國立暨南國際大學資訊工程學系

碩士論文

支援網路電話即時監聽機制之分散式架構

Distributed Architecture for Real-Time Lawful Intercept in SIP-based VoIP Systems

指導教授：吳坤熹博士

研究生：陳柏州

中華民國九十七年 六月

致謝

本研究論文得以完成，本人首先要感謝我的恩師 吳坤熹教授的指導。在吳老師兩年的栽培之下，讓學生我在這段追求學問的路上得以學習到實事求是的研究方式。雖然在學習過程中常有過錯，老師始終以寬容的心予以指導，指引學生朝正確的方向邁進。

此外，也要感謝在論文撰寫的過程中所有幫助過我的人士：在論文上或程式上有疑慮時，提供意見或技術諮詢的陳泰良學長及王鐘逸同學。和我一同在實驗室共同為各自的論文努力熬夜、彼此加油打氣的王嘉裕學弟。還有，感謝陳彥州同學/學弟，在研究之餘照料我、陪同我到處散心。以及感謝 IP 網路電話實驗室的部份同學及學弟妹。

最後，感謝我的家人們，由於你們的支持，讓我可以在這求學之路能無後顧之憂，沒有你們一路上的支持與鼓勵，我無法有今天的成果，一切成就終歸你們。

論文名稱：

支援網路電話即時監聽機制之分散式架構

校院系：國立暨南國際大學資訊工程所　　頁數：49

畢業時間：97 年 6 月　　學位別：碩士

研究生：陳柏州　　指導教授：吳坤熹博士

中文摘要

網際網路語音（Voice over IP，簡稱VoIP）服務挾帶低通話成本的優勢，不斷地吸引傳統電話的使用者，包括個人及企業用戶，轉而使用VoIP這項新興的技術。當VoIP步入商業化階段並在市場上開始提供通話服務時，相關的電信法規要求提供電話監聽的機制。目前在傳統公眾電話網路上已有一套運作良好的監聽系統，而VoIP由於聲音的傳送並不像傳統電話需要透過中央交換機，相較之下較不易進行監聽。

近年來所提出的VoIP監聽系統，大都需要修改SIP代理伺服器架構，讓SIP代理伺服器將監聽訊息導向相關的監聽設備以進行監聽。但SIP為一開放的通訊協定，因此任何人都可以自行開發或使用各種廠牌的SIP伺服器；且由於將封包更改傳送路由的方式，容易讓被監聽者察覺。再者，將監聽的內容即時以檔案的型式儲存，會造成其磁碟讀寫的效能問題，在同時需要進行多人監聽時，會形成效能上的瓶頸。

針對上述問題，本論文提出一套 VoIP 的分散式監聽功能架構。我們將監聽功能佈建在網路基礎設備上，透過在網路設備上交換的封包做分析，建立監聽系統。由於監聽設備分散在各個子網路中，透過自動化回報及訊息交流，達到分散式監聽的目標，並且在執行監聽時由各個監聽設備即時將監聽內容傳至監聽者。因此本論文所提出的架構具有易於部署在不同的環境中，不需修改現有的 SIP 代理伺服器系統，能夠於通話進行中即時監聽內容等特點。

關鍵詞：SIP、VoIP、分散式系統、合法監聽、即時監聽

Title of Thesis：

Distributed Architecture for Real-Time Lawful Intercept in SIP-based VoIP Systems

Name of Institute：Department of Computer Science and Information Engineering, National Chi Nan University Pages：49

Graduation Time：06/2008 Degree Conferred：Master

Student Name：Po-Chou Chen Advisor Name：Quincy Wu

English Abstract

Due to the low cost and the convenience of Internet, the Voice over Internet Protocol（VoIP）services are also attracting the subscribers of the traditional telephone to turn to use this new technology. For VoIP to be commercialized, it must support Lawful Interception which is required by the Law Enforcement Agency of each country. In contrast to the Public Switched Telephone Network（PSTN）which has a good monitoring system, it is more difficult to develop a monitoring system for VoIP because the sound transmission in VoIP need not go through the central office (CO).

Recently, a few VoIP monitoring systems are proposed, but most of them need to revise the SIP proxy server behavior. They rely on the SIP proxy server to modify the SIP message body to conduct the monitoring. However, SIP is an open communication protocol; anyone may develop his/her own SIP server, so the server-based solution of monitoring will not always work. Moreover, because redirecting the packet flow to the monitoring device will change the route, it may be easy for people to detect that their conversation is being monitored. Users may also apply S/MIME to encrypt the contents of the SIP message body to prevent them from being modified by the SIP proxy server. Furthermore, the system stores the audio conversation in files, so the disk I/O will cause significant performance issues. As the requests for more monitoring increase, this will become the performance bottleneck.

To conquer the above pitfalls, this thesis proposes a distributed architecture of VoIP monitoring. We will distribute the monitoring equipment in many subnets, and wiretapping can be done by immediately passing the intercepted audio packets to the Law Enforcement Agency. In a word, this thesis proposes an architecture which is easy to deploy in different environments without amending the existing SIP proxy server, and it can support the real-time monitoring of audio conversation easily.

Keyword: Lawful Interception, Real-Time Interception, SIP, VoIP

目錄

圖目錄

表目錄

1. 導論

1.1 現況

網際網路語音服務（Voice over Internet Protocol，簡稱 VoIP）在這幾年快速成長，目前除了 MSN 之外，亦有蕃薯藤、新浪等入口網站推出網際網路語音服務。以最近迅速竄紅的 Skype 軟體為例，目前估計下載次數早已超過一億人次[1]，而這個數字還只是針對 Skype 這套網際網路語音服務軟體。從整個應用服務的發展來看，網際網路語音服務很明顯已逐漸成為大眾普遍使用的應用項目。

1.2 動機

VoIP 以其低成本獲得眾多使用者的喜愛，衝擊著原有的傳統電信市場。但由於 VoIP 普及後可能會成為犯罪的追查死角，因此 VoIP 在市場上提供服務時，電信法規會要求提供監聽機制。目前在傳統公眾電話網路（Public Switched Telephone Network，簡稱 PSTN）已經有一套行之多年的監聽架構；相對之下，VoIP 平台上僅有若干簡略的監聽機制，而且現有的 VoIP 監聽機制普遍具有會造成監聽設備負荷過重的缺點。本論文希望針對目前 VoIP 服務中最常使用的標準協定 SIP（Session Initiation Protocol）[2]，提出在 SIP 網路協定為基礎下的監聽機制，並經由分散式的架構來減少負荷。

1.3 網路電話

網路電話（Voice over Internet Protocol，簡稱 VoIP），簡單來說就是將語音訊號壓縮成數據資料封包後，在 Internet 網路上傳送的雙向語音服務；也就是說，透過開放性的網際網路，傳送語音的電信應用服務。利用網路電話不僅做到了即時提供語音服務，更可透過 Internet 連接至世界各地，讓使用者可以不需再透過傳統電話交換網路(PSTN)進行遠距離電話交談。

網路電話近年來快速成長，提供的服務也越來越多元化。例如 2005 年網路拍賣網站 eBay 以總值約 41 億美元收購網路電話廠商 Skype，想藉由語音通訊功能結合拍賣服務提供給自家用戶。而 2007 年台灣開始發放 070 網路電話執照，070 網路電話號碼為 E.164 通信編碼格式的 11 碼數字，能與傳統 PSTN 與行動電話互通。與 Skype 相比，070 網路電話每個用戶都有單獨門號；Skype 在撥打給傳統 PSTN 與行動電話時，僅能撥出去（撥出的號碼為 Skype 代表號），無法直接回撥給使用者[3]。為了讓網路電話更加普及，提供較多的服務與良好的通話品質固然是網路電話發展的重點，但這項新技術在管理上同時也產生了新的挑戰。因為傳統的 PSTN 系統中，通話都會經由電信業者的交換機來傳遞，因此執法機關要進行監聽時，只需要在電信機房就可以進行；但網路電話通話時是以點對點直接傳送語音資料，聲音並不會透過中央主機，因此傳統的監聽機制並不適用。相關的技術細節，我們將在 2.5 節予以說明。

在台灣，因應網路電話服務發展趨勢，電信法規詳細規範了需提供監聽機制：

- 第二類電信事業細部通訊監察需求表第一條[4]：需能將依通訊監察法特定之監察對象於通訊監察書許可期間內通話內容、時間及傳送路徑紀錄儲錄或轉送至通訊監察機關。
- 第二類電信事業管理規則第 27 條[5]：經營者對於調查或蒐集證據，並依法律程序查詢電信之有無及其內容者，應提供之。

2. 背景知識及相關研究

2.1 Session Initial Protocol

Session Initial Protocol，簡稱 SIP，是由網際網路工程任務小組（Internet Engineer Task Force ; IETF）制定的媒體會議訊息通訊協定。SIP 是以 ASCII 文字資料為基礎，用來建立、更改與結束兩點或多點之間通訊應用程式層次之控制協定。如同其他的 VoIP 通訊協定，SIP 的設計目的也是用來定義封包式資料電話網路中通訊階段管理工作。訊號取樣讓通訊內容能轉換成為數位化資訊，得以傳遞到網路各個地區；通訊階段管理則提供通話從開始到結束的所有狀態控制功能。

SIP 採用分散式架構，透過 URI（Uniform Resource Identifier）來命名位址與使用純文字格式來傳送訊息，使 SIP 能夠藉由網際網路的優點，來架構 VoIP 網路與應用程式。圖 1 為 SIP 網路的分散架構。

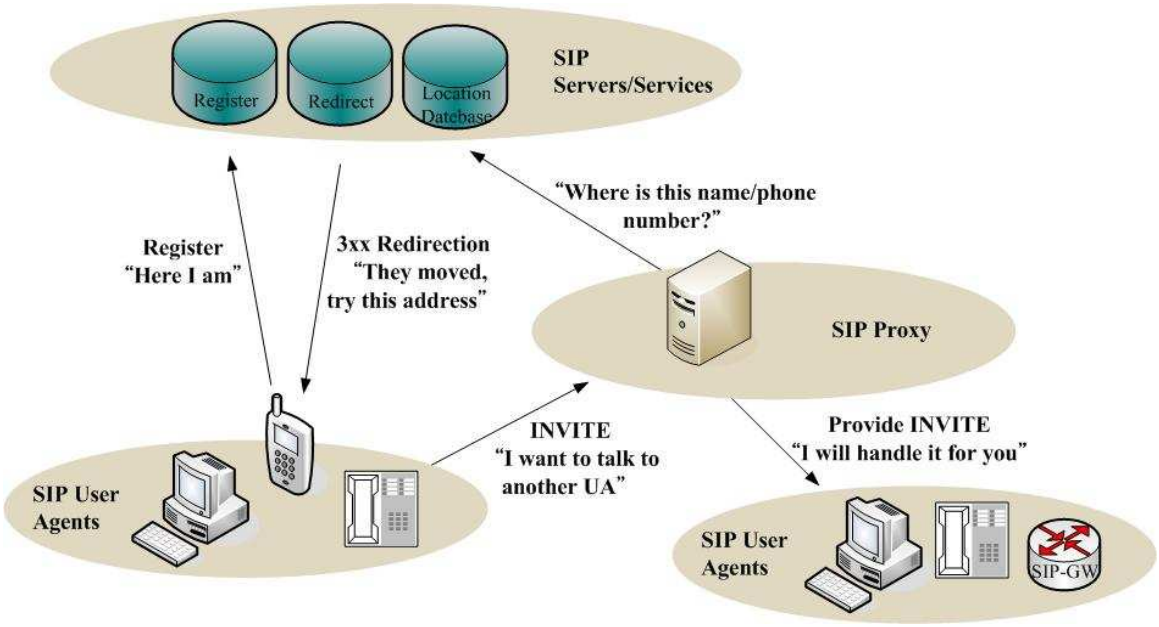

圖 1: SIP 網路的分散架構

在 SIP 網路中，我們會使用一個 SIP Proxy Server 來幫忙傳送 SIP 訊息。當發話端送出通話要求時，會經由 SIP Proxy Server 傳送給受話端，而受話端回傳的回應也會經由 SIP Proxy Server 回傳給發話端。圖 2 為 SIP 在建立通話及掛斷的訊息[6]。

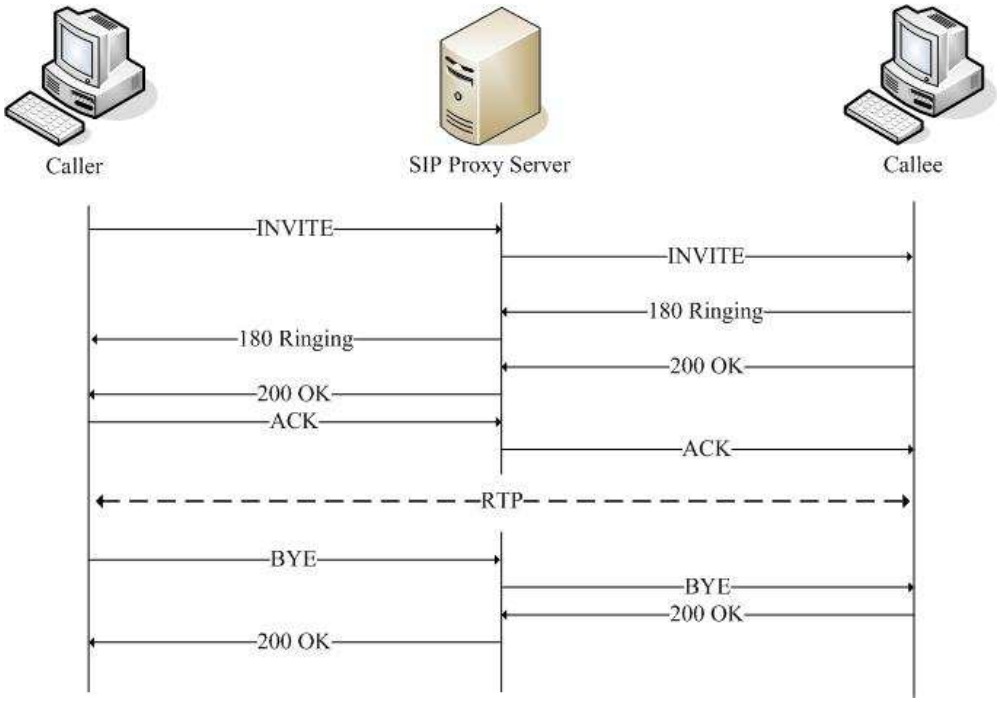

圖 2: SIP 建立通話及掛斷訊息

在 IETF 中所定義的 RFC 文件中有很多針對 SIP 在建立通話動作所定義的方法，於本論文中所使用的方法如表 1[2][7][8]。

表 1: SIP 所使用的方法

方法	說明	文件
INVITE	邀請建立 Session	RFC3261
ACK	回覆確認邀請(INVITE)	RFC3261
BYE	結束一個已連接的 Session	RFC3261
CANCEL	取消一個已發出邀請但尚未建立 Session 的請求	RFC3261
REGISTER	註冊使用者資訊	RFC3261
SUBSCRIBE	進行訂閱更新通知	RFC3265
NOTIFY	Server 用於回傳 UA 更新訊息	RFC3265

PUBLISH	傳送更新通知給 Server	RFC3903

2.1.1 SUBSCRIBE、NOTIFY 及 PUBLISH

這三種方法常被結合在一起使用在更新訊息的動作上；假設 UA1 要求 UA2 有訊息更新時，必須送出通知給 UA1。首先 UA1 會送出 SUBSCRIBE 給 Presence Server，表示要訂閱有關 UA2 的訊息；Server 會回傳 200 OK 及 NOTIFY 確認訊息後完成訂閱動作。此後當 UA2 傳送更新訊息時，會發出 PUBLISH 給 Presence Server，Server 會對所有訂閱 UA2 的人發送 NOTIFY 訊息。圖 3 說明了此流程。

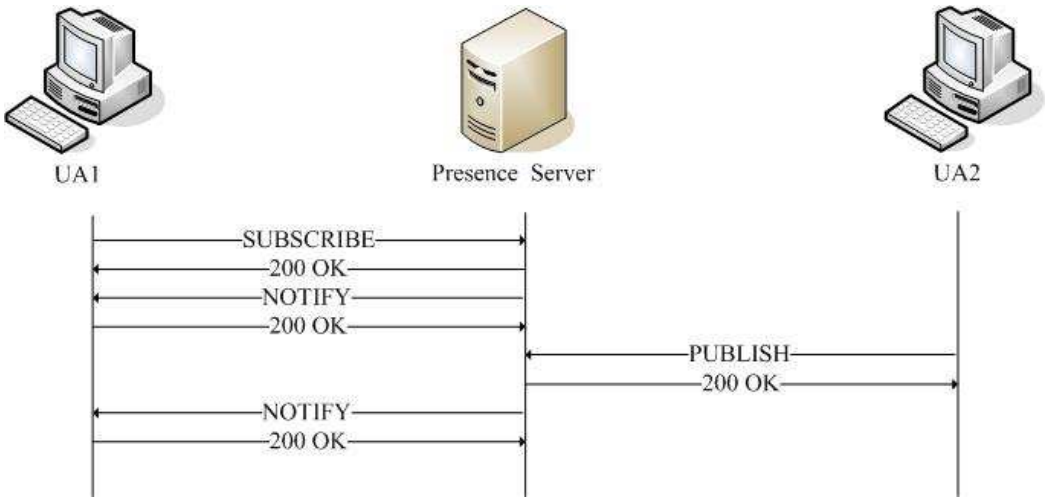

圖 3: SUBSCRIBE、NOTIFY 及 PUBLISH 流程

2.2 Session Description Protocol

Session Description Protocol[9]，簡稱 SDP，是在 RFC4566 中被提出，跟 SIP 一樣是以 ASCII 文字資料為基礎。SDP 只是一套描述多媒體訊息的格式，本身並沒不指定通話的建立或結束等動作，因此在 VoIP 裡，SDP 必須跟其他協定（如 SIP）合併使用。

一般來說 SDP 中主要有兩種類別的參數，分別為 Session Level 和 Media Level。

Session Level 包含 Session 資訊如：版本訊息、Session 名稱、Session 來源。Media Level 則包含了需要跟對方所溝通的媒體格式、連接埠號；通常不單只有一種媒體格式，所以會有多個媒體格式敘述。圖 4 為 SDP 結構示意圖。

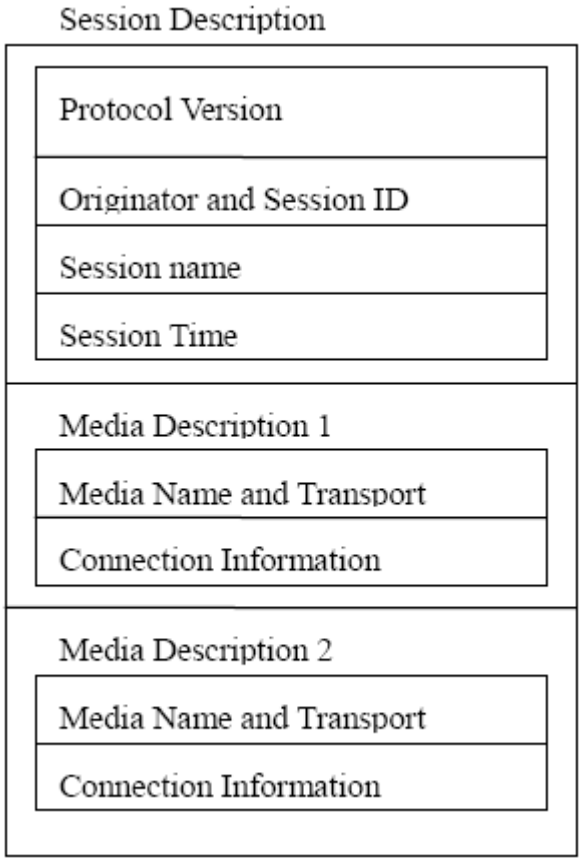

圖 4: SDP 結構圖

2.3 Real-Time Transport Protocol

Real-Time Transport Protocol[10]，簡稱 RTP，於 RFC3550 中被提出。RTP 是一種提供點對點傳輸服務的即時傳輸協定，用來支援在單目標廣播和多目標廣播網路服務中傳輸即時資料。

如圖 5 所示，RTP 封包的表頭包含了 RTP Version、Payload Type、Sequence Number、Timestamp、Synchronization Source Identifier（SSRC）等等，其中 CSRC 是用於混音器來將不同來源的 RTP 封包混合在同一個 RTP 封包中，因此在一般的點對

點（end to end）傳輸中，並不會使用 CSRC 這個欄位，所以 RTP 封包總共為 12 bytes。

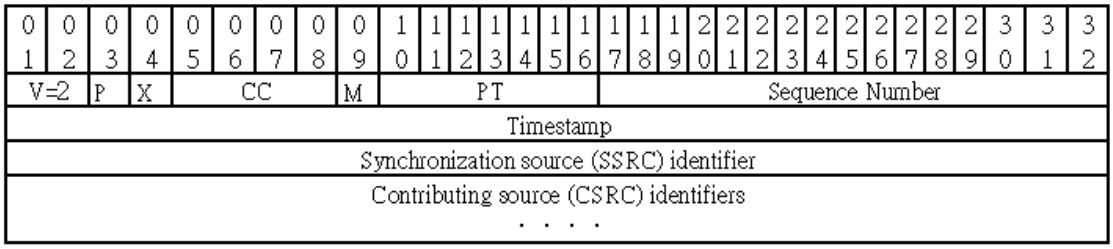

圖 5: RTP 封包格式

RTP 封包中 Payload Type（PT）欄位定義了封包中的資料為何種媒體格式，此欄位長度為 7 bits，因此可支援 128 種不同的媒體格式，表 2 為常見的 Payload Type 列表[11]。

表 2:常見的 Payload Type

Payload Type	聲音類型	採樣率(kHz)	數據率(kb/s)
0	PCMU	8	64
3	GSM	8	32
8	PCMA	8	64
15	G.728	8	16
18	G.729	8	8

2.4 監聽方法

2.4.1 PSTN 監聽方法

傳統電路交換網路(Public Switched Telephone Network，簡稱 PSTN)，是以電路交換(Circuit-Switched)方式進行傳輸，每次通話建立時系統會為通話雙方保留一條傳輸的頻寬，直到雙方結束通話為止。

依照通訊保障及監察法第 11 條[12]，只要有足夠的事實可以證明欲監聽的對象有相關罪嫌，就可以申請通訊監察書。通訊監察書應記載下列事項：

1. 案由及涉嫌觸犯之法條。

2. 監察對象。

3. 監察通訊種類及號碼等足資識別之特徵。

4. 監察處所。

5. 監察理由。

6. 監察期間及方法。

7. 聲請機關。

8. 執行機關。

偵查中由檢察官依司法警察機關聲請或依職權核發，審判中由法官依職權核發，但是該管檢察官可以口頭通知執行機關先予執行通訊監察。第五條之通訊監察期間，每次不得逾三十日；第七條之通訊監察期間，每次不得逾一年。圖 6 為公眾交換電話網路的監聽架構。

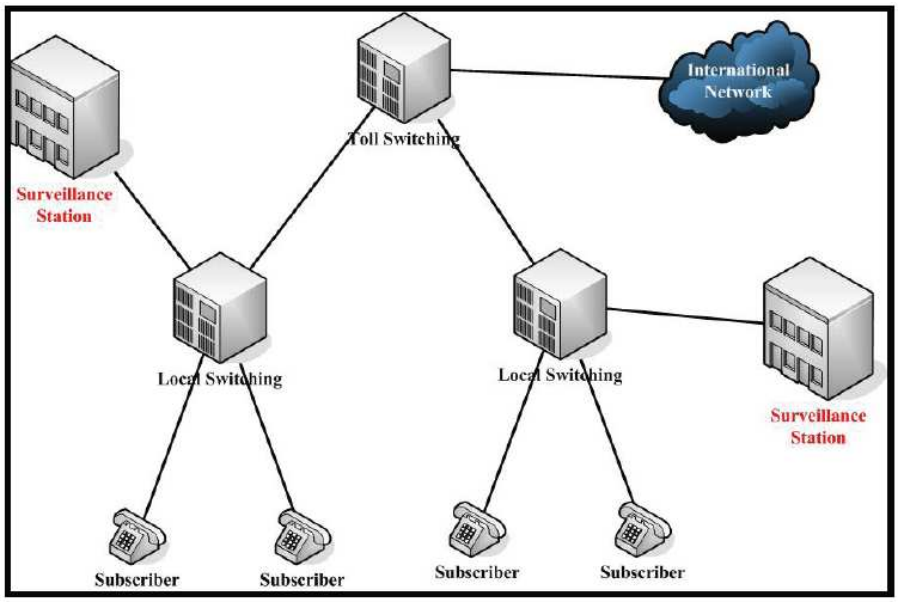

圖 6: PSTN 監聽架構

2.4.2 VoIP 監聽方法

VoIP 簡單來說，即是將聲音數位化之後，以一個個封包的方式，透過 IP 網路傳

送到另一端，因此 VoIP 與 PSTN 雖然希望達到的目的相同，但在架構上有明顯不同。監聽 VoIP 除了需要攔截 SIP 信令外，亦須攔截相關的媒體資料流(Media stream)。SIP 信令訊息可使用不同的傳輸協定(例如 UDP 或 TCP)，而埠號則由終端設備自行決定。Media stream 一般使用 UDP 搭配 RTP。

圖 7 為現行以 SIP 為基礎的 VoIP 系統監聽架構之一[13]。以圖 7 的架構，當 SIP 建立通話時，中間的 SIP Proxy Server 會將兩邊 UA 所發送的 INVITE 訊息做修改，將其 SDP 所描述的對方 UA 的 IP 位置改成 RTP Proxy 的 IP 位置，因此當通話建立成功之後，雙方的通話將通過 RTP Proxy，即可直接將通過 RTP Proxy 的通話做分析處理。

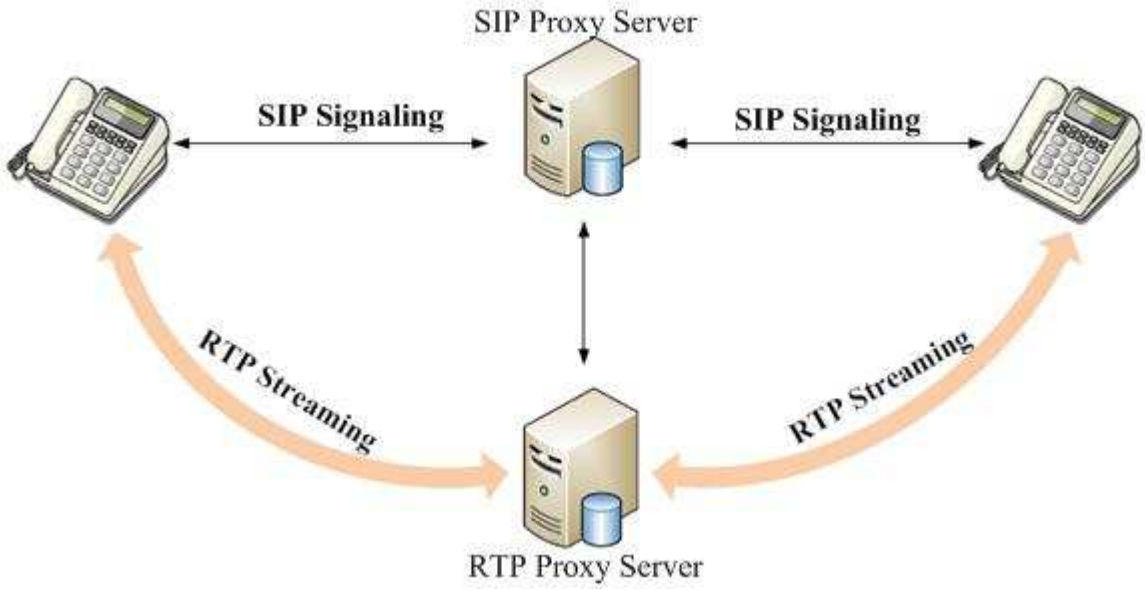

圖 7:VoIP 的監聽架構(RTP 透過中繼站)

而這個作法的缺點在於需要架設一個 RTP Proxy 來做聲音的中繼站，也就是將每次通話的 RTP stream 皆導向 RTP Proxy 中再經由檔案回傳至監聽者的電腦上收聽。此一方式會造成 RTP Proxy 負擔太大，而且在監聽過程中，若使用者使用適當的工具檢測，會發現聲音的傳送端並非對話者本人，而是一台正在監聽的 RTP Proxy 設備。這種會讓被監聽者察覺的機制，其實並不是個好的監聽機制。

圖 8 為另一種監聽架構[14]，由圖上可以很明顯看到此系統架構需在網路節點上佈建監聽設備。在通話時 SIP 訊息會存至資料庫做為紀錄，而 RTP 封包則會即時寫

入檔案伺服器。當監查人員要監聽其中的對話時，經由瀏覽器從檔案伺服器直接下載音效檔（wav）。

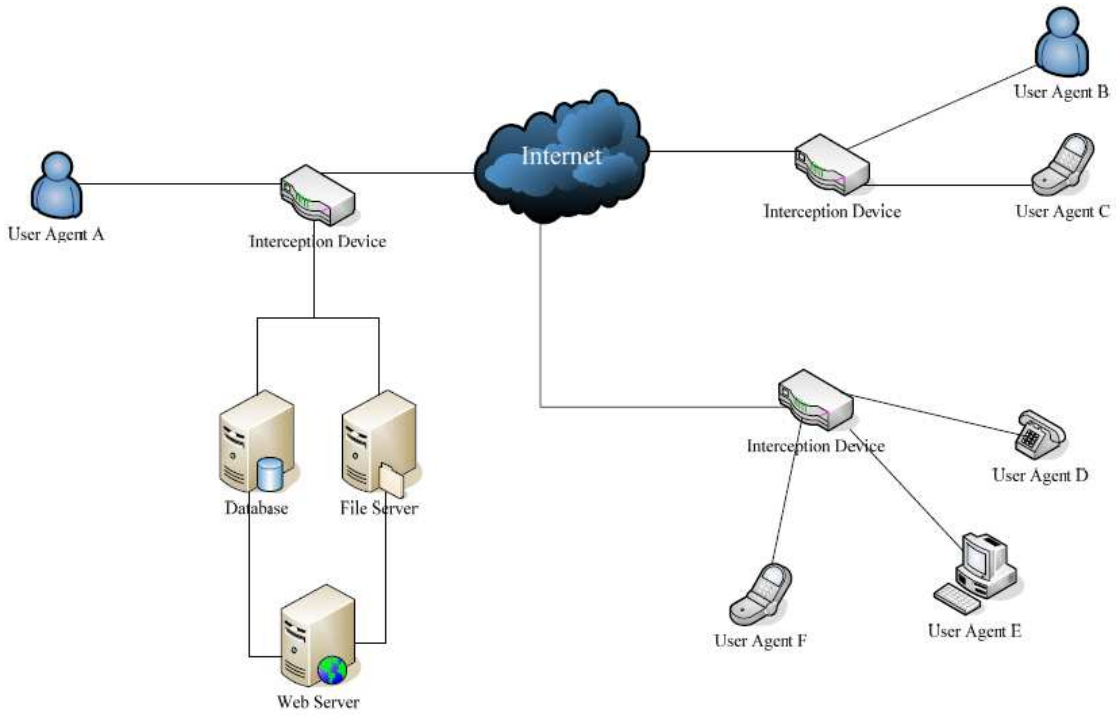

圖 8:VoIP 的監聽架構(網路節點佈建監聽設備)

這個做法不會讓監聽者發現自己正受到監聽，在原本的網路傳輸模式上也不會有任何改變。但是在儲存大量監聽內容時，大量的 I/O 會是影響其中效能的主要原因；再者事後才從檔案伺服器取得內容，在打擊犯罪的時效上並不夠即時快速；而當監聽檔案過大時，必須下載完成才能開始聽取內容，等待時間過久。因此我們提出一個新的監聽架構改善以上的缺點。

3. 核心技術

3.1 Libpcap

Pcap[15]是一個擷取封包的 Application Programming Interface（簡稱 API），在 Unix-like 的作業系統上的版本稱為 libpcap；而在 Windows 系統上的版本是 WinPcap，是廣為使用的函式庫。

程式設計師可以利用 C 或 C++語言撰寫擷取封包的程式，Pcap API 提供在不同網路介面上運作，以及透過正規過濾語法來設定過濾器 BPF（BSD Packer Filter）的功能。正規過濾語法的表示十分直覺。例如我們想要獲得 HTTP 的封包，可以設定 TCP Port 80，如此一來便只會擷取利用 TCP 經由 Port 80 傳送的封包。許多網路服務工具，大都使用 libpcap 或是 WinPcap 當作擷取及過濾封包的核心。

在我們的系統中，主要使用到的 libpcap 函式[16]如下：

- pcap_lookupdev()：傳回一張網卡以供 pcap_lookupnet()、pcap_open_live() 使用
- pcap_lookupnet()：設定網域
- pcap_open_live()：開啟網路介面卡以準備進行封包擷取
- pcap_compile()：編譯正規過濾語法
- pcap_setfilter()：設定過濾器
- pcap_loop()：持續地擷取封包

只要使用了這些基本的函式，就可以發展出一個具備有擷取封包功能的程式了。但是，在封包擷取下來後，使用者仍必需另外撰寫拆解封包、解讀封包的程式碼，才能將封包擷取的結果以容易閱讀的方式呈現給使用者。

3.2 oSIP2、eXosip2

oSIP2[17]與 eXosip2[18]是遵循 GPL 協議的開放程式庫，oSIP2 不提供任何快速產生請求訊息和回應訊息的方法，所有請求消息和附應消息的形成必須調用一連串的 SIP message API 來完成，因此作者在 oSIP2 上基礎上開發了 eXosip2，用 eXosip2 開發 VoIP 電話非常方便，僅需幾個函式就可以完成。

以 eXosip2 開發的 VoIP 電話相當的多，如 Linphone、SfSipUA 等等。而使用 eXosip2 來建立一個簡單的 VoIP UA 也相當簡單，主要要使用的函式[19][20]如下：

- eXosip_init ()：初始化 eXosip
- eXosip_listen_addr()：使用的協定及聆聽埠
- eXosip_event_wait()：偵測 SIP 訊息接收
- eXosip_call_build_initial_invite()：建立 INVITE 訊息
- eXosip_call_send_initial_invite()：傳送 INVITE 訊息
- eXosip_call_build_answer()：建立一個回應訊息
- eXosip_call_send_answer()：傳送所建立的回應訊息

只要使用了這些基本的函式，就可以使用接收及回應 SIP 訊息了。但是，在收到 SIP 訊息後，使用者仍必需另外撰寫針對 SIP 參數的程式碼，才能將完整的建立一個 SIP UA。

3.3 oRTP

oRTP[21]是遵循 GPL 協議的開放程式庫，依據 RFC3550，使用 C 語言編寫，提供了 RTP 相關的 API 函數，使用者可以在 Unix-like 或 Windows 利用 oRTP API 來建立 RTP 協定連線。實現一個簡單的 oRTP 程式所要使用的函式[22]如下：

- ortp_init ()：初始化 oRTP

- ortp_exit()：結束先前使用的 oRTP
- rtp_session_new()：建立一個 RTP Session
- rtp_session_set_ssrc()：設定 SSRC 值
- rtp_session_set_payload_type()：設定所使用的 Payload Type
- rtp_session_set_remote_addr()：設定目的地的 IP address 及 Port number
- rtp_session_recv_with_ts ()：根據 Timestamp 從 RTP stream 讀取數據
- rtp_session_send_with_ts()：從 buffer 裝入 RTP 封包並設定 Timestamp

3.4 libxml2

libxml2[23]是一個 C 語言的 XML 解析工具，主要是發展在 Gnome project 上。XML 是一個可延伸標示語言；XML 文件具有能夠很容易地讓人閱讀，同時又很容易讓電腦程式去辨識的語言格式和語法，但是在程式中要無錯誤的去取得所需的資訊，會花費不少功夫。而 libxml2 可以將這個時間縮短，並增加準確度。使用 libxml2 函式去解析 xml 主要使用的函式[24]：

- xmlReadMemory()：從 buffer 讀進 xmlDocPtr
- xmlDocGetRootElement()：從 xmlDocPtr 取得 xml 的 root 節點
- xmlNodeListGetString()：從 root 節點開始取得每個節點的資訊
- xmlFreeDoc()：釋放 xmlDocPtr
- xmlCleanupParser()：將解析結果 buffer 清除
- xmlMemoryDump()：將讀進 buffer 清除

3.5 Speex

Speex[25]是一個音訊壓縮的開放函式庫，Speex 主要是針對平常談話的環境的失真壓縮編碼器，所以他是一個很適合用於網路音訊通訊的一種壓縮格式。如要將一個

PCM 壓縮成 Speex 格式主要使用的函式[26]如下：

- speex_bits_init()：初始 structure bits
- speex_encoder_init()：初始 encoder
- speex_encoder_ctl()：設定 quality 數值
- speex_encode_int()：開始對此 frame encode
- speex_bits_write ()：將 encode 結果寫入 buffer
- speex_encoder_destroy()：釋放 encode 的 buffer
- speex_bits_destroy()：釋放 bits buffer

3.6 Third Party Call Control

Third Party Call Control（簡稱 3PCC）[27]，在 RFC3725 中被提出。顧名思義 3PCC 就是透過第三方的控制者來建立與另外兩方之間的連線，以及協助 SIP 達成許多的服務。例如使用者在瀏覽網頁時，點擊網頁上號碼的連結，伺服器即可替兩人建立通話連線。透過第三方在雙方通話的使用者之間以 SDP 訊息協調即將建立的通話，包括雙方使用的媒體格式，聲音傳輸的目的埠等等。

圖 9 是一個簡單的 3PCC 流程：

（1） 由第三方 Controller 發出一個不含 SDP 的 INVITE 訊息給 UA1。

（2） UA1 在收到 Controller 的 INVITE 訊息後回應 200 OK 並帶上 UA1 的 SDP 資訊。

（3） Controller 收到來自 UA1 的 200 OK 訊息，擷取其中的 SDP 資訊，以 INVITE 加上 SDP1 傳至 UA2。

（4） 回應 200 OK 並帶上 UA2 的 SDP 資訊。

（5） Controller 回應 ACK 並帶上 UA2 的 SDP 資訊給 UA1

（6） 至此 UA1 及 UA2 都有對方的 SDP 資訊，並建立通話。

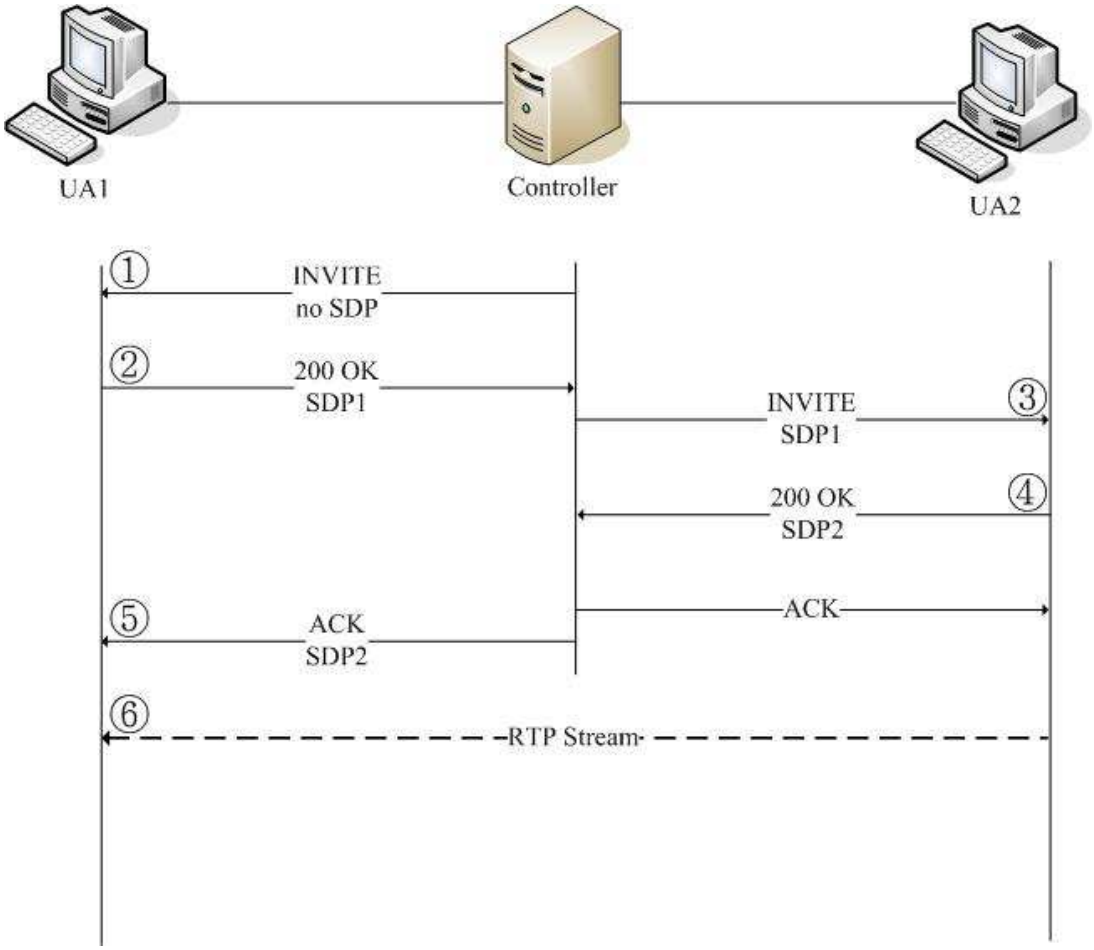
UA1
Controller
UA2
①
INVITE
no SDP
②
200 OK
SDP1
INVITE
SDP1
③
200 OK
SDP2
④
ACK
⑤
ACK
SDP2
⑥
RTP Stream

圖 9:簡易 3PCC 流程

4. 系統架構與實作成果

我們所提出的系統架構結合了 PSTN 的概念及 SIP 網路協定架構，讓使用 SIP 協定的網路電話也可以提供監聽功能。

這個架構主要達到幾個要求，第一個是監聽目標不會發現自己受到監聽；第二個是司法單位除了監聽通話內容之外，還能提供監聽的相關資訊；第三個是監聽的要求必須要經過 Central Server 確認才可開始監聽。圖 10 為改良的 VoIP 監聽架構。

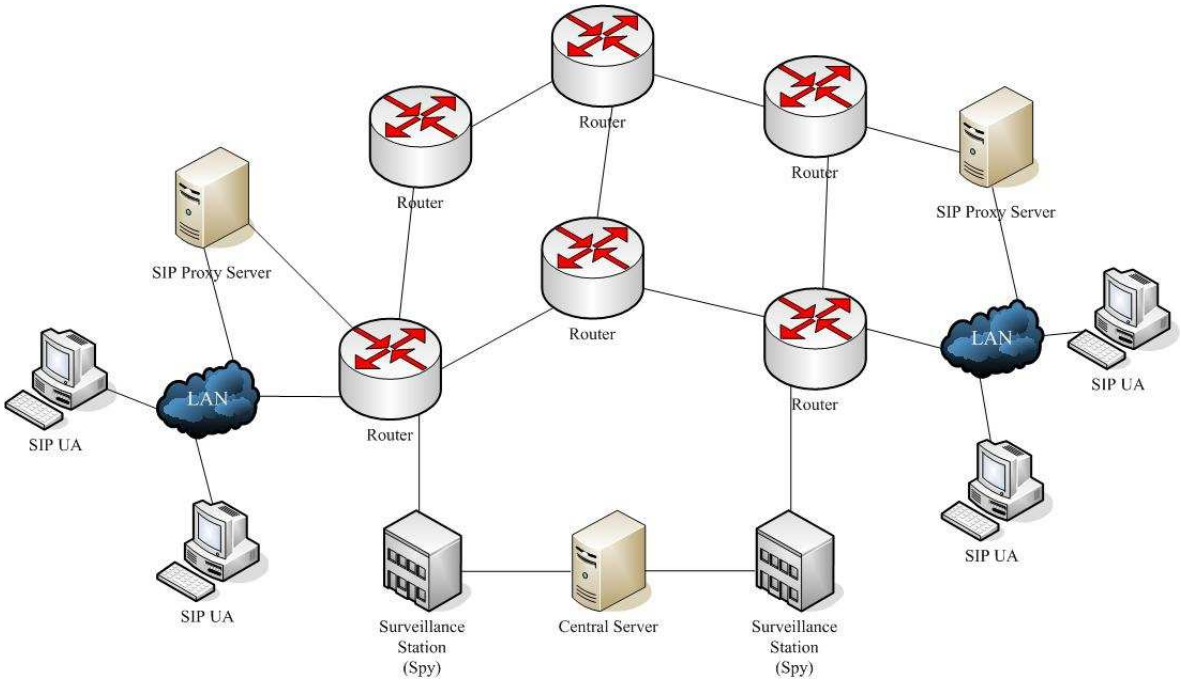

圖 10:改良的 VoIP 監聽架構

4.1 系統架構

4.1.1 VoIP Monitoring System

此 VoIP 監控平台是以 SIP 為基礎，圖 11 中以紅色虛線方框來表示主要的元件：

（1） 監聽人員（Supervisor；圖 11 ①）：可以在網路的任何一端使用網路連結到 Central Server，只需具備瀏覽器與 SIP User Agent，即可來聽取監聽內容。

（2） 網路電話伺服器&中央伺服器（SIP Proxy Server & Central Server；圖 11 ②）：SIP Proxy 主要功能是提供每個監聽設備及監聽人員 SIP 訊息的傳遞。Central Server 則是能將每個監聽伺服器所傳回的訊息整合，包括通聯訊息及監聽通話的建立。

（3） 監聽伺服器（Spy Server；圖 11 ③）：佈建在各個網域底下收集該網域的通聯訊息傳到 Central Server，負責將監聽人員所要監聽的通話傳至監聽人員電腦上。

（4） 網路電話使用者（User Agent；圖 11 ④）：受監測者網路電話端。

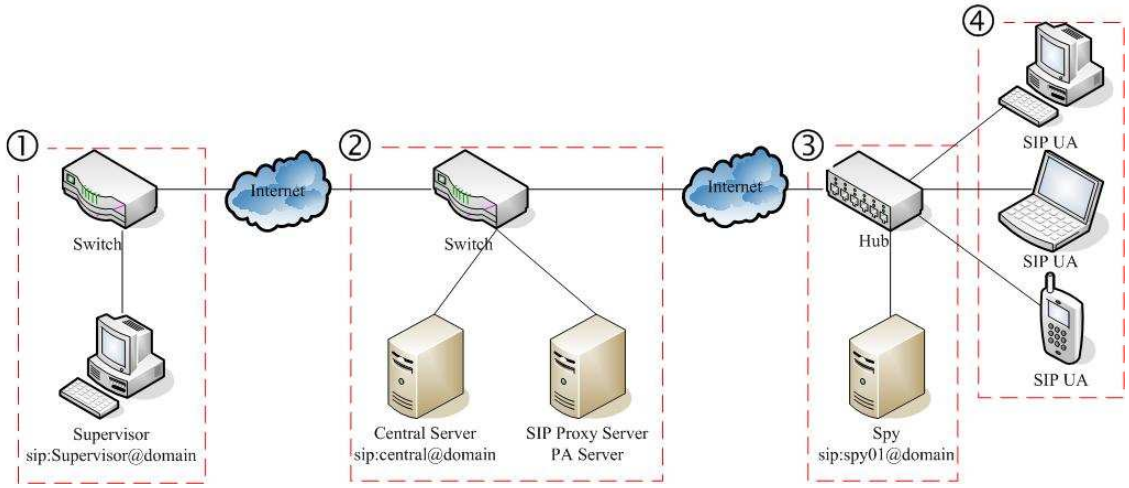

圖 11: VoIP Monitoring System 架構圖

4.1.2 Central Server and Spy Server component

圖 12 及圖 13 將描述每一個元件在實作上所使用的軟體函式庫。

（1） SIP Core module 裡面我們使用了 eXosip（圖 12①、圖 13②），主要負責 SIP 訊息的傳送及通話的建立。

（2） IRI（Intercept Related Information） Core module 是將來自各個 Spy Server 的更新訊息進一步分析，由於訊息內容為 XML 格式，因此我們使用了 libxml2（圖 12②）來加以解析。

（3） Multimedia Control module 使用 Speex（圖 13④）來進行聲音的壓縮，在 PCM 壓縮成 Speex 格式後，經由 oRTP（圖 13①）來實作 RTP，建

立 Spy Server 與監聽人員的 RTP sessions。

（4） Packet Capture module 使用的是 Libpcap（圖 13③），擷取網路上所流通的封包。

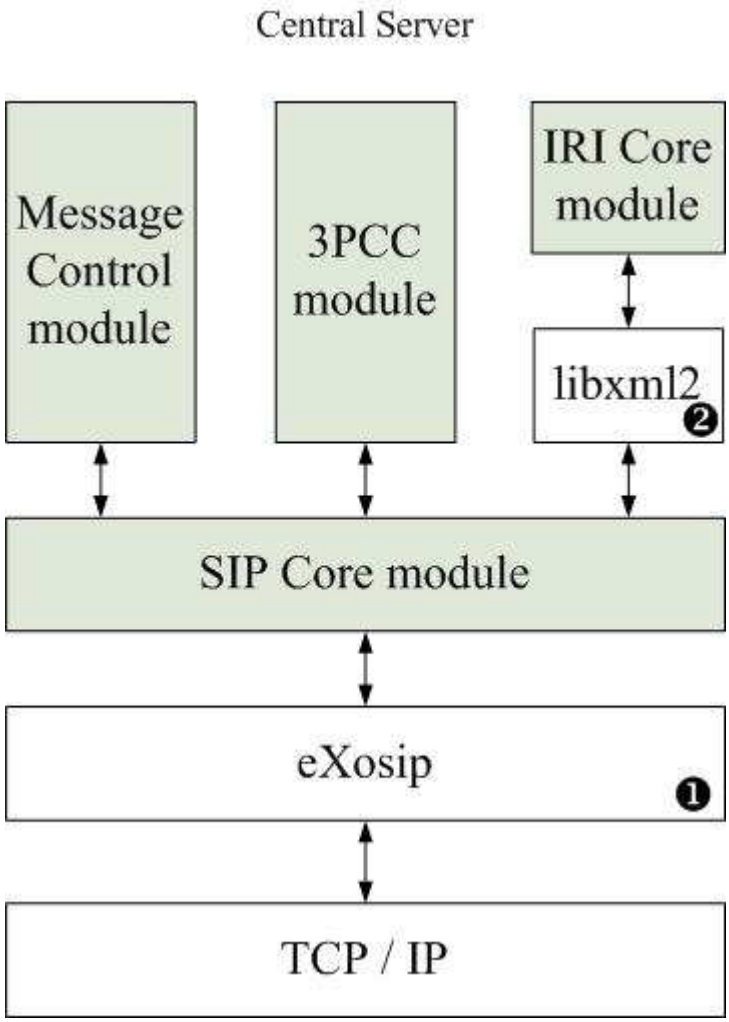

圖 12: Central Server component

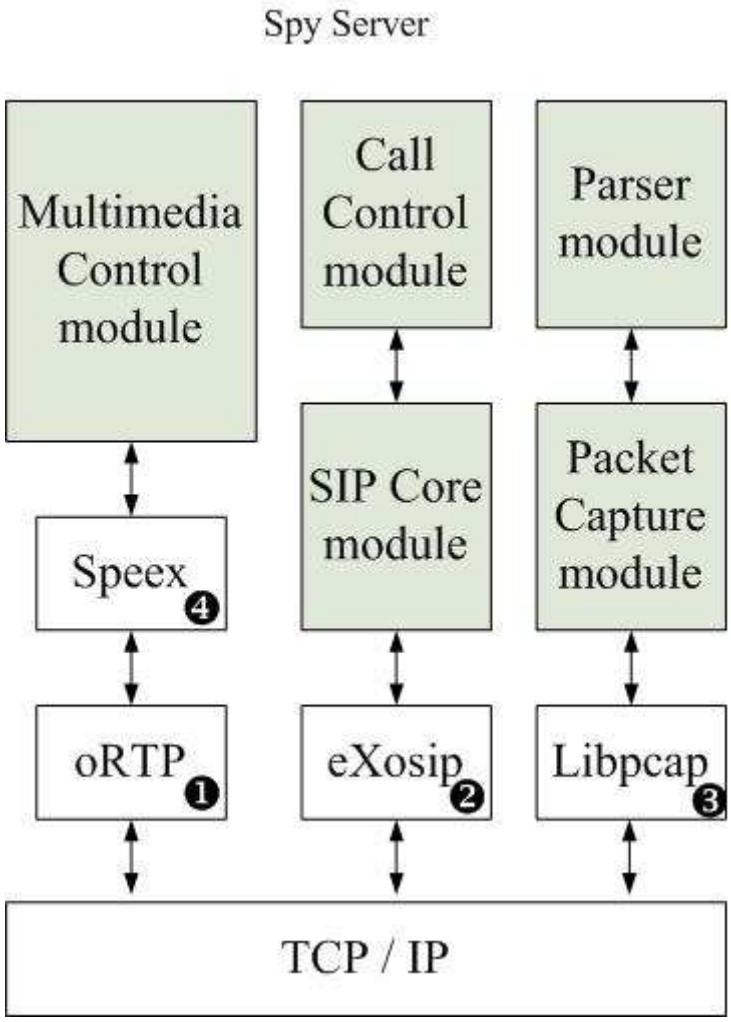

圖 13: Spy Server component

4.2 實作細節

4.2.1 VoIP Monitoring System Signaling Flow

圖 14 是 SIP-based 監控系統,我們可以發現此一監控系統可以分散在不同的網域底下進行監控的動作，其步驟說明如下：

Step 1：Spy 伺服器上線後向 Central Server 送出 MESSAGE 訊息，請求 Central Server 送出 SUBSCRIBE 動作。

Step 2：經由 Spy 伺服器的 MESSAGE 訊息的要求，向 SIP Proxy Server 送出 SUBSCRIBE Spy 伺服器的訊息。

Step 3：Spy 伺服器有更新訊息時，會經由 PUBLISH 訊息送至 SIP Proxy Server。

Step 4：SIP Proxy Server 收到 PUBLISH 訊息時，會以 NOTIFY 訊息傳給所有已 SUBSCRIBE 此一事件的 Central Server。

Step 5：監聽人員使用瀏覽器(圖 15)透過 HTTP 送出要求監聽某一通通話內容（Measurement-id）。建立通話的方式使用 Third party call control（3PCC）。

Steps 6~8：使用 3PCC 方式，開始由 Central Server 建立與監聽人員的通話過程，送出一個不含 SDP 的 INVITE 訊息，經由監聽人員電腦所送出的 200 OK 訊息中取得 SDP1。

Steps 9~12：收到 Step 8 的 200 OK 訊息中的 SDP1 後，再由 Central Server 送出帶有 SDP1 及 Measurement-id 的 INVITE 訊息給 Spy 伺服器，在 Spy 伺服器回傳的 200 OK 中取得 SDP2 再回傳給監聽人員。

Step 13：經由 Step 6~12 的通話建立完成後，會建立一個從 Spy 伺服器到監聽人員的單向 RTP Stream。

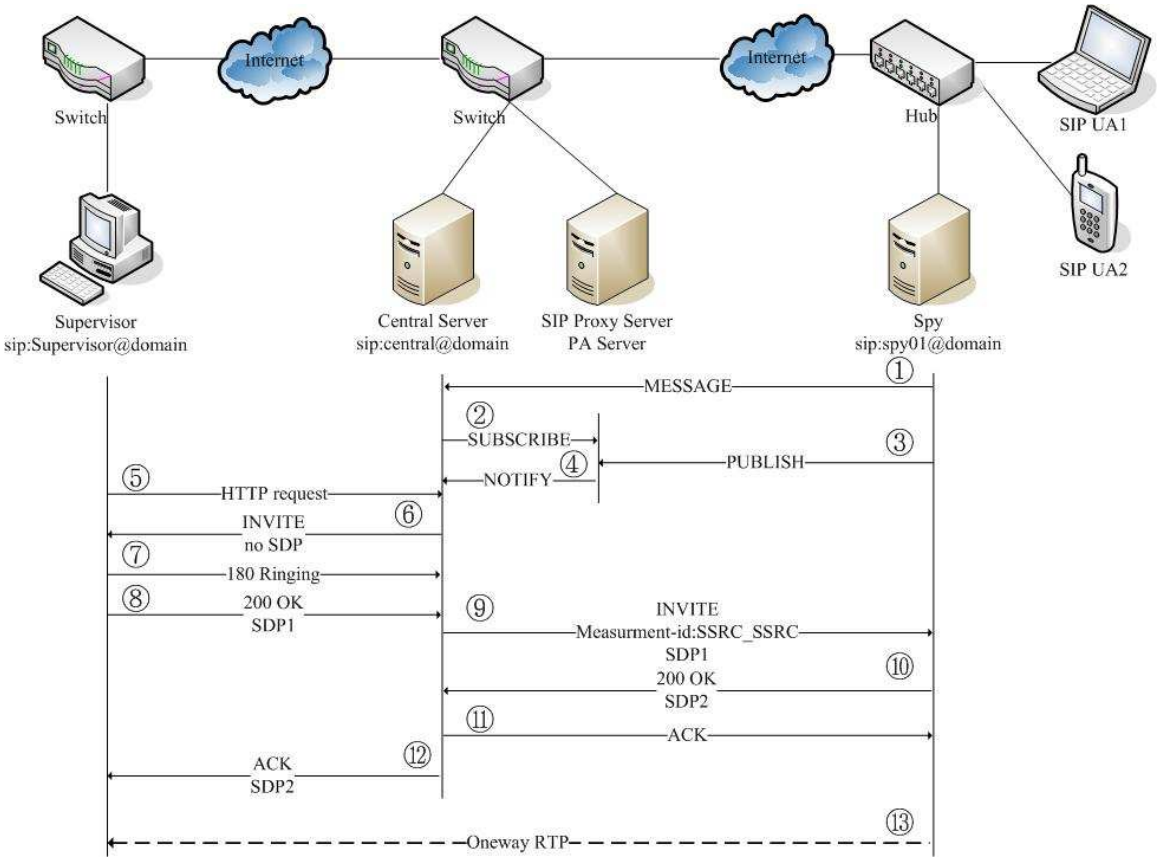

圖 14: VoIP Monitoring System 流程圖

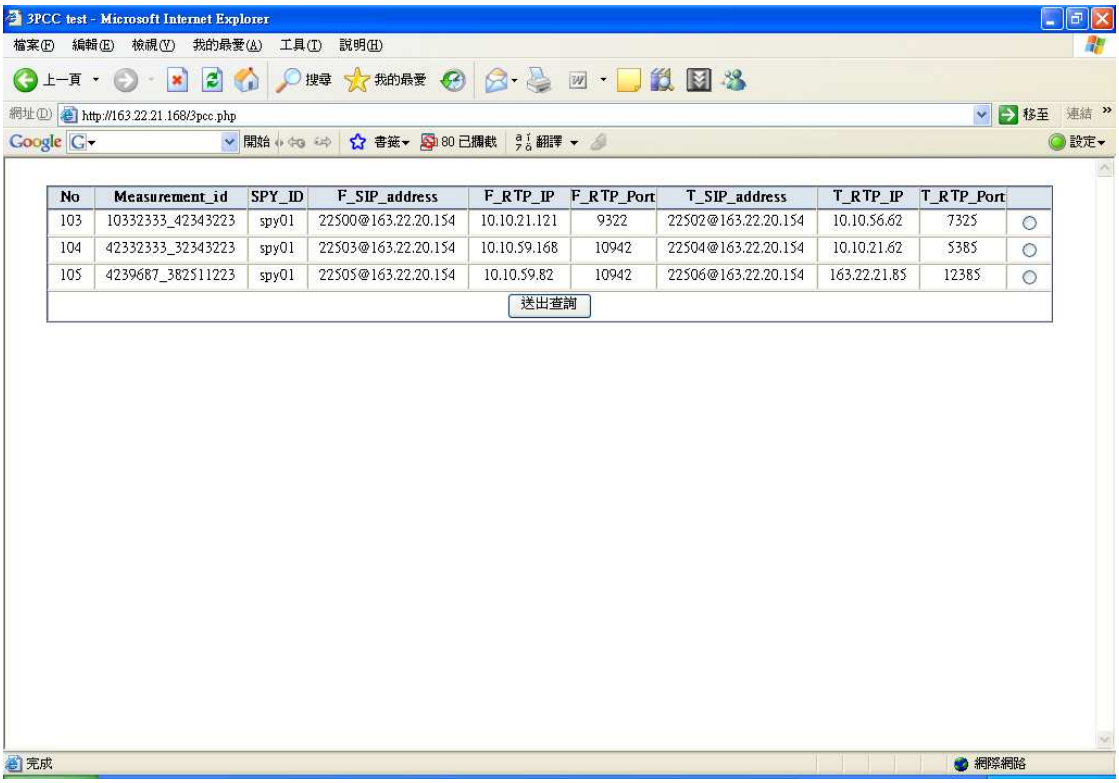

圖 15: 發送監聽要求介面

4.2.2 Central Server Signaling Flow

圖 16 解釋 Central Server 每個部份的完整流程及功能。

Step 1：Central Server 啟動後經過 Message Control module 接收來自 Spy Server 所發送的 PUBLISH 訊息。

Step 2：收到 Step1 的訊息以 libxml2 加以解析訊息中的資訊，更新到 IRI List 及資料庫。

Step 3：監聽人員透過 HTTP 送出監聽要求取得 Measurement-id 進行監聽的音訊。

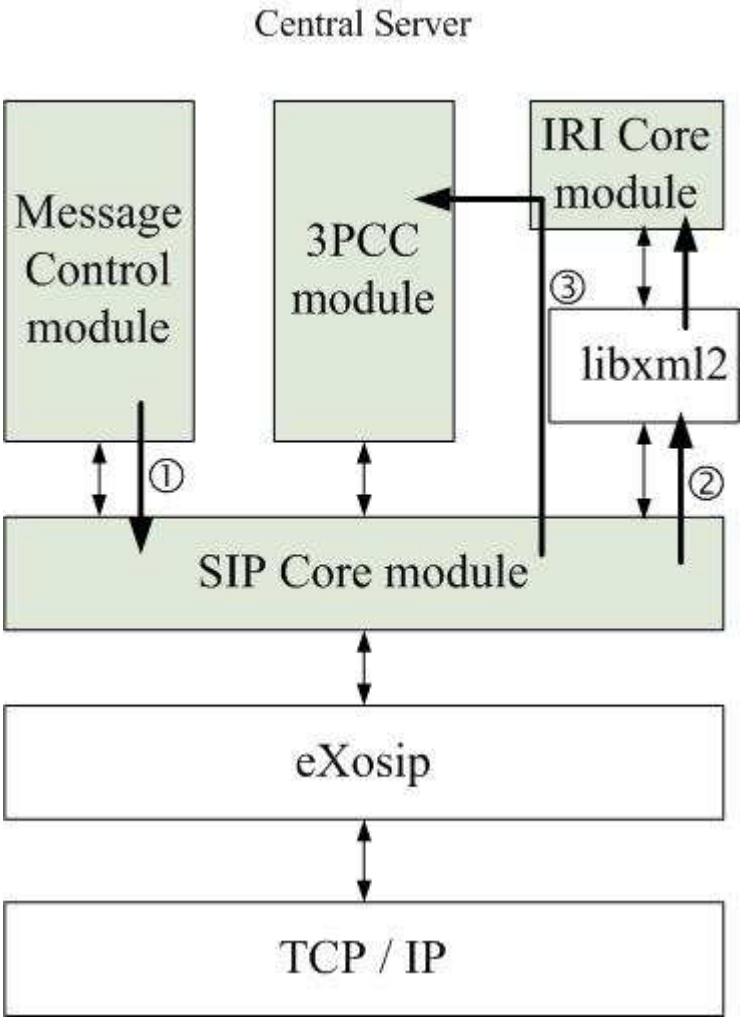

圖 16: Central Server 流程圖

4.2.3 Spy Server Signaling Flow

圖 17 解釋 Spy Server 每個部份的完整流程及功能。

Step 1：透過 Libpcap 從網路卡擷取封包，同時分析擷取的封包是否為 SIP 及 RTP 封包。

Step 2：分析 SIP 資訊包括 CALL-ID、SIP address 及 SDP，以建立通話關聯性。RTP 的資訊則從 SIP 中的 SDP 來與 RTP 的 IP address 及 Port number 建立關聯性，並將更新過的訊息傳到 Central Server。

Step 3：收到 Central Server 的通話要求，依照訊息建立通話。

Step 4：根據監聽人員傳來的 SDP 使用適合的音訊壓縮格式，並呼叫 oRTP library 來傳送音訊

Step 5：將從 Libpcap library 擷取的 RTP payload 經由 Speex 壓縮後，透過此介面進行壓縮及傳送。

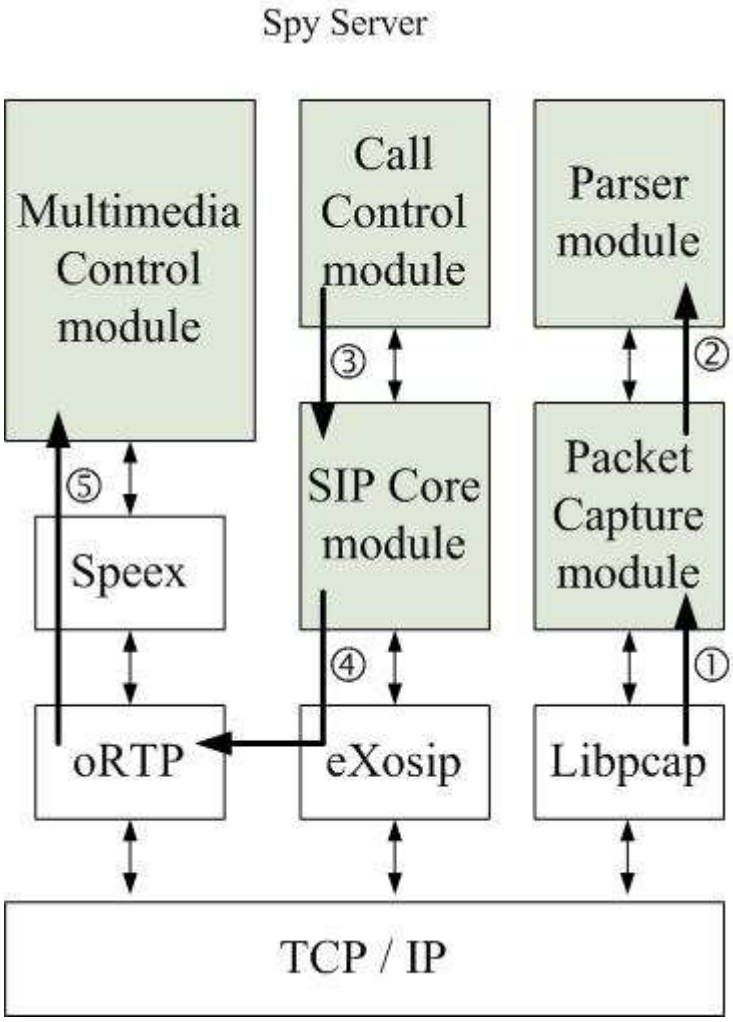

圖 17: Spy Server 流程圖

5. 效能測試

5.1 測試工具

我們將以兩個 User Agent 透過 SIP 伺服器建立通話的 INVITE 事件，讓兩個 User Agent 來進行 RTP stream 的傳輸，以進行壓力測試。要進行壓力測試，首先需要的是能夠大量製造 SIP 訊息的工具。而 SIPp 提供了這樣的功能。

SIPp 這個工具能夠模擬 User Agent Server（簡稱 UAS）及 User Agent Client（簡稱 UAC）的動作。在 SIPp 的功能當中，可以使用預設的功能來進行簡單的 INVITE 事件，由於預設的功能當中並沒有詳細設定進行 RTP stream 的傳輸，因此我們必須自行撰寫 XML(詳見附件 A)來訂定傳輸的 RTP 事件。

在圖 18、19 可以看到的是 UAC 在送出 INVITE 給 UAS 後，直到收到 200 OK 的訊息後會開始進行雙方就會開始 RTP 的傳送，在傳送期間會等待 40 秒，以完成 RTP 程序，在接收到 BYE 訊息後，結束此通話。

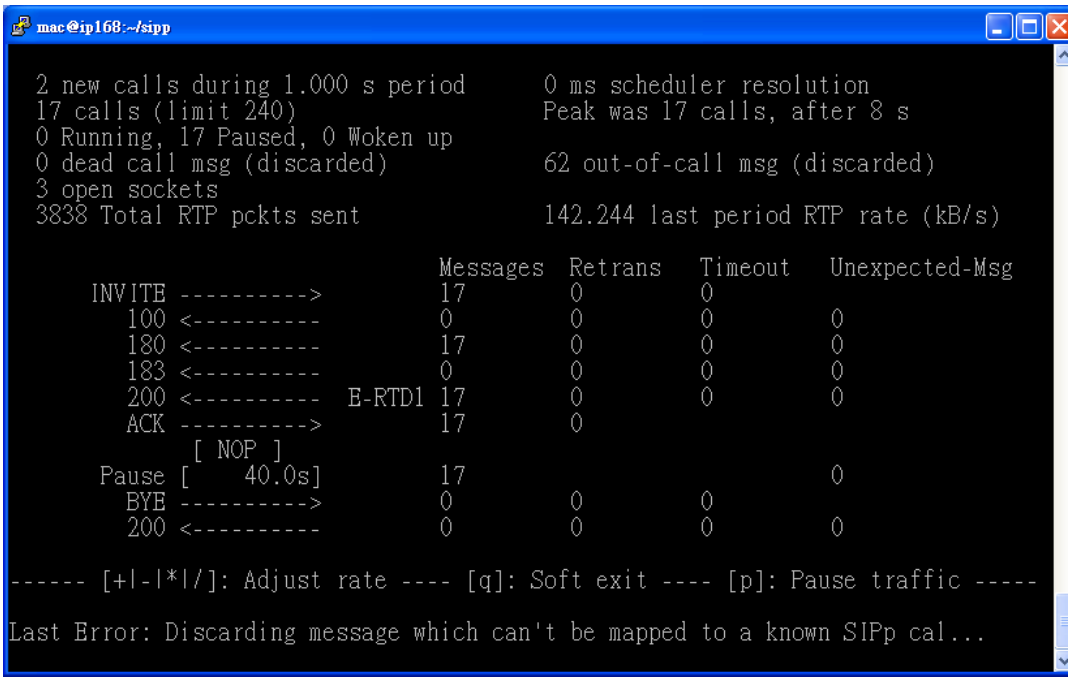

圖 18: UAC 示意圖

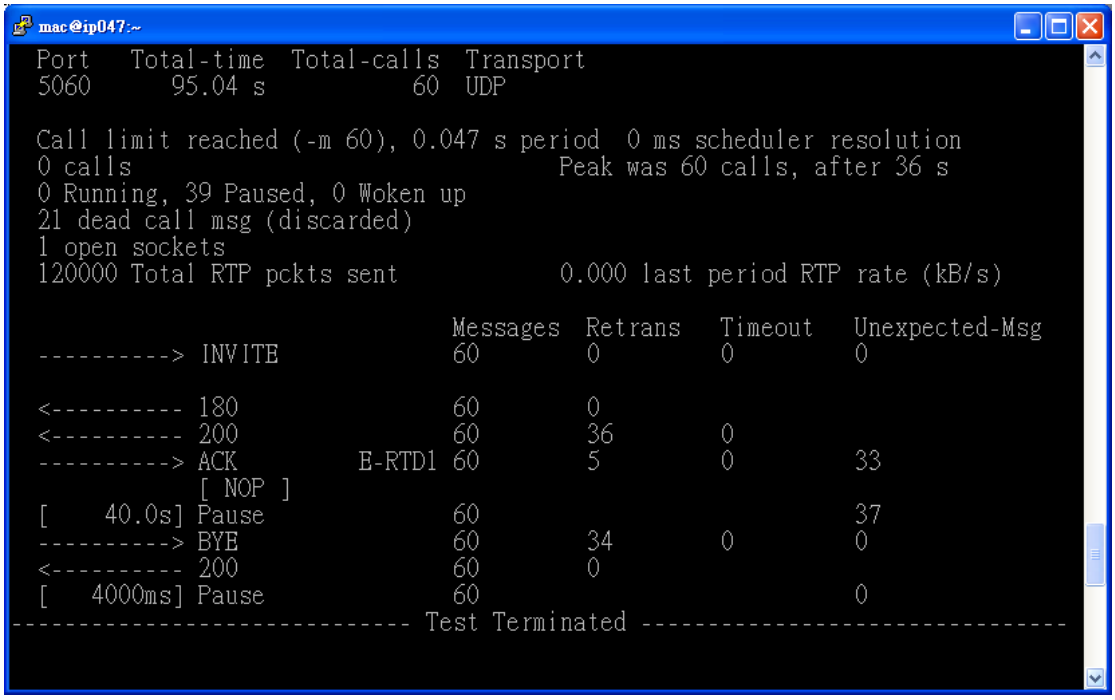

圖 19: UAS 示意圖

5.2 實驗環境

SPY Server 系統平台如下：

硬體：

（1）CPU：Intel(R) Pentium(R) 4 CPU 3.20GHz

（2）RAM：512MB

軟體：

（1）作業系統：LINUX Fedora Core 6

（2）libpcap：0.9.4-8.1

（3）libosip2：3.0.1-2.fc6

（4）libeXosip2：3.0.1-1.fc6

（5）ortp：0.13.0

（6）speex：1.2-0.1.beta1.fc6

Central Server 系統平台如下：

硬體：

（1）CPU：Intel(R) Pentium(R) Dual CPU E2140 @ 1.60GHz

（2）RAM：1024MB

軟體：

（1）作業系統：LINUX Fedora Core 7

（2）libosip2：3.0.3-2.fc7

（3）libeXosip2：3.0.3-1.fc7

（4）libxml2：2.6.28-2

（5）mysql：5.0.45-6.fc7

5.3 測試條件及結果

圖 20 為本系統之實驗與量測環境，其主要觀察的重點為 Spy Server 在進行大量通話監聽時，會不會造成擷取封包的不確實或者是進行多通監聽時對 Spy Server 的負擔。

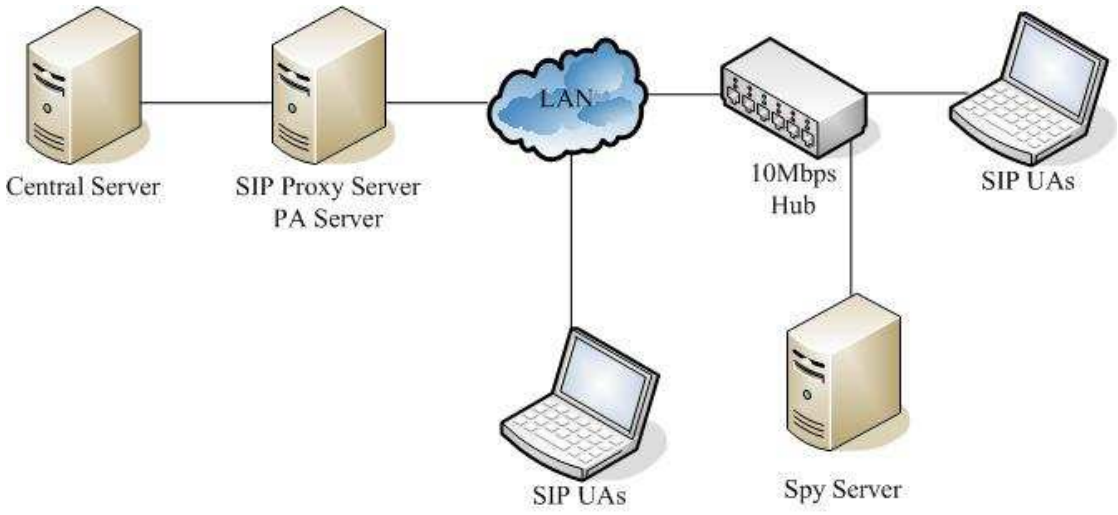

圖 20: 實驗與量測環境

我們使用 SIPp 這套軟體，在兩邊 UA 上同時產生大量的通話連線，藉此計算每次通話之封包遺失率及對 Spy Server 負擔。如圖 21 及表 3 所示，我們以 G.711 語音編碼格式做的量測結果。

我們可以由圖 21 看到藍色線條為兩邊 UA 所傳送的封包總數，黃色為 UA 端所接收到的封包數，粉色則是 SPY 端所接收到的封包數，由圖 21 及表 3 相互對照下，可以看出 Spy 在封包的擷取上可以很完整的擷取到。原因是出在於在取得 RTP 封包的方法是在網路節點上取得，因此只要封包在網路節點上經過，就能夠擷取得到。而我們可以再發現 concurrent calls 的增加，封包明顯的增加量變少，這是由於我們的實驗過程中是以每一個通話都是雙向 RTP 傳輸及使用 G.711 語音編碼，則是可能在瞬間眾多的封包湧入網路節點設備造成封包流量達到網路設備的極限及網路上封包碰撞所造成的遺失，無法再傳輸較多的封包量。

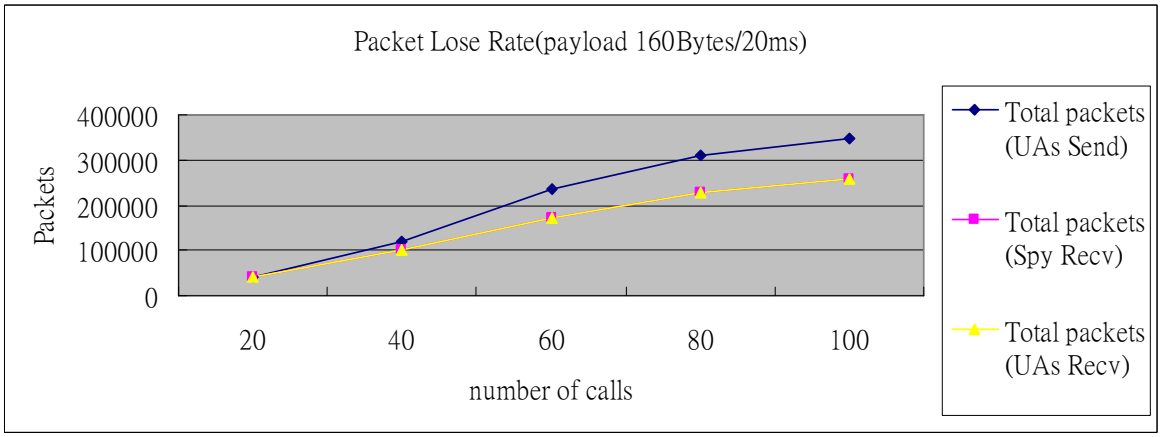

圖 21: Packet Lose Rate

表 3:Total packets and CPU、Memory loading

concurrent calls	Total packets (UAs Send)	Total packets (UAs Recv)	Total packets (Spy Recv)	CPU loading	Memory (MByte)
20	40000	40000	40000	4%	1.536
40	119999	101225	101225	10%	2.048
60	235998	170210	170210	12%	2.048
80	311998	227677	227677	12%	2.048
100	347999	258874	258874	14%	2.048

另一項實驗我們測試針對同一個監聽，SPY Server 能夠一次承受多少監聽人員的要求，在前一項實驗中，我們可以得知網路節點設備在 80~100 通間，封包數量增加的很慢，得知封包已達網路節點設備的極限，因此我們測試 SPY Server 在接受 20~80 通的監聽要求，程式的 CPU、Memory loading，由表 4 所示 CPU、Memory loading 在監聽要求數的增加，並無明顯的變化。

表 4: Multi monitor CPU、Memory loading

concurrent calls	CPU loading	Memory(MByte)
20	2%	1.536
30	2%	1.536
40	2%	2.048
50	2%	2.56
60	4%	3.072
70	4%	3.584
80	4%	4.096

6. 結論及未來方向

本系統能將眾多監聽設備放置在各個網域底下，將其收到的通聯訊息集中管理，在監聽時能由各個分散式的監聽設備負責聲音的傳送來減少中央伺服器的負擔。

在即時監聽時，能將聲音透過 Speex 壓縮，傳至監聽者的網路電話或行動電話上，並能透過 Call Out 的功能將可疑的通話傳至高階執法人員行動電話上，將可使監聽人員無須固定於監聽機房中，在犯罪偵察上可有更佳的彈性。

在各家網路電話廠商所開發的網路電話軟體，由於在建立通話過程加入了廠商所自行開發或自定欄位，例如 MSN 在建立過程中，使用兩層架構及自行開發的媒體格式；GTalk 則是使用另一種網路通訊協定。這使得資料判斷上不確實，但在聲音的傳輸上由於都是使用 RTP 協定，還是可以偵測出來，但使用自定的媒體格式則無法解譯出完整聲音。

未來我們希望能將所能監聽的媒體格式擴展到市面上通用的格式，在監聽的項目擴展到不只有聲音部份，並能將即時訊息（Instant message）和影像（Video）做同步監控。

參考文獻

[1] Logitech and Skype Announce Marketing Agreement <http://about.skype.com/2004/11/logitech_and_skype_announce_ma.html>

[2] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler, “SIP: Session Initiation Protocol,” IETF RFC 3261, Jun 2002.

[3] 自由電子報 - 第一家／ 070 網路電話 7.1 起哈啦 < http://www.libertytimes.com.tw/2007/new/apr/20/today-life1.htm>

[4] 第二類電信事業細部通訊監察需求表 <http://www.ncc.gov.tw/chinese/law_detail.aspx?sn_f=737>

[5] 第二類電信事業管理規則 <http://www.ncc.gov.tw/chinese/law_detail.aspx?sn_f=859>

[6] A. Johnston, S. Donovan, R. Sparks, C. Cunningham, and K. Summers, “Session Initiation Protocol (SIP) Basic Call Flow Examples,” IETF RFC 3665, Dec 2003.

[7] A. B. Roach, “Session Initiation Protocol (SIP)-Specific Event Notification,” IETF RFC 3265, Jun 2002.

[8] A. Niemi, “Session Initiation Protocol (SIP) Extension for Event State Publication,” IETF RFC 3903, Oct 2004.

[9] M. Handley, V. Jacobson, C. Perkins, “SDP: Session Description Protocol,” IETF RFC 4566, Jul 2006.

[10] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, “RTP: A Transport Protocol for Real-Time Applications,” IETF RFC 3550, Jul 2003.

[11] H. Schulzrinne, S. Casner, “RTP Profile for Audio and Video Conferences with Minimal Control,” IETF RFC 3551, Jul 2003.

[12] 全國法規資料庫-通訊保障及監察法

<http://law.moj.gov.tw/Scripts/Query4A.asp?FullDoc=all&Fcode=K0060044>

[13] 張慶龍, 彭耀民, 鄭玭吉, “SIP-based 網路電話監聽系統之設計與實現”, 二〇〇八數位生活科技研討會, pp. 58, Jun 2008.

[14] Chung-Hao Peng, “Lawful Interception Based on SIP”, Master Thesis, Institute of Computer Science and Information Engineering, National Chiao Tung University, Jul 2005.

[15] TCPDUMP public repository
<http://www.tcpdump.org/>

[16] Manpage of PCAP
<http://www.tcpdump.org/pcap3_man.html>

[17] The GNU oSIP library
<http://www.gnu.org/software/osip/>

[18] The eXtended osip library
<http://savannah.nongnu.org/projects/exosip>

[19] libosip Documentation
<http://www.gnu.org/software/osip/doc/html/index.html>

[20] libeXosip2 Documentation
<http://www.gnu.org/software/osip/doc/html/index.html>

[21] ortp, a Real-time Transport Protocol (RTP,RFC3550) library
<http://www.linphone.org/index.php/eng/code_review/ortp>

[22] oRTP API documentation
<http://download.savannah.gnu.org/releases/linphone/ortp/docs/>

[23] The XML C parser and toolkit of Gnome
<http://xmlsoft.org/>

[24] Reference Manual for libxml2

<http://xmlsoft.org/html/index.html>

[25] Speex: A Free Codec For Free Speech

<http://speex.org/>

[26] Speex: A Free Codec For Free Speech: Documentation

<http://speex.org/docs/>

[27] J. Rosenberg, J. Peterson, H. Schulzrinne, G. Camarillo, “Best Current Practices for Third Party Call Control (3pcc) in the Session Initiation Protocol (SIP),” IETF RFC 3725, Apr 2004.

附件

附件 A. SIPp scenario XML code

在撰寫自訂流程時，需要以標準的文法編輯，先定義此 scenario 的名稱，即可開始撰寫內容。主要的兩個命令為 send 及 recv，可以定義要送出什麼 SIP message，以及收到特定的 SIP message 時，要做出什麼回應。而傳送 RTP 封包則是使用<exec play_pcap_audio="*.pcap"/>語法，使用 wireshark 擷取到的 RTP 封包檔案來傳送。

A.1 UAC

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<scenario name="Basic Sipstone UAC">
  <send retrans="500">
    <![CDATA[

      INVITE sip:[service]@[remote_ip]:[remote_port] SIP/2.0
      Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
      From: sipp <sip:sipp@[local_ip]:[local_port]>;tag=[pid]SIPpTag00[call_number]
      To: sut <sip:[service]@[remote_ip]:[remote_port]>
      Call-ID: [call_id]
      CSeq: 1 INVITE
      Contact: sip:sipp@[local_ip]:[local_port]
      Max-Forwards: 70
      Subject: Performance Test
```

```
        Content-Type: application/sdp
        Content-Length: [len]

        v=0
        o=user1 53655765 2353687637 IN IP[local_ip_type] [local_ip]
        s=-
        c=IN IP[media_ip_type] [media_ip]
        t=0 0
        m=audio [auto_media_port] RTP/AVP 0
        a=rtpmap:0 PCMU/8000

    ]]>
</send>

<recv response="100"
          optional="true">
</recv>

<recv response="180" optional="true">
</recv>

<recv response="183" optional="true">
</recv>

<recv response="200" rtd="true">
```

```
</recv>

<send>
   <![CDATA[

      ACK sip:[service]@[remote_ip]:[remote_port] SIP/2.0
      Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
      From: sipp <sip:sipp@[local_ip]:[local_port]>;tag=[pid]SIPpTag00[call_number]
      To: sut <sip:[service]@[remote_ip]:[remote_port]>[peer_tag_param]
      Call-ID: [call_id]
      CSeq: 1 ACK
      Contact: sip:sipp@[local_ip]:[local_port]
      Max-Forwards: 70
      Subject: Performance Test
      Content-Length: 0

   ]]>
</send>

<nop>
   <action>
      <exec play_pcap_audio="uac.pcap"/>
   </action>
</nop>
<pause milliseconds="8000" />
```

```
  <!-- The 'crlf' option inserts a blank line in the statistics report. -->
  <send retrans="500">
    <![CDATA[

      BYE sip:[service]@[remote_ip]:[remote_port] SIP/2.0
      Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
      From: sipp <sip:sipp@[local_ip]:[local_port]>;tag=[pid]SIPpTag00[call_number]
      To: sut <sip:[service]@[remote_ip]:[remote_port]>[peer_tag_param]
      Call-ID: [call_id]
      CSeq: 2 BYE
      Contact: sip:sipp@[local_ip]:[local_port]
      Max-Forwards: 70
      Subject: Performance Test
      Content-Length: 0

    ]]>
  </send>

  <recv response="200" crlf="true">
  </recv>

  <ResponseTimeRepartition value="10, 20, 30, 40, 50, 100, 150, 200"/>
  <CallLengthRepartition value="10, 50, 100, 500, 1000, 5000, 10000"/>
</scenario>
```

A.2 UAS

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<scenario name="Basic UAS responder">
  <recv request="INVITE" crlf="true">
  </recv>

  <send>
    <![CDATA[

      SIP/2.0 180 Ringing
      [last_Via:]
      [last_From:]
      [last_To:];tag=[pid]SIPpTag01[call_number]
      [last_Call-ID:]
      [last_CSeq:]
      Contact: <sip:[local_ip]:[local_port];transport=[transport]>
      Content-Length: 0

    ]]>
  </send>

  <send retrans="500">
    <![CDATA[
```

```
      SIP/2.0 200 OK
      [last_Via:]
      [last_From:]
      [last_To:];tag=[pid]SIPpTag01[call_number]
      [last_Call-ID:]
      [last_CSeq:]
      Contact: <sip:[local_ip]:[local_port];transport=[transport]>
      Content-Type: application/sdp
      Content-Length: [len]

      v=0
      o=user1 53655765 2353687637 IN IP[local_ip_type] [local_ip]
      s=-
      c=IN IP[media_ip_type] [media_ip]
      t=0 0
      m=audio [auto_media_port] RTP/AVP 0
      a=rtpmap:0 PCMU/8000

    ]]>
</send>

<recv request="ACK"
          rtd="true" >
        <!-- crlf="true"> -->
```

```
</recv>

<nop>
   <action>
      <exec play_pcap_audio="uas.pcap" />
   </action>
</nop>

<pause    milliseconds="8000"/>

<recv request="BYE">
</recv>

<send>
   <![CDATA[

      SIP/2.0 200 OK
      [last_Via:]
      [last_From:]
      [last_To:]
      [last_Call-ID:]
      [last_CSeq:]
      Contact: <sip:[local_ip]:[local_port];transport=[transport]>
      Content-Length: 0
```

```
        ]]>
    </send>

    <timewait milliseconds="4000"/>
    <ResponseTimeRepartition value="10, 20, 30, 40, 50, 100, 150, 200"/>
    <CallLengthRepartition value="10, 50, 100, 500, 1000, 5000, 10000"/>
</scenario>
```