國立交通大學

資訊科學與工程研究所

碩 士 論 文

IEEE 802.11 無線網狀網路之分散式時槽分割式多重頻道協定

A Distributed Time-Slotted Multi-Channel Protocol for IEEE 802.11 Wireless Mesh Networks

研 究 生：陳威碩

指導教授：林正中　教授

曾煜棋　教授

中 華 民 國　九 十 五　年 六 月

IEEE 802.11 無線網狀網路之分散式時槽分割式多重頻道協定
A Distributed Time-Slotted Multi-Channel Protocol for IEEE 802.11 Wireless Mesh Networks

研 究 生：陳威碩　　Student：Wei-sho Chen

指導教授：林正中　　Advisor：Cheng-Chung Lin

曾煜棋　　Yu-Chee 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 2006

Hsinchu, Taiwan, Republic of China

中華民國九十五年六月

IEEE 802.11 無線網狀網路之分散式時槽分割式多重頻道協定

研究生：陳威碩　　指導教授：林正中 博士

曾煜棋 博士

國立交通大學資訊科學與工程研究所碩士班

摘要

IEEE 802.11 協定允許使用不同頻道來提升網路效能，儘管近年來有不少這方面的研究，但分散式的分配頻道仍然是個複雜而具有挑戰性的問題。無線網路網路(Wireless Mesh Network)近年來倍受國際矚目，它提供有線區網的另一種更方便、更便宜的選擇，而在無線網狀網路上使用多重頻道(Multiple Channel)是非常吸引人的一個議題，因為無線網狀網路必需提供很大的頻寬給使用者。我們提出一個適合無線網路網路的鏈結層的多重頻道管理協定來增進整個網路的效能，這個協定使用已普及的 IEEE 802.11 相容的網路卡介面，每個存取點只需一個介面便能順利運作，並且很容易能擴充至裝備多張介面卡的存取點。我們設計此協定使用接收端為主(Receiver-Based)的頻道分配演算法，並使用時槽來控制什麼時候該送、什麼時候該收。我們在真實的網路環境下實作這個協定來驗證我們方法，發現確實的提升了網路的吞吐量(Throughput)。

關鍵字：無線網狀網路、多重頻道、媒體存取控制、頻道分配、排程分配

A Distributed Time-Slotted Multi-Channel Protocol for IEEE 802.11 Wireless Networks

Student : Wei-sho Chen Advisor : Prof. Cheng-Chung Lin

Prof. Yu-Chee Tseng

Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

ABSTRACT

Multiple channels which are available for use in IEEE 802.11 can increase network capacity. Despite being the subject of many years of research, distributed channel assignment remains a challenging problem. The idea of exploiting multiple channels is particularly appealing in wireless mesh networks because of their high capacity requirements to support backbone traffic. We propose a link-layer protocol for wireless mesh network that utilizes multiple channels dynamically to improve performance. The protocol can be implemented in software over an IEEE 802.11-compliant wireless card. The protocol only needs one interface and can be easily extended to multiple interfaces. We design a receiver-based channel assignment algorithm and use time slot to control when to send and when to receive. Our protocol has been implemented in real environment and the result indeed shows improvement on the network performance.

Keywords: Wireless Mesh Network, Multi-Channel, Medium Access Control, Channel Assignment, Scheduling

誌謝

誠摯地感謝曾煜棋教授兩年來的指導與鼓勵，並提供良好的研究環境及充足的實驗設備，讓我得以順利完成此篇論文並取得碩士學位。

此外，也由衷地感謝指導我的林亭佑學姊、林致宇學長、郭博學長，以及一直和我一起共同討論解決問題的談偉航，感謝你們在我創作此篇論文時的參與討論與提供意見。

最後，感謝家人、實驗室以及關心我的人對我的期許及關懷，僅以此論文獻給你們。

在此向您們獻上我最誠摯的謝意。

陳 威 碩 謹識於

國立交通大學資訊科學與工程研究所碩士班

中華民國九十五年六月

論文目次

表目錄

圖目錄

一、緒論

近年來國際間「無線網狀網路」(wireless mesh networks)的相關研究備受矚目，更引起了學界與業界的廣泛討論。無線網狀網路系統是基於IEEE 標準協定實現的新型網路系統，可提升無線域區的覆蓋能力，並可為有線網路提供另一種便宜、方便的選擇。

1-1、研究背景

我們的研究背景立基於目前漸趨成熟的無線網狀網路，由於無線網狀網路的高頻寬需求，我們引進使用多重頻道的優點來改善其效能。

在多躍式隨建即連無線網路(multi-hop ad-hoc wireless network)裡，傳輸速率會因為鄰近網格點的同時傳輸而干擾降低速率，利用多重頻道(multiple channel)可以避免這種干擾進而有效提升無線網路的效能。在 IEEE 802.11 網路中有存在著數個不與其他頻道重疊(non-overlap)的頻道，例如在 IEEE 802.11b 中有三個不相互重疊的頻道可使用，而在 IEEE 802.11a 中更多達十二個不相互重疊的頻道可使用，雖然在 IEEE 802.11 Infrastructure mode 中，相鄰的基地台使用不同的頻道來降低干擾的方法已經被提出，例如交大研究團隊在文獻[14]中提出了動態調整無線網路基地台的頻道的方法，然而在 IEEE 802.11 Ad-hoc mode 中如何有效的利用多重頻道增進網路效能卻還是一個值得研究的領域。

我們可以從下圖 1 看出使用多重頻道的好處，在使用同一個頻道時，兩個連線(links)會互相競爭干擾這個頻道，雙方理想值只能得到這個頻道頻寬的一半，或許還會更低，但如果這兩個連線使用不同的頻道，便能獲得這個頻道的整個頻寬，進而提升整個網路的效能。

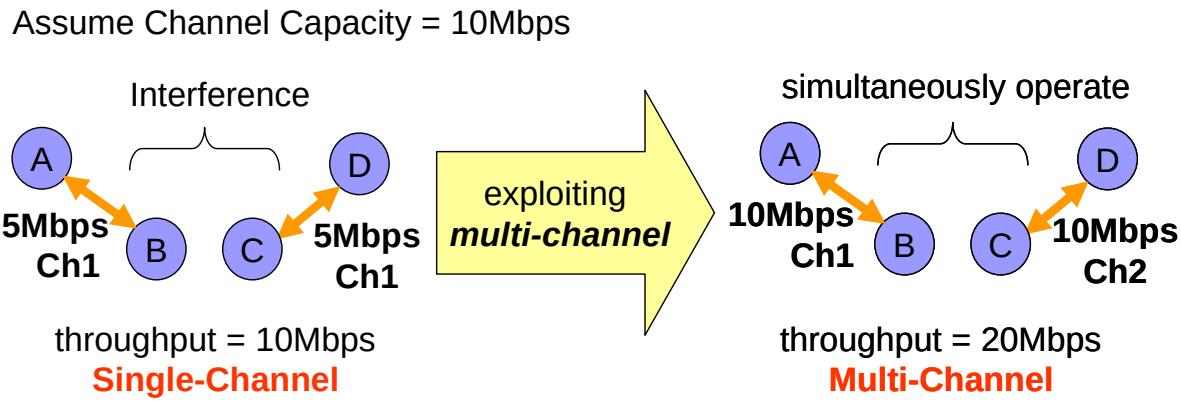

圖 1、頻道干擾範例

無線網狀網路(wireless mesh network)就像一個由一群路由器(routers)組成的網路，如圖 2，差別在於網狀網路之間是用無線網路的方式來連線，最近這幾年無線網狀網路也獲得相當大的重視，無線網狀網路可提供無線寬頻服務的網路，它融合了無線區域網路(wireless LAN)和隨建即連(Ad-Hoc)網路的優勢，無線網狀網路採用無線傳輸的方式連結了許多的網格點(mesh point)，有了無線網狀網路的架設可讓網路服務提供者透過少量的有線網路達成大範圍的服務區域，且因為少了使用有線網路時纜線的建置過程，時間與成本都可大量的減少。

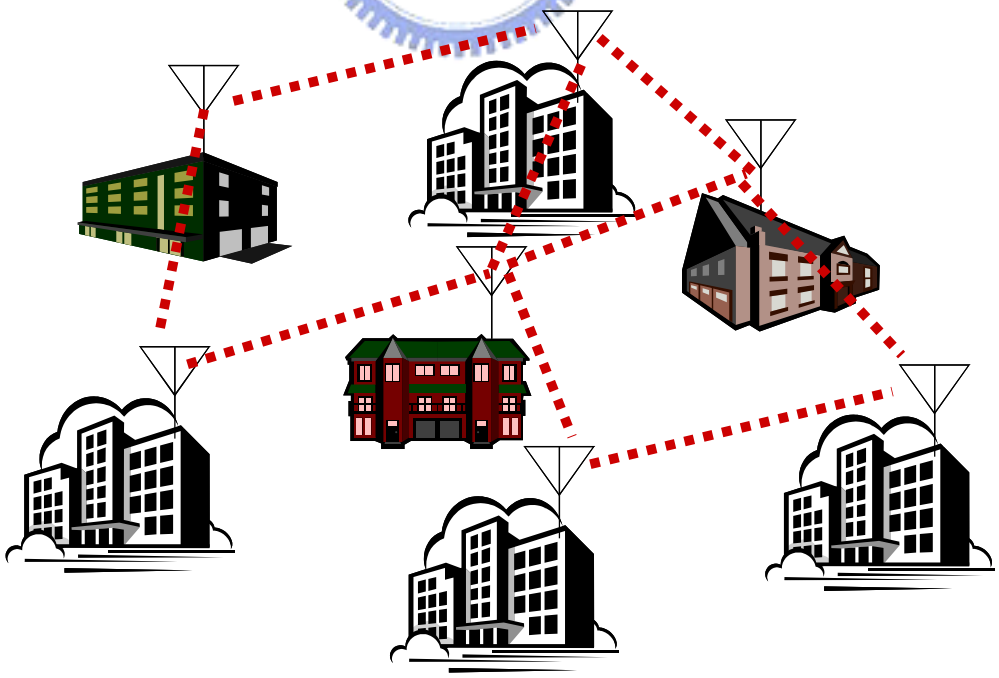

圖 2、無線網狀網路

1-2、研究動機與目的

在 IEEE 802.11 網路中有存在著數個不與其他頻道重疊(non-overlap)的頻道，雖然在 IEEE 802.11 Infrastructure mode 中，相鄰的基地台使用不同的頻道來降低干擾的方法已經被提出，然而在 IEEE 802.11 Ad-hoc mode 中如何有效的利用多重頻道增進網路效能卻仍是一個值得研究的領域，例如 IEEE 802.11 的媒介存取協定(Medium Access Control Protocol)的設計只針對單一頻道的使用，因此造成效能無法進一步地提升。當多個頻道被利用時，可預期的是網路的吞吐量(throughput)會增加，干擾可降低，空間中頻道的再使用率(spatial reuse)能提高。

我們最終的目的是要能夠利用多重頻道來提升無線網狀網路的吞吐量。由於在無線網狀網路中，有一些特性是和傳統無線區域網路或無線隨建即連網路不同的，例如網格點是不具移動能力的，此外在無線網狀網路中，通常會有一個連接至有線網路的閘道器(gateway)，因此在網路協定的設計上會有別於傳統的無線區域網路或隨建即連網路。例如由於網狀網路是以效能的提升(throughput)為考量，而非以省電或硬體成本為考量，因此通訊協定上的設計可以較為複雜，而多頻道的使用也更為合理，因此針對無線網狀網路的環境，在頻道的切換與管理方面我們提出一個可在鏈結層(link layer)實作的解決方法，以便提供整個無線網狀網路最大的效能。

1-3、研究範圍

我們將針對無線網狀網路提出一個頻道管理協定，並於鏈結層上實作出來，我們預期此頻道管理能夠在只有一個網路介面卡的環境下運作，如此能減少硬體的成本與硬體的相容性。

在頻道管理上要考量的主要問題是，要分配哪一個頻道給哪一個網路介面卡使用，分配好頻道後使用多久要再度切換頻道等問題，在決定如何分配頻道的問題上，我們必須考量每個鏈結(link)的負載量(load)，以及鄰近網格點所使用的頻道狀況，如此才能降低干擾進而提高整體網路效能。

而頻道的使用問題上，我們必須考量每個網格點都可隨著時間的變化而

動態的調整所使用的頻道，在這過程中要考慮如何跟鄰居交換頻道的使用資訊，另外在多重頻道環境下廣播(broadcast)封包的傳送也是我們必須考量的問題。

1-4、論文結構

本篇研究論文之架構區分為五個章節：第壹章為緒論，闡明本研究之背景、動機與目的，第貳章為相關研究，介紹目前有關使用多重頻道應用於的隨建即連網路(Ad-Hoc network)或無線網狀網路(wireless mesh network)的文獻，在第參章中，會詳盡說明我們提出的方法和管理協定，以及衍生出來的議題，第肆章利用真實的網路環境形成無線網狀網路，並修改網路卡的驅動程式，把我們的協定加進裡面，並測試成果，於第伍章中則是本研究論文的結論與未來發展方向，以供為後續研究之參考。

二、相關研究

下面我們將透過本章相關文獻的整理，進一步了解應用多重頻道在無線網路方面的發展，並分析它們的優缺點。

參考文獻	[1]	[2]	[3]	[4][5]	[12]	[13]
頻道數	有限	有限	有限	有限	無限	有限
MAC 協定	需新 MAC	802.11 相容	802.11 相容	802.11 相容	需新 MAC	需新 MAC
網路卡數	兩張	均可	單張	多張	單張	多張
時間同步	不需要	不需要	需要	不需要	不需要	不需要
演算法	分散式	集中式	分散式	分散式	分散式	分散式
應用網路	Ad-Hoc	Mesh	Mesh	Mesh	Ad Hoc	Ad-Hoc

表 1、多重頻道範疇

上表列出了相關多重頻道應用在無線網路的範疇，在[12]中假設頻道數無限多，在[1]中需要新的媒介存取層協定(Medium Access Protocol, MAC)，在[4][5]中網路卡數必須多於一張才能實行，[3]為單張網路卡，利用不停的切換頻道來提高空間的再使用率，但必需時間同步，[2]中的演算法是集中式演算法，必須知道整個網路的拓撲，[13]把多重頻道建立在繞徑(routing)上，應用的網路類型屬於隨建即連(Ad-Hoc)式網路。

我們的問題鎖定在無線網狀網路和不改變 IEEE 802.11 協定的條件下，所以以下僅簡介在此條件下的研究，頻道數無限多不是一個合理的假設，我們討論的是有限的頻道數。下面列出幾個相關的研究，並分析其缺優點，以便帶入我們的研究主題。

2-1、多重頻道多網卡無線網路（MCR）

[5]為美國伊利諾大學(University of Illinois at Urbana-Champaingn) Pradeep Kyasanur 及 Nitin H. Vaidya 提出的分散式方法，他們稱作MCR(Multi-Channel Routing)，不需要時間同步，使用兩張以上網路卡，一為固定式網路卡，另為可切換式網路卡，每個網格點都需為固定式網路卡設置一個頻道，這個頻道稱之為接收頻道，在鄰近區(neighborhood)的每個網格點的接收頻道需盡量不同，為了達到這個要求，每個網格點會廣播自己的接收頻道給鄰居知道，當鄰居收集到這些資訊，便可設定一個較適當的頻道給固定式網卡。這是一個接收頻道為主(receiver-based)的演算法，利用固定式網卡來接收資料，用可切換式網卡來送資料，當要送資料時，可切換式網卡必需切換到接收端的接收頻道上，這樣才能使用相同的頻道進行傳送。

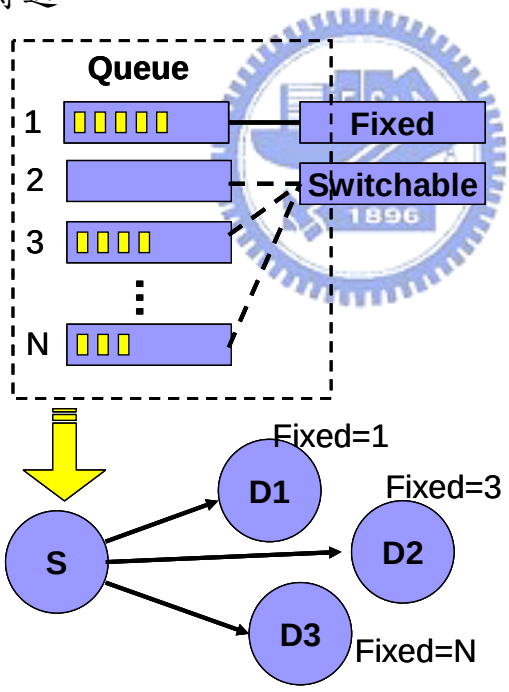

圖 3、多重頻道多網卡無線網路（MCR）

每個網格點會維護每個頻道佇列，如上圖 3，當上層有封包要求傳送時，封包會依照接收端的接收頻道來放進頻道佇列中，固定式網卡只需管理接收頻道的那個頻道佇列，而可切換式網卡必需適時的切換到其它的每個頻道佇列去把未送出的資料送出。另一個需要解決的問題是廣播問題，因為

每個網格點接收資料的頻道不同，所以當需要廣播的時候，每個頻道都要發送，所以廣播的封包必需在每個頻道佇列上都複製一份，才能確保接收端能收到資訊。

MCR 的缺點在於網路上每個網格點的接收和傳送的比例不盡相同，某些網格點可能時常在做傳送的動作或接收的動作，便會有一張網路卡是浪費的，而且在這個方法的運作下，廣播需要很大的花費，因為每個頻道都要廣播一次。

2-2、時槽式跳頻法（SSCH）

[3]為微軟公司(Microsoft) Raramvir Bahl、Ranveer Chandra 及 John Dunagan 的研究，簡稱 SSCH(Slotted Seeded Channel Hopping)，是一個分散式的演算法，只需用單張網卡，不過需要時間的同步。

SSCH 利用跳頻的方式來達成提升空間中頻道的再使用率(Spatial Reuse)，SSCH 定義一個時槽(slot)為在單一頻道上所花費的時間，他們選擇一個時槽為 10 毫秒，並定義頻道排程(channel schedule)為一連串的頻道作為跳頻順序，頻道排程包含目前的頻道和一個跳頻的規則，可利用規則計算出下一個時槽該跳至那一個頻道，利用規則來跳頻可以節省使用龐大的空間來儲存跳頻的順序，每個網格點必須儲存並維護其它鄰居區內所有網格點的頻道排程。

SSCH 用四組(channel,seed)配對來組成頻道排程，他們實驗結果顯示四組已經有很好的效能增進，用(xi,ai)來表示(channel,seed)，xi 表示[0,12]共 13 個可能的頻道，而 seed ai 為[1,12]個整數，每個網格點會一直根據頻道排程的資訊來跳頻，下個時槽的跳頻頻道公式為：

xi<-(xi+ai)mod 13 ---公式(一)

13 為可能的頻道，此篇假設可用頻道為 13 個，這個可用頻道數必需為質數個，如此一來，利用公式(一)的跳頻方法，任意兩個網格點的 ai 不同，便能保證在跳躍過程中，每 13 次的跳躍，一定會有某個時槽，而且只有一個，兩個網格點的頻道會跳到同一個頻道，如此即使頻道排程互不相同的兩個

網格點也能經由這種偶發性的頻道重疊而進行溝通，主要是用在廣播上。但如果兩個網格點頻道排程的頻道不同，但種子(seed)相同時，使可能發生兩個頻道永遠沒有重疊的情況，所以走訪完每個配對的所有頻道必須加上一個配類時槽(parity slot) ，這個配類時槽所需跳躍到的頻道為第一個配對的種子(seed)的數值，加了這個配類時槽就能避免兩個網格點的頻道不一樣但種子(seed)相同帶來頻道永遠無法重疊的情況，因為至少能在配類時槽能重疊。

下圖 4 為兩個網格點的頻道排程，此例共有 3 個頻道，2 組(channel, seed)配對，每一個週期(cycle)必須走訪完每個配對的所有頻道加上一個配類時槽(parity slot)，接著一直重覆著這個週期。如圖可以看到網格點 A 的第一組配對(x1, a1)和網格點 B 的第一組配對(x1, a1)均為(1, 2)，表示兩個網格點的第一組配對頻道排程相同，跳頻的順序均會相同，在此配對的時槽中，兩個網格點便可進行溝通，因為種子(seed)也相同，所以在配類時槽(parity slot)也會重疊。在圖中時槽裡的數字表示在這個時槽要跳躍到的頻道，網格點 A 的第一個配對(1, 1)表示目前這個時槽使用頻道是頻道 1，而更新的規則是加上種子(seed)1，所以下一次跳躍的頻道是(1+1)mod 3=2。

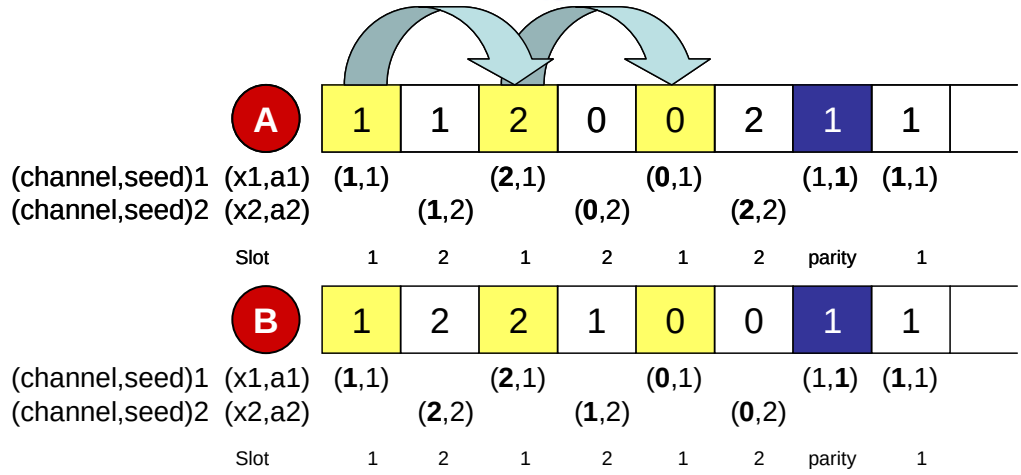

圖 4、時槽式跳頻法 (SSCH)

當某網格點想要傳輸資料給某鄰居網格點時，可以改變自己的某個(channel, seed)配對跟鄰居相對應的配對相同，如此，這個配對所屬的時槽便能對行溝通，而另外的配對可以再跟其它的鄰居配合傳輸，在他們的

模擬中是以四組配對作為實驗。在如此的架構下，除非(channel, seed)配對完全相同，否則最多頻道只會重疊一次，可降低干擾，至於廣播的問題，他們在每個時槽均廣播自己的頻道排程，因為每個週期(cycle)會至少頻道重疊一次，所以利用這個性質，便可確保在一個週期內可把廣播訊息廣播出去。

SSCH 的缺點在於廣播所需要的時間很長，假設每個時槽為 10 毫秒、13 個可用頻道，需 530 毫秒才能確保廣播至所有的鄰居網格點，對於繞徑(routing)尤其不利，找尋路徑需花費很長的時間。而且每個時槽都必需廣播自己的排程，花費高。另外，SSCH 並沒有規定那些需固定用那些頻道排程，不小心就一群人都使用相同的排程，造成頻道擁塞(channel congestion)，需要做反同步(de-synchronization)來把頻道使用再度打散。

三、時槽式多重頻道管理協定

整理了許多多重頻道相關的研究，我們想出了一個方法適用在有限頻道數、單張網路卡、IEEE 802.11 相容、分散式(distributed)、網狀網路(Mesh Network)的多重頻道管理協定，不過需要時間的同步，並可容易擴充至多張網路卡的環境。

3-1、基本原理

我們發展了一套適用於無線網狀網路(Wireless Mesh Network)，而可實作在鏈結層(link layer)上使用多重頻道的頻道管理協定(channel management protocol)，這是一個以接收端的頻道(receiver-based)做為傳輸頻道的方法，主要想法是假設當有一個網格點(mesh point) A 要傳送資料給另一個網格點 B 時，A 就要切換到 B 所使用的頻道上進行通訊。我們假設每個網格點都會分配到一個接收頻道(receiving channel)，此頻道應該與網格點的鄰居(neighbors)所使用的接收頻道(receiving channel)要盡量的不同。如圖 5，網格點 A, B, C, D, E, F 都盡量的使用不同的頻道做為接收頻道(receiving channel)，例如 A 使用 Channel 3，B 使用 Channel 5，C 使用 Channel 4 等，假如 B 要傳輸資料給 A，會用 Channel 3 去傳輸資料，同時 C 要傳資料給 D，會用 Channel 1 去傳輸，所以在同一個鄰居區內，CD 和 AB 的傳輸會用不同的頻道，而降低干擾使網路吞吐量(throughput)增加。至於一個網格點如何選取適當的頻道，稍後會做詳細的說明，下面先將設計頻道分配演算法所要用到的其它概念做敘述。

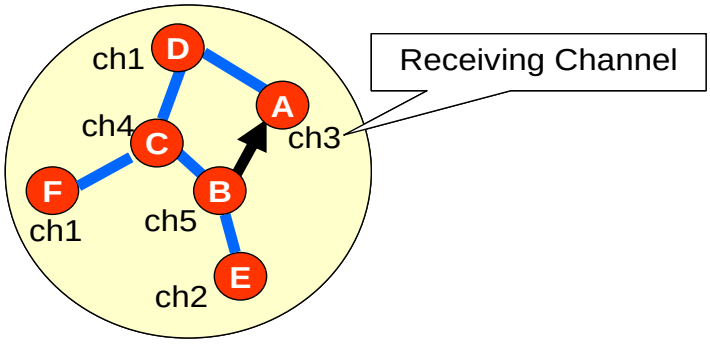

圖 5、 接收頻道示意圖

接收端為主(receiver-based)的設計會產生一個問題，如果網格點 A 要傳送資料給網格點 B，A 會切換到 B 的接收頻道(receiving channel)上，但假設此時 B 也正要傳送資料給 C，則 B 會切換到 C 的接收頻道(receiving channel)上，如此有可能形成死結(deadlock)的情形，為此我們加進“分時”的概念，我們把每個網格點的時間軸切成一個一個的時槽(time slot)，我們假設每個網格點的時槽的開始是同步的，並設定每 k 個時槽為一個週期(cycle)，然後重覆這 k 個時槽。在這 k 個時槽裡，每個網格點(假設為 A)需指定好那些時槽是用來傳送資料給其它網格點，那些時槽是用來接收其它網格點所送過來的資料，然後把這個資訊廣播給其鄰居(假設為 B)，當 B 收到這個資訊時，B 就能知道 A 的時槽使用情況，如此 B 有資料要傳送給 A 時，B 能夠知道 A 何時可接收資料，何時不能接收資料，當然 B 會選擇 A 可接收資料的時槽傳送資料給 A，除了傳送時槽、接收時候和廣播時槽外，我們還選了一個接收時槽當固定式接收時槽(fixed receiving slot)，因為我們可以動態改變排程，所以接收時槽可能轉變為傳送時槽，為了不讓所有的時槽都變成傳送時槽，固定式的接收時槽是不能變成傳送時槽的，至於固定式接收時槽的選取，也是在鄰居區內要盡量不同。

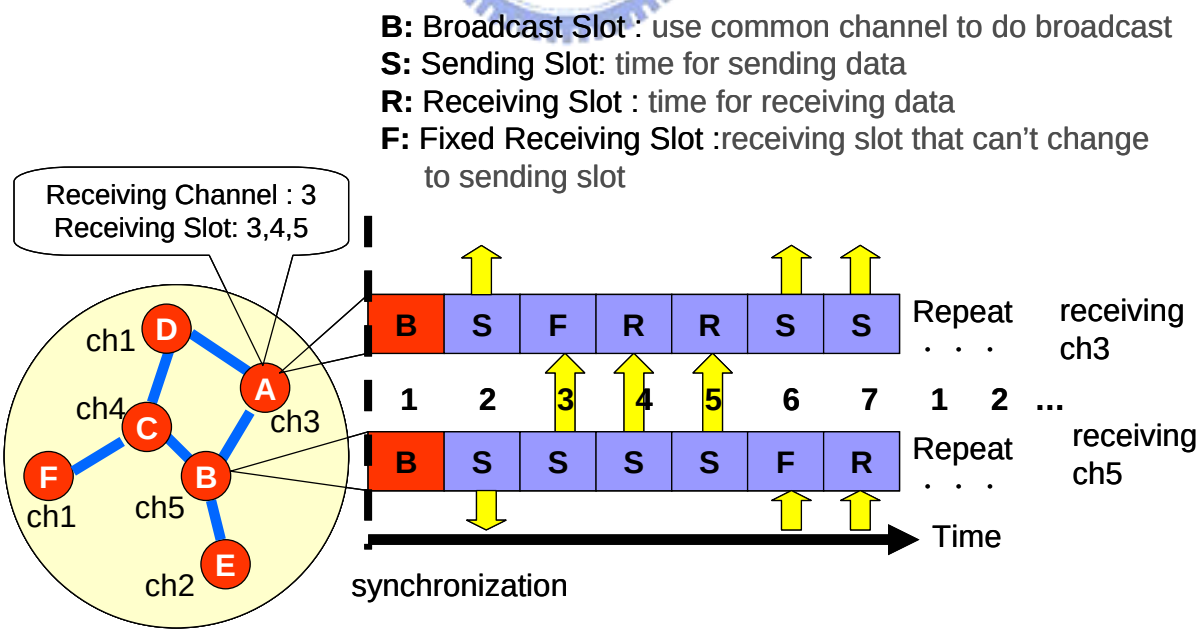

圖 6、channel model

如圖 6：在這個例子中 k 值為 7，當網格點 B 要傳送資料給 A，因為 B

有收到 A 的廣播說 A 的時槽 3,4,5 是用來接收資料的時槽，因此 B 要送資料給 A 時，B 會利用時槽 3,4,5 將頻道切換到 A 的接收頻道(receiving channel)，在此例中為 Channel 3，來進行資料的傳輸。

另外要解決的問題是廣播(broadcast)的問題，在多重頻道(multi-channel)的網路環境下，因為每個網格點可能正使用不同的頻道，如何做有效率的廣播就是一個問題，相關研究中廣播的方法大多是複製多個廣播封包(broadcast packet)在每個頻道都廣播出去，以便確保鄰居都能接收到此廣播封包，我們所採取的做法是選擇第一個時槽裡當成廣播時槽(broadcast slot)，在這個廣播時槽裡，頻道會切換到一個大家共同的頻道，其目的就是要把所有的網格點在這個時候同時切換至此共同頻道上，如此一來，所有的網格點便能同時接收或傳送廣播封包，因為我們並沒有改變 IEEE 802.11 的 MAC 協定，所以這時的接收和傳送是經由 IEEE 802.11 的競爭機制在傳送。這樣做的好處在於每一次的廣播只需廣播一次便所有的網格點都接收的到，並不需要在每個頻道上做廣播的動作，也不需要複製多個廣播封包。此外，我們必須考量一個問題：這樣是不是會造成頻道擁塞(channel congestion)？因為這個時候所有的網格點都切換到這個頻道上，造成封包過多超過這個頻道所能負荷的量。在我們的協定裡，這是可以避免的，因為我們定義的一個週期的時槽數可經由廣播封包和一般封包的比例來設定，廣播時槽可以在一個週期不一定只有一個，可以有兩個或三個，可視這個網路的特性去調整這個參數。廣播時槽帶來的好處還不只這些，在多重頻道上同步是有困難的，因為所有的人不在相同的頻道上，有了這個廣播時槽，順使可以在這個時槽發送信號彈(beacon)來達成時間同步的效果。

3-2、協定設計細節

發展這個管理協定，衍生出一些待解決的議題，第一，如何決定每個網格點的接收頻道(receiving channel)？第二，如何分配每個網格點傳送時槽(sending slot)和接收時槽(receiving slot)的比例？第三，如何分配

傳送時槽和接收時槽的順序？第四，進入某個傳送時槽時，要選擇傳送給那一個鄰居才不會造成不公平？

3-2-1、接收頻道分配

(Receiving Channel Assignment)

為什麼要選擇接收頻道？因為接收頻道的選擇關係到整個網路的效能，每個連線(wireless link)使用愈不同的頻道，干擾的情況就愈小，整體網路效能便能上升。干擾範圍半徑通常為溝通範圍半徑的二到三倍，現在我們來敘述一個網格點如何選擇接收頻道，其方法如下：

1. 加入網路後，先聽數個週期，但不發送訊息，得知 1~3-Hop 鄰居(neighbor)使用頻道的資訊。
2. 使用 Hop Weight 來決定干擾程度，Hop 愈長，Weight 愈小，選一個干擾程度最小的接收頻道當自己的接收頻道(receiving channel)。
3. 在廣播時槽廣播自己的接收頻道(receiving channel)並幫忙轉送 1~2-Hop 鄰居的接收頻道。
4. 收到其它網格點的資訊，更新自己的資料表，此時便知道其它 1~3-Hop 網格點的接收頻道分佈狀況。
5. 每隔一段時間檢查資料表，查詢每個頻道的干擾程度，發現自己的接收頻道干擾程度比其它頻道大時，重覆步驟 2~4。

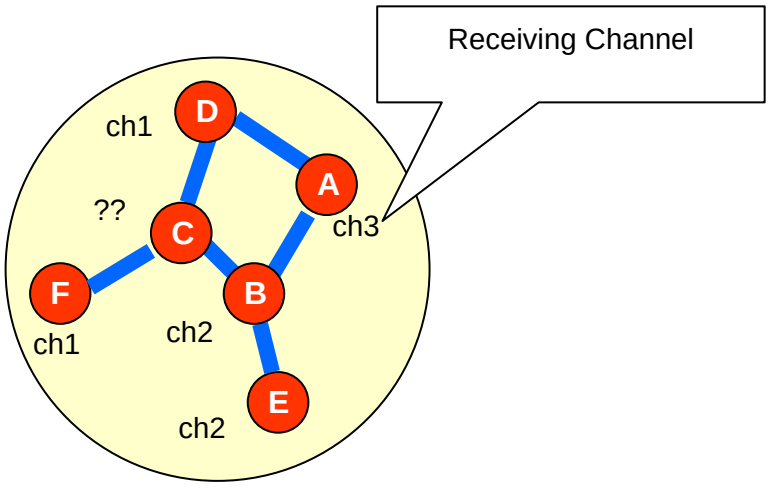

圖 7 、選擇接收頻道

以上圖 7 為例，A、B、C、D、E、F 節點代表的是每個網格點，旁邊的

數字代表接收頻道為幾號頻道，總頻道數為 3，以 ch1、ch2、ch3 表示。假設 1-Hop Weight=3, 2-Hop Weight=2, 3-Hop Weight=1，點 C 是剛加入網路的網格點，因為鄰居網格點會在廣播時槽廣播自己的接收頻道是那一個頻道，當 C 收集一段時間後便能得知 1-Hop 鄰居 B、D、F 的接收頻道為 ch1、ch1、ch2，2-Hop 鄰居 A,E 的接收頻道為 ch3,ch2，沒有 3-Hop 的鄰居，計算每個頻道的干擾狀況，ch1 干擾程度=3+3=6，ch2 干擾程度=3+2=5，ch3 干擾程度=2，所以選一個干擾程度最小的的頻道 ch3 當接收頻道，接著廣播自己的接收頻道讓其它網格點知道。因為我們的演算法是一個以接收端的頻道(receiver-based)做為傳輸頻道的方法，要傳輸就必需切換到接收端的接收頻道去傳輸，如果每個接收端的接收頻道均不同，則每條連線所使用的頻道就會不同，干擾便會大幅降低，使用 Hop Weight 的原因是因為距離愈遠，干擾的程度就愈小，變成比較不重要，比重就要低一點。如此一來，選擇到的接收頻道便能達到比較好的效果。

3-2-2、傳送接收時槽比例分配

(The Ratio of Sending and Receiving Slot)

我們觀察到並不是每個網格點傳送量和接收量總是均衡的，每個網格點的網路流量特性不同，某些網格點比較傾向接收資料，或傾向傳送資料，所以傳送時間和接收時間的比例我們設計成可變的，因此傳送時槽數(sending slot)和接收時槽數(receiving slot)變成可動態依照情況而改變，至於如何分配每個網格點傳送時槽和接收時槽的比例，我們使用單位時間內需傳送量和需接收量的比例來當傳送時槽和接收時槽的比例。公式為：

α* S/R < O/I ➔ S+1,R-1

β* S/R > O/I ➔ S-1,R+1

α、β為穩定度的參數，S 為目前傳送時槽的數目，R 為目前接收時槽的數目(包含固定接收時槽 fixed receiving slot)，O 為單位時間需傳送量，I 為單位時間需接收量。雖然說無線網狀網路(wireless mesh network)的網路流量穩定，但如果單位時間需傳送量和單位時間需接收量的比例剛好在

某個臨界數值之徘徊，會造成一下增開傳送時槽，一下子又減少傳送時槽，造成排程不穩定，這不是我們希望的，我們取α為1.2、β為0.8，表示如果比例沒有超過α倍的話，不會增開傳送時槽，沒有低於β倍的話，不會減少傳送時槽，避免時槽比例經常性更換。另外有一條規定是，接收時槽最小個數需保持一個來接收資料，我們使用固式定接收時槽來達成這個規定，因為這個時槽是不能改變成傳送時槽的，以免其它人永遠無法傳送資料給它。如此一來，每個網格點便依照著自己的流量特性來分配傳送和接收比例，達到較好的傳輸效果。

接下來我們探討I（單位時間內所需接收的量)要怎麼衡量？O（單位時間內要傳送的量）可以用上層(IP 層)要求傳送封包的量來衡量，但是 I 並不能用收到的流量來衡量，因為只能由接收時槽上獲得，無線網狀網路是高負載的網路，IP 層要求傳輸的量遠大於從接收時槽上接收到的流量，所以依照我們的演算法，傳送時槽會一直增開，所以我們衡量 I 需用別人想要送給我們的流量來計算，但是每個網格點並不知道別人有多少流量要傳給它，所以每個網格點在廣播頻道排程時，要順便將要傳送給其它網格點的流量含進排程，其它網格點收到時，便能清楚知道別人要傳給自己有多少流量，用這個一數值來衡量I，這個演算法便可以讓傳送時槽和接收時槽的比例正確的分配。

3-2-3、傳送接收時槽順序分配

(The Order of Sending and Receiving Slot)

當決定了傳送時槽(sending slot)和接收時槽(receiving slot)的比例，還必需決定傳送時槽和接收時槽要放在週期內的那一個時槽裡，如果排序的不好，如下圖 8，兩個網格點的排程幾乎都一樣時，會造成無法連線的情況，所以排程也必需詳細考慮。

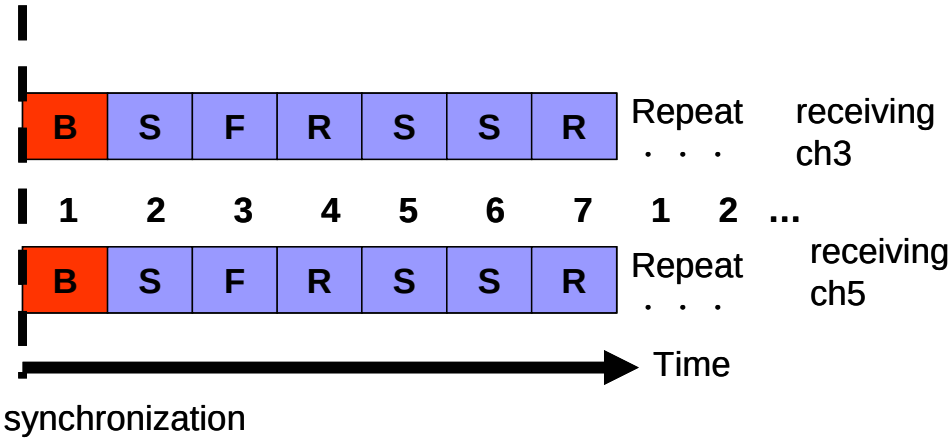

圖 8、排程相同造成無法溝通

初始時，除了廣播時槽(broadcast time slot)外，其它時槽(time slot)預設均為接收時槽(receiving time slot)包含固定接收時槽(fixed receiving slot)，隨著傳送流量的變化，會有三種情況，第一種情況，傳送時槽和接收時槽比例 S/R 沒有變，第二種情況，需要增開傳送時槽，第三種情況，需要減少傳送時槽。

第一種情況是最簡單的，排程根據之前傳送時槽和接收時槽的順序便可，除此，亦可接收時槽和傳送時槽交換，以便得到更好的結果。第二種情況，當需增加傳送時槽(sending slot)時，定義 Lj 為第 j 個時槽衡量值，這個值愈高代表這個時槽改成傳送時槽會有比較好的效果，下面會有 Lj 的正式定義，當要增開傳送時槽時，我們會選擇所有接收時槽(不含固定接收時槽)中 Lj 值最大的那個時槽。第三種情況，跟第二種情況反其道而行，選擇所有傳送時槽中 Lj 最小的時槽變成接收時槽。

接下來，我們正式的定義 Lj：

$$Lj = \sum_{Sij='R'or'F'} Wi/Gi \; - \sum_{Sij='S'} Oi/Ti$$

Sij 為鄰居 i 的第 j 個時槽的類型(接收時槽、傳送時槽或固定接收時槽)，Wi 為對鄰居 i 要傳送的量的比重(weight)，這個比重通常用封包到達速率(packet arrival rate)來代表，也可以用其它方式來代表，定義 Oi 為其它鄰居有多少流量要傳給我的比重，從鄰居發送排程時內含的資料獲得，

Gi 為自己的傳送時槽配合到鄰居 i 的接收時槽的總個數，意義上是說自己共有 Gi 個時槽可以跟 Ni 進行傳輸，如下圖 9，自己和鄰居 1 的 Gi 值為自己的傳送時槽數配合到鄰居相對應的接收時槽，此例，G_1 是 2，因為共有 2 個時槽有配合到。反之，Ti 為鄰居 i 的傳送時槽配合到自己的接收時槽的總個數，根據 Lj，我們便可以選出效果較好的時槽來改變成接收時槽或傳送時槽。

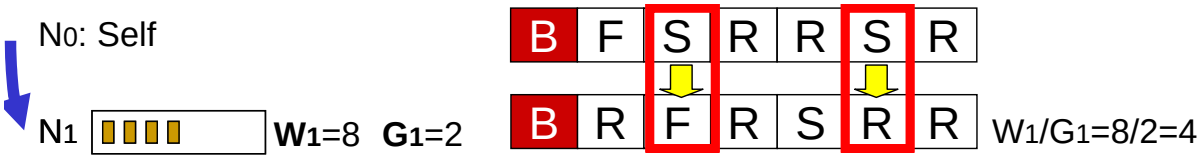

圖 9、Gi 值的計算

下圖 10 為某網格點增開新傳送時槽的範例，N_4 有資料要傳給 N_0，N_0 有資料要傳給 N_1、N_2、N_3，我們先對每個鄰居計算 Wi/Gi 和 Oi/Ti，現在是 N_0 要增開傳送時槽，所以我們考慮第 4、5、7 個接收時槽，L_4 為衡量第 4 個時槽的值，改變 L_4 變成傳送時槽會影響到 N_1、N_3、N_4，這個改變對 N_1 和 N_3 是好的，因為 N_1 和 N_4 便可接收到來自 N_0 的資料，但對 N_4 是不好的，因為 N_4 的資料便無法從這個時槽傳送到 N_0，所以 L_4 的值是 W_1/G_1 加 W_3/G_3 再減 O_4/T_4，算出來是 9，我們再計算其它時槽的 Lj，L_5 是 5，L_7 為無限大，因為 N_0 有資料要傳給 N_2，但 N_0 跟 N_2 並沒有可溝通的時槽，G_2 為 0，W_2/G_2 變成無限大，所以我們目前最迫切的是要能跟 N_2 溝通，所以改變第 7 個時槽成為傳送時槽是最好的。我們可以發現並不是只考慮比重(weight)最重的那一個鄰居，因為有可能這個鄰居已經可以用其它的時槽去溝通了，最迫切的是那些有資料要傳，但可溝通的時槽很少的那些鄰居。

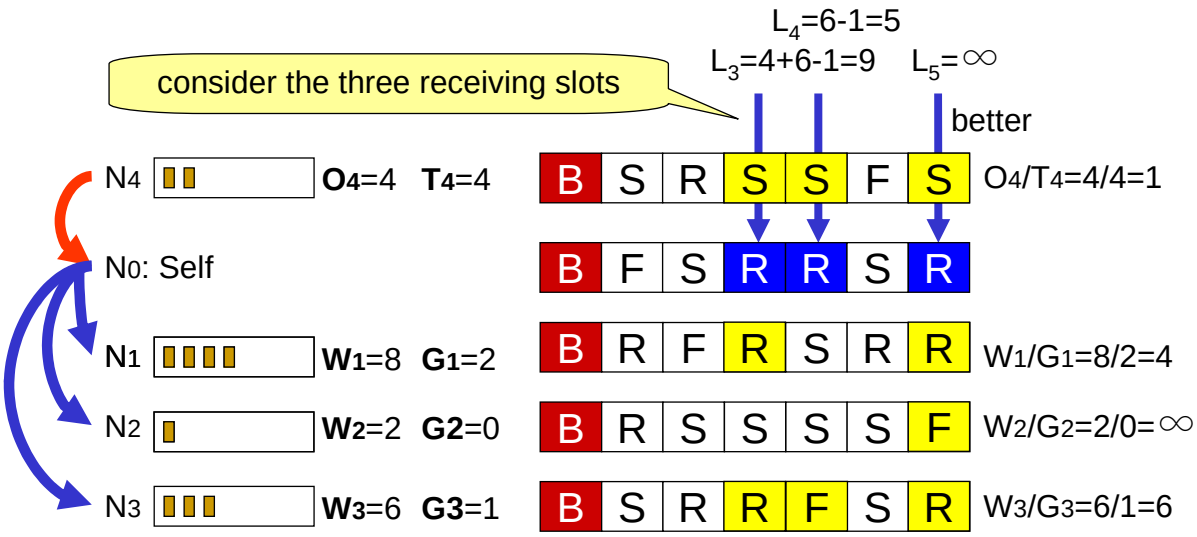

圖 10、增開傳送時槽範例

3-2-4、傳送端排程

(The Destination of Sending Slot Scheduling)

在接收時槽，只需靜待在自己的接收頻道上等待別人傳送資料，但在傳送時槽卻還需判斷要傳送給誰資料。在解決這個問題前，會先碰到我們用什麼資料結構來儲存將要送出去的封包，不同的資料結構會造成能判斷的能力不同。

我們用時槽來分割時間，但是封包並不是盲目的傳送出去，因為在某個傳送時槽，頻道是換到某個鄰居的接收頻道上，只有當鄰居是處在這個接收頻道上時才能把封包傳送出去，其它的封包必需用其它的緩衝區(佇列)暫存起來，這衍生出一個問題，用頻道佇列(per channel queue)還是鄰居佇列(per neighbor queue)來實作緩衝區會比較好？頻道佇列和鄰居佇列範例如下圖 11 所示，網格點 A 需暫存鄰居的資料，圖左為鄰居佇居，就是為每一個鄰居都開一個佇列來當緩衝區，圖右為頻道佇列，總頻道數固定時，佇列數也就固定，網格點 B 和網格點 E 使用相同的接收頻道 ch1，所以 B 和 E 必需共同一個緩衝區。使用頻道佇列和使用鄰居佇列各有一些優缺點，使用頻道佇列的優點是總頻道的數道固定，將來網路卡硬體可提供這類型的佇列，效能上會比較好。而使用鄰居佇列，當在自己的接收時槽時，可以視接收到的封包的傳送端為可溝通對象，把一些要回傳的封包從鄰居

佇列中拿出傳回去，但頻道佇列沒有辦法做到這點，因為他是以頻道為佇列，一定要等到傳送時槽到來換到該頻道，才會從該頻道佇列取出封包來傳送。

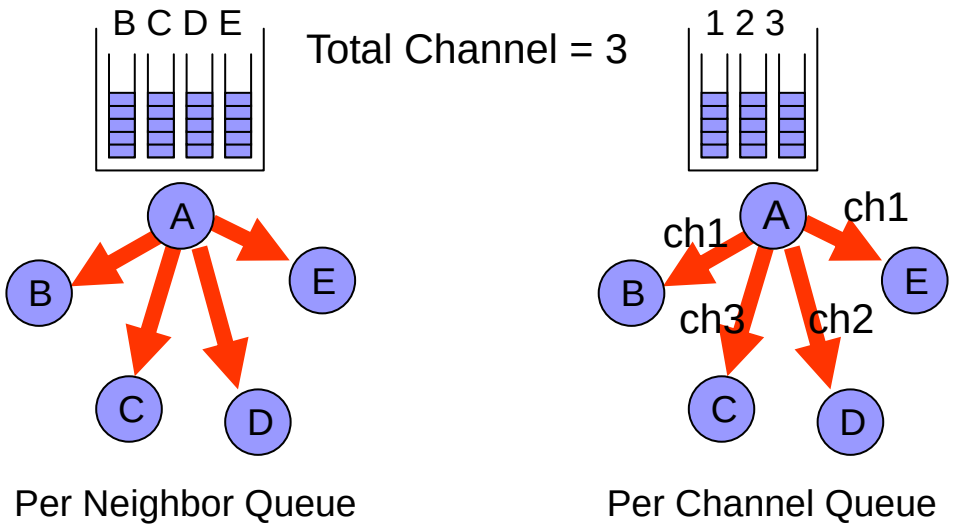

圖 11、鄰居佇列與頻道佇列

評估衡量的結果，我們決定以頻道佇列來實作，這關係到進入每一個傳送時槽選擇接收端(destination)的演算法，我們使用傳統的比重分配法(Weighted Round Robin Algorithm)，但這並不全適用於我們的架構之下，因為每一個傳送時槽，我們只能選擇正處理接收時槽的網格點來傳輸。我們為每一個頻道佇列設定一個比重(weight) Wi 和優先權(priority) Pi，這個比重和傳送接收時槽順序分配時用到的那個比重是一樣的。一開始每個頻道佇列的 Pi 相同，Wi 隨著網路流量的變化而改變，在無線網狀網路(mesh network)裡，這個變化量比較小。

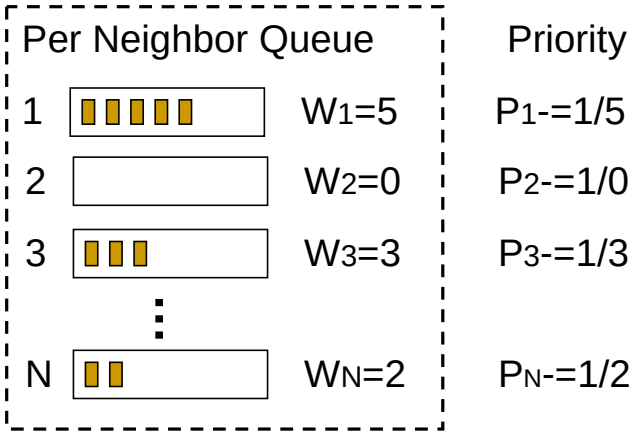

圖 12、傳送端排程及優先權

當進入每個傳送時槽時，先去查看排程資料表，得知那些鄰居正處於接收時槽，再比較這些鄰居誰的 Pi 最高，選擇這個鄰居當成接收端傳輸資料，當結束這個傳送時槽時，降低 Pi 的優先權 1/Wi，如此一來，Wi 重的鄰居優先權降的慢，能得到傳輸的機會比較多，而且一直沒有輪到可傳輸時槽的那些鄰居，因為沒有輪到機會傳輸，一輪到機會時也會因為 Pi 值比其它網格點高而競爭到傳輸的優先權，如圖 12，N_1 一次降 1/5 而 N_3 一次降 1/3，N_1 比較容易取得傳送權，但 N_3 也不會一直競爭不到傳送權。我們會週期性的把 Pi 重置(reset)，不會造成不公平的狀況發生。

四、實作

4-1、硬體和系統環境

為了在真的環境上面實作，查閱了相關的研究，去找尋有公開原始碼的驅動程式(open source driver)，Atheros 有公開晶片 Linux 驅動程式的原始碼，使用 Atheros 晶片的網卡都可以使用這個驅動程式來驅動。所以我們選擇了幾台筆記型電腦，作業系統均為 Linux Red Hat 9.0，每台上面均裝有 D-link DWL-AG650 的網路卡，它是使用 Atheros 的晶片，這讓我們可以修改公開的原始碼來把我們的管理協定實作在上面。

圖 13、D-link 網路卡

圖 14、實作環境

我們利用這些筆記型電腦當成是一個個的網格點(mesh point)，讓它們形成隨建即連網路(ad-hoc network)，並固定其位置模擬無線網狀網路(mesh network)，無線網狀網路和隨建即連網路有很大的共通性，差別在於無線網狀網路有閘道可以連上網際網路(internet)，且無線網狀網路沒有行動性(mobility)，我們讓這些筆記型電腦模擬一個無線網狀網路的雛形(prototype)來當成我們要的環境。

4-2、程式架構

我們的管理協定是屬於 OSI 七層網路模型的鏈結層(Data Link Layer)，而在 Linux 的網路架構下是簡易的分成四層，我們所要修改的是

最底層鏈結層。

下圖 15 為 Linux 使用者空間(user space)、核心空間(kernel space)和網路驅動程式(network driver)之間的關係，上層使用者空間的行程(user space processes)透過系統呼叫(system call)跟核心來通溝，而核心內行程的溝通都是使用函式呼叫(function call)的方式，在 Linux 的網路分層架構下，每一層之間都有佇列(queue)把出境封包暫存起來，等到有時間可以處理時，才使用函式呼叫去處理送出的動作，而外來的封包則是使用中斷(interrupt)的方式來通知驅動程式，驅動程式需處理鏈結層的一些必要動作，例如拿掉鏈結層的標頭(header)、過濾封包…，處理完後再使用軟體中斷(software interrupt)通知核心處理更上層的工作。

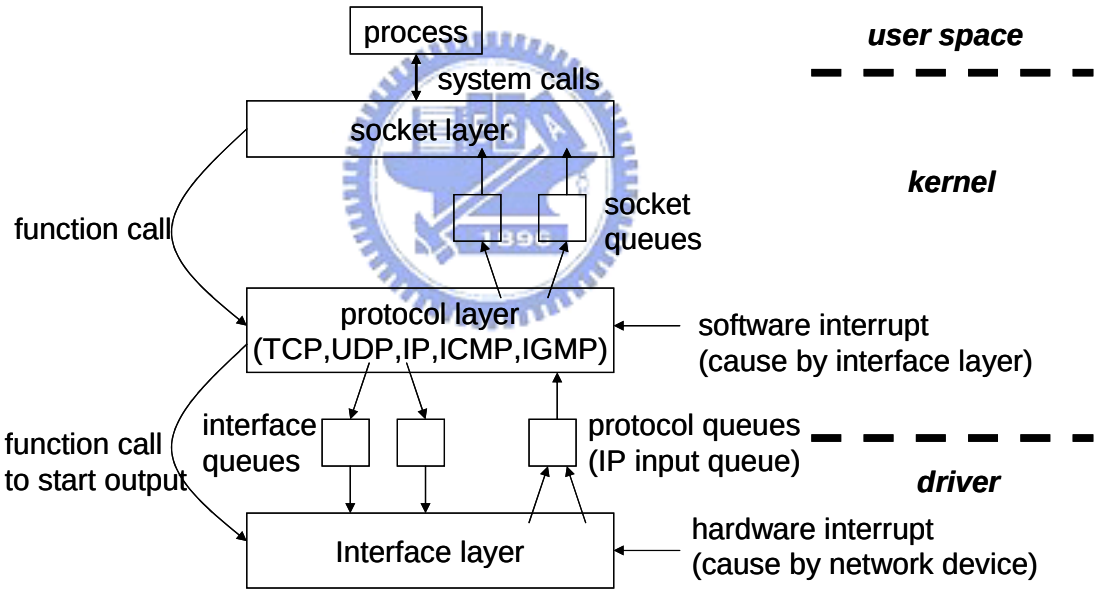

圖 15、 Linux User Space、Kernel 和 Network Driver 之間的關係

圖 16 是我們使用的 Madwifi 驅動程式和 Linux 核心之間的架構，驅動程式利用模組(module)方式掛載(mount)進入核心時，會跟核心註冊一些函式。當核心需要送出資料封包時，會先將資料排入「出境佇列」(outgoing queue)，然後呼叫網路介面的 hard_start_transmit()作業方法。在 Madwifi 驅動程式裡是註冊 ath_start()這個函式作為 hard_start_transmit()的函式指標。利用 request_irg()函式註冊 Madwifi 驅動程式的 ath_intr()為

中斷函式，而 netif_rx()是核心給驅動程式呼叫用來通知上層處理接收到的封包。

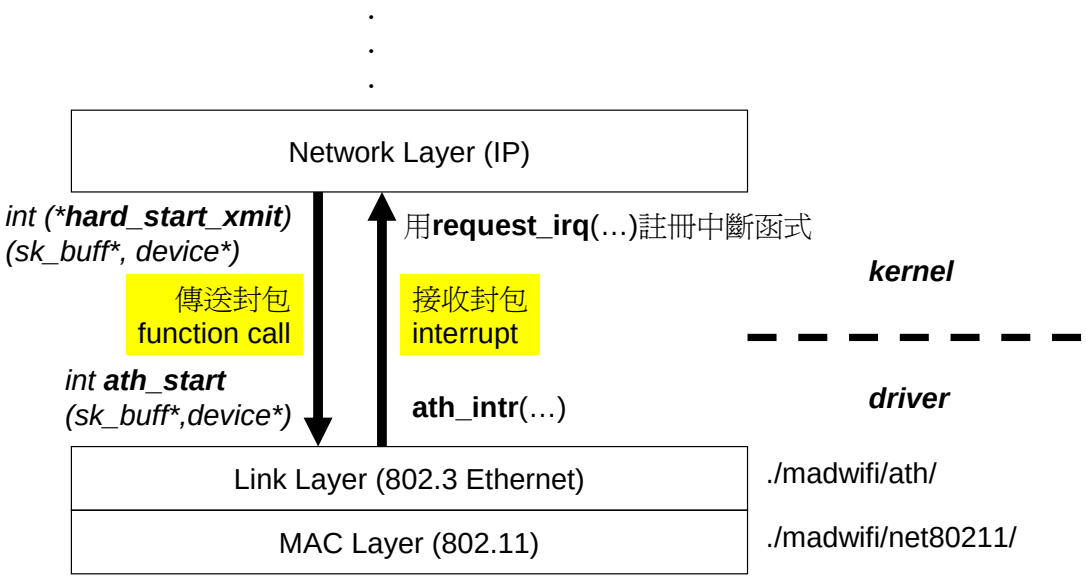

圖 16、Madwifi Driver 和 Linux Kernel 之間的架構

下圖 17 為對 Madwifi 程式我們所要修改新增的部分。當有封包要送出的時候，我們必須多加鄰居佇列(per neighbor queue)來暫存封包，分類的依據是檢查 MAC 位址，放進適當的佇列中，然後選擇一個此時該傳送的封包傳送出去，接著再觸發傳送下一個封包。當接收到封包時，檢查是否為時間同步或頻道排程的封包，如果是時間同步的封包則更新計時器，若是頻道排程的封包，則更新排程資料表，若都不是則交於上層去處理。新增切換頻道計時器，每 3 秒中斷一次，當切換頻道計時器中斷時，需去查詢排程資料表，如果進入的是廣播時槽(broadcast slot)，則切換至共同頻道上，然後檢查頻道使用情況是否該換接收頻道，也檢查網路流量是否需換排程，一定的週期需重設所有佇列的優先權，如果進入的是接收時槽，最簡單，就切換到接收頻道上去等待資料就好了，如果進入的是傳送時槽，必須依照佇列優先權和頻道排程選一個傳送端，並切換到此傳送端的接收頻道上。

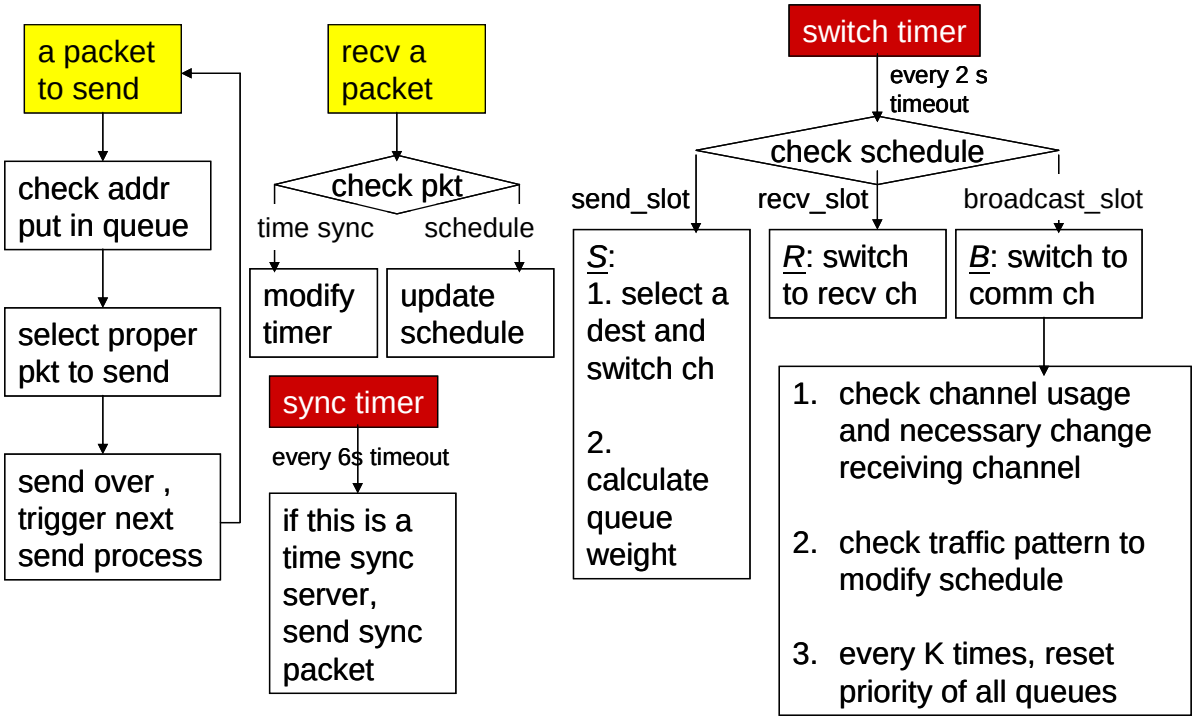

圖 17、程式架構圖

4-3、實作的過程與碰到的難題

萬事起頭難，驅動程式是屬於 Linux 核心的部分，跟一般我們碰到的程式限制多了不少，使用不當就會死當，挑戰變成很高，龐大的程式碼也讓我們不知從何下手，經過幾個月的努力查詢程式碼，並閱讀有關 Linux 網路驅動程式相關的書籍來封包了解，才漸漸了解 Madwifi 驅動程式的架構。

首先碰到的問題便是每 3 秒切換一次頻道的問題，因為 Linux 核心不會做置換(context switch)的動作，所以不用能 while 迴圈或 sleep 去實作，會耗盡 CPU 的時間，其它的工作都停滯不動，其它的行程也都受到牽涉。我們研究 Linux 核心可用的函式庫及語法，利用 timer_list 這個資料結構來實作，向核心註冊中斷時間和中斷服務函式，時間一到，核心會自動去呼叫註冊的中斷服務函式，就可以達成不吃 CPU 時間而且 5 秒中斷一次。我們最後成功的在驅動程式裡加入計時器中斷，完成分時的概念。

接下在我們在 Linux 核心和驅動程式間之間加新一層暫存區，把上層的封包用佇列(queue)暫存，等到適當的傳送時槽(sending slot)再傳送出去，佇列怎麼做？經過一番的努力，發現核心經手的每一個封包，都是包

裝成一個 struct sk_buff 結構，要把封包用佇列暫存，需要加上一個 sk_buff_head 把上層下來的封包串起來，便可行成一個佇列。分析封包要傳送的位置，便可利用多個 sk_buff_head 來實作成頻道佇列(per channel queue)或鄰居佇列(per neighbor queue)， 我們是以鄰居佇列為實作佇列的方式。

困擾著我們最久的是切換頻道的問題，我們試了諸多的切換頻道的方法，第一種，直接下 iwconfig 系統呼叫(system call)去切換頻道，是屬於使用者空間的方法，第二種，使用 Madwifi 驅動程式被 iwconfig 下命令著第一個呼叫的函式，並傳入相同的參數，第三種，在 Madwifi 驅動程式與 IEEE 802.11 無關的程式碼裡選一個最直接切換 channel 的函式去切換，發現一件事，切換頻道均沒有問題，但是同時兩個網格點同時換到同一個頻道時，有時連得起來，有時連不起來，有時連的很快，有時連的很慢，這對我們的管理協定有非常嚴重的效能折扣。第四種，我們直接去呼叫燒在網路卡的韌體函式去切換頻道，發現沒有效果，根本不能切換頻道。正當一切均失敗的時候，我去查閱了 IEEE 802.11 的書籍，找到 802.11 在 Ad-Hoc 模式下為了要同步的關係，需要產生信號彈(beacon)，我們發現，當一起切換到相同的頻道，當信號彈 BSSID 有合併的時候，網路才有辦法連線，所以我們去追蹤查詢，換頻道的時候會重新產生 BSSID，就是即使隨建即連(Ad-Hoc)網路的名稱相同，BSSID 也會不同，推測因為 BSSID 的不同，所以封包在過濾的時候被過濾掉了，所以連線連不起來，當 BSSID 合併才連得起來，我們查到網路卡支援一種 Ad-Hoc Demo 的模式可以把 BSSID 一直維持在 00:00:00:00:00:00，所有封包都會收進來，如此才成功的解決問題。

4-4、測試程式

我們的程式碼架構在驅動程式裡，為了監控驅動程式的狀態以及測試效能，我們在使用者空間透過共享記憶體的方式去監控驅動程式的狀態，並在上面使用 UDP 的封包進行頻寬的測試，測試畫面如下圖。

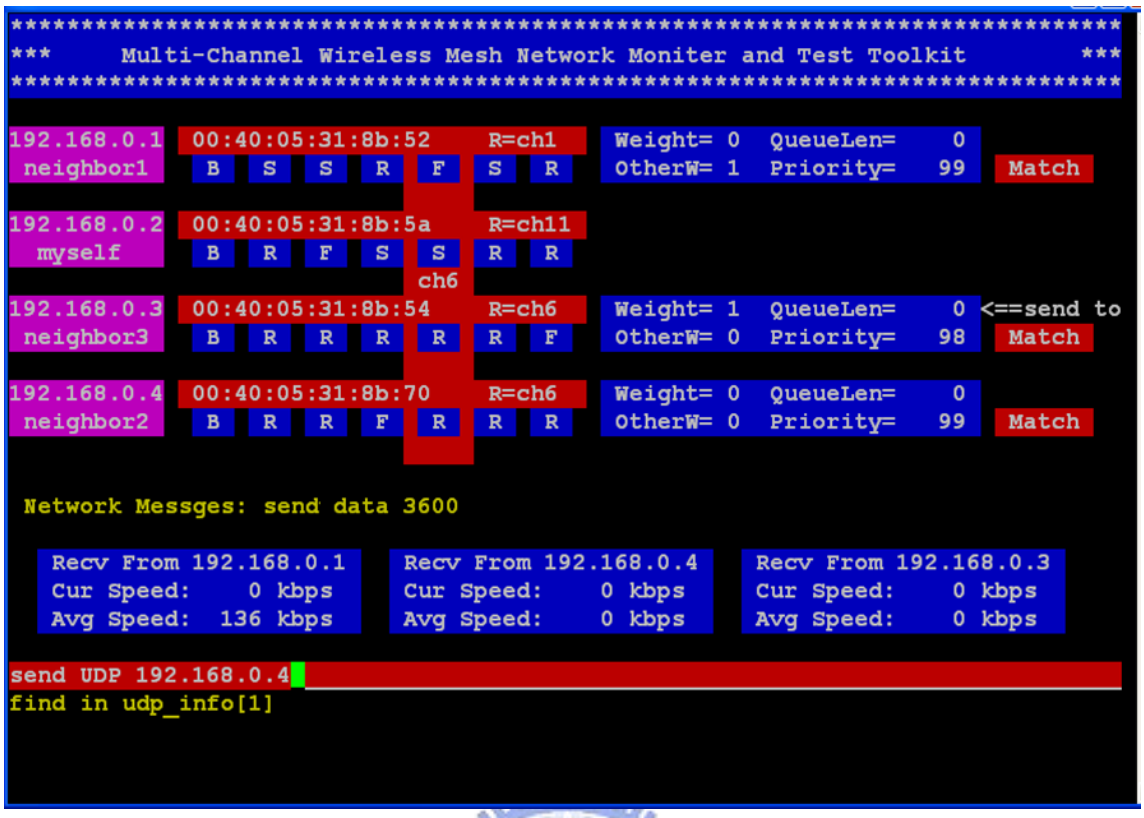

圖 18、測試監控程式

看圖 18，這是 192.168.0.2 筆記型電腦的測試監控程式的截取畫面，此存取會對其它鄰居紀錄一些資訊，驅動程式儲的資訊放在畫面中 Network Messages 的上方，例如排程、MAC 位址、比重(Weight)、對方對我的比重(OtherW)、目前佇列的長度(QueueLen)以及優先權(Priority)，為了實作上的方便，比重有就是 1，沒有就是 0，因為這個不是實作上的重點，所以用最簡單的方式。畫面中 R=ch1 表示網格點的接收頻道是頻道 1，一個個藍色小方框裡標著 B、R、F、S 是鄰居和自己網格點的排程，會有紅色的長條顯示目前在那一個時槽，而長條中標示的 ch6 表示自己目前這個時槽是用頻道 6，當自己在傳送時槽時而鄰居又剛好在接收時槽時，最右方會有”Match”的紅色標示，這個傳送時槽目前傳輸的對象會在最右邊標示”send to ”的字眼，以上是驅動程式的資訊，用共享記憶體的方式讀取出來顯示在畫面上。

在 Network Messages 的下方在網路層的監控和測試工具，在 Network Messages 可以看目前產生的封包序號或接收到的封包序號，而下方藍色大方框可以看目前網路層有接收到從那來的流量，Cur Speed 是目前這一秒鐘

的傳輸速度，而 Avg Speed 是過去一段時間傳輸速度的平均，因為我們有分時槽的關係，用 Cur Speed 來看傳輸速度是不準確的，有時是 700 KBps 而有時是 0 KBps，所以要加 Avg Speed 來測試頻寬。最下方的命令列可以用來下命令測試頻寬，例如”send UPD 192.168.0.4”可以起始一條新的流量(最佳傳輸的方法)給 192.168.0.4 這個網格點。

4-5、測試結果

我們利用測試監控程式來測試所設計的協定，下圖 19 為我們測試的第一個網路環境，四個網格點都在互相的干擾範圍內，A 使用最佳傳送的方式(Best Effort)傳送 UDP 封包給 B，而 C 也使用最佳傳送的方式傳送 UDP 封包給 D，由表 2 可看出，在相同未經修改的驅動程式下，使用不同頻道時的效能是使用同頻道的效能的 190%而已，但是 2 個連線此時使用調頻的方式是手動去改變頻道的方式，接著可以看到經過我們經改過的驅動程式(隨著不同環境自動調整頻道)的效能是單一頻道的 135%的效能。

圖 19、測試環境一

A-B 傳輸速度	C-D 傳輸速度	頻道使用	驅動程式
374KBps	358KBps	同頻道	未經修改的驅動程式
700KBps	698KBps	不同頻道	未經修改的驅動程式(手動調頻)
503KBps	486KBps	不同頻道	我們修改過的驅動程式

表 2、測試結果一

接著我們測試鏈狀的網路，如下圖 20。

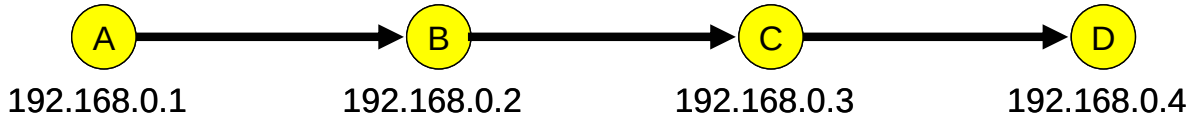

圖 20、測試環境二

在第二個測試環境下，因為我們無法把四台筆記型電腦拉開至幾百公尺之長來造成鏈型網路，所以我們透過寫死繞送路徑(Routing Path)的方式來達成這個效果，此時 A 使用最佳傳送的方法傳送 UDP 封包至 D，會經過 B 和 C 最後才到達 D。下圖 21 為四個網格點在此環境下排程變化的最終結果，可以看到 A 有三個時槽使用頻道 11 傳送到 B，同一時刻，C 也利用這些時槽使用頻道 1 傳送到 D，此時便有使用到多重頻道帶來的好處，增進鏈狀傳輸的效能。由表 3 可看出在我們的多重頻道協定是單一頻道效能的 122%。

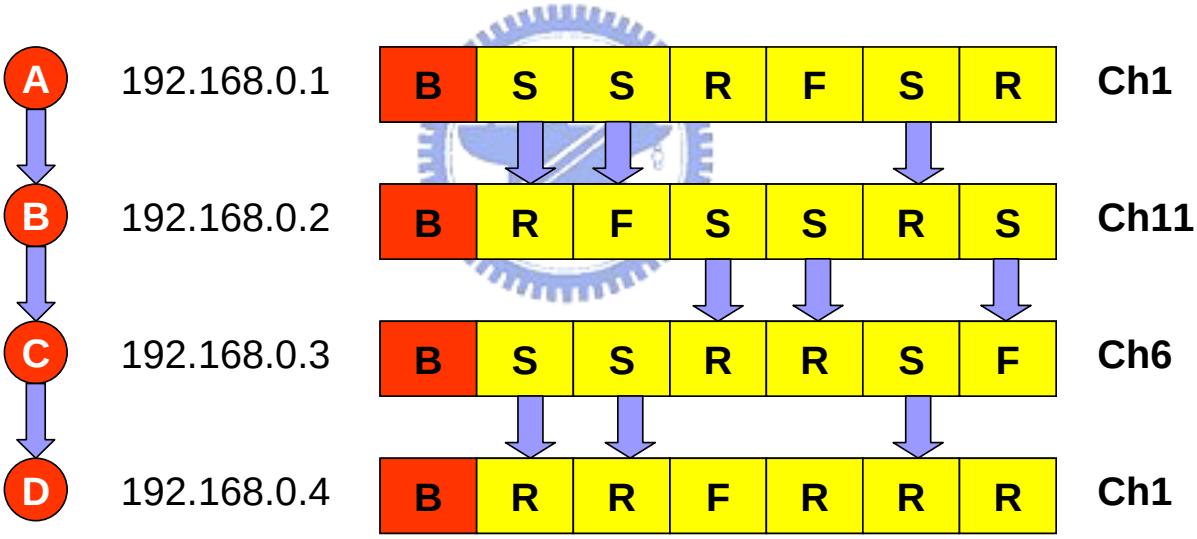

圖 21、測試環境下二的排程結果

A-B-C-D 傳輸速度	頻道使用	驅動程式
245KBps	單一頻道	未經修改的驅動程式
300KBps	多重頻道	我們修改過的驅動程式

表 3、測試結果二

五、未來展望與結論

5-1、本文研究貢獻

我們提出了一個可套用在 IEEE 802.11 MAC 層上的多重頻道管理協定，透過不斷的切換頻道提高空間中頻道的再使用率(Spatial Reuse)，只需單張網路卡便能達成這個效果，並且容易擴充至多張網路卡，因為在同一網格點上的每張網路卡都可使用不同的接收頻道，每張網路卡均個別使用我們的協定就可以降低干擾，並不需要另外使用其它協定來處理。

我們的研究並不是紙上談兵，因為我們模擬真正的環境去修改網路卡的驅動程式，把我們的管理協定實作在上面，並寫了測試程式去監控是否運作的正常，有成功的在實機上面套用我們的協定。

5-2、未來的研究方向

在我們提出的架構之下，衍生出了很多議題，像是傳送接收時槽比例的分配、傳送接收時槽順序的分配、傳送端的選擇，每個議題都是可以再繼續深入研究最佳化的方法，例如傳送接收時槽比例在分配時，α、β穩定的參數如果調整才能達到最好的效果，既穩定又能配合流量來分例。另外，廣播時槽和其它時槽的比例和時槽的總數也是另一個研究的議題，怎麼樣分配才不會造成時間的浪費，又不會造成廣播時槽內頻道擁塞(channel congestion)的問題。

在實作的方面可以擴大其規模，用更多的網格點或用多的頻道來測試我們的管理協定，適當的修改，讓它能夠更成熟更穩定。

六、參考文獻

[1] Shih-Lin Wu, Chih-Yu Lin, Yu-Chee Tseng, and Jang-Ping Sheu , “A New Multi-Channel MAC Protocol with On-Demand Channel Assignment for Multi-Hop Mobile Ad Hoc Networks” in I-SPAN, 2000.

[2] Ashish Raniwala, Kartik Gopalan, and Tzi-cker Chiueh, “Centralized Channel Assignment and Routing Algorithms for Multi-Channel Wireless Mesh Networks,” Mobile Computing and Communications Review, vol. 8, no. 2, pp. 50－65, April 2004.

[3] Paramvir Bahl, Ranveer Chandra, and John Dunagan, “SSCH: Slotted Seeded Channel Hopping for Capacity Improvement in IEEE 802.11 Ad-Hoc Wireless Networks,” in ACM Mobicom, 2004.

[4] Pradeep Kyasanur and Nitin H. Vaidya, "Routing and Interface Assignment in Multi-Channel Multi-Interface Wireless Networks", in WCNC 2005.

[5] Pradeep Kyasanur and Nitin H. Vaidya, "Routing in Multi-Channel Multi-Interface Ad-Hoc Wireless Networks", Technical Report, December 2004

[6] Richard Draves, Jitendra Padhye, and Brian Zill, “Routing in Multi- Radio, Multi-Hop Wireless Mesh Networks,” in ACM Mobicom, 2004.

[7] Sheng-Hsuan Hsu, Ching-Chi Hsu, Shun-Shii Lin, and Ferng-Ching Lin “A Multi-channel MAC Protocol Using Maximal Matching for Ad Hoc Networks” in ICDCSW, 2004.

[8] Ashish Raniwala, and Tzi-cker Chiueh , “Architecture and Algorithms for an IEEE 802.11-Based Multi-Channel Wireless Mesh Network” in Infocom, 2005.

[9] Jeremy Elson, Lewis Girod and Deborah Estrin “Fine-Grained Network Time Synchronization using Reference Broadcasts” in

OSDI 2002

[10] Romit Roy Choudhury, Xue Yang, Ram Ramanathan, Nitin H. Vaidya "Using Directional Antennas for Medium Access Control in Ad-hoc Network" in ACM Mobicom , 2005.

[11] Leiming Xu, Young Xiang, and Meillin Shi "On the Problem of Channel Assignment for Multi-NIC Multihop Wireless Networks" will appear in MSN, 2005.

[12] Sheng-Hsuan Hsu, Ching-Chi Hsu, Shun-Shii Lin, and Ferng-Ching Lin, "A Multi-Channel Mac Protocol Using Maximal Matching for Ad Hoc Networks" in ICDCSW, 2004.

[13] Michelle X. Gong and Scott F. Midkiff, "Distributed Channel Assignment Protocols a Cross-Layer Approach" in WCNC, 2005.

[14] 林炳榕,「新世代無線區域網路架構與技術」, Master Thesis, 2004