臺北市立大學 資訊科學系

專題報告

指導教授：梁世聰老師

QnA Session Helper --- Modules for seat arrangement and session conrtol

專題生：王若芸 撰

中 華 民 國 １０４ 年 6 月 9 日

摘 要

在一個大型的會議環境，當主持人想和來賓互動麥克風數量又不足時，來賓必須浪費一些時間在等待麥克風的傳遞上，此專題就是為了解決這樣的問題，只要在主持人的電腦上安裝一項應用程式，來賓的手持裝置安裝一項 App，再透過一連串的動作，就能解決麥可風不足的問題，讓會議更加流暢，此系統主要分成三個模組，而本論文主要就座位安排及審核發言申請兩個模組作說明。

Abstract

During a Q-and-A session of a conference organized in a large meeting venue, it takes much time for the session chair to wait conference attendants moving around and delivering a handheld microphone to a remote questioner. In this project, we propose QnA Session Helper as a total solution to solve this problem. With the pre-installations of the server application on the host computer and the client App on each of the handheld devices of the conference attendees, an efficient and well-organized Q-and-A session providing all attendees with their personal microphones can be achieved.This system is consisted of three modules,and this paper mainly illustrates two modules separately,namely the seat arrangememt and the session control.

目錄

圖目錄

表目錄

第一章 簡　介

本章共分成三小節，第一節先介紹研究背景與動機，第二節為研究目的，第三節則是研究環境限制。

1.1 研究背景與動機

會議是人類社會的一種社交、公關、政治、意見交流、訊息傳播及溝通的活動，有鑑於當今會議環境的多元性，會議的流暢度有時會受到會議環境的影響，尤其是大型的會議參與人數眾多，若採用互動問答的方式進行，會議流程的拿捏更顯重要，為了讓所有出席者清楚聽到發言者所說得話，常使用麥克風及擴音器等設備來輔助，但礙於成本的關係，很難達到人手一支麥可風的要求，在這樣的會議環境之下，勢必要浪費些許時間來傳遞麥克風，而本專題便是針對此種會議環境所設計，希望能解決此問題進而提升會議流暢度。

1.2 研究目的

本專題特別針對大型會議環境，欲設計一項工具來解決問答式發言的問題，並提升會議進行的流暢度，透過來賓的手持裝置取代麥克風的功能，此外，提供主持人審核發言的權利，運用事先建置好的會議座位表，讓主持人了解當前發言人所在位置以及提出發言申請人的位置，更能掌控會議流程及來賓狀況。

來賓取得發言權後即可透過手持裝置將聲音傳送到主控台，再藉由主控台之喇叭播放，省去傳遞麥克風之等待時間，主持人亦可控制來賓發言的時間，以免來賓發言過於冗長影響會議流程。

1.3 研究環境限制

因為牽涉到聲音的傳送及座位表的即時更新，此裝置必須在有網路的環境下才能運作，在此 Client 端為來賓的手持裝置，Server 端為主持人的主控台，以下推薦兩種網路環境，第一種為 LAN 環境如圖 1 所示，在此環境之下 Client 端及 Server 端連上同一個 AP，經由 Switch 作封包之收送，而聲音傳送的品質亦會受到 AP 訊號強度及穩定度的影響。

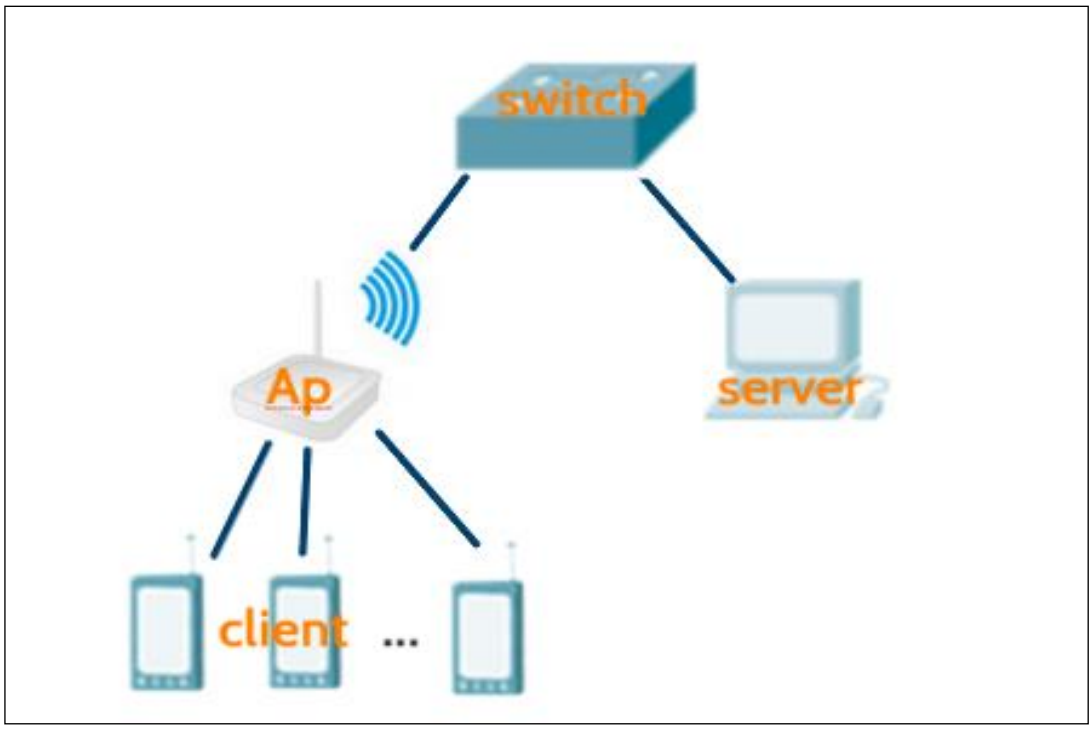


圖 1 LAN 環境

另一種為 Internet 環境如圖 2 所示，在此環境下聲音傳送的品質除了會受到 AP 訊號強度及穩定度的影響之外，某些封包可能會被防火牆擋掉，比較兩種環境後，發現使用 Internet 環境風險較高，所以比較推薦在 LAN 之環境下操作本系統。

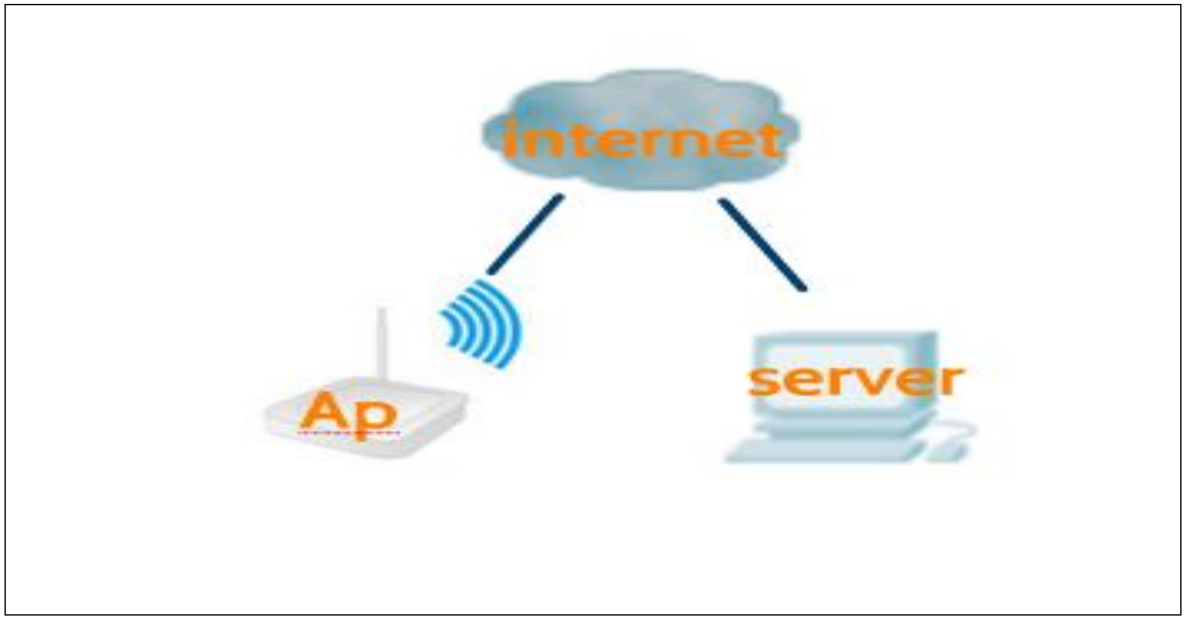


圖 2　Internet 環境

第二章　系統實作與設計

本章共分成三小節，第一節先介紹系統架構，第二節為系統開發與實作，第三節為系統介面展示，第二節又細分成兩個部分來說明，分別為 Client 端及 Server 端。

2.1 系統架構

本系統的架構部分主要分成兩個部分，Server 端及 Client 端，其中 Server 端為主持人的主控台，Client 端為來賓的手持裝置，Server 端包含三個步驟而 Client 端包含四個步驟，我們將這七個步驟組合成三個 Module 如圖 3 所示，各 Module 中之 Client 及 Server 端皆有封包之收送。

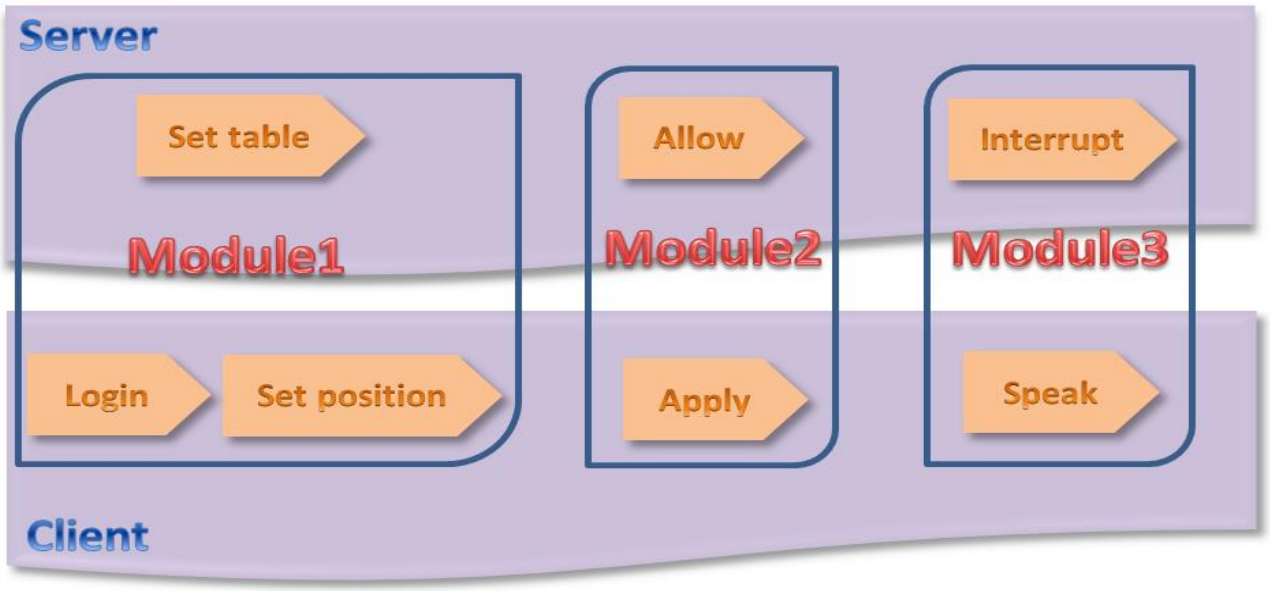


圖 3 系統架構圖

Module1. Server 端需先建置議場座位表，再傳送至 Client 端，Client 端接收到座位表後，來賓再將自己所在位置設置完成，之後傳回 Server 端，而議場座位表也會即時更新，主持人能清楚知道來賓所在位置。

Module2. 當來賓想發言時，需提出發言申請並等待 Server 端的許可，Server 端接收到 Client 端的申請後，可以選擇給予申請者發言權或拒絕給予，當 Client 端得到發言權後，來賓即可發言。

Module3. Server 端有權力控制發言人的發言時間，若發言人的發言過於冗長，占用過多會議時間，主持人可以直接中斷其發言，將其發言權取回，當 Client 端沒有發言權後便無法再發言。

2.2 系統開發與實作

以下就音頻傳送、module1 及 module2 來介紹，本專題之 Server 端為以 java 撰寫之應用程式，而 Client 端則為 android 開發之 App。

(一) 音頻傳送

要進行音頻的傳輸，首先就是要建立數據連結。常用的通訊協議中，TCP 較可靠，所以用在不允許數據丟失的應用上，而 UDP 則較多應用於處理速度要求較快、數據傳輸可靠性要求不是很高的應用上，此專題聲音部分運用 UDP 來傳送，Server 端音頻傳送程式碼如圖 4 所示

表 1 AudioFormat 參數定義

signed	若檔案採用可變位元率(VBR)進行編碼，則為 true
sampleRate	具有此格式的聲音每秒播放及錄製的樣本數
sampleSizeInBit	每個具有此格式的聲音樣本中的位數
channels	使用此格式的音頻信號(單聲道為 1，立體聲為 2)
bigEndian	指示以 big-endian 還是 little-endian 方式儲存音頻數據

其中 AudioFormat 為聲音串流中安排特定數據的類，透過檢查音頻格式存儲的信息，可以得知二進制聲音數據中解釋位元的方式，AudioFormat 中定義了一些參數在 AudioTrack 和 AudioRecord 中用到，如表一所示。

```
618
619     public static void startTalk() {
620         startThread = new Thread(new Runnable() {
621
622             @Override
623             public void run() {
624
625                 byte[] receiveData = new byte[8192];//8192
626                 format = new AudioFormat(sampleRate, 16, 1, true, false);
627                 startThread.setPriority(Thread.MAX_PRIORITY);
628                 while (status == true) {
629
630                     DatagramPacket receivePacket = new DatagramPacket(
631                             receiveData, receiveData.length);
632                     try {
633                         servervo.receive(receivePacket);
634                     } catch (IOException e) {
635                         // TODO Auto-generated catch block
636                         e.printStackTrace();
637                     }
638                     ByteArrayInputStream baiss = new ByteArrayInputStream(
639                             receivePacket.getData());
640                     ais = new AudioInputStream(baiss, format,
641                             receivePacket.getLength());
642                     toSpeaker(receivePacket.getData());
643
644                 }
645             }
646
647         });
648
649         startThread.start();
650
651     }
```

圖 4 Server 端音頻收送

表 2　AudioRecord 參數定義

audioSource	表示採集音頻之來源
sampleRateInHz	音頻採集之頻率，每秒能採樣的次数，頻率越高，音質越好
channelConfig	聲道設置，MONO 為單聲道，STEREO 立體聲
audioFormat	編碼和採樣大小，android 支持的採樣大小 16bit 或者 8bit。當然採樣大小越大，那麼信息量越多，音質也越高
bufferSizeInBytes	採集數據需要的缓冲區的大小，若不知道最小需要的大小可以在 getMinBufferSize()查看

Client 端運用 socket 傳送封包將兩端設定相同 port，連至 Server 端之 Ip 位置，在 android 中採集音頻的 API 為 android.media.AudioRecord，其中定義了一些參數如表 2 所示，採集到的數據保存在一個 byteBuffer 中。

Server 端音頻傳送程式中之 tospeaker function 負責對接收到 DatagramPacket 做處理，其中 FloatControl 為 java 針對聲音所提供之類別，控制各種聲音之音量和 MASTER_GAIN，tospeaker function 程式碼為如圖 5 所示。

```
513    public static void toSpeaker(byte soundbytes[]) {
514        try {
515
516            DataLine.Info dataLineInfo = new DataLine.Info(SourceDataLine.class, format);
517            SourceDataLine sourceDataLine = (SourceDataLine) AudioSystem.getLine(dataLineInfo);
518
519            sourceDataLine.open(format);
520
521            FloatControl volumeControl = (FloatControl) sourceDataLine
522                    .getControl(FloatControl.Type.MASTER_GAIN);
523            volumeControl.setValue(6.00f);
524
525            sourceDataLine.start();
526            sourceDataLine.open(format);
527            sourceDataLine.start();
528            sourceDataLine.write(soundbytes, 0, soundbytes.length);
529            sourceDataLine.drain();
530            sourceDataLine.close();
531
532        } catch (Exception e) {
533            System.out.println("Not working in speakers...");
534            e.printStackTrace();
535        }
536    }
```

圖　5 tospeaker function

Client 端音頻收送程式碼如圖 6 所示

```
112    public void startStreaming() {
113
114        Thread streamThread = new Thread(new Runnable() {
115            @Override
116            public void run() {
117                try {
118
119                    socket = new DatagramSocket();
120                    Log.d("VS", "Socket Created");
121                    byte[] buffer = new byte[minBufSize];
122                    Log.d("VS", "Buffer created of size " + minBufSize);
123                    DatagramPacket packet;
124                    // machine's IP
125                    final InetAddress destination = InetAddress.getByName(IP);// 163.21.245.164
126                    Log.d("VS", "Address retrieved");
127
128                    recorder = new AudioRecord(MediaRecorder.AudioSource.VOICE_RECOGNITION,
129                            sampleRate, channelConfig, audioFormat,minBufSize * 10);
130                    Log.d("VS", "Recorder initialized");
131                    recorder.startRecording();
132
133                    while (status == true) {
134                        // reading data from MIC into buffer
135                        minBufSize = recorder.read(buffer, 0, buffer.length);
136                        // putting buffer in the packet
137                        packet = new DatagramPacket(buffer, buffer.length,destination, 40000);
138                        socket.send(packet);
139                        System.out.println("MinBufferSize: " + minBufSize);
140
141                    }
142
143                } catch (UnknownHostException e) {
144                    Log.e("VS", "UnknownHostException");
145                } catch (IOException e) {
146                    e.printStackTrace();
147                    Log.e("VS", "IOException");
148                }
149            }
150
151        });
152        streamThread.start();
153    }
```

圖 6　Client 端音頻收送

(二) Module 1

Server 端建置座位表的方法有兩種，第一種為直接以滑鼠點擊來刻畫出單一位置，第二種為 mesh 的方式，輸入行數及列數來建置大量座位，建完議場座位表後，再將位置之 X 軸及 Y 軸座標傳送至 Client 端，Client 端依接收到的座標建立出座位表，且 Server 端之座位表會不斷更新，Server 端座標傳送程式碼如圖 7 所示，Client 端座標接收程式碼如圖 8 所示。

```
393     public static void sendX() {
394         new Thread() {
395             public void run() {
396                 try {
397
398                     serverSocket = new ServerSocket(53000);
399                     while (yes) {
400
401                         Socket connection = serverSocket.accept();
402                         String out = "";
403                         for (int j = 0; j < li; j++) {
404                             out += arrX[j] + "::";
405                         }
406
407                         DataOutputStream output = new DataOutputStream(
408                                 connection.getOutputStream());
409                         output.writeBytes(out + "\n");
410
411                         connection.close();
412                     }
413                 } catch (IOException e) {e.printStackTrace();}
414             }
415         }.start();
416     }
```

圖 7 Server 端座標傳送

```
50  private class connectX extends AsyncTask<Void, Void, String> {
51
52      protected String doInBackground(Void... params) {
53          String result = "";
54
55          try {
56              Socket client = new Socket(IP, 53000);// 192.168.1.69
57              BufferedReader input = new BufferedReader(
58                      new InputStreamReader(client.getInputStream()));
59              result = input.readLine();
60
61          } catch (IOException e) {
62              e.printStackTrace();
63          }
64          return result;
65      }
66
```

圖 8 Client 端座標接收

(三) Module 2

當 Client 端想發言時，需對 Server 端提出申請，Server 端接收到 Client 端之申請後，可允許或拒絕給予發言權，再將回應傳送回 Client 端，Client 端接收回應後，若回應為允許則取得發言權並得以發言，若為拒絕則無法發言，Client 端傳送申請發言請求程式碼如圖 9 所示，Server 端接收 Client 端發言申請程式碼如圖 10 所示。

```
155    public void sendApply() {
156        Thread nameThread = new Thread(new Runnable() {
157
158            public void run() {
159                try {
160
161                    DatagramSocket clientSocket = new DatagramSocket();
162                    byte[] sendStr = new byte[15];
163                    Log.e("VS", "sendApply");
164
165                    sendStr = "apply".getBytes();
166                    Log.e("VS", new String(sendStr, "UTF-8"));
167                    final InetAddress destination = InetAddress.getByName(IP);// 192.168.0.3,192.168.1.71
168                    DatagramPacket sendPacket = new DatagramPacket(sendStr,sendStr.length, destination, 55000);
169                    clientSocket.send(sendPacket);
170                    receiceturn();
171                } catch (IOException e) {
172                    e.printStackTrace();
173                    Log.e("VS", "IOException");
174                }
175            }
176        });
177        nameThread.start();
178    }
```

圖 9 Client 申請發言

```
533    public static void acceptApply() {
534        acceptThread = new Thread(new Runnable() {
535            public void run() {
536
537                try {
538                    byte[] accept = new byte[15];
539                    applySocket = new DatagramSocket(55000);
540                    DatagramPacket apply = new DatagramPacket(accept,accept.length);
541                    while (yes) {
542                        System.out.print("ready to accept");
543                        applySocket.receive(apply);
544                        System.out.print(new String(accept, "UTF-8").substring(
545                                0, 5));
546                        if ((new String(accept, "UTF-8")).substring(0, 5)
547                                .equals("apply")) {
548                            for (int i = 0; i < index; i++) {
549                                if (apply.getAddress().equals(clientIP[i])) {
550                                    b[i].setBackground(Color.RED);
551                                }
552                            }
553                        }
554
555                    }
```

圖 10 Server 端接收請求

2.3 系統介面展示

以下就 module1、module2 之功能介面來作展示。

(一)Module 1

首先 Server 端必須先建立座位表，Server 端建立座位表的方式有兩種，第一種為一次產生大量座位的表格方式，輸入行數及列數，按下 mesh 鈕產生如圖 11 所示。

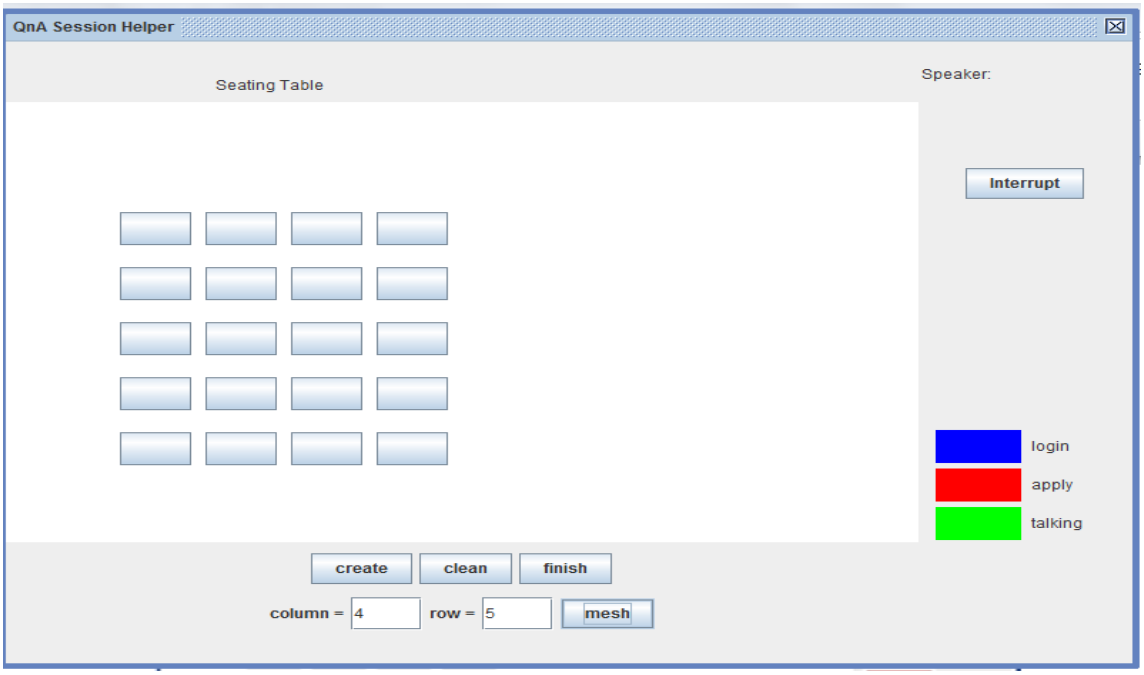


圖 11 mesh 產生座位

第二種為以滑鼠點擊一次產生一個座位，按下 create 鈕滑鼠才能在 seating table 區域中點擊出座位如圖 12 所示，此區域最多能產生 100 個座位，若建立的座位表有誤，可以刪除修改，再重新設置座位表，按下 clean 鈕會出現警告視窗如圖 13 所示，確認你真的要刪除，以免誤點 clean 鈕將整個座位表刪除，建置完畢按下 finish 便可將座位表傳送至 Client 端。

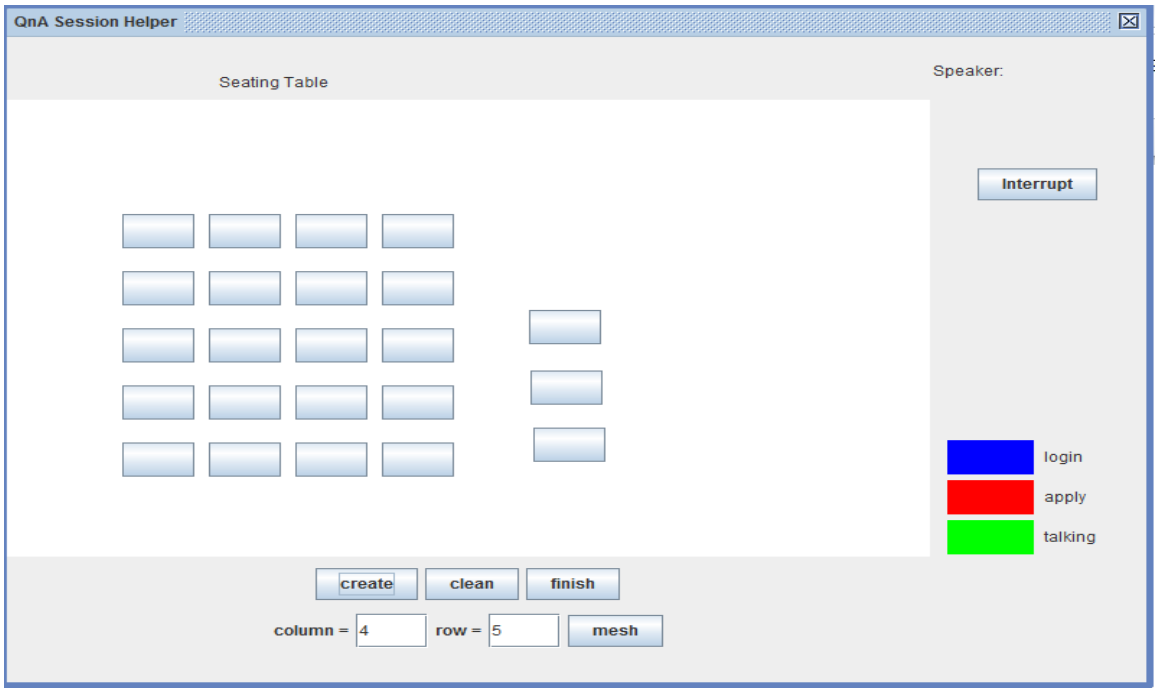


圖 12 滑鼠點擊產生座位

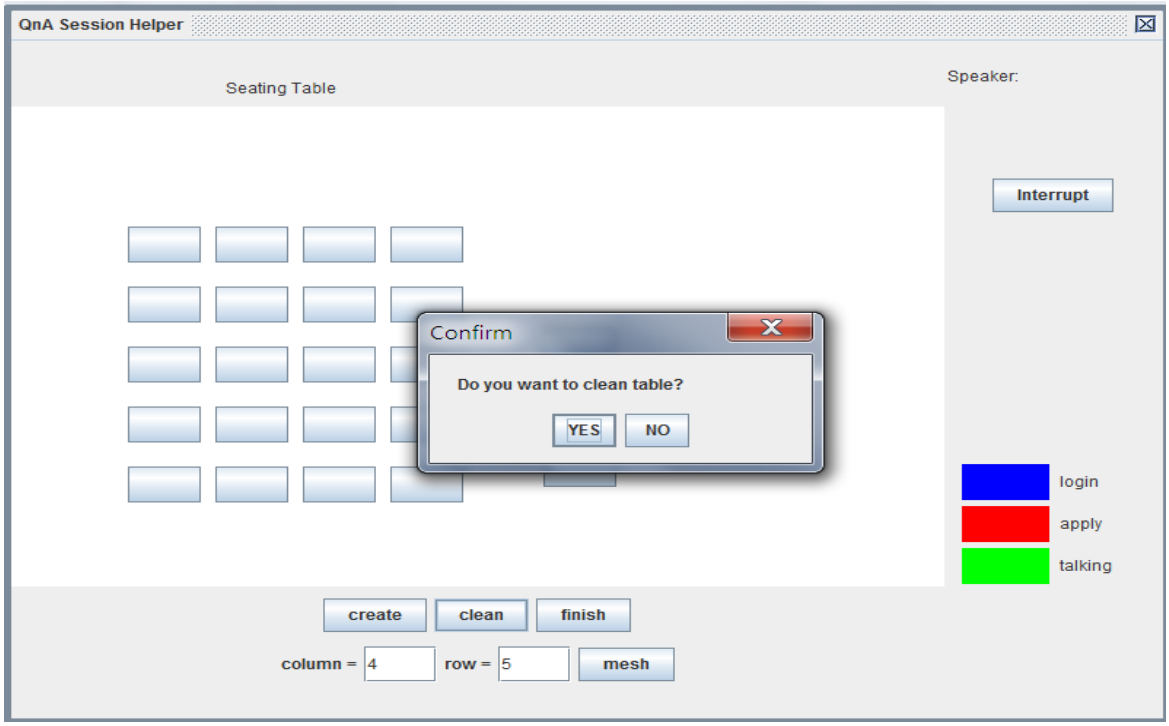


圖 13 刪除座位表

Client 端會先進入登入畫面，輸入帳號密碼後按 login 如圖 14 所示，登入成功後，會進入座位表畫面，此座位表為剛剛 Server 端傳入的，來賓依照自己的位置，按下後會跳出一個小視窗如圖 15 所示，若發現座位有誤可取消重選，確認後按下 finish 便會立即將你的座位資訊傳回 Server 端如圖 16 所示，兩端之座位表也會即時更新，你可以看到其他座位上坐的是誰如圖 17 所示，
此時 Server 端的來賓狀態皆為 login。

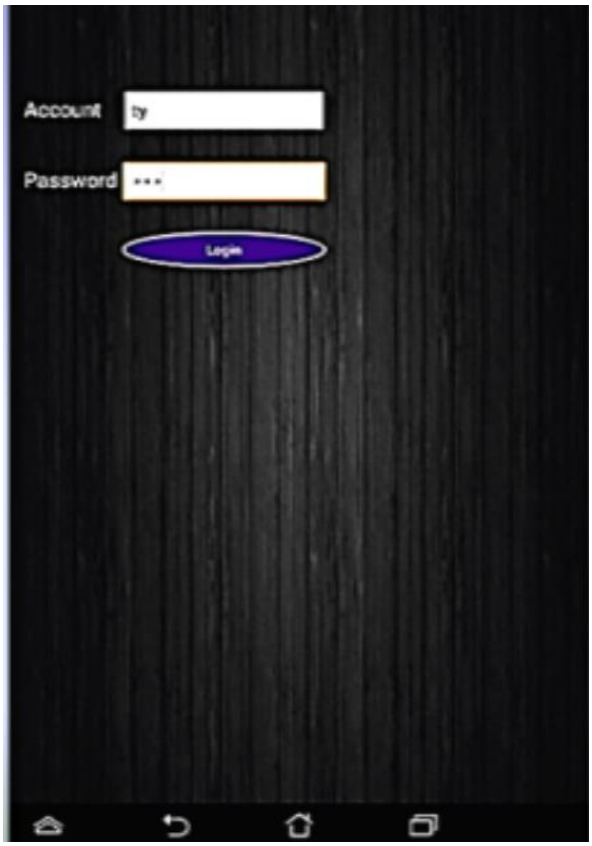


圖 14 登入畫面

圖 15 Client 設定座位

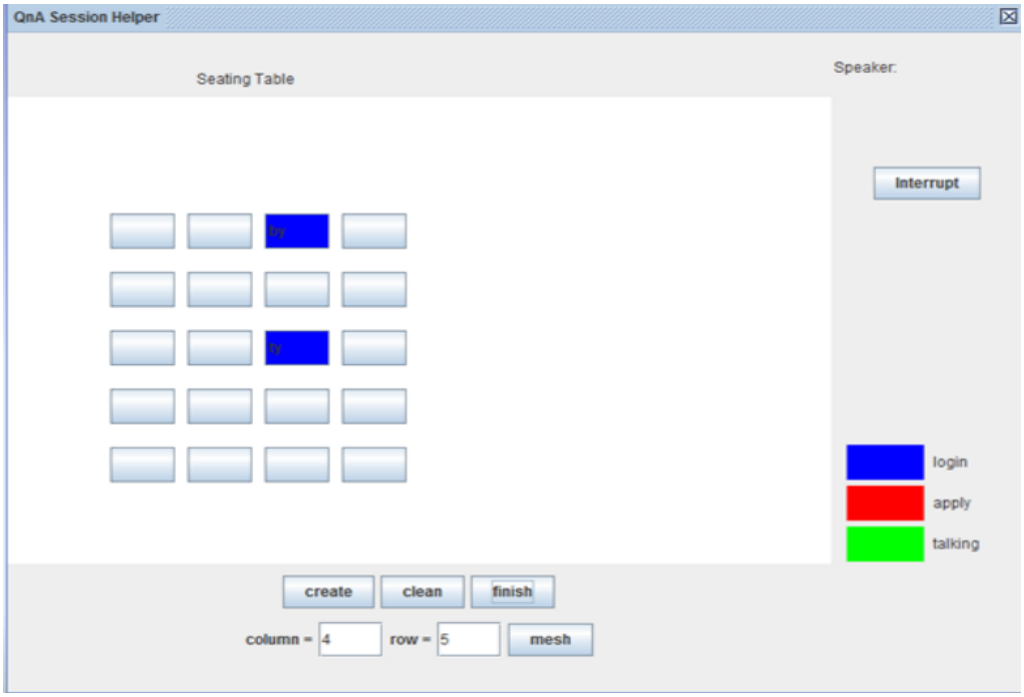


圖 16 Server 端更新座位表

圖 17 Client 端更新座位表

(二)Module2

Client 端完成座位設定後，按下 Next 會進入發言申請畫面如圖 18 所示，若想發言按下 Apply 作申請，此申請會傳回 Server 端，此時申請人的狀態會改為 apply，一次可能有多個人提出申請，主持人一次可選一人發言，只要按下申請人的座位會出現一個小視窗，確認你要給他發言權如圖 19 所示，確認後來賓的發言頁面會出現一個氣泡訊息如圖 20 所示，通知你已取得發言權，此時來賓只要按下發言頁面中的麥克風圖案即可將發言傳至主機端，而 Server 端發言人的狀態會改為 talking，speaker 欄位會改為發言人的名字。

當來賓按下 Stop，便可結束發言，聲音就傳不回 Server 端，發言停止後在 Server 端之狀態又會變成原始的 login 狀態，而 Speaker 人名也會清空。

圖 18 發言頁面

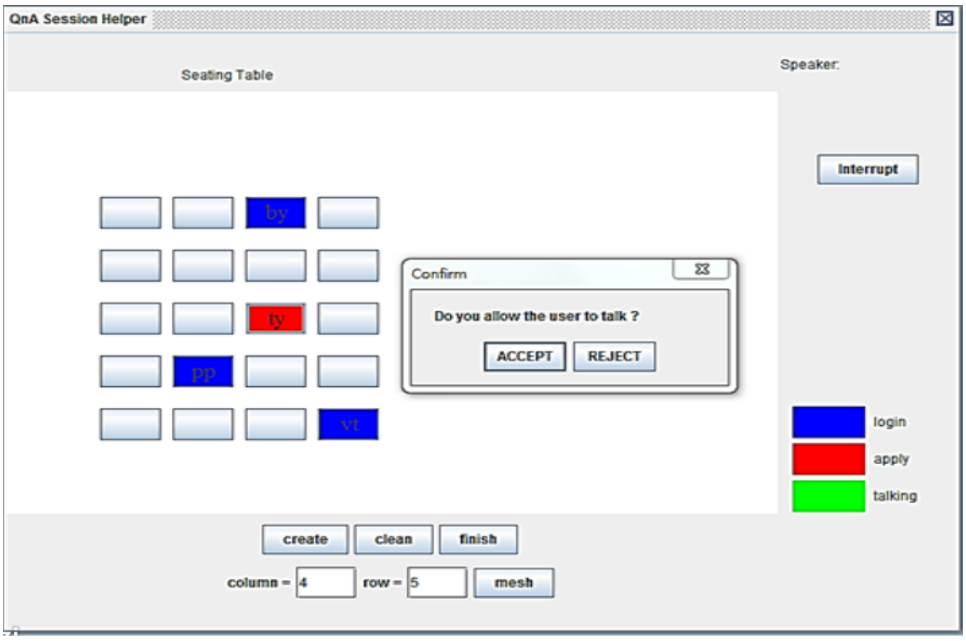


圖 19 選取發言人

圖 20 通知已取得發言權

第三章 研究結論及建議

本章共分成兩小節，第一節先介紹研究結論，第二節為系統改進及未來展望。

3.1 研究結論

藉由第一章之動機，經過一連串的實作，可以得到以下關於此系統之結論。

1. 此系統可以使得會議流程更加順暢。
2. 主持人可以選擇一位有發言意願之來賓，給予發言權。
3. 主持人可以清楚知道來賓所在位置和當前狀態。
4. 主持人可以控制來賓發言的時間。

3.2 系統改進及未來展望

礙於目前介面之設計，座位表最多只能建置 100 個座位，建置方式只有表格及滑鼠點擊兩種，未來希望能將座位表擴充到更大的議場環境，並且讓建置方式更有彈性，座位表清除目前只能整個刪除，未來希望能將部分或單一座位刪除，再者聲音的品質希望能再提升，讓音頻更加穩定順暢。

Client 端座位設定部分，設定完成傳送至 Server 端後便無法更改，若來賓不小心按錯位置便會影響其他人的設定及座位表的正確性，所以希望能就這個部分做改進，能讓來賓設置座位時有更正的功能。

就申請發言到給予發言權的部分，目前一輪可以讓多個有意願發言之來賓申請，但主持人一次只能選一人發言，選取後其他想發言的來賓便要重新提出申請，這邊的設計不是很好，未來希望能讓主持人不只一次選一個人，一輪可選取至多五人放入佇列中，就選取的順序來發言。

附錄

QnA Session Helper

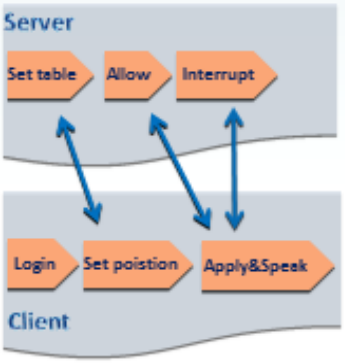
Student : Ruo-Yun Wang (王若芸) and Chia-hao Lien (連嘉豪)
Advisor : Prof. Shin-Tsung Liang (梁世聰)

Abstract

During a Q-and-A session of a conference organized in a large meeting venue, it takes much time for the session chair to wait conference attendants moving around and Delivering a handheld microphone to a remote questioner. In this project, we propose QnA Session Helper as a total solution to solve this problem. With the pre-installations of the server application on the host computer and the client App on each of the handheld devices of the conference attendees, an efficient and well-organized Q-and-A session providing all attendees with their personal microphones can be achieved.

Structure

- Server sets table,uses socket to transfer to client and keeps updating.
- Server gives client the right of speech.
- Server interrupts client's speech.



- Client gets table, sets the position,and transfers to server.
- Client applys to speak.
- Client gets the right of speech,and starts to speak.

Conclusion

i . This tool helps conference flow more smoothly.
ii. The host can choose one of the applicants and give her/him the right of speech.
iii. The host can clearly know the position of the speaker.
iv. To avoid speaking too long,the host can interrupt her/his speech.

How to use

References

[1]http://stackoverflow.com/questions/25527751/flowcontrol-whvalue-whowing-java- lang- illegalargumentexception
[2] http://www.tutorialspoint.com/java/java networking.htm
[3]http://beej.us/guide/bgnet/output/html/multipage/index.html

參考資料

[1] http://stackoverflow.com/questions/25927751/floatcontrol-setvalue-showing-java-lang- illegalargumentexception

[2] http://www.tutorialspoint.com/java/java_networking.htm

[3] http://beej.us/guide/bgnet/output/html/multipage/index.html