無線網路基礎與應用 2015
TEAM07 Lab 4 Report
Wireless Sensor Network

蘇彥 高偉傑 黃子軒 楊孟翰 林蔚城

June 1, 2015

1 Introduction

Lab 4 模擬一個無線傳輸環境，利用兩台 Zigduino，分別當作 Src 和 Dst，以及擺設在 4F 指定地點的 6 個 Zigduino 來傳送封包，由於兩台 Zigduino (Src 及 Dst) 距離過遠，因此必須藉由中間的 Zigduino 當作中繼來轉送 Packet，因此必須建立 Routing Algorithm 完成傳輸，並盡可能的設立 MAC Protocol 提升傳輸的速率及可靠性，如: CSMA/CA、Error detection coding、Random backoff 等機制，減少封包在空氣介質中發生碰撞的機率，提高封包傳送的正確性、成功率以及降低 Round Trip Time，並透過傳送 100 個 ping 封包所得之數據來評估我們實作的效果。

Figure 1: CSIE 4F Routing 示意圖

2 Roouting Method

首先我們將 mode 分為 3 種，0 代表在 broadcast，1 為要回傳 backstr，2 則是在 ping。Mode 是依據收到的封包來決定的，RxBuffer 的第 9 格存有 mode 的資訊，因此我們會將此資訊與目前的 mode 來比較，除了 dst 收到 broadcast，要開始回傳 back string 時，以及 src 收到 back string 要開始 ping 之外，只允許 mode 由 0->1 or 1->2 or 維持原值，因此，不符合前述規定的封包將直接 drop 掉。下頁 DETAIL 的部分會說明各個 mode 如何運作。下圖為我們 Routing Algorithm 的流程圖 (Fig.2) 及傳送的封包內容示意圖 (Fig.3)。

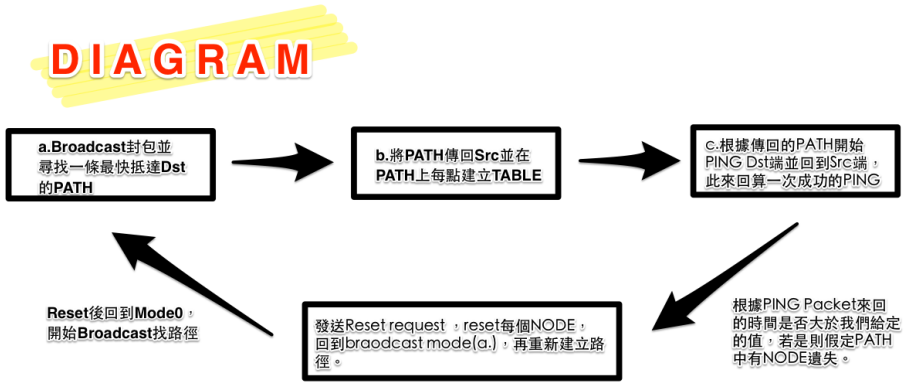

Figure 2: 演算法流程圖

Figure 3: 封包內容示意圖

簡單講解一下 code 的架構。在 void loop() 中，大致分為 Rx 、處理 Ping 相關、和 Tx 三個部分，而 Rx 和 Tx 因有 3 種 mode 和 Src、Dst、mid 而分成 $3 * 3$ 共 9 部分，Ping 則分為 Ping Index 為 1 至 100 和大於 100 兩部分。Rx 會依封包內容來決定是否要在 Tx 傳遞。Ping Index 介於 1 至 100 之間時，需要把 Tx_available 設為 1，如此才能在 Tx 繼續 Ping，大於 100 時則等待到 (第 100 個封包送出時間 +time out 長度) 時結束，最後印出 RTT 與成功率。

- **Detail**

a. **Mode** 0：**broadcast mode**(如 Fig.4)
收到其他 node 的 broadcast
Src Node：不做 transmit。
Mid Node：先檢查自己是否已被加入 path 中，若是已在 path 中則不做 transmit，不在的話就把自己加到 path 中，並繼續 broadcast 下去。
Dst Node：代表已有一條從 src 至 dst 的 path，將此 path 反轉成由 dst 至 src，並沿此 path 回傳 back string 給 src，最後填上 prenxt。

Figure 4: Mode 0 broadcast mode

b. **Mode** 1：**back mode**(如 Fig.5)
收到其他 node 的 back string
Src Node：填上 prenxt，將 mode 換為 ping mode，並準備 ping string 開始 ping。
Mid Node：填上 prenxt，並準備 back string，繼續朝 src 的方向傳回。
Dst Node：不做 transmit。

c. **Mode** 2：**ping mode**(如 Fig.6)
Src Node：收到封包時，先檢查 ping index 是否正確，再來檢查收到的封包是否 time out，如果沒有，則計算此封包 RTT 和紀錄成功封包數。要傳輸 ping 封包前，先紀錄封包出發時間，將 ping index 填入封包，最後將 ping index+1。
Mid Node：準備 ping string，並從封包判斷目前方向為從 src 到 dst 還是從 dst 到 src 後繼續往下一點傳。
Dst Node：準備 ping string 回傳。

Figure 5: Mode 1 backstr 傳遞

Figure 6: Mode 2 Ping Mode Begin

- **Rebuild Algorithm**
  當我們拔除一個點時，為了要讓所有點能知道這個點已經消失，我們設置了一個 ResetTime 的時間，開始 Ping 之後，Src 端每收到一次成功的 Ping 便會記錄下此時間值 lastpktt，而後只要 $|tnow - lastpktt| > ResetTime$，則表示封包已經不會再回來 Src 端，路徑極有可能出現中斷點，此時 Src 端會 broadcast 50 次 mode3 的封包去通知所有的 NODE 要 reset 所有值使大家回到 mode0。當所有點都已成功 reset 後，Src 也會將自己切換為 mode0 的模式並重新執行我們的主要演算法去計算新路徑。/bigskip

- **MAC Protocol**
  我們有針對以下機制做實作及原始碼改進，但或許是想法有誤，實測最後發現效果不如預期，甚至較原本 successful rate 及 RTT 低，也因最後花費大量時間在 Routing Protocol 部分的實作及實測，因此沒有針對 MAC Protocol 做的修正改進而選擇使用預設 Function:

  1. **Error detection code**
     在 Error detection 上，我們使用 $CRC-16$ 演算法，除數多項式採用常見的 16bits 多項式 $x^{16}+x^{15}+x^2+1$ $(0xA001)$，但實測後反而丟掉更多封包，因此暫時使用預設的 fcs function。
  2. **CSMA/CA**
     ZigdrunioExample 上已提供 CSMA API (getRssiNow() 及 doCca()) 可以實作，由於實作在 ZigduinoRadio.cpp 的 getRssiNow() 自行定義了 rssi 的值，並在 example 中 rssi 值為 -91 才做傳送。觀察過 SpectrumAnalyzer 分析後，rssi 經常落於 -91 至 -8X 中，因此設計了 threshold 機制。
     若 $|RSSI - RSSI\_BASE\_VAL| < threshold$ 即可傳送。原本以為這可以減少傳送時等待時間，但實測後發現預設等於 -91 再傳送效果較好。
  3. **Random backoff**
     在無線通訊系統中，backoff 機制是讓節點傳送時間錯開，減少 collision 次數之機制，我們的做法是模擬設定一個 content window，從中 random 一數乘上 slot time 後作為 delay 值，若無取得 Ack 再 double 取值。最後實測覺得在這個 lab 上效果沒有定值好，因此能採用定值。

- **Ping Test**

  a. **ping success rate :**
     當 SRC 端開始 ping100 次後，如果收到別的點傳來的 ping 且回傳的時間差在 Timeout 時間內，則表示有成功回傳，success 成功次數 $+1$。當 SRC 端送完 100 次之後，會在等待一個 Timeout 時間，當最後一個封包的時間加上 Timeout 時間到了後，立即計算當前的 success 個數，再除以 100 即為成功率。

  b. 我們的 RTT 計算圖示如下 (Fig.4)，初始我們先建立一個用來紀錄所有 ping packet 發送時間的 array，在 ping 從 src 端發送時利用 millis() 取得一個時間並記錄在對應的 array index 中，由於我們的 packet 中含有 ping 的 index 資訊以方便我們掌握 package，收到回來的 package 後擷取 ping 之 index，紀

錄目前時間，並與 array 中對應的位置時間做相減取得 RTT(以 ms 為單位)。

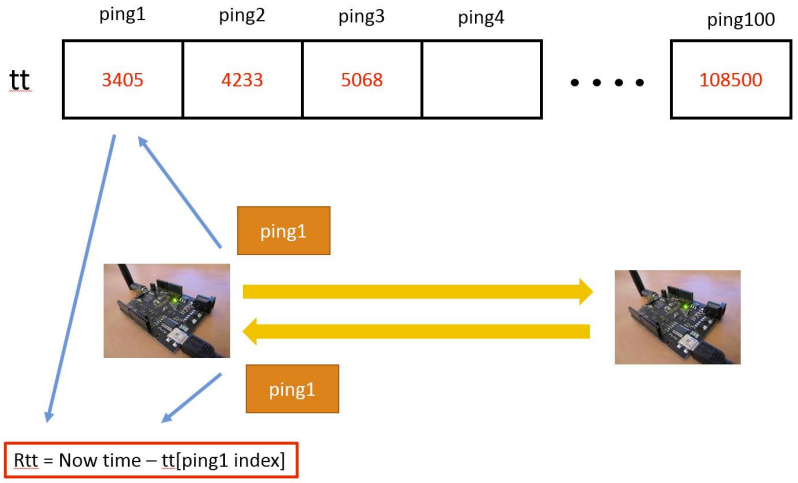

Figure 7: RTT Array & 計算示意圖

3 Problem & Solved

- 我們一開始 Routing 的方式是利用每次收到封包後，去檢查封包中特定位置的資訊，來取得接下來要傳送的 NEXTNODE 等資訊，但常常會因為封包遺失或 Crash 而導致整體路徑錯誤。

  解決方法: 我們利用 Mode.1 (如上圖 Fig.3) 傳回 PATH 時，在每個 ZIgduino 填 Routingtable 固定指向的 NODE，接下來每次收到封包，只需查 TABLE 即可得知下一點要傳給誰。也降低傳送失敗的機率。

- 因為我們是將經過的 NODE 一個個加進封包中，到 Dst 在 ReverseStr 傳回 Src 中間常常需要查找封包中特定 index 的資訊常常會找錯，DEBUG 下來花了許多時間，同時有時會也會因為很多環境變因造成。

  解決方法: 我們盡量減少查找 Packet 中資訊的次數，因為花時間且容易出錯，以及精算每次查 index 的位置 (多次 DEBUG 累積來)。

- 我們發現很多測試情況下，對傳或者只有一個中繼點的情況下出錯機率較低，但是當中繼點 -> 中繼點時，常常是問題癥結點的發生。

  解決方法: 我們只能藉由查看中繼點 Monitor 找出 BUG 所在，花了我們大部分時間，也有很多想到的想法但是仍有 BUG 導致最後無法使用，沒有在時間內能夠完成它。

4 Results

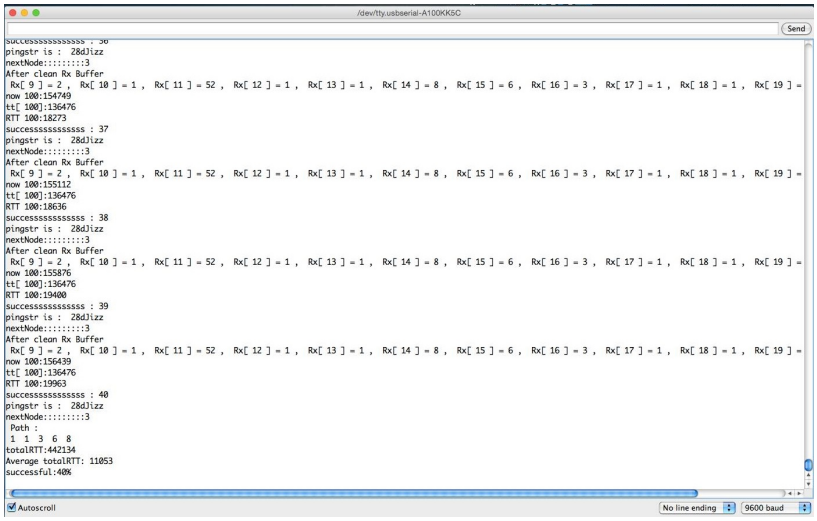

Figure 8: Result.

5 Work Division

蘇彥: MAC Protocol、Routing algorithm Debug and improve、Test、Report

黃子軒:Routing algorithm(broadcast, back to src)、packet format、reverse path、Test、Report

楊孟翰: MAC Protocol、Routing algorithm Debug and improve、Test、Write and Coordinate everyone's part of Report

林蔚城:Routing algorithm(broadcast, back to src)、clean noise in RxBuffer

高偉傑:Routing algorithm(ping)、Rebuild algorithm、mode design、drop packet mechanism、Test、Report

6 Reference

1. http://www.daxia.com/bibis/upload/ZigDuino%B5%C4Arduino%D1%A7%CF%B0%B1%CA%BC%C7.640.pdf