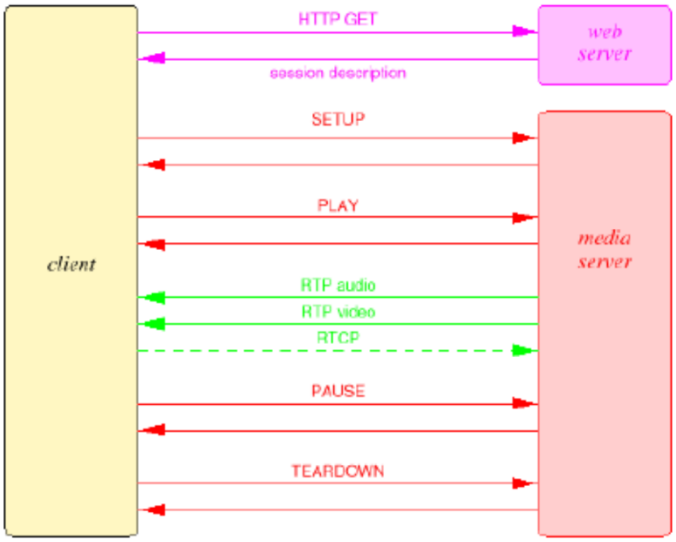
電腦網路導論_期末報告

伏宇寬 B07202024
王品壬 B07202026
何景盛 B08901123

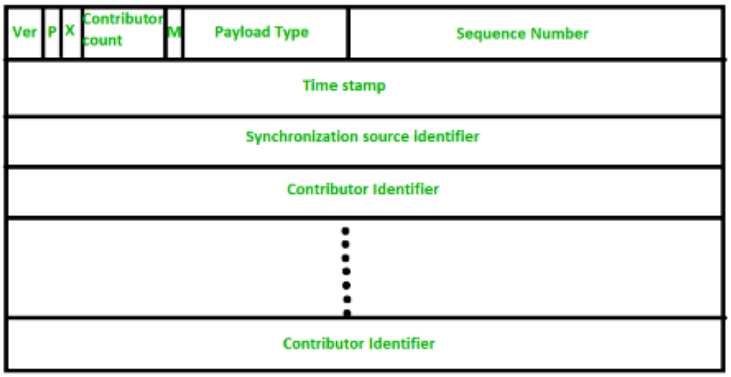
前言：

本次報告，我們目的是實現 video streaming，需要用到 Real-time Transport Protocol(RTP)和 Real Time Streaming Protocol(RTSP)這兩個 protocol 來架設 media server，除此之外，我們還需要用 HyperText Transfer Protocol(HTTP)來架設 web server。而需實作的結構如下圖。



RTP：

RTP 是建基於 UDP 上，主要用來傳輸多媒體，RTP 協議詳細地規劃好 video 和 audio 在 internet 的傳輸格式，RTP packet format 如下圖。



RTSP：

RTSP 是在網絡的應用層，主是用來控制 media server 的 protocol，它默認的端口是 544。而一些 request 和 response 都是跟 HTTP 相似。在這次 project 中，我們需要用到以下這四個 commands：SETUP、PLAY、PAUSE 和 TEARDOWN。

SETUP：

Request：需要在 PLAY request 之前發送，而 SETUP request 主要是詢問如何傳輸媒體流。 Response：Server 端會在 SETUP response 內加入一個 session 的參數送回 Client 端。

PLAY：

Request：向 server 提出播放媒體流的請求。

Response：則回應 Client 端的請求，把媒體流送到 Client 端。

PAUSE：

Request：向 server 提出暫停媒體流的請求。

Response：則回應 Client 端的請求，暫停媒體流送到 Client 端。

TEARDOWN：

Request：向 server 提出中止媒體流的請求。

Response：則釋放這個 session 的所有東西。

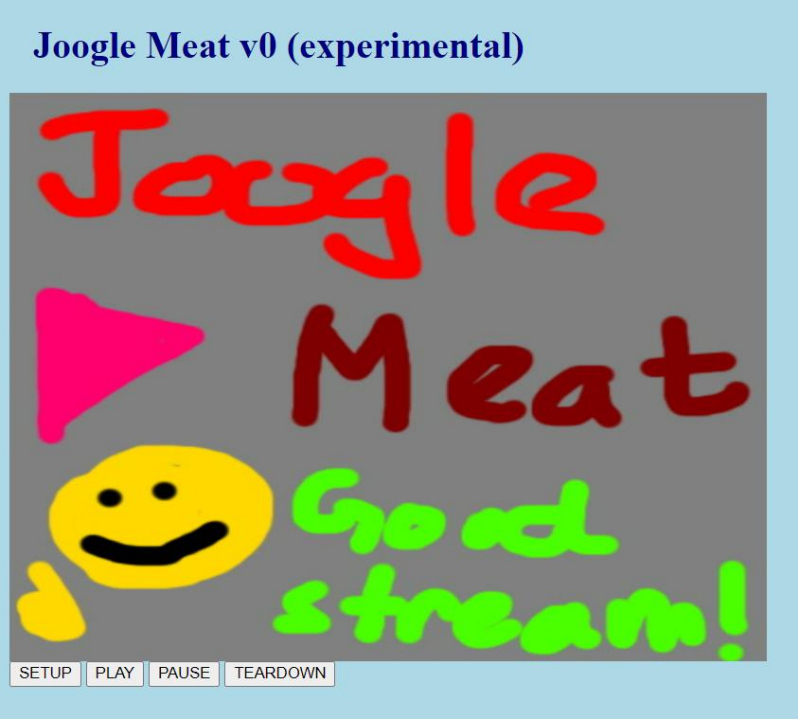
HTTP：

HTTP 也是網絡的應用層，可以用來發布和接收 HTML 的文件，從而可以在 Browser 上顯示，默認端口為 80，主要用 GET 和 POST 這兩個動作來顯示和提交數據。

Streaming 流程:

與 web server 建立 http 連線，接著當我們按下 setup，會與 media server 建立 RTSP 連線，按下 play 之後 media server 開始建立 RTP 封包，並把封包傳送至 Client 端，Client 收到 RTP 封包之後，將裡面 media data decode 出來，並播放影音出來。當按下 pause，client 會建立一個 RTSP 封包，告知 media server 現在要暫停傳送 RTP 封包，但 RTSP 連線依舊存在。若使用者按下 TEARDOWN，則 RTSP 連線被關閉，client 與 media server 將無法繼續溝通。

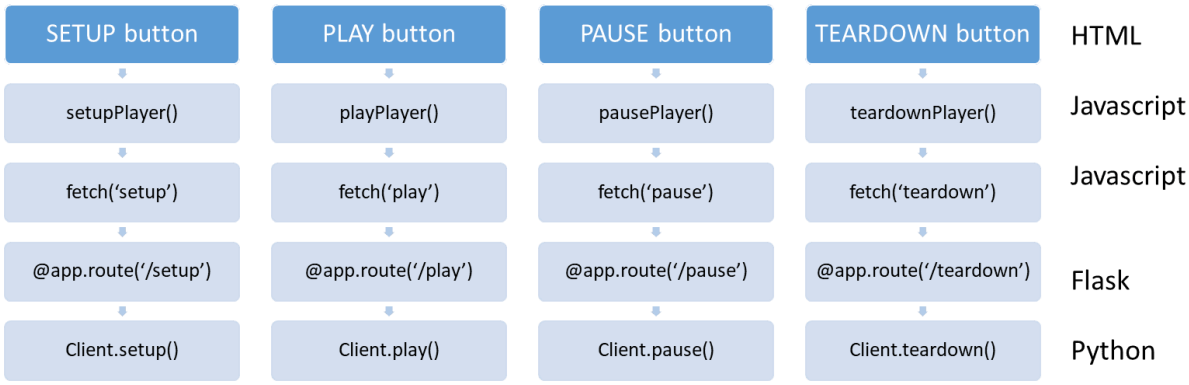
使用說明：



程式部分：

我們使用的主要語言為 python，以及少數的 html 和 javascript。網頁方面，我們使用 python 的 flask 模組來顯示網頁，而可以透過網頁下方四個按鈕(SETUP, PLAY, PAUSE, TEARDOWN)透過 javascript 呼叫 python 的函式，傳送 RTSP 給 server。

controls.js 中，有四個函式，分別是 setup(), play(), pause()以及 teardown()，當按鈕被按下第一時間會觸發這些函式。但是因為我們希望可以透過 python 去處理這些 input，就使用 fetch()這個方式去生產一個 HTTP 請求。當 flask 收到此請求時，就能夠在 flask 中進行 client 這個 class 的 setup(), play(), pause(), teardown()。



可以透過流程圖發現，從 HTML 的按鈕要能夠回饋到 client 的過程有點冗長，但給予我們在瀏覽器中即時的反應。Client 本身可以分為兩個 class，一個處理 video 一個處理 audio。Client 主要扮演著接收以及顯示 stream 的功能，而 flask 則是 client 與瀏覽器的橋樑。

當 server 接收到 RTSP 請求以後，會做相對的回應以及傳送資料給 client 的 UDP socket。Client 本身會有兩個 socket，個別接受 video 與 audio 兩個 stream。接受以後，video 部分會不斷更新 html 檔中的圖片，如同影片一樣；audio 部分則以 pyaudio 模組的方式直接在 python script 裡面播放。

Server 可以使用 cv2 以及 pyaudio 的模組去接受電腦鏡頭與麥克風的收入，傳給 client。若選擇 stream 一般的影片檔，就會切成很多 frame 傳送到 client 端。

socket methods：socket(socket.AF_INET, socket.SOCK_STREAM)：建立 TCP 連結。

bind((address, port))：綁定地址和端口。
listen(value)：等待連線，value 為連線個數。
accept()：接受連接。
connect()：與 server 端連線。
send(data)：發送資料，data 為資料。
recv(data)：接收資料，data 為資料。

cv2 methods：VideoCapture(filename)：捕抓 filename，如果 filename 是 mp4，則以 mp4 的方式開啟， 如果 filename 是 0，則開啟 webcam。

get(cv2.CAP_PROP_FPS)：獲取影片的 fps。
read()：讀取影片一幀的資料並 return。
resize(frame, (width, height))：調整這一幀的大小。
imencode(“.jpg”, frame)：把圖片資料解碼成 jpg。

audio methods：pyaudio.PyAudio()：設置 PyAudio。

open(format, channel, rate, output)：用 PyAudio 打開音頻檔，並規泛什麼樣的格式。
write(data)：寫入音頻的採樣。
read(data)：讀出音頻的採樣。

我們把 project 分成不同的 module 來執行，audio_client.py, video_client.py, main.py, RtpPacket.py, Server.py, ServerWorker.py, VideoStream.py

audio_client.py, video_client.py：內部 method 主要以處理從 server 端傳送來的資料。
main.py：以建立 socket 和調用兩個 client.py 模組，也包含 flask 來進行網頁設置。
RtpPacket.py：把資料處理成 RTP packet 的格式，可以把資料 encode 成 RTP 封包，也可以把資料從 RTP 封包 decode 出來。
Server.py：建立 RTSP socket 以及等待建立 RTSP 連線，並在建立連線後調用 ServerWorker.py。
ServerWorker.py：內部 method 以將需要 streaming 的資料打包成 RTP packet 並傳送到 Client 端。
VideoStream.py：主要以處理多媒體，例如 mp4、webcam 和 microphone。

由於一般 wav 的 sample rate 可能會到 44100Hz，因此在傳送的時候會需要傳輸許多資料，對於 streaming 而言並不合適，容易造成 congestion，因此我們先將影片的音訊取出來，並降低他的 sample rate 到 8000Hz，使得傳輸變得比較順暢。

Demo:

影片連結:

https://drive.google.com/file/d/1wZ4WyhUqn9AnQQ3jl_XUYWatYaC_g8Ee/view?usp=sharing

程式碼:

Github: peterwang0817/icn-final-project: 110-1 電腦網路導論期末專題 (github.com)

總結及心得：

在本次報告中，我們更清楚明白 video 和 audio 是如何構成的，就例如 video，一直以為 mp4 就是 video 的本身，但原來 mp4、mov 和 avi 這些檔是 video 的格式，只是一個容器，真正 video 的內容是一些編碼，就如現時普遍的 h.264 等等這編碼標準，同時也有 h.263、h.265 和 vp8 等等，而這些編碼同時也可以壓縮檔大小，因為它會記錄下一些關鍵幀，從而用算法去補充中間的幀數，這樣一來可以減低傳輸時的負荷。

參考資料：

RTP、RTSP concept reference:

https://codertw.com/%E4%BA%BA%E5%B7%A5%E6%99%BA%E6%85%A7/134652/

python socket reference:

https://ithelp.ithome.com.tw/articles/10205819

pyaudio reference:

http://people.csail.mit.edu/hubert/pyaudio/docs/#pyaudio.Stream.write

RTP、RTSP reference:

https://www.pearson.com/us/higher-education/product/Kurose-Computer-Networking-A-Top-Down-Approach-7th-Edition/9780133594140.html?tab=overview&fbclid=IwAR1nNCDxdylclYqty4_IFMKuwqH9MF4FCoUe_baTxne88CCu5kKqDrf5d8o

audio streaming reference:

https://pyshine.com/How-to-send-audio-video-of-MP4-using-sockets-in-Python/?fbclid=IwAR0nkbb_sWnqQQgrTcC1mcdXAu_8Xv4JF9rtxF4UskPHBKAgs4dhRM_IG6E