行政院國家科學委員會補助專題研究計畫 □ 成 果 報 告
■期中進度報告

ZigBee 無線樹狀感測網路之初始化及通訊問題研究

計畫類別：■ 個別型計畫　　□ 整合型計畫
計畫編號：NSC 97－2221－E－009－142－MY3
執行期間：2009 年 08 月 01 日至 2010 年 07 月 31 日

計畫主持人：曾煜棋
共同主持人：
計畫參與人員：

成果報告類型(依經費核定清單規定繳交)：■精簡報告　□完整報告

本成果報告包括以下應繳交之附件：
□赴國外出差或研習心得報告一份
□赴大陸地區出差或研習心得報告一份
■出席國際學術會議心得報告及發表之論文各一份
□國際合作研究計畫國外研究報告書一份

處理方式：除產學合作研究計畫、提升產業技術及人才培育研究計畫、列管計畫及下列情形者外，得立即公開查詢
■涉及專利或其他智慧財產權，□一年□二年後可公開查詢

執行單位：

中　華　民　國　99 年　05 月　31　日

無線感測網路相關的研究議題得到許多研究單位及學者的關注，近年來ZigBee通訊協定被視為最適用於感測網路的通訊協定，本計畫將研究以ZigBee 樹狀網路為基礎之無線感測網路相關通訊協定，本計畫為三年之計畫，其目標主要包含三個面向：(1) ZigBee無線感測網路生成之研究，(2) ZigBee樹狀網路資料傳遞排程，(3) ZigBee基礎之長鏈狀網路研究。在第二年中，本計畫將討論ZigBee網路中之資料收集所遇到之問題。Convergecast在無線感測網路中是一個很基本的應用，而目前的convergecast大多著重於降低回報延遲及能源的消耗，然而，一個好的設計除了考慮這些因素外，還必須和標準相容。因此，本計畫於ZigBee樹狀網路中，針對快速回報定義了一個最小延遲的訊框排程問題，並証明其為一個困難的問題，同時使此問題相容於IEEE 802.15.4之規範，並且於某些特殊的網路型態中，提出了最佳解法；於一般網路型態型，提出了集中式及分散式的時槽分配方法。

關鍵字：convergecast、圖形理論、IEEE 802.15.4、排程、無線感測器網路、ZigBee

Recently, a lot of research works have been dedicated to the *wireless sensor networks (WSNs)* field. ZigBee is a communication standard which is considered to be suitable for WSNs. In this project, we discuss initialization and communication protocols for ZigBee tree-based WSNs. This project contains three research topics including 1) formation of a ZigBee-based WSN, 2) scheduling for ZigBee tree-based networks considering data flows, and 3) ZigBee-based long thin networks. In the second year, we discuss the data gathering in ZigBee networks. Convergecast is a fundamental operation in wireless sensor networks. Existing convergecast solutions have focused on reducing latency and energy consumption. However, a good design should be compliant to standards, in addition to considering these factors. Based on this observation, this work defines a *minimum delay beacon scheduling problem* for quick convergecast in ZigBee tree-based wireless sensor networks and proves that this problem is NP-complete. Our formulation is compliant with the low power design of IEEE 802.15.4. We then propose optimal solutions for special cases and heuristic algorithms for general cases.

Keywords: convergecast, graph theory, IEEE 802.15.4, scheduling, wireless sensor network, ZigBee.

目錄

一、 前言

在許多無線網路的應用中，網路上的裝置需要將應用層所產生的感測資料回報至一資料收集基地台(Sink)，然而在某些特殊的網路應用中，如監控、保全等，裝置並不是隨時皆要傳遞資料，因此節點可依照需求進入休眠模式以減少電量消耗。ZigBee[1]協定中亦提供了一信標網路 (Beacon-enabled network) 架構來支援此類需求。在此類型的網路中，網路的拓樸必須為 ZigBee 樹狀網路，網路中的協調者(Coordinator)當作是根節點(root)，並同時為資料收集基地台(Sink)，Sink 可以連結若干中繼節點(Router)作為其子節點(Child)，中繼節點亦可以連結若干節點當作其子節點。在此網路中，裝置採取類似分時多工(Time Division Multiple Access，TDMA)的技巧來分配節點的傳輸時間，網路當中的 Sink 發送 Beacon 封包宣告網路超級訊框(Superframe)格式，在一個超級訊框(Superframe) 時間內，時間被切割成多組時槽(Time Slot)，而 Sink 佔領一時槽用以接收或傳送網路上其他裝置之資料，而剩餘之時槽可供網路中的其他中繼節點使用，中繼節點亦藉由發送 Beacon 封包宣告其所佔據之時槽，當一節點 D 收到某 Router A 之 Beacon，節點 D 可在 Router A 所佔領之時槽傳送給 Router A，或是接收 Router A 所要傳送給節點 D 之資料。

然而網路上的節點不能恣意地選擇時槽，當一網路上的節點D欲傳遞資料時至Sink時，它必須等待其父節點(假設為 Router A)之 Beacon，假若有另一Router B與Router A選擇相同之時槽，且Router B亦為節點D之鄰居，此時Router A與Router B會同時發出Beacon，則會造成Beacon在節點D處發生碰撞，由於節點D接收不到Router A之Beacon，則因此無法傳遞資料給Router A。倘若此時網路上有許多的Beacon發生碰撞，則整個網路的運作將為之癱瘓。因此設計一個能避免Beacon碰撞的排程演算法是相當重要的，同時，除了要避免Beacon 碰撞外，Beacon的排程演算法亦要考慮傳遞延遲之問題。

目前有幾篇文獻[2][3][4][5][6][7]也探討關於省電之網路資料回報之排程，在文獻[6]中，作者探討在基於欲達到低回報延遲以及低能量消耗的目標下，建構出一個平衡樹(balance tree)連接網路節點，並且指定分碼多工編碼(CDMA code)給網路節點用以降低傳輸干擾影響已達到節能目的。在文獻[7]中，作者探討在最小化能量消耗的情況下，該如何使得資料的回報能夠在一個指定的時間內到達資料收集伺服器，作者提出一個動態規劃演算法(Dynamic programming algorithm)來計算出一動態排程，但是該演算法必須假設每個節點可以同時間收集數個資料封包。由此可見文獻[6]及[7]皆必須假設節點的能力(可使用CDMA編碼及同時接收多個封包)，且所提出之演算法不符合ZigBee 規範。在文獻[5]中，作者提出一個低功耗及低延遲的媒體擷取層通訊協定(MAC protocol)，稱之為DMAC，網路中的感測器使用一個樹狀結構連結，並且節點們長時間皆處於睡眠狀態，當感測器由睡眠狀態切換至活動狀態時，該感測器先將無線模組狀態開啟於接收狀態然後再切換至傳輸狀態，DMAC 使用階梯式排程方式來達到低傳輸延遲之目的，也就是說一感測器先接收來自子節點之封包後，切換至傳送模式轉發封包給其父節點。類似於文獻[5]之方法，在文獻[4]中作者亦使用階梯式排程方式，並且考量網路資料流流量，在此網路中，父節點定期廣播其剩餘之時槽數目，其子節點可依據其流量大小，向父節點註冊所需要之時槽數。在文獻[3]中，作者提出一分散式回報排程演算法，其基本的概念為先將感測器使用一展延樹連接起來，

接著將該樹簡化成數個重疊之線性拓樸，針對每個線性拓樸，該演算法則由下到上(節點至資料收集伺服器)方向排程節點之傳送時間。而文獻[2]中，作者提出一個集中式演算法來排程封包傳遞，該演算法將感測器分割成數個區段，每個區段內的感測器之傳輸不會影響在同於該區段內的感測器之傳輸，簡單來說，文獻[2]中所提出的方法為創造出多個可平行處理之傳輸區段以降低回報延遲。雖然文獻[2][3][4][5]所提出的方法皆與本計畫之目標一致，但是所提出之演算法皆不適用於ZigBee 網路。

二、 背景知識

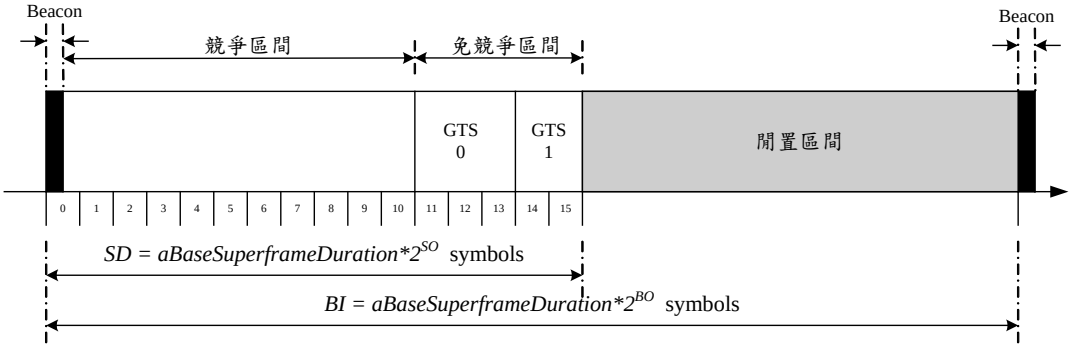


圖 1：IEEE 802.15.4 Superframe 架構

在 IEEE 802.15.4[8]中，定義了兩種網路設備，分為：Full-Function Device (FFD)與 Reduced-Function Device (RFD)。一個 FFD 具有 Personal Area Network (PAN) Coordinator、Router 與 RFD 等功能，它可以與 RFD 或是 FFD 進行資料傳輸。RFD 是一個極其簡單的網路設備，只具備與單一 FFD 進行資料傳輸的功能。

在 IEEE 802.15.4 中，FFD 允許採用 Superframe，所謂的 Superframe 是一段由同一個 Coordinator 所發的兩個連續的 Beacon 所限定的時間區段(Beacon interval, BI)。在 Beacon-enabled network 下發 Beacon 的目的是要作同步化，任何設備想要進行資料傳輸都必須在屬於 Superframe 內的活動區間內以 slotted CSMA/CA 的方式對傳輸通道進行存取。活動區間又可分為競爭區間與免競爭區間，協調者只在活動區間和個人區域網路中的裝置互動收送資料，而在閒置區間則可以進入省電模式(休眠)以減少電源消耗。對於協調者以外的裝置來說，它們在沒有資料要傳送時可以進入省電模式。

Superframe 的結構由 Beacon 訊框中包含的資訊來決定，Beacon Order (BO)決定 Beacon 發送的間隔而 Superframe Order (SO)則決定活動區間的長短，SO 及 BO 為介於 0 至 14 間之整數且需符合 SO≦BO 之限制。活動區間的長度被設定為：

$$aBaseSuperframeDuration \times 2^{SO} \text{ symbols}，$$

aBaseSuperframeDuration 為 IEEE 802.15.4 預設參數，其值為 960 symbols，而一個 symbol 的大小依據所使用之不同的操作頻帶有著不同的設定，例如操作於 2.4 GHz 的網路，一個 symbol 的大小為 16 *us*。而一個 Superframe 的長度為：

$$aBaseSuperframeDuration \times 2^{BO} \text{ symbols}。$$

由此設定一個操作於 2.4 GHz 頻帶的網路，Superframe 的長度可從 15.36 *ms* 到 215.7 秒不等。當 SO=15 時代表不使用 Superframe 的架構。目前在 IEEE802.15.4 標準中對於 Beacon-enabled 網路的說明僅限於星狀網路，對於多節網路尚未提出完整的解決方案。然而在 IEEE 802.15.4b 裡則會修正此問題。在 IEEE 802.15.4b 中，BI 的長度為所收到之父節點之 Beacon 間隔，裝置可在 Beacon 間隔中挑選一時槽來傳送其信標(如圖 1 所示)。

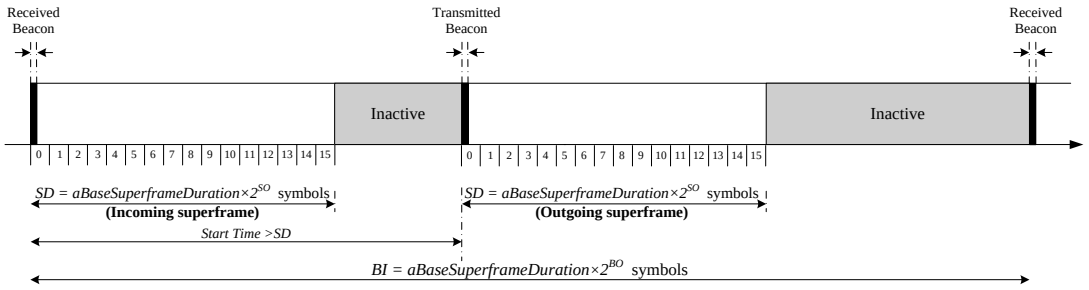


圖 2: IEEE 802.15.4b 中所允許之信標網路模式

在 IEEE802.15.4 標準中對於 Beacon-enabled 網路的說明僅限於星狀網路，對於多點跳躍網路(Multihop networks)尚未提出完整的解決方案。在新版規範 IEEE 802.15.4b[9]中則修正此問題。在 IEEE 802.15.4b 中，Superframe 的時間長短為所收到之父節點之 Beacon 間隔(BI)，裝置可在 Beacon 間隔中挑選一時段來傳送其 Beacon (如圖 2 所示)。當收到父節點的 Beacon 時，該裝置必須切換至 Active 模式並且保持該狀態直到該活動區間結束，這一段時間被稱作為 Incoming superframe，然而該裝置亦可以發送自己的 Beacon，並且可使得其子裝置保持清醒一段時間，該段時間稱作為 Outgoing superframe。由上討論可知一個父節點之 Outgoing superframe 即為其子節點之 Incoming superframe，而為避免子節點無法辨識 Incoming superframe 的到來，在選擇 Outgoing superframe 時，必須避免 beacon 發生干擾之情形。

三、 研究目的

除了要避免 Beacon 碰撞外，Beacon 的排程亦要考慮傳遞延遲之問題。以圖 3 之網路環境為例，圖上的虛線為不能使用相同時槽之路由器，以下我們展示不同的信標排程將會造成不一樣的傳輸延遲。

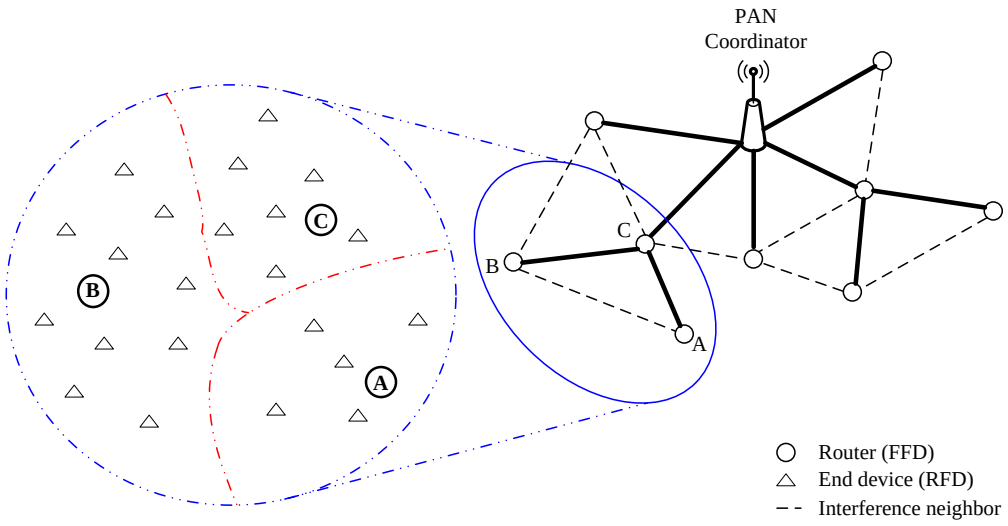


圖 3：ZigBee 網路資料傳遞情境

圖 4 為一信標排程之例子，網路上的資料流都是由裝置流向主裝置，圖 4

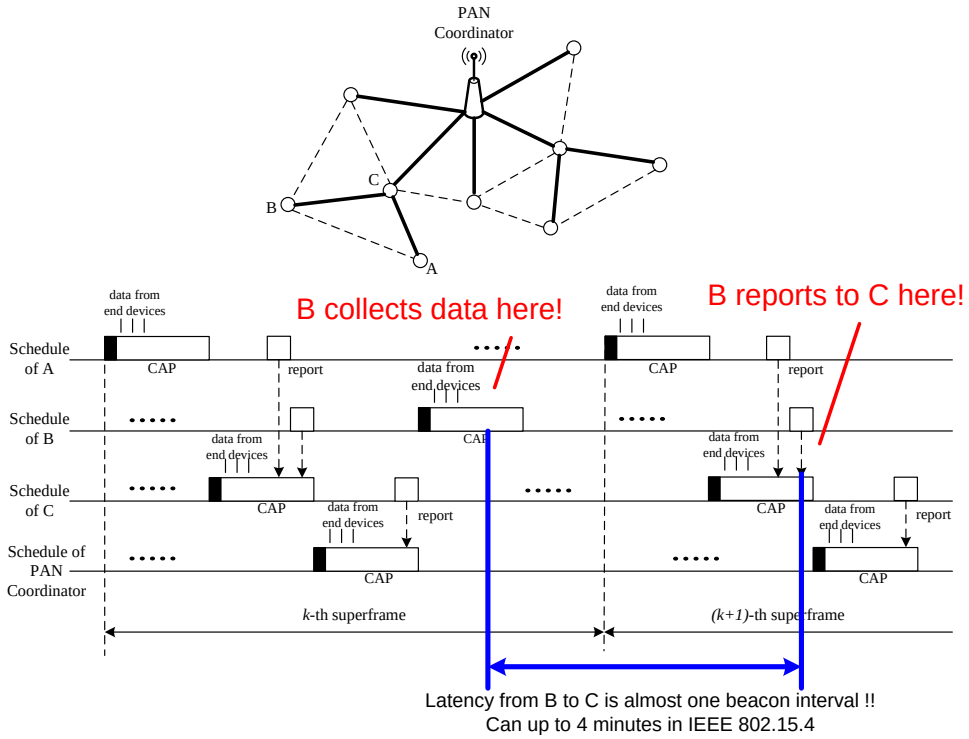


圖 4：一個信標排程例子

中的大方框為每個裝置所佔領之時槽，在時槽的一開始時佔領該時槽的裝置先發送信標，接著則開始競爭區間來讓其底下的裝置傳送資料給它。在圖 4 的例子中，路由器 A 及 B 聆聽路由器 C 的信標，並且在活動區間內利用競爭之方式將資料送給路由器 C。而路由器 C 也必須要聽取協調者之信標。由圖 4 的排程中，路由器 B 的時槽落於 C 之後，所以當路由器 B 有資料要送給 C 裝置時，必須等待將近一個 Beacon 間隔才會等到路由器 C 的下一個信標。當網路 Duty Cycle 很小時，Beacon 間隔可能較一時槽大數倍，例如在 2.4GHz 的頻帶下，當設定網路的 Duty cycle 為 1.56%，則一個 Beacon 間隔長度為 251.7 秒，而時槽長度僅為 3.9 秒。因此上述之例子整體網路的回報延遲將長達四分多鐘，然而如果網路的範圍更大，使用不佳的排程方法，由裝置到主裝置間訊息的傳遞可能會造成相當長的延遲。

圖 5 顯示了一個改善圖 4 信標排程之例子，在圖 5 的排程中，訊息傳遞的延遲的計量單位縮短為數個時槽，所以在修正過後的排程的回報延遲可以縮短至三個時槽長度(也就是 10 秒左右)，這顯示了不同之排程方式將可以有效的降低傳輸延遲，且同時間亦可以達到省電之目的。

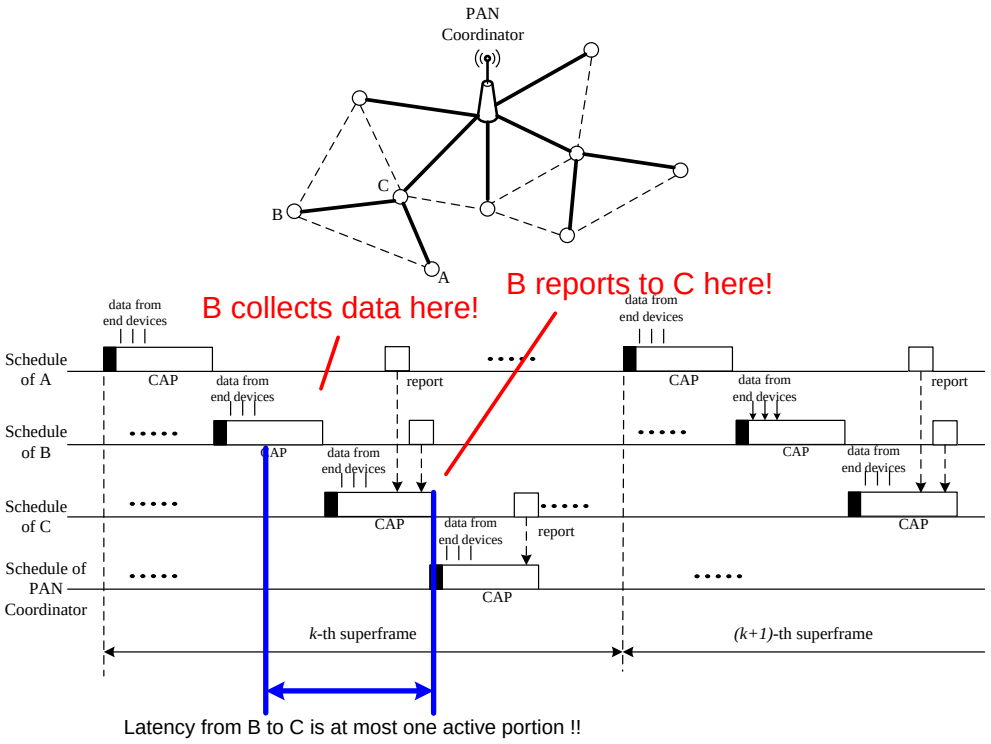


圖 5：一較佳的信標排程

四、 研究方法

首先，在此計畫中，我們先定義此時槽安排之問題為 Minimum Delay Beacon Scheduling (MDBS) Problem。給予一個 ZigBee 網路，針對具 Router 能力的裝置(Router-capable device)，且考慮所有具有 Router-capable 能力之裝置中其通訊連結皆為雙向時，則可將此節點和連結表式成一個圖 G=(V, E)。同時，於 G 中，我們再將 Router 間會產生直接干擾及間接干擾的兩點連立連線，最後變成產生圖 $G_I=(V, E_I)$。而此時槽安排問題的目標便是要替每一個 Router i 找出一個時槽 s(i)，而如果 Router i 和 Router j 於 GI 中存在有一邊時，則 s(i) 不等於 s(j)。在給定每一個 Router 一個時槽之後，Router i 到 Router j 之間的回報延遲便如下之定義：

$$d_{ij} = (s(j) - s(i)) \bmod k$$

因此，我們可以將圖 G 最後轉換成一有向之權重圖 $G_D=(V, E_D)$，如果(i, j)存在於 E 中時，則 E_D 將存在有(i, j)和(j, i)兩個有向邊，其中(i, j)之權重為 d_{ij}，而(j, i)之權重為 d_{ji}。由 G_D 更可得知網路上之每個節點至基地台之回報延遲時間為此節點至基地台的最短路徑權重總合，而整個網路之回報延遲時間 L(G)便為所以節點之回報延遲時間最大者。

定義一 給定一圖 G=(V, E)，MDBS 問題便為替每一個網路節點找出一個無干擾之時槽安排演算法，且可以使得網路之回報延遲時間 L(G)最小。

定義二 給定一圖 G=(V, E)以及一個延遲時間限制 d，則 Bounded Delay Beacon Scheduling

(BDBS)問題便為決定出是否存在一無干擾之時槽安排方法，可以使得 L(G) ≤ d。

因此，本計畫證明了一個定理如下：

定理一 BDBS 問題為一個困難(NP-complete)的問題。

MDBS 問題之解決方案

在本計畫中，根據上面的分析與定義，我們針對不同的 ZigBee 網路型態，分別提出不同之解決方法：

特殊網路型態之最佳解決方案

如圖 6 所示，當 ZigBee 網路型態為一個線狀(Regular Linear)或者為一環狀(Regular Ring)網路時，MDBS 問題將存在一多項式時間之最佳解。在此網路型態中，每一個節點之連結情形可影響至 h hop 遠之節點，所以於 G_I 中，每一個節點之最大 degree 數為 2h。因此，於線狀網路型態中，我們可以採用 bottom-up 的方式來安排時槽，最底下之節點(也就是離 Sink 最遠之節點)其時槽為 0，而其他節點 v 之時槽 s(v)安排方式如下：

$$s(v) = (k' + 1) \bmod k$$

而其中 k’ 為 child 節點之時槽，k 為網路中所有的時槽數。很明顯的可以看出，此時槽安排法的確為一最佳的時槽安排演算法。而圖 6(a)即為線狀網路時槽安排的一個例子。

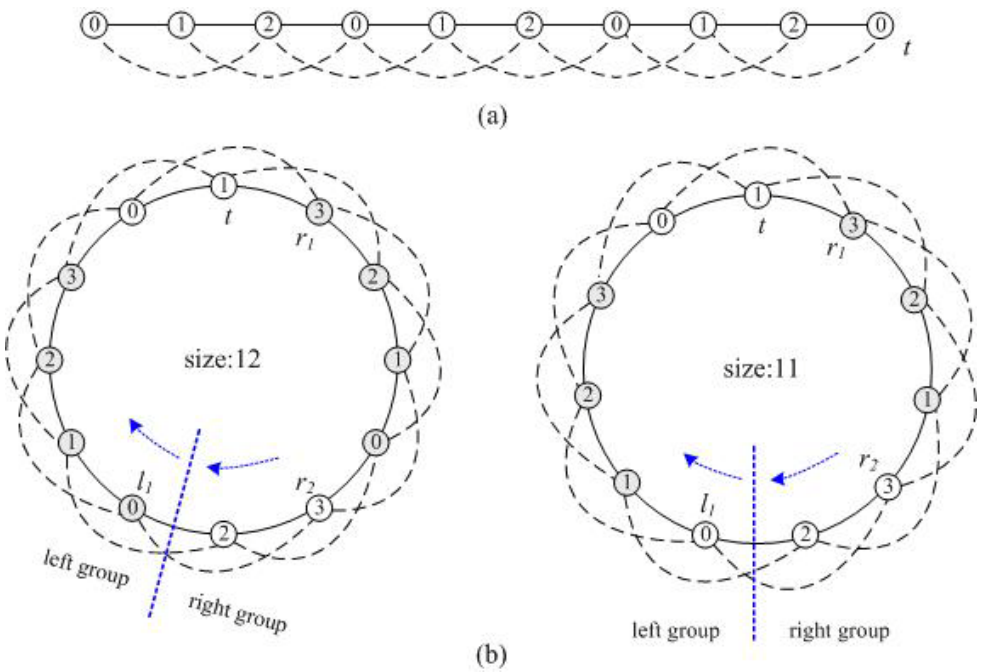


圖 6：線性及環狀網路之最佳時槽分配法範例

於環狀網路型態中，首先我們先依照 Sink 節點，將此網路分成左群和右群節點，而左

群節點的成員為 Sink 節點以及 Sink 往左開始算至第 $\left\lfloor \frac{|V|-1}{2} \right\rfloor$ 個節點，而其他節點便屬於右群節點。而決定出左群和右群節點之後，便可將其看成兩個線狀網路，而時槽安排方法如下：

1. 左群節點之時槽安排和線狀網路安排法相同。
2. 右群節點之時槽安排為 top-down 之方式由 Sink 開始往下安排，而對於每一個位於右群節點內之節點 v，其時槽為 s(v) = (j － c)，而其中 j 為其父親節點之時槽，而 c 為確保 s(v)不會和其互相干擾鄰居節點時槽重覆下中之最小整數。

圖 6(b)即為環狀網路時槽安排的一個例子。

集中式樹狀時槽安排解決方案

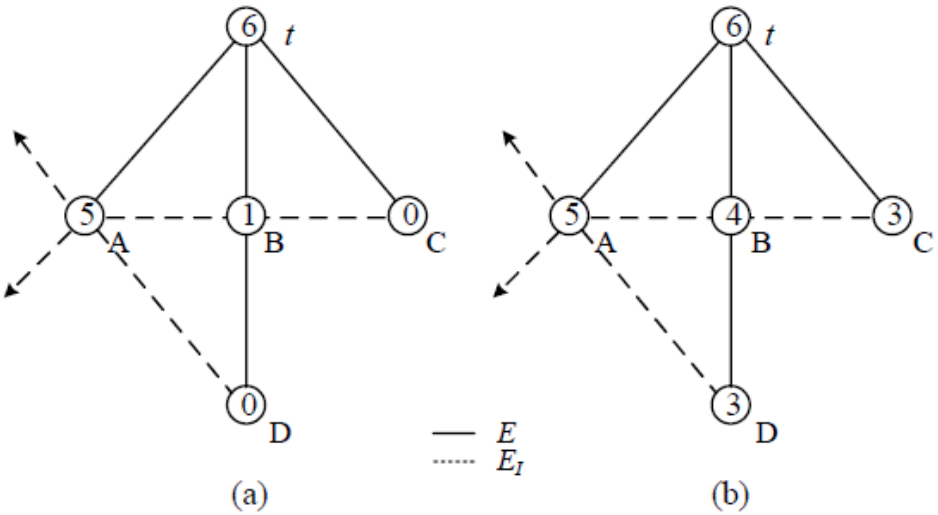


圖 7：嘗錯式時槽安排方法之(a)步驟 2 和(b)步驟 3

對於尋常之網路結構，本計畫亦提出一進中式的嘗錯式時槽安排解決方案 (Heuristic slot assignment solutions)，其基本精神為以最緊密之方式來安排節點之時槽。這一個演算法主要包含兩個步驟，假設 G 為網路上路由器們所組成的一個圖，k 為所有可用的時槽數量。

1. 由 G，首先建構一個寬度優先展延樹(Breadth first spanning tree)。
2. 接著以由下至上之方式(Bottom-up manner)瀏覽這顆樹上的成員。當拜訪到一個節點 v，首先以下列之方式指定一個暫存的時槽 $t(v)$給節點 v。

   i. 如果節點 v 是一個樹葉節點(leaf node)，則設定 $t(v)$為一個最小的非負整數 l，並且必須保證該 l 的數值不會與之前已經被拜訪過且與 v 為干擾鄰居節點的時槽相同。
   ii. 如果節點 v 是一個在樹中的節點(in-tree node)，則先設定一個變數 m 為 v 的所有小孩中最大的暫存時槽數值，也就是說 m=max{$t(child(v)$}，其中 $child(v)$為節點 v 的子節點之集合。接著我們指定 $t(v)$為一個最小的非負整數 $l > m$，並且必須保證該 l 的數值不會與之前已經被拜訪過且與 v 為干擾鄰居節點的時槽相同。

做完上述指定暫存時槽後，可以指定節點v的時槽$s(v)=t(v) \bmod k$。

3. 從Sink節點開始以top-down的方法來拜訪每個節點，當節點v被拜訪到時，我們便嘗試找尋一個新的時槽s’，使得(s(par(v)) – s’) mod k < (s(par(v)) – s(v)) mod k，若如可以找到此s’，則s(v) = s’。

在執行完這兩步驟後，網路上之路由器們即被指定到一個可盡量使得回報延遲降低之時槽安排方式。而圖 7(a)便為步驟 2 執行完後的一個例子，而圖 7(b)為步驟 3 中，節點 B、C、及 D 都能找到另一個時槽來降低各自的回報延遲時間的一個例子。

分散式時槽安排解決方案

本計畫亦針對尋常的網路結構提出一套分散式時槽安排方式，在此方法中，每一個節點會利用[22]中所提出之方法，來計算出其直接及間接干擾節點。而整個時槽安排方法由 Sink 發起，一開始 Sink 會將自己的時槽設定為 k – 1，之後便開始擴播 Beacon，而針對每一個收到 Beacon 之節點，便會依照下列步驟來決定自己的時槽：

1. 收到 Beacon 之節點 v 會傳送一 association request 封包至 Beacon 發起者。
2. 如果 v 無法成功 associate 上這個 Beacon 發起節點，則 v 會停止時槽安排演算法，而等待下一次收到 Beacon。
3. 如果 v 成功的 associate 上 Beacon 發起節點，則對於所有會和 v 產生直接和間間干擾之節點 u，v 會計算出一個最小的整個 m，使得(s(par(v)) – m) mod k ≠ s(u)，最後，v 會決定自己的時槽 s(v)為(s(par(v)) – m) mod k。
4. 在一段 t_{wait} 的時間內，v 會廣播自己所決定的時槽 s(v)，而其間如果發現有一鄰居節點 u 且(u, v) ∈ E_I，則如果下列的規則滿足的話，v 便會更改原本的時槽為一個新時槽，並且回到步驟 3。

   i. 於 GI 中，節點 u 之 degree 大於節點 v 之 degree。
   ii. 若 u, v 之 degree 相同，則節點 u 之深度比節點 v 之深度淺，也就是 u 比較靠近 Sink。
   iii. 若 u, v 之 degree 和深度皆相同，則 u 之 ID 比 v 還小

5. 在 t_{wait} 終止時，v 便可決定出自己的時槽，並開始廣播自己的 Beacon。

五、 結果與討論

在本計畫中，對於ZigBee網路環境中，針對convergecast我們定義了一個MDBS的問題，同時限制此問題之時槽排程安排上必須相容於ZigBee標準之規範。我們証明了此問題為一個困難的問題，並且於某些特殊的網路型態中，提出了最佳解法；於一般網路型態型，提出了集中式及分散式的時槽分配方法。

而在現實的網路中，亦有可能存在會有雙向資料流之應用存在，在[11]中，作者將我們的概念延伸為雙向 Beacon 排程問題，其目標為同時支援上傳(upstream, i.e., convergecast)以及下傳(downstream, i.e., broadcast)之資料流，作者修改了原本 IEEE 802.15.4b 中所定義之 Superframe 架構，並新增了一 Outgoing superframe 以及一 Incoming superframe，因此一個路由器可以分配到兩個時槽，一個時槽主要用以傳送上傳資料，另一時槽則主要用以傳送下傳資料。在[11]中，時槽安排方式之概念類似於我們所提出之方法，但由於需為每一路由器安排兩個時槽增加了演算法設計之複雜度，在[11]中作者亦觀察到一可以降低干擾鄰居數

量之方法，該方法可以使得時槽的安排能夠更加緊密，亦即能夠更降低上傳及下傳之延遲時間。

同時，在[11]中之作者亦觀察到可以更加減少干擾鄰居之方法，其概念為將網路上的路由器依照干擾的狀態以及其實體網路連結狀態做一分類，在安排時槽時即使某些節點互為鄰居節點，但是當他們使用相同的時槽有可能不會造成時槽的干擾，舉例來說：假設兩路由器節點互為鄰居，但是此兩路由器節點皆沒有小孩節點，則此兩路由器亦可以使用相同時槽，如此並不會造成任何 Beacon 碰撞之情形發生。

- 出席國際學術會議心得報告及發表之論文各一份

所出席之國際學術會議為「IEEE International Conference on Communications Conference」，其出國之報告書如附錄一

- 本計畫目前的研究成果為二篇論文如下：

附錄二：

Y.-C. Tseng and M.-S. Pan, "Quick Convergecast in ZigBee Beacon-Enabled Tree-Based Wireless Sensor Networks", Computer Communications, Vol. 31, No. 5, Mar. 2008, pp. 999-1011. (SCIE, EI)

附錄三：

Y.-C. Tseng and M.-S. Pan, “Quick Convergecast in ZigBee/IEEE 802.15.4 Tree-Based Wireless Sensor Networks”, ACM Int’l Workshop on Mobility Management and Wireless Access (ACM MobiWac), 2006. (selected as a candidate of the Best Paper Award)

- 計畫成果自評

第二年的主要工作目標有兩項：(1) ZigBee網路快速回報之研究方法及問題定義，(2) 針對所定義之問題提出適切的解決方案。而在本計畫中，針對 ZigBee快速回報問題，除了考慮了降低回報延遲及能源的消耗，並且符合了ZigBee標準的規範，因為本計畫定義了一個MDBS的問題。同時，我們提出了定理一來證明 MDBS為一個困難的問題。

此外，針對MDBS問題，針對特殊的網路型態：線狀網路及環狀網路，我們提出了一個多項式時間之最佳同，同時，針對一般的網路型態，我們也分別提出了集中式及分散式演算法來解之，因此本研究可利用我們所提出之方法來降低 ZigBee網路回報的時間。

六、 參考文獻

1. ZigBee specification version 2006, ZigBee document 064112, 2006.
2. H. Choi, J. Wang, and E. A. Hughes, “Scheduling for information gathering on sensor network,” ACM/Springer Wireless Networks, 2007, in press.
3. S. Gandham, Y. Zhang, and Q. Huang, “Distributed minimal time convergecast scheduling inwireless sensor networks,” In Proc. of IEEE Int’l Conference on Distributed Computing Systems (ICDCS), 2006.
4. B. Hohlt, L. Doherty, and E. Brewer, “Flexible power scheduling for sensor networks,” In Proc. of ACM/IEEE Int’l Conference on Information Processing in Sensor Networks (IPSN), 2004.
5. G. Lu, B. Krishnamachari, and C. S. Raghavendra, “An adaptive energy-efficient and low-latencyMAC for data gathering in wireless sensor networks,” In Proc. of IEEE Int’l Parallel and Distributed Processing Symposium (IPDPS), USA, 2004.
6. S. Upadhyayula, V. Annamalai, and S. K. S. Gupta, “A low-latency and energy-efficient algorithm for convergecast in wireless sensor networks,” In Proc. of IEEE Global Telecommunications Conference (Globecom), USA, 2003.
7. Y. Yu, B. Krishnamachari, and V. K. Prasanna, “Energy-latency tradeoffs for data gathering in wireless sensor networks,” In Proc. of IEEE INFOCOM, 2004.
8. 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), 2003.
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)(revision of IEEE Std 802.15.4-2003), 2006.
10. M. Yarvis, N. Kushalnagar, H. Singh, A. Rangarajan, Y. Liu, and S. Singh. Exploiting heterogeneity in sensor networks. In Proc. of IEEE INFOCOM, Miami, USA, 2005.
11. L.-H. Yen, Y. W. Law, and M. Palaniswami, "Risk-aware beacon scheduling for tree-based ZigBee/IEEE 802.15.4 wireless networks", in Proc. of International Wireless Internet Conference (WICON), 2008.

出國報告書

撰寫時間： 2010 年 6 月 3 日

姓　　名	梁家銘	單　　位	資訊工程系
連絡電話	03-5712121 ext.54747	出生年月日	71 年　4 月　25 日
職　　別	博士生		
出席國際會議名稱	IEEE International Conference on Communications Conference (ICC) 2010		
到達國家及地點	South Africa, Cape Town (南非, 開普敦)		
出國期間	自 2010 年 5 月 21 日 迄 2010 年 5 月 30 日		
內容	議程為5/23~5/27 Main Conference。本人參與該會議有以下兩項主要任務： 1. 發表論文，題目為：An Energy Efficient Sleep Scheduling Considering QoS Diversity for IEEE 802.16e Wireless Networks 2. 會議期間： ・聆聽最新無線網路研究成果及趨勢發展 ・嘗試挖掘具潛力的研究方向及題目著手研究 ・多方面與各國學者專家交流本論文內容 ・吸收各方給與的意見與建議 ・有機會的話，參與跨國際的合作與資訊交流 ・拓展研究的國際視野 ・精進自己聽說能力		

An Energy Efficient Sleep Scheduling Considering QoS Diversity for IEEE 802.16e Wireless Networks

Jen-Jee Chen, Jia-Ming Liang, and Yu-Chee Tseng
Department of Computer Science
National Chiao Tung University, Hsin-Chu 30010, Taiwan
E-mail: {chencz,jmliang,yctseng}@cs.nctu.edu.tw

***Abstract*—Power management is one of the most important issues in IEEE 802.16e wireless networks. In the standard, it defines three types of *power saving classes (PSCs)* for flows with different QoS characteristics. It allows a mobile device to turn off its wireless radio when all its PSCs are in sleep states. In this paper, we consider the scheduling of power saving classes of type II in an IEEE 802.16e network with a BS and multiple MSSs (mobile subscriber stations). Previous work proposes to enforce all MSSs to have the same sleep cycle, thus leading to higher energy cost for those MSSs with less strict delay bounds. We observe that if the sleep cycles of MSSs can be assigned according to their delay bounds, MSSs can significantly reduce their duty cycles. We propose an efficient *tank-filling algorithm*, which is standard-compliant and can allocate resources to MSSs according to their QoS characteristics with the least number of active frames. Simulation results verify that our algorithm incurs less power consumption and leads to higher bandwidth utilization than the previous schemes.**

***Index Terms*—IEEE 802.16e, power management, power saving class (PSC), quality of service (QoS), WiMAX, wireless network.**

I. Introduction

The IEEE 802.16e [1], [2] is a promising standard for providing broadband wireless access to *mobile subscriber stations (MSSs)* with high mobility. Like most other wireless mobile systems, how to conserve energy for battery-powered MSSs is a critical issue in IEEE 802.16e. In IEEE 802.16e, three types of PSCs (power saving classes) are defined. A PSC can be associated to one or more flows in an MSS. When a PSC is activated, it repeatedly switches between sleep and listening windows, where only during a listening window, can its member flows transmit/receive data. When all PSCs of an MSS are in their sleep windows, the MSS can turn off its radio transceiver to save energy.

The three types of PSC in IEEE 802.16e are reviewed below. In type I, the sizes of listening windows are fixed while the sizes of sleep windows grow exponentially when no data arrives. Once any traffic arrives, the PSC will be deactivated, until all queued traffics are delivered. So, PSCs of type I are more suitable for non-real-time traffic variable-rate (NRT-VR) and best-effort (BE) flows. In type II, the sizes of both listening and sleep windows are fixed. However, unlike type I, the arrival of traffics will not deactivate the PSC. This type II is more suitable for unsolicited grant service (UGS) and real-time traffic variable-rate (RT-VR) flows. In type III, it is only valid for one sleep window, after which the PSC is deactivated. This type is more suitable for multicast services and management operations. Among these three types, we are more interested in PSC of type II because one may dynamically adjust such PSCs' sleeping behaviors to maximize MSSs' energy efficiency.

In the literature, performance analyses for PSCs in an IEEE 802.16e network are conducted in [3]–[5]. For an MSS-BS pair, [6]–[8] focus on the design of PSCs of type I. How to adaptively adjust the initial sleep window is addressed in [6]. Assuming that the distribution of the response packet arrival time is known, [7] proposes a decision algorithm such that the MSS can stay asleep until response packets are expected to arrive. In [8], how to adjust the minimum and the maximum sleep windows is discussed. For type II, assuming that PSCs are already given, a Maximum Unavailability Interval scheme is proposed in [9] for selecting the optimal start frame for each PSC to maximize its unavailable duration. References [10], [11] propose to apply one single PSC to accommodate all real-time flows in an MSS; parameters of the PSC are selected to meet the flow with the strictest bandwidth and packet delay bound. Considering multiple MSSs under the same BS, [12] proposes a *Longest-Virtual-Burst-First (LVBF)* scheme, which always selects a primary MSS in the burst mode to serve and only gives the necessary bandwidth to the other MSSs to meet their requirements. However, it does not take delay constraints of flows into consideration. Reference [13] proposes to serve each MSS by a PSC of type II, but all of them share the same sleep cycle. This results in PSCs without overlapping in their active frames. However, since the common sleep cycle is bounded by the strictest delay bound of all MSSs, this way causes some MSSs to have too many active frames.

In this work, we focus on PSCs of type II. Given multiple MSSs under a BS, we consider the arrangement of PSCs for these MSSs according to their delay bounds and bandwidth requirements. This involves not only the selection of each PSC's parameters, but also the selection of their listening windows to reduce the overall active frames of MSSs. We propose a tank-filling algorithm, which regards the resources of the BS as a sequence of periodical tanks, each being able to provide a fixed amount of bandwidth. The result outperforms that of [13] because we relax the constraint that all PSCs should share the same sleeping cycle. Simulation results are provided to verify these claims.

The rest of this paper is organized as follows. Section II

gives some motivations and formally defines the problem. Our tank-filling algorithm is presented in Section III. Simulation results are shown in Section IV. Section V concludes this paper.

II. Motivation and Problem Definition

In this section, we first motivate our work by discussing previous work [13]. Then we formally define our problem. In [13], assuming that there are multiple MSSs, each to be served by a PSC of type II, it enforces each MSS to adopt a PSC of the same sleep cycle length. The sleep cycle is selected to meet the MSS with the tightest delay bound. While the solution is easy to implement, this is too restricted and may incur too many active frames to some MSSs. Fig. 1 shows an example with two MSSs M_1 and M_2, which have data arrival rates of $\tau_1 = 0.2\Omega$/frame and $\tau_2 = 0.075\Omega$/frame and delay bounds of $D_1 = 4$ (frames) and $D_2 = 12$ (frames), respectively, where Ω is the capacity of a frame. Fig. 1(a) shows the schedule computed by [13]. Since $\min(D_1, D_2) = 4$, in every four frames, M_1 and M_2 will be active for one frame and be allocated of bandwidths $\tau_1 \times 4 = 0.8\Omega$ and $\tau_2 \times 4 = 0.3\Omega$, respectively, per frame. Also, their active frames are shifted to avoid overlapping. As Fig. 1(b) shows, by assigning each MSS a sleep cycle adaptive to its delay bound, M_1 and M_2 can have sleep cycles of 4 and 12 frames, respectively, where in each active frame, they receive $\tau_1 \times 4 = 0.8\Omega$ and $\tau_2 \times 12 = 0.9\Omega$ of bandwidths. Still we can manage to incur no overlapping among their active frames, so M_2's duty cycle is significantly reduced.

The above observation motivates us to study a power management problem as follows. We consider a BS serving n MSSs M_i, $i = 1..n$. Each M_i has a data arrival rate of τ_i bits/frame and each data arrival has a delay bound of D_i frames. Assuming the available bandwidth per frame is Ω bits and $\sum_{i=1..n} \tau_i \leq \Omega$, the goal is to assign each M_i a PSC of type II with a sleep cycle of T_i^S, a listening window of T_i^L, and an offset of T_i^O, such that $T_i^S \leq D_i$ and the total number of active frames for all MSSs is minimized. Also, there is implicit requirement that whenever a listening window of an MSS arrives, the BS should be able to serve all its backlog data that would be overdue otherwise.

III. The Proposed Tank-Filling Scheme

In an IEEE 802.16e wireless network, the BS is responsible for scheduling the sleep frames of the MSSs associated with it. Initially, each $M_i, i = 1..n$, will send a request to the BS containing its D_i. We propose a *tank-filling (TF)* algorithm for the BS to determine the following parameters for each M_i: (1) (T_i^S, T_i^L, T_i^O) and (2) amount of bandwidth $B_{i,j}$ allocated to M_i in the j-th active frame in each listening windows, $j = 1..T_i^L$ (noth that $B_{i,j}$ is a real number between 0 and 1). Then these parameters are sent to each M_i. Then these MSSs will behave accordingly.

Our TF algorithm considers the resources of the BS as a sequence of repetitive *tanks*, each being able to hold Ω amount of water per frame. It maintains an important property that T_i^S of each M_i is an integer multiple of its previous T_{i-1}^S for each $i = 2..n$. So, we call T_1^S as the basic cycle, or simply T_{basic}, of the network. Intuitively, this property helps make MSSs' sleeping behaviors regular and increase the overlapping of their listening windows. Assuming that $T_1^S, T_2^S, ..., T_n^S$, are known (recall that $T_1^S = T_{basic}$), the resources controlled by the BS are represented by an array $R[1 : \frac{T_n^S}{T_{basic}}, 1 : T_{basic}]$, where each $R[k, \ell]$, $k = 1..\frac{T_n^S}{T_{basic}}$ and $\ell = 1..T_{basic}$, is to record the amount of remaining resource in the ℓ-th frame of the k-th basic cycle. Initially, $R[k, \ell] = \Omega$ is regarded as an empty tank. Gradually, we will fill in more data to each tank. Below, we present our TF algorithm in three steps. (A) Assuming that T_{basic} is known, we will choose T_i^S of each $M_i, i = 1..n$. (B) Determine T_i^L, T_i^O, and $B_{i,j}, j = 1..T_i^L$, of each $M_i, i = 1..n$. (C) In the end, we will come back and search for the most energy-efficient basic cycle T_{basic}.

A. Determining T_i^S of M_i

To decide T_i^S, we first sort MSSs by their delay bounds. Without loss of generality, let $D_1 \leq D_2 \leq \cdots \leq D_n$. Supposing that $T_1^S = T_{basic}$ is known and $T_1^S \leq D_1$, we set T_i^S, $i = 2..n$, as follows:

$$T_i^S = T_{i-1}^S \times \left\lfloor \frac{D_i}{T_{i-1}^S} \right\rfloor. \quad (1)$$

It is not hard to see that Eq. (1) implies $T_i^S \leq T_{i-1}^S \times \frac{D_i}{T_{i-1}^S} = D_i$. So, T_i^S guarantees the delay bound of M_i. In fact, Eq. (1) also ensures that T_i^S is an integer multiple of T_{i-1}^S.

Lemma 3.1: Eq. (1) guarantees that each T_i^S is an integer multiple of T_{i-1}^S, $i = 2..n$, and $T_i^S \leq D_i$, $i = 1..n$.

B. Scheduling T_i^L, T_i^O, and $B_{i,j}$ of M_i

Recall the array $R[\cdot, \cdot]$, which represents the resource of the BS. We will sequentially allocate resources for M_i, $i = 1..n$, by updating $R[\cdot, \cdot]$. Our algorithm is called 'tank-filling' when M_i is being considered, we will test every 'starting' tank in R by sequentially filling its data to the empty part of that tank and continuing to next tank in R, until all the data is drained. Note that here R is regarded as T_n^S tanks is a row-major way. Among these testing starting tank, the one resulting in the least active frames to M_i is selected. The detail procedure for M_i is follows, where i starts from 1 and end at n:

a) Calculate the bandwidth requirement of M_i per T_i^S by $\gamma_i = \tau_i \times T_i^S$.

b) When M_i enters the step, $R[\cdot, \cdot]$, if regarded in a row-major order, has a period of T_{i-1}^S (see that note at the end of step (d)). So we let $j = 1..T_{i-1}^S$ as the potential indices of the starting tanks and run the following steps for each j.

   i) Starting from the j-th tank in $R[\cdot, \cdot]$, we fill in the bandwidth requirement γ_i of M_i into the empty part of the tank. If there is sufficient space for γ_i, we are done; otherwise we fill the j-th tank up proceed to the $(j+1)$-th tank. We continue the process until all γ_i is satisfied.

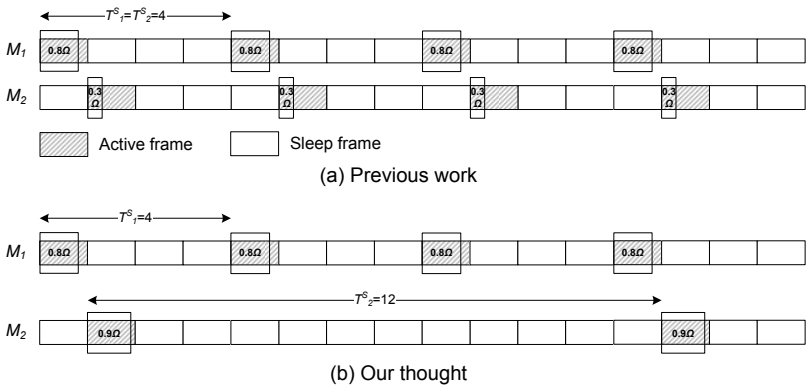


Fig. 1. Sleep scheduling for two MSSs M_1 and M_2 using (a) a common sleep cycle and (b) different sleep cycles.

ii) Let $f(j)$ be the number of tanks that have been used to serve M_i's data. This is regarded as the cost function to start with the j-th tank.

c) Among all possible js in step (b), let j^* be the index which induces the smallest cost $f(\cdot)$. We then place M_i's demand starting from the j^*-th tank according to above procedure. Note that in case of a tie, we will give priority to the one which leaves the least remaining resource in the last frame where M_i's demand is placed.

d) Then we set $T_i^L = f(j^*)$ and $T_i^O = j^*$. Also, we set $B_{i,j}$ to the bandwidth allocated to M_i in the j-th tank, $j = 1..T_n^S$, and subtract $B_{i,j}$ from the corresponding entry in R (note that the allocation in step (b) should be repeated $\frac{T_n^S}{T_i^S}$ for array R, so R has a period of T_i^S at the end of this step).

As noted in step (d), after the allocation of M_i, array R has a period of T_i^S. This would simplify our next allocation for M_{i+1} since T_{i+1}^S is an integer multiple of T_i^S.

Example 1: Fig. 2 shows an example of step B. There are 5 MSSs M_1, M_2, M_3, M_4, and M_5 with sleeping cycles of $T_1^S = T_{basic}$, $T_2^S = 2T_{basic}$, $T_3^S = 2T_{basic}$, $T_4^S = 2T_{basic}$, and $T_5^S = 4T_{basic}$ and required resources per cycle of $\gamma_1 = 0.5\Omega$, $\gamma_2 = 1.25\Omega$, $\gamma_3 = 0.4\Omega$, $\gamma_4 = 0.4\Omega$, and $\gamma_5 = 2.5\Omega$, respectively, where $T_{basic} = 3$ frames. Initially, $R[k, \ell] = \Omega$ for $k = 1..4$ and $\ell = 1..3$. Then, each M_i is scheduled as follows. For M_1, we can only set $j^* = 1$. Then, the BS reserves $\gamma_1 = 0.5\Omega$ resource for M_1 in every basic cycle as shown in Fig. 2(1) and set $B_{1,1} = B_{1,4} = B_{1,7} = B_{1,10} = 0.5\Omega$, $T_1^O = j^* = 1$, and $T_1^L = f(1) = 1$; so $R[1,1] = R[2,1] = R[3,1] = R[4,1] = 0.5\Omega$ and $R[k, \ell] = \Omega$ for $k = 1..4$ and $\ell = 2, 3$. For M_2, its j^* can be 1 or 2 or 3 and allocating γ_2 by starting from any of the two basic cycles are the same. Since $\lceil 0.5\Omega + 1.25\Omega \rceil - (0.5\Omega + 1.25\Omega) = 0.25\Omega < \lceil 1.25\Omega \rceil - 1.25\Omega = 0.75\Omega$, setting $j^* = 1$ and 3 would create the least number of active frames and leave the least remaining resource in the last frame. So we select $j^* = 1$. After the allocation, shown as Fig. 2(2), we set $B_{2,1} = B_{2,7} = 0.5\Omega$, $B_{2,2} = B_{2,8} = 0.75\Omega$, $T_2^O = j^* = 1$, and $T_2^L = f(1) = 2$ and update $R[1,1] = R[3,1] = 0$ and $R[1,2] = R[3,2] = 0.25\Omega$. For M_3, choosing $j^* = 3$, 4, 5, and 6 would create the same and least number of active frames (i.e., $f(3) = f(4) = f(5) = f(6) = 1 < f(2) = 2 < \cdots$), but setting $j^* = 4$ would leave less remaining resource in the last allocated frame (i.e., 0.1Ω). So we set $j^* = 4$ and update $B_{3,4} = B_{3,10} = 0.4\Omega$, $T_3^O = j^* = 4$, and $T_3^L = f(4) = 1$, as shown in Fig. 2(3). Then update $R[2,1] = R[4,1] = 0.1\Omega$. For M_4, setting $j^* = 3$, 5, and 6 would create the same and least number of active frames (i.e., $f(3) = f(5) = f(6) = 1 < f(2) = f(4) = 2 < \cdots$) and leave the same remaining resource in the last frame. So we choose $j^* = 3$ and set $B_{4,3} = B_{4,9} = 0.4\Omega$, $T_4^O = j^* = 3$, and $T_4^L = f(3) = 1$, as shown in Fig. 2(4). Then we update $R[1,3] = R[3,3] = 0.6\Omega$. For M_5, choosing $j^* = 3$ would add the least number of active frames (i.e., $f(3) = 4 < f(2) = f(5) = 5 < f(4) = 6 < \cdots$). So we choose $j^* = 3$ and set $B_{5,3} = 0.6\Omega$, $B_{5,4} = 0.1\Omega$, $B_{5,5} = \Omega$, $B_{5,6} = 0.8\Omega$, $T_5^O = j^* = 3$, and $T_5^L = f(3) = 4$, as shown in Fig. 2(5). Then update $R[1,3] = R[2,1] = R[2,2] = 0$ and $R[2,3] = 0.2\Omega$.

C. Selecting T_{basic}

Clearly, different values of T_1^S will lead to different duty cycles for MSSs. Here we adopt an exhausted search by setting $T_1^S = 1..D_1$ and trying to find the sum of the total number of active frames of all MSSs over a windows of T_n^S frames. Then $T_1^{S^*}$ leading to the least number of active frames is chosen as T_{basic}.

IV. Performance Evaluation

We have developed a simulator by C++ to verify the effectiveness of our PMSS scheme. Unless otherwise stated, the following assumptions are made in our simulation. The number of MSSs is ranged from 5 to 45. Each MSS M_i has a data rate τ_i of $1000 \sim 3000$ bits/frame and delay bound D_i of $10 \sim 200$ frames, where 1000 is the minimum data rate, 3000 is the maximum data rate, 10 is the minimum delay bound, and 200 is the maximum delay bound of the MSS. The available bandwidth per frame of the system is $\Omega = 80000$ bits (16Mbps) and the length of an OFDM/OFDMA frame is set

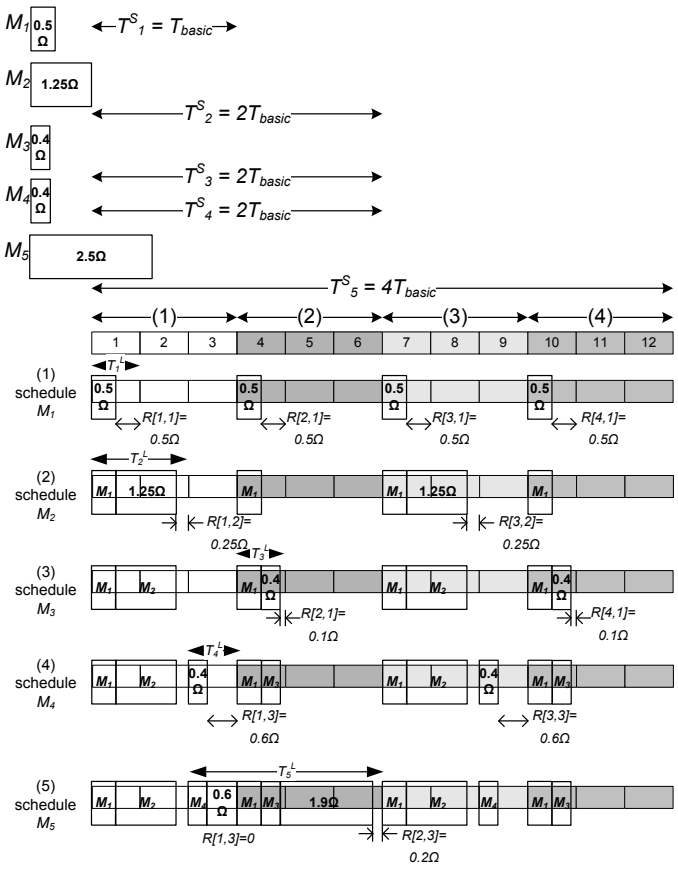


Fig. 2. Example of scheduling $B_{i,j}$, T_i^L, and T_i^O for five MSSs M_1, M_2, M_3, M_4, and M_5.

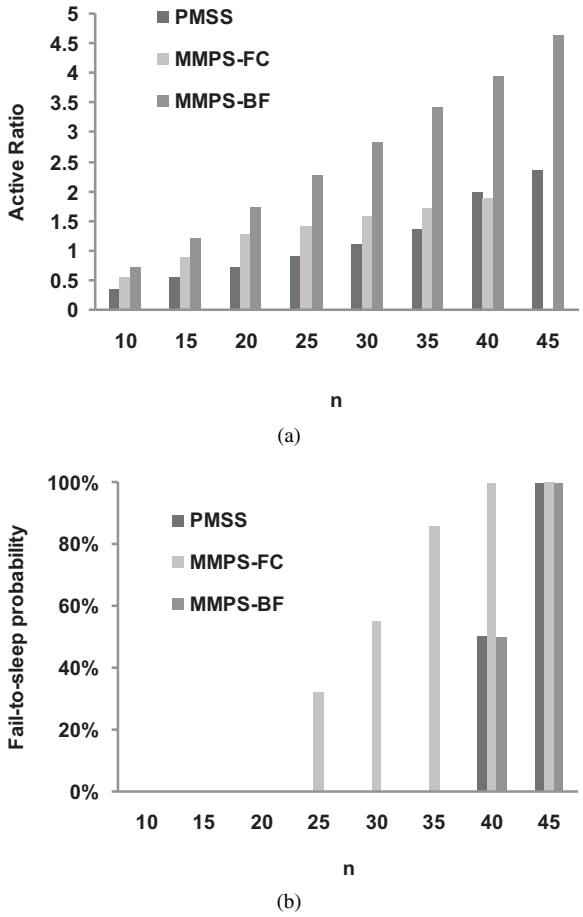


Fig. 3. Effects of number of MSSs on (a) active ratio and (b) fail-to-sleep probability.

to 5 ms [14]. We consider two performance metrics: (i) *active ratio*: the ratio of active frames for the system and (ii) *fail-to-sleep probability*: the ratio of failure to schedule MSSs' sleep. We will compare our PMSS against the MMPS-FC (Multiple MSSs Power-saving Scheduler with Fragment Collection) and MMPS-BF (Multiple MSSs Power-saving Scheduler with Boundary Free) schemes in [13].

A. Effects of n

In this experiment, we study the effect of n on the active ratio and fail-to-sleep probability. Fig. 3(a) shows the active ratio decreases when n increases. Our PMSS almost always performs the best in all three schemes, except at $n = 40$, MMPS-FC performs better than our PMSS. However, when $n = 40$, the fail-to-sleep probability of MMPS-FC is almost 100% (Fig. 3(b)). Fig. 3(b) shows the fail-to-sleep probability increases when n increases. MMPS-BF and our PMSS schemes perform the same and the best in the fail-to-sleep probability. The fail-to-sleep probabilities of the two schemes is zero when $n < 40$. For MMPS-FC, it can 100% successfully schedule MSSs into sleep when $n < 25$.

B. Effects of Maximum Delay Bound

Then, we investigate the effect of maximum delay bound on the active ratio by fixing $n = 20$. Fig. 4 shows the active ratio decreases when the maximum delay bound increases. Our PMSS performs the best in all three schemes. For the three schemes, our PMSS benefits the most when the maximum delay bound is increased from 50 ms to 3000 ms (70%); for MMPS-FC and MMPS-BF, the improvement is 52% and 47%, respectively. This is because our scheme can more accurately capture the traffic characteristics of MSSs.

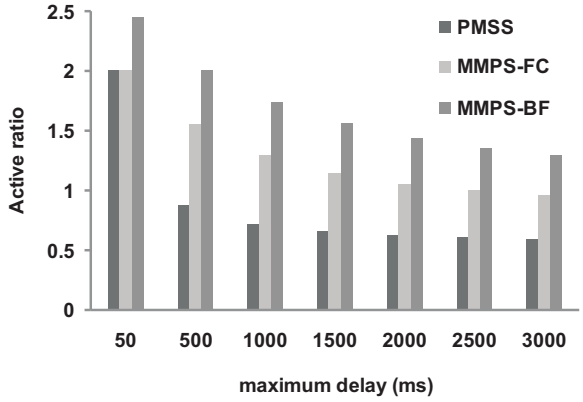
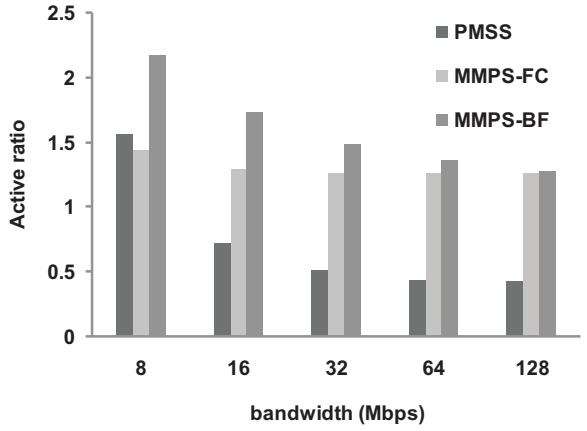
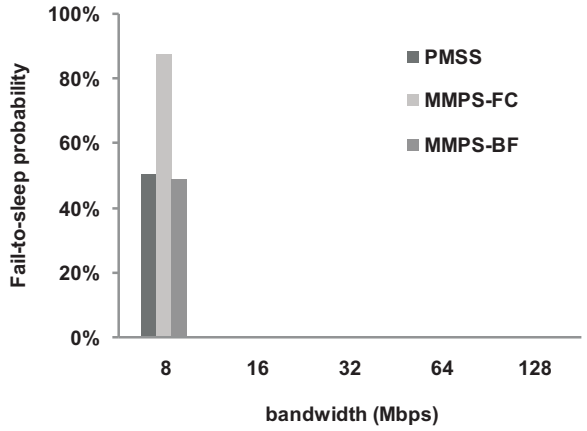


Fig. 4. Effects of maximum delay on active ratio.



(a)



(b)

Fig. 5. Effects of system bandwidth on (a) active ratio and (b) fail-to-sleep probability.

C. Effects of System Bandwidth

In this experiment, we investigate the effect of system bandwidth on the active ratio and fail-to-sleep probability by fixing $n = 20$. Fig. 5(a) shows the active ratio decreases when system bandwidth increases. Our PMSS outperforms other two schemes except when the system bandwidth is 8Mbps. When the system bandwidth is 8Mbps, MMPS-FC performs the best but its fail-to-sleep probability is much higher (88%) than other two schemes (about 50%). For the three schemes, our PMSS benefits the most when system bandwidth is increased from 8Mbps to 128Mbps (73%); for MMPS-FC and MMPS-BF, the improvement is 13% and 42%, respectively.

V. Conclusions

In this paper, we propose a per-MSS sleep scheduling scheme for multiple MSSs in IEEE 802.16e wireless networks such that the overall power consumption of the system is minimized while the QoS of each MSS can be guaranteed. Compared to the previous work, our approach assigns and schedules type II PSCs for each MSS by considering each of their QoS characteristics such that the sleep scheduling can more accurately capture each MSS's QoS requirement. This leads to each MSS can sleep more and the total power consumption of the system is significantly reduced. Also, the proposed scheme is easy to implement and compatible to the standard.

Acknowledgements

Y.-C. Tseng's research is co-sponsored by MoE ATU Plan, by NSC grants 96-2218-E-009-004, 97-3114-E-009-001, 97-2221-E-009-142-MY3, and 98-2219-E-009-005, by MOEA 98-EC-17-A-02-S2-0048, and 98-EC-17-A-19-S2-0052, and by ITRI, Taiwan.

References

[1] IEEE Std 802.16e-2005, "IEEE Standard for Local and metropolitan area networks Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems Amendment 2: Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands and Corrigendum 1," Feb. 2006.

[2] IEEE Std 802.16-2009, "IEEE Standard for Local and metropolitan area networks Part 16: Air Interface for Broadband Wireless Access Systems," May 2009.

[3] Y. Xiao, "Energy saving mechanism in the IEEE 802.16e wireless MAN," *IEEE Communications Letters*, vol. 9, no. 7, pp. 595–597, 2005.

[4] Y. Zhang and M. Fujise, "Energy management in the IEEE 802.16e MAC," *IEEE Communications Letters*, vol. 10, no. 4, pp. 311–313, 2006.

[5] Y. Zhang, "Performance Modeling of Energy Management Mechanism in IEEE 802.16e Mobile WiMAX," in *Proc. of IEEE Wireless Communications and Networking Conference*, 2007, pp. 3205–3209.

[6] J. Xiao and S. Zou and B. Ren and S. Cheng, "An Enhanced Energy Saving Mechanism in IEEE 802.16e," in *Proc. of IEEE GLOBECOM*, 2006.

[7] J.-R. Lee and D.-H. Cho, "Performance Evaluation of Energy-Saving Mechanism Based on Probabilistic Sleep Interval Decision Algorithm in IEEE 802.16e," *IEEE Transactions on Vehicular Technology*, vol. 56, no. 4, pp. 1773–1780, July 2007.

[8] M.-G. Kim and J.-Y. Choi and M. Kang, "Adaptive power saving mechanism considering the request period of each initiation of awakening in the IEEE 802.16e system," *IEEE Communications Letters*, vol. 12, no. 2, pp. 106–108, 2008.

[9] T.-C. Chen, J.-C. Chen, and Y.-Y. Chen, "Maximizing Unavailability Interval for Energy Saving in IEEE 802.16e Wireless MANs," *IEEE Transactions on Mobile Computing*, vol. 8, no. 4, pp. 475–487, Apr. 2009.

[10] S.-L. Tsao and Y.-L. Chen, "Energy-efficient packet scheduling algorithms for real-time communications in a mobile WiMAX system," *Computer Communications*, vol. 31, no. 10, pp. 2350–2359, June 2008.

[11] H.-L. Tseng, Y.-P. Hsu, C.-H. Hsu, P.-H. Tseng, and K.-T. Feng, "A Maximal Power-Conserving Scheduling Algorithm for Broadband Wireless Networks," in *Proc. of IEEE WCNC*, 2008, pp. 1877–1882.

[12] J. Shi, G. Fang, Y. Sun, J. Zhou, Z. Li, and E. Dutkiewicz, "Improving Mobile Station Energy Efficiency in IEEE 802.16e WMAN by Burst Scheduling," in *Proc. of IEEE GLOBECOM*, 2006.

[13] S.-C. Huang, R.-H. Jan, and C. Chen, "Energy Efficient Scheduling with QoS Guarantee for IEEE 802.16e Broadband Wireless Access Networks," in *Proc. of the International conference on Wireless Communications and Mobile Computing (IWCMC)*, pp. 547–552.

[14] H. S. Kim and S. Yang, "Tiny MAP: an efficient MAP in IEEE 802.16/WiMAX broadband wireless access systems," *Computer Communications*, vol. 30, no. 9, pp. 2122–2128, 2007.

附錄二

Quick Convergecast in ZigBee Beacon-Enabled Tree-Based Wireless Sensor Networks

Y.-C. Tseng and M.-S. Pan

Computer Communications

Quick Convergecast in ZigBee Beacon-Enabled Tree-Based Wireless Sensor Networks

Meng-Shiuan Pan and Yu-Chee Tseng
Department of Computer Science
National Chiao-Tung University
Hsin-Chu, 30010, Taiwan
Email: {mspan, yctseng}@cs.nctu.edu.tw

Abstract

Convergecast is a fundamental operation in wireless sensor networks. Existing convergecast solutions have focused on reducing latency and energy consumption. However, a good design should be compliant to standards, in addition to considering these factors. Based on this observation, this paper defines a *minimum delay beacon scheduling problem* for quick convergecast in ZigBee tree-based wireless sensor networks and proves that this problem is NP-complete. Our formulation is compliant with the low-power design of IEEE 802.15.4. We then propose optimal solutions for special cases and heuristic algorithms for general cases. Simulation results show that the proposed algorithms can indeed achieve quick convergecast.

Keywords: convergecast, graph theory, IEEE 802.15.4, scheduling, wireless sensor network, ZigBee.

1 Introduction

The rapid progress of wireless communication and embedded micro-sensing MEMS technologies has made *wireless sensor networks (WSNs)* possible. A WSN consists of many inexpensive wireless sensors capable of collecting, storing, processing environmental information, and communicating with neighboring nodes. Applications of WSNs include wildlife monitoring [3, 4], object tracking [16, 18], and dynamic path finding [15, 19].

Recently, several WSN platforms have been developed, such as MICA [6] and Dust Network [2]. For interoperability among different systems, standards such as ZigBee [24]

have been developed. In the ZigBee protocol stack, physical and MAC layer protocols are adopted from the IEEE 802.15.4 standard [13]. ZigBee solves interoperability issues from the physical layer to the application layer.

ZigBee supports three kinds of networks, namely *star*, *tree*, and *mesh* networks. A *ZigBee coordinator* is responsible for initializing, maintaining, and controlling the network. A star network has a coordinator with devices directly connecting to the coordinator. For tree and mesh networks, devices can communicate with each other in a multihop fashion. The network is formed by one ZigBee coordinator and multiple *ZigBee routers*. A device can join a network as an *end devices* by the associating with the coordinator or a router. In a tree network, the coordinator and routers can announce beacons. However, in a mesh network, regular beacons are not allowed. Beacons are an important mechanism to support power management. Therefore, the tree topology is preferred, especially when energy saving is a desirable feature. To support ZigBee beacon-enabled tree networks, the IEEE 802.15 WPAN Task Group 4 further defines a revision of the IEEE 802.15.4 [14] specification in 2006. One of the major changes is structure of superframes to support power management. On the contrary, to our understanding, power management is still impossible for mesh-based ZigBee networks in the current specification. Therefore, we will focus on tree-based, beacon-enabled ZigBee networks in this work.

Considering that data gathering is a major application of WSNs, *convergecast* has been investigated in several works [8, 9, 11, 17, 20, 23]. With the goals of low latency and low energy consumption, reference [20] shows how to connect sensors as a balanced reporting tree and how to assign CDMA codes to sensors to diminish interference among sensors, thus achieving energy efficiency. The work [23] aims to minimize the overall energy consumption under the constraint that sensed data should be reported within specified time. Dynamic programming algorithms are proposed by assuming that sensors can receive multiple packets at the same time. As can be seen, both [20] and [23] are based on quite strong assumptions on communication capability of sensor nodes and they do not fit into the ZigBee specification. In [17], the authors propose an energy-efficient and low-latency MAC, called *DMAC*. Sensors are connected by a tree and stay in sleep state for most of the time. When sensors change

to active state, they are first set to the receive mode and then to the transmit mode. *DMAC* achieves low-latency by staggering wake-up schedules of sensors at the time instant when their children switch to the transmit mode. Similar to [17], reference [11] arranges wake-up schedule of sensors by taking traffic loads into account. Each parent periodically broadcasts an advertisement containing a set of empty slots. Children nodes request empty slots according to their demands. In [9], the authors propose a distributed convergecast scheduling algorithm. The basic concept is to connect nodes by a spanning tree. Then the algorithm reduces the tree to multiple lines. For each line, the algorithm schedules nodes' transmission times in a bottom-up manner. Reference [8] presents a centralized solution to convergecast. The algorithm divides nodes into many segments such that the transmission of a node in a segment does not cause interference to other transmissions in the same segment. The aim is to increase the degree of parallel transmissions to decrease latencies. Although these results [8, 9, 11, 17] are designed for quick convergecast, the solutions are not compliant to the ZigBee standard for the following two reasons. Firstly, in these works, nodes' wake/sleep times are dynamically changed according to their schedules. However, in a ZigBee beacon-enabled tree network, nodes' wake/sleep times must be fixed in the way that each router wakes up twice in each cycle to receive its children's packets and to transmit packets to its parent, respectively. The coordinator (resp., an end device) wakes up once to receive its children's packets (resp., to transmit packets to its parent). Secondly, the scheduling of [8, 9, 11, 17] is transmission-based, while ours are receiving-based. The implication is that the former may cause a router to be active multiple times per cycle. This is incompatible with the ZigBee specification.

This paper aims at designing quick convergecast solutions for ZigBee tree-based, beacon-enabled WSNs. This work is motivated by the following observations. First, we see that most related works are not compliant to the ZigBee standard. Second, we believe that tree-based topology is more suitable if power management is a main concern in WSNs. The network scenario is shown in Fig. 1. The network contains one *sink* (ZigBee coordinator), some ZigBee routers, and some ZigBee end devices. Each ZigBee router is responsible for collecting sensed data from end devices associated with it and relaying incoming data

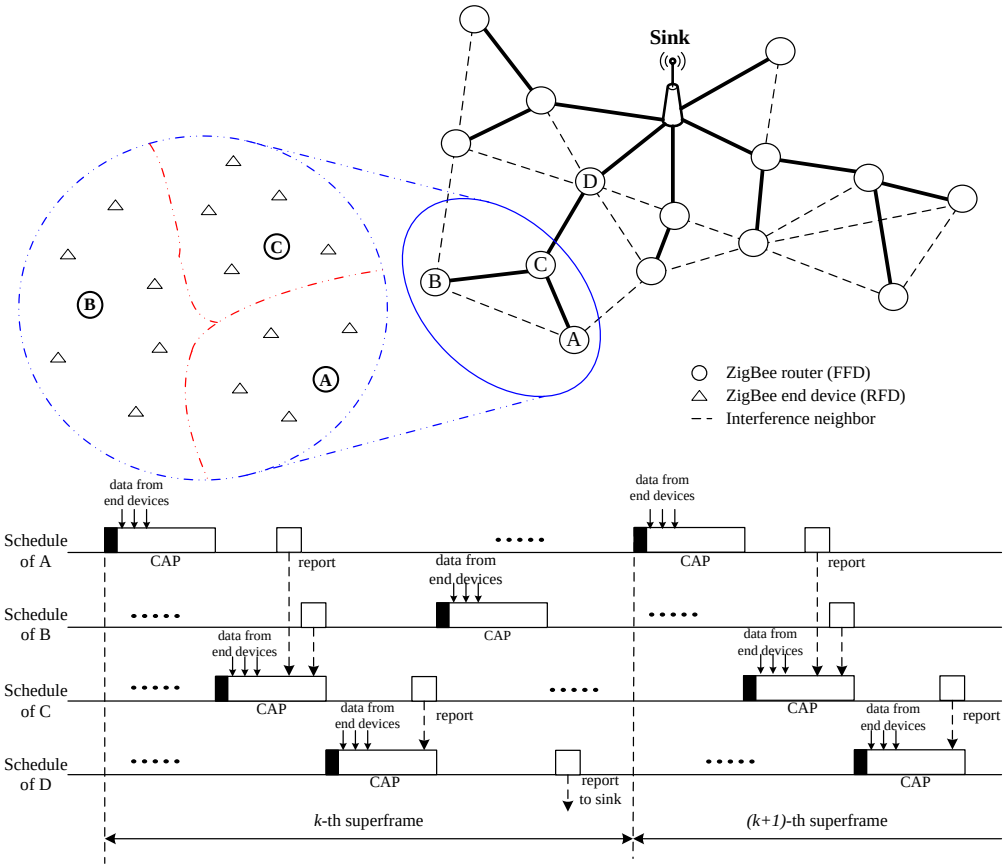


Figure 1: An example of convergecast in a ZigBee tree-based network.

to the sink. According to specifications, a ZigBee router can announce a beacon to start a superframe. Each superframe consists of an *active portion* followed by an *inactive portion*. On receiving its parent router's beacon, an end device has to wake up for an active portion to sense the environment and communicate with its coordinator. However, to avoid collision with its neighbors, a router should shift its active portion by a certain amount. Fig. 1 shows a possible allocation of active portions for routers A, B, C, and D. The collected sensory data of A in the k-th superframe can be sent to C in the same superframe. However, because the active portion of B in the k-th superframe appears after that of C, the collected data of B in the k-th superframe can only be relayed to C in the $(k+1)$-th superframe. The report delay from B to C is almost the length of one superframe. The delay can be eliminated if the active portion of B in the k-th superframe appears before that of C. The delay is not negligible because of the low duty cycle design of IEEE 802.15.4. For example, in 2.4 GHz PHY, with 1.56% duty cycle, a superframe can be as long as 251.658 seconds (with an active portion of 3.93 seconds). Clearly, for large-scale WSNs, the convergecast latency could be significant if the problem is not carefully addressed. The quick convergecast problem is to schedule the beacons of routers to minimize the convergecast latency. We prove that this problem is NP-complete by reducing the 3-CNF-SAT problem to it. We show two special cases of this problem where optimal solutions can be found in polynomial time and propose two heuristic algorithms for general cases. To the best of our knowledge, this is the first result that provides convergecast solutions in ZigBee beacon-enabled tree networks.

The rest of this paper is organized as follows. Section 2 briefly introduces IEEE 802.15.4 and ZigBee. The quick convergecast problem is formally defined in Section 3. Section 4 presents our scheduling solutions. Simulation results are given in Section 5. Finally, Section 6 concludes this paper.

2 Overview of IEEE 802.15.4 and ZigBee Standards

IEEE 802.15.4 [13] specifies the physical and data link protocols for *low-rate wireless personal area networks (LR-WPAN)*. In the physical layer, there are three frequency bands with 27 radio channels. Channel 0 ranges from 868.0 MHz to 868.6 MHz, which provides a data

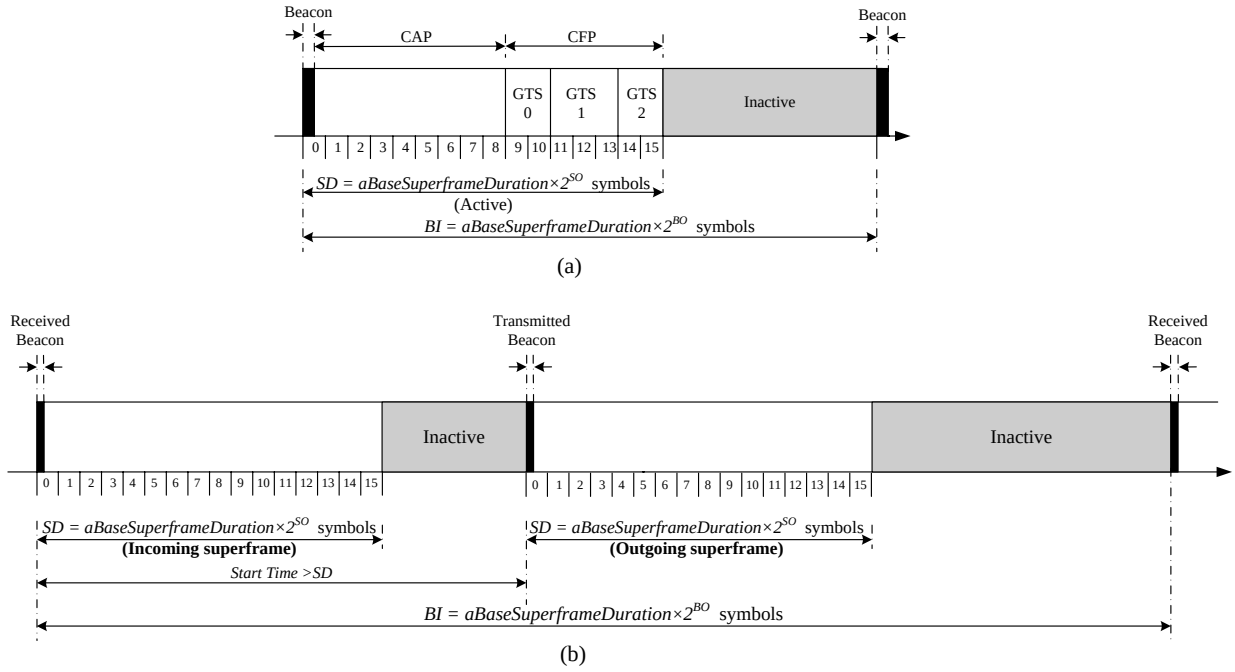


Figure 2: IEEE 802.15.4 superframe structure.

rate of 20 kbps. Channels 1 to 10 work from 902.0 MHz to 928.0 MHz and each channel provides a data rate of 40 kbps. Channels 11 to 26 are located from 2.4 GHz to 2.4835 GHz, each with a data rate of 250 kbps.

IEEE 802.15.4 devices are expected to have limited power, but need to operate for a longer period of time. Therefore, energy conservation is a critical issue. Devices are classified as *full function devices* (FFDs) and *reduced function devices* (RFDs). IEEE 802.15.4 supports star and peer-to-peer topologies. In each PAN, one device is designated as the *coordinator*, which is responsible for maintaining the network. A FFD has the capability of serving as a coordinator or associating with an existing coordinator/router and becoming a router. A RFD can only associate with a coordinator/router and can not have children.

The ZigBee coordinator defines the superframe structure of a ZigBee network. As shown in Fig. 2(a), the structure of superframes is controlled by two parameters: *beacon order* (BO) and *superframe order* (SO), which decide the lengths of a superframe and its active potion, respectively. For a beacon-enabled network, the setting of BO and SO should satisfy the relationship $0 \leq SO \leq BO \leq 14$. (A non-beacon-enabled network should set $BO =$

$SO = 15$ to indicate that superframes do not exist.) Each active portion consists of 16 equal-length slots, which can be further partitioned into a *contention access period* (CAP) and a *contention free period* (CFP). The CAP may contain the first i slots, and the CFP contains the rest of the $16-i$ slots, where $1 \leq i \leq 16$. Slotted CSMA/CA is used in CAP. FFDs which require fixed transmission rates can ask for *guarantee time slots (GTSs)* from the coordinator. A CFP can support multiple GTSs, and each GTS may contain multiple slots. Note that only the coordinator can allocate GTSs. After the active portion, devices can go to sleep to save energy.

In a beacon-enabled star network, a device only needs to be active for $2^{-(BO-SO)}$ portion of the time. Changing the value of $(BO-SO)$ allows us to adjust the on-duty time of devices. However, for a beacon-enabled tree network, routers have to choose different times to start their active portions to avoid collision. Once the value of $(BO-SO)$ is decided, each router can choose from 2^{BO-SO} slots as its active portion. In the revised version of IEEE 802.15.4 [14], a router can select one active portion as its *outgoing superframe*, and based on the active portion selected by its parent, the active portion is called its *incoming superframe* (as shown in Fig. 2(b)). In an outgoing/incoming superframe, a router is expected to transmit/receive a beacon to/from its child routers/parent router. When choosing a slot, neighboring routers' active portions (i.e., outgoing superframes) should be shifted away from each other to avoid interference. This work is motivated by the observation that the specification does not clearly define how to choose the locations of routers' active portions such that the convergecast latency can be reduced. In our work, we consider two kinds of interference between routers. Two routers have *direct interference* if they can hear each others' beacons. Two routers have *indirect interference* if they have at least one common neighbor. Both interferences should be avoided when choosing routers' active portions. Table 1 lists possible choices of $(BO-SO)$ combinations.

Table 1: Relationship of $BO - SO$, duty cycle, and the number of active portions in a superframe.

$BO - SO$	0	1	2	3	4	5	6	7	8	≥ 9
Duty cycle (%)	100	50	25	12.5	6.25	3.13	1.56	0.78	0.39	≤ 0.195
Number of active portions (slots)	1	2	4	8	16	32	64	128	256	≥ 512

3 The Minimum Delay Beacon Scheduling (MDBS) Problem

This section formally defines the convergecast problem in ZigBee networks. Given a ZigBee network, we model it by a graph $G = (V, E)$, where V contains all routers and the coordinator and E contains all symmetric communication links between nodes in V. The coordinator also serves as the sink of the network. End devices can only associate with routers, but are not included in V. From G, we can construct an *interference graph* $G_I = (V, E_I)$, where edge $(i, j) \in E_I$ if there are direct/indirect interferences between i and j. There is a duty cycle requirement α for this network. From α and Table 1, we can determine the most appropriate value of $BO - SO$. We denote by $k = 2^{BO-SO}$ the number of active portions (or slots) per beacon interval.

The beacon scheduling problem is to find a slot assignment $s(i)$ for each router $i \in V$, where $s(i)$ is an integer and $s(i) \in [0, k-1]$, such that router i's active portion is in slot $s(i)$ and $s(i) \neq s(j)$ if $(i, j) \in E_I$. Here the slot assignment means the position of the outgoing superframe of each router (the position of the incoming superframe, as clarified earlier, is determined by the parent of the router). Motivated by Brook's theorem [21], which proves that n colors are sufficient to color any graph with a maximum degree of n, we would assume that $k \geq D_I$, where D_I is the maximum degree of G_I.

Given a slot assignment for G, the report latency from node i to node j, where $(i, j) \in E$, is the number of slots, denoted by d_{ij}, that node i has to wait to relay its collected sensory data to node j, i.e.,

$$d_{ij} = (s(j) - s(i)) \bmod k. \tag{1}$$

Note that the report latency from node i to node j (d_{ij}) may not by equal to the report latency

from node j to node i (d_{ji}). Therefore, we can convert G into a weighted directed graph $G_D = (V, E_D)$ such that each $(i, j) \in E$ is translated into two directed edges (i, j) and (j, i) such that $w((i, j)) = d_{ij}$ and $w((j, i)) = d_{ji}$. The report latency for each $i \in V$ to the sink is the sum of report latencies of the links on the shortest path from i to the sink in G_D. The latency of the convergecast, denoted as $L(G)$, is the maximum of all nodes' report latencies.

Definition 1 *Given $G = (V, E)$, G's interference graph $G_I = (V, E_I)$, and k available slots, the* Minimum Delay Beacon Scheduling (MDBS) problem *is to find an interference-free slot assignment $s(i)$ for each $i \in V$ such that the convergecast latency $L(G)$ is minimized.*

To prove that the MDBS problem is NP-complete, we define a decision problem as follows.

Definition 2 *Given $G = (V, E)$, G's interference graph $G_I = (V, E_I)$, k available slots, and a delay constraint d, the* Bounded Delay Beacon Scheduling (BDBS) problem *is to decide if there exists an interference-free slot assignment $s(i)$ for each $i \in V$ such that the convergecast latency $L(G) \leq d$.*

Theorem 1 *The BDBS problem is NP-complete.*

Proof. First, given slot assignments for nodes in V, we can find the report latency of each $i \in V$ by running a shortest path algorithm on G_D. We can then check if $L(G) \leq d$. Clearly, this takes polynomial time.

We then prove that the BDBS problem is NP-hard by reducing the 3 conjunctive normal form satisfiability (3-CNF-SAT) problem to a special case of the BDBS problem in polynomial time. Given any 3-CNF formula C, we will construct the corresponding G and G_I. Then we show that C is satisfiable *if and only if* there is a slot assignment for each $i \in V$ using no more than $k = 3$ slots such that $L(G) \leq 4$ slots.

Let $C = C_1 \wedge C_2 \wedge \cdots \wedge C_m$, where clause $C_j = x_{j,1} \vee x_{j,2} \vee x_{j,3}$, $1 \leq j \leq m$, $x_{j,i} \in \{X_1, X_2, ..., X_n\}$, and $X_i \in \{x_i, \bar{x}_i\}$, where x_i is a binary variable, $1 \leq i \leq n$. We first construct G from C as follows:

1. For each clause C_j, $j = 1, 2, ..., m$, add a vertex C_j in G.

2. For each literal X_i, $i = 1, 2, ..., n$, add four vertices x_{i1}, x_{i2}, $\bar{x}_{i1}$, and $\bar{x}_{i2}$ in G.

3. Add a vertex t as the sink of G.

4. Add edges (t, x_{i2}) and $(t, \bar{x}_{i2})$ to G, for $i = 1, 2, ..., n$.

5. Add edges (x_{i1}, x_{i2}) and $(\bar{x}_{i1}, \bar{x}_{i2})$ to G, for $i = 1, 2, ..., n$.

6. For each $i = 1, 2, ..., n$ and each $j = 1, 2, ..., m$, add an edge (C_j, x_{i1}) (resp., $(C_j, \bar{x}_{i1})$) to G if x_i (resp., $\bar{x}_i$) appears in C_j.

Then we construct G_I as follows.

1. Add all vertices and edges in G into G_I.

2. Add edges $(x_{i1}, \bar{x}_{i1})$ and $(x_{i2}, \bar{x}_{i2})$ to G_I, for $i = 1, 2, ..., n$.

3. Add edges (C_j, x_{i2}) and $(C_j, \bar{x}_{i2})$ to G_I, for $i = 1, 2, ..., n$ and $j = 1, 2, ..., m$.

Then we build a one-to-one mapping from each truth assignment of C to a slot assignment of G. We establish the following mapping:

1. Set $s(t) = 0$.

2. Set $s(C_j) = 0$, $j = 1, 2, ..., m$.

3. Set $s(x_{i1}) = 1$ and $s(\bar{x}_{i2}) = 1$, $i = 1, 2, ..., n$, if x_i is true; otherwise, set $s(x_{i1}) = 2$ and $s(\bar{x}_{i2}) = 2$.

4. Set $s(x_{i2}) = 1$ and $s(\bar{x}_{i1}) = 1$, $i = 1, 2, ..., n$, if $\bar{x}_i$ is true; otherwise, set $s(x_{i2}) = 2$ and $s(\bar{x}_{i1}) = 2$.

The above reduction can be computed in polynomial time. By the above reduction, vertices x_{i1} or $\bar{x}_{i1}$, $i = 1, 2, ..., n$, that are assigned to slot 1 (resp. slot 2) will have a report latency of 2 (resp. 4) and vertices x_{i2} or $\bar{x}_{i2}$, $i = 1, 2, ..., n$, that are assigned to slot 1 (resp. slot 2) will have a report latency of 2 (resp. 1). Hence, for those vertices x_{i1}, $\bar{x}_{i1}$, x_{i2}, and $\bar{x}_{i2}$, $i = 1, 2, ..., n$, the longest report latency will be 4.

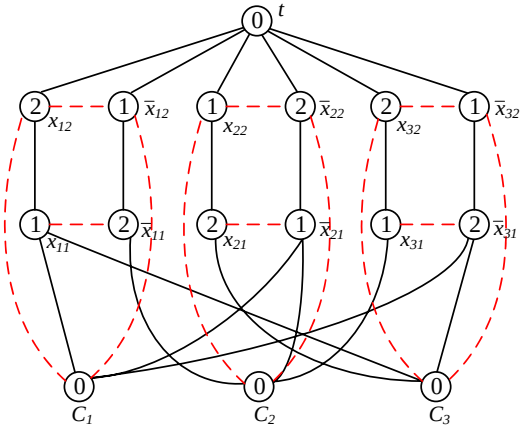


Figure 3: An example of reduction from the 3-CNF-SAT to the BDBS problem.

To prove the *if* part, we need to show that if C is satisfiable, there is a slot assignment such that $k = 3$ and $L(G) \leq 4$. Since C satisfiable, there must exist an assignment such that each clause C_j, $j = 1, 2, ..., m$, is true. If a clause C_j is true, at least one variable in C_j is true. According to the reduction, C_j can always find an edge (C_j, x_{i1}) or $(C_j, \bar{x}_{i1})$ with $w((C_j, x_{i1})) = 1$ or $w((C_j, \bar{x}_{i1})) = 1$, where $i = 1, 2, ..., n$. Thus, when C is satisfiable, the reporting latency for each clause is 3. This achieves $L(G) = 4$.

For the *only if* part, if each vertex C_j, $j = 1, 2, ..., m$, can find at least an edge with weight 1 to one of x_{i1} and $\bar{x}_{i1}$, for $i = 1, 2, ..., n$, to achieve a report latency of 3, it must be that each clause has at least one variable to be true. So formula C is satisfiable. Otherwise, the report latency of C_j, $j = 1, 2, ..., m$, will be 6. □

For example, given $C = (x_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_3) \wedge (x_1 \vee x_2 \vee \bar{x}_3)$, Fig. 3 shows the corresponding G. The truth assignment $(x_1, x_2, x_3) = (T, F, T)$ makes C satisfiable. According to the reduction and the mapping in the above proof, we can obtain the network G and its slot assignment as shown in Fig. 3 such that $L(G) = 4$. □

4 Algorithms for the MDBS Problem

4.1 Optimal Solutions for Special Cases

Optimal solutions can be found for the MDBS problem in polynomial time for regular linear networks and regular ring networks, as illustrated in Fig. 4. In such networks, each vertex is connected to one or two adjacent vertices and has an interference relation with each neighbor within h hops from it, where $h \geq 2$. In a regular linear network, we assume that the sink t is at one end of the network. Clearly, the maximum degree of G_I is $2h$. We will show that an optimal solution can be found if the number of slots $k \geq h + 1$. The slot assignment can be done in a bottom-up manner. The bottom node is assigned to slot 0. Then, for each vertex v, $s(v) = (k' + 1) \bmod k$, where k' is the slot assigned to v's child.

Theorem 2 *For a regular linear network, if $k \geq h + 1$, the above slot assignment achieves a report latency of $|V| - 1$, which is optimal.*

Proof. Clearly, the slot assignment is interference-free. Also the report latency of $|V| - 1$ is clearly the lower bound. □

For a regular ring network, we first partition vertices excluding t into left and right groups as illustrated in Fig. 4(b) such that the left group consists of the sink node t and $\lfloor \frac{|V|-1}{2} \rfloor$ other nodes counting counter-clockwise from t, and the right group consists of those $\lceil \frac{|V|-1}{2} \rceil$ nodes counting clockwise from t. Now we consider the ring as a spanning tree with t as the root and left and right groups as two linear paths. Assuming that $\lfloor \frac{|V|-1}{2} \rfloor \geq 2h$ and $k \geq 2h$, the slot assignment works as follows:

1. The bottom node in the left group is assigned to slot 0.

2. All other nodes in the left group are assigned with slots in a bottom-up manner. For each node i in the left group, we let $s(i) = (j + 1) \bmod k$, where j is the slot of i's child.

3. Nodes in the right group are assigned with slots in a top-down manner. For each node i in the right group, we let $s(i) = (j - c) \bmod k$, where j is the slot assigned to i's

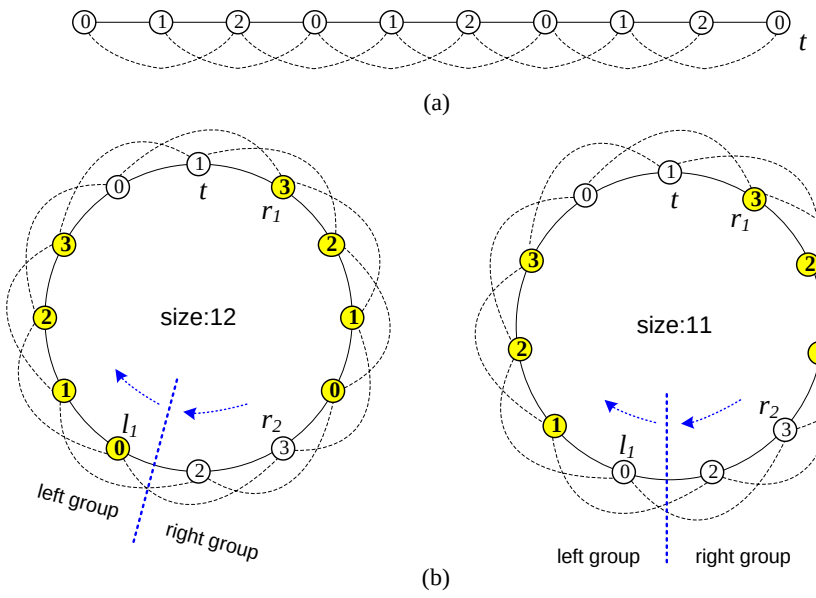


Figure 4: Examples of optimal slot assignments for regular linear and ring networks ($h = 2$). Dotted lines mean interference relations.

parent and c is the smallest constant ($1 \leq c \leq k$) that ensures that $s(i)$ is not used by any of its interference neighbors that have been assigned with slots.

It is not hard to prove the slot assignment is interference-free because nodes receives slots sequentially and we have avoided using the same slots among interfering neighbors. Although this is a greedy approach, we show that c is equal to 1 in step 3 in most of the cases except when two special nodes are visited. This gives an asymptotically optimal algorithm, as proved in the following theorem.

Theorem 3 *For a regular ring network, assuming that* $k \geq 2h$ *and* $\lfloor \frac{|V|-1}{2} \rfloor \geq 2h$, *the above slot assignment achieves a report latency* $L(G) = \lfloor \frac{|V|-1}{2} \rfloor + h$, *which is optimal within a factor of 1.5.*

Proof. We first identify three nodes on the ring (refer to Fig. 4(b)):

- l_1: the bottom node in the left group.
- r_1: the first node in the right group.

- r_2: the node that is h hops from l_1 counting counterclockwise.

The report latency of each node can be analyzed as follows. The parent of node x is denoted by $par(x)$.

A1. For each node i in the left group except the sink t, the latency from i to $par(i)$ is 1.

A2. The latency from r_1 to t is h.

A3. For each node i next to r_1 in the right group but before r_2 (counting clockwise), the latency from i to $par(i)$ is 1.

A4. The latency from r_2 to $par(r_2)$ is 1 if the ring size is even; otherwise, the latency is 2.

A5. For each node i in the right group that is a descendant of r_2, the report latency from i to $par(i)$ is 1.

It is not hard to prove that A1, A2, and A3 are true. To see A4 and A5, we make the following observations. The function $par^i(x)$ is to apply i times the $par()$ function on node x. Note that $par^0(x)$ means x itself.

O1. When the ring size is even, the equality $s(par^{i-1}(l_1)) = s(par^i(r_2))$ holds for $i = 1, 2, ..., \lfloor \frac{|V|-1}{2} \rfloor - h - 1$. More specifically, this means that (i) l_1 and $par(r_2)$ will receive the same slot, (ii) $par(l_1)$ and $par^2(r_2)$ will receive the same slot, etc. This can be proved by induction by showing that the i-th descendant of t in the right group will be assigned the same slot as the $(h + i - 1)$-th descendant of t in the left group (the induction can go in a top-down manner). This property implies that when assigning a slot to r_2 in step 3, $c = 1$ in case that the ring size is even. Further, r_2 and its descendants will be sequentially assigned to slots $k-1$, $k-2$, ..., $k-h$, which implies that $c = 1$ when doing the assignments in step 3. So properties A4 and A5 hold for the case of an even ring.

O2. When the ring size is odd, the equality $s(par^i(l_1)) = s(par^i(r_2))$ holds for $i = 1, 2, ...,$ $\lfloor \frac{|V|-1}{2} \rfloor - h$. This means that (i) $par(l_1)$ and $par(r_2)$ will receive the same slot, and

(ii) $par^2(l_1)$ and $par^2(r_2)$ will receive the same slot, etc. Again, this can be proved by induction as in O1. This property implies that $c = 2$ when assigning a slot to r_2 in step 3, and $c = 1$ when assigning slots to descendants of r_2. So properties A4 and A5 hold for the case of an odd ring.

The equality of slot assignments pointed out in O1 and O2 is illustrated in Fig. 4(b) by those numbers in gray nodes. In summary, the report latency of the left group is $\lfloor \frac{|V|-1}{2} \rfloor$. When the ring size is even, the report latency of the right group is the number of nodes in this group, $\frac{|V|}{2}$, plus the extra latency $h-1$ incurred at r_1. So $L(G) = \frac{|V|}{2} + h - 1 = \lfloor \frac{|V|-1}{2} \rfloor + h$. When the ring size is odd, the report latency of right group is the number of nodes in this group, $\frac{|V|-1}{2}$, plus the extra latency $h-1$ incurred at r_1 and the extra latency 1 incurred at r_2. So $L(G) = \lfloor \frac{|V|-1}{2} \rfloor + h$.

A lower bound on the report latency of this problem is the maximum number of nodes in each group excluding t. Applying $\lfloor \frac{|V|-1}{2} \rfloor$ as a lower bound and using the fact that $\lfloor \frac{|V|-1}{2} \rfloor \geq 2h$, $L(G)$ will be smaller than $1.5 \times \lfloor \frac{|V|-1}{2} \rfloor$, which implies the algorithm is optimal within a factor of 1.5. Note that the condition $\lfloor \frac{|V|-1}{2} \rfloor \geq 2h$ is to guarantee that t will not locate within h hops from r_2. Otherwise, the observation O2 will not hold. □

4.2 A Centralized Tree-Based Assignment Scheme

Given $G = (V, E)$, $G_I = (V, E_I)$, and k, we propose a centralized slot assignment heuristic algorithm. Our algorithm is composed of the following three phases:

phase 1. From G, we first construct a BFS tree T rooted at sink t.

phase 2. We traverse vertices of T in a bottom-up manner. For these vertices in depth d, we first sort them according to their degrees in G_I in a descending order. Then we sequentially traverse these vertices in that order. For each vertex v in depth d visited, we compute a temporary slot number $t(v)$ for v as follows.

1. If v is a leaf node, we set $t(v)$ to the minimal non-negative integer l such that for each vertex u that has been visited and $(u, v) \in E_I$, ($t(u)$ mod k) $\neq l$.

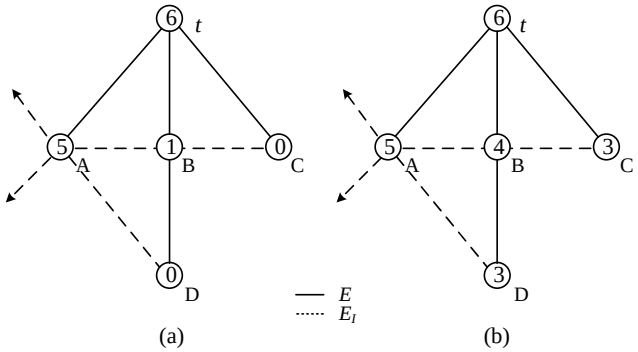


Figure 5: (a) Slot assignment after phase 2. (b) Slot compacting by phase 3.

2. If v is an in-tree node, let m be the maximum of the numbers that have been assigned to v's children, i.e., $m = \max\{t(child(v))\}$, where $child(v)$ is the set of v's children. We then set $t(v)$ to the minimal non-negative integer $l > m$ such that for each vertex u that has been visited and $(u, v) \in E_I$, ($t(u)$ mod k) $\neq$ (l mod k).

After every vertex v is visited, we make the assignment $s(v) = t(v)$ mod k.

phase 3. In this phase, vertices are traversed sequentially from t in a top-down manner. When each vertex v is visited, we try to greedily find a new slot l such that ($s(par(v)) - l$) mod $k <$ ($s(par(v)) - s(v)$) mod k, such that $l \neq s(u)$ for each $(u, v) \in E_I$, if possible. Then we reassign $s(v) = l$.

Note that in phase 2, a node with a higher degree means that it has more interference neighbors, implying that it has less slots to use. Therefore, it has to be assigned to a slot earlier. Also note that, the number $t(v)$ is not a modulus number. However, in step 2 of phase 2, we did check that if $t(v)$ is converted to a slot number, no interference will occur. Intuitively, this is a temporary slot assignment that will incur the least latency to v's children. At the end, $t(v)$ is converted to a slot assignment $s(v)$. Phase 3 is a greedy approach to further reduce the report latency of routers. For example, Fig. 5(a) shows the slot assignment after phase 2. Fig. 5(b) indicates that B, C, and D can find another slots and their report latencies are decreased. This phase can reduce $L(G)$ in some cases.

The computational complexity of this algorithm is analyzed below. In phase 1, the complexity of constructing a BFS tree is $O(|V| + |E|)$. In phase 2, the cost of sorting is at most $O(|V|^2)$ and the computational cost to compute $t(v)$ for each vertex v is bounded by $O(kD_I)$, where D_I is the degree of G_I. So the time complexity of phase 2 is $O(|V|^2 + kD_I|V|)$. Phase 3 performs a similar procedure as phase 2, so its time complexity is also $O(kD_I|V|)$. Overall, the time complexity is $O(|V|^2 + kD_I|V|)$.

4.3 A Distributed Assignment Scheme

In this section, we propose a distributed slot assignment algorithm. Each node has to compute its direct as well as indirect interference neighbors in a distributed manner. To achieve this, we will refer to the *heterogeneity* approach in [22], which adopts power control to achieve this goal. Assuming routers' default transmission range is r, interference neighbors must locate within range $2r$. From time-to-time, each router will boost its transmission power to double its default transmission range and send HELLO packets to its neighbor routers. Each HELLO packet further contains sender's 1) depth[1], 2) the location of outgoing superframe (i.e., slot), and 3) number of interference neighbors. Note that all other packets are transmitted by the default power level. When booting up, each router will broadcast HELLO packets claiming that its depth and slot are *NULL*. After joining the network and choosing a slot, the HELLO packets will carry the node's depth and slot information. The algorithm is triggered by the sink t setting $s(t) = k - 1$ and then broadcasting its beacon. A router $v \neq t$ that receives a beacon will decide its slot as follows.

1. Node v sends an association request to the beacon sender.

2. If v fails to associate with the beacon sender, it stops the procedure and waits for other beacons.

3. If v successfully associates with a parent node $par(v)$, it computes the smallest positive integer l such that $(s(par(v)) - l) \bmod k \neq s(u)$ for all $(u, v) \in E_I$ and $s(u) \neq$ *NULL*. Then v chooses $s(v) = (s(par(v)) - l) \bmod k$ as its slot.

[1]The depth of a node is the length of the tree path from the root to the node. The root node is at depth zero.

4. Then, v broadcasts HELLOs including its slot assignment $s(v)$ for a time period t_{wait}. If it finds that $s(v) = s(u)$ for any $(u, v) \in E_I$, v has to change to a new slot if one of the following rules is satisfied and goes back to step 3.

   (a) Node u has more interference neighbors than v.

   (b) Node u and v have the same number of interference neighbors but the depth of u is lower than v, i.e. u is closer to the sink than v.

   (c) Node u and v have the same number of interference neighbors and they are at the same depth but the u's ID is smaller than v's.

5. After t_{wait}, v can finalize its slot selection and broadcast its beacons.

In this distributed algorithm, slots are assigned to routers, ideally, in a top-down manner. However, due to transmission latency, some routers at lower levels may find slots earlier than those at higher levels. Also note that the time t_{wait} is to avoid possible collision on slot assignments due to packet loss.

5 Simulation Results

This section presents our simulation results. We first assume that the size of sensory data is negligible and that all routers generate reports at the same time, and compare the performances of different convergecast algorithms. Then we simulate more realistic scenarios where the size of sensory data is not negligible and routers need to generate reports periodically or passively driven by events randomly appearing in certain regions in the sensing field. More specifically, sensors generate reports according to certain application specifications. Devices all run ZigBee and IEEE 802.15.4 protocols to communicate with each other. Routers can aggregate child sensors' reports and report to their parents directly. Each router has a fix-size buffer. When a router's buffer overflows, this router will not accept further incoming frames. We also measure the *goodput* of the network, which is defined as the ratio of sensors' reports successfully received by the sink. Some parameters used in our simulation are listed in Table 2.

Table 2: Simulation parameters.

Parameter	Value
length of a frame's header and tail	18 Bytes
length of a sensor's report	16 Bytes
beacon length	18 Bytes
maximum length of a frame	127 Bytes
bit rate	250k bps
symbol rate	62.5k symbols/s
aBaseSuperframeDuration	960 symbols
aUnitBackoffPeriod	20 symbols
aCCATime	8 symbols
macMinBE	3
aMaxBE	5
macMaxCSMABackoffs	4
maximum number of retransmissions	3

5.1 Comparison of Different Convergecast Algorithms

We compare the proposed slot assignment algorithms against a random slot assignment (denoted by RAN) scheme and a greedy slot assignment (denoted by GDY) scheme. In RAN, the slot assignment starts from the sink and each router, after associating with a parent router, simply chooses any slot which has not been used by any of its interference neighbors. In GDY, routers are given a sequence number in a top-down manner. The sink sets its slot to $k-1$. Then the slot assignment continues in sequence. For a node i, it will try to find a slot $s(i) = s(j) - l \bmod k$, where j is the predecessor of i and l is the smallest integer letting $s(i)$ is the slot which does not assign to any of i's interference neighbors. In the simulations, routers are randomly distributed in a circular region of a radius r and a sink is placed in the center. Our centralized tree-based scheme and distributed slot assignment scheme are denoted as CTB and DSA, respectively. We compare the report latency $L(G)$ (in terms of slots).

Fig. 6 shows some slot assignment results of CTB and DSA when $r = 35\ m$ and $k = 64$. Devices are randomly distributed. The transmission range of routers is set to $20\ m$. In this case, CTB performs better than DSA.

Next, we observe the impact of different r, C_R (number of routers), and T_R (transmission

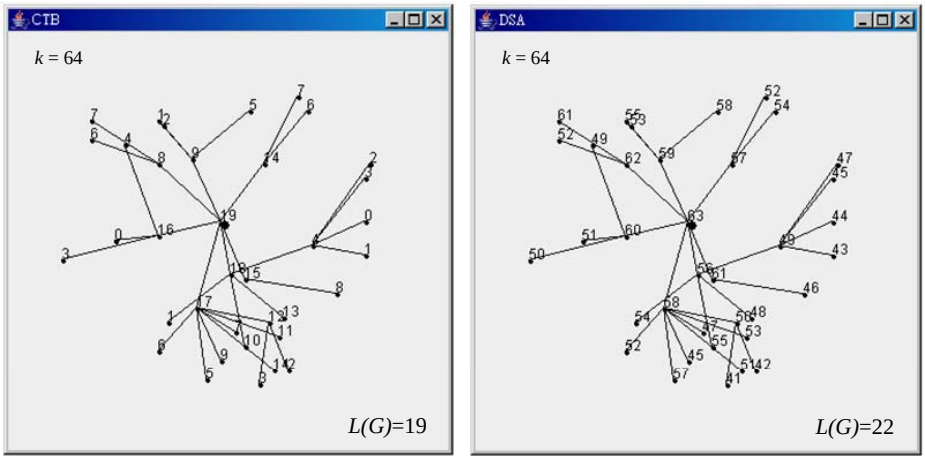


Figure 6: Slot assignment examples by CTB and DSA.

distance). Fig. 7(a) shows the impact of r when $k = 64$, $T_R = 25\ m$, and $C_R = 3 \times (r/10)^2$. CTB performs the best. DSA performs slightly worse than CTB, but still significantly outperforms RAN and GDY. It can be seen that RAN and GRY could result in very long convergecast latency. Both CTB and DSA are quite insensitive to the network size. But this is not the case for RAN and GDY. Fig. 7(b) shows the impact of T_R when $C_R = 300$, $r = 100\ m$, and $k = 64$. Since a larger transmission range implies higher interference among routers, the report latencies of CTB and DSA will increase linearly as T_R increases. The report latency of RAN also increases when $T_R = 17 \sim 21\ m$ because of the increased interference. After $T_R \geq 22\ m$, the latency of RAN decreases because that the network diameter is reduced. Basically, GDY behaves the same as CTB and DSA. But when the transmission range is larger, the report latency slightly becomes small.

Fig. 7(c) shows the impact of C_R when $r = 100\ m$, $T_R = 20\ m$, and $k = 128$. As a larger C_R means a higher network density and thus more interference, the report latencies of CTB and DSA increase as C_R increases. Since the network diameter is bounded, the report latency of RAN is also bounded. GDY is sensitive to the number of routers when there are less routers. This is because that each router can own a slot and the report latency increases proportionally to the number of routers. With $r = 100\ m$, $C_R = 300$, and $T_R = 20\ m$, Fig. 7(d) shows the impact of routers' duty cycle. Note that a lower duty cycle means a larger number of available slots. Interestingly, we see that the report latencies of CTB, DSA,

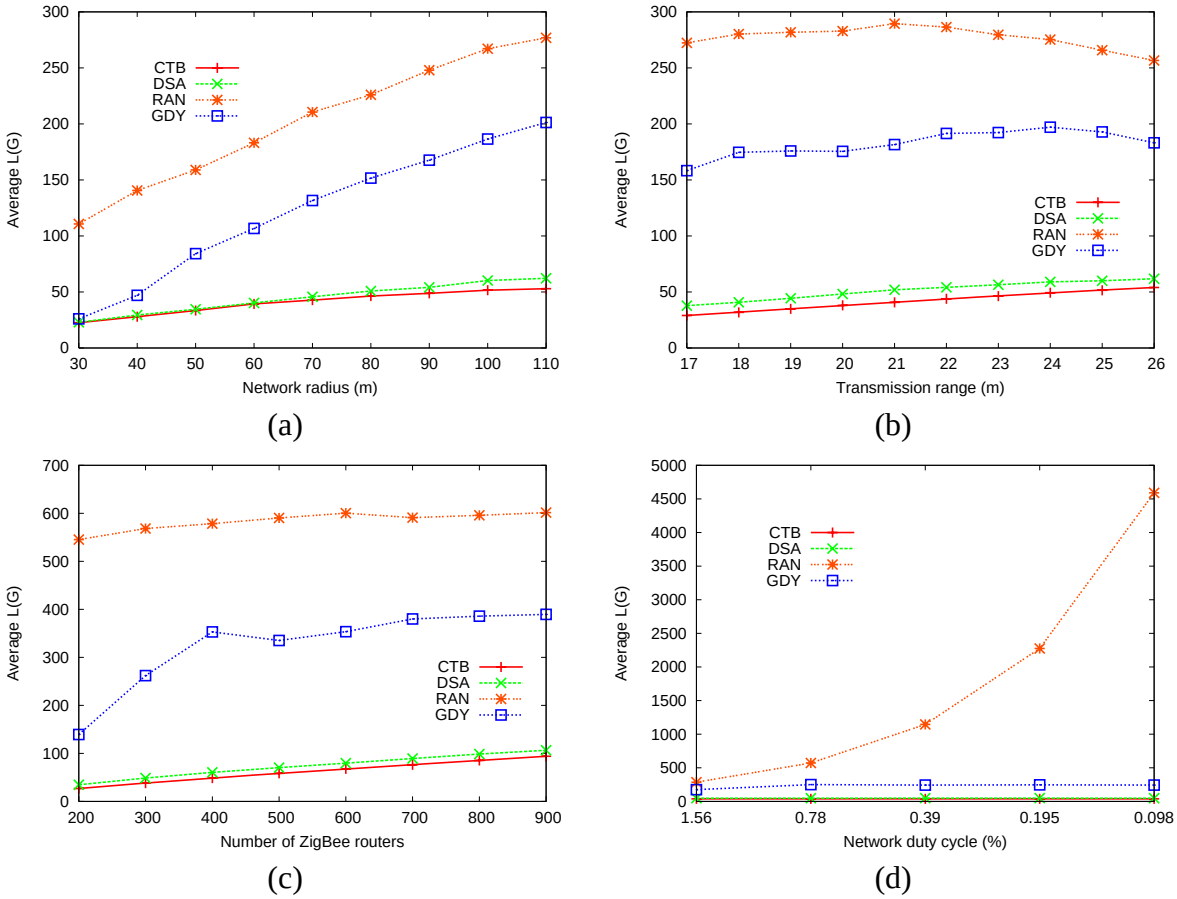


Figure 7: Comparison of report latencies under different configurations.

and GDY are independent of the number of slots. Contrarily, with a random assignment, RAN even incurs a higher report latency as there are more freedom in slot selection.

5.2 Periodical Reporting Scenarios

Next, we assume that sensors are instructed to report their data in a periodically manner. We set $r = 100\ m$, $T_R = 20\ m$, and $C_R = 300$ with 6000 randomly placed sensors associated to these routers, and we further restrict a router can accept at most 30 sensors. $BO - SO$ is fixed to six, so $k = 2^{BO-SO} = 64$. Since the earlier simulations show that CTB and DSA perform quite close, we will use only CTB to assign routers' slots. Sensors are required to generate a report every 251.66 second (the length of one beacon interval when $BO = 14$). We set the buffer size of each router is 10 KB.[2] We allocate two mini-slots for each child router of the sink as the GTS slot. [3]

Since $(BO - SO)$ is fixed, a small BO implies a smaller slot size (and thus a smaller unit size of $L(G)$). So, a smaller slot size seemingly implies higher contention among sensors if they all intend to report to their parents simultaneously. In fact, a smaller BO does not hurt the overall reporting times of sensors if we can properly divide sensors into groups. For example, in Fig. 8, when $BO = 14$, all sensors of a router can report in every superframe. When $BO = 13$, if we divide sensors into two groups, then they can report alternately in odd and even superframes. Similarly, when $BO = 12$, four groups of sensors can report alternately. Since the length of superframes are reduced proportionally, the report intervals of sensors actually remain the same in these cases. In the following experiments, we groups sensors according to their parents' IDs. A sensor belongs to group m if the modulus of its parent's ID is m.

Fig. 9 shows the theoretical and actual report latencies under different BOs. Note that a report may be delayed due to buffer constraint. As can be seen, the actual latency does not always favor a smaller BO. Our results show that $BO = 10 \sim 12$ performs better. Fig. 9(b) shows the goodput of sensory reports, channel utilization at the sink, and the number of

[2]Currently, there are some platforms which are equipped with larger RAMs. For example, Jennic JN5121 [5] has a 96KB RAM and CC2420DBK [1] has a 32KB RAM.

[3]There are sixteen mini-slots per active portion (slot).

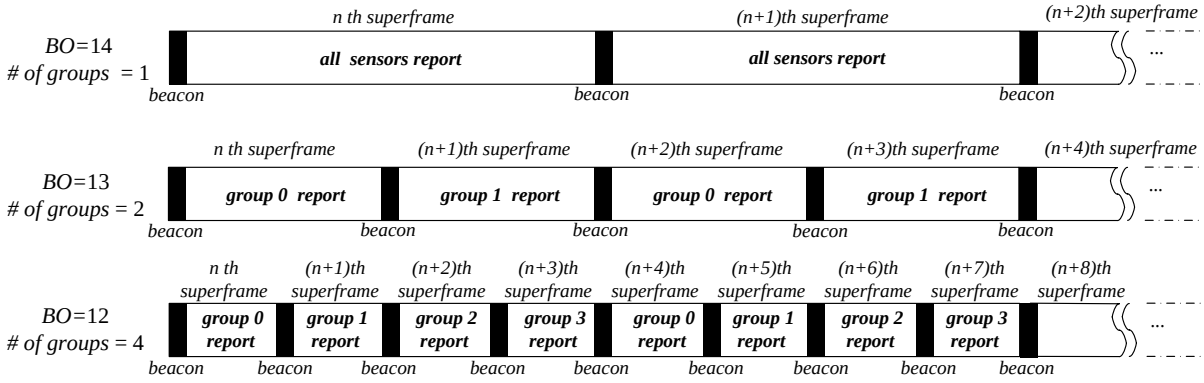
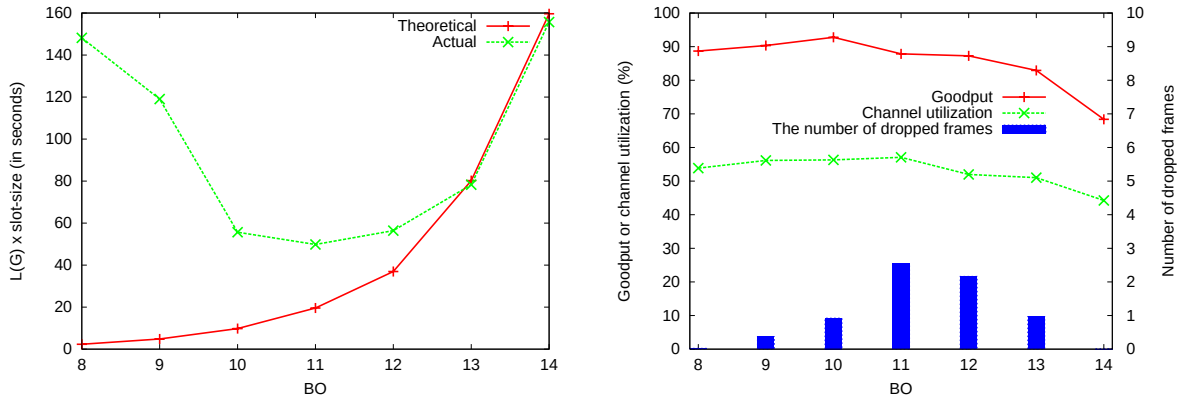


Figure 8: An example of report scheduling under different values of BO.



(a) (b)

Figure 9: Simulations considering buffer limitation and contention effects: (a) theoretical v.s. actual report latencies and (b) goodput, channel utilization, and number of dropped frames.

dropped frames at the sink. When $BO = 14$, although there is no frames being dropped at the sink, the goodput is still low. This is because a lot of collisions happen inside the network, causing many sensory reports being dropped at intermediate levels (a frame is dropped after exceeding its retransmission limit). Fig. 10 shows a log of the numbers of frames received by a sink's child router when $BO = 14$. We can see that more than half of the active portion is wasted. Overall, $BO = 10$ produces the best goodput and a shorter report latency.

Some previous works can be also integrated in this periodical reporting scenario, such as the adaptive GTS allocation mechanism in [12] and the aggregation algorithms for WSNs in

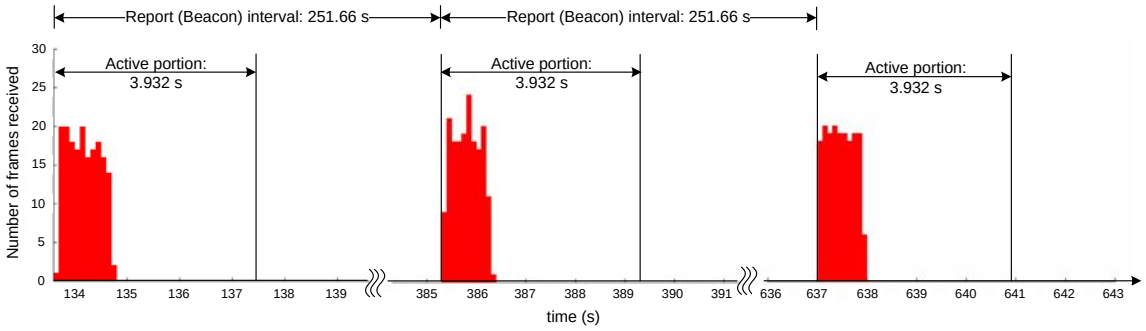
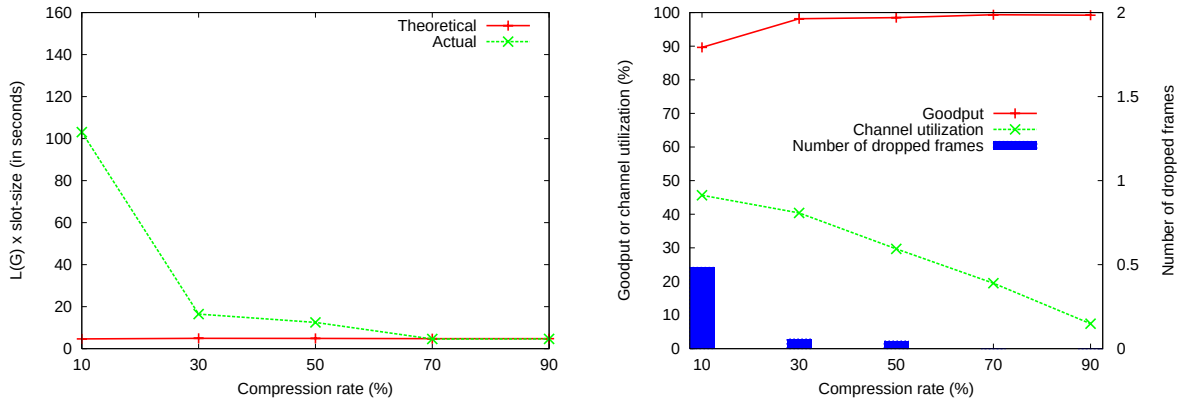


Figure 10: A log of the number of frames received by a sink's child router when $BO = 14$.



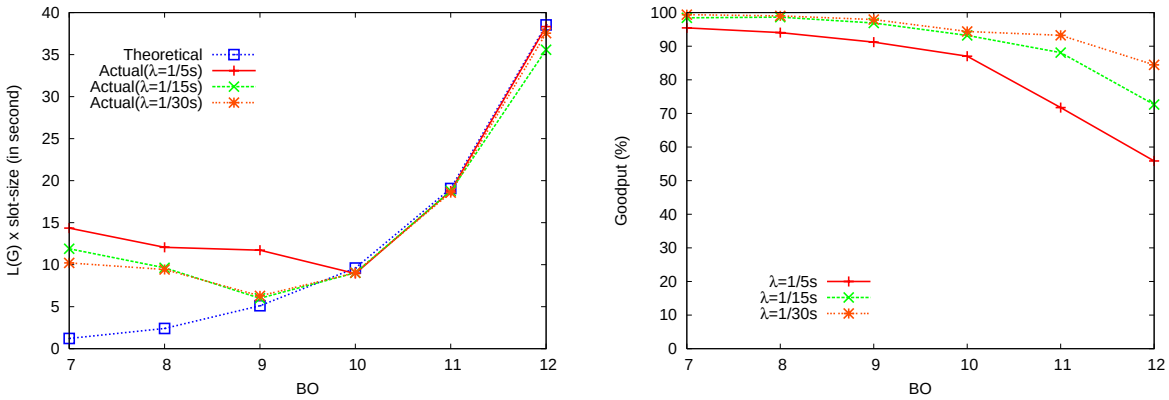
(a) (b)

Figure 11: Simulations considering data compression: (a) theoretical v.s. actual report latencies and (b) goodput, channel utilization, and number of dropped frames.

[7][10]. Fig. 11 shows an experiment that routers can compress reports from sensors with a rate cr when $BO = 10$. If a router receives n reports and each report's size is 16 Bytes (as in Table 2), it can compress the size to $16 \times n \times (1 - cr)$. The report latencies decrease when the cr becomes larger. By compressing the report data, the goodput can up to 98% and the report can arrive to the sink more quickly.

5.3 Event-Driven Reporting Scenarios

In the following, we assume that sensors' reporting activities are triggered by events occurred at random locations in the network with a rate λ. The sensing range of each sensors is 3



(a) (b)

Figure 12: Simulation results of event-driven scenarios: (a) theoretical v.s. actual report latencies and (b) goodput.

meters and each event is a disk of a radius of 5 meters. A sensor can detect an event if its sensing range overlaps with the disk of that event. Each router has an 1 KB buffer. When a sensor detects an event, it only tries to report that event once. All other settings are the same as those in Section 5.2.

Fig. 12 shows the simulation results when $\lambda = 1/5s, 1/15s,$ and $1/30s$. From Fig. 12(a), we can observe that when BO is small, the report latency can not achieve to the theoretical value. This is because that an active portion is too small to accommodate all reports from sensors, thus lengthening the report latency. When BO becomes larger, the theoretical and actual curves would meet. However, the good put will degrade, as shown in Fig. 12(b). This is because reports are likely to be dropped due to buffer overflow. How to determine a proper BO, which can contain most of the reports and guarantee low latency, is an important design issue for such scenarios.

6 Conclusions

In this paper, we have defined a new minimum delay beacon scheduling (MDBS) problem for convergecast with the restrictions that the beacon scheduling must be compliant to the ZigBee standard. We prove the MDBS problem is NP-complete and propose optimal so-

lutions for special cases and two heuristic algorithms for general cases. Simulation results indicate the performance of our heuristic algorithms decrease only when the number of interference neighbors is increased. Compared to the random slot assignment and greedy slot assignment scheme, our heuristic algorithms can effectively schedule the ZigBee routers' beacon times to achieve quick convergecast. In the future, it deserves to consider extending this work to an asynchronous sleep scheduling to support energy-efficient convergecast in ZigBee mesh networks.

7 Acknowledgements

Y.-C. Tseng's research is co-sponsored by Taiwan MoE ATU Program, by NSC grants 93-2752-E-007-001-PAE, 96-2623-7-009-002-ET, 95-2221-E-009-058-MY3, 95-2221-E-009-060-MY3, 95-2219-E-009-007, 95-2218-E-009-209, and 94-2219-E-007-009, by Realtek Semiconductor Corp., by MOEA under grant number 94-EC-17-A-04-S1-044, by ITRI, Taiwan, by Microsoft Corp., and by Intel Corp.

References

[1] Chipcon CC2420DBK. http://www.chipcon.com/.

[2] Dust network Inc. http://dust-inc.com/flash-index.shtml.

[3] Design and construction of a wildfire instrumentation system using networked sensors. http://firebug.sourceforge.net/.

[4] Habitat monitoring on great duck island. http://www.greatduckisland.net/technology.php.

[5] Jennic JN5121. http://www.jennic.com/.

[6] Motes, smart dust sensors, wireless sensor networks. http://www.xbow.com/.

[7] S.-J. Baek, G. de Veciana, and X. Su. Minimizing energy consumption in large-scale sensor networks through distributed data compression and hierarchical aggregation. *IEEE Journal on Selected Areas in Communications*, 22(6):1130–1140, 2004.

[8] H. Choi, J. Wang, and E. A. Hughes. Scheduling for information gathering on sensor network. *ACM/Kluwer Wireless Networks*, 2007, in press.

[9] S. Gandham, Y. Zhang, and Q. Huang. Distributed minimal time convergecast scheduling in wireless sensor networks. In *Proc. of IEEE Int'l Conference on Distributed Computing Systems (ICDCS)*, Lisboa, Portugal, 2006.

[10] D. Ganesan, B. Greenstein, D. Perelyubskiy, D. Estrin, and J. Heidemann. An evaluation of multiresolution storage for sensor networks. In *Proc. of ACM Int'l Conference on Embedded Networked Sensor Systems (SenSys)*, Los Angeles, USA, 2003.

[11] B. Hohlt, L. Doherty, and E. Brewer. Flexible power scheduling for sensor networks. In *Proc. of ACM/IEEE Int'l Conference on Information Processing in Sensor Networks (IPSN)*, Berkeley, USA, 2004.

[12] Y.-K. Huang, A.-C. Pang, and T.-W. Kuo. AGA: Adaptive GTS allocation with low latency and fairness considerations for IEEE 802.15.4. In *Proc. of IEEE Int'l Conference on Communications (ICC)*, Istanbul, Turkey, 2006.

[13] 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), 2003.

[14] 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)(revision of IEEE Std 802.15.4-2003), 2006.

[15] Q. Li, M. DeRosa, and D. Rus. Distributed algorithm for guiding navigation across a sensor network. In *Proc. of ACM Int'l Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, Maryland, USA, 2003.

[16] C.-Y. Lin, W.-C. Peng, and Y.-C. Tseng. Efficient in-network moving object tracking in wireless sensor networks. *IEEE Trans. Mobile Computing*, 5(8):1044–1056, 2006.

[17] G. Lu, B. Krishnamachari, and C. S. Raghavendra. An adaptive energy-efficient and low-latency MAC for data gathering in wireless sensor networks. In *Proc. of IEEE Int'l Parallel and Distributed Processing Symposium (IPDPS)*, New Mexico, USA, 2004.

[18] Y.-C. Tseng, S.-P. Kuo, H.-W. Lee, and C.-F. Huang. Location tracking in a wireless sensor network by mobile agents and its data fusion strategies. *The Computer Journal*, 47(4):448–460, 2004.

[19] Y.-C. Tseng, M.-S. Pan, and Y.-Y. Tsai. Wireless sensor networks for emergency navigation. *IEEE Computer*, 39(7):55–62, 2006.

[20] S. Upadhyayula, V. Annamalai, and S. K. S. Gupta. A low-latency and energy-efficient algorithm for convergecast in wireless sensor networks. In *Proc. of IEEE Global Telecommunications Conference (Globecom)*, San Francisco, USA, 2003.

[21] D. B. West. *Introduction to Graph Theory*. Prentice Hall, 2001.

[22] M. Yarvis, N. Kushalnagar, H. Singh, A. Rangarajan, Y. Liu, and S. Singh. Exploiting heterogeneity in sensor networks. In *Proc. of IEEE INFOCOM*, Miami, USA, 2005.

[23] Y. Yu, B. Krishnamachari, and V. K. Prasanna. Energy-latency tradeoffs for data gathering in wireless sensor networks. In *Proc. of IEEE INFOCOM*, Hong Kong, 2004.

[24] ZigBee specification version 2006, ZigBee document 064112, 2006.

附錄三

Quick Convergecast in ZigBee/IEEE 802.15.4 Tree-Based Wireless Sensor Networks

Y.-C. Tseng and M.-S. Pan

ACM Int’l Workshop on Mobility Management and Wireless Access (ACM MobiWac)

Quick Convergecast in ZigBee/IEEE 802.15.4 Tree-Based Wireless Sensor Networks

Yu-Chee Tseng*
Department of Computer Science
National Chiao Tung University
Hsin-Chu, 30010, Taiwan
yctseng@cs.nctu.edu.tw

Meng-Shiuan Pan
Department of Computer Science
National Chiao Tung University
Hsin-Chu, 30010, Taiwan
mspan@cs.nctu.edu.tw

ABSTRACT

Convergecast is a fundamental operation in wireless sensor networks. Existing convergecast solutions have focused on reducing latency and energy consumption. However, a good design should be compliant to standards, in addition to considering these factors. Based on this observation, this paper defines a *minimum delay beacon scheduling problem* for quick convergecast in ZigBee/IEEE 802.15.4 tree-based wireless sensor networks and proves that this problem is NP-complete. Our formulation is also compliant with the low-power design of IEEE 802.15.4. We then propose optimal solutions for special cases and heuristic algorithms for general cases. Simulation results show that the proposed algorithms can indeed achieve quick convergecast.

Categories and Subject Descriptors

C.2.1 [**[Computer-Communication Networks**]: Network Architecture and Design—*Distributed networks*, *Wireless communication*; G.2.2 [**Discrete Mathematics**]: Graph Theory

General Terms

Algorithms, Design, Theory.

Keywords

convergecast, graph theory, IEEE 802.15.4, scheduling, wireless sensor network, ZigBee

1. INTRODUCTION

The rapid progress of wireless communication and embedded micro-sensing MEMS technologies has made *wireless sensor networks (WSNs)* possible. A WSN consists of many inexpensive wireless sensors capable of collecting, storing, processing environmental information, and communicating with neighboring nodes. Applications of WSNs include wildlife monitoring [1, 3], object tracking [9, 11], and dynamic path finding [8, 12].

*Corresponding author.

MobiWac'06, October 2, 2006, Torremolinos, Malaga, Spain.

Recently, many WSN platforms have been developed, such as MICA [4] and Dust Network [2]. For interoperability among different systems, standards such as ZigBee/IEEE 802.15.4 [5, 7] protocols have been developed. ZigBee/IEEE 802.15.4 specifies a global standard on physical, MAC, and network layers for WSNs requiring high reliability, low cost, low power, scalability, and low data rate.

Considering that data gathering is a major application of WSNs, *convergecast* has been investigated in several works [6, 10, 13, 16]. With the goals of low latency and low energy consumption, reference [13] shows how to connect sensors as a balanced reporting tree and how to assign CDMA codes to sensors to diminish interference among sensors, thus achieving energy efficiency. The work [16] aims to minimize the overall energy consumption under the constraint that sensed data should be reported within specified time. Dynamic programming algorithms are proposed by assuming that sensors can receive multiple packets at the same time. As can be seen, both [13] and [16] are based on quite strong assumptions on communication capability of sensor nodes. In [10], the authors propose an energy efficient and low latency MAC, called *DMAC*. Sensors are connected by a tree and stay in sleep mode for most of the time. When waking up, sensors are first set to the receive mode and then to the transmit mode. *DMAC* achieves low-latency by staggering wake-up schedules of sensors at the time instant when their children switch to the transmit mode. Similar to [10], reference [6] arranges wake-up schedule of sensors by taking traffic loads into account. Each parent periodically broadcasts an advertisement containing a set of empty slots. Children nodes request empty slots according to their demands. Although these results [6, 10] are designed for quick convergecast, the solutions are not compliant to ZigBee/IEEE 802.15.4 standards.

This paper aims at designing efficient convergecast solutions for WSNs that are compliant with the ZigBee/IEEE 802.15.4 standards. Assuming a tree topology, Fig. 1 shows the problem scenario. The network contains one *sink* (ZigBee coordinator), some *full function devices* (ZigBee routers), and some *reduced function devices* (ZigBee end devices). Each ZigBee router is responsible for collecting sensed data from end devices associated with it and relaying incoming data to the sink. According to specifications, a ZigBee router can announce a beacon to start a superframe. Each superframe consists of an *active portion* followed by an *inactive portion*. On receiving its parent router's beacon, an end device has also to wake up for an active portion to sense the environment and communicate with its coordinator. However, to avoid collision with its neighbors, a router should shift its active portion by a certain amount. Fig. 1 shows a possible allocation of active portions for routers A, B, C, and D. Assuming that routers relay packets in contention-free *guarantee time slots (GTSs)*, the collected sensory

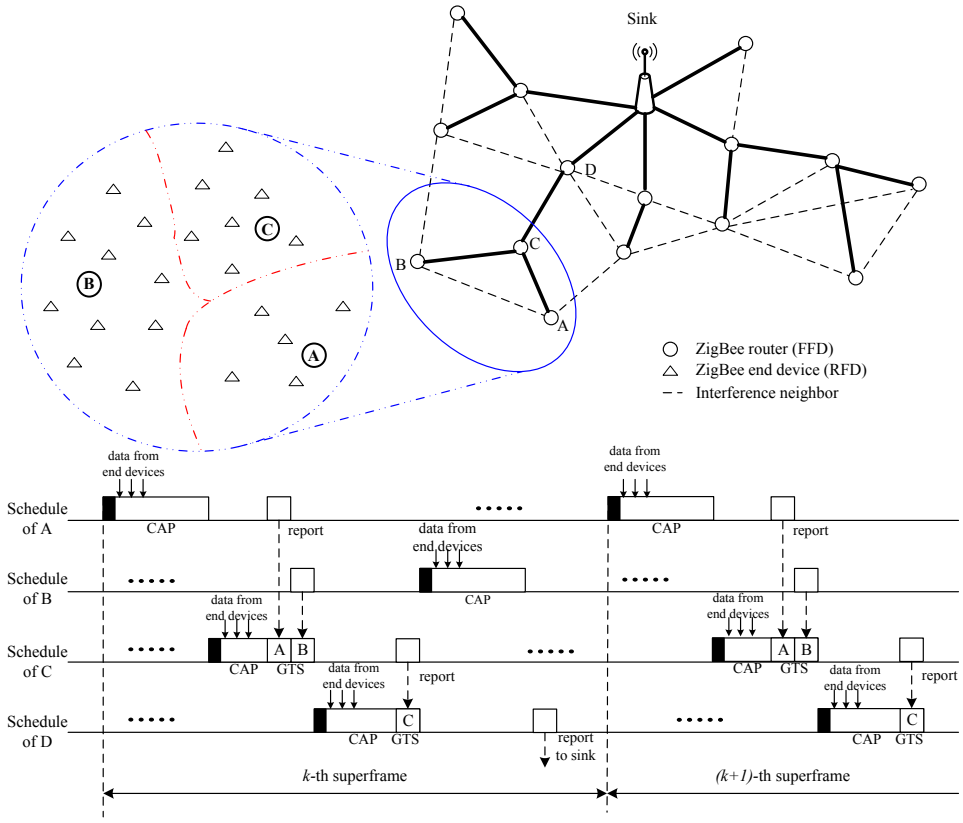


Figure 1: An example of convergecast in a ZigBee/IEEE 802.15.4 tree-based network.

data of A in the k-th superframe can be sent to C via the GTS of C in the k-th superframe. However, because the active portion of B in the k-th superframe appears after that of C, the collected data of B in the k-th superframe can only be relayed to C along the GTS of C in the $(k+1)$-th superframe. The delay can be eliminated if the active portion of B in the k-th superframe appears before that of C. The delay is not negligible because of the low duty cycle design of IEEE 802.15.4. For example, in 2.4 GHz PHY, with 1.56% duty cycle, a superframe can be up to 251.658 seconds (with an active portion of 3.93 seconds). Clearly, for large-scale WSNs, the convergecast latency could be significant. The purpose of this paper is to solve the beacon scheduling problem to minimize the convergecast latency. We prove that this problem is NP-complete by reducing it to the 3-CNF-SAT problem. We show two special cases of this problem where optimal solutions can be found in polynomial time and propose some heuristic algorithms for general cases.

The rest of this paper is organized as follows. Section 2 briefly introduces IEEE 802.15.4 and ZigBee. The convergecast problem is formally defined in Section 3. Section 4 presents our convergecast algorithms. Simulation results are given in Section 5. Finally, Section 6 concludes this paper.

2. OVERVIEW OF IEEE 802.15.4 AND ZIGBEE STANDARDS

IEEE 802.15.4 [7] specifies the physical and data link protocols for *low-rate wireless personal area networks (LR-WPAN)*. In the physical layer, there are three frequency bands with 27 radio channels. Channel 0 ranges from 868.0 MHz to 868.6 MHz, which provides a data rate of 20 kbps. Channels 1 to 10 work from 902.0 MHz to 928.0 MHz and each channel provides a data rate of 40 kbps. Channels 11 to 26 are located from 2.4 GHz to 2.4835 GHz, each with a data rate of 250 kbps.

IEEE 802.15.4 devices are expected to have limited power, but need to operate for a longer period of time. Therefore, energy conservation is a critical issue. Devices are classified as *full function devices* (FFDs) and *reduced function devices* (RFDs). IEEE 802.15.4 supports star and peer-to-peer topologies. In each PAN, one device is designated as the *coordinator*, which is responsible for maintaining the network. A FFD has the capability of becoming a coordinator or associating with an existing coordinator. A RFD can only associate with a coordinator.

ZigBee defines the communication protocols above IEEE 802.15.4. In its version 1.0, star, tree, and mesh topologies are supported. A ZigBee coordinator is responsible for initializing, maintaining, and controlling the network. In a star network, devices must directly connect to the coordinator. For tree and mesh networks, devices can communicate with each other in a multihop fashion. The network backbone is formed by one ZigBee coordinator and multiple ZigBee routers (which must be 802.15.4 FFDs). RFDs can only join the network as end devices by associating with the ZigBee coordinator or ZigBee routers. In a tree network, the coordinator and routers can announce beacons. However, in a mesh network, regular beacons are not allowed. Beacons are an important mechanism to support power management. Therefore, the tree topology is preferred, especially when energy saving is a desired feature.

The ZigBee coordinator defines the superframe structure of a network. As shown in Fig. 2, the structure of superframes is controlled by two parameters: *beacon order* (BO) and *superframe order* (SO), which decide the length of a superframe and its active

Table 1: Relationship of $BO - SO$, duty cycle, and the number of active portions in a superframe.

$BO - SO$	0	1	2	3	4	5	6	7	8	9	≥ 10
Duty cycle (%)	100	50	25	12.5	6.25	3.13	1.56	0.78	0.39	0.195	< 0.1
Number of active portions (slots)	1	2	4	8	16	32	64	128	256	512	≥ 1024

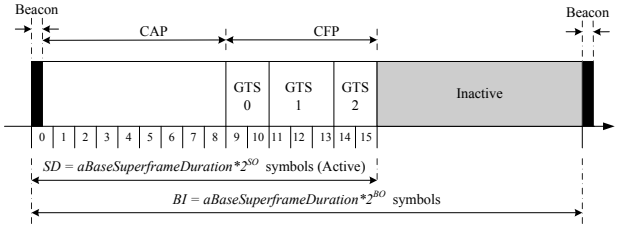


Figure 2: IEEE 804.15.4 superframe structure.

potion, respectively. For a beacon-enabled network, the setting of BO and SO should satisfy the relationship $0 \leq SO \leq BO \leq 14$. A non-beacon-enabled network should set $BO = SO = 15$ to indicate that superframes do not exist. Each active portion consists of 16 equal-length slots and can be further partitioned into a *contention access period* (CAP) and a *contention free period* (CFP). The CAP may contain the first i slots and the rest of the $16 - i$ slots belong to the CFP, where $1 \leq i \leq 16$. Slotted CSMA/CA is used in CAP. FFDs which require fixed transmission rates can ask for *guarantee time slots (GTSs)* from the coordinator. A CFP can support multiple GTSs, and each GTS may contain multiple slots. After the active portion, devices can go to sleep to save energy.

Since a device only needs to be active for $2^{-(BO-SO)}$ portion of the time, changing the value of $(BO - SO)$ allows us to adjust the on-duty time of devices. In a beacon-enabled network, routers do not have to choose the same time to start their active portions (and thus their superframes). Once the value of $(BO - SO)$ is decided, each router can choose one of the 2^{BO-SO} slots to send its beacon and start its active portion. Neighboring routers' active portions should be shifted away from each other to avoid interference. This work is motivated by the observation that the specification does not clearly define how to choose locations of routers' active portions to reduce convergecast latency. In our work, we consider two kinds of interference between routers. Two routers have *direct interference* if they can hear each others's beacons. Two routers have *indirect interference* if they have at least one common neighbor. Both interferences should be avoided when choosing routers' active portions. Table 1 lists possible choices of $(BO - SO)$ combinations.

3. THE MINIMUM DELAY BEACON SCHEDULING (MDBS) PROBLEM

This section formally defines the convergecast problem in ZigBee network. Given a ZigBee network, we model it by a graph $G = (V, E)$, where V contains all routers and the coordinator and E contains all symmetric communication links between nodes in V. The coordinator also serves as the sink of the network. End devices can only associate with routers, but are not included in V. From G, we can construct an *interference graph* $G_I = (V, E_I)$, where edge $(i, j) \in E_I$ if there are direct/indirect interferences between i and j. Also, there is a duty cycle requirement α for this network. From α and Table 1, we can determine the most appropriate value of $BO - SO$. We denote by $k = 2^{BO-SO}$ the number of active portions (or slots) per superframe. The beacon scheduling problem is to find a slot assignment $s(i)$ for each router $i \in V$, where $s(i)$ is an integer and $s(i) \in [0, k-1]$, such that router i's active portion is in slot $s(i)$ and $s(i) \neq s(j)$ if $(i, j) \in E_I$. Motivated by Brook's theorem [14], which proves that n colors are sufficient to color any graph with a maximum degree of n, we would assume that $k \geq D_I$, where D_I is the maximum degree of G_I.

Given a slot assignment for G, the report latency from node i to node j, where $(i, j) \in E$, is the number of slots, denoted by d_{ij}, that node i has to wait to relay its collected sensory data to node j, i.e.,

$$d_{ij} = (s(j) - s(i)) \bmod k. \quad (1)$$

Note that the report latency from node i to node j (d_{ij}) may not by equal to the report latency from node j to node i (d_{ji}). Therefore, we can convert G into a weighted directed graph $G_D = (V, E_D)$ such that each $(i, j) \in E$ is translated into two directed edges (i, j) and (j, i) such that $w((i, j)) = d_{ij}$ and $w((j, i)) = d_{ji}$. The report latency for each $i \in V$ to the sink is the sum of report latencies of the links on the shortest path from i to the sink in G_D. The latency of the convergecast, denoted as $L(G)$, is the maximum of all nodes' report latencies.

DEFINITION 1. *Given $G = (V, E)$, G's interference graph $G_I = (V, E_I)$, and k available slots, the* Minimum Delay Beacon Scheduling (MDBS) problem *is to find an interference-free slot assignment $s(i)$ for each $i \in V$ such that the convergecast latency $L(G)$ is minimized.*

To prove that the MDBS problem is NP-complete, we define a decision problem as follows.

DEFINITION 2. *Given $G = (V, E)$, G's interference graph $G_I = (V, E_I)$, k available slots, and a delay constraint d, the* Bounded Delay Beacon Scheduling (BDBS) problem *is to decide if there exists an interference-free slot assignment for each $i \in V$ such that the convergecast latency $L(G) \leq d$.*

THEOREM 1. *The BDBS problem is NP-complete.*

PROOF. First, given slot assignments for nodes in V, we can find the report latency of each $i \in V$ by running a shortest path algorithm on G_D. We can then check if $L(G) \leq d$. Clearly, this takes polynomial time.

We then prove that the BDBS problem is NP-hard by reducing the 3 conjunctive normal form satisfiability (3-CNF-SAT) problem to a special case of the BDBS problem in polynomial time. Given any 3-CNF formula C, we will construct the corresponding G and G_I. Then we show that C is satisfiable *if and only if* there is a slot assignment for each $i \in V$ using no more than $k = 3$ slots such that $L(G) \leq 4$ slots.

Let $C = C_1 \wedge C_2 \wedge \cdots \wedge C_m$, where clause $C_j = x_{j,1} \vee x_{j,2} \vee x_{j,3}$, $1 \leq j \leq m$, $x_{j,i} \in \{X_1, X_2, ..., X_n\}$, and $X_i \in \{x_i, \bar{x}_i\}$, where x_i is a binary variable, $1 \leq i \leq n$. We first construct G from C as follows:

1. For each clause C_j, $j = 1, 2, ..., m$, add a vertex C_j in G.
2. For each literal X_i, $i = 1, 2, ..., n$, add four vertices x_{i1}, x_{i2}, $\bar{x}_{i1}$, and $\bar{x}_{i2}$ in G.

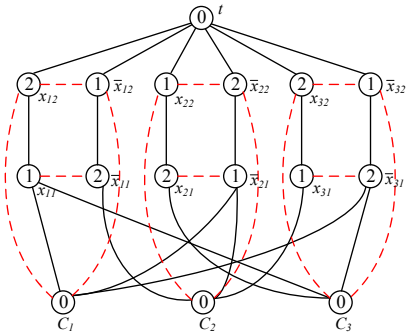


Figure 3: An example of reduction from the 3-CNF-SAT to the BDBS problem.

3. Add a vertex t as the sink of G.
4. Add edges (t, x_{i2}) and $(t, \bar{x}_{i2})$ to G, for $i = 1, 2, ..., n$.
5. Add edges (x_{i1}, x_{i2}) and $(\bar{x}_{i1}, \bar{x}_{i2})$ to G, for $i = 1, 2, ..., n$.
6. For each $i = 1, 2, ..., n$ and each $j = 1, 2, ..., m$, add an edge (C_j, x_{i1}) (resp., $(C_j, \bar{x}_{i1})$) to G if x_i (resp., $\bar{x}_i$) appears in C_j.

Then we construct G_I as follows.

1. Add all vertices and edges in G into G_I.
2. Add edges $(x_{i1}, \bar{x}_{i1})$ and $(x_{i2}, \bar{x}_{i2})$ to G_I, for $i = 1, 2, ..., n$.
3. Add edges (C_j, x_{i2}) and $(C_j, \bar{x}_{i2})$ to G_I, for $i = 1, 2, ..., n$ and $j = 1, 2, ..., m$.

Then we build a one-to-one mapping from each truth assignment of C to a slot assignment of G. We establish the following mapping:

1. Set $s(t) = 0$.
2. Set $s(C_j) = 0$, $j = 1, 2, ..., m$.
3. Set $s(x_{i1}) = 1$ and $s(\bar{x}_{i2}) = 1$, $i = 1, 2, ..., n$, if x_i is true; otherwise, set $s(x_{i1}) = 2$ and $s(\bar{x}_{i2}) = 2$.
4. Set $s(x_{i2}) = 1$ and $s(\bar{x}_{i1}) = 1$, $i = 1, 2, ..., n$, if $\bar{x}_i$ is true; otherwise, set $s(x_{i2}) = 2$ and $s(\bar{x}_{i1}) = 2$.

The above reduction can be computed in polynomial time. By the above reduction, vertices x_{i1} or $\bar{x}_{i1}$, $i = 1, 2, ..., n$, that are assigned to slot 1 (resp. slot 2) will have a report latency of 2 (resp. 4) and vertices x_{i2} or $\bar{x}_{i2}$, $i = 1, 2, ..., n$, that are assigned to slot 1 (resp. slot 2) will have a report latency of 2 (resp. 1). Hence, for those vertices x_{i1}, $\bar{x}_{i1}$, x_{i2}, and $\bar{x}_{i2}$, $i = 1, 2, ..., n$, the longest report latency will be 4.

To prove the *if* part, we need to show that if C is satisfiable, there is a slot assignment such that $k = 3$ and $L(G) \leq 4$. Since C satisfiable, there must exist an assignment such that each clause C_j, $j = 1, 2, ..., m$, is true. If a clause C_j is true, at least one variable in C_j is true. According to the reduction, C_j can always find an edge (C_j, x_{i1}) or $(C_j, \bar{x}_{i1})$ with $w((C_j, x_{i1})) = 1$ or $w((C_j, \bar{x}_{i1})) = 1$, where $i = 1, 2, ..., n$. Thus, when C is satisfiable, the reporting latency for each clause is 3. This achieves $L(G) = 4$.

For the *only if* part, if each vertex C_j, $j = 1, 2, ..., m$, can find at least an edge with weight 1 to one of x_{i1} and $\bar{x}_{i1}$, for $i = 1, 2, ..., n$, to achieve a report latency of 3, it must be that each clause has at least one variable to be true. So formula C is satisfiable. Otherwise, the report latency of C_j, $j = 1, 2, ..., m$, will be 6. □

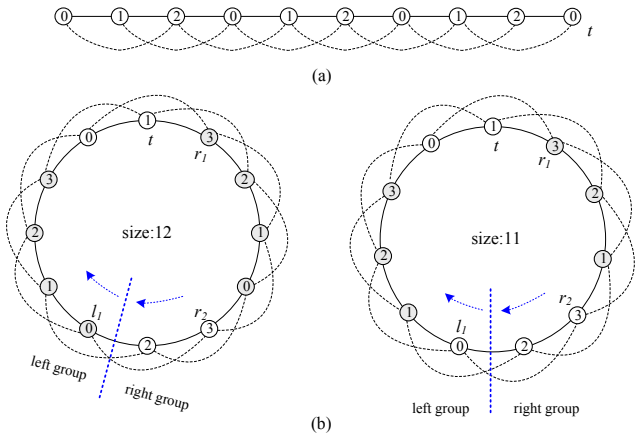


Figure 4: Examples of optimal slot assignments for regular linear and ring networks ($h = 2$). Dotted lines mean interference relations.

For example, given $C = (x_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_3) \wedge (x_1 \vee x_2 \vee \bar{x}_3)$, Fig. 3 shows the corresponding G. The truth assignment $(x_1, x_2, x_3) = (T, F, T)$ makes C satisfiable. According to the reduction and the mapping in the above proof, we can obtain the network G and its slot assignment as shown in Fig. 3 such that $L(G) = 4$.

4. ALGORITHMS FOR THE MDBS PROBLEM

4.1 Optimal Solutions for Special Cases

Optimal solutions can be found for the MDBS problem in polynomial time for regular linear networks and regular ring networks, as illustrated in Fig. 4. In such networks, each vertex is connected to one or two adjacent vertices and has an interference relation with each neighbor within h hops from it, where $h \geq 2$. In a regular linear network, we assume that the sink t is at one end of the network. Clearly, the maximum degree of G_I is $2h$. We will show that an optimal solution can be found if the number of slots $k \geq h + 1$. The slot assignment can be done in a bottom-up manner. The bottom node is assigned to slot 0. Then, for each vertex v, $s(v) = (k' + 1) \bmod k$, where k' is the slot assigned to v's child.

THEOREM 2. *For a regular linear network, if $k \geq h + 1$, the above slot assignment achieves a report latency of $|V| - 1$, which is optimal.*

PROOF. Clearly, the slot assignment is interference-free. Also the report latency of $|V| - 1$ is clearly the lower bound. □

For a regular ring network, we first partition vertices excluding t into left and right groups as illustrated in Fig. 4(b) such that the left group consists of the sink node t and $\lfloor \frac{|V|-1}{2} \rfloor$ other nodes counting counter-clockwise from t, and the right group consists of those $\lceil \frac{|V|-1}{2} \rceil$ nodes counting clockwise from t. Now we consider the ring as a spanning tree with t as the root and left and right groups as two linear paths. Assuming that $\lfloor \frac{|V|-1}{2} \rfloor \geq 2h$ and $k \geq 2h$, the slot assignment works as follows:

1. The bottom node in the left group is assigned to slot 0.

2. All other nodes in the left group are assigned with slots in a bottom-up manner. For each node i in the left group, we let $s(i) = (j + 1) \bmod k$, where j is the slot of i's child.

3. Nodes in the right group are assigned with slots in a top-down manner. For each node i in the right group, we let $s(i) = (j - c) \bmod k$, where j is the slot assigned to i's parent and c is the smallest constant ($1 \leq c \leq k$) that ensures that $s(i)$ is not used by any of its interference neighbors that have been assigned with slots.

It is not hard to prove the slot assignment is interference-free because nodes receives slots sequentially and we have avoided using the same slots among interfering neighbors. Although this is a greedy approach, we show that c is equal to 1 in step 3 in most of the cases except when two special nodes are visited. This gives an asymptotically optimal algorithm, as proved in the following theorem.

THEOREM 3. *For a regular ring network, assuming that $k \geq 2h$ and $\lfloor \frac{|V|-1}{2} \rfloor \geq 2h$, the above slot assignment achieves a report latency $L(G) = \lfloor \frac{|V|-1}{2} \rfloor + h$, which is optimal within a factor of 1.5.*

PROOF. We first identify three nodes on the ring (refer to Fig. 4(b)):

- l_1: the last node in the left group.
- r_1: the first node in the right group.
- r_2: the node that is h hops from l_1 counting clockwise.

The report latency of each node can be analyzed as follows. The parent of node x is denoted by $par(x)$.

A1. For each node i in the left group except the sink t, the latency from i to $par(i)$ is 1.

A2. The latency from r_1 to t is h.

A3. For each node i next to r_1 in the right group but before r_2 (counting clockwise), the latency from i to $par(i)$ is 1.

A4. The latency from r_2 to $par(r_2)$ is 1 if the ring size is even; otherwise, the latency is 2.

A5. For each node i in the right group that is a descendant of r_2, the report latency from i to $par(i)$ is 1.

It is not hard to prove that A1, A2, and A3 are true. To see A4 and A5, we make the following observations. The function $par^i(x)$ is to apply i times the $par()$ function on node x. Note that $par^0(x)$ means x itself.

O1. When the ring size is even, the equality $s(par^{i-1}(l_1)) = s(par^i(r_2))$ holds for $i = 1, 2, ..., \lfloor \frac{|V|-1}{2} \rfloor - h - 1$. More specifically, this means that (i) l_1 and $par(r_2)$ will receive the same slot, (ii) $par(l_1)$ and $par^2(r_2)$ will receive the same slot, etc. This can be proved by induction by showing that the i-th descendant of t in the right group will be assigned the same slot as the $(h + i - 1)$-th descendant of t in the left group (the induction can go in a top-down manner). This property implies that when assigning a slot to r_2 in step 3, $c = 1$ in case that the ring size is even. Further, the descendant of r_2 will be sequentially assigned to slots $k - 1$, $k - 2$, ..., $k - h$, which implies that $c = 1$ when doing the assignments in step 3. So properties A4 and A5 hold for the case of an even ring.

O2. When the ring size is odd, the equality $s(par^i(l_1)) = s(par^i(r_2))$ holds for $i = 1, 2, ..., \lfloor \frac{|V|-1}{2} \rfloor - h$. This means that (i) $par(l_1)$ and $par(r_2)$ will receive the same slot, and (ii) $par^2(l_1)$ and $par^2(r_2)$ will receive the same slot, etc. Again, this can be proved by induction as in O1. This property implies that $c = 2$ when assigning a slot to r_2 in step 3, and $c = 1$ when assigning slots to descendants of r_2. So properties $A4$ and $A5$ hold for the case of an odd ring.

The equality of slot assignments pointed out in O1 and O2 is illustrated in Fig. 4(b) by those numbers in gray nodes. In summary, the report latency of the left group is $\lfloor \frac{|V|-1}{2} \rfloor$. When the ring size is even, the report latency of the right group is the number of nodes in this group, $\frac{|V|}{2}$, plus the extra latency $h - 1$ incurred at r_1. So $L(G) = \frac{|V|}{2} + h - 1 = \lfloor \frac{|V|-1}{2} \rfloor + h$. When the ring size is odd, the report latency of right group is the number of nodes in this group, $\frac{|V|-1}{2}$, plus the extra latency $h - 1$ incurred at r_1 and the extra latency 1 incurred at r_2. So $L(G) = \lfloor \frac{|V|-1}{2} \rfloor + h$.

A lower bound on the report latency of this problem is the maximum number of nodes in each group excluding t. Applying $\lfloor \frac{|V|-1}{2} \rfloor$ as a lower bound and using the fact that $\lfloor \frac{|V|-1}{2} \rfloor \geq 2h$, the algorithm is optimal within a factor of 1.5. Note that the condition $\lfloor \frac{|V|-1}{2} \rfloor \geq 2h$ is to guarantee that t will not locate within h hops from r_2. Otherwise, the observation O2 will not hold. □

4.2 A Centralized Tree-Based Assignment

Given $G = (V, E)$, $G_I = (V, E_I)$, and k, we propose a centralized slot assignment heuristic algorithm. Our algorithm is composed of the following three phases:

phase 1. From G, we first construct a BFS tree T rooted at sink t.

phase 2. We traverse vertices of T in a bottom-up manner. For each vertex v visited, we first compute a temporary slot number $t(v)$ for v as follows.

1. If v is a leaf node, we set $t(v)$ to the minimal non-negative integer l such that for each vertex u that has been visited and $(u, v) \in E_I$, $(t(u) \bmod k) \neq l$.
2. If v is an in-tree node, let m be the maximum of the numbers that have been assigned to v's children, i.e., $m = \max\{t(child(v))\}$, where $child(v)$ is the set of v's children. We then set $t(v)$ to the minimal non-negative integer l such that for each vertex u that has been visited and $(u, v) \in E_I$, $(t(u) \bmod k) \neq (l \bmod k)$.

After every vertex v is visited, we make the assignment $s(v) = t(v) \bmod k$.

phase 3. In this phase, vertices are traversed sequentially from t in a top-down manner. When each vertex v is visited, we try to greedily find a new slot l such that $(s(par(v)) - l) \bmod k < (s(par(v)) - s(v)) \bmod k$, such that $l \neq s(u)$ for each $(u, v) \in E_I$, if possible. Then we reassign $s(v) = l$.

Note that in phase 2, the number $t(v)$ is not a modulus number. However, we will check that if $t(v)$ is converted to a slot number, no interference will occur. Intuitively, this is a temporary slot assignment that will incur the least latency to v's children. At the end, $t(v)$ is converted to a slot assignment $s(v)$. Phase 3 is a greedy approach to further reduce $L(G)$. For example, Fig. 5(a) shows the slot assignment after phase 2, which induces a $L(G)$ of 6. Fig. 5(b)

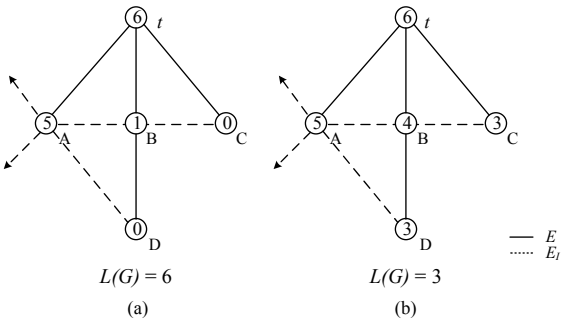


Figure 5: (a) Slot assignment after phase 2. (b) Slot compacting by phase 3.

indicates that C and D can find another slots and $L(G)$ is decreased to 3.

The computational complexity of this algorithm is analyzed below. In phase 1, the complexity of constructing a BFS tree is $O(|V| + |E|)$. In phase 2, the computational cost to compute $t(v)$ for each vertex v is bounded by $O(kD_I)$, where D_I is the degree of G_I. So the time complexity of phase 2 is $O(kD_I|V|)$. Phase 3 performs a similar procedure as phase 2, so its time complexity is also $O(kD_I|V|)$. Overall, the time complexity is $O(kD_I|V|)$.

4.3 A Distributed Slot Assignment

In the following, we propose a distributed slot assignment algorithm. Each node has to compute its direct as well as indirect interference neighbors in a distributed manner. To achieve this, we will refer to the *heterogeneity* approach in [15]. From time-to-time, each router will boost its transmission power to double its default transmission range and exchanges HELLO packets, containing its slot information with its interference neighbors. Note that all other packets are transmitted by the default power level. When booting up, each router will broadcast HELLO packets claiming that its current slot is *NULL*. After receiving a slot the HELLO packets will carry the node's slot assignment. The algorithm is triggered by the sink t setting $s(t) = 0$ and then broadcasting its beacon. A router $v \neq t$ that receives a beacon will find itself a slot as follows.

1. Node v sends an association request to the beacon sender.
2. If v fails to associate with the beacon sender, it stops the procedure and waits for other beacons.
3. If v successfully associates with a parent node $par(v)$, it computes the smallest positive integer l such that $(s(par(v)) - l) \bmod k \neq s(u)$ for all $(u, v) \in E_I$ and $s(u) \neq NULL$. Then v chooses $s(v) = (s(par(v)) - l) \bmod k$ as its slot.
4. Then, v broadcasts HELLOs including its slot assignment $s(v)$ for a time period t_{wait}. If it finds that $s(v) = s(u)$ for any $(u, v) \in E_I$ such that u's ID is larger than v's ID, then v has to choose another slot assignment and going back to step 3.
5. After t_{wait}, v can finalize its slot selection and broadcast its beacons.

In this distributed algorithm, slots are assigned to routers, ideally, in a top-down manner. However, due to transmission latency, some routers at lower levels may find slots earlier than those at higher levels. Also note that step 4 is to correct possible collision on slot assignments due to packet loss.

5. SIMULATION RESULTS

We compare the proposed slot assignment algorithms against a random slot assignment (RAN) scheme. The slot assignment starts from the sink and each router, after associating with a parent router, simply chooses any slot which has not been used by any of its interference neighbors. In the following simulations, routers are randomly distributed in a square region and a sink is placed in the center of the network. Each router can have at most 7 children. Our centralized tree-based scheme and distributed slot assignment scheme are denoted as CTB and DSA, respectively. We compare the report latency $L(G)$ (in unit of active portions).

In the following two simulations, we set $k = 64$. Fig. 6(a) shows the results when the transmission range of routers is set to 25 meters and $(n/10)^2$ routers are generated in an $n \times n$ square region, where n ranges from $60m$ to $240m$. CTB always performs the best. DSA performs slight worse than CTB, but still significantly outperforms RAN. It can be seen that a random slot assignment could result in very long convergecast latency. Both CTB and DSA are quite insensitive to the network size. But this is not the case for RAN. Next, we simulate a $200m \times 200m$ network and place 400 routers in it. Fig. 6(b) shows the results when we vary the transmission range. Since a larger transmission range implies higher interference among routers, the report latencies of CTB and DSA will increase slightly as the transmission range increases. However, RAN will benefit from a larger transmission range because it is more sensitive to the network diameter than other factors.

Next, with a network size of $200m \times 200m$ and a router transmission range of $20m$, and $k = 128$, we vary the number of routers in the network. As Fig. 6(c) shows, when there are more and more routers, the report latencies of CTB and DSA will sightly increase because the interference among routers will increase. But the report latency of RAN will drop first, and then increase slightly as there are more routers. This is because more routers will decrease the network diameter first, but since we limit the number of children per router to 7, the network diameter can not keep on decreasing after a certain number of routers. In Fig. 6(d), we fix the number of routers to 400 and vary routers' duty cycle. Note that a lower duty cycle means a larger number of available slots. Interestingly, we see that the report latencies of CTB and DSA are independent of the number of slots. Contrarily, with a random assignment, RAN even incurs a higher report latency as there are more freedom in slot selection.

6. CONCLUSIONS

In this paper, we have defined a new minimum delay beacon scheduling (MDBS) problem for convergecast with the restrictions that the beacon scheduling must be compliant to the ZigBee/IEEE 802.15.4 standards. We prove the MDBS problem is NP-complete and propose optimal solutions for special cases and two heuristic algorithms for general cases. Simulation results indicate the performance of our heuristic algorithms decrease only when the number of interference neighbors is increased. Compared to the random slot assignment scheme, our heuristic algorithms can effectively schedule the ZigBee routers' beacon times to achieve quick convergecast.

7. ACKNOWLEDGEMENTS

Y.-C. Tseng's research is co-sponsored by NSC under grant numbers 93-2752-E-007-001-PAE, 95-2623-7-009-010-ET, 95-2218-E-009-020, 95-2219-E-009-007, 94-2213-E-009-004, and 94-2219-E-007-009, by MOEA under grant number 94-EC-17-A-04-S1-044, by ITRI, Taiwan, and by Intel Inc.

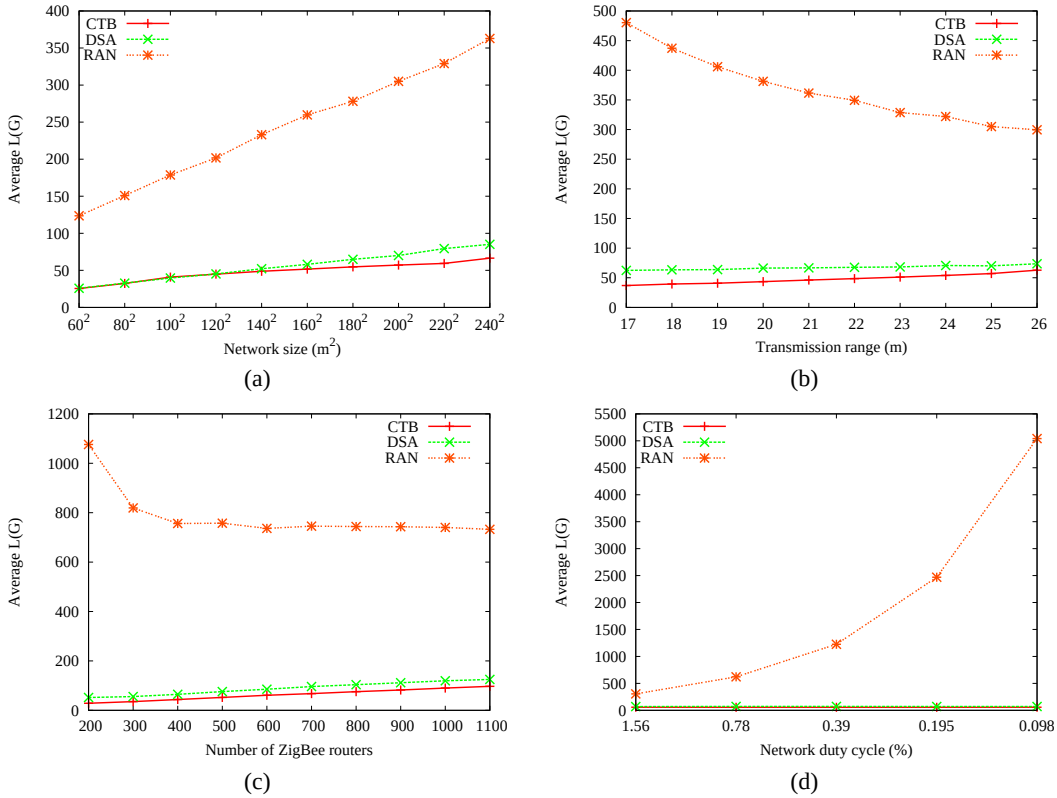


Figure 6: Simulation results on the average report latencies under different configurations.

8. REFERENCES

[1] Design and construction of a wildfire instrumentation system using networked sensors. http://firebug.sourceforge.net/.

[2] Dust network Inc. http://dust-inc.com/flash-index.shtml.

[3] Habitat monitoring on great duck island. http://www.greatduckisland.net/technology.php.

[4] Motes, smart dust sensors, wireless sensor networks. http://www.xbow.com/Products/productsdetails.aspx?sid=3.

[5] Zigbee alliance. http://www.zigbee.org/.

[6] B. Hohlt, L. Doherty, and E. Brewer. Flexible power scheduling for sensor networks. In *Proc. of Int'l Symp. on Information Processing in Sensor Networks (IPSN)*, 2004.

[7] 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), 2003.

[8] Q. Li, M. DeRosa, and D. Rus. Distributed algorithm for guiding navigation across a sensor network. In *Proc. of ACM Int'l Symp. on Mobile Ad Hoc Networking and Computing (MobiHOC)*, 2003.

[9] C.-Y. Lin and Y.-C. Tseng. Structures for in-network moving object tracking in wireless sensor networks. In *Proc. of Broadband Wireless Networking Symp. (BroadNet)*, 2004.

[10] G. Lu, B. Krishnamachari, and C. S. Raghavendra. An adaptive energy-efficient and low-latency mac for data gathering in wireless sensor networks. In *Proceedings of Int'l Parallel and Distributed Processing Symp.*, 2004.

[11] Y.-C. Tseng, S.-P. Kuo, H.-W. Lee, and C.-F. Huang. Location tracking in a wireless sensor network by mobile agents and its data fusion strategies. In *Proc. of Int'l Symp. on Information Processing in Sensor Networks (IPSN)*, 2003.

[12] Y.-C. Tseng, M.-S. Pan, and Y.-Y. Tsai. Wireless sensor networks for emergency navigation. volume 39, pages 55–62, 2006.

[13] S. Upadhyayula, V. Annamalai, and S. K. S. Gupta. A low-latency and energy-efficient algorithm for convergecast in wireless sensor networks. In *Proc. of IEEE Global Telecommunications Conference (Globecom)*, 2003.

[14] D. B. West. *Introduction to Graph Theory*. Prentice Hall, 2001.

[15] M. Yarvis, N. Kushalnagar, H. Singh, A. Rangarajan, Y. Liu, and S. Singh. Exploiting heterogeneity in sensor networks. In *Proc. of IEEE INFOCOM*, 2005.

[16] Y. Yu, B. Krishnamachari, and V. K. Prasanna. Energy-latency tradeoffs for data gathering in wireless sensor networks. In *Proc. of IEEE INFOCOM*, 2004.