國立交通大學

資訊科學與工程研究所

碩 士 論 文

802.11 無線網路下具網路拓樸知覺的頻道掃描機制之設計與實作

Design and Implementation of a Topology-Aware Scan Mechanism for 802.11 Wireless Networks

研 究 生：陳家銘

指導教授：曾建超　教授

中 華 民 國　九 十 六　年 六 月

802.11 無線網路下具網路拓樸知覺的頻道掃描機制之設計與實作

Design and Implementation of a Topology-Aware Scan Mechanism for 802.11 Wireless Networks

研 究 生：陳家銘　　Student：Jia-Ming Chen

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

中 華 民 國 九 十 六 年 六 月

802.11 無線網路下具網路拓樸知覺的頻道掃描機制之設計與實作

研究生： 陳家銘　　指導教授： 曾建超 教授

國立交通大學資訊學院資訊科學與工程研究所

摘　要

本論文針對 IEEE 802.11 無線區域網路提出一個具網路拓樸知覺的頻道掃瞄機制，這個機制不僅可以來縮短換手延遲時間，同時也可以更有效率地搜尋候選基地台。

近年由於行動運算技術的普及化，如今市面上許多的可攜式裝置如筆記型電腦、個人數位助理或是智慧型手機等，皆已具備存取 802.11 無線網路的能力。當使用這些具移動性的行動裝置在 802.11 網路漫遊時，將會遭遇到基地台換手的問題。尤其是使用 VoIP 等即時性應用服務，能夠快速且順暢地在基地台間換手是一項重要的需求。

802.11 網路基地台換手過可以分為基地台搜尋(discovery)與基地台連結(commit)兩個階段。研究顯示，基地台搜尋佔據整個換手過程的絕大部分時間。如果能有效改善基地台搜尋所需的時間，就可以降低因基地台換手造成的封包遺失和延遲等影響，進而達到快速換手的目的。目前有相當多的研究根據此現象而致力於降低基地台搜尋的時間。部分研究引進 NG (neighbor graph)的概念，透過 NG 紀錄的基地台相鄰關係與使用頻道，行動端點在基地台搜尋階段只需要針對鄰近的基地台進行掃瞄即可，避免掃瞄全部頻道或在一個頻道等待而造成過長的延遲時間。

然而 NG 的方法需要的額外負擔太大也未善用基地台部署的「地理性資訊」，NG 的建立是藉由系統中 MSs 漫遊所產生的基地台換手行為，動態紀錄並更新基地台的相鄰關係。但是這種動態建立的方式其實是不必要的額外負擔，因為基地台的部署是一種不會經常改變的靜態資訊。此外 NG 只是紀錄基地台的相鄰關係，並未提供真實環境底下鄰近基地台彼此位置的相對關係，因此行動裝置在基地台搜尋的過程仍會掃瞄不必要的頻道。有鑒於此，我們提出「具網路拓樸知覺的頻道掃描機制」，讓行動裝置可以利用基地台的網路拓樸資訊，在適當的時機量測特定鄰近基地台的訊號，觀察其衰減或增強的變化，再根據鄰近基地台的網路拓樸資訊，判斷出使用者可能的移動方向，找出合適的目標基地台，達到減少掃瞄所有鄰近基地台的目的，進而降低搜尋的時間。本論文以校園大樓環境為例，在 Linux 作業系統上搭配使用 Atheros 無線網路卡實作此機制。

測試與分析的結果顯示，此機制的加入的確可以達到降低換手動作對即時性應用服務品質造成的影響。

Design and Implementation of a Topology-Aware Scan Mechanism for 802.11 Wireless Networks

Student : Jia-Ming Chen Advisor : Dr. Chien-Chao Tseng

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

Abstract

This thesis proposes a topology-aware scan approach that not only can reduce handoff latency but also scan candidate access points more intelligently in IEEE 802.11 WLAN. With the advance of wireless technology and mobile devices, nowadays many portable devices on the market such as notebooks, PDAs or smart phones, are already equipped with IEEE 802.11 wireless LAN (WLAN) adapters. However, when a device roams in IEEE 802.11 networks, it needs to perform handoff procedure and change its point of network attachment from one WLAN access point (AP) to another. The handoff procedure is very time consuming and may render real-time services infeasible in IEEE 802.11 networks.

The handoff procedure consists of two phases: discovery and commit phases. Previous researches have already shown that the discovery phase contributes most of the handoff delay. Therefore, researchers have proposed various approaches to reduce the latency of the discovery phase. Among the previous proposals, Neighbor Graph (NG) is one of the most effective schemes. NG records the adjacency relations among APs and the channel used by each AP. With NG, a mobile station (MS) can determine which channels it need to scan and whether it has received all responses from the APs in a particular channel during the discovery phase. As a consequence, the number of channels probed decreases and so does the waiting time an MS staying in a channel.

However, the dynamic maintenance cost of NG and the lack of the geographic information of APs in NG make the NG approach less effective in

the next candidate AP discovery. Therefore, in this thesis, we propose a novel topology-aware scan mechanism where an MS can selectively scan only the APs the MS may possibly visit next according to the MS's moving orientation and the topology information of APs. With the topology information, an MS not only can shorten the handover delay but also can reduce the impact of AP discovery phase on normal packet transmission. We use a campus building as the test environment and implement our mechanism on the Linux OS platform with WLAN cards of Atheros chipsets. Experimental result shows the proposed mechanism can achieve fast handoff without introducing extra packet jitter and delay time.

誌　謝

本論文得以順利完成首要感謝我的指導教授-曾建超 博士，在我碩士研究兩年間的教誨，指導我正確的研究方法與態度，引領我走向正確的研究方向，並且提供一個完整且自由的研究環境，以及經由計畫的實作增加我的經驗。同時也要向我的論文口試委員：曹孝櫟 博士、嚴力行 博士及張弘鑫 博士致上謝意，感謝他們在百忙中撥空審查我的論文並且提供寶貴的意見。

另外還要感謝無線網際網路實驗室的同學、學長以及學弟，在我碩士求學過程中給予的意見、支持與鼓勵，讓我的碩士生涯豐富且充實。除此之外也要特別感謝 crocodile 以及許多好友，多年來一路陪著我成長，給我精神上與實質上的關心與協助，讓我度過許多的困難與挫折，謝謝你們。

最後，僅以此文獻給我摯愛的雙親，感謝你們在背後無止盡的支持，讓我能夠無後顧之憂完成學業。

目　錄

圖目錄

表目錄

第一章　緒論

1.1　研究動機

近年來，無線寬頻技術快速蓬勃發展，像是 IEEE 802.11 [1]、3G、3.5G 或是 WiMax 等，都希望能夠讓使用者透過無線的方式來享用寬頻的網路服務。其中，內建符合 802.11 b/g 規格的無線網路晶片已經是一台筆記型電腦必備的項目之一，這讓使用無線網路的人數快速成長。加上 802.11 基地台架設容易，價格低廉，因此讓許多人願意自行架設 802.11 基地台。而台北市正推行的無線新都計畫，目標建構一個涵蓋整個市區的 Wi-Fi 無線城市，為市民、校園和社區，甚至是來自境外的旅客，提供無線寬頻網路存取、互動式多媒體內容以及行動商務等應用服務平台。來自西班牙號稱帶動全球無線上網革命的 FON [2]，也在創辦人 Martin Varsavsky 以社群的方式號召下，逐漸開始蔓延開來，FON 以「免費分享、分享免費(Share for free, access for free)」為宗旨，希望創造一個開放與共享的無線網路環境。因此我們可以瞭解，以 802.11 為基礎的 WLAN 無線網路服務，不管在現在或是未來，都會在無線網路領域佔有一席地位。

802.11 無線網路的核心精神就是移動性，讓使用者能夠在使用過程中到處漫遊。但是，使用者常常會遇到一個問題，就是 802.11 無線網路在移動的情況下會發生斷線、訊號不穩定的問題。這種因為使用者漫遊而需要在基地台間換手的問題，對於需要即時性要求高的網路服務而言，是一種嚴重的考驗。因為當換手過程時間過久時，會造成即時性的網路服務(如 VoIP、DVB-H)停頓、延遲、甚至服務中斷，讓使用者感到不便。因此，解決換手所造成的上述問題，一直是許多相關研究的目標。

從以往的研究中我們得知，改進換手過程時須考慮的因素，主要可分為二。第一是何時進行換手？根據不同的換手時間點，所接收的無線訊號強弱會有不一樣的表現，使用者都是希望能夠有一個穩定並且夠好的訊號可以使用，無線網路卡何時開始尋找下一個基地台與何時換到的適合的基地台，這些時間點的決定，都會影響到換手的過程是否能夠快速且順暢。第二個因素則是換手的程序能多快？在換手的過程中，與目前基地台斷線後到連線上新的基地台，這中間會有一段時間是無法收送封包，這段時間的長短，決定了換手過程所造成的封包遺失(packet loss)、延遲時間(delay time)等影響會有多嚴重，我們都是希望這段時間能夠越短越好，如此才能達到快速、順暢甚至使用者無感覺的換手。

而根據[3]~[5]的研究報告我們可以得知，目前市面上網路卡所提供的作法，尚未能

有效的改善換手所造成的問題，無法達到即時性網路服務使用上要求。因此本論文即是從此處出發，希望能夠改善解決此問題。

1.2 研究目的

在上一節裡，我們已經瞭解所要研究的問題以及需求所在，因此本節中將簡單敘述我們想要達成的目標。

根據 A. Mishra 等[3]的研究，目前普遍建立布置的 802.11 無線網路服務，在基地台換手過程中，可以區分為基地台搜尋(discovery)與基地台連結(commit)兩個過程。研究顯示基地台搜索佔據整個換手過程的大部分 90%時間,如果能有效改善基地台搜尋所需的時間，就能夠縮短換手所需的時間，降低對網路服務的影響，因此先前已經有許多的研究[7]~[16]著重於此議題。

而基地台搜尋主要的目的與精神，是在適當的時間點去尋找出適合的基地台，接著再讓基地台連結程序去完成換手動作，在 802.11 的標準中並沒有規範無線網路卡要如何去進行搜尋，這部分屬於實作層面的問題。目前市面上的網路卡作法大多是透過發送 broadcast probe request 訊息的方式，去詢問 802.11 提供服務的 11 個頻道，再從收集到的 probe response 結果中去選擇訊號較好的基地台，而這樣的過程大概需要 350~500ms 的時間，對於即時性網路服務來說，這可能已經造成封包遺失、延遲等問題了。

本論文所提出的「具網路拓樸知覺的頻道掃描機制」，目的就是要改善這段搜尋所需的時間，透過所在位置網路拓樸資訊的輔助，縮短基地台搜尋的時間，達到降低換手過程會造成的影響。我們會將所提出的機制，在 Linux 作業系統下進行實作，修改開放原始碼的無線網路卡驅動程式，搭配上自行開發的換手管理程式，完成支援此機制的平台。因為此機制與作法的設計精神並沒有更改 802.11 標準的規範或是修改到基地台提供的功能，所以修改過的無線網路卡可以相容運行於目前布建的 802.11 無線網路環境，因此我們最後將利用實際大樓所建置的 802.11 開放系統式(open system)無線網路環境，測試與分析此機制運作的結果與效能，驗證此作法的可行性。

1.3 章節簡介

本篇論文的章節編排與內容簡介如下：

第一章： 緒論，簡介本篇論文的研究動機及希望達成的目標。

第二章： 背景知識概述，介紹本篇論文所要探討主題以及所應用到的相關知識，包括 802.11 無線網路下的換手流程的延遲時間分析，以及實作上會應用到的 Linux 作業系統下使用者空間與核心空間訊息溝通機制、Wireless Extension API。

第三章： 相關研究，簡介本篇論文主題之前發表過的相關研究文獻資料。

第四章： 具網路拓樸知覺的頻道掃描機制之設計，說明我們所提出具網路拓樸知覺的頻道掃描機制的原理與運作流程，以及機制當中所運用到的相關技術。

第五章： 具網路拓樸知覺的頻道掃描機制之實作，描述我們在 Linux 作業系統下進行實作的各項工作，包括無線網路硬體與驅動程式的選擇、系統的整體架構，以及各個元件的運作機制。

第六章： 效能分析與結果，針對我們所提出與實作的具網路拓樸知覺的頻道掃描機制，進行相關的實際環境測試，並提供效能分析報告。

第七章： 結論與未來工作，總結本篇論文的結果，以及未來可繼續研究的方向。

第二章　背景知識介紹

2.1 IEEE 802.11 無線網路

IEEE 802.11 [1]無線網路的全名是無線區域網路(Wireless LAN)，顧名思義是將區域網路(Local Area Network, LAN)無線化，不需要透過實體傳輸線即可以使用，是目前相當普及網路服務。802.11 標準定義了透過無線電波溝通的媒體存取控制層(MAC layer)和實體層(PHY layer)介面，此標準在 1997 年由 IEEE 制訂之後急速發展，目前根據此規格延伸出的 802.11 家族包括了有：

- 802.11： 應用在無線的區域網路上，使用 2.4GHz 的頻帶作為無線電波的使用範圍。802.11 的實體層可由 Frequency Hopping Spread Spectrum (FHSS)或由 Direct Sequence Spread Spectrum (DSSS)所完成。
- 802.11a： 802.11a 是 802.11 原始標準的一個修訂標準，於 1999 年獲得批准。802.11a 在實體層所使用的是 Orthogonal Frequency Division Multiplexing (OFDM)的方式，而非原 802.11 所使用的 FHSS 或 DSSS。
- 802.11b: 與 802.11a 一起在 1999 年 9 月通過的修訂標準, 使用的是 2.4GHz 的頻帶。802.11b 在實體層的技術是採用 DSSS，並採取與乙太網路(Ethernet)相同的通訊協定 CSMA/CA (carrier sense multiple access with collision avoidance)。
- 802.11g： IEEE 在 2003 年 7 月通過的第三種修訂標準。延伸自 802.11b 的架構, 所使用的頻帶是在 2.4GHz 的範圍, 並具有可提供 54Mbps 的傳輸能力，因此相較於 802.11b 的 11Mbps 其傳輸率大大的提升了不少。在實體層上採用 OFDM 的技術，因此可以獲得較高的資料傳輸率。
- 802.11n： IEEE 標準制定組織於 2003 年成立新的工作小組 802.11n, 主要目的為制定下一世代的高速無線網路, 希望增加無線網路的傳輸率與解決多重路徑的干擾，讓無線網路資料傳輸速度提高到 540Mbps。802.11n 也將會比目前的無線網路傳送到更遠的距離。802.11n 實體層是以 MIMO (multiple-input multiple-output) OFDM-based 為技術主軸，MIMO 使用多個發射和接收天線來允許更高的資料傳輸率, 許多研究已明確地證明 MIMO OFDM-based 系統架構在具有多路徑干擾的環境下可以有效地增加傳輸速率。

總結 802.11 家族的資料如 Table 2.1：

Table 2.1: IEEE 802.11 wireless protocol summary.

(Adapted from [26])

Protocol	Release Date	Op. Frequency	Data Rate (Max)	Throughput (Typ)	Range (Indoor)	Range (Outdoor)
Legacy	1997	2.4-2.5 GHz	2 Mbps	0.7 Mbps	~25 meters	~75 meters
802.11a	1999	5.15-5.35/5.47-5.725/5.725-5.825 GHz	54 Mbps	23 Mbps	~30 meters	~100 meters
802.11b	1997	2.4-2.5 GHz	11 Mbps	4 Mbps	~35 meters	~110 meters
802.11g	2003	2.4-2.5 GHz	54 Mbps	19 Mbps	~35 meters	~115 meters
802.11n (draft)	2007(D2.0)	2.4 GHz and/or 5 GHZ	248 Mbps (2x2 ant)	74 Mbps	~70 meters	~160 meters

2.1.1 802.11 無線網路架構

在 802.11 網路底下，不論是固定式或可攜式電腦都被視為一台 Station (STA)，當兩台或更多的 STAs 聚在一起彼此通訊時，這些 STAs 的集合被視為一個 Basic Service Set (BSS)，BSS 是 802.11 底下的基本建構元件(building block)。

當一個 BSS 是獨立存在運作且沒有連接上網路時，稱為一個 Independent Basic Service Set (IBSS)，也就是 802.11 所定義的一種網路架構類型：Ad-hoc wireless network，如 Figure 2.1。在 IBSS 中，STAs 彼此間直接點對點通訊，通常 IBSS 是由少數幾部 STAs 針對特定目的而組成的暫時性網路，持續時間不長且規模較小。

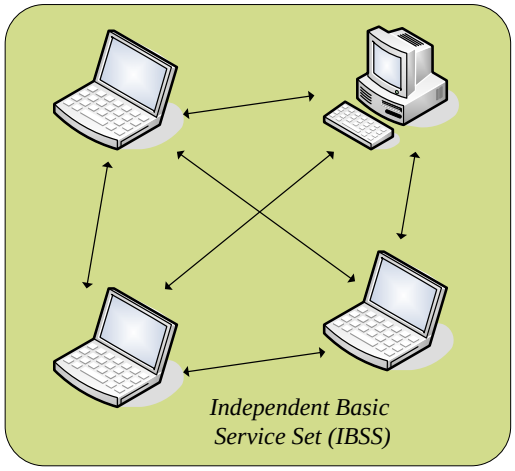

Figure 2.1: Ad-hoc wireless network.

802.11 所定義的另外一種網路架構類型，稱為 Infrastructure wireless network。在這種網路架構裡，存在一台特殊的 STA 被稱為是基地台(access point，AP)，基地台

所負責的是將 BSS 與現存的 Distribution System (DS)相連接，並提供 Station Service 與 Distribution Service，網路架構下的其他 STA 透過此基地台來連接其他的無線網路設備，或是存取有線網路的資源。

多個 BSSs 可以經由 DS 相互連結而擴大 WLAN 的服務涵蓋範圍，這個由 BSS 拼湊出來、擴大的無線網路在概念上還是屬於同一資料鍵結層(data link layer)，因此在不同 BSS 下的 STA 可以使用 link layer address 直接通訊；而這個由多個 BSSs 與 DS 所組成的無線網路就稱為 Extended Service Set (ESS)，整個網路架構如 Figure 2.2 所示。

以 Infrastructure wireless network 架構為主的 802.11 網路服務，目前在機場、學校、企業大樓、大飯店或是商店等環境中廣泛的被建置與使用，本篇論文所探討的 802.11 無線網路架構也就是此類型。

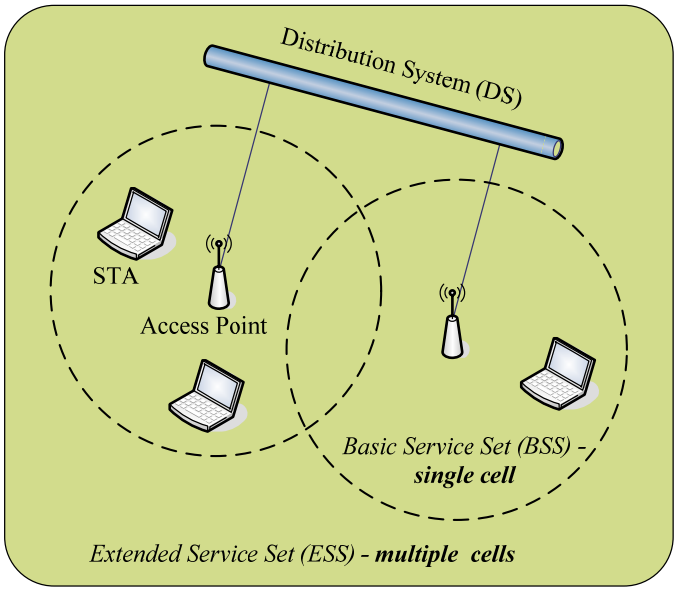

Figure 2.2: Infrastructure wireless network.

而根據 802.11 標準的定義，當使用者在無線網路中移動時，有三種可能的情境會發生，描述如下：

a) No Transition：

使用者不動或只是在所連結基地台附近移動，沒有超過基地台所能提供服務的涵蓋範圍。

b) BSS Transition：

在同一個 ESS 底下，使用者從某一個 BSS 移動到另一個 BSS，也就是

從某一台基地台更改連結到另一台基地台，這正是本篇論文所探討的換手情形。

c) ESS Transition：

使用者從某一個 BSS 移動到另一個 BSS，而這兩個 BSS 分屬於不同的 ESS，也就是說使用者跨越了 ESS 所提供的服務範圍，在這種情況底下，802.11 標準並不保障上層的連線狀況，此時需要上層通訊協定的額外支援才行。

在接下來的內容，我們將針對 BSS Transition 情況，介紹整個換手的流程。

2.1.2 802.11 換手流程

根據 A. Mishra 等[3]的研究觀察，802.11 基地台換手過程主要可分為兩個階段：基地台搜尋(discovery)與基地台連結(commit)。基地台搜尋階段主要是執行 scanning 動作，而基地台連結階段則包含了 authentication 與 association 兩個動作，各個動作的說明如下：

a) Scanning：

在此階段，STA 主要目的是要掃瞄找出附近有哪些基地台可供使用，並且量測基地台的訊號強度。而在 802.11 中規範了兩種 scan mechanisms：passive scan 和 active scan。

在 passive scan 中，STA 並不主動發出任何 packet，而是在 channel 中等候一段時間，聆聽基地台週期性發出的 beacon 封包，根據所收集到的 beacon(s)，量測訊號並讀取出基地台的資訊。

而相對的在 active scan 之下，STA 就扮演比較積極的腳色。STA 會依序對每一個 channel 發送 broadcast probe request 封包，然後等待一段時間收集運作在此 channel 的基地台所回應的 probe response 封包，藉此得到基地台的資訊與訊號。

b) Authentication：

認證(authentication)的目的是為了確認 STA 身分的合法性，保證存取網路的使用者已獲得授權。認證是連結(association)基地台前的必要程序，唯有經過身分認證成功的使用者才准許使用網路。在 802.11 中，定義了兩種認證的機制：開放式系統認證(open system)與及共享金鑰式認證

(shared key)。

c) Association：

STA 完成與目標基地台的身份認證成功之後，接著與目標基地台進行連結，取得 AID (association identifier)後，就可以使用網路資源。

整個換手過程中 STA 與基地台之間的訊息交換流程如 Figure 2.3。

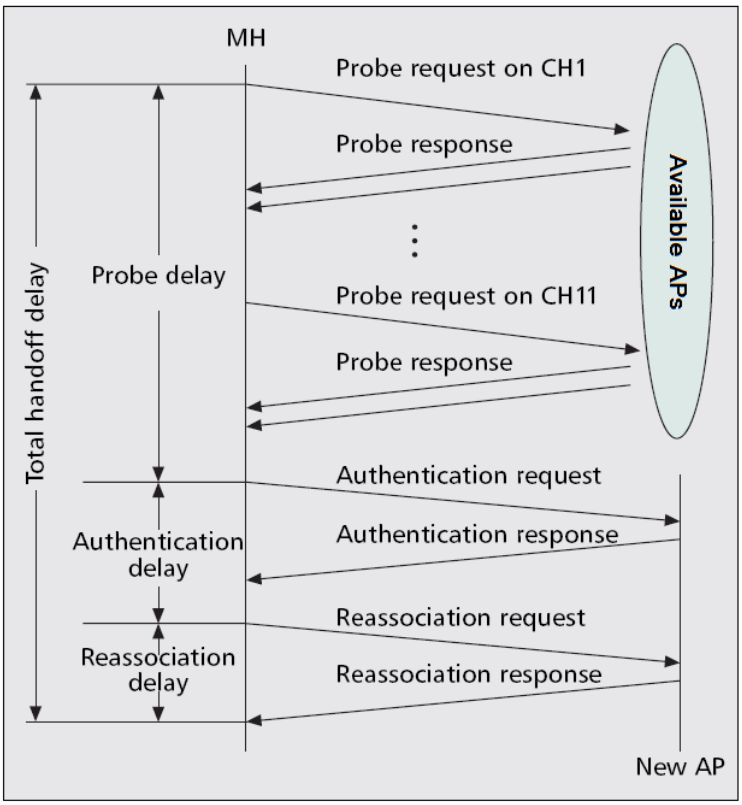

Figure 2.3: Handoff message flow in IEEE 802.11 networks.

(Adapted from [6])

Table 2.2: Latencies of handoff procedures.

(Adapted from [16])

Phase	Time
Scanning	350~500 msec
Authentication	< 10 msec
Association	< 10 msec

依據 A. Mishra 等[3]的實驗分析報告，將換手過程裡每一個階段所花的時間整理如 Table 2.2。從表中我們得知整個換手過程大概需要 370~520ms 的時間，這對於即

時性網路應用服務如 VoIP 所建議的延遲時間 50ms [20]而言，已經是 7~10 倍多的時間，會造成封包遺失與延遲等影響是不得不面對的課題；進一步我們更可以發現 scanning 動作佔據了整個換手過程的絕大部分約 90%時間，所以如果能有效的改善 scanning 所需要的時間，就能夠縮短整個換手所需的時間，降低對網路應用服務的影響。

2.2 Linux 下使用者空間和核心空間的訊息溝通機制

在 Linux 作業系統下，使用者執行的應用程式是跑在使用者空間，而核心程式與模組(包含驅動程式)則是在核心空間執行，彼此並不能直接存取資料結構或是記憶體位址。當應用程式想要呼叫驅動程式模組去執行某項動作或是取得下層裝置資料與狀態時，都需要透過 Linux 核心所提供的標準界面來進行，底下我們將簡單介紹 Linux 所提供的幾項常見機制。

ioctl system call :

Ioctl 是 input/output control 的意思，在 Linux 環境中, 裝置是被當作檔案一樣看待的，因此有基本的 file operations，如 open()、close()、read()、write()...等等，而在這些操作函式功能之外，不同的裝置存在著不同的操作功能，這些個別裝置專屬的指令動作，通常就是用 ioctl 來完成。

從使用者空間來看，ioctl 指令具有下列的原型：

int ioctl(int fd, unsigned long cmd, ...);

fd 代表著是一個檔案描述子，cmd 則是和裝置相關的指令編號(command code)，裝置驅動程式可以自行定義私有的指令編號，編號範圍從 SIOCDEVPRIVATE 到 SIOCDEVPRIVATE+15。最後則是一個選擇性參數，通常是指標 char *arg，指向要傳遞資料的記憶體位置。

應用程式可以透過 ioctl 傳送命令給驅動程式，而當上層程式呼叫 ioctl 時, 相對應的驅動程式裡會有一個 callback function 被呼叫起來回應，這個 callback function 專門處理 ioctl 指令，根據所傳送的不同指令碼，執行相對的程式動作。一些網路管理程式就是使用 ioctl 來控制網路，如 ifconfig、iwconfig、route、dhclient...等。

Netlink Socket：

Netlink socket 是一種特殊的 Linux 特有的 socket，是比較新且適合網路相關程式跟核心溝通的機制，定義在 RFC 3549 [21]。在 Linux 核心版本 2.6.14 之後，像是路由 daemon (NETLINK_ROUTE)、用戶 socket 協議(NETLINK_USERSOCK)、防火牆(NETLINK_FIREWALL)、ipsec 安全策略(NETLINK_XFRM)…等等，都是透過此機制來跟核心溝通。使用 netlink socket 的好處是他可以讓應用程式與核心進行雙向資料傳輸，並且應用程式只要使用標準的 socket API 就可以。

我們將 netlink socket 跟 ioctl 指令作比較，其特性不同處可以分為以下幾點：

1) Netlink 是一種非同步通信機制，而 ioctl 是同步通信機制。

2) Netlink 支援廣播功能，核心模組或應用程式可以把訊息廣播給某個 netlink 群組，屬於該群組的任何核心模組或應用程式都能接收到該訊息，核心事件往上層應用程式的通知機制就使用了這一特性，任何對核心事件感興趣的都能收到該子系統發送的事件。

3) 核心可以使用 netlink 主動傳送訊息通知上層應用程式，但是 ioctl 只能由上層應用程式主動呼叫。

4) Netlink 使用標準的 socket API，如 bind()、sendmsg()、recvmsg()…等，因此容易許多。ioctl 則需要特別注意使用，誤用指令編號或是傳遞錯誤參數記憶體位址都有可能會讓驅動程式執行錯誤，甚至造成當機。

Procfs

Procfs 代表著是 process file system，這是核心模擬出來的軟體檔案系統，通常掛載在/proc 目錄下，核心透過檔案的方式將核心內部資訊傳遞給使用者空間的程式，像是 cpuinfo、interrupts、meminfo…等。因為是以檔案的形式存在，因此我們可以一般檔案操作的方式來使用他，像是用 cat 或是 more 指令來讀取核心訊息，用導向符號“>”將訊息傳遞給核心。與一般檔案不一樣的地方是，在/proc 下的每一個檔案，都各自對應到核心內部的專屬函示，當應用程式或使用者讀取檔案時，核心才會即時的產生檔案內容。

Linux 中有相當多的程式都是利用/proc 檔案系統，例如 ps、top、w、vmstat…等等系統工具。

Sysfs

Linux 在核心版本 2.6 之後，導入了 sysfs 檔案系統，這是一個特殊的檔案系統，通常掛載在/sys 目錄下。與/proc 類似，sysfs 不僅可以像/proc 一樣允許使用者空間讀取核心的資料，同時 sysfs 是以更結構化的方式向使用者呈現資訊，提供一條管道讓使用者來改變某些作業參數。

透過 sysfs 檔案系統，核心把 kobject (kernel object)物件的階層關係與物件所擁有的屬性開放給使用者讀取，而驅動程式模組亦是一個 kobject 物件，所以也被對應到 sysfs 系統中，而模組參數則是作為物件屬性對應到 sysfs 下的一個文件。因此，經由 sysfs 系統，程式設計師在撰寫裝置驅動程式時就可以將一些跟裝置操作相關的參數輸出到 sysfs 系統下，讓使用者通過 sysfs 系統來讀取或設定更改這些參數，如此就可以達到在裝置進行中動態控制裝置行為。

根據 J. Corbet 等[22]的說法，針對 Linux 現在的發展趨勢走向，已經不再繼續鼓勵在/proc 下增加檔案，/proc 檔案系統的發展在核心開發團隊眼中，已經是相當混亂且失去控制的，不再符合當初的設置目的，因此對於新的驅動程式，應該是盡量使用 sysfs 系統來提供裝置的狀態資訊。

2.3 Linux Wireless Extension API

從使用者的角度來看，802.11 無線網路介面與一般的 Ethernet 網路介面並沒有什麼太大差異，只不過 802.11 有許多 Ethernet 所沒有的操作參數與特性。與其讓每一個驅動程式各自實現自己的設定工具，不如將所有共同的功能定義成一個 Wireless Extension API，讓驅動程式有一個統一的軟體介面來提供功能，這樣使用者便可以有一套一致的設定工具。Linux Wireless Extension [24]與 WPA Supplicant [25]就是這種構想下的產物。

Wireless Extension API 一開始是由 HP Labs 以開放原始碼的方式開發與維護，在 Linux 核心版本 2.2.14 之後，Wireless Extension API 被收入核心當中，因此現今使用的 Linux 核心都有支援提供此 API。使用者除了可以透過 Jean Tourrilhes 所實作出的 Wireless Tools [24]設定工具來操作使用 802.11 無線網路卡外，也可以自行依據 Wireless Extension API 的規範撰寫相關程式，透過 ioctl 指令，傳入相對應的指令編號與資料結構(這部分資料可參考 wireless.h 標頭檔)，如此就可以控制與設定無線網路卡。

第三章　相關研究

在以往的研究中，縮短換手延遲時間主要可分為兩個方向，第一個主要是想辦法縮短 scanning 所需的時間，減少基地台搜尋階段的時間，這部分的研究如[7]~[16]等，進一步的內容在本章節之後會介紹；另一個方向是朝降低基地台連結過程所需的時間，減少 authentication、association 過程訊息交換的延遲時間，像是[17]~[19]的研究就屬於此類。本篇論文主要探討的是降低基地台搜尋時間方法，因此接下來將對這部分的相關研究作簡介。

3.1　頻道掃瞄機制的時間分析

在 2.1.2 小節中，我們已經介紹過了 802.11 所提供的兩種掃瞄機制：passive scan 與 active scan，而這兩項機制除了行為不同之外，所需要花費的時間也不盡相同，因此採用什麼樣的掃瞄機制，也就會造成不同的延遲時間。

在 passive scan 機制中，主要依靠的是 STA 在每一個 channel 聆聽基地台週期所發出的 beacon 封包，讀取出基地台的資訊。STA 除了所連結的基地台外，在沒有其他資訊的輔助下，並不會知道其他基地台發出 beacon 的時間點，所以為了確保能收到 beacon，勢必至少要保守的等待一個 beacon interval 時間。假設以目前市面上基地台常見 beacon interval 的預設時間 100ms 來計算，STA 除了正連結中的基地台不需要外，要掃瞄完另外 10 個 channels 就需要花費 1 sec 的時間。

而另一方面對於 active scan，802.11 所定義的流程如下頁 Figure 3.1。其中與掃瞄花費時間最有關係的是(e)項動作，當中包含了兩個重要的時間：MinChannelTime 以及 MaxChannelTime。STA 在(c)動作送出 broadcast probe request 訊息之後，會重新啟動一個 ProbeTimer，接下來如果在 MinChannelTime 的時間內目標 channel 都沒有 busy 的情況發生，就表示目標 channel 是空的，沒有任何基地台存在，這時 STA 就結束這個 channel 的掃瞄動作，切換到下一個 channel 繼續掃瞄；相反的，如果目標 channel 有 busy 的情況發生，則表示目標 channel 有基地台存在著，因此要等待到 MaxChannelTime 時間結束，處理完收到的 probe responses 訊息之後，再進行下一個 channel 的掃瞄。我們可以將 active scan 流程所需要的時間用 T_{active_scan} 表示，N 代表要掃瞄的 channel 個數，歸納出如下面式子：

$$N \times MinChannelTime \le T_{active\_scan} \le N \times MaxChannelTime$$

由上述討論可以知道，active scan 花費的時間主要受制於 MinChannelTime 與 MaxChannelTime 參數的影響，在 802.11 標準中並沒有硬性規定這兩個時間的數值，廠商可以根據不同因素的考量自行調整，因此這部分不同 device 會有不同的結果，根據 P. J. Huang 等[13]指出，部分實作產品預設 MaxChannelTime 為 30ms。總結來說，一般而言研究人員大多數都認為，active scan 所需要花費的時間是比 passive scan 來的短，而市面上實作產品也大多是透過 active scan 機制來進行頻道掃瞄的工作。

For each channel to be scanned:

(a) Wait until the ProbeDelay time has expired or a PHYRxStart.indication has been received;

(b) Perform the Basic Access procedure (Distributed Coordination Function, DCF);

(c) Send a probe with the broadcast destination, SSID, and broadcast BSSID;

(d) Clear and start a ProbeTimer;

(e) If PHYCCA.indication (busy) has not been detected before the ProbeTimer reaches **MinChannelTime**,
then clear NAV and scan the next channel,
else when ProbeTimer reaches **MaxChannelTime**,
process all received probe responses;

(f) Clear NAV and scan the next channel.

Figure 3.1: Active scanning procedure.
(Source from [1])

3.2 基地台搜尋階段的改良機制

在改良基地台搜尋階段的方法中，直覺上想最直接方式可能是減少掃瞄的 channel 個數，如果 STA 可以有某種機制的輔助，掃瞄少數幾個 channels 就可以找到合適的基地台，如此也就達到縮短時間的目標。而依據 S. H. Pack 等[6]的研究，策略上將改良機制分為兩種方向，第一種就是減少所要掃瞄的 channel 個數(reduce the number of channels to be probed)，利用掃瞄少數 chnnels 的方法來取代以往掃瞄全部 11 個 channels 的方法。第二種則是透過數據分析與實驗找出最佳的 MinChannelTime 與 MaxChannelTime，或是經由額外資訊的幫助，讓 STA 不需要等到 MaxChannelTime 期滿，及早就能結束這次的掃瞄，開始往下一個 channel 執行，達到減少在 channel 中的等待時間(reduce channel waiting time)。接下來我們將根據這兩種策略分類介紹相關的研究文獻。

3.2.1 Reduce the number of channels to be probed

- Neighbor Graph scheme :

M. Shin 等[9]提出 NG (neighbor graph)的概念，提議建立有關基地台與基地台間的佈署關係以及相關資訊，藉由提供這樣的資訊給 STA，STA 可以知道目前所連結基地台附近存在有哪些其他基地台資訊，而這些鄰近基地台就是有可能換手的目標，有了這個資訊的輔助，STA 在進行 active scanning procedure 時，就只要掃瞄有目標基地台存在的 channel 即可，省略掃瞄無目標基地台存在的 channel。STA 進一步也可以從 NG 資訊中知曉某一個 channel 中有幾台基地台的存在，在發出 broadcast probe request 封包後，只要收到相同數量基地台所回應的 probe responses 之後，就可以結束此 channel 的掃瞄，不需要等到 MaxChannelTime 期滿，如此也達到了降低 channel waiting time 的目的。整個機制運作流程如 Figure 3.2 所示。

```
for all channel i where any neighbor AP is running do
    Broadcast probe request on channel i
    Start probe timer
    while True do
        Read probe responses
        if Medium is idle until MinChanTime expires then
            break
        else if all APs on channel i have replied then
            break
        else if MaxChanTime expires then
            break
        end if
    end while
end for
```

Figure 3.2: NG probing algorithm.

(Source from [9])

透過 NG 的機制，雖然需要額外付出管理基地台以及維護 NG relationship 資訊的費用，STA 也需要額外多儲存一份這樣的資訊；但根據其實驗分析結果顯示，適當的使用 NG 資訊來協助 active scanning procedure 的進行，可以比單純使用 Full Channel Scanning 所需花費的時間大幅縮減 80.7%，因此雖然有其他 overhead 需要付出，但只要控制得宜，應該是值得投資的。

NG-pruning scheme :

在 M. Shin 等[9]提出的 NG 機制後，根據觀察得到的心得，再提出了進階的改良方法，稱為 NG-pruning algorithm，所根據的資訊是基地台與基地台之間 Non-overlapping relationship 構成的 Non-overlap Graph (NOG)。所謂的 Non-overlapping 關係是指：基地台 A 和基地台 B 在 NOG 中有 edge 存在，表示在這兩台基地台間，STA 如果可以到達基地台 A 與之通訊，就確定其無法與基地台 B 通訊，反之亦然。透過 NOG 這一項進階資訊的幫助，STA 可以瞭解每一個 channel 中鄰近基地台的 non-overlapping 關係，依此可以更進一步來判斷縮短在 channel 中等待時間。我們以 Figure 3.3 為例簡單說明此機制的運作。

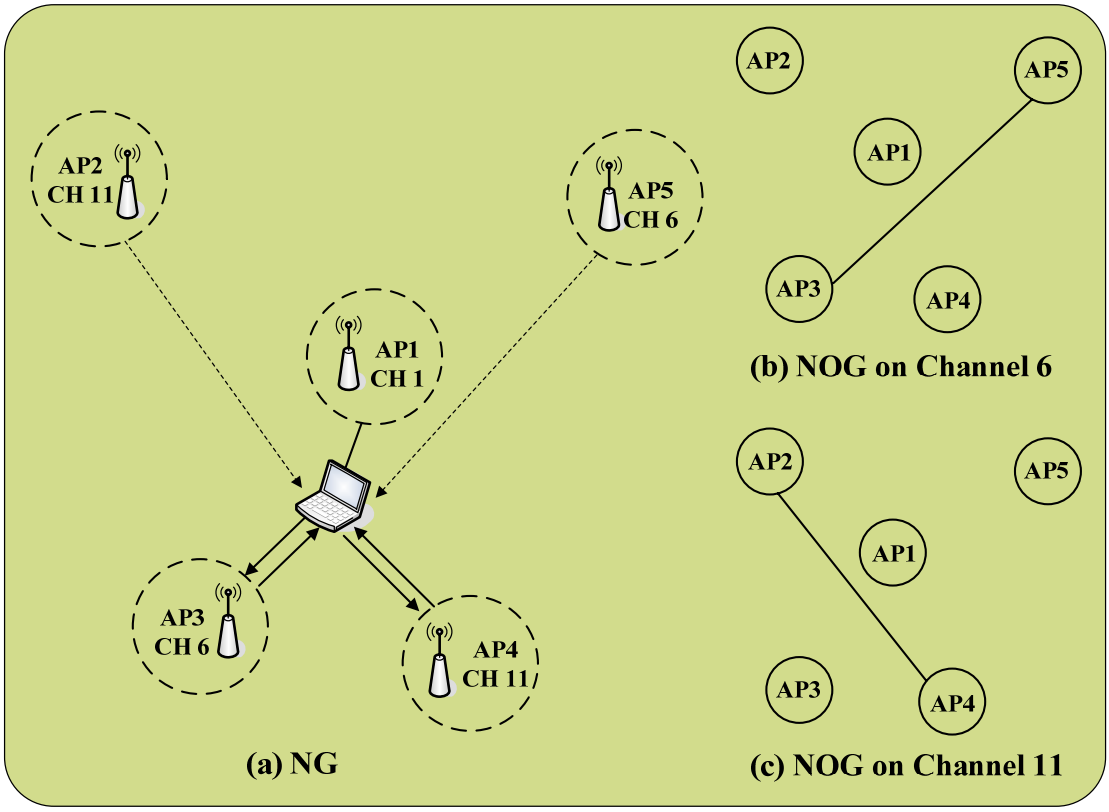

Figure 3.3: NG-pruning scheme.

(Adapted from [6])

在這個例子中，我們使用 1、6、11 三個 channels，STA 目前所連結的基地台 AP1 有四個 neighbors (AP2~AP5)。當使用之前所介紹的 NG probing algorithm 機制時，STA 只需要掃瞄 6、11 這兩個 channels，可以明顯發現要比 full scanning 來的少很多。而使用 NG-pruning 機制時，STA 先取得在 channel 6、11 的 NOG 關係，如同(b)(c)所表示。當 STA 首先掃瞄 channel 6 時，收到 AP3 所回應的 probe response 後，依據(b)的 non-overlapping 關係，STA 不會收到 AP5 的回應，因此 STA 不需要

在等待下去，可以立即結束 channel 6 的掃瞄；同樣的，接著進行 channel 11 的掃瞄，STA 在收到 AP4 所回應的 probe response 後，也不需要等待同個 channel 的另一台 neighbor AP2，也可以立即結束掃瞄。透過這樣的機制，就可以在進一步精簡縮短頻道掃瞄時間。

要提醒一下的是，Figure 3.3 的例子中 non-overlapping 關係都是在同個 channel 底下，但這並不代表不同 channel 基地台之間不能有 non-overlapping 關係，不同 channel 基地台間要是存在著 non-overlapping 關係，就可以達到減少需要掃瞄的 channel 個數。最後根據實驗分析結果，使用 NG-pruning scheme 可以比單純使用 Full Channel Scanning 所需花費的時間縮短 83.9%，比起 NG probing algorithm 則是多改善了 3.2%。

Selective Channel Scanning Using Neighbor Graph :

H. S. Kim 等[11]也是根據 NG 的概念，提出 selective scanning 的作法，讓 STA 透過 NG Server 去取得 graph 的資訊，省略不去掃瞄閒置無基地台存在的 channel，降低掃瞄所需要的時間。接著 H. S. Kim 等更進一步指出，利用 broadcast 傳送 probe request 時必須要等待 MinChannelTime 或是 MaxChannelTime 的時間，因為 STA 不知道會有幾台 AP 回應 response，如果透過 unicast 的方式來 probe channel 時，就不需要等待那麼長時間，因此最後提出 selective channel scanning with unicast scheme，整個基地台搜尋階段所花費的時間 T_{delay} 可以改為下列計算公式：

$$T_{delay} = N' \times rtt + \alpha$$

其中 N'為 graph 中所記錄可能的鄰近 AP 個數，rtt 是 unicast probe req/resp 來回加起來的 round trip time，α則是訊息處理的時間。最後經由實際測試，根據 sniffer 所攔截到的 probe req/resp 封包來衡量計算延遲時間，結果如 Table 3.1，可以發現搭配使用 unicast 的 selective scanning 機制所需的花費時間是最短的。

Table 3.1: Average probe delay of each method.

(Source from [11])

Neighbors	Method	delay [ms]
1	Basic active scanning	322
	Selective Scanning	55
	Selective Scanning with Unicast	12
2	Basic active scanning	322
	Selective Scanning	332
	Selective Scanning with Unicast	21
3	Basic active scanning	322
	Selective Scanning	145
	Selective Scanning with Unicast	30

3.2.2 Reduce channel waiting time

- Optimize Probe-Waiting time :

A. Mishra 等[3]進行一連串的實驗測量並且分析其結果，認為可以依據統計的結果選取出適當的 MinChannelTime 以及 MaxChannelTime 時間參數，讓一定比例之上的基地台都會在選取的時間內成功回傳 probe response。透過事先量測統計出來的 MinChannelTime 與 MaxChannelTime，可以在 channel waiting time 以及 probe loss 之間取得很好的平衡點。根據這篇報告的建議，MinChannelTime 設為 7ms，而 MaxChannelTime 設為 11ms，這樣就可以讓大部分的基地台都成功的回應 probe response。

- Intelligent Channel Scan :

K. Kwon and C. Lee [8]所提出的的 intelligent channel scan scheme，利用 802.11 標準中 DCF 機制的原理，讓 STA 偵測 AP 送出的 probe response 是否有 collision 的情況發生，以及利用 monitor data frame 擷取出 BSSID，用此來判斷有幾台 AP 存在此 channel 中，透過以上兩個動作智慧的判斷出該 channel 下的目標基地台是否已經全部偵測完畢，不需要等到 MaxChannelTime 結束即可以判斷出是否要切換到下一個 channel 去執行掃瞄了。

但此作法中有一個假設條件是 probe response 有較高的傳送優先權，即使基地台的 queue 中有其他封包要送，收到 probe request 後，基地台依然必須優先送出 probe response frame。而在 802.11 規範中，probe response 都是透過 DCF 利用 unicast 機制傳送，並沒有較高的優先權，因此這樣的假設勢必要修改基地台的功能，實務上的可行性並不高。另外 probe response 發生 collision 的情形是機率問題，並不全然會發生。而透過 monitor data frame 估算來有幾台 AP 存在，也是有可能會發生遺漏的情況。

3.2.3 Background scanning mechanism

在換手過程中基地台搜尋與基地台連結兩個階段，在 802.11 標準裡並沒有規定說這些過程一定要連在一起執行，尤其是前面基地台搜尋的階段，佔據了整個 90%時間，如果可以事先在換手之前處理的話，這樣換手過程就只要執行基地台連結的程序，STA 處於沒有連線狀態的時間將可以大幅降低。而將基地台搜尋階段提前處理，在原本正常傳輸的狀態下分散式的進行頻道掃瞄的動作，一般文獻上就簡稱為 background

scanning、preemptive AP discovery 或者 pre-scanning 機制，本篇論文名詞統一稱為 background scanning。

至於 background scanning 要如何進行，該採用 active scan 或者是 passive scan，以及如何有效的分散執行才不會影響到原本的資料傳輸品質，這些都是屬於開放式 issues，目前也有許多的研究提出利用此機制來提供快速換手。

P. J. Huang 等[13]基於 H. S. Kim 等[11]的方法，提出結合 background scanning 的機制，將 selective channel scanning with unicast scheme 在正常傳輸的狀態下分散式的進行，而為了避免進行掃瞄時造成原本傳輸中的資料封包遺失，P. J. Huang 等利用 802.11 標準中規範的 power-saving mode，在進行掃瞄前，通知基地台 STA 進入 power-saving mode，讓基地台幫忙 buffering 傳送給 STA 的封包，STA 趁此時進行頻道掃瞄動作，結束後再通知基地台 STA 返回 normal mode，繼續正常的資料傳輸。透過此機制在 power-saving mode 中執行選擇性掃瞄，再結合換手之前預先執行認證和經由加強 IAPP 而延伸出來的 forwarding-and-buffering 機制，有效的減少 90% 換手時間，並且不會有封包遺失。

第四章　具網路拓樸知覺的頻道掃描機制之設計

4.1　掃瞄機制之設計概念

在之前 3.2 章節裡，我們已經介紹了利用 NG 資訊來改善基地台搜尋階段的機制，在 NG 當中，基地台與基地台之間有 edge 關係所代表的含意可能是此兩台基地台的通訊範圍有互相涵蓋到，或是 STA 可以在這相鄰兩台基地台之間換手(可能非雙向)，透過此 NG 資訊可以讓 STA 減少需要掃瞄的 channel 個數，也可以降低 channel waiting time。然而考慮真實環境底下使用者移動的過程，其移動的方向並非完全是隨機(random)的方式，而是會受制於地形地物的影響；例如在校園中有建築物牆壁的影響、大樓建築物中受限於走道的設計，或是使用者大多是從某個地方移動到特定目的地(如教室)，都有可能造成說雖然在 NG 中有 edge 的關係，但是實際上並不會有換手的機會存在著。這樣的現象提示我們，是否在 NG 中的所有鄰近基地台都是換手時可能的目標基地台呢？是否真的需要去執行掃瞄所有的鄰近基地台呢？

進一步我們以「人」的角度來思考，以下頁 Figure 4.1 為例。當使用者往左邊方向移動時，如果無線網路卡可以自由控制，這時只需要下命令去執行掃瞄 AP2、AP3 即可，完全不須理會 AP4、AP5，縱使它們也都是屬於目前所連結基地台的鄰近基地台。這樣判斷所依據的條件是什麼，就是對於基地台部屬位置的瞭解，以及使用者移動方向的資訊。利用這兩項資訊，我們可以從 NG 中選取出部分基地台成為一個子集合，這個子集合包含了位於使用者移動方向的基地台，而這些基地台有較高的機率被選擇中成為最後的目標基地台，因此我們可以不需要掃瞄 NG 中全部的鄰近基地台，就可以選擇出換手的目標了。

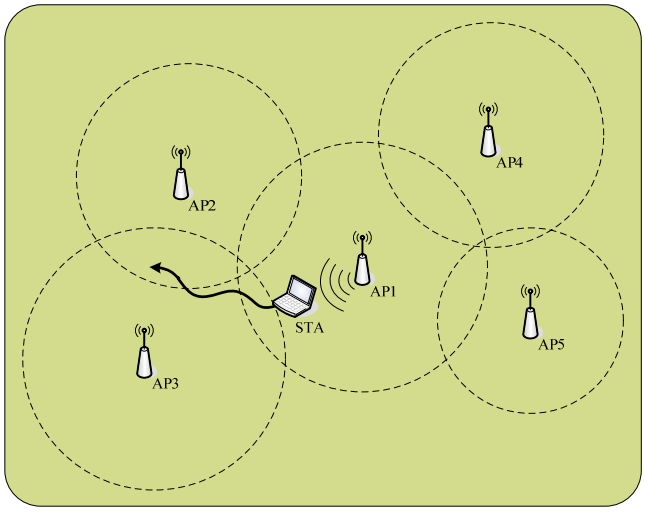

Figure 4.1: User movement case.

然而 STA 要如何知道這些資訊呢？針對基地台部屬位置彼此間相對關係資訊，我們可以擴充 NG 表格所記載的內容，加入真實網路拓樸的環境資訊，把鄰近基地台根據其位置方向予以區分，讓 STA 可以清楚地明白在各個方向有哪些基地台存在著。

而另外一個重點訊息，瞭解 STA 的移動方向，這項資訊可以透過定位(positioning)的技術來取得，藉由多次定位出來的點來計算移動方向。在室外環境 STA 可以透過 GPS 的輔助，但是對於室內環境而言，GPS 就無法有效的達成目的；而針對室內環境定位這項課題以往也有許多的研究，透過不同的技術如 Bluetooth [27]~[28], infrared [29], RFID [30]來達到定位的目的，但是這些技術都需要額外設備的輔助才行。另外也有部分研究是結合 WLAN [31]~[32]來達到定位的目的，但是其所付出的 overhead 往往過大，不適合拿來作為縮短基地台搜尋時間問題的解決方案。

我們重新釐清我們的需求，是否真的需要利用到定位技術來知道準確的移動方向呢？我們希望達到的是只需要經由簡單的判斷，不會造成消耗太多處理時間與其他 overhead 的解決方案，如此用在降低基地台搜尋時間才有優勢與價值。再之前我們已經把鄰近基地台的相對方位區別出來，所以應該就只需要知道 STA 是往哪一個方向的基地台靠近就可以了。而根據觀察可以得知，一般情況下靠近基地台時 STA 所接收到的 RSSI 訊號值會有漸漸遞增變好的現象，因此我們可以透過這樣的方式來判斷 STA 是往哪一個方向的基地台移動中；相反的也可以利用偵測到的 RSSI 是漸漸遞減中的，瞭解 STA 不是往這個方向移動，透過網路拓樸所記錄的基地台位置相對關係，去猜測出另外哪些方向的基地台是有較高的換手可能性。

因此本篇論文提出「具網路拓樸知覺的頻道掃描機制」，透過網路拓樸記錄的資訊輔助，在 background 事先利用 unicast probe mechanism 執行掃瞄特定鄰近基地台，接著利用所量測到的訊號衰減或增強變化幅度，搭配網路拓樸表所儲存的基地台部屬位置彼此間相對關係，幫助我們猜測使用者可能的移動方向，根據此結果來縮小所須的掃瞄範圍，盡快地選擇出換手目標基地台，達到降低基地台搜尋階段所需的時間。本章節接下來將進一步描述此機制所利用到的技術以及所提出的作法流程。

4.2 針對特定基地台訊號變化的量測

在提出的機制中，首先要解決的就是瞭解鄰近基地台訊號變化的情況，因為這是我們機制判斷的依據。在無線網路的環境底下，STA 可以透過持續監聽(monitoring) channel 的方式來取得基地台訊號的變化情形；監聽的方式則可以使用 802.11 標準所定義的掃瞄機制：passive scan 或 active scan。而如同先前 3.1 小節中所分析的結果，active scan 所需要花費的時間比 passive scan 來的短，大部分的網路卡實作上也都是

採用 active scan 為主，因此我們採用 active scan 機制去執行掃瞄 channel 來取得特定基地台的訊號 RSSI 值。

而在原本 802.11 的 active scan 機制中，是利用 broadcast 的方式去進行掃瞄，那是在 STA 並不知道有哪些基地台存在的情況下，因而盡可能保守一點的去蒐集到較多的資訊。但是我們已經有了網路拓樸資訊的輔助，知道鄰近有哪些基地台存在著，並且我們所須的只是要針對某一個基地台去詢問取得訊號強度，這時應該採用 unicast 的方式即可，不僅可以縮短掃瞄的時間，也減少了訊息傳輸的量。有了 unicast 掃瞄方法後，接著考慮要執行這個動作的時機問題，我們要在資料封包傳輸過程中 background 去進行掃瞄基地台動作，如果目標 channel 與目前連結中的基地台是相同的話也許還好，但是實際上基地台在部署時考慮到頻率重疊(frequency overlapping)的問題，相鄰的基地台 channel 多會是設定成不同，因此很多時候我們是需要去進行不同 channel 的掃瞄，這時切換到另外一個 channel 去執行掃瞄，就有可能會造成封包遺失的情形發生，所以我們會需要利用 802.11 提供的省電模式機制來避免這種情況的發生。底下將介紹實際 unicast probe 與省電模式機制的運作方式。

Unicast Probe Function :

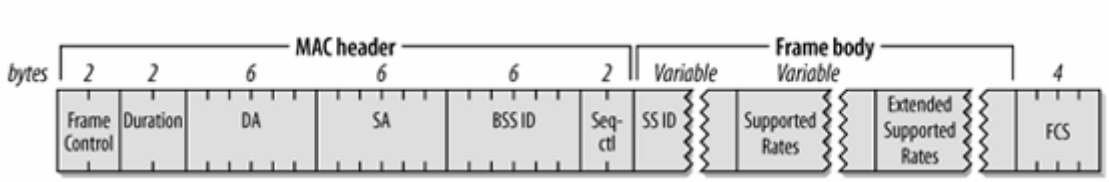

Figure 4.2: 802.11 probe request frame.

(Source from [33])

如 Figure 4.2 所示的 probe request frame format，DA 欄位指的是接收者的 MAC address，802.11 預設 active scan 是填入 broadcast address。而 BSSID 欄位代表著 BSS 的代號，通常是填入基地台的 MAC address，預設依然是填入 broadcast address，代表不指定特別的接收基地台。SSID 欄位則是可供 STA 指定目標基地台搜尋的無線網路代號，預設通常是使用 Null SSID 以表示不限制。而當要使用 unicast probe 機制時，我們必須修改 active scan 送出的封包，使其 DA 與 BSSID 欄位填入鎖定的基地台 MAC address，而 SSID 一樣填入鎖定的基地台 SSID。

而根據 802.11 的傳送機制規定，收到 unicast frame 後，都必須在 SIFS 時間後回應 Ack，因此 STA 送出 unicast probe request frame 給基地台後，基地台會先回應 Ack，接著一樣也透過 unicast 回應 probe response frame，STA 最後回應 Ack 作為

此次動作的結尾。整個流程就如同 Figure 4.3 所顯示的。

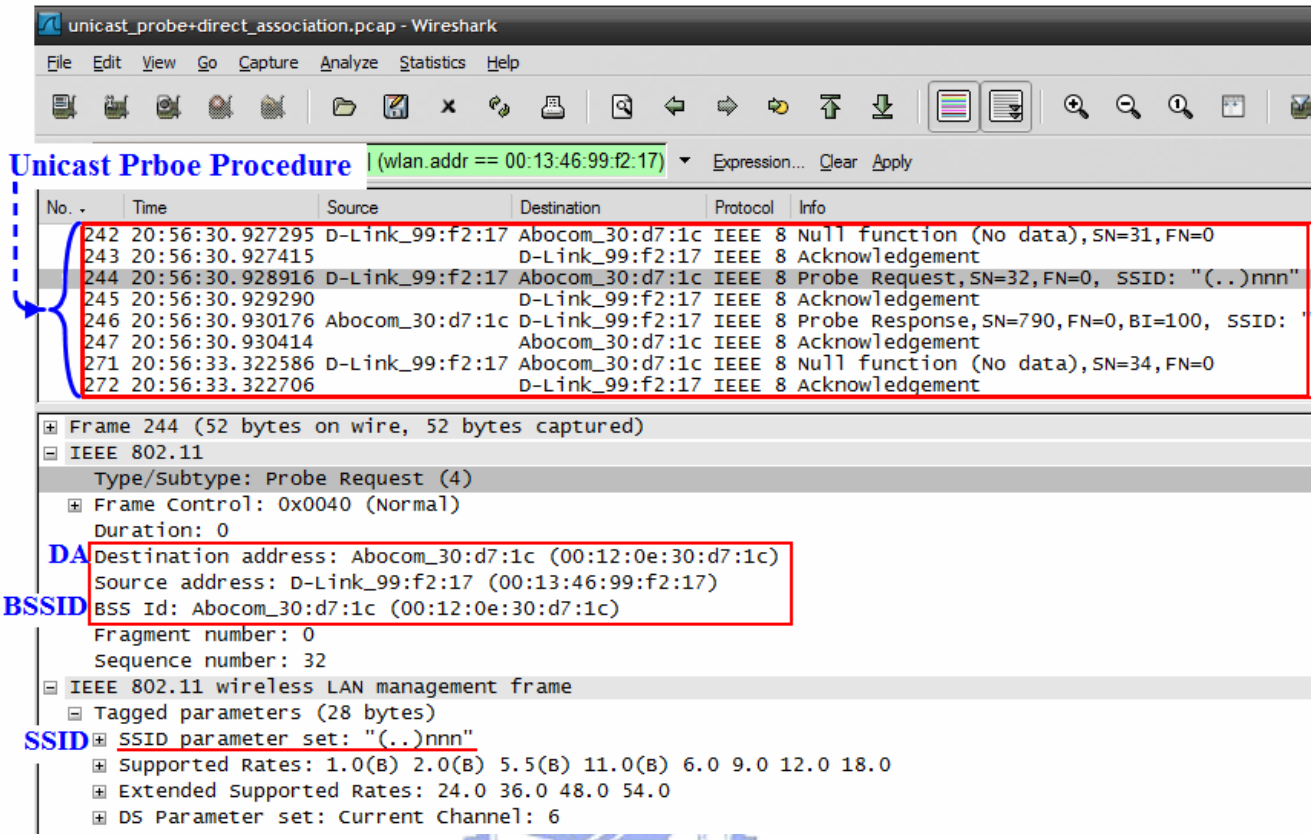

Figure 4.3: Unicast probe message flow (captured by wireshark).

Power-Saving Mode :

一般的可攜式裝置，在沒有接上其他電源供應來源的情形下，其電源主要來自其內建的電池設備，所以裝置可以使用的時數就會因為電池電力消耗的速度而有不同，802.11 針對電力的管理提供了一個 power-saving mode 機制，可以讓 STA 動態的依據自己通訊或電力消耗情況，適當的調整切換其 power management state 為 power-saving mode 或是 active mode，來達到節省裝置電力消耗的目的。一般而言進入 power-saving mode 的 STA，不會傳輸接收一般資料封包，而當有其他人想與之通訊時，所連結中的基地台會幫其 buffering，並等到 STA 醒來回到 active mode 之後，再將封包傳送給 STA。這機制的原始設計目的是為了節省不必要的電力消耗考量，然而也提供了一個機會讓 STA 可以趁著睡眠狀態時去做一些其他事情，而部分研究就是將這樣的機會利用在降低 background scanning 時所造成的封包遺失影響。

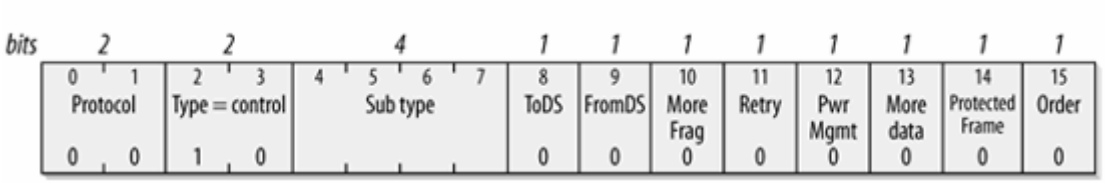

Figure 4.4: Power management bit in Frame Control field.

如 Figure 4.4 所示，Power management state 是依據位於 802.11 定義的 Frame Control field 中第 12 bits 來決定，當其設定為 1 時，代表著 STA 即將進入 power-saving mode，反之設定為 0 則是回到 active mode。STA 可以透過傳輸中的資料封包或是額外多發送一個 Null Data 封包來夾帶這項訊息去通知基地台，接著再切換到其他 channel 去執行掃瞄的動作，如此就可以避免封包遺失的影響。

4.3 網路拓樸的建立

在 NG 的作法中，一般主要是記錄鄰近基地台的三個資訊，分別為 SSID、BSSID 以及運作的 channel。而在我們的機制中，我們擴充 NG 表格所記載的內容，把鄰近基地台根據其位置方位予以區分開來，讓 STA 可以清楚地明白在各個方向有哪些基地台存在著。觀察我們所針對的大樓建築物-交通大學工程三館的環境，其走廊是呈現ㄇ字形，因此建置在走廊上的基地台，彼此間 link 主要是直線的關係，另外可能包含了轉角以及位於不同樓層的情況。

根據直線的關係，我們可以先將鄰近基地台區分為左右兩個方向，這時所記錄的 network topology table 就如同 Table 4.1。Neighbors 的欄位被劃分為 left、right 兩欄，分別記錄著這兩個方向可能的下一個基地台，代表著 STA 可由目前的基地台換手過去。

Table 4.1: A temporary prototype of network topology table.

Index	SSID	Channel	BSSID	Index of Neighbors	
				left	right
1	WL1	6	000F3DE1037E	2	
2	WL1	1	000F3DF73765	3	1
3	WL1	11	000F3DEC1D4F	4	2
4	WL1	1	0060B3166890		3

接著我們分析跨樓層的情境，當使用者經由樓梯在不同樓層移動時，上下樓層最靠近樓梯口的那兩台基地台，就會有 STA 在這之間換手的情況發生，因此在 network topology table 中是屬於鄰近基地台關係，然而，其訊號變化的模式卻是與一般平面環境不一樣。在平面環境下，STA 從目前連結的基地台移動到新的目標基地台，這兩台基地台訊號的變化模式如 Figure 4.5，可以發現舊基地台的訊號會漸漸遞減，而新基地台則是會漸漸地增，因此我們可以利用這樣的變化趨勢來評估 STA 往那個方向移動。

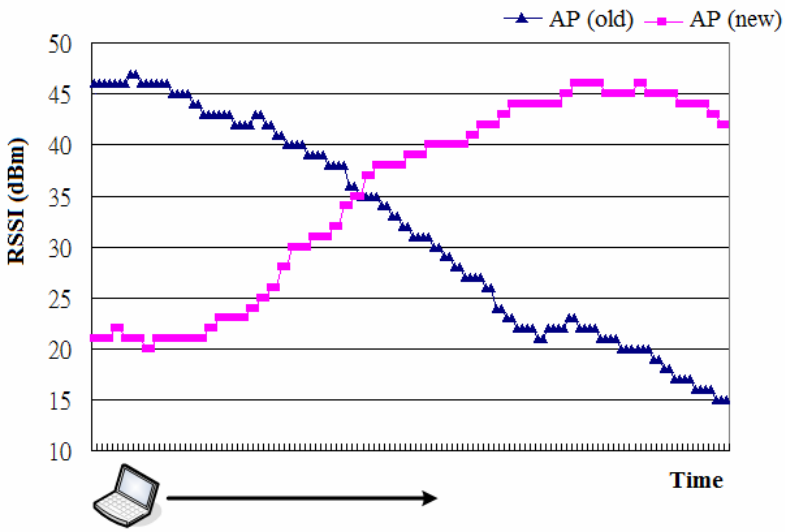

Figure 4.5: RSSI change between old/new APs.

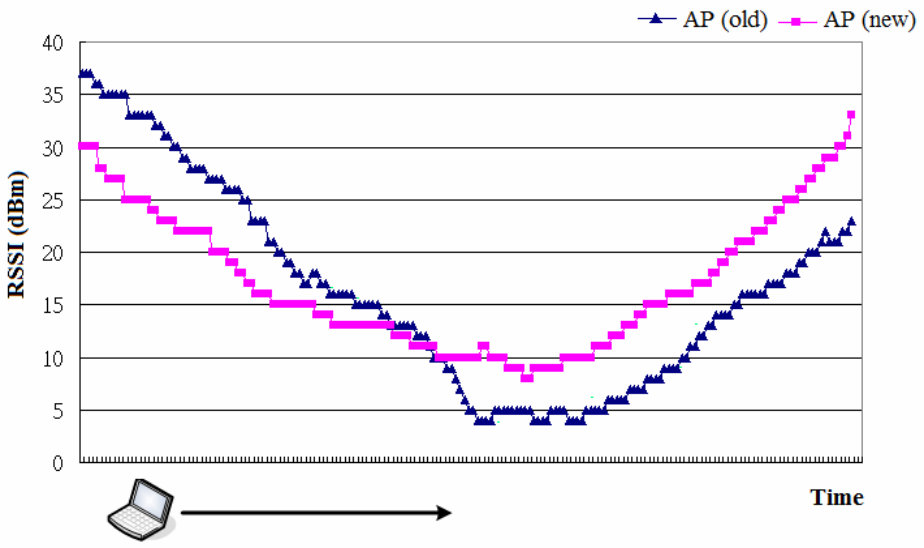

Figure 4.6: RSSI change between old/new APs with inter-floor movement case.

但是在跨樓層的情況下，STA 從目前連結的基地台移動到樓下新的基地台，訊號的變化模式是如 Figure 4.6 的結果。我們可以發現在下樓梯的過程中，兩台基地台訊號都是呈現先遞減然後再遞增的現象，中間交會的部分大約就是 STA 下樓梯走一半到達

樓梯中間的位置。這原因是因為真實世界是立體 3D 的環境，STA 進入樓梯的時候，其實對基地台來講都是在遠離中的狀態，相對的從樓梯中回到樓層的時候也都是處於接近中的狀態，加上基地台雖然與 STA 間隔了一個樓層，但依然還是可以接收到訊號，所以會呈現出這樣的變化趨勢。因此我們針對這樣的鄰近基地台關係，在 network topology table 中增加 Inter-Floor Neighbors 欄位來記錄上下樓層最靠近樓梯口的鄰近基地台關係。

再來我們觀察轉角的情況；當使用者在經過ㄇ字形轉角的位置時，會發現最靠近轉角的兩台相鄰基地台其訊號強度會變化的較劇烈，經常是使用者轉過方向之後，訊號遞增或遞減的速度比一般直線情況來的快，這是因為訊號強度受到地形影響的結果。

而我們所提的掃瞄機制，是屬於 threshold-based handoff algorithm，也就是當 STA 接收到連結基地台的訊號強度低於某個 threshold 值之後，才會進行預定的 background scanning 機制去找尋新的目標基地台，並不是永遠都在背景執行著。針對上面所描述轉角造成的現象，我們希望能夠提前一點動作，在訊號還沒很快的遞減之前，早一步先去執行掃瞄轉角過後的目標基地台，如此在 STA 經過轉角後，便可以很快的連上新的目標基地台，讓 STA 可以一直保持在較好的訊號強度環境底下，因此這時候我們可能會針對這樣的情況多定義一組 threshold 值。進一步我們根據實際測量的結果觀察到，事實上基地台所處的位置彼此間距離都不盡然相同，當使用相同的 threshold 值來判斷時，有時會變成太晚進行掃瞄而慢一步才換手，造成有一段時間 STA 有另外較好訊號的基地台可選擇卻無法去使用；或是因為太早執行掃瞄機制而會產生多餘的掃瞄次數。因此我們認為考量到真實網路拓樸的建置情形，每一個基地台應該有屬於它自己的 threshold 值，代表在這個基地台管轄下訊號的參考值，當 STA 收到的訊號低於此值時，就是需要去進行 handoff 事先準備工作的時候了。

Table 4.2: An example of network topology table.

Index	SSID	Channel	BSSID	Index of Neighbors		Inter-Floor Neighbors		Threshold (dBm)
				left	right	up	down	
1	WL1	6	000F3DE1037E	2				35
2	WL1	1	000F3DF73765	3	1			30
3	WL1	11	000F3DEC1D4F	4	2		5	35
4	WL1	1	0060B3166890		3			30
5	WL1	6	0060B3166863			3		25

總結以上說明，我們依據地形地物環境的影響，建置出如 Table 4.2 的 network topology table，將原本 NG 概念下單純的邏輯性 link 關係(有或沒有，能 handoff 過去或不行)轉變成能夠反應網路拓樸的 link 關係，並依據其相對方向與之區分；另外也根據每台基地台的環境去決定不同的 threshold 值，根據此值來決定何時 STA 要進行 background scanning 機制去搜尋新的基地台。

4.4 頻道掃描機制之運作

當 STA 利用頻道掃瞄機制去搜尋目標基地台時，是執行 Searching_Target_AP Function，當中主要是利用到前面 4.2 小節介紹的基地台訊號變化的量測方法，以及 4.3 所建置出來的 network topology table。我們分上下兩部分來說明這個 Function 的運作。

Searching_Target_AP FUNCTION

```
1)  Set Target_AP as Null;
2)  Select AP from one direction;
3)
4)  while loop is smaller than Max_Retry do
5)      loop add by 1;
6)      Measure selected AP signal variation;
7)      if signal is increasing then
8)              Selected AP as Target_AP;
9)              break
10)     else
11)             if no AP exist in opposite direction then
12)                     continue
13)             else
14)                     Select AP from opposite direction;
15)                     continue
16)             end if
17)     end if
18) end while
19)
```

一開始我們都是先假設使用者是在同一樓層當中移動，因此會先從左右兩邊的鄰近基地台開始去詢問。首先我們先選擇某一邊方向當作起始，這部分可以是隨機選取或是固定從某一邊開始，接著透過 background unicast probe 去量測並計算出其訊號的變化，而當結果是遞增的情況時，即可以判斷此鄰近基地台為我們所要找的目標基地台；相反的當結果是遞減時，我們會嘗試看看有沒有相反方向的鄰近基地台可以選擇，如果有則選擇之，並一樣進行訊號量測的動作。然而因為無線網路的訊號變化有時是不

太穩定的，所以可能會發生誤判的情況，當經過一輪兩邊方向的基地台都量測過但是卻都沒有出現遞增的現象時，我們會嘗試多確認幾次，迴圈重複執行的次數是用 Max_Retry 數值予以限制，而當重複次數大於所允許的次數，結果還是找不到適合的目標基地台時，這是我們就進入 Function 的下半部分來進一步判斷這時是什麼情況。

```
20)  if loop is equal to Max_Retry and Target_AP is Null then
21)      if Inter-floor neighbor AP exist then
22)              Monitor current AP until signal is increasing;
23)              Measure signal of inter-floor neighbor APs;
24)              Compare current AP with inter-floor neighbor APs;
25)              Select one AP whose signal is the best;
26)              if selected AP is current AP then
27)                      goto Monitoring_Current_AP state
28)              else
29)                      Selected AP as Target_AP;
30)              end if
31)      else
32)              Return FAILURE;
33)      end if
34)  end if
35)
36)  Return SUCCESS and Target_AP;
37)
```

在經過 Function 上半部分後還是找不到目標基地台，這時可能是使用者移動超出我們所知曉的，來到的位置其附近基地台訊號都不好，在這種情況下就回傳 FAILURE 資訊。另一個比較有可能的情況是使用者不在同一樓層中移動了，而是進入樓梯準備到另外一個樓層去，此時我們就得參考 network topology table 所記錄的 Inter-Floor Neighbors 資訊了。

在偵測跨樓層基地台的部分，根據先前 4.3 小節 Figure 4.6 的觀察結果，在到達樓梯中間之前，目前連線中的與下一個目標基地台都是呈現遞減的趨勢，所以我們的作法一開始會先避開此區段，而是等到目前連線基地台重新出現遞增現象，這時才去測量跨樓層鄰近基地台的訊號強度，接著再與目前連線基地台的訊號作比較，從中選一個訊號最好的當作候選基地台。這時如果選中的結果一樣是目前連線基地台，代表著 MN 走回原本的樓層，遇到此種情況我們就結束搜尋 Function 的動作，轉換 MN 狀態為 Monitoring_Current_AP 狀態去做相對應處理；否則的話此選中的鄰近基地台就為我們所要找的目標基地台，因此即可結束此 Function 動作，回傳成功及候選基地台資訊。

4.5 STA 整體運作流程

經過之前 4.4 小節所介紹的頻道掃瞄機制，STA 加入此機制後的整體運作流程，我們用 Figure 4.7 狀態轉換圖來說明。

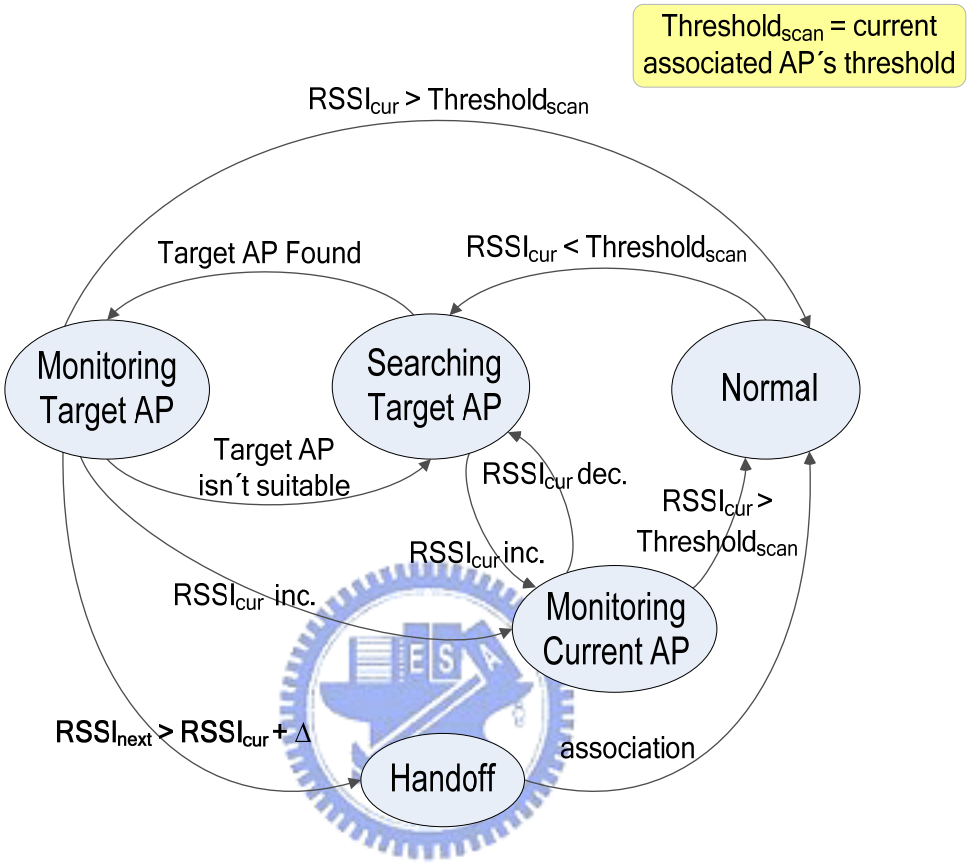

Figure 4.7: State transition diagram of proposed mechanism.

STA 一開始連線上基地台後是處於 Normal 狀態，進行正常的資料傳輸，此時我們的機制並不會作任何動作，而當所接收到的訊號小於 Threshold$_{scan}$ 時，才會進行 Searching Target AP 去搜尋目標基地台。而找到目標基地台後，STA 會進入 Monitoring Target AP 狀態，這時會週期的去維護確認所找到目標基地台是正確的，直到其訊號比目前連線的基地台來的好，並且超過Δ值(避免 ping-pong effect)後，就會驅動 STA 去進行 direct association 動作換手到目標基地台。

而在 Monitoring Target AP 狀態中要是發現目標基地台不適合，STA 就會回到 Searching Target AP 狀態去重新進行搜尋，而所謂不適合的條件，指的是目標基地台訊號出現遞減的現象。另外要是目前連線基地台出現遞增的現象時，並且比目標基地台訊號還要好，代表著使用者可能往回走了，此時我們會來到 Monitoring Current AP 狀態，等待其他的變化出現再去作動作。

最後，在 Searching Target AP 以及 Monitoring Target AP 執行的過程中，要是發現目前連結的訊號持續遞增變好並大於 $Threshold_{scan}$ 時，便會中斷 Function 的執行，因為此時代表著 STA 已經不再遠離原本的連結基地台，所以換手的事先準備工作也就可以先停止了。

第五章　具網路拓樸知覺的頻道掃描機制之實作

本節當中將介紹我們如何實作第四章所提及的頻道掃瞄機制軟體元件。首先在 5.1 會說明實作上所使用的軟硬體設備，5.2 介紹實作環境的網路拓撲，接著 5.3 說明實作的系統架構。

5.1　實作軟硬體環境

- 硬體環境：

我們將所提出的機制實作在行動端點上，行動端點為一台筆記型電腦，搭配一張 PCMCIA 介面 Atheros chipset 的無線網路卡。詳細配備如下表：

Table 5.1: Mobile Station hardware equipment table.

Category	Specification
PC	Acer TravelMate 3010 Notebook
Processor	Intel Core™ Duo mobile T2300 (1.66GHz)
Memory	Samsung DDR2 1GB
HDD	Seagate Serial ATA 5400rpm 100GB
WiFi	D-Link DWL-G630 802.11g 11/54Mbps

- 軟體開發環境：

因為我們需要修改 WLAN card 的 driver，實作網路卡本身沒提供的機制，所以我們選擇開放原始碼的 Linux 作業系統，搭配 Atheros 在 Linux 底下專用的驅動程式 madwifi driver[34]。開發與測試過程使用到的詳細軟體規格如下表：

Table 5.2: Software development environment.

Category	Specification
Operating System	Linux Fedora Core 5 with kernel 2.6.20
WiFi driver	MADWiFi 0.9.3
Development tool	GCC 4.1.1
Sniffer software	Wireshark 0.99.5
Packet generator	BillSniff 2.0

5.2 網路環境

本篇論文所針對的是校園大樓建築，因此我們的實作環境就是以交大工程三館為主，利用其所部署的無線網路環境作為實驗的平台。在實驗的過程中，我們主要是以工程三館五、六樓層的南半邊當作 STA 移動的範圍，這兩樓層的平面圖及基地台部署的相關位置，其簡單繪製的示意圖如 Figure 5.1、5.2。

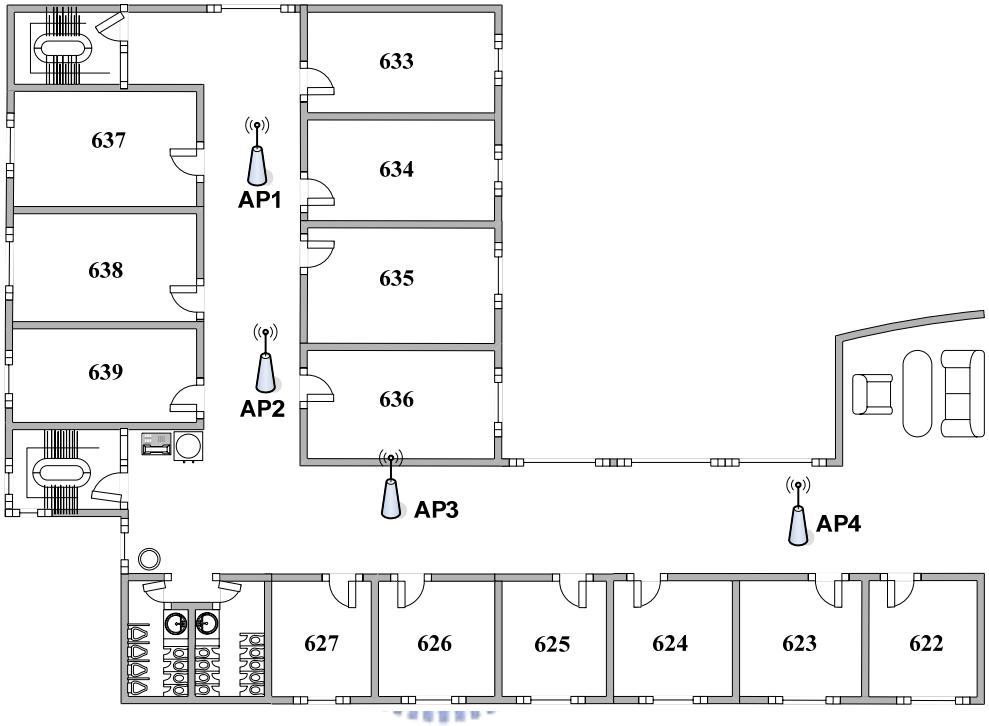

Figure 5.1: The southern half layout of the 6th floor.

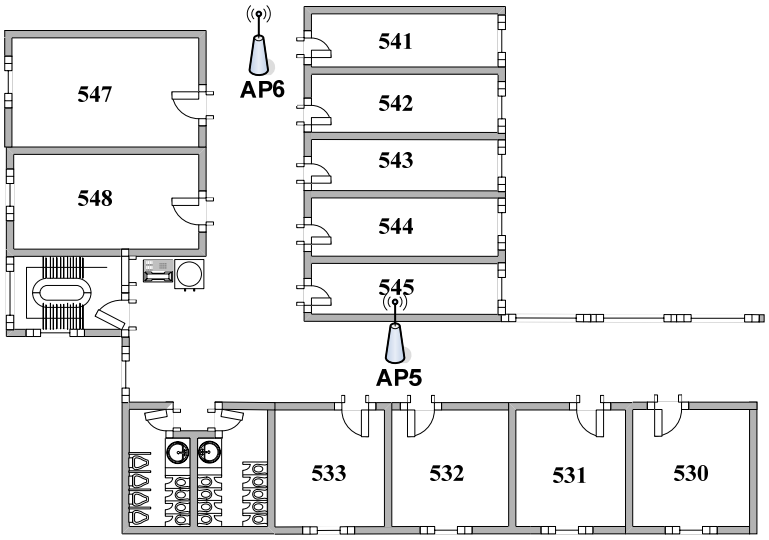

Figure 5.2: The southern half layout of the 5th floor.

根據這樣的環境，我們建立出的網路拓樸中，同一樓層裡相鄰的基地台彼此之間都是鄰居，而靠近樓梯口的基地台則是另外會有 inter-floor neighbor 存在，另外這些基地台其 SSID 都是相同的"WL1"，並且也都是屬於 140.113.24.0/24 子網域中。

5.3 系統架構

我們實作出的系統架構如 Figure 5.3 所示，左邊淺綠色無陰影的五個元件，從下方實體網路裝置開始，到 device driver、Linux Networking Stack，以及位在 user level 的網路應用程式，這是 Linux 作業系統下簡單的網路子系統示意圖。而我們在原本的架構中加入了兩個元件來完成我們提出的機制，一個是 Driver Extend Entity，另一個則是位在 user level 的 TopologyScan Management 控制程式，分別是 Figure 5.3 圖中深藍色有陰影的兩個元件，底下我們將進一步介紹這兩元件。

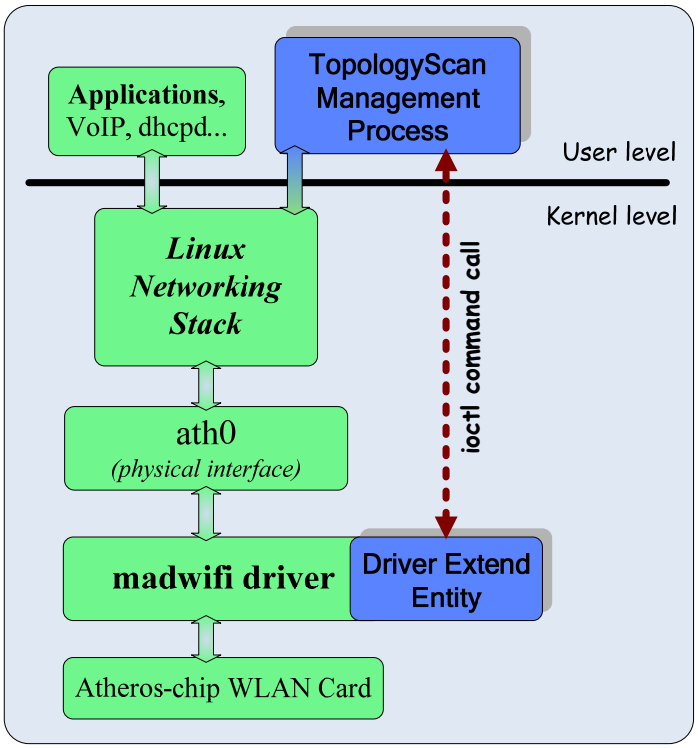

Figure 5.3: System architecture of proposed scheme implementation.

5.3.1 Madwifi Driver Extend Entity

Madwifi Driver Extend Entity 最主要的目的，就是擴充無線網路卡的功能，加入原本 driver 沒有支援的 unicast probe 以及 direct association 機制，並且註冊 ioctl callback function 來等待上層管理程式的呼叫；另外當無線網路卡狀態改變時，會透過之前 2.2 章節介紹的 Netlink Socket 去通知上層管理程式做處理。

5.3.2 TopologyScan Management Process

我們將提出的機制實作成 TopologyScan Management 程式，運作在 user level 中，經由這隻程式來控制無線網路卡的行為，透過 ioctl 指令去呼叫 driver 執行掃瞄、連結等動作，並且讀取回傳的結果。而 network topology information 則是儲存在另外一個獨立的檔案中，TopologyScan 在開始執行時會先進行讀取並根據此檔案資訊建立出 table，之後的動作就是依據此 table 來進行頻道掃瞄與換手的判斷。

第六章　效能分析與結果

本章節將對於我們所設計與實作的機制進行實驗與結果評估。6.1 節中我們會先對於所使用的 madwifi driver 進行分析，針對單一動作所需要花費的時間作量測。在 6.2 則是運行我們的頻道掃瞄機制並觀察其對資料封包傳輸所造成的影響。

6.1　Madwifi driver 行為之時間分析

無線網路卡因為使用的 chipset、driver 不同，其功能不盡相同，也會造成不一樣的表現結果，因為我們是選擇使用 Atheros chip 搭配 madwifi driver 來實作，所以我們先對於其效能的表現進行瞭解。我們透過修改 driver 加入偵測點輸出時間來觀察，從程式執行的角度來瞭解在執行掃瞄、連結時所需要的時間。

Probe Channel Cost :

當我們在執行 unicast probe 去執行掃瞄一個基地台時，其所需花費的時間可以列出如下面公式：

2*Set_Power_Saving + 2*Channel_Switch + Unicast_Probe

其中 Set_Power_Saving 為送出 Null data packet 給基地台，告知基地台說 STA 將近入睡眠狀態或醒來回到正常模式，在進入掃瞄動作前跟結束後各需要作一次；而 Channel_Switch 時間是 STA 切換所在 channel 需要的時間，最後的 Unicast_Probe 則是進行 4.2 小節所介紹的 unicast probe procedure 需要的時間。

式子中 Channel_Switch 時間是受到硬體裝置本身能力的限制，根據我們量測的結果需要 3.5~4ms；而設定 Power_Saving_Mode 需要 0.5ms 的時間，至於 unicast probe procedure 所需時間則是需要 2~3ms。因此最後我們可以得知，STA 利用 unicast probe 去執行掃瞄一個基地台大約是需要 10~12ms 的時間。

Channel Waiting Time :

如 3.1 小節所介紹的，802.11 定義了 max/min channel time 來決定 STA 在執行 active scan procedure 時需要在 channel 等待的時間。而在 madwifi 裡，針對 active scan procedure 一樣定義了兩個時間，稱為 max/min dwell time，預設值 max 為

200ms，min 則是為 20ms，然而其行為模式卻跟 802.11 標準所規範的不盡相同。

Madwifi 在執行 active scan procedure 時，當其收到 probe response(s)或 beacon(s)後，會確認 channel 等待時間是否已經超過 min dwell time，如果已超過則會在處理完收到的回應後就立即結束此 channel 的動作，切換到下一個 channel 去進行；如果沒有超過則會等到 min 時間期滿，之後也立即切換到下個 channel。而萬一沒收到任何 probe response 或 beacon，STA 則會一直等到 max dwell time 期滿才結束此 channel 的動作。跟 802.11 標準比較起來，在基地台部署密集的環境下，madwifi 所需要花費的時間會比 802.11 標準定義的來的短；相反的在只有少數基地台的環境下，madwifi 所需要花費的時間會比 802.11 標準的來的長。

而 Madwifi 如此的作法對我們機制有一點小幫助，因為我們採用 unicast probe 作法時，希望能夠在收到 probe response 後就馬上結束，不需要等到 802.11 標準所定義的 max channel time 結束。因此在 madwifi 底下，我們只需要將 max/min dwell time 的值都更改設定成很小，如此在收到 response 的情況下就不須再等待即可結束；萬一是沒收到回應的情形，STA 也能很快的在 max dwell time 期滿後就結束。

Direct Association：

當 STA 選定目標基地台後，決定要換手時，會直接執行 direct association 動作，不需要在 probe 一次，而 direct association 的過程包含了 authentication 與 association 兩個動作，我們從程式執行的流程去觀察，要完成此項動作大概需要 4~6ms 的時間。

6.2 封包傳輸 jitter 的影響

為了瞭解我們的機制對於正常資料傳輸的影響，我們模擬 VoIP 即時性服務的情境，利用一台筆記型電腦擔任 Correspondent Node (CN)角色，透過有線乙太網路連接上 140.113.24.0/24 子網域，並且透過封包產生器傳送 512Bytes 的 UDP 封包給 MN，週期時間為 20ms；而 MN 端則執行 sniffer 軟體，攔截所接收的 UDP traffic，根據其到達終點端的間隔時間(Inter-Arrival Time, IAT)資訊，進行下列的分析。

Probe、Handoff 動作的影響

首先我們先觀察 MN 在進行 background probe 與 handoff 過程中，對於資料封

包傳輸的影響。結果如 Figure 6.1 所示，MN 在 packet sequence no.90 ~ no.91 這段時間進行了 handoff 動作，花費了額外 20ms 的時間，因此造成 IAT 變成 40ms。而在這之前，MN 從 packet no.15 開始進行了數次的 probe 動作，根據圖表可以觀察得知對於封包 IAT 會有些微 5~10ms 的 jitter 效應。這樣的影響對於即時性應用服務來說，都應該還屬於可接受的範圍。

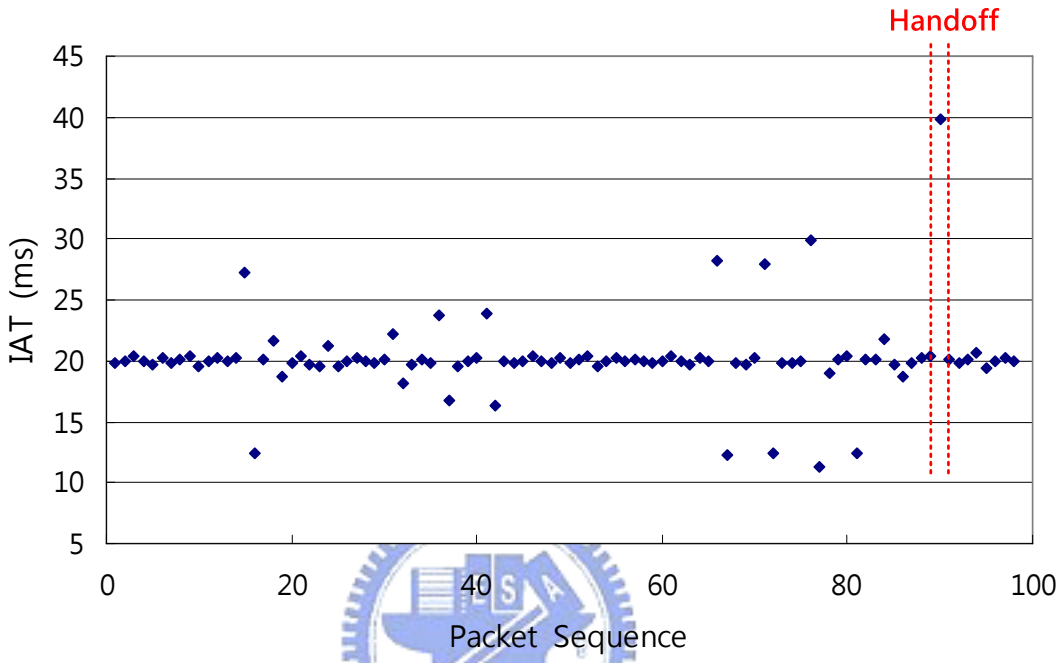

Figure 6.1: IAT of packets during probe and handoff.

不同情境 roaming 影響情況

我們利用兩種案例：同樓層中移動與不同樓層間移動，來實際測試整個移動過程中會有多大的 overhead 產生，以累積分佈函數的方式製成圖表來說明之。

Figure 6.2 是 MN 在同樓層中移動的情況，其間總共經歷了 4 次換手過程，經由圖表可以發現我們機制對於資料的傳輸只會有些微的 jitter 影響，統計資料顯示 IAT 為 20ms 左右大約是 96.70%。這些 jitter 影響一部份有可能是來自於無線網路本身傳輸時的不穩定，另外主要的是我們機制在進行 background scanning 時，會暫時切換到 power saving mode，去執行掃瞄目標基地台訊號，而這時有可能會跟正常資料傳輸的時間相撞，因此產生了這樣的影響。

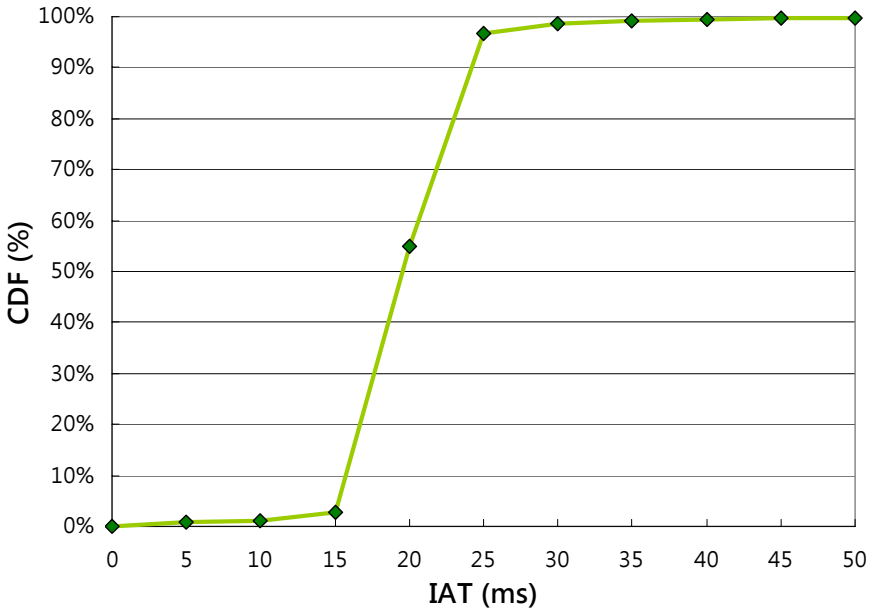

Figure 6.2: Cumulative distribution function of packets with IAT=20ms.

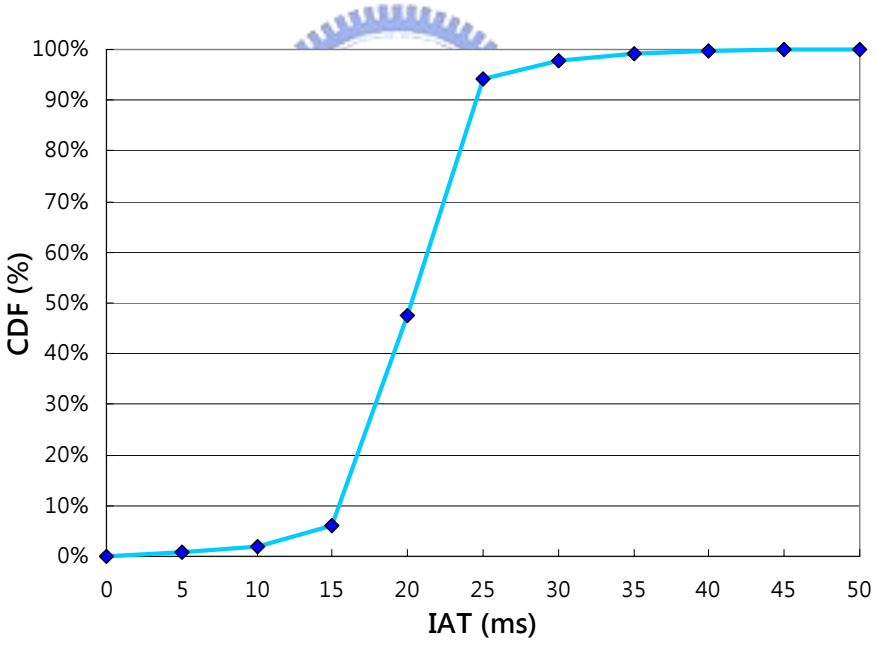

Figure 6.3: Cumulative distribution function of packets with IAT=20ms during inter-floor roaming.

而 Figure 6.3 則是 MN 在不同樓層中移動的情況，從六樓的基地台換手到五樓基地台，經由圖表可以發現 jitter 的影響比起同樓層移動時來的大，統計資料顯示 IAT 為 20ms 左右大約是 94.12%，這部分可能是在樓梯間移動的時候，收到的訊號本身就相對的較差，因此封包傳輸比較不穩定，產生 packet delay 的機會較高。

第七章　結論與未來工作

7.1　結論

自從 Intel 推出迅馳行動運算技術之後，帶動了 802.11 無線網路的市場，如今已經有越來越多的行動終端裝置如筆記型電腦，平板電腦,個人數位助理甚至是智慧型手機皆擁有存取 802.11 無線網路的能力，使用者對於隨時隨地存取網路的需求也日益提升。然而當使用者在 802.11 網路漫遊時，會遭遇到基地台換手的問題，導致使用中的應用服務中斷或造成服務品質降低；因此本論文便是針對這樣的問題，透過降低基地台搜尋階段的時間來達到提供快速且順暢的換手目的。

我們提出「具網路拓樸知覺的頻道掃描機制」，讓行動端點在適當的時機量測特定鄰近基地台的訊號，觀察其衰減或增強的變化，再利用鄰近基地台的網路拓樸資訊，判斷使用者可能的移動方向，找出此方向最有可能的目標基地台，透過如此的機制縮短基地台搜尋的時間，達到降低換手過程會造成的影響。我們並以校園大樓環境為例，利用交大工程三館所部署的 802.11 開放系統式無線網路環境作為實驗平台，在 Linux 作業系統上搭配使用 Atheros 無線網路卡，修改開放原始碼的 madwifi 驅動程式，加上自行開發的換手管理程式，完成支援此機制的平台。

我們所設計的掃瞄機制，可以針對 STA 進行訊號的量測，而所須花費的時間很短，並且是在換手前 background 中分散著進行，對於正常資料的傳輸影響不大；最後測試與分析的結果也顯示說，此機制的加入的確可以達到降低換手動作對即時性應用服務品質造成的干擾，讓使用者在換手過程中幾乎不會察覺。另外此機制與作法並沒有更改 802.11 標準的規範或是修改到基地台提供的功能，所以可以相容運行在目前已存在的 802.11 無線網路環境中。

7.2　未來工作

目前我們所設計實作的 802.11 無線網路下的頻道掃瞄機制，是以大樓建築環境為主，透過實驗確認其作法的可行性後，在未來工作方面我們有以下幾點可以延伸：

1) 針對其他環境進一步去做探討。目前所完成的是建築物內移動的情況，接下去我們可以研究在平面移動的狀況，例如在全平面環境下我們可以將鄰近基地台區分為六個方向，而這時利用我們的機制根據訊號變化去猜測使用者的移動方向時，可以盡快的找出較有可能的目標基地台集合，省略不

用掃瞄全部鄰近基地台來達到快速搜尋的效果也會更加明顯。

2) 結合 security 安全協定。目前我們是在開放系統式環境下進行實作，而在需要安全協定支援的無線網路下，STA 在換手過程需要額外執行初始驗證金鑰 PMK 或其他安全機制的動作，這些動作都會進一步造成額外的換手延遲。而我們機制可以盡早判斷出使用者移動的方向，因此也可以利用此資訊去完成事先認證的機制，縮短換手時間。

Reference

[1] IEEE, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," *IEEE Standard 802.11*, 1999.

[2] FON: the largest Wi-Fi community in the world. *Available from: http://www.fon.com/tw/*

[3] A. Mishra, M. Shin, and W. A. Arbaugh, "An Empirical Analysis of the IEEE 802.11 MAC Layer Handoff Process," *ACM SIGCOMM Computer Communications Review*, Vol. 33, No. 2, pp. 93-102, Apr. 2003.

[4] Z. H. Chen, "Effects of Handover Schemes on the Latencies of Inter-network Handovers," *NCTU Master Thesis*, *WIN Lab*, Jun. 2006.

[5] C. C. Hung, "An Empirical Analysis of IEEE 802.11 Wireless Networks and SIP-based VoIP Handovers," *NCTU Master Thesis*, *WIN Lab*, Jun. 2007.

[6] S. H. Pack, J. Y. Choi, T. Y. Kwon, and Y. H. Choi, "Fast Handoff Support in IEEE 802.11 Wireless Networks," *IEEE Communications Surveys & Tutorials*, Vol. 9, No. 1, pp. 2-12, First Quarter 2007.

[7] H. Velayos, and G. Karlsson, "Techniques to Reduce IEEE 802.11b MAC Layer Handover Time," *Proc. IEEE ICC 2004*, Jun. 2004.

[8] K. Kwon and C. Lee, "A Fast Handoff Algorithm using Intelligent Channel Scan for IEEE 802.11 WLANs," *The 6th International Conference on Advanced Communication Technology*, Vol. 1, pp. 46-50, 2004.

[9] M. Shin, A. Mishra, and W. A. Arbaugh, "Improving the Latency of 802.11 Handoffs using Neighbor Graphs," *in Proceedings of the ACM MobiSys Conference*, Jun. 2004.

[10] S. Shin, A. S. Rawat, and H. Schulzrinne, "Reducing MAC Layer Handoff Latency in IEEE 802.11 Wireless LANs," *in Proceeding of ACM MobiWac 2004*, Oct. 2004.

[11] H. S. Kim, S. H. Park, C. S. Park, J. W. Kim, and S. J. Ko, "Selective Channel Scanning for Fast Handoff in Wireless LAN using Neighbor Graph," *International Technical Conference on Circuits/Systems, Computers and Communications 2004*, Jul. 2004.

[12] N. Mustafa, W. Mahmood, A. Chaudhry, and M. Ibrahim, "Pre-Scanning and Dynamic Caching for Fast Handoff at MAC Layer in IEEE 802.11 Wireless LANs," *IEEE International Conference on Mobile Adhoc and Sensor Systems Conference*, Nov. 2005.

[13] P. J. Huang, Y. C. Tseng, and K. C. Tsai, "A Fast Handoff Mechanism for IEEE 802.11 and IAPP Networks," *IEEE 63rd Vehicular Technology Conference 2006-spring*, Vol. 2, 2006.

[14] C. C. Tseng, K. H. Chi, M. D. Hsieh, and H. H. Chang, "Location-based Fast Handoff for 802.11 Networks," *IEEE Communication Letters*, Vol. 9, No. 4, pp. 304-306, Apr. 2005.

[15] C. C. Tseng, L. H. Yen, H. H. Chang, and K. C. Hsu, "Topology-aided cross-layer fast handoff designs for IEEE 802.11/Mobile IP environments," *IEEE Communications Magazine*, Vol. 43, No. 12, pp. 156-163, Dec. 2005.

[16] I. Ramani and S. Savage, "SyncScan: Practical Fast Handoff for 802.11 Infrastructure Networks," *IEEE INFOCOM 2005*, Vol. 1, pp. 675-684, Mar. 2005.

[17] S. Pack and Y. Choi, "Fast Handoff Scheme based on Mobility Prediction in Public Wireless LAN Systems," *IEE Proc. Communications*, Vol. 151, No. 5, pp. 489–495, Oct. 2004.

[18] A. Mishra, M. Shin, and W. Arbaugh, "Context Caching using Neighbor Graphs for Fast Handoffs in a Wireless Network," *IEEE INFOCOM 2004*, Vol. 1, Mar. 2004.

[19] S. Pack, H. Jung, T. Kwon, and Y. Choi, "SNC: A selective neighbor caching scheme for fast handoff in IEEE 802.11 wireless networks," *ACM SIGMOBILE Mobile Computing and Communications Review (MC2R)*, Vol. 9, No. 4, pp. 39-49, Oct. 2005.

[20] International Telecommunication Union, "General Characteristics of International Telephone Connections and International Telephone Circuits," *ITU-TG.114*, 1988.

[21] J. Salim, H. Khosravi, A. Kleen, and A. Kuznetsov, "Linux Netlink as an IP Services Protocol," *IETF RFC 3549*, Jul. 2003.

[22] J. Corbet, A. Rubini, and G. Kroah-Hartman, "Linux Device Drivers, 3rd Edition," *O'Reilly Media, Inc.*, Feb. 2005.

[23] C. Benvenuti, "Understanding Linux Network Internals," *O'Reilly Media, Inc.*, Dec. 2005.

[24] J. Tourrilhes, "Linux Wireless Extension and Tools," *Available from: http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Tools.html*

[25] J. Malinen, "Linux WPA/WPA2/IEEE 802.1X Supplicant," *Available from: http://hostap.epitest.fi/wpa_supplicant/*

[26] Wikipedia, *Available from: http://en.wikipedia.org/wiki/Main_Page*

[27] G. Anastasi, R. Bandelloni, M. Conti, F. Delmastro, E. Gregori, and G. Mainetto, "Experimenting an Indoor Bluetooth-based Positioning Service," *IEEE Proc. Distributed Computing Systems Workshops*, pp. 480-483, May 2003.

[28] J. Hallberg, M. Nilsson, and K. Synnes, "Positioning with Bluetooth," *The 10th International Conference on Telecommunications (ICT)*, Vol. 2, pp. 954-958, Mar. 2003.

[29] A. Krohn, M. Beigl, M. Hazas, and H.W. Gellersen, "Using Fine-Grained Infrared Positioning to Support the Surface-Based Activities of Mobile Users," *IEEE Distributed Computing Systems Workshops*, pp. 463-468, June 2005.

[30] H.D. Chon, S. Jun, H. Jung, and S.W. An, "Using RFID for Accurate

Positioning," *Journal of Global Positioning Systems (CPGPS)*, Vol. 3, pp. 32-39, Feb. 2005.

[31] P. Bahl and V. Padmanabhan, "RADAR: An In-Building RF-Based User Location and Tracking System," *IEEE INFOCOM*, Mar. 2000.

[32] R. H. Jan and Y. R. Lee, "An indoor geolocation system of wireless LANs," *International Conference on Parallel Processing Workshops*, pp. 29-34, Oct. 2003.

[33] M. Gast, "802.11 Wireless Networks: The Definitive Guide, 2nd Edition," *O'Reilly Media, Inc.*, Apr. 2005.

[34] MADWiFi: Multiband Atheros Driver for WiFi, *Available from: http://madwifi.org/*

[35] Wireshark: The World's Most Popular Network Protocol Analyzer, *Available from: http://www.wireshark.org/*

[36] BillSniff: Network Protocol Analyzer, *Available from: http://secureinf.com/billsniff.php*