國立交通大學

資訊科學與工程研究所

碩 士 論 文

IEEE 802.11a 以及 IEEE 802.11p 車間通訊網路效能之比較

Performance comparison between IEEE 802.11(a)-based and IEEE 802.11(p)-based wireless vehicular networks.

研 究 生：洪維駿

指導教授：王協源　教授

中 華 民 國　九 十 八 年 六 月

IEEE 802.11a 以及 IEEE 802.11p 車間通訊網路效能之比較

Performance comparison between IEEE 802.11a-based and IEEE 802.11p-based wireless vehicular networks

研 究 生：洪維駿　　Student：Wei-Jyun Hong

指導教授：王協源　　Advisor：Shie-Yuan Wang

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 2009

Hsinchu, Taiwan, Republic of China

中華民國九十八年六月

中文摘要

IEEE 802.11p/1609 標準為了讓車與車 (V2V) 之間的 ad-hoc 網路做訊息的交換而定義了 WIBSS (WAVE Independent Basic Service Set) 。但是，這種設計無法善用 802.11p/1609 網路 multi-channel 的特性。如果我們想使用 multi-channel 勢必要有一種機制來溝通並且同步 nodes 所使用的 channel，但這可能會增加 MAC layer 設計的複雜度。

有鑑於此，我們找設計一種更為簡單的方法來解決 channel 同步的問題。本篇論文首先利用模擬的方法來觀察常用的 ad-hoc 路由協定 (AODV，DSDV) 在 802.11p/1609 網路上的效能。在我們的模擬結果中發現，如果將一般路由協定所傳送/廣播的控制訊息藉著 WBSS (WAVE Basic Service Set) 的建立來傳送，將造成路由協定的效能低落。為了解決此問題我們最終將這些路由協定所傳送/廣播的控制訊息以 WSM 格式來包裝傳送。另外為了將封包正確的送達收端，我們在 WBSS-based 網路採用了兩種 MAC 層的 multi-hop forwarding 機制來幫助 WBSS 的建立。最後我們比較這兩種機制在 802.11p/1609 網路下運行的效能，並以 802.11a 網路為基準做比較。

我們的模擬結果顯示，在 802.11p/1609 networks 所運作的 routing protocol 效能上與 802.11a ad-hoc networks 並沒有太大的差異。這是由於我們將所有路由協定所 傳送/廣播 的控制訊息利用 WSM 的格式來發送。另外，雖然 WBSS-based V2V networks 相較於 802.11a 網路只能利用一半的頻寬來傳輸資料，但是因為運用到為 802.11p/1609 網路所設計的 multi-channel，在某些情況下傳輸資料的效率反而優於 802.11a ad-hoc networks。

關鍵字：車間通訊網路，802.11p/1609 網路，路由協定，無線網路，ad-hoc，multi-hop forwarding，802.11a networks。

ABSTRACT

The IEEE 802.11p/1609 standards define the WAVE Independent basic service set (WIBSS) for vehicular ad-hoc networks to carry out vehicle-to-vehicle (V2V) communication. However, the design of a WIBSS cannot utilize the multi-channel property of an 802.11p/1609 network. To make use of these channels, an extra protocol is needed to negotiate and synchronize the channel used among nodes, which may complicate the design of the MAC layer.

These observations motivated us to find an easy-to-implement solution for it.In this thesis, we first observed the performances of several common routing protocols (such as, AODV and DSDV) in 802.11p/1609 networks using simulations. From our simulation results, we concluded that transmitting/broadcasting control messages of routing protocols in WAVE Basic Service Sets (WBSSs) is inefficient for routing protocols. To solve this problem, we propose a framework that transmits/broadcasts routing control messages using WSMs while data are still transmitted in WBSSs. Based on this framework, we further propose two MAC-layer WBSS-creating schemes to realize multi-hop data forwarding in WBSS-based networks. We compare the performances of 802.11p/1609 networks using these two WBSS-creating schemes with those of 802.11a networks.

Our simulation results show that the performances of routing protocols under 802.11p/1609 networks can be the same as those over 802.11a networks as long as in 802.11p/1609 networks routing control messages can be transmitted using WSMs. Because 802.11p/1609 networks only use a half of link bandwidth to transmit data, the bandwidth that can be used to transmit data is a half of the bandwidth that can be used in 802.11a networks. However, due to the multi-channel design WBSS-based V2V networks, can outperform 802.11a ad-hoc networks in several conditions.

Keywords: Vehicle networks, 802.11p/1609 networks, routing protocol, wireless networks, Ad-hoc, multi-hop forwarding, 802.11a networks.

致謝

感謝指導教授王協源在這兩年內對我的悉心教導與照顧，無論是在課業或是待人處事方面都讓我獲益良多。另外，感謝周智良學長以及林志哲學長，學長們給我的建議以及幫助讓我的碩士論文可以更加的完善。

接下來，感謝口試委員藍崑展教授、鄭振牟教授以及陳裕賢教授特地撥冗前來聽取我的論文報告並且給予建議，讓我可以從教授們的觀點來得知論文不足之處。

感謝家人對我的支持與照顧，讓我可以順利的走完這兩年，而不留下任何遺憾。

最後，感謝實驗室的所有成員，大家的相互幫助以及扶持，充實了我這兩年的碩士生活。

目錄

圖目錄

表目錄

Chapter 1 Introduction

在無線網路的發展領域中 IEEE 802.11(a)(b)(g) 已經相當成熟，但是考量到現今車間通訊網路的需求，舊有的 IEEE 802.11 標準並不適合用在這類型的網路，因此由 IEEE 802.11 團隊所制定的 IEEE 802.11p [1][2] 應運而生。車間通訊網路除了較底層的 IEEE 802.11p 其上層還包括 IEEE 1609 [3][4][5][6][7][8][9] 系列，我們將其通稱為 WAVE (Wireless Access in the Vehicular Environment) standards。另外，車間通訊網路也可稱為 WAVE network。

WAVE network 有兩種模式，其一為 Infrastructure mode，在一般的情形下，通常會由 RSU/OBU (roadside unit/onboard unit) 建立 WBSS，OBU 便可藉由加入此 WBSS (WAVE Basic Service Set) 來存取 802.11p/1609 網路。通常建立 WBSS 的 WAVE device 稱之為 provider 而加入 WBSS 的 WAVE device 稱之為 user。WBSS 的建立或是加入是為了交換一般的 IP 封包，在沒有加入或是建立 WBSS 的狀態下 WAVE device 就只能利用 WAVE standards 所制定的 WSM (WAVE short message) 來交換資料。

IEEE 802.11p 另外定義了一種 ad-hoc mode 的網路存取方式。在這種網路上的所有的節點都是 OBU，OBU 與 OBU 之間透過 WIBSS (WAVE independent Basic service set) 的建立來交換訊息。WIBSS 的建立不需要發送 beacons 就可以直接開始傳輸資料，意即 OBU 可以自由的交換訊息。

如果 WAVE network 的 ad-hoc mode 只是單純的使用上述的方式來達成，那將無法善用到 WAVE network 所擁有的眾多 channels，這是因為 WIBSS 在建立之前並沒有發送 beacons 就直接傳輸資料。在上述的情形下，送端以及收端必須一開始就停留在相同的 channel 才能互相通訊。另外，如果送端和收端距離很遠必須要靠中介點 (forwarder) 來幫助封包的傳送那麼 forwarder 也需要固定在此 channel 上。因此，如果想要透過 WIBSS 的建立來交換訊息最保險的方式便是所有的 node 一開始就留在相同的 channel。

如果我們想要在 WAVE network 的 ad-hoc mode 上使用 multi-channel 來交換訊息也是可以的，但這需要一種機制來溝通與同步 node 與 node 之間的 channel。如此勢必會增加 MAC layer 設計與實作的複雜度。

有鑒於此，本篇論文的目的便是要找出一種簡單得方法來解決上述的問題，並且使用 multi-channel 來傳輸資料。經過我們的研究後發現， WAVE standard 所定義的 WBSS mode 便已經內建了 channel 同步的機制。WBSS 所使用的方法如下，provider (WBSS 的建立者) 在 control channel(CCH) interval 時發送 beacon 通知 user (WBSS 的加入者) 之後的 service channel (SCH) interval 双方要到哪一個 SCH 交換訊息。在此機制下 provider 和 user 會在 CCH interval 交換控制訊息並做 SCH 的同步，之後到達 SCH interval 時便轉換到同一個 SCH。WAVE device 會持續的在 CCH 與 SCH 間切換。

但是這樣還不足以達到類似 WAVE ad-hoc mode 的能力。如果送端和收端需要中介點的幫助來交換訊息，送端必須先透過 WBSS 的建立將封包送至中介點之後再經過數次 WBSS 的建立與加入最後才將封包送達收端。每一次的 WBSS 建立與加入都需要有相對應的 provider 與 user，因此我們需要一種機制來做 provider 與 user 的選擇。我們最終選擇 [10] 所設計的機制 (multi-hop forwarding)，並且修改它以適用在我們所建立的網路環境 (WBSS-based V2V networks) 之上。

另外為了在 WBSS-based V2V network 上交換訊息我們還需要 routing protocol (AODV [11] 以及 DSDV [12]) 的幫助，為我們選擇封包的傳送路徑。但是 routing protocol 的運行必須要 傳送/廣播 IP 封包來尋找路徑。而在我們的 WBSS-based 架構中傳送 IP 封包必須要建立或是加入 WBSS。如果大家都想廣播 IP 封包，那麼所有的 node 都需要建立 WBSS，另一方面如果想要收到這些廣播的 IP 封包，所有的 node 也都需要加入其他 node 所建立的 WBSS。最後這個 WBSS 又該由誰建立呢？為了解決這個問題我們將 routing protocol 所廣播的 IP 封包都封裝成 WSM，並且將其放在 CCH interval 上傳送。如此我們便不需要考

慮到 WBSS 建立的問題。

透過 WBSS 的建立我們可以自由的運用到 WAVE network 上所有 channel，但是為了讓收端與送端間溝通好下次要到哪個 SCH 來傳送資料，OBU 必須要常常在 CCH 以及不同 SCH 間切換，對於資料傳送的 delay 也會比較大，當然也只能利用到一半的時間來傳輸資料。相較於原本 WAVE ad-hoc mode 透過 WIBSS 的建立來交換訊息，本篇論文所使用機制卻可以讓資料分散在不同 channel 上傳送，使得 WAVE network 理論上可以多出 3 倍的頻寬。這是因為 WBSS-based V2V networks 可使用的 channel 資源為 WAVE ad-hoc mode 的 6 倍，但 WBSS-based V2V networks 能利用 SCH interval 來交換訊息，可使用的時間資源為 WAVE ad-hoc mode 的 0.5 倍。

WSM 是 WAVE standard 專門為 WAVE network 所定義。WSM 的能力相當強大，applications 在傳送 WSM 時可以直接指定要在哪一個 channel 上傳送 WSM。另外 WSM 也可以在任何的 channel 上傳送，比起只能在 SCH 上傳送的 IP data 顯然 WSM 的運用可以相當的靈活。如果上層所有的資料都利用 WSM 的格式來傳送，我們就可以直接利用到所有的 channel 資源。但是如果將所有的資料包裝成 WSM 的格式來傳送，那麼在此之前利用 IP 封包來傳送資料的 application 將要全部改寫，這顯然不太可行。

Chapter 2 Related Work

自從 WAVE Standard 被發表以來，在其上的研究便不曾間斷過。其中，有一部分的研究是注重在 IEEE 802.11p 以及 IEEE 1609.4。這部分的研究偏向較底層的資料傳輸效能以及 WAVE MAC 的特性，其中還可分成非安全性相關 [13][14] 與安全性相關 [15] 的研究。[13] 的研究內容是在 802.11p 環境下測量封包的碰撞機率，傳輸 delay 以及 throughput。[14] 的研究內容是改善 WAVE MAC layer 的 protocol，使得在其上運作的 application 可以達到更佳的封包傳送效率。[15] 的研究內容是 MAC layer 利用自己所設計的演算法有效率的將重要的安全性訊息 broadcast 到整個網路。

有一部分的研究是專注在較上層 IEEE 1609.1 以及 IEEE 1609.3 與安全性向關的研究題目 [16][17]，另外也有研究題目是與 WBSS 相關的論文 [18]。[16] 的研究是利用 roadside unit 來收集車輛所傳出來的訊息，並且利用這些訊息來分析每兩台車輛的相對移動方向以及位置，最後計算兩台車是否有發生碰撞的可能性。[17] 的研究是改善車輛碰撞偵測系統的效能，這種系統是利用 GPS 服務來得知車輛位置，並且與附近車輛交換這些 GPS 訊息，藉以分析自己是否有遭受碰撞的危險。[18] 的研究是在 WBSS 的架構下建立一種封包傳送的機制，使得每個封包的發送者都可以享有較高而且穩定的 throughput。

我們所做的研究與 [10] 的研究有相當大的相關性，但其中的不同點在於 [10] 所設計的 multi-hop forwarding 演算法與上層的 routing protocol 必須有非常密切的關聯，而且 [10] 所使用的 topology 上必須要有 RSU 的存在。我們改善了 [10] 所使用的 multi-hop forwarding 機制，並且適當的運用在我們所支持的 WBSS-based V2V networks，並提升了 WAVE network 傳輸資料的效能與自由度。

Chapter 3 Background

3.1 Wireless Access in the Vehicular Environment

IEEE 802.11p (又稱 WAVE；Wireless Access in the Vehicular Environment) 標準，是由 IEEE 802.11 標準所擴充的通訊協定，這個通訊協定主要是使用在車用電子上的無線通訊系統。WAVE 是從 IEEE 802.11 標準來擴充延伸，來符合智慧型運輸系統(ITS：Intelligent Transportation System)的相關應用，應用的層面包含高速率的車輛與車輛 (Vehicular-to-Vehicular：V2V) 以及車輛與路邊基礎設施(Vehicular to Roadside Unit：V2R) 間的封包交換，其使用的波段為 5.9 千兆赫(5.85-5.925 千兆赫)，而 IEEE 1609 標準則是以 IEEE 802.11p 通訊協定為基礎的高層標準。

在 IEEE 801.11p/IEEE 1609 標準中詳細制定了 WAVE device 中 protocol 層面的溝通方式，其架構如圖 3.1 所示。IEEE 1609.0 定義的整個 WAVE 系統的架構，參考圖 3.1，它將整個 WAVE 系統分成兩大塊，其中一塊是數據平面 (Data Plane)，有 WSMP (Wave Short Message Protocol)，WAVE MAC 以及 WAVE PHY，另為一塊是管理平面 (Management Plane)，主要由 WME (WAVE management entity) 所構成。本篇論文的所探討的範圍主要為 IEEE 1609.0，IEEE 1609.3 以及 IEEE 1609.4。

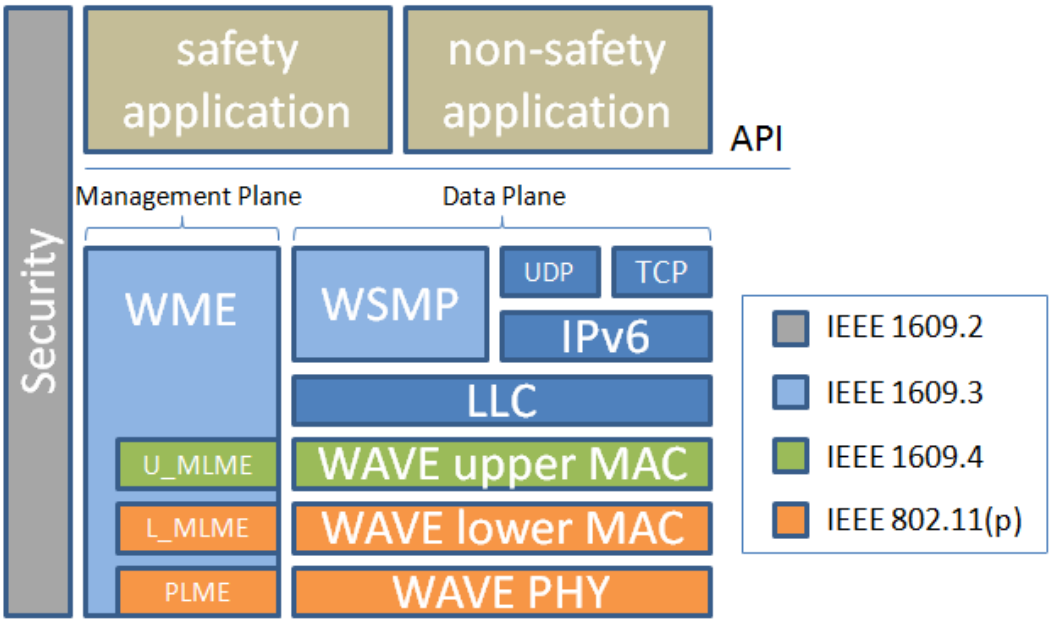


圖 3.1 The Protocol Stack of an IEEE 802.11p/1609 Network

3.1.1 IEEE 1609.0

IEEE 1609.0 規劃了整個 WAVE 系統功能以及的規則，在此分成 WAVE 系統概要以及 WAVE 系統操作兩部分來介紹。

WAVE 系統概要：

- WAVE device 以及 WAVE service

  WAVE device 分成車載裝置 (onboard unit：OBU) 以及路邊基礎設施 (roadside unit：RSU)兩種。WAVE service 是由應用程式 (applications) 透過 WAVE device 所提供。提供 service 的裝置稱做提供者 (provider)，使用 service 的裝置則稱為使用者 (user)。當 applications 下的 WAVE device 形成 provider 以及 user 的關係必定是兩個 WAVE device 上的 applications 有交換資料的需求。另外 OBU/RSU 的身分可以是 provider 或是 user。

- 頻道種類

  在 WAVE 系統中有兩種頻道可以使用，一個管理頻道 (control channel：CCH)以及六個服務頻道 (service channel：SCH)。WAVE device 預設會停留在

CCH 上收送 Wave Short Message (WSM) 以及系統控制訊息(service 廣播：service advertisement)，SCH 則是用來收送一般的應用層資料 (IP data) 封包。

- 通訊協定

  WAVE 系統提供 WSMP 以及 IPv6 兩種資料交換的通訊協定，WSM 可以在任何 WAVE 所定義的頻道上傳輸，IP 封包的傳輸則只能在 SCH 上。WSMP 提供送端 (source) application 直接控制 WAVE PHY 的能力 (例如：發送頻道，傳送功率 …)，WSM 的收端 (destination) 則是根據 WSM 所帶的 Provider Service Identifier (PSID) 來判斷要送給哪一個 application。每一個 PSID 代表著不同的 service 類別，它是由 IEEE Registration Authority 所管理。WSMs 是專門為 WAVE 所設計。

- 通訊服務

  Application 可以自行選擇是否要使用 WAVE services 來交換資料封包，如果使用的話 application 就可以在 SCH 上交換一般的 IP 封包，否則 application 就只能在 CCH 上利用 WSM 來交換資料。

- Device 規則

  一個 WAVE deivce 在某一個 service 中只能扮演 provider 或是 user 其中一個角色，至於扮演哪一種角色則是由 application 來決定。Provider 必須定時 (以每個 CCH interval 為單位) 發出 service advertisement 並且選定一個 SCH 作為資料傳輸所用。User 則根據 CCH 上收到的 service advertisement 來決定是否要加入這個 service 。

- 優先權

  在 WAVE device 上有兩種類型的優先權

  - Application 有自己的優先權等級，主要是利用在網路的 service 上，這可以用來讓 user 選擇高優先權的 service。
  - WAVE MAC 對於不同的封包給定其不同的優先權，IP 封包的優先權以發送的 application 為依據來決定，WSM 封包則是以每個封包為單位來

給定優先權，即使發送不同 WSM 封包的 application 為同一個。

- 頻道同步

當 WAVE device 間要使用 SCH 來傳輸資料時必須同步頻道切換的時間，它會利用一個大家都可以存取到的時間資源來做同步，例如：由全球定位系統 (Global Positioning System：GPS) 所提供的國際標準時間 (Universal Time Coordinated：UTC) 。被同步的間隔 (interval) 如圖 3.2 所示，其中包含一個 CCH interval 後面再接著一個 SCH interval。在 CCH interval 時 WAVE device 監聽著 CCH，而在 SCH 時加入某 service 的 WAVE device 就可以利用 SCH 來交換訊息。WAVE device 一旦切換到 SCH 就無法收到 CCH 上的封包。

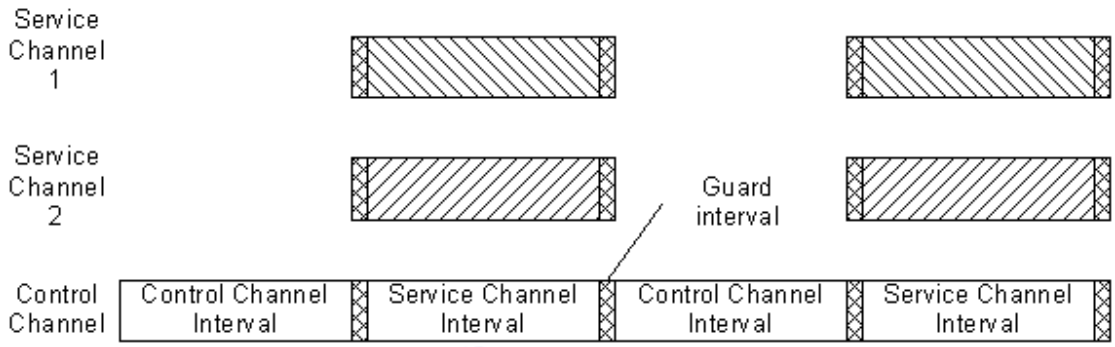


圖 3.2 The Bandwidth Division of an 802.11p/1609 Network

- 服務開始

WAVE service 的起始是由某個 provider application 向 WME 發出要求，provider 要把選定的 SCH 資訊加在 advertisement 中。如果選擇的 service 是持續的 (persistent) 那麼發送出 advertisement 的 destination MAC address 必須是廣播的 (broadcast) 而且要決定每個 CCH interval 要發送多少次 (repeats) advertisement。如果不是 persistent service 則 destination 的 MAC address 可以是 broadcast 或是單點傳播 (unicast)，而且只有在第一個 CCH interval 才需要發送一次 advertisement。

WAVE device 在收到 advertisement 後根據其內的 PSID 來判斷是哪種 provider application，並且檢察 service 的優先權 (priority)，認證 (credentials) 等等。如果條件符合那麼 WME 就會告知 MAC 下次 SCH interval 要轉換到 advertisement 所提供的 SCH。

Service advertisement 的 destination 不需要發送確認 (acknowledgement：ACK) 給 source，所以 provider application 不會知道有哪些 WAVE device 收到了 advertisement，也不會知道有哪些 WAVE device 成為 user 加入了這個 service。

WAVE 系統操作：

- 無 service 狀態下運行

  WAVE device 在沒有建立 service 的情況下就只能使用 WSMs 在 CCH 上交換資料。以下從 source 以及 destination application 的實際運行的情境來介紹。

  - Source 把想送出的資料準備好之後設定其目地的 MAC address 為 broadcast MAC address，之後設定 WSMP 的參數。透過 WME 所提供的基本操作 (primitive) WSM-WaveShortMEssage.request 把資料以及參數告知 WME。WME 會幫助 application 將 WSM 送到底層，最後資料會以 WSMP 封包的形式在 CCH 上傳送。
  - Destination 收到 WSM 後，WSMP 會跟據其中所帶的 PSID 來將 WSM 轉送給正確的 application，同時 destination 也會知道 source 的 WAVE device 位置。之後 destination 就可以根據此來發送封包，MAC address 則可以選擇用 unicast 或是 broadcast MAC address。

- 在 service 下運行

  WAVE service 是利用時間以及 channel 所提供的資源來分配給這些 WAVE device 所建立的 service。WAVE service 的起始是由某個 application 向 WAVE device 發出請求，並且在 CCH 上發送 advertisement 宣告 service

的成立。

在之前有提到 WAVE service 分成 persistent 以及 non-persistent 兩種，persistent service 為持續性的 service (例如：路況報告)，non-persistent service 則為臨時性的 service (例如：碰裝偵測通知)。

WAVE advertisement 為 application 用來發佈 provider service 建立的公告，其組成步驟如圖 3.3 所示。所有 provider service 的相關訊息都包含在由 WME 所提供的 WAVE Service Advertisement (WSA)。

上層的協定 (Higher layer) 會監視由 WME 維護的有效的服務 (available service) 選出其中有興趣的來加入。

WAVE device 停止提供 service 時，它並不會發出任何的訊息來告知其他 WAVE device。它只會在 device 內部利用 primitive 告知 WME 結束 service。User 端的 WME 會利用一些機制來得知 service 已經不存在了，並將此 service 從 available service 列表中刪除。

不同的 service 可以在選擇同一個 SCH 來交換資料，而且在 service 的運作期間可以切換到其他的 SCH 來傳送封包，只要把取代的 SCH 填在 WAVE advertisement 告知其他 WAVE device 即可。

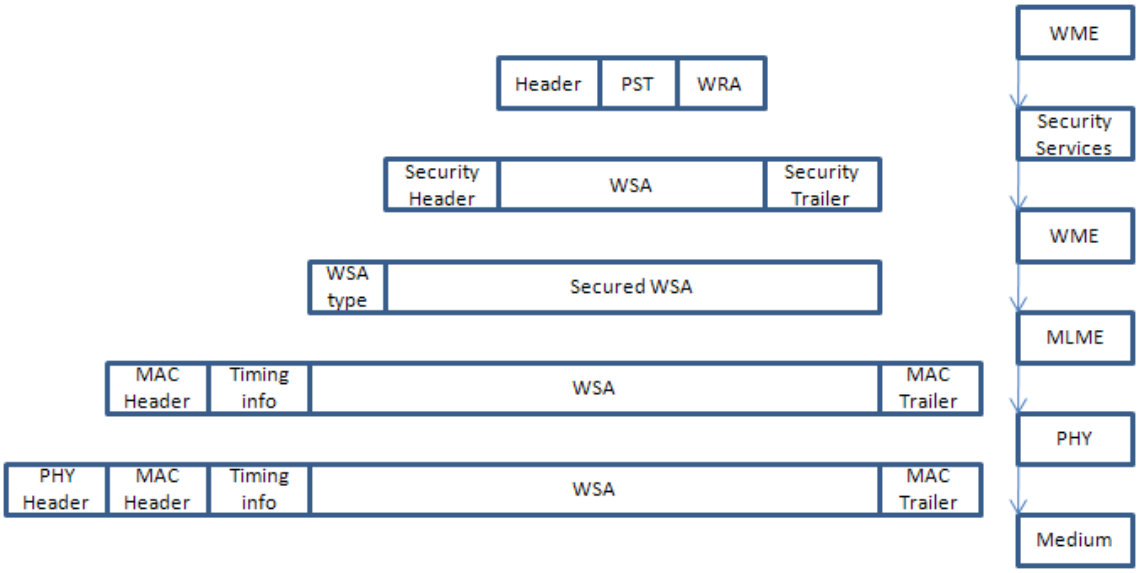


圖 3.3 The Message Encapsulation for a WAVE Service Advertisement (WSA)

- RSU/OBU 上的發散系統 (Distribution System)

RSU/OBU 不僅可以在 WAVE 網路上傳送封包它還可以透過固網連接其他 WAVE device，透過其他的裝置在和遠端電腦溝通如圖 3.4 所示。這代表著遠端的電腦不但可以用任何型態的網路來存取 RSU/OBU 上的封包，甚至可以提供或是使用 WAVE service，只要中介的裝置支援即可。此項功能賦予使用者極大的便利性，以中央氣象局所提供的氣象預報 service 為例，中央氣象局不需要為 WAVE 而特別開發一套硬體設備，只要將提供 service 的那台電腦連接上 WAVE device 即可。

以下介紹幾種 WAVE device 配合 Distribution System 的情境：

- 圖 3.5 展示了標準的 service 模式，RSU 為 provider 以及 advertiser OBU 為 user。
- 圖 3.6 的 Host 為提供 service 的 provider，RSU 為 advertiser，OBU 為 user。
- 圖 3.7 的 provider，advertiser 以及 user 同圖 3.6 只是 Host 和 RSU 間多了一個 router，host 可以透過任何 router 所支援的網路來與 OBU 傳送資料。

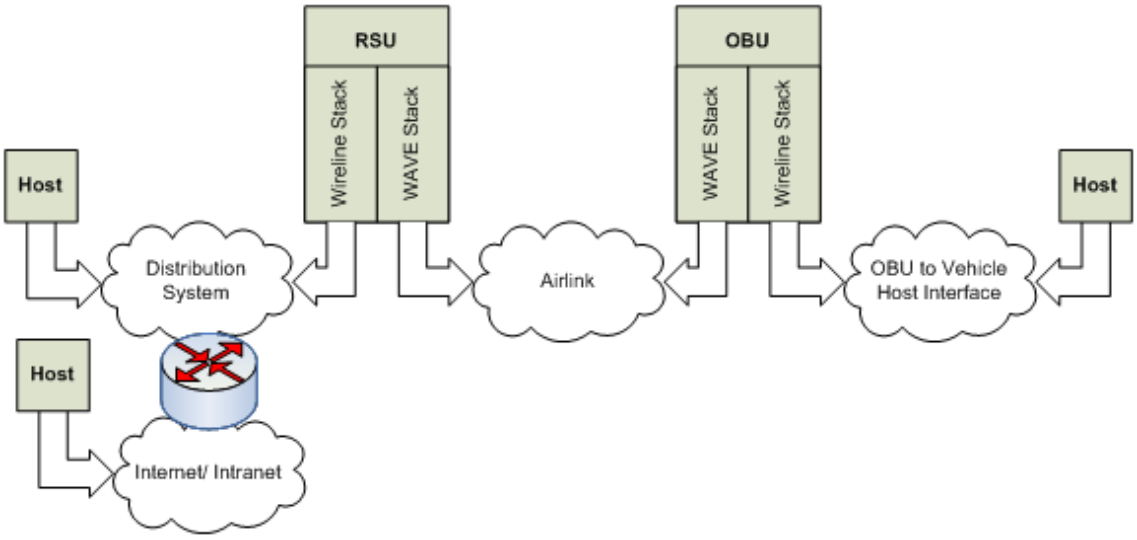


圖 3.4 An Example Scenario of the 802.11p/1609 Network

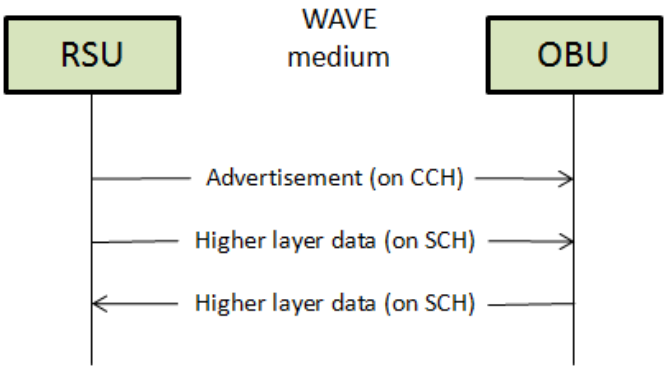


圖 3.5 The Data Exchange Process between a RSU and OBU

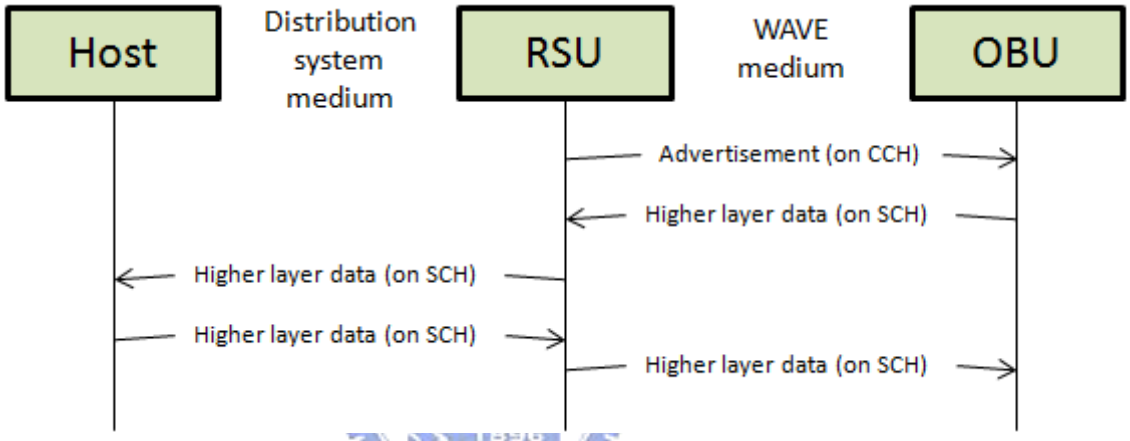


圖 3.6 The Data Exchange Process between an End Host and OBU (The RSU serves as a Relay Node)

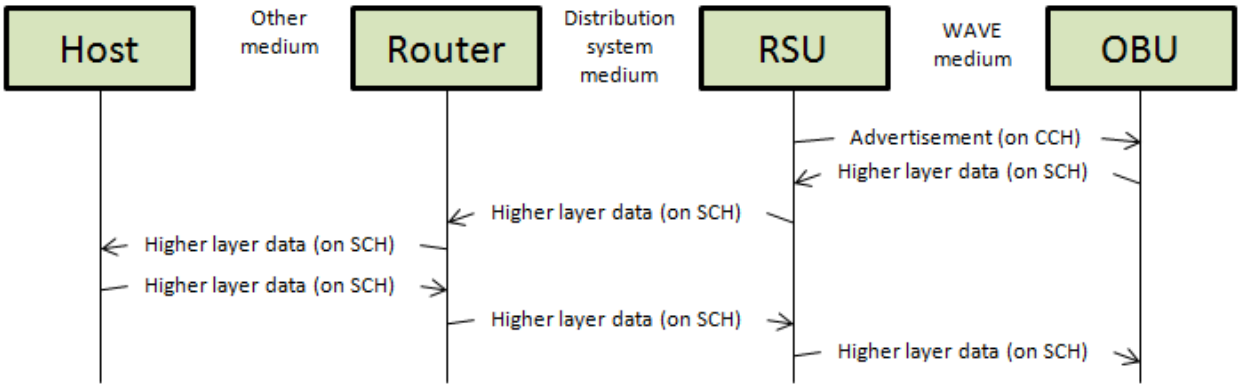


圖 3.7 The Data Exchange Process between an End Host and OBU (Across a Subnet)

3.1.2 IEEE 1609.3

IEEE 1609.3 規範了 WME 所具備的功能，WME 向上提供 application primitive，像下連結 WAVE MAC 與 WAVE PHY，可以說是管理整個 WAVE device 的核心。其負責整個 WAVE service 的運作流程，以下將個別介紹 WME 所需具備的能力：

- 服務廣播監視(Advertised service monitoring)

  WME 必須收取其他 WAVE device 所發出來的 service advertisement，提供給上層以及 WME 自己的管理函式運作。藉由 advertisement 的收集，WME 利用 management information base (MIB) 作 available service 的維護。

  WME 利用連結品質 (link quality) 來判斷 service 是否為 available service 目前 IEEE 1609.3 提供兩種方式來決定 link quality：

  - RCPI 以及平均 RCPI。WME 可以利用 RCPI 來計算兩個 WAVE device 間的無線訊號強度。WME 可以將 RCPI 轉換成 advertised WAVE device 與自己所在的 WAVE device 間的 link quality。至於 RCPI 將會在稍後的項目做詳細的介紹。
  - WSA count (這次 CCH interval 所收到的 WSA 個數)。WME 利用 WSA 所帶的 repeats (WSA 在一次 CCH interval 所發送的次數 - 1)，persistence (WSA 是否要在每次 CCH interval 都發送) 以及 WSA count 來轉換成自己所需要的 link quality。例如 repeats 為 3 WSA count 為 4 表示 provider 所發出的 WSA 都被收到了，link quality 為最佳。

- 服務的需求以及頻道的分配(Service requests and channel assignment)

  WME 必須處理 Higher layer 所要求的 service request，以及規劃 SCH 給這個 service 使用。如果 service request 的類別為 provider 那麼 WME 則要開始送出 service advertisement。

  以下將分別介紹 WAVE device 所提供的 management service，data service 以及 primitive 三大項目。

- WAVE data service 提供資料傳輸的能力，必須配合 WAVE management service 才能將資料正確的送到 destination。資料類型有 IP data 以及 WSM data 兩種，它們與 WAVE management service 配合傳送的流程請如圖 3.8 以及圖 3.9 所示。

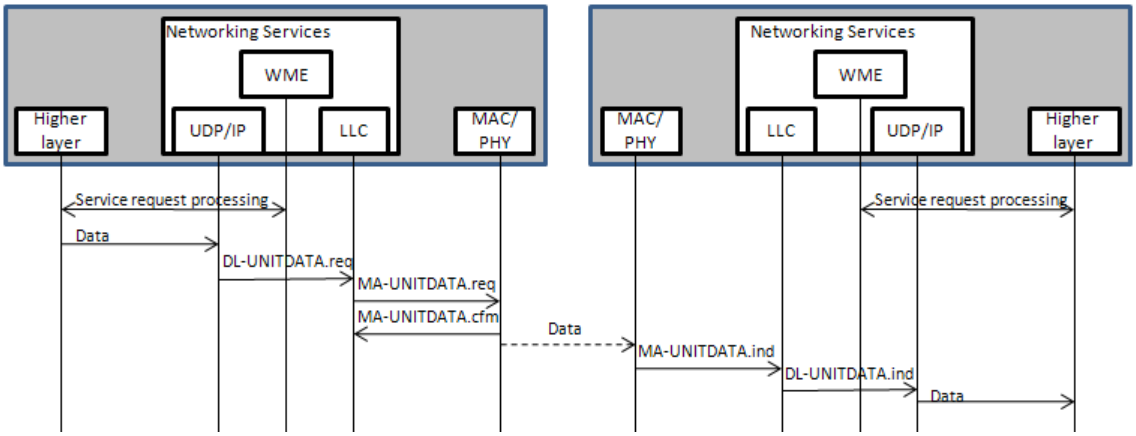


圖 3.8 The Transmission Flow of an IP Packet

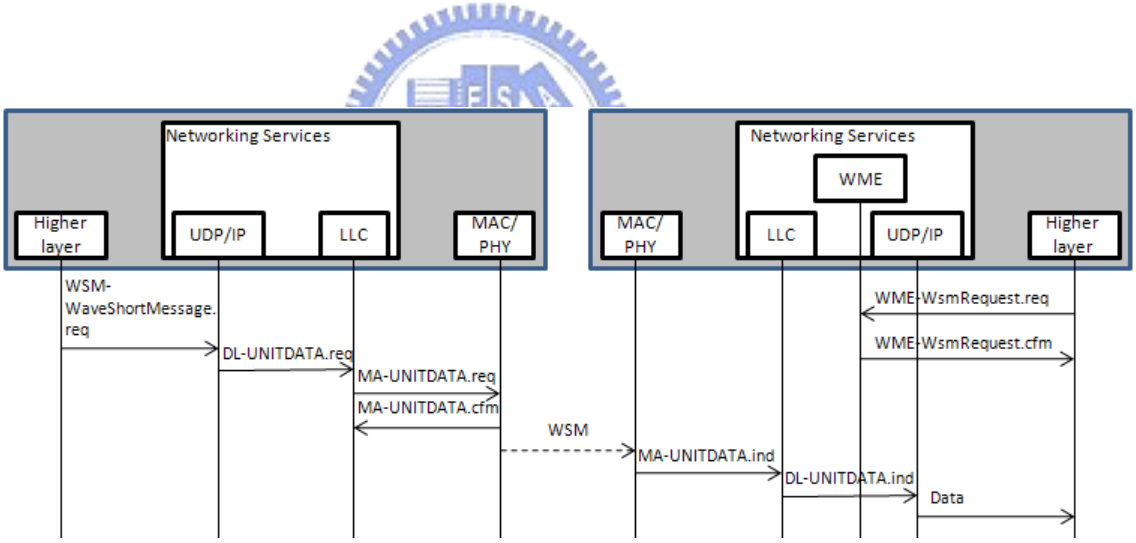


圖 3.9 The Transmission Flow of a WSM

- WAVE management service 是由 WME 來提供。其中包含 provider service，user service，WSM service 以及 CCH service 四種類型的 service request primitive。這四種 service 都包含了新增 (add) service 以及刪除 (delete) service 的選項，provider service 另外提供了修改 (change) 的功能用來做為臨時修正 provider service 的能力。Service request 的流程如圖 3.10 所

示，以下將分別介紹四種 service request：

- Provider service request 是 application 告知 WME 提供 service 的需求，WME 收到之後便將 application 所提供的資訊包成 WSA 送給下層，並且在 CCH 上傳送 service advertisement。之後到了 SCH interval，device 會留在 SCH，CCH interval 則會留在 CCH。Channel 的切換如圖 3.11 所示。
- User service request 是 higher layer 告知 WME 它對哪些 service 有興趣。如果 WME 收到關於這些 service 的 advertisement 可以告知 higher layer 或是自動加入這些 service。
- WSM service request 為 higher layer 向 WME 註冊哪些 WSM 要向上送。WSM 則以 PSID 來分類。
- CCH service 為以 WAVE device 建立/加入 service 的前提下，higher layer 要求在 CCH interval 時 WAVE device 留在 CCH 的 priority。如果 CCH priority 大於 service 的 priority，WAVE device 將會在 CCH interval 時強制切回 CCH，但是在 SCH interval WAVE device 必須留在 SCH。當 WAVE device 使用 Access Immediate 或是 Access Indefinite 這兩種方式切換 channel 時，CCH service 才會發生功用。

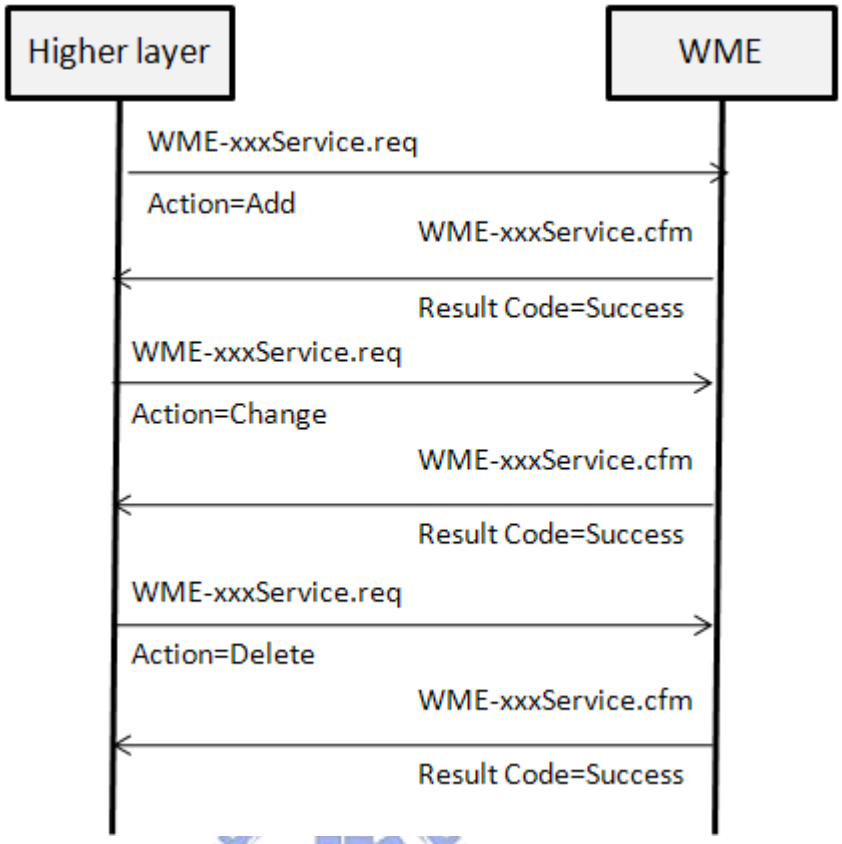


圖 3.10 Service Request Information Flow

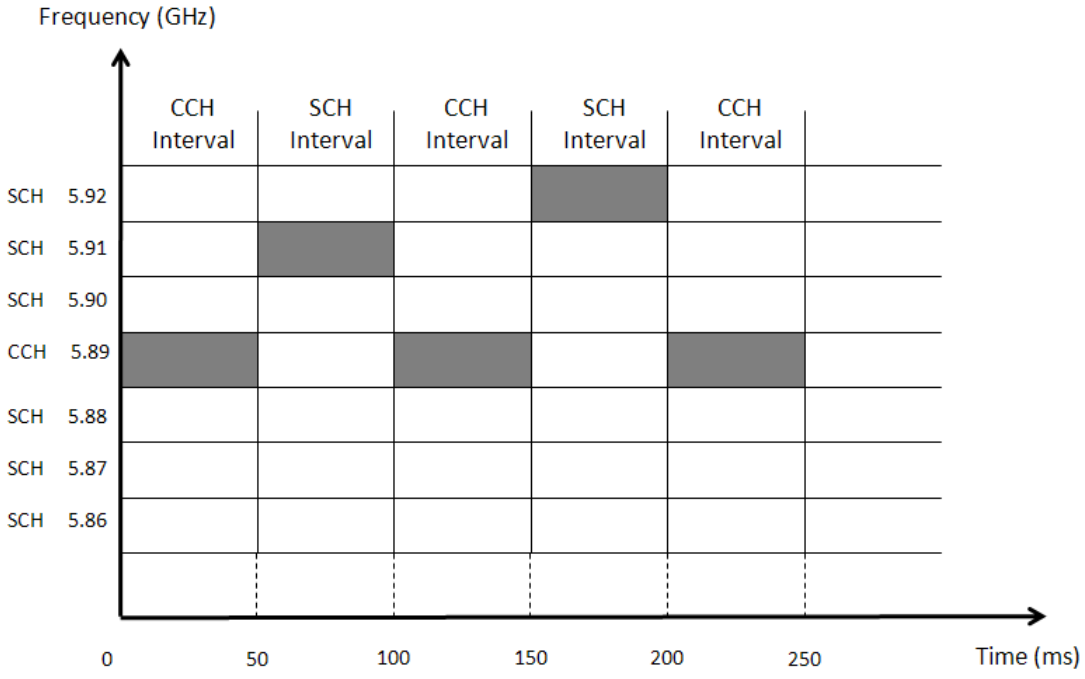


圖 3.11 An Example of Channel Access in 802.11p/1609 Network

- 在 WAVE device 中各個協定間的 控制/資料 訊息的交換是依靠 primitive。Primitive 整理於表 3.1。SAP 為 service access point 代表著協定與協定之間的介面，primitives 便是連接這些協定的基本操作。這些 primitive 大致上可以分成封包傳送，service 操作以及 MIB 的存取三類。

  WME-ProviderService.request 的操作中會導致 WSA 的產生或修改，其封包格式如圖 3.12 所示。

  WSM-WaveShortMessage.request 的操作會產生 WSM 封包，其封包格式如圖 3.13 所示。

SAP	Primitive	SAP	Primitive
WSMP	WSM-WaveShortMessage.request	LSAP	DL-UNITDATA
	WSM-WaveShortMessage.indication	MLME	MLME-DELETETXPROFILE
WME	WME-ProviderService.request		MLME-REGISTERTXPROFILE
	WME-ProviderService.confirm		MLME-WSA
	WME-UserService.request		MLME-WSADIGEST
	WME-UserService.confirm		MLME-WSAEND
	WME-WSMService.request		MLME-SCHSTART
	WME-WSMService.confirm		MLME-SCHEND
	WME-CchService.request		MLME-GET
	WME-CchService.confirm		MLME-SET
	WME-Notification.indication		MLME-MREPORT
	WME-Get.request		MLME-MREQUEST
	WME-Get.confirm	MAC	MA-UNITDATA
	WME-Set.request		

	WME-Set.confirm
	WME-RCPIREQUEST.request
	WME-RCPIREQUEST.indication

表格 3.1 Summary of WAVE Primitives

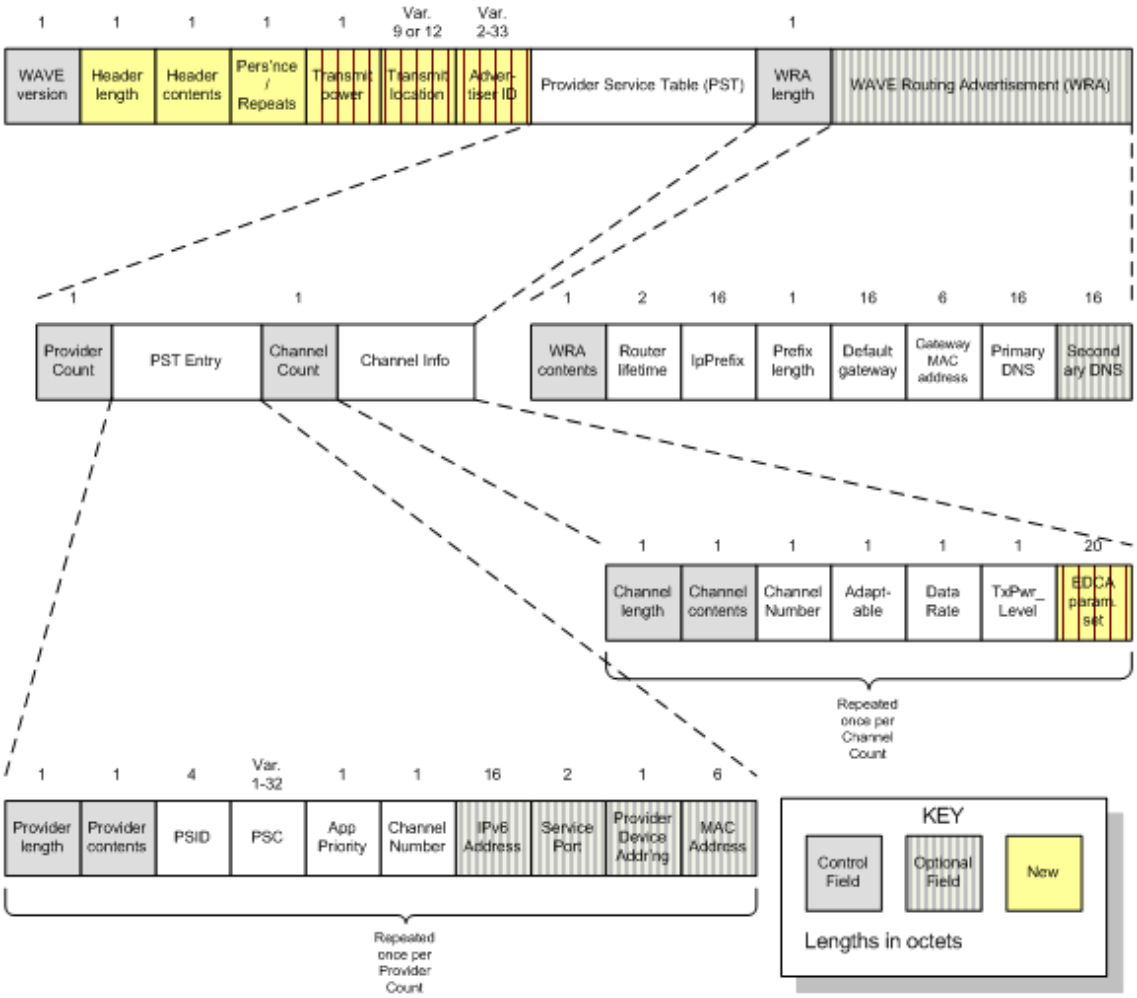


圖 3.12 The Format of a WSA

圖 3.13 The Format of the WSM Header

- 評估服務頻道的品質(Service channel quality evaluation)

  WME 提供一些機制來選擇品質較好的 SCH。

- IPv6 設定(IPv6 configuration)

  WME 設定自己的 IPv6 協定堆疊，用來收取其他 WAVE device 所發出的 IP 封包。

- 輪詢頻道功率的指示(Received Power Indicator(RCPI) polling)

  WME 可以和其他 WAVE device 上的 WME 交換特殊的封包來計算兩者間的訊號強度。

- MIB 的維護(MIB maintains)

  WME 維護自己的 MIB 其中包含 WAVE device 的設定以及狀態，WME 必須根據 primitive 的指示來 存取/修改 MIB 的內容。

3.1.3 IEEE 1609.4

1609.4 定義了 WAVE multi-channel 運作以及 MAC 對於資料的處理，以下將分別介紹 WAVE MAC 以及 multi-channel access：

- WAVE MAC

  WAVE MAC 是基於 IEEE Std 802.11 修改而來。目前 WAVE MAC 標準分兩部分，其一為 IEEE 1609.4 另一為 IEEE 802.11p，IEEE 1609.4 較偏重與 WME 間的溝通以及 WAVE system 的運行相關，IEEE 802.11p 則較偏重 MAC 與 WAVE phy 間的溝通。

  WAVE system 為了讓封包在不同的 channel (CCH/SCH) 上傳送。MAC 必須將封包 (IP/WSM) 在正確的 channel 上傳送 (channel routing)，在 CCH 傳送控制封包 (例如：WSA) 以及 WSM，在 SCH 上傳送 WSM 以及 IP 封包。這些封包可能要在不同的頻道上傳送，而且這些封包的優先權也許都不盡相同。為了解決這些特殊的要求 IEEE 1609.4 提供一個範例架構如圖 3.14 所示 (實作的時候可以自行設計)。首先以封包的 channel 來分成兩類，再根

據個別的 priority 最後進入不同的 queue，每個 queue 有不同的參數設定導致在不同 queue 內的封包有不同的傳送機率。

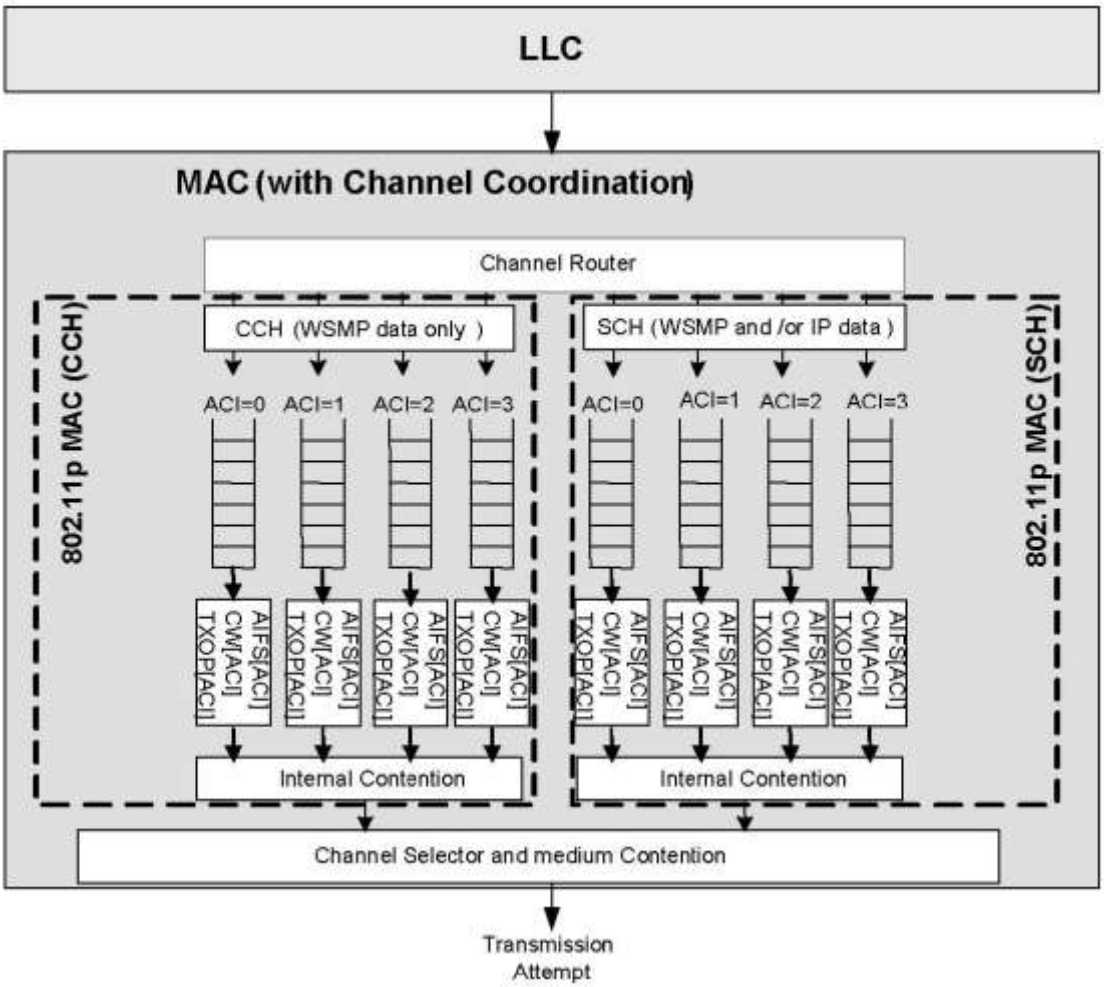


圖 3.14 The Internal Structure of an 802.11p MAC

- Multi-channel access

當 WAVE service 開始被提供或是使用時，WME 會要求 WAVE device 切換到 SCH 以收送資料。WAVE device 共提供三種 SCH 的存取方式：

- 一般模式 (Normal service channel access)

即使 WME 在 CCH interval 要求存取 SCH，WAVE device 會等到 SCH interval 才切換到 SCH，直到 CCH interval 才會切換回 CCH。CCH interval 留在 CCH，SCH interval 留在 SCH。

- 立即模式 (Immediate SCH access)

WME 一要求存取 SCH，WAVE device 馬上切換到 SCH，即使目前是 CCH interval，等到 CCH interval 時才會切換回 CCH。除了一開始之外其他都跟一般模式一樣。

- 無限模式 (Indefinite SCH access)

WME 一要求存取 SCH，WAVE device 馬上切換到 SCH，即使目前是 CCH interval。之後就一直留在 SCH。

3.2 Ad-hoc Routing Protocol

Ad-hoc 網路是一個沒有有線基礎設施 (Infrastructure) 所支持的網路，所有的節點都是由可移動的主機 (mobile node) 所構成。這種網路的特點是動態的拓樸結構，通常在需要通訊時才會開始尋找路由 (on-demand routing)。目前已經發展出許多的路由協定，以下將介紹常用的幾種 Ad-hoc 路由協定 (AODV，DSDV)。

3.2.1 DSDV (Destination Sequence Distance Vector)

DSDV 是以 Bellman-Ford 演算法所改進而來。Bellman-Ford 演算法是 Distance Vector (DV) Routing 演算法的一種，以下將個別介紹。

DV 演算法是一種反覆、非同步和分散式的演算法，每個 node 會從一個或多個相鄰的 node 取得資訊，並做處理最後將結果送回相鄰 node，直到相鄰 node 間不再交換資訊為止。其中每個 node 都維護一個表格記錄著自己到所有可到達的 node 距離，並且定期 broadcast 給相鄰的 node，node 收到相鄰 node 的 routing 資訊之後反映到自己的表格，最後根據最小路徑的原則來決定每個目的地的 next hop node。當 node 要把封包送往目的地時他只是往 next hop node 丟。DV 演算法範例如圖 3.15 所示。

Bellman-Ford 演算法的演算法如圖 3.16 所示。演算法最終的目的便是計算出發到所有可到達點的最小花費。以圖 3.15 所示的 topology 為例，出發點為 X，第一

回合 X 只知道相到鄰點的花費，X-Y 花費 2 (暫定)，X-Z 花費 9 (暫定)。第二回合將 X 花費最小可到達的點 Y 納入考量，並且不再改變 X-Y 的路徑花費，再將 Y 的相鄰點花費納入考量，Y-Z 花費 1。此時便可以得知 X-Y 再由 Y-Z 花費會比 X-Z 小因此修正 X-Z 的花費為 3。第三回合將 Z 納入考量後發現沒有剩餘的相鄰點演算結束。

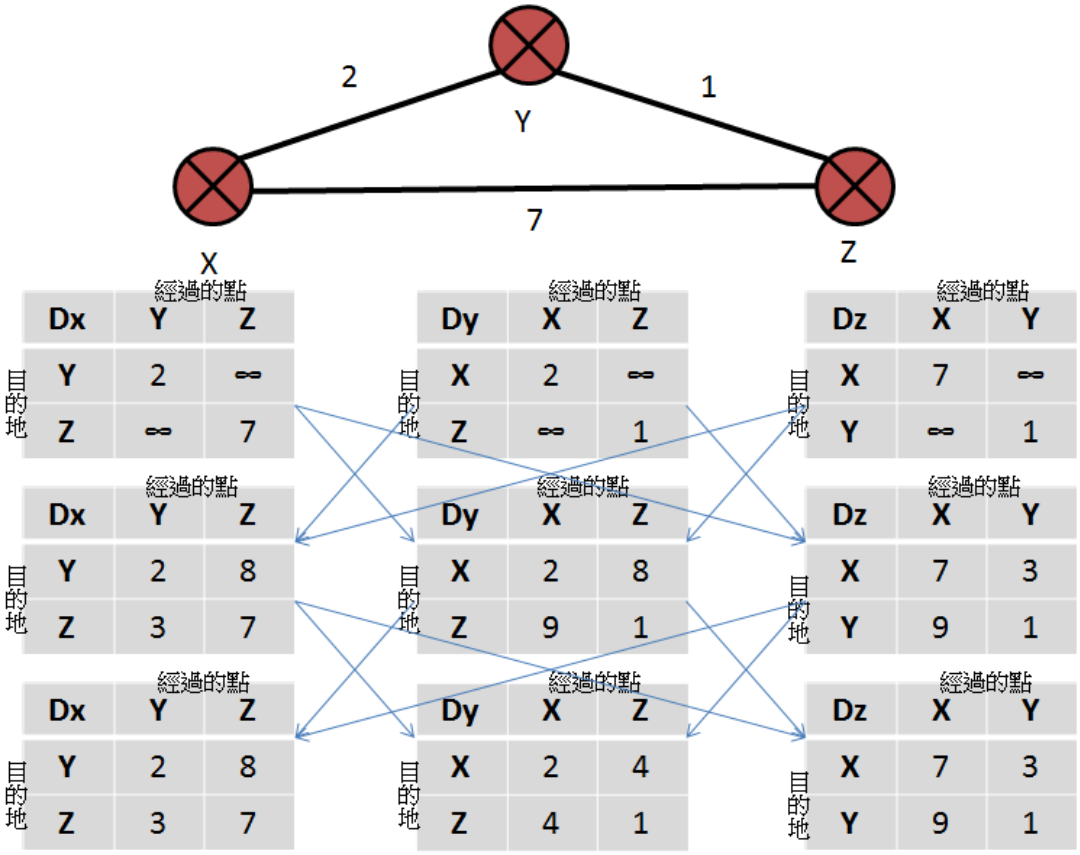


圖 3.15 An Example of the DV Algorithm

```
procedure BellmanFord(list vertices, list edges, vertex source)
  // This implementation takes in a graph, represented as lists of vertices
  // and edges, and modifies the vertices so that their distance and
  // predecessor attributes store the shortest paths.

  // Step 1: Initialize graph
  for each vertex v in vertices:
    if v is source then v.distance := 0
    else v.distance := infinity
    v.predecessor := null

  // Step 2: relax edges repeatedly
  for i from 1 to size(vertices)-1:
    for each edge uv in edges: // uv is the edge from u to v
      u := uv.source
      v := uv.destination
      if u.distance + uv.weight < v.distance:
        v.distance := u.distance + uv.weight
        v.predecessor := u

  // Step 3: check for negative-weight cycles
  for each edge uv in edges:
```

圖 3.16 Bellman-Ford algorithm-WIKI

當 Bellman-Ford 演算法穩定之後 routing table 就不會改變，只有當 topology 改變或是收到的 routing table 有所改變時才需要更新自己的 routing table。

DSDV 以 Bellman-Ford 演算法為基礎並加以改良。與一般的 DV routing 演算法相同，每個 node 都要維護自己的 routing table，試圖包含網路中所有可到 node 的路徑資訊，而且要定期 broadcast route update。與一般 DV routing 演算法不同的地方是，每一筆路徑資訊會附加一個目的 node 所指定的數字 (sequence number)，作為保證路徑資訊的時效性以及正確性。Node 收到路徑資訊後和自己得路徑資訊相比，保留 sequence number 較大者並以其作為新的路徑資訊。若是 sequence number 相同則保留最短路徑得那筆路徑資訊。此舉不但可以確定路徑資訊是最新的，也可以避免發生 route looping。Loop 發生的原因為連結成本的改變。如圖 3.17 所示。當 Y-X 花費從 4 變為 60 時，Y 由之前 Z 送來的 routing information 得知 Z-X 花費 5 因此得出 Y-Z 再由 Z-X 花費為 1+5 之後將改變

送回給 Z。Z 收到之後發現 Y-X 的花費改變了因此 Z 改變 Z-X 的花費為 Z-Y 加上 Y-X 為 7。此時可以明顯發現 loop 發生了，因為 Y-X 會經過 Z 而 Z-X 會經過 Y。DSDV 因為額外帶了其他的 routing information 因此可以避免 loop。DSDV 的 routing table 如圖 3.18 所示。

DSDV 的缺點為當 topology 非常大時，控制訊息的傳送將會佔掉很大的網路頻寬。另外每個 node 所維護的 routing table 也會非常的大，某些封包永不會經的 node 也要去維護其路徑，不僅浪費了儲存空間也浪費了 CPU 的資源。

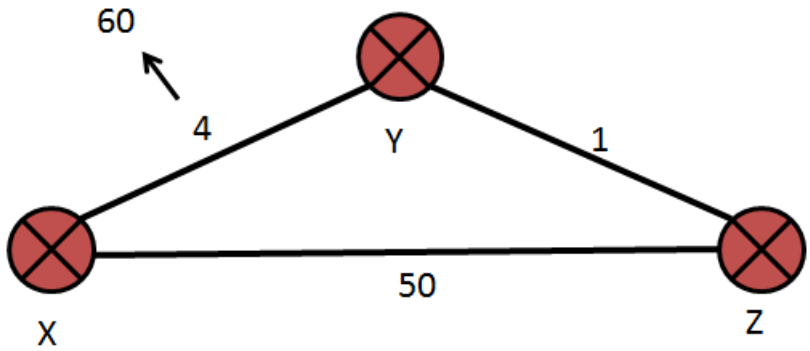
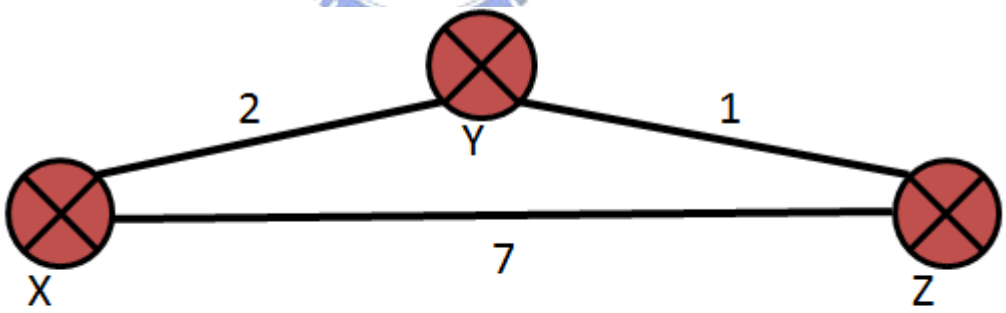


圖 3.17 An Example of Route Looping



Destination	Next Hop	Number of Hops	Sequence Number
X	X	0	X14
Y	Y	1	Y12
Z	Y	2	Y9

圖 3.18 NODE X's DSDV Routing Table

3.2.2 AODV (Ad-hoc On-Demand Distance Vector Routing)

AODV 是由 DSDV 演進而來。AODV 和 DSDV 不同之處為 AODV 只在有需求時 (on-demand)才會去建立路徑 (以 source 和 destination 來區別不同的路徑)， 而且也不必像 DSDV 一樣需要維護整個網路的 routing table。AODV 只維護選擇路徑中的 node，也不會和非選擇路徑上的 node 交換 routing table。

在開始時，整個網路都是靜止的狀態，直到有傳送封包的要求 AODV 才會開始運行。首先 source 會先在自己的 routing table 尋找是否已有到達 destination 的路徑。若無，便會開始啟動路徑尋找 (path discovery) 的程序，找到路徑之後 source 就會沿著這條路徑來送封包，為了保證封包可以在有效的路徑上傳送 AODV 會去維護這些路徑，當路徑斷掉時 route error (RERR) 會傳回給 source，如果 source 還需要此條路徑就會重新啟動 path discovery。

Path discovery 的運作如圖 3.19 所示。一開始 source node 會 broadcast route request (RREQ) 給相鄰的 node，相鄰 node 收到之後再 broadcast 出去，以此類推，直到 RREQ 到達 destination 為止，或是傳送 RREQ 的過程中某個中間的 node 已有到達 destination 的 path，並且此 path 還是有效的，此時 RREQ 的 broadcast 才會停止，之後便開始回覆 route reply (RREP) 給 source。RREP 的傳送方式為 unicast。

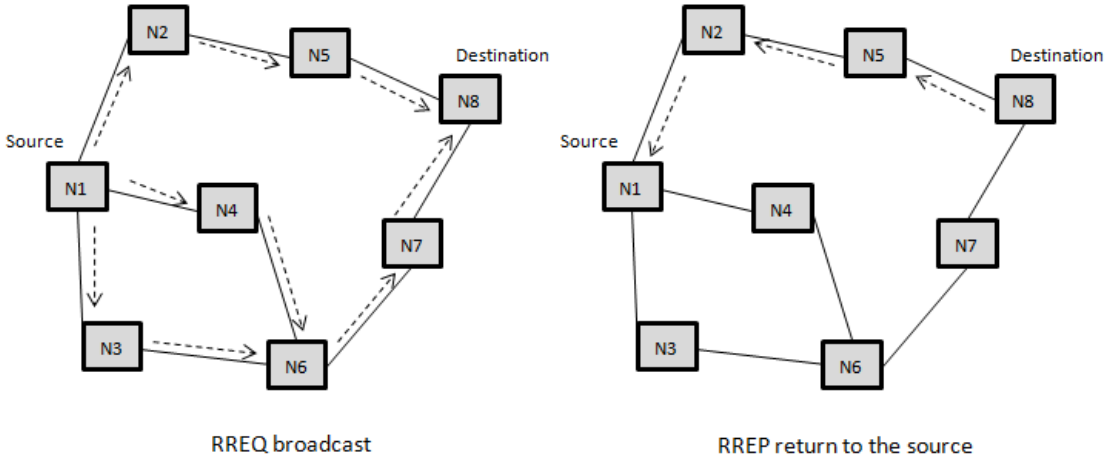


圖 3.19 AODV Path Discovery

RREQ 所帶的資訊有 source_addr，source_sequence_num，broadcast_id，dest_addr，dest_sequence_num 以及 hop_count。每個唯一的 RREQ 可以利用 source_addr 加上 broadcast_id 來辨識。每個 node 有 node sequence number 以及 broadcast id 兩個計數器 (counter)。

在轉送 RREQ 的過程中，每個 node 會自動設定反向的路徑 (例如 node B 收到 node A 的 RREQ，node B 會把 node A 的 IP 紀錄在自己的 routing table 中)，之後把 hop_count 加 1 再 broadcast 出去。如果 node 收到重複的 RREQ 則直接丟掉即可，避免發生 flooding。RREQ 中的 source_sequence_num 以及 dest_sequence_num，分別用來代表正向以及反向路徑的時效。如果收到 RREQ 的 node 發現 destination path 已經紀錄在自己的 routing table 中，先檢查 RREQ 中的 dest_sequence_num 是否比自己 routing table 中所記錄的還大，如果是則繼續 broadcast RREQ，如果不是則往反向路徑回覆 RREP。

RREP 所帶的資訊有 source_addr，dest_addr，dest_sequence_num，hop_count 以及 lifetime。RREP 沿著反向路徑回到 source 的過程中，每一個在此路徑上的 node 會設定一個轉送點 (forward point)，指向送來 RREP 的 node，並且更新 routing table 中此路徑的時效 (timeout)。

當 node 收到 RREP 其中的路徑已經被記錄在自己的 routing table 中時。如果 RREP 中的 dest_sequence_num 比自己小則直接丟棄，如果比自己的大或是 dest_sequence_num 相同但是 hop_count 比自己的小 node 才會更新 routing table 並且把 RREP 傳送出去。另外每一條路徑都有一個 timer 如果在 lifetime 內都沒有用到這條路徑，才會刪除之。

當 topology 改變了，如果移動的是路徑中的 source node 則 source 會重新做 path discovery，如果移動的是路徑中的 node 則所有 forwarding point 指向它的相鄰 node 會發現，然後依照反向路徑把 RERR 傳送給前一個 node 直到 source 為止，告知所有在此路徑上的 node 此條路徑已經失效請從 routing table 中刪除。當 source 收到 RERR 時，如果 source 還需要此一路徑，source 才會

重新啟動 path discovery。

3.3 Multi-hop Forwarding over WAVE Networks

在 WAVE system 上兩個 node 可以透過 WBSS 的建立來傳輸一般的 IP 封包，但是在兩個 node 距離 1 個 hop 以上的情形下，而且 node 之間的 node 都沒有提供 service 時該怎麼辦呢？

Multi-hop forwarding 正是為了解決此問題而設計出來的。Multi-hop forwarding 有兩種形式。其一為以送端為中心的形式 (sender-centric multi-hop forwarding scheme：SMFS)，另一種為以收端為中心的形式 (receiver-centric multi-hop forwarding scheme：RMFS)。SMFS 的意義是，由 sender 來建立 WBSS 讓 receiver 加入以提供封包傳輸的 service，RMFS 則相反，是由 recevier 來建立 WBSS 讓 sender 加入以提供封包傳輸的 service，SMFS 由 sender 自主建立 WBSS 是很自然的一件事，但是 RMFS 由 receiver 來建立 WBSS 卻要通過其他的幫助，因為在 service 還未建立前不能傳送 IP 封包，receiver 也不會知道自己是 receiver，因此必須要由 sender 在 CCH 先發送一個 WSM 給 receiver，請 receiver 建立 WBSS。

RMFS 以及 SMFS 都有提供 service channel 的選擇方式。SMFS 必須由 sender (provider) 來決定。RMFS 則可以由 sender (receiver：channel 選擇資訊帶在 WSM) 或是 receiver (provider) 來決定，如果 channel 的選擇是由 provider 來決定，則 provider 必須收集其他 advertisement 所帶的 channel number 來記錄 channel 的使用情況，如果 channel 的選擇是由 user 來決定，則 user 必須收集其他 WSM 內所帶的 channel number。

3.3.1 SMFS

SMFS 是較為直接的 multi-hop forwarding 方式。當某個 OBU 想要透過

WAVE network 傳送 IP 封包給另一個 OBU 時，首先 OBU (original sender) 會檢查另一個 OBU (target receiver)，是否在自己的傳輸範圍內 (one hop)。如果是的話那就直接由 original sender 建立一個 WBSS，並且等待 target receiver 加入即可。如果 target receiver 距離 original sender two hop 以上那麼 original sender 會試圖把封包先傳送到 original sender 附近的 RSU 再由 RSU 透過固網將封包送至 target receiver 附近的 RSU，最後再由 RSU 送給 target receiver。

RSU 通常都會有固定的 WAVE service 運作。因此，OBU 只要直接加入便可以透過 RSU 來傳送封包。RSU 在背後使用固網來傳送資料，網路的品質較為可靠，速度也較快。重要的是如果 original sender 和 target receiver 的 forwarding point 都是 OBU 的話，每經過一個 forwarding OBU 都要經歷一次 SMFS 的運作，會浪費掉相當多的時間。但是若 original sender 和 RSU 的間隔距離過遠 (大於 1 hop)，那麼 original sender 還是要透過 RMFS 來將封包送至距離 RSU 更近的 forwarding OBU，再由此 forwarding OBU 將封包送至 RSU。

SMFS 的運作情形如圖 3.20 以及圖 3.21 所示。在圖 3.20 的情境 OBU 並不需要 SMFS 的機制，由 original sender 直接將封包送至 RSU，再由附近的 RSU (轉)送至 target receiver 即可。圖 3.21 中，original sender 沒有辦法直接使用 RSU 所提供的 service，而只能利用 SMFS 的機制將封包傳到附近的 OBU (forwarder)，再由 forwarder 送至 RSU，最後由 target receiver 附近的 RSU 轉送至 target receiver。

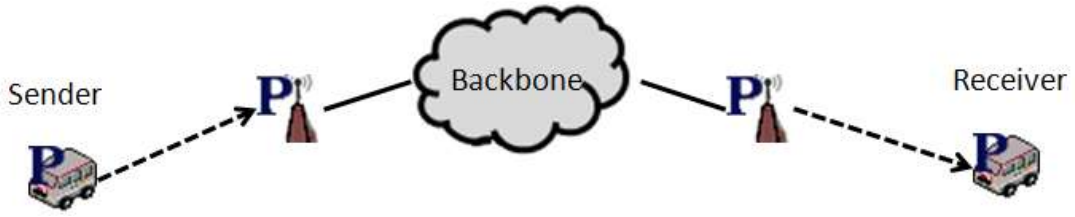


圖 3.20 An SMFS scenario where the Sender node can Directly Communicate with RSU node

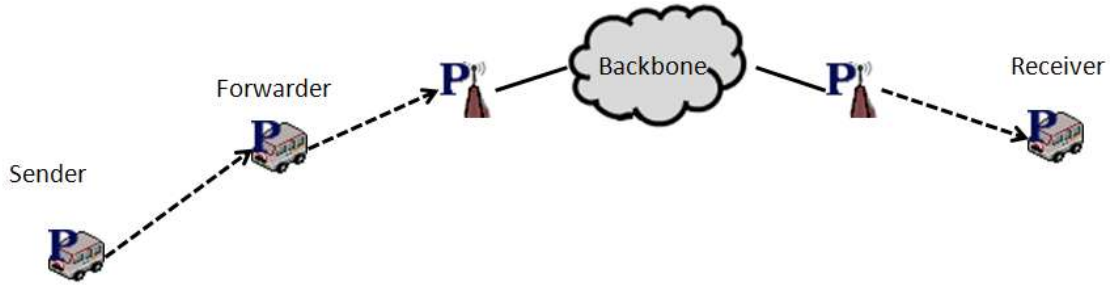


圖 3.21 An SMFS scenario where the Sender node cannot Directly Communicate with RSU node

SMFS 的內部運作如圖 3.22 所示。首先由 sender (original sender or forwarder) 在 CCH interval 發送 advertisement，通知 receiver(forwarder or target receiver)，加入 WBSS 以收取封包。等到 SCH interval 時再由 sender 將要送出的 IP data broadcast 出去。在此必須要使用 broadcast，因為 sender 不知道哪一個 receivers 會加入 service，receivers 也不會知道自己是否是目前加入此 WBSS 中離 RSU 最近的。Receivers 只知道自己是不是比 sender 更接近 RSU (使用 position-based routing)。因此 SMFS 只能將封包利用 MAC-layer 的 broadcast 送出去，讓所有加入此 WBSS 的 user 收到。如此才能讓封包逐漸的接近 RSU。

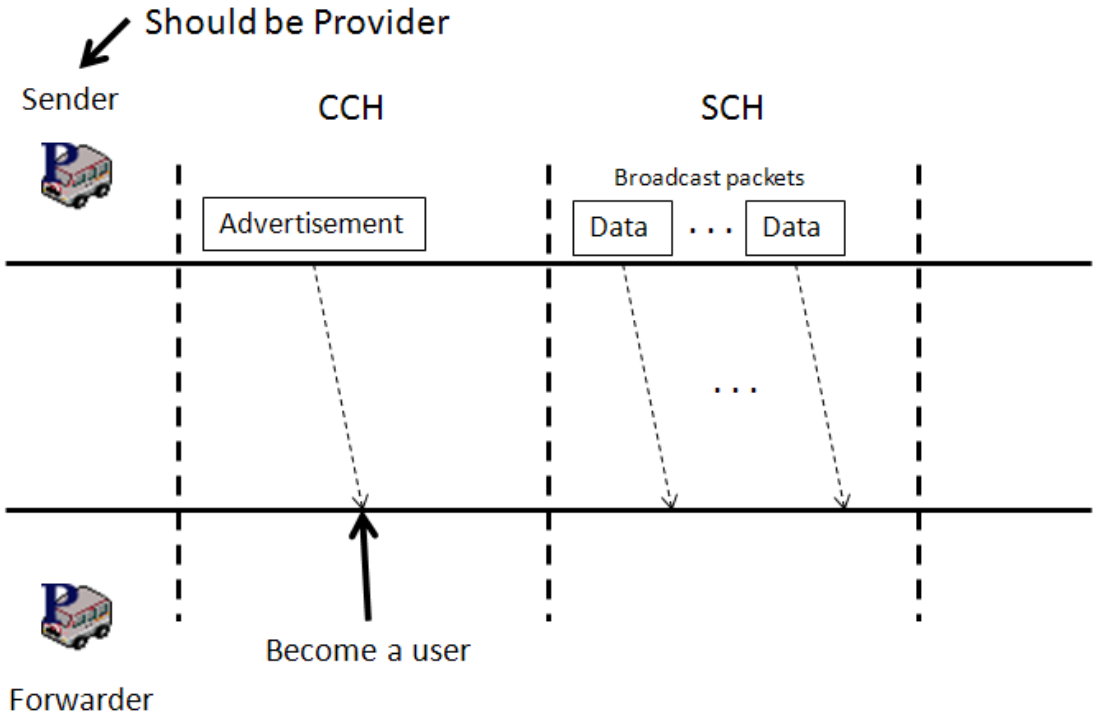


圖 3.22 The Procedure of SMFS

3.3.2 RMFS

RMFS 相對於 SMFS 較為間接，但是它卻沒有 SMFS 所存在的問題。假設現在有三個 node，A，B 以及 C。在 A 和 B 同時想透過 forwarder C 送出封包的形形下，依照 SMFS 的方法 A 和 B 都會發 WSA 給 C。在 A 和 B 都用不同 SCH 來傳輸封包的情況下，C 只能替其中一個 node forward 封包。而且，由於 SMFS 中 sender 使用 MAC-layer 的 broadcast 封包，所以即使 A 或 B 的封包掉了，也不會被發現。

RMFS 運作的時機與 SMFS 相同，但是 RMFS 的 WBSS 建立的過程與 SMFS 正好相反。如圖 3.23 所示，首先由 sender (original sender or forwarder) 在 CCH interval 發送 WSM (broadcast) 告知 receiver (forwarder or target receiver)，Receiver 收到此 WSM 之後便開始建立 WBSS。如此 sender 便可利用此 service 在 SCH interval 傳送 IP data。使用 broadcast WSM 的理由是最接近 RSU 的 OBU 不一定有機會成為 forwarder，因此凡是在附近收到此 WSM 的 OBU 都可

以提供 service 讓 sender 選一個最接近 RSU 的 WBSS 加入。另外， forwarder 也可以藉由收到其他 forwarder 所發出的 advertisement 來判斷自己是否比別的 forwarder 更接近 RSU，如此 forwarders 也可以自行判斷是否有必要繼續提供 service 讓 sender 傳送封包。

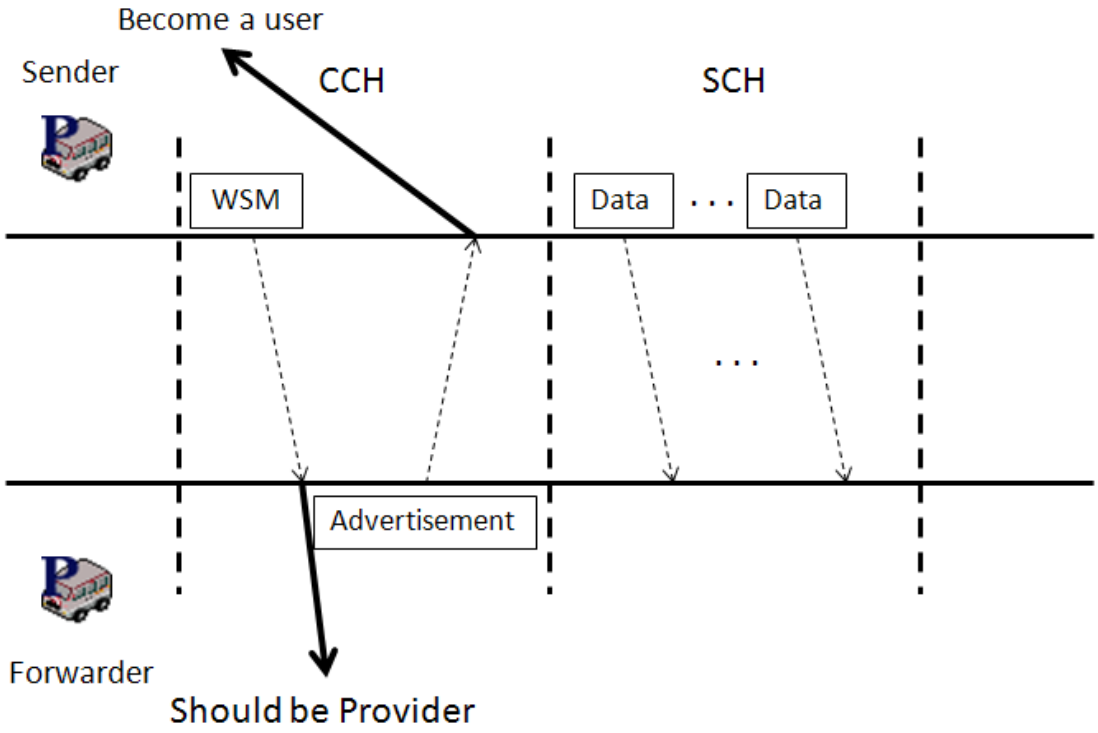


圖 3.23 The Procedure of RMFS

Chapter 4 Design and Implementation

如同第一章的介紹，本篇論文的目的便是，在 WBSS-based V2V networks 上交換訊息，其中包含了利用 routing protocol 來尋找路徑以及使用 multi-hop forwarding 機制來傳輸資料。本章節會介紹我們所做的改變，設計，想法以及實作。本篇論文使用了 NCTUns [19] 平台來實做這些功能。

4.1 IEEE 1609.3 Implementation

章節 3.1.2 所介紹的 WME 以及 WSMP 架構以及運作流程都是由 IEEE 1609.3 所定義，以下將使用兩個章節分別介紹我們在 NCTUns 上的實作。

4.1.1 WME Implementation

在 NCTUns 中，大部分的硬體設施都是實作成 module 的形式來模擬。WME 是 WAVE device 的一部分，因此我們將它實做成 NCTUns module。由於 WME 向下負責控制 WAVE MAC 向上負責與 application 溝通，因此我們將 WME module 的位置放在 WAVE MAC module 之上 ARP module 之下。如圖 4.1 所示，4.1(a) 是一般的 OBU 它只有 WAVE device 這個介面，4.1(b) 則是 RSU，其除了可以透過 WAVE network 傳送封包，也可以透過有線網路來傳送封包，因此包含了兩種介面。在章節 3.1.1 曾經介紹過此種情並且稱之為 Distribution System。

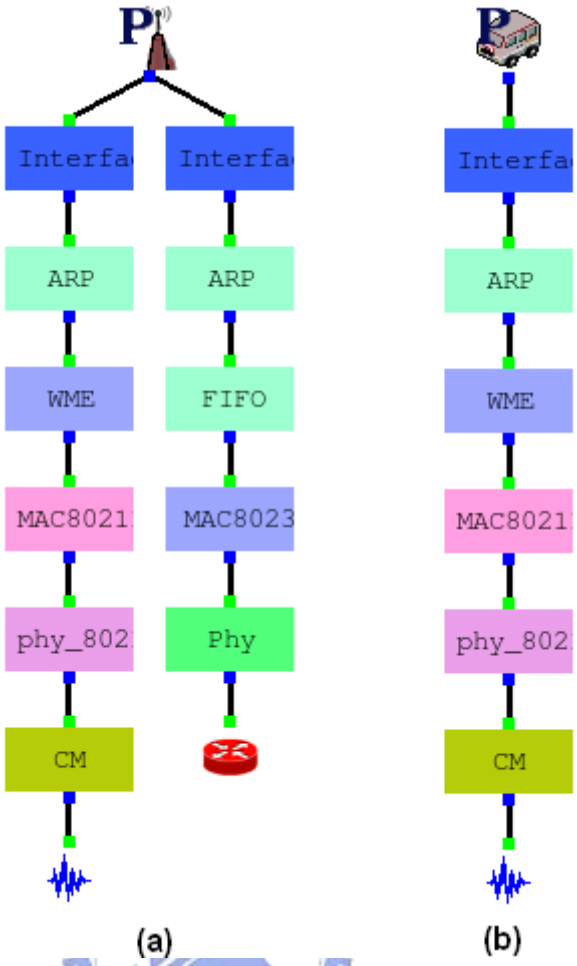


圖 4.1 The Protocol Stack of a WAVE Device in NCTUns

WME 透過 primitive 與 higher layer 以及 MAC layer 溝通，但是我們實做的 WME 並沒有完整的支援 WME 與 higher layer 間的溝通。目前只能支援 application 單向的透過 primitive 來告知 WME WAVE service 的訊息 (例如 provider service 的新增)，其原因要從 WAVE standard 的發展歷史來說明。一開始 IEEE 1609.1 定義了 application 以及 WME 的中介層，稱為 resource manager，application 必須透過 resource manager 來存取 WAVE device 所提供的資源，而在 IEEE 1609.3 所說的 higher layer 正是 resource management。但是，IEEE 1609.0 在之後卻被部分的人認為制定失敗。雖然後來又制定了 IEEE 1609.5 的 communications manager 來取代 IEEE 1609.0 中 resource management，但是在實做 WAVE system 時 IEEE 1609.5 還未發表，因此我們捨棄了在 WAVE device 上

application 透過 higher layer 與 WME 間動態使用 primitive 溝通的功能。最後，我們選擇了折衷的辦法，那就是，一開始就必須設定好 higher layer 何時去使用 WME 提供的 primitives。實際的情形為使用者必須一開始就在指定的檔案 (primitive file) 中說明 WME 要在何時執行何種 primitives。在我們的實作中只提供 WME-ProviderService.request ， WME-UserService.request ， WME-CchService.request 以及 WME-WSMService.request 四種較重要的 service request primitive。檔案的格式如圖 4.2 所示，SIB_Begin 以下 SIB_End 以上為設定檔的內容，所有 node 的 primitives 都設定在此檔案內。NID X 以下 NID Y 以上為 NID X 的 primitives 設定。CDB 與 CDE 中間所夾的內容就是一筆完整的 primitive 設定，其中時間 (Time) 的單位為 $10^{(-7)}$ second，表示 primitive 啟動的時間。

```
SIB_Begin
        NID 1
                CDB
                        Time 20000000
                        Primitive provider_service_req
                        Action add
                        PSID 1
                        PSC ""
                        AppPriority 1
                        Channel 174
                        Presistence 1
                        Repeats 0
                        IPService 1
                        IPAddr 1.0.1.1
                        ServicePort 1
                        MacAddr 0:1:0:0:0:2
                        RecipientMacAddr 0:1:0:0:0:3
                CDE
                CDB
                        Time 10000000
                        Primitive user_service_req
                        Action add
                        UserReqType auto_access_on_service_match
                        PSID 1
                        PSC "test"
                        Channel 174
                        SourceMac 0:1:0:0:0:2
                        AID ""
                        ImmediateAccess 1
                        IndefiniteAccess 1
                        Notify 1
                CDE
        NID 2
                CDB
                        Time 10000000
                        Primitive wsm_service_req
                        Action add
                        PSID 1
                CDE
                CDB
                        Time 10000000
                        Primitive cch_service_req
                        Action add
                        AppPriority 1
                CDE
SIB_End
```

圖 4.2 WME Primitive Setting File

一開始每個 node 的 WME module 會去讀取 primitive file，並且把屬於自己的 primitives 設定按照時間順序存在 queue 中，直到 primitive 的啟動時間到了，WME 就依照 primitive 的 type 去執行不同的 function。

所有 service 的新增，移除或是修改都是藉由 primitives file 所指定的 primitives 來運作。以下將詳細介紹這些 service request 的運作流程：

- Provider Service：

  若 provider service request 的 Action 為 add，其代表的意思為 application 想提供 service，請 WME 幫助它完成。收到此 primitive 之後，WME 首先會將 WME-ProviderService.request 的內容記錄在 MIB 中並且開

始準備發送 advertisement。如果 WAVE device 之前就是 provider 那麼 WME 就要修改之前的 WSA，並且在原本的 WSA 之後附加此 provider 的資訊，否則就要自己建立新的 WSA。WSA 建立完成之後就要根據 primitive 中所帶的 persistence 以及 repeats 來發送 WSA。最後便是告知 MAC 到達 SCH interval 時要跳到哪個 SCH。

若 provider service request 的 Action 為 change，其代表的意思為 application 想改變 service 的參數，請 WME 幫助它完成。此時 WME 會根據 primitive 的內容來修改 MIB 中相對應的欄位，並且修改已存在的 WSA 成為 primitive 所要的樣子。如果 SCH 有改變的話則要重新通知 MAC，另外 WSA 的發送頻率有改變的話也要做處理。

若 provider service request 的 Action 為 del，其代表的意思為 application 想停止 service 的提供，請 WME 幫助它完成。此時 WME 會根據 primitive 的內容來刪除 MIB 中相對應的欄位，並且將 WSA 中相應的 provider 訊息刪除。如果 WSA 內沒有其他的 provider 的訊息則停止發送 WSA，並且告知 MAC SCH interval 請留在 CCH。

- User Service：

  WME-UserService.request 有新增與刪除兩種 action。新增表示 application 對於 primitive 所指定的 service (以 PSID 來區分) 有興趣，刪除則相反。收到 user service request 時 WME 一開始只會去修改 MIB 的欄位，之後如果 WME 收到 WSA 中包含 application 所感興趣的 service， WME 會自動加入此 service 直到此 service 不再提供，或是 application 對這類型的 service 不再感興趣為止。如果 application 對很多 service 都感到興趣的話，WME 會選擇最高 priority 的 service 加入。

- Cch Service：

  WME-CchService.request 有新增與刪除兩種 action，用來新增或是刪除 CCH 的 priority。由於 CCH 只有一個，因此 WME 只會使用到最大的

priority。WME 收到此 primitive 時只會去修改 MIB 的欄位，直到 SCH interval 轉換到 CCH interval 那一瞬間，如果 WAVE device 想要繼續留在 SCH 它必須去比較目前所使用的 provider service 的 priority 是否比 CCH priority 大，如果是則會在繼續留在 SCH 否則就會切換回 CCH。

- WSM service：

  WME-WSMService.request 有新增與刪除兩種 action，新增或是刪除 PSID。WME 可以用 PSID 來判別 WSM 由哪一種 application 送出，application 可以藉此決定只收哪種 PSID 的 WSM。WME 收到此 primitive 時只會去修改 MIB 的欄位，直到 WAVE device 收到 WSM 時，WME 會根據 application 之前所註冊的 WSM 的 PSID，來決定要不要收此 WSM。

  目前 WME 只能服務一個有能力收發 WSM 的 application，因此只要是 WSM service 有註冊過的 PSID 都會送到同一個 application，WME 並沒有 route WSM 到正確 application 的能力。使用這種作法的原因是因為在實做 WME 當時考慮到 WAVE 標準對於 application 的定義並不明確。

WME 必須要維護 available service。在此使用的方法為收集其他 WAVE device 所發出的 advertisement，如果在此次 CCH interval 有收到關於某 service 的 advertisement 表示此 service 可以使用，WME 會在 CCH interval 結束時搜尋 MIB 中記錄的 available service，如果 available service 中有對應到 application 使用 user service request primitive 註冊的 service 則 WME 會自動加入此 service，詳細的運作情況如圖 4.3 所示。

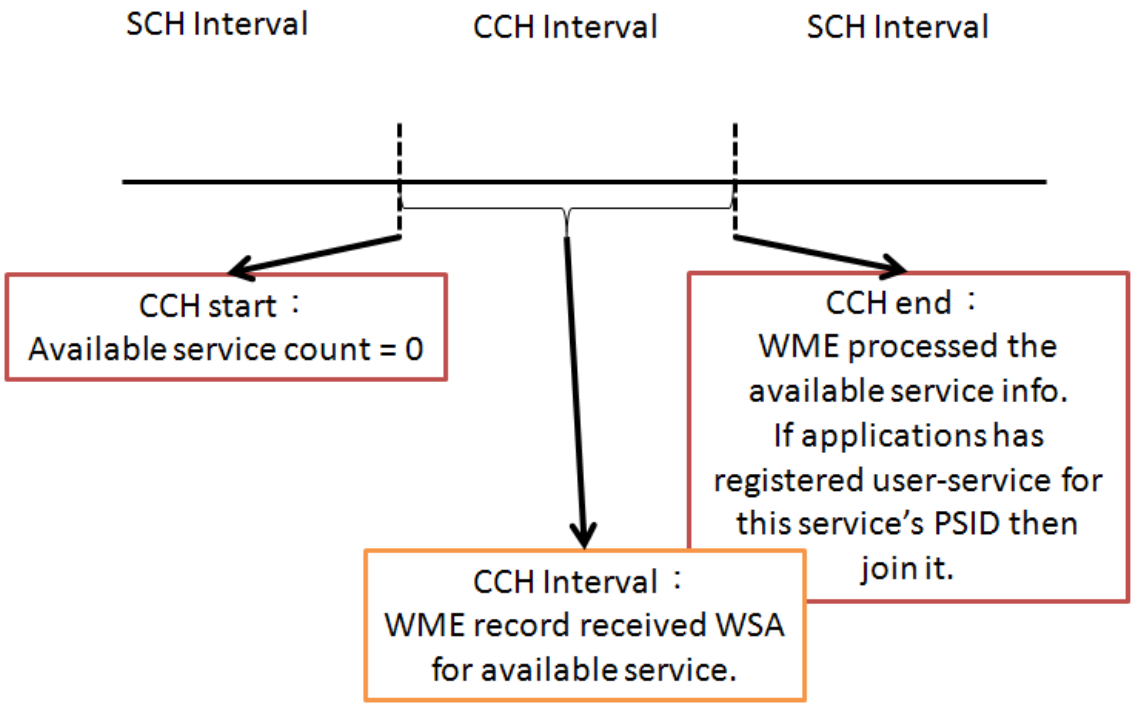


圖 4.3 Recording Available Services in CCH intervals

IEEE 1609.3 定義了 WAVE system 發送 IP 封包的條件限制，這個機制目前實作在 WME 中。其實際運作情況如圖 4.4 所示，如果 IP 封包是由 higher layer 送往 WME，則 WME 會檢查目前 WAVE device 是否有加入或是提供 service (user or provider)。若 WAVE device 目前為 user 則只能將封包送往提供自己 service 的 provider，若 WAVE device 為 provider 則直接將封包送出去，發生其他的情況則直接將封包丟棄。如果 IP 封包是由 lower layer 送往 WME，WME 同樣要檢查目前 WAVE device 是否有加入或是提供 service (user or provider)。若 WAVE device 目前為 user 則只能收取自己 provider 所發出的封包，若 WAVE device 為 provider 則直接將封包向上送，其他情況則直接將封包丟棄。

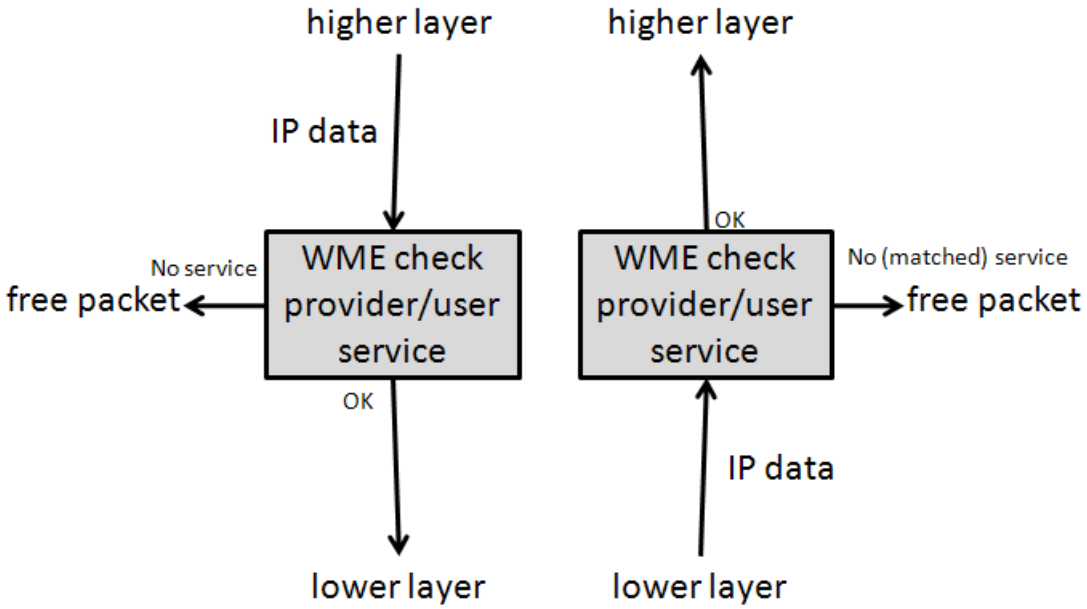


圖 4.4 The Processing of IP Packets in WME

4.1.2 WSMP Implementation

WSMP 是專門為了 WAVE system 所設計的 protocol，它與 IP protocol 的地位相同。在 NCTUns 所使用的 IP protocol 是修改 Linux kernel 內建的 IP protocol 而來，但是 Linux kernel 並沒有內建 WSMP，因此必須自行實作。為了讓 WAVE applications 可以透過 function call 的方式來發送 WSM，以下有兩種方法來實作 WSMP。第一種方法比較直覺，將 WSMP 實作在 Linux kernel 中，但是缺點是實作的難度較高以及每次 Linux 改版都要重新維護。第二種方法是將 WSMP 實作在 NCTUns 中，application 要發送 WSM 時利用 IPC 將 WSM 送至 NCTUns 內。

在此使用上述的第二種實作方式。WSM 收送的情形如圖 4.5 所示，當 application 要發送 WSM 時，其會透過 IPC 將 WSM 送至相對應的 WME，之後再由 WME 將 WSM 送出。當另一方的 WME 收到 WSM 時，它會透過 signal 來通知 application，最後 application 透過 IPC 讀取 NCTUns 內的 WSM。

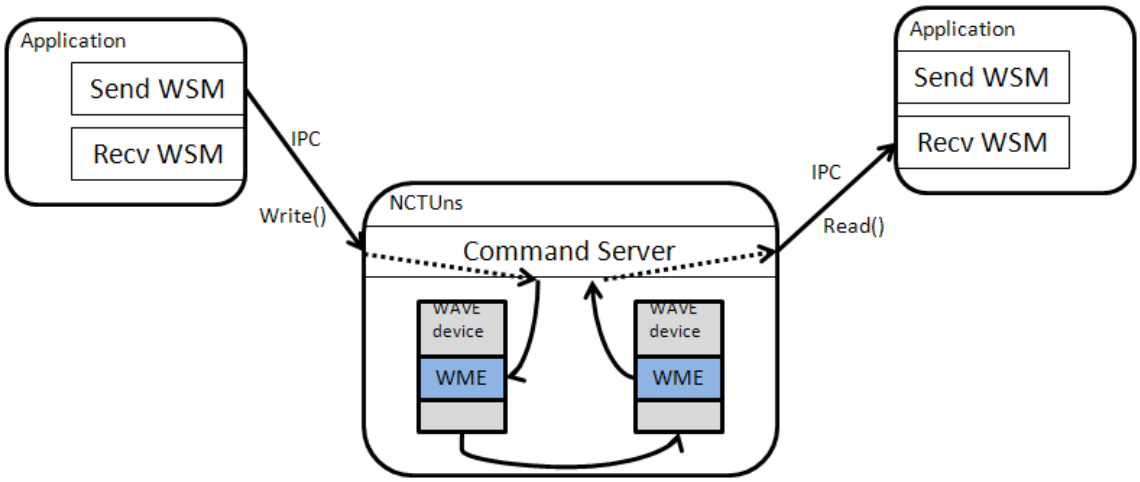


圖 4.5 Send/Receive a WSM

NCTUns 透過其內的 command server 來收送 WSM，之後再由 command server 來將 WSM 送至相對應的 WME module 或是 application，也就是由 command server 來負責做 IPC。

NCTUns 在開始模擬前會先將其所有將要運行的 module 初始化，接下來才會開始 fork 出即將要在 NCTUns 上模擬執行的 process。為了要讓 application 將 WSM 送到正確的 WME，以及 WME 將 WSM 送到正確的 application，因此我們利用 NCTUns 會先初始化完所有的 module 後才會 fork process 的特性。其細節如圖 4.6 所示，在 WME 作初始化時先在 command server 內建立一個資料結構 WME_REG_INFO，記錄 WME 所在的 node id 以及 WME module 所在的位置。當 application 被 fork 出來時 application 會透過 IPC 向 command server 告知此 application 是由哪個 node 所執行，application 和 command 間溝通所利用的 file descriptor 以及 application 的 process id (PID)。Command server 會根據 application 所給的訊息 (node id) 去找相對應的 WME_REG_INFO，找到之後便將 application 的 PID 紀錄在內。如此，WME_REG_INFO 內便有 WME 以及 application 的共同資訊，以及個別資訊 (共同資訊為 node id，個別資訊為 WME module 位置以及 application 之 PID)。

Application 想發送 WSM 時先透過 IPC 傳送給 command server，由

commander 去尋找相對應的 WME_REG_INFO，利用其內的資訊將 WSM 送至正確的 WME module。

WME module 想把 WSM 傳給 application 時，WME 直接將 WSM 交給 command server 處理，讓 command server 去尋找相對應的 WME_REG_INFO，利用其內的資訊將 WSM 透過 IPC 送至正確的 application process，並且發送一個 signal (kill(PID, SIGANL_TYPE)) 通知此 application 來收取 WSM。

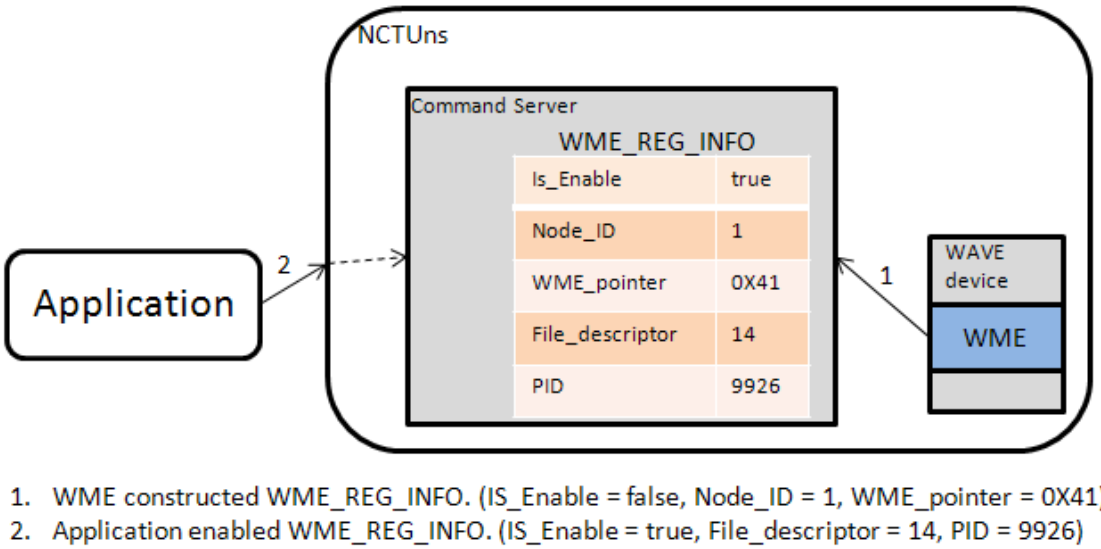


圖 4.6 Construction of the WME_REG_INFO Structure

為了配合 WME 的設計因此所有想發送 WSM 的 application 都必須依照圖 4.7 的格式撰寫。首先 application 要先利用 IPC 開通和 NCTUns 間的通道，接下來要傳送註冊訊息給 NCTUns，並且要對 NCTUns 所發出的 signal 做處理 (signal handler)。Application 收到 NCTUns 所發出的 signal 表示 NCTUns 已經將 WSM 送到之前所建立的 IPC 通道，此時 application 要中斷目前程式的執行，並起去執行 signal handler 以收取 WSM。另外 application 只要把想送的 WSM 包裝成正確的格式，再透過 IPC 送給 NCTUns 便可完成 WSM 的發送。

```
void sig_usr1(int signo)        //signal handler function
{
        sigprocmask(SIG_BLOCK, &usr1_mask, NULL);
        //防止收取封包時遭到 signal 的中斷

        Recv_WSM();

        sigprocmask(SIG_UNBLOCK, &usr1_mask, NULL);
}

int Recv_WSM()
{
        read(file_descriptor, buffer, buffer_size);
        //透過 IPC 收取 WSM
}

int Send_WSM(data, ...)
{
        sigprocmask(SIG_BLOCK, &usr1_mask, NULL);
        //防止發送封包時遭到 signal 的中斷

        WSM_data = packing_WSM(data, ....);
        //將發送的 data 包裝成  WSM 格式

        write(file_descriptor, WSM_data, size);
        //透過 IPC 發送 WSM 給  NCTUns

        sigprocmask(SIG_UNBLOCK, &usr1_mask, NULL);
}

int main()
{
        createTCPSocketForCommunicationWithSimulationEngine(...)
        //開通和 NCTUns 間的連線

        Enable_WME_REG_INFO(...);
        //向 NCTUns 註冊 process 的資訊

        signal(SIGUSR1, sig_usr1);
        //註冊 signal 的 handler function

        ...
        ...
        Send_WSM(data, ...);
        ...
        ...
}
```

圖 4.7　WAVE Application for Sending or Receiving WSMs

4.2　Support WBSS-based V2V Networks

在 WBSS-based V2V networks 交換資料有兩個流程，首先是資料傳送路徑的尋找，找到路徑之後才能依據此路徑作資料的傳送。章節 3.3 所介紹的 multi-hop

forwarding 是建立在 sender OBU 可以特過 RSU 所提供的 WAVE service 來傳送封包給 receiver OBU，但是本篇論文所討論的東西有點不太一樣。首先所有在 topology 上的 node 都是 OBU，而且必須要透過一般的 ad-hoc routing protocol 來尋找路徑，因此我們必須要對原本 multi-hop forwarding 的設計做修正。

我們即將要介紹的內容有兩部分，其中一項為我們是如何讓 ad-hoc routing protocol 在 WAVE networks 上運作，另一項為，在已知封包傳送路徑的情形下，我們如何修改 multi-hop forwarding scheme 來將封包傳給 next hop node。

4.2.1 Support Ad-hoc Routing Protocol

一般的 routing protocol 在尋找路徑時都會發出一種特殊的 broadcast IP 封包，以章節 3.2.1 所介紹的 DSDV 為例，DSDV 會定時和鄰居做 routing table 的交換。從整個 topology 的觀點來看，所有的 node 都會定時 broadcast IP packet，而在 WAVE 網路上如果想傳送 IP 封包必須加入或是提供 service，但是 topology 上只有 OBU，而且 OBU 也不會自動去建立 service。

Multi-hop forwarding scheme 解決了一部分 routing protocol 會定期 broadcast IP packet 的問題，以 SMFS 加上 DSDV 為例，DSDV 可以利用 SMFS 來定期 broadcast routing table 封包給鄰居。但是萬一所有的 node 都同時想 broadcast routing table 時，根據 SMFS 的設計，此時所有的 node 都應該要成為 provider 並且建立 WBSS，但是為了收到其他 node 的 broadcast 訊息所有的 node 也應該加入在自己附近的 WBSS。如果我們沒辦法決定最終 WBSS 該由哪些 node 來建立，那麼 broadcast routing table 就會失敗。另外，根據 WAVE 標準的定義，IP 封包都只能在 SCH interval 發送，DSDV 所 broadcast 的 routing table 其鄰居指必須要等到 SCH Interval 才有機會收到，這也許會對 DSDV 的路徑建立造成很大的 delay。

上述的兩個問題可能會拖慢甚至造成 routing protocol 尋找路徑的失敗，為了要解決這些問題，我們將這些為了 route 所 broadcast 的封包用 WSM 來取代。

WSM 不受 channel 以及 WBSS 的限制，因此我們固定讓這些 routing protocol 的 hello message 以 WSM 的形式在唯一 CCH 上傳送，如此一來只要到了 CCH interval 時所有的 node 都可以自由的交換這些 hello message。

但是，如果只是將 broadcast 封包以 WSM 的形式來發送還是會對 routing protocol 造成影響。我們在測試中發現 routing protocol 還有一些 control message 是 unicast 傳送的，例如 AODV 的 RREP。如過我們忽略了這種封包那麼還是會對某些 routing protocol 的運作造成影響，因此我們最終決定將所有 routing protocol 所使用的 control message 一律包裝成 WSM 的格式來發送。

我們的實作方法為，當 WME 抓到上層 routing protocol 要傳送的 control 封包時就將之包裝成 WSM 的格式，WME 將整個 IP 封包放在 WSM header 中的 WSM data，並且將其中的 WSM version 填入自定義的數字 2 來告知 WSM 的收端，此封包為 routing protocol 所 broadcast 的 control 封包。當 WME 收到 WSM version 為 2 的 WSM，就將其中的 WSM data 取出送給上層的 routing protocol。WME 對於此類型封包的處理流程圖請參見圖 4.8。

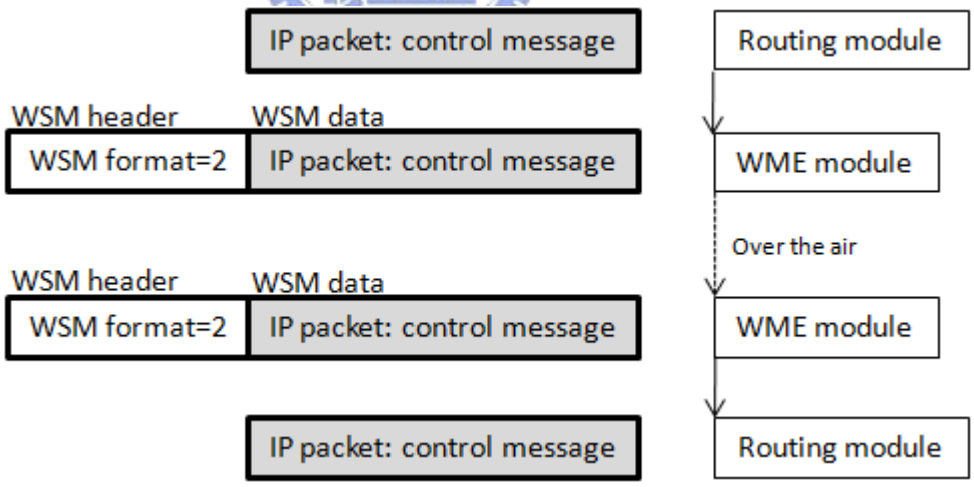


圖 4.8 WME Broadcast Control Packets for Routing

4.2.2 Support Multi-hop Forwarding Scheme

為了要在 WAVE WBSS system 上支援 multi-hop forwarding，我們必須要對原本的 WAVE MAC 以及 WME 做修正。首先我們需要把 WME 對於 IP 封包的限

制改掉，原本的 WME 會在沒有加入或是建立 WBSS 的情形下把所有送入 WME 的 IP 封包丟掉，除非使用者主動在 primitive setting file 要求 WME 建立或是加入 WBSS。我們將這種限制拿掉，並且加入自動建立 WBSS 的機制，當 WME 收到封包時它會依據 RMFS 或是 SMFS 的機制選擇來建立 WBSS，而且其它的 WAVE device 也必須要自動的加入相對應的 WBSS。

原本 WAVE MAC 有維護一個 priority queue，在 MAC de-queue 時就是我們要發動 RMFS 或是 SMFS 的時機。另外，為了使 WAVE device 可以充分的利用 SCH interval，我們必須調整 MAC queue 的 size，使得 MAC 將 queue 中的封包全部傳給對方的總時間大於 SCH interval (50 ms) 。整個系統的運作流程如圖 4.9 所示。

原本的 [10] 定義的 multi-forwarding scheme 中 sender 要將資料送往哪個 forwarder 是依據 position based routing protocol 來決定。現在因為在上層有 ad-hoc routing protocol，因此我們將就可以將封包的傳輸路徑交由其決定。

另外，為了將大家所使用的 channel 錯開，provider 在建立 WBSS 之前會先去尋找目前使用率最小的 channel 來使用。我們使用的方法為，平常就去收集其他 node 所發出的 WSA，並且找出此 WSA 所帶的 channel 參數，藉此紀錄 channel 的使用率。

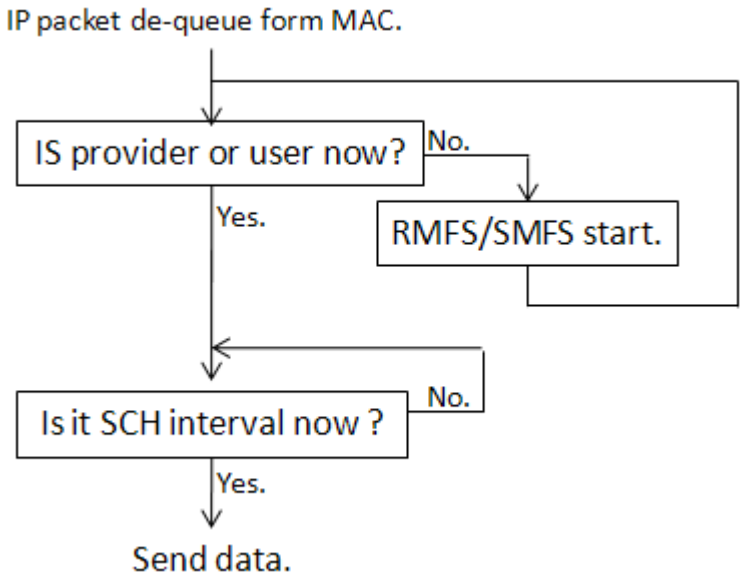


圖 4.9 The Processing Flow of an IP packet in MAC

4.2.3 RMFS Implementation

首先我們先定義 RMFS 中的 sender 與 receiver，在此所說的 sender 是表示想傳輸 MAC queue 中 data 的 node，sender 可能是原本封包的發送者也可能是 forwarder，receiver 則是要接收 sender 送出 data 的 node，receiver 可能最後 data 的目的端也可能是 forwarder。

以下將介紹 RMFS 中 sender 和 receiver 的狀態。每個 node 同時只能在一個狀態上停留，不論是 sender 或是 reveiver 其初始狀態都是 RMFS_NONE。所有的狀態除了初始狀態外其餘都是由其中一種狀態轉變而來。

Sender：

- RMFS_SENDER_WAIT_WBSS

  表示 sender 正在等 receiver 建立 WBSS。

- RMFS_SENDER_IS_USER

  表示 sender 已經加入 receiver 所建立的 WBSS 成為 user。

Reveiver：

- RMFS_RECEIVER_IS_PROVIDER

表示 receiver 因為收到 sender 的 WSM 指示而成為 provider。

章節 3.3.2 所介紹的 RMFS 並不完全相容於 WBSS-based V2V networks。以下將介紹一些 RMFS 可能發生的問題，以及解決方法：

- 假設在 topology 上有 3 個 node，A，B 以及 C，node B 在 A 和 C 之間。A 和 C 必須透過 B 才能傳送資料給對方，如果現在 A 要傳送封包給 B，同時 B 也要傳送封包給 C，此時 A 會要求 B 建立 WBSS，於此同時 B 也會要求 C 建立 WBSS，當然最佳的情形就是 B 建立 WBSS 讓 A 和 C 加入，如此既可滿足 A 與 B 也可以滿足 B 與 C 間的封包傳輸需求。但是在 RMFS 的定義中是不可能的，因為 C 是 receiver 它只會依照 B 的要求建立 WBSS 而不會去加入任何的 WBSS。在上述的情形下只能選擇由 B 或是 C 其中一個 node 建立 WBSS。我們的作法是依據 A 或是 B 的 WSM 先送出來決定，如果 A 先送出 WSM 就由 B 建立 WBSS 如果 B 先送出就由 C 建立 WBSS。
- 另外為了讓每個想傳輸資料的 node 都有機會將封包傳送出去，我們在 sender 要求 receiver 建立 WBSS 所發送的 WSM 中加入了一個 priority number，當 sender 的所要求的 WBSS 沒有建立成功，也就是 RMFS 機制失敗時 priority number 將會遞增，如果 RMFS 機制成功時則 priority number 將會歸 0。當 receiver 收到多個 sender 的要求時 reveicer 會選擇去服務擁有最高 priority 的 sender。

接下來將介紹 RMFS 中 sender 以及 receiver 的 state 轉變，配合三種事件 (sender 有資料想送，收到 WSM 要求建立 WBSS 以及收到 WSA 選擇是否加入此 WBSS) 的發生，來描述整個 RMFS 運作的流程。所有可能的組合如圖 4.10 所示，而 state 的轉變如圖 4.11 所示。

一般的情況下 sender 一開始的 state 為 RMFS_NONE，直到 sender 的 MAC de-queue packet 時，sender 的狀態變為 RMFS_SENDER_WAIT_WBSS 並且發送 WSM 給 receiver。等到 receiver 建立好 WBSS 並且 sender 確定收到

receiver 的 service advertisement 之後，sender 便加入此 WBSS 準備傳送封包，並且狀態變為 RMFS_SENDER_IS_USER。Receiver 的狀態一開始也是 RMFS_NONE 直到收到 sender 的 WSM 指示便開始建立 WBSS 發送 service advertisement 給 sender 並且狀態變成 RMFS_RECEIVER_IS_PROVIDER，如圖 4.12 所示。

在此所使用的 RMFS 因為加入了 priority 的改良而導致整個 RMFS 機制變得相當複雜，以下將藉由圖 4.10 來介紹整個 RMFS 的運作：

- 表格 (A-1)

  (A->B)：sender 想要將封包送出去因此將先發送 WSM，要求 receiver 建立 WBSS。

- 表格 (A-2)

  (A->D)：Receiver 收到 sender 建立 WBSS 的要求，receiver 開始建立 WBSS 並且將開始發送 WSA。

- 表格(B-2)

  WAVE device 收到來自其他 sender 的要求自己將身分轉變為 receiver 並且建立 WBSS。此時 WAVE device 必須要選擇自己繼續做 sender 然後把 WSM 丟棄或是 WAVE device 決定服務其他 sender 將自己從 sender 轉變為 receiver。

  首先 WAVE device 會比較所收到 WSM 內所帶的 priority 和自己的 priority：

  - (B->B)：如果自己的 priority 較大，則將此 WSM 丟棄。
  - (B->D)：如果自己的 priority 較小，則依照此 WSM 的要求來建立 WBSS 並將自己的身分由 sender 轉變為 receiver。
  - 如果 priority 相同，則判斷自己的 WSM 是否已發送出去 (可能還在 MAC queue 中等待發送)：
    - (B->D)：如果自己的 WSM 還未發送則依照此 WSM 的要求來建立

WBSS 並將自己的身分由 sender 轉變為 receiver。

◆ (B->B)：如果自己的 WSM 已經發送則將此 WSM 丟棄。

- 表格(B-3)

(B->C)：Sender 收到來自 receiver 所發送的 WSA，加入 receiver 所建立的 WBSS。

- 表格(C-2)

WAVE device 收到來自其他 sender 的要求自己將身分轉變為 receiver 並且建立 WBSS，此時 WAVE device 必須要選擇自己繼續做 sender 然後把 WSM 丟棄或是 WAVE device 決定服務其他 sender 將自己從 sender 轉變為 receiver。

■ (C->C)：如果自己的 priority 較大或相等，則將此 WSM 丟棄。

■ (C->D)：如果自己的 priority 較小，則依照此 WSM 的要求來建立 WBSS 並將自己的身分由 user 轉變為 receiver。

- 表格(C-3)

(C->B)：如果此時 sender 收到 receiver 所發送的 WSA 並且發現此 WSA 不是給自己的，表示自己之前找到的 receiver 已經去服務其他的 sender 了，自己所加入的 WBSS 可能已經無效了。

(C->C)：其他情況則維持不變。

- 表格(D-1)

當 receiver 自己想送資料時 receiver 會先比較自己的 priority 和正服務中 sender 的 priority：

■ (D->B)：如果自己的 priority 較大，則將自己轉變為 sender。

■ (D->D)：如果自己的 prority 較小或相等則維持原本的狀態。

- 表格(D-2)

Receiver 收到其他 sender 的要求，此時 receiver 會比較收到 WSM 內的 priority 正服務中 sender 的 priority：

- (D->D)：如果 WSM 的 priority 較大，receiver 將重新建立 WBSS 並且發送 WSA 給要服務的 sender。
- (D->D)：如過正服務中 sender 的 priority 較大或是相等則將此 WSM 丟棄。

- 表格(D-3)

  (D->A)：如果 receiver 收到 WSA 而且這個 WSA 是由目前 receiver 正服務中的 sender 所發送，表示自己所服務的 sender 已經變成 receiver 去服務其他的 sender 了，那麼自己所建立的 WBSS 將不再需要了。

  (D->D)：其他情況則維持不變。

在 RMFS 中不論 sender 或是 reveicer 目前為何種狀態，一旦 channel state 由 SCH 轉變為 CCH 都要將 state 變回 RMFS_NONE。此舉是為了讓所有的 node 都有機會傳輸封包，在每經歷過一段時間後想發送封包的 node 就必須要重新發送 WSM 以取得封包的發送權。

另外，為了要維護 RMFS 的這些 state，所有發送的 WSA 以及 WSM 雖然是 broadcast 但其內必須要攜帶這個封包是送給誰的資訊。至於為何不直接 unicast 來傳送 WSA 以及 WSM，原因是為了讓其他 node 也可以收到這些封包並藉此做效能的最佳化 (SCH 的選擇，WBSS 是否失效 …)。另外，這些 RMFS 的控制封包還需要帶入 priority，WAVE device 會在每次 CCH interval 開始時決定自己的 priority 是該遞增或是歸零。

另外，如果 sender 的狀態一直都維持在 RMFS_SENDER_WAIT_WBSS，表示 sender 沒有找到他想要的 receiver 為其建立 WBSS，因此 sender 如果留在 RMFS_SENDER_WAIT_WBSS 的狀態一段時間之後，sender 必須重新發送 WSM，去要求 receiver 建立 WBSS。

相較於 SMFS，RMFS 的 sender 在送封包之前要多發送 WSM 給 user，看起好像 SMFS 較占優勢，其實不然，在 RMFS 中多餘的 WSM 可以用來作一些控制資料的傳輸 (例如 sender 的優先權，sender 要求建立 WBSS 的時間 …)，

RMFS 可以很容易的使用這些訊息來幫助其盡可能的找到一個 receiver。相較之下 SMFS 並無此能力，一但網路上的 sender 一多 RMFS 就可以比 SMFS 更穩定而且精準的為 sender 找到 receiver。

WAVE system 天生就有個缺點，provider 不會知道有那個 user 加入了此 WBSS 除非 user 傳輸資料給 provider。從 WAVE service 中 provider 的觀點來看，使用 RMFS 的機制可以讓其確定 user 為誰，減少 WBSS 資源的浪費。RMFS 這彌補了 WAVE system 先天的缺憾，如此一來 service 的資源的控制將可以非常精準，不會像一般的 WAVAE service 一樣常常有 provider 建立 WBSS 但是卻沒有 user 加入之。

State \ Event	None (A)	RMFS_SENDER_WAIT_WBSS (B)	RMFS_SENDER_IS_USER (C)	RMFS_RECEIVER_IS_PROVIDER (D)
Want to send packet (1)	A->B	X	X	(My priority V.S Serviced sender's priority) D->B D
Received WSM (2)	A->D	(Received WSM's priority ?) (My WSM has sent ?) B->D B	(Received WSM's priority ?) C->D C	(Received WSM's priority ?) D
Received WSA (3)	X	B->C	(Received WSA's priority ?) (Is WSA sent by my provider/receiver ?) C->B C	(Received WSA's priority ?) (Is WSA sent by my user/sender ?) D->A D

圖 4.10 The State-transition Table of RMFS

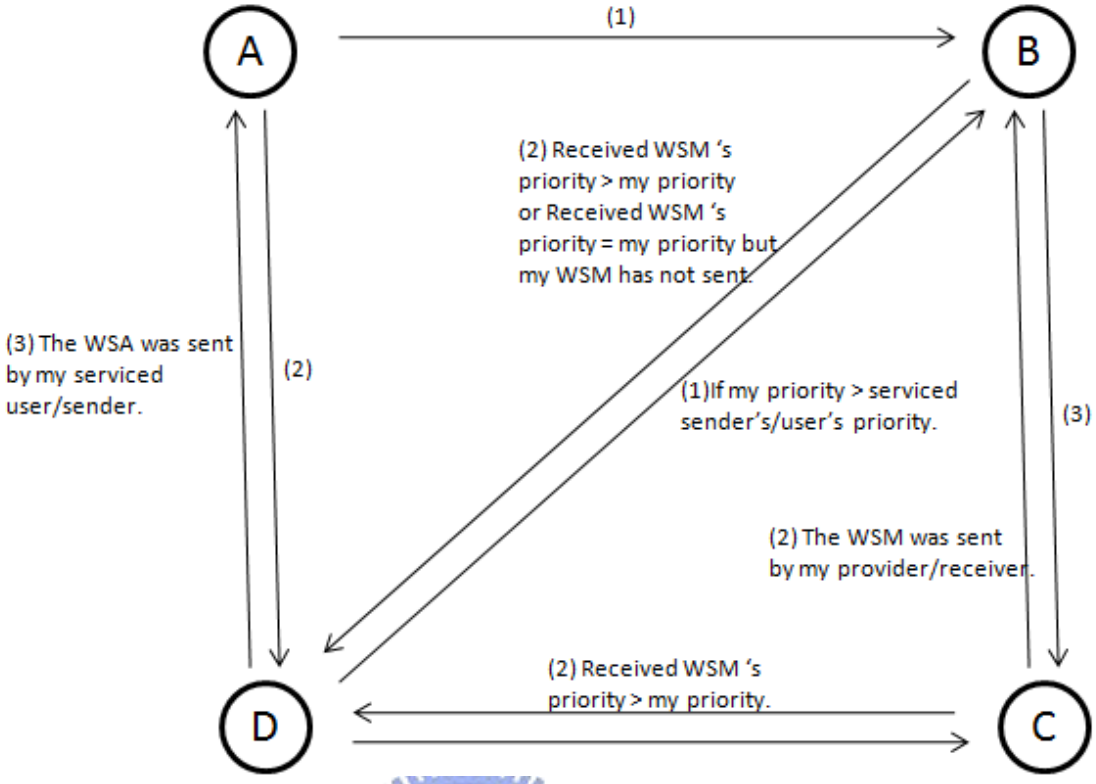


圖 4.11 The State-transition Diagram of RMFS

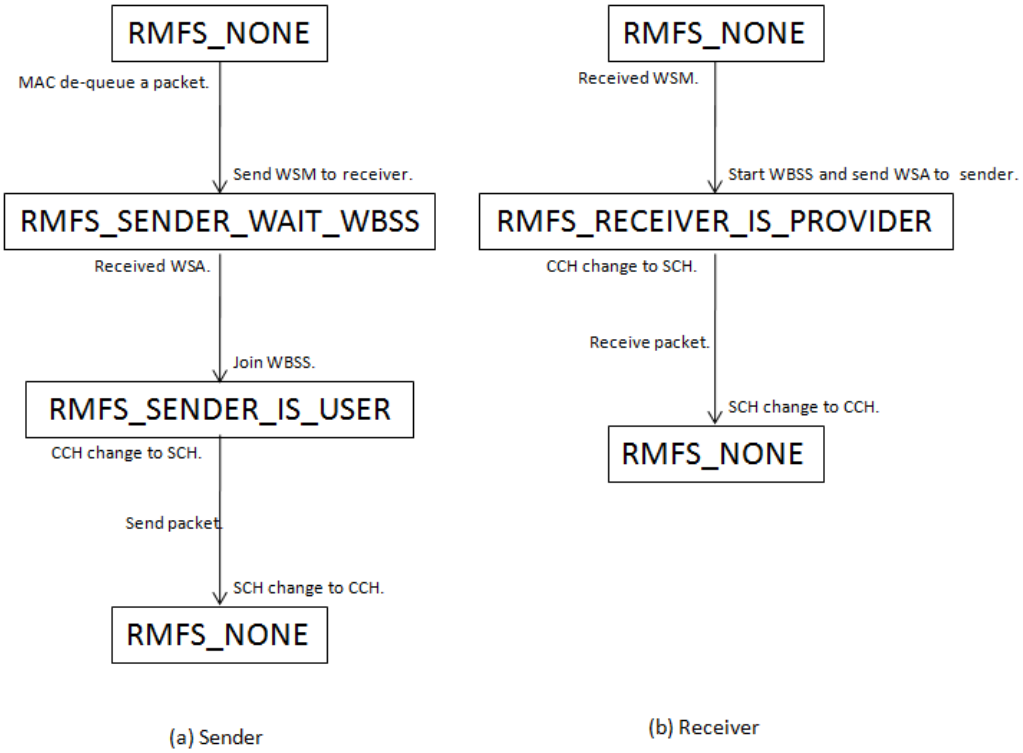


圖 4.12 The State-transition Diagram of RMFS from the Perspectives of a Sender and a Receiver

4.2.4 SMFS Implementation

SMFS 少了 RMFS 一開始發送的 WSM 以及 provider 和 user 對於 receiver 和 sender 的身分對調外，其他的部分幾乎一模一樣。以下先介紹 SMFS 的狀態，同樣的，sender 和 receiver 有一個共同的初始狀態 SMFS_NONE 所有的狀態都是由此狀態轉變而來：

Sender：

- SMFS_SENDER_IS_PROVIDER

  表示 sender 想要發送封包給 receiver，建立 WBSS。

Receiver：

- SMFS_RECEIVER_IS_USER

  表示 receiver 收到 sender 的 service advertisement 並且加入 WBSS。

在 WAVE ad-hoc network 上運行 SMFS 有個很大的問題。我們考慮一種情況如下，如果現在有兩個 node 同時想送資料給對方，此時双方會同時建立 WBSS，最後沒有人會成為 user 導致雙方的封包永遠傳不出去。我們的修正如下，node 在自己的 service advertisement 還未發送出去之前如果先收到對方的 service advertisement 就取消 WBSS 的建立並且加入對發所建立的 WBSS。

以下將介紹 SMFS 中 sender 以及 receiver 的 state 轉變，配合兩種事件 (sender 有資料想送以及收到 WSA 選擇是否加入此 WBSS) 的發生，來描述整個 SMFS 運作的流程，所有可能的組合如圖 4.13 所示，而完整的 state 轉變如圖 4.14 所示。

在一般的情況下，sender 一開始的 state 為 SMFS_NONE。當 sender 的 MAC de-queue packet 時 sender 將建立 WBSS 以提供 service 並且將狀態變為 SMFS_SENDER_IS_PROVIDER 直到 CH state 由 CCH 轉變為 SCH sender 的 state 才又變回 SMFS_NONE。Receiver 的狀態一開始也是 RMFS_NONE 直到收到 sender 的 service advertisement 便加入此 WBSS 並且將狀態改為 SMFS_RECEIVER_IS_USER，如圖 4.15 所示。

在此所使用的 SMFS 因為加入了 priority 的改良而導致 SMFS 機制變得比較複雜，以下將藉由圖 4.13 來介紹整個 RMFS 的運作：

- 表格(A-1)

  (A->B)：Sender 想要將封包發送出去，因此建立 WBSS 並且發送 WSA 期望 receiver 加入此 WBSS。

- 表格(A-2)

  (A->C)：Receiver 加入 sender 所建立的 WBSS。

- 表格(B-2)

  WAVE device 收到來自其他 sender 的 WSA 要求自己將身分轉變為 receiver 並且加入此 WBSS。此時 WAVE device 必須要選擇自己繼續做 sender 然後把 WSA 丟棄或是 WAVE device 決定加入其他 sender 所建立的 WBSS。

  首先 WAVE device 會比較所收到 WSA 內所帶的 priority 和自己的 priority：

  - (B->B)：如果自己的 priority 較大，則將此 WSA 丟棄。
  - (B->C)：如果自己的 priority 較小，則依照此 WSA 的要求來加入 WBSS 並將自己的身分由 sender 轉變為 receiver。
  - 如果 priority 相同，則判斷自己的 WSA 是否已發送出去 (可能還在 MAC queue 中等待發送)：
    - (B->C)：如果自己的 WSA 還未發送則依照此 WSA 的要求來加入此 WBSS 並將自己的身分由 user 轉變為 receiver。
    - (B->B)：如果自己的 WSA 已經發送則將此 WSA 丟棄。

- 表格(C-2)

  (C->C)：如果此時又收到其他 sender 的 WSA 在此 WSA 所帶的 channel 資訊與原本自己所加入的 WBSS 相同的情況下，直接加入此 WBSS，也就是 reveiver 可以同時服務多個 sender。如果 WBSS 使用的 channel 不

同則 reveiver 會選擇 priority 最高的 WBSS 加入。

SMFS 優於 RMFS 之處在於其不需要事先準備動作就開以開始建立 WBSS，而且在 SMFS 中 receiver 可以同時服務多個 sender。同樣的 SMFS 的缺點也很明顯，它不能夠傳輸額外的控制封包，provider 也不知道到底 user 有沒有加入 WBSS，如果 user 有加入那還好，如果 user 沒加入的話就表示 sender 浪費了整個 SCH interval。但是 SMFS 還是有其存在的價值，畢竟 SMFS 在 WBSS-based V2V networks 上使用最簡單的機制來建立 WBSS。

State \ Event	None (A)	SMFS_sender_provider (B)	SMFS_receiver_user (C)
Want to send packet (1)	A->B	X	X
Received WSA (2)	A->C	(Received WSA's priority ?) (My WSM has sent ?) B->C B	C

圖 4.13 The State-transition Table of SMFS

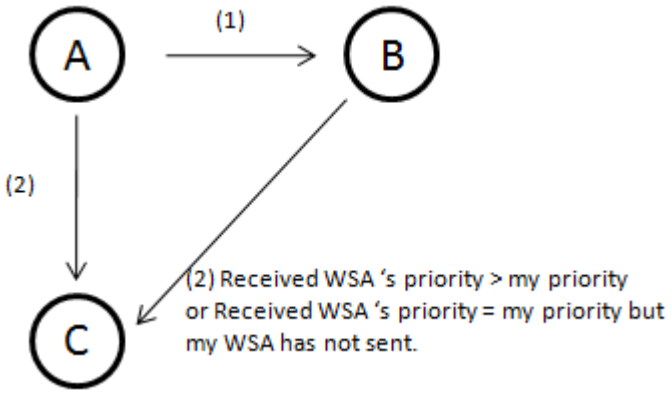


圖 4.14 The State-transition Diagram of SMFS

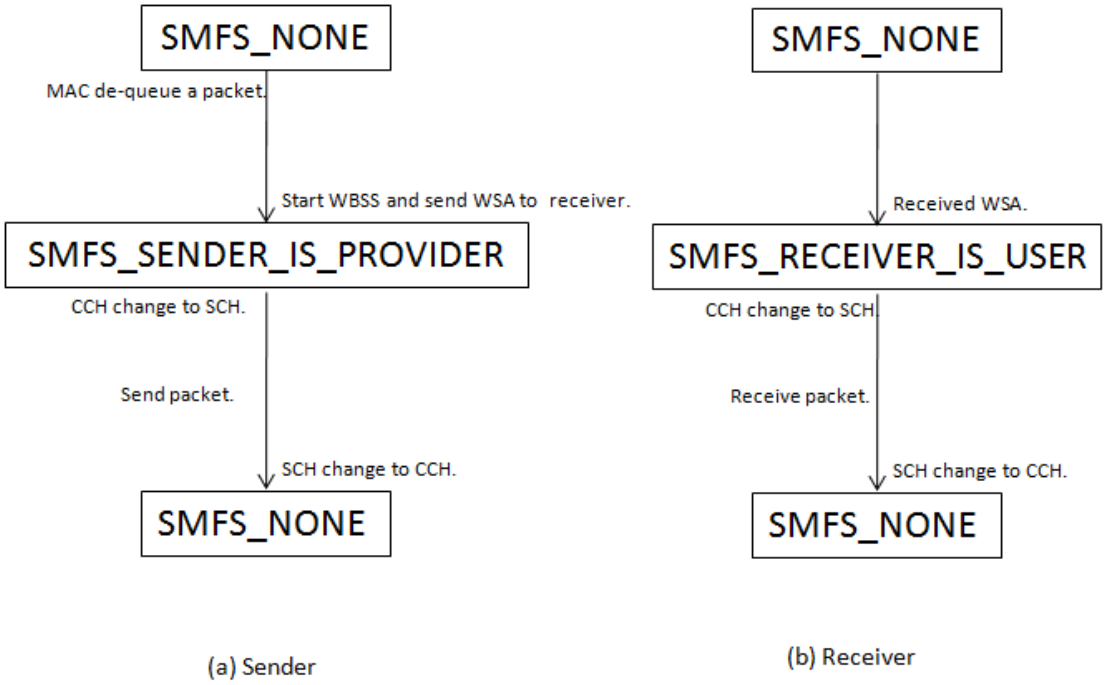


圖 4.15 The State-transition Diagram of SMFS from the Perspectives of a Sender and a Receiver

4.2.5 The priority design for SMFS and RMFS

在章節 4.2.4 以及 4.2.5 中我們曾經提到 priority 的設計，但是並沒有做詳細的介紹。我們所加入的 priority 機制主要是為了讓所有的 node 都可以公平的享有封包發送的機會，當某個 node 收到許多個來自其他 node 的 multi-hop forwarding 需求時，它會先去服務 priority 較大的 node。

一開始大家的 priority 都是 1，priority 只有在某個開始 multi-hop forwarding 機制的 node 失敗時才會遞增。另外，如果 node 發現上個 channel interval 自己已經成功的運行 multi-hop forwarding 機制，那麼 node 的 priority 就會設為 0。

我們利用一個 2-hop forwarding 的例子來說明特地將 priority 降為 0 的好處，圖 4.16 中 node 1 會透過 node 2 將封包傳送給 node 3。一開始大家的 priority 都是 1 而且只有 node 1 想送資料給 node 2，當 node 2 收到來自 node 1 的封包之後 node 1 和 node 2 都同時想發送封包。如果我們沒有將 node 1 的

priority 設為 0，那麼 此時 node 1 與 node 2 的 priority 就會相同為 1，node 1 會與 node 2 競爭發送封包的機會，如果 node 1 贏了就會導致 node 3 必須要等 3 個 channel interval 才能收到封包，最後 node 3 常常需要等待 2 個 channel interval 以上才能收到封包。如果依照我們的設計方式，在 node 1 傳送封包給 node 2 之後就將 node 1 的 priority 設為 0，便可以讓 node 2 在此時一定會獲得封包發送的機會，最後 node 3 每等待 2 個 channel interval 就一定可以收到封包，也使得 node 3 平均收到的封包個數更多。使用一發送完封包就把 node 的 priority 設成 0 的機制，在圖 4.16 的 topology 中最多可以讓 throughput 提昇 50%。

圖 4.16 2-hop forwarding example

4.2.6 SCH Selection for RMFS and SMFS

本章節所要介紹的是我們所使用的 SCH 選擇演算法。在此是由 provider/WBSS 的建立者來做 SCH 的選擇，provider 平時會收集其他 node 所發出的 WSA，並且將其所使用的 SCH number 記錄在自己的 SCH table 中，直到自己要建立 WBSS 時便會去 SCH table 中尋找使用率最小的 SCH，並且將此 SCH number 填入自己所要發出的 WSA 中。

以圖 4.17 的情況為例，provider 收到一個 WSA 並且將此 WSA 所使用的 SCH (SCH 2) 記錄在自己的 SCH table 中。直到 provider 自己想建立 WBSS 時

便在自己的 SCH table 中尋找使用率最小的 SCH (SCH 5)，並且將此 SCH number 填入自己的 WSA 中。

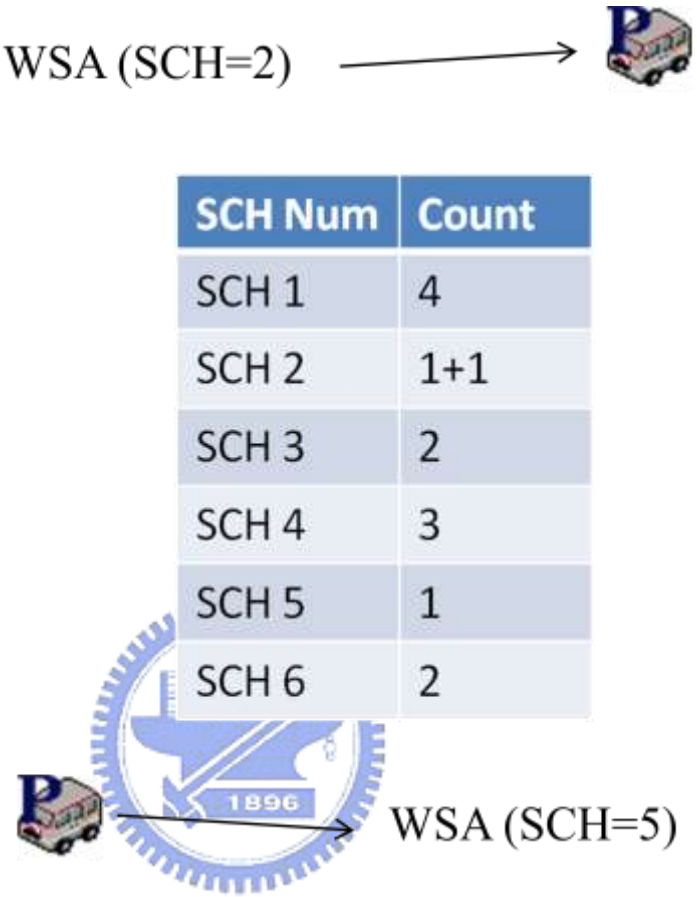


圖 4.17 The scenario for SCH selection

Chapter 5 Performance Evaluation

本章節主要展示的結果有兩項。其中之一為模擬常用的 ad-hoc routing protocol 運行在 WAVE ad-hoc network 之上，並且比較這些適用在一般的無線網路 (IEEE 802.11a) 上的 ad-hoc routing protocol 是否也適用在 WBSS-based V2V networks。另一項則是在 WAVE MAC 依據上層所決定的路徑利用 multi-hop forwarding 機制來傳輸封包，並且將比較 802.11a 與 802.11p 的封包傳輸效率。

5.1 Simulation Setting

我們挑選 NCTUns 作為實驗平台，並在其上開發我們所需要的功能，最終利用模擬的方法來呈現結果。

我們將 802.11(a)(p) 的 bandwith 固定為 6 Mbit/sec。另外，考量到 802.11a 的傳輸範圍較 802.11p 還小，因此我們調整 802.11a 的發送功率，使得 802.11a 的傳輸範圍與 802.11p 相同。

5.1.1 Simulation Tool

本篇論文使用 NCTUns-5.0 網路模擬器來進行實驗。NCTUns 是一個開放原始碼的網路模擬器，並提供完整網路架構以及開發環境，我們可以很快的在 NCTUns 上修改或是新增想要的功能。另外，NCTUns 所提供的 GUI 相當方便，我們可以很快速利用其建立 topology 並且設定好模擬參數。

5.1.2 Simulation Topology

本篇論文主要有三個類型的 topology，所有 topology 中的 node 都不會移動，圖 5.1 所顯示的 topology 類型中所有的 OBU 都只能收到相鄰 OBU 所發送的封包。這類型的 topology 共有五種，其差別只在於送端到收端的 hop count 個數 (從 1 個 hop 到 5 個 hop)。以 3 個 hop 為例，此時 topology 上就只有 node 1，

node 2 以及 node 3。Topology 中 node 與 node 間的距離為 700 公尺。

圖 5.2 所示的 topology 中，node 1 要與 node 4 交換訊息必須透過 node 2 或是 node 3 的幫助 (forwarding)。

圖 5.3 所示的 topology 中有 12 個 node 彼此都在互相的干擾範圍內。以 node 1 為例，noed 1 在傳送資料的同時如果其他 11 個 node 也同時有傳送資料的需求，node 1 將會受到其他 11 個 node 的干擾。

圖 5.4 所示的 topology 為網狀網路，其中每個 node 的傳輸範圍為其進鄰近的上下左右四個 node，以 node 8 為例，node 8 的傳輸範圍為 node 2，node 7，node 9 以及 14。

圖 5.1 Chain Topology

圖 5.2　Broken Link Topology

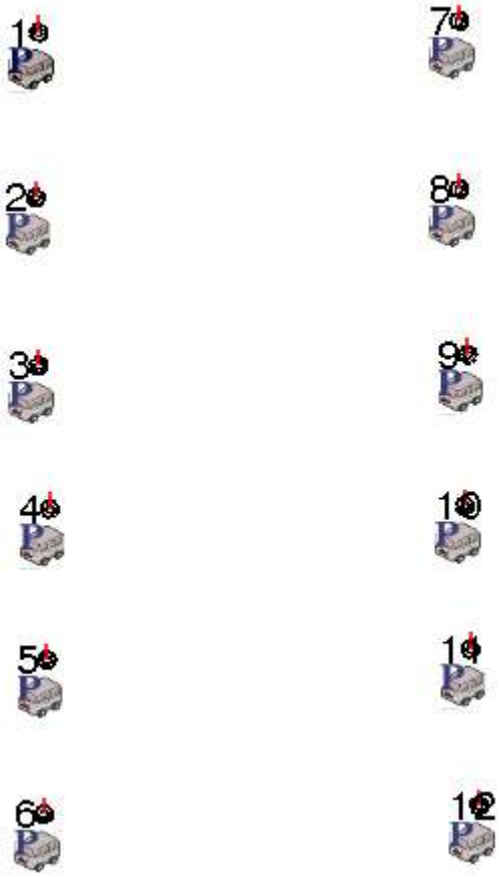


圖 5.3　One-to-one Topology

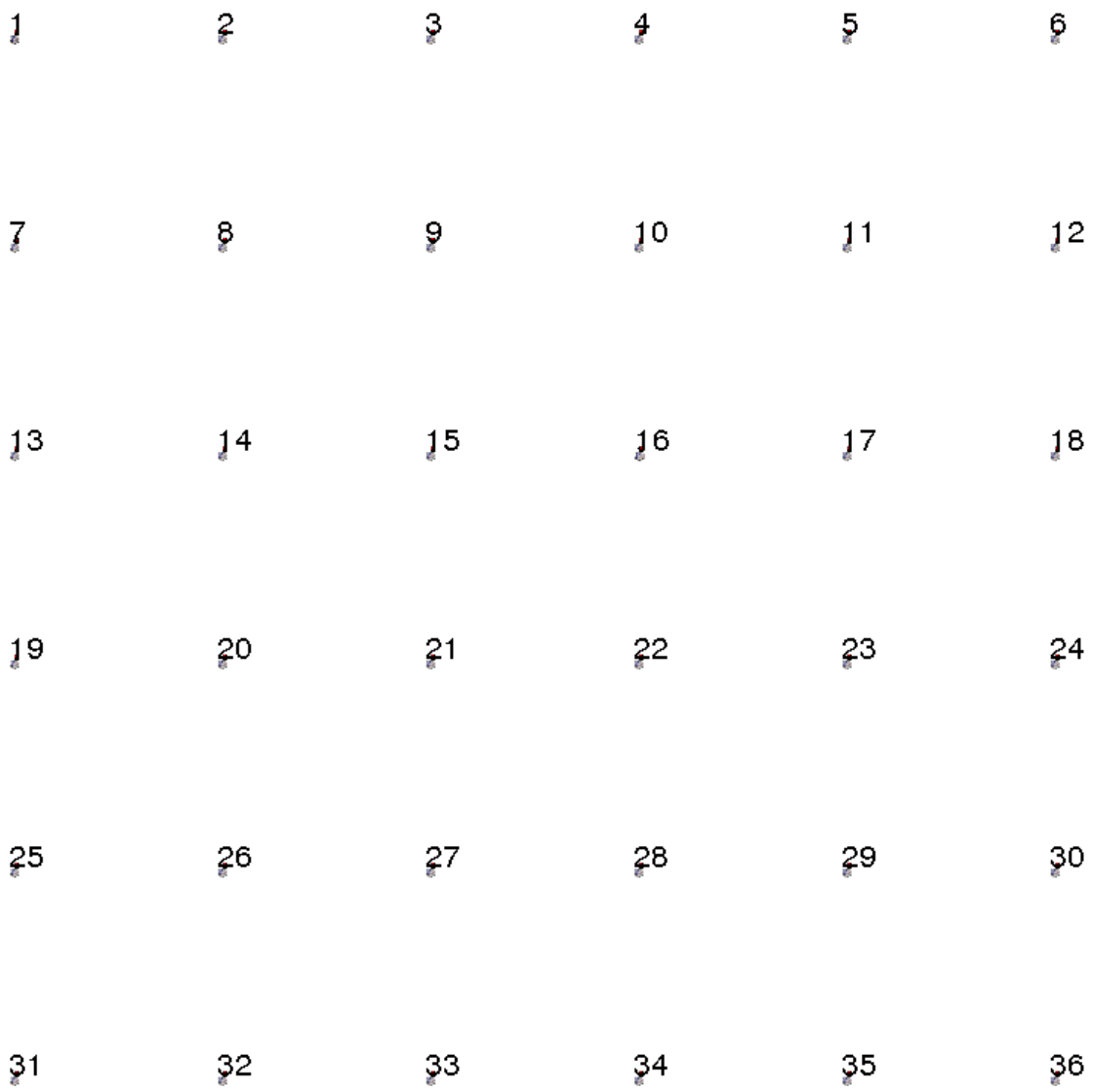


圖 5.4　Grid Topology

5.1.3　Simulation Metrics

以下將定義模擬過程中所有效能的計算方式：

- 流量 (Throughput)

  Throughput 的計算是由收端 application 以秒為單位統計每一秒所收到的封包大小加總。

- 延遲時間 (Delay)

在此我們需要量測的 delay time 有兩種，其中一種是 routing protocol 尋找封包由送端到達收端的路徑所需的時間 (Path finding time)。另一種是當封包傳送路徑斷掉時 routing protocol 回復路徑所需要的時間(Path recovery time)。

- 流量下降比率 (Throughput reduction percentage)

  流量下降比率是我們用來測量由於我們設計的演算法不完美而造成的流量降低比率，我們會用某些特定的方式來測量最佳的 throughput 以做為比較的基準。

5.1.4 Traffic Generation

以下將定義模擬過程中所有封包的產生方式：

- Greedy UDP

  Application 將會不斷的產生封包，目的為佔滿可用的頻寬。

5.2 Simulation Results

5.2.1 Routing Protocol Performance Results over Chain Topology

我們想要知道在 802.11p/1609 的網路環境下，利用我們所設計的方法對於 routing protocol 路徑的建立有何影響。在此將分別比較 AODV 與 DSDV 在 802.11p 以及 802.11a 網路上的效能。

利用圖 5.1 的 topology 我們可以測量 AODV 與 DSDV 在不同 hop count 的情形下建立路徑所需要的時間。AODV 與 DSDV 的結果將分別以圖 5.5 以及圖 5.6 展現。

在 802.11p 網路上 AODV 建立 ptah 的流程中 sender 會先 floading RREQ 之後才由 receiver 將 RREP 用 unicast 的方式傳回給 sender。依照我們

在章節 4.2.1 一開始所做的設計，RREQ 會包裝成 WSM 並且在 CCH interval 傳送。但是由於 RREP 由於是 unicast 的 IP 封包，因此所們將利用 multi-hop forwarding 的方式來傳送，而這將會對 AODV 造成很大的影響。因為 RREP 每經過一個 hop 就會多 100ms (SCH interval 加上 CCH interval) 的 delay，因此 AODV 的路徑建立時間便會隨著送端與收端間 hop 的增加而增加。我們可以從圖 5.5 中的 802.11p AODV-RREP-RMFS 與 802.11p AODV-RREP-SMFS 這兩條線看出趨勢。

如果因為我們的設計不良而造成 AODV 效能的低落，那麼我們就必須要做修正。我們的改進方法是將這些 routing protocol 的 control 封包全部都包裝成 WSM 的格式，並且將這些 control 封包都在 CCH interval 傳送，如此便可以讓 routing protocol 順暢的在 802.11p 網路上運行並且不受到 channel 切換的影響。我們為 AODV 所做的修正效能呈現在圖 5.5 中的 802.11p AODV-RREP-WSM 這條上。我們可以利用圖 5.5 來證明使用我們所設計的方法之後，AODV 在 WBSS-based V2V networks 上建立路徑所需要的時間與 802.11a ad-hoc network 沒有太大的差異。

另外，DSDV 建立路徑的過程中只會 broadcast 自己的 routing table 給附近的 node，使用我們所設計的方法也不會對 DSDV 建立路徑的流程造成負面影響，圖 5.6 的結果顯示我們的推測是正確的。

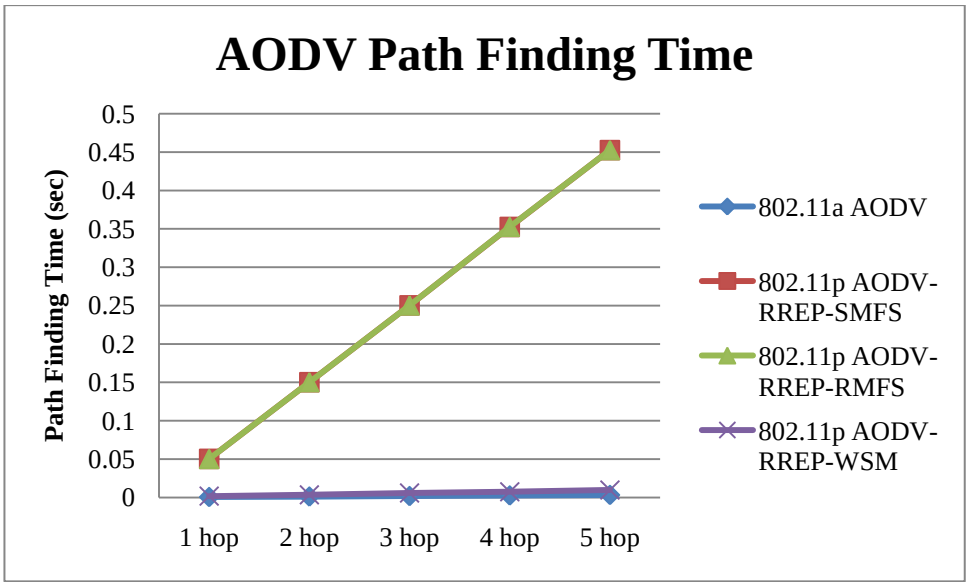


圖 5.5 AODV Path Finding Time over Chain Topology

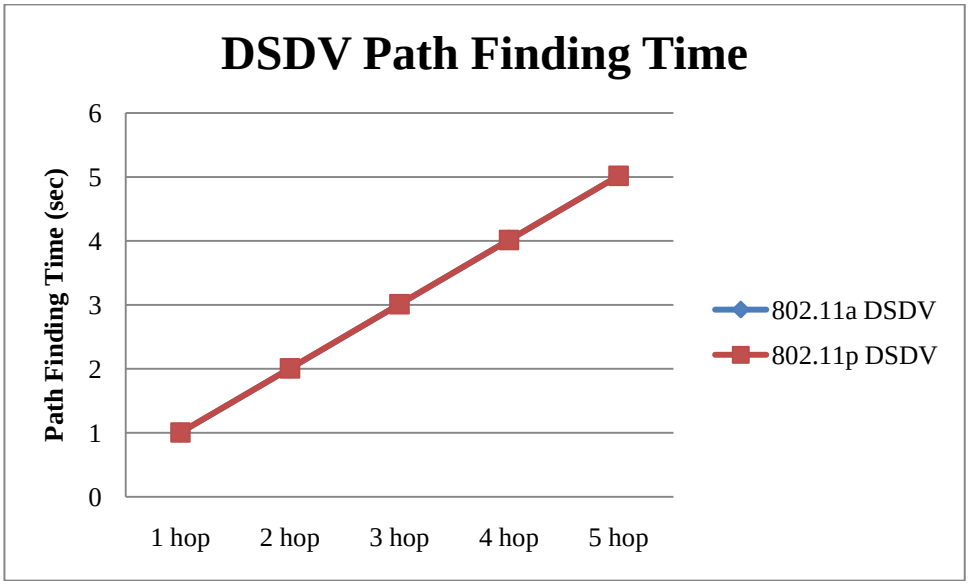


圖 5.6 DSDV Path Finding Time over Chain Topology

5.2.2 Routing Protocol Performance Results over Broken Link Topology

Routing protocol 除了路徑的尋找時間可以用來作為效能量測的比較基準外，路徑的回復時間也可以用來作為效能量測的指標。

我們想要知道在 802.11p/1609 的網路環境下，利用我們所設計的方法對於 routing protocol 路徑的回復有何影響。在此將分別比較 AODV 與 DSDV 在 802.11p 以及 802.11a 網路上的效能。

路徑有的回復的需求通常是因為原本的路徑不見了。在此我們利用圖 5.2 的 broken link topology 來模擬路徑斷掉的情境。在圖 5.2 中原本 node 1 是透過 node 2 將封包傳送給 node 4，而 node 3 在一開始並不存在 topology 中，直到第 10 秒我們將 node 2 移出 topology 並且將 node 3 移入，強迫 node 1 將原本傳輸封包的路徑 1-2-4 改為 1-3-4，並且量測 node 回復路徑所需要的時間。

我們已經可以預測使用 DSDV 不論是在 802.11a 或是 802.11p 網路上結果都不會有太大的差別，因為 DSDV 是靠每一秒發送自己的 routing table 給附近的 node 來維護路徑的原因。而 DSDV 發送 hello message 的時間又剛好是在 CCH interval，因此就算是在 802.11p 上一直做 channel 的切換也不會造成任何的影響。

如果使用的 routing protocol 是 AODV，802.11p 不斷的切換 channel 就會對 AODV 的路徑回復造成影響。最糟的情形是路徑斷掉的時間在 SCH interval，此時 AODV 必須要等待 50ms 直到 CCH interval 時才能發現路徑的錯誤。AODV 必須要在 CCH interval 才能做路徑的重建。如果 WAVE device 在 SCH interval 發送 routing protocol 的 control 封包來重建路徑，可能會發生收端與送端在不同 SCH 的問題，最後導致這些 control 封包無法送達收端。我們只有在 CCH interval 才能確保所有的 node 都留在相同的 channel (CCH)，也只有在此時發送 routing protocol 的 control 封包才能發揮效用。圖 5.7 的結果證明了我們的論述。

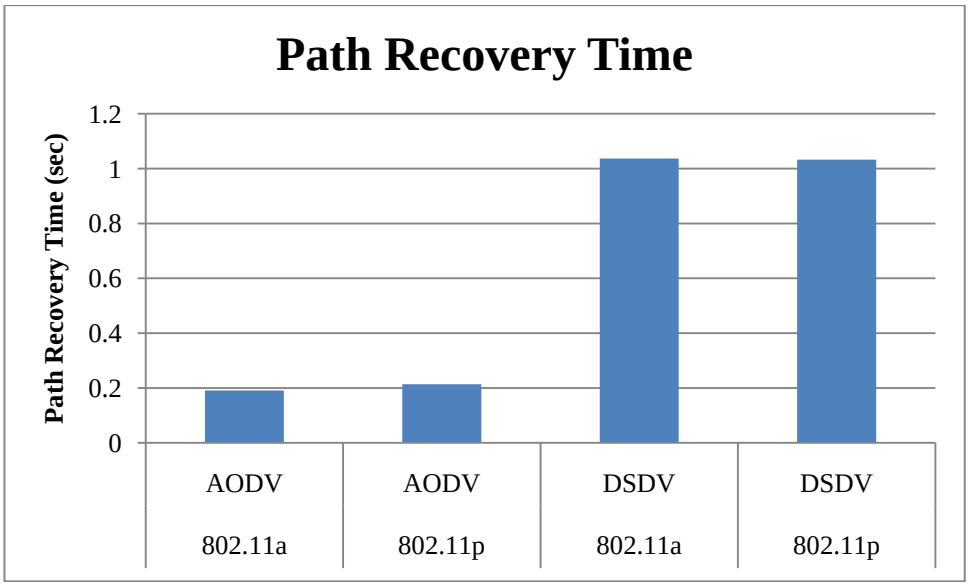


圖 5.7 Path Recovery Time over Broken Link Topology

5.2.3 Data Transfer Performance Results over Chain Topology

在上層的 routing protocol 為我們找到路徑之後，接下來我們便要考慮 MAC layer 要如何將資料傳送出去了，本章節將呈現 802.11a 與 802.11p MAC layer 對於資料傳的效能結果。

我們利用 chain topology 來做模擬，並且比較 802.11a 與 802.11p 對於資料的傳輸在不同 hop count 下的效能。另外，為了呈現 802.11p 的優點，在 802.11p 上傳輸資料時，無論是利用 RMFS 或是 SMFS 我們都將開啟 SCH 自動選擇。

圖 5.8 的結果顯示了 802.11p 分別利用 SMFS 以及 RMFS 來幫助資料的傳輸與 802.11a 資料傳輸的效能比較，圖中我們可以發現隨著 hop count 的增加 throughput 會逐漸的下降，但是在 2 個 hop 之後 802.11p 的 throughput 下降比率比 802.11a 還來的小，造成這種結果正是因為我們開啟了 SCH 自動選擇的關係。

我們以 chain topology 來舉例說明在 802.11p 將 SCH 錯開的好處。假設現在在 chain topology 上有 4 個 node，node 1 將透過 node 2 以及 node 3 的幫助將資料傳送給 node 4。如果我們沒有將 SCH 錯開，當 node 1 傳送資料給 node 2 時，node 3 就不能傳送資料給 node 4。這是因為在無線網路的環境下資料是利

用能量波的形式來傳送，如果 node 1 和 node 3 同時使用相同的 channel 傳輸資料到空氣中，node 2 最終所收到的資料為兩者的混合。如果我們將 node 1 傳送資料給 node 2 的 SCH 與 node 3 傳送資料給 node 4 的 SCH 錯開，那麼資料就可以同時傳輸，throughput 在兩個 hop count 之後就不會隨著 hop count 的增加而繼續向下掉了。

在圖 5.8 中，802.11p 的 throughput 在兩個 hop count 之後還是繼續向下掉，這有兩種可能，其中一個為我們所設計的 SCH 自動選擇演算法並不完美，導致至 channel 無法完全錯開所造成，另一個為我們所改進的 RMFS/SMFS 並不完善，造成 sender 無法與 reveicer 配合建立/加入 WBSS 的情形。

為了要分析 RMFS/SMFS 以及 SCH 自動選擇演算法的對於 throughput 所造成的影響，我們首先量測 SCH 自動選擇演算法所造成的 throughput 下降比率。我們讓 chain topology 上所有的 node 在建立 WBSS 時都會為不同的 node 選擇不同的 SCH，藉此以達到 SCH 的最佳選擇。結果如圖 5.9 所示，在圖 5.9 中我們發現 RMFS 的設計相當優異，即使超過 2 個 hop 之後 throughput 也不會向下掉。如此，我們便可以排除因為 RMFS 的設計不良而造成 throughput 的下降。利用 RMFS 的結果，我們可以從圖 5.8 中的 ”802.11p RMFS” 與圖 5.9 中的 “802.11p RMFS-Best-SCH-Selection” 分析出單純因為 SCH 選擇演算法的不佳而造成 throughput 的下降率，結果如圖 5.10 所示。

因為 RMFS 在 SCH 選擇演算法完美的情況下不會造成 throughput 的下降，因此我們可以 RMFS 為基準從圖 5.9 中分析出因為 SMFS 的設計不良所造成的 throughput 下降率，結果呈現在圖 5.11 中。

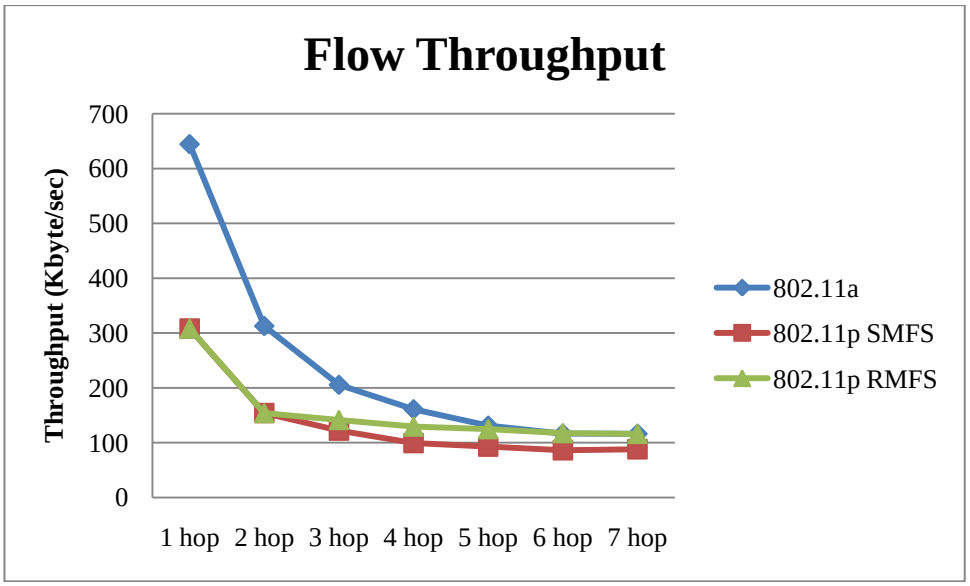


圖 5.8 Throughput over Chain Topology

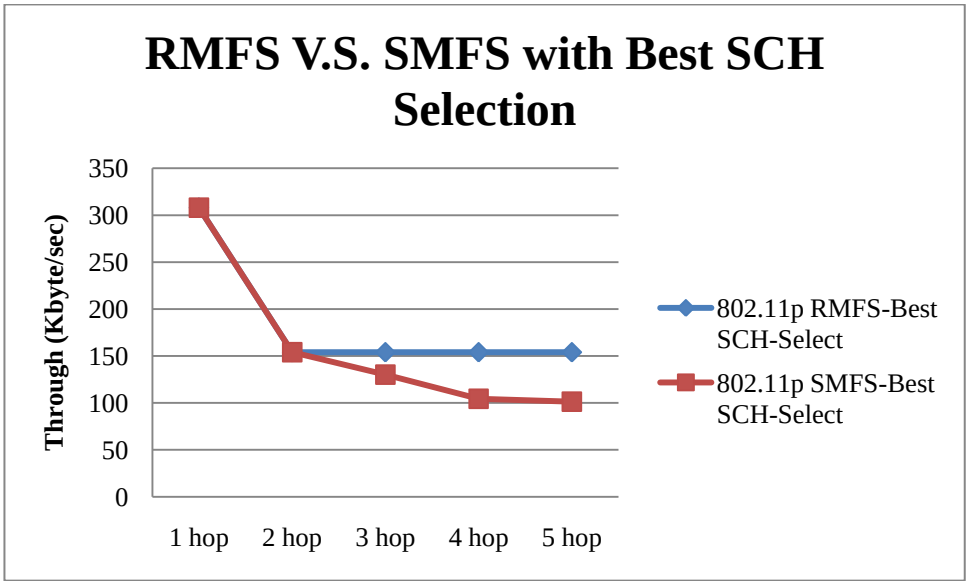


圖 5.9 Throughput over Chain Topology with Best SCH Selection

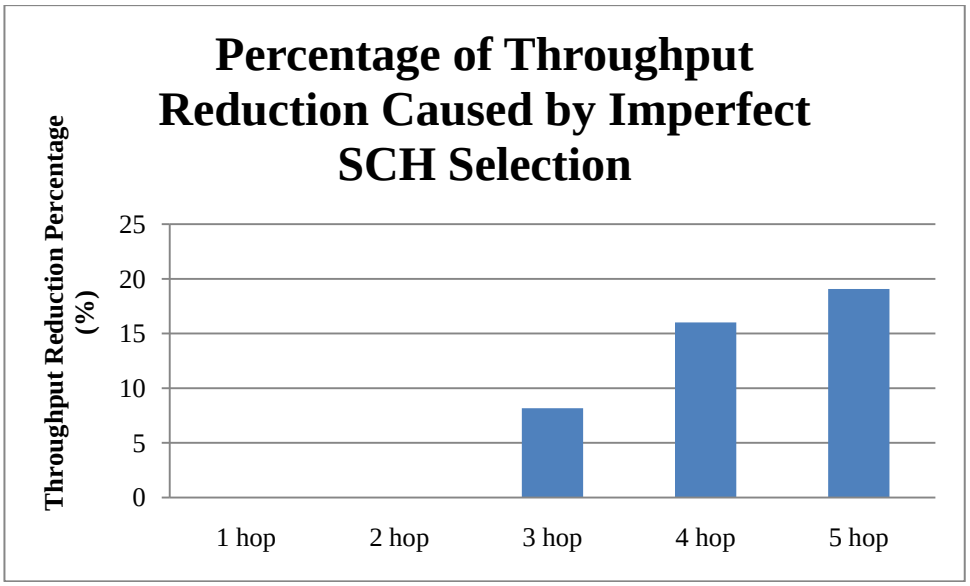


圖 5.10 Percentage of Throughput Reduction Caused by Imperfect SCH Selection over Chain Topology

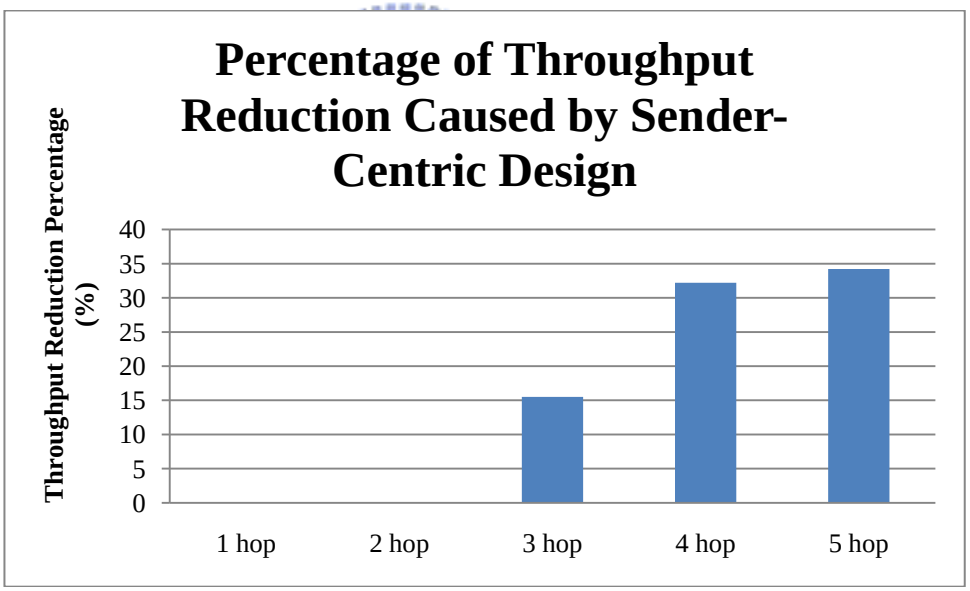


圖 5.11 Percentage of Throughput Reduction Caused by Center-Centric Design over Chain Topology

5.2.4 Data Transfer Performance Results over One-to-one Topology

在章節 5.2.3 的圖表展示中，我們看到了在 chain topology 在 5 個 hop count 之後 802.11p 的 throughput 會優於 802.11a 。在本章節我們想要量測利用我的所設計的方法在 WBSS-based V2V networks 中可使用的最大 throughput 之極限。為了展示 802.11p 的能力我們利用圖 5.3 的 one-to-one topology 來模擬隨著 data flow 從一條增加到六條，理論上 802.11p 的 throughput 將逐漸提升至 802.11a 的 3 倍。

圖 5.12 與圖 5.13 證實了我們的論點，雖然在 6 條 flow 的情況下 802.11p 配合 RMFS 所得到的 throughput 並沒有達到 802.11a 的 3 倍，但那是由於我們 SCH 自動選擇演算法所造成的失誤。

圖 5.12 以及圖 5.13 我們額外測試了一種隨機選擇 SCH 的 SCH 選擇方式，這種隨機選擇 SCH 的方法並沒有比我們所設計的 SCH 演算法來得好。以下將利用圖 5.12，圖 5.13 的結果以及數學證明來解釋。最後我們發現數學所計算出來的結果與模擬所顯示的結果相符，同時也證明了我們所設計的 SCH 選擇演算法是優於隨機選擇 SCH 的方法。

我們利用數學方法來計算在隨機選擇 SCH 演算法下每一條 flow 可以利用的 throughput 比率。以 2 條 flow 為例，我們以其中一條 flow 為觀點來考量，考慮沒有其他 node 與自己使用同一 SCH 所得到的 throughput 比率為 $\frac{6*5}{6^2}*\frac{1}{1}$，考慮有一個 node 與自己使用同一 SCH 所得到的 throughput 比率為 $\frac{6*1}{6^2}*\frac{1}{2}$，最後兩者相加再乘上單一 flow 的最大 throughput 所得到的結果即為兩條 flow 的平均 throughput。上述的計算方式在多條 flow 的情況下也適用，最後我們整理出一個公式來計算平均 throughput 的比率，$\sum_{i=1}^{n}\left(\frac{1}{\mathrm{i}}*\frac{6*1^{\mathrm{i}-1}*5^{\mathrm{n}-1}}{6^{\mathrm{n}}}*\binom{n-1}{i-1}\right)$，其中 n 代表 flow 個數。公式所算出來的結果我們呈現在圖 5.12 以及圖 5.13 中的 802.11p RMFS-RAND-THEORETICAL 這條線，並且將之與 802.11p RMFS-RAND-SCH 作比較發現兩條線幾乎是重合的，這證明了我們利用數學所

證明的結果與模擬的結果相符。

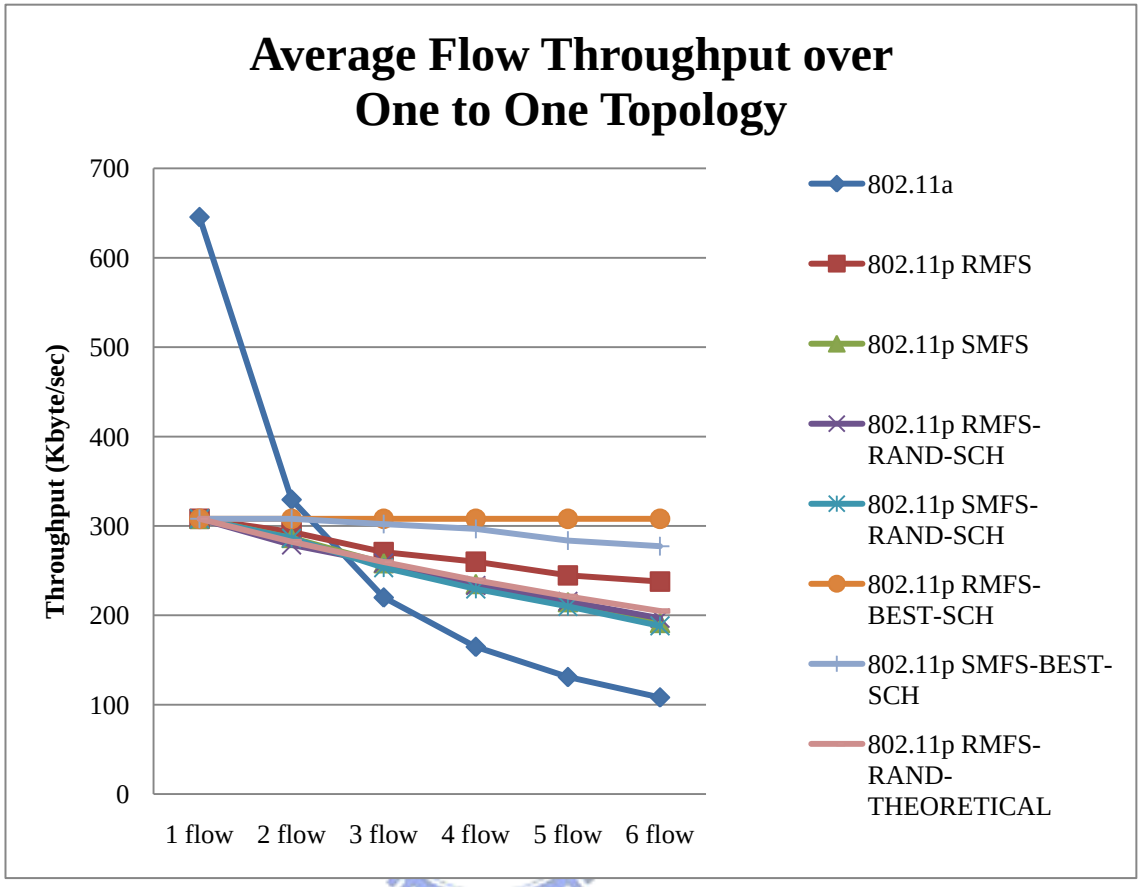


圖 5.12 Average Flow Throughput over One-to-one Topology

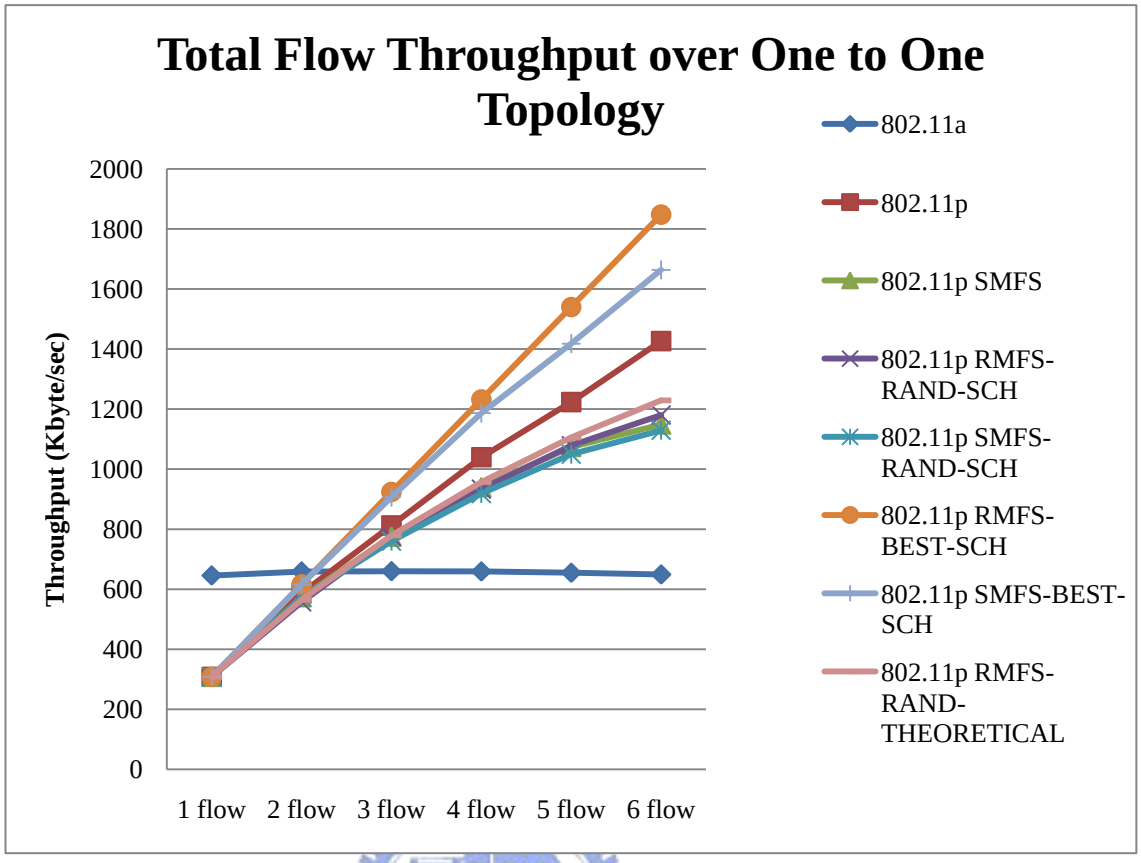


圖 5.13 Total Flow Throughput over One-to-one Topology

5.2.5 Data Transfer Performance Results over Grid Topology

在前面的章節我們使用了簡單的 topology 就可以展現 802.11p 優於 802.11a 之處，不論是在 chain topology 或是 one-to-one topology 我們都可以看見 WBSS-Based V2V networks 利用 multi-channel 特性的好處。

本章節我們利用圖 5.4 的 grid topology 來測試 802.11p 在較複雜的網路環境下所得到的效能是否比會 802.11a 來的高。在 grid topology 中我們讓每一行以及每一列都產生一條 flow，列 flow 的流向是由每一列的最左邊那一個 node 流向每一列的最右邊那一個 node，例如 node 1 傳送封包給 node 6，行 flow 的流向是由每一行的最上面那一個 node 流向每一行的最下面那一個 node，例如 node 1 傳送封包給 node 31，如果 grid topology 是由 4 乘 4 個 node 所組成那就會有 8 條 flow。

圖 5.14 以及 圖 5.15 所展示的結果說明了 802.11p 配合 RMFS 再加上我們所設計的 SCH 選擇演算法所得到的效能已經大幅超越了 802.11a，同時也說明了 WBSS-Based V2V networks 在較複雜的網路環境中所展現出來的效能比起傳統的 ad-hoc network 還來的優異。

802.11p 相對於 802.11a 的 throughput 提升在 grid topology 中並沒有像 one-to-one topology 一樣好，那是因為 grid topology 中同時會有 row flow 以及 column flow 以圖 5.4 為例，node 2 以及 node7 同時都會傳封包給 node 8，而在 RMFS 以及 SMFS 的機制中 node 8 在一個 channel interval 只能選擇服務 node 2 或是 node 7 其中一個，因此 RMFS 以及 SMFS 在 grid topology 中並不能如同在 one-to-one topology 中運行的這麼好。從圖 5.14 以及圖 5.15 的結果說明了我們所支援的 WBSS-based V2V networks 的效能還是比 802.11a 好，原因主要是我們利用了 WAVE multi-channel 的特性。

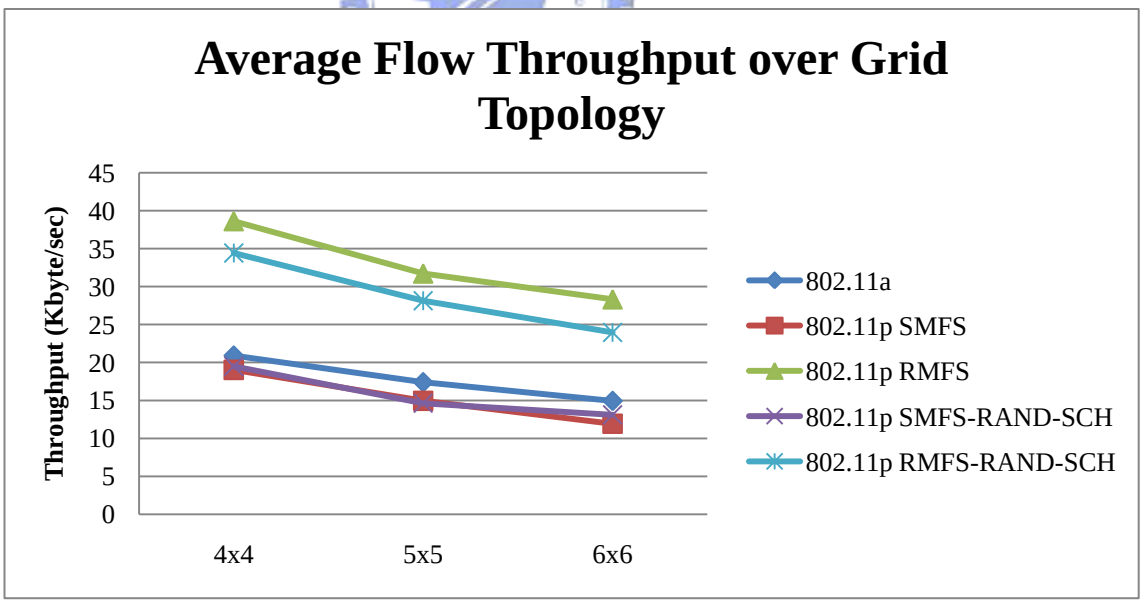


圖 5.14 Average Flow Throughput over Grid Topology

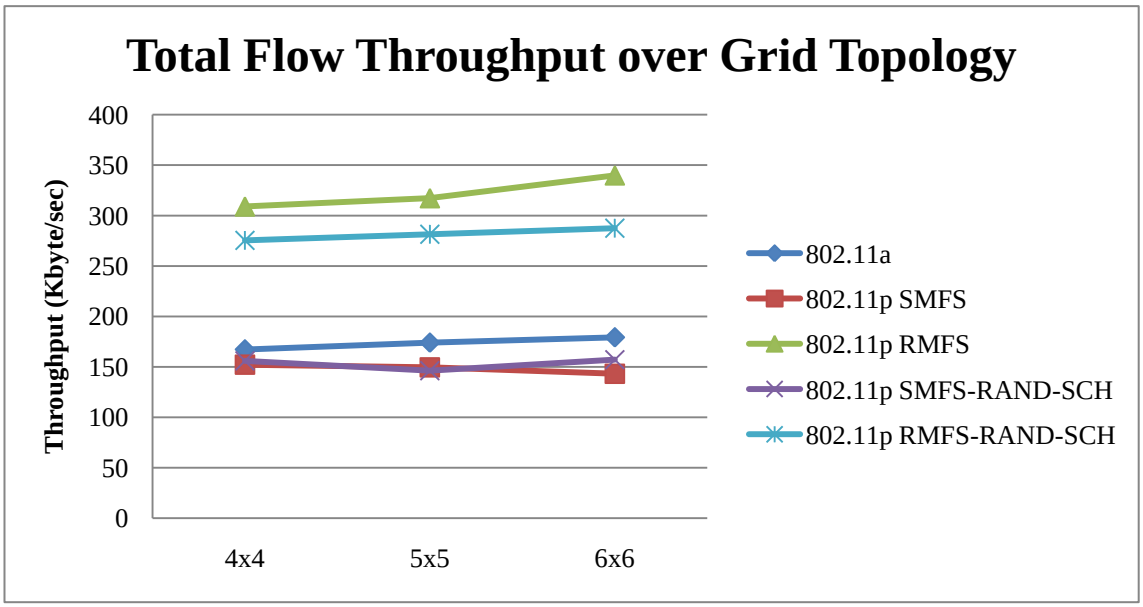


圖 5.15 Total Flow Throughput over Grid Topology

Chapter 6 Future Work

- 提升 WBSS-based V2V networks 的 routing 效能

  對於 WBSS-based V2V networks 來說 AODV 與 DSDV 並非最合適的 routing protocol，這是因為在 MAC layer 使用的 multi-hop forwarding scheme 無法完美的配合上層的 AODV 與 DSDV。若 MAC layer 發現無法與收端建立 provider 與 user 的關係時，MAC layer 可以立即回報上層的 routing protocol，讓上層做替代路徑的選擇，如此便可以避免浪費掉整個 SCH interval 的封包傳輸機會。

- 提升 WBSS-based V2V networks 的資料傳輸效能

  為了要讓 WBSS-based V2V networks 達到最佳的效能必須要有一套效能更好的 SCH 選擇演算法，以改善由於我們所提出的 SCH 自動選擇演算法不佳而造成 throughput 的降。

- 在會動的網路上分析 802.11p 資料傳輸的效能以改善 multi-hop forwarding scheme

  在本篇論文中我們所分析的 topology 都是屬於靜態的 topology，如果能在動態的 topology 上模擬資料的傳輸，並且可以完美的分析所有可能造成 throughput 下降的原因，我們還可以更加完善 RMFS 與 SMFS 的設計。

Chapter 7 Conclusion

從研究 802.11p standard 所定義的 WAVE ad-hoc network 之中我們看到它的缺點。雖然 WAVE ad-hoc network 不用建立 WBSS 就可以直接傳輸資料，但是他缺少了 multi-channel 的使用。另外，在 WAVE ad-hoc network 中所有的封包將混雜在同一個 channel 上傳送，可能導致重要的安全訊息 (例如車輛防撞訊息) 較不容易被送出去。

為此我們使用了 WBSS-based V2V networks 作為替代，改善了 WAVE ad-hoc network 種種的缺點。為了要實現 WBSS-based V2V networks，我們在其上做了 ad-hoc routing protocol 的支援並且改良 multi-hop forwarding scheme 讓 WBSS-based V2V networks 可以正常的運作。

最終，我們成功了讓 WBSS-based V2V networks 達到與 WAVE ad-hoc network 有相同的能力，並且可以用到所有 WAVE networks 提供的 channels。不論是在 routing protocol 的運作與資料的傳輸都不會比 WAVE ad-hoc network 來的差，在某些情況下 WBSS-based V2V networks 可提供的總 throughput 甚至優於 WAVE ad-hoc network。這結果不但表示了我們的想法是可行的，也為 WAVE networks 找到了一個新的研究的方向。

Chapter 8 References

[1] "Draft P802.11p/D3.0," the IEEE 802.11 Working Group of the IEEE 802 Committee, July 2007.

[2] "IEEE Std 802.11-2007 (Revision of IEEE Std 802.11-1999)," June 12, 2007.

[3] "IEEE 1609.1 Trial-Use Standard for Wireless Accesses in Vehicular Environments (WAVE) - Resource Manager," IEEE Vehicular Technology Society, October 2007.

[4] "IEEE 1609.2 Trial-Use Standard for Wireless Accesses in Vehicular Environments (WAVE) - Security Services for Applications and Management Messages," IEEE Vehicular Technology Society, October 2006.

[5] "IEEE 1609.3 Trial-Use Standard for Wireless Accesses in Vehicular Environments (WAVE) - Networking Services," IEEE Vehicular Technology Society, October 2006.

[6] "IEEE 1609.4 Trial-Use Standard for Wireless Accesses in Vehicular Environments (WAVE) - Multi-channel Operation," IEEE Vehicular Technology Society, October 2006.

[7] "IEEE P1609.0™/D0.7 Draft Standard for Wireless Access in Vehicular Environments (WAVE) – Architecture," IEEE WAVE Working Group, January 2009.

[8] "IEEE P1609.3™/D1.0 Draft Standard for Wireless Access in Vehicular Environments (WAVE) - Networking Service," IEEE WAVE Working Group, December 2008.

[9] "IEEE P1609.4™/D1.0 Draft Standard for Wireless Access in Vehicular Environments (WAVE) – Multi-channel Operation," IEEE Dedicated Short Range Working Group, December 2008.

[10] Kuang-Che Liu, “Supporting Multi-Hop Forwarding over 802.11p Networks,” M.S. thesis, National Chaio Tong University, September 2008.

[11] Charles E. Perkins, Elizabeth M, Belding-Royer, and Samir Das, “Ad Hoc On Demand Distance Vector (AODV) Routing,” IETF RFC 3561, July 2003.

[12] Charles E, Perkins, Pravin Bhagwat, “Highly dynamic Destination-Sequenced Distance-vector routing (DSDV) for mobile computers,” In Proc. ACM SIGCOMM Conference (SIGCOMM ’94), pages 234-244, August 1993.

[13] Eichler, S., “Performance Evaluation of the IEEE 802.11p WAVE Communication Standard,” Vehicular Technology Conference, VTC-2007 Fall, 2007 IEEE 66th, pp.2199-2203, Sept. 30 2007-Oct. 3 2007.

[14] Yunpeng Zang, Stibor L., Walke B., Reumerman H.-J., Barroso A., “A Novel MAC Protocol for Throughput Sensitive Applications in Vehicular Environments,” Vehicular Technology Conference, VTC2007-Spring, IEEE 65th, pp.2580-2584, 22-25 April 2007.

[15] Taha M.M.I., Hasan Y.M.Y., “A Novel Headway-Based Vehicle-to-Vehicle Multi-Mode Broadcasting Protocol,” Vehicular Technology Conference, VTC 2008-Fall, IEEE 68th, pp.1-5, 21-24 Sept. 2008.

[16] Shie-Yuan Wang, Yow-Wei Cheng, Chih-Che Lin, Wei-Jyun Hong, Ting-Wei He, “A Vehicle Collision Warning System Employing Vehicle-To-Infrastructure Communications,” Wireless Communications and Networking Conference, WCNC 2008, IEEE , pp.3075-3080, March 31 2008-April 3 2008.

[17] Maruoka T., Sato Y., Nakai S., Wada T., Okada H., “An Extended Collision Judgment Algorithm for Vehicular Collision Avoidance Support System (VCASS) in Advanced ITS,” Vehicular Technology Conference, VTC 2008-Fall, IEEE 68th, pp.1-5, 21-24 Sept. 2008.

[18] Nakjung Choi, Sungjoon Choi, Yongho Seok, Taekyoung Kwon, Yanghee Choi, “A

Solicitation-based IEEE 802.11p MAC Protocol for Roadside to Vehicular Networks," 2007 Mobile Networking for Vehicular Environments, pp.91-96, 11-11 May 2007.

[19] S. Wang, C. Chou, C. Huang, C. Hwang, Z. Yang, C. Chiou, and C. Lin. "The Design and Implementation of the NCTUns 1.0 Network Simulator," Computer Networks, 42(2):175–197, June. 2003.