國立暨南國際大學資訊工程學系

碩士論文

以 IPv6 多播功能進行隨按即說服務(PTT)之效能提升手段

Performance Improvement of Push to Talk (PTT) Service with IPv6 Multicast

指導教授:吳坤熹博士

研究生:吳怡蓓

中華民國 109 年 4 月

doi:10.6837/ncnu202000205

國立暨南國際大學碩士論文考試審定書

資訊工程學系（研究所）

研究生 吳怡蓓 所提之論文

Performance Improvement of Push to Talk (PTT) Service with IPv6 Multicast

以 IPv6多播功能進行隨按即說服務(PTT)之效能提升手段

經本委員會審查，符合碩士學位論文標準。

學位考試委員會

高志隆 委員兼召集人

張瑞杰 委員

吳坤熹 委員

中華民國 一〇九 年 四 月 十五 日

doi:10.6837/ncnu202000205

致謝詞

首先，最感謝的就是從大一至研究所都一路幫助我、指導我的 吳坤熹 老師，其中經歷了課業與打工兩頭燒、大三研究專題不確定，甚至是該不該念研究所等等階段，都是老師一直在後面督促我、鼓勵我，讓我在每一個猶豫不前的分岔路上，都有了另一個可能的選項。當我擔心生活所需，老師提供了我機會，讓我能把時間花在跟科系相關的技術上，卻又能負擔學費生活費。在我想休學的時候，在我覺得挫折的時候，老師的話總能讓我又拾回一些信心。每一次跟老師開會前，雖然總是覺得壓力大，老師也並不會直接把答案給我，但是總能從老師的話中得到一些想法，反而開會完心情是非常舒暢的。而除了學業上，老師也帶領我參加鐵人比賽以及馬拉松等，讓我除了課業之外還發現自己其實在其他不熟悉的領域上還是能表現得不錯，除了為自己帶來成就感也能維持身體的健康。在每一次上到了講台上或者比賽之前，老師總會提醒我要睡飽才能做到最好，並且提醒我要吃早餐。這些小小的關心帶給我莫大的力量，真得非常謝謝老師。

再來，要感謝我的家人，不管我做什麼決定，總是給予我支持，並且在我失敗的時候鼓勵我，要我再繼續加油就好。

最後，我知道自己一直都不是一個很稱職的學生，不管是提早畢業還是五年一貫都沒能準時完成。但還是很感謝自己，畢竟完成還是得靠自己。從大學到研究所中間所經歷了很多無法言喻的事情也只能對自己訴說，很感謝雖然是那麼不完美的，但我還是走過來了。希望以後也能帶著這些過去，繼續努力，不要因為安逸的生活而忘了自己曾經努力過。

doi:10.6837/ncnu202000205

論文名稱：以 IPv6 多播功能進行隨按即說服務(PTT)之效能提升手段

校院系：國立暨南國際大學科技學院資訊工程學系　　頁數：53

畢業時間：中華民國 109 年 4 月　　學位別：碩士

研究生：吳怡蓓　　指導教授：吳坤熹博士

摘要

隨按即說(Push to Talk, PTT)的技術從早期的無線電對講機(Walkie-Talkies)到現在常用的手機對講服務(Push to Talk over Cellular, PoC)已有久遠的歷史。最早，對講機是作為軍事用途，近年來各國則致力於研究關鍵任務一按通服務(Mission Critical Push-to-Talk, MCPTT)，也就是將 PTT 應用於公共安全上，改善第一線人員使用公共安全通訊系統(Public Safety Communication)的運作效能。由於現今的 PTT 是建立在網路電話(Voice over IP, VoIP)的技術上，所以 PTT 聲音的延遲程度、多人通話是否造成效能降低以及對行動通訊的支援能力也是大家所關注的。

在本篇論文中，我們設計出在 IPv6 (Internet Protocol Version 6)的環境下，使用 SIP 協定(Session Initiation Protocol) 以及 TBCP 協定的概念(Talk Burst Control Protocol) 來完成系統中信令(Signaling)控制，並以 GLM (Group List Management) 及 3PCC 協定(Third Party Call Control)完成多播群組(Multicast Group)的建立以及管理。有別於過往電信業者的 PoC 系統，本篇論文在 IPv6 環境中利用 SIP 代理伺服器 (Proxy)服務以及 IPV6 多播的方式，讓使用者的聲音串流無須透過伺服器，也無須複製多份資料給群組內其他使用者，就能達到隨按即說的功能，並且利用多播方式減少往返伺服器所需傳送的信令。因此提供一個傳輸資料更有效率，信令交換時間更簡短的即時 PTT 系統。

關鍵字: 3PCC, GLM, IPv6, Multicast, PTT(Push-to-Talk), SIP, TBCP, VoIP

doi:10.6837/ncnu202000205

Title of Thesis: Performance Improvement of Push to Talk (PTT) Service with IPv6 Multicast

Name of Institute: Department of Computer Science and Information Engineering, College of Science and Technology, National Chi Nan University Pages: 53

Graduation Time: 04/2020 Degree: Master

Student Name: Yi-Bei Wu Advisor Name: Dr. Quincy Wu

Abstract

From the early Walkie-Talkies, to the recent Push to Talk over Cellular (PoC), it has been a long history where the PTT (Push to Talk) technology is applied. Although Walkie-Talkies have been used in military operations in decades, many countries are recently devoted to the Mission Critical Push-to-Talk (MCPTT) technology, which applies PTT to improve the performance of public safety communication for first responders. Because PTT is based on voice over IP (VoIP) technologies, the performance, security, and mobility of PTT are also of great concerns to researchers and manufacturers.

This paper proposed a design and implementation of a PTT system, using the SIP (Session Initiation Protocol) and TBCP (Talk Burst Control Protocol) to transport signaling and using GLM (Group List Management) and 3PCC (Third Party Call Control) protocol to create/manage multicast groups. Additionally, in contrast to commercial PoC systems, this paper fully utilizes a subtle design in the SIP protocol and the IPv6 multicast feature, which allows users to send audio data without going through the PTT server, which was the major factor of transmission delay. In summary, this paper provides a more efficient PTT system.

Keywords: 3PCC, GLM, IPv6, Multicast, PTT(Push-to-Talk), SIP, TBCP, VoIP

doi:10.6837/ncnu202000205

目次

doi:10.6837/ncnu202000205

doi:10.6837/ncnu202000205

doi:10.6837/ncnu202000205

表目次

doi:10.6837/ncnu202000205

圖目次

doi:10.6837/ncnu202000205

doi:10.6837/ncnu202000205

第一章 緒論

第一節 研究動機

最早，無線電對講機是作為軍事用途，近年來各國則致力於研究關鍵任務一按通服務，也就是將 PTT(Push to Talk)應用於公共安全上，改善第一線人員使用公共安全通訊系統的運作效能[1]。

國際標準組織開放行動聯盟(Open Mobile Alliance, OMA)針對 PTT，成立了 PTT over Cellular (PoC) [2] 工作群組來制定相關標準機制。源自於傳統無線電的概念，在通話之前，必需先將成員所使用的通訊器材調整至同一頻道(Channel) ，在無線電波的有效範圍內，當使用者按下發話鈕時，所有同頻道的成員才會接收到該使用者發出的訊息。而在 PoC 機制中，與無線電通訊不同之處在於，PoC 透過網際網路協定(Internet Protocol, IP)來建立連線，必須先將同一群組的使用者註冊到伺服器上後，透過通用封包無線服務(General Packet Radio Service, GPRS)或是 3G/4G 無線通訊服務傳輸語音(Audio)。PoC 與傳統無線電最大的差別在於不會有共用頻道或是頻道干擾的問題，也不會受到無線電波在物理上的距離限制。

現今的 PoC 工作原理類似網路電話[3]，主要是基於網際網路工程任務小組(Internet Engineering Task Force, IETF)所定義的會議初始協定 (Session Initiation Protocol, SIP) [4]和即時傳輸協定(Real-time Transport Protocol, RTP)以及發話權控制協定(Talk Burst Control Protocol, TBCP) [5]。要在網路環境下進行通話的控制，最重要的部分是信令交換，SIP 協定就是負責語音傳輸時的信令交換。而在 OMA 的架構中，是透過背靠背用户代理（Back-to-Back User Agent）為 SIP 伺服器運作方式，並利用 Unicast 來作聲音資訊的傳輸，不但增加用戶代理與伺服器的負擔，也使聲音資訊傳輸距離變遠。因此本論文的研究動機，希望提出不同的架構，改善整體運作的效能。

doi:10.6837/ncnu202000205

第二節 研究目的

本篇是基於 OMA 提出之 PoC，加以改良，以達到聲音傳輸效率提升之手段。主要使用了 SIP 伺服器做為語音串流起始(Initiate)、修改(Modify)與終止(Terminate)信號的控制伺服器，並且管理終端設備彼此之間的連線狀態。群組管理部分由 GLM 伺服器(Group List Management Server)分配 IPv6 多播位址，並經由 3PCC 控制器(Third Party Call Control Controller)[6]來邀請使用者加入通話，發話權(Floor)則是利用 TBCP 來做管理，其中 TBCP 是利用 SIP 來傳遞訊息。最後，即時傳輸協定是以用戶資料包協定(User Datagram Protocol, UDP) [7]為基礎來傳遞聲音資訊，透過 IPv6 的多播群組來完成整個 PTT 系統運作。

本論文主要探討在 IPv6[8]的網路環境底下，利用多播群組[9]，結合 SIP 協定、TBCP 協定、GLM[10]以及 3PCC 控制器運作，提供 PTT 服務。除了信號管理需要透過 SIP 模組以及 TBCP 模組，在語音傳遞的部分可由端點至端點直接通訊，不需透過伺服器轉送；並且可使用多播方式傳送，降低伺服器在遠處所造成的傳輸延遲問題，也不會因為群組人數增加造成使用者送出的封包數增加。不但提高了即時語音的便利性，更提供了一種即時語音的解決方案。

doi:10.6837/ncnu202000205

第三節　論文架構

本篇論文之其餘章節分為七章。第二章為研究背景；第三章回顧前人文獻；第四節為系統架構及系統運作流程之細節；第五章為系統實作；第六章為系統功能分析與比較；最後則為本篇論文的結論及未來展望。

doi:10.6837/ncnu202000205

第二章 研究背景

無線對講機(Walkie-Talkie)至今已經有久遠歷史，而其特色在於必需先將成員所使用的通訊器材調整至同一頻率的頻道(Channel)，在無線電波的有效範圍內，同時間只有一個人擁有發話權。當使用者按下發話鈕時，所有同頻道的成員才會接收到該使用者發出的訊息，並且當其他成員拿到發話權後，也能回話。前述所謂「一人擁有發話權」就是半雙工(Half-Duplex) 系統概念，而讓多人在相同頻率的頻道上彼此通話則為雙向無線電 (Two-Way Radio)的概念。

現今的 Push to Talk over IP 則是沿用了對講機的優點與特色，並解決了對講機通話距離有限的缺點，使得 PTT 這個服務不管在哪裡都能使用。PTT 的運作原理主要是基於 Voice over IP(VoIP) ，而 VoIP 是基於 SIP 這個協定，用來建立、修改及終止多媒體會議，SIP 將會在第一節作介紹。而在 PTT 中還有一個很重要的角色，就是發話權的控制，第二節介紹的就是由 OMA 為處理發話權控制所提出的 TBCP。而現今的 PTT 架構中應屬 OMA 所提出的 PoC 最為有名，會在第三節作介紹。此外，本篇使用了 IPv6 作為網路層(Network Layer)的協定，將會在第四節作介紹。最後在本篇論文中使用了 3PCC 做為建立群組的方法，會在第五節作介紹。

第一節 SIP 協定

目前常用的網路電話協定為 H.323、SIP、MEGACO 和 MGCP 等。其中 H.323 是最早被全世界廣為採用的網路電話與視訊會議協定，但因複雜性高，在許多技術上的問題受限，不容易針對新的應用作擴展，而有了後續新協定的產生。SIP 協定用於多媒體會談(Multimedia Session)的建立、修改及終止。SIP 為會談發起時的初始化協定，在協定初始化後，傳遞影音等資料則需要其他傳輸協定。例如即時傳輸協定(RTP)[11]能即時傳輸影像與聲音資料，而即時傳輸控制協定(RTCP)則

doi:10.6837/ncnu202000205

提供服務品質(Quality of service, QoS)；若與即時串流協定 (Real-time Streaming Protocol, RTSP)結合則可以用來控制多媒體語音的播放暫停。會話描述協定(Session Description Protocol, SDP)[12]可以用來描述多媒體的串流（Streaming），例如媒體形式(Media Type)，傳輸協定(Transport Protocol)以及編碼格式(Codec)等等。

SIP 協定是基於用戶資料包協定(UDP)與網際網路協定(IP)運作的信令協定。SIP 通訊模式與超文本傳輸協定(HyperText Transfer Protocol, HTTP)[13]類似，都是採用請求(Request)、回應(Response)的模式，並提供數種的要求命令(Command)和回應代碼(Status Code)，配合夾帶的標頭欄位(Header Fields)訊息(如表一)和通訊內容(Content)來完成信令控制。SIP 協定在建立會談時使用一組類似 HTTP 文字格式的訊息傳遞和交換多個用戶之間的網際網路位址、媒體能力、編碼格式等資訊。SIP 與網際網路工程任務小組(IETF)的其他協定都有密切關係，例如即時傳輸協定(RTP)、會談公告協定(Session Announcement Protocol, SAP)、會話描述協定(SDP)。而一個完整的 SIP 服務系統，還需要網域名稱系統(Domain Name System, DNS)、動態主機設定協定(Dynamic Host Configuration Protocol, DHCP)、資源預留協定(Resource Reservation Protocol, RSVP)等協定的配合才能正常運作。其中會話描述協定可視為 SIP 訊息傳送的內容，專門負責處理媒體格式的協商。

doi:10.6837/ncnu202000205

表 一、SIP 標頭[4]

General-headers	Entity-headers	Request-headers	Response-headers
Call-ID	Content-Encoding	Accept	Allow
Contact	Content-Length	Accept-Encoding	Proxy-Authenticate
Cseq	Content-Type	Accept-Language	Retry-After
Date		Authorization	Server
Encryption		Contact	Unsupported
Expires		Hide	Warning
From		Max-Forwards	WWW-Authenticate
Record-Route		Organization	
Timestamp		Proxy-Authorization	
To		Proxy-Require	
Via		Route	
		Require	
		Response-Key	
		Subject	
		User-Agent	

SIP 伺服器的運作模式如圖一，首先用戶代理(User Agent, UA)必須向 SIP 伺服器進行註冊(REGISTER)；當撥出(Call)電話時，則會送出邀請(INVITE)給其他用戶代理；如果使用者接起電話，用戶代理就會回傳 200 OK 表示接受，最後再由發起者回傳一個 ACK 代表對話建立；結束通話時，則送出(BYE)的信令。

doi:10.6837/ncnu202000205

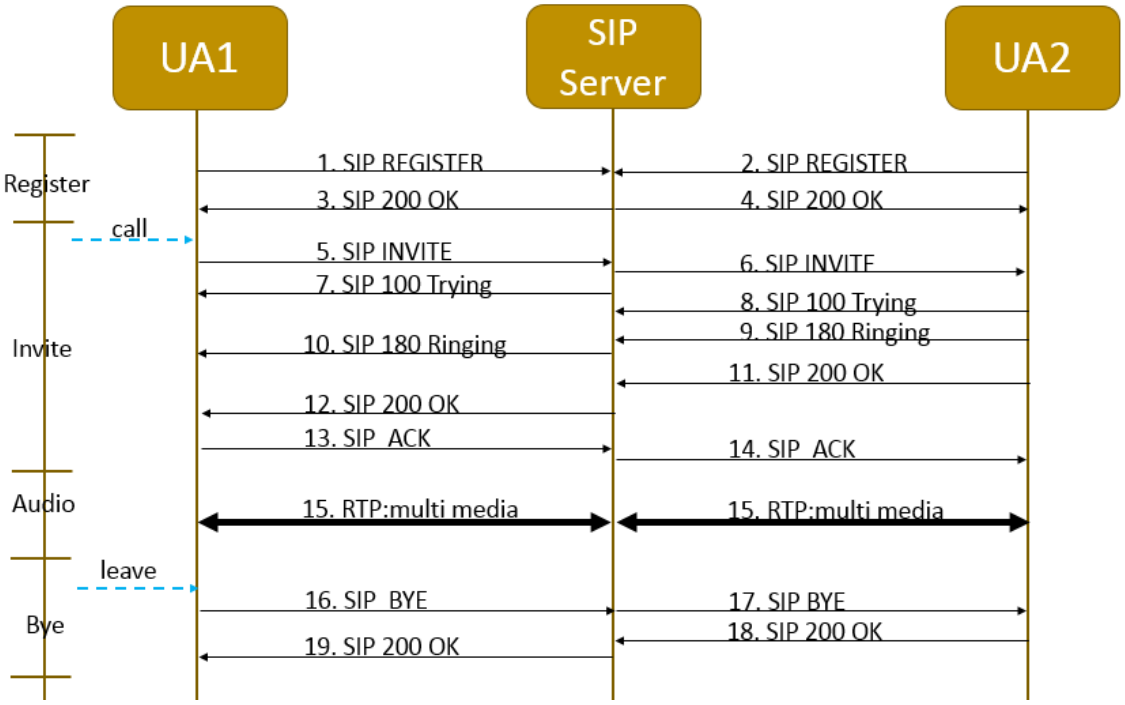

圖 一、SIP 運作流程圖

第二節 TBCP 協定

國際標準組織開放行動聯盟(OMA)推出之 PoC 架構是運用即時傳輸控制協定(RTCP)的 APP Message 封包來夾帶 TBCP 發話權的請求。其特殊之處在於，TBCP 每送出一個請求，都會有一個時間計時器(Timer)被啟動。當時間到了(Timeout)以後會再重送(Retransmit)請求。

圖二為即時傳輸控制協定的 APP Message 之標頭，其中 Message Subtype 用來傳遞 TBCP 的訊息(Message)，例如 TBCP Request、TBCP Granted、TBCP Deny、TBCP Release、TBCP Idle、TBCP Taken 等，來達到傳遞發話權控制訊息的效果。

doi:10.6837/ncnu202000205

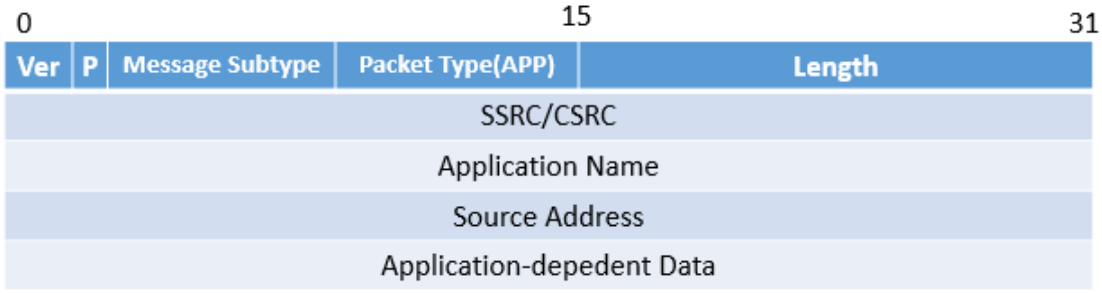

圖 二、即時傳輸控制協定的 APP Message 標頭

TBCP 伺服器的運作模式如圖三。當使用者按下 PoC 按鈕，用戶代理(UA1)會發出請求(Request)。若現在無人持有發話權，伺服器就會允許(Granted)，並且向同群組的所有其他用戶代理發送發話權已被持有的訊息(Taken)；之後，當使用者放開按鈕，也就是發話結束， UA1 會發送一個發話權釋放的訊息(Release)，伺服器也會通知同群組所有用戶代理目前發話權已閒置(Idle)。

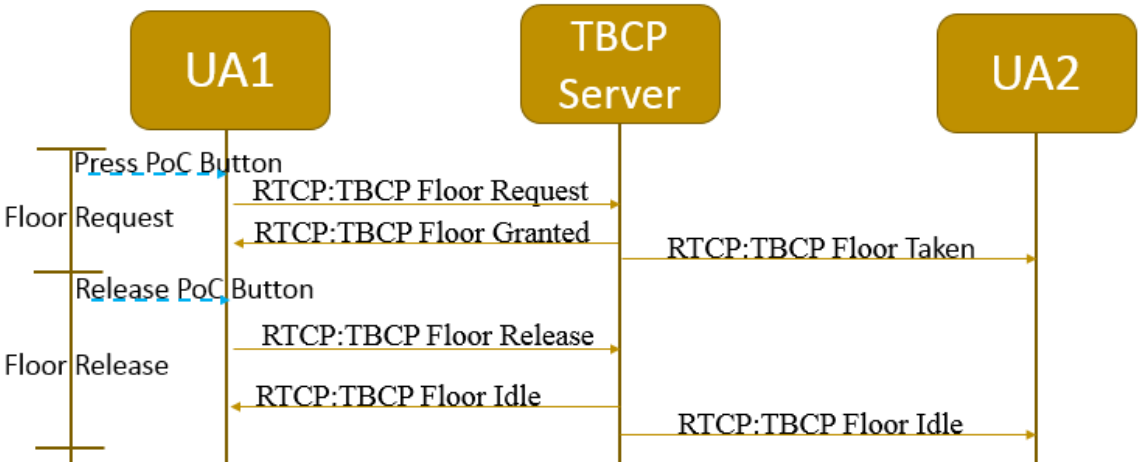

圖 三、TBCP 運作流程

第三節　OMA 架構之 PoC 運作流程

OMA 的 PoC 架構，可分為四個部分，分別為 PoC 客戶端(PoC Client)、GLM 伺服器 、SIP 伺服器以及 PoC 伺服器，如圖四。其中使用到之 SIP 以及 TBCP 協定的運作模式可參考前兩節之介紹。而 GLM 伺服器之運作可參考[10]，使用者如需要建立、修改、閱讀群組資訊或刪除群組時，會使用 HTTP 傳送 GET 封包給 GLM 伺服器，並由 GLM 伺服器回傳 200 OK，代表所欲執行的功能順利完成。

doi:10.6837/ncnu202000205

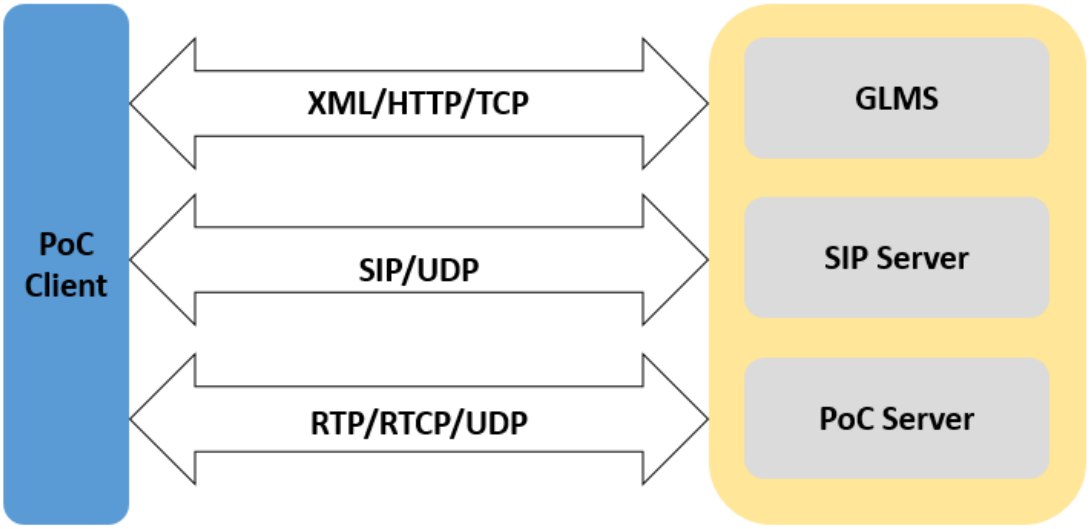

圖 四、OMA 之 PoC 架構圖

PoC 運作流程，如圖五所示。一開始，使用者按下按鈕，用戶代理(UA1)向 PoC 伺服器發送 INVITE 以及 TBCP Request 的信令，而 PoC 伺服器也會將 INVITE 信令轉送給被邀請的其他用戶代理。接著 PoC 伺服器會使用 RTCP 傳送 TBCP Granted 封包給 UA1，而其他用戶代理則會收到 TBCP 的 Taken 封包。此時即可開始以即時傳輸協定(RTP)來傳遞聲音訊號。當使用者放開按鈕， UA1 會向 PoC 伺服器傳送 TBCP Release 的信令，而 PoC 伺服器則會向群組內所有用戶代理傳送 TBCP Idle 的封包。如使用者再次按下按鈕，UA1 會傳送 TBCP Request 給 PoC 伺服器，接著 PoC 伺服器傳送 TBCP Granted 給 UA1，而其他用戶代理則會收到 TBCP 的 Taken。最後，使用者按下離開， UA1 會發送(BYE)信令給伺服器，並經由 PoC 伺服器轉送給其他用戶代理。

doi:10.6837/ncnu202000205

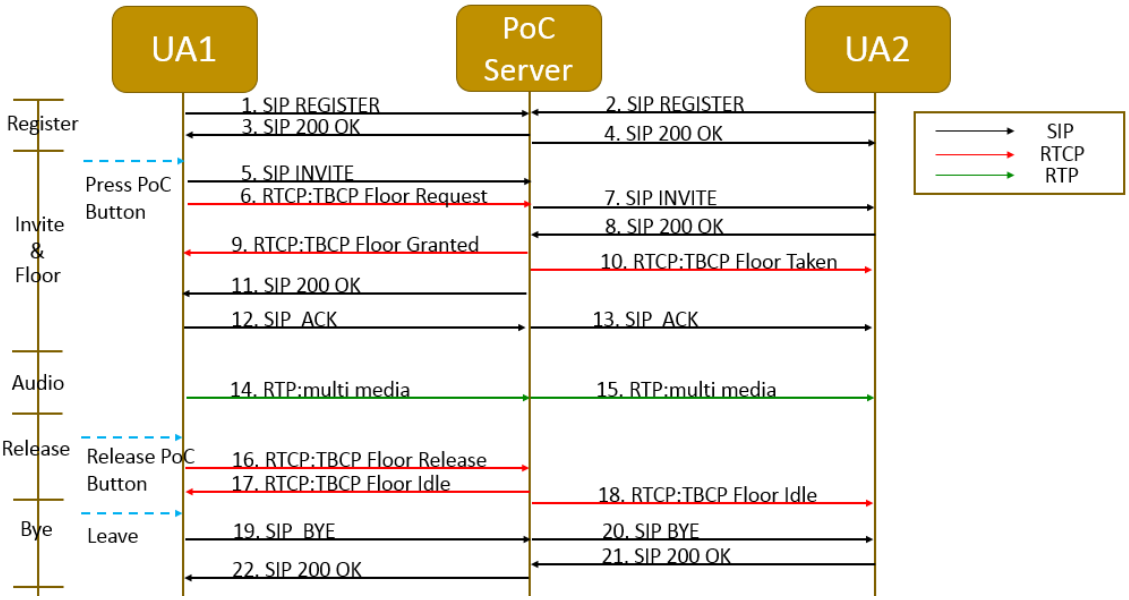

圖 五、OMA PoC 運作流程圖

第四節 IPv6

現今的網際網路發展蓬勃，因此 IP 位址已有逐漸用罄的趨勢，最初所設計的 IP 位址長度為 4 個位元組，最多分配給大約 40 億個設備，雖然目前的網路位址轉換(Network Address Translation, NAT)及無類別域間路由(Classless Inter-Domain Routing, CIDR)等技術可延緩網路位置匱乏之現象。但是除了大量位址空間，IPv6 還增加了自動設定位址(Auto-Configuration)、內建網際網路安全協定 (Internet Protocol Security, IPsec)加密機制、行動 IPv6(Mobile IPv6)以及服務品質機制，IPv6 多播(Multicast)也比以往 IPv4 的廣播(Broadcast)更加彈性，非常有優勢。

RFC 2373 中定義 IPv6 封包傳送有三種型式，分別為 Unicast、Anycast、Multicast。多播(Multicast)是多點傳輸(Multi-point Transmission)機制，當封包傳往多播位址時，所有聽在此多播位址的節點都能收到。因此，使用者只要送出一份資料，就能由路由器(Router)自動複製，轉發給所有群組內的節點(Node)。在有多個接收者時，可大幅降低傳送端的負擔。圖六所示為 IPv6 多播位址之格式(Format)：前面 8 個位元(Bit)為 FF，代表此為多播位址；其後 4 個位元為多播位址旗標(Flags) ，主要用來區分是否為網際網路號碼分配局(Internet Assigned Numbers

doi:10.6837/ncnu202000205

Authority, IANA)指派之多播位址(Well-known Multicast Address) ，表示為 0000，或是暫時性多播位址(Temporary Multicast Address) ，表示為 0001 ；再後面 4 個位元代表可傳送範圍(Scope)； 最後 112 個位元為 IPv6 多播位址的群組識別(Group ID) ，由分配局指派或者使用者自行定義之。

8 bits	4 bits	4bits	112 bits
1111 1111	Flags	Scope	Group ID

圖 六、IPv6 多播位址

第五節 3PCC 協定

第三方通話控制(Third Party Call Control, 3PCC)是由第三方(控制者)來對要通話的兩方進行通話邀請，雙方的 SDP 也由控制者負責協商。所以控制者會讓兩邊建立起 SIP session。在 RFC 3725 中定義了四種通話流程(call flow) 。而本論文所提出的 Multicast Push to Talk (MPTT)系統，則是參考其第一種的運作方式，如圖七。

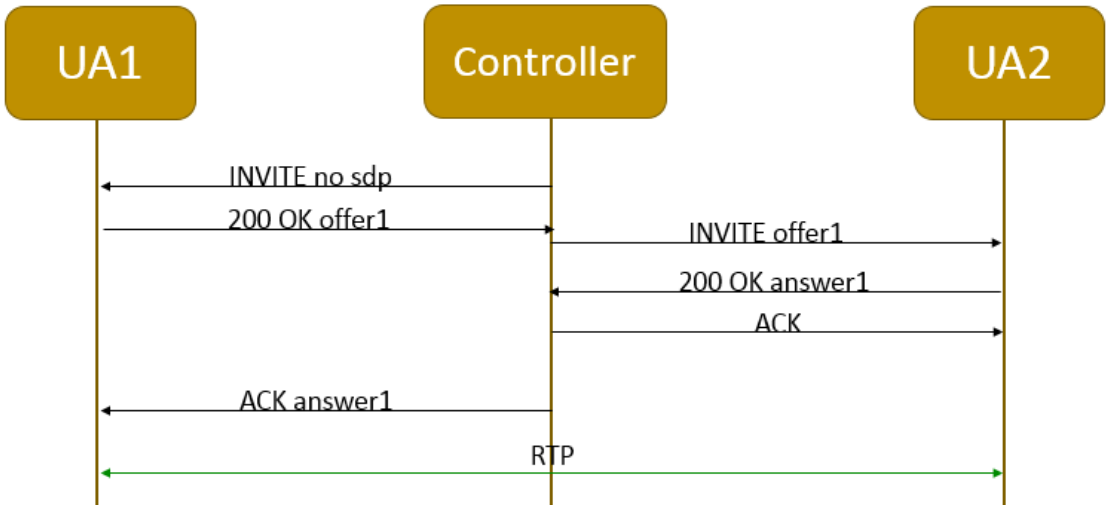

圖 七、3PCC Flow I

doi:10.6837/ncnu202000205

第三章　文獻探討

在本節中，將針對目前國內外學者先前所提出之 SIP 伺服器、TBCP 流程以及 IPv6 多播進行回顧；並綜合前人曾採用之架構，提出一個新的 PTT 架構，在第四章中說明。

第一節　點對點之語音傳輸(Point to Point Audio Streaming)

Akshai Parthasarathy 在[14]中提到，利用 SIP 伺服器作信令控制，而後利用即時傳輸協定(RTP)點對點傳送語音資訊。其想法奠定了本篇論文中語音資訊傳遞的方式，由 SIP 伺服器做信令控制，但傳輸語音資訊時，則是用戶代理對用戶代理，不再透過中間的 SIP 伺服器。現今個人電腦運算能力越發強大，所以讓伺服器只著重在信令控制，不需要浪費效能在語音傳遞上，既可以減輕伺服器負擔，對於伺服器與用戶代理的距離影響自然減少，語音傳輸也能更順暢。

第二節　多播 PTT

Jen, Chien-Maw 在[15]提出了利用多播即時傳輸協定與 SIP 協定的 PTT 實作，SIP 伺服器的部分是建立會談，而 PoC 伺服器除了做發話權控制也負責維護及儲存群組名單(Group List)。然而其 PoC 伺服器利用多播即時傳輸協定發送給各用戶端的，除了得到發話權的客戶端(Client)來源 ID(Source ID)，還有音訊。此架構的缺點在於，如果伺服器與用戶端距離較遠，則音訊的傳輸將極為耗時。故本論文參考其架構加以修改，TBCP 信令傳遞使用 SIP 傳送給所有用戶代理，而音訊傳遞的部分則不需透過任何伺服器，直接由用戶端點對點傳送。

第三節　多播位址群組(Multicast Address Group)

HaiYan Liu、Ying Li 與 YunFeng Zhang 在[16]中提出了實現 IPv6 多播通訊過

doi:10.6837/ncnu202000205

程，包括 IPv6 的 socket option，以及如何建立 socket 及收送資料，本篇論文也參考其提出之 IPv6 多播方法，用於多播隨按即說系統。

總合此章與第一章之內容，本篇論文與 OMA 架構不同的是，本篇的用戶代理需與 GLM 伺服器、SIP 伺服器、TBCP 伺服器進行資料交換或信令傳遞。其次，本篇論文中是先邀請用戶代理加入群組後才能開始按下按鈕擁有發話權，而非等使用者按下按鈕後才邀請其他用戶代理加入群組，這樣更符合原本傳統無線對講機的概念(預先調整為相同頻率)。第三，本篇參考以上各學者提出之架構與改進，取各篇之優點，點對點傳輸語音訊息、TBCP 伺服器與 SIP 伺服器的信令整合以及多播 IPv6，提出下一章之系統架構。

doi:10.6837/ncnu202000205

第四章　Multicast Push to Talk 系統

根據上一章各學者提出的研究，本論文提出的 Multicast Push to Talk (MPTT) 系統，是利用 SIP 模組為管理所有用戶代理的主要模組，TBCP 模組做為管理發話權的模組，GLM 模組為管理群組的模組，擁有 IPv6 多播位址對照表。最後，透過 IPv6 多播將語音訊息傳送給群組內所有用戶代理，不管是一對一或一對多的情境下，都能適用。以下兩節，將對於 MPTT 系統，詳細說明設計架構與各模組之描述。

第一節　系統架構

圖八所示為 MPTT 的系統架構圖，包含四部份：第一部份為用戶代理(UA)，第二部份為 SIP 模組(SIP Module)，第三部份為 GLM 模組(GLM Module)，第四部份為 TBCP 模組(TBCP Module)。

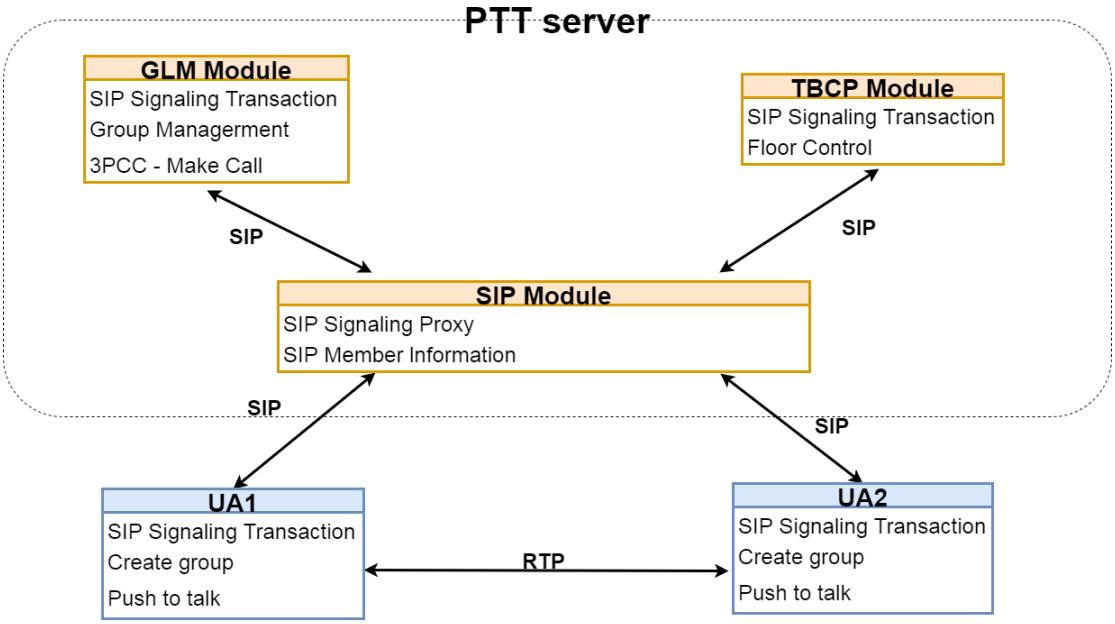

圖 八、系統架構圖

在 MPTT 系統中，有一台 PTT 伺服器，裡面有三個模組，分別為 SIP 模組、GLM 模組、TBCP 模組，模組與模組之間是透過 SIP 信令做溝通，模組與用戶代理之間也是透過 SIP 信令，模組的功能會在下個章節作介紹。而用戶代理與用戶

doi:10.6837/ncnu202000205

代理之間只需傳送使用者的 RTP 封包語音 (Audio Data)。

第二節　系統功能

表二所示為 MPTT 的系統功能表。SIP 模組的功能有 SIP Signaling Proxy，負責 SIP 信令(Signaling)轉送。SIP Member Information 負責用戶代理註冊。用戶代理之群組管理 Group Management 則是透過 GLM 模組，並且由 GLM 模組分配群組的 IPv6 多播位址給用戶代理，另外 GLM 模組也負責 3PCC - Make Call 的功能，負責邀請使用者加入群組。而每當使用者要發話時，則透過 TBCP 模組 Floor Control 功能來獲得發話權(Floor)。系統中的每個節點，不論是伺服器或用戶代理，都有 SIP Signaling Transaction 的功能，用來收送 SIP 信令。最後，用戶代理負責與使用者互動 Create Group 以及 Push to Talk 的功能，其中 Push to Talk 功能也包含了傳送使用者音訊資料。

表 二、系統功能表

	功能名稱	描述
SIP	SIP Signaling Proxy	轉送所有 SIP 封包
	SIP Member Information	擁有所有 SIP UA 資訊
GLMS	3PCC - Make Call	送出邀請給名單內的 UA
	Group Managerment	群組管理以及 IPv6 Multicast Address 的指派
TBCP	Floor Control	控制每一個群組的發話權
ALL	SIP Signaling Transaction	收送 SIP Signaling
UA	Create group	建立群組並送出群組人員名單
	Push to talk	偵測使用者按下的按鍵，控制使用者發話

此外，底層則是採用了 IPv6 協定，除了可提供廣大的位址，IPv6 能透過鄰居發現協定(Neighbor Discovery Protocol, NDP)以及無狀態位址自動設定(Stateless Address Auto-configuration)之自動定址(Auto-Configuration)的特性，讓使用者不須手動設定位址，此外多播功能可以有效減少網路流量，更是本篇論文選擇 IPv6 作為網路層協定的原因。

doi:10.6837/ncnu202000205

第三節 SIP 模組

在本篇之 MPTT 系統中，採用 SIP 伺服器架構，與前章所提到之 SIP 架構相同，用戶代理需向 SIP 模組註冊(Register)。其中當用戶代理與各模組溝通時都須透過 SIP 模組來轉送 SIP 信令，SIP 模組也同時擁有群組中所有用戶代理的資訊。

第四節 GLM 模組

本篇所提出之系統中，GLM 伺服器與 OMA 所提出的 PoC 架構有些微不同。首先 MPTT 是由用戶代理利用 SIP Message[17]送出群組名單，而當 GLM 模組收到名單後，會將群組名單存成 XML 格式並且指定一 IPv6 多播位址給群組，之後邀請名單內的使用者加入群組(3PCC)，而非如 OMA 是透過 HTTP 傳遞訊息，也不是由使用者發起邀請。

第五節 TBCP 模組

本篇論文與 OMA 所提出之 TBCP 實作上有所不同，但目標理念是相同的。OMA 提出之 TBCP 協定是利用 RTCP 的 APP 傳送，而在 MPTT 系統中用戶代理是利用 SIP Message 向 TBCP 模組發出請求來得到發話權，其中請求的格式為 XML，裡面的資訊包含 IPv6 多播位址，也就是群組的 ID ，以及使用者的 SIP URI，讓 TBCP 模組能清楚得知使由哪個群組的哪一個使用者所送出的請求，以便日後加上優先權(Priority)的功能。

第六節 信令流程

MPTT 有別於 VoIP 技術，除了利用 SIP 協定中的註冊伺服器以及用戶代理以外，也參考原本 OMA 之 PoC 機制，更為重要的則是應用了 IPv6 多播位址實現點對點傳送。

圖九為本論文所提出隨按即說系統之 MPTT 信令流程，其中可以分為六個階

doi:10.6837/ncnu202000205

段，分別為註冊(Register)階段、發起群組(Invite) 、取得發話權(Floor)階段、聲音(Audio)發送、結束發話權(Release)以及取消群組(Bye)。完整信令流程可參考圖十。

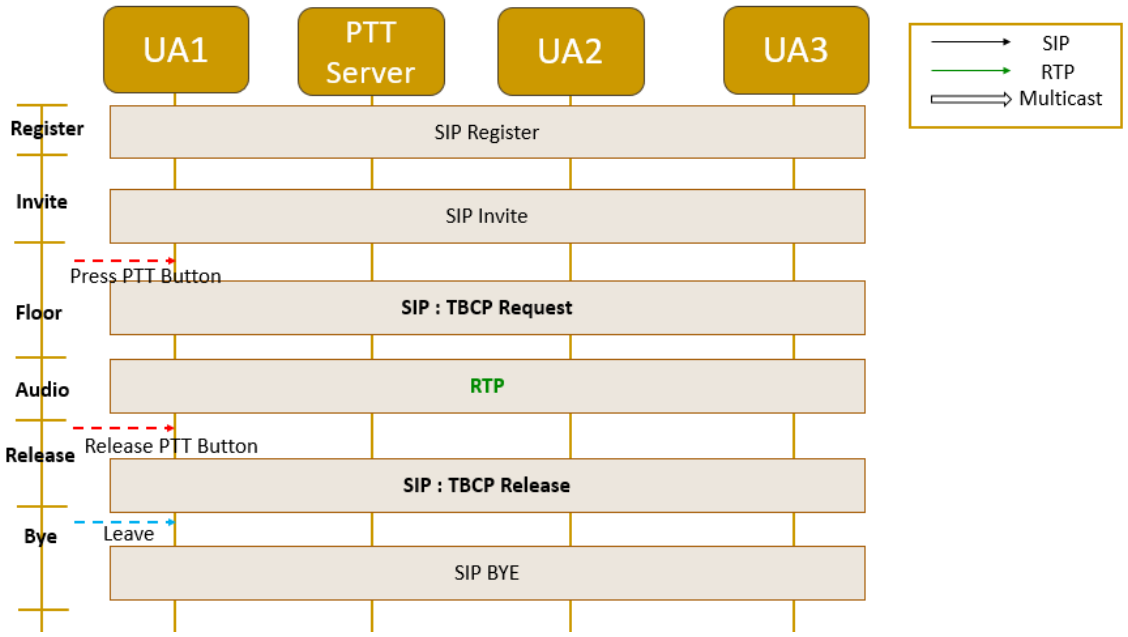

圖 九、MPTT 信令流程

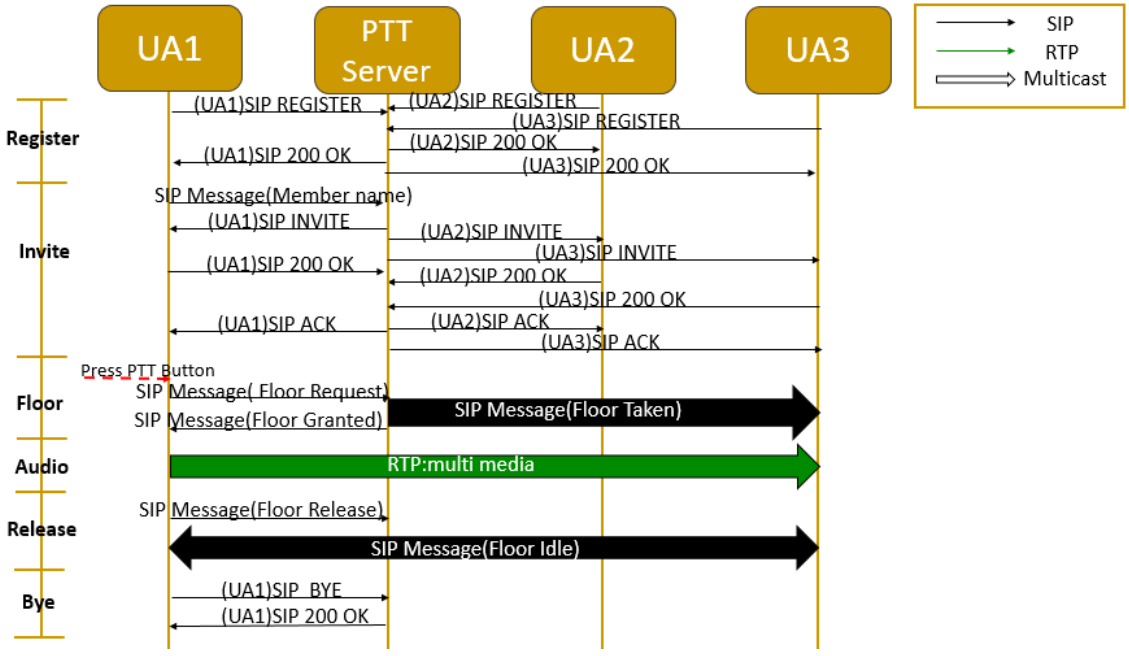

圖 十、MPTT 信令流程(詳細)

doi:10.6837/ncnu202000205

Step 1 (Register): 首先，使用者會進行註冊，因此用戶代理(UA1)需要向 SIP 伺服器發送 REGISTER 信令，在 SIP 伺服器回應 200 OK 的訊息後，則成功加入 MPTT 群組。

Step 2 (Invite): 註冊成功後，使用者可自行輸入人員名單並按下邀請，UA1 會先透過 SIP Message 傳送成員名單(Member list)，PTT 伺服器收到人員名單後，GLM 模組會分配一 IPv6 多播位址給此群組，並利用 GLM 模組的 3PCC-Make Call 功能向名單內的使用者發送夾帶著 IPv6 多播位址的 SDP 之 SIP INVITE，來告訴所有用戶代理多播群組的位址，當用戶代理收到 INVITE，會聽在此 IPv6 多播位址，並在回應 200 OK 之後，所有用戶代理即加入了此群組。

Step 3 (Floor): 取得發話權階段，當使用者按下隨按即說之按鈕時，會利用 SIP Message 的封包，將 TBCP Request 以及 IPv6 多播位址夾帶在 Content 中發送給 PTT 伺服器。在 PTT 伺服器收到此封包時，TBCP 模組會利用收到 IPv6 多播位址來察看此群組的發話權(Floor)是否有人佔用。若無人占用，TBCP 模組則會回送 TBCP Granted 訊息的封包給 UA1，並將 TBCP Taken 的 SIP Message 封包發給群組內所有用戶代理。UA1 收到 TBCP Granted 後，才能發話；反之則無法發話。

Step 4 (Audio): 發送對話階段，使用者得到發話權，UA1 會利用已知的多播位址，將使用者的語音串流利用多播的方式發送給此群組內的所有成員(也就是所有聽在這個 IPv6 多播位址的用戶代理) ，所以用戶代理只需送出一份聲音資料，就能讓所有群組成員都聽到。

Step 5 (Release): 當使用者放開按鈕，發話權結束，UA1 會傳送 TBCP Release 以及群組的 IPv6 多播位址給 PTT 伺服器。而 TBCP 模組會依據 IPv6 多播位址，把 TBCP Idle 的信令多播傳給群組內其他用戶代理。代表發話權現在沒有人使用。

Step 6 (Bye): 最後，在取消群組階段，UA1 可以任意離開已加入的群組。在送出 BYE 信令之後，GLM 模組會結束與 UA1 所建立的會話。雖然發起者 UA1 已離開此群組，其他群組內使用者仍能繼續通話。

doi:10.6837/ncnu202000205

第五章 系統實作

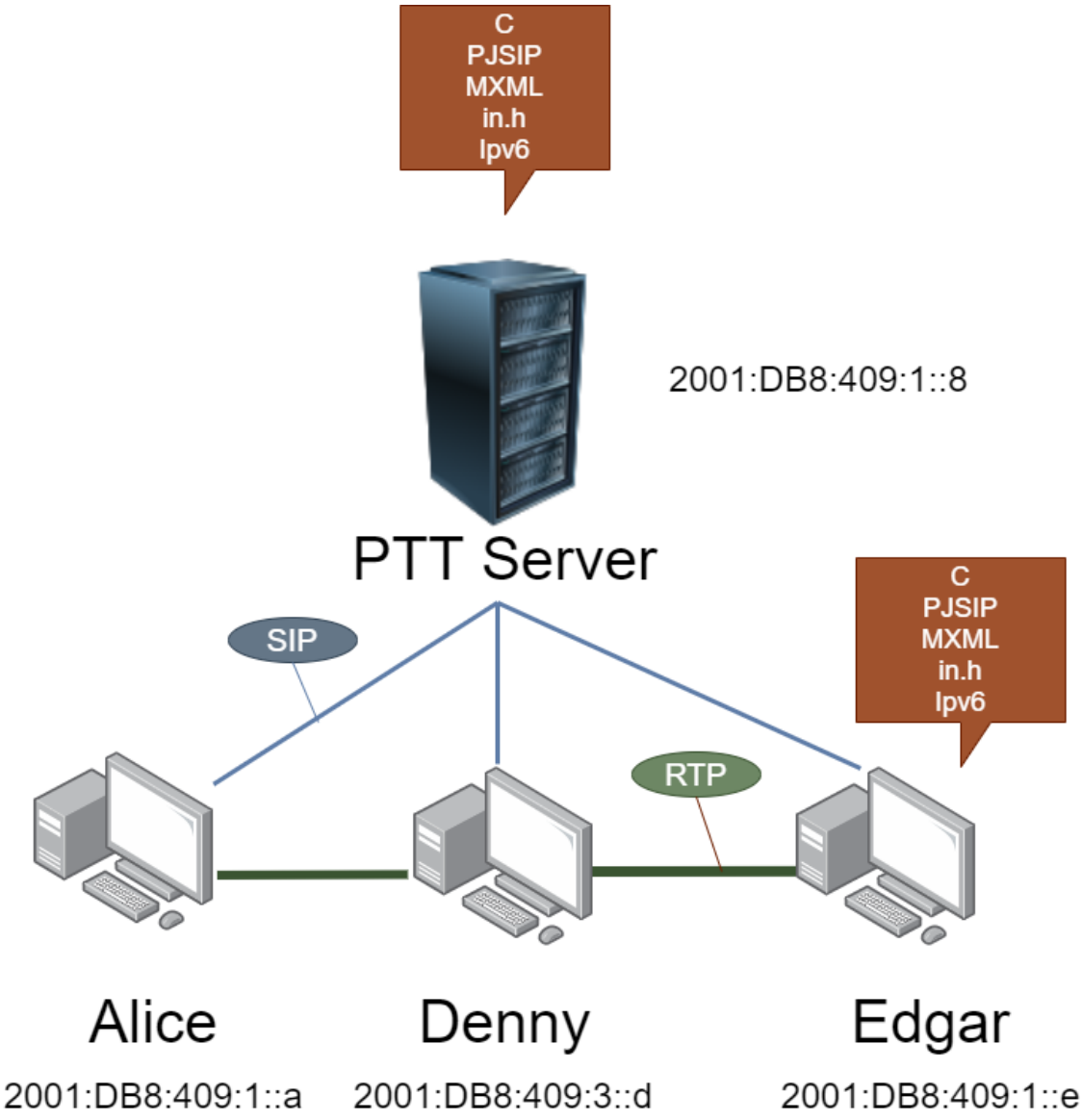

圖 十一、系統實作架構圖

基於第四章的系統架構，接下來將開始介紹實作的細節。圖十一為系統實作架構圖，在實作架構中，有一台 PTT 伺服器包含上一章所提之 SIP、GLM、TBCP 三個模組，及三個使用者，分別為 Alice、Denny 以及 Edgar。伺服器上三個模組，以及使用者的用戶代理都是由 C 語言撰寫完成，並用 PJSIP 做為主要函式庫，另外還使用了 MXML 來夾帶訊息，最後當然使用了 IPv6 傳送。其中因為 PJSIP 不支援 IPv6 Multicast[18]，所以這一部份自行撰寫後加入了 PJSIP Library。

doi:10.6837/ncnu202000205

其中要特別注意的是，當目的位址為 Multicast Address 時，Hop Limit 預設為 1，因此在做 IPv6 Multicast 時，Socket Option `IPV6_MULTICAST_HOPS()` 一定要做設定，否則 Multicast 封包只會在同一個子網段(subnet)中傳送。而圖十二為實作的網路拓樸圖，在拓樸圖中會有兩個 router 形成三個 subnet(子網路)，Alice、Edgar 與 PTT Server 都在同一個 subnet(2001:db8:409:1::1/64)，IPv6 address 分別為 2001:db8:409:1::a、2001:db8:409:1::e 與 2001:db8:409:1::8，而 Denny 則是 2001:db8:409:3::d。

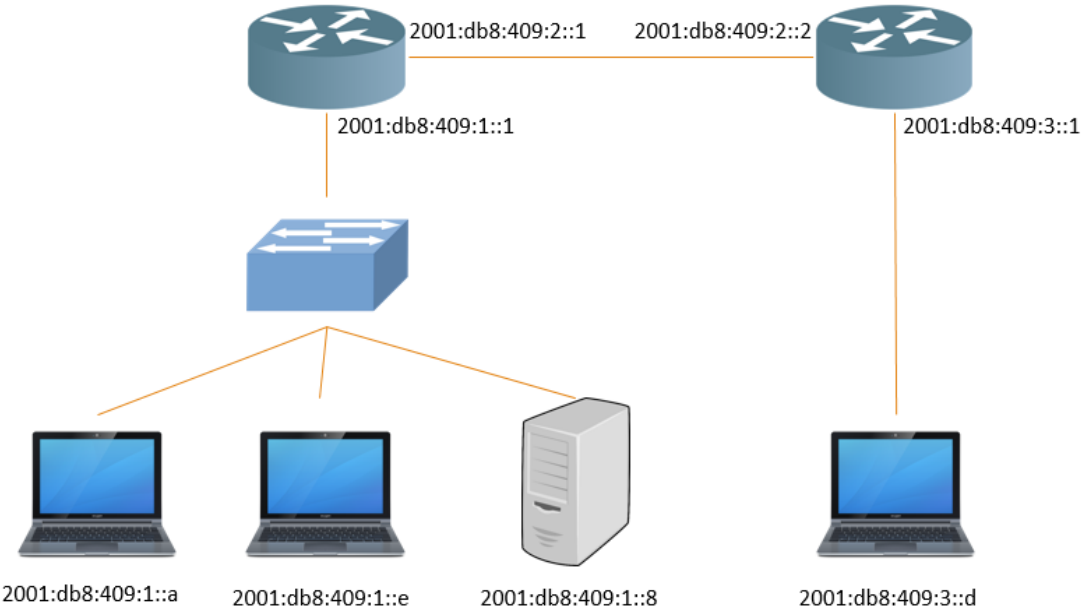

圖 十二、實作網路拓樸圖

以下依照表三次序，逐一呈現上一章所提系統之實作流程。

doi:10.6837/ncnu202000205

表 三、實作運作流程

	UA	Server
Register	三位使用者向 Server 進行註冊	SIP : 查看SIP Signaling
Invite	Denny 建立群組 (Member : Edgar and Alice)	GlM : 收到建立群組 Message 並送出邀請給群組成員
Floor	Denny按下按鈕想要發話	TBCP : 收到 Floor Request
Audio	1. Denny 收到 Floor Granted 開始發話 2. 查看 Edgar 與 Alice 的 Wireshark	
Release	Denny 結束發話	TBCP : 收到 Floor Release
Floor	Edgar想要發話	
Bye	Denny 離開群組	SIP : 查看SIP Signaling

第一節 Register

首先三位使用者會向 PTT 伺服器的 SIP 模組進行註冊，如圖十三為 Alice 的使用者畫面，所以在 SIP 模組的 Wireshark 畫面可以看到會有來自三位使用者的 SIP Register，如圖十四有來自 2001:db8:409:1::a(Alice)、2001:db8:409:1::e(Edgar)以及 2001:db8:409:3::d(Denny)的 SIP 封包，以及由 SIP 模組所回應的 200 OK。

```
Alice > ./ptt
22:59:31.079        os_core_unix.c !pjlib 2.9 for POSIX initialized
22:59:31.081        sip_endpoint.c  .Creating endpoint instance...
22:59:31.081                 pjlib  .select() I/O Queue created (0x55755ac63c60)
22:59:31.081        sip_endpoint.c  .Module "mod-msg-print" registered
22:59:31.081       sip_transport.c  .Transport manager created.
22:59:31.081          pjsua_core.c  .PJSUA state changed: NULL --> CREATED
Registration Finished!
Press 'h' to hangup all calls, 'm' to make call, 'q' to quit
```

圖 十三、使用者端完成註冊畫面

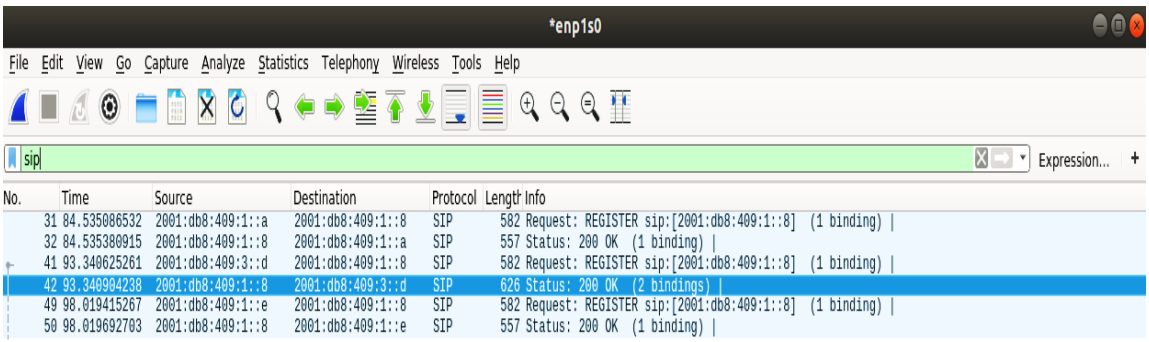

圖 十四、SIP Module 收到三位使用者 Register(Wireshark)

doi:10.6837/ncnu202000205

第二節 Invite

其次 Alice 建立群組(輸入 m)，故而 PTT 伺服器的 GLM 模組會收到來自 Alice 的使用者名單，GLM 模組這邊會指派(assign)給群組一個 IPv6 Multicast Address。圖十五所示即為群組的 IPv6 Multicast Address ff08::10 以及成員名單(SIP URI)。這時候 GLM 模組會根據收到的使用者名單來向三位使用者送出邀請，所以我們在使用者的畫面，如圖十六為 Alice 的畫面，會有 Incoming Call 代表現在有電話進來。三位使用者會接起電話(輸入 a)，如圖十七，此時三位使用者就都加入群組了。

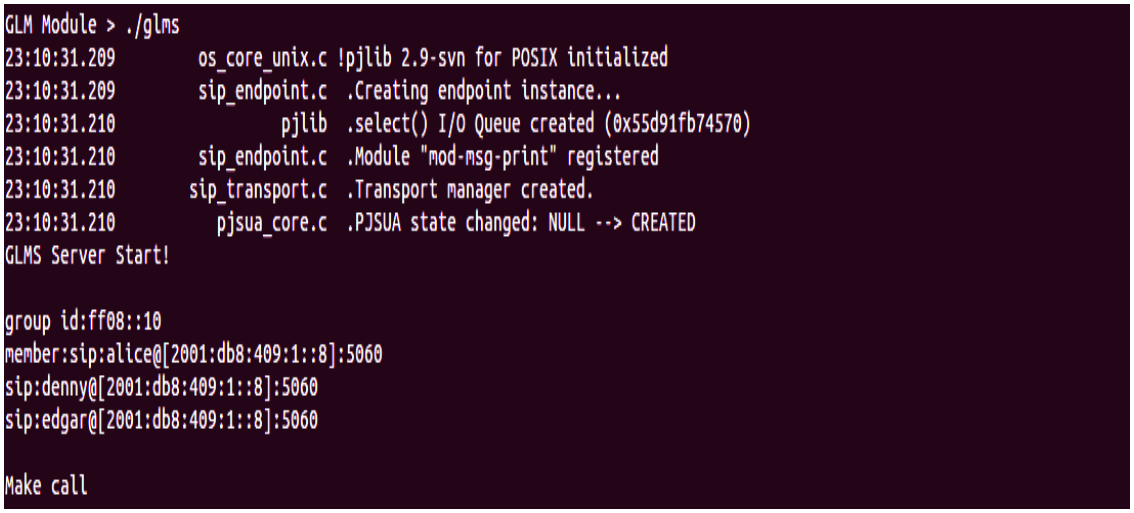

圖 十五、GLM Module 畫面

doi:10.6837/ncnu202000205

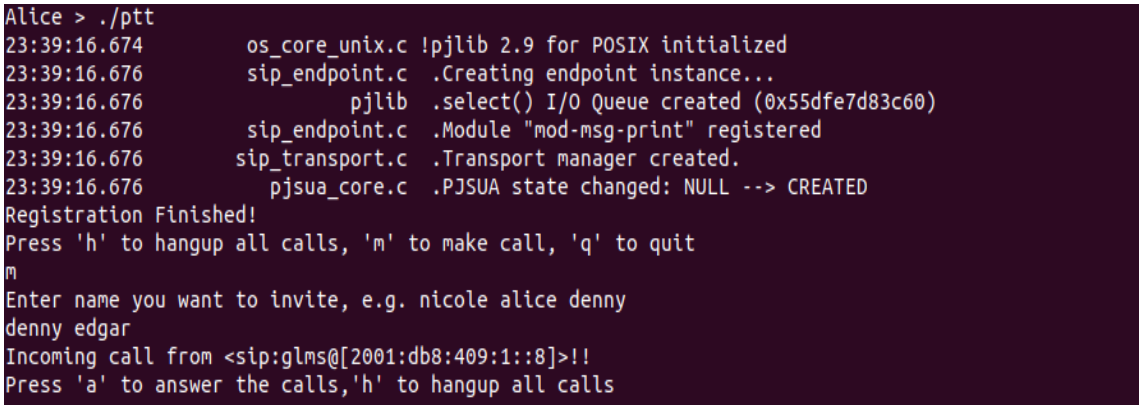

圖 十六、使用者收到邀請

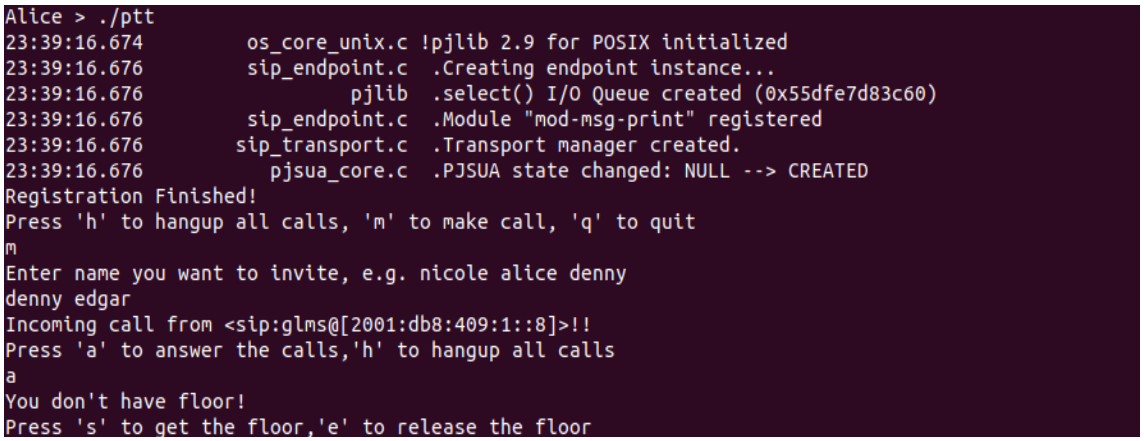

圖 十七、使用者輸入 a 接起電話

第三節 Floor

三位都加入群組後，Alice 想進行發話，所以當 Alice 按下發話按鈕 s，會送出 TBCP Request 給 PTT 伺服器的 TBCP Module，所以我們可以在 TBCP Module 模組的畫面上看到來自群組 ff08::10 的 Alice 送出了一個 TBCP Request，並且 TBCP Module 回應了 TBCP Granted，如圖十八，所以這時候 Alice 就能開始發話了。

doi:10.6837/ncnu202000205

```
TBCP Module > ./tbcps
23:36:53.999          os_core_unix.c !pjlib 2.9-svn for POSIX initialized
23:36:54.000          sip_endpoint.c  .Creating endpoint instance...
23:36:54.000                   pjlib  .select() I/O Queue created (0x559bc12fd570)
23:36:54.000          sip_endpoint.c  .Module "mod-msg-print" registered
23:36:54.000         sip_transport.c  .Transport manager created.
23:36:54.000           pjsua_core.c  .PJSUA state changed: NULL --> CREATED
TBCP Server Start!
Press 'h' to hangup all calls, 'q' to quit
Request from group ID ff08::10
 member <sip:alice@[2001:db8:409:1::8]>
Floor Granted
```

圖 十八、TBCP Module - TBCP Request & Granted

第四節　Audio

當 Alice 開始發話，我們可以從 Denny 的 Wireshark 畫面上清楚看到，有很多 RTP 封包由 2001:db8:409:1::a(Alice) 傳送到 ff08::10，如圖十九，而 Edgar 因為也加入了這個群組，所以當然也會收到 Alice 的聲音封包，如圖二十。

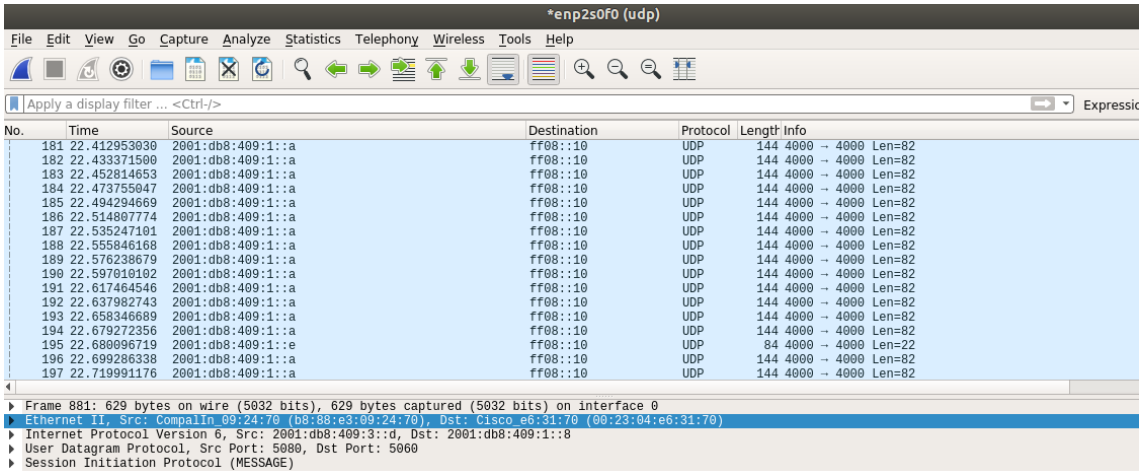

圖 十九、Denny 收到送給 ff08::10 的 RTP 封包

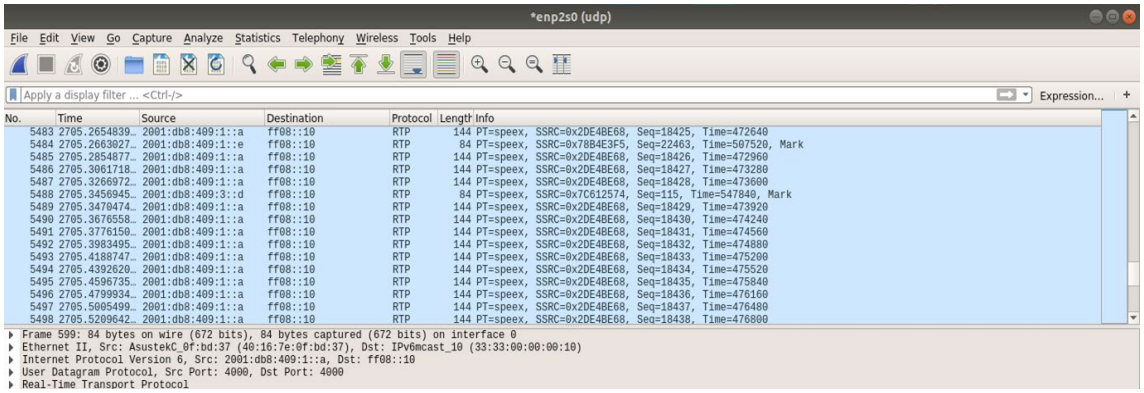

圖 二十、Edgar 收到送給 ff08::10 的 RTP 封包

doi:10.6837/ncnu202000205

第五節　Release

Alice 發話結束時，可按下 e 來釋出發話權，此時她的 UA 會向 PTT 伺服器的 TBCP Module 送出 TBCP Release，所以我們可以在 TBCP Module 模組的畫面上看到來自群組 ff08::10 的 Alice 送出了一個 TBCP Release，如圖二十一。所以這時候 Alice 就不能再發話了，我們可以從 Denny 的 Wireshark 畫面上清楚看到，如圖二十二，雖然還是有收到 RTP 封包，但是封包大小為 84 bytes，payload 是空的(如果有傳聲音是 144 bytes)，這是一個代表目前還 alive 的訊息，並非聲音的封包，也不會像聲音封包傳送的那麼頻繁。

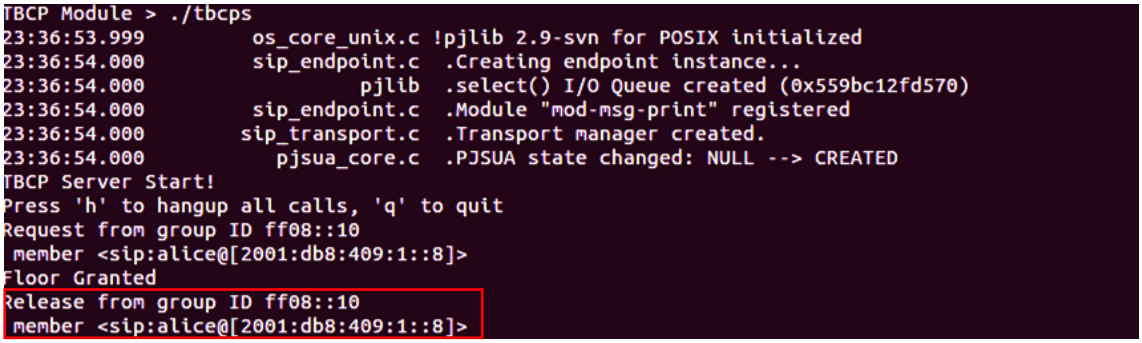

圖 二十一、TBCP Module - TBCP Release

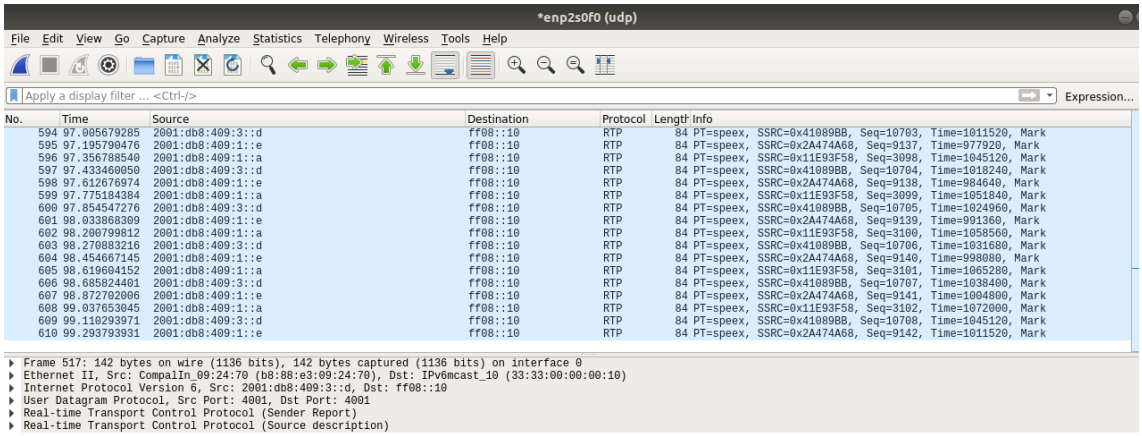

圖 二十二、Denny 收到送給 ff08::10 的 RTP alive 封包

第六節　BYE

當 Alice 想要離開群組，會向 PTT 伺服器的 SIP module 送出 SIP BYE 封包，如圖二十三，我們可以看到來自 2001:db8:409:1::a(Alice)的 SIP 封包，結束與

doi:10.6837/ncnu202000205

GLM Module 建立的會話。

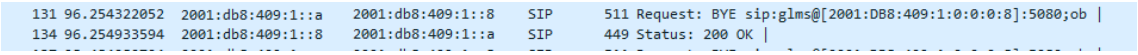

圖 二十三、SIP Module 收到 Alice 的 SIP BYE

第七節 Floor from Denny

Alice 離開群組後，並不影響剩下的人通話，這時換 Denny 想發話了，我們一樣可以在 Edgar 的 Wireshark 畫面清楚看到由 2001:db8:409:3::d(Denny)傳送到 ff08::10 群組的 RTP 封包，如圖二十四，並且這時候只有 2001:db8:409:1::e(Edgar)的 alive 封包已經沒有來自 2001:db8:409:1::a(Alice)的 alive 封包。

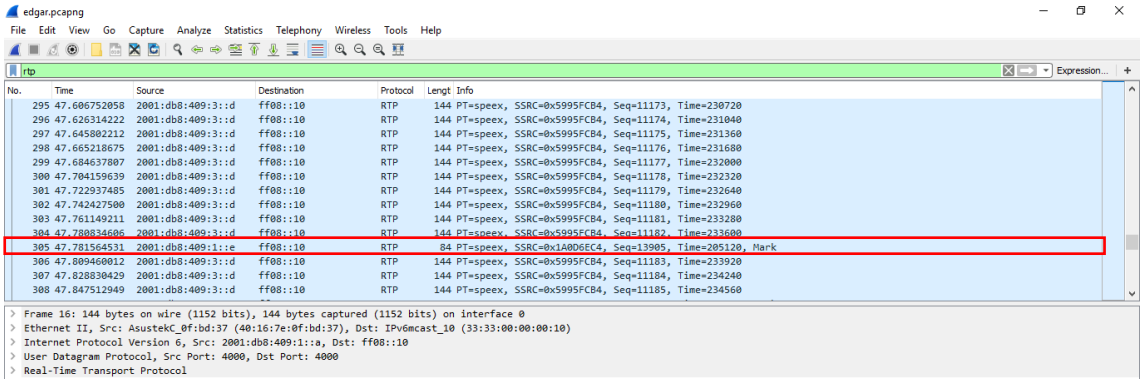

圖 二十四、Edgar 收到送給 ff08::10 的 RTP 封包

doi:10.6837/ncnu202000205

第六章　系統功能分析與比較

表四列出 OMA 提出之 PoC 架構與本篇論文提出之 MPTT 系統架構的比較表，而詳細流程比較將會在之後的小節介紹，其中群組人數會以 3 人為例子。

表 四、OMA PoC 與 MPTT 比較表

OMA PoC (p.10)	MPTT(p.17)
按下 PoC 按鈕才邀請群組成員加入	事先邀請群組成員建立群組
由使用者送出邀請	由Server送出邀請
RTCP 傳送 TBCP Floor	SIP 傳送 Floor
TBCP Floor 為 Unicast 傳送	Floor 為 Multicast 傳送
RTP 為 Unicast 傳送	RTP 為 Multicast 傳送
RTP 須經過 Server	RTP 不需經過 Server
建立群組的使用者離開群組後群組解散	使用者離開群組並不影響群組運作

第一節　建立群組的方式

OMA 架構中，如圖二十五，是按下按鈕後，才開始邀請成員加入群組以及請求發話權；

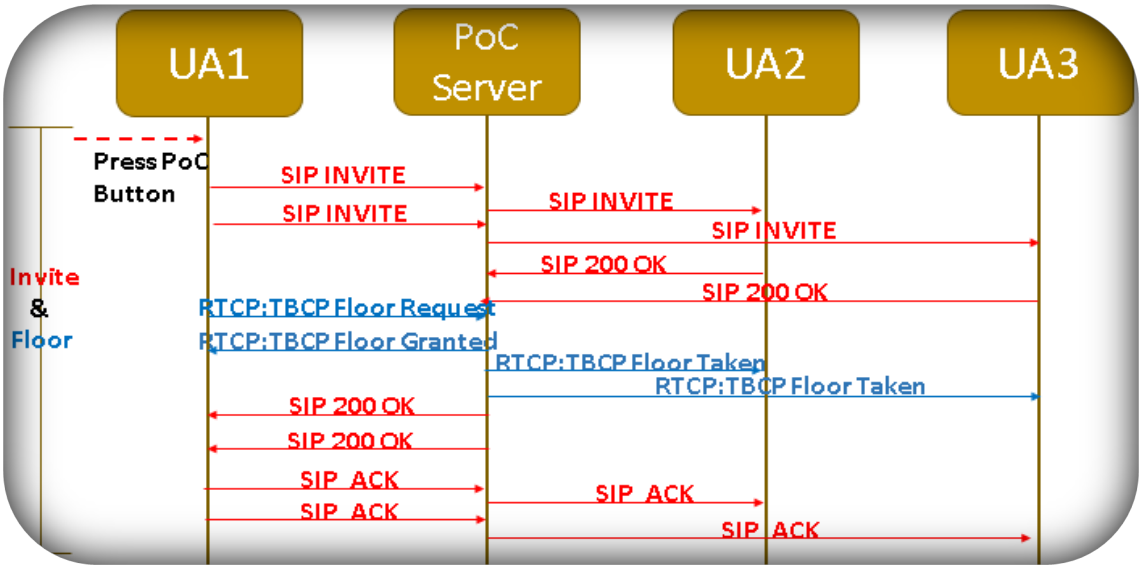

圖 二十五、OMA PoC Invite

doi:10.6837/ncnu202000205

而在 MPTT 系統中，如圖二十六，為事先邀請使用者建立群組才開始請求發話權，所以按下發話按鈕後拿到發話權就能馬上發話，不需等待邀請完其他使用者，減少按下按鈕後所需等待的時間。

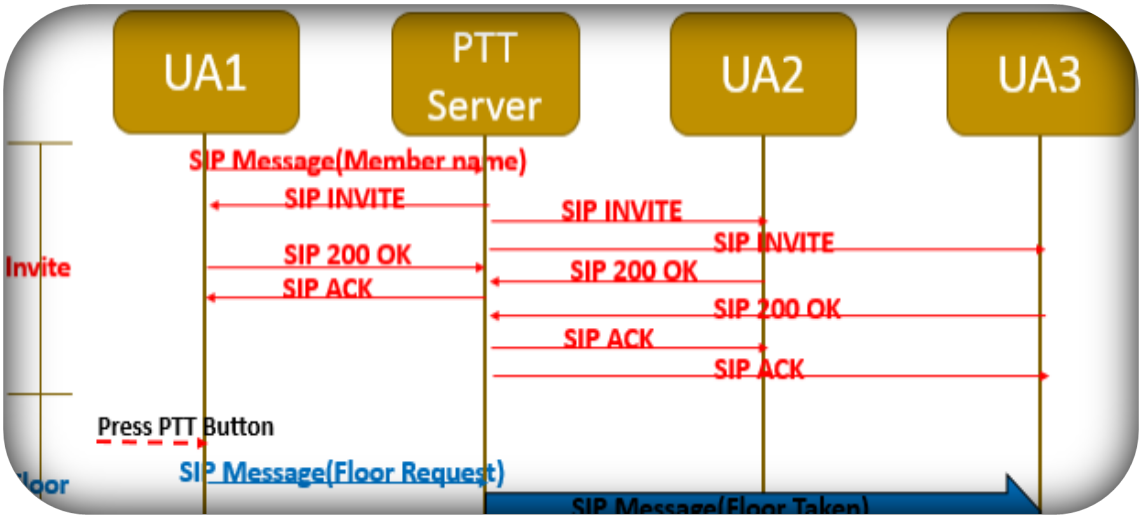

圖 二十六、MPTT Invite

第二節　發起邀請方式

OMA 架構中，如圖二十五，是由用戶代理送出邀請，經 PTT 伺服器轉送給其他用戶代理，而邀請一個使用者都需要經過 SIP INVITE、200 OK 以及 ACK 並且經過伺服器轉送，所以群組人數為 n 時，Invite 步驟所需的封包數為 6(n-1)，當群組人數為 3，所需封包數為 12；而在 MPTT 系統中，如圖二十六，首先由使用者送出夾帶群組名單的 SIP Message 給 PTT 伺服器，PTT 伺服器中的 GLM Module 收到群組名單後會送出邀請給所有用戶代理，不需再經過伺服器轉送的步驟，所以群組人數為 n 時，Invite 步驟所需的封包數為 3n+1 ，1 為一開始送出的 SIP Message，當群組人數為 3，所需封包數為 10。

第三節　傳送 Floor 的協定

OMA 架構中，如圖二十七，是用 RTCP 的 APP 封包夾帶 TBCP Floor 的封包送出；

doi:10.6837/ncnu202000205

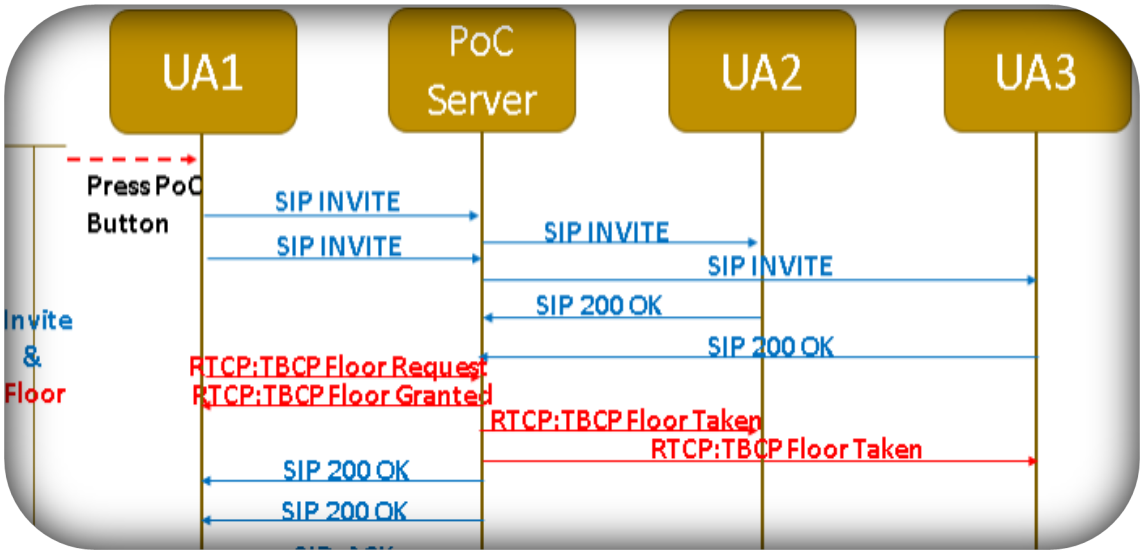

圖 二十七、OMA PoC Floor

而在 MPTT 系統中，如圖二十八，是利用 SIP Message 傳送 Floor 訊息。由於 TBCP 是信令(Signaling)訊息，而 RTCP 協定原設計是負責計算 QoS，兩者相較之下， SIP 更適合傳送 TBCP 信令，也比較統一。

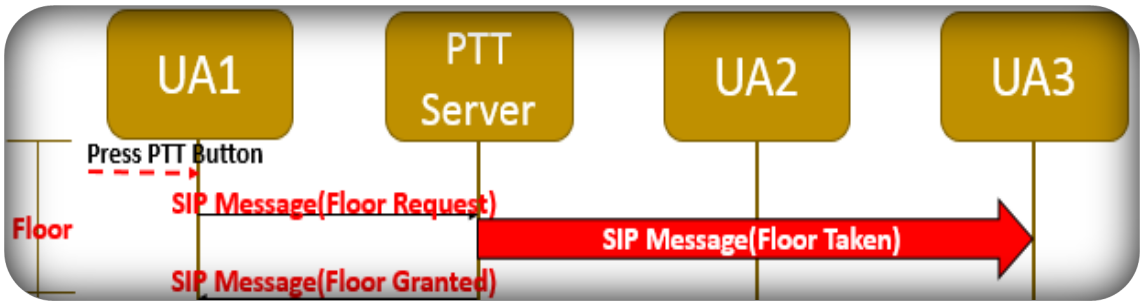

圖 二十八、MPTT Floor

第四節　Floor 為 Unicast /Mulitcast

OMA 架構中，如圖二十七，PoC 伺服器是用 Unicast 的方式傳送 TBCP Taken 的封包，PoC 伺服器需要送出兩個 Taken 封包以及一個 Granted 封包，所以群組人數為 n 時，取得發話權(Floor) 步驟所需的封包數為 n+1 ，當群組人數為 3，所需封包數為 4；而在 MPTT 系統中，如圖二十八，PTT 伺服器是用 Multicast 傳送 Floor Taken 訊息，只需要送出一份，所以 Floor 步驟不因成員增加而需要送出更多 Floor Taken 封包。

doi:10.6837/ncnu202000205

OMA 架構中，如圖二十九，PoC 伺服器是用 Unicast 的方式傳送 TBCP Idle 的封包，PoC 伺服器需要送出三個 Idle 封包，所以群組人數為 n 時，Release 步驟所需的封包數為 n+1 ，當群組人數為 3，所需封包數為 4；

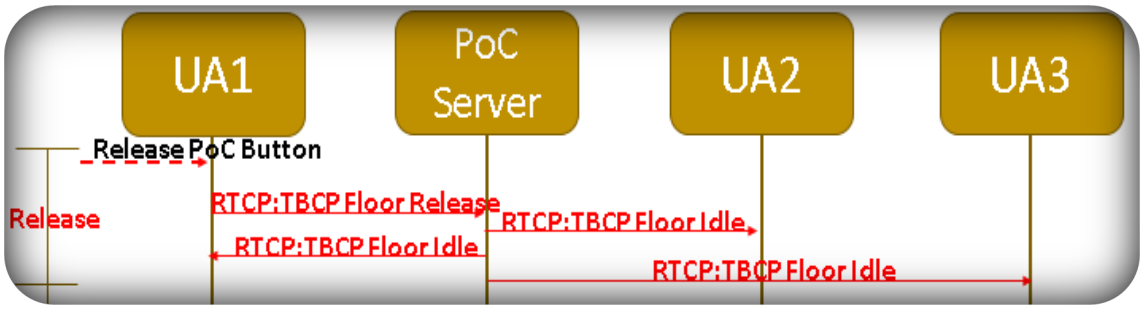

圖 二十九、OMA PoC Release

而在 MPTT 系統中，如圖三十，PTT 伺服器是用 Multicast 傳送 Floor Idle 訊息，只需要送出一份，所以 Release 步驟不因成員增加而需要送出更多 Floor Idle 封包。

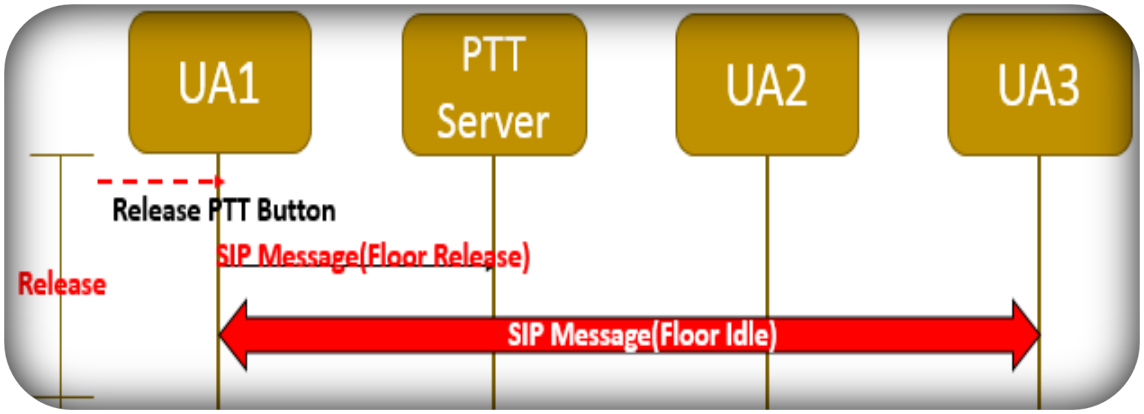

圖 三十、MPTT Release

第五節　RTP 的 Unicast /Mulitcast

OMA 架構中，如圖三十一，用戶代理是用 Unicast 的方式傳送 RTP 的封包，經過 PoC 伺服器轉送給另外兩個用戶代理，所以當成員增加，伺服器所需送出的聲音封包就會倍數成長，所以群組人數為 n 時，t 為時間，a 為每秒封包數，Audio 步驟所需的封包數為 nta ，當群組人數為 3，所需封包數為 3 ta；

doi:10.6837/ncnu202000205

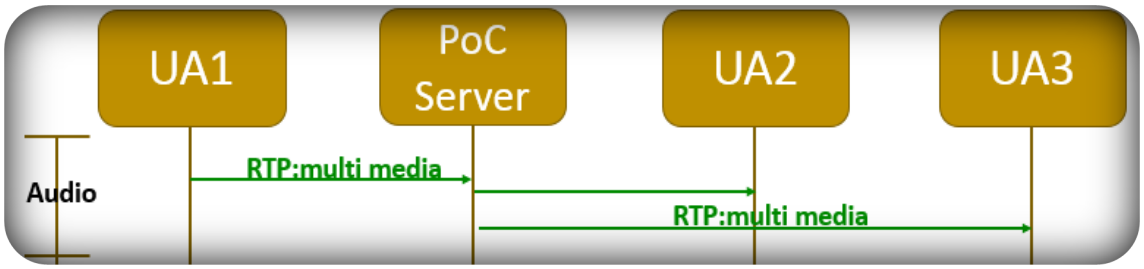

圖 三十一、OMA PoC Audio

而在 MPTT 系統中，如圖三十二，用戶代理是用 Multicast 傳送，聲音訊息只需要送出一份，*t* 為時間，*a* 為每秒封包數，Audio 步驟不因成員增加而需要送出更多的封包，所需封包數為 *ta*。

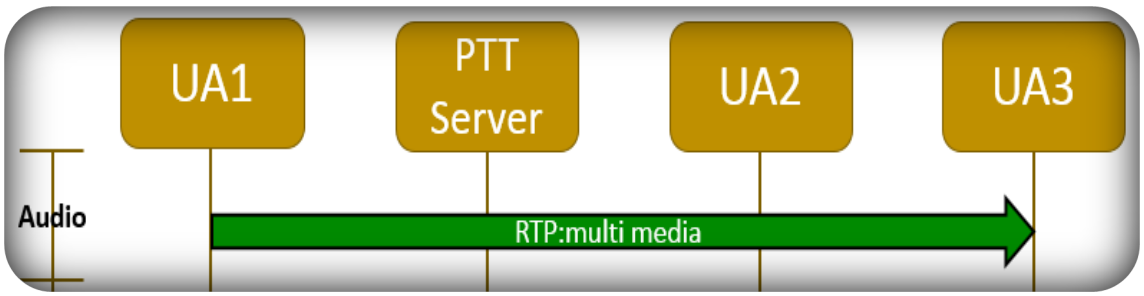

圖 三十二、MPTT Audio

第六節　Back to Back /Proxy

OMA 架構中，如圖三十一，用戶代理傳送 RTP 的封包，需經過 PoC 伺服器轉送給另外兩個用戶代理，所以當伺服器在遠處，轉送所需的時間也增加；而在 MPTT 系統中，如圖三十二，聲音訊息不需經過伺服器轉送，減少經過伺服器所需等待的時間。

第七節　離開群組的方式

OMA 架構中當發起通話的使用者離開群組，如圖三十三，群組也會跟著解散，每結束一個會話都需要經過 SIP BYE 以及 200 OK，所以群組人數為 n 時，Bye 步驟所需的封包數為 2n，當群組人數為 3，所需封包數為 6；

doi:10.6837/ncnu202000205

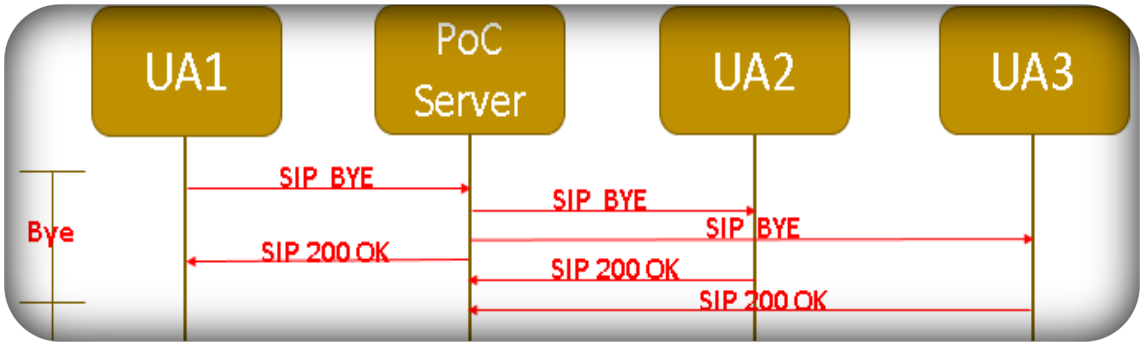

圖 三十三、OMA PoC Bye

而在 MPTT 架構中，如圖三十四，使用者送出 SIP BYE 以及收到 200OK，就能離開群組，Bye 步驟所需封包數為 2，而原本群組內的使用者還是能繼續通話，不因為人數而有改變。

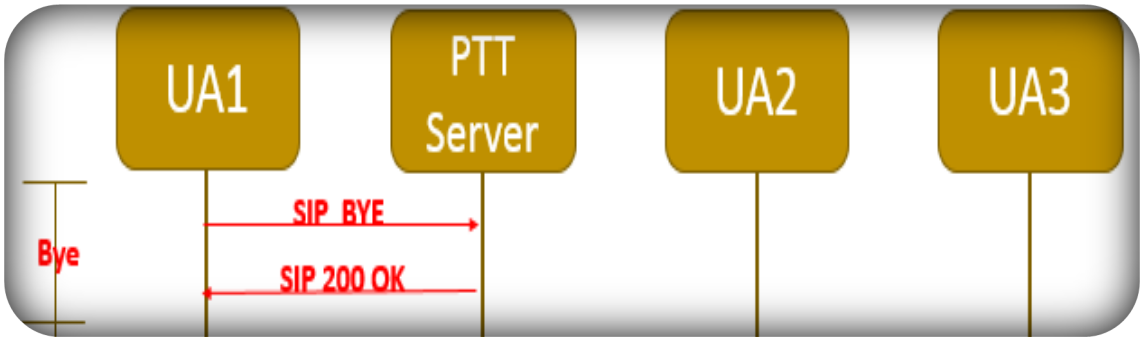

圖 三十四、MPTT Bye

第八節　效能比較

最後依每步驟的封包數總結如表五，可以得知，在原 PoC 架構中從建立群組到通話並離開群組需要 12n-4+n*ta*，而在本篇提出之 MPTT 架構中只需要 5n+9+n*ta*。

doi:10.6837/ncnu202000205

表 五、各步驟封包數比較(人數為 n，*t* 為時間，*a* 為每秒封包數)

	OMA PoC	MPTT
Register	2n	2n
Invite	6(n-1)	3n+1
floor	n+1	3
Audio	nta	ta
Release	n+1	2
Bye	2n	3
Total	12n-4+n*ta*	5n+9+*ta*

doi:10.6837/ncnu202000205

第七章　結論與未來展望

第一節　未來展望

新的應用也希望能相容於舊的架構，所以為了能讓原有的 VoIP 電話能使用 MPTT 架構中，本篇參考[19, 20]提出了以下的架構，將於未來實現。

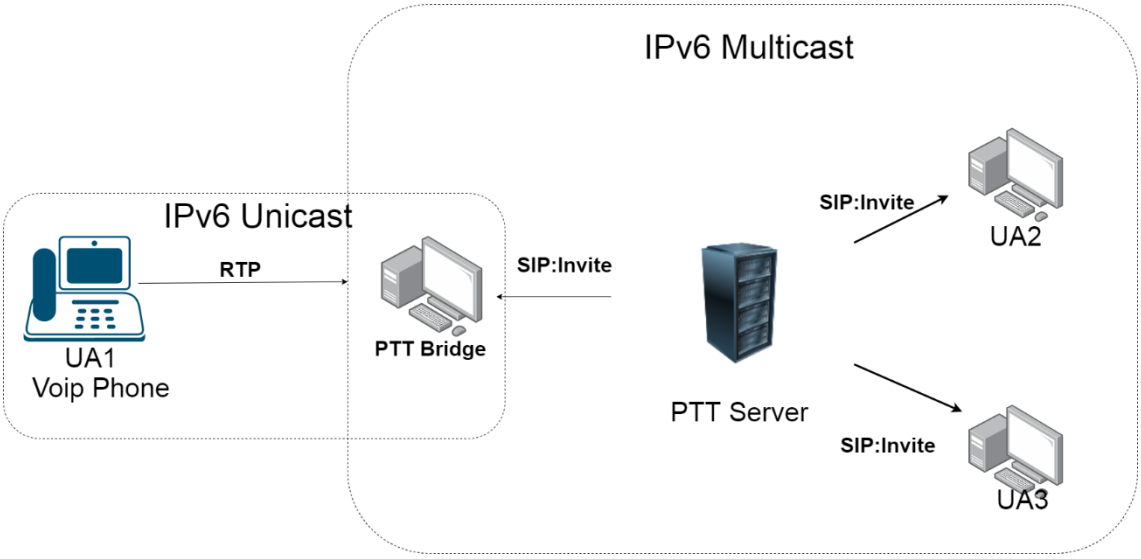

圖 三十五、VoIP 電話使用於 MPTT 系統

如圖三十五，在 VoIP 電話以及 PTT 伺服器中間，需要一台 PTT Bridge，當 VoIP 電話 UA1 接上網路以後會送出 SIP 註冊封包，PTT Bridge 收到後會直接轉送給 PTT 伺服器的 SIP Module，完成註冊。

如果 UA1 想打普通網路電話 UA2，PTT Bridge 不管是收到 SIP INVITE 還是夾帶電話號碼 DTMF 的 RTP 都會直接轉送給 PTT 伺服器的 SIP Module，由 SIP Module 再轉送 INVITE 給 UA2，完成邀請後，就能開始通話。

如果 UA1 想使用 MPTT 服務，當按下指定好的代號後(例如 3 次井字號)，就能使用 MPTT 服務，這時當 PTT Bridge 收到 SIP INVITE 以及夾帶 DTMF 的 RTP 後，會把所要邀請的使用者名單用 SIP Message 送給 PTT 伺服器，PTT 伺服器的 GLM 模組收到後會邀請 PTT Bridge 以及被邀請的其他 UA 加入群組，而當 PTT

doi:10.6837/ncnu202000205

Bridge 收到 SIP 邀請也會與 UA1 建立會話，當 UA1 回應 200 OK，PTT Bridge 也會回應 200 OK 給 GLM 模組，這時群組就建立完成了。

當 UA1 想要發話會按下指定好的代號後(例如 2 次星字號)，這時 PTT Bridge 會送出用 SIP 送出 Floor Request 給 PTT 伺服器的 TBCP 模組，如果收到 Floor Granted，PTT Bridge 會向 UA1 送出一段聲音表示可以開始說話了，這時當 UA1 開始說話，其送出的聲音都會先送到 PTT Bridge，PTT Bridge 會將 RTP 送往它所加入的多播群組。反之，當 PTT Bridge 收到由其他 UA 送往多播群組的聲音，也會用 Unicast 的方式傳送給 UA1，完成普通網路電話也能整合於 MPTT 系統的應用。

第二節　結論

行動通訊的推廣以及無線網路的蓬勃發展，都使得上網更為便利，且頻寬的限制也不再是問題，因此促成 PTT 的發展與使用更為容易。功能面而言，PTT 是一個被廣為使用的功能。雖然目前在網際網路上語音串流服務非常普遍，但如何更方便快速即時的傳送語音訊息，是本篇論文想要達成的目標。尤其針對 IPv6 網路環境，透過多播的傳輸方式，達成點對點即時語音的交談，是本篇論文主要的研究議題。

doi:10.6837/ncnu202000205

參考文獻

[1] Y. Lair, and G. Mayer, “Mission Critical Services in 3GPP,” June 2017. (https://www.3gpp.org/news-events/1875-mc_services)

[2] O. M. Alliance, “Push to talk over Cellular (PoC)-Architecture,” *OMA-AD-PoC-V1 0-20060609-A*, 2006.

[3] B. Goode, “Voice over Internet protocol (VoIP),” *Proceedings of the IEEE,* vol. 90, no. 9, pp. 1495-1517, 2002.

[4] H. S. J. Rosenberg, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler, “Session initiation protocol,” *IETF RFC 3261,* June 2002.

[5] M. Tsai, and Y. Lin, “Talk burst control for push-to-talk over cellular,” *IEEE Transactions on Wireless Communications,* vol. 7, no. 7, pp. 2612-2618, 2008.

[6] J. Rosenberg, J. Peterson, H. Schulzrinne, and G. Camarillo, "Best Current Practices for Third Party Call Control (3pcc) in the Session Initiation Protocol (SIP)," *IETF RFC 3725*, April 2004.

[7] W. Hongtao, J. Yuehui, W. Wendong, M. Jian, and Z. Dongmei, "The performance comparison of PRSCTP, TCP and UDP for MPEG-4 multimedia traffic in mobile network," *Proceedings of the International Conference on Communication Technology (ICCT) 2003*, vol.1, pp. 403-406.

[8] R. Hinden, and S. Deering, “IP version 6 addressing architecture,” *IETF RFC 2373,* July 1998.

[9] S. Myung-Ki, K. Ki-Il, K. Dong-Kyun, and K. Sang-Ha, “Multicast delivery using explicit multicast over IPv6 networks,” *IEEE Communications Letters,* vol. 7, no. 2, pp. 91-93, 2003.

[10] M. Dawei, L. Binde, Z. Fengbin, and S. Chen, "Research and Realization of PoC Group List Management Server," August 2012. (https://doi.org/10.2991/iccasm.2012.252)

[11] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, " RTP: A transport protocol for real-time applications," *IETF RFC 3550*, July 2003.

[12] M. Handley, V. Jacobson, and C. Perkins, "SDP: session description protocol," *IETF RFC 2327*, April 1998.

[13] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, " Hypertext Transfer Protocol--HTTP/1.1," *IETF RFC 2616*, June 1999.

[14] A. Parthasarathy, "Push to talk over cellular (PoC) server," *Proceedings of the IEEE Networking, Sensing and Control*, pp. 772-776, March 2005.

[15] C.-M. Jen, "Method for implementing push-to-talk over SIP and multicast RTP related system," *US Patent 7,620,413*, November 2009.

doi:10.6837/ncnu202000205

[16] H. Liu, Y. Li, and Y. Zhang, "IPv6 multicast implementation and security solutions base on .Net platform," *International Conference on Mechatronic Science, Electric Engineering and Computer (MEC)* , pp. 1780-1782, August 2011.

[17] B. Campbell, J. Rosenberg, H. Schulzrinne, C. Huitema, and D. Gurle, "Session Initiation Protocol (SIP) Extension for Instant Messaging"," *IETF RFC 3428*, December 2002.

[18] R. Gilligan, S. Thomson, J. Bound, and W. Stevens, "Basic socket interface extensions for IPv6," *IETF RFC 3493*, February 2003.

[19] D. Ostroff, and A. Shaham, "Access gateway, softswitch and telephone for push-to-talk telephony," *US Patent 20060178138A1*, August 2006.

[20] G. Bhutiani, and A. Seth, "Push to talk over cellular (half-duplex) to full-duplex voice conferencing," *US Patent 20060229093A1*, October 2006.

doi:10.6837/ncnu202000205

附錄

附錄一　架設 MPTT 系統

本篇論文實作系統的安裝方式如下：

1. 需要環境：

   I. Ubuntu 18.04

   II. Kamailio 5.3

2. Kamailio 安裝(SIP Module)

   I. 安裝

      $ sudo apt install kamailio kamailio-mysql-modules

      $ sudo apt install kamailio-websocket-modules

   II. 檢查版本

      $ kamailio –V

   III. 設定 Kamailio

      $ sudo -i

      $ cd /etc/kamailio/

      $ cp kamailio.cfg kamailio.cfg.bkg

      $ vim kamailio.cfg

      # listen=udp:10.0.0.10:5060

      listen = [你的 IPv6 Address]

   IV. 重啟 Kamailio

      $ service kamailio restart

   V. 秀出現在 Kamailio 狀態以及註冊的人員

      $ kamctl ul show

doi:10.6837/ncnu202000205

3. MXML 安裝

$ wget

https://github.com/michaelrsweet/mxml/releases/download/v3.1/mxml-3.1.tar.gz

$ tar -xzvf mxml-3.1.tar.gz && cd mxml-3.1

$ ./configure

$ make

$ sudo make install

$ export LD_LIBRARY_PATH=/lib:/usr/lib:/usr/local/lib

4. IPv6 Multicast PJSIP 安裝

$ apt-get install build-essential libasound2-dev pkg-config

$ sudo apt install git

$ git clone https://github.com/IishaWu/push-to-talk-with-pjsua.git

$ cd push-to-talk-with-pjsua/

$ ./configure --enable-shared

$ make dep

$ make

$ sudo make install

$ export LD_LIBRARY_PATH=/lib:/usr/lib:/usr/local/lib

5. GLM Module 啟用

$ cd ../PTT/

$ gcc glms.c -o glms `pkg-config --cflags --libs libpjproject mxml`

$ ./glms

6. TBCP Module 啟用

$ cd ../PTT/

$ gcc tbcps.c -o tbcps `pkg-config --cflags --libs libpjproject mxml`

doi:10.6837/ncnu202000205

$ ./tbcps

7. 如遇到 Make Error

$ make distclean

$ make clean

$ make realclean

8. 如果出現找不到 Default Audio Device

請新開一個 Terminal

$ aplay -l

$ sudo apt-get install libasound2-dev

$ cd push-to-talk-with-pjsua/

$ ./configure --enable-shared

$ make dep

$ make

$ sudo make install

附錄二 使用者端安裝

本篇論文實作之使用者的用戶代理的安裝方式如下：

1. MXML 安裝

$ wget

https://github.com/michaelrsweet/mxml/releases/download/v3.1/mxml-3.1.tar.gz

$ tar -xzvf mxml-3.1.tar.gz && cd mxml-3.1

$ ./configure

$ make

$ sudo make install

$ export LD_LIBRARY_PATH=/lib:/usr/lib:/usr/local/lib

2. IPv6 Multicast PJSIP 安裝

$ apt-get install build-essential libasound2-dev pkg-config

$ sudo apt install git

$ git clone https://github.com/IishaWu/push-to-talk-with-pjsua.git

doi:10.6837/ncnu202000205

$ cd push-to-talk-with-pjsua/

$ ./configure --enable-shared

$ make dep

$ make

$ sudo make install

$ export LD_LIBRARY_PATH=/lib:/usr/lib:/usr/local/lib

$ cd ../PTT/

$ gcc ptt.c -o glms `pkg-config --cflags --libs libpjproject mxml`

$ ./ptt

3. 如遇到 Make Error

$ make distclean

$ make clean

$ make realclean

4. 如果出現找不到 Default Audio Device

請新開一個 Terminal

$ aplay -l

$ sudo apt-get install libasound2-dev

$ cd push-to-talk-with-pjsua/

$ ./configure --enable-shared

$ make dep

$ make

$ sudo make install

5. How to make call

依畫面指示按下 m 可以輸入想要邀請的使用者 e.g. nicole alice

依畫面指示按下 a 可以接受通話

依畫面指示按下 s 可以取得發化權並發話

依畫面指示按下 e 可以釋放發化權

doi:10.6837/ncnu202000205

依畫面指示按下 h 可以結束通話

依畫面指示按下 q 可以結束通話

附錄三　IPv6 Multicast PJSIP 實現 IPv6 Multciast

因為 PJSIP 不支援 IPv6 Multicast，所以須修改其原始碼，新增 IPv6 Multicast Socket Option，以及 Struct pj_ipv6_mreq 並新增一個 Flag ipv6_media_mcast_use 作為是否啟用 IPv6 Multicast。

1. push-to-talk-with-pjsua -2.9/pjsip/include/pjsua-lib/pjsua.h

```
/**
* Specify whether IPv6 multicast should be used on media.
*/
 pjsua_ipv6_mcast_use            ipv6_media_mcast_use;
 /**
  * Specify how IPv6 ipv6 transport should be used in account config.
  */
 typedef enum pjsua_ipv6_mcast_use
 {
    /**
     * IPv6 is not used.
     */
    PJSUA_IPV6_MCAST_DISABLED,

    /**
     * IPv6 is enabled.
     */
```


doi:10.6837/ncnu202000205

```
        PJSUA_IPV6_MCAST_ENABLED


    } pjsua_ipv6_mcast_use;
```

2. push-to-talk-with-pjsua -2.9/pjlib/src/pj/sock_bsd.c

```
    /* ipv6 Multicast Socket option */
    #ifdef IPV6_MULTICAST_IF
    const pj_uint16_t PJ_IPV6_MULTICAST_IF = IPV6_MULTICAST_IF;
    const pj_uint16_t PJ_IPV6_MULTICAST_HOPS= IPV6_MULTICAST_HOPS;
    const pj_uint16_t PJ_IPV6_MULTICAST_LOOP = IPV6_MULTICAST_LOOP;
    const pj_uint16_t PJ_IPV6_JOIN_GROUP  = IPV6_JOIN_GROUP;
    const pj_uint16_t PJ_IPV6_LEAVE_GROUP = IPV6_LEAVE_GROUP;
    #else
    const pj_uint16_t PJ_IPV6_MULTICAST_IF    = 0xFFFF;
    const pj_uint16_t PJ_IPV6_MULTICAST_HOPS   = 0xFFFF;
    const pj_uint16_t PJ_IPV6_MULTICAST_LOOP  = 0xFFFF;
    const pj_uint16_t PJ_IPV6_JOIN_GROUP  = 0xFFFF;
    const pj_uint16_t PJ_IPV6_LEAVE_GROUP = 0xFFFF;
    #endif
```

3. push-to-talk-with-pjsua -2.9/pjlib/src/pj/sock_common.c

```
    PJ_DEF(pj_uint16_t) pj_IPV6_MULTICAST_IF(void)
    {
       return PJ_IPV6_MULTICAST_IF;
    }
```


doi:10.6837/ncnu202000205

```
PJ_DEF(pj_uint16_t) pj_IPV6_MULTICAST_HOPS(void)
{
   return PJ_IPV6_MULTICAST_HOPS;
}

PJ_DEF(pj_uint16_t) pj_IPV6_MULTICAST_LOOP(void)
{
   return PJ_IPV6_MULTICAST_LOOP;
}


PJ_DEF(pj_uint16_t) pj_IPV6_JOIN_GROUP(void)
{
   return PJ_IPV6_JOIN_GROUP;
}

PJ_DEF(pj_uint16_t) pj_IPV6_LEAVE_GROUP(void)
{
   return PJ_IPV6_LEAVE_GROUP;
}
```

4. push-to-talk-with-pjsua -2.9/pjlib/include/pj/sock.h

```
/** IPV6 multicast interface. @see pj_IPV6_MULTICAST_IF() */
extern const pj_uint16_t PJ_IPV6_MULTICAST_IF;
```


doi:10.6837/ncnu202000205

```
    /** IPv6 multicast hop. @see pj_IPV6_MULTICAST_HOPS() */
    extern const pj_uint16_t PJ_IPV6_MULTICAST_HOPS;
    /** IPv6 multicast loopback. @see pj_IPV6_MULTICAST_LOOP() */
    extern const pj_uint16_t PJ_IPV6_MULTICAST_LOOP;
    /** Join an IPv6 group. @see pj_IPV6_JOIN_GROUP() */
    extern const pj_uint16_t PJ_IPV6_JOIN_GROUP;
    /** Leave an IPv6 group. @see pj_IPV6_LEAVE_GROUP() */
    extern const pj_uint16_t PJ_IPV6_LEAVE_GROUP;

/** Get #PJ_IPV6_MULTICAST_IF constant */
PJ_DECL(pj_uint16_t) pj_IPV6_MULTICAST_IF(void);

  /** Get #PJ_IPV6_MULTICAST_HOPS constant */
  PJ_DECL(pj_uint16_t) pj_IPV6_MULTICAST_HOPS(void);
  /** Get #PJ_IPV6_MULTICAST_LOOP constant */
  PJ_DECL(pj_uint16_t) pj_IPV6_MULTICAST_LOOP(void);
  /** Get #PJ_IPV6_JOIN_GROUP constant */
  PJ_DECL(pj_uint16_t) pj_IPV6_JOIN_GROUP(void);
  /** Get #PJ_IPV6_LEAVE_GROUP constant */
  PJ_DECL(pj_uint16_t) pj_IPV6_LEAVE_GROUP(void);

      /** Get #PJ_IPV6_MULTICAST_IF constant */
    #   define pj_IPV6_MULTICAST_IF()    PJ_IPV6_MULTICAST_IF
       /** Get #PJ_IP_MULTICAST_TTL constant */
    #   define pj_IPV6_MULTICAST_HOPS()
```


doi:10.6837/ncnu202000205

```
PJ_IPV6_MULTICAST_HOPS
        /** Get #PJ_IPV6_MULTICAST_LOOP constant */
    #   define pj_IPV6_MULTICAST_LOOP()  PJ_IPV6_MULTICAST_LOOP
        /** Get #PJ_IPV6_ADD_MEMBERSHIP constant */
    #   define pj_IPV6_JOIN_GROUP()  PJ_IPV6_JOIN_GROUP
        /** Get #PJ_IPV6_DROP_MEMBERSHIP constant */
    #   define pj_IPV6_LEAVE_GROUP() PJ_IPV6_LEAVE_GROUP

    /**
     * This structure provides multicast group information for IPv6 addresses.
     */
    typedef struct pj_ipv6_mreq {
       pj_in6_addr ipv6mr_multiaddr;   /**< IPv6 multicast address of group. */
       int ipv6mr_interface;   /**< local IPv6 address of interface. */
    } pj_ipv6_mreq;
```

5. /pjsip/src/pjsua-lib/pjsua-media.c

I. create_rtp_rtcp_sock()

```
    /* if ipv6 mcast REUSEADDR*/
  if (acc->cfg.ipv6_media_mcast_use == PJSUA_IPV6_MCAST_ENABLED){
        int reuse = 1;
        status = pj_sock_setsockopt(sock[0], pj_SOL_SOCKET(),
              pj_SO_REUSEADDR(), &reuse, sizeof(reuse));
     }

    /* Specify the multicast group */
```


doi:10.6837/ncnu202000205

```
        if (acc->cfg.ipv6_media_mcast_use ==
PJSUA_IPV6_MCAST_ENABLED){
            struct pj_ipv6_mreq imr;
            imr.ipv6mr_multiaddr = public_addr.ipv6.sin6_addr;
            imr.ipv6mr_interface = 0;
            status = pj_sock_setsockopt(sock[0],PJ_SOL_IPV6,
pj_SOL_IPV6(),pj_IPV6_JOIN_GROUP(),&imr, sizeof(struct pj_ipv6_mreq));
            if (status != PJ_SUCCESS)
              return status;
        }

    /* if ipv6  mcast REUSEADDR*/
     if (acc->cfg.ipv6_media_mcast_use ==PJSUA_IPV6_MCAST_ENABLED){
            int reuse = 1;
            status = pj_sock_setsockopt(sock[1], pj_SOL_SOCKET(),
                  pj_SO_REUSEADDR(), &reuse,sizeof(reuse));
      }

    /* Specify the multicast group */
     if (acc->cfg.ipv6_media_mcast_use == PJSUA_IPV6_MCAST_ENABLED){
            struct pj_ipv6_mreq imr;
            imr.ipv6mr_multiaddr = public_addr.ipv6.sin6_addr;
            imr.ipv6mr_interface = 0;
            status = pj_sock_setsockopt(sock[1],PJ_SOL_IPV6,//
pj_SOL_IPV6(),pj_IPV6_JOIN_GROUP(),&imr, sizeof(struct pj_ipv6_mreq));
```


doi:10.6837/ncnu202000205

```
        if (status != PJ_SUCCESS)
            return status;
    }
```

附錄四　IPv6 Multicast PJSIP 實現 Multicast PTT

PJSUA 的 pjsua_media.c 中，收到 Incoming 封包並且確認沒問題後，會開始建立 Media Session，預設是把 Local Device 的 IPv6 Address 作為 Local RTP Address，但是在 MPTT 當中是把對方 SDP 中的 Mcast Address 作為 Local RTP address，為了達到此功能而新增了 ptt_mcast_use，作為判斷依據。

因為 PJSUA 的 pjmedia 預設會把 Remote RTP Address 設為對方的 IP Address(source address)，但是本論文中的 MPTT 運作方式是 Multicast Address，所以需要把 PJMEDIA_TRANSPORT_SWITCH_REMOTE_ADDR 設為 0。

1. pjsip/include/pjsua-lib/pjsua.h

```
/**
    * Mcast Push to talk set rtp address with remote sdp conn addr
    */
   pjsua_ptt_mcast_use          ptt_mcast_use;

/**
 * Mcast Push to talk set rtp address with remote sdp conn addr
 */
typedef enum pjsua_ptt_mcast_use
{
```


doi:10.6837/ncnu202000205

```
    /**
     * PTT MCAST is not used.
     */
    PJSUA_PTT_MCAST_DISABLED,

    /**
     * PTT MCAST is enabled.
     */
    PJSUA_PTT_MCAST_ENABLED

} pjsua_ptt_mcast_use
```

2. /pjsip/src/pjsua-lib/pjsua-media.c

I. pjsua_media_channel_init()

```
/* ptt multicast enable*/
    if (acc->cfg.ptt_mcast_use ==
PJSUA_PTT_MCAST_ENABLED){
        pjmedia_sdp_media *m = rem_sdp->media[rem_sdp-
>media_count-1];
        acc->cfg.rtp_cfg.public_addr= m->conn->addr;
    }
```

II. create_rtp_rtcp_sock()

```
pj_sockaddr public_addr;
pj_sockaddr_init(af, &public_addr, NULL, 0);
   if (cfg->public_addr.slen) {
```


doi:10.6837/ncnu202000205

```
                status = pj_sockaddr_set_str_addr(af, &public_addr, &cfg-
    >public_addr);
                if (status != PJ_SUCCESS) {
                    pjsua_perror(THIS_FILE, "Unable to resolve transport public
    address",
                                 status);
                    return status;
                }
            }
```

3. /include/pjmedia/config.h

```
PJMEDIA_TRANSPORT_SWITCH_REMOTE_ADDR 0
```

4. /include/pjmedia/sock.h

```
extern const struct pj_in6_addr pj_in6addr_any;
extern const struct pj_in6_addr pj_in6addr_loopback;
extern const struct pj_in6_addr pj_in6addr_any;
extern const struct pj_in6_addr pj_in6addr_loopback;
```

```
/*
 * SIP multicast socket option.
 */
PJ_DEF(void) pjsua_sip_multicast(pjsip_rx_data *rdata)
{
   /* Parse SDP from incoming request */
```


doi:10.6837/ncnu202000205

```
if (rdata->msg_info.msg->body) {
    pj_status_t status;
    pjmedia_sdp_session *offer=NULL;
    pjsip_rdata_sdp_info *sdp_info;
    sdp_info = pjsip_rdata_get_sdp_info(rdata);
    offer = sdp_info->sdp;
    pjmedia_sdp_media *m = offer->media[offer->media_count-1];
    struct pj_ipv6_mreq imr;
    pj_sockaddr public_addr;

    pj_sockaddr_init(pj_AF_INET6(), &public_addr, NULL, 0);
    if (m->conn->addr.slen) {
      status = pj_sockaddr_set_str_addr(pj_AF_INET6(), &public_addr, &m-
>conn->addr);
      if (status != PJ_SUCCESS) {
    pjsua_perror(THIS_FILE, "Unable to resolve transport bind
address",status);
        return status;
      }
    }
    imr.ipv6mr_multiaddr = public_addr.ipv6.sin6_addr;
    imr.ipv6mr_interface = 0;
    int reuse = 1;
    status = pj_sock_setsockopt(*sip_sock, pj_SOL_SOCKET(),
     pj_SO_REUSEADDR(), &reuse, sizeof(reuse));
```


doi:10.6837/ncnu202000205

```
status = pj_sock_setsockopt(*sip_sock,PJ_SOL_IPV6,//
pj_SOL_IPV6(),pj_IPV6_JOIN_GROUP(),&imr, sizeof(struct pj_ipv6_mreq));
```


doi:10.6837/ncnu202000205