BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC CẦN THƠ

TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

NIÊN LUẬN CƠ SỞ

NGÀNH MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG DỮ LIỆU

Đề tài

HỆ THỐNG TRUYỀN HÌNH ẢNH VÀ ÂM THANH TRÊN INTERNET

Phân hệ 1: Truyền hình ảnh trên Internet

Giảng viên hướng dẫn:

ThS. TRẦN DUY QUANG

Sinh viên thực hiện:

LÊ THỊ NGỌC DIỆP

MSSV: B2013462

KHOÁ 46

Cần Thơ, 11/2023

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

MỤC LỤC

CHƯƠNG 1: GIỚI THIỆU 1
1.1. Lý do chọn đề tài 1
1.2. Mục tiêu 1
1.3. Đối tượng 1
1.4. Phạm vi nghiên cứu 1
1.5. Phương pháp nghiên cứu 2
1.5.1. Phân tích yêu cầu 2
1.5.2. Phân tích và thiết kế 2
1.5.3. Hướng giải quyết 2
1.5.4. Lý thuyết 2
1.6. Ý nghĩa thực tiễn của đề tài 2
1.7. Bố cục niên luận cơ sở 2
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT 3
2.1. Đường Truyền Internet 3
2.2. Giao thức TCP 3
2.3. Giao thức UDP 3
2.4. Hình ảnh 4
2.5. Âm thanh 4
2.6. Cân Bằng Tốc Độ Đường Truyền: 5
CHƯƠNG 3: PHƯƠNG PHÁP THỰC HIỆN 6
3.1. Phân tích yêu cầu 6
3.2. Thu thập thông tin và dữ liệu 6
3.3. Lựa chọn ngôn ngữ lập trình và môi trường thiết kế 6
3.4. Xem xét hệ thống 6
3.5. Các bước tổng quan cài đặt và lập trình 7
3.6. Tiến hành thực hiện dựa vào các thông tin dữ liệu 8
3.6.1. Cài đặt Python 8
3.6.2. Cài đặt Visual Studio Code 8
3.6.3. Cài Đặt Extension Python 8
3.6.4. Cài đặt các thư viện 8
3.6.5. Tiến hành lập trình máy chủ (server) 9
3.6.6. Tiến hành lập trình máy khách (client) 13

CHƯƠNG 4: KẾT QUẢ THỰC NGHIỆM 18
4.1. Kết quả tốc độ truyền dữ liệu 18
4.2. Kết quả chất lượng hình ảnh 18
4.3. Đánh giá chất lượng của hệ thống 19
4.4. Hạn chế 19
CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 20
5.1. Kết luận 20
5.2. Hướng phát triển 20
Tài liệu tham khảo: 21

MỤC LỤC HÌNH ẢNH

Hình 1. UDP socket của server *9*

Hình 2. Hàm generate_video() của server *9*

Hình 3. Hàm send_video() của server *10*

Hình 4. Hàm get_video() của server *12*

Hình 5. UDP socket của client *13*

Hình 6. Hàm generate_video() của client *13*

Hình 7. Hàm get_video() của client *15*

Hình 8. Hàm send_video() của client *17*

Hình 9. Kết quả đạt được *18*

DANH MỤC TỪ VIẾT TẮT

Từ viết tắt	Viết đầy đủ
LAN	Local Area Network
UDP	User Datagram Protocol
TCP	Transmission Control Protocol
OpenCV	Open Source Computer Vision Library
OSI	Open Systems Interconnection
CNTT	Công Nghệ Thông Tin
ICT	Information & Communications Technologies

LỜI CẢM ƠN

Em xin chân thành gửi lời cảm ơn thầy Trần Duy Quang – khoa Mạng máy tính và truyền thông đã nhiệt tình hướng dẫn em trong xuyên suốt quá trình thực hiện đề tài niên luận cơ sở ngành. Thầy đã có những lời chỉ bảo nhiệt tình và đóng góp ý kiến để em được hoàn thiện hơn.

Em cũng xin chân thành cảm ơn các thầy cô trong trường Công Nghệ Thông Tin và Truyền Thông, những người đã dạy dỗ và trang bị cho em kiến thức để hoàn thành đề tài này.

Trong quá trình nghiên cứu và thực hiện đề tài em có những cố gắng nhất định, nhưng vẫn còn khá nhiều thiếu sót. Vậy nên, em rất mong được lắng nghe các ý kiến đóng góp từ các thầy cô và các bạn để em được hoàn thiện hơn trong tương lai.

TÓM TẮT

Hiện nay, nhu cầu ngày càng cao về các trải nghiệm chất lượng trên Internet đã đặt ra thách thức đối với hệ thống truyền tải âm thanh và hình ảnh. Đề tài này nhằm tìm hiểu và đánh giá các phương pháp, giải pháp hiện tại cũng như những hướng phát triển tiềm năng để cải thiện trải nghiệm người dùng. Mục tiêu của nghiên cứu là khám phá cách tích hợp linh hoạt giữa các giao thức để tối ưu hóa cân bằng tốc độ truyền tải. Đồng thời, nghiên cứu cũng như chú trọng vào khả năng áp dụng của các công nghệ để cải thiện hiệu suất và tốc độ truyền tải của hệ thống. Phương pháp nghiên cứu được thiết kế dựa trên việc đánh giá tốc độ truyền thông tin của hai giao thức TCP và UDP trong môi trường thực nghiệm. Thông qua việc phân tích và đánh giá, hệ thống được tính hợp bởi hai giao thức trên. Ngoài ra, em tiến hành thực hiện trong không gian mạng LAN và thực hiện các thử nghiệm để đánh giá khả năng tích hợp của hai giao thức trong hệ thống. Kết quả quan trọng của nghiên cứu bao gồm việc chứng minh rằng sự kết hợp giữa hai giao thức TCP và UDP có thể cung cấp trải nghiệm người dùng mượt mà hơn, không bị ngắt hoãn trong thời gian sử dụng. Đồng thời, hệ thống hứa hẹn mở ra những khả năng mới để cải thiện băng thông và độ trễ. Kết luận chính của hệ thống là sự linh hoạt trong tích hợp giữa các giao thức có thể là chìa khóa để đáp ứng ngày càng cao của người dùng. Trong tương lai, hệ thống sẽ tiếp tục phát triển và hoàn thiện, đồng thời tăng cường thử nghiệm và triển khai thực tế để đảm bảo tính ứng dụng cao.

ABSTRACT

Nowadays, the increasing demand for high-quality experiences on the Internet poses a challenge to audio and video transmission systems. This project aims to explore and evaluate current methods, solutions, as well as potential development directions to enhance user experiences. The research goal is to investigate the flexible integration of protocols to optimize transmission speed balance. Simultaneously, the study focuses on the applicability of technologies to improve the performance and transmission speed of the system. The research methodology is designed based on evaluating the information transmission speed of two protocols, TCP and UDP, in an experimental environment. Through analysis and evaluation, the system is integrated by these two protocols. Additionally, experiments are conducted within a LAN network to assess the integration capabilities of the two protocols in the system. Key results of the research include demonstrating that the combination of TCP and UDP protocols can provide users with a smoother experience without interruptions during usage. Moreover, the system holds the promise of opening up new possibilities to improve bandwidth and latency. The main conclusion of the system is that flexibility in the integration of protocols can be the key to meeting the increasing demands of users. In the future, the system will continue to develop and refine, while enhancing real-world testing and deployment to ensure high application adaptabilit.

CHƯƠNG 1: GIỚI THIỆU

1.1. Lý do chọn đề tài

Em chọn đề tài "Hệ thống truyền hình ảnh và âm thanh trên Internet" vì ngày nay, trên thế giới đang phát triển với nền công nghiệp 4.0, công nghiệp công nghệ thông tin (CNTT) hay công nghiệp ICT được hiểu là công nghiệp công nghệ số. Hiện nay để thuận tiện trong công việc cũng như giao tiếp, mọi người thường giao tiếp với nhau thông qua Internet. Khi biết việc giao tiếp với nhau trên Internet rất quan trọng về độ chính xác và tốc độ xử lý của hệ thống, nên việc điều chỉnh về tốc độ kết nối cũng cần được tối ưu. Để đáp ứng nhu cầu giao tiếp qua Internet ngày càng tăng nhanh cần phải có các ứng dụng phụ vụ và hỗ trợ đến nhiều người dùng. Cũng như đảm bảo các các thông tin giao tiếp phải chính xác và được bảo mật cho người dùng.

1.2. Mục tiêu

Mục tiêu của đề tài nhằm hướng đến việc đáp ứng các nhu cầu giao tiếp qua Internet như họp trực tuyến, học trực tuyến hoặc trò chuyện với nhau nhanh chóng và tiện lợi.

1.3. Đối tượng

Đối tượng của đề tài là mọi người có nhu cầu trao đổi thông tin qua Internet.

1.4. Phạm vi nghiên cứu

Đề tài thực hiện truyền hình ảnh và âm thanh trên Internet trong phạm vi học phần Niên luận cơ sở ngành Mạng máy tính và truyền thông dữ liệu, và tham khảo các thông tin sẵn có trên Internet.

1.5. Phương pháp nghiên cứu

1.5.1. Phân tích yêu cầu

- Đối với công việc: Thuận lợi cho việc thảo luận và làm việc nhóm từ xa.
- Đối với giáo dục: Phương tiện hỗ trợ để tổ chức dạy, họp, học tập.
- Đối với cuộc sống: Trao đổi, trò chuyện với gia đình và bạn bè.

1.5.2. Phân tích và thiết kế

- Phân tích các hàm cần sử dụng để hệ thống hoạt động như: hình ảnh, âm thanh.
- Thiết kế dựa vào các hệ thống có sẵn như Google Meet, Zoom Meeting,…

1.5.3. Hướng giải quyết

Thu thập thông tin trên Internet và xem các video làm các hệ thống tương tự.

1.5.4. Lý thuyết

- Tìm về các thư viện được Python hỗ trợ như: OpenCV, Pyaudio, Socket, Threading,…
- Cách thức giao tiếp bằng giao thức UDP/TCP và kết nối bằng ngôn ngữ Python.

1.6. Ý nghĩa thực tiễn của đề tài

Hệ thống giúp ích cho việc trò chuyện trao đổi thông tin một cách nhanh chóng, gặp gỡ nhau nhưng không cần tốn nhiều thời gian di chuyển. Một số công việc có thể được giải quyết từ xa thông qua việc giao tiếp trực tuyến trên hệ thống.

1.7. Bố cục niên luận cơ sở

Bố cục gồm có 5 phần:

1. Giới thiệu.
2. Cơ sở lý thuyết.
3. Phương pháp thực hiện.
4. Kết quả thực nghiệm.
5. Kết luận và hướng phát triển.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

Cân bằng tốc độ đường truyền âm thanh và hình ảnh trên Internet là đều quan trọng của việc cải thiện trải nghiệm của người dùng khi sử dụng dịch vụ trực tuyến. Để hiểu rõ hơn về đề tài này, lý thuyết là một phần quan trọng liên quan đến đường truyền, âm thanh, hình ảnh, và cách chúng tương tác với nhau trên mạng Internet.

2.1. Đường Truyền Internet

Mạng Internet là hệ thống kết nối máy tính toàn cầu, sử dụng nhiều giao thức truyền tải dữ liệu như TCP hay UDP, để truyền âm thanh, hình ảnh và dữ liệu giữa các thiết bị. Tốc độ đường truyền, băng thông, và độ trễ đều ảnh hưởng đến việc truyền và nhận dữ liệu qua Internet.

2.2. Giao thức TCP

- TCP (Transmission Control Protocol) là giao thức định tuyến trong mô hình OSI, hoạt động ở tầng vận chuyển, cung cấp dịch vụ giao vận đáng tin cậy thông qua cơ chế thiết lập kết nối, kiểm soát lưu lượng và xác nhận lỗi. Khi sử dụng TCP, dữ liệu được chia thành các phân đoạn và đóng gói thành nhiều gói tin. TCP đảm bảo gói tin đến nơi đích trong trật tự và nguyên vẹn. TCP rất phù hợp cho các ứng dụng cần truyền dữ liệu lớn mà không mất gói tin như truyền văn bản.
- So với UDP, TCP có độ trễ cao hơn nhưng đảm bảo tính nhất quán, toàn vẹn dữ liệu. Rất phù hợp với truyền dữ liệu lớn cần chất lượng cao.

2.3. Giao thức UDP

- UDP (User Datagram Protocol) là giao thức không kết nối trong mô hình OSI, hoạt động ở tầng vận chuyển. UDP không yêu cầu thiết lập kết nối trước khi gửi dữ liệu, cũng không xác nhận hoặc đảm bảo dữ liệu đến đích. UDP có lợi thế về tốc độ truyền so với TCP do không phải duy trì kết nối. Phù hợp với ứng dụng yêu cầu thời gian phản hồi nhanh. Truyền phát hình ảnh, âm thanh trực tuyến như cuộc gọi video, trò chuyện video online thường sử dụng UDP. Các game trực tuyến, video hội nghị sử dụng UDP để truyền luồng dữ liệu liên tục với độ trễ thấp.
- So với TCP, UDP không đảm bảo tính nhất quán, có thể mất gói dữ liệu. Song phù hợp với nội dung yêu cầu truyền nhanh hơn chất lượng.

2.4. Hình ảnh

- Khi giao tiếp trực tuyến, dữ liệu hình ảnh được chuyển thành tín hiệu số để truyền qua Internet. Hình ảnh thường là dữ liệu hình ảnh trong video. Trong trường hợp video, dữ liệu hình ảnh liên tục được chia thành các khung hình (frames), và mỗi khung hình đều được số hóa để tạo thành dữ liệu số liệu cần truyền đi. Các phương tiện truyền thông trực tuyến thường sử dụng các giao thức và codec để nén và giải nén dữ liệu video, giúp giảm dung lượng tệp và tăng tốc độ truyền tải.
- Khi tín hiệu số của hình ảnh đã được tạo ra, nó sẽ được truyền qua Internet thông qua giao thức truyền tải dữ liệu như TCP hoặc UDP

2.5. Âm thanh

- Số hóa âm thanh: Dữ liệu âm thanh, được ghi lại từ microphone hoặc nguồn âm thanh khác, cần được chuyển đổi từ dạng sóng âm thành dạng số. Quá trình này gọi là số hóa âm thanh.
- Mã hóa âm thanh: Dữ liệu số hóa sau đó có thể được mã hóa để giảm dung lượng tệp và tối ưu hóa cho việc truyền tải qua mạng. Các codec âm thanh thường được sử dụng để thực hiện công việc này. Các codec như MP3, AAC, hoặc Opus là những ví dụ phổ biến.
- Truyền qua mạng: Dữ liệu âm thanh số sau khi được mã hóa sẽ được truyền qua mạng Internet. Các giao thức truyền tải dữ liệu như TCP hoặc UDP thường được sử dụng để chuyển gửi dữ liệu âm thanh.
- Giải mã âm thanh: Tại đầu nhận, dữ liệu âm thanh số được giải mã để khôi phục thành dạng sóng âm, và sau đó có thể được phát ra qua tai nghe hoặc loa.
- Khi các quá trình được thực hiện đầy đủ sẽ đảm bảo người tham gia giao tiếp, trao đổi trực tuyến có thể nghe và tương tác với âm thanh từ đối tác trò chuyện một cách hiệu quả qua Internet.

2.6. Cân Bằng Tốc Độ Đường Truyền:

Cân bằng tốc độ đường truyền là quá trình điều phối tài nguyên trong mạng, để đảm bảo rằng âm thanh và hình ảnh được truyền qua mạng một cách mượt mà và không bị gián đoạn. Cân bằng tốc độ đường truyền có thể được thực hiện bằng cách điều chỉnh băng thông, tối ưu hóa độ trễ, và sử dụng các giao thức truyền tải hiệu quả.

Các phương pháp cân bằng tốc độ đường truyền thường bao gồm:

- Round Robin: Phương pháp này chia đều lượng dữ liệu gửi đi qua tất cả các đường truyền có sẵn. Mỗi đường truyền lấy lần lượt để gửi dữ liệu.
- Least Connections: Hệ thống chọn đường truyền có ít kết nối nhất để gửi dữ liệu. Điều này giúp tránh tình trạng quá tải trên một số đường truyền.
- Weighted Distribution: Các đường truyền được gán trọng số khác nhau dựa trên hiệu suất hoặc dung lượng, và lượng dữ liệu được phân phối tương ứng với trọng số của từng đường truyền.
- Dynamic Load Balancing: Hệ thống tự động cân bằng tốc độ dựa trên tình trạng hiện tại của đường truyền. Điều này đôi khi liên quan đến việc giám sát và đánh giá liên tục về hiệu suất của các đường truyền.
- Cân bằng tốc độ đường truyền thường được áp dụng trong các môi trường mạng lớn, hệ thống máy chủ, và các ứng dụng yêu cầu độ tin cậy và khả năng mở rộng cao. Điều này giúp đảm bảo rằng không có đường truyền nào bị quá tải trong khi các đường truyền khác còn có thể được sử dụng.

CHƯƠNG 3: PHƯƠNG PHÁP THỰC HIỆN

3.1. Phân tích yêu cầu

Đề tài yêu cầu một hệ thống có thể truyền đi dữ liệu ở dạng văn bản, hình ảnh từ thiết bị này đến thiết bị khác thông qua đường truyền trên mạng Internet. Các dữ liệu truyền đi được thực hiện thông qua các giao thức kết nối trên Internet, một số dữ liệu cần độ chính xác của đường truyền, một số dữ liệu khác lại cần tốc độ của đường truyền phải đảm bảo nhanh. Nên tuỳ theo loại dữ liệu cần có cách thức truyền phù hợp với loại dữ liệu đó và cũng như lựa chọn cách thức kết nối trên mạng Internet hợp lý.

3.2. Thu thập thông tin và dữ liệu

Thông tin và dữ liệu về nhu cầu và mong muốn được thu thập trên các diễn đàn lớn như Facebook, Youtube,…

Sử dụng hai máy tính đóng vai trò là máy chủ (server) và máy khách (client) cùng kết nối chung mạng LAN - mạng máy tính nội bộ. Giao tiếp này cho phép hai máy tính kết nối với nhau để cùng làm việc và chia sẻ dữ liệu.

3.3. Lựa chọn ngôn ngữ lập trình và môi trường thiết kế

Ngôn ngữ lập trình Python được chọn để thiết kế hệ thống vì có những thư viện tích hợp hỗ trợ về hình ảnh có thư viện OpenCV, với các cú pháp ngắn gọn dễ thực hiện nên em chọn ngôn ngữ này giúp thuận tiện và tiết kiệm thời gian hơn trong quá trình thực hiện.

Môi trường thiết kế em dùng Visual Studio Code để lập trình vì môi trường khá quen thuộc với hầu hết mọi người trong lĩnh vực lập trình.

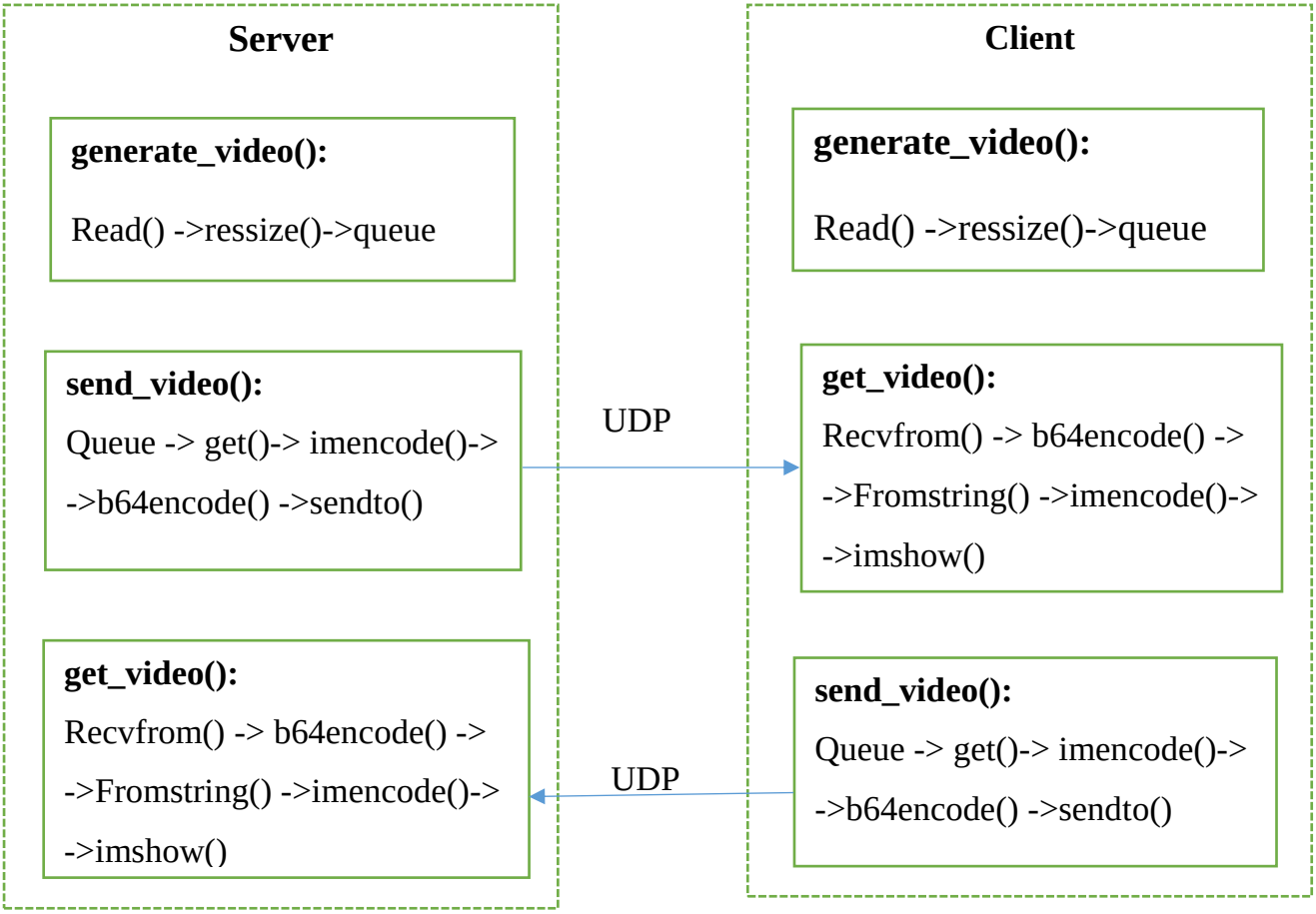
3.4. Xem xét hệ thống

Xem xét các chức năng có trong hệ thống như: nhận tin nhắn, gửi tin nhắn, gửi hình ảnh, nhận và hiển thị hình ảnh.

Các chức năng sẽ kết nối bằng giao thức UDP, để thông tin truyền và nhận được đảm bảo hơn.

Các chức năng sẽ được kết nối tại các số hiệu cổng khác nhau để trách xảy ra đụng độ trong quá trình truyền tải.

Sơ đồ truyền và nhận hình ảnh:



3.5. Các bước tổng quan cài đặt và lập trình

- Cài đặt Python
- Cài đặt Visual Studio Code
- Cài Đặt Extension Python
- Cài đặt các thư viện
- Tiến hành lập trình máy chủ (server)
- Tiến hành lập trình máy khách (client)

3.6. Tiến hành thực hiện dựa vào các thông tin dữ liệu

3.6.1. Cài đặt Python

- Tải Python: Truy cập trang web chính thức của Python tại https://www.python.org/. Tải bản Python 3.x.
- Chạy bản cài đặt.

3.6.2. Cài đặt Visual Studio Code

- Truy cập trang web chính thức của Visual Studio Code tại https://code.visualstudio.com/ và tải bản cài đặt phù hợp với hệ điều hành Windows.
- Chạy bản cài đặt.
- Khởi động Visual Studio Code.
- Trên thanh tìm kiếm tìm "Python: Select Interpreter" rồi chọn phiên bản Python đã cài đặt.

3.6.3. Cài Đặt Extension Python

- Mở Visual Studio Code.
- Trong thanh bên, nhấn vào biểu tượng "Extensions".
- Tìm kiếm "Python" trong ô tìm kiếm.
- Chọn "Python" của Microsoft và nhấn nút "Install".

3.6.4. Cài đặt các thư viện

- Những thư viện cần sử dụng cho hệ thống như: socket, time, base64, threading, wave, pickle, struct, queue, và os là các thư viện mặc định của Python.
- Cài đặt các thư viện được hỗ trợ của Python trên terminal như: openCV, imutils, numpy.

pip install opencv-python

pip install imutils

pip install numpy

3.6.5. Tiến hành lập trình máy chủ (server)

- Khởi tạo các thư viện cần thiết:

  import cv2, imutils, socket

  import numpy as np

  import time, queue, os, base64, threading

- Khởi tạo hàng đợi để lưu trữ và chia sẻ dữ liệu hình ảnh.

  q = queue.Queue(maxsize=10)

- Sử dụng thư viện cv2 khởi tạo một đối tượng VideoCapture để nhận video đầu vào từ camera.

  vid = cv2.VideoCapture(0)

- Khởi tạo một biến BUFF_SIZE = 65536 dùng để nhận dữ liệu từ socket.
- Khởi tạo socket UDP cho việc truyền dữ liệu và thiết lập tùy chọn socket là một hằng số chỉ định kích thước buffer với kích thước BUFF_SIZE.

```
server_socket = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
server_socket.setsockopt(socket.SOL_SOCKET,socket.SO_RCVBUF,BUFF_SIZE)
host_name = socket.gethostname()
host_ip = '192.168.43.210'
print(host_ip)
port = 80
socket_address = (host_ip,port)
server_socket.bind(socket_address)
print('Listening at:',socket_address)
```

Hình 1. UDP socket của server.

- Hàm "generate_video()" sử dụng thư viện cv2 và thư viện imutils để thao tác với video. Đọc các khung hình từ camera và đặt chúng vào hàng đợi.

```
def generate_video():
    WIDTH=400
    while(vid.isOpened()):
        try:
            _,frame = vid.read()
            frame = imutils.resize(frame,width=WIDTH)
            q.put(frame)
        except:
            os._exit(1)
        time.sleep(0.001)
    print('Player closed')
    vid.release()
```

Hình 2. Hàm generate_video() của server.

- Hàm “send_video()” để gửi video qua socket UDP cho client

  - Khởi tạo các biến theo dõi và tính toán tốc độ FPS (khung hình / giây).

  - Sử dụng thư viện cv2 tạo cửa sổ để hiển thị video.

  - Nhận tin nhắn và địa chỉ client từ socket máy chủ.

  - Đặt chiều rộng của khung hình là 400.

  - Cho 1 vòng lặp vô hạn để gửi hình ảnh sang client và phát hình ảnh.

    + Lấy một khung hình từ hàng đợi.

    + Mã hóa khung hình thành một định dạng hình ảnh nén JPEG.

    + Mã hóa dữ liệu đã nén thành Base64 để truyền đi.

    + Gửi dữ liệu đã mã hóa đến client sử dụng giao thức UDP.

    + Kiểm tra số khung hình đã gửi và cập nhật FPS bằng cách chia số khung hình cho thời gian từ lúc bắt đầu đến hiện tại.

    + Hiển thị chỉ số FPS lên video.

    + Thiết lập thời gian ngủ 0.01 giây giữa các vòng lặp để giảm tải CPU và giữ cho FPS ổn định.

```
def send_video():
    fps,st,frames_to_count,cnt = (0,0,20,0)
    cv2.namedWindow('SERVER TRANSMITTING VIDEO')
    cv2.moveWindow('SERVER TRANSMITTING VIDEO', 400,30)
    msg,client_addr = server_socket.recvfrom(BUFF_SIZE)
    print('connection from ',client_addr)
    WIDTH=400
    while(True):
        frame = q.get()
        encoded,buffer = cv2.imencode('.jpeg',frame,[cv2.IMWRITE_JPEG_QUALITY,80])
        message = base64.b64encode(buffer)
        server_socket.sendto(message,client_addr)
        frame = cv2.putText(frame,'FPS: '+str(round(fps,1)),(10,40),
                            cv2.FONT_HERSHEY_SIMPLEX,0.7,(0,0,255),2)

        if cnt == frames_to_count:
            try:
                fps = round(frames_to_count/(time.time()-st),1)
                st=time.time()
                cnt=0
            except:
                pass
        cnt+=1
        cv2.imshow('SERVER TRANSMITTING VIDEO', frame)
        key = cv2.waitKey(1) & 0xFF
        if key == ord('q'):
            os._exit(1)
        time.sleep(0.01)
```

Hình 3. Hàm send_video() của server.

- Hàm “get_video()” để nhận video từ client qua socket UDP

  - Sử dụng thư viện cv2 tạo cửa sổ để hiển thị video

  - Khởi tạo các biến theo dõi và tính toán tốc độ FPS (khung hình / giây).

  - Khởi tạo lại biến BUFF_SIZE = 65536

  - Khởi tạo lại socket UDP cho việc nhận dữ liệu và thiết lập tùy chọn socket là một hằng số chỉ định kích thước buffer với kích thước BUFF_SIZE.

  - Giữ nguyên địa chỉ nối kết, chỉ thay đổi cổng dịch vụ khác để tránh xảy ra tình trạng đụng độ.

  - Cho 1 vòng lặp vô hạn để nhận hình ảnh từ client và phát hình ảnh.

    + Dùng hàm recvfrom() Nhận gói tin từ client với kích thước BUF_SIZE.

    + Giải mã dữ liệu từ Base64.

    + Chuyển đổi dữ liệu ở dạng chuỗi thành dạng mảng numpy có kiểu dữ liệu là uint8.

    + Sử dụng cv2.imdecode() để giải mã mảng numpy thành khung hình ảnh.

    + Kiểm tra số khung hình đã nhận đủ chưa, nếu đã đủ thì cập nhật FPS bằng cách chia số khung hình cho thời gian từ lúc bắt đầu đến hiện tại.

    + Hiển thị chỉ số FPS lên video.

    + Nếu người dùng nhấn phím 'q', thì thoát khỏi vòng lặp và kết thúc chương trình.

    + Thời gian ngủ 0.01 giây giữa các vòng lặp để giảm tải CPU và giữ cho FPS ổn định.

  - Kết thúc: Thoát chương trình.

```python
def get_video():
    cv2.namedWindow('SERVER RECEIVING VIDEO')
    cv2.moveWindow('SERVER RECEIVING VIDEO', 400,360)
    fps,st,frames_to_count,cnt = (0,0,20,0)
    BUFF_SIZE = 65536
    server_socket = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
    server_socket.setsockopt(socket.SOL_SOCKET,socket.SO_RCVBUF,BUFF_SIZE)
    socket_address = (host_ip,port-3)
    server_socket.bind(socket_address)

    while True:
        packet,_ = server_socket.recvfrom(BUFF_SIZE)
        data = base64.b64decode(packet,' /')
        npdata = np.frombuffer(data,dtype=np.uint8)

        frame = cv2.imdecode(npdata,1)
        frame = cv2.putText(frame,'FPS: '+str(fps),(10,40),
                            cv2.FONT_HERSHEY_SIMPLEX,0.7,(0,0,255),2)
        cv2.imshow("SERVER RECEIVING VIDEO",frame)
        key = cv2.waitKey(1) & 0xFF

        if key == ord('q'):
            server_socket.close()
            break

        if cnt == frames_to_count:
            try:
                fps = round(frames_to_count/(time.time()-st),1)
                st=time.time()
                cnt=0
            except:
                pass
        cnt+=1
        time.sleep(0.001)
    server_socket.close()
    cv2.destroyAllWindows()
```

Hình 4. Hàm get_video() của server.

- Tạo và khởi chạy các luồng

```
t3 = threading.Thread(target=generate_video, args=())
t4 = threading.Thread(target=send_video, args=())
t5 = threading.Thread(target=get_video, args=())
t3.start()
t4.start()
t5.start()
```

3.6.6. Tiến hành lập trình máy khách (client)

- Khởi tạo các thư viện cần thiết:

  import cv2, imutils, socket, numpy as np

  import time, queue, os, base64, threading, pickle, struct

- Khởi tạo queue để lưu trữ và chia sẻ dữ liệu hình ảnh

  q = queue.Queue(maxsize=10)

- Sử dụng thư viện cv2 khởi tạo một đối tượng VideoCapture để nhận video đầu vào từ camera.

  vid = cv2.VideoCapture(0)

- Khởi tạo một biến BUFF_SIZE = 65536 dùng để nhận dữ liệu từ socket.
- Khởi tạo socket UDP, thiết lập tùy chọn socket là một hằng số chỉ định kích thước buffer với kích thước BUFF_SIZE.

```
client_socket = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
client_socket.setsockopt(socket.SOL_SOCKET,socket.SO_RCVBUF,BUFF_SIZE)
host_name = socket.gethostname()
host_ip = '192.168.43.210'
print(host_ip)
port = 80
```

Hình 5. UDP socket của client.

- Hàm "generate_video()" sử dụng thư viện cv2 và thư viện imutils để thao tác với video. Đọc các khung hình từ camera và đặt chúng vào hàng đợi.

```
def generate_video():
    WIDTH=400
    while(vid.isOpened()):
        try:
            _,frame = vid.read()
            frame = imutils.resize(frame,width=WIDTH)
            q.put(frame)
        except:
            os._exit(1)
        time.sleep(0.001)
    print('Player closed')
    vid.release()
```

Hình 6. Hàm generate_video() của client.

- Hàm “get_video()” để nhận video từ server qua socket UDP

  - Sử dụng thư viện cv2 tạo cửa sổ để hiển thị video
  - Khởi tạo các biến theo dõi và tính toán tốc độ FPS (khung hình /giây).
  - Sử dụng hàm sendto() gửi 1 tin nhắn có nội dung “Hello” tới server.
  - Khởi tạo vòng lặp vô hạn để nhận các gói tin từ server.

    + Dùng hàm recvfrom() Nhận gói tin từ server với kích thước BUF_SIZE.

    + Giải mã dữ liệu từ Base64.

    + Chuyển đổi dữ liệu ở dạng chuỗi thành dạng mảng numpy có kiểu dữ liệu là uint8.

    + Sử dụng cv2.imdecode() để giải mã mảng numpy thành khung hình ảnh.

    + Kiểm tra số khung hình đã nhận đủ chưa, nếu đã đủ thì cập nhật FPS bằng cách chia số khung hình cho thời gian từ lúc bắt đầu đến hiện tại.

    + Hiển thị chỉ số FPS lên video.

    + Nếu người dùng nhấn phím 'q', thì thoát khỏi vòng lặp và kết thúc chương trình.

    + Thời gian ngủ 0.01 giây giữa các vòng lặp để giảm tải CPU và giữ cho FPS ổn định.

  - Kết thúc: Thoát chương trình.

```
def get_video():

    cv2.namedWindow('CLIENT RECEIVING VIDEO')
    cv2.moveWindow('CLIENT RECEIVING VIDEO', 10,360)
    fps,st,frames_to_count,cnt = (0,0,20,0)
    message = b'Hello'
    client_socket.sendto(message,(host_ip,port))
    while True:
        packet,_ = client_socket.recvfrom(BUFF_SIZE)
        data = base64.b64decode(packet,' /')
        npdata = np.frombuffer(data,dtype=np.uint8)

        frame = cv2.imdecode(npdata,1)
        frame = cv2.putText(frame,'FPS: '+str(fps),(10,40),
                            cv2.FONT_HERSHEY_SIMPLEX,0.7,(0,0,255),2)
        cv2.imshow("CLIENT RECEIVING VIDEO",frame)
        key = cv2.waitKey(1) & 0xFF

        if key == ord('q'):
            client_socket.close()
            os._exit(1)

        if cnt == frames_to_count:
            try:
                fps = round(frames_to_count/(time.time()-st),1)
                st=time.time()
                cnt=0
            except:
                pass
        cnt+=1
        time.sleep(0.001)

    client_socket.close()
    cv2.destroyAllWindows()
```

Hình 7. Hàm get_video() của client.

- Hàm “send_video()” để gửi video qua socket UDP cho server

  - Đặt lại địa chỉ socket, trong đó giữ nguyên địa chỉ nối kết, thay đổi cổng giống với cổng hàm “get_video” của server.
  - Khởi tạo các biến theo dõi và tính toán tốc độ FPS (khung hình /giây).
  - Sử dụng thư viện cv2 tạo cửa sổ để hiển thị video.
  - Đặt chiều rộng của khung hình là 400.
  - Cho 1 vòng lặp vô hạn để gửi hình ảnh sang server và phát hình ảnh.
    + Lấy một khung hình từ hàng đợi.
    + Mã hóa khung hình thành định dạng hình ảnh nén JPEG.
    + Mã hóa dữ liệu đã nén thành Base64 để truyền đi.
    + Gửi dữ liệu đã mã hóa đến server sử dụng giao thức UDP.
    + Kiểm tra số khung hình đã gửi và cập nhật FPS bằng cách chia số khung hình cho thời gian từ lúc bắt đầu đến hiện tại.
    + Hiển thị chỉ số FPS lên video.
    + Thiết lập thời gian ngủ 0.01 giây giữa các vòng lặp để giảm tải CPU và giữ cho FPS ổn định.

```python
def send_video():
    socket_address = (host_ip,port-3)
    print('server listening at',socket_address)

    fps,st,frames_to_count,cnt = (0,0,20,0)
    cv2.namedWindow('CLIENT TRANSMITTING VIDEO')
    cv2.moveWindow('CLIENT TRANSMITTING VIDEO', 10,30)

    while True:
        WIDTH=400
        while(True):
            frame = q.get()
            encoded,buffer = cv2.imencode('.jpeg',frame,[cv2.IMWRITE_JPEG_QUALITY,80])
            message = base64.b64encode(buffer)
            client_socket.sendto(message,socket_address)
            frame = cv2.putText(frame,'FPS: '+str(round(fps,1)),(10,40),
                                cv2.FONT_HERSHEY_SIMPLEX,0.7,(0,0,255),2)

            if cnt == frames_to_count:
                try:
                    fps = round(frames_to_count/(time.time()-st),1)
                    st=time.time()
                    cnt=0
                except:
                    pass
            cnt+=1

            cv2.imshow('CLIENT TRANSMITTING VIDEO', frame)
            key = cv2.waitKey(1) & 0xFF
            if key == ord('q'):
                os._exit(1)
            time.sleep(0.001)
```

Hình 8. Hàm send_video của client.

- Tạo và khởi chạy các luồng

```
t3 = threading.Thread(target=get_video, args=())
t4 = threading.Thread(target=send_video, args=())
t5 = threading.Thread(target=generate_video, args=())
t3.start()
t4.start()
t5.start()
```

CHƯƠNG 4: KẾT QUẢ THỰC NGHIỆM

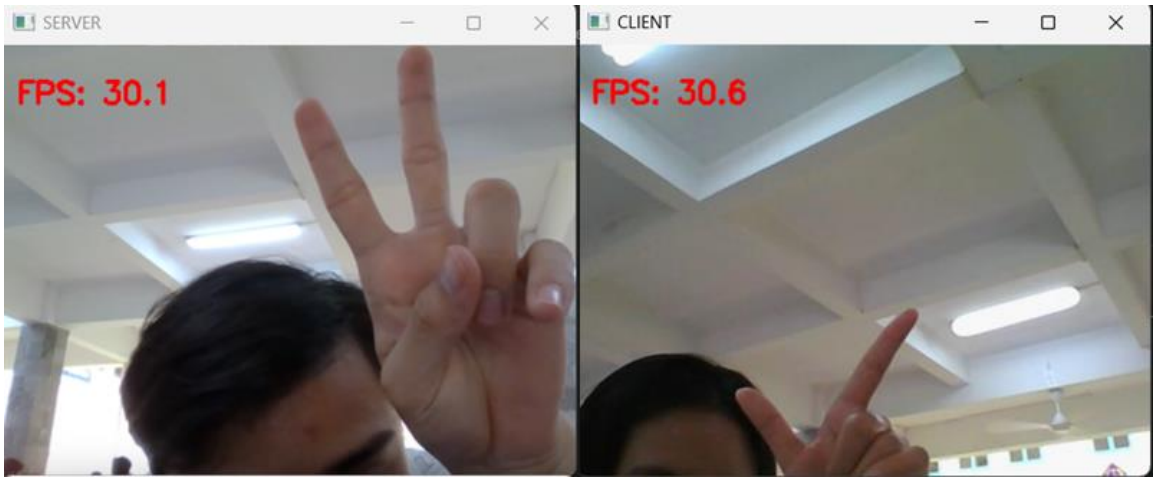
Em đã tiến hành sử dụng các phương pháp và công cụ được mô tả trong chương 3. Các kết quả sau đây là sự kết hợp của nhiều lần thực hiện để đảm bảo sự đáng tin cậy và khách quan.

4.1. Kết quả tốc độ truyền dữ liệu

Tốc độ truyền và nhận dữ liệu được thực nghiệm trong nhiều tình huống sử dụng các kỹ thuật cân bằng tốc độ đường truyền khác nhau. Kết quả cho thấy rằng việc cân bằng tốc độ đường truyền có thể cải thiện đáng kể tốc độ truyền dữ liệu trong các cuộc họp trực tuyến. Kỹ thuật này giúp đảm bảo cho người dùng có trải nghiệm mượt mà hơn, hạn chế việc bị gián đoạn.

4.2. Kết quả chất lượng hình ảnh

Em đã tiến hành đánh giá chất lượng hình ảnh trong các thực nghiệm sử dụng các kỹ thuật cân bằng tốc độ đường truyền khác nhau. Kết quả thu được là cân bằng tốc độ đường truyền có thể cải thiện chất lượng hình ảnh đối với người dùng. Đặc biệt, trong các trường hợp có sự giảm tốc độ truyền dữ liệu, việc cân bằng tốc độ đường truyền có thể giảm hiện tượng giật hình và ngắt quãng. Chỉ số đo tốc độ (FPS) thu được xấp xỉ tương đồng giữa truyền và nhận hình ảnh thông qua mạng LAN.



Hình 9. Kết quả đạt được

4.3. Đánh giá chất lượng của hệ thống

- Sự ổn định: Cân bằng tốc độ đường truyền đã giúp tạo ra sự ổn định trong kết nối Internet, giảm thiểu sự ngắt quãng gián đoạn.
- Tăng Hiệu Suất**:** Sự cải thiện trong tốc độ truyền dữ liệu đã tạo điều kiện cho việc sử dụng ứng dụng và dịch vụ trực tuyến tốt hơn, bao gồm gọi video, và cuộc họp trực tuyến.

4.4. Hạn chế

Hệ thống vẫn chưa thể hoàn thiện về mặt giao diện, truyền âm thanh. Chất lượng và tốc độ truyền hình ảnh được cải thiện đáng kể nhưng còn phụ thuộc vào đường truyền mạng.

CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1. Kết luận

Trong quá trình nghiên cứu , tiến hành xây dựng và thực nghiệm hệ thống truyền âm thanh và hình ảnh trên Internet, em đã đạt được các kết quả quan trọng và có những kết luận về cách cân bằng tốc độ đường truyền có thể cải thiện chất lượng dịch vụ và trải nghiệm của người dùng. Các điểm kết luận bao gồm:

- Cải thiện chất lượng hình ảnh: Việc cân bằng tốc độ đường truyền có cải thiện đáng kể chất lượng hình ảnh trong hệ thống truyền âm thanh và hình ảnh trực tuyến, có giảm hiện tượng giật hình và gián đoạn.
- Tăng hiệu suất: Cân bằng tốc độ đường truyền cũng đã giúp tăng hiệu suất của hệ thống trực tuyến, giúp người dùng có một trải nghiệm tốt hơn.

5.2. Hướng phát triển

Mặc dù hệ thống đã đạt được kết quả tích cực, nhưng còn nhiều thiếu sót, sau đây là một số hướng phát triển tiềm năng cho tương lai:

- Tích hợp các giao thức mạng mới: Thêm các phương pháp cân bằng tốc độ truyền dữ liệu sử dụng các giao thức mạng mới có thể tối ưu hóa hiệu suất và khả năng cân bằng tốc độ trong thời gian thực.
- Tăng khả năng bảo mật cho hệ thống: Nghiên cứu và phát triển các giải pháp an toàn để bảo vệ dữ liệu và thông tin cá nhân của người dùng trong quá trình truyền dữ liệu.
- Mở rộng ứng dụng: Nâng cấp giao diện và mạng nối kết để áp dụng rộng rãi hơn, bao gồm các lĩnh vực như giáo dục, công nghiệp, nông nghiệp và y tế.

Tài liệu tham khảo:

1. Youtube
2. https://vi.wikipedia.org/wiki/UDP
3. https://www.cit.ctu.edu.vn/~dtnghi/oss/python3
4. https://codelearn.io/sharing/top-30-libraries-packages-4-beginner-p1