國立交通大學

資訊學院　資訊學程

碩士論文

IETF/3GPP SIP 使用者代理程式之研製

Design and Implementation of an IETF/3GPP SIP User Agent

研究生：蔡國泰

指導教授：林一平　教授

陳懷恩　教授

中華民國九十九年八月

IETF/3GPP SIP 使用者代理程式之研製

Design and Implementation of an IETF/3GPP SIP User Agent

研 究 生：蔡國泰　　Student：Kuo-Tai Tsai

指導教授：林一平 博士　　Advisor：Dr. Yi-Bing Lin

陳懷恩 博士　　Dr. Whai-En Chen

國 立 交 通 大 學

資訊學院　資訊學程

碩 士 論 文

A Thesis

Submitted to College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master of Science

in

Computer Science

August 2010

Hsinchu, Taiwan, Republic of China

中 華 民 國 九 十 九 年 八 月

IETF/3GPP SIP 使用者代理程式之研製

學生：蔡國泰　　指導教授：林一平 博士

陳懷恩 博士

國 立 交 通 大 學　　資 訊 學 院　　資 訊 學 程 碩 士 班

摘要

VoIP (Voice over IP) 為現行最重要的網路服務之一，此服務建置於 3GPP (3rd Generation Partnership Project) IP 多媒體子系統 (IP Multimedia Core Network Subsystem; IMS) 及 IETF (Internet Engineering Task Force) SIP (Session Initiation Protocol) 網路架構上。為了在此兩種網路架構中評估 UE (User Equipment) 與核心網路之間的服務效能，本論文在 Windows Mobile 為作業系統的 UE 上研製了一套 VoIP UA (User Agent) 軟體。並以通話建立 (call setup) 為例。利用此 UA 軟體來測量 UE 於不同無線網路、不同傳輸協定、有無啟用身份認證、QoS (Quality of Service) 協商與 TLS (Transport Layer Security) 安全機制下的撥號後延遲 (Post Dialing Delay; PDD)。最後的測量結果可提供網路業者建置 VoIP 服務時的參考。

關鍵字：VoIP、IMS、SIP、PDD、QoS、TLS

Design and Implementation of an IETF/3GPP SIP User Agent

Student：Kuo-Tai Tsai　　Advisor：Dr. Yi-Bing Lin

Dr. Whai-En Chen

Degree Program of Computer Science

National Chiao Tung University

ABSTRACT

Voice over IP (VoIP) is one of the most important network services in recent years. This service is deployed in the 3rd Generation Partnership Project (3GPP) IP Multimedia Core Network Subsystem (IMS) and Internet Engineering Task Force (IETF) Session Initiation Protocol (SIP) network architectures. To evaluate the VoIP performance between the User Equipment (UE) and core network, this thesis designs and implements a VoIP User Agent (UA) on the UE with Windows Mobile operating system. This thesis uses the post dialing delay (PDD) as the measurement for evaluating the service quality of the call setup under different wireless networks, transport protocols, and negociation procedures (authentication, quality of service (QoS), and transport layer security (TLS)). The results would be meaningful for network operators to deploy VoIP service.

Keywords: VoIP, IMS, SIP, PDD, QoS, TLS

誌謝

首先我要感謝指導教授林一平博士與陳懷恩博士。林一平博士在研究上的細心嚴謹，以及做研究的認真熱情，一直是我學習的典範。陳懷恩教授擁有豐富的學術知識與專業經驗，對研究充滿創意。從他身上我獲得許多研究想法與學習多元思考。我很榮幸能成為他們的學生，沒有他們在研究上耐心指導，我無法完成此碩士論文。

接著感謝林風博士與陳元凱博士在口試時給予的寶貴建議，讓此篇論文能更加完整。以及感謝 117 實驗室的學長姊、同學、與學弟妹們，如家人般帶給我關懷與歡樂。而實驗室勤學不倦的風氣，也深深影響我積極向上。論文撰寫過程中，我得到孟勳學長、皇甫學長、依寰學長、品任學長、及仁煌同學的幫助，與他們討論，我總能獲得解決問題的方向。特別是仁煌同學，這段期間我們一起分享研究經驗、討論問題，他是我研究所生涯不可或缺的朋友。我也要感謝之皓學妹提供 3.5G SIM 卡，讓真實網路下的實驗得以順利完成。論文校閱過程中，也感謝彥能學弟與明興學弟幫忙掃描論文。

另一方面，我要感謝公司主管 Jason 與 Charles 對我就讀研究所的支持與關心，及同事們的鼓勵。讓我在完成學業的同時，也能在工作上充滿活力。

最後，我要感謝我的家人。敬愛的父母親，在我求學過程中，總是默默全力地支持我。岳父岳母假日幫忙我們照顧小孩，讓我有更多時間做研究。老婆宜君，感謝妳這幾年的陪伴、打理家務、教養博特和安棋兩個孩子。有妳為這個家辛苦的付出，我才能順利完成學業。博特和安棋，爸爸終於把論文寫完了，謝謝你們每天幫爸爸加油打氣，讓爸爸克服萬難，勇往直前。

目錄

圖目錄

表目錄

第一章 簡介

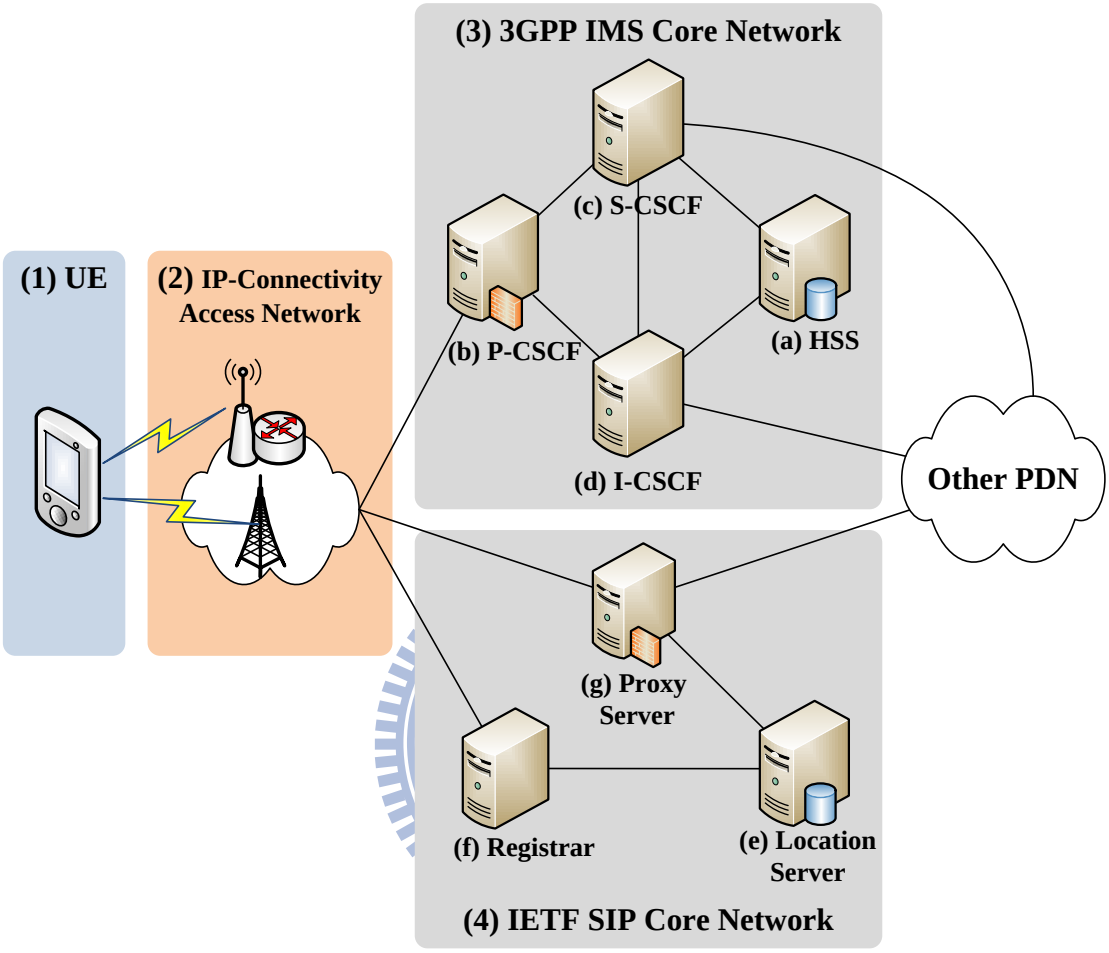

圖 1.1: 3GPP IMS 與 IETF SIP 網路架構

VoIP (*Voice over IP*) 為現行最重要的網路服務之一，此服務建置於 IP 多媒體子系統 (*IP Multimedia Core Network Subsystem*; IMS) [1] 與 SIP (*Session Initiation Protocol*) [2] 網路架構上。IMS 為第三代合作夥伴計畫 (*3rd Generation Partnership Project*; 3GPP) 所規劃的多媒體服務基礎架構，它在第三代行動通訊系統 (*Third Generation*; 3G) 中扮演極重要的角色。簡化的 IMS 架構如圖 1.1 (1)(2)(3) 所示，此架構包含 UE (*User Equipment*; 圖 1.1 (1))、IP 連結存取網路 (*IP-Connectivity Access Network*; 圖 1.1 (2)) 與核心網路 (*Core Network*; 圖 1.1 (3))。UE 為具行動通訊裝置之個人電腦或手機。UE 提供 UA (*User Agent*)

軟體讓終端用戶 (end user) 透過 IP 連結存取網路連結到核心網路。核心網路包含 HSS (*Home Subscriber Server*; 圖 1.1 (a)) 與 CSCFs (*Call Session Control Functions*; 圖 1.1 (b)(c)(d))。HSS 為 IMS 中最主要的資料庫，負責儲存使用者相關資訊。CSCFs 的角色類似於 SIP 代理伺服器。CSCFs 依不同功能，分成 P-CSCF (*Proxy CSCF*; 圖 1.1 (b))、S-CSCF (*Serving CSCF*; 圖 1.1 (c)) 與 I-CSCF (*Interrogating CSCF*; 圖 1.1 (d))。P-CSCF 為 UE 連結核心網路的第一個節點，負責決定信令傳遞的路徑，並提供承載資源 (bearer resource) 的認證。S-CSCF 負責服務 UE，以進行註冊、設定通話線路與提供增添服務 (supplementary service) 等功能。I-CSCF 是來話 (incoming call) 從客籍網路 (visited network) 進入 IMS 的連結點，其功用為本籍網路 (home network) 的防火牆與閘道，可對外隱藏本籍網路內部的資訊。

另一種網路架構為網際網路工程專案小組 (*Internet Engineering Task Force*; IETF) 所制定的 SIP 網路架構 (圖 1.1 (4))，目前被廣泛使用在 VoIP 服務上。IETF SIP 核心網路由三種伺服器所組成：位置伺服器 (Location Server; 圖 1.1 (e)) 用於儲存 UA 目前位址。註冊伺服器 (Registrar; 圖 1.1 (f)) 處理 UA 發出的註冊要求，並在 UA 成功註冊後，向位置伺服器更新 UA 所在的位址。代理伺服器 (Proxy Server; 圖 1.1 (g)) 除了向位置伺服器詢問受話端位址，並負責 SIP 信令的轉送。

現行 VoIP 服務建置於上述 SIP 為基礎的 3GPP IMS 及 IETF SIP 兩種網路架構上。例如行動電信業者採用前者，而網路電話服務業者 (*Internet Telephony Service Provider*; ITSP) 採用後者。在此兩種架構中，終端用戶必須以 UE 為媒介，透過 UE 上的 UA 軟體來與核心網路進行服務。為了評估 UE 與核心網路之間的服務效能，本論文參考 [3, 4] 提出的實驗方法，以通話建立 (call setup) 為例，分別在可控制與實際網路下測量 UE 與核心網路間的通話建立效能。相較於 [3, 4]，本論文採用智慧型手機 (smartphone) 做為實

驗的 UE (將於後續說明)，並於通話建立程序上加入 TCP 連結傳輸、身份認證與安全加密機制等延伸功能來進行測量。藉由通話建立複雜度的增加，探討延伸功能對原有服務效能的影響。

由於近年來智慧型手機的興起 (如 Windows Mobile、iPhone 與 Android phone)，以及在行動裝置上使用個人服務的接受度與普及率攀升，未來智慧型手機勢必成為重要的服務媒介之一。本論文考量到 Windows Mobile 智慧型手機較早被推行，因此在開發資源上較為完整，所以決定以 Windows Mobile 智慧型手機做為實驗的 UE。接著，本論文評估現行可運作於行動通訊平台上的 UA 軟體，如 fring [5] 與 Nokia N95 手機 [6] 內建的 SIP UA 等。fring 支援多種行動平台，及整合多種服務業者提供的 VoIP 功能，如 MSN、GoogleTalk 與 Facebook。但 fring 不支援 3GPP 服務標準與 TLS (*Transport Layer Security*) [7] 安全加密保護。Nokia N95 手機內建的 SIP UA 雖然支援了 3GPP IMS 與 IETF SIP 兩種服務標準以及安全加密保護，但未提供 QoS (*Quality of Service*) 資源協商功能。上述 UA 軟體雖然已有部份可利用的功能，但本論文無法取得這些軟體的原始碼，以致無法在軟體上加入效能測量相關程式。

而本實驗室在 Windows Mobile 上已有基本的 UA 軟體開發成果 [8]。此 UA 軟體可提供註冊認證、通話建立等基本 VoIP 功能。不過此 UA 軟體尚未支援 3GPP IMS 服務標準、TCP 傳輸協定、QoS 協商與 TLS 安全保護。所以，本論文決定基礎於此 UA 軟體來進行功能的開發，並對所開發之新 UA 軟體稱作 SIP/IMS UA。SIP/IMS UA 支援了 3GPP IMS 與 IETF SIP 兩種網路服務架構，及上述原 UA 軟體未提供的功能，並具備即時訊息 (*Instant Message*) [9]、通話保留 (*Communication Hold*) [10]、通話轉接 (*Communication Transfer*) [11] 與 NAT (*Network Address Translation*) [12] 穿越等能力。後續本論文將利用 SIP/IMS UA 來進行各種效能測量實驗。

本論文的章節組織描述如下。第二章說明本論文所實作之 SIP/IMS UA 的系統設計與內部模組功能。第三章描述 SIP 通話建立的信令流程。第四章使用 SIP/IMS UA 來進行實驗與分析實驗結果。第五章為總結，並提出未來工作方向。

第二章 SIP/IMS UA 設計與實作

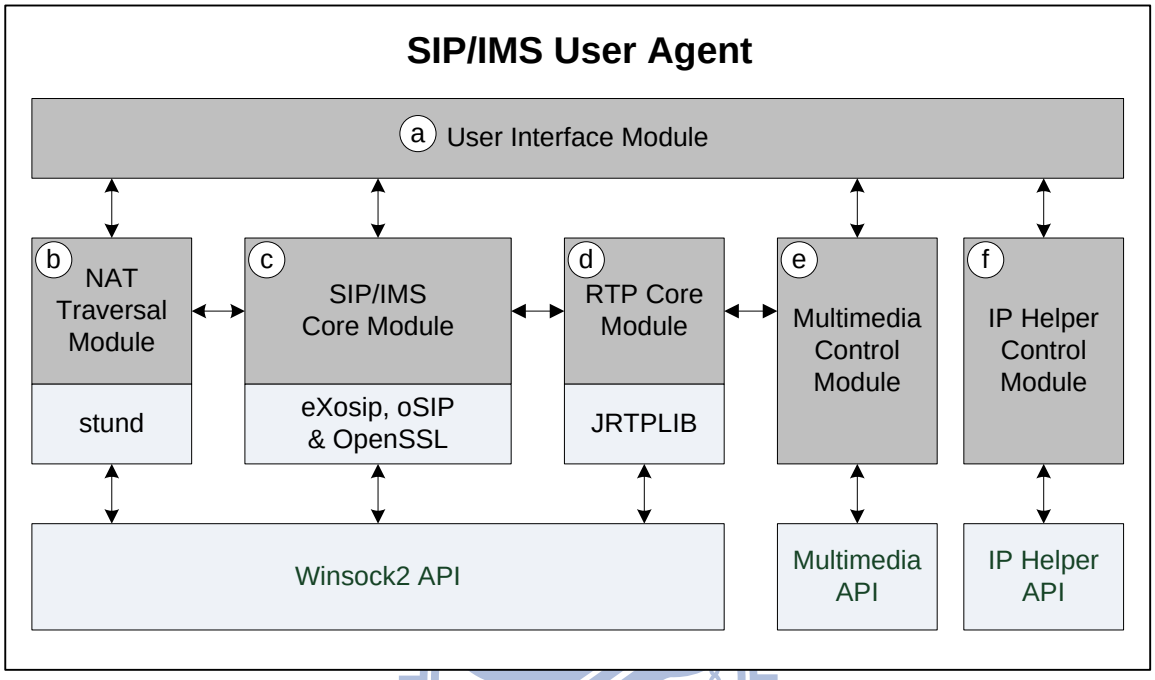

圖 2.1: SIP/IMS UA 之軟體架構圖

本章節說明本論文實作之 SIP/IMS UA 的軟體架構與內部模組功能。本論文參考 [8] 所設計之 VoIP 軟體架構來發展 SIP/IMS UA，加強了原有 User Interface Module (圖 2.1ⓐ) 與 SIP/IMS Core Module 的功能 (圖 2.1ⓒ)，並新增 NAT Traversal Module (圖 2.1ⓑ)。後續小節將詳述以上三個模組與簡介其他模組。

2.1 User Interface Module

User Interface Module (圖 2.1ⓐ) 提供圖形化使用者介面，讓使用者透過此介面的操作 (例如按下按鈕或選取選單)，來啟動服務功能或取得 UA 狀態資訊。本論文對原有

VoIP 軟體的使用者介面進行修改，以新增 3GPP IMS 服務標準、即時訊息 [9]、通話保留 [10] 與通話轉接 [11] 等功能。接著說明 User Interface Module 於修改後提供的使用者介面。

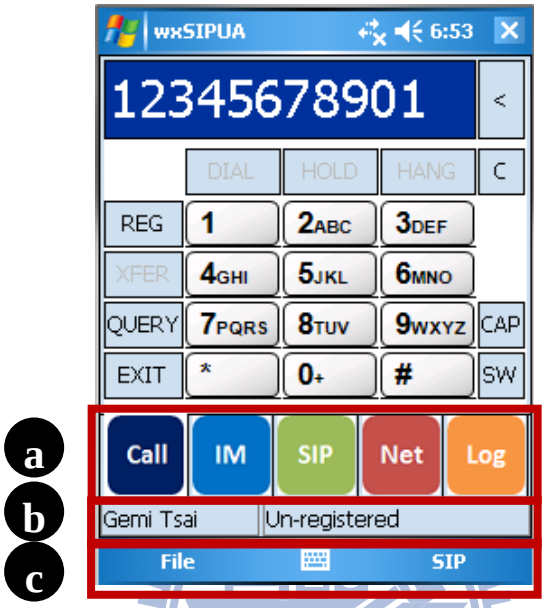

圖 2.2: 工具列

如圖 2.2 所示，User Interface Module 將使用者介面規劃成三個工具列 (toolbar) 與五種頁面 (page)。工具列包含頁面選擇列 (Page Bar; 圖 2.2ⓐ)、狀態列 (Status Bar; 圖 2.2ⓑ) 與功能表列 (Menu Bar; 圖 2.2ⓒ)。頁面選擇列用於切換目前操作頁面，利用列上的圖示鈕，由左至右可切換至通話功能頁面、即時訊息頁面、SIP/IMS 設定頁面、網路設定頁面與訊息日誌頁面。狀態列可顯示服務狀態資訊，左欄為使用者名稱 (如圖 2.2ⓑ之 “Gemi Tsai”)，也是發起通話與進行即時通訊所用的識別名稱；右欄為 UA 目前狀態 (如圖 2.2ⓑ之 “Un-registered”)，會依不同事件而改變。功能表列提供表單型態的功能操作。以下分別介紹上述五種頁面。

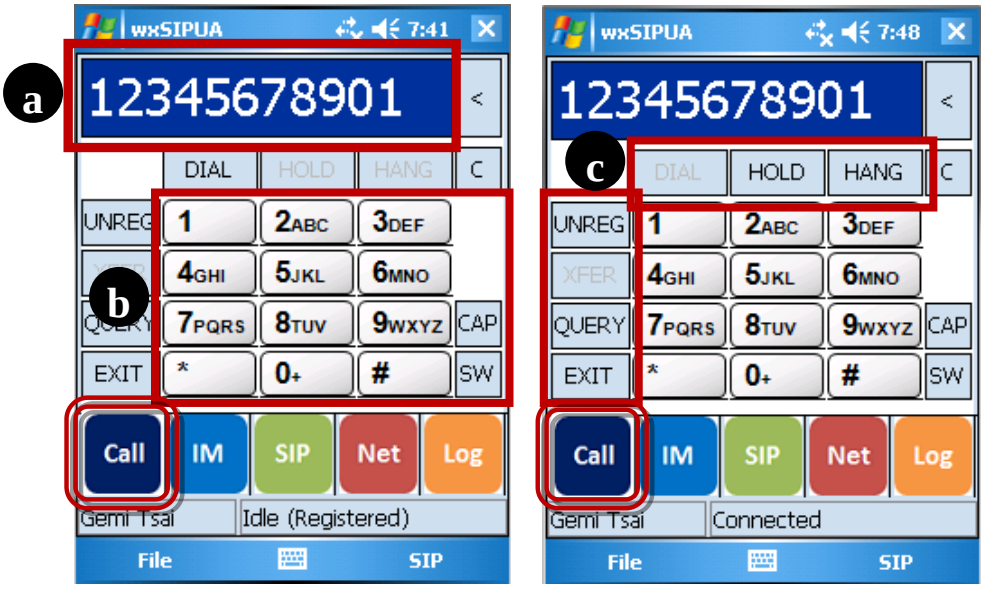

圖 2.3: 通話功能頁面

通話功能頁面 (圖 2.3) 包含撥號顯示列 (Dial Display; 圖 2.3ⓐ)、撥號鍵 (Dial Keypad; 圖 2.3ⓑ) 與功能鍵 (Function Keypad; 圖 2.3ⓒ)。撥號顯示列用於顯示使用者輸入的撥號資料或來電者的名稱。撥號鍵協助使用者輸入撥號資料，並可利用 CAP 鍵來切換撥號顯示列最後一字元的英文大小寫 (若為數字則不受改變)，或以 SW 鍵來切換撥號鍵所輸出的字元 (如撥號鍵 “2ABC” 之輸出字元將依序切換為 “2”、“A”、“B”、“C”)。若資料輸入錯誤時，可利用撥號顯示列右邊的倒退鍵 “<” 或清除鍵 “C” 來刪除輸入的資料。功能鍵用於啟動各種服務功能，各功能鍵說明如下：REG/UNREG 用於註冊與解除註冊；DIAL 用於撥號建立通話；HANG 用於終止目前通話；HOLD/RESUME 用於保留通話與恢復通話；XFER 負責啟動通話轉接；QUERY 可查詢 SIP/IMS UA 執行中各參數目前狀態值；EXIT 用於結束 UA 的執行。

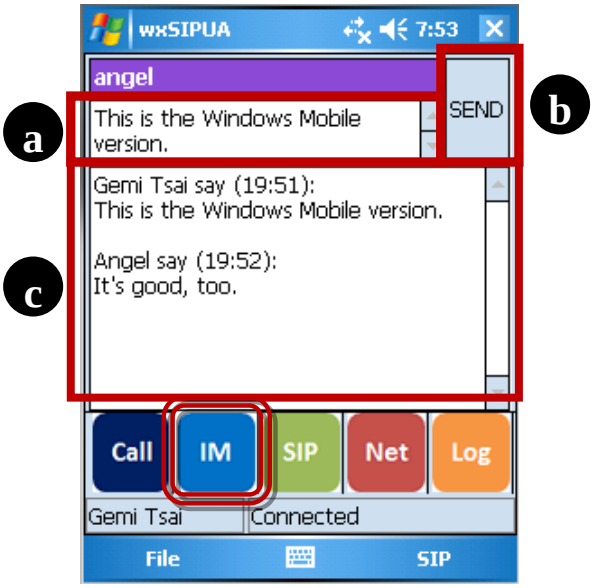

圖 2.4: 即時訊息頁面

即時訊息頁面 (圖 2.4) 提供即時訊息服務功能。使用者由訊息輸入區 (Message Input Area; 圖 2.4ⓐ) 輸入接收者名稱與欲發送的訊息，再利用 SEND 鈕 (圖 2.4ⓑ) 進行發送。對話區 (Conversation Area; 圖 2.4ⓒ) 記載使用者已收發的訊息記錄。每筆記錄包含訊息的所有者、傳送時間與內容。

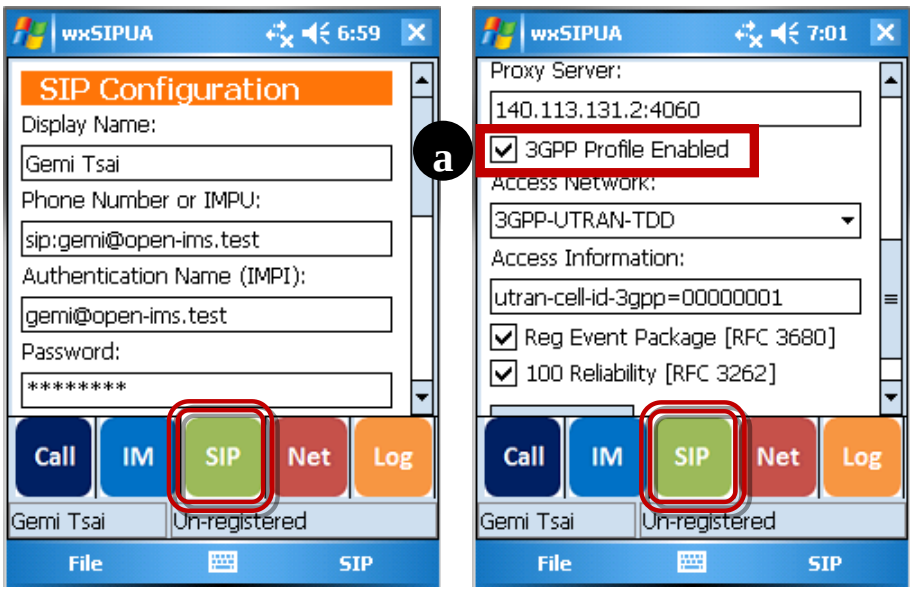

圖 2.5: SIP/IMS 設定頁面

SIP/IMS 設定頁面 (圖 2.5) 提供 SIP 服務標準的參數設定，包括：

- **Display Name**：使用者進行服務時所顯示的名稱。
- **Phone Number or IMPU**：通話服務時所用的電話號碼，也可使用 IMPU (*IP Multimedia Public Identity*) 的格式，如 “sip:12345@host.com”。
- **Authentication Name (IMPI)**：使用者進行身份認證所用的名稱，在 IMS 服務之下稱為 IMPI (*IP Multimedia Private Identity*)。
- **Password**：使用者進行身份認證所用的密碼。
- **Domain**：本籍網路的網域名稱。
- **3GPP Profile Enabled**：開啟或關閉 3GPP IMS 服務標準的支援。
- **Proxy Server**：代理伺服器位置。當啟用 3GPP Profile Enabled 選項 (圖 2.5ⓐ) 時，此值將代表 P-CSCF 的位址；當關閉 3GPP Profile Enabled 選項時，此值則代表 SIP Proxy 的位址。
- **Access Network** 與 **Access Information**：UE 目前連結網路的資訊。此參數只在 IMS 服務架構下 (啟用 3GPP Profile Enabled 選項) 才可進行設定。
- **Reg Event Package**：是否啟用註冊事件訂閱 [13] 功能。
- **100 Reliability**：是否啟用 SIP 臨時回覆 (SIP 101~199 訊息) 可靠度 [14] 機制。當 UA 雙方都開啟此機制，一旦 UA 收到 180 Ringing 或 183 Session Progress 訊息時，將會立即以 PRACK 訊息來回應此訊息已被接收。

網路設定頁面 (圖 2.6) 用於調整網路相關的設定參數，包括：

- 網路位址 (圖 2.6ⓐ)：選擇 UA 網路傳輸所用的 IP 位址、通訊埠 (port) 與 IP 版本 (IPv4 或 IPv6)。
- 資料傳輸的協定類型 (圖 2.6ⓑ)：選擇資料傳輸的協定，包括 UDP 即時傳輸、TCP 可靠連結傳輸、與 TLS 安全連結傳輸。

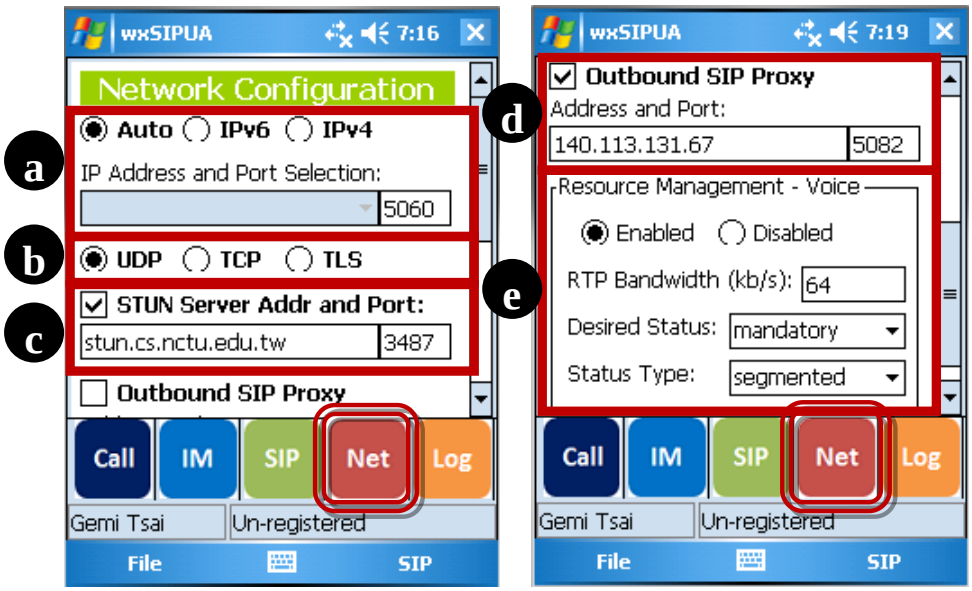

圖 2.6: 網路設定頁面

- STUN 啟用與相關參數設定 (圖 2.6ⓒ)：是否使用 STUN (*Simple Traversal of UDP Through NATs*) [15] 解決方案來穿越 NAT。啟用時需輸入遠端 STUN 伺服器之位址與通訊埠，以便向 STUN 伺服器取得 UA 穿越 NAT 所需的資訊。詳細說明參見第 2.3.1 節。
- Outbound SIP Proxy 啟用與相關參數設定 (圖 2.6ⓓ)：是否使用 Outbound SIP Proxy 來協助 SIP 訊息穿越 NAT。詳細說明參見第 2.3.2 節。
- QoS 資源管理參數 (圖 2.6ⓔ)：設定 QoS 相關參數 [16]，第一個參數為保留頻寬大小；第二個參數為對 QoS 的要求 (desired)，可選擇無 (none)、非強制 (optional) 或強制 (mandatory)；第三個參數為資源保留的進行狀態，可要求 UA 雙方以端到端 (end-to-end) 或分為本地與遠端兩段 (segmented) 來進行。

訊息日誌頁面 (圖 2.7) 記載 UA 執行時所產生的歷史訊息，每筆記錄包含訊息發生的時間與內容。使用者可利用歷史訊息來除錯或檢查執行狀態。

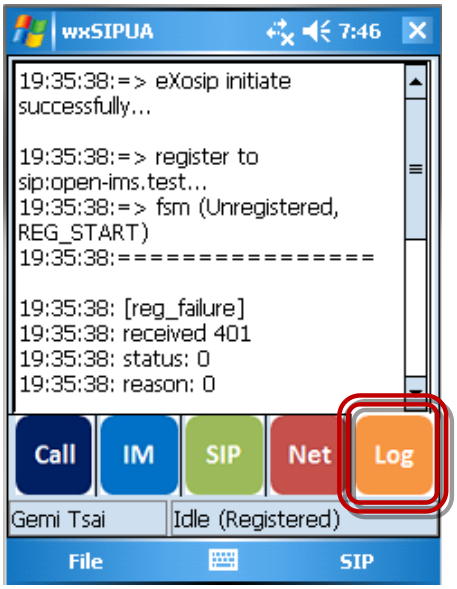

圖 2.7: 訊息日誌頁面

2.2 SIP/IMS Core Module

SIP/IMS Core Module (圖 2.1ⓒ) 為 UA 收送與處理 SIP 訊息的核心模組，負責 SIP 初始化、eXosip [17]、oSIP [18] 與 OpenSSL [19] 的函式包裝 (encapsulation)、SIP 事件的取得與處理、以及 RTP Core Module 的初始化。

2.2.1 SIP 初始化

在 SIP 帳號註冊前，SIP/IMS Core Module 會先進行 SIP 初始化，如圖 2.8 所示。圖 2.8①呼叫 eXosip_init 函式來初始化 eXosip 函式庫。此 eXosip 函式庫包裝了 oSIP 與 OpenSSL 函式庫，用於隱藏 oSIP 與 OpenSSL 複雜的 SIP 訊息建置與處理步驟，以提供簡易的函式介面讓 SIP/IMS Core Module 使用。接著在圖 2.8②中的 switch 語法，會依據使用者在圖 2.6 所選擇的傳輸協定，將本地端的 IP 位址與通訊埠等參數，傳入 eXosip_listen_addr 函式 (圖 2.8③④⑥) 來設定本地端用於接收 SIP 訊息的網路位址。若使用者選擇 TLS 做為傳輸的方式，則必須再使用 eXosip_set_tls_ctx 函式 (圖 2.8⑤) 來設

```
int wxSIPCore::SipInit (const char *addr, int port, int family, int transport) {
    ... // declaration
①   if (eXosip_init () != OSIP_SUCCESS)
        return -1;
    ... // set other options
②   switch (transport)  {
        case TRANSPORT_UDP:
③           i = eXosip_listen_addr (IPPROTO_UDP, addr, port, family, 0);
            break;
        case TRANSPORT_TCP:
④           i = eXosip_listen_addr (IPPROTO_TCP, addr, port, family, 0);
            break;
        case TRANSPORT_TLS:
            eXosip_tls_ctx_t ctx;
            memset (&ctx, 0, sizeof (eXosip_tls_ctx_t));
            sprintf (ctx.server.priv_key, "%s", SERVER_PRIV_KEY);
            sprintf (ctx.server.priv_key_pw, "%s", SERVER_PASSWD);
            sprintf (ctx.server.cert, "%s", SERVER_CERT);
            sprintf (ctx.client.priv_key, "%s", CLIENT_PRIV_KEY);
            sprintf (ctx.client.priv_key_pw, "%s", CLIENT_PASSWD);
            sprintf (ctx.client.cert, "%s", CLIENT_CERT);
            sprintf (ctx.root_ca_cert, "%s", ROOT_CA_CERT);
⑤           if ((i = eXosip_set_tls_ctx (&ctx)) != TLS_OK)
                break;
⑥           i = eXosip_listen_addr (IPPROTO_TCP, addr, port, family, 1);
            break;
    }
    if (i != 0) {
        eXosip_quit ();
        return i;
    }
    return 0;
}
```

圖 2.8: wxSIPCore::SipInit 函式

定私密金鑰 (private key)、憑證 (certificate) 與認證機構 (certificate authority) 憑證，以提供 SIP 訊息加解密與身份認證使用。

2.2.2 eXosip 與 oSIP 函式的包裝

在本論文的實作中，SIP/IMS Core Module 扮演 eXosip 及 oSIP 函式庫包裝者 (wrapper) 角色。SIP/IMS Core Module 會將通話功能所需的 eXosip 及 oSIP 相關函式作一包裝，來提供簡化的函式給 User Interface Module 使用。例如 SIP/IMS Core Module 提供的 wxSIPCore::Invite 函式被用來產生與發送 INVITE 訊息。參見圖 2.9，說明如下。

在 wxSIPCore::Invite 函式中，圖 2.9①宣告一個 SIP 訊息結構 (osip_message_t) 指標 *invite*，接著圖 2.9②利用 eXosip_call_build_initial_invite 函式建置一個初始化的 INVITE 訊息結構，並將記憶體位址指派給 *invite*。在初始化之後，雖然 INVITE 訊息結構已具備基本的 SIP 標頭 (header field)，但為了支持延伸的規範，須再透過 oSIP 函式加入額外的 SIP 標頭。圖 2.9③呼叫 osip_message_set_route 函式來設定 *Route* 標頭 [2]，此標頭用於指定 SIP 訊息將經過的代理伺服器，如 IMS 核心內部的 P-CSCF 與 S-CSCF。圖 2.9④-⑥利用 osip_message_set_header 函式建置 IMS 所需的標頭 [20]，包括 *P-Access-Network-Info* [21]、*Privacy* [22, 23] 與 *P-Preferred-Identity* [23]。在 IMS 核心網路中，UA 會以 *P-Access-Network-Info* 標頭記載目前連結網路的資訊。*Privacy* 標頭用於指定要隱私的資訊，如 Caller ID。當使用者擁有多個 Public User Identity 時，*P-Preferred-Identity* 標頭用於告知核心網路，欲使用那個 Public User Identity 來建立此語音會期 (session)。圖 2.9⑦的 Proc_LocalSdpInfo 函式用於產生 SDP (*Session Description Protocol*) [24] 內容。最後，圖 2.9⑧的 eXosip_call_send_initial_invite 函式會依照指標 *invite* 所指向的 osip_message_t 結構內容，開始發送 INVITE 訊息。

```
int wxSIPCore::Invite (…) {
①   osip_message_t *invite = NULL;
    ... // other declaration

    eXosip_lock ();
②   i = eXosip_call_build_initial_invite (&invite, toSipUri, fromSipUri, proxy, NULL);
    eXosip_unlock ();
    if (i != OSIP_SUCCESS)    return i;

    ...// set outbound sip proxy

    for (j = 0; j < RouteList->count; j ++)
③       osip_message_set_route (invite, RouteList->Item[j]);

    if (isIMS) {
④       osip_message_set_header (invite, "P-Access-Network-Info", PAccessNwkInfo);
⑤       osip_message_set_header (invite, "Privacy", "none");
⑥       osip_message_set_header (invite, "P-Preferred-Identity", fromSipUri);
    }
    ... // set other additional SIP headers

⑦   Proc_LocalSdpInfo (invite, NULL);

    eXosip_lock ();
⑧   i = eXosip_call_send_initial_invite (invite);
    eXosip_unlock ();
    if (i < 0)   return i;

    PostStateEvent (SEND_INVITE);
    return 0;
}
```

圖 2.9: wxSIPCore::Invite 函式

2.2.3 SIP 事件的取得與處理

eXosip 底層在接收到 SIP 訊息後，會將 SIP 訊息以事件 (簡稱為 SIP 事件) 的形式放到 FIFO (First In, First Out) 佇列。由於取出 SIP 事件之函式 (eXosip_event_get) 為一不會立刻回傳 (blocking) 的函式，所以必須等待此函式處理完畢才可繼續下個程序。為了使 UA 能即時回應使用者的操作，本實作設計 SIP/IMS Core Module 以 wxSIPEvtThread 執行緒處理 SIP 事件的取得，並依照事件種類進行不同的處理。

```
void* wxSIPEvtThread::Entry () {
      eXosip_event_t *event = NULL;

①    while ((event = eXosip_event_get ()) != NULL) {
②          switch (event->type) {
③          case EXOSIP_CALL_RINGING: // received 18x response
④                eXosip_Call_Ringing_Handler (event);
                 break;
⑤          case EXOSIP_CALL_ANSWERED:    // received 200 OK
⑥                eXosip_Call_Answered_Handler (event);
                 break;
           ... // other events
           }
⑦          eXosip_event_free (event);
      }
      return 0;
}
```

圖 2.10: wxSIPEvtThread::Entry 函式

wxSIPEvtThread::Entry 函式會將 SIP 事件取出並分送 (dispatch) 給負責處理的函式，參見圖 2.10。其中，eXosip_event_get 函式 (圖 2.10①) 負責從 eXosip 維繫的 FIFO 佇列取出 SIP 事件，並由圖 2.10②的 switch 語法，依不同事件呼叫不同的函式。例如，

若事件種類為 EXOSIP_CALL_RINGING (圖 2.10③)，表示收到對方所發出的 180 RINGING 或 183 Session Progress 訊息，此時將交由 eXosip_Call_Ringing_Handler 函式 (圖 2.10④) 來進行後續處理。同理，若取得事件之種類為 EXOSIP_CALL_ANSWERED (圖 2.10⑤)，表示 SIP/IMS Core Module 接收到 INVITE 訊息所對應的 200 OK 回覆，代表受話方已接受通話請求。在此情形下，程式會以 eXosip_Call_Answered_Handler 函式 (圖 2.10⑥) 來回應 ACK 訊息，並依照 200 OK 內的 SDP 資訊建立 RTP (*Real-time Transport Protocol*) [25] 語音會期。當事件處理完畢後，圖 2.10⑦的 eXosip_event_free 函式會負責釋放 SIP 事件所使用的記憶體。本論文接著以振鈴為例，說明事件處理函式的運作。參見圖 2.10④之 eXosip_Call_Ringing_Handler 函式對 EXOSIP_CALL_RINGING 事件的處理。此函式內容如圖 2.11 所示。

在圖 2.11 函式中，首先程式會檢查此訊息的 *CSeq* 標頭與內容 (圖 2.11①)，以避免重覆處理相同的訊息。接著 sipCore->Proc_RemoteSdpInfo 函式 (圖 2.11②) 處理訊息中夾帶的 SDP 資訊。在 SDP 資訊處理完畢後，程式將接著檢查 *Require* 標頭是否帶有 *100rel* 內容，以及 UA 是否啟用此功能 (圖 2.11③)，若有則必須回傳 PRACK 訊息 [14]。eXosip_call_build_prack 函式 (圖 2.11④) 被用來建立一個基本的 PRACK 訊息結構。每當建立新訊息時，UA 須檢查雙方語音會期能力是否協商完成，若未完成則必須再以 SDP 繼續進行協商，這部份的工作是由 sipCore->Proc_LocalSdpInfo 函式 (圖 2.11⑤) 負責。當 PRACK 訊息準備完成後，程式會以 eXosip_call_send_prack 函式 (圖 2.11⑥) 來發送此訊息。最後，若接收訊息為 180 Ringing，則程式會發佈振鈴 (alerting) 事件 (圖 2.11⑦) 至 User Interface Module 要求產生振鈴。

```
int wxSIPEvtThread::eXosip_Call_Ringing_Handler (eXosip_event_t *event) {
      ... // other declaration

①    if (CheckCSeq (event) != 0)   return -1;

②    sipCore->Proc_RemoteSdpInfo (event);

③    if (IsRequired100rel (event)) {
           osip_message_t *prack;
           eXosip_lock ();
④          eXosip_call_build_prack (event->tid, &prack);

           if (isIMS) {
           osip_message_set_header (prack, "P-Access-Network-Info", PAccessNwkInfo);
           }
           ... // set other additional SIP headers

⑤          sipCore->Proc_LocalSdpInfo (prack, event);

⑥          eXosip_call_send_prack (event->tid, prack);
           eXosip_unlock ();
      }

      if (event->response->status_code == 180)
⑦          PostEvent (ID_SIP_ALERTING);
      return 0;
}
```

圖 2.11: wxSIPEvtThread:: eXosip_Call_Ringing_Handler 函式

2.2.4 RTP Core Module 之初始化

當建立通話的 SIP 信令程序完成後，SIP/IMS Core Module 會開始初始化 RTP Core Module，並通知 RTP Core Module 建立 RTP 語音會期。本節以 SIP/IMS Core Module 收到對方回覆通話請求的 200 OK 訊息所呼叫的處理函式 eXosip_Call_Answered_Handler 為例

來說明，程式內容如圖 2.12 所示。

```
int wxSIPEvtThread::eXosip_Call_Answered_Handler (eXosip_event_t *event) {
      ... // declaration and find the dialog

①    eXosip_call_send_ack (event->did, ack);
②    sipCore->Proc_RemoteSdpInfo (event);
③    retVal = sipCore->CreateRTPSession (sipCore ->sdpInfo->c_addrtype,
                  sipCore ->sdpInfo->c_addr, atoi (sipCore ->mediaInfo->m_port));
      if (retVal < 0) {
            ... // error handling
      }
      PostStateEvent (SEND_ACK);
      PostUiEvent (UI_SIPINVITECTRL);
      return 0;
}
```

圖 2.12: wxSIPEvtThread:: eXosip_Call_Answered_Handler 函式

圖 2.12 的 eXosip_Call_Answered_Handler 函式中，程式會先以 eXosip_call_send_ack 函式 (圖 2.12①) 來回應 ACK 訊息，並利用 sipCore->Proc_RemoteSdpInfo 函式 (圖 2.12②) 來處理 200 OK 訊息所夾帶的 SDP 資訊。接著，SIP/IMS Core Module 會將處理後的 SDP 內容傳入 sipCore->CreateRTPSession 函式 (圖 2.12③)，以進行 RTP Core Module 的初始化，並通知 RTP Core Module 依據 SDP 內容建立 RTP 語音會期。

2.3 NAT Traversal Module

NAT Traversal Module (圖 2.1ⓑ) 提供 STUN (*Simple Traversal of UDP Through NATs*) [15] 與 SBC (*Session Border Controller*) [26] 兩種穿越 NAT 的解決方案，來解決 SIP 與 RTP 封包穿越 NAT 的問題。

2.3.1 STUN 解決方案

```
int wxNatTraversal::MasqueradeByStun (char *addr, int *port) {
      StunAddress4 stunServer;
      ... // other declarations

①    if (stunParseServerName (stunServerAddr, stunServer) == false)
            return -1;
②    stunServer.port = stunServerPort;

③    type = stunNatType (stunServer, false, false, false, *port, &nic);
      if (type == StunTypeBlocked || type == StunTypeUnknown || type == StunTypeFailure)
            return -1;
④    sprintf (addr, "%d.%d.%d.%d", (nic.addr >> 24) & 0xFF, (nic.addr >> 16) & 0xFF,
                        (nic.addr >> 8) & 0xFF, nic.addr & 0xFF);
⑤    *port = nic.port;
      return 0;
}
```

圖 2.13: wxNatTraversal::MasqueradeByStun 函式

STUN 解決方案用於協助 SIP/IMS Core Module 修改 SIP 與 RTP 的聯繫資訊 (SIP 訊息中的 *Contact* 與 *Via* 標頭，以及 SDP 的 *c* 與 *m* 欄位)，將訊息中 UA 於私有 (private) 網路下的 IP 位址與通訊埠，替換成 NAT 轉換後的公眾 (public) 網路 IP 位址與通訊埠。本論文使用 wxNatTraversal::MasqueradeByStun 函式 (圖 2.13) 來包裝 STUN 函式庫 [27] 以提供上述功能。在此函式中，圖 2.13①②負責將圖 2.6ⓒ所輸入的 STUN 伺服器 IP 位址與通訊埠轉換為 STUN 函式庫所用的格式。接著以 STUN 函式庫提供的 stunNatType 函式 (圖 2.13③)，來取得 UA 於 NAT 轉換後的公眾 IP 位址與通訊埠。最後，再由圖 2.13④⑤程式碼，替換原有的私有 IP 位址與通訊埠。

2.3.2 SBC 解決方案

SBC 解決方案利用了外部 Outbound SIP Proxy 與 RTP Proxy [28]，讓 UA 透過這些伺服器來轉送封包，以解決 SIP/RTP 穿越 NAT 的問題。在 SIP 方面，當使用者啟用圖 2.6ⓓ的 SBC 解決方案，UA 會將所有 SIP 訊息送往 Outbound SIP Proxy，交由 Outbound SIP Proxy 來轉換 IP 位址與轉送。此功能的實作是由 wxSIPCore::SetOutboundSipProxy 函式負責，程式內容如圖 2.14 所示。

```
int wxSIPCore::SetOutboundSipProxy (osip_message_t *msg, char *obSipProxy) {
        osip_route_t *route = NULL;
①       osip_route_init (&route);
②       if (osip_route_parse (route, obSipProxy) < 0) {
                osip_route_free (route);
                return -1;
        }
③       osip_uri_uparam_add (route->url, osip_strdup ("lr"), NULL);
④       osip_route_to_str (route, &obSipProxy);
⑤       osip_message_set_route (msg, obSipProxy);
        osip_route_free (route);

        return 0;
}
```

圖 2.14: wxSIPCore::SetOutboundSipProxy 函式

圖 2.14 的 wxSIPCore::SetOutboundSipProxy 函式會將 Outbound SIP Proxy 位址加入至 SIP 訊息路由的第一筆。在此函式中，首先圖 2.14①-④會透過 oSIP 提供的函式集，對此筆路由加入 *lr* 參數 [2]。接著再利用 osip_message_set_route 函式 (圖 2.14⑤)，將此筆路由加入至 SIP 訊息中。

在 RTP 方面，當通話雙方 UA 協商 RTP 語音會期時，外部的 RTP Proxy 會從中修改 SDP 內容，將用於接收 RTP 封包的 IP 位址 *c* 欄位與通訊埠 *m* 欄位，改為由 RTP Proxy 來負責接收 RTP 封包。如此一來，原本通話雙方以點對點 (peer-to-peer) 方式傳輸 RTP 封包，變為透過 RTP Proxy 來轉送。

2.4 其他模組簡介

此小節簡介延用原 UA 軟體的三個模組。在本論文的實作中，這些模組的運作方式並沒有改變或被新功能影響。各模組說明如下：

RTP Core Module (圖 2.1ⓓ) 為 UA 軟體用於收送 RTP 封包的核心模組。RTP Core Module 會將 JRTPLIB [29] 提供之函式包裝成支援 IPv4 與 IPv6 的 RTP 封包收發函式。使 SIP/IMS Core Module 能使用上述函式，進行 RTP Core Module 的初始化，及依照 SDP 中的連線資訊設定 RTP Core Module 參數。

Multimedia Control Module (圖 2.1ⓔ) 負責控制音效裝置的錄音與播音。此模組將從音效裝置錄製且編碼後的語音資料送至 RTP Core Module 包裝成 RTP 封包，並由 RTP Core Module 傳送至收話端。另一方面，當 RTP Core Module 收到 RTP 封包後，會交由此模組來進行解碼與控制音效裝置播音。

IP Helper Control Module (圖 2.1ⓕ) 將 IP Helper API 作一包裝，讓 User Interface Module 透過此模組來取得設備上所有 IP 位址，以供使用者選擇使用。

第三章　SIP 通話建立信令流程

本章節說明 3GPP IMS 與 IETF SIP 網路架構中，主要的幾種 SIP 通話建立信令流程。其中第 3.1 節並敘述了撥號後延遲 (*Post Dialing Delay*; PDD) [30] 的定義。本章節將由基本通話建立開始介紹，向後延伸至幾種不同的功能程序。為了著重於 UA 訊息收送與 PDD 的說明，流程中簡化了核心網路內部伺服器間訊息交換的敘述。

3.1　基本通話建立與 PDD

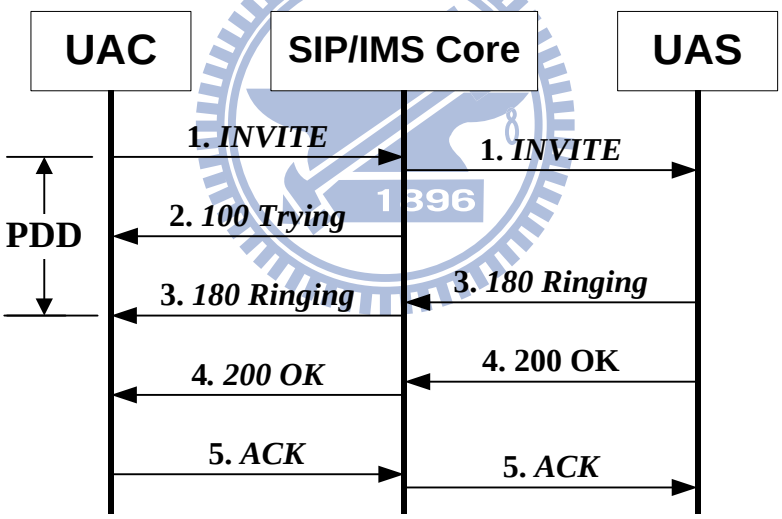

圖 3.1: 基本通話建立訊息流程

SIP 通話建立是由發話端 (originator) 與受話端 (terminator) 雙方之 UA 共同完成。發話端 UA 扮演 UAC (*User Agent Client*) 角色，負責向受話端發送通話建立請求，並由受話端 UA 扮演 UAS (*User Agent Server*) 角色，用以回覆 UAC 之請求。SIP 基本通話建立訊息流程如圖 3.1 所示，包含以下步驟：

步驟 1. UAC 透過 INVITE 訊息向 UAS 發送通話建立請求，並由核心網路轉送至 UAS。此訊息包含初始語音會期的 SDP 資訊，用以向 UAS 提供 UAC 的語音會期處理能力。

步驟 2. 當核心網路接收到 INVITE 訊息時，會立即回傳 100 Trying 來指示已接收 INVITE 訊息，避免 UAC 因等待回覆過久而重送訊息。

步驟 3. 在 UAS 收到 INVITE 訊息後，即開始振鈴通報使用者，並回傳 180 Ringing 訊息至 UAC。另一方面，當 UAC 收到 180 Ringing 訊息時，會立即產生回鈴音 (ringback tone) 提示使用者。

步驟 4. 若 UAS 使用者接受此通話請求，UAS 將透過 200 OK 訊息來告知 UAC。此訊息夾帶了 SDP 內容，用以答覆 UAS 所接受的媒體能力。

步驟 5. 在 UAC 收到 200 OK 訊息後，會立即傳送 ACK 訊息向 UAS 做最後確認回應。而 UAS 收到此 ACK 訊息時，表示通話建立已完成，UA 雙方可開始啟動語音會期進行通話。

圖 3.1 中，從 UAC 發送 INVITE 訊息 (步驟 1) 至收到 180 Ringing 回覆 (步驟 3) 之間所等待的時間，稱為撥號後延遲 (*Post Dialing Delay*; PDD)。PDD 是一種測量通話建立效能的指標 [30]，其定義為發話端從撥號結束開始，至聽到回鈴音或忙線音 (busy tone) 所經過的時間。本論文第四章將藉由 PDD 的測量來評估不同實驗環境下通話建立效能的表現。

3.2 使用 TCP 傳輸協定建立通話

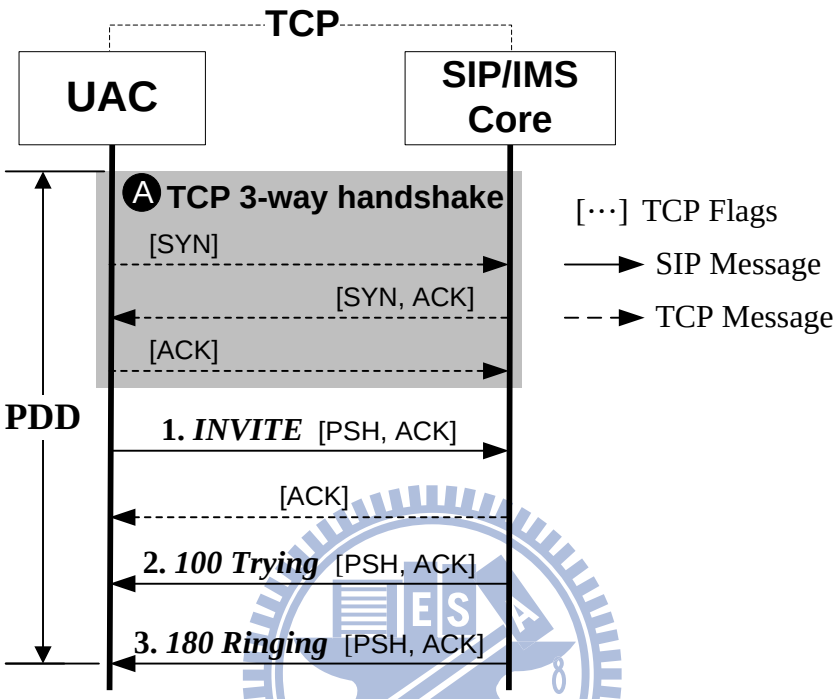

圖 3.2: 通話建立流程 – UAC 與核心網路間使用 TCP 傳輸協定

SIP 可透過 UDP 或 TCP 協定來傳輸訊息。UDP 為 SIP 基本的傳輸協定，是一種非連線導向與非可靠型的傳輸方式，提供較簡易與快速的封包收送。而 TCP 是一種連線導向 (connection oriented) 的可靠型傳輸方式。TCP 在傳送訊息前，必須先以三向交握 (3-way handshake) 來建立收發兩端的連結，並在連線建立完成後才可開始收送訊息。接收端也須回應 (acknowledgement) 訊息的接收，來確保傳輸的可靠性。圖 3.2 展示了 UAC 與核心網路之間使用 TCP 協定傳輸的通話建立流程。此流程中，PDD 增加了 TCP 三向交握 (圖 3.2Ⓐ) 與回應訊息接收所等待的時間。

3.3 具資源管理機制之通話建立流程

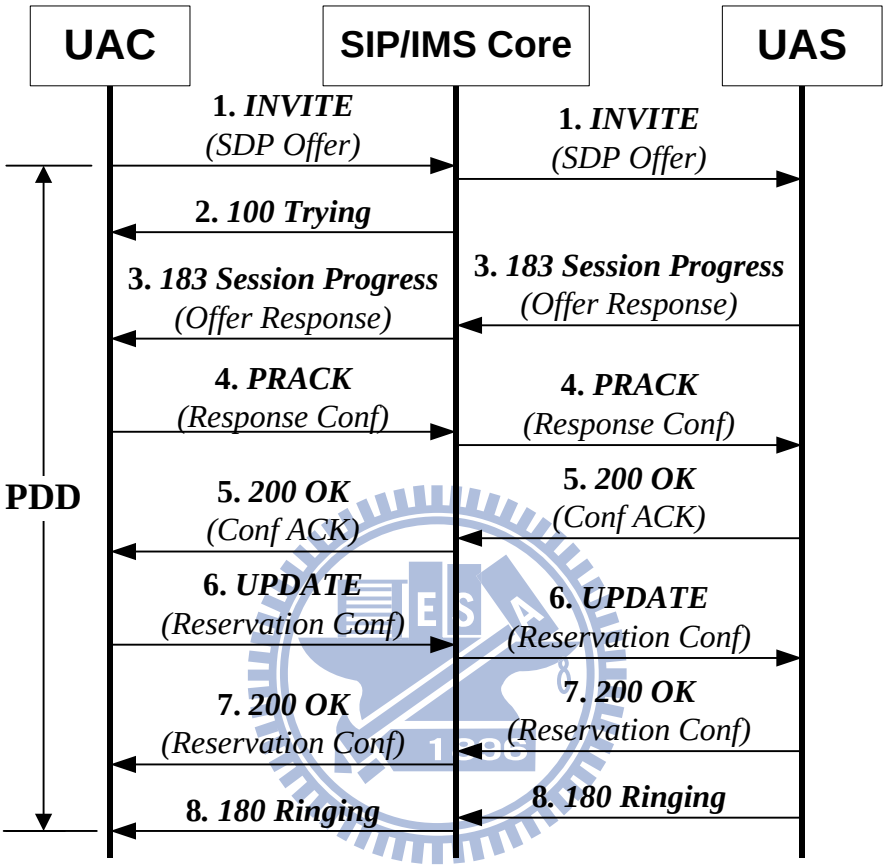

圖 3.3: 通話建立流程 – 具資源管理機制

基本 SIP 通話建立程序由於沒有考慮目前網路資源狀況，所以在行動通訊網路下，可能發生通話建立雖然完成，卻因資源不足而無法順利傳送語音封包的情況。3GPP 為了改善上述問題，在通話建立程序中規範了資源管理機制 [16, 20, 31]，用以確保通話雙方擁有足夠的網路資源來傳輸語音封包，其訊息流程如圖 3.3 所示，步驟如下：

步驟 1. 與圖 3.1 之步驟 1 相似，但在 INVITE 訊息的 SDP 敘述中，UAC 必須再提供自己目前的資源狀態與期許的資源需求給 UAS。

步驟 2. 當核心網路收到 INVITE 訊息時，會立即回傳 100 Trying 來指示已接收 INVITE 訊息，避免 UAC 因等待回覆過久而重送訊息。

步驟 3. 在 UAS 收到 INVITE 訊息後，會利用 183 Session Progress 訊息來回覆自己目前的資源狀態與期許的資源需求給 UAC，並指示 UAC 在完成資源保留時，必須向 UAS 發出確認通知。

步驟 4. UAC 從 183 Session Progress 訊息中得到 UAS 對媒體能力需求的答覆，此時 UAC 開始與核心網路進行資源保留。在資源管理機制下，為了確保訊息的可靠度，當 UAC 收到 183 Session Progress 訊息時，必須回傳 PRACK 訊息，用於回覆接收確認。

步驟 5. 若 UAS 收到 PRACK 訊息，會發送 200 OK 訊息做為確認回應。同時 UAS 開始與核心網路進行資源保留。

步驟 6-7. 當 UAC 完成資源保留時，會立即透過 UPDATE 訊息發送資源保留確認 (reservation confirmation) 至 UAS。另一方面，UAS 接收 UPDATE 訊息後，會以 200 OK 訊息回應 UPDATE 訊息。

步驟 8. 若 UAS 完成資源保留，並且已收到 UAC 完成資源保留的確認，此時 UAS 即可振鈴通報使用者，同時傳送 180 Ringing 訊息至 UAC。UAC 在收到 180 Ringing 訊息後，將產生回鈴音提示使用者。

在資源管理機制下，UA 於通話建立時會協商 QoS 資訊，因而增加圖 3.3 步驟 3 至 7 的程序與 QoS 協商所需的 SDP 內容。UA 雙方並會在此過程中進行網路資源的保留。PDD 也因此多出上述步驟所產生的等待時間。

3.4 具認證機制之通話建立流程

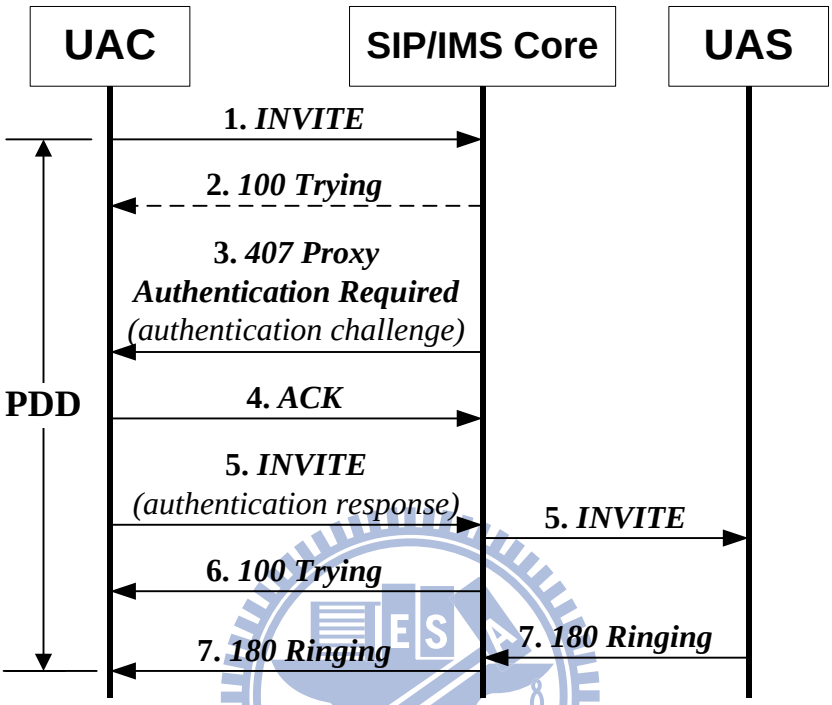

圖 3.4: 通話建立流程 — 啟用認證機制

認證 (authentication) 機制能讓核心網路在處理 UAC 請求前，先依安全需求對 UAC 進行認證，防止他人的偽裝。SIP 認證是一種查問式 (challenge-based) 機制 [2]。認證的方式中，IETF SIP 網路架構使用 HTTP Digest [32] 為主要的認證方式。而 3GPP IMS 則採用安全性較高的 IMS AKA (*Authentication and Key Agreement*) [20, 33, 34] 為主。核心網路為了加強認證的安全性，除了註冊時要求 UAC 進行認證外，在 UAC 建立通話前也會再次要求認證。啟用認證機制的通話建立訊息流程如圖 3.4 所示，步驟如下：

步驟 1. UAC 對 UAS 發起通話建立請求，傳送 INVITE 訊息至核心網路。

步驟 2. 若核心網路為 IMS，則 IMS 核心會由 UAC 所指派的 P-CSCF 來轉送 INVITE 訊息給服務 UAC 的 S-CSCF。接著由 S-CSCF 來進行認證查問 (authentication

challenge)。在 P-CSCF 等待 S-CSCF 回覆時，必須先回應 100 Trying 訊息給 UAC，避免 UAC 因等待回應過久而重送 INVITE 訊息。

步驟 3. 核心網路會以 407 Proxy Authentication Required 訊息夾帶 realm (如 host.com)、algorithm (如 MD5) 與 nonce 等參數 [32]，傳送至 UAC 進行認證查問。

步驟 4. UAC 在收到認證查問的 407 Proxy Authentication Required 訊息後，會隨即發送 ACK 訊息來結束此次通話請求。

步驟 5. UAC 將所接收的認證查問參數，依據 [32] 的推算方式求出認證回覆值 (authentication response)，並利用新的 INVITE 訊息將此值傳送至核心網路進行驗證。若 UAC 通過核心網路的驗證，則核心網路將繼續進行通話建立程序，轉送此 INVITE 訊息至目的端。後續的步驟與基本通話建立程序相同。除此之外，基於安全考量，核心網路在轉送此 INVITE 訊息前，可能會移除訊息中認證相關的資料再進行傳送。

步驟 6, 7. 與圖 3.1 步驟 2 與 3 相同。

在上述認證機制中，通話建立程序將多出認證 UAC 身份所用的步驟 2 至 5 訊息 (100 Trying、407 Proxy Authentication Required、ACK 與 INVITE)。而 PDD 也增加了收送此四個訊息與驗證身份所等待的時間。

3.5 啟用 TLS 安全機制之通話建立流程

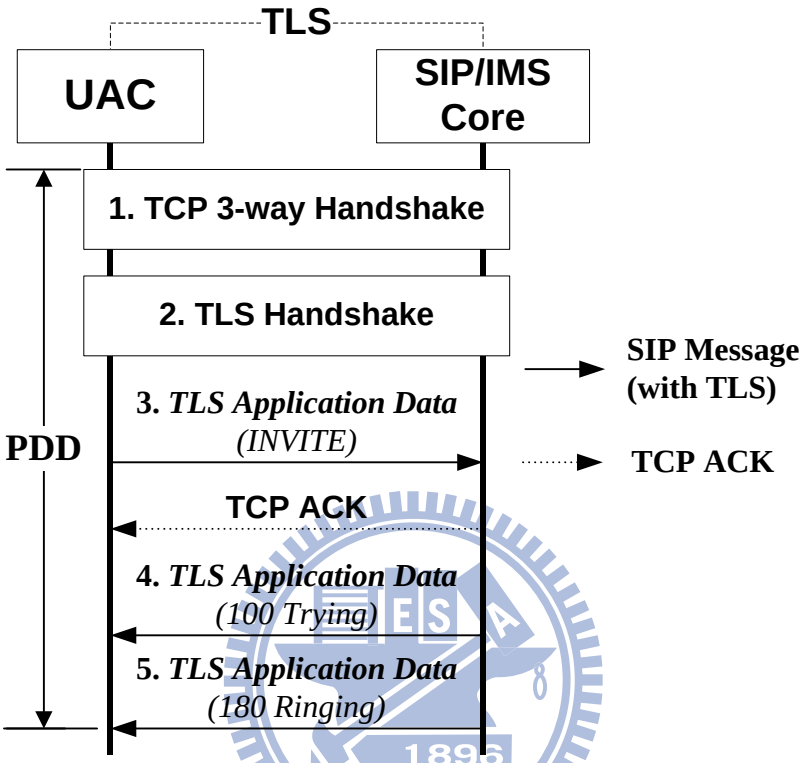

圖 3.5: 通話建立流程 – 啟用 TLS 安全機制

TLS 協定提供資料完整性 (integrity) 與機密性 (confidentiality) 的保護，以及收送雙方身份真實性的認證。IETF 之 RFC 3261 [2] 將 TLS 定為 SIP 主要的安全機制，3GPP 也將 TLS 納入 IMS 規範 [20] 中，並可與 IPSec (*Internet Protocol Security*) [35] 安全機制共同運作。圖 3.5 說明 UAC 與核心網路之間使用 TLS 安全機制進行傳輸的通話建立流程。由於 TLS 建置於 TCP 連線之上，所以 UAC 與核心網路必須先以 TCP 三向交握建立兩端的連結 (圖 3.5 步驟 1)，接著才使用 TLS 交握程序來建立安全連線 (圖 3.5 步驟 2)。

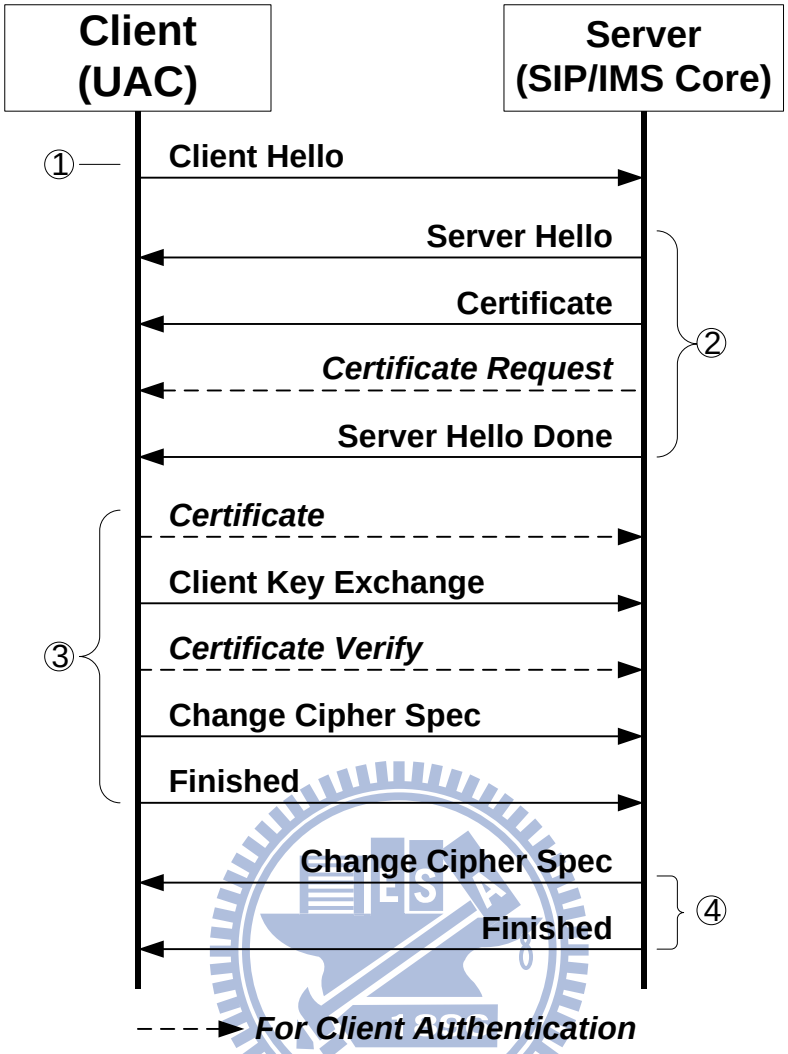

圖 3.6: TLS 交握程序

TLS 交握程序包含了客戶端與伺服器兩種角色，由發起安全連線的一方扮演客戶端，另一方則扮演伺服器，負責回應客戶端的請求。客戶端與伺服器可在 TLS 交握程序中協商加解密所用的演算法與金鑰，並且認證對方身份的真實性。TLS 共有「只認證伺服器身份」及「伺服器與客戶端相互認證」兩種身份認證方式。包含身份認證的 TLS 交握程序如圖 3.6 所示，步驟如下：

步驟 1. 客戶端傳送 Client Hello 訊息給伺服器，此訊息用於告知伺服器，客戶端所支援的 TLS 版本號碼、加密機制 (cipher suite) 與資料壓縮方法。

步驟 2. 伺服器由客戶端的 Client Hello 訊息中選擇接受的參數，並且以 Server Hello 訊息向客戶端回覆。Certificate 訊息用於傳送伺服器公開金鑰 (public key) 與認證機構之憑證，以提供客戶端認證伺服器身份的真實性。若伺服器也要求客戶端進行身份認證，會在傳送 Certificate 訊息之後，再發送 Certificate Request 訊息告知客戶端身份認證的請求。最後伺服器以 Server Hello Done 訊息來結束此初步協商。

步驟 3. 若伺服器傳送至客戶端的訊息中，含有要求客戶端身份認證的 Certificate Request 訊息，客戶端將利用 Certificate 訊息來傳送自己的公開金鑰與認證機構憑證給伺服器。Client Key Exchange 訊息用來告知伺服器後續通訊所用的金鑰資訊，此訊息會使用伺服器的公開金鑰來加密傳送，以確認伺服器公開金鑰正確性。若客戶端先前已傳送 Certificate 訊息，此時會再以 Certificate Verify 訊息向伺服器證實其公開金鑰與私密金鑰為自己所擁有。Change Cipher Spec 訊息用於啟動雙方所協商的安全服務。最後客戶端再以加密的 Finished 訊息，告知伺服器完成協商。

步驟 4. 伺服器利用 Change Cipher Spec 訊息，告知客戶端開始啟動安全服務，最後傳送加密的 Finished 訊息，告知客戶端完成協商。

在 TLS 交握過程中，若採取「伺服器與客戶端相互認證」的方式，伺服器需要多傳送一個 Certificate Request 訊息向客戶端要請求認證，客戶端也需多傳送 Certificate 與 Certificate Verify 訊息給伺服器。當 UAC 與核心網路之間的安全連線建立完成，後續訊息將會由發送方加密並包裝成 TLS Application Data 訊息傳送，再由接收方負責解密。如圖 3.5 步驟 3 至 5，其 TLS Application Data 訊息分別包裝了 SIP 之 INVITE、100 Trying 與 180 Ringing 訊息加密後的結果。UAC 在 TLS 安全機制下，PDD 會多出 TCP 三向交握、TLS 交握以及上述步驟 3 至 5 訊息加解密所使用的時間。

第四章　實驗環境與結果

本章節使用本論文所實作的 SIP/IMS UA 軟體，測量第三章所描述的五種通話建立程序 PDD。

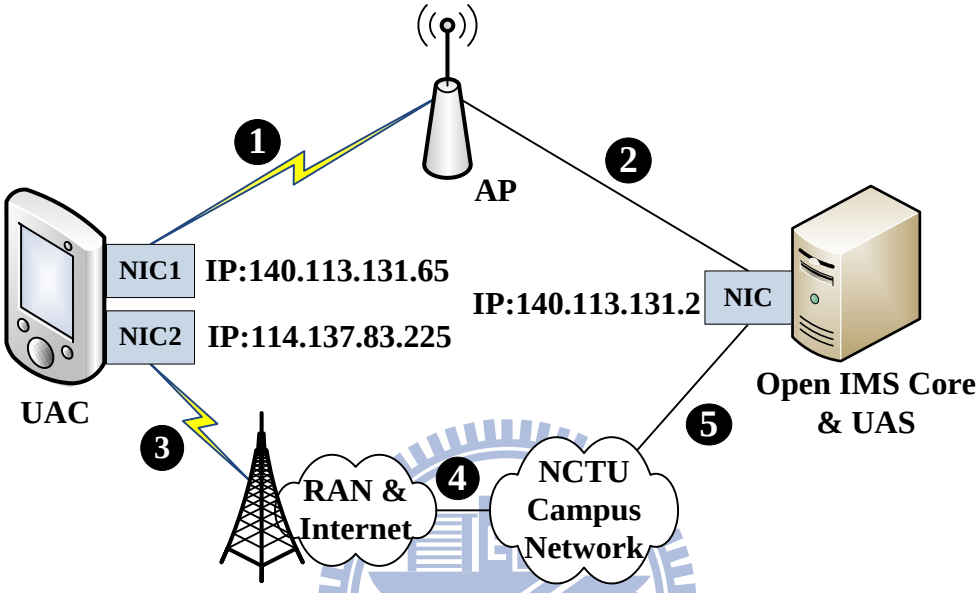

圖 4.1: 實驗環境

本論文實驗環境如圖 4.1 所示，主要包含了 UAC、核心網路軟體 (Open IMS Core) 與 UAS 三個元件。本論文使用 CHT 9110 智慧型手機做為實驗 UE 來執行 SIP/IMS UA 軟體，負責扮演 UAC 角色。CHT 9110 智慧型手機規格如表 4.1 所示，採用 Samsung™ 500MHz 處理器以及 256MB ROM 與 64MB RAM，使用的作業系統為 Microsoft Windows Mobile® 6.0 Professional，並內建 IEEE 802.11 b/g 無線網路 (表 4.1 之 NIC1) 與 HSDPA (High Speed Downlink Packet Access) 蜂巢式無線電通訊模組 (表 4.1 之 NIC2)。本實驗使用開放原始碼 Fraunhofer FOKUS Open IMS Core [36] 做為核心網路軟體，並安裝於 Ubuntu 9.10 作業系統上。Open IMS Core 以 3GPP IMS 網路架構為主，並能同時服務 IETF 所規範的 SIP UA，提供了後續實驗所需的各種功能。本論文所開發之 SIP/IMS UA 程式碼可針對

Windows 或 Linux 作業系統，建置出相對應的 SIP/IMS UA 軟體。為了減低 UAS 與 Open IMS Core 之間傳遞封包的延遲，本實驗以 Linux 作業系統版本之 SIP/IMS UA 來擔任 UAS 角色，並與 Open IMS Core 安裝於同一部電腦設備中。Open IMS Core 與 UAS 使用之電腦設備規格如表 4.2 所示，硬體使用 Intel® Core™ 2 2.13GHz 處理器、1GB 記憶體以及 Realtek RTL8111B Gigabit Ethernet 網路介面控制器 (Network Interface Controller; 簡稱 NIC)。

表 4.1: 實驗設備 – UAC (CHT 9110)

CPU	Samsung™ 2442, 500 MHz
Memory	256 MB ROM, 64 MB SDRAM
OS	Microsoft Windows Mobile® 6.0 Professional
NIC1	IEEE 802.11 b/g Compatible WiFi
NIC2	HSDPA/UMTS/GSM/GPRS/EDGE Module
Software	SIP/IMS UA (Windows Mobile Version)

表 4.2: 實驗設備 – Open IMS Core & UAS

CPU	Intel® Core™ 2 6400 2.13GHz
Memory	1 GB RAM
OS	Ubuntu 9.10
NIC	Realtek RTL8111B Gigabit Ethernet NIC
Software	Fraunhofer FOKUS Open IMS Core & SIP/IMS UA (Linux Version)

在實驗網路的配置上，本論文在 UAC 與 Open IMS Core 之間分別使用了可控制網路與實際網路來進行測試。UAC 在可控制網路上採用 IEEE 802.11g 無線區域網路 (*Wireless Local Area Networ*k; WLAN) 來進行資料傳輸，配置的網路 IP 位址為 140.113.131.65。此環境下 UAC 的信令傳送路徑為圖 4.1①→②，並在測量時沒有其他無關之設備及網路接入。實際網路為 UAC 使用網路服務業者 (中華電信) 所提供的 3.5G 行動通訊網路 (HSDPA) 來與其他設備溝通，配置的網路 IP 位址為 114.137.88.225。此環境下 UAC 的信

令傳送路徑為圖 4.1③→④→⑤，其中 UAC 會透過行動通話基地台與無線接取網路 (*Radio Access Network*; RAN)，經由網際網路 (Internet) 與交通大學校園網路 (NCTU Campus Network)，來連結架設於交通大學電資大樓 610 室的 Open IMS Core 及 UAS 設備進行測試。Open IMS Core 與 UAS 所配置的網路 IP 位址為 140.113.131.2。

在實驗項目上，本論文共設計六種實驗，其中實驗 1 至 5 使用可控制的 802.11g 無線區域網路來進行測試，實驗 6 則使用實際 3.5G HSDPA 行動通訊網路。在實驗過程中，每個實驗項目程序將被連續執行 500 次[1]，由 UAC 來測量與記錄每次所測得之 PDD 數值，同時於 Open IMS Core 設備上執行 Wireshark 封包分析器 [37] 記錄收送封包的內容。最後，從測量結果中計算收送封包大小，及 PDD 之最小值、平均值、最大值與標準差。

在實驗數值的比較上，本論文定義 *Diff* (α, β)代表數值 α 至 β 所增加的大小，也就是數值 α 與 β 之間的差異：

$$Diff(\alpha,\beta) = \left|\frac{\beta-\alpha}{\alpha}\right| \times 100\% \qquad (4.1)$$

本論文透過公式 4.1 來計算兩實驗數值的差異。接著後續小節將說明本論文進行的六種實驗。

[1] 附錄 A 分析出，當實驗次數到達 500 次時，之後每次測量得到的平均 PDD 值最大差異會小於 0.1%。

4.1 實驗 1：開啟與關閉 3GPP Profile 對 PDD 之影響

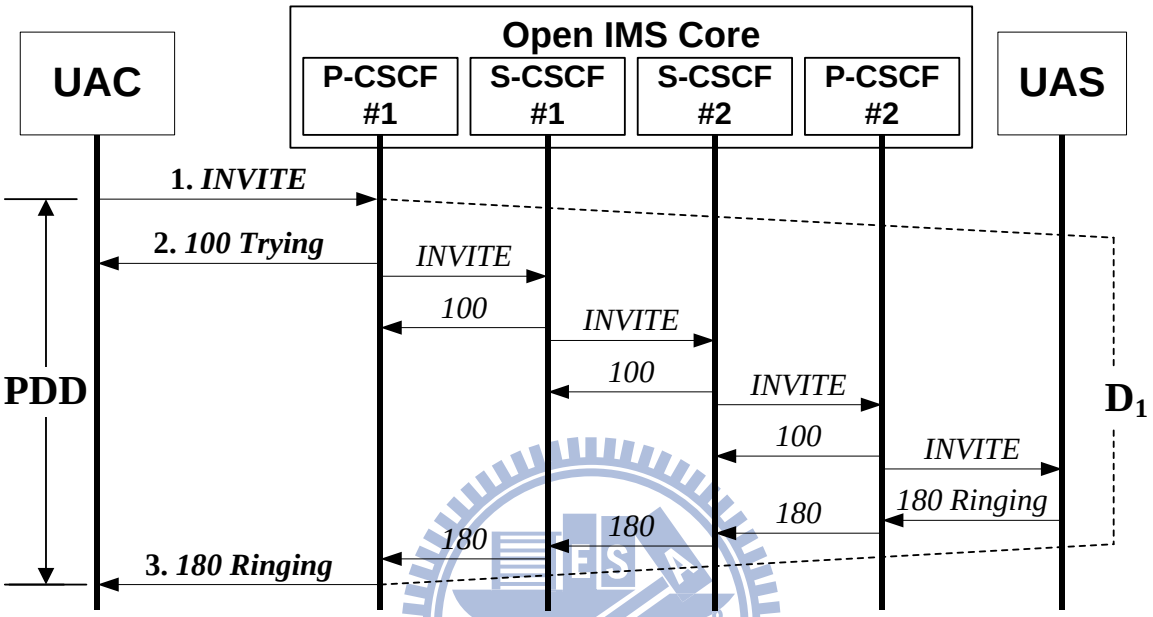

圖 4.2: 實驗 1 訊息流程

實驗 1 以第 3.1 節描述之通話建立程序，研究 UAC 開啟與關閉 3GPP Profile 對 PDD 之影響。圖 4.2 為實驗 1 之訊息流程，在本論文的實驗環境中，UAC 與 UAS 發送的 SIP 訊息皆會經由 Open IMS Core 處理與轉送。以 UAC 發送 SIP 訊息至 UAS 為例，首先 UAC 須將 SIP 訊息送至 Open IMS Core 中被指派的 P-CSCF (圖 4.2 P-CSCF#1)，並由此 P-CSCF 轉送 SIP 訊息至服務 UAC 的 S-CSCF (圖 4.2 S-CSCF#1)。本實驗環境中 UAC 與 UAS 位於同一個本籍網路，所以 UAC 發送給 UAS 的 SIP 訊息不須經由 UAS 的 I-CSCF 來進入 UAS 的本籍網路，S-CSCF#1 可直接將訊息轉送至服務 UAS 的 S-CSCF (圖 4.2 S-CSCF#2)。接著此 S-CSCF 負責傳送 SIP 訊息至 UAS 被指派的 P-CSCF (圖 4.2 P-CSCF#2)，最後再由此 P-CSCF 轉送 SIP 訊息給 UAS。反之亦然，若 UAS 發送 SIP 訊息至 UAC，也是以相同方式傳送。

4.1.1 實驗 1 收送訊息比較

本論文在 SIP/IMS UA 的實作上，參考了 RFC 3261 [2] 與 3GPP TS 24.229 [20] 所提出的 SIP 標頭使用規範，並配合實際的使用需求來進行開發。使用者可利用 SIP/IMS 設定頁面上的 3GPP Profile Enabled 選項 (圖 2.5ⓐ)，切換 SIP/IMS UA 建構 SIP 標頭的方式。當此選項關閉時，SIP/IMS UA 會遵循 3GPP TS 24.229 所規範的「RFC status」內容建構 SIP 標頭，此時 SIP/IMS UA 可在 IETF SIP 網路環境下進行服務。相反地，若開啟此選項，則 SIP/IMS UA 就會改以 3GPP TS 24.229 規範的 3GPP「Profile status」來建構 SIP 標頭，以便支援 3GPP IMS 網路架構下的服務標準。表 4.3 記錄了 UAC 所收送的 SIP 訊息及訊息中夾帶的 SIP 標頭，說明如下。

表 4.3: 實驗 1 之 UAC 收送訊息與 SIP 標頭

RFC	SIP Header	3GPP Profile 選項	
		關閉	開啟
圖 4.2 步驟 1: UAC → P-CSCF (INVITE)			
3261	Call-ID, Contact, Content-Length, Content Type, CSeq, From, Max.-Forwards, Require, Route, Supported, To, Via, Allow, User-Agent	有	有
3323	***Privacy***	無	有
3325	***P-Preferred-Identity***	無	有
3455	***P-Access-Network-Info***	無	有
圖 4.2 步驟 2: UAC ← P-CSCF (100 Trying)			
3261	Call-ID, Content-Length, CSeq, From, Server, To, Via, Warning	有	有
圖 4.2 步驟 3: UAC ← P-CSCF (180 Ringing)			
3261	Call-ID, Contact, Content-Length, CSeq, From, Record-Route, Require, To, Via, Allow, User-Agent	有	有
3455	***P-Asserted-Identity*** (由 Open IMS Core 產生)	有	有

如表 4.3 所示，在關閉 3GPP Profile 時，UAC 發送的訊息皆為 RFC 3261 所規範的 SIP 標頭。若開啟 3GPP Profile，則 UAC 除了建構既有 RFC 3261 規範的 SIP 標頭外，會再建構 3GPP IMS 服務所需的延伸標頭。如步驟 1 INVITE 訊息會建構 *P-Preferred-Identity*、*Privacy* 與 *P-Access-Network-Info* 標頭。此外，由於實驗使用 Open IMS Core 做為核心網路軟體，所以無論 UAC 是否啟用 3GPP Profile，步驟 3 收到 Open IMS Core 所轉送的 180 Ringing 訊息中，皆會包含 Open IMS Core 產生的 *P-Asserted-Identity* 延伸標頭 [23]。表 4.4 為實驗中 UAC 開啟與關閉 3GPP Profile 時收送訊息的比較，說明如下。

表 4.4: 實驗 1 之 UAC 收送訊息比較

Step	Direction	Message Type	Packet Size (bytes)		
			3GPP Profile		Increment
			Disabled	Enabled	
1	UAC → P-CSCF	SIP INVITE	683	806	123
2	UAC ← P-CSCF	SIP 100	546	546	0
3	UAC ← P-CSCF	SIP 180	627	627	0
Total			1856	1979	123

由於 SIP 訊息在 *Via* 標頭的 branch 參數、*From* 與 *To* 標頭的 tag 參數，以及 *Call-ID* 標頭的內容皆為自動隨機產生，所以每次測量到的訊息長度不相同。因此本論文以平均值表示訊息長度。

表 4.4 中，當 UAC 啟用 3GPP Profile 後，步驟 1 INVITE 訊息將多了 3GPP IMS 服務所需的延伸標頭，以致訊息長度增加 123 bytes。其中包含 *Privacy* 標頭 15 bytes、*P-Preferred-Identity* 標頭 50 bytes、及 *P-Access-Network-Info* 標頭 58 bytes。步驟 2 與 3 訊息長度維持不變。總計 UAC 關閉 3GPP Profile 時共收送訊息大小 1856 bytes，若開啟則變為 1979 bytes，增加 123 bytes。

4.1.2 實驗 1 之 PDD 測量結果與分析

此小節分析本實驗 PDD 測量結果。PDD 測量結果如表 4.5 所示，此表格包含以下欄位，由左至右依序為：

表 4.5: 實驗 1 之 PDD 測量結果

Test Item	PDD (ms)			
	Min	Avg	Max	StD
UDP-Basic	19.36	29.60	52.95	8.44
UDP-3GPP	19.65	29.72	53.59	8.15

- **Test Item**：依據實驗項目之特性所給予的簡稱，例如在 UDP 傳輸協定下的基本通話實驗項目 (關閉 3GPP Profile) 簡稱為 ***UDP-Basic***，而啟用 3GPP Profile 的實驗項目簡稱為 ***UDP-3GPP***。
- **PDD**：實驗所測量之 PDD 數據，單位為 ms。PDD 數據包含了最小值 **Min**，平均值 **Avg**，最大值 **Max**，以及標準差 **StD**。

表 4.5 測得 ***UDP-Basic*** 與 ***UDP-3GPP*** 之平均 PDD 分別為 29.60 與 29.72 ms。本論文將「P-CSCF[2]從收到 UAC 的 INVITE 發話請求開始，至傳送 180 Ringing 振鈴回覆給 UAC 所經過的時間」定義為 D_1 (參見圖 4.2)，並利用 D_1 來輔助驗證 PDD 正確性。表 4.6 為上述項目測量結果比較，說明如下。

表 4.6: 實驗 1 各項目測量比較

Test Item	PDD	D_1
UDP-Basic	29.60	3.81
UDP-3GPP	29.72	3.90
Diff (UDP-Basic, UDP-3GPP)	0.405%	2.362%

單位: ms

[2]本章節與後續章節所提到之 P-CSCF 與 S-CSCF，若無特別說明則為 UAC 所屬的 P-CSCF (圖 4.2 P-CSCF#1) 與服務 UAC 的 S-CSCF (圖 4.2 S-CSCF#1)。

表 4.6 推算出 ***UDP-Basic*** 與 ***UDP-3GPP*** 之 PDD 平均值差異為 0.405%，並測量出兩者 D_1 平均值分別為 3.81 與 3.90 ms，差異 2.362%。本論文於附錄 C 利用附錄 B 提出的方法，驗證了此 PDD 與 D_1 測量結果正確性。經由以上分析，本實驗觀察到 UAC 在啟用 3GPP Profile 時，UAC 需多建置與傳送 3GPP IMS 服務所需的延伸標頭 123 bytes。這些因素造成 PDD 增加 0.405%的延遲。

4.2 實驗 2：基本通話建立在 UDP 與 TCP 協定下對 PDD 之影響

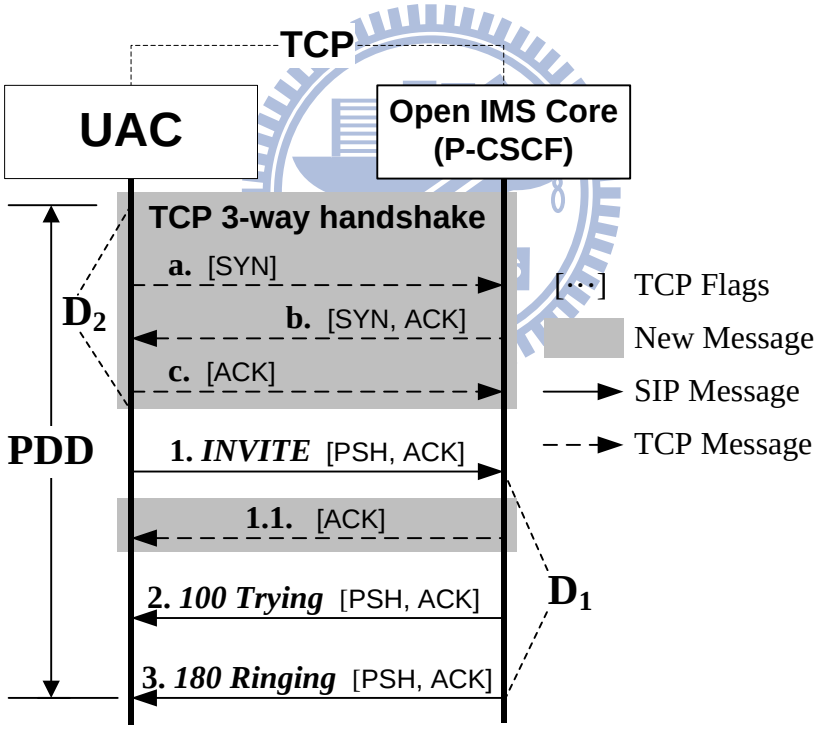

圖 4.3: 實驗 2 訊息流程 –UAC 與核心網路間使用 TCP 協定建立通話

實驗 2 以第 3.1 與 3.2 節描述之通話建立程序，研究通話建立時，UAC 與核心網路間使用 UDP 與 TCP 傳輸協定對 PDD 的影響。本論文於實驗 1 已說明 UDP 傳輸協定下的通話建立流程，而圖 4.3 為 UAC 與核心網路間使用 TCP 傳輸協定的流程。在測量 TCP 傳

輸協定時，由於 TCP 為連結導向的傳輸模式，所以每次測量都須完全中斷 TCP 連結後才可繼續下個測量，確保每次測量均會重新建立 TCP 連結。

4.2.1 實驗 2 收送訊息比較

本論文將 UAC 建立 TCP 連結所用的 TCP 三向交握封包記錄於表 4.7 步驟 a 至 c。其中步驟 a 的 TCP SYN 封包長度為 66 bytes，步驟 b 的 TCP SYN,ACK 為 66 bytes，步驟 c 的 TCP ACK 為 60 bytes，此部份共計收送了 192 bytes。TCP 連結建立完成後，後續 UAC 收送的訊息如表 4.7 步驟 1 至 3 所示，並與 UDP 協定比較。

表 4.7: 實驗 2 之 UAC 收送訊息比較

Step	Direction	Message Type	Packet Size (bytes)		
			UDP	TCP	Increment
a	UAC → P-CSCF	TCP SYN	-	66	66
b	UAC ← P-CSCF	TCP SYN,ACK	-	66	66
c	UAC → P-CSCF	TCP ACK	-	60	60
1	UAC → P-CSCF	SIP INVITE	683	709	26
1.1	UAC ← P-CSCF	TCP ACK	-	60	60
2	UAC ← P-CSCF	SIP 100	546	558	12
3	UAC ← P-CSCF	SIP 180	627	639	12
Total			1856	2158	302

表 4.7 中，由於 TCP 標頭 (20 bytes) 比 UDP 標頭 (8 bytes) 多出 12 bytes，所以改用 TCP 協定時，步驟 1 至 3 訊息皆會增加 12 bytes。並且 UAC 在步驟 1 INVITE 訊息的 SIP *Contact* 標頭會多記錄傳輸協定資訊「;transport=TCP」，長度 14 bytes。所以步驟 1 訊息共增加 26 bytes (12+14)。此外，當 P-CSCF 收到 UAC 發送的 INVITE 訊息時，會額外發送 TCP ACK 來回應 UAC，長度 60 bytes。總計 UAC 在 UDP 協定下共收送訊息 1856 bytes，而在 TCP 協定下收送 2158 bytes，增加 302 bytes。

4.2.2 實驗 2 之 PDD 測量結果與分析

此小節分析本實驗 PDD 測量結果，並與前 ***UDP-Basic*** 測量結果進行比較。PDD 測量結果如表 4.8 所示。

表 4.8: 實驗 2 之 PDD 測量結果

Test Item	PDD (ms)			
	Min	Avg	Max	StD
UDP-Basic	19.36	29.60	52.95	8.44
TCP-Basic	26.65	36.23	46.59	2.43

表 4.8 中，本論文簡稱 UAC 與 Open IMS Core 間使用 TCP 協定傳輸訊息的基本通話建立測量為 ***TCP-Basic***，測得平均 PDD 為 36.23 ms。為了探討 ***UDP-Basic*** 與 ***TCP-Basic*** 的 PDD 差異，除了實驗 1 所定義的 D_1 外，本論文再將「TCP 三向交握延遲」定義為 D_2 (參見圖 4.3)。表 4.9 為 ***UDP-Basic*** 與 ***TCP-Basic*** 上述項目測量結果比較。

表 4.9: 實驗 2 各項目測量結果比較

Test Item	PDD	D_1	D_2
UDP-Basic	29.60	3.81	-
TCP-Basic	36.23	3.88	6.23
Diff (UDP-Basic, TCP-Basic)	22.399%	1.837%	-

單位: ms

表 4.9 測量出 ***TCP-Basic*** 之 D_1 平均值為 3.88 ms，相較於 ***UDP-Basic*** 的 3.81 ms，差異 1.837%。表 4.9 也測量出 ***TCP-Basic*** 之 TCP 三向交握延遲 D_2 平均值為 6.23 ms。為了驗證 D_2 的正確性，本實驗另外以程式在此實驗環境下進行 TCP 三向交握 100,000 次，測得總平均延遲為 6.19 ms。D_2 與此值誤差 0.04 ms，誤差在可接受範圍，故推論 D_2 為合理值。本論文也於附錄 C 驗證 ***TCP-Basic*** 之 PDD 與 D_1 測量結果正確性。

經由以上分析，本實驗觀察到 UAC 在此網路環境下建立通話，使用 TCP 協定會較使用 UDP 協定多出 TCP 的三向交握、ACK 確認回應與訊息標頭差異，因而造成 PDD 增加 22.399%的延遲。

4.3 實驗 3：QoS 協商機制在 UDP 與 TCP 協定下對 PDD 之影響

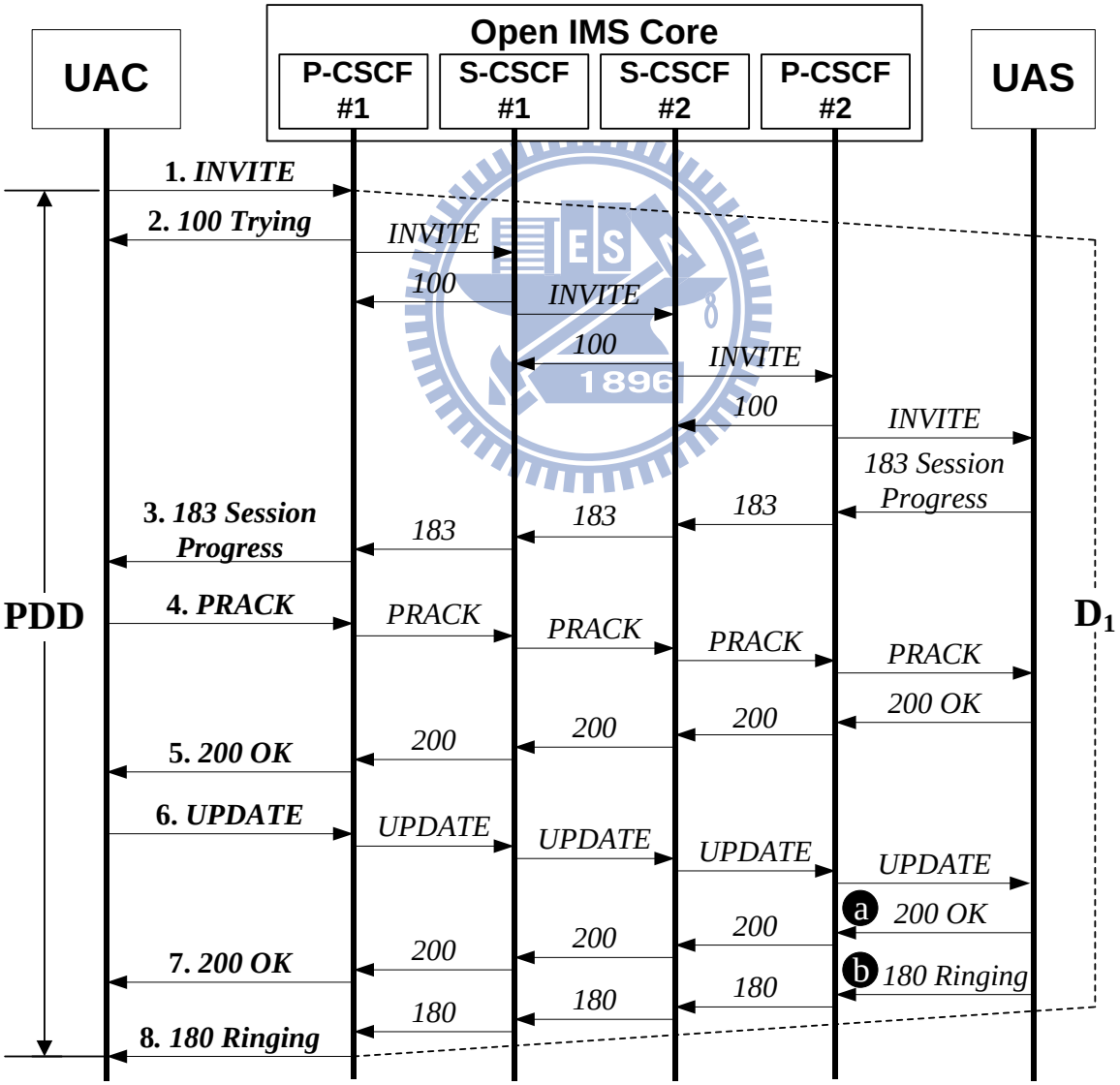

圖 4.4: 實驗 3 訊息流程 – UAC 於 UDP 協定下協商 QoS 資訊

實驗 3 以第 3.3 節描述之通話建立程序，研究通話建立時若啟用 QoS 協商，在 UDP 與 TCP 協定下對 PDD 的影響。圖 4.4 為 UAC 於 UDP 協定下協商 QoS 資訊的訊息流程。由於 QoS 資源保留程序不在本論文探討的範疇，因此本實驗僅分析 UA 雙方使用 SIP 與 SDP 協商 QoS 資訊的部份，不包括資源保留信令。所以當 UAC 收到 PRACK 的 200 OK 回覆後 (圖 4.4 步驟 5)，即可馬上發送 UPDATE 訊息 (圖 4.4 步驟 6)，以及 UAS 在發送 UPDATE 的 200 OK 回覆後 (圖 4.4ⓐ)，將隨即發送 180 Ringing 訊息 (圖 4.4ⓑ)。

4.3.1 實驗 3 收送訊息比較

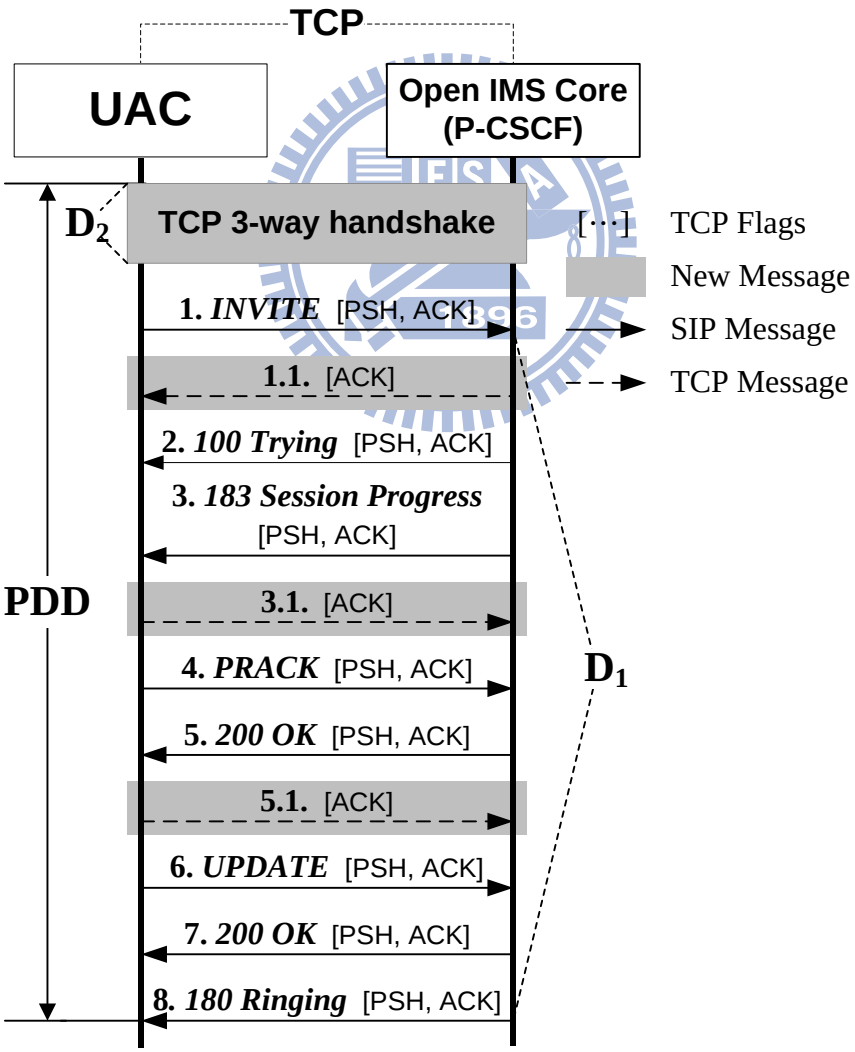

圖 4.5: 實驗 3 訊息流程 – UAC 於 TCP 協定下協商 QoS 資訊

在協商 QoS 資訊的流程中，若將 UAC 與核心網路間的傳輸協定由 UDP 改為 TCP，其訊息流程如圖 4.5 所示。表 4.10 比較了 UAC 於此兩種傳輸協定下收送的訊息，說明如下。

表 4.10: 實驗 4 之 UAC 收送訊息比較

Step	Direction	Message Type	Packet Size (bytes)		
			UDP	TCP	Increment
a	UAC → P-CSCF	TCP SYN	-	66	66
b	UAC ← P-CSCF	TCP SYN,ACK	-	66	66
c	UAC → P-CSCF	TCP ACK	-	60	60
1	UAC → P-CSCF	SIP INVITE	849	875	26
1.1	UAC ← P-CSCF	TCP ACK	-	60	60
2	UAC ← P-CSCF	SIP 100	546	558	12
3	UAC ← P-CSCF	SIP 183	990	1002	12
3.1	UAC → P-CSCF	TCP ACK	-	60	60
4	UAC → P-CSCF	SIP PRACK	853	865	12
5	UAC ← P-CSCF	SIP 200	626	638	12
5.1	UAC → P-CSCF	TCP ACK	-	60	60
6	UAC → P-CSCF	SIP UPDATE	859	885	26
7	UAC ← P-CSCF	SIP 200	606	618	12
8	UAC ← P-CSCF	SIP 180	653	665	12
Total			5982	6478	496

由表 4.10 可知，TCP 協定下的通話建立流程會多出 TCP 三向交握 (步驟 a 至 c) 與額外三個 TCP ACK 回應訊息 (步驟 1.1、3.1 與 5.1)。而原有 SIP 訊息標頭長度也皆增加 12 bytes 的傳輸協定標頭差距。在步驟 1 INVITE 訊息與步驟 6 UPDATE 訊息中，UAC 會於 SIP *Contact* 標頭內夾帶傳輸協定資訊「;transport=TCP」14 bytes，所以此兩訊息長度將增加 26 bytes。總計 UAC 在 UDP 協定下共收送訊息 5982 bytes，而在 TCP 協定下收送 6478 bytes，一共增加了 496 bytes。

4.3.2 實驗 3 之 PDD 測量結果與分析

此小節分析本實驗 PDD 測量結果。PDD 測量結果如表 4.11 所示。

表 4.11: 實驗 3 之 PDD 測量結果

Test Item	PDD (ms)			
	Min	Avg	Max	StD
UDP-QoS	79.14	103.59	189.17	19.70
TCP-QoS	91.61	120.74	177.70	11.31

表 4.11 中，本論文簡稱 UAC 以 UDP 傳輸協定協商 QoS 的實驗項目為 ***UDP-QoS***，而以 TCP 協定協商 QoS 的實驗項目為 ***TCP-QoS***。並測得兩實驗項目平均 PDD 分別為 103.59 與 120.74 ms。本論文接著比較兩實驗項目測量結果，如表 4.12 所示。

表 4.12: 實驗 3 各項目測量結果比較

Test Item	PDD	D_1
UDP-QoS	103.59	77.68
TCP-QoS	120.74	88.35
Diff (UDP-QoS, TCP-QoS)	16.556%	13.736%

單位: ms

表 4.12 推算出 ***UDP-QoS*** 與 ***TCP-QoS*** 之平均 PDD 差異為 16.556%，並測量出 D_1 平均值分別為 77.68 與 88.35 ms，差異 13.736%。本論文在實驗 2 已測量及驗證此實驗環境下的 TCP 三向交握延遲 D_2 為 6.23 ms。並於附錄 C 將 TCP 協定較 UDP 協定多出的訊息刪除後比較，驗證了 PDD 與 D_1 測量結果正確性。

經由上述分析，本實驗觀察到 UAC 建立通話時若啟用 QoS 協商，使用 TCP 協定會較使用 UDP 協定對 PDD 增加 16.556%的延遲。

4.4 實驗 4：通話認證機制在 UDP 與 TCP 協定下對 PDD 之影響

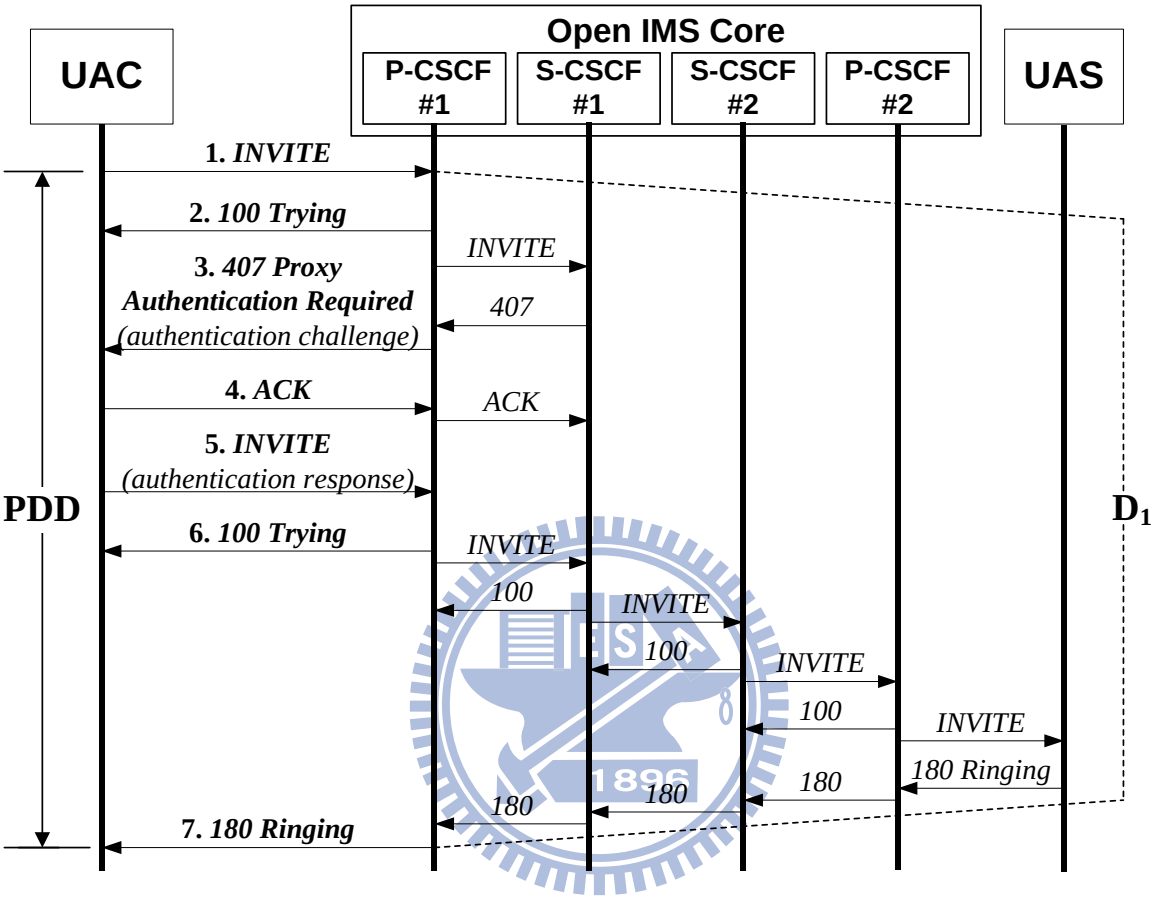

圖 4.6: 實驗 4 訊息流程 – UAC 於 UDP 協定下進行通話認證

實驗 4 以第 3.4 節描述之通話建立程序，研究通話建立時若對 UAC 進行身份認證，在 UDP 與 TCP 協定下對 PDD 的影響。圖 4.6 為 UAC 於 UDP 協定下進行通話認證的訊息流程。在 3GPP TS 24.229 [20] 提到 IMS 核心網路中，當服務 UAC 的 S-CSCF 收到 UAC 建立通話的 INVITE 訊息時，若 S-CSCF 欲進行身份認證，則 S-CSCF 須繼續延用 UAC 先前註冊認證時所用的認證參數，不再向 HSS 要求提供 UAC 的認證參數 (認證參數請參見第 3.4 節說明)。因此 S-CSCF 會直接回覆 407 Proxy Authentication Required 訊息，並夾帶先前記錄的認證參數，經由 P-CSCF 轉送至 UAC 進行認證查問。

4.4.1 實驗 4 收送訊息比較

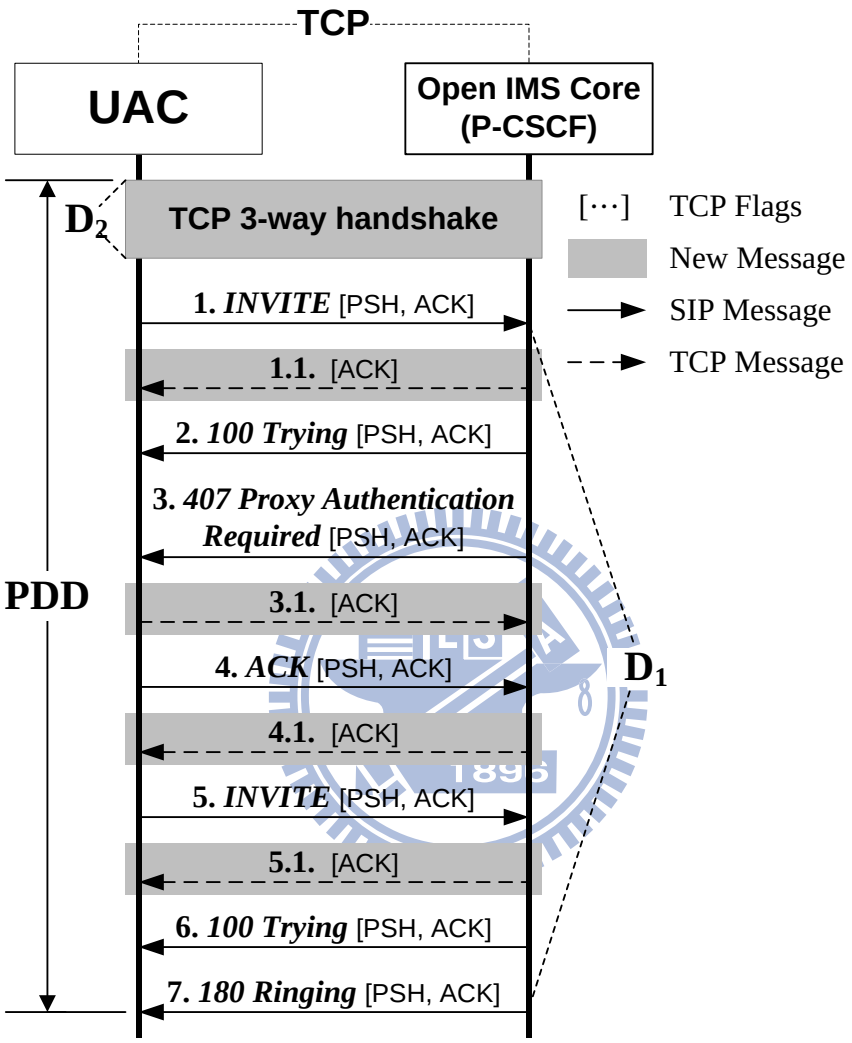

圖 4.7: 實驗 4 訊息流程 – UAC 於 TCP 協定下進行通話認證

通話認證的流程中，若將 UAC 與核心網路間的傳輸協定由 UDP 改為 TCP，其訊息流程如圖 4.7 所示。表 4.13 比較了 UAC 於此兩種傳輸協定下收送的訊息，說明如下。

表 4.13: 實驗 4 之 UAC 收送訊息比較

Step	Direction	Message Type	Packet Size (bytes)		
			UDP	TCP	Increment
a	UAC → P-CSCF	TCP SYN	-	66	66
b	UAC ← P-CSCF	TCP SYN,ACK	-	66	66
c	UAC → P-CSCF	TCP ACK	-	60	60
1	UAC → P-CSCF	SIP INVITE	683	709	26
1.1	UAC ← P-CSCF	TCP ACK	-	60	60
2	UAC ← P-CSCF	SIP 100	546	558	12
3	UAC ← P-CSCF	SIP 407	876	888	12
3.1	UAC → P-CSCF	TCP ACK	-	60	60
4	UAC → P-CSCF	SIP ACK	433	445	12
4.1	UAC ← P-CSCF	TCP ACK	-	60	60
5	UAC → P-CSCF	SIP INVITE	943	969	26
5.1	UAC ← P-CSCF	TCP ACK	-	60	60
6	UAC ← P-CSCF	SIP 100	546	558	12
7	UAC ← P-CSCF	SIP 180	627	639	12
Total			4654	5198	544

由表 4.13 可知，TCP 協定下的通話建立流程會多出 TCP 三向交握 (步驟 a 至 c) 與額外四個 TCP ACK 回應訊息 (步驟 1.1、3.1、4.1 與 5.1)。而原有 SIP 訊息標頭長度也皆增加 12 bytes 的傳輸協定標頭差距。在步驟 1 與步驟 5 的 INVITE 訊息中，UAC 會於 SIP *Contact* 標頭內夾帶傳輸協定資訊「;transport=TCP」14 bytes，所以兩訊息長度皆增加了 26 bytes。總計 UAC 在 UDP 協定下共收送訊息 4654 bytes，而在 TCP 協定下收送 5198 bytes，一共增加了 544 bytes。

4.4.2 實驗 4 之 PDD 測量結果與分析

此小節分析本實驗 PDD 測量結果。PDD 測量結果如表 4.14 所示。

表 4.14: 實驗 4 之 PDD 測量結果

Test Item	PDD (ms)			
	Min	Avg	Max	StD
UDP-Auth	38.89	63.76	105.63	14.25
TCP-Auth	64.23	75.29	132.19	4.66

表 4.14 中，本論文簡稱 UAC 於 UDP 傳輸協定下進行通話認證的實驗項目為 ***UDP-Auth***，而 TCP 傳輸協定下為 ***TCP-Auth***。並測得兩實驗項目平均 PDD 分別為 63.76 與 75.29 ms。本論文接著比較兩實驗項目測量結果，如表 4.15 所示。

表 4.15: 實驗 4 各項目測量結果比較

Test Item	PDD	D_1
UDP-Auth	63.76	38.09
TCP-Auth	75.29	43.31
Diff (UDP-Auth, TCP-Auth)	18.083%	13.704%

單位: ms

表 4.15 推算出 ***UDP-Auth*** 與 ***TCP-Auth*** 之平均 PDD 差異為 18.083%，並測量出 D_1 平均值分別為 38.09 與 43.31 ms，差異 13.704%。本論文在實驗 2 已測得及驗證此實驗環境下的 TCP 三向交握延遲 D_2 為 6.23 ms。並於附錄 C 將 TCP 協定較 UDP 協定多出的訊息刪除後比較，驗證了 PDD 與 D_1 測量結果正確性。經由上述分析，本實驗觀察到 UAC 建立通話時若進行身份認證，在使用 TCP 較使用 UDP 對 PDD 增加 18.083%的延遲。

回顧實驗 1 至 4 的測量結果，本論文另外發現在 802.11g 實驗環境中，雖然實驗項目於 TCP 協定下測得的 PDD 平均值皆較 UDP 協定下大，但 PDD 標準差卻皆較 UDP 協定小。表示此實驗環境下 TCP 協定產生的 PDD 分佈較 UDP 協定平均。

4.5 實驗 5：啟用 TLS 安全機制對 PDD 之影響

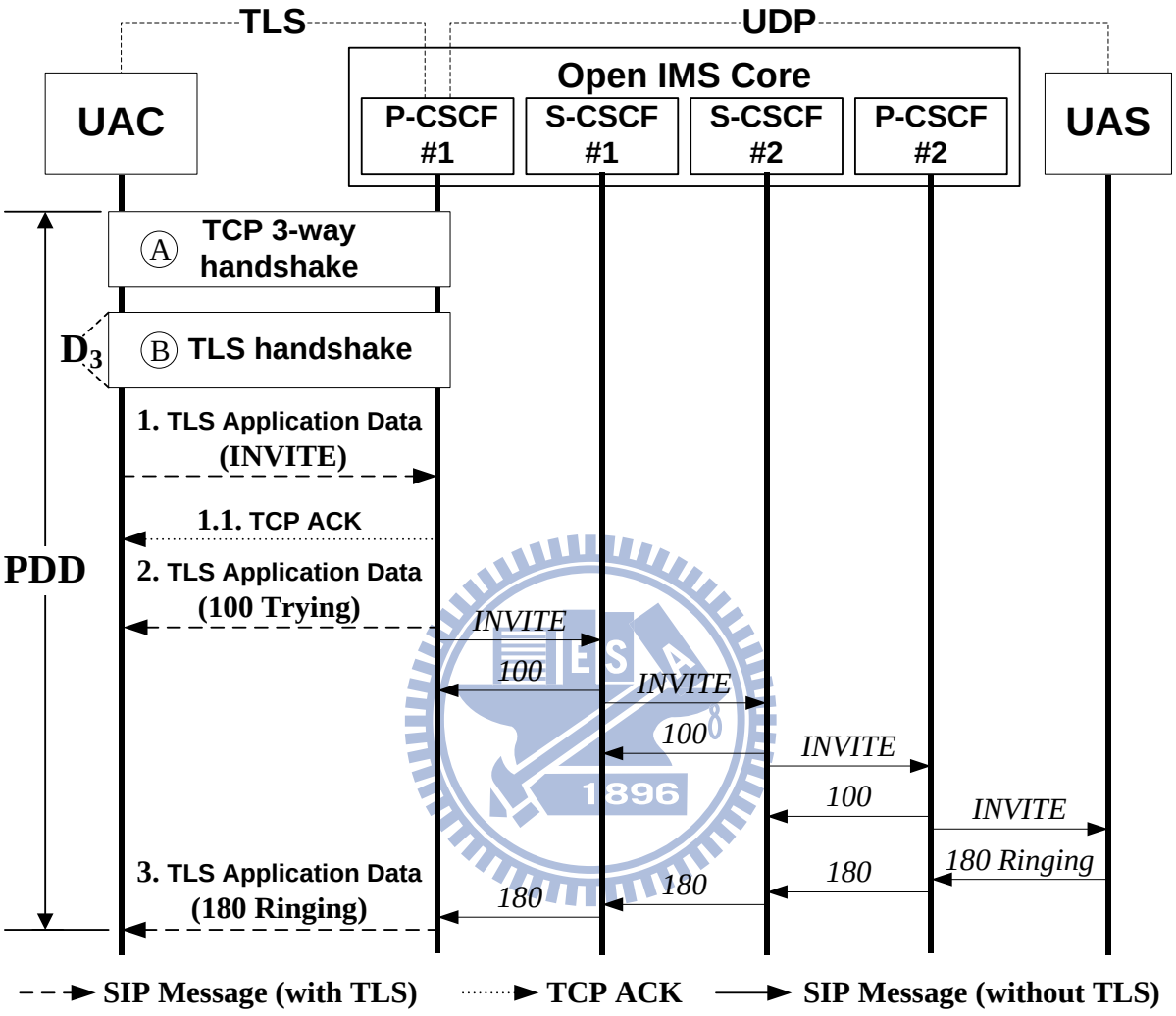

圖 4.8: 實驗 5 訊息流程 – UAC 啟用 TLS 安全機制

實驗 5 以第 3.5 節描述之通話建立程序，研究 UAC 與核心網路之間啟用 TLS 安全機制對 PDD 的影響。本實驗分為三部份，首先測量 TLS 交握的兩種 TLS 認證方式與 PDD，接著比較有無 TLS 安全機制的差異，最後結合 TLS 與 SIP 認證來比較兩種雙向認證的方法。

圖 4.8 為 UAC 啟用 TLS 安全機制的通話建立流程，當 UAC 開始發送 SIP 訊息前，會先以 TCP 三向交握 (圖 4.8Ⓐ) 及 TLS 交握 (圖 4.8Ⓑ) 來建立安全連結。本論文在實驗 2 已說明 TCP 三向交握，所以本節後續將接著探討未說明的 TLS 交握部份。

4.5.1 TLS 交握與 PDD 之測量

第 3.5 節敘述 TLS 交握的認證方式有「只認證伺服器身份」與「伺服器與客戶端相互認證」兩種，因此本實驗分成此兩種方式來測量。首先本實驗分析「只認證伺服器身份」的 TLS 交握，並簡稱使用此認證方式建立通話的實驗項目為 ***TLS-A***。***TLS-A*** 的 TLS 交握訊息如表 4.16 所示。

表 4.16: 實驗項目 TLS-A 之 TLS 交握訊息

Step	Direction	Message Type	Packet Size (bytes)
1	UAC → P-CSCF	TLS Client Hello	148
1.1	UAC ← P-CSCF	TCP ACK	60
2	UAC ← P-CSCF	TLS Server Hello, Certificate, Server Hello Done	2169 (1514+655)
2.1	UAC → P-CSCF	TCP ACK	60
3	UAC → P-CSCF	TLS Client Key Exchange, Change Cipher Spec, Finished	252
3.1	UAC ← P-CSCF	TCP ACK	60
4	UAC ← P-CSCF	TLS Change Cipher Spec, Finished	113
4.1	UAC → P-CSCF	TCP ACK	60
Total			2922

表 4.16 中，步驟 1 為 UAC 發送給 P-CSCF 的 Client Hello 訊息，長度 148 bytes。步驟 2 為 P-CSCF 發送給 UAC 的封包，此封包由 Server Hello、Certificate 與 Server Hello Done 三訊息組成，總長度 2169 bytes。由於此封包長度超出 MTU (Maximum Transmission Unit) 限制，被 P-CSCF 分段 (fragment) 成長度 1514 與 655 bytes 兩封包發送。步驟 3 為 UAC 發送給 P-CSCF 的封包，此封包由 Client Key Exchange、Change Cipher Spec 與 Finished 三訊息組成，總長度 252 bytes。步驟 4 為 P-CSCF 發送給 UAC 的封包，內部包含 Change Cipher Spec 與 Finished 訊息，長度 113 bytes。上述步驟在發送方傳送封包後，皆會由接收方回應 TCP ACK 確認封包，長度皆 60 bytes。總計 ***TLS-A*** 在 TLS 交握過程共收送封包 2922 bytes。接著第二種 TLS 認證方式為「伺服器與客戶端相互認證」，本論文簡稱以方式建立通話的實驗項目為 ***TLS-B***。***TLS-B*** 的 TLS 交握訊息如表 4.17 所示。

表 4.17: 實驗項目 TLS-B 之 TLS 交握訊息

Step	Direction	Message Type	Packet Size (bytes)
1	UAC → P-CSCF	TLS Client Hello	148
1.1	UAC ← P-CSCF	TCP ACK	60
2	UAC ← P-CSCF	TLS Server Hello, Certificate, *Certificate Request*, Server Hello Done	2308 (1514+794)
2.1	UAC → P-CSCF	TCP ACK	60
3	UAC → P-CSCF	TLS *Certificate*, Client Key Exchange, *Certificate Verify*, Change Cipher Spec, Finished	2450 (1514+936)
3.1	UAC ← P-CSCF	TCP ACK	60
4	UAC ← P-CSCF	TLS Change Cipher Spec, Finished	113
4.1	UAC → P-CSCF	TCP ACK	60
Total			5259

將表 4.17 與 4.16 比較，表 4.17 在步驟 2 封包中多夾帶了 Certificate Request 訊息 275 bytes，使封包總長度增至 2308 bytes。此外，表 4.17 在步驟 3 封包中也多夾帶 Certificate 與 Certificate Verify 兩訊息，造成封包總長度增至 2450 bytes。此封包大小因為超出 MTU 限制而被分段成長度 1514 與 936 bytes 兩個封包發送。總計 ***TLS-B*** 在 TLS 交握過程共收送封包 5259 bytes，較 ***TLS-A*** 多出 2337 bytes。本論文接著測量 ***TLS-A*** 與 ***TLS-B*** 之 PDD，如表 4.18 所示。

表 4.18: 實驗 5 之 PDD 測量結果

Test Item	PDD (ms)			
	Min	Avg	Max	StD
TLS-A	85.69	111.19	162.64	13.97
TLS-B	135.51	162.98	212.75	14.61

表 4.18 測得 ***TLS-A*** 與 ***TLS-B*** 之平均 PDD 分別為 111.19 與 162.98 ms。為了驗證兩實驗項目 PDD 正確性，本實驗將「TLS 交握延遲」定義為 D_3 (參見圖 4.8)。並於附錄 C 利用兩實驗項目只有 TLS 交握不同的差別，將 TLS 交握部份刪除後比較，驗證兩者 PDD 正確性。表 4.19 為 PDD 與 D_3 的測量結果比較。

表 4.19: 實驗 5 各項目測量結果比較

Test Item	PDD	D_3
TLS-A	111.19	66.63
TLS-B	162.98	118.46
Diff (TLS-A, TLS-B)	46.578%	77.788%

單位: ms

表 4.19 推算出 ***TLS-A*** 與 ***TLS-B*** 之 PDD 差異為 46.578%，並測得兩實驗項目的 TLS 交握延遲 D_3 平均值分別為 66.63 與 118.46 ms，差異 77.788%。經由上述分析可知，啟用 TLS 安全機制下，若 TLS 交握認證使用「伺服器與客戶端相互認證」的方式，會較使用「只認證伺服器身份」對 PDD 增加 46.578%的延遲。

4.5.2 有無 TLS 安全機制之比較

TLS 安全機制基礎於 TCP 傳輸協定，所以接下來實驗以 TCP 協定為基礎，比較有無啟用 TLS 安全機制對 PDD 的影響。本實驗使用前 ***TCP-Basic*** 測量結果做為關閉 TLS 安全機制的實驗數據。而啟用 TLS 安全機制的項目上，由於 ***TLS-A*** 與 ***TLS-B*** 只有 TLS 認證方式不同，***TLS-A*** 在 TLS 交握的封包總傳輸量與延遲都較 ***TLS-B*** 小，所以本實驗選擇用 ***TLS-A*** 來進行分析。將 ***TCP-Basic*** 與 ***TLS-A*** 比較，兩項目在發送 SIP 訊息前皆會先進行 TCP 三向交握，接著 ***TLS-A*** 再進行 TLS 交握，後續兩項目收送的訊息如表 4.20 所示。

表 4.20: 實驗項目 TCP-Basic 與 TLS-A 收送訊息比較

Step	Direction	Message Type	Packet Size (bytes)		
			TCP-Basic	TLS-A (Encrypted)	Increment
1	UAC → P-CSCF	SIP INVITE	709	749	40
1.1	UAC ← P-CSCF	TCP ACK	60	60	0
2	UAC ← P-CSCF	SIP 100	558	589	31
3	UAC ← P-CSCF	SIP 180	639	670	31
Total			1966	2068	102

TLS-A 在建立 TLS 安全連線後，UAC 與 P-CSCF 之間的 SIP 訊息都將被加密成 TLS 訊息傳送，使用的加密機制為 TLS 交握協商結果 TLS_RSA_WITH_AES_256_CBC_SHA [7]。所以表 4.20 的 SIP 訊息在加密後，步驟 1 INVITE 長度增加 40 bytes，步驟 2 的 100 Trying 與步驟 3 的 180 Ringing 也皆增加 31 bytes。總計 ***TLS-A*** 在此部份共收送封包 2068 bytes，較 ***TCP-Basic*** 多出 102 bytes。

表 4.21 將 ***TCP-Basic*** 與 ***TLS-A*** 的 PDD 進行比較。此表格計算出 ***TCP-Basic*** 與 ***TLS-A*** 之 PDD 差異為 206.731%，代表 UAC 若於 TCP 協定中啟用 TLS 安全機制，PDD 將增加

206.731%的延遲。

表 4.21: 實驗項目 TCP-Basic 與 TLS-A 之 PDD 比較

Test Item	PDD (ms)
TCP-Basic	36.25
TLS-A	111.19
Diff (TCP-Basic, TLS-A)	206.731%

4.5.3 雙向認證方法比較

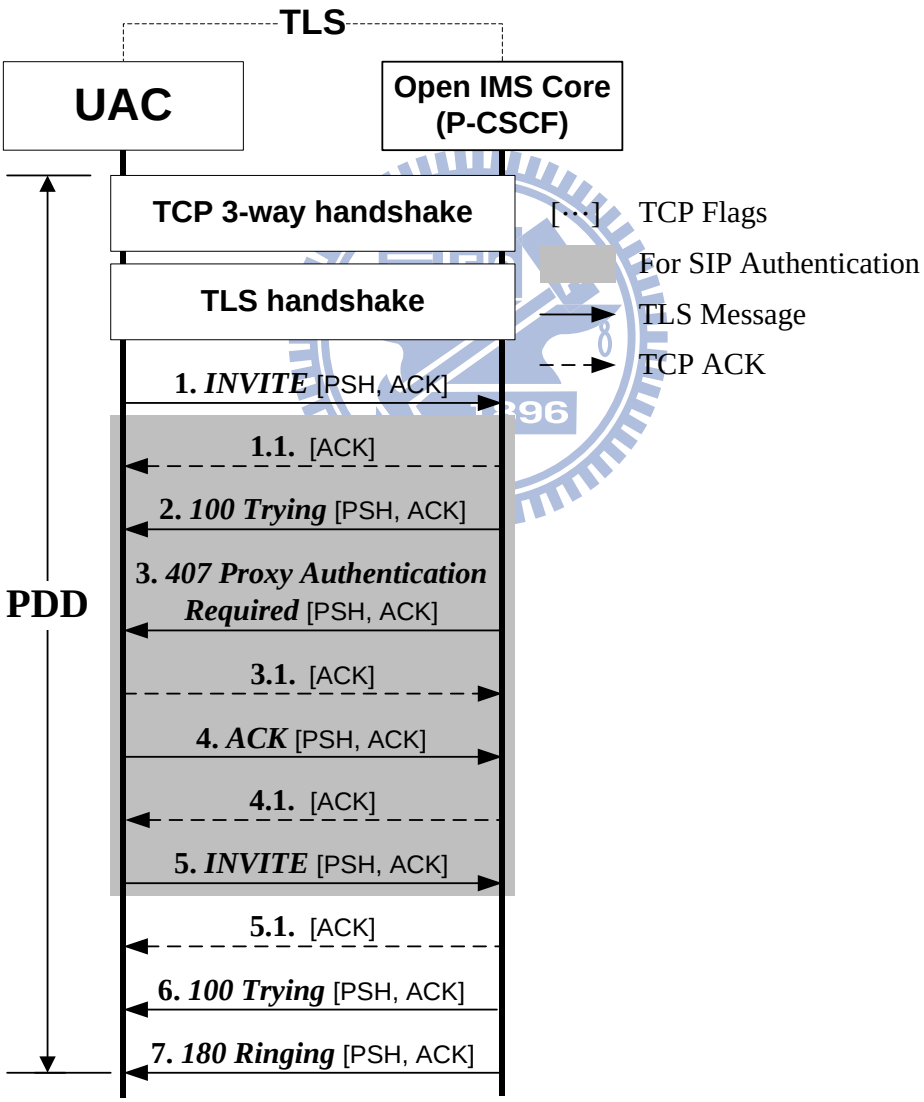

圖 4.9: 實驗 5 訊息流程 – TLS-A 結合 SIP 通話認證

通話建立時，UAC 與核心網路若要認證彼此身份的真實性，除了可使用 TLS 交握的雙向身份認證 (參見 ***TLS-B***)，也能在 TLS 交握時先認證核心網路身份 (參見 ***TLS-A***)，再結合第 3.4 節的 SIP 通話認證來認證 UAC (簡稱 ***TLS-A*+SIP Auth**)，達到相互認證的效果。後續小節比較上述兩種雙向認證方法。

TLS-A 結合 SIP 通話認證的訊息流程如圖 4.9 所示。第 4.5.1 節的表 4.16 與 4.17 已比較過 ***TLS-A*** 與 ***TLS-B*** 的 TLS 交握訊息差異，所以本論文接著比較兩項目從 UAC 發送 INVITE 訊息至收到 180 Ringing 回覆過程中收送的訊息，如表 4.22 所示，說明如下。

表 4.22: 雙向認證方法之收送訊息比較

Step	Direction	Message Type	Packet Size (bytes)		
			TLS-B	TLS-A+ SIP Auth	Increment
1	UAC → P-CSCF	SIP INVITE	749	749	0
1.1	UAC ← P-CSCF	TCP ACK	60	60	0
2	UAC ← P-CSCF	SIP 100	-	589	589
3	UAC ← P-CSCF	SIP 407	-	923	923
3.1	UAC → P-CSCF	TCP ACK	-	60	60
4	UAC → P-CSCF	SIP ACK	-	475	475
4.1	UAC ← P-CSCF	TCP ACK	-	60	60
5	UAC → P-CSCF	SIP INVITE	-	1003	1003
5.1	UAC ← P-CSCF	TCP ACK	-	60	60
6	UAC ← P-CSCF	SIP 100	589	589	0
7	UAC ← P-CSCF	SIP 180	670	670	0
Total			2068	5238	3170

表 4.22 列出的 SIP 訊息皆為 TLS 加密後的訊息。比較此兩種認證方法，***TLS-B*** 在此部份共收送封包 2068 bytes，而 ***TLS-A*** 結合 SIP 通話認證方法多出步驟 2 至 5.1 訊息，總計收送封包 5238 bytes，較 ***TLS-B*** 多了 3170 bytes。將此封包傳輸量加上 TCP 三向交握與

TLS 交握的部份，得到表 4.23 結果。

表 4.23: 雙向認證方法之封包傳輸量

Message	Packet Size (bytes)		
	TLS-B	TLS-A+SIP Auth	Increment
TCP 3-way handshake	192	192	0
TLS handshake	5259	2922	-2337
INVITE to 180 Ringing	2068	5238	3170
Total	7519	8352	833

由表 4.23 可知 ***TLS-B*** 的封包總傳輸量為 7519 bytes，而 ***TLS-A*** 結合 SIP 通話認證為 8352 bytes，較 ***TLS-B*** 多出 833 bytes。表 4.24 為此兩種方法的 PDD 測量結果。

表 4.24: 雙向認證方法之 PDD 測量結果

Test Item	PDD (ms)			
	Min	Avg	Max	StD
TLS-B	135.51	162.98	212.75	14.61
TLS-A+SIP Auth	161.73	202.92	267.15	21.11

表 4.24 測量出 ***TLS-A*** 結合 SIP 通話認證之平均 PDD 為 202.92 ms，較 ***TLS-B*** 的 162.98 ms 大。而由表 4.23 的計算已知 ***TLS-A*** 結合 SIP 通話認證的封包傳輸量比 ***TLS-B*** 大，所以 ***TLS-A*** 結合 SIP 通話認證產生的 PDD 較 ***TLS-B*** 大為合理的結果。

在實驗 4 與 5 的分析中，本論文觀察到在單向認證的比較下，SIP 通話認證無論使用 UDP 協定 (***UDP-Auth***) 或 TCP 協定 (***TCP-Auth***)，所表現出的 PDD 皆較 TLS 認證的 ***TLS-A*** 小。但在雙向認證上，***TLS-A*** 結合 SIP 通話認證所產生的 PDD，卻比雙向皆使用 TLS 認證的 ***TLS-B*** 還大。

4.6 實驗 6：以 3.5G HSDPA 環境進行測試

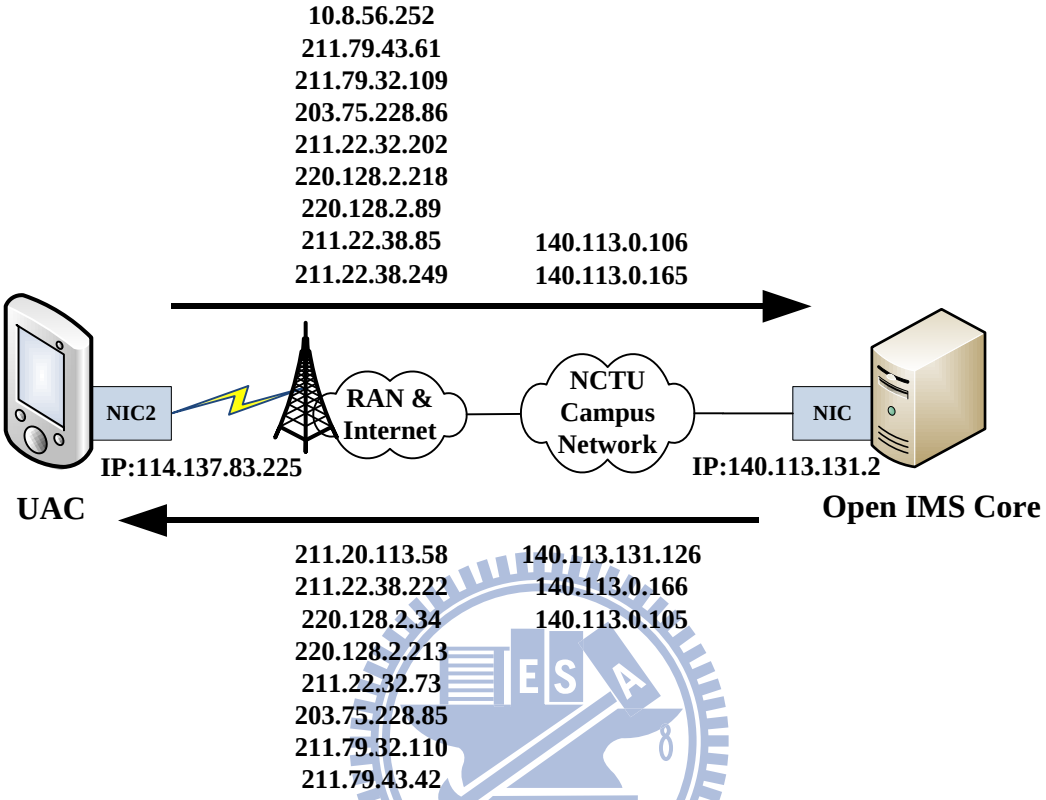

圖 4.10: 3.5G HSDPA 傳送路徑

實驗 6 將 UAC 的存取網路由 802.11g 無線區域網路改為網路服務業者提供的 3.5G 行動通訊網路 (HSDPA)，來重新測量先前的實驗項目[3]。圖 4.1 已說明此行動通訊網路的信令傳送路徑，而圖 4.10 展示實驗過程中訊息經過的網路節點。其中 UAC 對 Open IMS Core 發送的訊息會依序經過 IP 位址 10.8.56.252、211.79.43.61、211.79.32.109、203.75.228.86、211.22.32.202、220.128.2.218、220.128.2.89、221.22.38.85 與 221.22.38.249 的 9 個服務業者網路節點，以及 IP 位址 140.113.0.106 與 140.113.0.165 的 2 個交大校園網路節點，共計

[3] 實驗 6 不測量 ***UDP-3GPP*** 項目，原因為 ***UDP-3GPP*** 與 ***UDP-Basic*** 收送的訊息大小十分相近，不易在 3.5G HSDPA 網路上有明顯的差異比較。

11 個節點。另一方面，Open IMS Core 對 UAC 發送的訊息則會依序經過 IP 位址 140.113.131.126、140.113.0.166 與 140.113.0.105 的 3 個交大校園網路節點，以及 IP 位址 211.20.113.58、211.22.38.222、220.128.2.34、220.128.2.213、211.22.32.73、203.75.228.85、211.79.32.110 與 211.79.43.42 的 8 個服務業者網路節點，共計 11 個節點。由此觀察到 3.5G HSDPA 實驗網路在 UAC 與 Open IMS Core 之間是以非對稱路由 (asymmetric routing) 方式收送封包。各實驗項目於此網路下的測量結果如表 4.25 所示。

表 4.25: 各實驗項目於 3.5G HSDPA 網路下測量結果

Test Item	PDD (ms)			
	Min	Avg	Max	StD
UDP-Basic	125	238	730	107
TCP-Basic	466	552	1608	63
UDP-QoS	433	892	2615	221
TCP-QoS	1867	2220	4291	341
UDP-Auth	375	583	1443	100
TCP-Auth	1360	1574	3511	151
TLS-A	1285	1528	4763	242
TLS-B	1602	2560	6166	257
TLS-A+SIP Auth	2347	2729	6405	340

表 4.25 測得各實驗項目平均 PDD 分別為 238 ms (***UDP-Basic***)、552 ms (***TCP-Basic***)、892 ms (***UDP-QoS***)、2220 ms (***TCP-QoS***)、583 ms (***UDP-Auth***)、1574 ms (***TCP-Auth***)、1528 ms (***TLS-A***)、2560 ms (***TLS-B***) 與 2729 ms (***TLS-A*+SIP Auth**)。本論文接著將 802.11g 與 HSDPA 網路下的 PDD 測量結果進行綜合比較。首先比較基本通話建立於不同傳輸協定與不同存取網路下產生的 PDD 差異，如表 4.26 所示，說明如下。

表 4.26: 基本通話建立之 PDD 綜合比較

Test Item	PDD (ms)		
	802.11g	HSDPA	*Diff* (802.11g, HSDPA)
UDP-Basic	29.60	238	704%
TCP-Basic	36.23	552	1424%
Diff (UDP-Basic, TCP-Basic)	22%	132%	-

表 4.26 中，通話建立所用的傳輸協定若從 UDP 改為 TCP，在 802.11g 網路下 PDD 會增加 22%的延遲，而在 HSDPA 網路下會增加 132%。本論文觀察發現，上述兩種網路相比，由於 802.11g 的傳輸速度較快，所以當傳輸協定由 UDP 改為 TCP 時，相對產生的 PDD 差異較小。而實際 3.5G HSDPA 網路由於頻寬較小、封包傳送路徑較長、以及網路傳輸延遲較大等因素，所以產生的 PDD 差異較大，並且超過一倍。

另一方面，如果 ***UDP-Basic*** 的存取網路從 802.11g 改為 HSDPA，PDD 會增加 704%的延遲，若為 ***TCP-Basic*** 則會增加 1424%。***UDP-Basic*** 與 ***TCP-Basic*** 相比，***UDP-Basic*** 由於傳送的訊息數量較少，所以當存取網路由較快的 802.11g 改為較慢的 HSDPA 時，影響 PDD 的程度相對也較小。而 ***TCP-Basic*** 因為傳送的訊息數量較多，加上 TCP 需要可靠傳輸的回應確認，所以當存取網路由較快的 802.11g 改為較慢的 HSDPA 時，對 PDD 的影響程度相對較大，增加一倍的 PDD 差異。在表 4.27 的啟用 QoS 協商與表 4.28 的啟用通話認證的綜合比較上，也可得到相同的結論。

表 4.27: 啟用 QoS 協商之 PDD 綜合比較

Test Item	PDD (ms)		
	802.11g	HSDPA	*Diff* (802.11g, HSDPA)
UDP-QoS	103.59	892	761%
TCP-QoS	120.74	2220	1739%
Diff (UDP-QoS, TCP-QoS)	17%	149%	-

表 4.28: 啟用通話認證之 PDD 綜合比較

Test Item	PDD (ms)		
	802.11g	HSDPA	*Diff* (802.11g, HSDPA)
UDP-Auth	63.76	583	814%
TCP-Auth	75.29	1574	1991%
Diff (UDP-Auth, TCP-Auth)	18%	170%	-

透過上述的綜合比較，本論文觀察 802.11g 網路上若採用不同傳輸協定 (UDP 與 TCP) 來建立通話，對 PDD 的影響在 17%~22%。而在實際的 HSDPA 網路上，對 PDD 的影響在 132%~170%。由此發現，在實際 HSDPA 網路上，傳輸協定對 PDD 的影響較為明顯，而使用 UDP 協定可比 TCP 協定減少一半以上的 PDD 延遲。(實驗測得 TCP 協定較 UDP 協定多出 132%~170%的 PDD 延遲)

接著表 4.29 將本論文提及的三種 TLS 安全機制實驗項目進行比較。

表 4.29: TLS 安全機制之 PDD 比較

Test Item	PDD (ms)		
	802.11g	HSDPA	*Diff* (802.11g, HSDPA)
TLS-A	111.19	1528	1274%
TLS-B	162.98	2560	1471%
TLS-A+SIP Auth	202.92	2729	1245%

由表 4.29 發現，***TLS-A*** 結合 SIP 通話認證無論在 802.11g 或 HSDPA 網路下，產生的 PDD 皆為三者中最大。另一方面，表 4.29 也發現當存取網路由 802.11g 改為 HSDPA 時，***TLS-A*** 結合 SIP 通話認證的 PDD 差異為 1245%，竟是三者中最小。

第五章　結論與未來工作

本論文在 Windows Mobile 行動通訊平台上，實作了一個支援 3GPP IMS 與 IETF SIP 兩種網路架構的 VoIP 軟體 SIP/IMS UA。藉由 SIP/IMS UA 對多種服務功能的支援，測量 UE 與核心網路之間的通話建立效能。本論文也分析了不同傳輸協定以及啟用通話認證、QoS 協商與 TLS 安全機制下的通話建立信令程序。並於可控制無線區域網路與實際行動通訊網路下進行測量，研究與分析這些程序所產生的 PDD。最後將觀察結果於第四章實驗中報告，其中包括三個發現：

1. 通話建立若要進行身份認證，在單向認證使用 SIP 通話認證會較 TLS 認證有較小的 PDD。而在雙向認證上，使用 TLS 雙向認證會較使用 TLS 單向認證結合 SIP 通話認證有較小的 PDD。
2. 在實際 3.5G HSDPA 網路上建立通話，使用 UDP 傳輸協定，會比 TCP 傳輸協定減少約一半的 PDD 延遲。
3. 在實際 3.5G HSDPA 網路上，發送封包數量的增加，會比封包大小的增加對 PDD 有較大的影響。

目前 SIP/IMS UA 已可在 3GPP IMS 與 IETF SIP 網路架構下，提供網路電話與即時訊息服務功能。未來可在 SIP/IMS UA 上開發影像功能，如視訊電話與網路電視，並可進行各種相關實驗與測量，提供行動服務業者建置多媒體服務時的參考。除此之外，未來也可於實際 3.5G HSDPA 網路下，試著使用信令壓縮等技術，來減少信令傳送的大小。測量 UA 軟體在此網路中能否有更好的效能表現。

附錄 A　平均 PDD 值與實驗次數關係

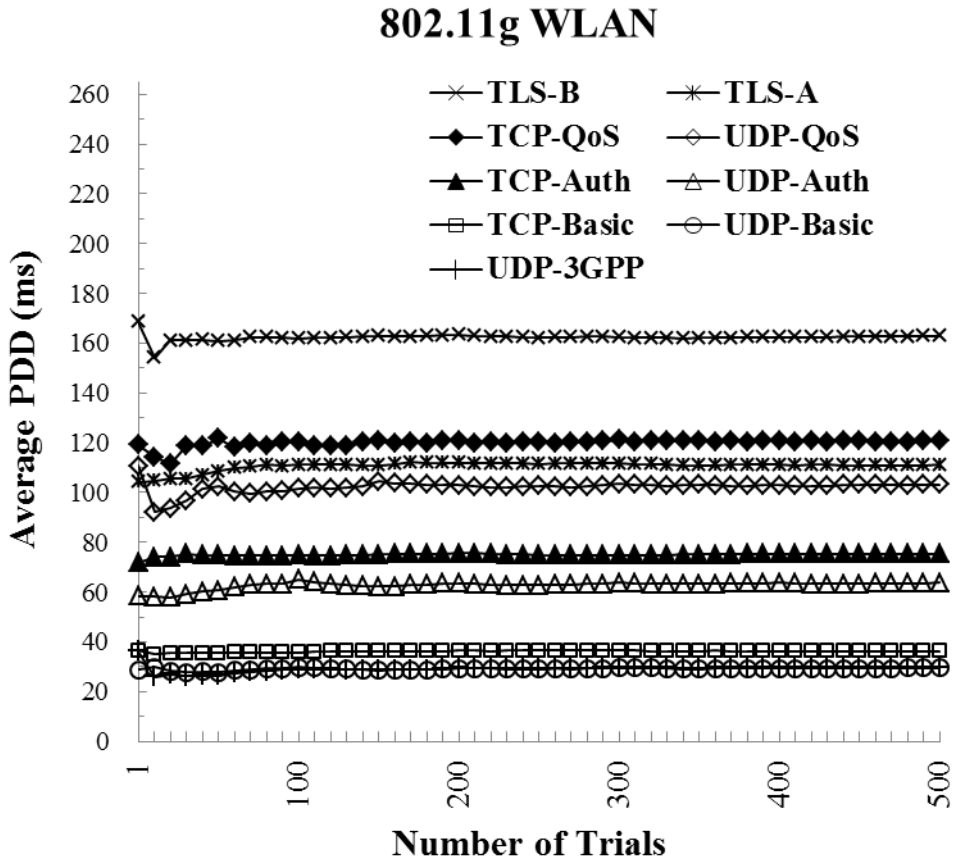

圖 A.1: 平均 PDD 值與實驗次數關係圖

為了評估足夠的實驗測試次數，本論文觀察所有實驗項目的平均 PDD 值與實驗次數的關係。IEEE 802.11g 無線網路環境下的實驗結果如圖 A.1 所示，本論文觀察到實驗次數逐漸增加後，平均 PDD 值會慢慢收斂。在實驗次數到達 500 次時，之後每次得到的平均 PDD 值最大差異會小於 0.1%。根據此實驗結果，本論文採用 500 做為實驗的測試次數。

附錄 B　實驗環境基本傳輸效能評估

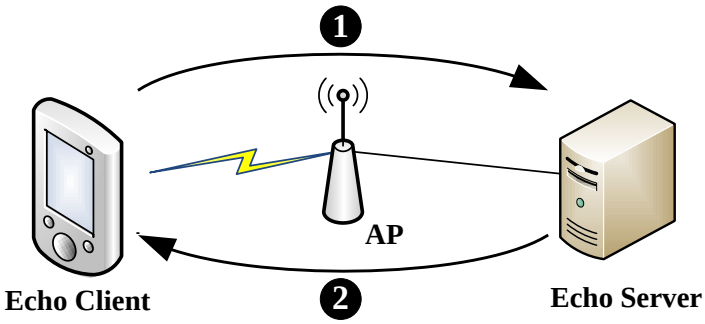

圖 B.1: Echo Client 與 Server 測量程序

在無線網路中，使用不同規格設備或不同環境架設方式，會造成傳輸效能的差異。因此，本論文利用 Round Trip Time (RTT) 來評估目前網路環境的基本傳輸效能，用於驗證實驗測量結果的正確性。本論文利用 Echo Client 與 Server 來測量 RTT，測量程序如圖 B.1 所示，封包之傳輸路徑為圖①→②。每次的測量都由 Client 發送固定大小封包給 Server，Server 在收到封包後會立即回傳相同封包給 Client。依照此步驟測量 100,000 次，由 Client 記錄 RTT 測量平均值。表 B.1 為本論文無線網路實驗環境下，三種不同大小封包測量之 RTT 結果。將 RTT 除以 2 則可算出單向封包傳送的時間。

表 B.1: 802.11g WLAN 實驗環境之 RTT 測量

Packet Length (bytes)	512	768	1024
RTT (ms)	2.392	2.716	2.962

附錄 C　實驗測量結果驗證

附錄 C 驗證了第四章實驗的測量結果。各實驗項目驗證方法如下：

實驗一： 此部份驗證了 ***UDP-Basic*** 與 ***UDP-3GPP*** 之 PDD 與 D_1 正確性。本論文於表 4.6 已測得 ***UDP-Basic*** 與 ***UDP-3GPP*** 之 PDD 與 D_1。利用此結果可計算出兩實驗項目的 PDD – D_1，分別得到 25.79 與 25.82 ms，差距 0.03 ms。表 4.4 中已知 ***UDP-Basic*** 與 ***UDP-3GPP*** 的訊息流程差別只在於 UAC 傳送的 INVITE 訊息長度 (分別為 683 與 806 bytes)，所以 ***UDP-Basic*** 與 ***UDP-3GPP*** 的 PDD – D_1 差距值 0.03 ms 可表現出兩實驗項目傳送 INVITE 訊息的傳輸時間差距。接著本論文使用附錄 B 方法進行驗證，首先分別測量此無線網路實驗環境下傳送封包 683 與 806 bytes 的平均時間，測得 1.38 與 1.43 ms，並相減推算出兩封包傳送時間差距為 0.05 ms。先前算出 ***UDP-Basic*** 與 ***UDP-3GPP*** 傳送 INVITE 訊息的時間差距 0.03 ms 應該與此值相等。比較之下 0.03 ms 與 0.05 ms 誤差為 0.02 ms，此誤差在可接受範圍，故推論本論文測得之 PDD 與 D_1 為合理值。

實驗二：此部份驗證了 ***TCP-Basic*** 之 PDD 與 D_1 正確性。本論文於表 4.9 已測得 ***TCP-Basic*** 之 PDD、D_1 與 D_2。利用此結果可計算 ***TCP-Basic*** 之 PDD – D_1 – D_2，得到 26.12 ms。此值為 UAC 發送 INVITE 訊息與接收 180 Ringing 訊息所產生的延遲總和。將此值與先前 ***UDP-Basic*** 的 PDD – D_1 計算結果 (25.79 ms) 相比，兩數值差異為 1.28%。理論上此兩數值應非常相近，本論文測量結果差異為 1.28%，在可接受的差異範圍內，加上已知 D_2 為合理值，故可推論 ***TCP-Basic*** 之 PDD 與 D_1 同為合理值。

實驗三： 此部份驗證了 ***UDP-QoS*** 與 ***TCP-QoS*** 之 PDD 與 D_1 正確性。本論文於表 4.12 已測得 ***UDP-QoS*** 與 ***TCP-QoS*** 之 PDD 與 D_1，再利用實驗 2 測得之 TCP 三向交握延遲 D_2 (6.23 ms)，來計算兩實驗項目的 PDD – D_1 – D_2。結果分別為 25.91 與 26.16 ms，差異 0.965%。理論上此差異值應該非常小，差異 0.965%在可接受範圍，故推論本論文測得之 PDD 與 D_1 皆為合理值。

實驗四： 此部份驗證了 ***UDP-Auth*** 與 ***TCP-Auth*** 之 PDD 與 D_1 正確性。本論文於表 4.15 已測得 ***UDP-Auth*** 與 ***TCP-Auth*** 之 PDD 與 D_1。再利用實驗 2 測得之 TCP 三向交握延遲 D_2 (6.23 ms)，來計算兩實驗項目的 PDD – D_1 – D_2，結果分別為 25.67 與 25.75 ms，差異 0.312%。理論上此差異值應該非常小，差異 0.312%在可接受範圍，故推論本論文測得之 PDD 與 D_1 皆為合理值。

實驗五： 此部份驗證了 ***TLS-A*** 與 ***TLS-B*** 之 PDD 正確性。本論文由第 4.5.1 節已知兩實驗項目訊息流程差別只有 TLS 交握不同，所以若不考慮 TLS 交握，兩者的延遲應該相等。本論文於表 4.19 已測得 ***TLS-A*** 與 ***TLS-B*** 之 PDD 與 TLS 交握延遲 D_3，將 PDD – D_3 來推算兩實驗項目不考慮 TLS 交握下的延遲。計算結果分別為 44.56 與 44.52 ms，差異 0.09%。理論上兩延遲值應該相等，差異 0.09%在可接受範圍，故推論本論文測得之 PDD 結果為合理值。

參考文獻

[1] Lin, Y.-B. and Pang, A.-C. *Wireless and Mobile All-IP Networks*. John Wiley & Sons, Inc., 2005.

[2] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley and E. Schooler, SIP: Session Initiation Protocol. RFC 3261, Internet Engineering Task Force, June 2002.

[3] D. Vingarzan and P. Weik. IMS Signaling over Current Wireless Networks: Experiments Using the Open IMS Core. Vehicular Technology Magazine, IEEE, March 2007.

[4] D. Waiting, R. Good, R. Spiers, and N. Ventura, "Open Source Development Tools for IMS Research," 2008 4th International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities, March 2008.

[5] fring. http://www.fring.com/

[6] Nokia N95. http://en.wikipedia.org/wiki/Nokia_N95

[7] T. Dierks and C. Allen. The TLS Protocol Version 1.0, RFC 2246, Internet Engineering Task Force, January 1999.

[8] Chan, Y.-H, *VoIP Quality Measurement for Multiple Codecs*. NCTU, July 2007.

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

[10] 3GPP. 3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; Communication HOLD (HOLD) using IP Multimedia (IM) Core Network (CN) subsystem. Technical Specification 3G TS 24.610 version 8.3.0 (2009-03), 2009.

[11] 3GPP. 3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; Explicit Communication Transfer (ECT) using IP Multimedia (IM) Core

Network (CN) subsystem. Technical Specification 3G TS 24.629 version 8.2.0 (2009-03), 2009.

[12] P. Srisuresh and M. Holdrege. IP Network Address Translator (NAT) Terminology and Considerations. RFC 2663, IETF, August 1999.

[13] J. Rosenberg. A Session Initiation Protocol (SIP) Event Package for Registrations. RFC 3680, Internet Engineering Task Force, March 2004.

[14] J. Rosenberg and Schulzrinne. H. Reliability of Provisional Responses in Session Initiation Protocol (SIP), RFC 3262, Internet Engineering Task Force, June 2002.

[15] J. Rosenberg, R. Mahy, P. Matthews, and D. Wing. Session Traversal Utilities for NAT (STUN). RFC 5389, Internet Engineering Task Force, October 2008.

[16] G. Camarillo, W. Marshall, and J. Rosenberg, Integration of Resource Management and Session Initiation Protocol (SIP), RFC 3312, Internet Engineering Task Force, October 2002.

[17] The eXtended osip library. http://savannah.nongnu.org/projects/exosip

[18] The GNU oSIP library. http://www.gnu.org/software/osip/.

[19] OpenSSL: Cryptography and SSL/TKS Toolkit. http://www.openssl.org/

[20] 3GPP. 3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; IP multimedia call control protocol based on Session Initiation Protocol (SIP) and Session Description Protocol (SDP); Stage 3. Technical Specification 3G TS 24.229 version 8.7.0 (2009-03), 2009.

[21] M. Garcia-Martin, E. Henrikson and D. Mills. Private Header (P-Header) Extensions to the Session Initiation Protocol (SIP) for the 3rd-Generation Partnership Project (3GPP). RFC 3455, IETF, January 2003.

[22] J. Peterson. A Privacy Mechanism for the Session Initiation Protocol (SIP). RFC 3323,

Internet Engineering Task Force, November 2002.

[23] C. Jennings, J. Peterson, and M. Watson. Private Extensions to the Session Initiation Protocol (SIP) for Asserted Identity within Trusted Networks. RFC 3325, Internet Engineering Task Force, November 2002.

[24] M. Handley, V. Jacobson and C. Perkins. SDP: Session Description Protocol. RFC 4566, Internet Engineering Task Force, June 2006.

[25] H. Schulzrinne, S. Casner, R. Frederick and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. RCF 3550, IETF, July 2003.

[26] G. Camarillo, R. Penfield, A. Hawrylyshenm, and M. Bhatia. Requirements from SIP (Session Initiation Protocol) Session Border Control Deployments. Internet-Draft, January 2009. draft-ietf-sipping-sbc-funcs-08.

[27] STUN Client and Server library. http://sourceforge.net/projects/stun/

[28] The Sippy RTPproxy. http://www.rtpproxy.org/

[29] JRTPLIB. http://research.edm.uhasselt.be/~jori/page/index.php?n=CS.Jrtplib

[30] International Telecommunication Union. Service Quality Assessment for Connection Set-up and Release Delays. Recommendation E.431, Geneva, 1992

[31] 3GPP. 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; IP Multimedia Subsystem (IMS); Stage 2. Technical Specification 3G TS 23.228 version 8.8.0 (2009-03), 2009.

[32] J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, E. Sink and L. Stewart. HTTP Authentication: Basic and Digest Access Authentication, RFC 2617, Internet Engineering Task Force, June 1999.

[33] A. Niemi, J. Arkko, and V. Torvinen. Hypertext Transfer Protocol (HTTP) Digest Authentication Using Authentication and Key Agreement (AKA), RFC 3310, Internet

Engineering Task Force, September 2002.

[34] 3GPP. 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; 3G security; Security architecture. Technical Specification 3G TS 33.102 version 8.2.0 (2009-03), 2009.

[35] S. Kent and K. Seo. Security Architecture for the Internet Protocol, RFC 4301, Internet Engineering Task Force, December 2005.

[36] The Open Source IMS Core project. http://www.openimscore.org/

[37] Wireshark. http://www.wireshark.org/