無線感測網路下非同步動態調整睡眠排程機制

江俊樺
台中教育大學資訊工程學系
研究生
luckbook222@gmail.com

張林煌*
台中教育大學資訊工程學系
教授
lchang@mail.ntcu.edu.tw

李宗翰
台中教育大學資訊工程學系
副教授
thlee@mail.ntcu.edu.tw

摘要

由於無線感測網絡（WSN）被能源消耗與作業系統的性能所侷限，為了確保無線感測網絡在任何時候都能有效率的運作，有眾多的睡眠排程機制被提出來解決耗能的相關問題。其中在感測器中接收與發送封包是最消耗能源的，因此藉由使用多媒體存取控制（MAC）協議來控制每個感測節點睡眠排程，來降低功耗以提升每個節點的壽命。在 X-MAC 機制中，縮短的 preamble 的方法能夠顯著降低能源的消耗，降低每跳延遲。在 ContikiMAC 中則是嚴格的限制封包的長度，藉由兩次的 CCA 來偵測頻道是否在忙碌。在我們提出的方法中，撰寫了許多程序和執行模擬陣列來驗證我們所提出的機制優於傳統提供 X-MAC 與 ContikiMAC 機制，並證明我們提出的方法在有隱藏節點的情境下有顯著的增益。

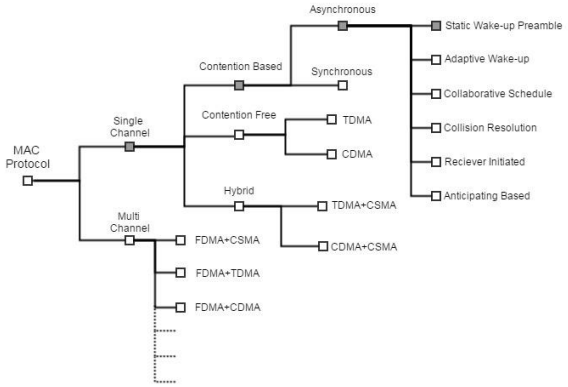
關鍵詞：無線感測網路、睡眠排程、能量消耗、即時流量。

Abstract

Because the wireless sensor network (WSN) is limited by the energy consumption and the performance of the operating system, in order to ensure that the wireless sensor network at any time can be efficient operation, there are a number of sleep schedules that have been proposed to address energy-related issues. Which in the sensor to receive and send packets is the most energy-consuming, thus, each sensor node sleep schedule is controlled by using a multimedia access control (MAC) protocol, to reduce power consumption to enhance the life of each node. In the X-MAC mechanism, the shortened preamble approach can significantly reduce energy consumption, reduce the delay per hop, in ContikiMAC it is strictly limited to the length of the packet, with two CCA to detect whether the channel is busy. In our approach, Compiling a number of programs and performing a simulation array to verify that the mechanisms we propose are superior to the traditional X-MAC and ContikiMAC mechanisms, and prove that our proposed method has significant gain in the case of hidden nodes.

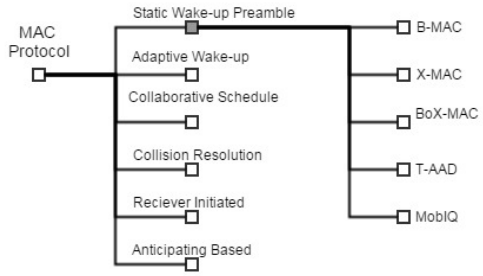
Keywords: Wireless sensing network, sleep scheduling, Energy Efficiency, Runtime Traffic

1. 前言



圖一、MAC 協議架構

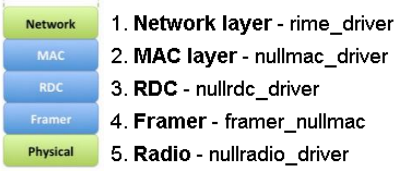
如圖一[1]所示，在多媒體存取層主要分為單通道與多通道兩種。Single Channel 是整個硬體主要都由一個天線做接收與發送，其中可以分為以避免碰撞機制 Carrier Sense Multiple Access (CSMA/CA)為基礎的同步與非同步機制；免競爭的分時多工 Time Division Multiple Access (TDMA)與對於編碼加以修改的 Code Division Multiple Access (CDMA)；最後將上述兩者結合的混合型。多頻道(multichannel)指的是硬體上的天線為兩根以上，藉由兩根以上的天線同時進行發送與接收，多頻道還能細分為將頻率上直接切割成多份的 Frequency Division Multiple Access (FDMA)，其中 FDMA 又可與 CSMA、TDMA、CDMA 做結合。在考量到在山區非必要時不進行傳輸，在傳輸需求低的情況下，雙天線造成額外的耗能，因此本篇主要以單通道非同步競爭型作為主要的研究方向。



圖二、MAC 協議靜態喚醒機制

在非同步喚醒機制中分為：Static Wake-up Preamble [2]固定長度的preamble；Adaptive Wake-up [3]會根據鄰居節點的排程方案來調整preamble的長度；Collaborative Schedule[4] 接收端接收到其他節點發送過來的封包後，設定與發送端相同的喚醒時間，並調整preamble長度；Collision Resolution 主要在處理CMSA中binary exponential backoff (BEB)的問題；Receiver Initiated(beacon enabled)[5] 當接收端醒來就主動的發送beacon告知發送端可以傳送資料，當發送端未偵測到有醒來的節點就可以直接進入睡眠，節省發送端的能源；Anticipating Based(beacon enabled)[6]機制是當目的端還未醒來，發送端將原先預計傳給目的端的資料傳給距離目的端一跳的節點，之後再由距離目的端一跳的節點進行轉送，節省發送端的能源，詳細如圖二所示。在本篇研究中主要研究的議題是Static Wake-up Preamble，注重節點的能量消耗與考量到程式碼的複雜性。

在 contiki 作業系統下，與傳統 TCP/IP 不同於傳輸協定被分為五層[7]，如圖三所示。Physical 層負責傳送與接收無線感測器所發送出來的訊號；Framer 層負責確認封包格式是否為 802.15.4 的格式；RDC 層來控制感測節點的睡眠排程；MAC 層則是決定是否要啟用 CSMA/CA 機制；最後由網路層則是將資料送到網路上。



圖三、Contiki 作業系統下網路五層架構

2. 相關文獻

本篇研究主要著重在 RDC 層的睡眠排程機制，在 X-MAC[8]是將醒來與睡眠時間設定為定值，分別將其舍為 6.25ms 與 118.75ms，在 preamble 得長度與傳統的 LPL[9]相較之下較短，藉此來降低 preamble 的耗能。在 X-MAC 使用到 fast sleep 和 phase optimization 這兩種方法來提升效能與能降低耗能，fast sleep 是當節點收到目的位址不是自己的封包時，就會進入睡眠，而本次的睡眠周期為 125ms；phase-lock optimizations 最早是由 WiseMAC[10]提出，目的是將記錄 ACK 的回傳時間，藉此來調整下次發送資料的時間，省去不必要的封包傳輸。

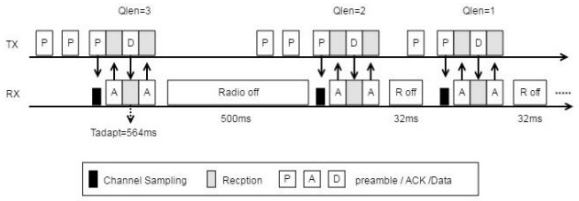
T-ADD[11]機制與 X-MAC 機制主要的差異在於 T-AAD 有兩種睡眠模式，第一種是還未收到資料封包之前的睡眠模式，此種模式預設的睡眠時間為 500ms；第二種是在接收到資料後的睡眠模式，預設的睡眠時間為 32ms 在每次發送端傳輸完資料後，接收端的睡眠模式會根據有無接收到資料來改變，演算法如圖四所示，之後再根據感測節點的預期要發送的資料來動態的調整睡眠排程。

Algorithm 1: Functionality of the T-AAD mechanism

```
begin
    when reception of packet P then
        if ST_max then
            if P.Q_len > 1 then
                calculate T_adapt;
                set_ST(ST_min) for T_adapt;
            else
                continue with (ST_max);
            end
        else
            if P.Q_len > 1 then
                calculate new T_adapt;
                if new T_adapt > current T_adapt then
                    set_ST(ST_min) for new T_adapt;
                else
                    continue with current T_adapt;
                end
            else
                continue with current T_adapt;
            end
        end
    end
end
```

圖四、T-AAD 演算法

在 T-AAD 機制下的當發送端有資料要發送時，會發送一連串的 preamble 來確定接收節點是否醒來，如圖五所示。

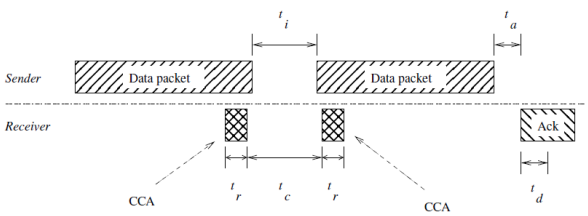


圖五、T-AAD 機制

假設現在共有三筆封包需要發送，在第一次接收端接收到 premable 時會先回傳 ACK 告

知發送端可以發送，之後將佇列中的封包數量(queue len)寫進 Data 的檔頭，讓接收端在收到後計算 T_{adapt}，如公式(1)所示，讓之後的睡眠周期由原先的 500ms 改為 32ms，並讓資料連續的接收。這種方法可以動態的根據資料發送的多寡來決定睡眠時間，但是還需要多等待一個 ST_{max} 才能反應。

$$T_{adapt} = ST_{max} + (Q_{len} - 1)\times(ST_{min})\times(1 + M_{err})..(1)$$



圖六、ContikiMAC 傳輸與 CCA 時間限制

在 ContikiMAC[12]中需要嚴格的限制封包的長度，如圖六所示。首先是T_c與T_i之間的關係，T_c必須比T_i長，藉此保證第一次與第二次的 CCA 可以檢測到有封包正在傳輸，而封包的長度也不能比 CCA 的長度T_r還短，確保兩次的 CCA 都能檢測到封包，因此封包最短的長度被制定為T_{rcr}因此可以得到如公式(2)所示的時間關係。

$$T_a + T_d < T_i < T_c < T_c + 2T_r < T_s .....(2)$$

在資料傳送完後的等待時間為T_a，T_a在 IEEE 802.15.4 中被定義為 12 個 symbol，其中一個 symbol 等於 4bit，其中 1bit 的資料傳輸時間為 1/250ms，因此T_a長度被定義為 0.192ms，而在接收到 ACK 的時候會先檢查同步標頭(SHR)，SHR 是由前置序列(preamble sequence)和區隔字元(start of frame delimiter，SFD)所組成，其中 SHR 長度為 5 個 symbol，因此可以確認從封包在接收到後，回傳 ACK 到接收端確認完畢的時間需等待 0.384ms，而在 CC2420 晶片定義 T_r時間為 0.192，因此可以得到如公式(3)

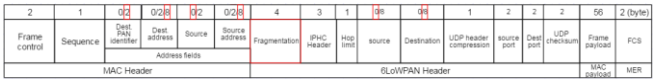
$$0.352 < T_i < T_c < T_c + 0.384 < T_s .....(3)$$

藉由公式(3)可以得知每個封包傳輸時間最短必須在 0.736ms，在傳送速率為 250Kbps 的情況下每個封包長度被限制至少為 23bytes，這個長度限制會根據 contiki 版本的不同而有所調整，藉由嚴格限制的封包長度，來降低整體耗能。

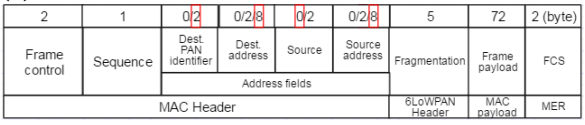
在本篇結合了 X-MAC 的 fast sleep 與 phase-lock optimizations，並使用 ContikiMAC 的睡眠機制，最後再結合類似 T-AAD 機制中預計傳輸封包的數量來動態的調整睡眠時間。

3. 研究方法

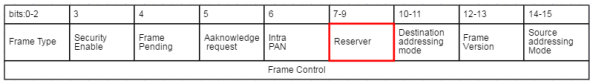
本篇研究主方法主要會以 ContikiMAC 作為基礎，因此每筆封包的最短長度都被限制在 23byte，若當資料小於 23bytes 時，則會在不足的部分則會補零，藉此確保每一筆資料都有符合長度限制。其中還修改了 X-MAC 中提到的 fast sleep，fast sleep 原先是當節點收到目的位址不是自己的封包時就會進入睡眠，在本篇修改的部分如圖七(a)所示，(a)指的是 6LoWPAN 下未分割的第一筆封包，當發送端有兩筆以上的封包要傳輸時，在 6LoWPAN 的機制下的第一筆封包是不做切割的，目的是保留來源與目的位址，並告知接收端之後還有多筆封包需要傳輸，其中在 6LoWPAN 下 Fragmentation 欄位佔了 4bytes，其中的 2bytes 是告知接收端這筆封包的 tag，另外 2bytes 則是記錄發送資料的總長度，在此則是透過計算 Datagram size 與之後每筆發送資料的 payload 來決定預期發送幾筆封包，之後發送封包的長度如(b)所示。最後要讓 Data 與 ACK 可以在 MAC 層做預期發送資料的判斷，因此需要將預期資料的參數放入 Frame Control 中所保留的 3bit 中，在此讓接收端接收時可以解析。



(a)6LoWPAN 下未分割的第一筆封包



(b)6LoWPAN 下分割後的封包



(c)Frame control

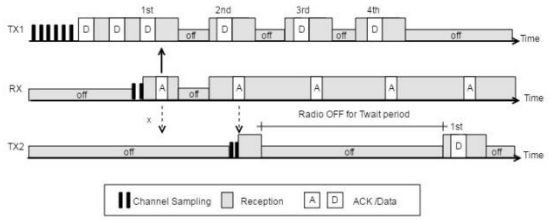
圖七、(a)(b)(c)6LoWPAN 下封包資料格式

最後當接收端解析目的位址不是給自己時，會檢查 Frame Control 欄位中的 reserve，當得知預期發送封包數量(F_{len})時，則會透過公式(4)計算 T_{wait}來調整睡眠時間。

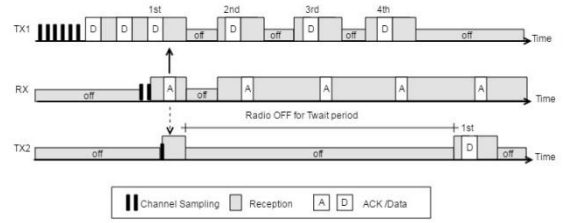
$$T_{wait} = F_{len}(TX_{time} + ACK_{time})(1+ M_{err}).....(4)$$

考量到隱藏節點的問題，如圖八(a)(b)所示，TX1 與 TX2 存在著隱藏節點的問題。(a)指的

是當 TX1 發送資料時，接收端 RX 在收到後回傳 ACK 告知 TX1 有收到，此時 TX2 隱藏節點正處於睡眠周期，因此沒有收到來自 RX 所傳遞過來的 ACK，TX2 醒來要傳送封包到 RX 時，接收到 RX 所傳遞過來的 ACK 後得知 T_{wait}，在等待 T_{wait} 後醒來在開始傳遞資料，避免掉不必要的發送封包來偵測 RX 的狀態，(b)指的是在 TX2 要發送封包時，正好接收到 RX 所傳遞的 ACK，提早進入睡眠，相比(a)下能省去一次的 CCA 時間。透過(a)(b)來避免掉隱藏節點與不必要封包所消耗的能量。



(a)隱藏節點 TX2 未收到 ACK

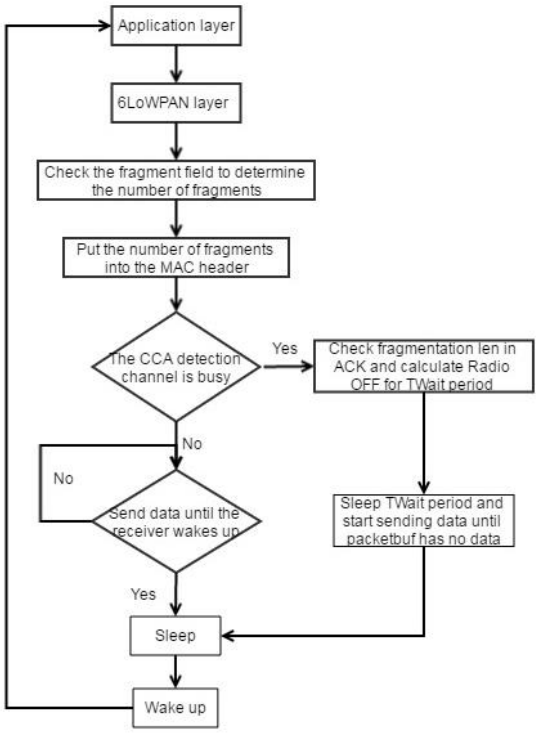


(b)隱藏節點 TX2 收到 ACK

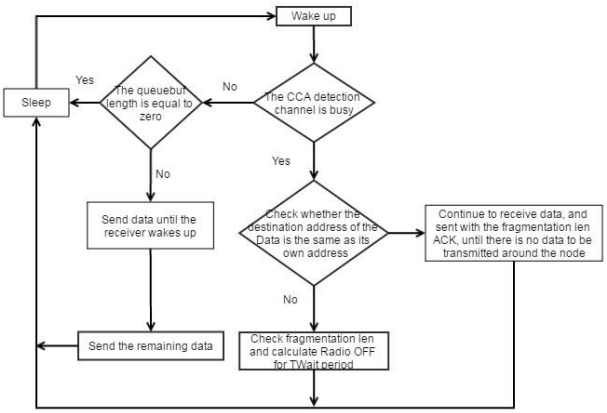
圖八、(a)(b)隱藏節點同時要發送資料

在發送端流程如圖九所示，在發送端一開始要傳輸封包時，應用層會先發送大量的資料下來，透過 6LoWPAN 來分割應用層所傳下來的封包，並將預期傳輸的封包數量解入 MAC 檔頭，讓資料能夠在 MAC 層解析，之後因為要傳輸封包開始 CCA 偵測通道是否在忙碌，沒有在忙碌的話就直接開始封送資料直到接收端醒來，接收端在醒來後會開始把剩餘的資料發送完，若此時應用層又有新的資料進來，則會到下一次醒來週期在開始發送，避免節點一直佔據通道的問題，在 CCA 偵測時若通道是忙碌的狀態，則代表有其他節點在傳輸，因此會判斷接收端是否有傳輸 ACK 過來，這個 ACK 表示的不是自己傳遞資料而傳遞過來的 ACK，而是在接收端收到 TX1 所傳遞過來而傳的 ACK，藉由解析這個 ACK 來計算 T_{wait}，之後就睡眠 T_{wait} 直到醒來，醒來後就開始直接發送封包。

轉送端流程如圖十所示，在節點醒來後會先檢查 CCA 是否忙碌，若沒有忙碌的話會檢查自己的 queuebuf 是不是等於零，若資料等於零表示沒有資料要發送，之後就會進入睡眠；當 packetbuf 不等於零時表示有封包要發送，此時會開始發送一連串的未被 6LoWPAN 的封包，偵測接收節點是否醒來，第一筆封包發送最多可以傳送 31 個封包，剛好就一個週期，確保接收端接收到，之後接收端會透過 phase-lock optimizations 的技術來記錄第一次傳輸封包被接收的 ACK 時間，藉此來調整下次發送第一筆封包的時機，當接收節點醒來時就會將剩餘的資料傳遞完畢，並進入睡眠等待下次醒來。當 CCA 是忙碌時，代表有其他節點正在傳輸，此時會先檢查 ACK 封包目的位址是不是跟自己的位址相同，相同的話就繼續接收資料直到結束，不相同則是計算 T_{wait} 並進入睡眠。



圖九、發送節點流程圖



圖十、轉送節點流程圖

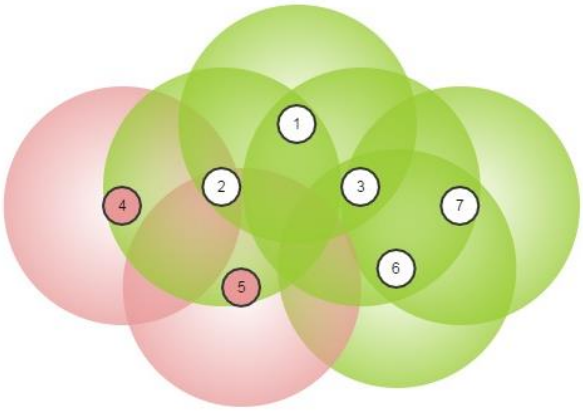
4.實驗環境

本研究使用 Contiki 所提供的 cooja 模擬器來進行數值分析，如表一所示，模擬硬體使用 SKY 無線感測模組；其中 SKY 使用的微控制器與發送晶片分別為 MSP430 與 CC2420；作業系統為 Contiki2.7。模擬網路環境使用具有重傳機制的 CSMA/CA；傳輸協議使用 Zigbee。

表一、實驗環境

MCU/SoC	MSP430
Radio	CC2420
Platforms	SKY Mote
OS	Contiki2.7
Adaptation layer	6LoWPAN
MAC layer	CSMA/CA
RDC layer	ContikiMAC
Frames layer	802.15.4
Channel check rate	8 Hz
Shortest packet size	43bytes
The first packet of the payload	56byte
The remaining payload size	72bytes
CCA count MAX	2
CCA count MAX TX	6

拓樸情境使用 6 個 sender 與 1 個 sink 做為模擬情境，節點拓樸如圖十一所示，node4、node5、node6、node7 每十秒分別發送一筆資料，每筆資料大小為 400byte，其中 node6 與 node7 作為 node4 與 node5 的比照組，cooja 模擬時間為三十分鐘。



圖十一、網路拓樸情境

$$T_{cpu} * P_{cpu} + T_{radioRX} * P_{radioRX} + T_{radioTX} * P_{radioTX} + T_{lpm} * P_{lpm} \ldots\ldots(5)$$

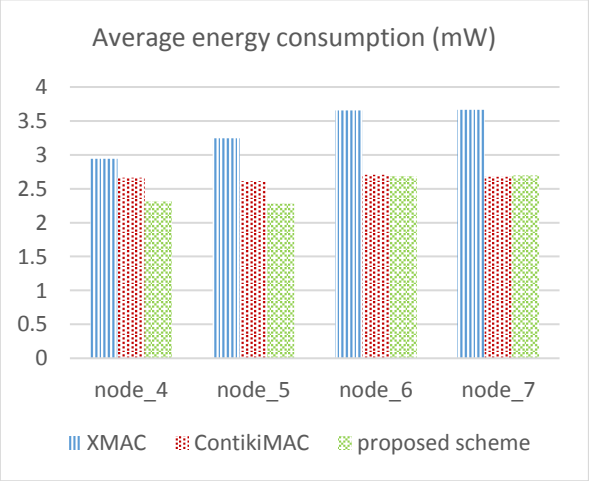
在耗能計算的部分如公式(5)所示，本篇研究主要將透過 Cooja 模擬器取得每一個狀態的時間。在 CC2420 晶片發送與接收都是在晶片上進行運作，而 cpu 與 low power mod(LPM)則是在 MSP430 運行，因此總耗能由 CPU 運作時間加上低耗能模式(LPM)的運作時間，每一次接收與發送則是另外進行運算，再由 SKY 的模組的各模式下的電流與電壓來算出功率，電流與電壓的設定如圖十二所示，最後在將其乘上模擬時間，藉此來分析能源消耗。

Typical Operating Conditions

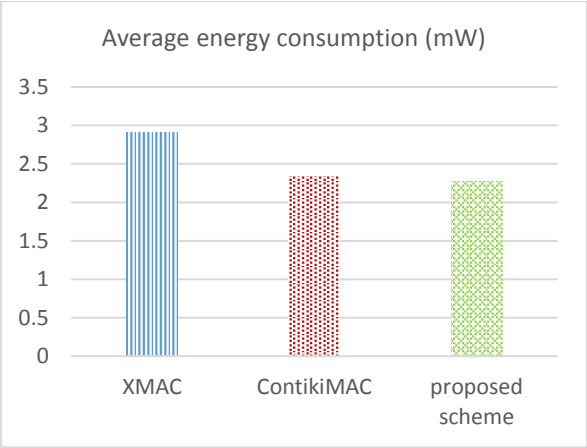
	MIN	NOM	MAX	UNIT
Supply voltage	2.1		3.6	V
Supply voltage during flash memory programming	2.7		3.6	V
Operating free air temperature	-40		85	°C
Current Consumption: MCU on, Radio RX		21.8	23	mA
Current Consumption: MCU on, Radio TX		19.5	21	mA
Current Consumption: MCU on, Radio off		1800	2400	µA
Current Consumption: MCU idle, Radio off		54.5	1200	µA
Current Consumption: MCU standby		5.1	21.0	µA

圖十二、SKY mote 的電流與電壓

分析結果如圖十三所示，我們所提出來的方法在一般情況下與 ContikiMAC 的能源消耗相近，但隱藏節點 node4 與 node5 相較於 X-MAC 與 ContikiMAC，能源的部分有顯著的下降。最後在比較整體能源消耗，如圖十四所示，相較於 ContikiMAC 差異不大，主要原因接收端在接收到封包時不會立刻進入睡眠，還會等待一段時間後才會進入睡眠，因此之後研究還會改善程式碼的流程，藉此降低整體耗能。



圖十三、一般組與對照組能量消耗



圖十四、整體平均能量消耗

5.結論

此篇研究主要在分析無線感測網路上不同睡眠排程的能源消耗與延遲分析，在 X-MAC 下藉由縮短前導封包 preamble 的長度來降低每一跳的延遲，藉此也縮短每個節點的等待時間，降低能源消耗；在 ContikiMAC 則嚴格的限制封包長度，讓整體的工作週期下降；在 T-AAD 中在接收端收到封包後降低睡眠時間，在根據每個節點的發送封包來動態的調整睡眠排程，而本研究則使用類似於 T-AAD 預期傳輸封包的概念，以 ContikiMAC 作為主要改良的機制，透過 6LoWPAN 來預測傳輸的封包數量，藉由此方式提高整體傳輸效能與降低能源消耗。並透過 Cooja 來模擬感測節點在不同情境，並與 X-MAC 和 ContikiMAC 比較，結果顯示我們所提出的機制在有隱藏節點的情境下，隱藏節點上的能源消耗有顯著的下降，未來會改善程式執行的程序，在接收端在接收到封包時的判斷與執行序，藉此降低整體耗能。

參考文獻

[1] Messaoud, Doudou, Djenouri Djamel, and Badache Nadjib. "Survey on latency issues of asynchronous MAC protocols in delay-sensitive wireless sensor networks." IEEE Trans. on Communications Surveys Tutorials, vol. PP 99 (2012): 1-23.

[2] J. Polastre, J. Hill and D. Culler, “Versatile low power media access for wireless sensor networks,” Proc. 2nd international conference on Embedded networked sensor systems (SensSys04), USA. November 03- 05, 2004, pp 95-107.

[3] A. El-Hoiydi and J.-D. Decotignie, ‘Low Power Downlink MAC Protocols for Infrastructure Wireless Sensor Networks,” Mobile Networks and Applications (MONET), ACM/Kluwer Academic Publishers, 2005, Vol 10, No 06, pp 675-690.

[4] Q. Yu, C. Tan, H. Zhou, “A Low-Latency MAC Protocol for Wireless Sensor Networks,” International Conference on Wireless Communications, Networking and Mobile Computing (WiCom07), Shanghai, Sept. 21-25, 2007, pp. 28162820.

[5] Y. Sun, O. Gurewitz, and D.B. Johnson, “RI-MAC: a receiver-initiated asynchronous duty cycle mac protocol for dynamic traffic loads in wireless sensor networks,” Proc. 6th ACM conference on Embedded network sensor systems (SenSys’08), 2008, pp. 1-14.

[6] S. Rashwand, J.V. Misic, V.B. Misic, S. Biswas, and Md.M. Haque, “A Novel Asynchronous, Energy Efficient, Low Transmission Delay MAC Protocol for Wireless Sensor Networks, ” ICDCS Workshops Proceedings, 2009, pp. 186-193.

[7] http://anrg.usc.edu/contiki/index.php/MAC_protocols_in_ContikiOS

[8] Buettner, Michael, et al. "X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks." Proceedings of the 4th international conference on Embedded networked sensor systems. ACM, 2006.

[9] Merlin, Christophe J., and Wendi B. Heinzelman. "Duty cycle control for low-power-listening MAC protocols." IEEE Transactions on Mobile Computing 9.11 (2010): 1508-1521.

[10] El-Hoiydi, Amre, and J-D. Decotignie. "WiseMAC: an ultra low power MAC protocol for the downlink of infrastructure wireless sensor networks." Computers and Communications, 2004. Proceedings. ISCC 2004. Ninth International Symposium on. Vol. 1. IEEE, 2004.

[11] Papadopoulos, Georgios Z., et al. "T-AAD: Lightweight traffic auto-adaptations for low-power MAC protocols." Ad Hoc Networking Workshop (MED-HOC-NET), 2014 13th Annual Mediterranean. IEEE, 2014.

[12] Adam Dunkels. The ContikiMAC Radio Duty Cycling Protocol. Technical Report T2011:13, Swedish Institute of Computer Science, December 2011.