổng thể đã đảm bảo sự liên kết chặt chẽ giữa phía Server và phía Client, tạo nên một hệ thống ổn định và dễ bảo trì. Ứng dụng phía Server được thiết kế để xử lý các yêu cầu từ phía Client một cách hiệu quả, đồng thời đảm bảo an toàn và bảo mật thông tin. Việc triển khai, đánh giá trên cả mạng LAN và Internet đã cung cấp cái nhìn tổng quan về hiệu suất của ứng dụng. Thông qua quá trình này, có thể đánh giá được thời gian phản hồi, độ ổn định và khả năng chịu tải của hệ thống. Đánh giá này sẽ là cơ sở để tối ưu hóa và điều chỉnh các thành phần của hệ thống để đáp ứng yêu cầu và mong đợi từ người dùng.

KẾT LUẬN

1. Kết quả đạt được

Sau quá trình nghiên cứu, xây dựng đồ án Lập trình ứng dụng Client – Server lấy số thứ tự khám bệnh đã thực hiện đầy đủ các yêu cầu đặt ra của nhiệm vụ đồ án tốt nghiệp, cụ thể là:

- Nghiên cứu hoạt động của mô hình Client – Server và các giao thức giao tiếp.
- Nghiên cứu các công nghệ lập trình theo mô hình Client – Server.
- Nghiên cứu nghiệp vụ bài toán lấy số khám bệnh tại cơ sở y tế.
- Thiết kế giao diện, xử lý logic phía Client gửi yêu cầu lấy số khám bệnh; phía Server xử lý yêu cầu từ Client và cấp số thứ tự mới cho người dùng, quản lý danh sách các số thứ tự đang chờ xử lý và cập nhật danh sách khi một số thứ tự đã được xử lý.

Như vậy, thông qua ứng dụng này, khách hàng khi đến giao dịch tại các địa điểm công cộng như ngân hàng, trung tâm hành chính công, bệnh viện,... giúp tạo ra sự công bằng và minh bạch trong việc xếp hàng, đồng thời giúp tiết kiệm thời gian chờ đợi cho khách hàng. Với mục đích xếp hàng lấy số thứ tự, hướng đến một nền văn minh, lịch sự cho những nơi công cộng, dịch vụ chăm sóc khách hàng được tốt hơn. Hệ thống chỉ cần cắm điện và kết nối dây mạng, không cần lắp ráp gì thêm vì các thiết bị sử dụng toàn bộ tài nguyên mạng LAN có sẵn. Việc kết nối qua LAN này cũng rất thuận lợi để lắp đặt, bảo trì hay hiệu chỉnh ngay cả với người không cần nhiều chuyên môn. Nếu sử dụng hệ thống Internet, ứng dụng cũng hoạt động ổn định, đảm bảo tài nguyên và khả năng chịu tải cũng như tốc độ xử lý.

2. Hạn chế

Để tăng tính ứng dụng đồ án cần được thiết kế dạng hệ thống IoT hoàn chỉnh như hệ thống in số, chế độ ưu tiên và chọn dịch vụ tại máy lấy số, lưu trữ và thống kê lịch sử phục vụ để có hướng điều chỉnh và báo cáo khi cần.

3. Kiến nghị

Đề tài thực hiện phân tích, cài đặt và đánh giá ứng dụng thành công. Kết quả và giải pháp của đề tài có thể ứng dụng vào nghiệp vụ lấy và xử lý số tự động tại các cơ quan hành chính công. Tuy nhiên, cần có một hệ thống đồng bộ cần thiết kế dạng hệ thống IoT hoàn chỉnh với Server chuyên dụng để phục vụ tối ưu vấn đề này.

TÀI LIỆU THAM KHẢO

- [1] Đại học Sao Đỏ (2022), Giáo trình Lập trình mạng.
- [2] Michael J. Donahoo and Kenneth L. Calvert (2017), *Distributed Systems: Principles and Paradigms*, ISBN-10: 1543057381.
- [3] Michael J. Donahoo, Kenneth L. Calvert (2009), *TCP/IP Sockets in C, 2nd Edition*. ISBN: 9780080923215.

Đại học Sao Đỏ