**ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**TRƯỜNG ĐIỆN - ĐIỆN TỬ**



**BÁO CÁO BÀI TẬP LỚN**

**MẠNG MÁY TÍNH VÀ IOT**

**ĐỀ TÀI:**

**LẬP TRÌNH SOCKET VỚI ỨNG DỤNG VIDEO STREAMING**

**Mã lớp : 131588**

**Nhóm sinh viên thực hiện:**

|  |  |
| --- | --- |
| Thành viên | MSSV |
| 1. Trần Thị Thảo Anh | 20193287 |
| 1. Nguyễn Đình Phương | 20193314 |
| 1. Phạm Đức Anh | 20193286 |

**Giảng viên hướng dẫn**: PGS.TS Nguyễn Tài Hưng

**Hà Nội, 7/2022**

LỜI MỞ ĐẦU

Hiện nay mạng máy tính đã và đang phát triển rất mạnh, đáp ứng rất tốt các nhu cầu về thông tin khoa học, giải trí, liên lạc, mua bán, quảng cáo v.v... của con người. Nó đã trở thành một thành phần không thể thiếu với cuộc sống hiện đại. Internet là một hệ thống **hệ thống chia sẻ thông tin toàn cầu** liên kết các máy tính trên toàn thế giới lại theo những chuẩn chung. Trong phạm vi nhỏ hơn chúng ta có những mô hình mạng khác như là WAN, LAN.., mỗi mô hình đó đều có những ứng dụng trên nhiều tiện ích khác nhau . Ngày nay nhu cầu giao tiếp và truyền thông là cần thiết hơn bao giờ hết. Có nhiều cách để truyền đạt thông tin từ người này sang người khác: trực tiếp, thoại, video, thư từ… Để truyền thông tin một cách hiệu quả và nhanh chóng, việc truyền video stream đã dần trở thành một công cụ hữu ích phục vụ nhu cầu giao tiếp của con người.

Mục tiêu của bài tập lớn là tìm hiểu về mạng máy tính và kĩ thật lập trình Socket với ngôn ngữ lập trình là Python, trên cơ sở đó xây dựng ứng dụng truyền video, nên chúng em đã quyết định chọn đề tài ” Lập trình Socket với ứng dụng Video Streaming ”. Bài tập lớn sẽ xây dựng thử nghiệm một hệ thống cho phép nhắn tin trao đổi dữ liệu trong mô hình Client/Server, chúng em đã thực hiện chương trình stream video giữa Client và Server, mã hóa trước khi truyền.

Chúng em xin cảm ơn thầy PGS. TS. Nguyễn Tài Hưng đã chỉ dạy cho chúng em những kiến thức trong bộ môn “ Mạng máy tính và IoT” để chúng em có thể hoàn thành bài tập này. Rất mong có được những sự góp ý của thầy để chúng em có thể sửa đổi và rút ra kinh nghiệm cho bản thân.

**MỤC LỤC**

[DANH MỤC HÌNH VẼ 4](#_Toc109340106)

[DANH MỤC BẢNG BIỂU 5](#_Toc109340107)

[CHƯƠNG 1. TỔNG QUAN 6](#_Toc109340108)

[1.1 . Đặt vấn đề 6](#_Toc109340109)

[1.2 . Cơ sở lập trình 6](#_Toc109340110)

[1.3 . Phân công công việc 6](#_Toc109340111)

[CHƯƠNG 2. CƠ SỞ LÝ THUYẾT 7](#_Toc109340112)

[2.1 . Mô hình Client – Server 7](#_Toc109340113)

[2.1.1 Nguyên tắc hoạt động 8](#_Toc109340114)

[2.1.2 Ưu nhược điểm 8](#_Toc109340115)

[2.2 Giao thức TCP/IP 10](#_Toc109340116)

[2.2.1 Phương thức hoạt động 12](#_Toc109340117)

[2.2.2 Giới thiệu về IP 13](#_Toc109340118)

[2.2.3 Giới thiệu về Port 14](#_Toc109340119)

[2.3 Giới thiệu về Socket 16](#_Toc109340120)

[2.3.1 Phân loại Socket 16](#_Toc109340121)

[CHƯƠNG 3. TRIỂN KHAI 20](#_Toc109340122)

[3.1 . Mô tả 20](#_Toc109340123)

[3.1.1 Công việc phải làm 20](#_Toc109340124)

[3.2 . Ngôn ngữ và thư viện sử dụng 20](#_Toc109340125)

[3.3 . Sơ đồ khối 21](#_Toc109340126)

[3.3.1 Khung data 21](#_Toc109340127)

[3.3.2 Server 21](#_Toc109340128)

[3.3.3 Client 22](#_Toc109340129)

[3.4 . Code 22](#_Toc109340130)

[3.4.1 Server 22](#_Toc109340131)

[3.4.2 Client 25](#_Toc109340132)

[3.5 . Demo 26](#_Toc109340133)

[3.5.1 Khởi tạo Server 26](#_Toc109340134)

[3.5.2 Sau khi kết nối Client với Server 26](#_Toc109340135)

[KẾT LUẬN 28](#_Toc109340136)

# DANH MỤC HÌNH VẼ

[Hình 2.1 Mô hình Client – Server 7](#_Toc109339701)

[Hình 2.2 Các tầng trong giao thức TCP/IP 11](#_Toc109339702)

[Hình 2.3 Quá trình đóng mở gói dữ liệu trong TCP/IP 12](#_Toc109339703)

[Hình 2.4 Cấu trúc dữ liệu trong TCP/IP 13](#_Toc109339704)

[Hình 2.5 Stream Socket 17](#_Toc109339705)

[Hình 2.6 Datagram Socket 19](#_Toc109339706)

[Hình 3.1 Khung data được đóng gói 21](#_Toc109339707)

[Hình 3.2 Sơ đồ khối của Server 22](#_Toc109339708)

[Hình 3.3 Sơ đồ khối của Client 23](#_Toc109339709)

[Hình 3.4 Sau khi khởi tạo Socket ở chương trình Server 27](#_Toc109339710)

[Hình 3.5 Các bit kiểm tra được in trên màn hình 28](#_Toc109339711)

[Hình 3.6 Video được truyền 29](#_Toc109339712)

# DANH MỤC BẢNG BIỂU

[Bảng 1 Phân công công việc 6](#_Toc109339745)

[Bảng 2 Thư viện sử dụng 20](#_Toc109339746)

# TỔNG QUAN

## . Đặt vấn đề

Trên thực tế, Video Stream là một ứng dụng mạng đã được phổ biến rộng rãi. Một số ứng dụng như FaceTime, Zalo, Messenger, … Người ta sử dụng các chương trình này nhằm trao đổi thông tin, bàn bạc với nhau thông qua môi trường Internet – một cách đơn giản để kết nối mọi người với nhau. Khi xây dựng chương trình này, việc đảm bảo về các yêu cầu về kết nối dữ liệu trong quá trình truyền đi cần phải đảm bảo toàn vẹn và an toàn.

## . Cơ sở lập trình

Socket là giao diện lập trình ứng dụng mạng được dùng để truyền và nhận dữ liệu trên internet. Giữa hai chương trình chạy trên mạng cần có một liên kết giao tiếp hai chiều (two-way communication) để kết nối 2 process với nhau. Điểm cuối (endpoint) của liên kết này được gọi là socket. Một chức năng khác của socket là giúp các tầng TCP hoặc TCP Layer định danh ứng dụng mà dữ liệu sẽ được gửi tới thông qua sự ràng buộc với một cổng port (thể hiện là một con số cụ thể), từ đó tiến hành kết nối giữa client và server.

Socket có thể hỗ trợ hầu hết các hệ điều hành bao gồm MS Windows, Linux, … Ngoài ra, socket cũng được sử dụng với nhiều ngôn ngữ lập trình, gồm C, C++, Java, Python, … Vì vậy, socket có thể tương thích với hầu hết mọi đối tượng người dùng với những cấu hình máy khác nhau.

## . Phân công công việc

Bảng 1 Phân công công việc

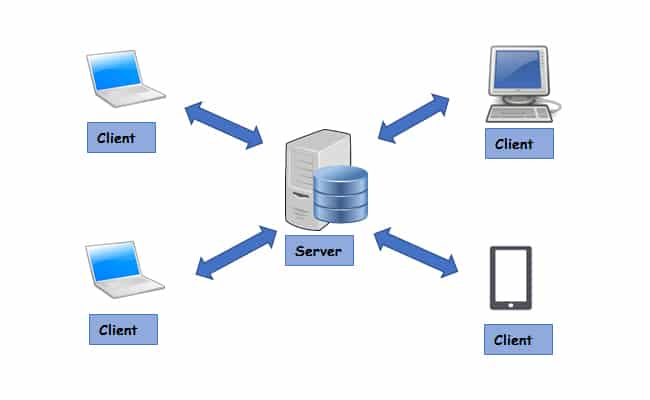
|  |  |
| --- | --- |
| **Nội dung** | **Thành viên** |
| Tìm hiểu tài liệu: Lý thuyết TCP/IP, socket | Thảo Anh, Đình Phương, Đức Anh |
| Thiết kế sơ đồ truyền nhận 1 gói | Thảo Anh, Đình Phương |
| Thiết kế khối chức năng | Đức Anh |
| Lập trình socket cơ bản và kiểm tra kết quả | Đức Anh, Đình Phương |
| Lập trình thêm các bit kiểm tra vào video | Thảo Anh |
| Hoàn thiện phần mềm | Phương |
| Viết báo cáo | Đức Anh |
| Thiết kế slide | Đình Phương |

# CƠ SỞ LÝ THUYẾT

Chương này sẽ nêu ra các cơ sở lý thuyết được áp dụng trong dự án này.

## . Mô hình Client – Server

***Client and Server*** - Mô hình khách – chủ, là một cấu trúc ứng dụng phân tán mà ở đó công việc được phân chia giữa bên cung cấp tài nguyên hoặc dịch vụ, được gọi là máy chủ, và bên yêu cầu dịch vụ, được gọi là máy khách. Thường thì các máy khách và máy chủ giao tiếp vói nhau thông qua mạng máy tính trên các phân cứng riêng biệt, nhưng cả máy khách và máy chủ cũng có thể nằm trên cùng một hệ thống. Một máy chủ nối chủ chạy một hoặc nhiều chương trình máy chủ, đồng thời chia sẻ tài nguyên với các máy khách. Một máy khách thường không chia sẽ bất kì tài nguyên nào của nó cả, mà chỉ yêu cầu nội dụng hoặc dịch vụ từ một máy chủ. Vì vậy, các máy khách sẽ thiết lập các phiên giao tiếp với máy chủ, và máy chủ sẽ chờ các yêu cầu đến .



Hình . Mô hình Client – Server

***Server*** – máy chủ, là một phần của máy tính phần cứng hoặc phần mềm (chương trình máy tính) mà cung cấp chức năng cho các chương trình khác hoặc các thiết bị, được gọi là "máy khách". Kiến trúc này được gọi là mô hình khách – chủ. Máy chủ có thể cung cấp các chức năng khác nhau, thường được gọi là "dịch vụ", chẳng hạn như chia sẻ dữ liệu hoặc tài nguyên giữa nhiều máy khách hoặc thực hiện tính toán cho một máy khách. Một máy chủ duy nhất có thể phục vụ nhiều máy khách và một máy khách có thể sử dụng nhiều máy chủ. Quy trình máy khách có thể chạy trên cùng một thiết bị hoặc có thể kết nối qua mạng với máy chủ trên một thiết bị khác.

***Client*** – máy khách, là nơi gửi yêu cầu đến máy chủ - server. Máy khách – client sẽ tổ chức giao tiếp với người dùng, máy chủ và môi trường bên ngoài tại trạm làm việc. Bên cạnh đó nó còn tiếp nhận yêu cầu của người dùng và lập những yêu cầu nhằm gửi đến máy chủ - server. Thể hiện các phản hồi nhận được từ máy chủ.

### Nguyên tắc hoạt động

Server chấp nhận tất cả các yêu cầu hợp lệ từ mọi nơi khác nhau trên Internet, sau đó trả kết quả về máy tính đã gửi yêu cầu đó.

Máy tính được coi là máy khách khi chúng làm nhiệm vụ gửi yêu cầu đến các máy chủ và đợi câu trả lời được gửi về.

Để máy khách và máy chủ có thể giao tiếp được với nhau thì giữa chúng phải có một chuẩn nhất định, và chuẩn đó được gọi là giao thức. Một số giao thức được sử dụng phổ biến hiện nay như: HTTPS, TCP/IP, FTP,...

Nếu máy khách muốn lấy được thông tin từ máy chủ, chúng phải tuân theo một giao thức mà máy chủ đó đưa ra. Nếu yêu cầu đó được chấp nhận thì máy chủ sẽ thu thập thông tin và trả về kết quả cho máy khách yêu cầu. Bởi vì Server - máy chủ luôn luôn trong trạng thái sẵn sàng để nhận request từ client nên chỉ cần client gửi yêu cầu tín hiệu và chấp nhận yêu cầu đó thì server sẽ trả kết quả về phía client trong thời gian ngắn nhất.

### Ưu nhược điểm

* ***Ưu điểm***

1. Tập trung

Ưu điểm chính của mô hình mạng khách chủ là khả năng kiểm soát tập trung (Centralization) được tích hợp sẵn. Với mô hình này, tất cả thông tin cần thiết đều được đặt ở một vị trí duy nhất. Việc này rất hữu ích cho những quản trị viên mạng. Vì họ có được toàn quyền quản lý và điều hành.

Bằng tính năng này, mọi sự cố trong mạng đều có thể được giải quyết ở một nơi duy nhất. Và do đó, việc cập nhật tài nguyên, dữ liệu cũng trở nên dễ dàng hơn.

2. Bảo mật

Trong mạng Client Server, dữ liệu được bảo vệ tốt do kiến trúc tập trung của mạng. Nó có thể được áp dụng các biện pháp kiểm soát truy cập, sao cho chỉ những người dùng được ủy quyền mới có thể truy cập.

Một trong những cách để làm vậy là áp đặt thông tin đăng nhập, chẳng hạn như Username hay Password. Hơn nữa, nếu dữ liệu bị mất, các file có thể được khôi phục dễ dàng từ chỉ một bản sao lưu duy nhất.

3. Khả năng mở rộng

Mô hình mạng Client server có khả năng mở rộng tốt. Bất cứ khi nào người dùng cần, họ có thể tăng số lượng tài nguyên, chẳng hạn như số Client hay Server. Do đó có thể dễ dàng tăng kích thước của Server mà không bị gián đoạn nhiều.

4. Khả năng truy cập

Không có sự phân biệt giữa các vị trí hay nền tảng khác nhau, mọi client đều có thể đăng nhập vào hệ thống. Bằng cách này, tất cả nhân viên đều có thể truy cập thông tin của công ty của họ, không cần phải dùng một terminal mode hay một bộ xử lý nào.

**Nhược điểm :**

1. Tắc nghẽn lưu lượng

Nhược điểm chính của mô hình mạng Client Server là tắc nghẽn lưu lượng. Nếu có quá nhiều Client tạo request từ cùng một Server, nó có thể làm chậm kết nối. Hoặc tệ hơn là dẫn đến hiện tượng crash. Một server bị quá tải có thể tạo ra nhiều vấn đề trong việc truy cập thông tin.

2. Độ bền

Như ta đã biết, mạng Client Server là mạng tập trung. Nên nếu Server chính xảy ra sự cố hay bị nhiễu, toàn bộ hệ thống mạng sẽ bị gián đoạn. Do đó, các mạng client server sẽ thiếu tính ổn định và độ bền.

3. Chi phí

Chi phí thiết lập và bảo trì server trong các mạng client server thường khá cao. Vì các hệ thống mạng có sức mạnh lớn có thể có giá rất đắt. Do đó, không phải tất cả người dùng đều có thể chi trả được.

4. Bảo trì

Khi các Server được triển khai, nó sẽ hoạt động không ngừng nghỉ. Có nghĩa là nó cần được quan tâm đúng mức nếu có bất kỳ vấn đề gì thì phải giải quyết ngay. Vì vậy, cần có một nhà quản lý mạng chuyên biệt để duy trì hoạt động của Server.

5. Tài nguyên

Không phải tất cả tài nguyên hiện có ở trên Server đều có thể sử dụng được. Ví dụ như bạn không thể in trực tiếp tài liệu trên web, hoặc chỉnh sửa bất kỳ thông tin nào trên ổ cứng của Client.

## Giao thức TCP/IP

Bộ giao thức TCP/IP (Internet Protocol Suite) là bộ giao thức mà Internet và các mạng máy tính đang sử dụng và chạy trên đó. Nó gồm 2 giao thức chính là TCP (Transmission Control Protocol - Giao thức điều khiển giao vận) và IP (Internet Protocol - Giao thức liên mạng).

Bộ giao thức TCP/IP được coi là một tập hợp các tầng. Mỗi tầng giải quyết một tập các vấn đề liên quan đến việc truyền tải dữ liệu, và cung cấp cho các giao thức tầng cấp trên một dịch vụ được định nghĩa rõ ràng dựa trên việc sử dụng các dịch vụ của các tầng thấp hơn. Hay nói cách khác, các tầng trên gần với người dùng hơn và làm việc với dữ liệu trừu tượng hơn, chúng dựa vào các tầng cấp dưới để biến đổi dữ liệu thành các dạng mà cuối cùng có thể truyền đi một cách vật lý.

Diagram

Description automatically generated

Hình . Các tầng trong giao thức TCP/IP

* **Tầng liên kết:**

Tầng liên kết (còn được gọi là tầng liên kết dữ liệu hay tầng giao tiếp mạng) là tầng thấp nhất trong mô hình TCP/IP, bao gồm các thiết bị giao tiếp mạng và các chương trình cung cấp các thông tin cần thiết để có thể hoạt động, truy nhập đường truyền vật lý qua các thiết bị giao tiếp mạng đó.

* **Tầng Internet:**

Tầng Internet ( hay còn gọi là tầng Mạng) xử lý quá trình truyền gói tin trên mạng, các giao thức của tầng này bao gồm : IP ( Internet Protocol) , ICMP ( Internet Control Message Protocol) , IGMP ( Internet Group Message Protocol ).

* **Tầng giao vận:**

Tầng giao vận phụ trách luồng dữ liệu giữa 2 trạm thực hiện các ứng dụng của tầng trên, tầng này có 2 giao thức chính là TCP ( Transmisson Control Protocol) và UDP ( User Datagram Protocol ) .

TCP cung cấp luồng dữ liệu tin cậy giữa 2 trạm, nó sử dụng các cơ chế như chia nhỏ các gói tin ở tầng trên thành các gói tin có kích thước thích hợp cho tầng mạng bên dưới, báo nhận gói tin, đặt hạn chế thời gian timeout để đảm bảo bên nhận biết được các gói tin đã gửi đi. Do tầng này đảm bảo tính tin cậy nên tầng trên sẽ không cần quan tâm đến nữa .

UDP cung cấp một dịch vụ rất đơn giản hơn cho tầng ứng dụng . Nó chỉ gửi dữ liệu từ trạm này tới trạm kia mà không đảm bảo các gói tin đến được tới đích. Các cơ chế đảm bảo độ tin cậy được thực hiện bởi tầng trên Tầng ứng dụng

* **Tầng ứng dụng**:

Là tầng trên của mô hình TCP/IP bao gồm các tiến trình và các ứng dụng cung cấp cho người sử dụng để truy cập mạng. Có rất nhiều ứng dụng được cung cấp trong tầng này, mà phổ biến là Telnet: sử dụng trong việc truy cập mạng từ xa, FTP ( File Transport Protocol )dịch vụ truyền tệp tin., EMAIL : dịch vụ truyền thư tín điện tử. WWW (Word Wide Wed).

### Phương thức hoạt động



Hình 2. Quá trình đóng mở gói dữ liệu trong TCP/IP

Cũng tương tự như trong mô hình OSI, khi truyền dữ liệu , quá trình tiến hành từ tầng trên xuống tầng dưới, qua mỗi tầng dữ liệu được thêm vào thông tin điều khiển gọi là Header. Khi nhận dữ liệu thì quá trình xảy ra ngược lại, dữ liệu được truyền từ tầng dưới lên và qua mỗi tầng thì phần header tương ứng sẽ được lấy đi và khi đến tầng trên cùng thì dữ liệu không còn phần header nữa.

Diagram

Description automatically generated

Hình 2. Cấu trúc dữ liệu trong TCP/IP

Hình trên cho ta thấy lược đồ dữ liệu qua các tầng. Trong hình ta thấy tại các tầng khác nhau dữ liệu được mang những thuật ngữ khác nhau

* Trong tầng ứng dụng: dữ liệu là các luồng được gọi là stream.
* Trong tầng giao vận: đơn vị dữ liệu mà TCP gửi xuống gọi là TCP segment.
* Trong tầng mạng, dữ liệu mà IP gửi xuống tầng dưới gọi là IP Datagram
* Trong tầng liên kết, dữ liệu được truyền đi gọi là frame.

### Giới thiệu về IP

IP Là 1 trong những giao thức quan trọng nhất của bộ giao thức TPC/IP, là giao thức hướng không liên kết (connectionless): dữ liệu của IP được truyền đi ngay lập tức nếu có thể (best effort), không có bất kì cơ chế thiết lập kết nối , không có cơ chế báo nhận hay điều khiển luồng nào được sử dụng với IP, các gói tin IP cũng không được đánh số thứ tự khi trao đổi trên mạng…Mỗi gói tin IP được xử lý một cách hoàn toàn độc lập với các gói tin IP khác .Giao thức IP sử dụng cơ chế định địa chỉ theo kiểu phân cấp, trong đó phần NetworkId của địa chỉ giống như tên của một con đường và phần hostId của địa chỉ sẽ là số nhà của một căn nhà trên con đường ấy.Không có cơ chế khôi phục lại gói tin bị mất trên đường truyền. Việc này được giao lại cho các giao thức tầng trên để đảm bảo độ tin cậy (TCP) .

### Giới thiệu về Port

Port mạng là điểm cuối của quá trình giao tiếp trong hệ thống mạng máy tính. Port là giao thức bit 16 đứng đầu (chèn vào phần đầu header) của mỗi gói tin trong giao thức TCP, UDP hay còn gọi là cổng port, nơi quy định các tập dữ liệu riêng biệt. Port là một dạng thuật toán đã được định sẵn và mỗi máy tính cần phải có thì mới có thể nhận và gửi các gói tin đi được. Khi hoạt động Port cũng được quy đổi giống với số bit của bất kì một mã dữ liệu nào đó. Đơn giản thì đây giống như cánh cổng có quyền cho vào hay không với các dữ liệu muốn ra vào hệ thống máy tính của bạn. Khi có 1 địa chỉ IP, bạn sẽ xác định được một máy trong một mạng, nhưng nếu chúng chạy cùng nhiều dịch vụ khác nhau thì phải có cách để phân biệt được chúng và lúc này bạn cần đến Port.

Cổng mạng được cung cấp bởi các giao thức Lớp Truyền tải (Transport Layer) trong mô hình OSI. Mỗi giao thức và tổ hợp địa chỉ truyền tải được xác định thông qua port với giao thức 16-bit được gọi là số cổng, chẳng hạn như Giao thức TCP/IP và Giao thức UDP là một số phục vụ giao tiếp điểm cuối giữa hai máy tính.

**Hoạt động của Port**

Port hoạt động bằng cách xác định cổng cấp quyền cho dữ liệu được ra hay vào trong hệ thống máy tính của bạn trong mạng. Tất cả các thiết bị được kết nối mạng đều được trang bị các port được tiêu chuẩn hóa có số hiệu được ấn định. Những con số này được dành riêng cho một số giao thức nhất định và chức năng liên quan của chúng. Các port luôn được liên kết với địa chỉ IP của máy chủ cũng như các giao thức sử dụng cho việc giao tiếp trong mạng máy tính với nhau. Mỗi port được đặt tên bằng 1 số thứ tự nhất định và mỗi cổng sẽ phục vụ cho những dịch vụ cụ thể khác nhau nhằm mục đích cho phép các gói tin dễ dàng đến đúng nơi quy định. Bởi vì mục đích này mà các port có số thấp (từ 1024 trở xuống) thường được sử dụng cho các dịch vụ phổ biến từ trước đến nay. Còn các port có số cao hơn (từ 1024 trở lên) thì lại thường được sử dụng làm cổng tạm cho các dịch vụ ít phổ biến hơn.

Để xác định lưu lượng đến nên được hướng đến giao thức nào, các số cổng khác nhau được sử dụng. Chúng cho phép một máy chủ lưu trữ với một địa chỉ IP duy nhất chạy các dịch vụ mạng. Mỗi số cổng có một dịch vụ riêng biệt và đối với mỗi máy chủ lưu trữ có thể có 65535 cổng cho mỗi địa chỉ IP. Tổ chức Internet Assigned Numbers Authority (IANA) chịu trách nhiệm quản lý việc sử dụng các cổng này.  
**Ứng dụng của PORT:**

Port mạng là chìa khóa, địa chỉ nhận diện tập tin, dữ liệu Giống như một người lính gác, số địa chỉ nhà ở, khi bạn đăng ký các loại port trên hệ thống máy tính của bạn sẽ giúp cho các tập tin được truy cập, được đưa vào đúng với địa chỉ port khớp với đầu bit tập tin đó.

Port mạng có tác dụng chọn lọc tin Port sẽ quy định chỉ những tập tin nào mới được nhập vào máy và được thông qua xuất nhập trong hệ thống máy. Nếu như tập tin đúng với cổng bit thì sẽ được xâm nhập vào máy nhưng tất nhiên máy tính sẽ từ chối việc nhập dữ liệu nào đó nếu như không đúng cổng port. Điều này sẽ giúp phân loại và lựa chọn luôn những tập tin an toàn.

Port mạng có tác dụng bảo vệ xâm nhập có hại cho máy tính.

Port cũng được xem là một trong những cổng bảo vệ an toàn cho máy tính của bạn.

Port có thể phát hiện những tập tin xấu, có chứa virus làm ảnh hưởng đến các tập tin và máy tính. Nó sẽ loại bỏ những tập tin đó đi, loại bỏ virus xâm nhập, giúp máy tính luôn giữ được sự an toàn và đảm bảo tránh được những thông tin nhiễu.

**Phân nhóm ba loại port mạng theo quy định của IANA**

* Port từ 0 đến 1023

Các cổng từ 0 đến 1023 là những cổng nổi tiếng hoặc cổng mạng lưới hệ thống.

* Port từ 1024 đến 49151

1024 đến 49151 là những cổng đã ĐK được IANA chỉ định cho một dịch vụ đơn cử theo đơn của một thực thể nhu yếu.

* Port từ 49152 đến 65535

49152 đến 65535 là những cổng động ( riêng, cao ) nằm trong khoảng chừng từ 49,152 đến 65,535. Có thể được sử dụng cho mục tiêu tư nhân hoặc dịch vụ người mua hoặc trong thời điểm tạm thời.

## Giới thiệu về Socket

Socket là điểm cuối end-point trong liên kết truyền thông hai chiều (two-way communication) biểu diễn kết nối giữa Client – Server. Socket được sử dụng để gửi dữ liệu qua lại giữa Client và Server thông qua mạng (network).

Mạng (network) có thể là logical, mạng cục bộ tới máy tính hoặc 1 kết nối vật lí tới mạng bên ngoài, với kết nối riêng của chúng tới các mạng khác. Ví dụ điển hình là Internet, cái mà bạn kết nối thông qua [ISP (Internet Service Provider)](https://thuthuat.taimienphi.vn/isp-la-gi-phan-loai-isp-23684n.aspx) .

### Phân loại Socket

* Socket được chia làm 2 loại chính :
  + Stream Socket

Stream Socket dựa trên giao thức TCP việc truyền dữ liệu chỉ thực hiện giữa 2 quá trình đã thiết lập kết nối. Giao thức này đảm bảo dữ liệu được truyền đến nơi nhận một cách đáng tin cậy, đúng thứ tự nhờ vào cơ chế quản lý luồng lưu thông trên mạng và cơ chế chống tắc nghẽn.

* Ưu điểm của Stream Socket là gì?
  + - Dữ liệu truyền đi được đảm bảo truyền đến đúng nơi nhận, đúng thứ tự.
    - Mỗi thông điệp gửi đi đều có xác nhận trả về để thông báo cho người dùng thông tin về quá trình truyền tải.
* Nhược điểm:
  + - Giữa máy chủ và máy nhận chỉ có 1 IP, nên khi kết nối, 1 máy phải chờ máy còn lại chấp nhận kết nối.
    - Tốc độ không nhanh

Nó phù hợp ứng dụng trong các ứng dụng yêu cầu độ chính xác cao, không yêu cầu cao về tốc độ. Ta có thể kể đến 1 vài ứng dụng như: Mail, thanh toán ngân hàng,...

Diagram

Description automatically generated

Hình 2. Stream Socket

Có thể phân thành 4 giai đoạn như sau:

* **Giai đoạn 1**: Server tạo Socket, gán số hiệu cổng và lắng nghe yêu cầu nối kết. Server sẵn sàng phục vụ Client.socket(): Server yêu cầu tạo một socket để có thể sử dụng các dịch vụ của tầng vận chuyển.
  + bind(): Server yêu cầu gán số hiệu cổng (port) cho socket.
  + listen(): Server lắng nghe các yêu cầu nối kết từ các client trên cổng đã được gán.
* **Giai đoạn 2**: Client tạo Socket, yêu cầu thiết lập một nối kết với Server.
  + socket(): Client yêu cầu tạo một socket để có thể sử dụng các dịch vụ của tầng vận chuyển, thông thường hệ thống tự động gán một số hiệu cổng còn rảnh cho socket của Client.
  + connect(): Client gởi yêu cầu nối kết đến server có địa chỉ IP và Port xác định.
  + accept(): Server chấp nhận nối kết của client, khi đó một kênh giao tiếp ảo được hình thành, Client và server có thể trao đổi thông tin với nhau thông qua kênh ảo này.
* **Giai đoạn 3**: Trao đổi thông tin giữa Client và Server.
  + Sau khi chấp nhận yêu cầu nối kết, thông thường server thực hiện lệnh read() và nghẽn cho đến khi có thông điệp yêu cầu (Request Message) từ client gởi đến.
  + Server phân tích và thực thi yêu cầu. Kết quả sẽ được gởi về client bằng lệnh write().
  + Sau khi gởi yêu cầu bằng lệnh write(), client chờ nhận thông điệp kết quả (ReplyMessage) từ server bằng lệnh read().
* **Giai đoạn 4**: Kết thúc phiên làm việc.
  + Các câu lệnh read(), write() có thể được thưc hiện nhiều lần (ký hiệu bằng hình ellipse).
  + Kênh ảo sẽ bị xóa khi Server hoặc Client đóng socket bằng lệnh close().

**Datagram Socket:**

Diagram

Description automatically generated

Hình 2. Datagram Socket

Dựa trên giao thức UDP việc truyền dữ liệu không yêu cầu có sự thiết lập kết nối giữa 2 quá trình.

* Ưu điểm :
  + - Quá trình kết nối và truyền tải thông tin đơn giản, không cần thực hiện nhiều thao tác.
    - Thời gian truyền tải dữ liệu cực nhanh.
* Nhược điểm :
  + - Quá trình truyền thông tin không đảm bảo tin cậy, thông tin có thể truyền sai thứ tự hoặc bị lặp.

Phù hợp với các ứng dụng như live stream, video call, ... vì những ứng dụng này yêu cầu tốc độ nhanh, nếu mất mát 1 vài gói tin trong lúc truyền thì vẫn không ảnh hưởng quá nhiều.

# TRIỂN KHAI

## . Mô tả

Lập trình socket video streaming với việc kiểm tra 3 bit đầu được thêm vào cùng data để kiểm tra sự chính xác của dữ liệu. Với việc bit Parity được random là gửi thành công và không thành công ngẫu nhiên, frame từ video lúc đó sẽ không được gửi đồng thời bit Send và Read cũng được giữ nguyên và tiếp tục gửi những frame tiếp theo, khi đó bit Send và Read sẽ được tăng lên để đối chiếu với bên nhận là server.

### Công việc phải làm

Tạo mô hình Server chờ kết nối từ Client

Tạo mô hình Client để gửi video cho Server

Triển khai kết hợp các bit Send Read Parity vào data truyền từ Client với Server

Mô phỏng gửi 1 video với việc in ra 3 bit kiểm tra trong cửa sổ Console.

Sau khi kết thúc gửi sẽ ngắt kết nối

## . Ngôn ngữ và thư viện sử dụng

Để thực hiện yêu cầu bìa toán, nhóm sử dụng ngôn ngữ Python. Python là ngôn ngữ được sử dụng rộng rãi, với các module được xây dựng sẵn cho video streaming. Vì thế tạo sự thuận tiện cho việc lập trình về mạng máy tính.

Bảng Thư viện sử dụng

|  |  |
| --- | --- |
| **Thư viện** | **Chức năng** |
| cv2 | Xử lí ảnh và video |
| socket | Xử lí phương thức lập trình socket |
| struct | Xử lí |
| pickle | Chuyển đổi cấu trúc sang binary |
| random | Khởi tạo random cho biến parity |
| numpy | Xử lý mảng và ma trận |

## . Sơ đồ khối

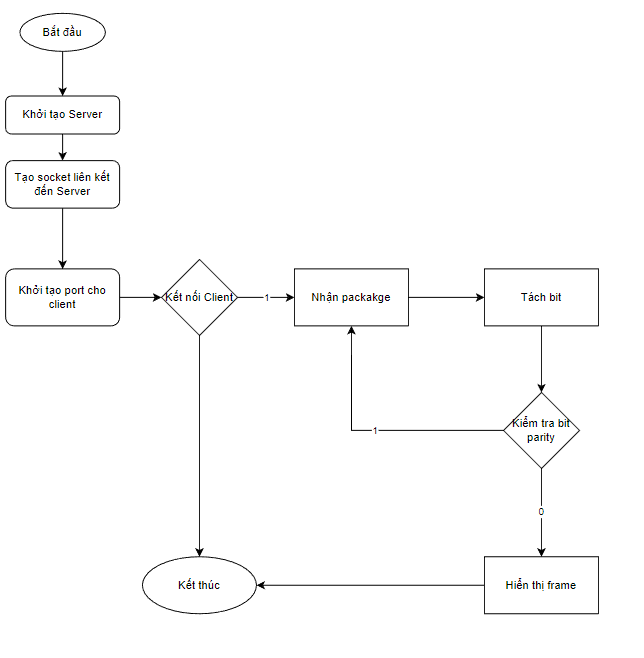
### Khung data

Ảnh có chứa văn bản

Mô tả được tạo tự động

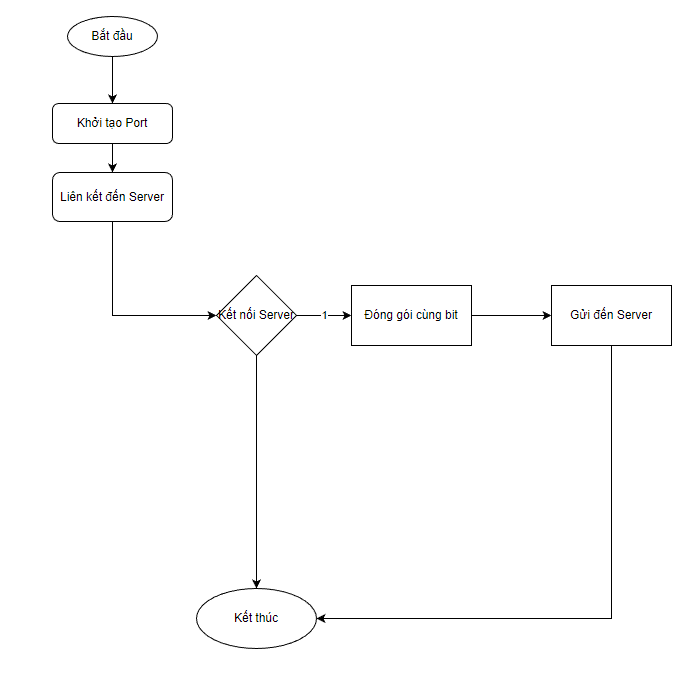
Hình . Khung data được đóng gói

### Server



Hình . Sơ đồ khối của Server

### Client



Hình . Sơ đồ khối của Client

## . Code

### Server

|  |
| --- |
| import socket  import sys  import threading  import cv2  import pickle  import numpy as np  import struct  import zlib  import time  import queue  from threading import Thread  q = queue.Queue(maxsize=100000) # hang doi cac frame cua video  HOST='127.0.0.1'  PORT=8485  s=socket.socket(socket.AF\_INET,socket.SOCK\_STREAM)  print('Socket created')  s.bind((HOST,PORT))  print('Socket bind complete')  s.listen(10)  print('Socket now listening')  conn,addr=s.accept()  # Nhan so frame cua video  data = conn.recv(1024)  totalFrame = int(data.decode("utf8"))  # FPS cua video  data = conn.recv(1024)  FPS = int(data.decode("utf8"))  payload\_size = struct.calcsize(">L")  print("payload\_size: {}".format(payload\_size))  done\_flag = 0;  def frame\_recv():      data = b""      stop\_flag = 0;      fps,st,frames\_to\_count,cnt = (0,0,20,0)      while stop\_flag == 0:          while len(data) < payload\_size:              print("Recv: {}".format(len(data)))              data += conn.recv(4096)            print("Done Recv: {}".format(len(data)))          packed\_msg\_size = data[:payload\_size]          data = data[payload\_size:]          try:              msg\_size = struct.unpack(">L", packed\_msg\_size)[0]          except:              continue          print("msg\_size: {}".format(msg\_size))          while len(data) < msg\_size:              data += conn.recv(4096)            msg\_data = data[:msg\_size]          # 15 la de gui 3bit s,r,p moi bit dang de fix length la 5          frame\_data = msg\_data[15:]          bit\_seq = msg\_data[:15]          bit\_seq = bit\_seq.decode('utf-8')          data = data[msg\_size:]          # ktra bit p xem loi khong          if int(bit\_seq[10:]) == 1:              resp = 'nak'              conn.sendall(resp.encode('utf-8'))              continue          else:              resp = 'ack'            frame=pickle.loads(frame\_data, fix\_imports=True,encoding="bytes")          try:              frame = cv2.imdecode(frame, cv2.IMREAD\_COLOR)          except:              continue          q.put(frame)    #day frame vao queue          cnt+=1          conn.sendall(resp.encode('utf-8'))          if (totalFrame == cnt):              stop\_flag = 1              break      print("done thread")      global done\_flag      done\_flag = 1  def frame\_show():      fps,st,frames\_to\_count,cnt = (0,0,20,0)      while (done\_flag == 0):          frame = q.get()  #In ra FPS cua video          frame = cv2.putText(frame,'FPS: '+str(fps),(10,40),cv2.FONT\_HERSHEY\_SIMPLEX,0.7,(0,0,255),2)          cv2.imshow('Received',frame)          cv2.waitKey(1)          if cnt == frames\_to\_count:              try:                  fps = round(frames\_to\_count/(time.time()-st))                  st=time.time()                  cnt=0              except:                  pass          cnt+=1          time.sleep(1/(FPS+5)) # VD: 30fps -> load 1 frame sau 1/30s      print("end thread")      s.close()  # Chay da luong 2 ham: mot ham buffer frame, mot ham lay frame ra -> on dinh FPS  t1 = threading.Thread(target=frame\_recv, name='t1')  t2 = threading.Thread(target=frame\_show, name='t2')  t1.start()  t2.start()  t1.join()  t2.join() |

### Client

|  |
| --- |
| import cv2, imutils  import io  import socket  import struct  import time  import pickle  import sys  # print(sys.path)  from queue import Queue  import random  client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  client\_socket.connect(('127.0.0.1', 8485))  # client\_socket.connect(('10.0.2.15', 8485))  # connection = client\_socket.makefile('wb')  # Thay video\_path vao nhe - neu input file tu command line: sys.argv[1]  cap = cv2.VideoCapture(sys.argv[1])  # Gui di so frame cua video  totalFrame = cap.get(cv2.CAP\_PROP\_FRAME\_COUNT)  client\_socket.sendall(str(int(totalFrame)).encode('utf-8'))  # Gui di FPS cua video  FPS = cap.get(cv2.CAP\_PROP\_FPS)  print(FPS)  client\_socket.sendall(str(int(FPS)).encode('utf-8'))  img\_counter = 0  sBit = 0  rBit = 0  pBit = 0  encode\_param = [int(cv2.IMWRITE\_JPEG\_QUALITY), 50]  frame\_dist = {}  while (cap.isOpened()):      # Doc video theo frame      ret, frame = cap.read()      try:          frame = imutils.resize(frame,width=WIDTH)      except:          pass      frame\_dist[img\_counter] = frame      # Ma hoa      result, frame = cv2.imencode('.jpg', frame\_dist[sBit], encode\_param)      # ghep bit moi bit co do dai 5, chay duoc toi da 99999 frame      print(sBit, rBit, pBit)      bit\_seq = format(sBit, '05d') + format(rBit, '05d') + format(pBit, '05d')      bit\_seq = bit\_seq.encode('utf-8')      # Chuyen thanh dang bytes      data = bit\_seq + pickle.dumps(frame, 0)      size = len(data)        print("{}: {}".format(img\_counter, size))      # print(struct.pack(">L", size))      client\_socket.sendall(struct.pack(">L", size) + data)      resp = client\_socket.recv(1024)      # Nhan ket qua thanh cong hoac that bai      if (resp.decode('utf-8') == 'nak'):          # Khong tang, lan sau gui lai frame sBit          sBit = sBit          rBit = rBit      else:          # Thanh cong, frame sBit da duoc gui, xoa di cho nhe, tang sBit gui frame tiep theo          del frame\_dist[sBit]          sBit += 1          rBit += 1        # Radom pBit gia lap kenh truyen      pBit = random.randint(0,1)      img\_counter += 1      if rBit == totalFrame:          break      print('total frame:',totalFrame)  cap.release() |

### Giải thích đoạn code liên quan đến kiểm tra lỗi

3.4.3.1 Đoạn code kiểm tra bit lỗi ở server

    while stop\_flag == 0:

        while len(data) < payload\_size:

            print("Recv: {}".format(len(data)))

            data += conn.recv(4096)

        print("Done Recv: {}".format(len(data)))

        packed\_msg\_size = data[:payload\_size]

Chờ nhận đủ gói từ socket

        data = data[payload\_size:]

        try:

            msg\_size = struct.unpack(">L", packed\_msg\_size)[0]

        except:

            continue

        print("msg\_size: {}".format(msg\_size))

        while len(data) < msg\_size:

            data += conn.recv(4096)

Lấy ra đoạn dữ liệu cần xử lí

        msg\_data = data[:msg\_size]

        # 15 la de gui 3bit s,r,p moi bit dang de fix length la 5

        frame\_data = msg\_data[15:]

Tách ra phần bộ 3 số kiểm tra lỗi, mỗi số dài 5bit

        bit\_seq = msg\_data[:15]

        bit\_seq = bit\_seq.decode('utf-8')

        data = data[msg\_size:]

        # ktra bit p xem loi khong

Kiểm tra bit lỗi p

-nếu p = 1 thì gói vừa gửi bị lỗi, gửi lại bản tin **‘nak’**

- nếu p = 0 thì gói vừa gửi không bị lỗi, gửi lại bản tin **‘ack’**

        if int(bit\_seq[10:]) == 1:

            resp = 'nak'

            conn.sendall(resp.encode('utf-8'))

            continue

        else:

            resp = 'ack'

3.4.3.2 Đoạn code ghép bit kiểm tra vào phần data ở client

img\_counter = 0

sBit = 0

Bộ 3 số kiểm tra lỗi

rBit = 0

pBit = 0

encode\_param = [int(cv2.IMWRITE\_JPEG\_QUALITY), 50]

frame\_dist = {}

while (cap.isOpened()):

    # Doc video theo frame

    ret, frame = cap.read()

sBit chính là số thứ tự frame gửi đi

    try:

        frame = imutils.resize(frame,width=WIDTH)

    except:

Mã hoá nội dung 1 frame video

        pass

    frame\_dist[img\_counter] = frame

    # Ma hoa

    result, frame = cv2.imencode('.jpg', frame\_dist[sBit], encode\_param)

    # ghep bit moi bit co do dai 5, chay duoc toi da 99999 frame

    print(sBit, rBit, pBit)

    bit\_seq = format(sBit, '05d') + format(rBit, '05d') + format(pBit, '05d')

    bit\_seq = bit\_seq.encode('utf-8')

Ghép 3 số kiểm tra với data gủi đi

Mã hoá 3 số kiểm tra

    # Chuyen thanh dang bytes

    data = bit\_seq + pickle.dumps(frame, 0)

    size = len(data)

    print("{}: {}".format(img\_counter, size))

    # print(struct.pack(">L", size))

Gửi đi qua socket

    client\_socket.sendall(struct.pack(">L", size) + data)

    resp = client\_socket.recv(1024)

    # Nhan ket qua thanh cong hoac that bai

Nếu sau khi gửi, nhận được bản tin phản hồi là **‘nak’**, tức là tin gửi đi đã lỗi, sBit và rBit không tăng

    if (resp.decode('utf-8') == 'nak'):

        # Khong tang, lan sau gui lai frame sBit

        sBit = sBit

        rBit = rBit

    else:

        # Thanh cong, frame sBit da duoc gui, xoa di cho nhe, tang sBit gui frame tiep theo

Ngược lại bản tin **‘ack’** có nghĩa là gửi thành công, tiến hành tăng sBit và rBit

        del frame\_dist[sBit]

        sBit += 1

        rBit += 1

    # Radom pBit gia lap kenh truyen

Sinh ngẫu nhiên bit lỗi cho data tiếp theo

    pBit = random.randint(0,1)

    img\_counter += 1

    if rBit == totalFrame:

        break

    print('total frame:',totalFrame)

cap.release()

## . Demo

### Khởi tạo Server



Hình . Sau khi khởi tạo Socket ở chương trình Server

### Sau khi kết nối Client với Server

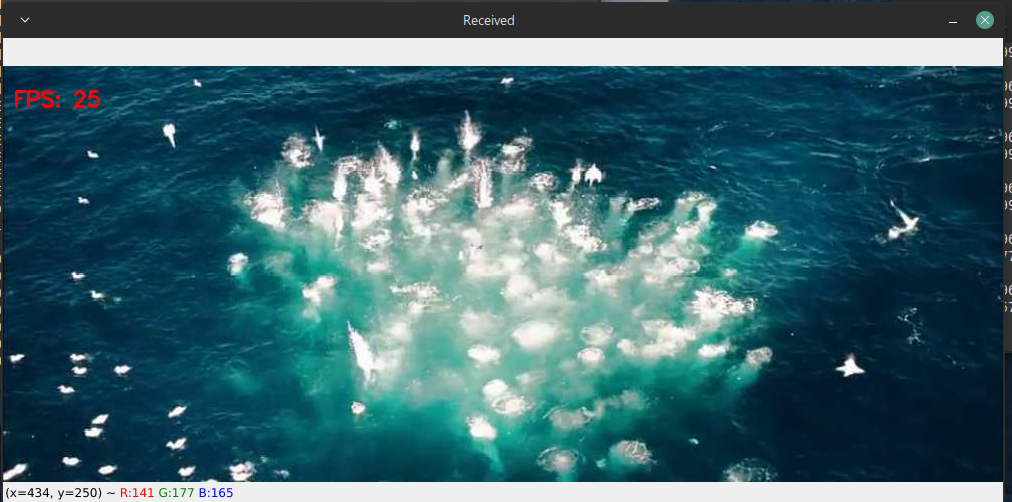
Ảnh có chứa văn bản

Mô tả được tạo tự động

Data được gửi lại cho đến khi nào bit lỗi được kiểm tra đúng thì mới sang data tiếp theo

Hình . Các bit kiểm tra được in trên màn hình

Với 2 bit đầu là Send bit và Read bit, bit thứ 3 sẽ là bit parity, khi là 0 thì tiếp tục gửi frame và tăng 2 bit còn lại, nếu là 1 thì sẽ bỏ qua frame, khi đó 2 bit còn lại giữ nguyên cho tới frame tiếp theo được gửi.



Hình . Video được truyền

# KẾT LUẬN

Trong bài tập lớn của nhóm, chúng em đã thực hiện được thành công một ứng dụng mạng TCP/IP kết hợp với các bit kiểm tra được đóng gói vào cùng dữ liệu. Thông qua học phần Mạng máy tinh và IoT, chúng em đã trau dồi, tìm hiểu thêm nhiều kiến thức bổ ích về kĩ năng lập trình, kĩ năng làm việc nhóm, kĩ năng quản lý…

Do còn thiếu sót trong kiến thức, tư duy và kinh nghiệm về lập trình cũng như kế hoạch nhóm và thực tế về hệ thống tổ chức quản lý nên nhóm còn nhiều sai sót trong dự án. Nhóm chúng em rất mong nhận được sự đánh giá khách quan, chính xác và góp ý của thầy và các bạn để hoàn thiện hơn.

Nhóm chúng em xin chân thành cảm ơn sự giúp đỡ tận tình của thầy **PGS.TS Nguyễn Tài Hưng** đã hướng dẫn và truyền đạt các kiến thức cần thiết để hoàn thành đề tài bài tập lớn .