# Linux 核心中 IEEE 802.15.4 通訊協定之設計與實做
# Design and Implementation of IEEE 802.15.4 Protocol Stack in Linux Kernel

陳信養、李宗憲、馮立琪　　沈仲九、李俊賢
長庚大學資訊工程研究所　　工研院電通所
lcfeng@mail.cgu.edu.tw

## 摘要

近年來在無線感測網路的研究越來越盛行，無線感測網路設置的主要目的是透過感測器來收集資料，並且使用無線的傳輸方式將其所收集到的資料傳回。ZigBee 是目前無線感測網路通訊協定標準中，極為令人矚目的一種，可應用在家庭、工業及醫療等相關環境中，包括監視、自動化及控制等。其定義的範圍為 Network 層以上以至於 AP 層，而底層則是以 IEEE802.15.4 通訊協定為基礎。

目前市面上開發相關產品技術的公司越來越多，但是有開放 IEEE 802.15.4 原始碼的卻少之又少，而且都是採用額外的一顆 micro processor 去執行 IEEE 802.15.4 的軟體堆疊。

在本論文中，我們在 Linux 環境下設計並實做了 IEEE 802.15.4 通訊堆疊，並整合到 Linux 核心當中。在這過程中我們克服了許多有關無線通訊協定及作業系統中時序與資源競爭等問題，成功的使用 Host CPU 來同時執行 IEEE 802.15.4 通訊堆疊與作業系統，提供給研究開發者另一種可行的方案。

## 1. 序論

無線網路在最近這幾年來的蓬勃發展，使得一些在無線網路上應用的新興技術如雨後春筍般的一一嶄露頭角。其中像是 ZigBee 此可應用於無線感測網路、無線個人區域網路(Wireless Personal Area Network，簡稱 WPAN)的技術，其主要是在短距離以及低功率的環境上使用，底層即是以 IEEE 802.15.4 通訊協定為基礎的應用。

放眼觀望目前市面上有關 IEEE 802.15.4 的產品，大多需要收費，很難找到開放的原始碼可供人開發使用。而且大多數的產品都是使用 Micro processor 去執行 IEEE 802.15.4 的通訊協定堆疊。

在本論文中我們使用工研院電通所所開發的 SCAN-ZB 作為研究平台，在 Linux 核心中加入 IEEE 802.15.4 通訊協定堆疊的設計與實作，使用 Host CPU 同時執行 IEEE 802.15.4 的通訊協定堆疊，以及 Linux 作業系統，這樣不但可以節省一顆 micro processor 的成本，並且將來想在 Linux 上開發新的應用程式也比較容易。依據初步的評估顯示，系統已經可以正常的運作。

本篇論文共為五節。第二節為相關研究，針對市面上有實做 IEEE 802.15.4 通訊堆疊的公司來做簡單的介紹。第三節為 IEEE 802.15.4 通訊協定，主要針對 IEEE 802.15.4 通訊協定做簡介。第四節則是系統設計與實作，主要介紹我們實做程式的部分以及比較值得探討之部分詳細說明。第五節為系統評量及討論。

## 2. 相關研究

以下為目前有關於實做 IEEE 802.15.4 通訊堆疊的介紹。

Microchip Technology Inc.[10]是 1989 年在美國成立的一家單晶片與模擬半導體的一家供應商。其主要產品有硬體平台以及相容於 Zigbee 的軟體堆疊，是以 micro processor 來執行 Zigbee 軟體堆疊並且是在一個沒有 OS 的環境下所執行。

不過這部份的資源必須要有買其產品才有辦法獲得，所以沒有詳細的資料。

Freescale[7]為目前市面上和 Microchip 互相競爭的廠商之一，主要在開發相容於 Zigbee 標準的軟硬體，讓應用程式開發工程師能更容易的開發各種不同應用在 Zigbee 上的應用產品，而其也是使用一顆 micro processor 來執行 Zigbee 軟體堆疊，也是在一個沒有 OS 的環境下執行。他們所開發的 Zigbee 軟體堆疊在 NWK 層以上有部分開放原始碼，但是 IEEE 802.15.4 所規定的 MAC 層以及 PHY 層則是只有 object code 可用，並沒有 source code 的提供。

Figure 8[4]是一個強大的軟體堆疊供應商，Figure 8 已經在 2005 年一月被 Chipcon 公司所併購。而此公司所開發的 Z-Stack 是目前 Zigbee 協定軟體中，最負盛名的一個，提供經濟、耐用且容易部署的解決方案。

## 3. IEEE 802.15.4 通訊協定

本節則是針對 IEEE 802.15.4 通訊協定來做介紹，第一節會先簡單的介紹一下 ZigBee 以及 IEEE 802.15.4 之間的關係，第二節會介紹一下其特性以及優點，第三節則介紹組成 IEEE 802.15.4 的元件及組成的拓樸形狀，第四節則簡介 IEEE 802.15.4 的架構，最後一節則是簡單介紹 IEEE 802.15.4 比較特有的功能。

### 3.1 Zigbee

Zigbee 規範 Network 層以上的架構。其制定的架構特點如下：

(1)數據傳輸速率低，速率在 10Kbps~250Kbps。

(2) 功耗低，低耗電。

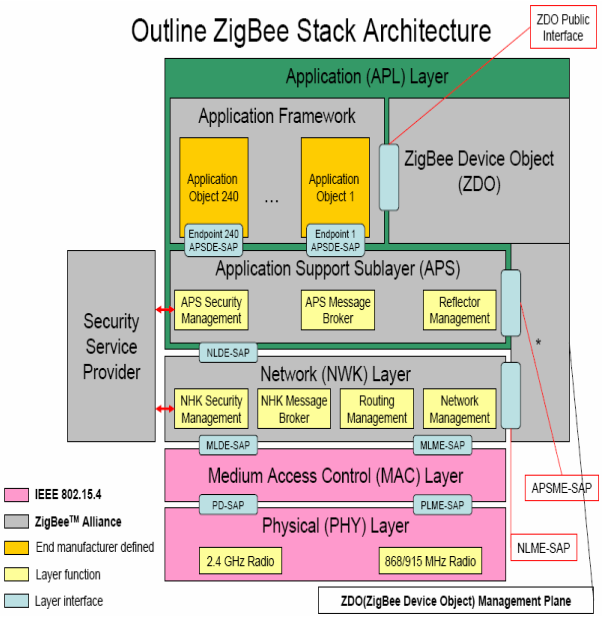
(3) 成本低，因為其架構簡單所以成本降低。

(4) 網路容量大，每個設備皆可與 255 個設備連接。

(5) 有效範圍小，其有效涵蓋範圍約為 10~75m 左右。

(6) 使用頻帶廣，分別為 2.4GHz、868MHz、915MHz。

而 Zigbee 所規定的架構主要有 Network(NWK)層、Application 層(APL)兩層，而在 APL 中又定義有三個部分。第一部分，Zigbee Device Object(ZDO)，主要是掌管其下的 device 為何種型態的裝置，如 Coordinator、EndDevicec 或 Router 等型態。第二部分，Application Support Layer(APS)，主要是傳輸 Data 的管理以及 security 的運作另外還有透過 ZDO 來做 Binding 與 Discovery 的動作。第三部分，則為管理 AP 層物件的 Application Framework。其架構如下圖所示：




。

圖 1、Zigbee 架構圖 [11]

從上圖可看到 ZigBee 目前囊括了從 NWK 層到 APL 的協定架構，而底層 Physical(PHY)層和 Medium Access Control(MAC)層兩層則是引用了 IEEE 802.15.4 通訊協定。

## 3.2 IEEE 802.15.4 之特性與組成

IEEE 802.15.4 主要是應用在一個電源必須持久但 throughput 不高的設備上，可以使其在一個很簡單且花費甚低的環境下使用無線網路與其他設備溝通。這個特性亦使得 IEEE 802.15.4 這個通訊協定稱作為 Low Rate Wireless Personal Area Networks(LW-WPANs)。而 LR-WPANs 最主要的幾個優點為：易於安裝、可靠的資料傳送、短距離的資料傳輸、非常低的花費及合理的電力持久性。

而 LR-WPANs 的特性如下：

(1) 其傳輸速率有 20kb/s、40kb/s 最高可達到 250kb/s。

(2) 其拓樸形式提供星狀與點對點兩種。其擁有 16 bits 或 64 bits 兩種位址模式，可供省電使用

(3) 其擁有 guaranteed time slots(GTSs) 的能力，GTSs 主要是保證傳輸的頻寬，可以使得此設備擁有 real time communication 的能力。

(4) 在頻道的存取方面，其使用 Carrier sense multiple access with collision avoidance (CSMA/CA)方式避免碰撞。

(5) 提供 acknowledged 的機制，使得資料傳輸更可靠。

(6) 提供 Energy detection(ED)的功能，選擇一個環境較好的頻道傳輸資料。提供 Link quality indication (LQI)，使其可以選擇品質較好的 coordinator 來連結。

(7) 在 2450 MHz 頻帶提供 16 個頻道、915 MHz 頻帶提供 10 個頻道、868 MHz 頻帶提供 1 個頻道。而目前台灣開放的為 2450 MHz 這個頻帶。

主要組成 IEEE 802.15.4 網路的基本單位為一個 device，而 device 的型態又可分為 full-functional device (FFD) 與 reduced-function device (RFD)其差別在於 FFD 必須擁有 IEEE 802.15.4 MAC 層的所有功能，但是 RFD 卻可選擇性的擁有。當兩個或兩個以上之 device 在同一個頻道上而且互相在各自的 personal operating space(POS)範圍裡面，並且有一個以上的 FFD 作為 Network 中的 Coordinator 即可形成一個 WPAN。

根據應用層面 LR-WPAN 之網路拓樸可以形成兩種形式：一為星狀(Start)另一為點對點(Peer-to-Peer)連接。如下圖所示：

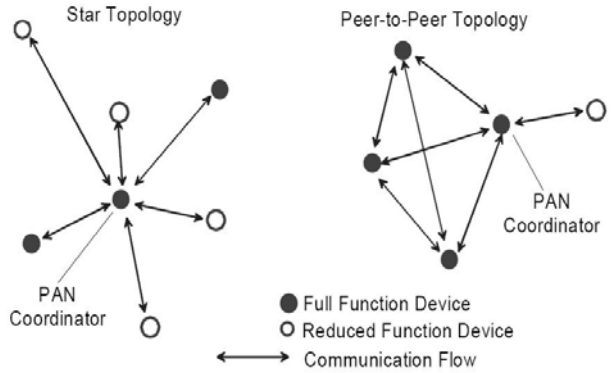



圖 2、網路拓樸

在星狀的網路連接型態中，必須有一個 FFD 來做為此網路的 PAN Coordinator 其負責管理並維護此網路之架構，而其他的終端點若要溝通皆須透過此 PAN Coordinator 來相互連結。而在這樣的網路中各個終端點可以為 RFD 也可為 FFD。在這個網路中的各個終端點裝置可以使用 64 bits extend address 直接與 Coordinator 來溝通，這個 64 bits extend address 通常為硬體廠商所給的 address 隨著硬體出廠就設定在其中；或者也可以讓 Coordinator 配置一個 16 bits short address 給已經與其連結上的終端點使用，在 Zigbee 中這個 short address 含有特別的意義，其為 Coordinator 用演算法算出來，可以根據這個 address 來取得這個 device 是在幾個 hop 之外，不過這已經超過 IEEE 802.15.4 的範圍。

而另一種點對點(Peer-ot-Peer)型態的網路拓樸也同樣必須要擁有一個 PAN Coordinator 來維持其網路架構，而這個 Coordinator 必須要管理這個 PAN 下的成員。也就是能在網路形態或行為改變時，Coordinator 能夠通知每個加入此網路的 Device。但與星狀不同的是在這個網路中的任兩個

device 只要相互在各自的 Personal Operating Space(POS)中即可直接的互傳封包，而不一定需要透過 PAN Coordinator 來轉送。

## 3.3 IEEE 802.15.4 之架構

在 IEEE 802.15.4 的標準中，定義了 PHY 與 MAC 兩層間的介面與其所需提供的服務。PHY 層最主要是用來操作 radio frequency (RF) transceiver 以及實現一些低階的控制機制。MAC 層主要的功能則是解析從 PHY 層所送上來的資料來判斷做何相對應的動作，以及使用 PHY 層提供的控制機制去控制 RF transceiver，其架構如下圖 3 所示：

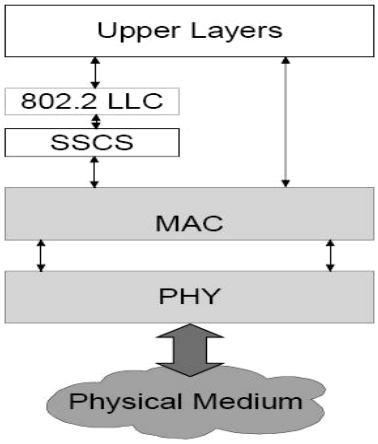



圖 3、IEEE 802.15.4 架構圖 [9]

### 3.3.1 PHY 層

在 IEEE 802.15.4 標準中，定義 PHY 層的特點如下：

(1) 啟動或關閉 RF 的動作。

(2) Energy Detection (ED)、Linking quality Indication (LQI)以便實現選擇品質良好之頻道。

(3) Clear Channel Assessment (CCA)，來監測此頻道是否正處於忙碌狀態。

(4) 提供之頻率如下：

868 - 868.6 MHz (e.g., Europe)

902 - 928 MHz (e.g., North America)

2400 - 2483.5 MHz (worldwide)

而 PHY 層主要提供兩種服務介面：其一為 PHY 層的專為資料傳輸所提供的服務介面，其工作是透過 RF 來傳送或接收封包，此封包稱為 Physical Protocol Data Units (PPDUs)，此介面稱為 Physical layer data service access point (PD-SAP)。其二為 PHY 層的提供的管理服務介面，而此界面稱為 Physical layer management entity service point (PLME-SAP)。另外還有一個專門儲存 PHY 層屬性的一個資料庫，稱為 PHY PAN information base (PIB)。

PHY 層之示意圖如下圖所示：

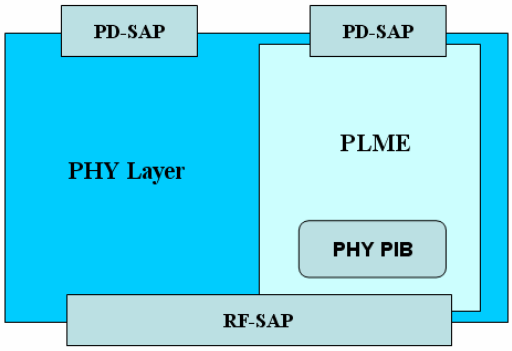



圖 4、IEEE 802.15.4 PHY layer 示意圖 [9]

### 3.3.2 MAC 層

在 IEEE 802.15.4 標準中 MAC 層擁有的特點有：

(1) Beacon 之管理。

(2) 頻道存取之機制(ie. CSMA/CA) 。

(3) 封包驗證。

(4) ACK 機制的支援與封包重送的機制。

(5) 連結與離開 PAN 的機制。

如同 PHY 層一般，MAC 層也提供兩種服務介面：一者為 MAC common part sublayer (MCPS) data SAP (簡稱 MCPS-SAP)，其主要是連接 PHY 層的 PD-SAP 將其所傳上來的 MPDU 再做進一步的處理以便再傳送給上層或接收到上層所要傳出的資料將其組合成 MPDU 再藉由 PHY 層的 PD-SAP 傳送出去。二者為 MAC sublayer management entity-service access point (MLME-SAP)，主要為提供上層管理整個 PAN 的一些管理機制。另外像 PHY 層一般 MAC

層也提供一個儲存 MAC 屬性的一個資料庫，稱為 MAC PIB。

MAC 層之示意圖，如下圖所示：

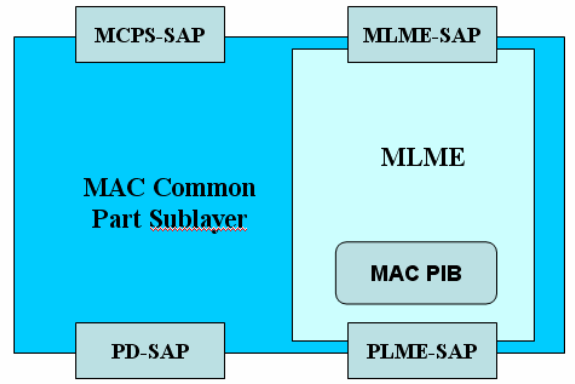



圖 5、IEEE 802.15.4 MAC layer 示意圖 [9]

## 3.4 IEEE 802.15.4 之功能性概述

本節主要是簡述 IEEE 802.15.4 中一些比較特別以及重要的功能。第一小節主要是介紹 Superframe 的架構，在 IEEE 802.15.4 中一個相當重要的概念，第二小節則是提到 PHY 層與 MAC 層的封包關係，第三小節則是介紹在 IEEE 802.15.4 中傳輸資料的一個重要的機制 CSMA/CA。

### 3.4.1 IEEE 802.15.4 Superframe 架構

在 LR-WPANs 中所定義的兩種網路型態，其一為 Beacon-disable network，其二為 Beacon-enable network，前者之 PAN Coordinator 不會定時發 beacon 而後者會。在後者的網路中，因為其會定時的發出 beacon，而 beacon 中包含了規定這個網路同步化的機制與定義這個 PAN 的行為與規則，這些限制就以一個 Superframe 來做規範。Superframe 的形成主要是根據兩個 Beacon 中間的間隔來規範，兩個 beacon 中間分成兩部分，第一部分為 Action 區塊而第二個部分則為 Inaction 區塊如下圖所示：

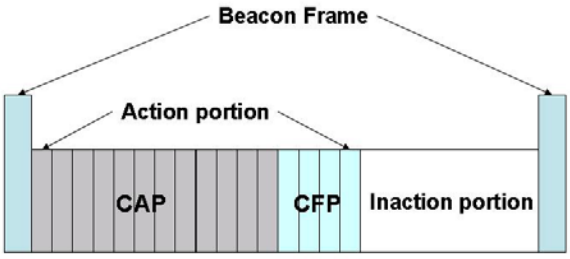



圖 6、Superframe Structure

在 Inaction 區塊中 Coordinator 為了省電選擇休息，而在這段期間 Coordinator 不接受任何封包的傳送與接收。在一個 Superframe 中所有的傳輸動作都必須在 Action 區塊中完成，如果此 PAN 允許 GTS 的要求則 Action 區塊中又可分為 Contention Access Period (CAP)區塊以及 Contention Free Period (CFP)區塊。如果 PAN 不允許 GTS 的要求則不會出現 CFP 區塊。在 CAP 區塊中所有封包的傳送皆須透過 CSMA/CA 的競爭方式來傳送，但是在 CFP 中 device 則必須依照 Coordinator 所分配的專用 slot 來傳送或者是接收資料，如此一來則可以保證在這個時間內此 device 可以確實的將其資料傳送而不需要與別人競爭。

### 3.4.2 IEEE 802.15.4 封包架構

在 IEEE 802.15.4 的標準中定義了四種不同封包的形式，分別為 Beacon、ACK、Data 與 Command 四種。而 MAC 層之封包與 PHY 層封包之關係如下圖所示：

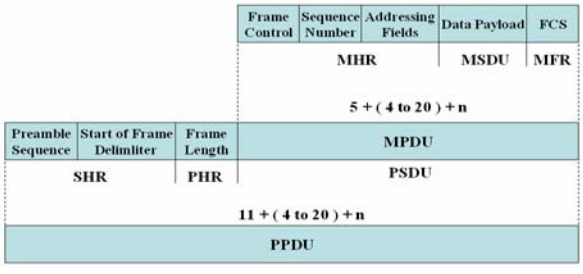



圖 7、IEEE 802.15.4 封包格式 [9]

### 3.4.3 CSMA/CA 機制

在 LR-WPAN 中有兩種對 channel 存取的機制，而決定使用哪一種存取機制是取決於其網路的架構。在 beacon-disable 的網路型態下使用的是 unslotted CSMA/CA 因為其沒有 superframe 的架構所以沒有讓 device 限定時間去送資料的功能。就因如此其 CSMA/CA 的機制就比較簡單，首先隨機取一個 backoff 的時間等過了這段時間後使用之前有介紹過的 PLME-CCA 去檢查 channel 上是否有人在傳送資料，如果沒有就可以把資料送出。如果碰到有人在傳資料也就是檢查 channel 的狀態為 BUSY，則再一次隨機取一個 backoff 時間來等待。另外在 beacon-enable 的網路架構下，因為其有 superframe 去規定整個時間的分配，所以必須依照 backoff slot 去作隨機的單位，而每次要做檢查動作的時候一定要是在每個 slot 的開始才能做此動作。在 beacon-enable 的網路架構中 CSMA/CA 的動作只能在 CAP 的區段裡面使用，而傳送 ACK 封包時則不需使用 CSMA/CA 機制。

## 4. 系統設計

在第一節中我們有提到，目前市面上的 IEEE 802.15.4 protocol stack 大多數是要收費的，而且他們的產品大多數是使用一個 micro processor。

為了節省開發成本，我們使用工業研究院電通所所開發出的 SCAN-ZB 作為硬體平台，將 IEEE 802.15.4 通訊堆疊實做到 Linux 系統核心中。我們所使用的 Linux 核心版本為 2.4.21 將其 patch 成 ARM 的架構，而 SCAN-ZB 主要是使用 Hynix 公司所生產的 HMS30C7202 SoC[8]當作核心，其使用 32 bits ARM 720T 中央處理器，時脈可到達 70 MHz 並擁有 UART、GPIO 等 IO port，使用 Chipcon CC2420 晶片來接收及發送相容於 IEEE 802.15.4 之封包。其外觀如圖 8 所示：

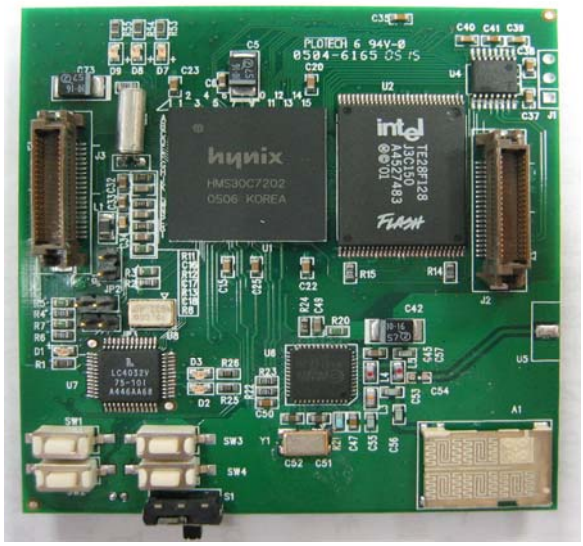



圖 8、SCAN-ZB 實照

本節主要是詳談我們所設計的系統，在 4.1 節先提出我們系統的架構，4.2 節則提出把 IEEE 802.15.4 實做在 Linux 核心中所遇到的問題，4.3 節則介紹封包的解析與組成，4.4 節則介紹我們所實做的高精確度 One-shut Timer，4.5 節則介紹重要的 CSMA/CA 機制，4.6 節則討論 Beacon 週期性實做，4.7 節則稍微探討 ISR 與 BH 的關係。

## 4.1 系統介面

實作介面的部分我們依照 IEEE 802.15.4 所規定的 primitive 函式並且參考 Freescale 的文件[5][6]，我們定義出來的介面架構如下圖所示：

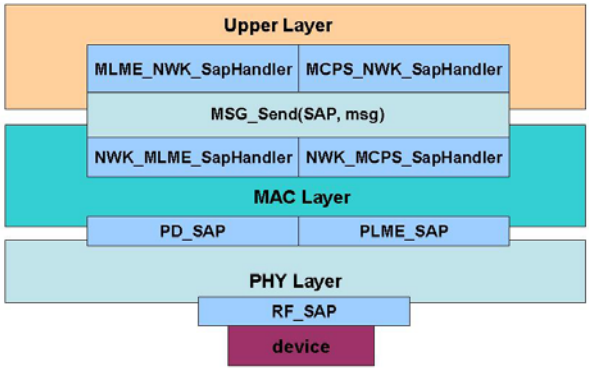



圖 9、實作各層介面圖

由圖 9 可看到，與下層 device 接觸的部分有 RF-SAP，這部分包含 driver 以及存取一些

transceiver 上的組態暫存器的 API。其上所接的就是 IEEE 802.15.4 所定義的 PHY 層，主要工作為拆解 PHY 層的標頭以及實現一些 Standard 上所定義的服務，封包部分會經過圖 9 上所標示的 PD-SAP 互傳，而提供服務的部分則是由 PLME-SAP 來負責。

再往上的 MAC 層則須拆解 MAC 的標頭，按照 Standard 上定義原本還必須去掉 2 個 byte 的 MFR，但是我們所使用的硬體也就是 Chipcon 這家公司所出的 CC2420 這塊晶片已經將 MFR 去除掉了，所以在實作上就省了這個步驟。

然後 MAC 層必須辨別這個封包是之前介紹的四種型態的哪一種，進而做不同的動作。如果是 Data 的封包則包成一個由 Freescale 定義的 Message 結構[5]，再經由 MSG_Send 此 Marco 送往上層的 NWK_ MCPS_Saphandler 然後交給上層做處理；如果是 Command 的封包則先判別需不需要往上轉送，如果需要則如同 Data 封包一般先拆解並包裝成一個 Message 再經由 NWK_ MLME_Saphandler 送給上層做處理，如果不需轉送上層則由 MAC 層處理並做相對應的動作。

## 4.2 設計策略

因為在相關文獻中找不到在作業系統核心中實做 IEEE 802.15.4 的先例，我們先針對可能遇到的問題進行分析並擬定初步的對策，以利後面系統的實作。

### ● 封包解析與組成：

IEEE 802.15.4 Standard 中所定義到的 primitive 函式必須要實做，而每個 primitive 的參數不一，所以我們參考 Freescale 所發表的 802.15.4 MAC/PHY Software User's Guide, Rev. 1.1 [6]來定義與 Network 的介面環境。定義完介面環境後，需要的分析封包以及組成封包的機制，這部分為在 4.3 節中詳細的介紹。

### ● 時間單位：

在 IEEE 802.15.4 中所使用的時間單位太小，不管是傳送封包、回應封包或者是計時都是使用這個單位，而這個單位稱作 symbol，其詳細解釋到後面章節會有說明，對於我們所使用的 Linux 來說是一個無法使用本身機制來處理的單位。因為 Linux 中皆是以一個 jiffies 作為時間基本單位並無法精確計算出 symbol 時間，所以我們準備實做一個高精準度的 One-Shut timer 來處理這些微小計時單位，這方面詳細的處理我們在 4.4 節中會詳細的介紹。

### ● CSMA/CA 機制：

是 IEEE 802.15.4 中另一個重要的機制，為了使封包發出時的碰撞機率減少，所以使用此機制來作為發出封包時該遵守的規則，但是這又牽扯到必須使用微小時間單位來計時，而這方面我們會在 4.5 節詳細的討論。

### ● Beacon 管理：

Beacon 在 IEEE 802.15.4 中所規定的 Beacon-enable WPAN 中扮演著很重要的角色，其發出的時間為週期性且必須要準確，所以我們利用高精準度的 One-Shut timer 來處理這個問題，關於此問題我們會在 4.6 節詳細討論。

### ● ISR 與 BH 之取捨：

在 Linux 中產生中斷後分兩部份去處理，第一個部份為中斷處理函式(ISR)，在這裡面處理的函式擁有即時性，但如果佔據太久則會導致中斷遺失。第二部份則為後續處理函式(BH)，這部份會延遲工作，但不會阻擋中斷發生。BH 部分我們無法預測他何時會被執行，但是在通訊協定標準中卻又有很多必須要即時的動作。在這部份，我們為了讓 ISR 盡量短，所以只讓 ISR 將資料從 chip 中搬移到 buffer 中，不過這又牽扯到效能問題。關於此問題我們會在 4.7 節詳細討論。

| Bits: 0–2 | 3 | 4 | 5 | 6 | 7–9 | 10–11 | 12–13 | 14–15 |
|---|---|---|---|---|---|---|---|---|
| Frame type | Security enabled | Frame pending | Ack. request | Intra-PAN | Reserved | Dest. addressing mode | Reserved | Source addressing mode |

圖 10、Frame control field 架構 [9]

## 4.3 封包解析與組成

IEEE 802.15.4 傳送資料皆是由封包形式來傳遞，所以首先我們必須要能解析封包才能進行下一步的判別。所以以下針對四種不同型態的封包去做介紹，並會提到實作中所用到的 structure。

### 4.3.1 Frame Control Field

下圖為一般性的 MAC 層封包：

| Octets: 2 | 1 | 0/2 | 0/2/8 | 0/2 | 0/2/8 | variable | 2 |
|---|---|---|---|---|---|---|---|
| Frame control | Sequence number | Destination PAN identifier | Destination address | Source PAN identifier | Source address | Frame payload | FCS |
| | | Addressing fields | | | | | |
| MHR | | | | | | MAC payload | MFR |

圖 11、一般性 MAC 封包架構 [9]

而圖 10 是 MAC 封包最前面兩個 byte 的 Frame control field(FCF)部分：

如圖 10 所示，能分辨此封包整個架構必須經由讀取 FCF 這個欄位來判別，首先讀取一個 byte 先判別此封包為何種型態的封包然後再判別，原本讀取 Ack request 欄位後應該要回 ACK 給對方，但是這邊我們使用 Chipcon CC2420 的晶片其硬體有自動回 ACK 功能，所以就不需再以軟體回 ACK。然後再讀取後面一個 byte 就可由 Intra-PAN、Destination addressing mode 、Source addressing mode 來判斷之後的 Addressing Field 的架構。

### 4.3.2 ACK Frame

判斷完 FCF 之後如果是 ACK 封包則檢查 MAC 層中的一個 Status Queue，這個 Queue 是用來存放送出以後正在等待更新的封包狀態，如果檢查到 Status Queue 中有與所接到之 ACK 相同的 Data sequence number 則更新此封包狀態，使其知道傳送動作已經完成。其流程圖如圖 12 所示：

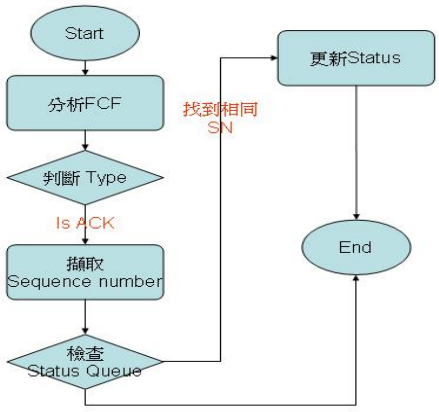



圖 12、ACK 封包處理流程

### 4.3.3 Data Frame

判斷完 FCF 之後如果是 Data 封包，則會進入處理 Data 封包的函式中。而 IEEE 802.15.4 的 Data 封包之格式如下圖所示：

| Octets: 2 | 1 | (see 7.2.2.2.1) | variable | 2 |
|---|---|---|---|---|
| Frame control | Sequence number | Addressing fields | Data payload | FCS |
| MHR | | | MAC payload | MFR |

圖 13、IEEE 802.15.4 Data 封包格式 [9]

其中 FCF 中的資訊和 Sequence number 已經被儲存起來，接下來就是將 Data payload 的部分告知上層。由於在 IEEE 802.15.4 中規定收到一個 Data 封包之後，必須要向上層發出一個 MCPS-Data.indication 的 primitive 來通知上層收到了一個 Data 封包並將資料送至上層。因此，此 Primitive 裡頭規定了必須要包含以下的欄位:

dstAddr：此欄位可以為 16 – bits short address，或者是 64 – bits extended address，取決於 FCF 中的 dst addr mode 欄位，這個欄位紀錄著此 Data 封包是要傳送給哪一個設備的。

dstPanId：此欄位之長度為 16-bits。其紀錄此 Data 封包是要傳送到哪一個 PAN 中，欄位的內容可以從 Data 封包 addressing fields 欄位取得。

dstAddrMode：此欄位紀錄著此封包採用的是哪一種 addressing mode，可以從 FCF 中的 dst addr mode 欄位取得。

srcAddr、srcPanId、srcAddrMode 這三個欄位和上面介紹的三個欄位雷同，在此不再贅述，不同的地方只有在這三個欄位紀錄的是此封包是從哪一個設備來的，以及遠端設備的相關資訊。

msduLength：此欄位紀錄著 MAC payload 的部分的長度，上層所看的到的封包長度就由此欄位來取得。

mpduLinkQuality：此欄位紀錄著收到此封包時，當時的 Link Quality 的值。

securityUse：此欄位紀錄著封包的內容是否需要使用加解密的方式來加強安全性。

msdu：此欄位中所紀錄的就是真正的 MAC payload 的內容，也就是要交付給上層處理的資料。

將 Data 封包拆解成以上的參數來填入 MCPS-Data.indication 之後就可以將其往上層傳送了。

### 4.3.4 Beacon Frame

Beacon 封包在 IEEE 802.15.4 中是一個重要的元素。這種型態的封包在 beacon-enabled 的 LR-WPAN 中，主要目的是讓一個 PAN 中所有 Device 能夠做同步的動作。如此就能達到讓 Device 的 Receiver 做重複開關的動作，但仍舊能夠正確的接收到應該要接收的封包，這個動作是讓 Device 在規定可以休息的時段中可以不需要打開 Receiver 以達到省電的動作。另外在 IEEE 802.15.4 規範中，Beacon 封包還可以由 PAN Coordinator 發出，Beacon 封包裡面會包含著維持此 PAN 運作的一些規範，例如 Device 應該要何時開啟 Receiver，Receiver 需要開啟多久的時間，開啟多久後可以關掉來達到省電的功能，是否有資料存放在 Coordinator 上等等。當一個 Device 想要加入一個 PAN 的時候，就必須要先遵循此 PAN 的 Coordinator 所規範的行為。而該如何在尚未加入 PAN 時能夠知道這些事情，答案就是在每個 Coordinator 發出的 Beacon 中獲得。Beacon 封包之內容如圖 14 所示：在我們接收到一個 Beacon 封包之後，首先我們會先判斷目前 Device 是否處於 SCAN 的狀態中。由於 IEEE 802.15.4 規範中提到，在非 SCAN 的狀態中，收到 Beacon 封包之後，必須要作過濾的動作，也就是只接收來自 Coordinator 的 Beacon 封包，若是在 SCAN 的狀態中，一般來說我們還是處於尚未選定加入哪一個 PAN 的情形之中，導致可能會過濾掉所有的 Beacon 封包( 因為尚未選定 Coordinator )因此在此情況下，我們會先把所有接收進來的 Beacon frame 存到一個陣列之中，過了 SCAN 的狀態之後，再把這些 Beacon frame 傳給上層，讓上層去作決定該做怎樣的動作。例如根據演算法選擇一個最適合此 Device 的 PAN 做 Associate 的動作。

| Octets: 2 | 1 | 4/10 | 2 | variable | variable | variable | 2 |
|---|---|---|---|---|---|---|---|
| Frame control | Sequence number | Addressing fields | Superframe specification | GTS fields (Figure 38) | Pending address fields (Figure 39) | Beacon payload | FCS |
| MHR | | | MAC payload | | | | MFR |

圖 14、IEEE 802.15.4 Beacon 封包格式 [9]

另外若是不在 SCAN 的狀態時，則我們就應該開啟過濾的機制，對於接收進來的 Beacon，檢查是否來自自己的 Coordinator。接下來則是判斷是否有 PAN-ID conflict 的情形發生。如果有的話，則應該利用 SYNC-LOSS-INDICATION 此 Primitive 告知上層做處理 。

接著會去 MAC PIB 中搜尋一個 attribute，稱為 MacAutoRequest。此 attribute 的意義是說，當 Device 收到一個 Beacon frame 時，如果從 Beacon 上面得知有資料在 Coordinator 上面，正常的情形下，必須請求 Coordinator 將資料送給自己。這個請求的動作會根據這個 Attribute 來決定 MAC 層是否應該自動完成這個請求的步驟。如果是 TRUE，則 MAC layer 就會自動發出一個稱為 Data Request command frame 送給 Coordinator，而如果此 Beacon 中存有 Beacon payload 時，要使用 MLME-BEACON.NOTIFY 告知上層有資料的到來。另外，如果當這個 Attribute 的值為 FALSE 的話，代表此動作不需要 MAC layer 費神，藉由 MLME-BEACON-NOTIFY 將是否發送 Data Request Command frame 的決定權交給上層。

而 Device 該如何得知是否有自己的資料放在 Coordinator 身上，而必須請求 Coordinator 將資料傳送給 Device 呢？這時就需要去解析 Beacon frame 中的 Pending Address fields。當 Device 收到一個 Beacon 之後，會去檢查 Beacon 中 Pending Address field 是否有自己的 Address 在上面，若是有的話則代表需要向 Coordinator 發出一個 Data Request Command frame。Pending Address field 之示意圖如下圖：

| Bits: 0–2 | 3 | 4–6 | 7 |
|---|---|---|---|
| Number of short addresses pending | Reserved | Number of extended addresses pending | Reserved |

圖 15、IEEE 802.15.4 Pending Address Field 架構 [9]

由圖 15 可知，當 Device 接收到 Beacon frame 之後，必須要解析最多總共七個長短的 Address，以得知自己是否有 Data 放在 Coordinator 上面。

而在我們實做的程式中，解析 beacon 封包的流程圖如圖 16 所示：

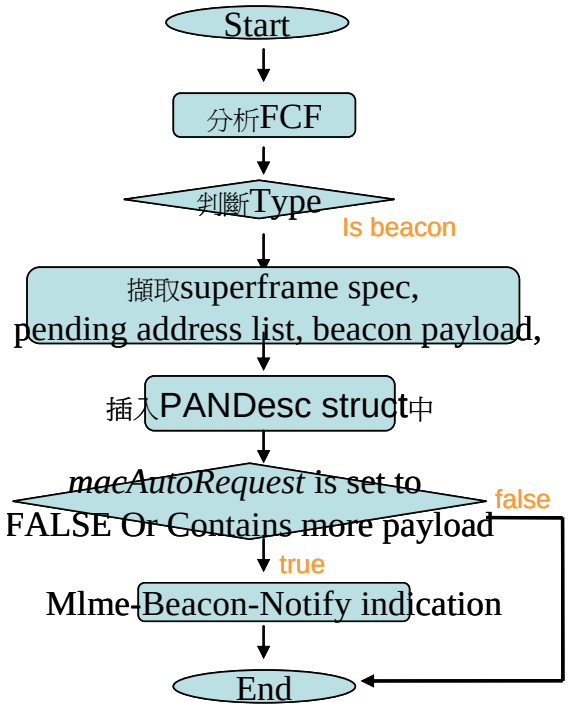



圖 16、Beacon 封包處理流程

## 4.3.5 Command Frame

接下來我們要描述的則是 Command Frame。此種 Frame 包含一些命令的封包，例如命令某一個 Device 離開、通知 Coordinator 想要連結上網路等等。此封包的格式如下圖所示：

| Octets: 2 | 1 | (see 7.2.2.4.1) | 1 | variable | 2 |
|---|---|---|---|---|---|
| Frame control | Sequence number | Addressing fields | Command frame identifier | Command payload | FCS |
| MHR | | | MAC payload | | MFR |

圖 17、IEEE 802.15.4 Command 封包架構 [9]

而其中 Command frame identifier 則敘述著此封包究竟是哪一種 Command Frame，在 IEEE 802.15.4 之中規範中有以下幾種 Command Frame：

| Command frame identifier | Command name | RFD | | Subclause |
|---|---|---|---|---|
| | | Tx | Rx | |
| 0 x 01 | Association request | X | | 7.3.1.1 |
| 0 x 02 | Association response | | X | 7.3.1.2 |
| 0 x 03 | Disassociation notification | X | X | 7.3.1.3 |
| 0 x 04 | Data request | X | | 7.3.2.1 |
| 0 x 05 | PAN ID conflict notification | X | | 7.3.2.2 |
| 0 x 06 | Orphan notification | X | | 7.3.2.3 |
| 0 x 07 | Beacon request | | | 7.3.2.4 |
| 0 x 08 | Coordinator realignment | | X | 7.3.2.5 |
| 0 x 09 | GTS request | | | 7.3.3.1 |
| 0 x 0a–0 x ff | Reserved | | | — |

圖 18、IEEE 802.15.4 Command 封包定義 [9]

在此我們不在贅述解析這些封包的過程，因為他們的處理流程和處理 Data Frame 的方式大同小異。

## 4.4 高精確度 One-Shut Timer

在 IEEE 802.15.4 標準中所用的基本單位為 symbols，從 Standard 中可得知處於 2.4G 頻帶時其 symbol rate 為每秒 62.5ksymbol，由此換算得知一個 symbol 約為 16μs 遠比 linux 中基本單位 jiffies 為 10ms 還要小上很多，所以光靠 linux 中的計時單位是不夠的。因此我們就從實做的硬體平台 SCAN-ZB 上找了一組 hardware timer 來做一個非週期性的 timer 中斷機制，其 request 單位為 symbol 這樣就可來滿足後面有關於時間方面的問題。

SCAN-ZB 硬體上總共有三組 hardware，其中一組被 Linux 的系統 timer 用去，我們就使用剩下的兩組使用其中一組來當作計時器。而我們的 timer interrupt 機制是使用一個 list 去紀錄進來的 request，而在接收到一個 request 時會先去判斷此 list 中是否有其他的 request 正在等待被喚醒。這個 list 是用一個 static 的 array 然後使用指標做成 ring buffer 的形式。原因是如果使用 linking list 則勢必要在 ISR 中去動態配置記憶體。而在 ISR 中呼叫過多 kernel 的函式會使得 ISR 過長，則會造成原本應該有 timer 的 interrupt 產生卻因為處在 ISR 中將 interrupt disable 而沒有發生。這情況對非常依賴此 timer 機制的 IEEE 802.15.4 來說是一個很嚴重的問題。

其處理流程圖如下圖所示：

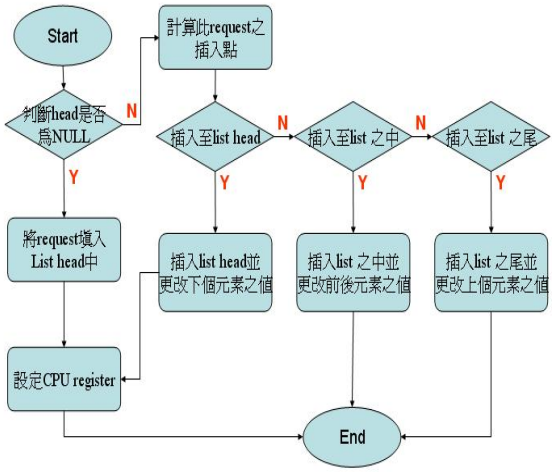



圖 19、Timer list structure

在我們實做這個高精準度 One-Shut timer 時並沒有考慮到如果它與系統的 timer 撞到的時候應該要如何處理。因此在我們剛完成 One-Shut timer 在測試的時候發生系統 timer 撞到我們的 One-Shut timer 之後就沒有再發生中斷了。因為我們的 timer 是單次觸發，如果這次沒有中斷產

生，其也不會去設下一次的中斷發生。發現了這個問題之後，我們著手去將這個問題解決，一開始我們使用的方法是讓我們的 timer 中斷處理函式在執行時，關掉其他的中斷。但這卻造成系統的 jiffies 會有延遲的情況發生，原因是原本系統的 timer 剛好來在我們的中斷處理函式中，但卻無法產生中斷去做它的處理函式。最後我們的解決方法為當系統的 timer 撞到我們的 timer 時，優先讓系統的 timer 執行，在它執行完之後再將我們的 timer 啟動。這樣一來便解決了之前 jiffies 延遲的問題。

## 4.5 CSMA/CA

在前面有提過 IEEE 802.15.4 的 CSMA/CA 機制有兩種，一種是在 beacon-enable 的 PAN 下所使用的 slotted CSMA/CA 另外一種則是在 beacon-disable 的 PAN 下所使用的 unslotted CSMA/CA。而 unslotted CSMA/CA 的演算法比 slotted CSMA/CA 來的簡單一些，他只需要隨機取一個 backoff time 單位為 symbol 並且等待這段時間後去使用 PHY 層的 PLME-CCA.request 來確定頻道上是否有人在傳輸資料。若沒有則直接將資料送出，如果有人正在傳送資料則再一次去隨機取一個等待的時間，其隨機的方法跟 slotted CSMA/CA 一樣，所以等一下詳細介紹 slotted CSMA/CA 的時候再做說明。

在 slotted CSMA/CA 的演算法中，它將原本 Superframe 所規定的 slot 劃分成更小的單位稱做一個 backoff period，而對應到 MAC PIB 中此值為 aUnitBackoffPeriod 其單位為 symbol。因為原本劃分的 slot 很大所以如果以其做 backoff 時間的單位則會浪費太多時間導致整個輸出量變小，碰撞機率也會變大。在 slotted CSMA/CA 中必須要維護三個變數以利演算法的進行，其說明如下：

NB：全名為 Number of Backoff，也就是總共執行了多少次隨機取 backoff 的次數，每當執行一次此值就會加 1。如果遇到使用 CCA 去偵測頻道為忙碌時，就必須要重新去隨機取一個 backoff，而此時 NB 就會增加。NB 最大值為 4 如果在這四次中偵測頻道都為忙碌的話就放棄傳送此資料，其是為了避免系統有過大的負擔。

CW：全名為 Content Window Length，其單位為 backoff period 在 IEEE 802.15.4 中的預設值為 20 個 symbols。意義為必須執行幾次偵測頻道皆為閒置時才可將資料送出，其初始值為 2 會隨著偵測到頻道閒置而遞減。

BE：全名為 Backoff Exponent，是用來計算 Backoff 時間的一個參數。其公式為：Backoff Time = $2^{BE} - 1$。

其演算法如下圖 20 所示：

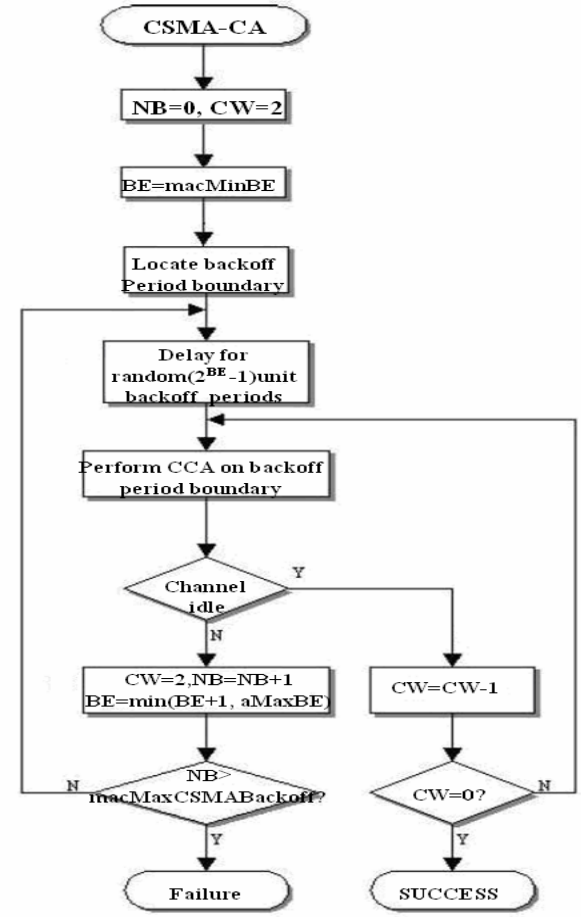



圖 20、IEEE 802.15.4 slotted CSMA/CA 演算法 [9]

而在我們實做的時候，我們將其拆解成兩個部分，第一個部分為做 CSMA/CA 前置動作的函式名叫 CSMACA( )，主要動作為設定一些變數的初值如上面所提到的 NB、BE、CW 等參數並呼叫我們所寫的

sys_setimmtimer( )這個函式來設定 interrupt 發生時間，這個時間也就是隨機產生的等待時間。而第二部分為實際去執行 CCA 的這個部分我們寫了一個名叫 DoCCA( )的函式，我們將其掛載成 timer interrupt 的 callback function，當 timer interrupt 發生且發覺其 callback function 為此函式時則會去執行 CCA 若連續二次都發覺到頻道為閒置時就將其送出。若發覺其中一次 CCA 偵測到頻道忙碌則依照上述之演算法去更改各個變數的值，然後在呼叫一次 sys_setimmtimer( )來重新執行這個流程，如果發現 NB 已經超過規定的數值，則須回報給上層一個訊息告知。

而呼叫 CSMA/CA 的時機有二，其一是從上層需要發送出資料或命令封包時，這時做完 CSMA/CA 之後會有相對應的流程需要執行但是以我們所時做的 CSMACA( )函式並無法回到呼叫點去執行其所要做的動作，所以當上層下資料或命令下來需要用到 CSMA/CA 的機制時(通常為 direct 傳輸)，我們另外用一個 MacCSMACA( )的函式將他包起來以便重新回到呼叫 CSMA/CA 的那個點去做接下來的動作。 其二是接到封包之後必須使用 CSMA/CA 機制將封包送出，例如說 Coordinator 接到一個 Data Request command 時必須從 Transaction Queue 中找到對應封包並將此封包用 CSMA/CA 的機制送出(通常為 indirect 傳輸)，此流程皆在 ISR 與 BH 的流程中，並不能使用等待的機制去等待對方回 ACK，所以我們另外寫了一個函式叫 macResend( )用來檢查是否接到 ACK 與重送機制。

另外如果有一個以上的資料需要做 CSMA/CA 動作時，後面進來的資料必須等待前一項資料的 CSMA/CA 動作做完之後才能執行。所以我們必須先將此資料封包放在一個 list 中，並在 DoCCA( )這個 ISR 後掛上一個 BH 專門來處理這個問題。當 DoCCA( )做完進入 BH 之後會先去檢查此 list 是否還有資料需要傳送，如果有就再呼叫 CSMACA 這個函式讓他繼續將後面的封包送出。

而 DoCCA 的程式流程如下圖：

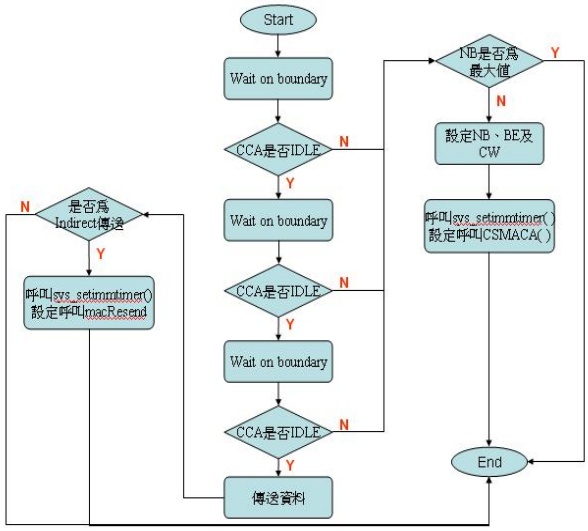



圖 21、DoCCA 程式流程

## 4.6 週期性 Beacon

Beacon 在 beacon-enable 的 PAN 中是一個很重要的封包，而其發出的準確性也是重要的一個因素。所以我們也是使用前面所提到的函式 sys_setimmtimer 來設定發出 beacon 的時間，當 interrupt 發生時則會依照我們所掛載的函式來發出 beacon。另外 IEEE 802.15.4 中有規定資料存在 Transaction Queue 中超過一段時間便需要被丟棄，而我們將這個機制掛載在此 ISR 結束之後的 BH 中去實做。因為定義的這段時間與發出 beacon 的時間有相關聯性。所以我們就如此實做。

而發出 beacon 封包除了組成封包外還需檢查 MAC 層中的 Queue 是否存在著別人的資料，如果有則必須加入到 beacon 封包中 pending address 的位置。

## 4.7 ISR 與 BH 之間的取捨

一開始我們以 IEEE 802.15.4 所定義的 PHY 層與 MAC 層的完整功能性為前提來實做我們的系統，又考慮到必須要有即時性所以將其都放到 ISR 中去實做。但這樣遇到一個問題，當 ISR 佔據太久導致下個封包來的時候無法產生中斷，就會使這個封包流失掉。這個問題在我們做 Assocaite 過程的

時候尤其明顯。有此經驗後，我們將 MAC 層與 PHY 層移出 ISR，使得 ISR 中只將資料從 chip 搬移到 buffer 中，這樣就不會使 ISR 佔據太久。

但是這又浮現出另外一個問題，原本在 ISR 中收到資料必須回應 ACK 可以立即的動作，但是將 MAC 層放到 BH 中則會延遲一段不短的時間，這段時間已經遠遠超過定義的標準值。這個實驗結果將會在第五節實驗 5.1.1 圖 23 中將會被展示出來。延續上列問題在 IEEE 802.15.4 中規定如果有使用 ACK 機制時，當一個封包送出去以後必須要接到一個 ACK，如果沒有接到則必須要重送這個封包，如此重送到標準中定義的最大重送次數。而我們實做的方法為每一個送出去的封包都會在 MAC 層中的一個 Status Queue 中去紀錄它的封包狀態，包括何時送出、執行 CSMA/CA 的狀態、有無接到 ACK 的狀態等。當 device 發出一個封包後便將其 status 存在這個 queue 中，等到接到其 ACK 之後便去更改 status。讓上層知道它已經接到 ACK 完成了傳送的動作。而接到 ACK 去更新 status 這個動作原本應該是在 MAC 層 BH 裡面要做掉，因為解析封包的動作是在 MAC 層該做的。但是根據實驗 5.1.1 圖 24 中可以來說明，如果暫時忽略完整性，將這樣的動作不放在 MAC 層 BH 裡面而併到 ISR 來做則會更好。根據上列討論，我們最後完成當接收到封包 IEEE 802.15.4 的處理架構對應到 Linux 上如下圖 22 所示：

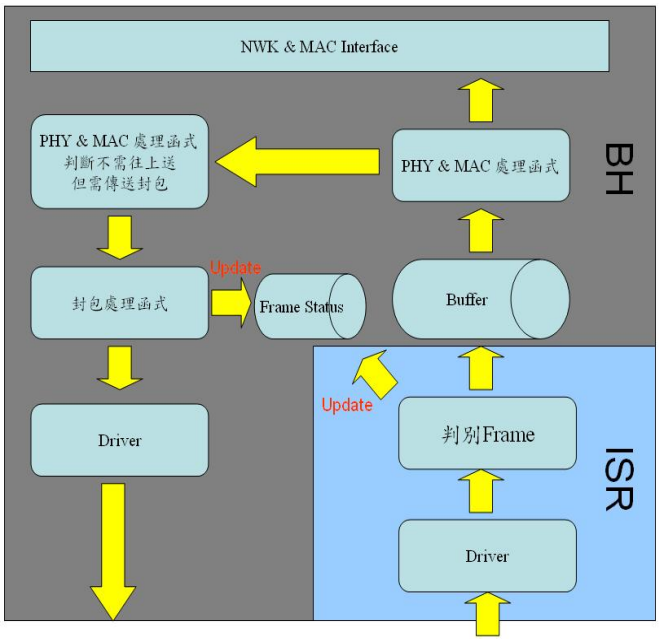



圖 22、封包處理對應在 Linux 上之架構圖

當 RF 收到一個封包以後，產生一個 interrupt 此時控制權轉移到 ISR，第一道處理則是由 Driver 將資料從 Chip 中拿出，並判斷是哪一種封包，如果是 ACK 封包則直接去更新系統中一個紀錄封包的 status queue，如果不是則將資料封包放到 buffer 中此時控制權交由 BH，當系統判斷一個安全時間後會呼叫 BH 來執行，此時我們掛在 BH 中的 PHY 與 MAC 處理函式就會起來做封包的解析，如果需要立即送出一個封包則再透過封包處理函式送出一個封包，否則送到與 NWK 層的介面時 BH 即結束。

# 5. 系統評量與討論

## 5.1 規格驗證及基本運作

在第一小節中，我們將驗證我們所設計的架構均能夠符合 Standard 所要求，並且以實驗數據來顯示。

### 5.1.1 ACK 的封包處理

在 4.7 節中，我們對於 ISR 和 BH 的取捨做了解釋。在這邊我們展示出兩個實驗來證明我們修改的系統架構是的確可以增進系統的準確性以及效能。

第一個實驗主要是來比較當 SCAN-ZB 接到一個封包產生中斷後在 ISR 中將封包搬到 Buffer 中，交由 Bottom half 進到 MAC 層函式處理一直送到與 Network 層的介面所需花費的時間。

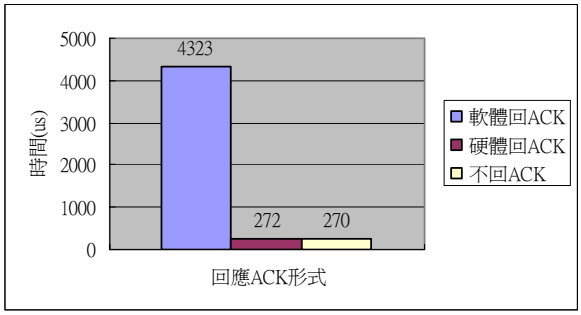



圖 23、處理 ACK 所需時間比較圖

在 IEEE 802.15.4 的 Standard 中規定需要回 ACK 的為 MAC 層，所以一開始我們的設計解析這個資料封包需不需要回 ACK 的動作是做在 MAC 層，而 MAC 層的處理函式是位在 Bottom half 中，所以從圖 23 中可看到原本 ACK 由 MAC 層回覆的時候需要 4323 個 μs。比起不做任何動作直接把資料送到與 Network 層介面的 270μs 還要多上許多，所以我們就找了 CC2420 的 Datasheet 中發現這塊 chip 擁有硬體回 ACK 的功能，之前沒有使用這個功能的原因是由硬體所發出的 ACK 都只有一種型態無法滿足 IEEE 802.15.4 的所有需求。但是如果由 MAC 層來回 ACK，則從接到資料封包至發出 ACK 的時間長達 5186μs 遠超過 MAC PIB 中所定義的 ACK 回覆時間。所以為了符合 Standard 中的規定我們使用硬體回 ACK，以此方法則從接到資料封包以至發出 ACK 時間約為 900μs，很接近 MAC PIB 中所規定的 54 個 symbol。

第二個實驗主要是比較更新 Status Queue 這個原本該在 MAC 層 BH 中處理的動作若是拆開移到 ISR 來做的差別。

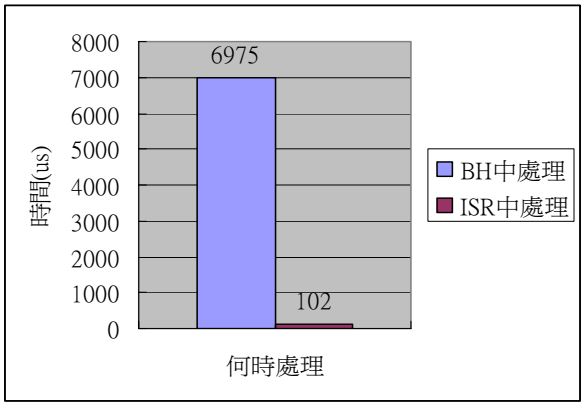



圖 24、處理 ACK 所需時間比較圖

當 device 接到一個封包後，將封包丟給 MAC 層的函式解析其為 ACK 然後再去更新 Status Queue 中所屬的狀態，而這個動作是在 Bottom half 中去動作。由上圖可以清楚的看到，如果這個動作交由身處在 BH 中的 MAC 層來做則需要 6975μs 才可以更新完成遠遠超過 IEEE 802.15.4 中所定義的 54 symbols 而此時測試的環境是當 Status Queue 中有三個不同封包的 status，同樣的情況如果這個 ACK 解析移往 ISR 中去判斷則可由上圖 23 得知只需要 102μs 符合 IEEE 802.15.4 中的規範。基於以上理由我們將判斷 ACK 封包與更新 status 的動作移在 ISR 的前頭去做以便符合 Standard 中的規範。

### 5.1.2 Beacon 的準確性

我們在這個實驗中，測試我們自製的即時性 timer 是否都有照著規定的時間精準的發出 Beacon。我們設定一個變數為 Beacon Order 的值為 6。這個值會決定兩個 Beacon 中間差距的時間為多少，根據以下的公式：

$$BI=aBaseSuperframeDuration*2^{BO}$$

symbols，我們可以算出兩個 Beacon 之間的時間差距為 983.04ms。在我們的實驗中，我們用 Chipcon 的 CC2400 Evaluation Board 來當 sniffer[2]，用來監測封包傳送狀態。我們可以利用此軟體來記錄兩個封包發出時所間隔的時間，下圖即為結果。

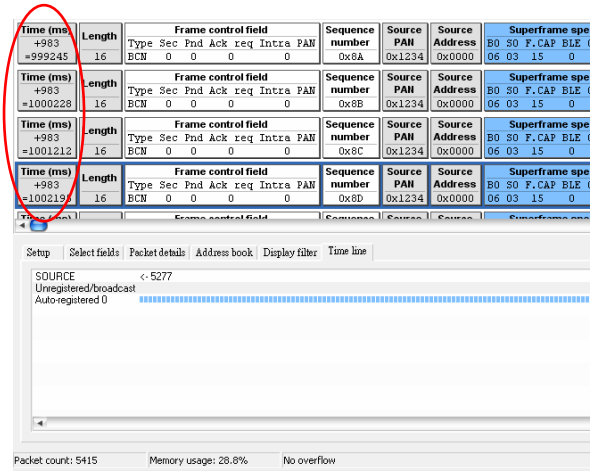



圖 25、Beacon 時間圖


圖 26、Beacon 時間圖(2)

在圖 25 中最左邊那一欄可以看到每一個封包和前一個封包所距離的時間均為 983ms。而在圖 26 中(也就是圖 25 中的左下角)可以看到總共有 5415 個封包被記錄，而所有的封包間隔時間均為 983mss 和 Standard 所規定的公式中算出來的結果均符合。

### 5.1.3 基本運作

這個實驗的目的主要是希望能夠測試兩個 Nodes 之間彼此可以做正確的資料傳輸。我們在兩台 Nodes 中間彼此傳送 Data，當一個 Nodes 接到對方的 Data 並回傳一個 ACK 之後，然後己方送出一個 Data 給對方。等待對方回傳一個 ACK 之後，對方會再送一個新的 Data 過來，而我就把這一份 Data 拷貝過來，然後在將此資料傳送過去，接著重複再做下一輪的動作。根據 Sniffer 上的觀察，此測試的結果是正常的。

## 5.2 Interoperability

為了測試我們開發的程式符合 Standard 所規定，我們利用 Freescale 所開發的平台和我們的平台互動來達到測試的目的。

我們的實驗環境為使用 Chipcon 的 CC2400 Evaluation Board 來當 sniffer[2]，用來監測網路封包傳送狀態。使用 Freescale 的 Dig536-2 平台，載入 Freescale 所開發的程式，當做 Coordinator，負責接收命令封包、判斷是否接受 Device 連結的判斷、回 ACK 以及發送 Beacon。使用 SCAN-ZB 當作一個 End device，它會作包括搜尋、連結 Coordinator、解析 Beacon、發出命令封包的動作。

圖 27 為 Sniffer 顯示出其中的連結過程，流程如下：

(A)End Device 發出一個 Associate Request 命令封包給 Freescale 的 Coordinator。

(B)Freescale 的 Coordinator 收到此封包之後回應了一個 ACK。

(C)再來 SCAN-ZB 所代表的 End Device 發出一個 Data Request 命令包封要求 Coordinator 回應。

(D)而 Coordinator 首先回應了一個 ACK，然後回傳一個 Associate Response Command 給 End Device。

## 5.3 效能量測

接下來我們所要做的是將我們的效能量測出來，並與 Freescale 所開發的套件比較，主要是針對 Data Frame 傳送的 throughput 來比較。

我們的實驗環境為使用 Chipcon 的 CC2400 Evaluation Board 來當 sniffer[2]，用來監測網路封包傳送狀態。使用 SCAN-ZB 當 Coordinator 負責接收封包、回 ACK 以及發送 beacon 等工作。而 Coordinator 的環境設定為 Beacon Order 為 6 並且 Superframe Order 也為 6，在這個情況下 beacon 的間隔時間為 983,040 $\mu$s。等於在兩個 beacon 之間 End device 都可以使用 CSMA/CA 機制將資料封包送出。而我們所使用的實驗組為 SCAN-ZB 而使用的對照組為 Freescale，皆當作 End device 來發封包，總共發出 1000 個資料封包給 Coordinator。然後取每一次 beacon 與 beacon 中間所發的封包總長度(包含資料封包以及 ACK 封

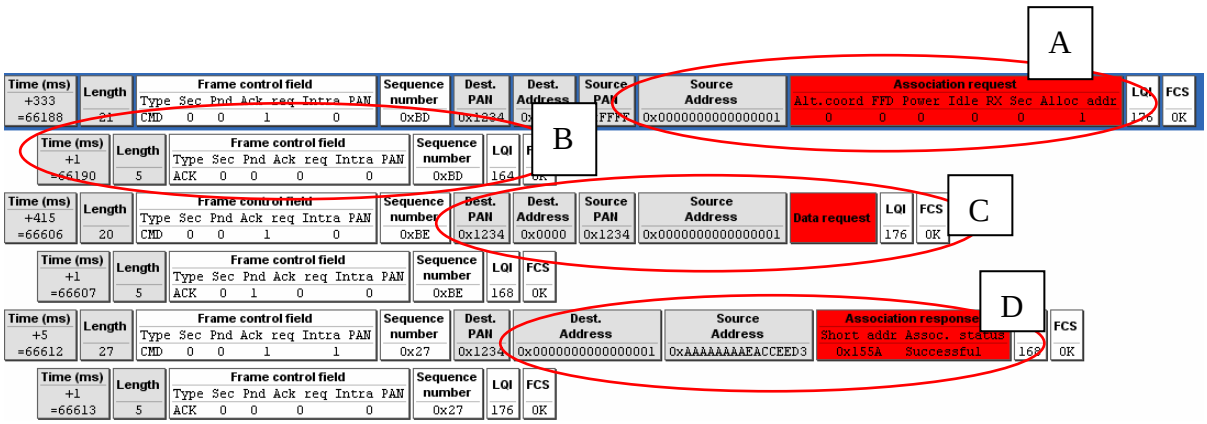



圖 27、Sniffer 紀錄 SCAN-ZB (End Device) 和 Freescale (Coordinator)連結的動作

包)，除以 beacon 間隔時間來換算出資料流量，再算出平均流量其單位為 kbps。其量測結果如下圖所示：

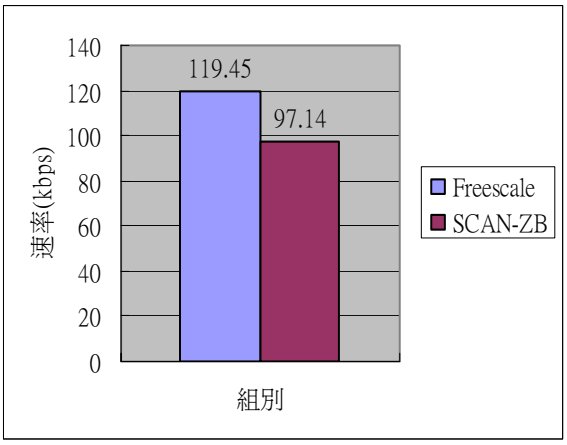



圖 28、資料流量比較

我們量測出來的流量值雖然比較低，但是 Freescale 是使用 micro processor 去執行 IEEE 802.15.4 protocol stack 且其沒有 OS 的負擔，所以我們效能比其低落，這方面我們會再去做效能的改善。

## 6. 結論與未來工作

由於近幾年越來越多人投入研究無線感測網路的領域，但是目前市面上廠商大部分有開放 NWK 層以上的 source code 卻很少有開放 MAC 層以下的 source code，使研究此領域的人需要花費一些金錢來得到這個技術，卻還不能了解其中的運作原理。所以我們使用 open source 的 Linux 為開發平台來開發這套 IEEE 802.15.4 protocol stack，並且不用 micro controler 改用 host CPU 來執行，這樣就可以減少需要一個 micro controler 的花費也大大的降低了開發成本。

目前我們實做的 IEEE 802.15.4 還有欠缺一些功能，例如說 GTS 的部分，如果完成了這部分的功能。還可以繼續在 IEEE 802.15.4 的標準上研究上 real time communicate 的議題；另外一個功能則是有關傳送加密的部分，這部分則又有通訊安全方面的研究可探討。未來將會朝這兩個方面去努力。另外在資料流量方面，比起 Freescale 的流量我們的確低了一點，所以這也是後續要改善的部份。

致謝：本文部份係工研院電通所執行經濟部委託之特殊具時效工業技術發展計畫”Embedded Linux 共通規格訂定及設計計劃”之成果

## 7. 參考文獻

[1] A. Rubini & J. Corbet ”Linux Device Drivers, 2e”, Jane 2001
[2] Chipcon at http://www.chipcon.com/
[3] D. P. Bovet & M. Cesati “Understanding the Linux Kernel” January 2001
[4] Figure 8 at http://www.f8w.com/
[5] Freescal 802.15.4 MAC/PHY Software

Reference Manual 802154MPSRM/D Rev. 0.5, April 2004
[6] Freescal 802.15.4 MAC/PHY Software User’s Guide 802154MPSUG/D Rev. 0.0, November 2004
[7] Freescale at http://www.freescale.com/webapp/sps/site/overview.jsp?nodeId=02XPgQhHPRjdyB, 2004
[8] Hynix at http://www.hynix.com/eng/index.jsp
[9] IEEE Standard for Information technology Telecommunications and information exchange between systems Local and metropolitan area networks Specific requirements Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs) IEEE Std 802.15.4™-2003，October 2003
[10] Microchip at http://www.microchip.com/zigbee, December 2004
[11] Motorola Confidential ProprietaryMotorola WMSG 2003
[12] Daintree Networks http://www.daintree.net/