國 立 交 通 大 學

資訊科學與工程研究所
碩 士 論 文

多介面裝置之遍佈式虛擬區域網路的設計與實作

Design and Implementation of a Ubiquitous Virtual LAN Platform for Multi-Interface Devices

研 究 生： 黃勇智

指導教授： 曾建超 教授

中 華 民 國 九 十 六 年 六 月

多介面裝置之遍佈式虛擬區域網路的設計與實作

Design and Implementation of a Ubiquitous Virtual LAN Platform for Multi-Interface Devices

研 究 生：黃勇智　　Student：Yong-Zhi Huang

指導教授：曾建超　　Advisor：Chien-Chao Tseng

國 立 交 通 大 學

資 訊 科 學 與 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

June 2007

HsinChu, Taiwan, Republic of China (R.O.C.)

中 華 民 國　九 十 六　年　六　月

多介面裝置之遍佈式虛擬區域網路的設計與實作

研究生：黃勇智　　指導教授：曾建超

國立交通大學
資訊學院
資訊科學與工程研究所

摘要

本研究針對多網路介面的行動裝置，設計與實作一套虛擬區域網路平台以支援行動裝置進行連線不中斷(Session Continuity)的網路換手(Handoff)。隨著多種有線及無線網路的發展以及 VoIP 的出現，在異質性網路裡漫遊成了最時尚的需求。在網路漫遊時，一定要面對的問題就是網路換手。對於移動終端來說，在不同的網路介面之間做網路換手時，則原本所建立好的 Session，會出現斷線的問題，這會使服務中斷，大大地影響服務的品質。

在本論文中，我們提出 Ubiquitous Virtual LAN (U-VLan)的概念，在多介面的移動終端上實作一個虛擬網路卡，利用這個虛擬網路卡可以形成一個虛擬區域網路，移動終端藉由虛擬網路卡彼此傳送資料，讓連線的兩端均認為對方在同一個網域下。而實際情形是 U-VLan 會把收到的資料經過處理，透過 socket 轉送到其他實體的介面輸出。U-VLan 可以整合許許多多的異質網路介面，達到延續連線的目的並提供無縫式的漫遊服務。現有絕大部分的網路程式皆無須經過任何的修改便可正常運作，而在設計應用程式時也不用考慮換手的問題，只須專心於應用層面的開發。

我們在微軟的 Windows XP SP2 上，實作出一個 U-VLan 的網路環境，實驗結果說明 U-VLan 對於整合許多種的網路介面很有幫助。

Design and Implementation of a Ubiquitous Virtual LAN Platform for Multi-Interface Devices

Student：Yong-Zhi Huang Advisor：Chien-Chao Tseng

Institute of Computer Science and Engineering
College of Computer Science
National Chiao Tung University

ABSTRACT

In this paper, we design and implement a virtual LAN platform that can support session continuity handoff for multi-interface devices. With the advance of network technologies and portable devices, a mobile node (MN) equipped with multiple network adapters may now roam among heterogeneous networks. However when an MN encounters a handoff, that is, changing its point of attachment, from one network to another, the ongoing session of the MN may be disrupted or even broken. Such session disruption may influence significantly the quality of services of real-time services.

In this thesis we introduce the concept of Ubiquitous Virtual LAN (U-VLan) to support session continuity handoffs for MNs with multiple interfaces. We first implement a virtual network interface (NIC) on an MN. Then the MNs that want to form a virtual LAN (VLan) can configure their virtual NICs with the same subnet prefix. Two MNs on the same VLan can communicate with each other using the virtual NICs (VNICs) as they were on the same physical LAN. However, physically, the U-VLan platform software tunnels an outgoing packet through a physical NIC of the sending node to a physical NIC of the destination node. When receiving the tunneled packet via the physical NIC, the U-VLan platform software on the destination node will de-tunnel the packet and delivers the original packet to the virtual NIC of the destination node.

The U-VLan platform provides the upper-layer network applications an illusion of an always connected NIC. Most of the existing network applications running on a U-VLan enabled MN can work continuously without any modification even if the MN switches the underlying physical NICs and attaches to a new network. We have implemented the U-VLan Platform on Microsoft's Windows XP SP2. Experimental results show that U-VLan is very effective in integrating heterogeneous NICs and can indeed provides session continuity for U-VLan enabled MNs.

致謝

研究所兩年過的很快也很充實，在這段期間最主要是要感謝我的指導教授-曾建超教授，在研究、求知方法等給我的建議、指導與輔導，尤其是在報告與表達上，老師不厭其煩的提供他的經驗、想法，帶領我完成兩年研究生生崖，並在論文的撰寫期間給了我許多重要的意見，使得本論文得以順利的完成。

還要感謝在這兩年中修課的教授，在這兩年修課當中，各位教授，認真辛苦，費盡心思的授課與教導，讓我受益良多。此外還要感謝實驗室的每一位學長，在我剛進實驗室時，親切地關心，給我許多協助與幫忙。還要特別感謝同窗兩年的同學們，在課業與生活上對我的照顧。感謝在實驗室的每一位，陪我度過實驗室的生活。也感謝我的家人，特別是養育我、栽培我的父母，讓我全心全意地完成學業與論文，在此獻上我最衷心的感謝。另外對所有關心我的人，敬上最誠摯的謝意。

謹以此論文，獻給我最親愛的家人與關心我的師長朋友們。

願 平安 健康 成就感

目錄

圖目錄

第一章 緒論

1.1 研究動機

目前有線網路以及無線網路的普及使得一個移動裝置上會有著許許多多不同的網路存取介面。以往的網際網路，均是透過有線的方式來連接，並不會想到漫遊的需求。如今，有了無線網路之後，漫遊的需求明顯增加。例如：可能現在在室內辦公，所以就接有線的網路，因爲比較快，也比較穩定，之後如有需要，必須離開一下，到另外一間會議室開會，因此在移動的時候，就換成無線區域網路，也因爲有了無線網路的環境而有不同的使用習慣，換手 (Handoff/Handover)便成了新環境中需要考慮的問題之一。

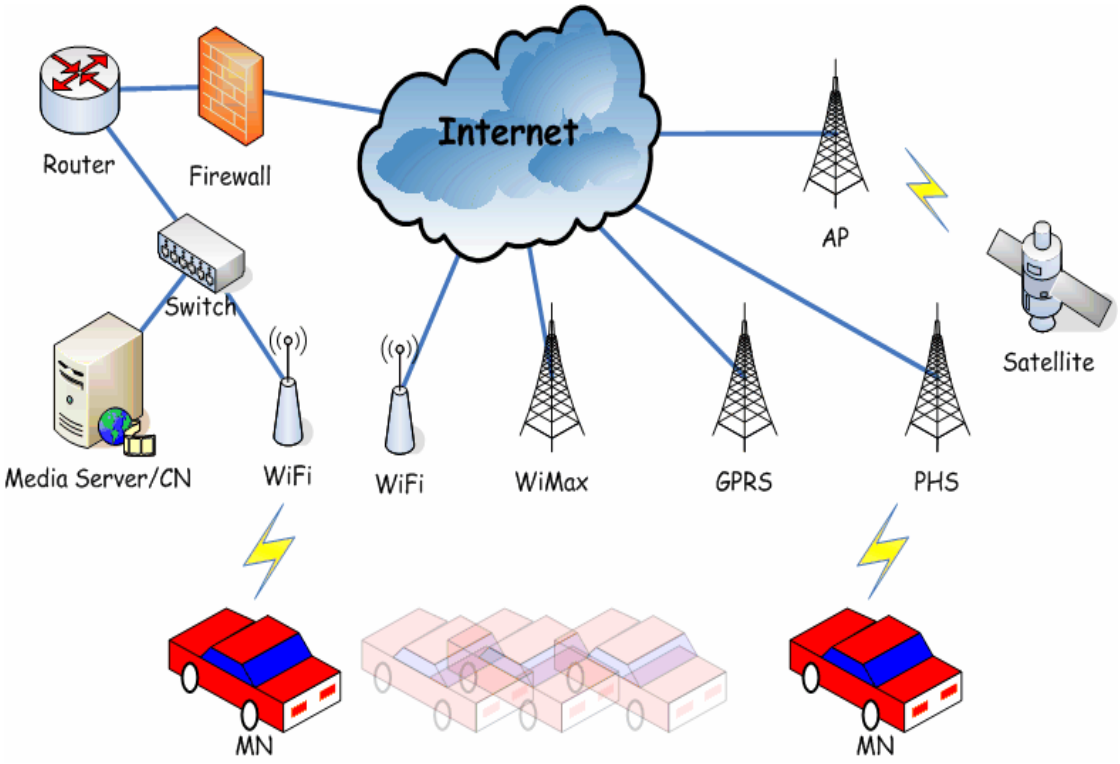

圖1: 多種網路介面環境

當一個行動裝置有著許許多多的網路存取介面，如 Ethernet、Wifi、GPRS、WiMAX、PHS…，就會有異質網路的整合性問題，這又包含了垂直換手 (Vertical Handoff)以及水平換手 (Horizontal Handoff)的部份。所以對於多介面網路存取裝置，要漫遊時，必須考量許多部分。再者，因爲多介面的換手，一個 session 的連線勢必會受到斷線的衝擊，斷線的問題就產生了。所以要如何提供多介面網路終端一個 Session Continuity 的網路環境是本論文想要達到的目標。

如圖 1，一台汽車-MN-上裝有一套監控軟體的客戶端 (client)，CN 上裝有監控軟體的伺服端 (server)。MN 先透過 Wifi 連上網際網路，並連線到 CN，MN 接著就透過攝影機把車上看到的畫面傳送到 CN，當 MN 一直持續的把影像畫面傳送給 CN 的時候，因爲 MN 在移動，所以 Wifi 的網路介面便失靈，還好還有另一個介面，也就是 GPRS 撥接的網路，因此必須換一個 IP 才能上網繼續和 CN 連線，然而變換 IP 對於上層的應用程式或是以 TCP 通訊協定來說，先前連線的 session 勢必會斷線，必須重新建立，因此 session 斷線問題是存在的。所以如果可以提供 Session Continuity 的網路環境，對應用程式來說是好的，這樣應用程式在設計的時候便可以不用考慮斷線的問題。

要提供 Session Continuity 的網路環境，則必須考量換手的效應。而換手是許多網路層的問題 (multi-layer issue)，包含實體層 (Physical Layer)、資料鏈結層 (Data Link Layer)、網路層 (Network Layer)、傳輸層 (Transport Layer)、以及應用層 (Application Layer)等。對於換手來說，如果底層的通訊協定對換手有處理，上層的通訊協定就可以不需要考慮換手的問題，如果底層通訊協定並沒有換手的機制，則上層通訊協定就必須要考慮換手的問題。

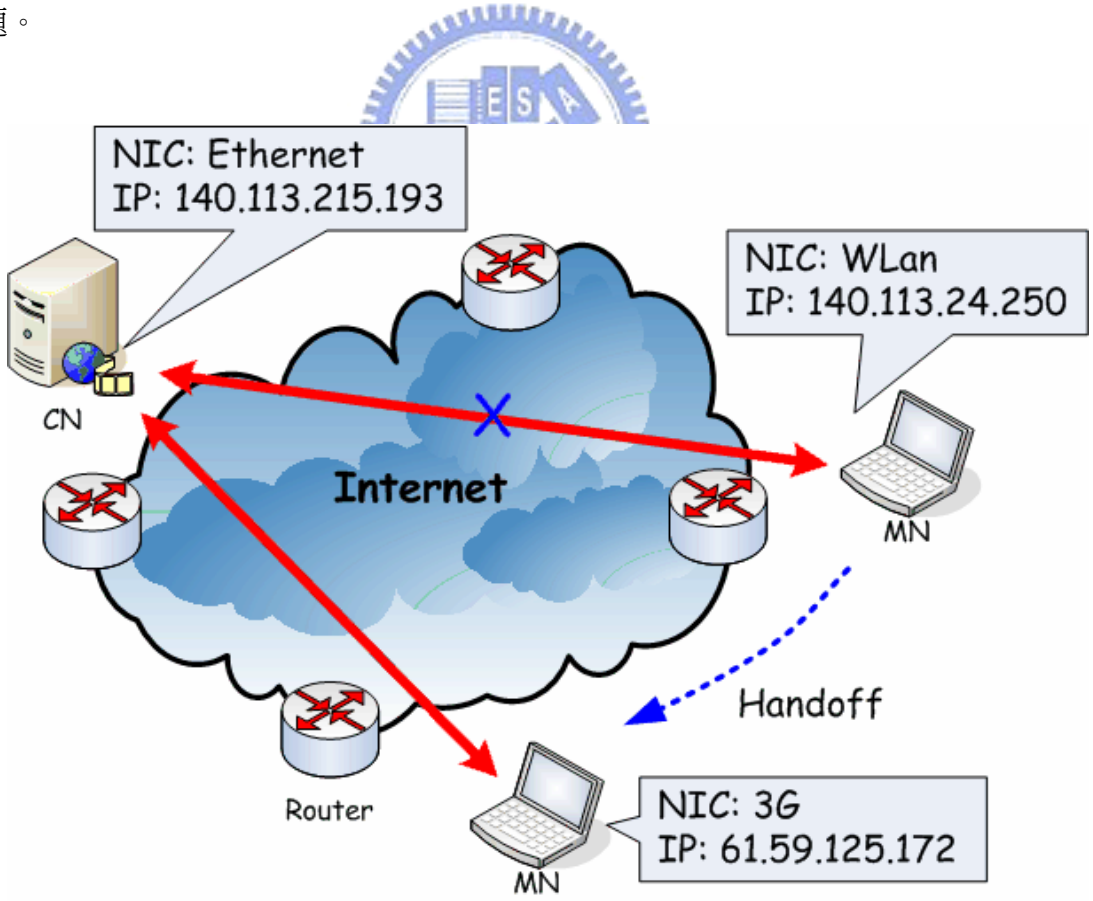

圖2: 換手斷線示意

舉個例子(圖 2)，以無線區域網路 (WLAN)來說，當一個網路終端已經連上一個基

地台，並且得到了一個網路的 IP，在上面的應用程式便用此 IP 和另外一台電腦溝通，然而這台終端因爲必要，需移動到另外一個網域，也就是切換到 3G 所撥接的網域，此時 WLAN 斷線了，網路的實體層切換過去，然對上層來說，並沒有幫助，上層的網路層也跟著改變 (IP 改變)，對於傳輸層，以 TCP 爲例，下層的 IP 一旦改變，整個 session 的連線就會斷掉，而應用層，原本連線的 socket 也會斷掉，因此必須要重新建立，如圖中所示。因此，對於換手的情形，應用程式如果自己解決，自行自動重新連線，自動處理換手的事務是解決換手問題的最好辦法，因爲實際在運作的功能就是應用程式本身最知道，當然自己來處理是最實在的。然而現今已經存在許多應用程式，許多以往較老的應用程式，這些應用程式如果要重新設計或是重新編譯是很費功夫的事，甚至要耗費的成本需要許多，不合乎效益，所以必須要另找方案。本論文所要提出的解決方案，就是要解決上述的問題。

1.2 研究目標與特色

1.2.1 目標

本論文的主要目標就是對於有多網路介面的裝置來說，可以設計一個遍佈式的虛擬區域網路平台，用以建構一個具有 Session Continuity 的網路環境，然後實作出來，並測試該平台之性能。

1.2.2 特色

- Session Continuity。當換手的時候可以繼續保持原本 session 的連線，而不會有斷線的問題。
- No modification to applications。對應用程式來說，可以不必重新設計，重新編譯，甚至新的應用程式，可以不需要考慮換手的問題。
- No modify protocol modules。對於現有的網路通訊協定，可以不必更動修改的盡量不要更動修改，而自行定義的通訊協定盡可能的相容於現有的環境下。
- Efficient handoff。有效率的換手，對於換手的延遲，可以最小化當然是最好，如果能避免這個延遲就避免。

1.3 章節概要

在此之後，第二章會介紹一些相關的背景知識，第三章會介紹目前的一些解決方案。第四章會介紹本系統是如何設計以及運作，並說明本系統的一些概念及想法。第五章會詳細的介紹實作的部份，也就是實際軟體元件在作業系統中所扮演的角色，以及有關的運作流程。接著就是第六章，也就是實驗的部份，這部份會測試本系統對於換手延遲的時間，以及本系統的效能。第七張就是結論以及未來的發展。最後是參考文獻。

第二章 背景知識

2.1 無線通訊技術簡介

在成熟的有線網路下，乙太網路已經成爲最多人使用的連接方式之一，而興新的光纖網路，也有成長的趨勢，因爲光纖的好處是可以不受環境電磁波的干擾，所以一般的海底電纜，現在都改用光纖網路來連接。有線的存取因爲發展的比較早，所以該解決的問題都解決了，而無線呢，目前無線網路的裝置以及設備越來越多，在最近的幾年內已經發展出許許多多的無線網路的標準，有的是屬於小範圍的網路，有的是中等距離的，當然也有大範圍的，有的可以提供高品質的傳輸速率，有的提供穩定的存取，也有的提供較低成本的連線。像現在最常看到的無線網路有：GPRS，3G，PHS，Wifi，WiMAX …等各式各樣的網路標準。下面簡單介紹一下各種標準。

- GSM/GPRS

GSM 全名爲：Global System for Mobile Communications，由歐洲所開發的數位行動電話網路標準，它的開發目的是讓全球各地共同使用一個行動電話網路標準，讓用戶使用一部手機就能行遍全球。GSM 系統包括 GSM900、GSM1800及GSM1900 等幾個頻段。GSM 系統有幾項重要特點：防盜拷能力佳、網路容量大、手機號碼資源豐富、通話清晰、穩定性強不易受干擾、資訊靈敏、通話死角少。

而 GPRS 也就是 General Packet Radio Service。簡單地說，GPRS 將 Packet Switching 的概念引進到 GSM 的系統中，而 Packet Switching 使用的是一種頻道分享的概念。傳統的 GSM 是一個 Circuit Switching 的網路。GPRS 具有的優勢包括：改善無線電頻道的使用現狀。提供低成本，品質穩定的服務給更多的客戶。快速的連接。GSM 及 GPRS 可以同時存在而互不干擾。可與其他 IP (網際網路協定)相連接。

- 3G

3G是 Third Generation 的簡稱是指第三代行動通訊。第一代行動通訊是類比無線網路，第二代是目前廣爲使用的 GSM 和 CDMA。3G將具有更寬的頻寬，其傳輸速度最低爲 384 Kbps，最高爲 2 Mbps，不僅能傳輸語音，還能傳輸資料，從而提供快速、方便的無線應用，如無線接取網際網路。從第二代行動通訊到 3G，其中的過渡技術便是上述的GPRS。

- PHS

PHS 中文全名爲低功率行動電話。英文名稱爲 Personal Handy-phone System。PHS 系統是日本自行研發的數位式無線電話系統。發射功率遠低於一般的 GSM/GPRS 行動電話，所以它是唯一可在醫院使用之行動通訊系統。以日本的系統建設經驗來看，PHS 最適用於高密度的都會區，可提供與 GSM/GPRS 系統同等的功能。其 64 Kbps 的行動數據傳速率，更是 GSM/GPRS 望塵莫及的，目前日本的通訊傳輸速率更達到 128 Kbps。而相較於 GSM/GPRS 行動電話將 64K 的語音壓縮至 9.6K 而導致失真較爲嚴重的情況，PHS 只將語音壓縮至 32K，通話音質佳，PHS 以接近市話的音質略勝 GSM 一籌。PHS 手機省電，其待機時間和通話時間長，一般 PHS 手機通常最低待機時間都在300 小時以上，因爲手機發射功率較低，PHS 手機的發射功率僅是 GSM 手機的六十分之一，即使長時間使用也不會有頭痛、目眩等使用 GSM 手機後的不適症狀。

- Wifi

Wifi 是 Wireless Fidelity 的縮寫，是一種無線資料傳輸技術與規格，一般可稱爲 WLAN (Wireless Local Area Network)，也就是無線區域網路。它是 IEEE (Institute of Electrical and Electronics Engineers，電機及電子工程師學會)所定義的無線網路通信的工業標準 IEEE 802.11，目前市面上常見的產品規格爲 IEEE 802.11 a/b/g/g+，最快可以提供 108Mbps 的連線速度，無線範圍適合在室內使用 (無障礙物 200 公尺以內)。依這些規格生產的無線產品通過 WECA (Wireless Ethernet Compatibility Alliance，無線乙太網路相容聯盟)實驗室的相容性測試後，WECA 就會授與無線相容性認證，也就是一般所謂的 Wifi 認證，產品上就可以冠上這個標章。

- WiMAX

WiMAX (World Interoperability for Microwave Access)是一種無線都會網路(WMAN)技術，是針對微波頻段和毫米波頻段提出的一種新的通訊標準，可以視爲無線網路的最後一哩，提供 Cable 及 xDSL 等高速有線網路的另一種選擇。WiMAX 最大傳輸距離是三十英里，在 20MHz 的頻道上，最大資料傳輸速率可達 75Mbps，非可視範圍（non-line-of-sight）內約可傳遞 3 到 5 英里，採用 QPSK 調變模組下，可視範圍（line-of-sight）內約可傳遞高達 30 英里的距離。因爲 WiMAX 的傳距遠，傳速高，加上 IEEE 802.16 具有 QoS 的頻寬管理能力，以及追加行動通訊的支援，這也使許多人臆測 WiMAX 恐將阻礙 3G 的普及，甚至成爲 3G 殺手。

2.2 NDIS

要在微軟的平台上設計網路相關的系統，那不能不了解微軟在網路核心所定義的軟體架構，接下來將介紹 NDIS 這個 Windows 作業系統裡都會存在的元件。

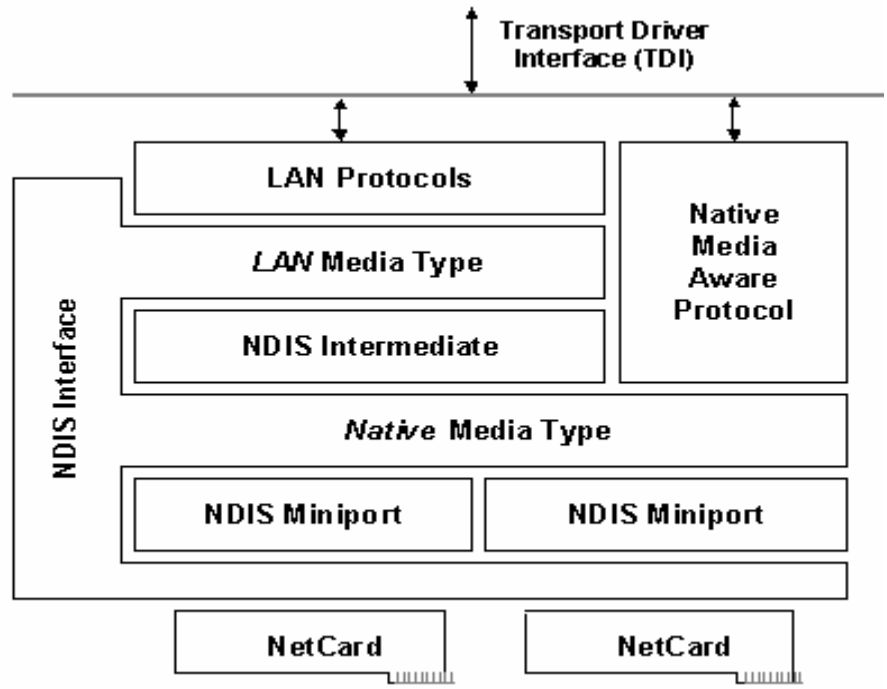

圖 3: NDIS drivers

NDIS 是 Network Driver Interface Specification (網路驅動介面規格)的縮寫爲 Microsoft 和 3Com 兩家公司所定義的，它爲網路驅動程式抽象了網路硬體，並指定了分層網路驅動程式間的標準介面，因此，它爲上層驅動程式抽象了管理硬體的下層驅動程式。NDIS 也維護了網路驅動程式的狀態訊息和參數，這包括到函數的指標，識別碼等。NDIS 本身就是一個函式庫，提供許許多多的網路 API 介面。NDIS 支援三種類型的驅動程式，迷你阜驅動程式 (Miniport driver)、中間層驅動程式 (Intermediate driver)、通訊協定驅動程式 (Protocol driver)，如圖 3 所示，下面將個別說明。

- **迷你阜驅動程式**

此驅動程式就是常說的網路卡驅動程式，它負責管理網路卡，包括透過網路卡發送和接收資料，它也爲上層驅動程式提供介面。此驅動程式一般由硬體製造商提供。而第四章要介紹的，也就是本論文所要開發的虛擬網路卡就屬於這一類的驅動程式，只是本論文裡的這支驅動程式並沒有實際的控制硬體。

- **中間層驅動程式**

此驅動程式通常位於迷你阜驅動程式和通訊協定驅動程式之間，是基於資料連結層和網路層之間的驅動程式。由於中間層驅動程式位在驅動層次的中間，所以上下兩層的驅動程式要溝通的資料均會經過此驅動程式，所以中間層驅動程式主要的用途之一就是過慮封包，其優點是能夠擷取所有的網路封包而且完整。此外中間層驅動程式還可以用來實現 VPN、NAT、PPPoE 以及 VLan，而本論文所提的虛擬網路卡爲何不開發在此，原因有二，其一因爲中間層，勢必所有的網路封包皆經過，需要處理不必要的封包，其二中間層驅動程式，事實上需要開發兩類的驅動程式，對上層來說，要開發迷你阜驅動程式的介面，對下層來說，要開發通訊協定驅動程式的介面比較麻煩，所以直接開發在迷你阜驅動程式那層比較實際。

- **通訊協定驅動程式**

此層的驅動程式，是位於 NDIS 架構的最高層，經常用做實現傳輸驅動堆疊的最底層，最常見的如 TCP/IP 或 IPX/SPX 堆疊。底層網路卡所收到的封包，會分別送到此層的每個驅動程式，所以通訊協定驅動程式會接收到所有的網路封包，至於要怎麼處理視通訊協定本身的設計而定。因此有些網路監控程式會設計一個沒有實際功能的通訊協定驅動程式

第三章 相關研究

本論文的目標就是要提供一個 Session Continuity 的網路環境，也就是可以提供無縫式的換手，而關於此類的研究相當多，大致上可以依照網路的階層來做區分，包含應用層 (Application Layer) [1][2][3]、傳輸層 (Transport Layer) [4][5][6][7][8][9][24] [28]、網路層 (Network Layer) [10][11][12][13][14][15]…，有研究提出虛擬網路介面來達到無縫式的換手 [31]，該架構提供了快速換手的機制。另外也有研究針對 VoIP 的應用，提出了 SCTP-based + 802.21 MIH function + Device Virtualization [28] 的 handoff 方法，該結果對於 VoIP 的環境表現很好。其上述的虛擬網路介面或是裝置虛擬化，和本論文要設計的系統類似，皆提供一個虛擬的網路介面，讓上層的通訊協定皆認爲底層有一個可以使用的網路介面。而下面將介紹目前常見的無縫式換手作法與優缺點。

3.1 Proxy Based

3.1.1 Proxy Base 簡介

此想法是利用一台 proxy server [1]幫忙，proxy server 上面會維護一個 IP 的位址表，上面會紀錄 MN (mobile node)端可以連上網路的所有 IP，當然會區分爲主要的以及備用的，只要 MN 移動到不同網域，就會向 proxy 註冊新的 IP 位址，所以 CN (Correspondent Node) 要傳送資料給 MN，就可以透過 proxy 很正確的傳送給 MN。

3.1.2 Proxy Base 分析討論

此方法很方便，只要在 MN 上稍微修改加入一個元件，它用來取得新網域的 IP，以及向 proxy 註冊新 IP，很容易實作。不過還是有些許的問題。

- 應用程式必須支援 Proxy

對於應用程式來說，必須支援 proxy，也就是應用程式本身需要考慮是否透過 proxy 來連線亦或是直接連線，如瀏覽網頁、檔案傳輸、…等應用。

- Proxy 的位置

因爲所有的傳送皆需要透過 proxy server，不管是 MN 送給 CN 或是 CN 送給 MN 都要經過它，如果 Proxy 離 MN 或是 CN 很遠，這樣就會影響網路的傳輸品質，所以它所在的位置便是重要的考量。

3.2 SIP Based

3.2.1 SIP 簡介

SIP (Session Initiation Protocol) [16][17]是一個由 IETF MMUSIC 工作組開發的通訊協定，應用於建立，修改和終止包括視訊，語音，即時通信，線上遊戲…等多種多媒體元素在內的互動式用戶會話。2000 年 11 月，SIP 被正式批准成爲 3GPP 通訊協議標準之一。SIP 與 H.323 一樣，是用於 VoIP 最主要的通訊協定之一。下面列出 SIP 的成員。

- SIP 成員

[User Agents]

User Agents 是 SIP 網路環境中的終端設備，它可以是 SIP 電話機或者在個人電腦端的 SIP 客戶端軟體，它包函 User Agent Client (UAC)以及 User Agent Server (UAS)，UAC 負責產生(建立)請求(Request)，而 UAS 負責產生依照請求產生應答(Response)。每個 SIP User Agent 都包含 UAC 以及 UAS 的功能。

[SIP Proxy]

SIP Proxy 負責將 User Agent 或者其他的 SIP Proxy 發出的請求代爲傳遞到另外一個 SIP 元件。當 User Agent 發出請求的時候，請求並不是直接傳送到目的端的 User Agent，而是經由一層層的 SIP Proxy 後才將請求訊息傳送到目的端的 User Agent，每個 SIP Proxy 都會決定出下一個路由且對請求訊息做適當的加工處裡以利訊息的傳遞。目的端的 User Agent 回覆結果的時候也是一樣會經由相反的路由將結果回覆給請求端的 User Agent。

[Redirect Server]

當 User Agent 或者 Proxy 所發出的請求傳送到 Redirect Server 時，Redirect Server 回覆 Redirect 訊息(3xx)，讓 User Agent 或者 Proxy 知道需要將訊息重新導向至另一個 SIP 元件。

[Registrar Server]

Registrar Server 提供 User Agent 進行註冊的介面，用以進行管理以及特定的服務，並且更新 Location Server 上的 User Agent 資訊或者更新其他的資料庫。

[Location Server]

Location Server 負責儲存 User Agent 的資訊，例如：URL、IP 位址、身分、特性等等。

由上面介紹的成員組成 SIP 的環境，在 MN 以及 CN 上皆會有一支 User Agent 存在，當 MN 移動到不同網域的時候，皆會向 SIP 的 Registrar Server 註冊他所在的 IP 位址，和 3.1 章 proxy base 的方法一樣，只是不同的是，MN 以及 CN 要建立連線的時候，並不是和 proxy base 的方法一樣，而是 CN 會去向 SIP 的 server 詢問如何和 MN 連絡，SIP server

回應 CN 之後，CN 便依照 SIP server 所提供的方式和 MN 連絡，可能是直接連線，也可能是透過 SIP proxy，一旦 MN 換手了，CN 可能就要再詢問 SIP server，然後重新建立連線，在 SIP 裡有一個詞 REINVITE，就是要重新建立 session 用的。

3.2.2 SIP 分析討論

SIP 的通訊協定是建立在應用層的方案，它解決了部分應用層的漫遊，這也難怪 3GPP 以它爲標準之一，然而它還是有缺點，就是對於應用程式來說，必須支援 SIP，也就是應用程式本身需要處理 SIP 的通訊協定，這點和 proxy 一樣。對於應用層的解決方案一般都是這樣，也就是應用程式是需要自己考慮換手的問題。

3.3 Multi-TCP Connections

3.3.1 Multi-TCP 簡介

此方法是藉由建立多條的 TCP 連線 [2]，切割封包然後同時使用這些連線來做傳輸，如果有其中一條連線中斷，則會交由其他條連線來補齊傳送的工作，該方法是抽換底層的函式庫，也就是 socket API，把單純的一條連線改成多條。

3.3.2 Multi-TCP 分析討論

此方法的優點就是實作在應用層，所以是很容易部署的，但也因爲在應用程式上面開發，所以必須設計一套 buffering 與 ACK 的機制，當某一條連線壅塞或是斷線，需要做放慢速度、重傳等動作，而要設計這樣的機制是相當複雜的。另外上述的動作在傳輸層已經做過的事再做一次，對系統來說很沒有效率。

3.4 Mobile IP

3.4.1 Mobile IP 簡介

因應個人電腦發展快速，行動手持式裝置日益普遍，網路節點離開原網路或進行網路漫遊的現象十分普遍，因此網際網路標準組織 IETF 成立了行動網際網路位址工作小組 (Mobile IP Working Group)以制訂支援網際網路位址行動能力 (Mobility)的相關標準，Mobile IP [10]的協定與相關技術因此得以制訂。

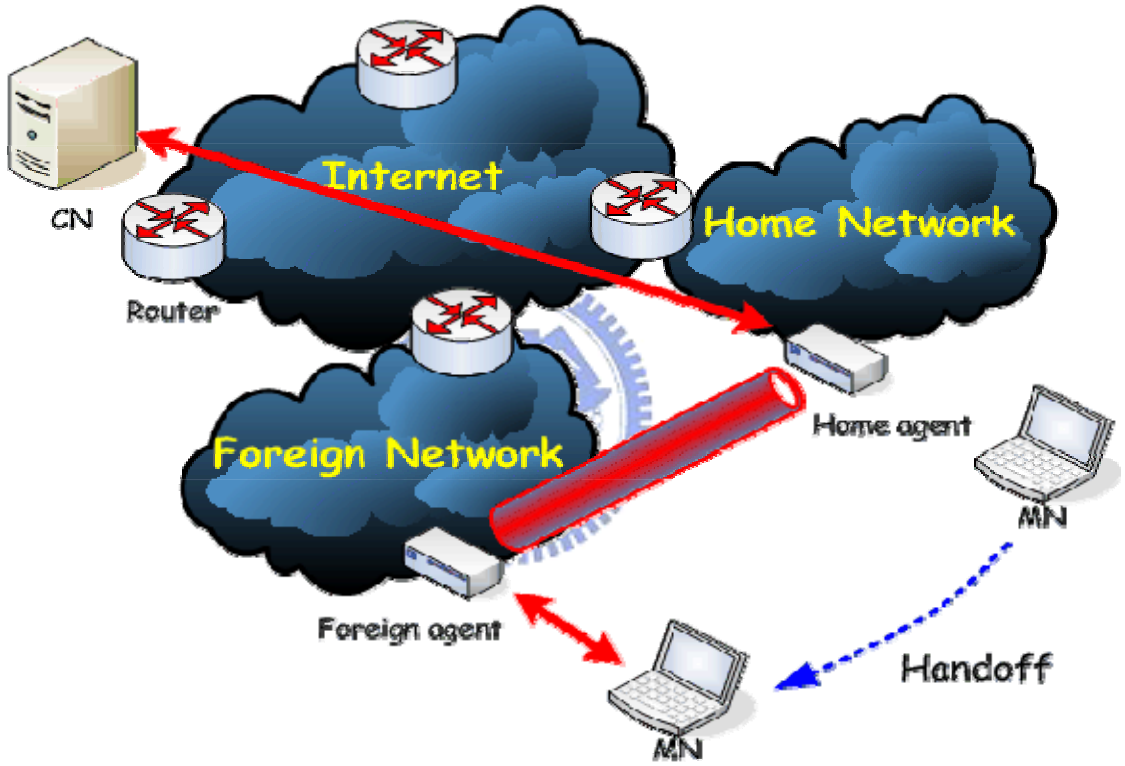

圖4: Mobile IP 架構圖

- Mobile IP 成員

[Mobile Node (MN)] 具備 IP 行動能力之網際網路節點，該節點可以漫遊其他網路但仍可以透過其原位址(Home Address)進行封包傳遞。

[Correspondent Node (CN)] 與 MN 進行通訊之任何一個網際網路節點。

[Home Network (HN)] MN 所在的家網路。

[Foreign Network (FN)] MN 漫遊到的其他網路。

[Home Agent (HA)] 在 MN 家網路上負責記錄 MN 位置並轉送封包到 FA。

[Foreign Agent(FA)] 在漫遊至其他網路，負責封包轉送之代理人。

[Care of Address (CoA)] MN 在漫遊至其他網路之暫時網路位址。

由 Mobile IP 架構圖來看，Mobile IP 中，每個 MN 都會有一個固定且唯一的 home address，所有要和 MN 連線的 CN 均可以直接連接該固定 IP 即可，因爲每個人看到 MN 的 IP 皆是同一個固定的 IP，所以對於 TCP 連線來說都一樣並不會有斷線的問題。而當 MN 從家網路移動出來，並切換到其他網域的時候，會跟當地的 FA 要求一個 Care of address，接著 MN 會跟 HA 註冊 CoA，讓 HA 知道 MN 移動到哪裡，Home agent 就可以將要送給 MN 的封包透過 FA 以隧道 (Tunnel)的方法順利送給 MN。

3.4.2 Mobile IP 分析討論

Mobile IP，顧名思義就是可移動的 IP，此方法因爲設計實作是在網路層 (Network Layer)，所以並不需要改變現存的 TCP 等通訊協定架構，應用層也不需要修改，但是還是有一些問題存在，下面簡單介紹之。

- **沒有效率的資料路徑**

爲什麼會說沒有效率呢，因爲 Mobile IP 會產生三角繞路的問題。對 CN 來說，所有的封包都還是一樣要送到 MN 的 home address，因此，不管 MN 在哪，封包均需要經過 HA，這樣就變的沒有效率。

- **需要部署網路元件**

以 Mobile IP 的架構來說，需要部署 Home Agent 及 Foreign Agent，這需要相當大的花費，很不實在。

- **NAT 以及防火牆穿透問題**

這個問題，對 Mobile IP 來說是個特別的問題，因爲 NAT 對內部私有網路來說如果沒有主動往外部網路送出封包，外部網路送進來的封包會被過濾丟掉。雖然有人提出一些解決的辦法，如美國專利編號 US2003/0123421，范榮軒學長 [22]…等屬於 (IP in IP) 的技巧，但是如果遇到需要分辨網路封包型態的 NAT，只要 IP 之上不是 TCP，UDP，ICMP 等基本的網路封包，便會過濾掉，那些方法便行不通，只有 RFC 3519 [23]所定義的方法 (IP in UDP)可以正常運作，然而如果遇到會看內容的防火牆 (如公司內部)，UDP 類型的封包如果不是 DNS 的查詢，可能就會被丟掉這樣對 Mobile IP 也是個問題。

- **封包遺失**

封包會遺失，是因爲當 MN 在移動的時候，也就是當 MN 離開原本網路到取得 Care of Address 並向 HA 註冊的中間時間，HA 還是以爲 MN 依然在原本的地方，所以會繼續傳送封包過去，然而這些封包 MN 接收不到，所以產生了封包遺失。

- **換手所需的時間長(延遲長)**

Mobile IP 的換手時間是大家公認的，至少在 1、2 秒以上，對於某些需要時效性的應用軟體來說，這麼長的時間可能就會影響到該軟體的功能，如遊戲就是一個最好的例子，還有 VoIP 的應用也不能有太長時間的延遲。

目前有些研究對於 Mobile IP 的缺失效能等有進一步的改善 [18][19][20][21]...，但因爲本質上的設計，有些問題還是存在並不會因爲改善了某些缺點而消失，只能比單純只有 Mobile IP 概念還來的有效率而已。

3.5 SCTP

3.5.1 SCTP 簡介

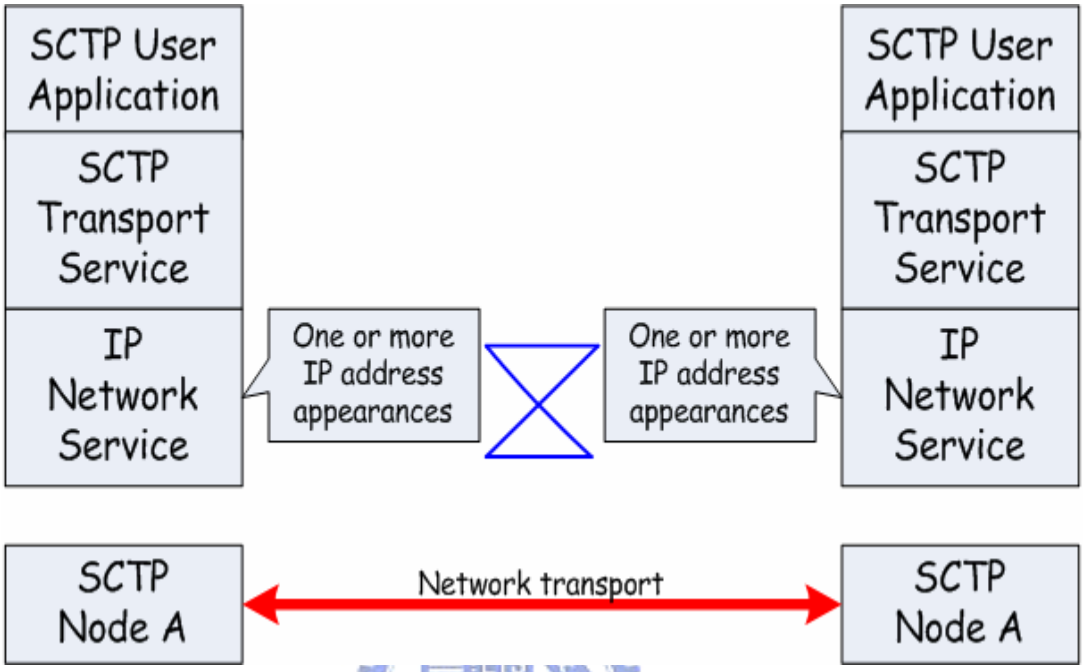

圖5: SCTP 連結概念

SCTP [9][25][26][27]是由 IEFT 的 SIGTRAN 提出的一種多媒體通訊控制傳輸協議，用於在 IP 網路上傳輸 PSTN 信令訊息，即通常所說的 SS7 over IP。目前，IEFT 將 SCTP 傳輸層協定作爲主要研究目的，與 TCP 和 UDP 共築於 IP 層之上。如圖 5 所示，SCTP 是一個傳輸層 (Transport Layer)的通訊協定，其最大的特點就是一個傳輸端可以有一個到多個 IP 同時運用。

- **SCTP 的特點**

SCTP 是新定義的一個通訊協定，它整合了 TCP 以及 UDP 的特性，包含 reliable、flow controlled、congestion controlled、data exchange 這些和 TCP 一樣，unordered、unreliable data exchange、這些和 UDP 一樣，不過它還有其他 TCP 以及 UDP 所沒有的功能。Multi-homing、Multi-streaming、Message boundaries (with reliability*)、Improved SYN-flood protection、A range of reliability and order (full to partial to none)…等。

*

UDP: message boundaries, not reliable

TCP: reliable, no message boundaries

圖6: SCTP Multi-homing

其中 Multi-homing 是一個對於換手來說很重要的特性，如圖 6 所示，當終端 B 的主要 IP 不能使用後，終端 A 依然可以把封包送給 B 的另一個 IP，因此 AB 之間所建立的 session 連線並不會因此而中斷掉。

- SCTP 的封包格式

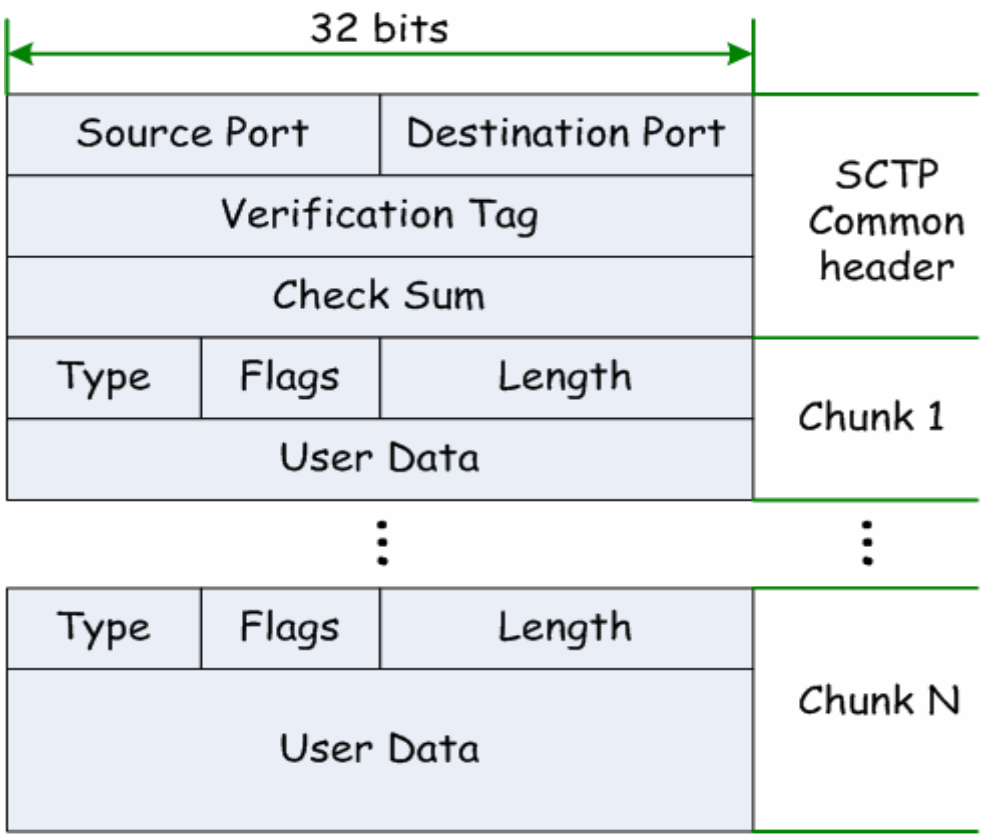

圖7: SCTP 封包格式

如圖 7，SCTP 的每一個封包包含 SCTP common header，以及後面接連著許許多多的 chuck，每個 chuck 包含 chuck header 以及資料。常見的 chuck 類型有：Data、SACK、HB、HB ACK。SCTP 可以把每個 chunk 全都包含在同一個封包裡，以減少封包傳送的次數。

- **SCTP 的重傳機制**

SCTP 的重傳機制是選擇性確認 (Selective ACK)。以例子說明，假設有封包編號從 1 開始到 10，而封包傳送從 1 開始一直傳送到編號 8，其中的封包 5 遺失了，對於 TCP 來說，它會認爲封包 6、7、8 均未收到，並要求從編號 5 開始重傳。而對於 SCTP 來說，因爲只有遺失封包 5，所以 SCTP 只會要求重傳封包 5 而已，至於 6、7、8 已經收到了皆暫存在緩衝區裡，所以說 SCTP 是選擇性的重傳確認。

- **SCTP 的斷線偵測**

SCTP 對於路徑 (path)來說，定義爲 path = destination，也就是說遠端如果有一個 IP 就會有一個 path 存在，SCTP session 在建立的時候，會告知彼此的 IP 資訊，因此對於每個 path 皆會定期的發送 heartbeat chunk，約 30 秒發送一次，用來偵測此路徑是否有效，經過五次的失敗，也就是沒有 heartbeat ACK 回應，就會認定此路徑失效。

3.5.2 SCTP 分析討論

SCTP 的 Multi-homing 對於可以同時使用多個網路介面是個很大的優點，而且他在傳輸層 (Transport Layer)上面實作，效率會比在應用層 (Application Layer)上實作來的好，和 Mobile IP 的作法比較，它並不需要在網路上部署 Home agent 及 Foreign agent，不過它在某些方面是有缺點的。以下將對一些缺點作說明。

- **網路介面的優先權太過簡單**

SCTP 的優先順序太過簡單，它會以先建立的連線爲主要連線，如果有問題需要切換才會換到備用路徑，因此並沒有考慮其他因素，以 WLAN 與 GPRS 爲例，一開始 SCTP 可能會先以 WLAN 當主要的傳輸路徑，當 WLAN 斷線後，就會轉換到 GPRS，如果此時 WLAN 恢復，SCTP 並不會換回 WLAN，而是繼續使用 GPRS 當主要路徑。WLAN 基本上頻寬、費用都比較具有優勢應該是首選，沒有換回來滿可惜的。

- **對於連線的狀態反應遲鈍**

SCTP 對於連線狀態探知的方法是計算重傳次數，當重傳達到某個次數且皆遺失或還是需要重傳的話，便當作此路徑有問題不能使用。以 heartbeat 來說，30 秒一次，也要等 5 次失敗了才會認爲有問題，時間相當長。

- **既有的應用程式需要重新設計以及編譯**

因爲現有的網路應用程式，基本上皆是使用 TCP 或 UDP 通訊協定，然而 SCTP 是新定義的通訊協定，如果既有的程式需要支援此通訊協定，勢必需要重新改寫編譯，這很花成本也不符合效益，這點是新通訊協定常有共通的毛病，尤其是既有傳輸層的通訊協定 (TCP/UDP…)使用的量太大了，需要改的程式就多的數不完。

- **NAT 及防火牆問題**

以現有的 NAT 機器，或是防火牆等網路中的元件，對於新定義的通訊協定，大致上來說均不支援，也就是說看不懂格式的網路封包均會丟棄不予理會。以 NAT 來說，當位於 NAT 內部的 LAN 要對外連線時，發送 SCTP 封包出去，這時大部分的 NAT 不了解該封包爲哪一種，所以均丟棄，不會作 IP 以及 port 的轉換對應 (mapping)。NAT 要區別 TCP 以及 UDP 的封包是需要的，因爲 TCP 在要求連線的時候，封包會有一個 SYN 的 flag，可以讓 NAT 知道要建連線了，便會對外打開一個 port 讓封包進來，要結束的時候也會有一個 flag 通知，NAT 就可以關閉該 IP 以及 port 的對應，讓封包進不來。然而 UDP，並沒有連線導向的機制，也就是說對於連線出去的 UDP 必須要知道是 UDP 型態才能設定 IP 以及 port 對應的 timeout 時間，如果不設 timeout 時間的話，不會知道什麼時候結束連線，對 NAT 來說，UDP 的 IP 以及 port 需要繼續維持對應關係，這樣會佔用 NAT 的 session 數量以及對外的 port 數，導致效能不彰，另一個原因是區分通訊協定，可以統計分析網路流量，進而執行過濾、監控等網路管理的動作，如限制某些網站的內容等，所以區別通訊協定對於 NAT 裝置是必要的。

第四章 遍佈式虛擬區域網路(U-VLAN)的設計

4.1 系統環境架構

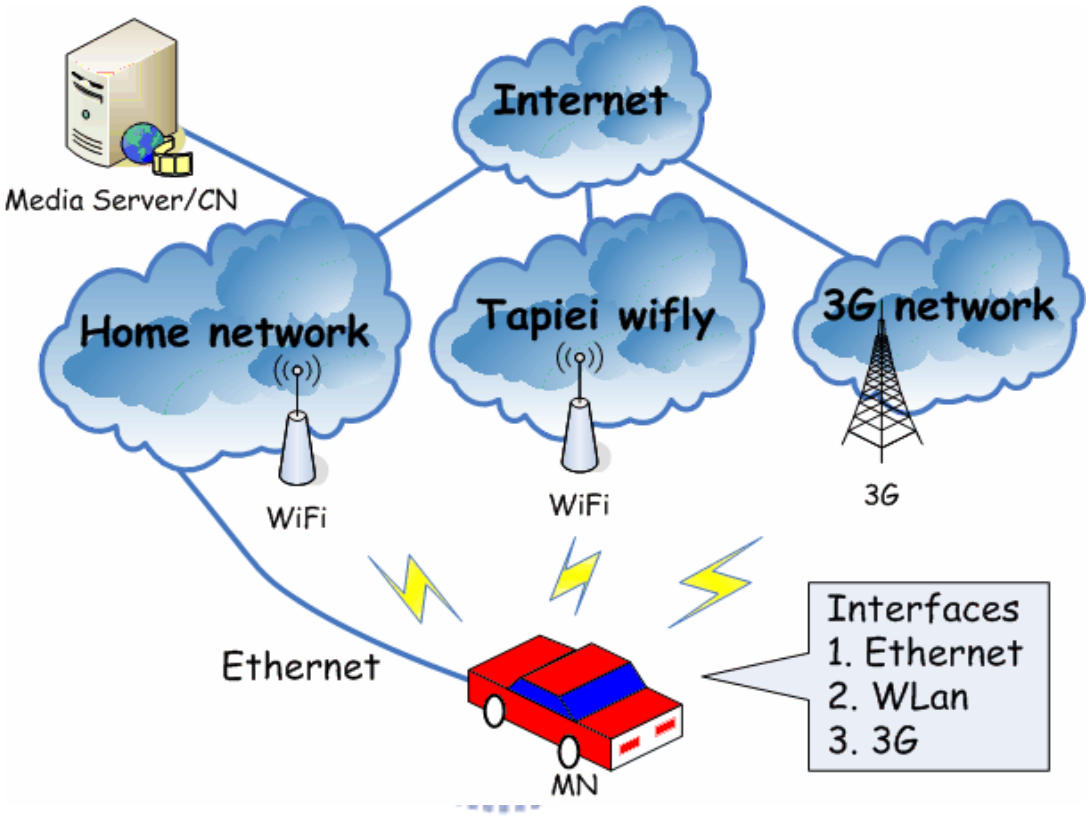

圖8: 系統環境架構

如圖 8 所示，移動端 MN 有三個網路介面可以存取網路，MN 爲一台巡邏車，當然巡邏車會把目前所看到及時的畫面傳回給 CN，而 CN 爲監控中心，專門收集每一台巡邏車的畫面，即時了解每台巡邏車上面的情況。

首先 MN 在家網路 (IP 爲 A)透過乙太網路和 CN 連接在同一個交換器 (switch)，接著 MN 以及 CN 啓動監控系統 (MN 傳送攝影機畫面給 CN)。系統啓動之後 MN 要開始巡邏車的工作，便開車離去，當然 MN 和家網路的連線就切斷。不過 MN 透過另外的網路介面重新連線到 CN，當然這時原本使用的 IP-A 就消失了，而換成 IP-B，而這是監控系統自行處理換手的問題，自行重新連線。然而，在本系統運作下，MN 上的客戶端和 CN 上面的伺服端皆不需要考慮換手的問題，因爲兩端的程式都會認爲彼此建立的連線(TCP/IP)沒有斷掉，而繼續維持連線狀態，因此只需要專心處理監控程式的事。4.3 節將介紹本系統是如何運作的。

4.2 U-VLan 特性

Ubiquitous Virtual LAN System 是用來達到特殊網路漫遊機制的軟體平台。只要電腦上面有安裝此系統，便可以建立一個虛擬區域網路。虛擬網路卡，會讓上層的元件如：TCP/IP 等認爲它是一個實體可以傳送的介面，並在路由表 (routing table)上增加幾個路由項目，讓應用程式可以正常的對該元件作存取。當然此虛擬介面所收到的封包會再經過一些機制轉送到其他實體的網路介面然後輸出。因此系統可以整合電腦上面多種異質網路介面的裝置，當電腦在漫遊時，可切換使用各種異質媒介網路卡，達到不斷線的漫遊服務。取名 Ubiquitous Virtual Lan 表示該系統可以遍佈存在於各個網路終端上，如行動端 (MN)以及對應端 (CN)。其 Ubiquitous Virtual LAN System 的主要特性與預期達到的功能如下所列：

- **支援各種型態之網路卡**

對於各種不同類型的網路介面卡，都可以使用。只要對於系統來說是一張網路型態的介面卡，在上面綁有 IP，可以被 TCP/IP 來運用的都可。常見的如：802.3 乙太網路卡，802.11 無線網路，GPRS，PHS，甚至是 WiMAX 等。不過前提是該硬體設備需要安裝好正確的驅動程式，且被 Windows 系統判別爲一個網路裝置，並有 IP 在上面才行。本系統可以支援一個(含)以上的網路介面，多個網路介面可以避免換手 (handoff)延遲的時間。

- **允許任一端(MN/CN)處於 NAT 的網路架構下**

可以允許 MN 或是 CN 某一端處在 NAT 的網路架構下，只要兩端有一端可以主動並成功連線到另一端即可，本系統的連線方式是以標準的 socket 介面做連線的動作，可以是 TCP 或是 UDP。

- **自動偵測目前系統可供使用之網路介面**

本係統會週期性的偵測各個正在使用中的網路介面，並及時通知負責連線的軟體元件。也就是說如果網路介面有多個，系統會自動選擇一個當作主要的網路介面來做連線，其他介面的會當作備案，一旦當主要的介面斷線了，系統會嘗試使用其他可以用的網路介面來傳送資料。

- **可使用多重介面來連線**

以 session 的角度來看，多重 IP 封包的傳送，皆視爲一個 session 的封包，這表示當 IP-A 不能用時，使用 IP-B 一樣可以繼續傳送原本的網路封包，就像 SCTP 裡面 multi-homing 的概念一樣，一個終端可能會有一個以上的 IP 可以使用，在建立連線後，

兩端必須知道彼此其他可用的 IP，知道之後當然就可以互相交替使用，這樣便不需要以 IP 來綁住兩端所建立起來的 session 連線，而漫遊的時候也不會因爲換 IP 而有斷線的問題。

- **無須修改原有系統核心與網路元件**

對於微軟 Windows 上面的網路核心，不管是公開的或是沒有公開的軟體介面，只要對其架構有所變更，或是對該系統做修補 (patch)的動作都會使微軟的系統不穩定，因爲我們對於微軟做了些什麼，或是在 Windows 裡面一些隱藏的軟體機制很不明確。而本系統是依照微軟提供的 DDK 所開發的標準虛擬網路卡驅動程式，對於原本核心的元件完全不去更動，保持 Windows 原有的機制也保持了穩定性。

- **不改變現有各種網路架構與終端裝置**

在本系統的整合架構中，無須修改現有網際網路環境的元件包括路由器，NAT，及電信網路內的各項元件等所有網路硬體設備。

4.3 運作機制

4.3.1 虛擬區域網路

本系統是建構在虛擬區域網路 (VLan)的架構上，和 IEEE 802.1Q 不同。

- IEEE 802.1Q

能夠在乙太網路交換器、第三層交換器 (Layer 3 Switch)、路由器、或是其他支援 802.1Q 的設備中，區隔區域網路的一種方式。傳統每個連接的機器必須要在同一個區域網路 (network)下，才能進行資料交換，因此，當我們想要在一台 HUB 或是 SWITCH HUB 能夠擁有兩個或是兩個以上完全獨立的區域網路時，是沒有辦法達成的，一般的解決方案是使用兩個或是兩個以上的 HUB 或是 SWITCH HUB 來解決；因此，支援 802.1Q 的交換器便提供了以一台設備區隔多個區域網路的功能。基本上 802.1Q 最大的用處是在乙太網路框架 (Frame)的表頭上加上一個標記(Tag)，用來分辨來自不同 VLan 的封包。

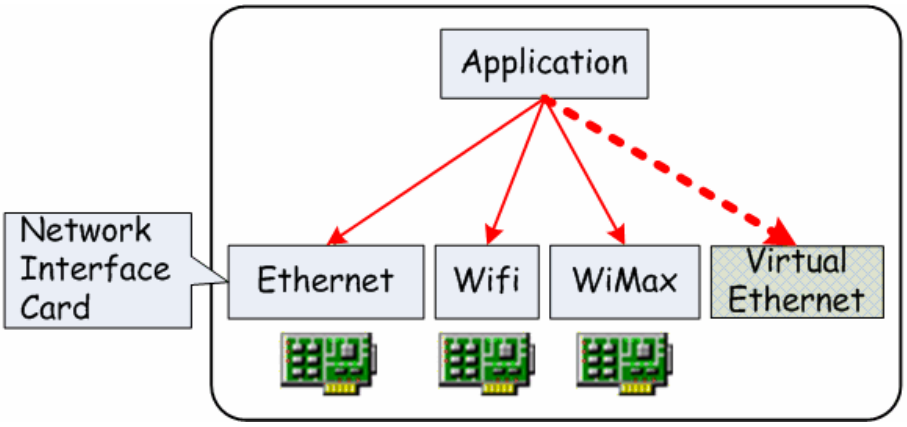

圖9: 虛擬網路卡

在業界當有人談到 802.1Q 就等於在說 VLan，談到 VLan 就等於在說 802.1Q，然而本系統的 VLan 並不是指 IEEE 所定義的 802.1Q，也不是分群標記，而是以系統的觀點下，產生一個子網域 (subnet)，並且使該網域不會有斷線的問題，可以永久使用。如圖 9 所示，在作業系統的觀點下，原本不存在的網路介面多出一個虛擬的網路卡，當然作業系統會在該虛擬介面上綁定一個 private IP 位址，因此多出一個網域可以存取。

- Virtual Lan

說到了區域網路 (Lan)，會出現的成員有三：交換器，網路卡以及網路線。三者皆需要存在，才能構成一個區域網路的實體。再來就是在該區域網路上面的 IP 位址，基本上都是屬於同一個網域，也就是說該網域所有封包上面的 IP 值和網路卡上設定的 MASK 值做 AND 運算之後，皆必須要相同，這樣每個終端設備才會收到該網域的封包。

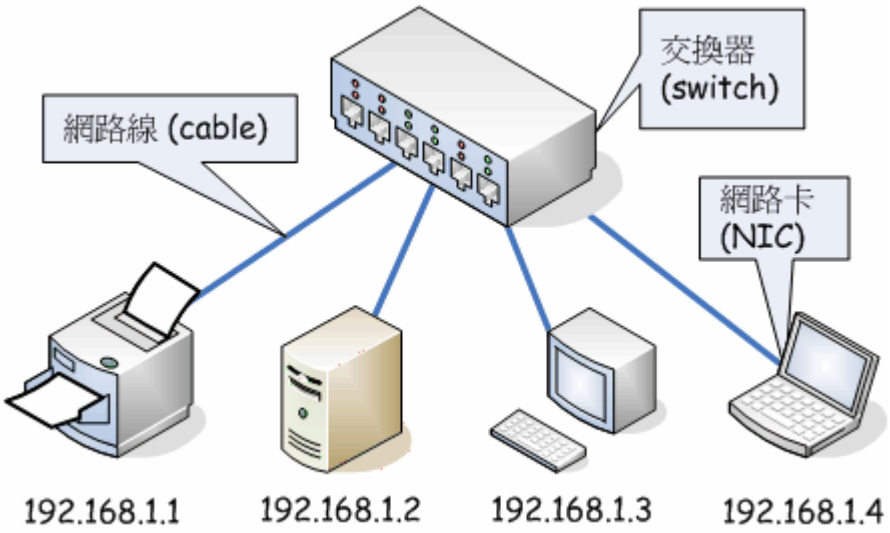

圖10: 區域網路

圖 10 是一個典型的區域網路，上述的三個元件都各自扮演著自己的角色。交換器負責集中所有設備的聯接，並把封包交換到正確的網路孔。每個終端設備皆有網路卡，做爲網路封包進出的窗口。交換器和網路卡中間要靠著網路線做連接。

而相對於實際上的區域網路，本系統就是要利用區域網路的概念，把這個概念轉化成爲虛擬的，也就是說上述的三個元件變成虛擬存在的。實際上，虛擬交換器，虛擬網路線及虛擬網路卡的機制皆是用軟體來達成的。以虛擬網路線爲例子，沒有實體的網路線，取而代之的是需要透過認證的 socket 連線，一旦 socket 認證通過，即表示該虛擬區域網路有一條虛擬網路線接到虛擬交換器上面，或是從虛擬交換器拉一條線接到終端設備的虛擬網路卡上。當然虛擬的網域是透過隧道 (Tunnel)的技術並附在 socket 連線上運作的，這點下一節會介紹此技術的運作方式。

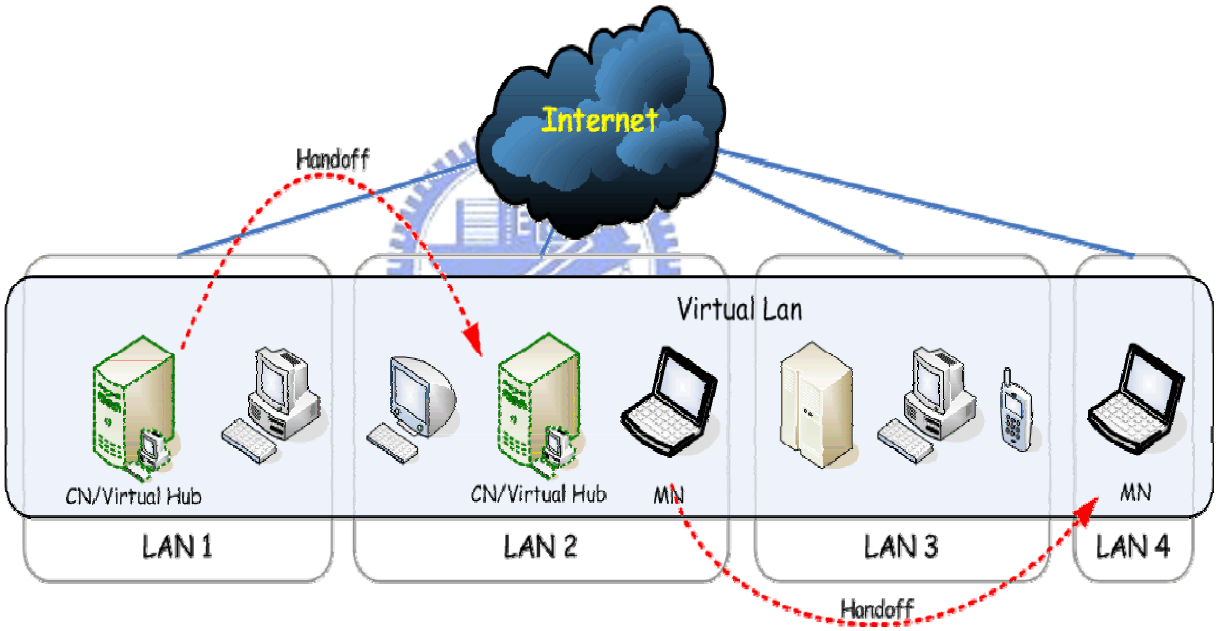

圖11: 虛擬區域網路

圖 11 所示爲虛擬區域網路的建立概念圖，許許多多的 LAN 可以透過網路閘道連上網際網路，然而不同的 LAN 彼此不能直接溝通，需要透過路由器才行。而本系統便是建立一個虛擬的區域網路，整合乙太網路的特性，讓實際上不同的區域網路也能變成同一個區域網路，這樣就能享有區域網路的便利性，再者，本系統基於系統的特性，可以讓終端到處漫遊。如 MN 從 LAN2 漫遊到 LAN4，而本系統建立的虛擬區域網路 (Virtual Lan)依然會還存在，所以只要是透過此虛擬區域網路所建立的連線均會維持住，也就是不會有斷線的問題。另外還有一點，本系統可以讓 MN 或是 CN 均可漫遊，只是不能同

時換手，需要等某一方換手穩定之後才能換另一方換手，如圖中的虛線部分。在 MN 與 CN 兩邊系統上的設計都一樣，只差了剛啓動時，是否會主動建立連線而已。基本上兩邊可以漫遊的概念有點像是 SCTP 的多 IP 終端，也就是說當 MN 與 CN 建立好連線之後，彼此的 IP 資訊皆會交換了解，並嘗試做排列組合的連線測試，偵測哪些 IP 組合可連線那些不行，用以當作往後如果需要換手時的備用連線，或是需要做負載平衡時多條連線用。

4.3.2 隧道機制

隧道機制 (Tunnel)是目前常出現的一個技術，一般都用來整合不同的通訊協定，常見的方式便是把通訊協定 A 放到通訊協定 B 的封包裡當成通訊協定 B 的資料。其中放入的動作稱爲編碼 (encode)，相對的拿出來的動作稱爲解碼 (decode)。如：IPv6 over IPv4。本系統使用的通道技術爲 Ethernet over TCP。

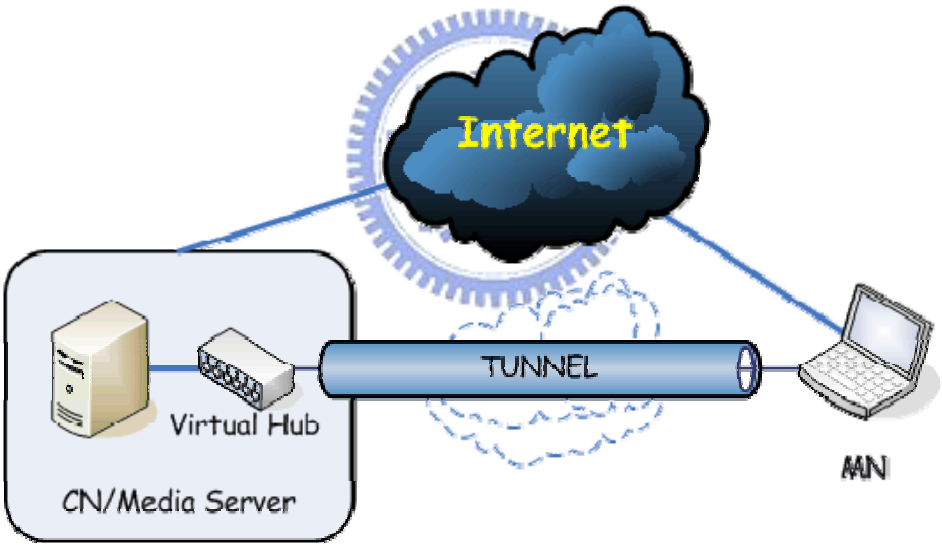

圖12: 隧道意示圖

如圖 12，MN 透過實體的網路介面和 CN 建立連線，也同時建立了一個隧道，該隧道專門用來傳送虛擬網域的資料，而 MN 和 CN 以虛擬的架構來說，就像用一條虛擬的網路線連接在同一台交換器上。和一般的區域網路一樣，可以透過乙太網路的機制傳送封包，而實際的隧道技術請看圖 13、圖 14。

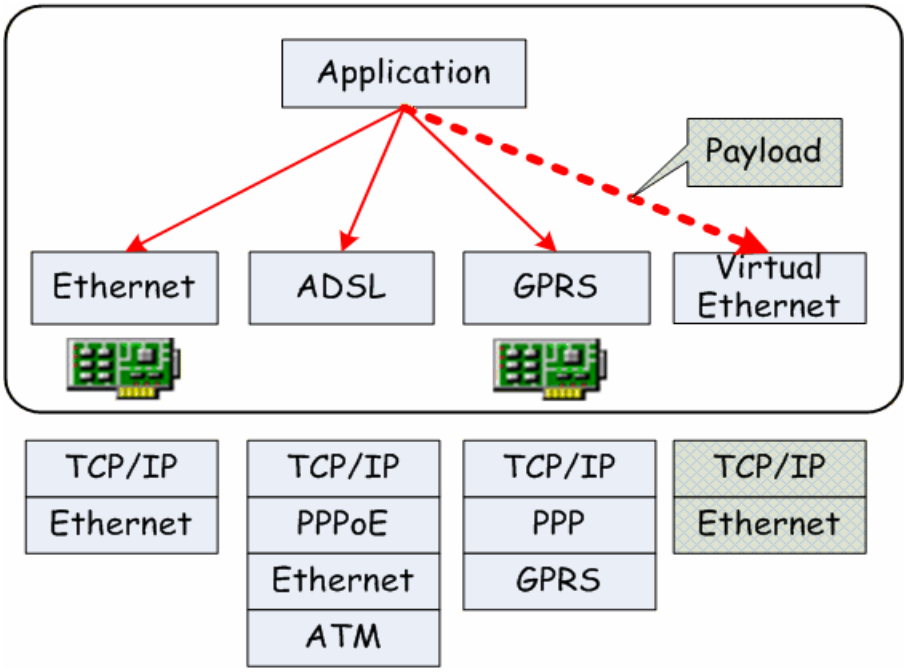

圖13: 封包堆疊與網路介面

當資料要從虛擬網路介面送出時，會經過本系統，並加上定義好的表頭 (U-VLan header)，之後再轉送給 socket 串流。如圖 13，在下面層左邊三個 TCP/IP 是實體網路介面所綁定的，在這些 TCP/IP 之下便是各種不同種類的網路介面所需要的通訊協定，如：ADSL 需要使用的 PPPoE，或是乙太網路的表頭等。而在下面層最右邊的 TCP/IP 爲虛擬網路卡所綁定的，所以只要封包經過本系統的虛擬網路介面，也就是虛擬網路卡上面設定的網域，就會送到本系統，之後就如之前所述的一樣。

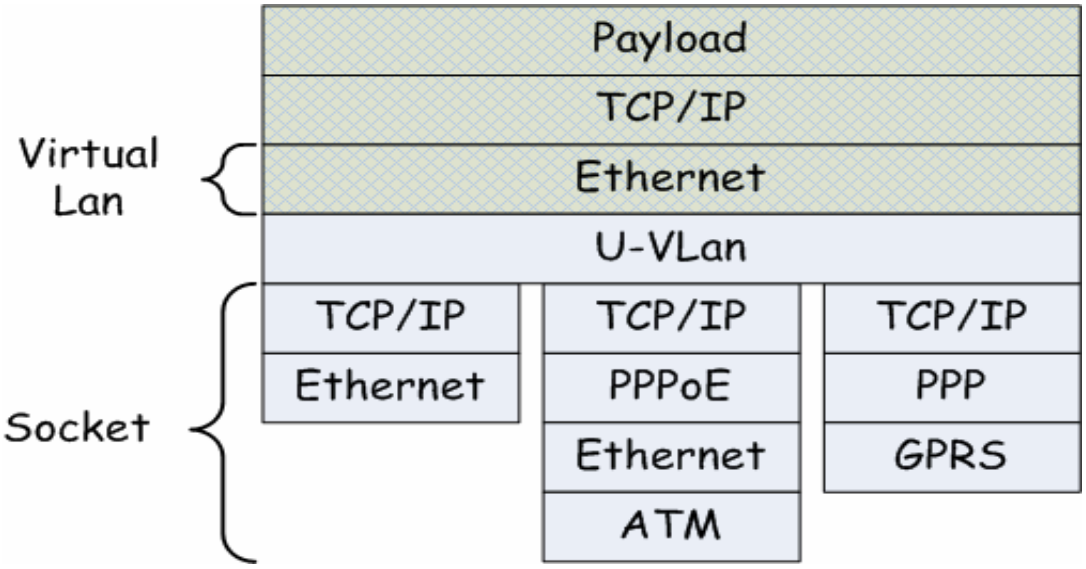

圖14: 隧道封包堆疊

圖 13 中的一些表頭 (header)以及資料，經過系統實際的運作之後，就會變成如圖 14 所示的關係，在上面三個爲整個虛擬網域的封包，在下面就是實體網域相關的表頭(視實際從哪種通訊介面送出去，下層就會附加該通訊介面的表頭)，至於中間，則加上了系統定義的一些通訊機制表頭。包含傳送封包資料的訊息、設定爲主要介面的訊息、本端所有相關 IP 的訊息、認證訊息等等！這些訊息可以幫助系統了解換手該處理的事情，以及如何維護該虛擬網域的連線等等。

本系統定義的 U-VLan 表頭，爲純文字格式 (text-based)，可以包含在 HTTP 的通訊協定裡，和 HTTP 相容。爲什麼要設計成純文字格式呢？簡單的說是因爲純文字的格式簡單，好維護，也好除錯，當然對於設計者來說，一看就懂。還有一個原因，便是防火牆的因素，由於現今有許許多多的防火牆都會限制或是管制連線，當然在最壞的情況下，有的甚至只限制 HTTP 類型的網路封包才能夠進出，因此如果所有的封包都建構在 HTTP 通訊協定的架構上，便可以很容易的穿過防火牆，有效的避開防火牆的限制，說到這，必須介紹一下日本的一個軟體 Softether [29]。**[Softether]**這是日本築波大學一年級學生登遊大自行開發的軟體，此軟體簡而言之，就是模擬乙太網卡的工作，甚至可以模擬 HUB 功能，使用隧道 (Tunnel)的特性，實現 VPN 的功能。使得系統把此軟體完全無礙的識別成一塊網路卡。有了這個東西，可以不再買 VPN 路由器，而實現從 Internet 訪問自家 LAN 的目標。主要想法是：A. 在 OSI Level 2 (Link Layer)上軟體模擬網路通訊，把乙太網路的通訊內容封裝到 TCP 封包裡去(軟體模擬乙太網路)。B. 再把自己的通訊包變成 SSL Session，用 HTTPS 協議穿越網際網路，甚至混過防火牆。基本上，這是個 clinet/server 的軟體，他的 clinet 端是一個虛擬的網路卡，只要設好了 server (virtual hub)，任何一個 clinet 連上去後，彼此間就成區域網路了，所以如果公司或是學校的區網，被防火牆給擋死了，便可以利用此軟體，透過第三者連上網路。

本論文所要提出的系統平台也是建立在上述 Softether 的技術上，並且修改部分架構，加入 mSCTP 的某些特性，並自行定義了換手的機制，以達到可以無縫漫遊的目的。其中 mSCTP 的特性，如某終端有多個 IP 可以使用，便會善加的利用多 IP 的環境。在 SCTP 的標準中，可以同時有多個 IP，不同 IP 傳送的資料可以視爲同一個 session 的連線，這個好處是可以解決 IP 固定而會斷線的問題。還有就是 SCTP 彼此連線的兩端，都是相對的，不分 Server 或是 Client，也就是不管是 Server 或是 Client 皆可以換手漫遊，只是兩端不可同時換到新的環境(會有新 IP，而舊 IP 不能使用)，兩端如果同時換的話就會有問題。

4.3.3 資料傳送流程

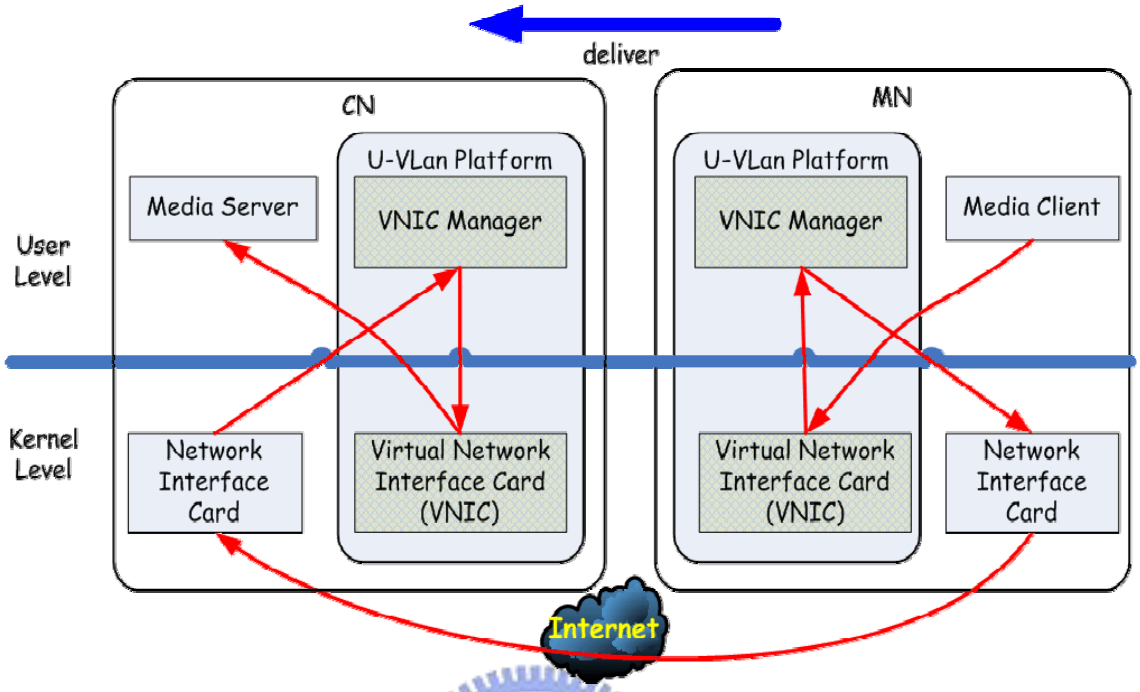

圖 15: U-VLan 平臺

透過前面所提到的隧道機制，以及虛擬區域網路的概念，組成了本系統基本的運作模式。以圖 15 爲例，MN 傳送資料給 CN，箭頭是指資料傳送方向。

在 MN 以及 CN 上都有本系統 U-VLan 之平台，對於應用程式如 Media Client 或是 Media Server 來說，會有一個虛擬不會斷線的網域可以連線，Media Client 可以直接對 CN 上的虛擬 IP 做 socket 的連線，因爲同網域，依據路由表 (routing table)的路由資訊，會從虛擬的網路卡送出封包，而本系統便取得送出的封包，進而送往虛擬網路卡的管理者，它會把封包再經過某種程度的包裝之後，透過 socket 的連線送出。因爲這個連線是經過真實的網域，所以會從實體網路卡丟出來。當 CN 的實體網路卡收到資料之後，因爲是虛擬網路卡管理員開啓的 socket port，所以就會轉送給虛擬網路卡管理員，它收到之後，會分析封包，還原成原本的乙太網路封包，然後轉送給虛擬網路卡，虛擬網卡便會通知 Media Server 所開啓的 socket 有封包進來了，這樣 CN 便成功的收到 MN 送過來資訊。上面一串連的動作稱爲隧道機制 (Tunnel)。

而圖中，虛擬網路卡管理員爲何要設計在使用者空間 (user space)，原因有幾點。第一點是如果放在核心 (kernel space)的話，程式設計會不好寫，而且不容易維護，甚至難除錯，再來就是對於微軟的 TCP 元件，如果寫在核心裡，不能使用該元件(因爲沒有微軟的原始碼或是技術支援)，如果是這樣，便要重新自己開發 TCP 的機制，那更麻煩，

而且寫在核心裡，如果有問題，會使得整個作業系統當機，這是很嚴重的問題。也因爲這樣，放在使用者空間好處多多。微軟近來發展的趨勢也是把驅動程式放在使用者空間(Windows User Mode Driver Framework)，用意就在說明：能不放在核心裡盡量不要。而虛擬網路卡管理員放在使用者空間，便可以透過標準的 socket 介面，使用 TCP 的機制，因爲是使用 Berkeley socket 的標準原型，所以以原始碼的角度來說，跨平台比較容易，而且在使用者空間，可以瞭解到許多系統的資訊，包含 TCP/IP 的參數，路由的資訊，ARP 的資料，實體網路卡的運作狀態等，皆容易取得。也許在核心也可以瞭解，不過對於同一層的驅動程式，要瞭解彼此的訊息是有點不容易的，甚至要取得使用者空間的資料更麻煩，所以總和以上的討論，把虛擬網路卡管理員設計在使用者空間下是可以接受的。

4.3.4 漫遊機制

本系統主要是要達到無縫隙的漫遊，當應用程式在連線時，發生換手的時候，透過本系統的機制，可以讓該應用程式認爲它使用的虛擬網路介面沒有斷線，沒有換手的動作，進而覺得它的連線還是繼續維持。以目前網路上，百分之 60 以上的封包來說幾乎都是 TCP/UDP 一旦換手了，對外連線所使用的 IP 一定會變更，一定有所不一樣，當然 session 就斷掉，需要重新連線。以例子說明本系統的運作概念。

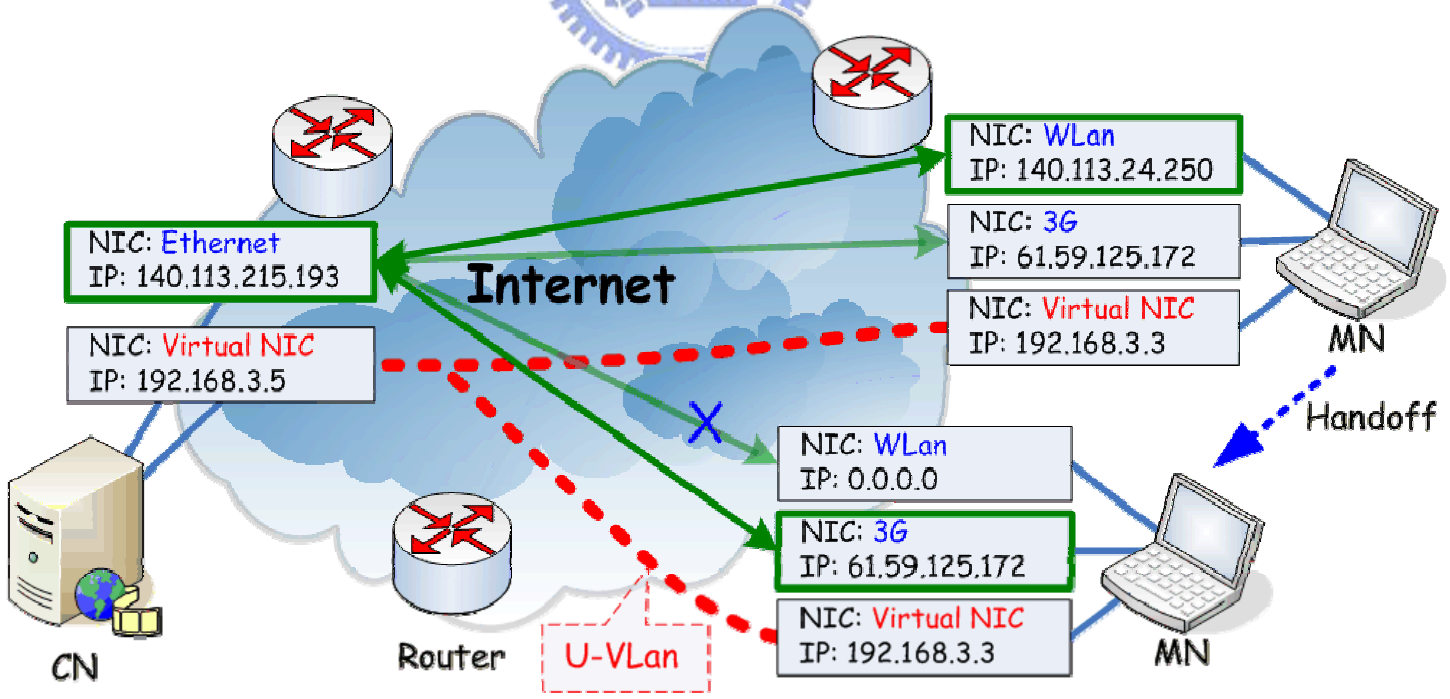

圖16: 漫遊意示圖

如圖 16，在這個例子中，CN 只有一個實體的網路介面，而 MN 有兩個實體的網路介面，一個是 WLan，另一個是 3G。在 MN 以及 CN 上皆有安裝本系統，因此產生了一個虛擬的網路介面，因此對 MN 上的應用程式來說有三個網域可以使用。接著 MN 上面的虛擬 IP 設定爲 192.168.3.3 而 CN 上面的虛擬 IP 設定爲 192.168.3.5。很顯然的，該虛擬的網路介面必須設定爲同一個網域，這樣才能產生一個虛擬區域網路，MN 上層的應用程式便可以透過該虛擬的介面和 CN 作溝通。當啓動本系統之後，兩端的虛擬區域網路便建立起來，先前提到的虛擬網路線就是圖 16 的虛線，它會一直存在，而實體的連線是實線，也就是虛線是以實線來達成的。另外在還沒有換手之前，MN 以 WLan 爲主要的網路介面，3G 爲備用介面，換手之後就變成以 3G 爲主要介面。下面將介紹 MN 換手的流程。

- Step 1

當本系統啓動之後，MN 以及 CN 會知道彼此所有的 IP 資訊，並嘗試對每組 IP 的組合作配對連線。如圖 17：MN 有 IP-A 以及 IP-B，而 CN 有 IP-X 及 IP-Y，總共有四個 IP (2x2)，而本系統會嘗試建立四條 socket 連線，其中一條連線爲主要，其餘的爲備用。這個概念就和 SCTP 中的 multi-homing 類似，有多重 IP 可以使用的終端設備，彼此連線可以透過不同的 IP，可以設定主要的 IP 及次要的 IP 路徑，而上層本身的 session 還是維持連線狀態，不受影響。以圖 16 爲例，MN 上主要的 IP 爲 140.113.24.250，所以主要的網路卡介面就是 WLan。

圖17: 多 IP 連線圖示

- Step 2

啓動 MN 及 CN 的監控程式後，MN 上的監控程式客戶端取得攝影機的畫面，利用 socket 連線，透過系統建立好的虛擬網域連線到 CN 的監控程式伺服端。這時 MN 以及 CN 的通訊便開始了。

- Step 3

當 WLan 出現問題，斷線時，接手的是 3G 的實體介面。當然實際對外的 IP 也就跟著換掉了，從 140.113.24.250 換到 61.59.125.172，而本系統在斷線前所建立的備用 socket 連線會繼續接手，如果備用連線尚未被建立起來，系統會依照之前所得知彼此 IP 的資訊，嘗試建立連線。

- Step 4

當備用連線或是嘗試建立的連線建立完成後，便恢復原本正常的狀態，也就是監控程式客戶端照常傳送畫面給伺服端，當然，在換手時無法傳送的封包會被暫存起來，直到這個時候再繼續傳送。對於虛擬介面來說，應用程式的感覺可能只有該介面的網路忽然延遲了一點而已，並沒有斷線的問題。

第五章 遍佈式虛擬區域網路(U-VLAN)的實作

5.1 開發環境

本系統平台的實作環境是建立在微軟的 Windows XP SP2 (5.1.2600)上，下面列出開發所需的工具以及程式元件，而本系統的軟體基本元件會在下一節介紹。

- Microsoft Windows Driver Development Kit 3790.1830

這個工具是用來開發虛擬網路卡，也就是 Visual Network Interface Card (VNIC)，要開發網路卡的驅動程式必須要依照微軟所定義好 NDIS 的一些標準介面，並依照標準流程，對作業系統註冊回呼 (callback)函式。這部份花比較多的時間是在看 DDK 的相關文件，以及了解微軟作業系統裡驅動程式的運作機制。

- Microsoft Visual Studio .NET 2003

此工具是微軟所開發的 IDE 程式開發工具，它有友善的介面可以做程式的編輯，是微軟平台上常用來開發應用程式的好工具。本系統在使用者空間的應用程式皆用此工具開發完成。U-VLan Manager (VNIC System)，是此程式最重要的部份，也就是虛擬網路卡管理員，它負責許多需要處理的事情，包含網路 socket 的建立、和虛擬網路卡的溝通、通訊協定的處理、實體網路卡的監控、換手的機制以及使用者介面等。

- Microsoft Platform SDK for Windows Server 2003 R2

此工具提供 Windows XP 上，和作業系統溝通的所有 API 以及一些好用的函式庫，包含本平台需要和 TCP/IP 元件溝通的橋樑-IPHelper，這個工具是開發微軟應用程式不可或缺的好幫手。

- WinPcap 4.0

此工具可以用來直接對網路卡發送封包以及收封包，可以避開微軟作業系統的安全檢查，直接對網路卡驅動程式發送封包，而此工具我們用來開發實驗的時候所需要用到的封包發送器，此封包發送器也是使用 Microsoft Visual Studio .NET 2003 的工作環境來開發，並配合此 WinPcap [30]的函式一起運作。

5.2 軟體基本元件

本章節將針對在 MN 以及 CN 上面所運作的各個元件做詳細的功能介紹與說明。

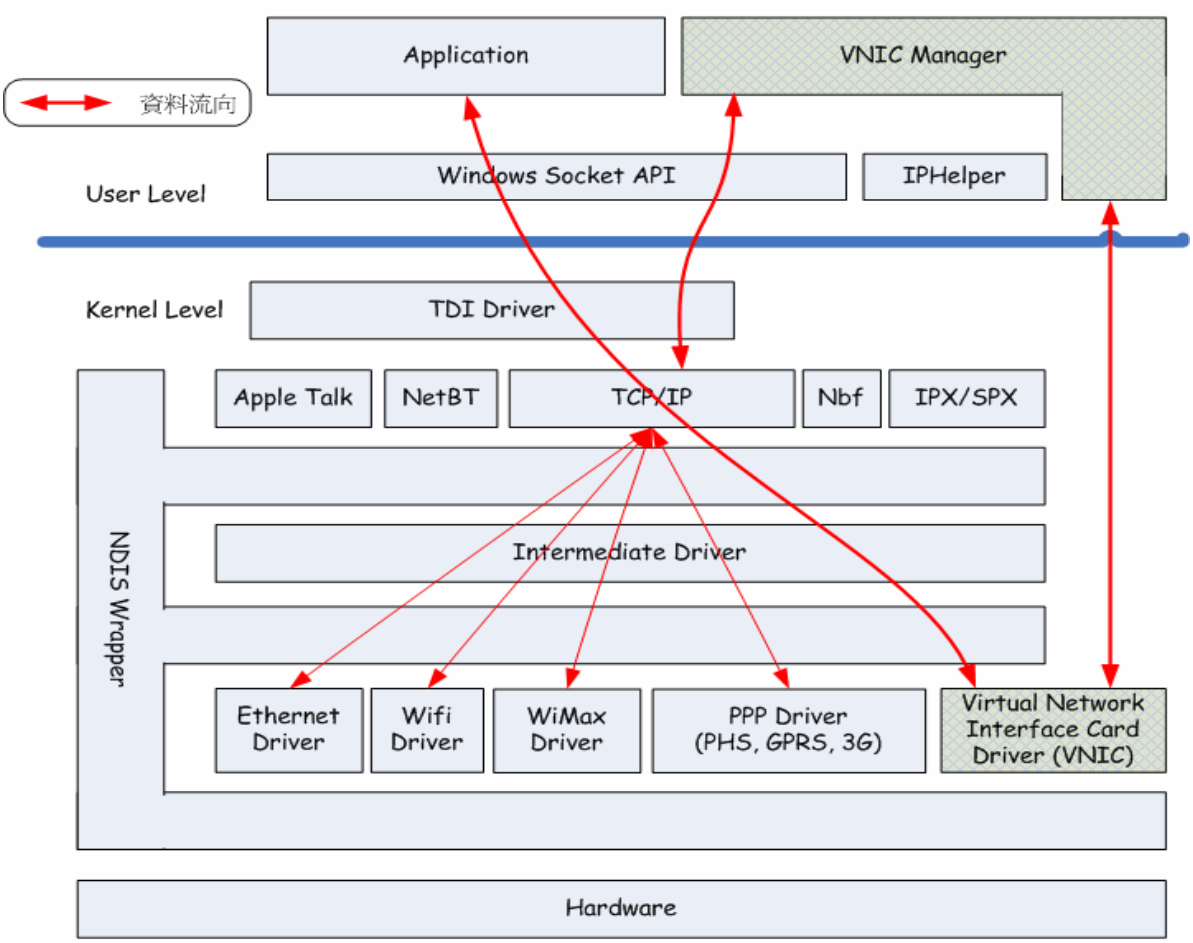

圖 18: 系統元件架構

如圖 18，VNIC Manager 以及 VNIC 兩塊有斜線部份的爲本系統所開發的元件，VNIC 爲虛擬的網路卡驅動程式，讓上層的所有元件認爲它是確實存在的一個網路介面，存在的一個子網路 (subnet)。而在使用者空間 (user level)，有一個 Manager 顧名思義就是管理那張虛擬網路卡，並透過它有所作爲。而其餘的部分都是微軟作業系統內建的網路元件，包含 TCP/IP 等通訊協定，當然也有各個硬體廠商所開發網路硬體的驅動程式如：802.3 乙太網路驅動程式，802.11 無線網路驅動程式，而 PPP 是指需要撥接用的網路介面，如：PHS, GRPS, 3G 等網路系統。Application 就是利用 socket 來建立連線的網路應用程式，這意味著 CN 以及 MN 上面要執行的監控程式。

5.2.1 虛擬網路卡

此虛擬網路卡驅動程式爲微軟 NDIS 架構中的 miniport driver，它和一般有實際硬體網路卡的驅動程式類似，是提供一個標準的介面(符合微軟所定義的介面)，好讓作業系統可以控制，存取該硬體。而不同的是，此虛擬網卡的驅動程式並沒有實際的接觸硬體設備，沒有向作業系統註冊硬體中斷的回乎機制 (callback)，也就是說該驅動程式並不會對硬體有直接控制存取。相反的，它只是讓上層作業系統的元件認爲它是一個實際的硬體，該有的硬體特性皆存在，就如同該名稱，顧名思義就是虛擬的，爲裝的，讓一般的元件都認爲它是有意義的存在。

此虛擬網路卡，定義了幾個簡單的ioctl介面讓管理者(VNIC Manager)方便存取。下面分別以網路應用程式的角度條例並說明之。

- **封包送出以及通知**

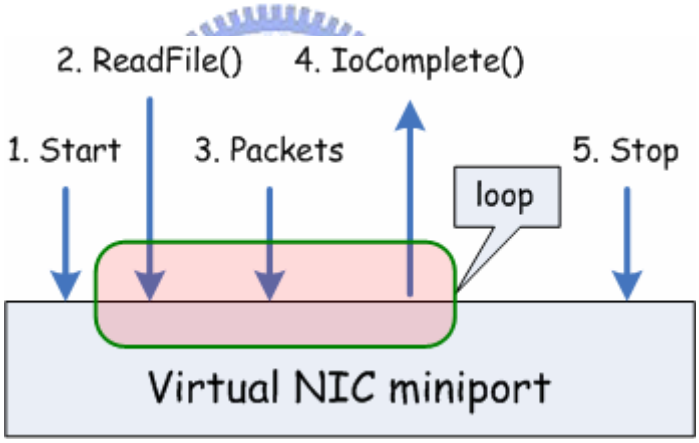

圖19: 封包送出以及通知流程

[Step 1]

當系統啓動之後，會對此虛擬網卡下一個控制命令，也就是 DeviceIoControl(IOCTL_WWTTVNIC_READ_DATA_START)，當虛擬網卡收到此命令後，會準備緩衝區，設定適當的大小，用來儲存所有上層應用程式送到此網路卡的封包，這就是圖 19 中的第 1 個步驟。

[Step 2]

緊接著，要取得所有送到此虛擬網路卡的封包，可以透過 ReadFile()的 IO 介面對此裝置下指令，當此裝置收到此呼叫後會判斷，如果在緩衝區有已經存在的網路封包，便直接回應，並把網路封包回傳給呼叫者，也就是 IoComplete()，如果沒有，便會進入

IRP_PENDING 的狀態，該狀態表示 IO 的動作尚未完成，需要等待，在此系統的狀況下，所指的就是在等待上層應用程式送封包到此虛擬網卡。

[Step 3]

此動作爲上層應用程式發送封包，經過路由表的判定，決定要送到此虛擬的網路卡，作業系統便會呼叫此虛擬網路卡所註冊的回乎函式 (CallBack)：SendPackets_Handler(NDIS_HANDLE MiniportAdapterContext, PPNDIS_PACKET PacketArray, UINT NumberOfPackets)。此函式就會把封包給儲存在緩衝區裡，並設定相關的記錄旗標，之後會去檢查是否有 IRP 在 Pending，有的話會進入步驟 4。

[Step 4]

有 IRP 的 Pending 以及緩衝區也有資料的時候，此動作就會發生，意思就是說，有讀取資料的動作，資料也存在，當然互相配合。此虛擬網卡便會把封包資料回傳給上層的應用程式，然後讓系統結束 IRP 的要求。

[Step 5]

最後的動作就是，當本系統要關閉的時候，會對此虛擬網卡下的最後一道指令，就是告訴虛擬網卡，可以不需要再儲存送過來的網路封包了，也不需要處理 IRP 的 Pending 了，意思就是直接把封包丟掉，當作沒事。

- **封包收到以及通知**

要通知上層的網路應用程式有封包到達了，這個動作比較簡單，因爲只要在虛擬網卡的驅動程式內呼叫 NdisMEthIndicateReceive() 或是 NdisMIndicateReceivePacket() 這兩個函式來通知上層的 protocol driver 有封包進來了，當然對外也要提供一個介面給虛擬網卡管理員來呼叫，虛擬網卡管理員可以透過 WriteFile() 對虛擬網卡寫入封包，當虛擬網卡收到指令之後，就會做上述的通知，告訴上層有封包到了。這樣的動作比較簡單，因爲很單純，不需要考慮同步的問題，只要虛擬網卡管理員收到資料，解開還原成原本乙太網路的資料，然後寫入虛擬網卡，這樣虛擬網卡就會有收到封包的事件出現了，之後 TCP 的通訊協定就會接手，再傳給位於使用者空間的 socket 元件，最後到網路應用程式。至於要暫存於緩衝區或是讀寫的同步問題，這個微軟的 NDIS 以及 TCP/IP 等元件會處裡，不需擔心，所以這部份比較容易完成，不需動用太多的資源。

5.2.2 虛擬網路卡管理員

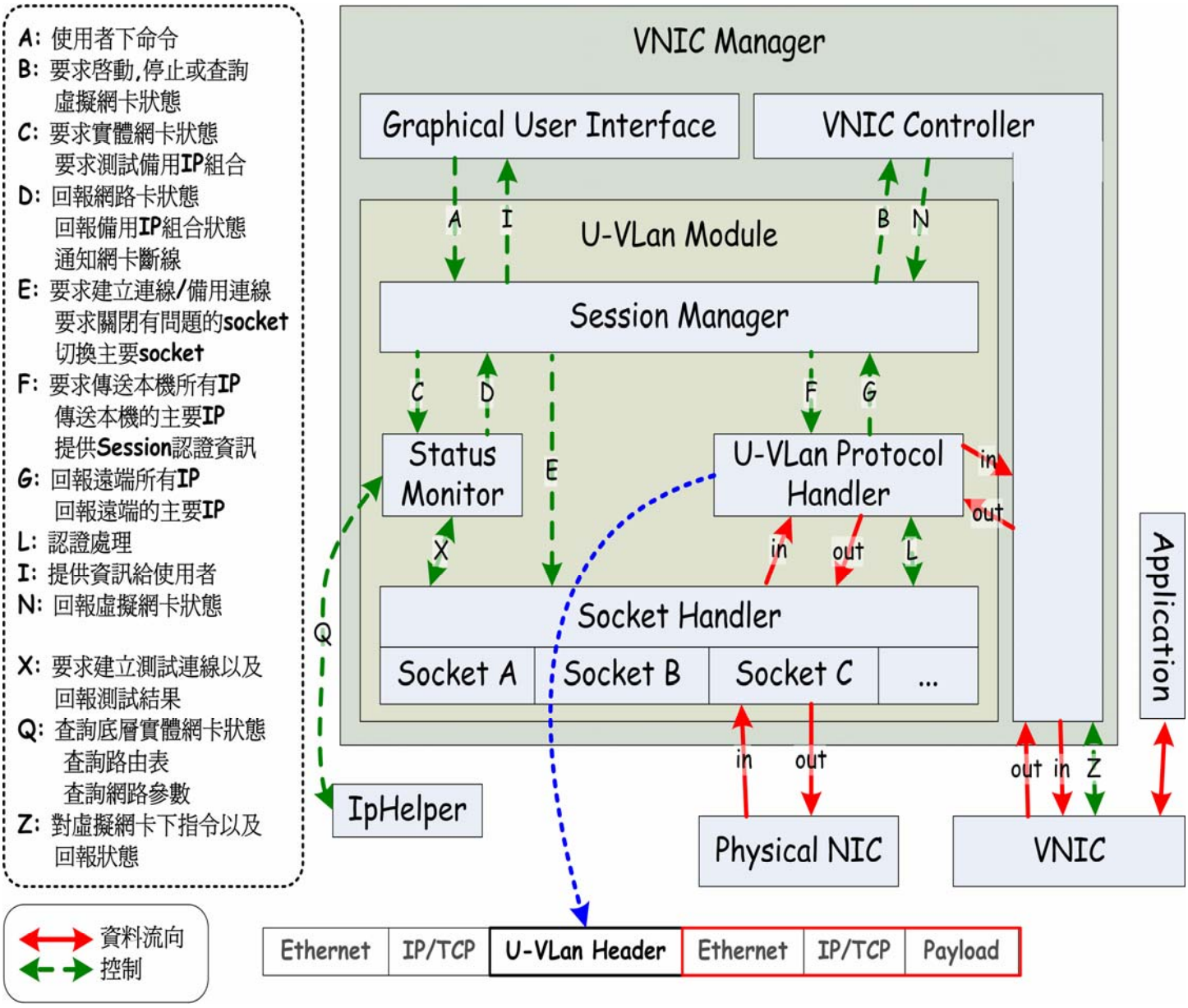

圖20: 虛擬網路卡管理員架構

如圖 20 所示，爲虛擬網路卡管理員得組成元件，包含有幾個重要的元件，GUI，U-VLan module、VNIC control，其中 U-VLan module 是本系統的核心，其包含 Session Manager，Status Monitor、U-VLan protocol handler、Socket handler，每個元件都有該負責的任務，下面將詳細介紹每個元件的功能以及流程。

- Graphical User Interface

使用者介面，讓使用者來控制此系統，有簡單的案鈕和設定程式，下達指令後，U-VLan module 便會依照使用者的設定運作，並且顯示目前的系統運作的狀態等資訊。

- VNIC Control

這個部份是用來和虛擬網路卡作溝通的元件，包含下達 IRP 中斷給 VNIC，以等待封包的通知，及傳送封包給虛擬網卡。因爲有輸出輸入 buffer 的問題，所以要有簡單的機制做同步化，防止資料順序亂掉，以及提高封包讀寫的效能。

- U-VLan module

這是本系統重要的核心元件，他負責維持虛擬網路的通暢，也就是說此元件必須要維持虛擬網路卡正常封包的傳送以及接收的動作。因爲虛擬網路卡負責蒐集應用程式所發出的網路封包，並透過此模組，封裝後再透過 socket 傳送到網路的另一方，所以要維持這樣的流程，是本模組該做的工作。以下分別介紹每個小元件的功能。

[Status Monitor]

此元件是專門監控網路硬體以及通報的程式，它會週期性監控所有網路介面的狀態，並及時通知 Session Manager 元件等，此元件也是和微軟作業系統的溝通橋樑，也就是和微軟內的 IpHelper 交流，透過微軟提供的 API 了解所以網路介面的運作情況。

[U-VLan protocol handler]

此元件用來做資料封裝以及解封裝用的，因爲 MN 以及 CN 會透過實體的網路作溝通，而這中間的通訊會經過隧道 (Tunnel)的技術傳輸。此元件定義了一些簡單的標頭，而這些標頭以文字字串爲主，和 HTTP 的通訊協定相容，可以利用 HTTP 的通續協定傳送，甚至是可以進階到 HTTPS 的加密。

[socket handler]

管理底層所有使用到的 socket，被其他元件所控制，此元件提供單純的 connect()、send()、recv()、…等介面，讓 U-VLan protocol handler 直接傳送封包資料以及接收封包資料，而不必考慮底層 socket 的建置，有點像是一個函式庫，供上層的元件使用。

[Session Manager]

此元件是整個虛擬網路卡管理員的核心程式，對於網路換手作一系列的動作，它主要的目的就是維持本系統的虛擬網域可以正常通訊。簡單介紹一下流程，首先，它會透

過 Status Monitor 取得底層所有可用的網路介面，當知道有那些介面之後，選擇其中一個當主要的介面，然後通知 socket handler 建立 socket 時要用那個介面 (bind IP)當網路傳送的出入口，當然 socket handler 也會試著建立其他的備用 socket (備用路徑)。當換手發生時，會通知 socket handler 改變主要的介面，如果有備用 socket 已經建立好了，當然就直接切換過去，沒有的話便查詢可建立 socket 的所有路徑表，並盡快的建立一個可以用的 socket，當 socket 建立好之後即表示可以繼續正常通訊。

- **幾個重要的流程**

[啟動系統]

A➔B➔Z➔C➔Q➔D➔E➔F➔L➔F➔G➔C➔X➔D➔E

A: 使用者下啓動指令給系統。

B: Session Manager 向 VNIC Controller 要求啓動虛擬網卡，並準備緩衝區暫存資料。

Z: VNIC Controller 對虛擬網卡下達一些控制指令。

C: Session Manager 向 Status Monitor 要目前實體網卡的運作狀態及本端可用的 IP。

Q: 透過 IpHelper 查詢上述要求。

D: 回報給 Session Manager

E: 對 socket handler 要求建立主要連線。

F: 提供使用者帳號以及密碼等認證資訊給 U-VLan Protocol Handler。

L: 透過 socket handler 所收送的封包，進行認證的動作。

F: 要求把本端上所有的 IP 資訊通知遠端，讓遠端知道本端的所有 IP。

G: 收到遠端主機上所有的 IP 資訊。

C: 請 Status Monitor 了解本端以及遠端所有 IP 的組合及連線狀況。

X: Status Monitor 著手處理 IP 組合的測試連線。

D: 回報可用的 IP 組合，也就是其他條可用的路徑。

E: Session Manager 要求 socket handle 建立備用的 socket 連線。

[介面斷線-換手]

D➔E

D: Status Monitor 偵測到有網路介面斷掉，也就是某個 IP 不能使用，便立即回報給 Session Manager。

E: 要求立即切斷所有使用該 IP 來通訊的 socket，並切換到備用的 socket 連線，如果沒有備用的 socket 連線，就會以目前已知遠端所有的 IP 資訊爲依據，嘗試建立主要的 socket。當 socket 連線建立起來之後，便恢復原本通訊的狀態。

[介面連上線]

D➔F➔C➔X➔D ➔E

D: Status Monitor 偵測到有網路介面啓動，也就是本端多一個 IP 可以使用，便立即回報給 Session Manager。

F: 要求把本端上所有的 IP 資訊通知遠端，讓遠端知道本端的所有 IP。

C: 請 Status Monitor 了解本端以及遠端所有 IP 的組合及連線狀況。

X: Status Monitor 著手處理 IP 組合的測試連線。

D: 回報可用的 IP 組合，也就是所有可用的路徑。

E: Session Manager 要求 socket handle 建立備用的 socket 連線。

第六章 實驗

設計好 Ubiquitous Virtual LAN Platform 後，接著測試本系統的效能。測試項目有二，包含換手的延遲，以及用 ICMP 封包測量回應時間以了解本系統的 overhead。本章的實驗結果說明，本系統能提供 Session Continuity 的網路環境、短的換手時間以及對系統並不會有太大的 overhead。實驗如下說明之。

6.1 換手的延遲

換手的延遲是許多換手機制設計的重點，因爲當換手發生的時候，有些應用程式只能接受較短的延遲時間，對延遲很敏感。本實驗就是要測試本系統的延遲效能，此實驗包含兩部份，一個是單介面網路裝置，就是說只有一個實體的網路介面可以連線，另外一個是雙介面網路裝置，也就是有兩個可以存取網路的實體。實驗介紹如下。

6.1.1 單介面網路裝置實驗

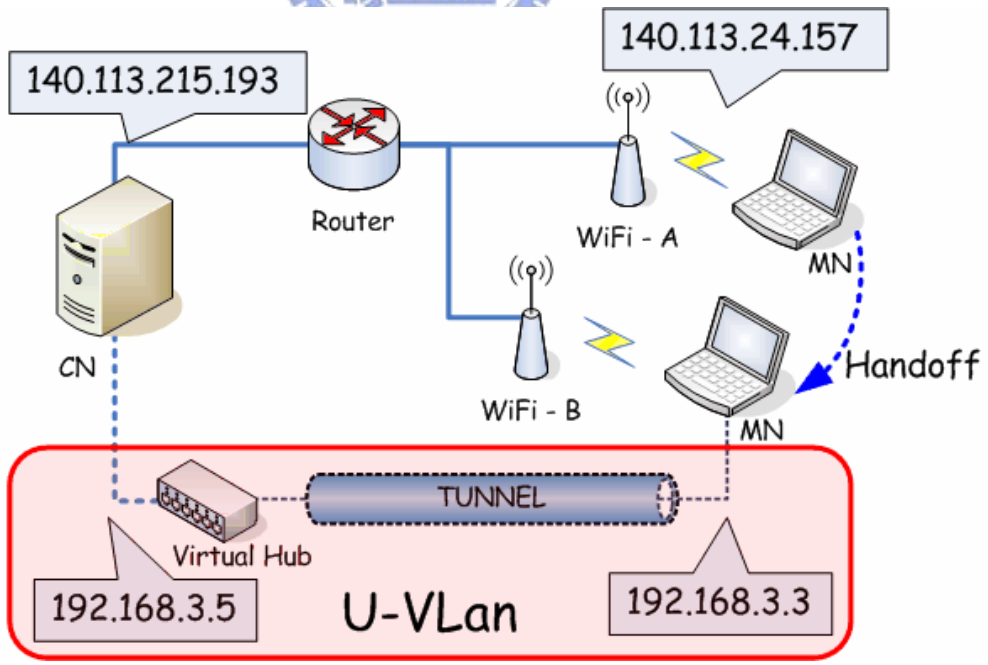

圖21: 單介面網路裝置實驗架構

實驗中的 MN，只啓動無線區域網路(WLAN)裝置，所以一次只能和一個基地台聯繫。首先 MN 和基地台 A 連接並透過它連線到 CN 建立起 U-VLan。當環境建立好之後，從 MN 週期性的對 CN 發送封包，每 10ms 發送一次，透過本系統的虛擬網卡發送，並在 CN 監控虛擬網路卡所收到的封包。相對的，也從 CN 週期性的發送封包給 MN，時間間隔也是 10ms，MN 也在虛擬網路卡監控所收到的封包。而發送程式爲自行撰寫陽春型的發送器，就很單純的發送封包，而監控程式，使用 Wireshark [32]，這是一套好用方便的網路封包監控器，爲 Ethereal [33]的進化版。當環境都用好了之後，等待一段時間之後，MN 從 Wifi-A 換手到 Wifi-B，下面爲實驗結果。

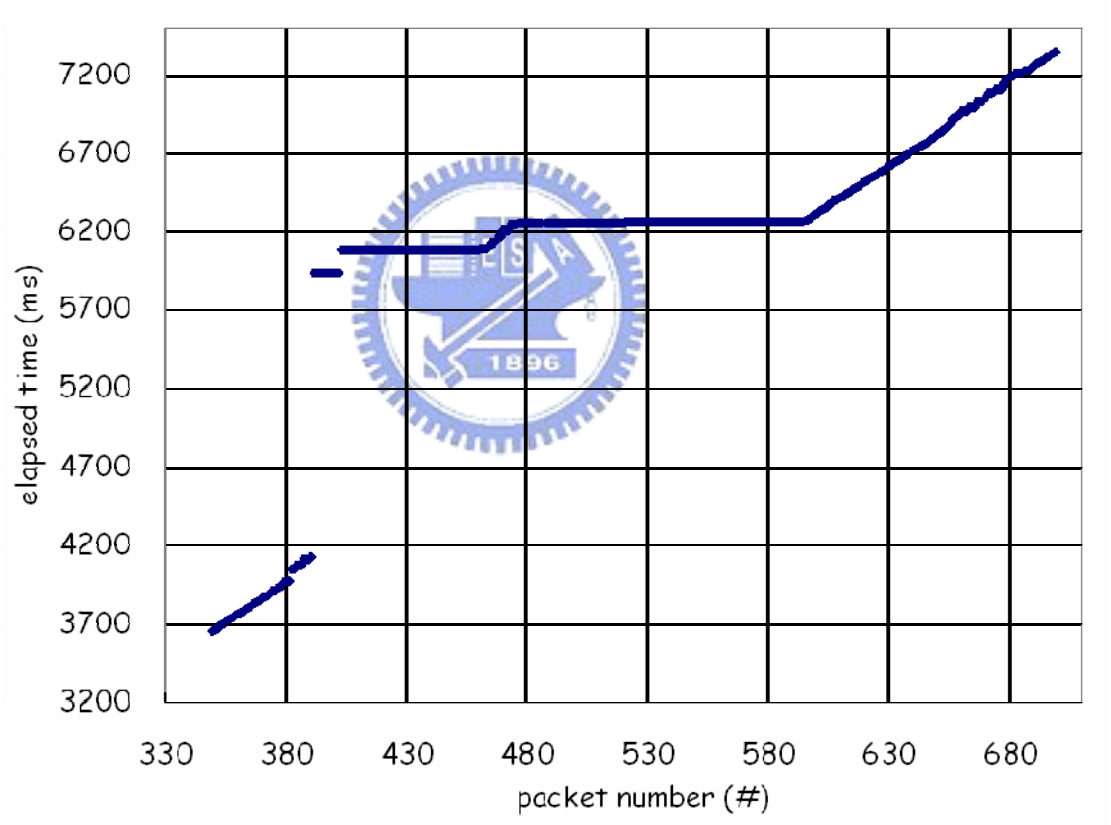

圖22: 單介面網路裝置換手延遲 (MN➔CN)

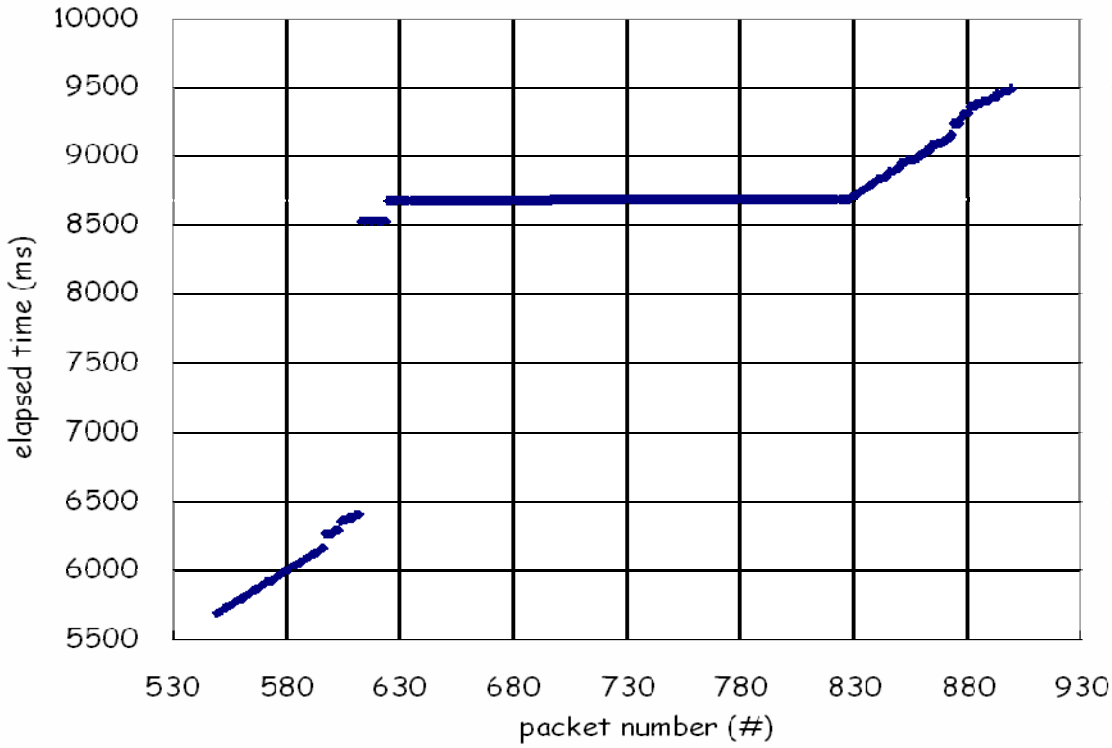

圖23: 單介面網路裝置換手延遲 (CN➔MN)

如圖 22，CN 所收到封包分佈的結果，由圖中得知，在約 4.136 秒鐘的地方爲 MN 換手的時間，約在 5.939 秒結束的附近又收到封包，這表示換手的延遲時間爲 1.803 秒，換手之後便又繼續維持正常的收送封包。再看圖 23，MN 所收到封包的分布結果也是類似，在約 6.407 秒鐘的時候 MN 和原本基地台斷線，也就是開始換手的時間點，之後約在 8.517 秒鐘的地方又再次收到封包，中間換手的時間爲 2.11 秒。

整體來看，換手的時間有點久，實際上在實驗的時候，該換手的時間很不穩定，最快有 900ms，最慢也有 5000ms，猜測可能有時延遲會更高，不過大多都在 2 秒附近，圖 23 的數據爲常出現的例子。而這個延遲基本上並不是本系統所造成，而是作業系統，硬體網卡以及網卡上面驅動程式的延遲，在這部份，因爲基地台 ESSID 的切換，802.11 所需要做的事情，以及在 PHY 層之上還有 IP 層需要考慮，尤其是 IP 層，微軟作業系統會對 IP 層琢磨比較多時間，Windows 會在剛連接的乙太網路，或是無線區域網路上廣播，問有沒有其他人在使用該網卡所綁定的 IP，沒有的話才會真正的綁定，之後該介面才算可以使用，所以延遲的比重也相對的多。而本系統的延遲可以說是很小可以忽略。另外，在恢復通訊之後，有出現很大量的傳輸，也就是在接近的時間點，收到許多封包，這表示啓動換手之後，還沒被傳送出去的封包，會被先存在緩衝區裡保留，直到又恢復通訊之後再繼續傳送，之後也就恢復正常。

6.1.2 雙介面網路裝置實驗

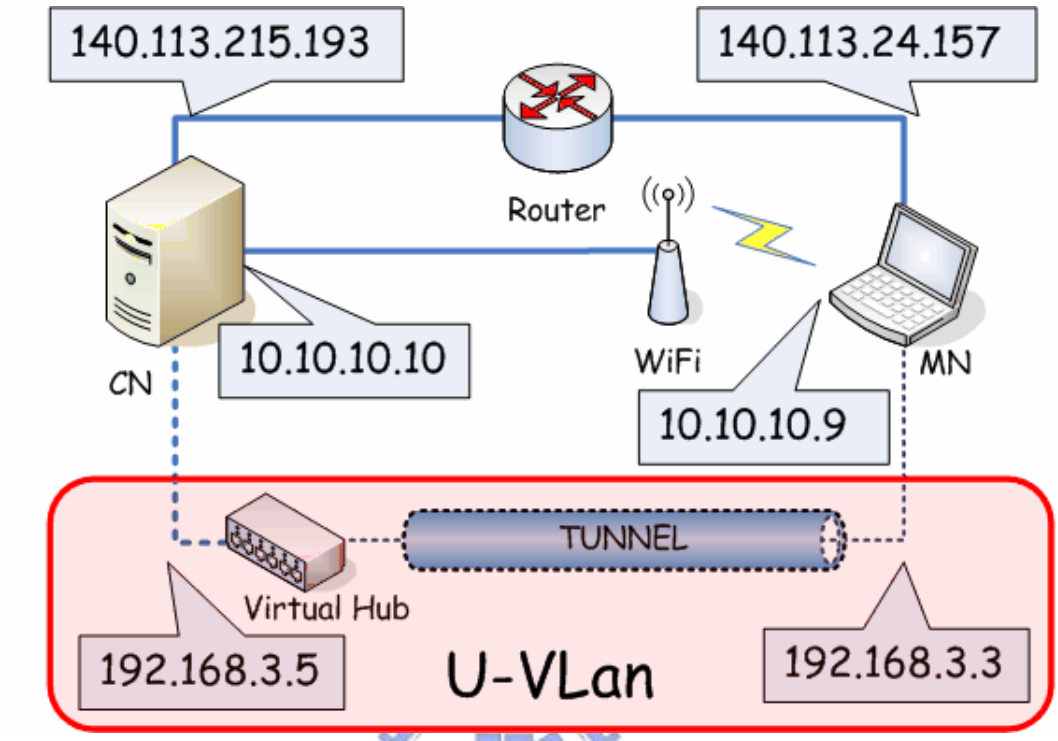

圖24: 雙介面網路裝置實驗架構

接下來這個實驗和前一個差不多，只是 MN 上面的網路介面改變了，變成兩個實體的網路裝置，如圖 24，一個是乙太網路，上面綁著 IP 爲 140.113.24.157，另外一個網路介面爲 IEEE 802.11 無線網路，上面綁著 IP 爲 10.10.10.9 之虛擬 IP。CN 上面也有兩個 IP，一個是 140.113.215.193，另一個爲 10.10.10.10 之虛擬 IP。接著同樣都在 MN 以及 CN 上面起動 U-VLan 平台，和上一個實驗一樣，MN 每隔 10ms 就透過 192.168.3.3 虛擬介面送出一個封包，CN 端也一樣在 192.168.3.5 虛擬網卡上監聽封包。反過來也是一樣，CN 每隔 10ms 透過 192.168.3.5 虛擬介面送出一個封包，MN 也一樣在 192.168.3.3 虛擬網卡上監聽封包。只是此次 MN 的換手不同，不同的是，單純的把乙太網路線拔除，也就是說 140.113.24.157 這個 IP 不能使用，依照 U-VLan 的特性，系統會自動切換到另一個可以連線的 IP，也就是 10.10.10.9 這個 IP，當然 CN 也會換到 10.10.10.10。下面是實驗的結果。

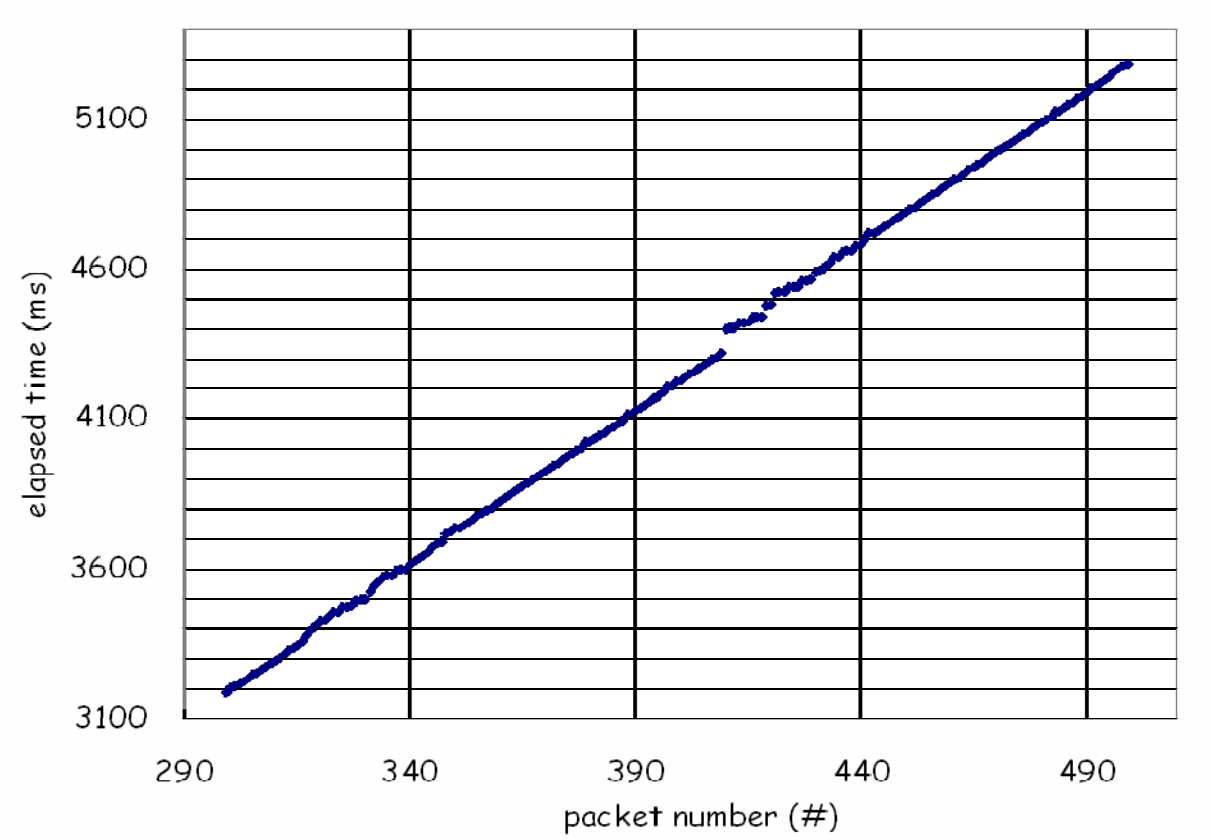

圖25: 雙介面網路裝置換手延遲 (MN➔CN)

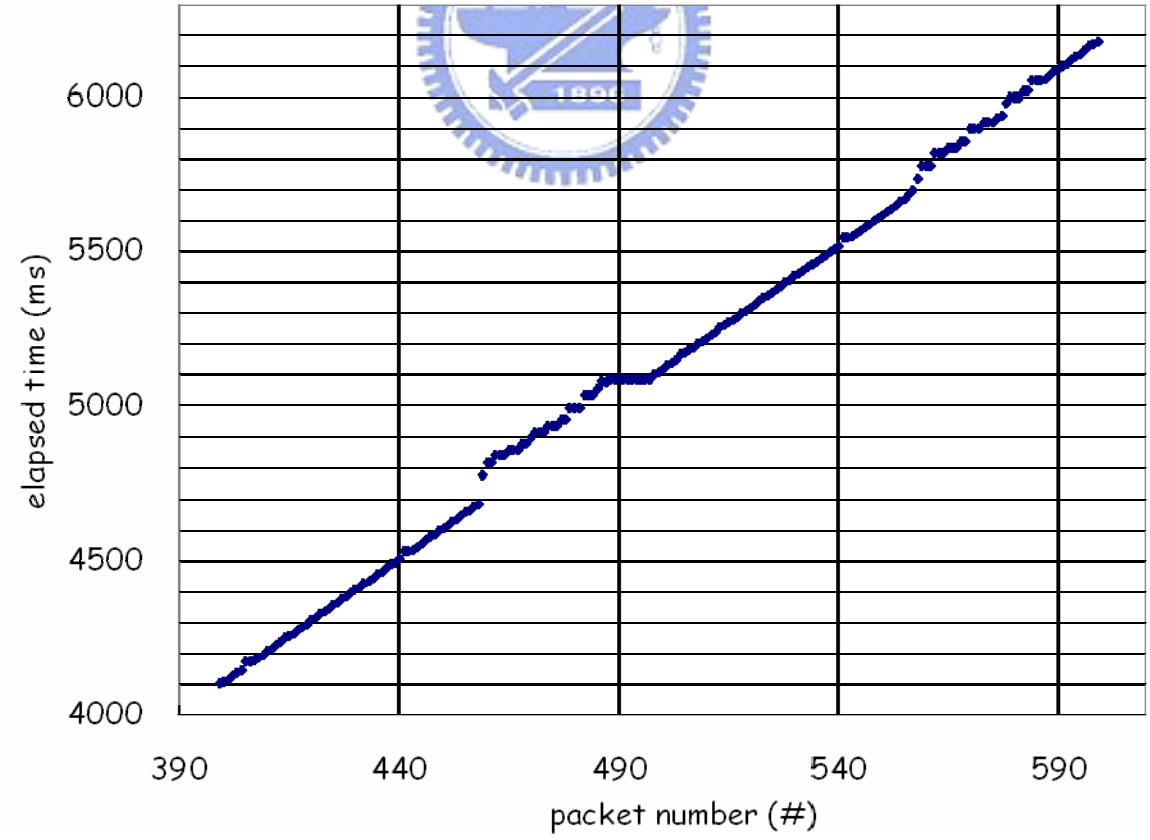

圖26: 雙介面網路裝置換手延遲 (CN➔MN)

先看圖 25，在換手的時機點，也就是拔掉網路線的時間大約在 4.318 秒的地方，中間停頓一下，接著在 4.398 秒又有封包的出現，停頓的時間約爲 80ms 左右，很少的時間。而圖 26 中，網路線被拔掉的時間點爲 4.688 秒，接著在 4.777 秒的時候又繼續收到，中間的延遲時間爲 89ms，也是很少的時間。時間會這麼小是因爲 U-VLan 有準備被用的替代方案，起初，此平台在 MN 以及 CN 上建立的時候，是透過 140.113.24.157 和 140.113.215.193 這兩個 IP 來建立的 (socket 連線)，建立之後，MN 以及 CN 會知道彼此的所有 IP，並嘗試建立備案的 socket 連線，當然 10.10.10.9 以及 10.10.10.10 就會有一 socket 連線存在，而此時原先所建立的 socket (140.113.24.157 和 140.113.215.193)爲主要的，因爲它是乙太網路，優先權比較高，而之後再建立的 socket 爲備用的。一旦斷線發生，意即 140.113.24.157 失效，以它爲基礎的 socket 連線就會斷掉，U-VLan 發現之後會立即切換到備用的連線，把備用的變成主要的連線，繼續運作。因此對系統來說，只是單純換一個 socket 來傳送資料而已，所以會立即傳送，不會有所等待。不過有另一種情況，就是當備用連線尚未被建立時，主要的連線斷掉，系統發現沒有備用連線，便會嘗試在所有已知的 IP 之間搜尋可以利用的所有路徑組合，試著找出一條可以通的路徑，如果是這種情形，時間可能延遲些，但 IP 所有的組合只要有一組可以通，建立 socket 連線再加上認證的動作，基本上不會太久。如果斷線後沒有任何 IP 組合可以使用，當然 timeout 就不可避免。

6.2 測量 U-VLan 的 overhead

要了解本系統的 overhead 便需要實際的測量，本實驗則是透過測量 ping 的回應時間來了解，也就是說封包經由本系統以及不經過本系統，中間的落差基本上就是經過本系統所需要多花的時間。ping 的測量也有其他好處，可以了解網路連線的穩定度，有些網路是需要反應時間快的，不可以延遲太久，像即時戰略遊戲，第一人稱射擊遊戲，以及目前最紅的 MMO RPG 等，都不允許網路太大的延遲，所以測量 ping 的反應是需要的。

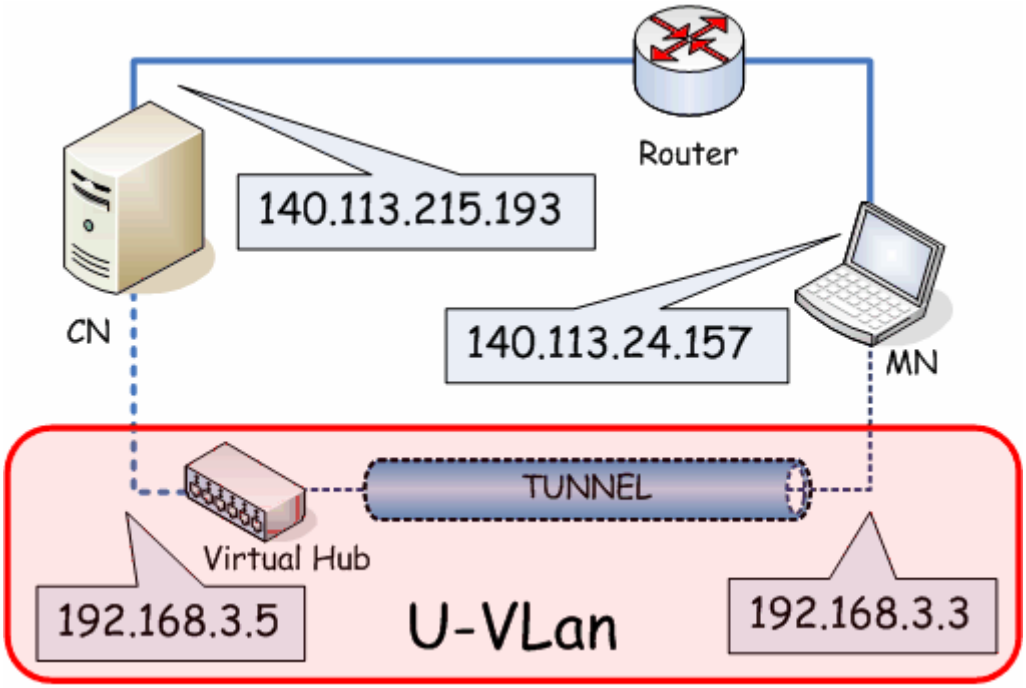

圖27: Ping 回應時間實驗架構

測試環境很簡單，只用單一的乙太網路介面，透過該實體網路，建立本系統平台。環境建立好之後，便在 CN 上頭下指令，對 MN 作 ping 的動作。再進一步說明，本實驗分成兩部分，一部份就是傳統乙太網路上對另外一台電腦 ping 的測量，也就是單純的從 IP 爲 140.113.215.193 發送 ICMP 的封包給 140.113.24.157 這端，來測量回應的時間。另外一部分就是對 192.168.3.3 的 IP 做 ping 的動作，網域的關係，ping 的封包會經過 CN 上頭的虛擬網路卡即 192.168.3.5 送出，再經過本系統的處理，最後送達 192.168.3.3 的虛擬網卡上。此實驗之 ICMP 的封包大小以遞增的方式測試，由小到大，指數成長的方式測量，從 32bytes 開始，每次增加都增爲兩倍，直到 ICMP 所規定的最大封包爲止。每次測量，發送 100 個單位的 ping，並紀錄回應的時間值，數據的收集以最大值，最小值，平均值來分析。下圖爲實驗的結果，B 爲本平台 U-VLan 的實驗數據，A 爲單純乙太網路的數據。

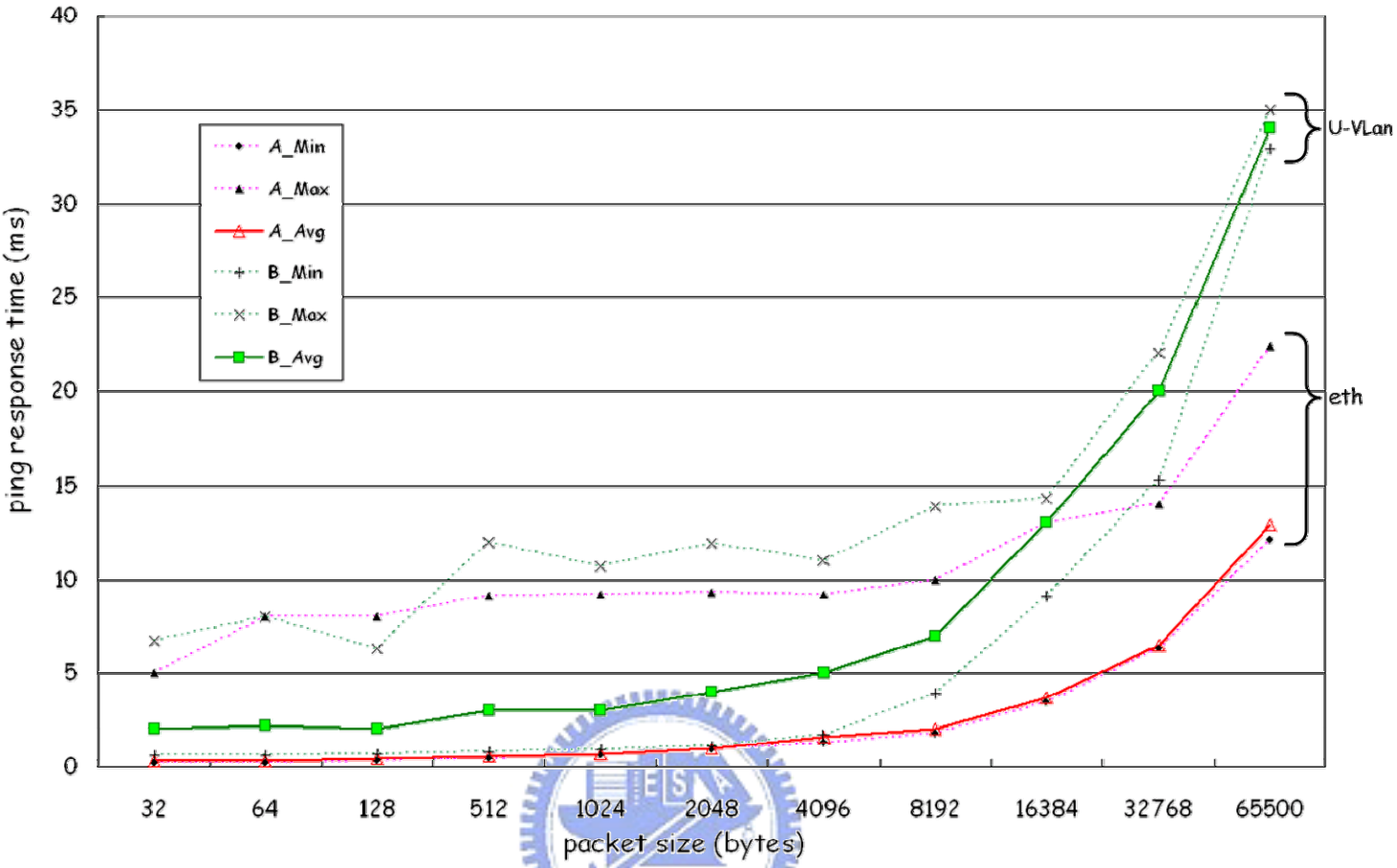

圖 28: Ping 回應的時間

圖 28 中，正常情況下的 A，當 ICMP 的封包比較小的時候，反應其實滿快的，由於乙太網路的設計，正常的封包最大爲 1514 bytes，如果再大的話，會被切割成兩個，甚至更多個封包，所以在圖中，對於 1024 bytes 以下的回應時間，基本上都差不多，因爲它並不會產生兩個以上的封包傳送。至於超過 1024 bytes 的數值，因爲送出的量是以指數的方式成長，所以如預期，圖形皆有指數成長的趨勢。圖中 A 類線條顯示，該最小值和平均值很接近，而和最大值中間有一個落差，這顯示可能是因爲硬體的不穩定，包含中間的交換器等因素，所以會忽然出現一個最大值，不然應該都是處於最小值的範圍。而本系統 U-VLan 和 A 有點類似，平均值也是偏向最小值，只不過沒有那麼近，這可能是因爲軟體的運作，時間不穩定的關係，所以不會常常都處於最小值的範圍。

另外，以最大值來比較，A 類的最大值以指數成長，且比較平滑，而本平台 U-VLan 的最大值，雖說也是指數的成長趨勢，但是會有不穩定的現象，也就忽高忽低的，這原因可以歸咎於 U-VLan 是以軟體的模式來模擬乙太網路的機制，所以既然是軟體，就不

會那麼的穩定。最小值，基本上是可以表現出此網路系統的最佳的狀態，因此該值可以直接當作最佳的反應時間，從圖中可知，最小值的曲線是最漂亮的一組，也是最符合指數方程式，較少偏差，也因此可以用數學式子來預測更大網路封包的來回所需要的時間，只是最小值往往在現實的情況下不會總是出現，就像理想氣體一樣，現實中的氣體會有一些微小的因素影響著，實際上的網路也一樣，也會有些微的小因子在影響著，因此看平均值比較可以了解真實的情況。

在 Ping 回應時間的圖中，兩條實線爲平均值，也代表著真實的情況。在較小資料的時候，回應時間也小，U-VLan 會比乙太網路的時間還多 2 ms ~ 3 ms，這意味著兩端系統多處理封包資料的時間約爲 2 ms ~ 3 ms，單一端處理的時間爲 1ms~1.5ms，滿小的，可以忽略，而當傳送的資料變大的時候，處理的時間就明顯的增加，從 1 Kbyte 的 2.5 ms，8 Kbytes 的 5 ms，16 Kbytes 的 10 ms，32 Kbytes 的 13.5ms 到最大的 64 Kbytes 的 21.5 ms，均繼續成長，值得注意的是值都小於 25ms，不會太大，而且在一般的網路封包，並不需要這麼大的封包送出後再等回應，一般都只有一個封包送出，之後就會有封包回應，所以平均的時間應該以乙太網路最大可接受的範圍爲基準，也就是 1.5 Kbytes，而此資料大小的來回反應時間，基本上都約在 3 ms 附近，反應算是很快的，也因爲小，增加出來的負擔開銷也就可以忽略掉。

實驗結果顯示 U-VLan 的平台並不會有太大的 overhead，且透過本平台運作的應用程式，不會因爲本系統的微小延遲而產生問題，這個微小的延遲可以假想爲中間多經過一些路由器的延遲即可。

第七章 結論與未來發展

7.1 結論

隨著有線以及無線網路的發展，越來越多裝置上內建有許多有線以及無線的網路介面，也因此使用者使用網路的習慣慢慢改變，而漫遊就是最常出現的例子。本篇論文隨著這樣的趨勢以及需求設計了一套網路平台，本平台可以自動的處理網路換手的問題，對於使用者來說，可以享受到漫遊於各個網路而不會產生斷線的問題。此外，在應用程式的配合上，現有絕大部分的網路程式皆無須經過任何的修改便可在此平台上運行並享有平台所提供的服務，而對於應用程式的設計也可以不用考慮換手的議題，專心於應用層面的開發。

本平台在微軟的作業系統 Windows XP (Service Pack 2)上開發成功，提供一個遍佈式的虛擬區域網路，成功的把 mSCTP 中，一個終端可以有多個 IP 的特性發揮出來，並應用在虛擬區域網路裡，再者本系統完全避開 Mobile IP 的缺點，包含部署代理人、沒有效率的路徑傳送、封包遺失以及防火牆 NAT 的穿透等缺點。另外對於單網路介面裝置來說，可以很順利的水平換手 (Horizontal Handoff)，並繼續保持原本的網路連線，而對於多種網路介面裝置來說可以達到垂直換手 (Vertical Handoff)且有著很小的換手延遲，因此成功的整合了異質網路，提供 Session Continuity 的網路環境。

7.2 未來發展

憑借著本平台的特性以及優點，可預期的將來，會有許多有淺力的應用發展在本平台上，如網路語音電話 (Voice over IP)，媒體點播 (Media on Demand)，網路遊戲 (Internet Game)…等等，因此需要加強某些部份，如 socket 建立時認證機制的安全性，虛擬交換器 (virtual switch/hub)中封包交換的處理機制可以更有效率，以及虛擬網路卡上 IP 分配的問題。另外可以試著整合 SIP 的特性使得 MN 以及 CN 均可以同時漫遊於網路間，以及提供使用者可以方便設定自己對於網路的偏好，並依照使用者的偏好，在適當的時機點主動換手，提供最友善的網路環境。

參考文獻

[1] Z. Jiang, K.K. Leung, B.J. Kim and P. Henry, "*Proxy Servers Based Seamless Mobility Management*," WCNC.2002, Orlando, FL, 2002.

[2] H. Sivakumar, S. Bailey and R.L. Grossman, "*PSockets: The Case for Application-level Network Striping for Data Intensive Applications using High Speed Wide Area Networks*," Proceedings of the 2000 ACM/IEEE Conference on Spuercomputing, 2000.

[3] 林義欽, "*行動無縫式 TCP 連線環境架構之研究*," 國立交通大學資訊工程系, 碩士論文, August 2004.

[4] T. Goff and D.S. Phatak, "*Unified Transport Layer Support for Data Striping and Host Mobility*," IEEE journal on selected areas in communications, Vol. 22, No. 4, May 2004.

[5] I. Aydin, W. Seok and C.C. Shen, "*Cellular SCTP: A Transport-Layer Approach to Internet Mobility*," IEEE Computer Communications and Networks (ICCCN) 2003, October 2003.

[6] S. Kashihara, K. Iida, H. Koga, Y. Kadobayashi and S. Yamaguchi, "*End-to-End Seamless Handover using Multi-path Transmission Algorithm*," Proc. Internet Conference 2003, October 2003.

[7] D.A. Maltz and P. Bhagwat, "*MSOCKS: An Architecture for Transport Layer Mobility*," IEEE INFOCOM, April 1998.

[8] L. Ma, F. Yu, V. Leung and T. Randhawa, "*A New Method to Support UMTS/WLAN Vertical Handover Using SCTP*," IEEE Wireless Communications, Vol. 11, No. 4, pp.44-51, August 2004.

[9] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang and V. Paxson, "*Stream Control Transmission Protocol*," IETF RFC 2960, October 2000.

[10] C. Perkins, "*IP Mobility Support*," IETF RFC 2002, October 1996

[11] H. Parikh, H. Chaskar, D. Trossen and G. Krishnamurthi, "*Seamless Handoff of Mobile Terminal from WLAN to cdma2000 Network*," Proc. of IEEE 3G Wireless2003, San Francisco, May 2003.

[12] R. Chakravorty, P. Vidales, K. Subramanian, I. Pratt and J. Crowcroft, "*Performance Issues with Vertical Handovers – Experiences from GPRS Cellular and WLAN Hot-spots Integration*," Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications (PERCOM'04), March 2004

[13] I.W. Wu, W.S. Chen, H.E. Liao and F.F. Young, "*A Seamlcss Handoff Approach of Mobile IP Protocol for Mobile Wireless Data Networks*," IEEE Transactions on Consumer Electronics. Vol. 48, no. 2, pp. 335-344. May 2002

[14] 陳伯剛, "*整合 GPRS 與 Wireless LAN 資料網路之兩階層式架構與通訊協定*," 國立交通大學資訊工程系, 碩士論文, August 2004.

[15] 邱文岳, "*WLAN-GPRS 整合網路下無接縫換手的方法*," 國立交通大學資訊工程系, 碩士論文, August 2004.

[16] M. Handley, H. Schulzrinne, E. Schooler and J. Rosenberg, "*SIP: Session Initiation Protocol*," IETF RFC 2543, March 1999.

[17] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley and E. Schooler, "*SIP: Session Initiation Protocol*," IETF RFC 3261, June 2002.

[18] J. Zhou and N. Sun, "*A Seamless Handoff Scheme for Mobile IP*," Vehicular Technology Conference, 2006. VTC 2006-Spring. IEEE 63rd, 2006.

[19] M. Samad and N. S. M. Kasim, "*Effect of Transmission Delay And Tunneled Traffic In A Wireless Mobile IP Network*," International RF and Microwave Conference (RFM) 2006, September 2006

[20] H.V. Nguyen, S.H. Roh, J.K. Ryu and S. Park, "*A Modification for Fast Handover in Hierarchical Mobile IPv6*," The 9th International Conference Advanced Communication Technology (ICACT) 2007, Feb. 2007.

[21] D.W. Zhang and Y. Yaom, "*A Predictive Handoff Approach for Mobile IP*," International Conference on Wireless and Mobile Communications (ICWMC) 2006, 2006.

[22] 范榮軒, "*異質網路漫遊系統整合平台之設計與實作*," 國立交通大學資訊工程系, 碩士論文, August 2005.

[23] H. Levkowetz and S. Vaarala, "*Mobile IP Traversal of Network Address Translation (NAT) Devices*," IETF RFC 3519, April 2003.

[24] S.J. Koh, M.J. Chang, and M.J. Lee, Member IEEE, "*mSCTP for Soft Handover in Transport Layer*," IEEE Communications Letters, vol. 8, no. 3, March 2004.

[25] J. Stone, R. Stewart and D. Otis, "*Stream Control Transmission Protocol Checksum Change*," IETF RFC 3309, Sept. 2002

[26] R. Stewart et al, "*Stream Control Transmission Protocol Partial Reliability Extensions*," IETF RFC 3758, May 2004

[27] R. Stewart et al, "*Stream Control Transmission Protocol Dynamic Address Reconfiguration*," IETF Internet Draft

[28] W.C. Wang, C.H. Hsu, Y.M. Chen and T.Y. Chung, "*SCTP-based Handover for VoIP over IEEE802.11 WLAN Using Device Virtualization*," The 9th International Conference Advanced Communication Technology (ICACT) 2007, Feb. 2007.

[29] "*Softether*," http://www.softether.com

[30] "*WinPcap*," http://www.winpcap.org

[31] K. Sethom, H. Afifi and G. Pujolle, "*VIP: Virtual Interface Prototype for Mobile Communication*," IEEE 16th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC) 2005, September 2005.

[32] "*Wireshark*", http://www.wireshark.org

[33] "*Ethereal*", http://www.ethereal.com