國立交通大學

網路工程研究所

碩 士 論 文

手機之自動以事件驅動指定轉接機制

Automatic Event-Triggered Call Forwarding Mechanism for Mobile Phones

研 究 生：吳政翰

指導教授：林一平　教授

中 華 民 國 一 〇 〇 年 七 月

手機之自動以事件驅動指定轉接機制
Automatic Event-Triggered Call Forwarding Mechanism for Mobile Phones

研究生：吳政翰　　Student：Zheng-Han Wu

指導教授：林一平 博士　　Advisor：Dr. Yi-Bing Lin

國立交通大學

網路工程研究所

碩士論文

A Thesis

Submitted to Institute of Network Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

July 2011

Hsinchu, Taiwan, Republic of China

中華民國一〇〇年七月

手機之自動以事件驅動指定轉接機制

學生：吳政翰　　指導教授：林一平博士

國立交通大學網路工程研究所碩士班

摘　要

指定轉接 (Call Forwarding) 是一項傳統的電信服務。該服務允許使用者將手機來電轉接至另一個電話號碼。但是指定轉接這項服務需要使用者手動地去設定啟用或是取消，這造成了使用上地不方便。為解決此問題，本篇論文針對手機端提出了一種自動的「指定轉接演算法」(Call Forwarding Algorithm；CFA)。只要在智慧型手機上面安裝一支 CFA 軟體，在某些事件發生時，例如當手機插上充電器或關機時，即可自動啟用指定轉接這項服務。而當手機拔除充電器或開機時，則可以自動取消指定轉接這項服務。此外，我們也從數學分析、模擬實驗與實際測量這三方面來評估「指定轉接演算法」的效能。我們的評估指出「指定轉接演算法」是可以被實際應用在商業用途上的。

關鍵字：指定轉接，電信，UMTS

Automatic Event-Triggered Call Forwarding Mechanism for Mobile Phones

Student: Zheng-Han Wu Advisor: Dr. Yi-Bing Lin

Institute of Network Engineering

National Chiao Tung University

ABSTRACT

Call forwarding is a traditional telecom service that allows a user to forward incoming calls to another telephone number. This service requires the user to manually activate and deactivate the feature, and therefore may not be very convenient. This paper proposes an automatic Call Forwarding Algorithm (CFA) for mobile phones. By installing a software in a smartphone, call forwarding is automatically triggered (e.g., when the phone is plugged in a charger or is turned off) or disabled (e.g., when the phone is unplugged from the charger or is turned on). We also investigate the performance of CFA through analytic analysis, simulation, and measurement. Our study indicates that CFA is very feasible for commercial usage.

Index Terms: Call Forwarding, mobile telecom, UMTS

誌　謝

首先我要感謝我的指導老師林一平博士。在兩年的碩士班生活中，老師認真熱情的研究態度，以及細心嚴謹的研究方法，一直是我學習的典範。從老師嚴格的教導與訓練中，我學習到了做事應有的多元思考方式。因為老師耐心的指導，我才能完成此篇論文。

接著我要感謝顏在賢博士、鄭枸澺博士、與宋映蓉博士，你們在碩士班口試時給了我許多寶貴的建議。此外，我也要感謝實驗室所有的學長姊、學弟妹，還有這兩年一起奮鬥的好同學們。感謝你們給予我許多的指導、建議、支持與幫忙。這兩年一起奮鬥的時光非常快樂。

最後，我也要感謝我的家人。因為有你們全力的支持與鼓勵，我才能無後顧之憂地專心做研究，完成此篇論文。

目錄

圖目錄

符號說明

t_c： 通話到達間隔時間 (inter-call arrival time)

t_a： CFA 啟用程序之延遲時間

τ_c： CFA 開始執行啟用程序與下一通來電到達的間隔時間

p_c： $\tau_c < t_a$ 的機率

T： 門檻值

p_s： $t_a < T$ 的機率

第一章　簡介

當一個手機用戶回到家時，他可能會將他的手機 (User Equipment；UE) 關機或是插上充電器。此時這個使用者雖然在家，但是他卻有可能因為沒有待在手機附近而導致了漏接電話的情況發生。在這個案例中，假如別人打來的電話能自動地被轉接至家裡的座機電話跟分機，則可以讓使用者避免因為離手機過遠而漏接了電話。傳統電信的指定轉接服務能讓使用者手動去設定啟用，但是這手動設定的方式是一道很瑣碎麻煩的程序。此外，當使用者不需要指定轉接時，他們也常常會因為忘記取消指定轉接這項服務的設定而造成了手機沒辦法正常地接收來電通話。

為了解決這個問題，我們提出了一個「自動以事件觸發之指定轉接演算法」(Call Forwarding Algorithm；CFA) 的軟體。使用者可以很輕易地在智慧型手機上安裝此軟體。而且我們的這個方法完全不需要去修改更動現存的電信網路設備。

1.1　指定轉接演算法 (CFA) 之概念

我們首先介紹 CFA 的概念以及在 Microsoft Windows CE (WinCE) 平台上實作 CFA 的方法。我們的 CFA 方法包含以下四個部分：

Part 1. 偵測觸發事件: 「當手機關機」或是「當手機插上充電器」的事件發生時，CFA 程式能自動地偵測這些觸發事件並啟用指定轉接服務。我們使用 WinCE 的 RegistryNotifyCallback 函式 [1] 來監測電池狀態，以實作自動偵測觸發事件的功能。此外我們也可將「手機收到一則特殊簡訊」的情況當作是另一種觸發事件 (這部份會在第四章再加以介紹)。

Part 2. 選擇轉接號碼 (即手機來電會被轉接至此電話號碼)**:** 當偵測到觸發事件時，手機會選擇對應的轉接號碼 (forwarded-to number)。而且此選擇號碼的功能也可以與定位服務 (location service) 搭配應用。例如家裡座機的電話號碼能與家的

Global Positioning System (GPS) 定位資訊做聯結。此時手機可以藉由 Assisted GPS (A-GPS) 得知目前所在的位置，並根據此位置來選擇對應的轉接號碼。但是測量位置時的誤差有可能導致定位不確定 (location ambiguity) 的情況發生。這將使得手機難以單從位置資訊來選擇轉接號碼。當此種情況發生時，手機會詢問使用者，請使用者從可能的號碼中選取一個以當做轉接號碼。假如沒有發生上述的定位誤差時，手機則可以自動選擇轉接號碼，不需要再去詢問、打擾使用者。

Part 3. 啟用指定轉接： 當選擇好轉接號碼之後，手機可以自動向電信網路執行標準的指定轉接註冊程序。我們使用 WinCE 的 lineForward 函式 [1] 來實作此功能。

Part 4. 取消指定轉接： 當啟用指定轉接的觸發事件消失時，我們需要取消指定轉接這項服務。例如當手機拔除充電器時，手機會自動向電信網路執行標準指定轉接刪除程序來取消指定轉接服務。同上面的 Part 3，我們也是使用 lineForward 函式來實作取消指定轉接服務的功能。

1.2 執行 CFA 之電信網路環境

我們使用簡化的 *Universal Mobile Telecommunications System* (UMTS) 網路架構來舉例說明指定轉接服務是如何運作的 [2][3]。根據圖 1.1 的架構圖，一個手機使用者 (UE1；圖 1.1 (1)) 藉由和基地台間的無線通訊與 serving *Mobile Switching Center* (MSC)/*Visitor Location Register* (VLR；圖 1.1 (2)) 連接，以接受電信服務。MSC 與 VLR 分別負責處理通話程序 (call processing) 與行動管理 (mobility management)。每一支手機都被指派了一個 E.164 行動電話號碼 (例如 UE1 的 0911111111)，而且每一個號碼都會對應到一個 *Gateway Mobile Switching Center* (GMSC；圖 1.1 (3))。換句話說，每一通撥打給 UE1 的電話，都會先被導向 UE1 所對應的 GMSC。此處的 *Home Location Register* (HLR；圖 1.1 (4)) 是一個可以提供手機使用者目前所屬的 MSC/VLR 位置的資料庫。而 MSC/GMSC 連接到公共電話網路 (Public Switched Telephone Network；PSTN；圖 1.1 (5))。PSTN 裡面的 *Service Switching Points* (SSPs；圖 1.1 (6) 跟 (7)) 則是支援通話程序的電話交換機。

基於這張架構圖，我們解釋使用中華電信無條件指定轉接服務 (call forwarding unconditional service) [3] 的 CFA 訊息流程 (**Part 3.** 與 **Part 4.**)。

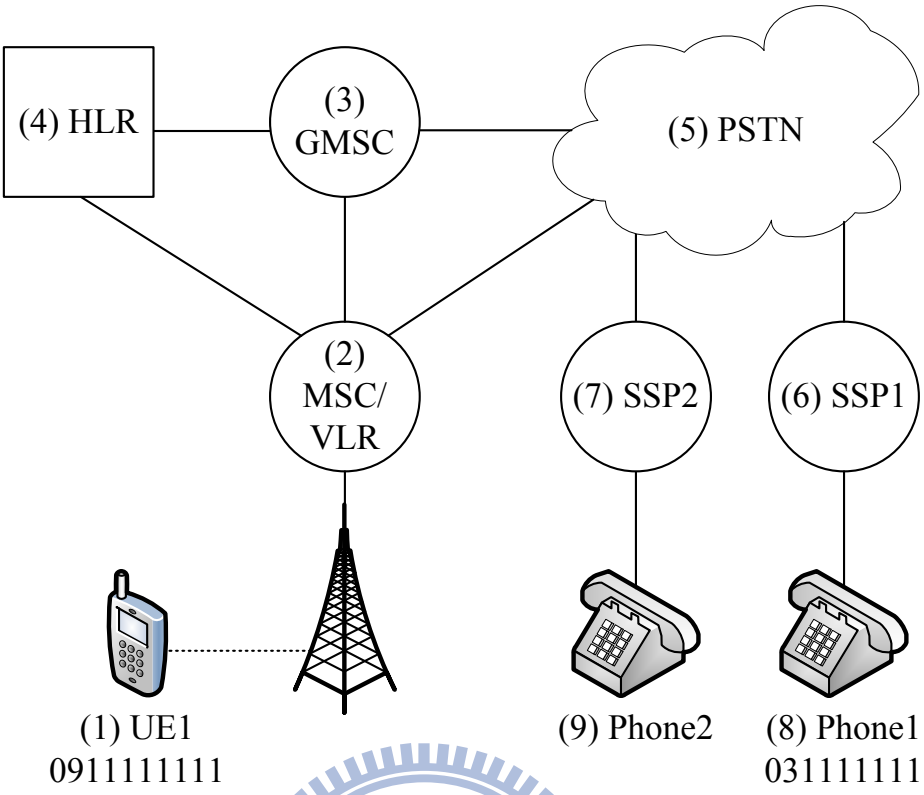


圖 1.1: 指定轉接之網路架構圖

1.3 論文章節架構

本篇論文針對手機端提出了一種自動的「指定轉接演算法」(Call Forwarding Algorithm；CFA)。我們的方法只要在智慧型手機上面安裝一支 CFA 軟體。當某些事件發生時，例如當手機插上充電器或關機時，即可自動啟用指定轉接這項服務。而當手機拔除充電器或開機時，則可以自動取消指定轉接這項服務。此外，我們也從數學分析、模擬實驗與實際測量這三方面來評估「指定轉接演算法」的效能。我們的評估指出「指定轉接演算法」是可以被實際應用在商業用途上的。本論文章節架構描述如下：

1. 第二章介紹 CFA 之訊息流程 (message flow)，並以數學分析 CFA 之延遲時間。
2. 第三章介紹 CFA 通知程序與 CFA 啟用程序之失敗偵測，接著亦以數學分析探討其效能。
3. 第四章介紹 CFA 與車載資訊 (Telematics) 之應用。
4. 第五章為本篇論文之總結，並提出未來工作方向。

第二章　CFA 之訊息流程 (Message Flow)

此章節介紹 CFA 啟用程序 (activation procedure)、來電通話建立程序 (incoming call setup procedure)、與取消程序 (deactivation procedure) 的訊息流程。我們假設 UE1 (user 1 的 UE) 已經安裝了 CFA 軟體，並且選擇 Phone1 (user 1 的座機；圖 1.1 (8)) 的 031111111 當做轉接號碼。即啟用指定轉接服務之後，UE1 的來電會被導至 Phone1。接著我們會推導在 CFA 啟用程序期間發生來電的機率，以觀察 CFA 效能。

2.1　CFA 啟用程序 (Activation Procedure)

當 user 1 將 UE1 插上充電器時，UE1 上的 CFA 程式會偵測到電池狀態已變為充電中。接著 CFA 啟用程序會自動執行標準 *3rd Generation Partnership Project* (3GPP) 指定轉接註冊程序 [3][4]。我們藉由圖 2.1 與下述兩步驟來說明 CFA 啟用程序：

Step A.1. 當 WinCE RegistryNotifyCallback 函式偵測到充電狀態時，CFA 會自動撥打一特殊號碼 **21*031111111#。此處的 21 為中華電信無條件指定轉接服務的服務號碼。而 031111111 代表轉接號碼。其他電信業者的指定轉接撥號方式類似，此處就不多做介紹。MSC/VLR (圖 1.1 (2)) 藉由 *Signaling System Number 7* (SS7) 的信令 MAP_REGISTER_SS request 送給 HLR (圖 1.1 (4)) 一指定轉接註冊請求。該請求指示 UE1 要啟用無條件指定轉接服務，並將轉接號碼設定為 031111111。

Step A.2. HLR 檢查 UE1 是否被允許啟用指定轉接服務。假如允許，則 HLR 會儲存轉接號碼。並回傳 SS7 MAP_REGISTER_SS response，通知 MSC/VLR 此註冊程序成功。否則 HLR 回傳一錯誤訊息。

當程序成功完成，所有 UE1 的來電都會被轉接至 Phone1。

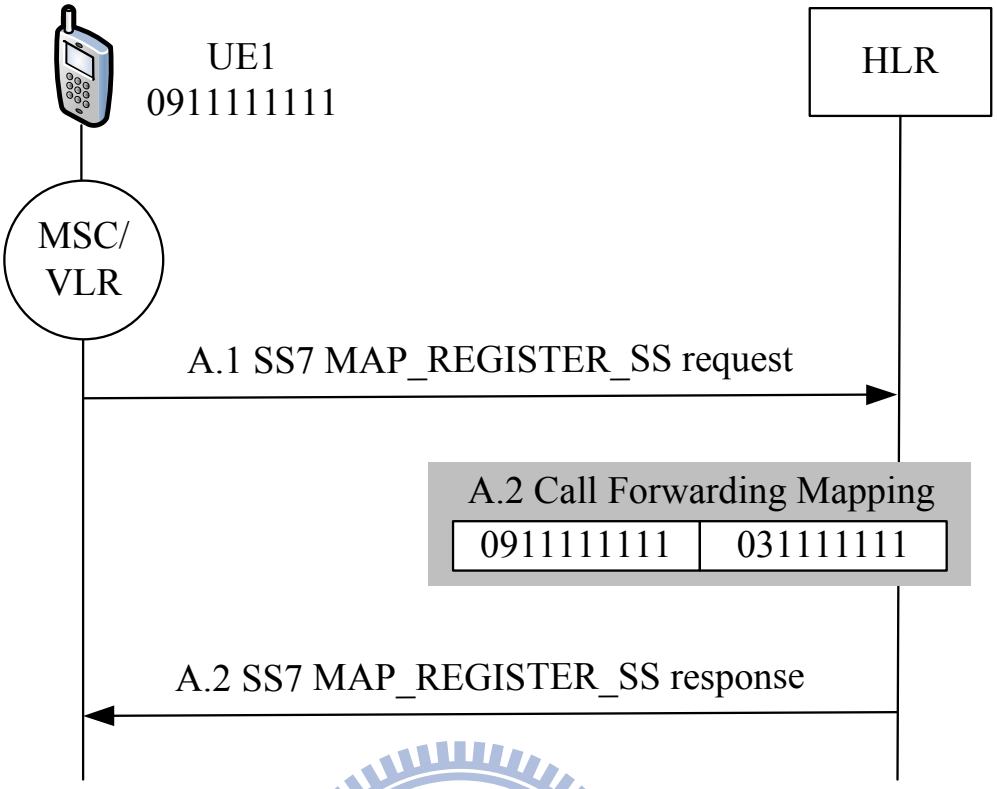


圖 2.1: CFA 啟用程序

2.2 來電通話建立程序 (Incoming Call Setup Procedure)

在 user 1 啟用了指定轉接服務之後，假如 user 2 (圖 2.2 (9)) 撥打 user 1 的手機號碼 091111111，此時的通話建立程序如同圖 2.2 與下列步驟說明：

Step B.1. SSP2 (圖 1.1 (7)) 發出 SS7 信令 *Initial Address Message* (IAM) 訊息給 091111111 (UE1) 對應的 GMSC (圖 1.1 (3))。

Step B.2. GMSC 藉由 SS7 MAP_SEND_ROUTING_INFORMATION request 向 HLR 詢問此通通話的路由 (routing) 資訊。

Step B.3. HLR 回覆一 SS7 MAP_SEND_ROUTING_INFORMATION response。此訊息包含服務轉接號碼 031111111 (Phone1) 的 SSP1 的 routing number (SS7 地址)。

Step B.4. GMSC 把 SS7 IAM 訊息轉送給 SPP1。

Step B.5. SSP1 對 Phone1 震鈴，並且將 SS7 *Address Complete Message* (ACM) 訊息經由 GMSC 回傳給 SSP2。

Step B.6. 當 user 1 接起 Phone1，SSP1 將 SS7 *Answer Message* (ANM) 訊息經由 GMSC 送給 SSP2。

在 Step B.6 完成後，user 1 與 user 2 電話已接通，開始會話。

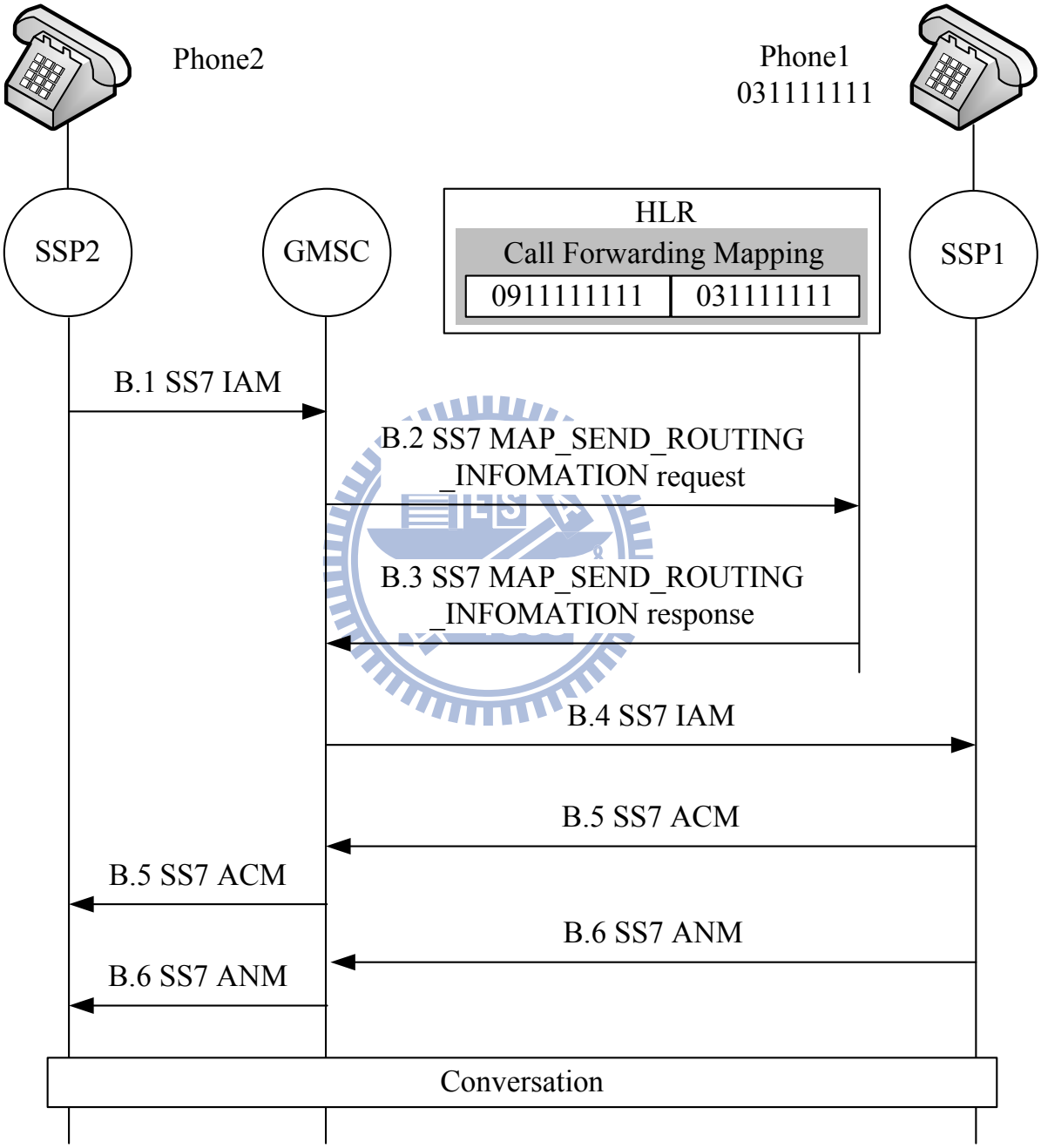


圖 2.2: 來電通話建立程序

2.3 CFA 取消程序 (Deactivation Procedure)

當 user 1 拔除 UE1 的充電器時，UE1 上的 CFA 程式會偵測到此事件並自動執行 CFA 取消程序。此 CFA 取消程序係依據 3GPP 指定轉接刪除程序 [3][4] 來取消指定轉接服務。我們以圖 2.3 跟以下兩步驟來介紹 CFA 取消程序：

Step C.1. 與 Step A.1 相似，當 WinCE RegistryNotifyCallback 函式偵測到拔除 UE1 充電器的觸發事件時，UE1 的 CFA 程式會自動撥打 ##21# 此特殊號碼。接著 MSC/VLR 送 SS7 MAP_ERASE_SS request 給 HLR，指示 UE1 想要取消指定轉接服務。

Step C.2. HLR 移除 UE1 對應的轉接號碼，並回覆 SS7 MAP_ERASE_SS response 告知此刪除程序成功。

在 CFA 取消完成後，所有 UE1 的來電不會再轉接至 Phone1，而會恢復導向 UE1。

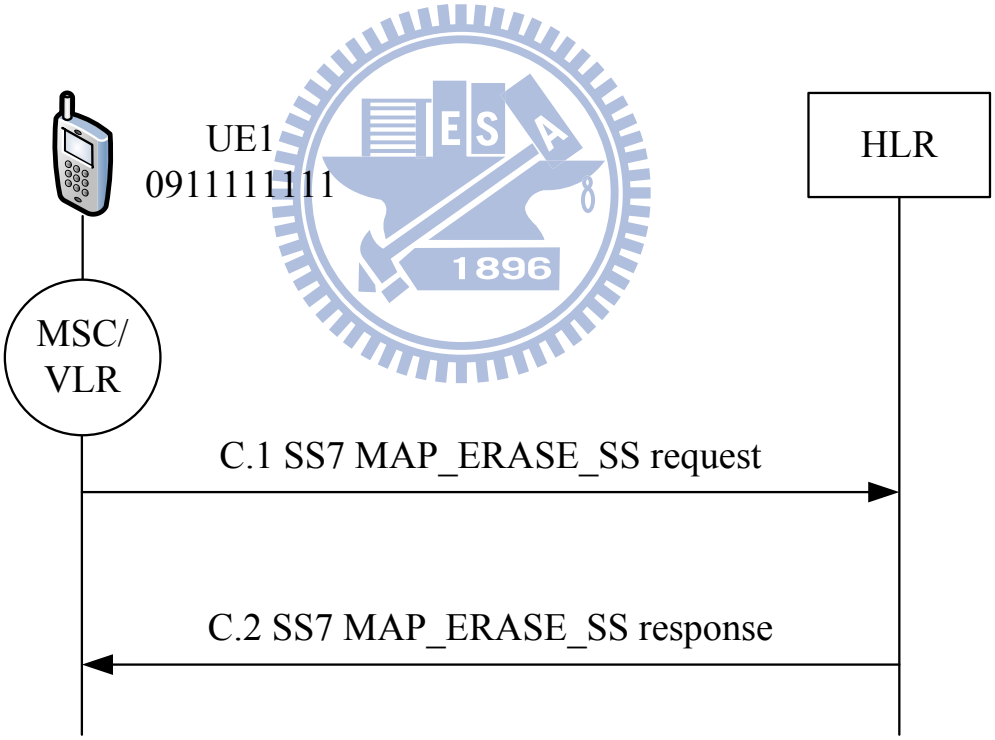


圖 2.3: CFA 取消程序

2.4 CFA 延遲分析

User 1 將 UE1 插上充電器之後，他會認為 UE1 的來電會被轉接至 Phone1。然而來電有可能在 CFA 完成啟用程序前就到達 UE1。在這個案例中，假如 user 1 預期當有來電時 Phone1 會響鈴，則他不會察覺到此通 UE1 的來電。直到 user 1 拔除 UE1 的充電器時，他才會發現漏接了此通電話。我們會以數學分析證明，因為上述情況而發生未接來電的機率是非常低的，因此可以被忽略。此外，我們也可以使用一通知機制 (notification mechanism) 來解決此問題。我們在第三章再加以介紹此通知機制。

假設 p_c 為來電在 CFA 啟用程序執行時 (在 3GPP 指定轉接註冊程序完成前) 到達 UE1 的機率。很明顯地，對使用者來說， p_c 值越小越好。

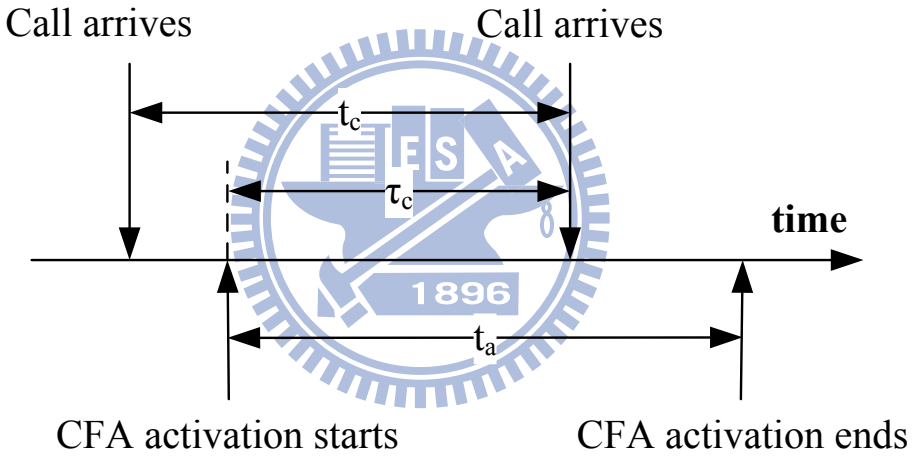


圖 2.4: 推導 p_c 之時間圖

圖 2.4 顯示推導 p_c 的時間圖。假設 t_c 為通話到達間隔時間 (inter-call arrival time) 且 t_a 為 CFA 啟用程序 (圖 2.1 的 Step A.1-A.2) 之延遲時間。則 τ_c 為 CFA 開始執行啟用程序與下一通來電到達的間隔時間。我們稱此 τ_c 為通話到達間隔時間的剩餘時間 (excess life)。故 p_c 為 $\tau_c < t_a$ 的機率。

假設 t_c 是平均值 (mean) 為 $1/\lambda$ 的指數分佈 (exponential distribution)，即來電到達情況為帕松過程 (Poisson process)。且 t_a 為任一隨機分佈 (arbitrary distribution)，其密度函數 (density function) 與拉普拉斯轉換 (Laplace transform) 分別為 $f_a(\cdot)$ 與 $f_a^*(s)$。根據指數分佈的無記憶 (memoryless) 特性， τ_c 和 t_c 的分佈相同，皆為同一個指數分佈。我們可

以推導 p_c 為：

$$\begin{aligned} p_c &= Pr[\tau_c < t_a] \\ &= \int_{t_a=0}^{\infty} f_a(t_a) \int_{\tau_c=0}^{t_a} \lambda e^{-\lambda \tau_c} d\tau_c dt_a \\ &= 1 - f_a^*(\lambda) \end{aligned} \tag{2.1}$$

假如 t_a 為伽馬隨機變數 (Gamma random variable)，則其拉普拉斯轉換為 $f_a^*(s) = \left(\frac{\mu}{s+\mu}\right)^k$。此處的 k 為形狀參數 (shape parameter)，而 μ 為速率參數 (rate parameter)。則方程式 (2.1) 可以改寫為

$$p_c = 1 - \left(\frac{\mu}{\lambda+\mu}\right)^k \tag{2.2}$$

我們使用伽馬分佈 (Gamma distribution) 是因為此分佈在電信模型 (telecom modeling) 中被廣泛地使用 (請見 [5][6] 與其參考文獻)。我們使用蒙地卡羅模擬法 (Monte Carlo simulation) 來驗證方程式 (2.2)。此模擬法產生延遲時間 τ_c 與 t_a，再藉由比較此二種延遲時間的長度來得出 p_c 的值。此模擬實驗顯示方程式 (2.2) 與模擬結果的誤差在 0.2% 以內。

我們也測量了中華電信商用 UMTS 系統的 t_a 值。我們在一型號為 CHT 9110、作業系統為 Microsoft Windows Mobile 6.0 的智慧型手機上面安裝 CFA 軟體。藉由執行 3000 次以上 CFA 啟用程序來收集延遲時間 t_a。我們得到 t_a 的平均值 $E[t_a] = 7.88266$ 秒，其變異數 (variance) 為 $V_a = E[t_a^2] - E[t_a]^2 = 0.0139717E[t_a]^2$ 的統計值。

根據測量到的平均值 $E[t_a]$ (即 7.88266 秒)，我們假設 $100E[t_a] \le E[t_c] \le 1000E[t_a]$ (即通話到達間隔時間範圍大約為 13 分鐘至 2.18 小時)。圖 2.5 依據 $E[t_c]/E[t_a]$ 與 V_a 的值標示出對應的 p_c 值 (來電在 CFA 啟用程序完成前到達 UE1 的機率)。此圖顯示當 $E[t_c]/E[t_a]$ 增加時 p_c 會減少。這結果很容易就可看出。此外我們也觀察到當 V_a 增加時 p_c 亦會減少。這一結果則不易看出。以下我們對此現象加以說明。針對一個固定的 $E[t_a]$ 值，當 V_a 增加時，短的 t_a 時段會比長的 t_a 時段增加更多。而短的 t_a 會使 $\tau_c < t_a$

的情況減少。因此，當 V_a 增加時， p_c 會減少。

測量顯示中華電信網路的 V_a 值非常小，而且有 0.1% 至 1% 的來電會在 CFA 啟用程序完成前到達 UE1。但是此種來電情況只有在 CFA 開始執行啟用程序的 8 秒以內才有可能發生，這時 user 1 仍然在 UE1 附近。因此，user 1 可以聽到手機響鈴而不會漏接這些電話。我們可以觀察到某些電信網路的 V_a 值非常大。巨大的 V_a 值會導致長的 t_a 出現。一個非常長的 t_a 值有可能會使得來電在 user 1 已經遠離了 UE1 之後 (但在 CFA 啟用程序完成之前) 才到達 UE1 的情況發生。在此案例中，使用者可能會因為沒有聽到手機響鈴而漏接了電話 (圖 2.5 顯示此機率 $p_c < 0.2\%$)。為了解決此問題，我們提出了一個 CFA 通知程序 (notification procedure)，並在第三章介紹此 CFA 通知程序。

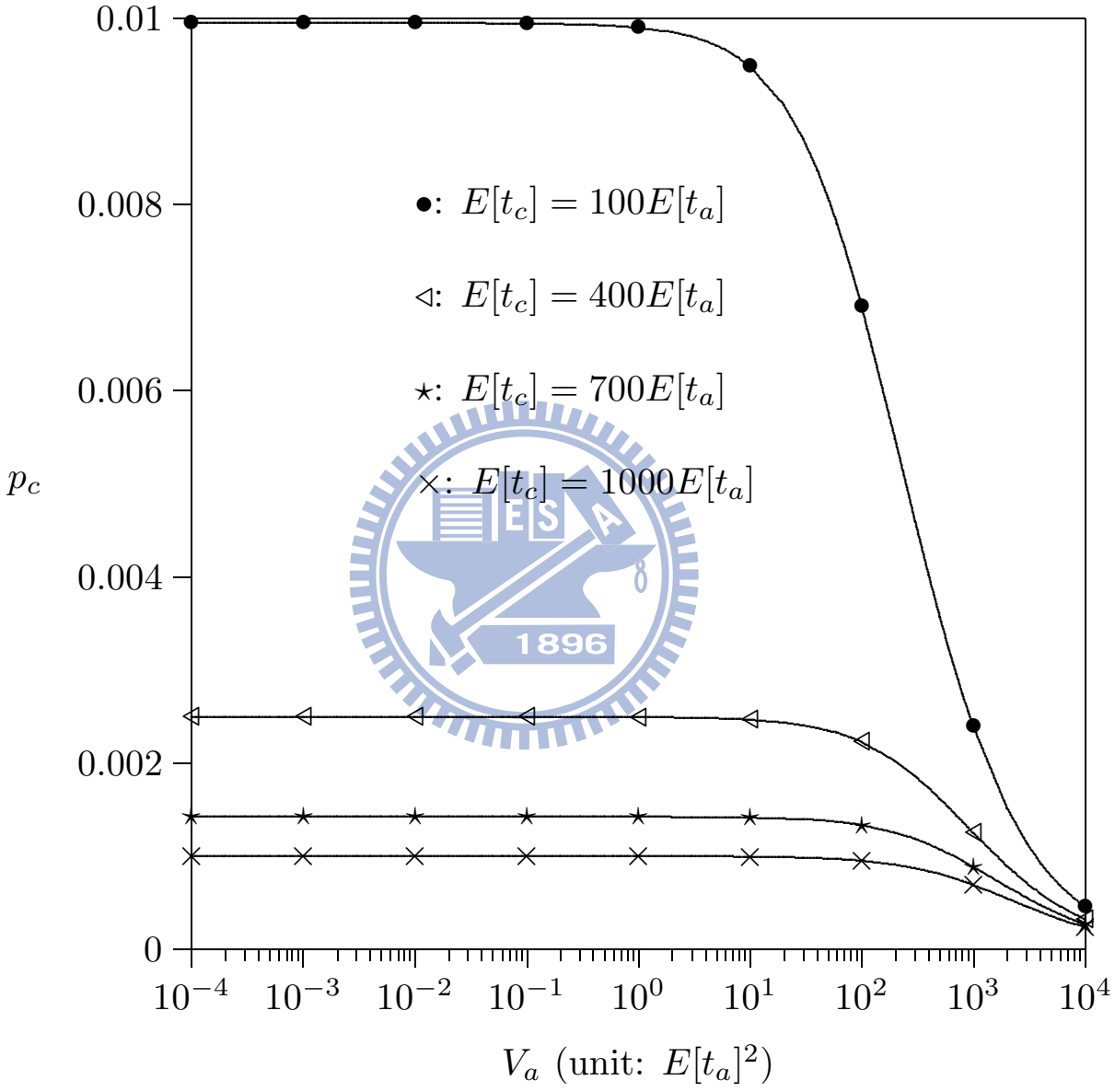


圖 2.5: $E[t_c]/E[t_a]$ 與 V_a 對 p_c 的影響 (t_c 為指數分佈)

第三章　CFA 通知 (Notification) 與失敗 (Failure) 偵測

User 1 將 UE1 插上充電器後，他可能會移動到其他房間 (例如從寢室移動到廚房)。使用者希望 CFA 啟用程序完成之後能被告知 CFA 程式已正確啟用指定轉接服務，將來電轉接至目標座機。我們在此章節提出一個 CFA 通知程序 (notification procedure) 來提供此功能。除了提醒使用者 CFA 啟用程序成功完成以外，此程序也藉由一個使用計時器 (timer) T 的門檻機制 (threshold mechanism) 來通知 user 1 指定轉接啟用程序執行失敗。即 CFA 通知程序可以用來告知 uesr 1 指定轉接啟用程序是執行成功還是失敗 (當 T 過期時)。

3.1　CFA 通知程序 (Notification Procedure)

我們以圖 3.1 與下列步驟介紹 CFA 通知程序：

Step D.1. UE1 的 CFA 程式自動撥打轉接號碼 031111111，開始建立與 Phone1 的通話連線。接著 MSC/VLR 送出 SS7 IAM 信令給 SSP1。

Step D.2. SSP1 對 Phone1 震鈴並回傳 SS7 ACM 信令給 MSC/VLR。然後 MSC/VLR 通知 UE1 Phone1 開始響鈴。

Step D.3. 在 user 1 接起 Phone1 之後，SSP1 送 SS7 ANM 信令給 MSC/VLR。藉由語音通告 (voice announcement) 等方式，UE1 的 CFA 程式告知 user 1 指定轉接的啟用狀態與是否有來電在 CFA 啟用程序執行期間到達 UE1。我們注意到，在此步驟，程式將那些 0.1% 至 1% 仍會到達 UE1 的來電 (介紹於第 2.4 章節) 通知給 user 1。語音通告要求 user 1 按下一個號碼 (例如按下「1」) 來確認已被告知指定轉接啟用狀態。假如程式不小心選擇到了一個錯誤的轉接號碼，則接起電話的人不會按下

號碼鍵「1」。此時程式取消指定轉接服務。

Step D.4. 在 user1 掛斷 Phone1 之後，SSP1 發出 SS7 *Release* (REL) 信令至 MSC/VLR，以結束此通話連線。

Step D.5. MSC/VLR 回覆 SS7 *Release Complete* (RLC) 信令給 SSP1。此程序結束。

在成功啟用指定轉接服務之後，UE1 進入充電狀態。針對「關閉 UE」這個情況，UE1 在完成上述程序之後才會真的關機。藉由向 user 1 做確認，Step D.3 保證指定轉接服務是正確地被啟用。

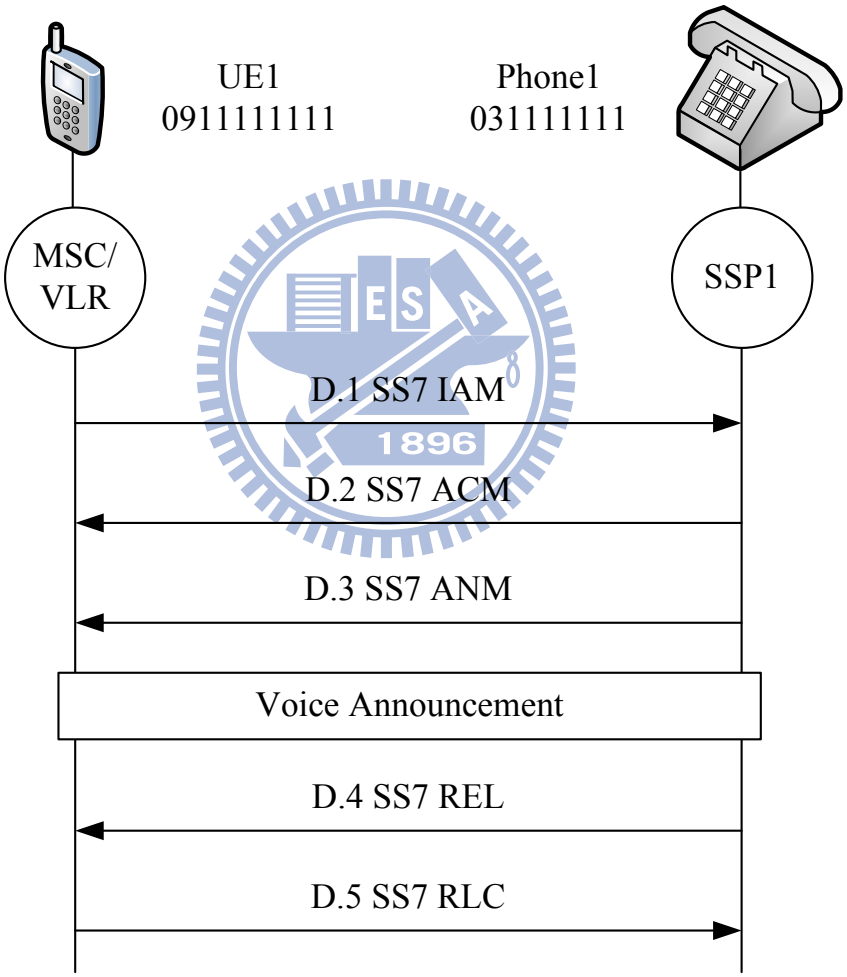


圖 3.1: CFA 通知程序

3.2 CFA 啟用程序之失敗偵測 (Failure Detection)

我們提出的 CFA 啟用程序之失敗偵測係使用一門檻值 T 來做判斷，此值可計算如下。當 CFA 啟用程序執行時，程式會測量並儲存執行所需時間 t_a。UE1 的 CFA 程式累計最近 m 個 t_a 樣本。假設 $t_{a,i}$ 為先前第 i 個 t_a 樣本。當 UE1 的 CFA 程式執行 CFA 啟用程序時，T 可計算如下：

$$T = \frac{\alpha\left(\sum_{i=1}^{m} t_{a,i}\right)}{m} \tag{3.1}$$

此處 $\alpha > 1$ 為加權因子，用來保證 T 不會比實際的 t_a 值還小。假如 CFA 啟用程序沒有在 T 的時間內完成 (即 CFA 程式在 T 時間內沒有收到 HLR 的回覆訊息)，則當作此 CFA 啟用程序執行失敗。在此案例，CFA 通知程序會告知 user 1 指定轉接服務啟用失敗。

根據方程式 (3)，假如 α 設定太小，CFA 程式會誤將成功啟用的指定轉接設定取消。相反地，假如 α 設定太大，則程式沒法及早偵測出 CFA 啟用程序的失敗。因此，如何選擇一個適當的 α 值，是一個重要的考量。我們以數學分析證明，在中華電信網路環境中，$\alpha = 1.5$ 足以適用我們的 CFA 啟用程序之失敗偵測。我們也觀察在不同的 V_a (t_a 的變異數) 值之下，CFA 啟用程序之失敗偵測的效能為何。

假設 $p_s = Pr[t_a < T]$ 為 CFA 啟用程序在 T 時間內執行成功的機率。很明顯地，p_s 值越大，則 CFA 啟用程序之失敗偵測的效能越好。

假設 t_a 為任一隨機變數 (random variable)，其密度函數 (density function) 與拉普拉斯轉換 (Laplace transform) 分別為 $f_a(\cdot)$ 與 $f_a^*(s)$。假設 T 為任一隨機變數，其密度函數與拉普拉斯轉換分別為 $f_T(\cdot)$ 與 $f_T^*(s)$。假如我們將方程式 (3.1) 改寫為 $T = \sum_{i=1}^{m}\left(\frac{\alpha t_{a,i}}{m}\right)$，則 T 分佈的拉普拉斯轉換為

$$f_T^*(s) = \left[f_a^*\left(\frac{\alpha s}{m}\right)\right]^m \tag{3.2}$$

假如 t_a 為耳朗隨機變數 (Erlang random variable)，其形狀參數 (shape parameter) 與速率參數 (rate parameter) 分別為 k 與 μ。則其密度函數與拉普拉斯轉換分別為

$$f_a(t_a) = \frac{\mu^k t_a^{k-1} e^{-\mu t_a}}{(k-1)!} \quad \text{與} \quad f_a^*(s) = \left(\frac{\mu}{s+\mu}\right)^k \tag{3.3}$$

將方程式 (3.3) 代進方程式 (3.2)，我們可以得到

$$f_T^*(s) = \left(\frac{\mu}{\frac{\alpha s}{m}+\mu}\right)^{km} = \left(\frac{m\mu}{\alpha s + m\mu}\right)^{km} \tag{3.4}$$

我們選擇耳朗分佈係因為此分佈能很容易地被擴展成超耳朗分佈 (hyper-Erlang distribution)。對許多其他分布與測量資料來說，超耳朗分佈已被證明是一個好的趨近法 [7][8]。

根據方程式 (3.3) 與方程式 (3.4)，我們可以將 p_s 推導為

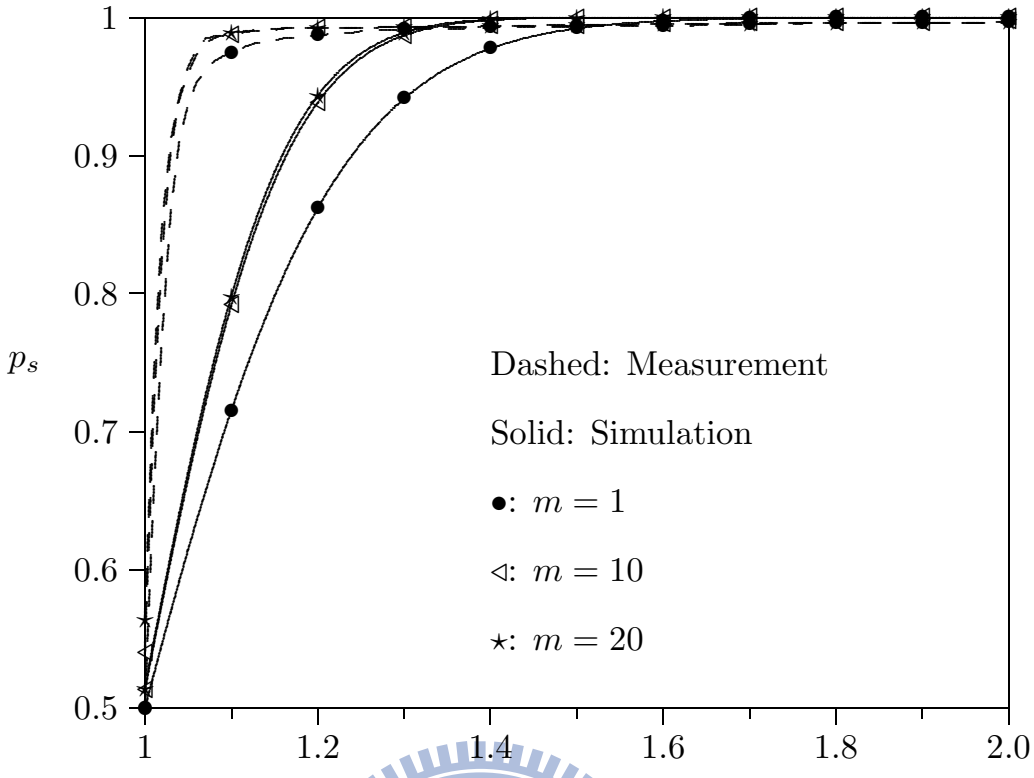
$$\begin{aligned}
p_s &= Pr[t_a < T] \\
&= \int_{T=0}^{\infty} f_T(T) \int_{t_a=0}^{T} f_a(t_a) dt_a dT \\
&= \int_{T=0}^{\infty} f_T(T) \left[1 - \sum_{i=0}^{k-1} \frac{e^{-\mu T}(\mu T)^i}{i!}\right] dT \\
&= 1 - \sum_{i=0}^{k-1} \left[\frac{\mu^i (-1)^i}{i!}\right] \left[\frac{d^i f_T^*(s)}{ds^i}\bigg|_{s=\mu}\right] \\
&= 1 - \sum_{i=0}^{k-1} \left(\frac{\alpha}{\alpha+m}\right)^i \left[\frac{(km+i-1)!}{i!(km-1)!}\right] \left(\frac{m}{\alpha+m}\right)^{km}
\end{aligned} \tag{3.5}$$

方程式 (3.5) 係用來驗證我們的模擬模型 (遵循第 2.4 章節介紹的蒙地卡羅模擬法)。模擬實驗顯示我們的分析 (方程式 (3.5)) 與模擬結果的誤差在 0.1% 以內。在這篇論文的剩餘部分，我們使用此被驗證過的模擬實驗來觀察 CFA 啟用程序之失敗偵測的效能。

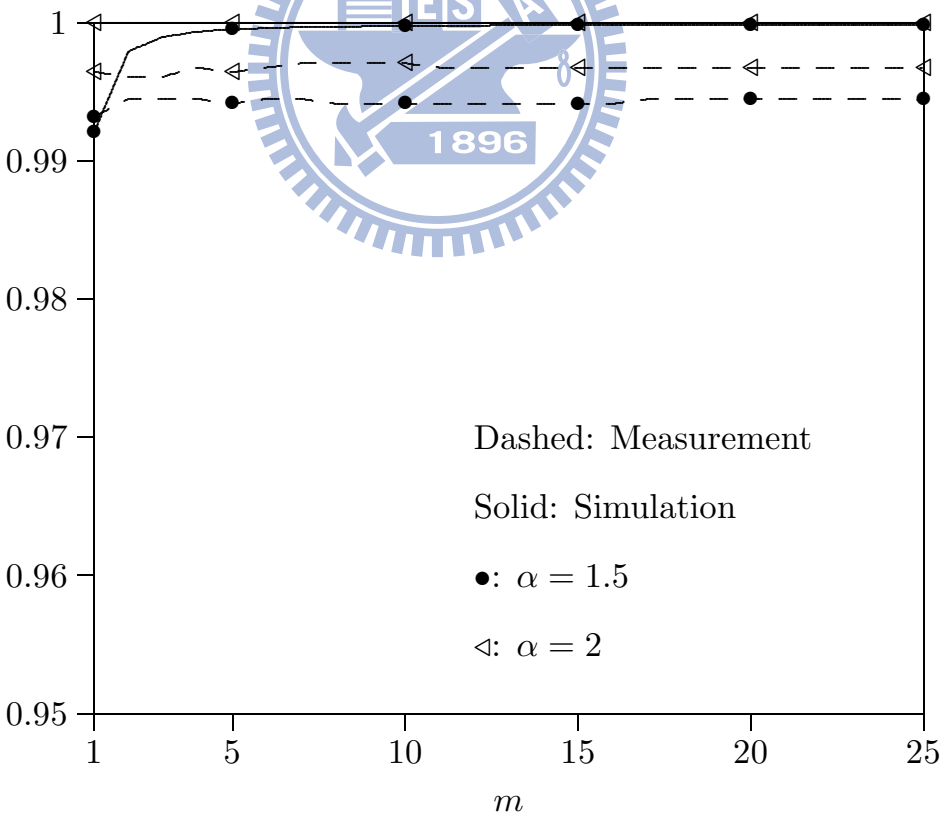
更明確地說，我們將此被驗證過的模擬模型從耳朗 t_a 分佈 (Erlang t_a distribution) 擴展成伽馬 t_a 分佈 (Gamma t_a distribution)。接著我們使用此伽馬模擬模型，將 t_a 的平均值與變異數分別設為 $E[t_a] = 7.88266$ 秒與 $V_a = 0.0139717E[t_a]^2$，藉此來趨近在中華電信網路環境中所測量到的資料 (於 2.4 章節介紹過)。針對模擬實驗與實際測量所得到的 t_a 值，圖 3.2 根據不同的 α 與 m 值，分別標示出相對應的 p_s 值。很明顯地，隨著 α 增加， p_s 亦會增加。圖 3.2 (a) 顯示當 α 較小時 ($\alpha \leq 1.3$)，模擬資料為測量資料的下限 (lower bounds)。當 α 較大時 ($\alpha \geq 1.5$) 結果則相反，模擬資料為測量資料的上限 (upper bounds)。伽馬 t_a 分佈與測量資料的 p_s 值的傾向很相似。而且當 α 值較大 ($\alpha \geq 1.5$) 時，兩者的 p_s 值會很接近。

此外，圖 3.2 (b) 顯示當 $m \geq 20$ 時， p_s 不易隨著 m 值之變動而有所改變。換句話說，儲存 CFA 最近 20 個 t_a 樣本即足以用來計算方程式 (3.1) 的 T 值。當 $m = 20$ 且 $\alpha \geq 1.5$ 時，測量與模擬的 p_s 值誤差在 0.5% 以內。圖 3.2 也顯示在中華電信網路環境裡，選擇一個小的 α ($\alpha = 1.5$) 值，即足以宣稱有好的 p_s 效能 (例如 $p_s > 0.99$)。

針對 V_a 值較大的電信網路環境，圖 3.3 顯示當 V_a 增加 (t_a 為伽馬分佈) 時， p_s 會先減少再增加。以下我們針對此現象加以說明。當 V_a 較小 (即 $V_a < E[t_a]^2$) 時，假如 V_a 增加，我們觀察到會有較多短 t_a 與長 t_a 出現。這些長 t_a 使得 p_s 變小。當 V_a 較大 (即 $V_a > 10E[t_a]^2$) 時，假如 V_a 增加，我們觀察到有更多短 t_a 出現。此外，我們也觀察到有更長的 t_a 出現。然而，這些非常長的 t_a 的個數遠小於短 t_a 的個數，因此我們可以觀察到 p_s 值變大。我們注意到假如選擇 $\alpha = 4.5$ (即 $E[T] = 35.47197$ 秒)，則不論 V_a 值為何， p_s 皆大於 0.9。



(a) α 的影響



(b) m 的影響

圖 3.2: α 與 m 對 p_s 的影響 ($E[t_a] = 7.88266$ 秒， $V_a = 0.0139717E[t_a]^2$)

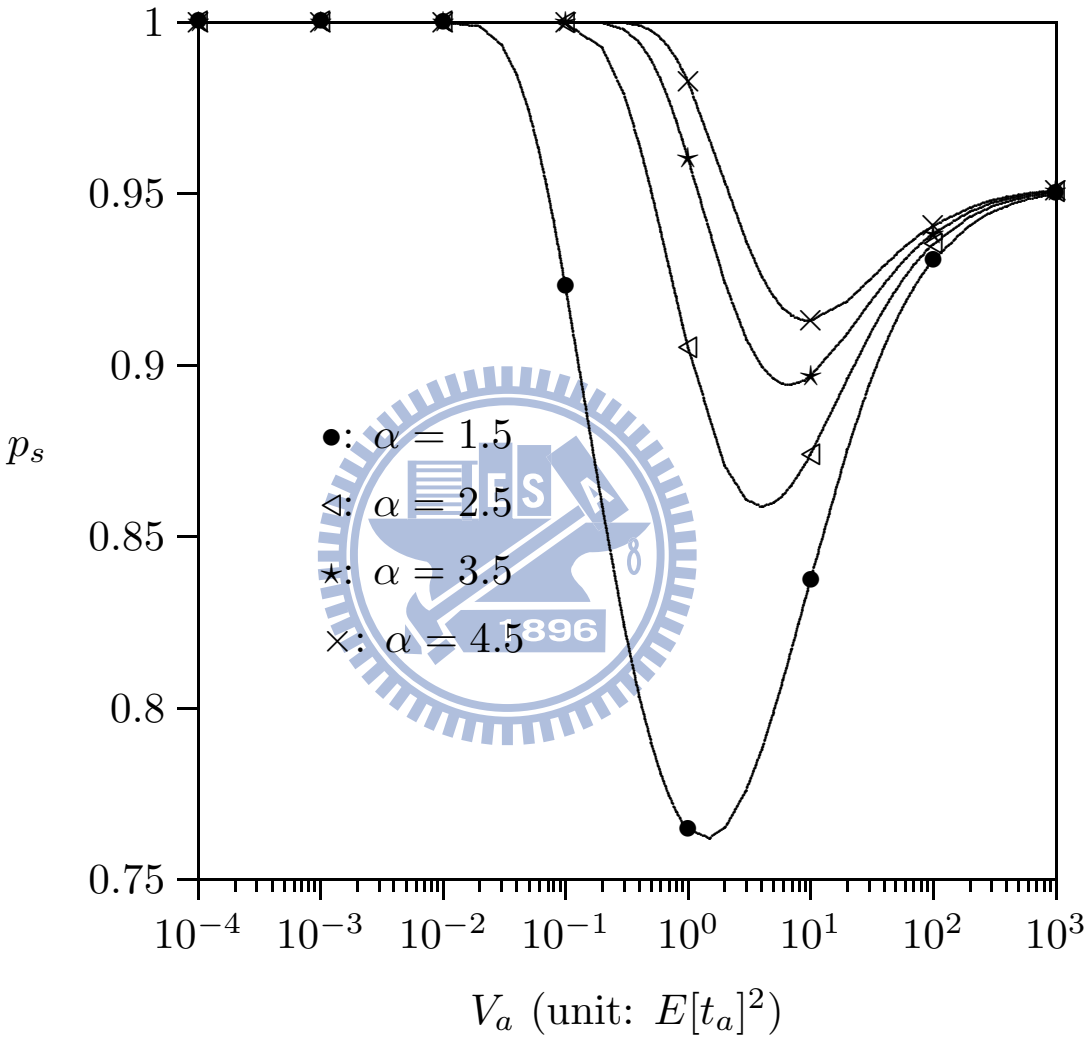


圖 3.3: α 與 V_a 對 p_s 的影響 ($m = 20$)

第四章　CFA 與車載資訊 (Telematics) 之應用

使用車載資訊服務時，使用者會在車上安裝個人導航設備 (Personal Navigation Device；PND)，其提供了 GPS 定位服務與行動通訊能力 (例如：GSM、GPRS、或是 UMTS)。使用免手持電話服務時，當使用者進到汽車裡並開啟 PND 之後，所有來電皆會被轉接至 PND。當有來電到達時，使用者不需使用手就能接電話 (換言之，使用者能藉由車上的揚聲器聽電話，並藉由 PND 的麥克風講電話)。現有的免手持車載電話服務大約可分為兩類：有線 (wire-line) 與藍芽 (bluetooth) 這二種方法。但是此二種方法皆需要使用者手動將行動電話與安裝於車內的通訊設備做連接。

假設 user 1 的 PND 已安裝一個軟體，使此設備能偵測觸發事件「當 PND 被開啟或關閉」，則我們能將 CFA 程式的自動指定轉接功能與車載資訊服務搭配應用。許多由台灣生產的 PND 都能配合電信業者的需求，允許如此的修改。當偵測到此觸發事件，PND 寄一封簡訊給 UE1，以啟用或取消指定轉接服務。CFA 程式執行如下。在 user 1 上車並開啟 PND 之後，以下步驟將會被執行：

Step E.1. PND 從 GPS 接收器取得目前位置，並寄簡訊給 UE1。此簡訊包含 PND 的 GPS 位置資訊與一請求，以啟用指定轉接服務將來電轉接至 PND 的電話號碼。

Step E.2. 在 UE1 收到簡訊之後，UE1 的 CFA 程式利用 A-GPS 機制獲得目前的位置資訊，並將此 UE1 的位置資訊與 PND 的位置資訊做比較。假如這兩個位置足夠接近 (例如 10 公尺以內)，則 CFA 程式會認為 UE1 在車裡，並響鈴通知 user 1，以確認使用者是否想要啟用指定轉接功能。User 1 簡單地按下一個鍵以接受 (或拒絕) 指定轉接啟用請求。接著，程式執行 CFA 啟用程序 (如 2.1 章節所介紹)。

Step E.3. UE1 的 CFA 程式回送給 PND 一則簡訊，告知 CFA 啟用程序執行結果。接著 PND 藉由語音通告 (voice announcement) 等方式，將結果告知 user 1。

在 Step E.2 時，假如 UE1 的 GPS 接收器沒有開啟，則當收到來自 PND 的簡訊時，

程式會將 GPS 接收器打開。等 UE1 的 CFA 程式取得它的 GPS 位置資訊後，再將 GPS 接收器關閉，以避免手機額外的電源消耗。

當 user 1 關閉 PND (將車熄火) 時，指定轉接服務會自動被取消，如下列步驟所述：

Step F.1. 在真的關機前，PND 送給 UE1 一則簡訊，以取消指定轉接服務。

Step F.2. 當接收到 PND 傳來的簡訊時，UE1 的 CFA 程式執行 CFA 取消程序 (如 2.3 章節所介紹)。

假如使用者離開汽車但卻沒有將 PND 關閉，則上述步驟將不適用。為了解決此問題，PND 可以定期執行類似 E.1 至 E.3 的步驟，以檢查 UE1 的位置。假如 GPS 位置資訊顯示 UE1 與 PND 距離過遠，則 UE1 詢問使用者是否要執行 CFA 取消程序。

第五章　結論與未來工作

此篇論文針對手機端提出一種自動的「指定轉接演算法」(Call Forwarding Algorithm；CFA)。固定住的座機只能手動啟用指定轉接功能，但此操作方式對使用者來說是一道瑣碎麻煩的程序。相反地，許多觸發事件有可能在手機上發生，例如：電池充電、關機、位置改變..... 等等。藉由偵測這些觸發事件，CFA 程式能自動啟用指定轉接服務。

CFA 程式讓使用者能避免繁瑣的指定轉接啟用與取消程序，而且不論程序是否執行成功，CFA 程式都會將執行結果告知使用者。我們推導時限 T 的值，使 CFA 程式能在 T 時間內，適當地告知使用者執行結果。在中華電信網路環境下，我們藉由模擬實驗與實際測量的方式，證明 CFA 程式能達到不錯的效能，可以被實際應用在商業用途上。此外，我們能很容易地在智慧型手機上安裝我們的 CFA 軟體，而且不需要對現有電信網路環境做任何的修改。

此外，我們目前也著手研究 IBM WebSphere software for Telecom (WsT)。IBM WsT 是一個能提供標準 Next Generation Network (NGN) / IP Multimedia Subsystem (IMS) 網路服務的平台。此平台讓服務提供者在開發時，只需知道「網路服務能做到那些功能」與「有那些應用程式介面 (Application Programming Interfaces；APIs)」即可，不需要知道「網路服務是如何被實作的」。目前 IBM WsT 與中華電信 NGN / IMS 做連接，我們正在此平台開發一個結合電信服務功能的揪團系統。指定轉接服務能與此揪團系統搭配應用。例如參加會議時，使用者會希望手機啟用指定轉接，將來電轉接至其他電話號碼 (例如秘書或助理的電話號碼..... 等等)。而當會議結束時，使用者會希望手機取消指定轉接。使用此揪團系統邀請參加會議，系統能在會議開始時間自動將使用者手機啟用指定轉接，並在會議結束時間自動取消指定轉接。目前我們正在 IBM WsT 平台上進行指定轉接與揪團系統的搭配應用，以期望將指定轉接服務更廣泛地提供給使用者。

參考文獻

[1] MSDN library. Available: http://msdn.microsoft.com/en-us/default.aspx

[2] Yi-Bing Lin and Ai-Chun Pang, *Wireless and Mobile All-IP Networks*. John Wiley & Sons, Inc., 2005.

[3] 3GPP. 3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; Call Forwarding (CF) supplementary services; Stage 3. Technical Specification 3G TS 24.082 version 9.0.0 (2009-12), 2009.

[4] 3GPP. 3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; Mobile Application Part (MAP) specification. Technical Specification 3G TS 29.002 version 9.2.0 (2010-06), 2010.

[5] Shun-Ren Yang, "Dynamic power saving mechanism for 3G UMTS system," *ACM/ Springer Mobile Networks and Applications*, 12(1): 5-14, 2007.

[6] Yen-Cheng Lai, Phone Lin, Yuguang Fang, Wei-Hao Chen, "Channel allocation for UMTS multimedia broadcasting and multicasting," *IEEE Transactions on Wireless Communications*, 7(11): 4375-4383, 2008.

[7] Yuguang Fang and Imrich Chlamtac, "Teletraffic analysis and mobility modeling of PCS networks," *IEEE Transactions on Communications*, 47(7):1062-1072, July 1999.

[8] Frank Kelly, *Reversibility and Stochastic Networks*. John Wiley & Sons, 1979.

附錄 A　程式碼

程式碼 A.1: my_Call_Forwarding_testing.h

```
1  // my_Call_Forwarding_testing.h : PROJECT_NAME 應用程式的主要標頭檔
2  //
3
4  #pragma once
5
6  #ifndef __AFXWIN_H__
7      #error "對 PCH 包含此檔案前先包含 'stdafx.h'"
8  #endif
9
10 #ifdef POCKETPC2003_UI_MODEL
11 #include "resourceppc.h"
12 #endif
13
14 ////////////////////////////////////////////////////////////////
15 #include <regext.h>
16 #include <snapi.h>
17 #include <tapi.h>
18
19 typedef struct __CALLFORWARDING_INFO_TAG__
20 {
21     DWORD    dwMode;
22     int      nSeconds;
23     //CString    strNumber;
24     TCHAR    strNumber[32];
25 }CALLFORWARDING_INFO, *PCALLFORWARDING_INFO;
```

```
26 ////////////////////////////////////////////////////////////////////
27
28 // Cmy_Call_Forwarding_testingApp:
29 // 請參閱實作此類別的 my_Call_Forwarding_testing.cpp
30 //
31
32 class Cmy_Call_Forwarding_testingApp : public CWinApp
33 {
34 public:
35     Cmy_Call_Forwarding_testingApp();
36
37 // 覆寫
38 public:
39     virtual BOOL InitInstance();
40
41 // 程式碼實作
42 protected:
43     CString str_dir_path;
44
45 public:     // battery state changing
46     HRESULT register_app ();              // register notification for
            battery state changing
47     HRESULT unregister_app ();            // unregister notification
48     CString get_dir_path ();
49
50 public:     // call forwarding
51     long initialize_TAPI ();
52     void shutdown_TAPI ();
53     DWORD GetCellularLineId ();
54     HLINE OpenTAPILine (DWORD dw_line_id);
55     DWORD ProcessCallForwarding(TCHAR *tmpPtr);
56     LPLINEFORWARDLIST AllocateCallForwardList(PCALLFORWARDING_INFO
            pInfo,int nEntries);
57
```

```
58      void test_func () { MessageBox (NULL, TEXT("test test"), TEXT("
           title"), MB_OK); }
59
60      DECLARE_MESSAGE_MAP()
61  };
62
63  extern Cmy_Call_Forwarding_testingApp theApp;
```

程式碼 A.2: my_Call_Forwarding_testingDlg.h

```
1  // my_Call_Forwarding_testingDlg.h : 標頭檔
2  //
3
4  #pragma once
5  #include "afxwin.h"
6
7  // Cmy_Call_Forwarding_testingDlg 對話方塊
8  class Cmy_Call_Forwarding_testingDlg : public CDialog
9  {
10 // 建構
11 public:
12     Cmy_Call_Forwarding_testingDlg(CWnd* pParent = NULL);   // 標準建
       構函式
13
14 // 對話方塊資料
15     enum { IDD = IDD_MY_CALL_FORWARDING_TESTING_DIALOG };
16
17 protected:
18     virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV 支
       援
19
20 protected:
21     virtual void OnOK();
22
23 // 程式碼實作
24 protected:
25     HICON m_hIcon;
26
27     // 產生的訊息對應函式
28     virtual BOOL OnInitDialog();
29 #if defined(_DEVICE_RESOLUTION_AWARE) && !defined(WIN32_PLATFORM_WFSP
    )
30     afx_msg void OnSize(UINT /*nType*/, int /*cx*/, int /*cy*/);
```

```
31 #endif
32     DECLARE_MESSAGE_MAP()
33
34 public:
35     BOOL update_state ();   // update window info & do call
          forwarding
36     afx_msg void OnEnChangeEditAddPhone();
37     afx_msg void OnBnClickedButtonStart();
38     afx_msg void OnBnClickedButtonDelete();
39     afx_msg void OnBnClickedButtonAdd();
40     afx_msg void OnBnClickedButtonQuit();
41
42 private:    // read or write saved data (phone number & comment)
43     HRESULT register_callback ();
44     HRESULT unregister_callback ();
45     bool write_phone_data ();
46     bool write_state_data ();
47
48 protected:
49     //System.Windows.Forms.Timer obj_timer;
50     ////Timer obj_timer;
51
52 public:
53     CComboBox obj_combo_forward_phone;
54     CStatic obj_static_status;
55     CEdit obj_edit_add_phone;
56     CEdit obj_edit_add_comment;
57 };
```

程式碼 A.3: my_Call_Forwarding_testing.cpp

```
1  // my_Call_Forwarding_testing.cpp : 定義應用程式的類別行為。
2  //
3
4  #include "stdafx.h"
5  #include "my_Call_Forwarding_testing.h"
6  #include "my_Call_Forwarding_testingDlg.h"
7
8  #ifdef _DEBUG
9  #define new DEBUG_NEW
10 #endif
11
12 ////////////////////////////////////////////////////////////
13 #define WM_OFFSET                    10000
14 #define WM_CHANGE_BATTSTRENGTH    (WM_OFFSET + 1)
15 #define WM_CHANGE_BATTSTATE       (WM_OFFSET + 2)
16 #define WM_CHANGE_BACKUPSTRENGTH  (WM_OFFSET + 3)
17 #define WM_CHANGE_BACKUPSTATE     (WM_OFFSET + 4)
18
19 const WCHAR c_wszAppBattStateName[]   = L"BattStat.BattState";   //
      RegistryNotifyApp names ///
20
21 #include <tapi.h>
22 #define CELLTSP_LINENAME_STRING (L"Cellular␣Line")
23 #define EXT_API_LOW_VERSION     0x00010000
24 #define EXT_API_HIGH_VERSION    0x00010000
25
26 HLINEAPP     m_hLineApp = NULL;
27 DWORD        m_dwDevices = 0;
28 DWORD        m_dwLowAPIVersion = 0;
29 long         lTapiReturn = 0;
30 DWORD        m_dwCellularId = 0;
31 DWORD        m_dwAPIVersion = 0;
32 DWORD        m_dwExtVersion = 0;
```

```
33 HLINE        m_hCellularLine = NULL;
34 DWORD        m_dwAddressId = 0;
35
36 void FAR PASCAL line_callback (DWORD h_device, DWORD dw_msg, DWORD
       dw_callback_instance, DWORD dw_param_1, DWORD dw_param_2, DWORD
       dw_param_3);
37 /////////////////////////////////////////////////////////////////////
38
39 // Cmy_Call_Forwarding_testingApp
40
41 BEGIN_MESSAGE_MAP(Cmy_Call_Forwarding_testingApp, CWinApp)
42 END_MESSAGE_MAP()
43
44 // Cmy_Call_Forwarding_testingApp 建構
45 Cmy_Call_Forwarding_testingApp::Cmy_Call_Forwarding_testingApp()
46     : CWinApp()
47 {
48     // TODO: 在此加入建構程式碼，
49     // 將所有重要的初始設定加入 InitInstance 中
50
51     // 取得程式資料夾路徑
52     int index;
53     GetModuleFileName (NULL, this->str_dir_path.GetBuffer (MAX_PATH),
           MAX_PATH);
54     this->str_dir_path.ReleaseBuffer ();
55     index = str_dir_path.ReverseFind ('\\') + 1;
56     str_dir_path.Delete (index, (str_dir_path.GetLength () - index));
57 }
58
59 // 僅有的一個 Cmy_Call_Forwarding_testingApp 物件
60 Cmy_Call_Forwarding_testingApp theApp;
61
62 // Cmy_Call_Forwarding_testingApp 初始設定
63
```

```
64 BOOL Cmy_Call_Forwarding_testingApp::InitInstance()
65 {
66     // 必須在應用程式初始化過程中呼叫一次，以初始化 SHInitExtraControls
67     // 任何 Windows Mobile 專用的控制項，如 CAPEDIT 和。 SIPPREF
68     SHInitExtraControls();
69
70     // 標準初始設定
71     // 如果您不使用這些功能並且想減少
72     // 最後完成的可執行檔大小，您可以
73     // 從下列程式碼移除不需要的初始化常式，
74     // 變更儲存設定值的登錄機碼
75     // TODO: 您應該適度修改此字串
76     // 例如，公司名稱或組織名稱()
77     SetRegistryKey(_T("本機 AppWizard 所產生的應用程式"));
78
79     Cmy_Call_Forwarding_testingDlg dlg;
80     m_pMainWnd = &dlg;
81     INT_PTR nResponse = dlg.DoModal();
82     if (nResponse == IDOK)
83     {
84         // TODO: 在此放置於使用確定 [] 來停止使用對話方塊時
85         // 處理的程式碼
86     }
87
88     // 因為已經關閉對話方塊，傳回，所以我們會結束應用程式， FALSE
89     // 而非提示開始應用程式的訊息。
90     return FALSE;
91 }
92
93 HRESULT Cmy_Call_Forwarding_testingApp::register_app ()
94 {
95     HRESULT hr;
96     NOTIFICATIONCONDITION nc;
97     TCHAR szExePath[MAX_PATH];
```

```
98      TCHAR szThisPath[MAX_PATH];
99
100     // 確定程式尚未註冊
101     unregister_app ();
102
103     // 取得程式的路徑
104     GetModuleFileName(NULL, szThisPath, MAX_PATH);
105
106     // 在路徑前後分別加上 \" 以防止路徑字串中含有空白字元
107     StringCchCopy(szExePath, MAX_PATH, L"\"");
108     StringCchCat(szExePath, MAX_PATH, szThisPath);
109     StringCchCat(szExePath, MAX_PATH, L"\"");
110
111     // 通知程式值發生改變 & 設定 bitmask 以檢查該值
112     nc.ctComparisonType = REG_CT_ANYCHANGE;
113     nc.dwMask           = SN_POWERBATTERYSTATE_BITMASK;
114     nc.TargetValue.dw   = 0;
115
116     // 註冊電池狀態通知
117     hr = RegistryNotifyApp(
118                 SN_POWERBATTERYSTATE_ROOT,
119                 SN_POWERBATTERYSTATE_PATH,
120                 SN_POWERBATTERYSTATE_VALUE,
121                 c_wszAppBattStateName,
122                 szExePath,
123                 NULL,
124                 NULL,
125                 WM_CHANGE_BATTSTATE,
126                 0,      // Command line 會包
                        含 "/notify <notification name>".
127                 &nc
128                 );
129     return hr;
130 }
```

```
131
132 HRESULT Cmy_Call_Forwarding_testingApp::unregister_app ()
133 {
134     RegistryStopNotification(c_wszAppBattStateName);
135     return S_OK;
136 }
137
138 CString Cmy_Call_Forwarding_testingApp::get_dir_path ()
139 {
140     return str_dir_path;
141 }
142
143 long Cmy_Call_Forwarding_testingApp::initialize_TAPI ()
144 {
145     LINEINITIALIZEEXPARAMS  sLineParam;
146
147     memset(&sLineParam,0,sizeof(LINEINITIALIZEEXPARAMS));
148
149     sLineParam.dwTotalSize   = sizeof(LINEINITIALIZEEXPARAMS);
150     sLineParam.dwOptions          =
        LINEINITIALIZEEXOPTION_USEHIDDENWINDOW;
151
152     m_dwLowAPIVersion    = TAPI_CURRENT_VERSION;
153
154     lTapiReturn/*--*/   = lineInitializeEx (&m_hLineApp, NULL,
        line_callback, TEXT("myTesting"), &m_dwDevices, &
        m_dwLowAPIVersion, &sLineParam);
155     if(0 == lTapiReturn)
156     {
157         m_dwCellularId  = GetCellularLineId();
158         if(0xFFFFFFFF != m_dwCellularId)
159         {
160             m_hCellularLine = OpenTAPILine(m_dwCellularId);
161             if(m_hCellularLine)
```

```
162             {
163                 LONG lError = lineGetAddressID(m_hCellularLine,&
                        m_dwAddressId,LINEADDRESSMODE_DIALABLEADDR,
164                                                 TEXT("9746495065"),11)
                                                    ; //
                                                    ////////////////////

165                 if(lError == 0)
166                     return 0;
167                 else
168                     ; // error handling for lineGetAddressID()
169             }
170             ; // error handling for OpenTAPILine()
171         }
172         ; // error handling for GetCellularLineId()
173     }
174     return -1;
175 }
176
177 void Cmy_Call_Forwarding_testingApp::shutdown_TAPI ()
178 {
179     if(m_hCellularLine)
180     {
181         lineClose(m_hCellularLine);
182     }
183
184     if(m_hLineApp)
185     {
186         lineShutdown(m_hLineApp);
187     }
188
189     m_hLineApp      = NULL;
190     m_hCellularLine = NULL;
191 }
```

```
192
193 DWORD Cmy_Call_Forwarding_testingApp::GetCellularLineId ()
194 {
195     DWORD               dwReturn        = 0xFFFFFFFF;
196     long                lResult         = 0;
197     LINEEXTENSIONID     sLineExt        = {0};
198     LPLINEDEVCAPS       lpLineDevCaps   = NULL;
199     BOOL                bContinue       = TRUE;
200
201     for(DWORD dwLine=0; dwLine<m_dwDevices && bContinue; ++dwLine)
202     {
203         lResult     = lineNegotiateAPIVersion(m_hLineApp,dwLine,
                m_dwLowAPIVersion,TAPI_CURRENT_VERSION,&m_dwAPIVersion,&
                sLineExt);
204
205         if(0 == lResult)
206         {
207             lpLineDevCaps   = (LPLINEDEVCAPS)LocalAlloc(LPTR,sizeof(
                    LINEDEVCAPS));
208             lResult         = LINEERR_STRUCTURETOOSMALL;
209
210             lpLineDevCaps->dwTotalSize  = sizeof(LINEDEVCAPS);
211             lpLineDevCaps->dwNeededSize = sizeof(LINEDEVCAPS);
212
213             while(LINEERR_STRUCTURETOOSMALL == lResult)
214             {
215                 lResult = lineGetDevCaps(m_hLineApp,dwLine,
                        TAPI_CURRENT_VERSION,0,lpLineDevCaps);
216
217                 if(LINEERR_STRUCTURETOOSMALL == lResult ||
                        lpLineDevCaps->dwTotalSize < lpLineDevCaps->
                        dwNeededSize)
218                 {
219                     lpLineDevCaps   = (LPLINEDEVCAPS)LocalReAlloc(
```

```
                        lpLineDevCaps,lpLineDevCaps->dwNeededSize,
                        LMEM_MOVEABLE);
220                     lResult          = LINEERR_STRUCTURETOOSMALL;
221
222                     lpLineDevCaps->dwTotalSize  = lpLineDevCaps->
                        dwNeededSize;
223                 }
224             }
225
226             if(0 == lResult)
227             {
228                 TCHAR szName[512];
229
230                 memcpy((PVOID)szName,(PVOID)((BYTE*)lpLineDevCaps +
                    lpLineDevCaps ->dwLineNameOffset),
231                         lpLineDevCaps->dwLineNameSize);
232
233                 szName[lpLineDevCaps->dwLineNameSize]   = 0;
234
235                 if(_tcscmp(szName,CELLTSP_LINENAME_STRING) == 0)
236                 {
237                     dwReturn    = dwLine;
238                     bContinue   = FALSE;
239                 }
240             }
241
242             LocalFree((HLOCAL)lpLineDevCaps);
243         }
244     }
245
246     return dwReturn;
247 }
248
249 HLINE Cmy_Call_Forwarding_testingApp::OpenTAPILine (DWORD dw_line_id)
```

```
250 {
251     DWORD    dwMediaMode = LINEMEDIAMODE_INTERACTIVEVOICE;
252     HLINE    hLine       = NULL;
253     long     lReturn     = lineOpen(m_hLineApp,m_dwCellularId,&hLine,
254                                     TAPI_CURRENT_VERSION,0,NULL,
255                                     LINECALLPRIVILEGE_OWNER,
                                        dwMediaMode,0);
256     lReturn              = lineNegotiateExtVersion(m_hLineApp,
           m_dwCellularId,m_dwAPIVersion,EXT_API_LOW_VERSION,
           EXT_API_HIGH_VERSION,&m_dwExtVersion);
257     return hLine;
258 }
259
260 DWORD Cmy_Call_Forwarding_testingApp::ProcessCallForwarding(TCHAR *
       tmpPtr)
261 {
262     CALLFORWARDING_INFO     sInfo[4]     = {0};
263     LPLINEFORWARDLIST       pInfo        = NULL;
264     LONG                    lError       = 0;
265     HCALL                   hCall        = NULL;
266     int                     nSeconds     = 5;
267     TCHAR                   strNumber[32] = {NULL};
268
269     if (tmpPtr != NULL)
270     {
271         sInfo[0].dwMode      = LINEFORWARDMODE_UNCOND;
272         sInfo[0].nSeconds    = 0;
273         wcsncpy (sInfo[0].strNumber, tmpPtr, 31);
274
275         pInfo    = AllocateCallForwardList(sInfo,1);
276         if(pInfo)
277             lError   = lineForward(m_hCellularLine,FALSE,m_dwAddressId
                ,pInfo,0,&hCall,NULL);
278     }
```

```
279     else
280     {
281         lError  = lineForward(m_hCellularLine,FALSE,m_dwAddressId,
                NULL,0,&hCall,NULL);
282     }
283
284     if(lError < 0)
285     {
286         ; // error handling for lineForward(...);
287     }
288
289     return lError;
290 }
291
292 LPLINEFORWARDLIST Cmy_Call_Forwarding_testingApp::
    AllocateCallForwardList(PCALLFORWARDING_INFO pInfo,int nEntries)
293 {
294     int                 nTextLen    = 0;
295
296     for(int nNumber=0; nNumber<nEntries; ++nNumber)
297     {
298         nTextLen    += ((lstrlen(pInfo[nNumber].strNumber) + 1) *
                sizeof(TCHAR));
299     }
300
301     DWORD               dwSize  = (sizeof(LINEFORWARDLIST));
302
303     dwSize  += nTextLen;
304     dwSize  += (sizeof(LINEFORWARD) * (nEntries - 1));
305
306     LPLINEFORWARDLIST   pList       = (LPLINEFORWARDLIST)LocalAlloc(
            LPTR,dwSize);
307
308     ZeroMemory(pList,dwSize);
```

```
309
310     DWORD   dwOffset    = sizeof(LINEFORWARDLIST) + (sizeof(
            LINEFORWARD) * (nEntries - 1));
311
312     pList->dwNumEntries = nEntries;
313     pList->dwTotalSize  = dwSize;
314
315     for(int nNumber=0; nNumber<nEntries; ++nNumber)
316     {
317         pList->ForwardList[nNumber].dwCallerAddressOffset   = 0;
318         pList->ForwardList[nNumber].dwCallerAddressSize     = 0;
319         pList->ForwardList[nNumber].dwDestCountryCode       = 0;
320         pList->ForwardList[nNumber].dwForwardMode           = pInfo[
                nNumber].dwMode;
321         pList->ForwardList[nNumber].dwDestAddressSize       = (
                lstrlen(pInfo[nNumber].strNumber) + 1) * sizeof(TCHAR);
322         pList->ForwardList[nNumber].dwDestAddressOffset     =
                dwOffset;
323
324         wcsncpy((TCHAR*)((LPBYTE)pList + dwOffset),
325                 pInfo[nNumber].strNumber,
326                 pList->ForwardList[nNumber].dwDestAddressSize);
327
328         dwOffset    += ((lstrlen(pInfo[nNumber].strNumber) + 1) *
                sizeof(TCHAR));
329     }
330     return pList;
331 }
332
333 void FAR PASCAL line_callback (DWORD h_device, DWORD dw_msg, DWORD
        dw_callback_instance, DWORD dw_param_1, DWORD dw_param_2, DWORD
        dw_param_3)
334 {
335 }
```

程式碼 A.4: my_Call_Forwarding_testingDlg.cpp

```
1  // my_Call_Forwarding_testingDlg.cpp : 實作檔
2  //
3
4  #include "stdafx.h"
5  #include "my_Call_Forwarding_testing.h"
6  #include "my_Call_Forwarding_testingDlg.h"
7
8  /////extern Cmy_Call_Forwarding_testingApp theApp;
9  #define EDIT_MAX              15
10 ////////////////////////////////////////////// my define &
       declaration {
11 #define ID_TIMER              1
12 #define AUTO_QUIT_TIME        30
13 #define FILE_PHONE_DATA       "phone_data.txt"
14 #define FILE_STATE_DATA       "state_data.txt"
15
16 #define MAX_LOADSTRING        100
17 #define STATUS_STRING_LEN     500
18
19 #define WM_OFFSET                       10000
20 #define WM_CHANGE_BATTSTRENGTH     (WM_OFFSET + 1)
21 #define WM_CHANGE_BATTSTATE        (WM_OFFSET + 2)
22 #define WM_CHANGE_BACKUPSTRENGTH   (WM_OFFSET + 3)
23 #define WM_CHANGE_BACKUPSTATE      (WM_OFFSET + 4)
24
25 //#define IDC_BATTSTRENGTH     (WM_OFFSET + 200)
26 #define IDC_BATTSTATE        (WM_OFFSET + 201)
27 //#define IDC_BACKUPSTRENGTH   (WM_OFFSET + 202)
28 //#define IDC_BACKUPSTATE      (WM_OFFSET + 203)
29 #define IDC_CMDLINE          (WM_OFFSET + 204)
30
31 HREGNOTIFY  g_hNotify;  // Handles to notifications
32 DWORD       g_start_time;
```

```
33
34 void CALLBACK timer_callback (HWND hwnd,UINT iMsg,UINT TimerID,DWORD
       Time);
35 void battery_state_callback (HREGNOTIFY h_notify, DWORD dw_user_data,
        const PBYTE p_data, const UINT cb_data); ///
36 void get_battery_state_str (DWORD dwBattState, LPTSTR
       pszBattStateStr); ///
37 void UpdateBattState(HKEY hKey, LPCWSTR wszSubKey, LPCWSTR wszName,
       DWORD dwBitMask, int nControlID);
38
39 ////////////////////////////////////////////// my define &
       declaration }
40
41 #ifdef _DEBUG
42 #define new DEBUG_NEW
43 #endif
44
45 // Cmy_Call_Forwarding_testingDlg 對話方塊
46
47 Cmy_Call_Forwarding_testingDlg::Cmy_Call_Forwarding_testingDlg(CWnd*
       pParent /*=NULL*/)
48     : CDialog(Cmy_Call_Forwarding_testingDlg::IDD, pParent)
49 {
50     m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
51 }
52
53 void Cmy_Call_Forwarding_testingDlg::DoDataExchange(CDataExchange*
       pDX)
54 {
55     CDialog::DoDataExchange(pDX);
56     DDX_Control(pDX, IDC_COMBO_FORWARD_PHONE, obj_combo_forward_phone
           );
57     DDX_Control(pDX, IDC_STATIC_STATUS, obj_static_status);
58     DDX_Control(pDX, IDC_EDIT_ADD_PHONE, obj_edit_add_phone);
```

```
59      DDX_Control(pDX, IDC_EDIT_ADD_COMMENT, obj_edit_add_comment);
60  }
61
62  BEGIN_MESSAGE_MAP(Cmy_Call_Forwarding_testingDlg, CDialog)
63  #if defined(_DEVICE_RESOLUTION_AWARE) && !defined(WIN32_PLATFORM_WFSP
        )
64      ON_WM_SIZE()
65  #endif
66      //}}AFX_MSG_MAP
67      ON_EN_CHANGE(IDC_EDIT_ADD_PHONE, &Cmy_Call_Forwarding_testingDlg
            ::OnEnChangeEditAddPhone)
68      ON_BN_CLICKED(IDC_BUTTON_START, &Cmy_Call_Forwarding_testingDlg::
            OnBnClickedButtonStart)
69      ON_BN_CLICKED(IDC_BUTTON_DELETE, &Cmy_Call_Forwarding_testingDlg
            ::OnBnClickedButtonDelete)
70      ON_BN_CLICKED(IDC_BUTTON_ADD, &Cmy_Call_Forwarding_testingDlg::
            OnBnClickedButtonAdd)
71      ON_BN_CLICKED(IDC_BUTTON_QUIT, &Cmy_Call_Forwarding_testingDlg::
            OnBnClickedButtonQuit)
72  END_MESSAGE_MAP()
73
74
75  // Cmy_Call_Forwarding_testingDlg 訊息處理常式
76
77  BOOL Cmy_Call_Forwarding_testingDlg::OnInitDialog()
78  {
79      CDialog::OnInitDialog();
80
81      // 設定此對話方塊的圖示。當應用程式的主視窗不是對話方塊時，
82      // 框架會自動從事此作業
83      SetIcon(m_hIcon, TRUE);          // 設定大圖示
84      SetIcon(m_hIcon, FALSE);         // 設定小圖示
85
86      // TODO: 在此加入額外的初始設定
```

```
87
88      // 從檔案讀取使用者之前儲存的電話號碼並將其設定到裡去Combobox
89
90      CString str_file;
91      CString str_phone;
92      CString str_temp;
93      CStdioFile file;
94      int index;
95
96      obj_combo_forward_phone.ResetContent ();
97      if (file.Open ((theApp.get_dir_path () + TEXT(FILE_PHONE_DATA)),
            CFile::modeRead))
98      {
99          while (file.ReadString (str_file) != FALSE /*|| str_file.
                GetLength () != 0*/)
100         {
101             obj_combo_forward_phone.InsertString (
                    obj_combo_forward_phone.GetCount(), str_file);
102         }
103     }
104     file.Close ();
105
106     // 從檔案中讀取目前程式狀態 (Start or Stop)
107     // 若為 Start 則設定一個 timer 並開始倒數
108     if (file.Open ((theApp.get_dir_path () + TEXT(FILE_STATE_DATA)),
            CFile::modeRead) && file.ReadString (str_file) != FALSE)
109     {
110         index = 0;
111         str_temp = str_file.Tokenize (TEXT("␣"), index);
112         if (str_temp.Compare (TEXT("Start")) == 0)
113         {
114             str_phone = str_file.Tokenize (TEXT(""), index);
115             SetDlgItemText (IDC_BUTTON_START, TEXT("Stop"));
116             KillTimer (ID_TIMER);
```

```
117             SetTimer (ID_TIMER, 1024, timer_callback);
118             g_start_time = GetTickCount ();
119         }
120     }
121     file.Close ();
122
123     // 設定下拉式選單選擇的號碼
124     for (index = 0; index < obj_combo_forward_phone.GetCount (); ++
            index)
125     {
126         obj_combo_forward_phone.GetLBText (index, str_temp);
127         if (str_phone.Compare (str_temp.Mid (0, str_phone.GetLength
                ())) == 0)
128         {
129             break;
130         }
131     }
132     if (index < obj_combo_forward_phone.GetCount ())
133     {
134         obj_combo_forward_phone.SetCurSel (index);
135     }
136     else
137     {
138         obj_combo_forward_phone.SetCurSel (0);
139     }
140
141     // 更新電池狀態
142     UpdateBattState(SN_POWERBATTERYSTATE_ROOT,
            SN_POWERBATTERYSTATE_PATH, SN_POWERBATTERYSTATE_VALUE,
            SN_POWERBATTERYSTATE_BITMASK, IDC_BATTSTATE);
143     // 偵測電池狀態是否有改變
144     register_callback ();
145
146     return TRUE;  // 傳回，除非您對控制項設定焦點 TRUE
```

```
147 }
148
149 #if defined(_DEVICE_RESOLUTION_AWARE) && !defined(WIN32_PLATFORM_WFSP
    )
150 void Cmy_Call_Forwarding_testingDlg::OnSize(UINT /*nType*/, int /*cx
    */, int /*cy*/)
151 {
152     if (AfxIsDRAEnabled())
153     {
154         DRA::RelayoutDialog(
155             AfxGetResourceHandle(),
156             this->m_hWnd,
157             DRA::GetDisplayMode() != DRA::Portrait ?
158             MAKEINTRESOURCE(
                IDD_MY_CALL_FORWARDING_TESTING_DIALOG_WIDE) :
159             MAKEINTRESOURCE(IDD_MY_CALL_FORWARDING_TESTING_DIALOG));
160     }
161 }
162 #endif
163
164 // 藉由 RegistryNotifyCallback 函式註冊，
165 // 當電池狀態改變時會呼叫 battery_stat_callback 函式
166 HRESULT Cmy_Call_Forwarding_testingDlg::register_callback ()
167 {
168     HRESULT hr;
169     NOTIFICATIONCONDITION nc;
170
171     // Make sure we aren't already registered.
172     unregister_callback ();
173
174     // Notify us of any change in the value and set the bitmask of
        the value to check.
175     nc.ctComparisonType = REG_CT_ANYCHANGE;
176     nc.dwMask           = SN_POWERBATTERYSTATE_BITMASK;
```

```
177     nc.TargetValue.dw    = 0;
178
179     // Register battery state notification.
180     hr = RegistryNotifyCallback(
181                 SN_POWERBATTERYSTATE_ROOT,
182                 SN_POWERBATTERYSTATE_PATH,
183                 SN_POWERBATTERYSTATE_VALUE,
184                 battery_state_callback,          // This notification
                        uses a callback.
185                 0,
186                 &nc,
187                 &g_hNotify
188                 );
189     return hr;
190 }
191
192 // 取消 RegistryNotifyCallback 函式的註冊
193 HRESULT Cmy_Call_Forwarding_testingDlg::unregister_callback ()
194 {
195     RegistryCloseNotification(g_hNotify);
196     return S_OK;
197 }
198
199 // 從 AUTO_QUIT_TIME 開始倒數的 callback 函式
200 // 並將剩餘秒數更新於 Quit button 上
201 void CALLBACK timer_callback (HWND hwnd,UINT iMsg,UINT TimerID,DWORD
       Time)
202 {
203     TCHAR temp[16];
204
205     Time -= g_start_time;
206     Time >>= 10;
207     if (Time < AUTO_QUIT_TIME)
208     {
```

```
209             swprintf (temp, TEXT("Quit (%d)"), (AUTO_QUIT_TIME - Time));
210             SetDlgItemText (*theApp.m_pMainWnd, IDC_BUTTON_QUIT, temp);
211         }
212         else
213         {
214             ((Cmy_Call_Forwarding_testingDlg*) theApp.m_pMainWnd)->
                    OnBnClickedButtonQuit();
215         }
216     }
217
218     // 用來偵測電池狀態的 callback 函式
219     void battery_state_callback (HREGNOTIFY h_notify, DWORD dw_user_data,
            const PBYTE p_data, const UINT cb_data)
220     {
221         DWORD dwBattState = 0;
222         PDWORD pdwData;
223         TCHAR szBattStateStr[STATUS_STRING_LEN];
224
225         StringCchCopy(szBattStateStr, STATUS_STRING_LEN, L"");
226
227         // Get the new value.
228         pdwData = (DWORD *) p_data;
229         dwBattState = *pdwData;
230
231         // Apply our bitmap mask and form the battery state as a string.
232         dwBattState = dwBattState & SN_POWERBATTERYSTATE_BITMASK;
233         get_battery_state_str (dwBattState, szBattStateStr); //
                ////////////
234
235         // Update the status text.
236         ((Cmy_Call_Forwarding_testingDlg*) theApp.m_pMainWnd)->
                obj_static_status.SetWindowTextW (szBattStateStr);
237     }
238
```

```
239 // 取得電池狀態，若狀態為 Charging 則啟用指定轉接；
240 // 若為一般狀態則停用指定轉接
241 void get_battery_state_str (DWORD dwBattState, LPTSTR
        pszBattStateStr)
242 {
243     // 啟用指定轉接前需要先將 TAPI 初始化
244     if (theApp.initialize_TAPI () != 0)
245     {
246         StringCchCat(pszBattStateStr, STATUS_STRING_LEN, L"
              initailize_TAPI()␣Error");
247         return;
248     }
249
250     if (dwBattState == 0)
251     {
252         ////////////////////////////////////////////////////
253         // 停用指定轉接
254         DWORD ttt = theApp.ProcessCallForwarding(NULL);
255         if (ttt > 0 && ttt < 0x80000000)
256         {
257             StringCchCat(pszBattStateStr, STATUS_STRING_LEN, L"Normal
                  ");
258         }
259         else
260         {
261             StringCchCat(pszBattStateStr, STATUS_STRING_LEN, L"
                  Normal_error");
262         }
263         ////////////////////////////////////////////////////
264     }
265     else
266     {
267         // Check each bit; specific battery drivers may set different
              combinations of bits.
```

```
268         if (dwBattState & 1)
269         {
270             StringCchCat(pszBattStateStr, STATUS_STRING_LEN, L"Not␣
                    Present␣");
271         }
272
273         if (dwBattState & 2)
274         {
275             ////////////////////////////////////////////////////////
276             // 從下拉式選單中取得目前選取的號碼
277             CString str_phone;
278             TCHAR temp[32]; //////////////
279             int index = ((Cmy_Call_Forwarding_testingDlg*) theApp.
                    m_pMainWnd)->obj_combo_forward_phone.GetCurSel ();
280             ((Cmy_Call_Forwarding_testingDlg*) theApp.m_pMainWnd)->
                    obj_combo_forward_phone.GetLBText (index, str_phone);
281             index = 0;
282             _tcscpy (temp, str_phone.Tokenize (TEXT("␣"), index));
283
284             // 啟用指定轉接
285             DWORD ttt = theApp.ProcessCallForwarding(temp);
286             if (ttt > 0 && ttt < 0x80000000)
287             {
288                 StringCchCat(pszBattStateStr, STATUS_STRING_LEN, L"
                        Charging␣␣␣(Call␣Forwarding)");
289             }
290             else
291             {
292                 StringCchCat(pszBattStateStr, STATUS_STRING_LEN, L"
                        Charging␣␣␣(Call␣Forwarding)␣␣Error");
293             }
294             ////////////////////////////////////////////////////////
295         }
296
```

```
297         if (dwBattState & 4)
298         {
299             StringCchCat(pszBattStateStr, STATUS_STRING_LEN, L"Low ")
                ;
300         }
301
302         if (dwBattState & 8)
303         {
304             StringCchCat(pszBattStateStr, STATUS_STRING_LEN, L"
                Critical");
305         }
306     }
307
308     // 結束使用 TAPI
309     theApp.shutdown_TAPI ();
310 }
311
312 // 更新電池狀態
313 void UpdateBattState(HKEY hKey, LPCWSTR wszSubKey, LPCWSTR wszName,
    DWORD dwBitMask, int nControlID)
314 {
315     HRESULT hr;
316     DWORD dwRegValue;
317     DWORD dwBattState;
318     TCHAR szBattStateStr[STATUS_STRING_LEN];
319
320     StringCchCopy(szBattStateStr, STATUS_STRING_LEN, L"");
321
322     // Read the new value.
323     hr = RegistryGetDWORD(hKey, wszSubKey, wszName, &dwRegValue);
324
325     if (hr == S_OK)
326     {
327         // Only look at specified bits.
```

```
328          dwBattState = dwRegValue  & dwBitMask;
329          get_battery_state_str(dwBattState, szBattStateStr);
330      }
331
332      // Update the status text.
333      ((Cmy_Call_Forwarding_testingDlg*) theApp.m_pMainWnd)->
            obj_static_status.SetWindowTextW (szBattStateStr);
334  }
335
336  // 將下拉式選單的號碼存於檔案
337  bool Cmy_Call_Forwarding_testingDlg::write_phone_data ()
338  {
339      CString str_file;
340      CStdioFile file_phone;
341      int i;
342
343      if (! file_phone.Open ((theApp.get_dir_path () + TEXT(
            FILE_PHONE_DATA)), CFile::modeCreate | CFile::modeWrite))
344      {
345          file_phone.Close ();
346          return false;
347      }
348      for (i = 0; i < obj_combo_forward_phone.GetCount (); ++i)
349      {
350          obj_combo_forward_phone.GetLBText (i, str_file);
351          file_phone.WriteString (str_file);
352      }
353      file_phone.Close ();
354      return true;
355  }
356
357  // 將程式狀態(Start/Stop)以文字方式存於檔案
358  bool Cmy_Call_Forwarding_testingDlg::write_state_data ()
359  {
```

```
360     CString str_file, str_action;
361     CStdioFile file_state;
362
363     if (! file_state.Open ((theApp.get_dir_path () + TEXT(
            FILE_STATE_DATA)), CFile::modeCreate | CFile::modeWrite))
364     {
365         file_state.Close ();
366         return false;
367     }
368     GetDlgItemText (IDC_BUTTON_START, str_action);
369
370     if (str_action.Compare (TEXT("Start")) != 0)    // != "Start" =>
            have registered the notification
371     {
372         obj_combo_forward_phone.GetLBText (obj_combo_forward_phone.
                GetCurSel (), str_file);
373         file_state.WriteString (TEXT("Start ") + str_file);
374     }
375     else
376     {
377         file_state.WriteString (TEXT("Stop"));
378     }
379
380     file_state.Close ();
381     return true;
382 }
383
384 void Cmy_Call_Forwarding_testingDlg::OnEnChangeEditAddPhone()
385 {
386     // TODO: 如果這是  RICHEDIT 控制項，控制項將不會
387     // 傳送此告知，除非您覆寫 CDialog::OnInitDialog()
388     // 函式和呼叫 CRichEditCtrl().SetEventMask()
389     // 讓具有 ENM_CHANGE 旗標 ORed 加入遮罩。
390
```

```
391     // TODO: 在此加入控制項告知處理常式程式碼
392 }
393
394 // 當 Start 或 Stop 鍵被按下時，
395 // 則啟用或停用指定轉接與電池狀態之偵測
396 void Cmy_Call_Forwarding_testingDlg::OnBnClickedButtonStart()
397 {
398     CString temp;
399     GetDlgItemText (IDC_BUTTON_START, temp);
400
401     if (temp.Compare (TEXT("Start")) == 0)
402     {
403         UpdateBattState(SN_POWERBATTERYSTATE_ROOT,
              SN_POWERBATTERYSTATE_PATH, SN_POWERBATTERYSTATE_VALUE,
              SN_POWERBATTERYSTATE_BITMASK, IDC_BATTSTATE);
404         SetDlgItemText (IDC_BUTTON_START, TEXT("Stop"));
405         theApp.register_app ();
406     }
407     else
408     {
409         KillTimer (ID_TIMER);
410         SetDlgItemText (IDC_BUTTON_QUIT, TEXT("Quit"));
411         SetDlgItemText (IDC_BUTTON_START, TEXT("Start"));
412         theApp.unregister_app ();
413     }
414 }
415
416 // 當 Delete 鍵被按下時，將下拉式選單的號碼刪除
417 void Cmy_Call_Forwarding_testingDlg::OnBnClickedButtonDelete()
418 {
419     int index;
420     if (obj_combo_forward_phone.GetCount () == 0)
421         return;
422     index = obj_combo_forward_phone.GetCurSel ();
```

```
423     if (index == CB_ERR || MessageBox (TEXT("Delete it?"), TEXT("
            Warning"), MB_OKCANCEL) != IDOK)
424         return;
425     obj_combo_forward_phone.DeleteString (index);
426     write_phone_data ();
427 }
428
429 // 當 Add 鍵被按下時，新增號碼至下拉式選單
430 void Cmy_Call_Forwarding_testingDlg::OnBnClickedButtonAdd()
431 {
432     CString str_add_phone, str_add_comment;
433
434     obj_edit_add_phone.GetWindowText (str_add_phone.GetBuffer (
            EDIT_MAX), EDIT_MAX);
435     obj_edit_add_comment.GetWindowText (str_add_comment.GetBuffer (
            EDIT_MAX), EDIT_MAX);
436     str_add_phone.ReleaseBuffer ();
437     str_add_comment.ReleaseBuffer ();
438     if (str_add_phone.Compare (TEXT("")) == 0)
439         return;
440
441     obj_combo_forward_phone.InsertString (obj_combo_forward_phone.
            GetCount (), (str_add_phone + TEXT(" ") + str_add_comment));
442     obj_edit_add_phone.SetWindowText (TEXT(""));
443     obj_edit_add_comment.SetWindowText (TEXT(""));
444     write_phone_data ();
445 }
446
447 // 當 Quit 鍵被按下時，將程式關閉
448 void Cmy_Call_Forwarding_testingDlg::OnBnClickedButtonQuit()
449 {
450     KillTimer (ID_TIMER);
451     write_state_data ();
452     unregister_callback ();
```

```
453     EndDialog (IDCLOSE);
454 }
455
456 // 當 OK 鍵被按下時，取消電池狀態之偵測，並將程式關閉
457 void Cmy_Call_Forwarding_testingDlg::OnOK()
458 {
459     KillTimer (ID_TIMER);
460     write_state_data ();
461     unregister_callback ();
462     EndDialog (IDOK);
463     CDialog::OnOK ();
464 }
```

附錄 B　CFA 使用者介面之介紹

本附錄藉由圖 B.1 至圖 B.7 介紹 CFA 應用之圖形化使用者介面。

圖 B.1 為 CFA 應用的首頁，提供下列資訊：**電池狀態** (圖 B.1 (1)) 以文字顯示手機目前的電池狀態。**轉接號碼列表** (圖 B.1 (2)) 列出使用者預設的號碼，並在每個號碼後面加上註解文字，以方便使用者辨識。當偵測到觸發事件時，CFA 應用將此列表中被選取的號碼作為轉接號碼，啟用指定轉接服務。**啟用按鈕** (圖 B.1 (3)) 被按下時啟用電池狀態之偵測，並確保當手機電池狀態發生改變時，此應用能處於執行狀態。意即按下此按鈕並將此應用關閉之後，若電池狀態發生改變，手機能自動將此應用執行起來。**結束按鈕** (圖 B.1 (4)) 被按下時將此應用關閉。**刪除按鈕** (圖 B.1 (5)) 被按下時將轉接號碼列表目前選取的號碼刪除。**新增預設號碼之輸入欄位** (圖 B.1 (6)) 讓使用者能輸入欲

圖 B.1: CFA 應用之首頁