Report

Walkie-Talkie

第五組

B06705059 魏任擇
B06902033 黃奕鈞
B06902039 賈本晧
B06902069 許博翔
B06902111 林慶珠
B06902128 鄭力誠

Computer Network Laboratory

目錄

一、摘要

本實驗透過python語言，結合socket programming，實作walkie-talkie的功能，讓若干個使用者可以同時在線，並且只有一位使用者可以透過麥克風將訊息傳播給其他使用者。訊息的封包只限在內網傳遞，確保接收者為特定對象。

關鍵字：Walkie-talkie、Socket、Multi-threading、push to talk

二、前言

常常看到保全人員拿著無線電通訊，通話方便且可以同時跟多人通訊，但礙於設備問題，無線電並非像智慧型手機一般，普遍率已達「人手一機」。

為了使無線電的優勢更為普及，而不需要另外買無線電機組，我們想用手機來實作無線電的功能，讓大家在不需要無線電機組的情況下，可以像是使用無線電般與其他人溝通。

透過這個實驗，我們可以更加熟悉socket以及multi-threading的控制，並嘗試讓資料能在TCP連線上即時同步。

三、實驗器材

- AP 一台(使用手機熱點分享也可)
- 電腦 (server) 一台
- 電腦 (client) 兩台

四、實驗方法與過程

首先，將一台電腦連接AP(或是手機開啟的熱點，不需要有行動網路)作為server，並建立socket。用戶端只需開起Wi-Fi並連上AP，開啟使用者介面並且註冊登入，按下錄音就可以將錄音內容傳遞給其他用戶，其他用戶同樣也可以傳遞語音回來，以此達成通訊的目的。

以下我們將分項詳細介紹每個環節：

- **使用者介面：**

  使用tkinter套件。頁面分為登入首頁、註冊頁、以及實際應用的頁面，在登入首頁會有輸入帳號密碼的entry，幫助程式得到input。並且有註冊和登入的按鈕，在按下後觸發其他工作。註冊頁也是相同道理，會有帳號、密碼、密碼確認的entry，以及確定註冊、返回的按鈕。實際應用的頁面則會有一個錄音按鈕，模擬使用對講機要按著按鈕講話，下方還有一個登出紐，方便使用者離開應用。

- **Client與Server連線處理：**

  Server會在指定的IP和port建立socket等待Client，當Server的socket有人要connect時(檢測有沒有人要求連線時用select)，server就會開一個thread來負責處理這個client的各種訊息溝通，原本server的thread就會繼續等待其他client進行連線，以達到不同thread進行不同的分工。

- **使用者登錄與註冊處理：**

  主要以json檔的方式，在server端儲存用戶的資料，包含名稱與密碼。而使用者登入時，會將輸入的帳號與密碼傳送給server，server收到後會比對Database中的用戶資料，將結果回傳給client，client收到的訊息如果是正確就會成功登入，並跳轉到使用畫面，反之，失敗就會顯示帳密錯誤。

至於註冊的部分，client會將要註冊的帳號密碼傳送給server，server比對Database後，只要沒有相同的使用者名稱 (即已註冊過的)，就會回傳正確，client收到後就能確認自己註冊成功了，另外，如果失敗的話 (帳號已被其他人使用)，client還是能夠繼續註冊，直到成功為止。

- **麥克風使用權處理：**

在server中設定一個global的變數，每個client在要求麥克風權限時，負責處理的thread會先查看該變數是否被起用(True)，如果已被啟用則代表此時有人在使用麥克風，則回傳拒絕的訊息給client，並讓他稍後再要求；如果該變數未被啟用，則將該變數設成True(此項操作為atomic)，直到client放開麥克風的按鍵時，負責處理的thread才會將他設回False，留待其他人再度要求。

- **語音訊息處理：**

利用sounddevice套件來進行錄音，錄音後sounddevice的rec()函數會回傳一個numpy array，這個numpy array可以傳進sounddevice的play()函數來放出聲音。

在得到麥克風使用權之後，程式會進入一個錄音迴圈直到使用者放開錄音的按鈕，在這個迴圈裡，每一次會錄製一小段的聲音，並加到一個等待傳送的queue中。另外在按下拿到麥克風使用權時，會建立一個新的thread，這個thread會將上述的queue中的錄音片段依序傳給server。

至於傳送的部分，首先會先把上述錄音片段的numpy array轉成pickle，client端會先傳一個錄音片段的byte數給server，讓server知道接下來要接收多少bytes，避免說話結束後傳送的"quit"訊息和錄音片段被一起收到並被當成錄音的一部分，收到byte數量之後server會傳送一個"start"的訊息，告訴client可以把錄音片段傳過去。

當server收到正確的錄音片段後，也會傳送"done"的訊息，這些訊息都能夠幫助資料正確送達對方，接著server會用跟client相同的方式，先傳送錄音片

段byte數量給其他clients，收到"start"之後把錄音片段broadcast出去，並在clients收到正確錄音片段後接收到"done"。

最後，接收方的clients會將收到的pickle轉換回numpy array，並且放進sounddevice.play()將語音訊息播出。

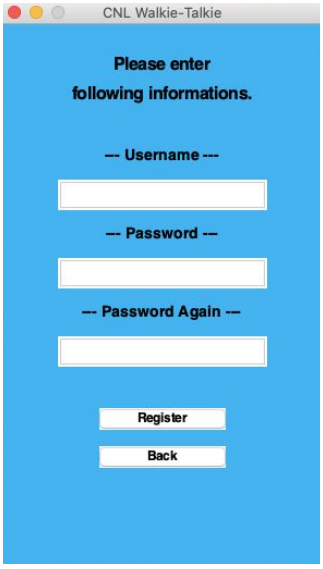
五、實驗結果

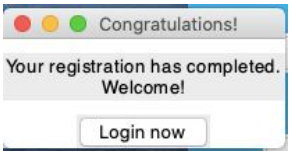
作為server端的電腦先連接AP，指定IP位置和port號碼，並建立socket等待client連線。作為client端的電腦只需開起Wi-Fi，指定IP位置和port號碼，即可連接到server。

- **使用者登入介面**

使用者可以透過已註冊的帳號密碼，登入使用無線電；如果沒有，也可以按"Register”按鈕，連到註冊帳號介面，進行註冊。輸入的帳號或密碼有誤，則會跳出Error的視窗提示使用者。

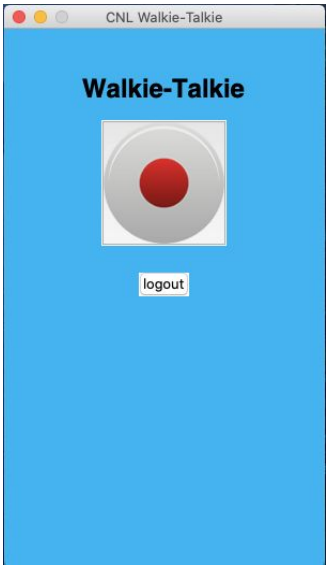
- **註冊帳號介面**





使用者可以輸入想要的帳號密碼，進行註冊。同時，我們的系統會確認使用者是否輸入未知字元、帳號已存在或輸入的兩次密碼不同，若沒有錯誤，則代表註冊成功。

- **通話介面**



這個介面就像是一般的無線電，只有一個通話按鈕。如果使用者按通話按鈕時，有其他使用者正在使用麥克風，系統會自動跳出訊息，通知該使用者有其他人在通話。另外設有登出按鈕可以登出。

六、實驗討論與未來展望

(一) 實驗討論

1. Client和Server之間的訊息分不清楚：

在兩端之間傳送的訊息可能是使用者加入\退出的訊息、使用\放棄麥克風使用權的訊息...等系統訊息，或是使用者要傳送語音訊息，只有單獨一個socket可能會讓語音訊息無法與其他訊息分開來處理。

解決方法：

一個client需要創建兩個socket，一個處裡上段說的系統訊息，另一個專門處理語音訊息。持有麥克風使用權的使用者，他的語音訊息才會傳給其他使用者。

2. 函式命名方式不統一：

因為大家都有參與coding的部分，因為使用習慣不同，在各class使用的函式名稱不盡相同，使各檔案在融合時，閱讀和除蟲的效率大幅降低。

解決方法：

我們的coding style該加強，並且應該在各自寫的時候，在取函式名稱上達到共識。

3. 錄音下來的聲音斷續且不清楚：

我們發現還不用到傳送錄音片段的部分，在本地端錄下的錄音片段連續撥放後就已經是斷斷續續的，原因是sounddevice的rec函式以及send都會拖延時間，造成下一段錄音開始時和前一段的結尾相隔有點久，中間那段時間的聲音就錄不到了。

解決方法：

我們把send的部分移到另外一個thread進行，讓它能和下一段錄音平行運作，並且增長錄音片段的duration，確保錄音片段至少有完整的幾個字，而不是一個字被分到很多個片段，導致無法接上時連字都聽不出來。應該錄製多少duration的片段再傳送，我們嘗試了不同大小的時間參數：

- 0.1秒～0.5秒：

  接收端會很快收到聲音，但因為錄製時間過短，麥克風可能尚未成功啟動，因此常發生只有雜音，或是說話聲音斷斷續續，造成聽者使用體驗不佳。

- 3秒以上：

  因為每次錄製的時間十分充足，聽者可以很清楚了解聲音內容，但代價是等待時間過長，換句話說，說話者講完至少需要3秒之後，聽者才會接收到內容，如果兩邊需要作出快速的回應的情況可能不適用。

- 1秒～2秒：

  這是我們認為使用體驗最佳的區間，雖然還是需要1秒的等待時間，但是這是在等待時間與語音品質最好的平衡點。

4. 聲音延遲：

承接上面一點，由於我們增加錄音片段的duration，而錄音片段會在錄音結束那一刻才被傳送，代表對方收到聲音時至少會延遲duration的時間，因此延遲的時間顯得較長。

解決方法：

我們認為問題還是在於sounddevice的rec函數執行時間，它可能要去要求硬體進行錄音而來回浪費許多時間，造成我們必須犧牲duration來維持聲音的清晰度，因此若要解決，可能要尋求更快速的package，或是更快速的硬體設備，不過這可能就和網路的部份較無關聯了。

(二) 未來展望

根據結果和討論，我們列出以下若干點，作為日後可以改進或加強的可能方向：

1. 增加重播功能：

由於我們的實作方式是透過AP將訊息進一步傳播給其他使用者，也就是說我們可以在server端增加紀錄的功能，讓AP保存所有使用者在各時間點的錄音紀錄。日後需要尋找之前的錄音紀錄時，可以在server的紀錄裡找到各音檔，甚至有進一步的相關資料，如：傳送人、傳送時間...等。

2. 導入手機製成app：

我們的目標是希望能讓普遍性較高的智慧型手機，可以結合無線電機組隨說隨傳的便利性，但是礙於程式語言的知識背景，本實驗是運用我們較熟悉的python語言實作。未來若是對於手機的程式語言(如Swift、Java...等)有進一步的學習，我們可以將這次實驗的方法，轉成應用在手機的程式語言，進而達到我們的理想。

3. 即時獲得成員資料與增加聊天室功能：

只要是能連上server並能夠成功登入的使用者，都可以接收或是傳遞語音資料。我們希望能讓server將當時連線的成員顯示在使用者介面，可以讓每個使用者一目了然目前有哪些使用者在線上，甚至是哪位使用者正在發言，進一步與特定的使用者創立聊天室，讓使用者可以與特別選擇的其他用戶聊天。

4. 自動連接上server：

我們目前是透過直接輸入server的IP及port的方式連接入系統，在實際應用上，我們可能部會直接知道server的IP與port，因此未來可以考慮當client開啟後，會向整個subnet內廣播，通知server自己來了，在server接收到時，會回傳自己的資訊，之後兩者就能開始溝通了。

5. 進行聲音資料加密：

目前我們傳聲音都是直接將音源的資料結構整個資料傳過去，這樣會有安全性的問題，如果有人攔截到我們的音源的封包的話，就可以直接透過python對應的套件進行播出。如果考慮到安全性的問題，我想我們未來可以在傳遞封包時進行加密，如此一來有心人士想要竊聽客戶端的資料時，就會難上許多。

6. 加入Mic拿取時間限制：

在某些情況可能會有人一直按住發話，使得其他想講話的人沒辦法拿到存取麥克風的權限，造成starvation的問題，因此若以後有所謂聊天室的功能的話，可以在聊天室上加上限制拿Mic的時間長短這個功能，使得在這一個聊天室上，一人最多持有Mic幾秒這樣，讓其他人有機會發言。

七、參考文獻

1. **socket programming**
   https://www.itread01.com/content/1541826445.html
2. **recording**
   https://codertw.com/%E7%A8%8B%E5%BC%8F%E8%AA%9E%E8%A8%80/491427/
3. **recording**
   https://stackoverflow.com/questions/38878908/play-and-record-sound-using-pyaudio-simultaneously
4. **GUI (tkinter)**
   https://blog.techbridge.cc/2019/09/21/how-to-use-python-tkinter-to-make-gui-app-tutorial/

八、組內分工

學號	姓名	工作
B06705059	魏任擇	server.py
B06902033	黃奕鈞	server與client間的整合、上台報告
B06902039	賈本晧	server.py、final report
B06902069	許博翔	GUI.py、錄音處理
B06902111	林慶珠	GUI.py
B06902128	鄭力誠	client.py