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：2010 年 08 月 01 日至 2011 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　100 年　10 月　28 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。在第三年中，我們探討了一個新的網路型態：長鏈狀網路(Long-Thin Network)。在這類的網路中，節點們由幾個網路的骨幹(Backbone)所連結，而這些骨幹則由幾個連接點給連接住進而散佈到整個所要監控檢測之區域。乍看之下，此網路型態於無線感測網路中似乎不常看見，然而，於實際中，我們可以發現此種網路型態遍佈於各式各樣的現實應用中。同時，我們發現，於長鏈狀網路下，ZigBee 標準所規範之網路生成及資料匯集協定皆不適用且擁有極差之效果，因此，我們重新設計了一符合ZigBee規範之網路生成協定，並設計了一資料匯集演算法於此網路型態下。同時，我們進一步的將此成果以實作成品展現，並可看出我們方法所帶來的實際效益。

關鍵字：位址分派、長鏈狀網路、資料收集、無線感測器網路、ZigBee

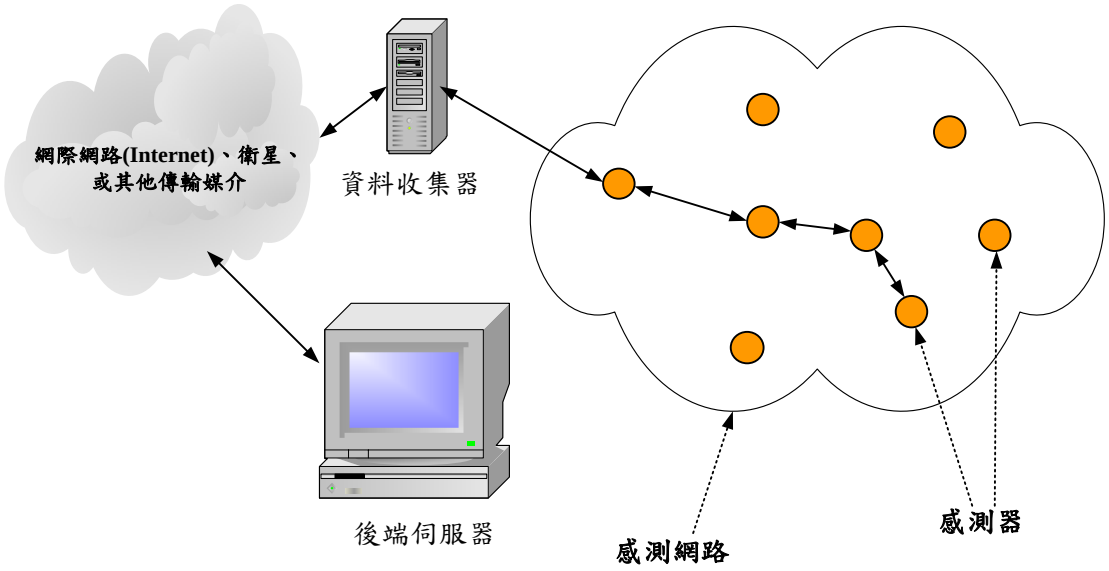
Recently, a lot of research works have been dedicated to the *wireless sensor networks (WSNs)* field. ZigBee is a communication standard which is considered to be suitable for WSNs. In this project, we discuss initialization and communication protocols for ZigBee tree-based WSNs. This project contains 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 last year, we promote a new concept of *long-thin (LT) topology* for WSNs, where a network may have a number of *linear paths* of nodes as backbones connecting to each other. These backbones are to extend the network to the intended coverage areas. At the first glance, a LT WSN only seems to be a special case of numerous WSN topologies. However, we observe, from real deployment experiments, that such a topology is quite general in many applications and deployments. We show that the *address assignment* and thus the *data aggregation scheme* defined in the original ZigBee specification may work poorly, if not fail, in a LT topology. We thus propose simple, yet efficient, address assignment and routing schemes for a LT WSN. Implementation results and prototyping experiences are also reported.

Keywords: address assignment, long-thin network, aggregation, wireless sensor network, ZigBee

目錄

一、 前言

近年來，隨著無線通訊、電池技術與嵌入式微處理器技術的進步與整合，帶動了無線感測器網路(Wireless Sensor Networks, WSNs)的發展。無線感測器網路是由許多的感測器節點(Sensor node)所組成，這些節點能夠在其感測範圍(Sensing range)內偵測目前的環境變數，例如：溫度、光度、濕度及磁場等。除此之外，節點們也能與其傳輸範圍(Communication rage)內的鄰居節點，透過無線通訊方式來溝通，並且也能把收集到的資訊使用多點跳躍的方式回報給資料收集端(Sink or Data collector)，接著網路管理者可分析感測器所回報的資料用於特定的應用，如環境監控、軍事或醫療用途等，圖一為無線感測器網路的基本架構圖。



圖一、無線感測器網路基本架構圖。

在許多無線感測網路之應用中，網路建置者需要將網路部署成一長鏈狀之網路。長鏈狀無線網路是由一些擁有線性路徑節點的骨幹連接彼此所形成，換句話說，此網路多數之網路節點以類似於長鏈方式連結，少部分之節點可與其他長鏈上之節點連接，作為長鏈間溝通之橋樑。長鏈狀網路型態常見於無線感測網路監測(Surveillance)應用中，例如圖二(a)為一個瓦斯管線偵測漏氣應用，圖二(b)為隧道內二氧化碳濃度偵測應用，圖二(c)為下水道水位偵測應用，圖二(d)為高速公路路燈或車況監控應用，圖二(e)為河流防洪水位監控，圖二(f)為橋樑震動監控應用。ZigBee 基礎之長鏈狀網路研究，其目的為探討該如何將ZigBee協定用於此一特殊但卻常見之樹狀拓樸中。在這一部份的計畫中，我們將探討一個特殊但是卻又常見之網路拓樸—長鏈狀網路(Long Thin network, LT)。

在這類的網路中，節點們由幾個網路的骨幹(Backbone)所連結，而這些骨幹則由幾個連接點給連接住 進而散佈到整個所要監控檢測之區域，每一個骨幹為一個線性的網路拓樸，並且可能包含數十甚至數百個節點，網路的範圍相當的大。因此，本計畫將以網路層的角度來切入研究適用於長鏈狀無線感測器網路的通訊協定，以下以不同層次的面向來討論：

- 網路生成(Formation)：在建置長鏈狀網路時，一個最重要的課題是我們該如何形成這

圖二、長鏈狀網路(Long-Thin Network)範例。

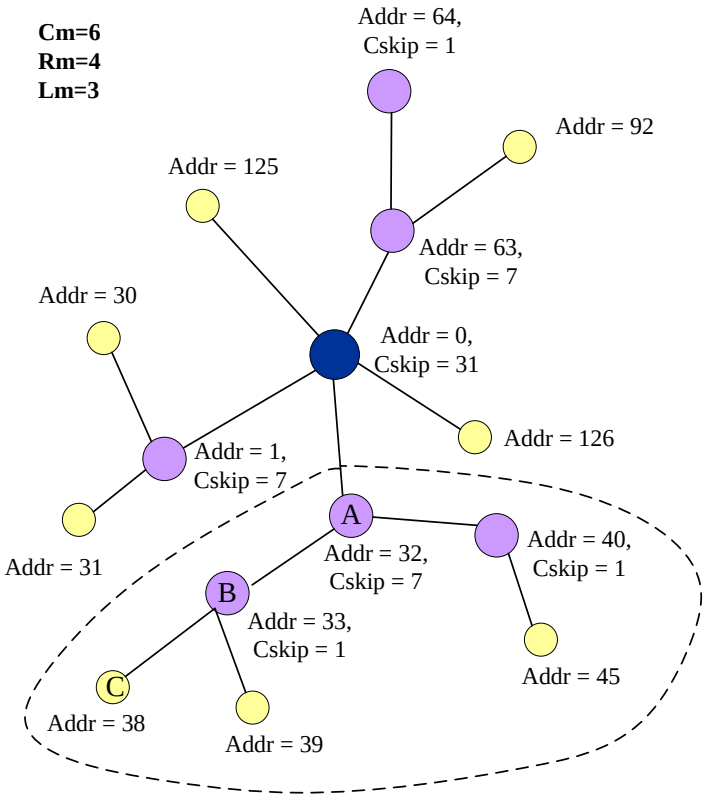
個網路。一個不佳的網路形成方式可能會使得網路無法真正成形，並且會造成網路斷點情況發生，我們觀察到ZigBee 網路生成方式並不能適用於長鏈狀網路，且因為ZigBee網路層規定網路位址只能有16 位元，為了更有效利用有限的網路位址，我們也需要一個新的位址分派演算法來解決此問題。

- 路由協定(Routing)：由於長鏈狀網路之網路拓樸，在網路節點的鄰居數太少使得現存的路由協定皆不適合使用於長鏈狀網路拓樸。

- 資料匯集及資料壓縮：在長鍊狀網路上若沒有適當的資料匯集及資料壓縮的機制，龐大的資料量會造成許多的問題，例如耗電以及因為高競爭而發生的封包遺失等，雖然在無線感測器網路上資料排程、資料匯集及資料壓縮的相關論文已被廣泛地討論，但是就我們所知並沒有相關的論文探討長鏈狀網路的上的資料匯集及資料壓縮。

因此，我們將以網路層的角度來切入研究適用於長鏈狀的通訊協定，首先我們將設計適用於長鏈狀網路之網路生成協定，並且提出用於長鏈狀網路之低負載及省電路由協定。現今沒有相關的文獻在探討長鏈狀網路型態，相信本計畫之成果可為ZigBee 基礎長鏈狀網路之研究尖端。

二、 背景知識



圖三、 ZigBee 網路位址分配。

- ZigBee 網路生成

在網路一啟動時，網路中的 FFD 們競爭成為 ZigBee 協調者，假定一裝置成功地成為協調者，該協調者將掃描所有的無線頻道並且決定一個合適的作為操作頻道，接著該協調者開始廣播 Beacon 訊框來讓其他的裝置能夠加入它所形成的網路。當裝置接收到一 Beacon 訊框，該裝置執行加入網路的步驟成為 ZigBee 路由器或者作為一個 ZigBee 末端設備。如果成為一 ZigBee 路由器，此路由器也能夠發送 Beacon 訊框來讓尚未加入網路的裝置加入此網路。當裝置成功加入一個網路後 Beacon 訊框的傳送者將會給定一網路位址，作為日後資料傳輸之識別碼。

ZigBee 網路層協定中規範了一分散式網路位址分配演算法用以分配網路位址給加入該網路之裝置。當網路形成時，ZigBee 協調者先定義一 ZigBee 路由器最多可容許連線之裝置個數(*Cm*)以及最多的子 ZigBee 路由器的數量(*Rm*)，以及網路的深度(*Lm*)。ZigBee 規定 Cm≧Rm，因此一 ZigBee 路由器至少可供(*Cm-Rm*) ZigBee 終端設備連結上它。在此規範中，裝置的網路位址是由其父節點(Parent router)所給定的。對 ZigBee 協調者而言，整個網路的位址空間被劃分成 *Rm*+1 塊，前 *Rm* 塊位址空間將會分配給其 Rm 個子路由器，而最後一部份則保留給與之連線之(*Cm-Rm*)個 ZigBee 終端設備。在此演算法中，ZigBee 路由器利用 *Cm*、*Rm* 和 *Lm* 來計算一個稱為 *Cskip* 的參數，接著再利用 *Cskip* 來計算其子路由器以及終端設備的網路位址，假定一路由器位於網路的第 *d* 層，*Cskip* 的數值可經由下式得到：

$$Cskip\ (d) = \begin{cases} 1 + Cm \cdot (Lm - d - 1), & \text{if } Rm = 1 \quad \text{(a)} \\ \dfrac{1 + Cm - Rm - Cm \cdot Rm^{\,Lm-d-1}}{1 - Rm}, & \text{Otherwise} \quad \text{(b)} \end{cases} \tag{1}$$

ZigBee 規範協調者位於深度 0，假設一路由器 *x* 位於深度 *d* 且一節點 *y* 為 *x* 的小孩，*Cskip(d)* 則代表以 *y* 為根(Root)之 subtree (包括 y 自己)的最多可容納節點個數。 例如，在圖三中，裝置 B 的 *Cskip*=1，C 的 subtree 最多可包含 1 個節點，A 的 Cskip=7 則 B 的 subtree 最多可包含 7 個節點。

位址的分配是由 ZigBee 協調者開始以及，ZigBee 協調者會先將自己的位址指定為 0 以及深度指定為 0，假設一個在深度 *d* 的父節點的位址已被指定為 A_{parent}，該父節點將指定他的第 *n* 個子路由器的位址為 $A_{parent}+(n-1)\times Cskip(d)+1$，並且指定他的第 *n* 個子終端設備的位址為 $A_{parent}+Rm\times Cskip(d)+n$。圖三中展示了一個位址分配的範例，在這範例中 ZigBee 協調者設定 *Cm*=6、*Rm*=4 以及 *Lm*=3，協調者的 *Cskip* 值可由 Eq. 1 得知為 31。因此協調者的第一個到第三個子路由器的位址將分別被指定為 0+(1-1)×31+1=1、0+(2-1)×31+1=32 及 0+(3-1)×31+1=63，而協調者的兩個子終端設備的位址將分別為 0+4×31+1=125 以及 0+4×31+2=126。

- 資料匯集及資料壓縮

在無線感測網路中，節點必須分散感測環境中的資訊，可能會有不同的節點感測到相同的資訊，造成資料的冗餘，此為空間上的冗餘，另外則是時間上的冗餘。為避免過多冗餘的資料在網路上傳輸，因此過去有許多探討資料匯集的文獻，如[1-5]。其中文獻[1]提出一種 Data-centric 的方法來選擇一條適當的路徑及利用快取來處理資料，以降低電源的消耗。但在長鏈狀網路中，路徑的選擇通常不是問題的核心，因為長鏈狀網路通常沒有多條路徑可供選擇，所以該方法可能並不適用於長鏈狀網路。

而過去在資料匯集系統的探討，大多採用以樹狀為基礎或群集為基礎的架構，參考文獻[2]認為維護一個特殊架構(例如樹狀架構)所需要的成本大大地降低了資料匯集所帶來的好處，所以提出一個不需要特定架構的資料匯集方法，主要精神在於避免維護架構所產生的成本，該文獻中提出兩種相關的機制，一為在媒介存取控制層(MAC layer)採用空間相關的資料匯集機制，稱為 Data-Aware Anycast (DAA)，另外則是在應用層(Application layer)考慮資料匯集系統中的等待問題，即一筆匯集的資料，需等待所有來源節點的封包都到達時，才能將所有的資料匯集成一筆，因此會造成時間上的浪費，故作者提出一種亂數等待(Randomized Waiting)的機制，以提升資料匯集的速率。雖然該文獻不需維護整個網路的架構，但是 Zigbee 對於架構維護的成本其實並不高，並且該作法必須更動 MAC 層，除了增加複雜度之外也增加了 MAC 層溝通的封包，加上 Application 層 Randomized Waiting 的機制也增加了封包的 delay。因此我們認為在長鏈狀網路中，文獻[2]的方法並不能完全適用。

文獻[3]提出的方法亦是利用 Data-centric routing 來降低電源的消耗。傳統 Address-centric 的路由是選取兩節點間的最短路徑（點對點），而 Data-centric routing 是針對多個不同的來源節點到達同一個目的節點的此種路由方式，選取最佳的路徑。因為跟文獻[1]一樣是利用路徑來降低電源的消耗，所以可能不適用長鏈狀的網路。

文獻[4]提出一種使用在 Data-centric storage (DCS)的雜湊表系統，稱為 Geographic Hash Table (GHT)。使用 Hash key 的概念，根據 key-value pair 找出存放備份資料的節點，並保證當網路拓樸變動時，仍可利用 key pairs 找到存放資料的節點，以提升網路的產出(throughput)。但長鏈狀網路通常是負責固定環境的偵測例如道路路燈監控、隧道二氧化碳指數監控、下水道水位變化，網路拓樸變動的機率不大，使用到 key pairs 的機率也大大地降低。

文獻[5]則是在一個稀疏的網路下，探討資料匯集方式，其假設感測器是均勻地佈置，而非呈長鏈狀。文獻[6]提出在 Ad-hoc 無線感測網路之下，每個感測點週期性地利用分散式的方式根據感測點剩下的電量來決定誰當群組的領導者，並且送到 Sink 的封包由領導者幫忙 relay，減少封包傳送，並強調他們提出的方法計算的時間並不受網路的大小影響，時間

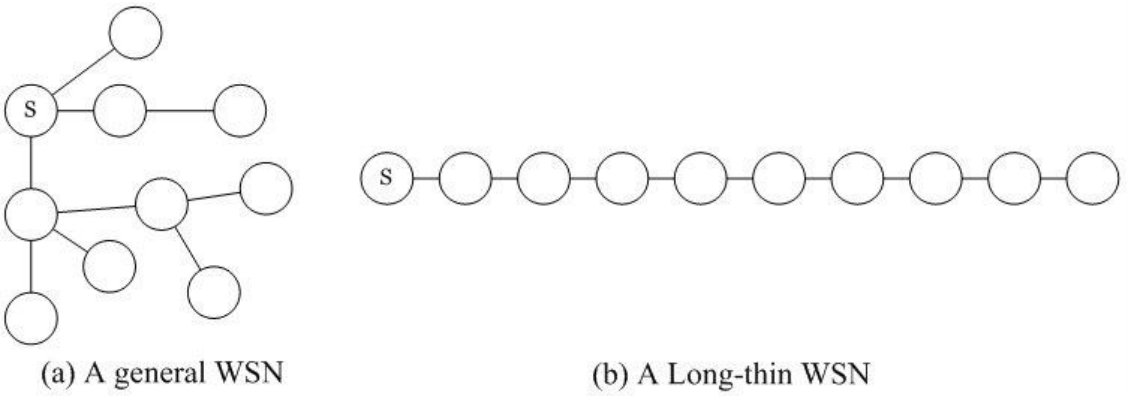
複雜度為 O(1)。但是此方法的 network model 假設所有的點跟群組的領導者只有 1-hop 的距離，在長鍊狀感測網路中，一個群可能包含了數個成員，且是以 multi-hop 的形式組成。文獻[7]根據感測點記憶體及電量使用情況動態地調整感測點感測的間隔時間(sense interval)，以及回報的間隔時間(send interval)，希望增加網路生命週期。文獻[7]的分群概念跟文獻[6]一樣，群內的感測點跟群內的領導者只有 1-hop 的距離。

文獻[8]根據網路大小以及感測點數量分成數個環形區域(ring)，再將每個環形區域分成數個區塊(section)，每個區塊內的感測點集結成一個群組，再從這些群組中挑選一個感測點當作群組的領導者負責群內感測點的資料匯集工作。但是該文獻分區塊的方法並不適用於長鍊狀網路的結構，並且該方法組內感測點的數量被區塊所限制，無法根據群內網路的流量動態調整群組的大小。

文獻[9]根據感測點記憶體以及電量使用的情形動態調整感測點收集資料(data-collection)以及傳送資料(data-sending)的時間間隔，藉以降低感測點能量的消耗，延長網路的生命週期。雖然該方法可以根據感測點的使用情境減少封包的傳遞。但在使用者的角度，我們希望由資料匯集的機制減少網路中封包的傳遞，而不是減少感測點收集資料及傳送資料的時間間隔影響資料收集的完整性。

文獻[10]根據能量消耗的預測建立樹狀匯集結構，並提供一個公式預測能量的消耗，而每個感測點根據該公式選擇父節點以建構樹狀匯集網路。但在長鍊狀網路中，每個節點通常沒有多條路可供選擇，所以該方法並不完全適用。

三、 研究目的



圖四、一般感測網路與長鏈狀感測網路之比較。

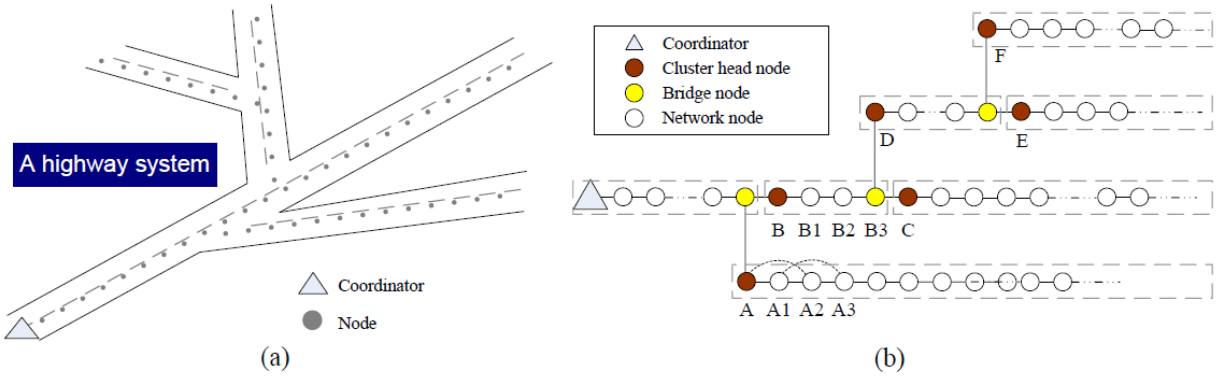
在建置長鏈狀網路時，一個最重要的課題是我們該如何形成這個網路。一個不佳的網路形成方式可能會使得網路無法真正成形，並且會造成網路斷點情況發生。我們觀察到 ZigBee 網路生成方式並不能適用於長鏈狀網路，因為ZigBee網路層規定網路位址只能有16位元，因此其最大的所能容納的網路節點個數為 $2^{16} = 65536$。然而，於 ZigBee 網路中，其分散式位址分派網路生成演算法，ZigBee協調者先定義一ZigBee路由器最多可容許連線之裝置個數(*Cm*)以及最多的子ZigBee路由器的數量(*Rm*)，以及網路的深度(*Lm*)。因此，當一個長鏈狀網路至少存在一個分支網路時，Cm 及 Rm 至少為 2，如此便會限制其網路深度最多為 15。然而，在這類的網路中，節點們由幾個網路的骨幹(Backbone)所連結，而這些骨幹則由幾個連接點給連接住 進而散佈到整個所要監控檢測之區域，每一個骨幹為一個線性的網路拓樸，並且可能包含數十甚至數百個節點，網路的範圍相當的大。為了更有效利用有限的網路位址，我們也需要一個新的位址分派演算法來解決此問題。

同時，在這樣網路架構下，我們發現如果沒有適當的資料匯集或資料壓縮機制，可能會有龐大的資料在網路上傳送，例如在圖四顯示了一個一般無線感測網路及一個長鏈狀感測網路的例子，兩個網路分別有十個感測節點（其中一個感測節點會扮演資料匯集點(Sink)的角色），我們假設每個感測節點都有一個封包要傳送到資料匯集點，且沒有任何匯集與壓縮的機制，在圖四(a)中會有 17 個封包在網路上傳送，而在圖四(b)中則有 45 個封包會在網路上傳送，由此例我們可看出資料匯集與資料壓縮在長鏈狀網路上是非常重要的，因為過多的封包在網路上傳送，不僅會造成電能的浪費(節能是無線感測網路上最為重要的議題) ，且會提升感測節點間為了傳送封包而產生的競爭，進而造成高的封包遺失率，因此本計畫即欲針對這種在長鏈狀無線感測網路的問題做討論並提出解決方案。針對這個問題，我們希望能在長鏈狀網路中加入水閘(Lock gate)的概念，並且將長鏈狀網路中的感測點作分群，由這些水閘負責執行資料匯集及資料壓縮，並且根據網路的流量來做到動態分群的效果，也就是能夠根據流量的計算動態地在長鏈狀網路中決定水閘的位置以及數量，讓每個水閘能夠有效率地發揮資料匯集及壓縮的效能，以降低長鏈狀網路特有的傳輸問題。

四、 研究方法

➢ ZigBee 網路生成

首先，我們先定義 LT Network 之網路型態。在這類的網路中，節點們由幾個網路的骨幹 (Backbone) 所連結，而這些骨幹則由幾個連接點給連接住 進而散佈到整個所要監控檢測之區域，每一個骨幹為一個線性的網路拓樸 (linear path)，並且可能包含數十甚至數百個 ZigBee Router 節點，如圖五(a)所示。而在本問題中，我們進一步假設所有的網路節點皆為 ZigBee Routers。於此網路架構中，我們會將整個網路切割成多個群集(Cluter)，每一個群集皆為一個線性拓樸，不存在有分支網路。而針對每一個群集，我們分別定義兩個特殊的節點型態：cluster head (群集領袖) 及 bridge (橋接點)，其中 cluster head 便為一群集中至網路協調中心最近之節點，而 bridge 為一群集中至網路協調中心最遠之節點，其他非此兩種型態之節點，我們統一定義為 network node。圖五(b)便為一個長鏈狀網路，經由我們節點型態定義後的示意圖。此外，如果群集領袖節點之父節點為另一群集中的橋接點時，我們稱此群集為橋接點所在群集之子群集(child cluster)；而橋接點所在群集為父群集(parent

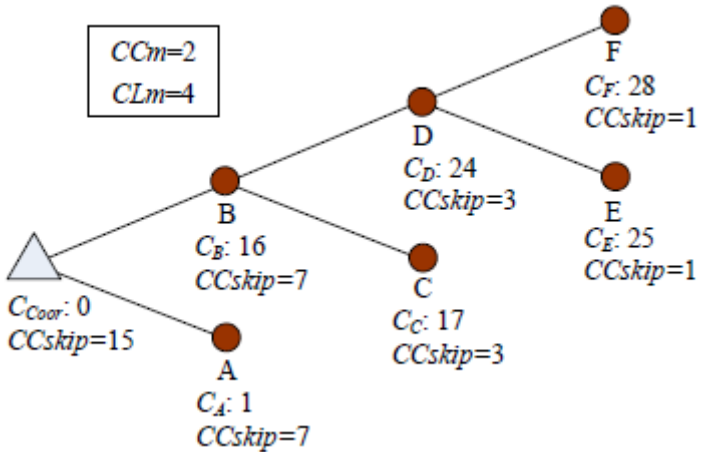


圖五、(a) 長鏈狀感測網路範例。(b) 網路節點角色定義。

cluster)。如圖五(b)中，群集 C 為群集 A 之子群集(child cluster)。同時，為了使得我們的設計可以相容於 ZigBee 標準之規範，我們將 ZigBee 16 位元之網路位址分成兩個部份：m 位元的 cluster ID (群集位址)，及 (16-m)位元的 node ID 節點位址。所以，網路中任一節點 v 之網路位址可以用下列的表示法下示：(Cv, Nv)，其中 Cv 為 v 節點之群集位址，而 Nv 為 v 之節點位址。

本方法中，於網路佈建階段，網路佈建者必需細心的佈建，且事先設置網路節點之角色，而其佈建及設置角色之基本原則如下：

1. 整個網路是由多個線性拓樸所組成。
2. 於每一個群集中，第一個節點為其 cluster head，而最後一個節點為 bridge。
3. 非網路協調中心節點之 cluster head 必定和一 bridge 互為通訊鄰居，而此 bridge 所在群集為其父群集。
4. 相反的，如果一 bridge 非 leaf 節點，則必有一 cluster head 為其通訊鄰居，而此 cluster head 所在之群集為其群集之子群集。
5. 於一群集中之網路節點，不會存在有別的群集之網路節點為其通訊鄰居。



圖六、圖五(b)之邏輯拓樸。

根據上述之佈建原則佈建下，網路佈建者便可建立出一個邏輯拓樸(logical network) GL，而其建立的方法為將每一個群集示為一個網路節點，而若群集間存在有子群集與父群集等關係時，則其相對應之網路節點便會存在一個連線相連。圖六便為一個邏輯拓樸例子，而其為圖五(b)的 GL。於此邏輯拓樸中，網路佈建者可以進一步定義出兩個網路參數：GL 中之多可容許連線之網路節點個數(*CCm*)，以及 GL 網路的深度(*CLm*)。

經過上述 GL 邏輯網路之網路參數設置及網路節點角色設置後，整個網路成生位址分派演算法便可分為兩個階段：1) 利用 GL 之網路參數，網路佈建者會先手動地替每一個群集領袖分派其 cluster ID。2) 每一個節點之 node ID 可利用 ZigBee 分散式位址分派演算法之概念來取得。因此，網路中央協調者(Coorinator)之cluster ID為0，其node ID也為0，接著對於每一個GL中之節點，若其深度為d，而其cluster ID為C，則其 i-th 子群集更會被分派到一cluster ID為 $C + (i - 1) \times CCskip(d) + 1$，其中 $CCskip(d) = (1 - CCm^{CLm-d}) / (1 - CCm)$。當每一個群集領袖都連接上網路並取得其cluster ID後，便可以開始利用 ZigBee 分取式位址分派演算法來分派 node ID 給其他群集中的網路節點。

因此，一開始 Coordinator 擁有(Cv, Nv)為(0, 0)，並開始定期廣播其Beacon。當存在一個網路節點u，其並未連上ZigBee網路，具還沒被分派網路位址(Cu, Nu)時，便會發送一Association Request給其Beacon發送者。因此，假設有一個節點v，已連上網路，並擁有網路位址為(Cv, Nv)，具定期的廣播Beacon封包，且送到 Association Request(s) 封包時，v 便會執行下列的程序：

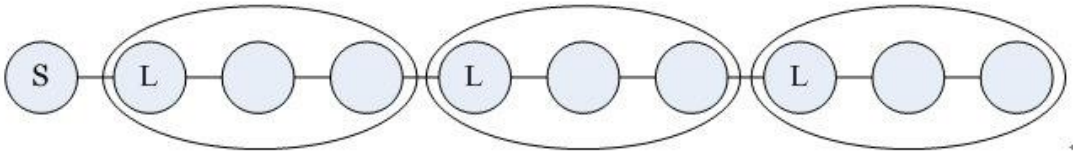
1. 若v並不是一個bridge，則v便會設定 N = Nv + 1。接著，v 會根據所收到的 Association Requests 之訊號強度執行一個排序，接著 v 會執行下列步驟：
   a) 若 Cu = Cv，則 v 會忽略此封包，查看下一個強度的 Association Request。
   b) 反之，便會設定 Nu = N，並設定 N = N + 1。接著便發送一Association Reply給 u。等所有的Association Request(s)都處理完畢後，若其中存在一個u為v的群集中的bridge，則v會選此節點為下一個廣播Beacon之節點。反之，則最後一個接收到Association Reply之節點，會被v選為下一個廣播Beacon之節點。
2. 若v為一個bridge，則v只接受其此群集之群集領袖節點之Association Request封包，同時並將其 Nu 設為0，並回應Association Reply封包，且選其為下一個Beacon封包廣播者。

➢ 資料匯集及資料壓縮

為了降低整個網路的傳輸量，首先將感測器分群(Group)，分群之後每個群會有一個領導者(Leader)負責群內的資料排程、資料匯集及資料壓縮，每個群應該有能力依目前網路狀況動態地選擇最適合自己的匯集演算法或壓縮演算法，當然這樣的選擇還必須將使用者的需求考慮進去，在本計畫中我們預計會設計一個演算法使得群的領導者可以選擇相關的匯集演算法或壓縮演算法。

首先用一個例子來說明分群的好處，圖七顯示了一個長鏈狀感測網路的例子，假設每個感測器皆想傳送一個封包，根據前面的說明，我們已經提到如果沒有適當的資料匯集或資料壓縮，整個網路上需要傳送 45 個封包，現在我們假設每個 Group 皆有個領導者 L，群內的感測器會先將資料送給群的領導者，領導者最後只會送一個封包給 Sink，依此傳送機制，只會有 21 個封包需要被傳送，由此例子我們可看出分群的確可以降低傳送封包的數量。

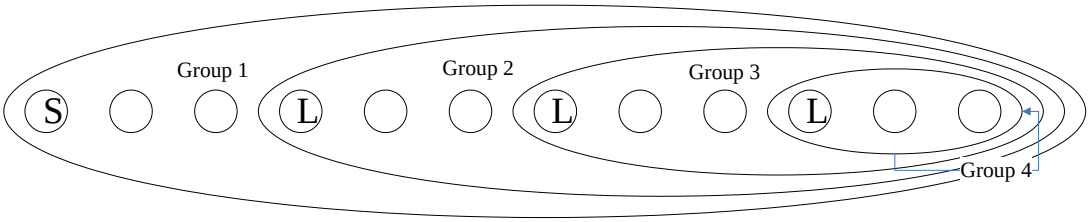
進一步觀察上述的例子，不難發現，這樣的運作類似於運河的運作，每個領導者扮演著水閘(Lock)的角色，控制著感測器的流量，使得長鏈網路中的資料流得以匯集在水閘處，領導者等待適當的時機將整群的資料作壓縮後送出，如同開放水閘般讓水流通過。以下說明資料匯集及壓縮方法。



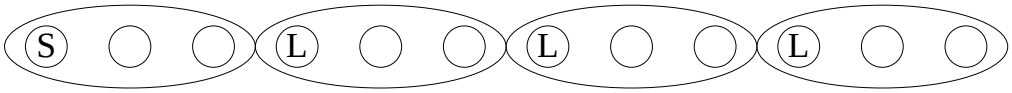
圖七、分群過後的長鏈狀無線感測網路。

在資料匯集與壓縮上，我們先說明我們所考慮的資料匯集型態，長鏈狀網路大致上可以分成兩種資料匯集型態。第一種型態如圖八所示，每個群組的閘門(Lock gate)除了負責自己群組的資料匯集及壓縮之外，也要順便處理下游群組的資料匯集；至於第二種型態如圖九所示，每個群組的閘門只要負責自己群組的資料匯集及壓縮，其他群組的封包只要幫忙傳遞(relay)即可。

第一種資料匯集型態的好處在於能夠最有效地減少長鏈狀網路的資料量，但是由於無線感測網路(Wireless Sensor Network)有時間相關性及空間相關性的特性，每個感測區域會有不同的代表值，如果採用型態一資料匯集模式會破壞這個特性，以圖 4.2 舉例子說明，假設每個群組的感測資料是環境的溫度，又閘門壓縮的方式是採用簡單地將收集到的溫度資料作平均之後傳給 sink，然而每個 Groups 之間的溫度可能相差甚遠(假設 Group 3 在冷氣房，Group 4 在暖氣室中)，如果 Group 3 將 Group 4 傳來的資料作平均，會變成最後 Sink 接收到的溫度值是介於冷氣房及暖氣室之間，這並不是我們所希望的結果，我們希望能夠知道冷氣房以及暖氣室溫度分別為多少。所以我們決定採用第二種資料匯集模式，保留無線感測網路的時間空間相關性，也保有資料匯集及壓縮的功能。



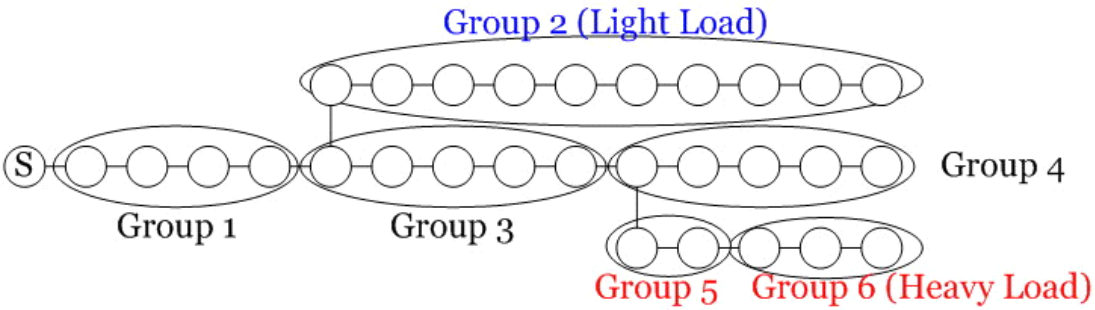
圖八、長鏈狀網路資料匯集型態一。



圖九、長鏈狀網路資料匯集型態二。

為因應網路流量變化，分群必須是動態的。由於每個感測器的資料量不同，例如在一個下水道系統中，偵測到高水位的感測器可能需要傳送較多的資料，因此分群時應該將負載平衡(Load Balancing)考量進去，我們將設計一考量負載平衡的動態分群機制。

每個群的領導者有兩個不同的功能，(1) 當群內的資料流量過多時，領導者可以抑制住過多的封包送往 Sink 以達到理想的傳輸，(2) 當群內的資料流量過小時，領導者可以先將資料儲存起來，等待資料量到達一定的值後再將資料傳送出去，以便減少傳輸的次數。然而這兩個功能會產生不同的問題，首先當群內感測器流量過大時，有可能使用任何的資料匯集技術或資料壓縮技述都無法降低資料量，或者當群內感測器流量過小時，領導者所儲存的資料可能會遭致過長的延遲，而無法被使用者所接受。因此這產生了另一個議題：負載平衡。



圖十、負載平衡的範例。

圖十顯示了一個負載平衡的範例，在此例中 Group 2 的感測器由於並沒有太多的資料量，因此 Group 2 包含了較多的感測器，反觀 Group 5 及 Group 6，由於有較多的資料量，因此 Group 內只包含了少數的感測器。從這個例子我們可以看出我們需要一個動態的分群演算法，此演算法可依據目前網路的狀況（即資料量的多寡）來決定分群的方式，而本計畫其中一個目標就是要設計並實作出這樣的演算法，在自動分群法的設計上可從兩個方向去著手，（一）集中式演算法：由 Sink 決定群的分配法，此方法在設計與實作上或許較為簡單，但是較無彈性，（二）分散式演算法：由每個群的領導者決定是否將成員收進來或釋放出去，甚至決定是否將領導者的身份轉移給別的感測器，由於是每個群的領導者自行決定，因此方法為分散式的，這樣的做法固然較有彈性，但也會產生一些問題，例如一個群變更其組成成員後，可能會連帶影響很多群也跟著要改變，如此對整個網路會造成不良的影響，因此在設計上必須要非常小心。

基於上述的討論，我們設計出一個分散式的長鏈狀網路動態分群演算法，此演算法能夠跟據網路流量狀態動態地做分群。在介紹方法之前，我們必須先解釋方法會用到的名詞：

- 邊界群組(boundary cluster)：網路中包含 leaf node 的群組即為邊界群組。
- 正常群組(normal cluster)：網路中不包含 leaf node 的群組即為正常群組。
- 分支點(branch node)：若感測點擁有多個 children，也就是位於樹狀結構的分支上，我們就稱該感測點為分支點。
- 上一層水閘(next lock gate)：假設網路中一個水閘 L，L 上游鄰近群組的群組負責人 L_n

即為水閘 L 的上一層水閘。

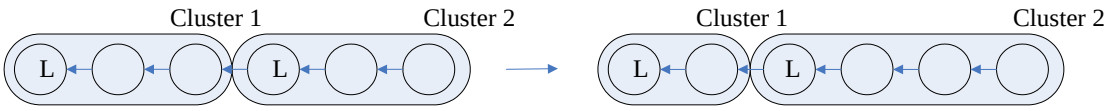
為了達到動態分群的最佳化效果，本方法對於網路中的水閘點做動態調整，移動水閘點的位置能夠調節各個感測點群的成員數量，也能平衡水閘處理的資料流量。為此我們針對樹狀網路結構設計一個機制讓長鏈狀網路中的水閘能夠定期的檢查群內感測點的資料傳送量，根據群內的資料傳送量來移動水閘的位置以調節成員數量。

我們定義 T 匯集時間、CR(Compression Rate)封包壓縮率、PS(Packet Size)封包大小。在每個水閘點週期性的檢查以下公式:

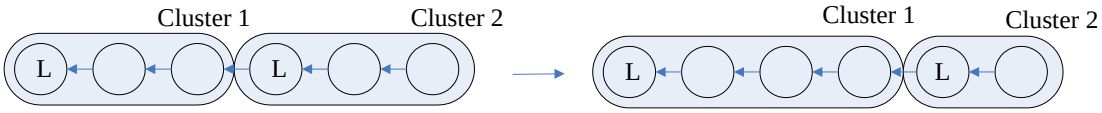
$$\left(T \times \sum_{i \in Cluster \quad (L)} \lambda_i \right) \times CR = PS$$

爲了節省電量及減少封包的傳送量，水閘會對匯集的資料作壓縮，並在匯集資料收集到一給定的封包大小 PS 時向下游轉送。匯集時間 T 代表水閘收集到給定的封包大小 PS 所需要的時間，群體內各感測節點的資料傳送率(data rate)則作為調整感測節點群大小的指標。由上述公式可以看出匯集時間 T 跟群內感測點資料傳送率的總和成反比，因此匯集時間 T 也間接地反應感測節點群的大小。

T_{min} 及 T_{max} 定義為能容忍的最小及最大匯集時間。當 $T < T_{min}$ 時，代表此群組中的感測點資料傳送率偏高，應該要減低該群組感測點的數目，提高水閘匯集壓縮的效率；反之 $T > T_{max}$ 時，群組中的感測點資料傳送偏低，應該要增加群組感測點的數目，以最佳化整體匯集效率。那要如何增加或是減少群內感測點的數目呢？在這裡我們希望藉由 lock gate 也就是群組負責人的移動來達到這個效果。移動的方向有兩種，一為上拉(pull)，水閘向下游方向移動，如圖十一(a)所示，此動作減少 cluster 1 的負責範圍。一為下推(push)，水閘向上游方向移動，如圖十一(b)所示，此動作擴大了 cluster 1 的負責範圍。由上述兩個例子可得知利用上一層水閘(next lock gate)的移動可以增加或是減少群內感測點的數目，進而增加匯集壓縮的效率。



圖十一(a)、水閘上拉移動(pull)。

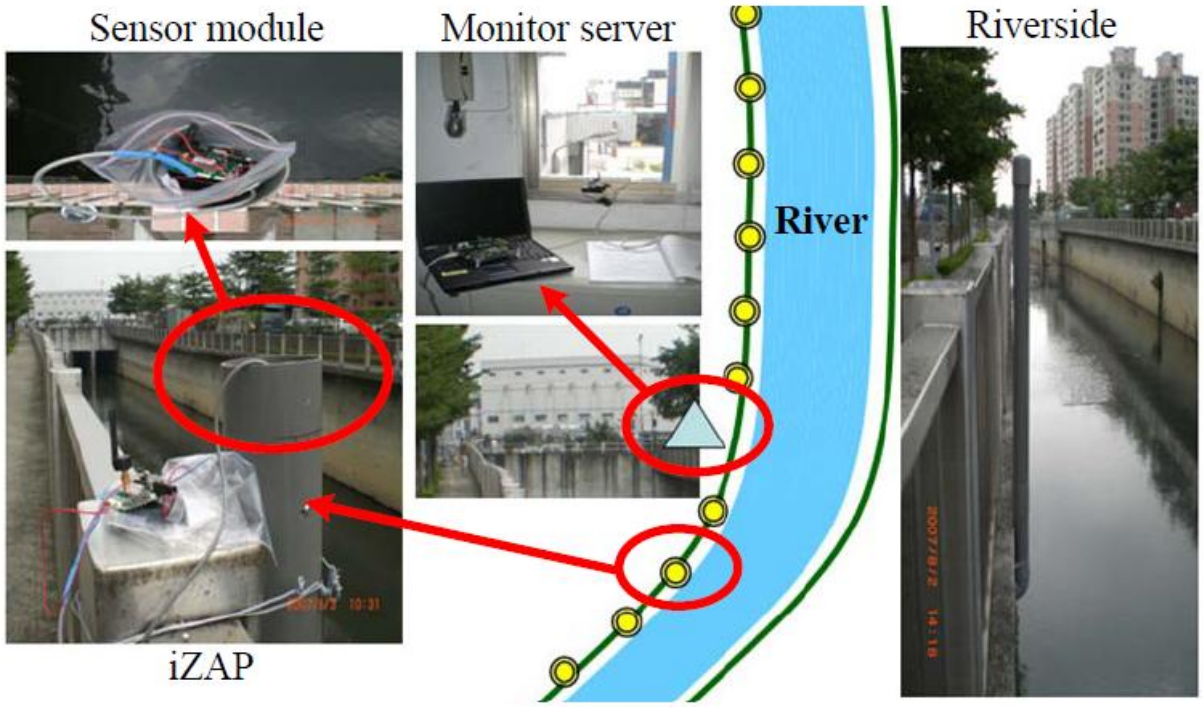


圖十一(b)、水閘下推移動(push)。

因此我們的作法就是希望水閘定期地經由上述公式來判斷群內感測點數目是否需要作調整，若需要，則藉由上一層水閘(next lock gate)的移動來調整群內感測點的數目。當水閘經由上述公式的判斷發現群內感測點過多或過少而需要調整時，水閘 L 會向上游各分支傳送詢問封包(request packet)，詢問上一層水閘(L_n)的狀態，若 L_n 也在調整狀態，則 L_n 視為忙碌，不回覆此詢問封包，反之，L_n 將回傳本身的匯集時間 T 以及群內所有感測點的資料傳送率(data rate)給 L。若 L 收到至少一回覆封包，此水閘 L 則根據收到的回覆候選之中，決定選擇要移動的 L_n。舉例來說，當 $T < T_{min}$ 時，L 會優先上拉(pull)T 值最大的 L_n，反之當 $T < T_{min}$ 時，L 會優先下推(push)T 值最小的 L_n。當水閘 L 選擇好移動的 L_n 以及移動的方向之後，就開始進行移動的動作，每次的移動 L 都必須更新群組的 T 值，直到 L 所負責的群組匯集時間 T 回歸正常值。

在上拉的過程中，若 L_n 移動至分支點(branch node)，會造成 L_n 負責的感測點數目遽增，我們稱之為分支點效應，如圖 4.9(a)所示。為避免此情形發生，當 L 發現 L_n 的下一次移動會到達分支點(branch node)時，L 將會改變移動候選，根據 T 值選擇下一個合適的 L_n，當分支點的所有 child 均為 L_n，則不會產生分支點效應，L 可如往進行 L_n 上拉的動作，如圖 4.9 (c)所示。在下推的過程中，也會發生分支點效應，當 L_n 由分支點(branch node)下移至非分支點時，L 負責的感測點數目也可能遽增，如圖 4.9 (b)所示。此時，L 應根據剩餘 L_n 的匯集時間 T 選出恰當的移動候選。跟上拉移動不同的是，即使所有 L_n 都移動到分支點(branch node)，分支點效應依舊存在，此時 L 應合併擁有最小資料傳送率的分支，避免 L 負責的感測點數目遽增。

五、 實驗與分析



圖十二、長鏈狀網路佈建實測。

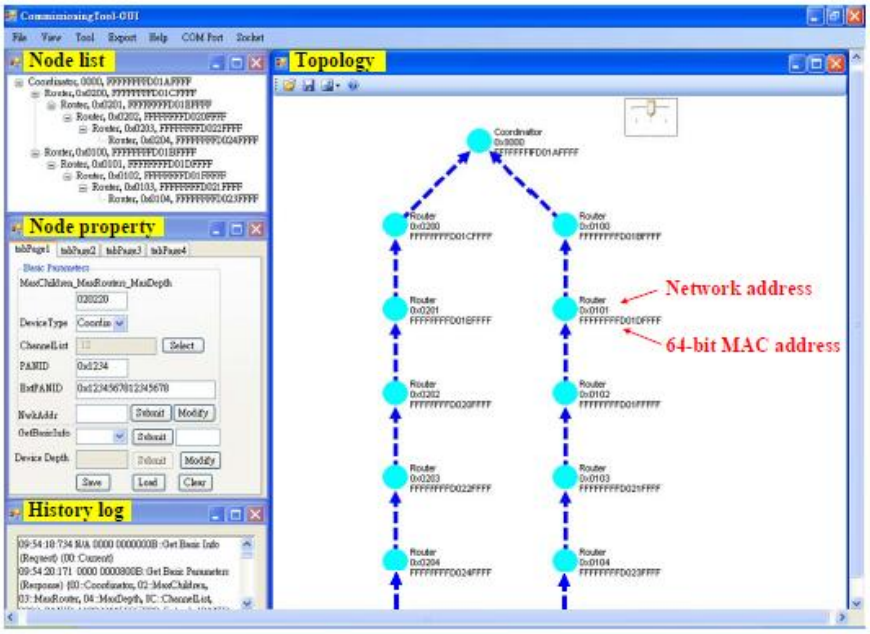
- **長鏈狀網路生成實作**

實作環境

- ✧ 硬體：
  - ✓ III ZigBee Advanced Platform (iZAP) [13]
    - ➢ RF module：Jennic JN5121
    - ➢ Battery：感測節點上的充電電池

- ✧ 軟體：
  - ➢ JN-SW-4031 SDK Toolchain：Jennic 平台的標準化開發工具。
  - ➢ JN-SW-4030 802.15.4 MAC：802.15.4 通訊協定媒體存取控制的 package。
  - ➢ Microsoft Visual Studio 2005：開發 PC 端程式使用的軟體。

於台北市中偵測一河流之水平面高度。而整個網路有41個節點，每個節點皆距離100 meters而網路深度為 20，如圖十二所示。整個網路共有三個群集，其中Coordinator所在之群集只有自己本身一個節點，而網路生成結果如圖十三所示。因此，透過實測的結果，可以看出我們的網路生成演算法可以適用於長鏈狀網路。



圖十三、長鏈狀網路生成結果。

➢ 資料匯集實作

實作環境

✧ 硬體：

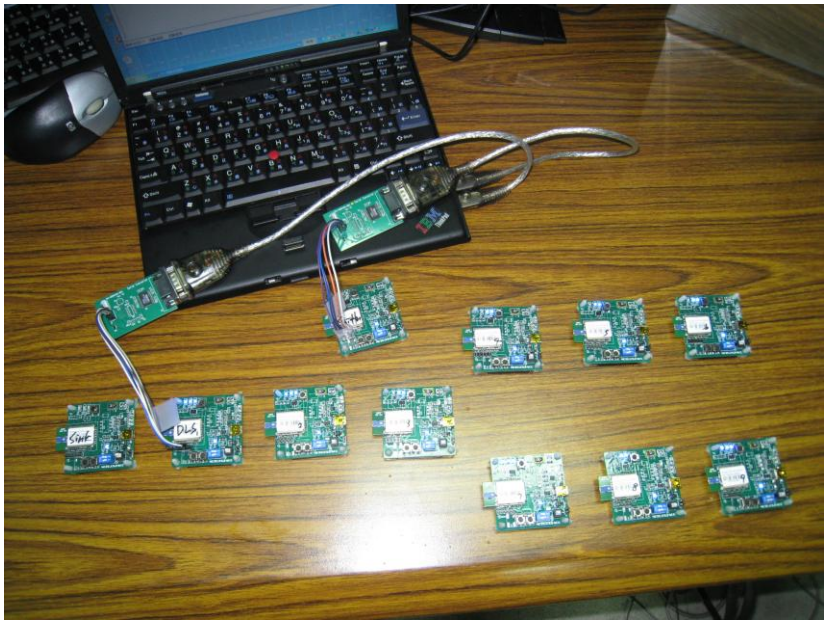
✓ JN5139-Z01-M00R1

- ➢ Serial Connector：連接感測節點與 PC 端的轉接器
- ➢ Battery：感測節點上的充電電池
- ➢ Antenna：感測節點的天線，用來溝通傳輸資訊。

✧ 軟體：

- ➢ JN-SW-4031 SDK Toolchain：Jennic 平台的標準化開發工具。
- ➢ JN-SW-4030 802.15.4 MAC：802.15.4 通訊協定媒體存取控制的 package。
- ➢ Microsoft Visual Studio 2005：開發 PC 端程式使用的軟體。

實作畫面：



圖十四、PC 端及感測節點。

實作數據

✓ 測量參數：

1. 總封包量
2. 總傳輸量（bytes）

✓ 比較對象：

i. DLS (Dynamic Lock Gate Selection)
水閘式資料匯集與壓縮並依負載平衡動態分群，水閘會跟據網路流量調整網路中水閘的數量以及位置。

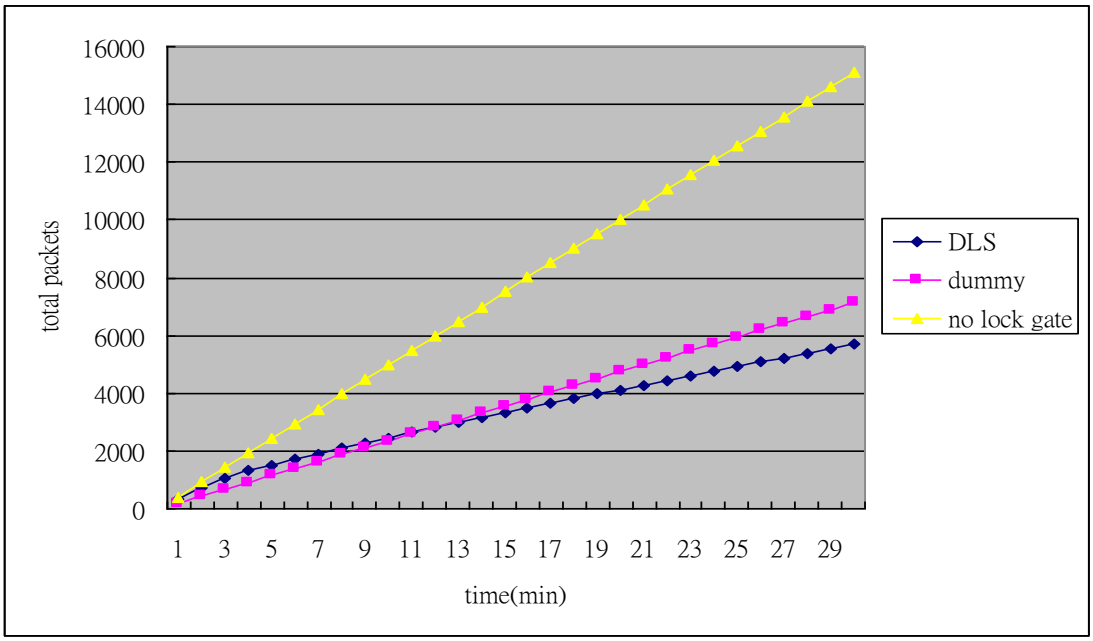
ii. Dummy (Fixed Lock Gate Selection)
水閘固定在分支點位置，並不根據網路流量調整水閘數量及位置。

iii. No lock gate
感測節點單純地將資料封包傳送給 sink，沒有任何封包匯集壓縮機制。

實驗一：總傳送封包

➢ 參數設定：

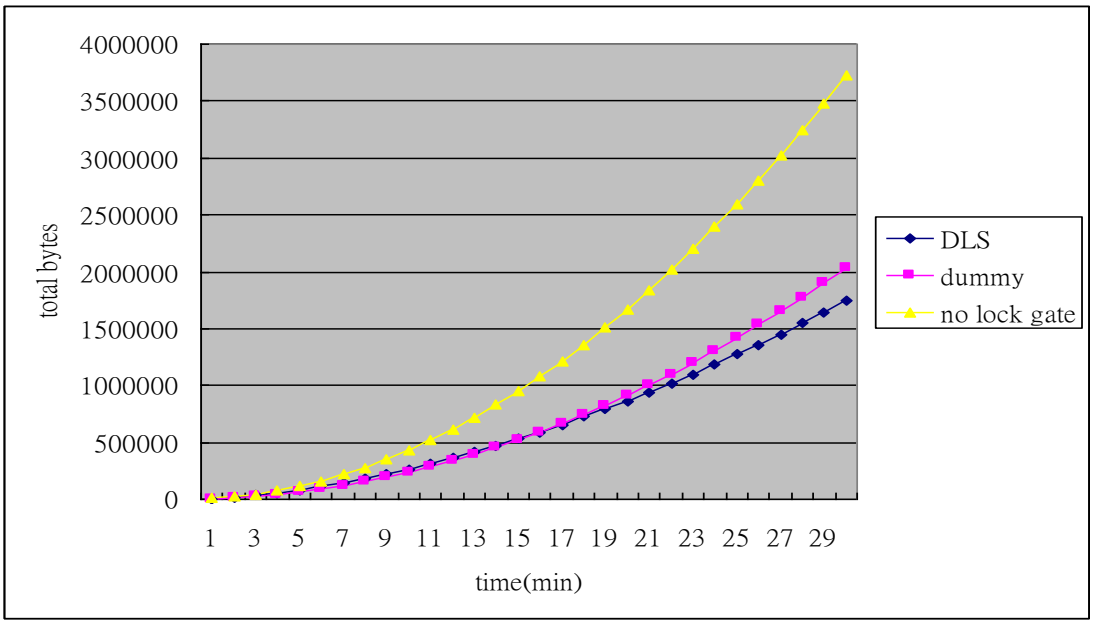
- 實驗時間：30 分鐘（半小時）
- 每隔 2 秒傳送一個資料封包



圖十五、總封包量與時間關係圖。

實驗二：總傳輸量（bytes）

- 參數設定：
  - 實驗時間：30 分鐘（半小時）
  - 每隔 2 秒傳送一個資料封包



圖十六、總傳輸量與時間關係圖。

六、 結果與討論

在這第三年度的計畫中，針對長鏈狀網路做了一些探討，此網路是由一些擁有線性路徑節點的骨幹連接彼此所形成，少部分之節點可與其他長鏈上之節點連接，作為長鏈間溝通之橋樑。長鏈狀網路型態雖然和一般正常的無網感測器網路型態大為不同，卻常見於無線感測網路監測(Surveillance)應用中。而針對此種網路型態，我們發現了兩種問題：1) ZigBee 網路生成問題，及 2) 資料匯集及資料壓縮問題。於 ZigBee 所規範網路生成演算法中，我們不難發現其不適用於長鏈狀網路，並會導致大量的網路節點無法連上 ZigBee 網路。而於長鏈狀網路的資料匯集上，我們分析長鏈狀網路所衍生出來之傳輸問題，發現若沒有資料匯集及資料壓縮的機制，長鏈狀網路封包的數量遠比其他一般無線感測網路架構的封包數量還要高，因此資料匯集及資料壓縮在長鏈狀網路中格外重要。因此，我們分別此兩個問題中，設計了一相容於 ZigBee 標準規範，且適用於長鏈狀網路型態之網路生成演算法；同時，於此生成的網路中，我們更進一步設計了一支援長鏈狀感測網路的動態分群、資料匯集、資料壓縮等功能。由實驗數據可看出，長鏈狀網路加入我們設計的演算法之後可以明顯地減少長鏈狀網路特殊架構所造成的大量資料量，間接降低該網路架構下無線感測網路電量的消耗，提高網路使用的生命週期。

在此次的三年計畫中，我們完整地探討了以 ZigBee 樹狀網路為基礎之無線感測網路相關議題：網路生成問題及資料收集回報問題。於無線感測器網路中，這兩個問題為其最基本且重要的議題，首先於網路生成上，ZigBee 所設計之分散式位址分配演算法，由於 *Cm*、*Rm* 及*Lm* 的限制可以簡化網路位址分派的複雜度，但是這三個參數亦可能使得裝置無法加入此網路，即使還有父節點有剩餘的位址空間，而造成了孤兒問題 (orphan problem)。同時，針對此問題，利用我們所提出的觀察，加上實測的結果，得知此問題並無法透過簡單的參數調整等方法來解決 orphan 的情形。因此，我們將此問題重新定義，並依照路由節點及終端節點兩種不同節點的特性，將此問題分成兩個子問題，分別為：分支度及深度受限之延展樹(BDDTF) 問題及終端設備最大的配對數 (EDMM) 問題。同時，我們也證明了 BDDTF 為一個複雜的問題 (NP-Complete)，因為我們提供了一集中式的 BDDTF 演算法來降低孤兒之數量，而因應無感感測器硬體限制，也提供了一分散式演算法。而針對 EDMM 問題，本計畫證明了其為一最大分配 (maximum matching) 的問題，因此可以簡單的利用最大分配演算法來得到最佳解，也就是使得終端節點成為孤兒的情形最小。同樣地，我們也提供了一分散式的分配演算法，使得 ZigBee 網路可以利用較輕量 (lightweight) 的方式，使得網路位址可以最有效的應用，降低節點形成孤兒的機會。

而上述的方法，雖然解決了 ZigBee 於一般網路上的 Orphan Problem，然而由於其仍保有 ZigBee 網路生成中所設定的三個網路參數：最多可容許連線之裝置個數(*Cm*)以及最多的子ZigBee路由器的數量(*Rm*)，以及網路的深度(*Lm*)。因此，當網路型態為長鏈狀網路時，上述的方法仍然會造成大量的網路節點無法連接上網路，使得無線感測網路無法正常運作。因此，我們透過將ZigBee網路位分割為兩個部份：m-bit cluster ID及(16-m)-bit node ID，提出了一個位址分派演算法，解決了 ZigBee 無線感測網路於長鏈狀網路架構下無法生成的問題。

接著，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)。此時，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。因此，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。

同樣地，此 convergecast 之解決，仍然只適合於一般 ZigBee 網路型態，我們分析長鏈狀網路所衍生出來之傳輸問題，發現若沒有資料匯集及資料壓縮的機制，長鏈狀網路封包的數量遠比其他一般無線感測網路架構的封包數量還要高，因此資料匯集及資料壓縮在長鏈狀網路中格外重要。為了減少長鏈狀網路特殊架構所造成的龐大資料量，我們首先將感測網路分群，並在每個群組中挑出一個領導者，讓該領導者對群內的資料做匯集以及壓縮的動作。而在資料匯集和壓縮方面，我們將設計出一種稱為水閘式(Lock)資料匯集和壓縮的演算法，每群的領導者會負責群內感測器的資料匯集及資料壓縮，接著依網路上的流量，以負載平衡的原則進行動態分群。

因此，本計畫最後根據了不同的網路型態，皆探討與設計了網路生成與資料匯集等演算法，可使得我們於不用的無線感測網路應用中，皆可以利用本計畫所設計之方法，生成無線感測網路，並藉由所生成之無線感測網路，透過所設計之資料匯集回報等路由協定，將無線感測器所感測之資值，成功且及時地回傳於資料收集端。

- 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「International Conference on Body Sensor Networks (BSN)」，其出國之報告書如附錄一

- 本計畫目前的研究成果為三篇論文如下：

附錄二：

Y.-C. Wang, C.-H. Chuang, Y.-C. Tseng, and C.-C. Shen, "A Lightweight, Self-Adaptive Lock Gate Designation Scheme for Data Collection in Long-Thin Wireless Sensor Networks", *Wireless Communications and Mobile Computing*. (SCIE) (to appear).

附錄三：

Y.-C. Wang, C.-H. Chuang, Y.-C. Tseng, and C.-C. Shen, "Dynamic Water Gate Assignment Scheme for Data Aggregation in Long-Thin Sensor Networks", *Int'l Computer Symp.*, 2010, Tainan, Taiwan.

附錄四：

M.-S. Pan, H.-W. Fang, Y.-C. Liu, and Y.-C. Tseng, "Address Assignment and Routing Schemes for ZigBee-Based Long-Thin Wireless Sensor Networks", ***IEEE VTC***, 2008-Spring.

- 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。

七、 參考文獻

[1] C. Intanagonwiwat, R. Govindan, and D. Estrin, “Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks,” Proc. Sixth Ann. Int’l Conf. Mobile Computing and Networking (MobiCOM), Aug. 2000.

[2] Fan, K., Liu, S., and Sinha, P., “Structure-Free Data Aggregation in Sensor Networks,” *IEEE Transactions on Mobile Computing* 6, 8 (Aug. 2007), 929-942.

[3] Krishnamachari, B., Estrin, D., and Wicker, S. B., “The Impact of Data Aggregation in Wireless Sensor Networks,” In *Proceedings of the 22nd international Conference on Distributed Computing Systems* (July 02 - 05, 2002). ICDCSW. IEEE Computer Society, Washington, DC, 575-578.

[4] Ratnasamy, S., Karp, B., Yin, L., Yu, F., Estrin, D., Govindan, R., Shenker, S., “GHT: A Geographic Hash Table for Data-Centric Storage,” In Proceedings of the First ACM International Workshop on Wireless Sensor Networks and Applications (WSNA2002), Atlanta, Georgia, September 28, 2002.

[5] J. Gao, L. J. Guibas, J. Hershberger, N. Milosavljevic, “Sparse Data Aggregation in Sensor Networks,” The 6th International Conference on Information Processing in Sensor Networks (IPSN) 2007.

[6] Ossama Younis and Sonia Fahmy, “Distributed Clustering in Ad-hoc Sensor Networks: A Hybrid, Energy-Efficient Approach”, INFOCOM 2004

[7] Angelo Brayner, Aretusa Lopes, Diorgens Meira, Ricardo Vasconcelos, Ronaldo Menezes, “An adaptive in-network aggregation operator for query processing in wireless sensor networks” The Journal of Systems and Software, 2007-July

[8] Yujie Zhu, Ramanuja Vedantham, Seung-Jong Park, Raghupathy Sivakumar, “A scalable correlation aware aggregation strategy for wireless sensor networks”, Wireless Internet, 2005

[9] Angelo Brayner, Aretusa Lopes, Diorgens Meira, Ricardo Vasconcelos Ronaldo Menezes, “An adaptive in-network aggregation aperator for query processing in wireless sensor networks”, The Journal of Systems and Software 2008

[10] Albert F. Harris III, Robin Kravets, Indranil Gupta, “Building trees based on aggregation efficiency in sensor networks”, Ad Hoc Networks, 2007-Nov

[11] Y.-C. Wang, Y.-Y. Hsieh, and Y.-C. Tseng, “Compression and Storage Schemes in a Sensor Network with Spatial and Temporal Coding Techniques”, IEEE VTC 2008-Spring.

[12] M.-S. Pan, H.-W. Fang, Y.-C. Liu, and Y.-C. Tseng, “Address Assignment and Routing Schemes for ZigBee-Based Long-Thin Wireless Sensor Networks”, IEEE VTC, 2008-Spring.

[13] III ZigBee Advanced Platform (iZAP). http://zigbee.iii.org.tw/.

附錄一

出國報告書

On Optimization of Accelerometers Deployment for Human Posture Tracking

C.-H. Wu, Y.-T. Chang, and Y.-C. Tseng

***Int'l Conf. on Body Sensor Networks*, 2011, Dallas, USA.**

出席國際學術會議報告

100 年 5 月 31 日

報告人姓名	吳鈞豪	就讀校院（科系所）	■博士班研究生 交通大學資工所 □碩士班研究生
時間 會議 地點	2011 5/20 到 2011 5/27 Dallas, Texas, US		
會議名稱	（中文)國際人體感測網路研討會 （英文)Int'l Conf. on Body Sensor Networks (BSN)		
發表論文題目	（中文）適用於人體姿態追蹤的加速度計部署最佳化 （英文)On Optimization of Accelerometers Deployment for Human Posture Tracking		

報告內容應包括下列各項：

一、參加會議經過

BSN2011 辦在美國中部的城市 Dallas, 加上開場的 reception 及之後的 tutorial，一共舉行了四天。主辦單位是 UTDallas，感覺還蠻用心的，有精美的海報範本、書包、及小冊子。在漫長的飛行後，會議過程如它的議程所示，第一天是 pre-conference reception，第二天是正式開始，有 opening，有 best paper session，poster session，和晚宴。Best paper 的得主是泰國的年輕女教授，做 ASL 的辨識。不過只做了 26 個英文字母，並不完全能用來溝通。我要報告的是第二天的 poster session，一開始有個 teaser，chair 要我們準備兩三張投影片，用五分鐘跟大家介紹一下。因為我們的題目頗數學的，希望我的解說能幫助其它人了解這個題目。再來就是在海報前跟其它人解說，陸續就有人來參觀了。會議第三天續繼聽 paper 報告，還有展示他們的系統，教大家怎麼用。

二、與會心得

這次參加 BSN 發現 MIT 的學生也會來這個 conference，還蠻酷的。聽 keynote speaker 講 tele-medicine，這個有點不切實際的方向，似乎還蠻有道理的。只是不知道台灣有沒有產業要發展。

三、考察參觀活動(無是項活動者省略)

無

四、建議

希望能在國內帶動 BSN 相關的產業，也提升我們專利佈署的廣度。

五、攜回資料名稱及內容

他們有發議程介紹，及會議的論文。另外也跟一些學者、業界交換了名片。

六、其他

On Optimization of Accelerometers Deployment for Human Posture Tracking

Chun-Hao Wu*, Yuan-Tse Chang*, and Yu-Chee Tseng*†

*Department of Computer Science, National Chiao Tung University, Hsin-Chu, 300, Taiwan
†Research Center for Information Technology Innovation, Academia Sinica, Taipei, 115, Taiwan
Email: {wchunhao, yuantse, yctseng}@cs.nctu.edu.tw

***Abstract*—We are interested in tracking human postures by deploying accelerometers on a human body. One fundamental issue in such scenarios is how to calculate the gravity, no matter when human body parts are moving or not. Assuming multiple accelerometers being deployed on a rigid part of a human body, a recent work proposes a data fusion method to estimate the gravity on that rigid part. However, how to find the optimal deployment of sensors that minimizes the estimation error of the gravity is still an open problem. In this paper, we formulate the deployment optimization problem. Since the problem is hard, we conduct experiments to observe what the optimal deployments should be.**

I. Introduction

A *body-area inertial sensor network (BISN)* consists of multiple accelerometers, magnetometers, and gyroscopes connected by wired/wireless links. An important usage in a BISN is to track human motions through these inertial sensors. Its applications include cyber-physical video game [1], robotic balancing [2], localization [3], sports training [4], medical care [5], and computer graphics [6].

A lot of works have studied how to track human postures. Basically, human motions can be tracked by estimating rotations of joints, via accelerometers, electric compasses, and/or gyroscopes [7]. Accelerometers sense the directions of the gravity, compasses sense the directions of the North, and gyroscopes estimate angular velocities. However, since a human body may continuously move, this is not an easy task. When the structures of articulated objects are unavailable, orientation tracking based on Kalman filtering data fusion techniques is studied in [3], [7], [8]. With the knowledge of human body structures, [9], [10] consider incorrect estimations of rotations that are out of the reachable regions of joints in a wearable micro-sensor network. Reference [11] further uses body model constraints to improve accuracy in motion capture. For velocity estimation in areas with magnetic disturbances, [12] uses four magnetometers forming an orthogonal trihedron to perceive the magnetic fields.

In this paper, we consider a fundamental issue in a BISN, the gravity measurement problem. Regarding a human body as multiple rigid parts connected by joints, we study the deployment of accelerometers on one rigid part and the estimation of the gravity on that rigid part. The acceleration perceived by an accelerometer is the gravity (which is the same for all accelerometers on the rigid part) plus its own acceleration seen by an outside observer. Since the orientation of the rigid part is unknown, how to measure the gravity is a challenging problem. Given a sensor deployment, [2] shows a data fusion scheme to extract the gravity. Based on [2], we formulate the deployment optimization problem as one of finding the best locations of accelerometers that minimize the estimation error of the gravity. Since the problem is hard, we conduct experiments to observe what the optimal deployments should be.

The rest of paper is organized as follows. Section II formulates the gravity measurement problem. Experimental results are given in Section III, and conclusions are drawn in Section IV.

II. Gravity Measurement Problem

We are interested in tracking human postures by deploying accelerometers on a human body. One fundamental issue in such scenarios is how to calculate the gravity, no matter when the body parts are moving or not. Fig. 1(a) shows an example. Clearly, the gravity as being measured by each sensor is relative to its placement and angle. We regard a human body as multiple movable *parts*, each being a *rigid body* connected to another part by a *rotational joint* [13], [14]. Accelerometers are placed on a human body for posture tracking. The concept is shown in Fig. 1(b).

To formulate the gravity measurement problem, we consider one rigid part and the sensors on it, as illustrated in Fig. 2. We assume that there is an Earth-fixed coordinate system, representing views of a fixed observer, with $\theta = (0, 0, 0)$ as its origin and x_θ, y_θ, and z_θ as its axes. The gravity g with respect to this θ-coordinate is thus -1 Guass along the z_θ axis. Let the joint of the rigid part be at location $r(t)$ at time t. For the rigid part, we assume a part-fixed coordinate with respect to the joint with $r(t)$ as its origin and x_r, y_r, and z_r as its axes. Therefore, for any point on the body, its coordinate with respect to the r-coordinate remains unchanged no matter how the body moves. For any point at location p with respect to the r-coordinate, its location with respect to the θ-coordinate changes over time t and can be written as $p'(t) = r(t) + R(t)p$, where $R(t)$ is the 3×3 rotation matrix translating from (x_r, y_r, z_r) to $(x_\theta, y_\theta, z_\theta)$ [13].

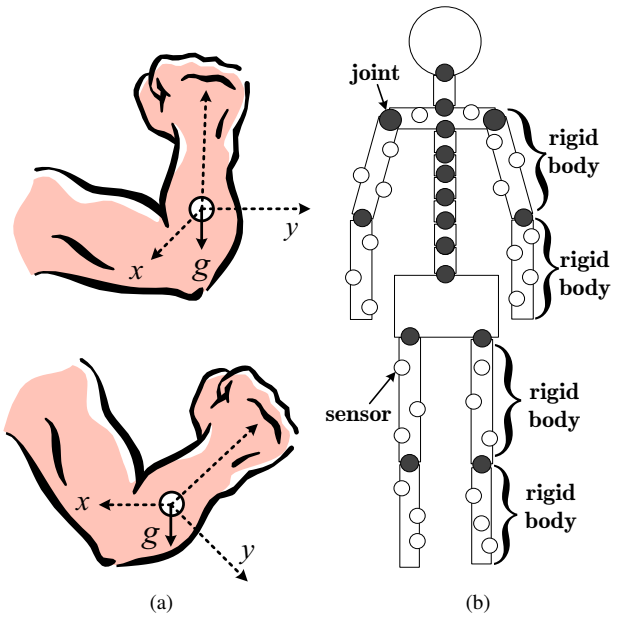


Figure 1. Gravity measurement and rigid body modeling.

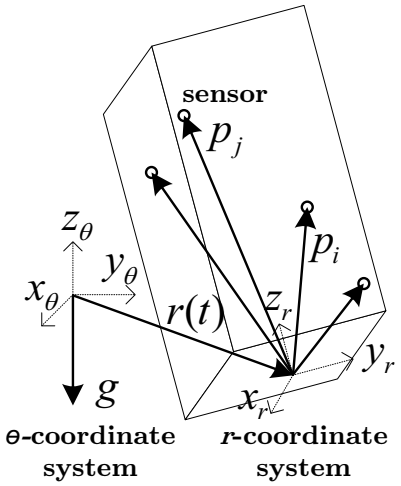


Figure 2. The kinematical model for a rigid body.

Let s_1, s_2, …, s_m be m accelerometers deployed on the rigid part and p_1, p_2, …, p_m be their locations with respect to the r-coordinate, respectively. Without loss of generality, we assume that these accelerometers are properly placed in the sense that their x-, y-, and z-axes are perfectly aligned to the x_r, y_r, and z_r axes of the r-coordinate, respectively (Otherwise, a rotation matrix from the r-coordinate to each sensor's coordinate would do the translation). This implies that the gravity being observed in the r-coordinate is the same as that being observed by any sensor.

Now, consider any sensor s_i, $i = 1, 2, \ldots, m$. Its location in the θ-coordinate at time t is $p'_i(t) = r(t) + R(t)p_i$ (note that p_i is time-invariant). Taking the second derivative of $p'_i(t)$, we have its acceleration in the θ-coordinate:

$$\ddot{p}'_i(t) = \ddot{r}(t) + \ddot{R}(t)p_i. \tag{1}$$

Since s_i is aligned to the r-coordinate and noise should be included, the actual reading $a_i(t)$ of s_i should be $a_i(t) = R^T(t)\left(-\ddot{p}'_i(t) + g\right) + n_i(t)$. Note that both $a_i(t)$ and $n_i(t)$ are 3×1 vectors, and each element of $n_i(t)$ has a zero mean with a standard deviation σ_n. (For example, when the rigid part is at rest, the reading of s_i should consist of only gravity, giving $a_i(t) = R^T(t)g + n_i(t)$; when it falls freely, the reading should be $a_i(t) = n_i(t)$.)

Below, assuming a fixed t, we will omit time information in our formulation. Plugging Eq. (1) into a_i, we have

$$\begin{aligned} a_i &= R^T\left(-\ddot{r} - \ddot{R}p_i + g\right) + n_i \\ &= \left[R^T\left(-\ddot{r} + g\right), \quad -R^T\ddot{R}\right]\begin{bmatrix}1\\ p_i\end{bmatrix} + \left[n_i\right]. \end{aligned}$$

Putting these m equations together, we have the equality:

$$A = QP + N, \tag{2}$$

where

$$\begin{aligned} A &= \begin{bmatrix} a_1 & \ldots & a_m \end{bmatrix} \in \Re^{3\times m}, \\ Q &= \left[R^T\left(-\ddot{r} + g\right), \quad -R^T\ddot{R}\right] \in \Re^{3\times 4}, \\ P &= \begin{bmatrix} 1 & \ldots & 1 \\ p_1 & \ldots & p_m \end{bmatrix} \in \Re^{4\times m}, \\ N &= \begin{bmatrix} n_1 & \ldots & n_m \end{bmatrix} \in \Re^{3\times m}. \end{aligned}$$

Here, A and P are known and Q is to be determined. P is called the *deployment matrix* of sensors s_1, s_2, …, s_m.

Since $\ddot{r} \ll g$ in Q, which is common for human motion, Q's first column vector $R^T(-\ddot{r} + g) \approx R^T g$. By estimating Q, we can determine $R^T g$. Let $\hat{Q}$ be an estimation of Q. Following [2], a $\hat{Q}$ that makes $\hat{Q} - Q$ as small as possible can be found by $\hat{Q} = AP^+$, where P^+ is the Moore-Penrose pseudoinverse of P when $m > 4$ and $P^+ = P^{-1}$ (the inverse of P) when $m = 4$ [15]. This implies that to find $\hat{Q}$ we need at least four sensors. Since $\hat{Q} - Q$ is a zero-mean vector, one needs to measure how $\hat{Q}$ is close to Q. In [2], an *error variance*, which solely depends on the deployment matrix P, is defined:

$$\sigma_e^2(P) = 3\sigma_n^2 \sum_{k=1}^{4} \frac{1}{\rho_k^2(P)}, \tag{3}$$

where $\rho_k(P)$ is the kth largest singular value of P. It is claimed that a smaller $\sigma_e^2(P)$ implies a more accurate $\hat{Q}$. Therefore, we formulate the deployment optimization problem as follows: given a rigid part and m accelerometers to be deployed on the surface of the part, the goal is to find the deployment matrix P such that the error variance $\sigma_e^2(P)$ is minimized. Note that since the $3\sigma_n^2$ in Eq. (3) is a constant, we only need to focus on the summation part.

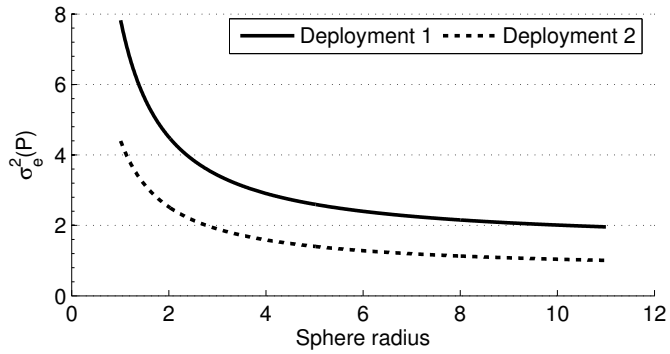


Figure 3. The radius of a sphere effects error variances.

III. Experimental Results

Since the problem is hard, we conduct experiments to observe what the optimal deployments should be.

A. Effects of Object Sizes

It is natural to ask how the size of a rigid part effects the estimation errors. To observe it, we vary the radius of a sphere from 1 to 10 and measure the error variance of some fixed deployment. That is, if the location of sensor s_i is p_i on a unit sphere, it becomes $10p_i$ on a sphere whose radius is 10. The results are shown in Fig. 3 for two deployments. We can see that the error variances decrease as the radius grows. The slope is quite steep at the beginning, implying that small objects are more vulnerable to sensor deployments.

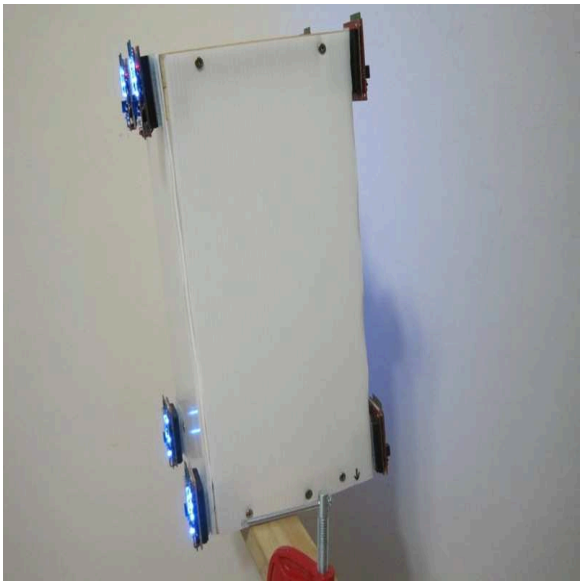
B. Gravity Estimation Results

We deploy eight accelerometers at the vertices of the box in Fig. 4 and want to measure the gravity when the box is moving. As shown in Fig. 4(a), a joint with one degree of freedom is at the left face of the box. We rotate the box quickly from Fig. 4(a) to Fig. 4(b) for many times, and record the sensing data for gravity measurement. The box and sensor positions in Fig. 4 are modeled as in Fig. 5.

Because the length of the true gravity $\|R^T(t)g\|$ is always 1000 mg (milliguass) no matter how we rotate the box, we compare different deployments by the lengths of their measured gravities. Fig. 6 compares the results of one sensor and four sensors. The estimated gravity of s_7 is its raw sensing data, and the estimated gravity of the four sensors is the first column of its $\hat{Q}$. Clearly, the measurement made by s_7 is significantly disturbed by motion, while the disturbance is reduced by using four sensors.

Let $\hat{g}(t)$ be the estimated gravity of a deployment at time t. We summarize its performance (as in Fig. 6) by the standard deviation of its errors $\|\hat{g}(t)\| - \|R^T(t)g\|$ (also called its standard error). The errors of each sensor's measurement is compared in Fig. 7. The sensors s_1, s_2, s_6, and s_8 have lower errors, whereas s_3, s_4, s_5, and s_7 have higher errors. It is natural to ask whether the four best positions result in an optimal measurement. Therefore, we compare the following five deployments:

(a)



(b)

Figure 4. Experiment setup.

1) Using one sensor at the position p_6.
2) Using four sensors at the positions p_3, p_4, p_5, and p_7.
3) Using four sensors at the positions p_1, p_2, p_6, and p_8.
4) Using four sensors at the positions p_1, p_3, p_6, and p_8.
5) Using all eight sensors.

The results are shown in Fig. 8. Using the four worst positions in deployment 2 is even worse than using only one sensor at p_6. Deploying at the four best positions, as in deployment 3, significantly reduces the errors. However, a better deployment can be found at deployment 4, which surprisingly combines a seemly erroneous position p_3. Finally, using all sensors is even more accurate.

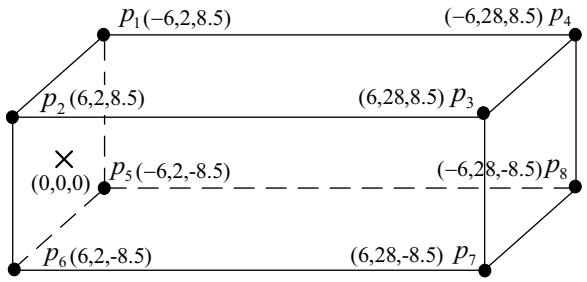


Figure 5. Sensors' locations on the box in Fig. 4.

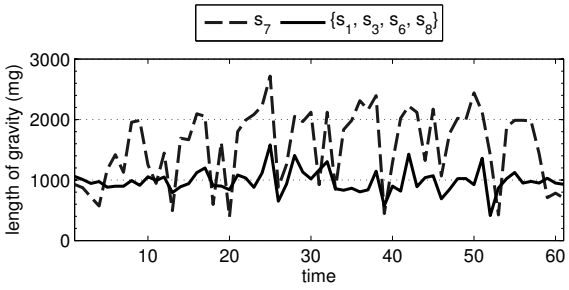


Figure 6. Comparison of using one sensor and four sensors.

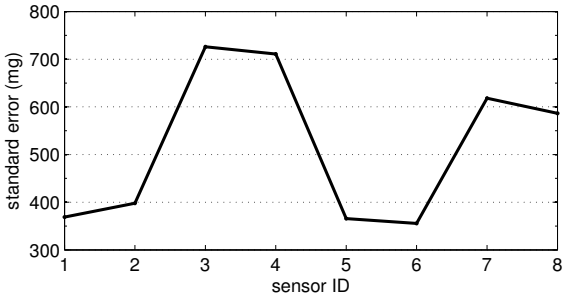


Figure 7. The errors of each sensor's measurement.

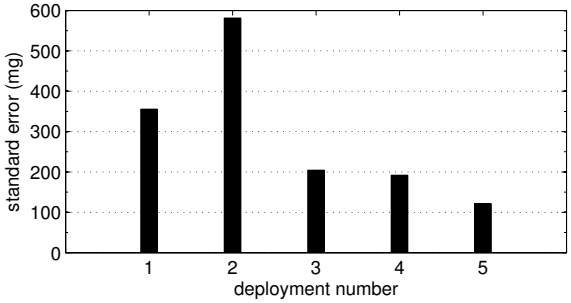


Figure 8. Comparison of different deployments.

IV. Conclusions and Future Work

In this paper, we have investigated how to perceive gravity by deploying multiple accelerometers on a rigid part. We have modeled the readings of an accelerometer in terms of the gravity and the sensor's own acceleration seen by an outside observer. Based on using the pseudoinverse scheme to extract the gravity from these readings, we have formulated the deployment optimization problem as one to find a deployment that minimizes the measurement error, which can be large for intuitive deployments. We have conducted experiments to observe solutions of this problem. It seems that deploying at the vertices of a box is the optimal. In our future work, we will investigate the optimal solutions for different geometries.

Acknowledgement

Y.-C. Tseng's research is co-sponsored by MoE ATU Plan, by NSC grants 97-3114-E-009-001, 97-2221-E-009-142-MY3, 98-2219-E-009-019, and 98-2219-E-009-005, 99-2218-E-009-005, by ITRI, Taiwan, by III, Taiwan, by D-Link, and by Intel.

References

[1] C.-H. Wu, Y.-T. Chang, and Y.-C. Tseng, "Multi-screen cyber-physical video game: An integration with body-area inertial sensor networks," in *Proc. of Int'l Conf. on Pervasive Comput. and Commun. (PerCom)*, 2010.

[2] S. Trimpe and R. D'Andrea, "Accelerometer-based tilt estimation of a rigid body with only rotational degrees of freedom," in *Proc. of IEEE Int'l Conf. on Robotics and Automation (ICRA)*, 2010.

[3] M. Sippel, A. Abduhl-Majeed, W. Kuntz, and L. Reindl, "Enhancing accuracy of an indoor radar by the implementation of a quaternion- and unscented kalman filter- based lightweight, planar, strapdown IMU," in *Proc. of the European Navigation Conf. (ENC-GNSS)*, 2008.

[4] D. T. W. Fong, J. C. Y. Wong, A. H. F. Lam, R. H. W. Lam, and W. J. Li, "A wireless motion sensing system using ADXL MEMS accelerometers for sports science applications," in *Proc. of World Congress on Intelligent Control and Automation*, 2004.

[5] C.-H. Wu and Y.-C. Tseng, "Data compression by temporal and spatial correlations in a body-area sensor network: A case study in pilates motion recognition," IEEE Trans. Mobile Comput., to appear.

[6] D. Vlasic, R. Adelsberger, G. Vannucci, J. Barnwell, M. Gross, W. Matusik, and J. Popović, "Practical motion capture in everyday surroundings," *ACM Trans. on Graphics*, vol. 26, no. 3, p. 35, 2007.

[7] J. K. Lee and E. J. Park, "A minimum-order kalman filter for ambulatory real-time human body orientation tracking," in *Proc. of IEEE Int'l Conf. on Robotics and Automation (ICRA)*, 2009.

[8] B. Huyghe, J. Doutreloigne, and J. Vanfleteren, "3D orientation tracking based on unscented kalman filtering of accelerometer and magnetometer data," in *IEEE Sensors Applications Symposium (SAS)*, 2009.

[9] Z. Zhang, L. W. Wong, and J.-K. Wu, "3d upper limb motion modeling and estimation using wearable micro-sensors," in

Proc. of Int'l Workshop on Wearable and Implantable Body Sensor Networks (BSN), 2010.

[10] Z. Zhang, "Ubiquitous human motion capture using wearable micro-sensors," in *Proc. of Int'l Conf. on Pervasive Comput. and Commun. (PerCom)*, 2009.

[11] A. D. Young, "Use of body model constraints to improve accuracy of inertial motion capture," in *Proc. of Int'l Workshop on Wearable and Implantable Body Sensor Networks (BSN)*, 2010.

[12] D. Vissiére, A. Martin, , and N. Petit, "Using distributed magnetometers to increase IMU-based velocity estimation in perturbed areas," in *Proc. of IEEE Int'l Conf. on Decision and Control*, 2007.

[13] J. Carig, *Introduction to Robotics*. Prentice Hall, New Jersey, 2005.

[14] A. D. Young, "Comparison of orientation filter algorithms for realtime wireless inertial posture tracking," in *Proc. of Int'l Workshop on Wearable and Implantable Body Sensor Networks (BSN)*, 2009.

[15] D. R. Basu and A. Lazaridi, "Stochastic optimal control by pseudo-inverse," *The Review of Economics and Statistics*, vol. 65, no. 2, pp. 347–350, 1983.

附錄二

A Lightweight, Self-Adaptive Lock Gate Designation Scheme for Data Collection in Long-Thin Wireless Sensor Networks

Y.-C. Wang, C.-H. Chuang, Y.-C. Tseng, and C.-C. Shen

Wireless Communications and Mobile Computing

A Lightweight, Self-Adaptive Lock Gate Designation Scheme for Data Collection in Long-Thin Wireless Sensor Networks

You-Chiun Wang[†], Che-Hsi Chuang[†], Yu-Chee Tseng[†], and Chien-Chung Shen[*]

[†]*Department of Computer Science, National Chiao-Tung University, Hsinchu 30010, Taiwan*
E-mail: {wangyc, chuangch, yctseng}@cs.nctu.edu.tw
[*]*Department of Computer and Information Sciences, University of Delaware, Newark, DE 19716, USA*
E-mail: cshen@cis.udel.edu

Abstract

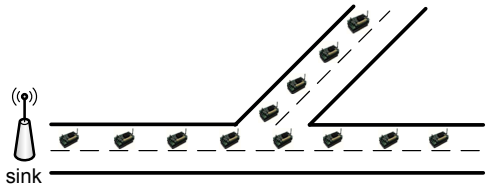
Constrained by the physical environments, the *long-thin* topology has recently been promoted for many practical deployments of wireless sensor networks (WSNs). In general, a long-thin topology is composed of a number of long branches of sensor nodes, where along a branch each sensor node has only one potential parent node toward the sink node. Although data aggregation may alleviate excessive packet contention, the maximum payload size of a packet and the dynamically changing traffic loads may severely affect the amount of sensor readings that may be collected along a long branch of sensor nodes. In addition, many practical applications of long-thin WSNs demand the exact sensor readings at each location along the deployment areas for monitoring and analysis purposes, so sensor readings may not be aggregated when they are collected. This paper proposes a lightweight, self-adaptive scheme that designates multiple collection nodes, termed *lock gates*, along a long-thin network to collect sensor readings sent from their respective upstream sensor nodes. The self-adaptive lock gate designation scheme balances between the responsiveness and the congestion of data collection while mitigating the funneling effect. The scheme also dynamically adapts the designation of lock gates to accommodate the time-varying sensor reading generation rates of different sensor nodes. A testbed of 100 Jennic sensor nodes is developed to demonstrate the effectiveness of the proposed lock gate designation scheme.

KEY WORDS: data collection, lock gates, long-thin networks, wireless sensor networks.

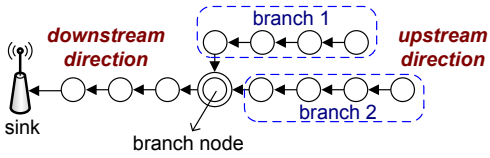
1. Introduction

A *wireless sensor network* (WSN) consists of a sheer number of sensor nodes, where each sensor node is a wireless device that reports sensor readings of its surroundings to a sink node via multi-hop ad hoc communications. Such networks facilitate pervasive monitoring of the physical environments to enable applications such as habitat monitoring, smart home, and surveillance [1, 2, 3].

In recent research, the *long-thin* topology has been promoted for many practical applications of WSNs where the sensor deployment is subject to environmental constraints [4]. For instance, a surveillance system of moving cars along streets, a monitoring system of carbon dioxide inside tunnels, and a monitoring system of water quality within underground sewer lines are typical applications. Fig. 1(a) depicts a physical deployment of sensor nodes along streets, and Fig. 1(b) depicts its corresponding long-thin network topology. In general, a long-thin topology is formed by a bunch of long *branches*, and each branch may be composed of tens



(a) the physical deployment of sensor nodes along streets



(b) the corresponding long-thin topology

Fig. 1. An example of long-thin WSNs.

or even hundreds of nodes. The structure of a branch is recursively defined, where a branch may contain other (sub)branches. For each sensor node along a branch, there exists only one potential parent node toward the sink node. Branches are grafted at *branch nodes*, and Fig. 1(b) shows an example where a branch node is denoted with double circles.

Let $\{s_1, s_2, \cdots, s_N\}$ denote the node IDs of a long-thin WSN, where N is the number of sensor nodes. By viewing a long-thin topology as a shortest-path tree rooted at the sink node, let d_i be the depth of sensor node s_i from the sink node. Assuming that each sensor node in the long-thin WSN delivers one sensor reading to the sink node without aggregating data, the network will incur $\sum_{i=1}^{N} d_i$ transmissions to collect all of the sensor readings. For instance, the long-thin WSN of Fig. 1 incurs 62 transmissions without data aggregation. Even worse, a long-thin WSN may suffer from the funneling effect where the hop-by-hop traffic over a long branch results in an increase in transit traffic intensity, collision, congestion, packet loss, and energy drain as packets move closer toward the sink node [5].

Now consider the following collection scheme. The leaf nodes start transmitting their sensor readings first. Each intermediate node *waits* to collect both its own sensor reading and the collected sensor readings sent from its children (or upstream nodes), and then forwards the collected packet toward the sink node. Such a scheme may result in only N transmissions in the network, assuming that successively collected sensor readings could be loaded into one huge packet. For instance, the long-thin WSN of Fig. 1 may incur only 12 transmissions using such a collection scheme.

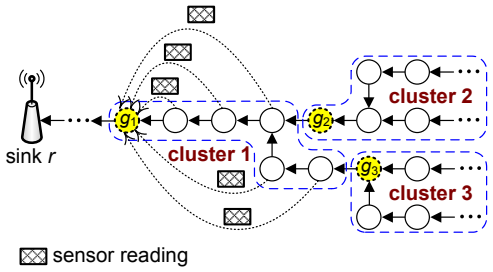


Fig. 2. A long-thin WSN and its lock gates.

Fewer transmissions benefit WSNs by reducing contention and conserving energy.

Although the above collection scheme maximally reduces the number of transmissions in a long-thin WSN, constraints imposed by the maximum payload size $L_{\max}$ of a packet and the compression ratio δ ($0 < \delta \leq 1$) make such a scheme impractical for a long-thin topology where a node can only collect up to $\lfloor \frac{L_{\max}}{\delta} \rfloor$ bytes of sensor readings, while the total data size of sensor readings generated by sensor nodes along a long branch can easily exceed this bound. Besides, many practical applications of long-thin WSNs demand the exact sensor readings of each location along the deployment areas for data monitoring and analysis purposes, so that sensor readings are not aggregated (such as computing the average, the minimum, or the maximum value) while they are being collected.

To mitigate the funneling effect and to comply with the maximum payload size, this paper suggests that in a long-thin network, multiple collection nodes, termed *lock gates*[*], should be designated, where each lock gate collects both its own sensor readings and all of the sensor readings sent from its upstream sensor nodes (up to immediate, upstream lock gates or the end of the branches) subject to the $\lfloor \frac{L_{\max}}{\delta} \rfloor$ bound. For instance, as shown in Fig. 2, assuming $L_{\max}$ is 3 bytes, δ is 0.5, and the size of sensor readings is 1 byte, we designate one lock gate every six sensor nodes so that lock gate g_1 collects the sensor readings of itself and its five upstream nodes into one collected packet[†]. Similar collections are done by lock gates g_2 and g_3. Such a collected packet (of maximum payload size $L_{\max}$) is

[*] We are inspired by the lock gates used in canals to withstand the water pressure arising from the level difference between adjacent pounds. In the context of long-thin WSNs, (irregular) network traffic from sensor readings corresponds to water pressure, which should be regulated to mitigate funneling effect, for instance.

[†] For the ease of explanation, in this example we simply assume that there is no 'protocol overhead' (or packet header/trailer) so that the size of a packet is exactly the same as the size of its payload.

said to be completely *filled up* with sensor readings and thus can be forwarded to the sink node without being padded with any other sensor readings along the way.

Another benefit of the above lock gate designation scheme is that it actually reduces contentions by spatially separating areas where packets are transmitted. In the above example, since lock gates g_2 and g_3 may *hold* their respective transmissions until enough sensor readings from the corresponding upstream nodes have been collected, the sensor nodes headed by lock gate g_1 can transmit their sensor readings to g_1 with less or even no interference coming from sensor nodes headed by lock gates g_2 and g_3.

Since sensor nodes may generate sensor readings at different *rates*, each lock gate may wait for a different amount of time to completely fill up one collected packet of payload size $L_{\max}$. Let λ_j denote the sensor reading generation rate (in bytes/second) of sensor node s_j. The rate λ_j may vary over time, but is assumed to change slowly so that a steady value of λ_j could be observed over a short period of time. Let $C(g_i)$ denote the *cluster* of nodes containing lock gate g_i and its upstream sensor nodes, up to the immediate, upstream lock gates or the end of the branches. Given λ_j for each sensor node s_j in $C(g_i)$, the expected time T_i that lock gate g_i takes to completely fill up one packet of maximum payload size $L_{\max}$ should satisfy:

$$\left(T_i \times \sum_{s_j \in C(g_i)} \lambda_j \right) \times \delta = L_{\max}. \qquad (1)$$

When T_i is large, the sink node is expected to wait for a longer time to receive a collected packet from lock gate g_i, which increases the application's response time. A large T_i may imply either the size of $C(g_i)$ is small or the total sensor reading generation rate of the sensor nodes in $C(g_i)$ is low, or both. On the other hand, a small T_i may imply either the size of $C(g_i)$ is large or the total sensor reading generation rate of the sensor nodes in $C(g_i)$ is high, or both. This may result in too many packet transmissions and thus congest the network. Therefore, lock gate designation should be made 'self-adaptive,' where T_i is bounded by dynamically designating the positions of lock gates to balance between the response time and the congestion of data collection. Furthermore, since sensor nodes are usually with limited computation power and small memory size [6], the self-adaptive lock gate designation scheme should be lightweight with simple operations.

This paper proposes a *lightweight, self-adaptive lock gate designation scheme*, termed *ALT*, to facilitate effective data collection in long-thin WSNs. Given a pair of time thresholds ($T_{\min}$, $T_{\max}$), ALT adaptively designates lock gates in a long-thin WSN such that for each lock gate g_i, the condition $T_{\min} \leq T_i \leq T_{\max}$ holds. The thresholds $T_{\min}$ and $T_{\max}$ are specified by the application of a long-thin WSN to avoid congestion from transmitting excessive packets and to impose an upper bound on response time, respectively. The ALT scheme possesses three characteristics. First, sensor nodes do not need to report their sensor reading generation rates λ_j to their corresponding lock gates. Instead, lock gates only need to *locally* observe their T_i values for the execution of ALT, so that ALT has low message overhead and good response time. The experimental results in Section 5.2 indeed show that ALT performs well even when λ_j changes. Second, ALT adopts a simple scheme to move lock gates in a 'hop-by-hop' manner so that the operations are light-weight and can be easily implemented on practical sensor platforms. Third, in contrast to conventional clustering protocols designed for WSNs with random topology, ALT incurs much less control overhead in long-thin WSNs because clusters are formed automatically whenever a lock gate and its upstream lock gates are designated. In addition, lock gates do not need to know the identities of their respective cluster members, and sensor nodes do not need to send their sensor reading generating rates to their corresponding lock gates.

The contributions of this paper are three-fold. First, we point out the necessity of lock gates to balance between the response time and the congestion of data collection in a long-thin WSN, and propose a lightweight, self-adaptive scheme to designate the lock gates. To the best of our knowledge, ALT is the first effort addressing efficient data collection in long-thin WSNs. Second, to evaluate its performance, we implement ALT on sensor nodes equipped with the Jennic wireless micro-controller supporting the IEEE 802.15.4 protocol [7], and develop a testbed containing 100 sensor nodes. Experimental results demonstrate that ALT adapts well to varying sensor reading generation rates and incurs fewer message transmissions than other schemes. In addition, we show the benefits of lock gates which significantly reduce both the number of data retransmissions and the amount of packet losses at the MAC layer. Third, since placing lock gates may increase the packet latency of sensor nodes, we derive the average extra packet latency caused by a lock gate via mathematical

analysis. We also validate the correctness of our analysis with measurements from real experiments.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 gives our problem statement. ALT and the analysis of its expected extra packet latency are described in Section 4. Section 5 presents our prototyping efforts and discusses experimental results. Conclusions are drawn in Section 6.

2. Related Work

Long-thin WSNs are widely used in many monitoring applications such as leakage detection within fuel pipes, stage measurements inside sewer, traffic adjustment along tunnels or highways, vibration detection of bridges, and flood protection of rivers. The long-thin topology is first studied in [4], which considers how to assign network addresses to sensor nodes in ZigBee-based long-thin WSNs to facilitate routing. In addition, a long-thin WSN is deployed along a river to monitor its water level. The work in [8] discusses how to detect and correct localization errors in a long-thin WSN. In addition, it adopts a weighted voting scheme to detect possible faulty sensor readings. However, none of the existing work on long-thin WSNs addresses the issue of data aggregation/compression.

The subject of data aggregation/compression in WSNs with random topologies has been extensively studied in the literature. Below, we categorize and review existing solutions.

Tree-based aggregation: The objective of tree-based aggregation schemes is to maximize a WSN's lifetime by jointly optimizing data aggregation and routing tree formation [9]. For instance, the work of [10] discusses how to find a set of data aggregation schedules to maximize the system's lifetime, where a *schedule* is defined as a collection of spanning trees rooted at the sink node. The work of [11, 12] proposes a data-centric approach to select an appropriate routing path to reduce energy consumption. TAG [13] organizes a WSN into a tree and proposes SQL-like semantics to aggregate streaming data into histograms. The work of [14] builds an aggregation tree according to the energy consumption of sensor nodes. Each node predicts the energy consumption of its potential parents and selects the one that can be left with the most energy as its parent. In the work of [15], one coding tree for raw data aggregation and one shortest-path tree for transmitting compressed data are built to deliver data to the sink node. In the work of [16], the effect of data aggregation on different routing schemes is studied. The work also proposes a static clustering scheme to achieve a near-optimal performance for various spatial correlations. The above research efforts focus on how to choose a good routing metric based on data attributes to facilitate aggregation. However, in long-thin WSNs, there usually exists at most one route from a sensor node to the sink node (*i.e.*, each sensor node has at most one potential parent node toward the sink node), so these existing tree-based solutions may not be directly applied.

Clustering-based aggregation: This category of schemes first group sensor nodes into clusters and then perform data aggregation within each cluster. The critical issue is how to select the *cluster head* in each cluster to aggregate data for its cluster members [17]. For instance, the scheme in [18] assigns weights to each node and the nodes with larger weights may become cluster heads, while the work of [19] favors nodes with more neighbors (*i.e.*, higher degrees) to become cluster heads. LEACH [20] assumes that each sensor node can be reachable in one hop and then assigns a fixed probability for each node to elect itself as a cluster head. In HEED [21], each sensor node uses its residual energy as the parameter to probabilistically select itself as a cluster head. The above research efforts discuss how to organize clusters such that nodes within a cluster are one-hop or k-hop away from the cluster head. In addition, SCT [22] proposes a ring-sector division clustering scheme, where sensor nodes in the same section are assembled into one cluster. Clearly, this ring-based approach cannot be used for long-thin WSNs. In contrast, ALT adaptively adjusts the size of clusters according to the amount of traffics generated from sensor nodes.

Chain-based aggregation: In these schemes, sensor nodes are organized into a linear chain for data aggregation. For instance, PEGASIS [23] organizes such a chain by adopting a greedy algorithm, where each sensor node selects its nearest neighbor (closer to the sink) as its *successor* along the chain and then sends its sensing readings to the successor. However, PEGASIS may not guarantee to minimize the total energy consumption of sensor nodes. Therefore, the work of [24] proposes a chain-construction scheme that minimizes the total energy consumption of sensor nodes by reducing the value of $\sum D^2$, where D is the distance between any two adjacent sensor nodes along the chain. The chain-based topology can be viewed as one special instance of the long-thin topology. Nevertheless, in the chain-based aggregation schemes, except for the node(s) at the end of the chain, all the

sensor nodes along the chain act as aggregators. In contrast, ALT dynamically selects a subset of sensor nodes to act as lock gates according to the sensor reading load.

Hierarchical aggregation: Several studies adopt a hierarchical architecture to aggregate or compress data in a WSN. The work of [25] first selects a subset of sensor nodes as level-1 aggregators. Then, among these level-1 aggregators, the scheme selects a subset of level-1 aggregators to act as level-2 aggregators. This procedure is repeated until level-h aggregators are selected. Then, sensor readings will be passed through each level of aggregators to the sink node. The studies [26, 27, 28] organize sensor nodes hierarchically and establish multi-resolution summaries of sensor data inside the network, through spatial and temporal compressions. However, such hierarchical architectures are not practical to be applied in long-thin WSNs.

Structure-free aggregation: The work of [29] considers aggregating data in a WSN without maintaining any structure. This work proposes a MAC protocol and studies the impact of randomized wait time to improve aggregation efficiency. However, this scheme may increase packet delays in long-thin WSNs.

Compared to prior aggregation/compression schemes, ALT exhibits two distinguishing features. First, while most of prior work consider selecting 'static' aggregators, ALT continuously adapts the positions of lock gates to balance between the responsiveness and the congestion of data collection. Second, while existing aggregation schemes are only to aggregate/compress the collected sensor readings, lock gates can control/adjust the amount of sensor readings generated within a cluster. By doing so, not only the amount of messages transmitted is significantly reduced but also concurrent data collections within individual clusters spatially isolated by lock gates take place.

With respect to energy conservation, many research efforts [30, 31, 32, 33, 34] exploit node redundancy to extend the network lifetime by selecting a subset of sensor nodes to be active while putting others to sleep to conserve energy. However, given the topology of long-thin WSNs where each sensor node usually has one potential parent node toward the sink node, such a sleep-active mechanism cannot be applied (otherwise, the network would be partitioned). In contrast, ALT strives to conserve energy by adapting the designation of lock gates to reduce the amount of messages transmitted for data collection.

3. Problem Statement

We model a long-thin WSN as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{r\} \cup \mathcal{S}$ contains the sink node r and the set of sensor nodes $\mathcal{S}$, and $\mathcal{E}$ contains all of the communication links. The topology of $\mathcal{G}$ may be represented as a tree rooted at sink node r. Each sensor node $s_j \in \mathcal{S}$ has a sensor reading generation rate λ_j, which may vary over time during the network operation. A sensor node is called a *branch node* if it has more than one child on $\mathcal{G}$. Fig. 1(b) gives an example, where the branch node is marked by double circles.

We define the direction toward the sink node as the *downstream direction* and the opposite direction as the *upstream direction*. Sensor nodes are grouped into non-overlapping clusters. For each cluster, the most downstream node is designated as a *lock gate*. For convenience, we call other nodes *regular sensor nodes*. A lock gate continuously collects the sensor readings from the upstream regular sensor nodes within its cluster. Whenever a lock gate collects enough sensor readings to completely fill up one packet with payload size $L_{\max}$, the lock gate sends out the packet, termed 'collected packet', toward the sink node. Such a collected packet will not be 'collected' again by other downstream lock gates, but will be directly relayed by the downstream nodes toward the sink node. Fig. 2 illustrates an example, where three clusters are formed and nodes g_1, g_2, and g_3 are designated as lock gates.

Given a pair of time thresholds $(T_{\min}, T_{\max})$, our objective is to designate lock gates (and thus adjust the corresponding clusters) such that for each lock gate g_i, the amount of time T_i to completely fill up one collected packet of payload size $L_{\max}$ satisfies the condition of $T_{\min} \leq T_i \leq T_{\max}$. Note that the thresholds $T_{\min}$ and $T_{\max}$ are specified by the applications of the long-thin WSN to prevent sensor nodes from transmitting excessive packets and to impose an upper bound on response time, respectively. For example, for non-time-critical applications where sensor nodes are requested to frequently report their monitoring data, we set a larger $T_{\min}$ value to avoid network congestion. On the other hand, in event-driven applications, a smaller $T_{\max}$ value is set to constrain the response time.

Table I summarizes the notations used in this paper. For the ease of presentation, the sensor reading generation rate λ_j is described as a steady-state variable, but in practice it may vary slowly during the operation of the network.

Table I. Summary of notations.

notation	definition
λ_j	the sensor reading generation rate of sensor node s_j
$C(g_i)$	the cluster of nodes containing lock gate g_i and its upstream sensor nodes
$L_{\max}$	the maximum payload size of a packet
δ	the compression ratio ($0 < \delta \leq 1$)
T_i	the amount of time for lock gate g_i to fill up one collected packet of payload size $L_{\max}$
$T_{\min}$	the lower-bound threshold for a lock gate to shrink its cluster
$T_{\max}$	the upper-bound threshold for a lock gate to expand its cluster
Δ_t	the waiting time for a lock gate to adjust one of its next lock gate if all of its next lock gates are busy
β	a system parameter to determine whether a lock gate enters the oscillating state or not

4. The Operations and Analysis of ALT

Given a long-thin WSN, ALT first randomly groups sensor nodes into several non-overlapping clusters that cover the entire network, and then designates their corresponding lock gates. Note that the sensor node closest to the sink node is always designated as a lock gate. Upon generating one sensor reading, each regular sensor node will send the reading toward its corresponding lock gate. Each lock gate g_i then collects the sensor readings from the regular sensor nodes within its cluster $C(g_i)$. After collecting enough sensor readings to fill up one packet of maximum payload size $L_{\max}$, lock gate g_i sends the collected packet toward the sink node. To reduce the latency of waiting to collect enough sensor readings to fill up one packet of maximum payload size $L_{\max}$, lock gate g_i may dynamically adjust the size of its cluster according to the duration T_i that it took to generate the previous collected packet (referring to Eq. (1)). When T_i is below the given lower-bound threshold $T_{\min}$, the total sensor reading generation rate within this cluster (*i.e.*, $\sum_{s_j \in C(g_i)} \lambda_j$) has become too high, and the collected packets will be sent to the sink node more often. In this case, lock gate g_i 'shrinks' its cluster by excluding certain sensor nodes to lower the total sensor reading generation rate. In contrast, when T_i is above the given upper-bound threshold $T_{\max}$, the total sensor reading generation rate within this cluster becomes too low. In this case, lock gate g_i 'expands' its cluster by including more sensor nodes to lower the latency of generating collected packets. Notice that within each cluster $C(g_i)$, the sensor readings sent from each regular sensor node $s_j \in C(g_i)$ may be relayed to lock gate g_i in a 'pipelining' manner subject to the contention of wireless transmissions. Thus, right before lock gate g_i sends out each (completely filled) collected packet toward the sink node, the percentage of sensor readings received from sensor s_j within this packet is approximately equal to

$$\frac{\lambda_j}{\sum_{s_k \in C(g_i)} \lambda_k}. \tag{2}$$

In other words, the amount of reported sensor readings from each sensor node is fairly proportional to the sensor reading generation rate of that sensor node.

Before describing ALT in details, we first define the terms used in the remainder of the paper. A lock gate g_k is called a *next lock gate* of lock gate g_i if g_k is an immediate upstream lock gate of g_i. In this case, g_i is the *previous lock gate* of g_k. For instance, in Fig. 2, g_2 is a next lock gate of g_1 while g_1 is a previous lock gate of g_2. Notice that each lock gate may have multiple next lock gates but has at most one previous lock gate. In addition, a lock gate is called a *leaf lock gate* if it has no next lock gate; otherwise, it is a *non-leaf lock gate*.

4.1. Adaptation of Lock Gate Designation

From an initial (random) lock gate designation, lock gates execute ALT *asynchronously*, while coordinating with previous and next lock gates. Each lock gate g_i measures its current T_i value, 'moves' one of its next lock gates (if necessary) either downstream or upstream by one hop, and recalculates its T_i value. This process is repeated until lock gate g_i settles at the condition of $T_{\min} \leq T_i \leq T_{\max}$. Specifically, for each non-leaf lock gate g_i, two possible cases need to be addressed: $T_i < T_{\min}$ and $T_i > T_{\max}$.

4.1.1. *Case of* $T_i < T_{\min}$

In this case, lock gate g_i 'shrinks' its cluster by first querying each of its next lock gates g_k for its T_k value. If lock gate g_k is also in the state of adjusting its own next lock gates, lock gate g_k will reply to lock gate g_i that itself is *busy*; otherwise, lock gate g_k will reply to lock gate g_i with its T_k value. If lock gate g_i concludes that all of its next lock gates are busy, it will wait for

a Δ_t time[‡] and then try again. Otherwise, lock gate g_i sends a *pull* message to one next lock gate g_k whose parent node, say, s_j on graph $\mathcal{G}$ is not a branch node *and* whose T_k value is the largest among all of lock gate g_i's non-busy next lock gates. Upon receiving such a pull message, lock gate g_k designates sensor node s_j to become a new lock gate and ceases being a lock gate. As a result, cluster $C(g_k)$ disappears and a new cluster $C(s_j)$ emerges. For convenience, we use the term 'move' to represent such an operation. However, in the case that lock gate g_i cannot find such a next lock gate (which means that the parent nodes of all of g_i's non-busy next lock gates on $\mathcal{G}$ are branch nodes), the next lock gate g_k that has the largest T_k value is asked to move one-hop downstream. These operations are repeated until lock gate g_i computes that $T_i \geq T_{\min}$.

Fig. 3(a) gives an example, where g_1 wants to adjust one of its next lock gates g_2 and g_3. Since the parent node of lock gate g_2 is a branch node, lock gate g_3 will be asked to move downstream. When the parent nodes of all of g_i's next lock gates are all branch nodes, as shown in Fig. 3(b), assuming $T_2 > T_3$, lock gate g_2 will be asked to move to node b.

Notice that, when shrinking a cluster, ALT gives priority to move a lock gate (one-hop downstream) whose parent node on $\mathcal{G}$ is *not* a branch node. Doing so avoids excluding too many sensor nodes all at once when a lock gate is shrinking its cluster, which may *otherwise* drastically increase its T_i value, and/or creating a new cluster with drastically increased cluster size (or decreased T_i value) due to merging of branches. Fig. 4(a) and Fig. 4(b) together depict one counterexample, where we assume $T_2 > T_3$. When lock gate g_1 simply moves one next lock gate whose T_i value is the largest, lock gate g_2 will be moved downstream, as shown in Fig. 4(b). As a result, the size of cluster $C(g_1)$ decreases drastically from 7 to 3, while the size of cluster $C(g_2)$ increases drastically from 8 to 12. In fact, ALT moves g_3 one-hop downstream instead.

4.1.2. *Case of* $T_i > T_{\max}$

In this case, lock gate g_i 'expands' its cluster by first querying each of its next lock gate g_k for its T_k value. If lock gate g_k is also in the state of adjusting its own next lock gates, lock gate g_k will reply to lock gate g_i that itself is busy; otherwise, lock gate g_k will reply to lock gate g_i with its T_k value. If lock gate g_i concludes that all of its next lock gates are busy, it will wait for a Δ_t time and try again. Otherwise, lock gate g_i sends a *push* message to move one next lock gate g_k, that is not a branch node *and* has the smallest T_k value, one-hop upstream. In the case that lock gate g_i cannot find such a next lock gate (which means that all of g_i's non-busy next lock gates are branch nodes), one next lock gate g_k that has the least T_k value will be moved one-hop upstream. These operations are repeated until lock gate g_i computes that $T_i \leq T_{\max}$.

Fig. 3(c) gives an example, where g_1 wants to adjust one of its next lock gates g_2 and g_3. Since lock gate g_2 is not a branch node, it will be moved upstream. When

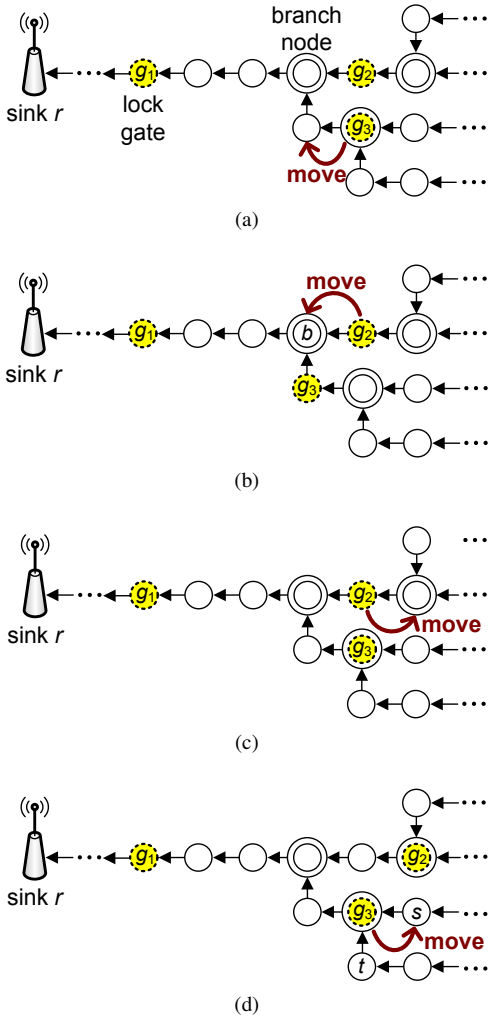


Fig. 3. Examples of moving next lock gates: (a) g_1 moves g_3 downstream since $T_1 < T_{\min}$, (b) g_1 moves g_2 to the branch node b since $T_1 < T_{\min}$, (c) g_1 moves g_2 upstream since $T_1 > T_{\max}$, and (d) g_1 moves g_3 to node s since $T_1 > T_{\max}$.

[‡]One possibility is to set $\Delta_t = T_{\max}$ so that one of g_i's next lock gates may become *non-busy*.

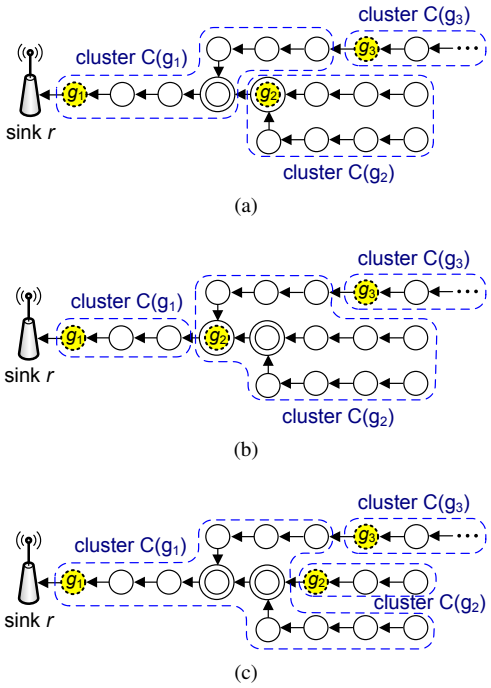


Fig. 4. Counterexamples of moving next lock gates: (a) the original clustering result, (b) g_1 simply moves g_2 downstream and thus drastically decreases the size of $C(g_1)$ while drastically increases the size of $C(g_2)$, and (c) g_1 simply moves g_2 upstream and thus drastically increases the size of $C(g_1)$ while drastically decreases the size of $C(g_2)$.

all of the next lock gates are branch nodes, as shown in Fig. 3(d), assuming that $T_3 < T_2$, lock gate g_3 will be asked to move to node s. Notice that in the latter case, the node in which lock gate g_3 used to reside, node t, and some of node t's upstream nodes will be merged into cluster $C(g_1)$.

In contrast to shrinking a cluster, ALT gives priority to move a lock gate (one-hop upstream) that is *not* a branch node when expanding a cluster. Doing so avoids including too many sensor nodes all at once when expanding a cluster, which may *otherwise* drastically decrease its T_i value due to merging of branches, and/or creating a new cluster with drastically decreased cluster size (or increased T_i value). Fig. 4(a) and Fig. 4(c) together show a different counterexample with the assumption that $T_2 < T_3$. When lock gate g_1 simply moves its next lock gate whose T_i value is the smallest, lock gate g_2 will be moved one-hop upstream, as shown in Fig. 4(c). As a result, the size of cluster $C(g_1)$ increases drastically from 7 to 12, while the size of cluster $C(g_2)$ decreases drastically from 8 to 3. In fact, ALT moves g_3 one-hop upstream instead.

In ALT, sensor nodes do not need to report their sensor reading generation rates λ_j to their corresponding lock gates. Thus, the control overhead is only incurred by transmitting query, reply, push, and pull messages between two adjacent lock gates. In Section 5.1, we will show that ALT's control overhead is only a small portion compared to the actual amount of data payloads.

4.2. Handling of Leaf Lock Gates

For each leaf lock gate g_l, it will be only moved according to the push or pull requests from its previous lock gate. However, two special cases should be considered. First, when lock gate g_l is asked to move upstream but itself is already a leaf node on $\mathcal{G}$, g_l will simply cease being a lock gate and work as a regular sensor node. In this case, the total number of lock gates in the network decreases by one. Second, after lock gate g_l has been moved downstream and computes that $T_l < T_{\min}$, lock gate g_l will select one leaf node, say, s_j from its cluster and designate s_j as a new (leaf) lock gate. In this case, the total number of lock gates in the network increases by one.

4.3. Handling of Oscillating Lock Gates

To prevent lock gates from 'oscillating' or moving back and forth between two adjacent nodes, each lock gate g_i maintains a short list recording its past positions on $\mathcal{G}$. If lock gate g_i finds that it has moved between two adjacent nodes (termed *oscillating nodes*) more than β times[§] and its previous lock gate still asks it to move to one of the oscillating nodes, lock gate g_i enters the *oscillating state* and requests its previous lock gate to stop asking it to move. Lock gate g_i will exit the oscillating state when either its previous lock gate asks it to move to one non-oscillating node or a pre-configured oscillating timer expires.

When each sensor node has a fixed sensor reading generation rate, the lock gates designated by ALT will eventually stabilize and converge (from an initial random designation) due to the following two factors. First, a lock gate can only move its next lock gates but cannot move its previous lock gate. In this case, clusters can stabilize *in sequence* from the downstream direction to the upstream direction. Second, ALT employs the above oscillation avoidance technique. In

[§]The β value will affect the convergence speed of ALT. When a fast convergence is desired, a smaller β can be set.

this case, if a lock gate finds that all of its next lock gates enter the oscillating state, the lock gate will stop moving its next lock gates (but may try later after Δ_t time).

4.4. Issue of Fault Tolerance

Till now, our discussion focuses on the assumption that each sensor node has only one potential parent node toward the sink node. For the reliability reason, a long-thin network may be deployed such that each sensor node can reach at least two downstream neighbors, as shown in Fig. 5(a). In such a deployment, sensor nodes will deliver their sensor readings to the sink node through the primary links. However, when some sensor nodes fail or run out of energy, neighboring nodes can use the secondary links to forward their data. Fig. 5(b) and Fig. 5(c) together depict an example. Initially, we have two paths $c \rightarrow b \rightarrow a \rightarrow \cdots \rightarrow r$ and $f \rightarrow e \rightarrow d \rightarrow \cdots \rightarrow r$ (formed via primary links), where node r is the sink node. Supposing that node b fails, the forwarding path from node c to sink node r changes from the original path $c \rightarrow b \rightarrow a \rightarrow \cdots \rightarrow r$ to the new path $c \rightarrow f \rightarrow e \rightarrow d \rightarrow \cdots \rightarrow r$. In this case, the upstream sensor nodes of node b can still transmit their sensor readings to sink node r, even though node b fails. Such a deployment ensures a higher degree of fault tolerance for long-thin WSNs.

ALT is designed to handle the following two cases:

- If a regular sensor node cannot reach its original lock gate, the sensor node will rejoin a nearest downstream cluster via a secondary link.
- If the new forwarding path from a regular sensor node s_i to its original lock gate g_j contains other lock gates, s_i will leave the original cluster $C(g_j)$ and join the new cluster $C(g_k)$, where g_k is s_i's nearest downstream lock gate.

The first case applies when the lock gate fails or the new forwarding path cannot reach the original lock gate. The second case applies when some regular sensor nodes fail and the new forwarding path to the original lock gate passes through other lock gates. Fig. 5(b) and Fig. 5(c) together depict an example. Initially, we have two clusters $C(a)$ and $C(d)$ with the lock gates a and d, respectively. When node b fails, the new forwarding path formed from node c to sink node r is no longer through lock gate a. This scenario fits the first case above and thus node c joins the new cluster $C(d)$. From Fig. 5(c), we can observe that lock gate a becomes a leaf lock gate and the size of cluster

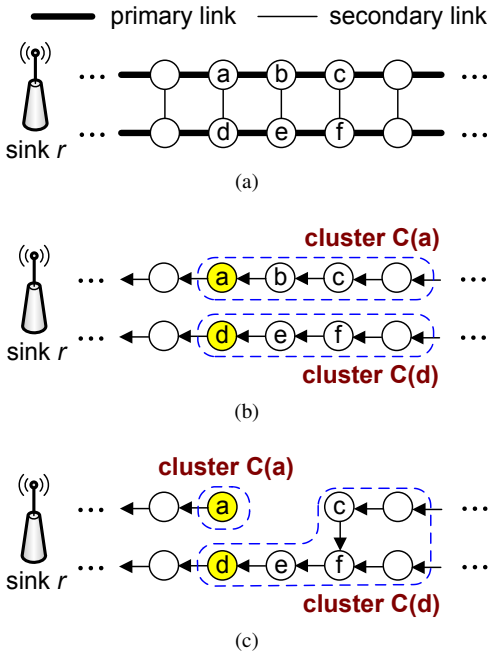


Fig. 5. Fault-tolerated deployment of a long-thin WSN: (a) the network topology, (b) sensor nodes transmit their sensor readings through the primary links, and (c) node c uses the secondary link to reach the sink node.

$C(d)$ increases. According to the rules of ALT, lock gate a may disappear and lock gate d may move its next lock gates toward the downstream direction.

4.5. Analysis of Expected Extra Packet Latency Caused by A Lock Gate

Clearly, a lock gate will delay the forwarding of sensor readings within its cluster toward the sink node. Below, we analyze the expected extra packet latency caused by a lock gate. We consider a cluster $C(s_1)$ that consists of a line of K sensor nodes s_1, s_2, $\cdots$, and s_K, where node s_1 is the lock gate, as shown in Fig. 6. To simplify the analysis, we assume that each sensor reading has the same payload size of L (bytes) and the average latency to forward a sensor reading over one-hop distance is τ. In addition, we assume that each sensor node s_j, $j = 1..K$, does not change its sensor reading generation rate (that is, λ_j enters the steady state) during which lock gate s_1 waits to collect enough sensor readings to fill up one packet of maximum payload size $L_{\max}$ (*i.e.*, T_i).

Let n_j be the number of sensor readings[¶] generated by sensor node s_j, $j = 1..K$, during T_i. Note that

[¶]Below, we use the term 'packet' for sensor reading.

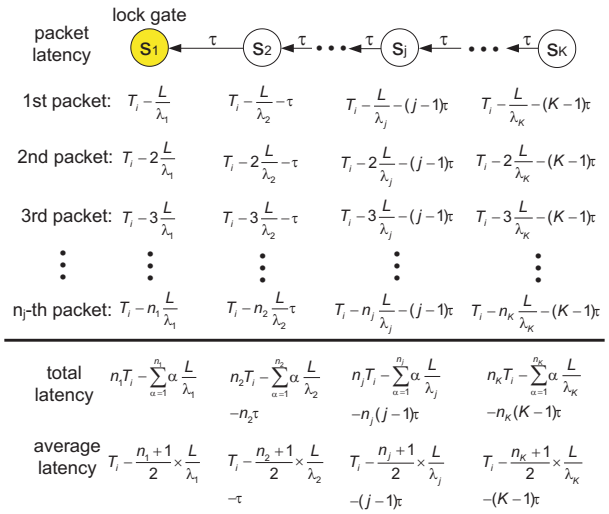


Fig. 6. Analysis of extra packet latency by a lock gate.

since λ_j does not change during T_i, we have $\lambda_j = \frac{n_j}{T_i}$. For node s_1, its first packet is generated after time $\frac{L}{\lambda_1}$. Since node s_1 is the lock gate, this first packet has to wait a duration of $T_i - \frac{L}{\lambda_1}$ before s_1 sends out a collected packet. Similarly, the second packet of node s_1 has to wait a duration of $T_i - 2\frac{L}{\lambda_1}$ and the last packet of node s_1 has to wait a duration of $T_i - n_1\frac{L}{\lambda_1}$. In this case, the average extra packet latency of node s_1 is

$$\frac{1}{n_1}\left((T_i - \frac{L}{\lambda_1}) + (T_i - 2\frac{L}{\lambda_1}) + \cdots + (T_i - n_1\frac{L}{\lambda_1})\right)$$
$$= \frac{1}{n_1}\left(n_1 T_i - \sum_{\alpha=1}^{n_1}\alpha\frac{L}{\lambda_1}\right) = T_i - \frac{n_1+1}{2}\times\frac{L}{\lambda_1}.$$

For node s_2, since it takes time τ to send a packet to node s_1, node s_2's first packet has to wait a duration of $T_i - \frac{L}{\lambda_2} - \tau$. Similarly, the second packet and the last packet of node s_2 have to wait durations of $T_i - 2\frac{L}{\lambda_2} - \tau$ and $T_i - n_2\frac{L}{\lambda_2} - \tau$, respectively. Thus, the average extra packet latency of node s_2 is

$$T_i - \frac{n_2+1}{2}\times\frac{L}{\lambda_2} - \tau.$$

Similarly, the average extra packet latency of node s_j is

$$T_i - \frac{n_j+1}{2}\times\frac{L}{\lambda_j} - (j-1)\tau.$$

Fig. 6 shows the extra packet latency of each sensor node. Therefore, the average extra packet latency of all sensor nodes within cluster $C(s_1)$ incurred by lock gate s_1 is

$$\frac{1}{K}\left((T_i - \frac{n_1+1}{2}\times\frac{L}{\lambda_1}) + (T_i - \frac{n_2+1}{2}\times\frac{L}{\lambda_2} - \tau) + \cdots + (T_i - \frac{n_K+1}{2}\times\frac{L}{\lambda_K} - (K-1)\tau)\right)$$
$$= \frac{1}{K}\left(KT_i - \frac{L}{2}\sum_{\alpha=1}^{K}\frac{n_\alpha+1}{\lambda_\alpha} - \sum_{\alpha=1}^{K-1}\alpha\tau\right)$$
$$= T_i - \frac{L}{2K}\sum_{\alpha=1}^{K}\frac{n_\alpha+1}{\lambda_\alpha} - \frac{K-1}{2}\tau. \quad (3)$$

According to Eq. (1), we can derive that

$$T_i = \frac{L_{\max}}{\delta \times \sum_{\alpha=1}^{K}\lambda_\alpha}. \quad (4)$$

Let n_{total} be the total number of packets 'generated' by all sensor nodes during T_i, so we have $n_{\text{total}} = \sum_{j=1}^{K} n_j$. Since each packet has a payload size of L, we can obtain that

$$(n_{\text{total}} \times L) \times \delta = L_{\max} \Rightarrow n_{\text{total}} = \left\lceil \frac{L_{\max}}{L \times \delta} \right\rceil.$$

From Eq. (2), each sensor node s_j, $j = 1..K$, will generate n_j packets during T_i:

$$n_j = \frac{\lambda_j}{\sum_{\alpha=1}^{K}\lambda_\alpha} \times n_{\text{total}}$$
$$= \left\lceil \frac{\lambda_j}{\sum_{\alpha=1}^{K}\lambda_\alpha} \times \frac{L_{\max}}{L \times \delta} \right\rceil. \quad (5)$$

Therefore, the average extra packet latency of all sensor nodes within a cluster can be calculated by substituting Eqs. (4) and (5) into Eq. (3).

5. Implementation and Experimental Results

In this section, we describe our prototyping efforts and discuss the experimental results. We use one hundred sensor nodes and one sink node. Fig. 7 pictures one deployment scenario of our prototype. Each sensor node is equipped with a Jennic JN5139 chip [7] containing a micro-controller and an IEEE 802.15.4 transceiver. The transmission power of each sensor node has been adjusted to have a communication distance of approximately 30 centimeters (cm). We place two adjacent nodes with a distance of 15 cm so that network connectivity can be guaranteed. An

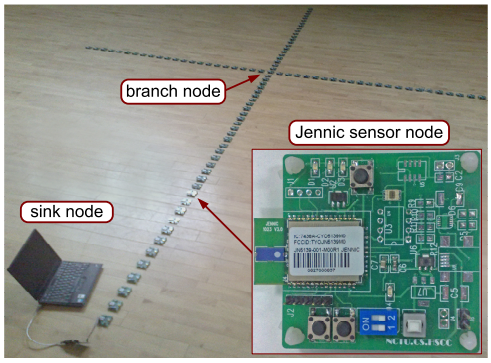
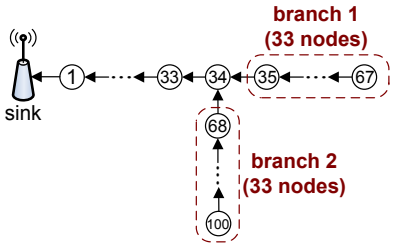
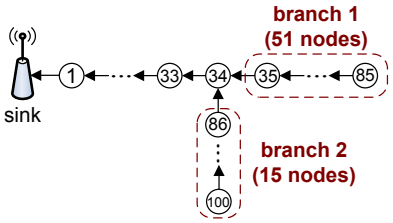


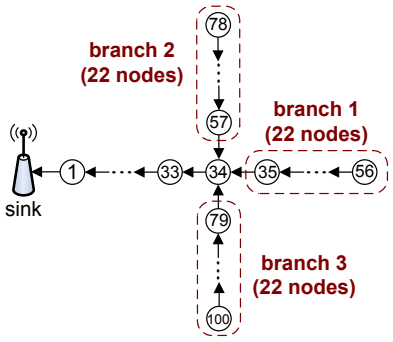
Fig. 7. The prototype of our long-thin WSN experiments.



(a) the balanced topology



(b) the unbalanced topology



(c) the cross topology

Fig. 8. Three long-thin topologies in our experiments.

application-layer protocol has also been implemented to enforce packet reception from adjacent neighbors only. Three long-thin topologies are deployed in our experiments. The *balanced* topology (referring to Fig. 8(a)) has two branches, where each branch contains 33 sensor nodes. The *unbalanced* topology (referring to Fig. 8(b)) has two branches, where the long branch contains 51 sensor nodes and the short branch contains 15 sensor nodes. The *cross* topology (referring to Fig. 8(c)) has three branches, where each branch contains 22 sensor nodes. We compare the performance of ALT against the *brute force* (BF) and the *fixed lock gate selection* (FLS) schemes using these three network topologies. BF does not apply any collection mechanism, so each sensor node simply relays the sensor readings from its upstream nodes to the sink node. Using FLS, each branch node is designated as a lock gate and we do not adjust the designation of lock gates during the experiments.

The size of a packet carrying one sensor reading is 15 bytes, which consists of a header of 12 bytes (according to the IEEE 802.15.4 standard [35]) and a sensor reading payload of 3 bytes. Each sensor node reports its sensor reading every Δ_s seconds, where Δ_s is randomly selected from $[(1-\gamma)\times 10, (1+\gamma)\times 10]$, and it may be changed every 30 seconds. Notice that when $\gamma = 0$, the sensor reading generation rate (*i.e.*, λ_i) is 0.3 bytes/second. For each lock gate, we adopt a simple collection scheme by removing the packet headers of the received sensor reading packets and then concatenating their payloads into one single packet. In this way, we have $\delta = 1$. Since the maximum payload size of a packet defined in the IEEE 802.15.4 standard is 118 bytes, each lock gate can collect at most 39 sensor readings, so we have $L_{\max} = 39 \times 3 = 117$ bytes. The total experiment time is 10 minutes. In the experiments of running ALT, the measurement of messages sent by sensor nodes includes all of the control messages (*e.g.*, query, reply, push, and pull) used to adjust the designation of lock gates. Other parameters used in the experiments are set as follows: $\beta = 3$, $\Delta_t = 2$ seconds, $T_{\min} = 24$ seconds, and $T_{\max} = 26$ seconds.

5.1. Communication Costs

We use the amount of messages and the number of packets successfully forwarded to the sink node to measure communication cost of data collection. Table II lists the total amount of messages (in bytes) successfully forwarded[‖] to the sink node by sensor nodes in different network topologies. Without using any collection scheme, BF exhibits the highest amount of messages. By dynamically adjusting the positions

[‖]Due to wireless contention and impairment, packet losses and retransmissions do occur, which are evaluated in Section 5.3.

Table II. Comparison on the total amount of messages (in bytes) sent by sensor nodes in different network topologies.

γ value	scheme	balanced	unbalanced	cross
$\gamma = 0$	ALT	498,530	581,550	468,982
	FLS	668,962	785,707	572,769
	BF	928,926	1,022,398	864,898
$\gamma = 0.3$	ALT	525,009	606,598	502,592
	FLS	680,349	801,486	596,287
	BF	943,384	1,096,538	871,232

Table III. Comparison on the total number of packets sent by sensor nodes in different network topologies.

γ value	scheme	balanced	unbalanced	cross
$\gamma = 0$	ALT	33,865	35,517	32,225
	FLS	99,915	128,396	78,767
	BF	238,899	267,690	217,392
$\gamma = 0.3$	ALT	35,878	35,788	33,033
	FLS	103,750	128,563	81,621
	BF	240,855	282,288	218,911

of lock gates according to the network condition, ALT enjoys a lower amount of messages compared with FLS. It can be observed that when $\gamma = 0$, ALT saves 18.1% $\sim$ 26.0% and 43.1% $\sim$ 46.3% of the amount of messages compared with FLS and BF, respectively. When $\gamma = 0.3$, ALT saves 15.7% $\sim$ 24.3% and 42.3% $\sim$ 44.7% of the amount of messages compared with FLS and BF, respectively. These results show the effectiveness of ALT. Notice that the three schemes all suffer from the highest amount of messages under the unbalanced topology, because this topology has the longest branch (with 51 sensor nodes).

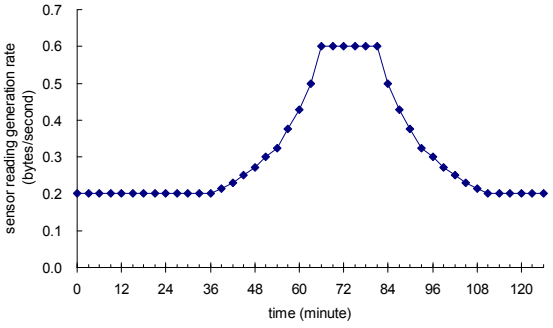
Table III lists the total number of packets successfully forwarded to the sink node by sensor nodes in different network topologies. Using BF, sensor nodes forward the most number of packets, because they simply relay sensor readings to the sink node. ALT incurs the smallest number of packets among all three schemes because it adaptively clusters sensor nodes via lock gates and collects their packets accordingly. It can be observed that when $\gamma = 0$, ALT saves 59.1% $\sim$ 72.3% and 85.2% $\sim$ 86.7% of the number of packets compared with FLS and BF, respectively. When $\gamma = 0.3$, ALT saves 59.5% $\sim$ 72.2% and 84.9% $\sim$ 87.3% of the number of packets compared with FLS and BF, respectively. These results demonstrate that ALT significantly reduces the number of packets forwarded by sensor nodes, which can greatly alleviate network congestion and conserve energy.

Note that our measurement includes all the control messages. From Tables II and III, we can observe that ALT incurs very low traffics even when control overheads are included. Thus, the impact of control overheads caused by ALT's adaption of lock gates is light-weight.

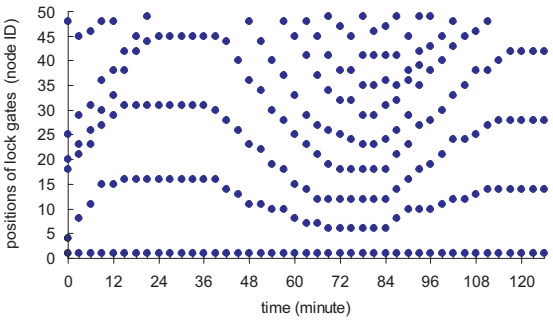
5.2. Adaptive Designation of Lock Gates

To demonstrate the adaptability of ALT to varying sensor reading generation rates, we deploy a sink node (of ID 0) and a line of 50 sensor nodes (of IDs 1 to 50), where the node with ID 1 is the most downstream sensor node. The duration of the experiment is 126 minutes and we monitor the (changing) number of lock gates and their designation (or positions) over time. All of the sensor nodes have the same sensor reading generation rate (λ), which changes every 3 minutes as shown in Fig. 9(a). For instance, starting at 0.2 bytes/second, λ remains at the same rate until the 36th minute, increases to 0.6 bytes/second at the 66th minute, remains at the same rate until the 81st minute, and then decreases. Fig. 9(b) depicts the changing designation of lock gates, as dots, over time. For instance, at the 0th minute, there are 6 lock gates randomly designated at nodes of IDs 1, 4, 18, 20, 25, and 48. Before the 36th minute, λ is not changed and thus the positions of lock gates stabilize at nodes of IDs 1, 16, 31, and 45 at the 24th minute. When λ increases, the size of the clusters decreases and thus the number of lock gates increases accordingly. Between the 66th and the 81st minutes, λ remains stable and thus the designation of lock gates are only slightly adjusted. For instance, at the 72nd minute, 8 lock gates are designated at nodes of IDs 1, 6, 12, 18, 25, 32, 38, and 47. After the 81st minute, when λ decreases, the number of lock gates also decreases and the size of clusters increases. After the 117th minute, the designation of lock gates remains stable because λ does not change. Since all of the sensor nodes have the same λ value, we also observe that the distance between any two adjacent lock gates is quite similar at most time instances. Such a phenomenon is more visible when the number of lock gates is smaller. These observations demonstrate that ALT can efficiently adjust the size of each cluster (and designate the lock gate accordingly) based on the traffic sent from the sensor nodes in that cluster.

Using the same network topology in the previous experiment, we also demonstrate the adaptability of ALT when sensor nodes have different sensor reading generation rates (λ). Specifically, the λ value of sensor nodes of IDs 1 to 25 increases while that of sensor nodes of IDs 26 to 50 decreases over time, as shown in Fig. 10(a). For convenience, we use the terms 'downstream part' and 'upstream part' to represent

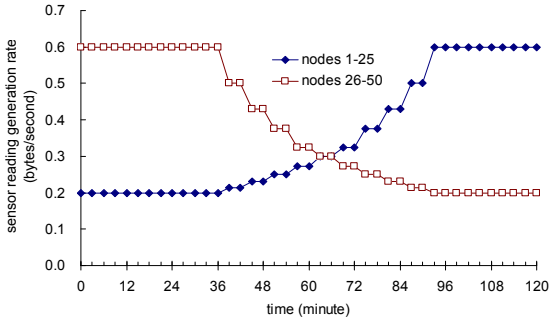


(a) the sensor reading generation rate

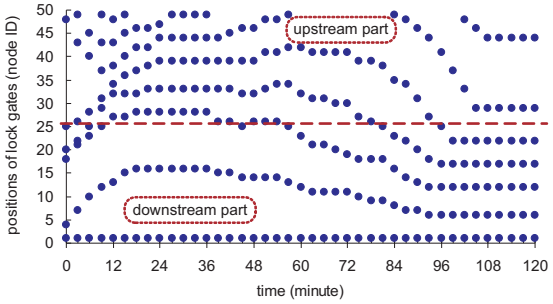


(b) the positions of lock gates

Fig. 9. The change of the positions of lock gates when all sensor nodes have the same sensor reading generation rate.



(a) the sensor reading generation rates



(b) the positions of lock gates

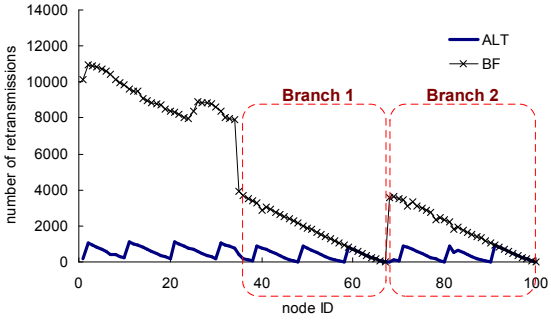
Fig. 10. The change of the positions of lock gates when sensor nodes have different sensor reading generation rates.

the sensor nodes of IDs 1 to 25 and of IDs 26 to 50, respectively. The duration of the experiment is 120 minutes. Beginning with the same random lock gate designation as the previous experiment, Fig. 10(b) shows the changes of lock gate designation over time. We observe that before the 60th minute, most lock gates are located at the upstream part because sensor nodes in the upstream part have a higher sensor reading generation rate. Thus, ALT shrinks the sizes of clusters in the upstream part and thus designates more lock gates. After the two sensor reading generation rates cross around the 66th minute, the behavior reverses itself such that most lock gates move to the downstream part because sensor nodes in the downstream part have a higher sensor reading generation rate.

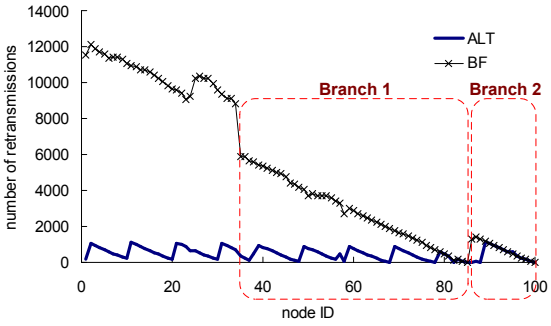
5.3. Impact on MAC-layer behaviors

We also evaluate the impact of lock gates on the MAC-layer behaviors in terms of the number of packet retransmissions and the number of packet losses by reusing the three network topologies in Fig. 8. We assign sensor nodes with IDs 1 to 100, where the node with ID 1 is the most downstream sensor node. In the balanced topology, 10 sensor nodes with IDs 1, 10, 20, 30, 38, 48, 58, 70, 80, and 90 are designated as lock gates. In the unbalanced topology, 11 sensor nodes with IDs 1, 10, 20, 30, 38, 48, 58, 68, 78, 88, and 98 are designated as lock gates. In the cross topology, 10 sensor nodes with IDs 1, 10, 20, 30, 37, 47, 59, 69, 80, and 90 are designated as lock gates. Each sensor node reports its sensor reading every 5 seconds, so the sensor reading generation rate λ_i is 0.6 bytes/second. The duration of each experiment is 10 minutes.

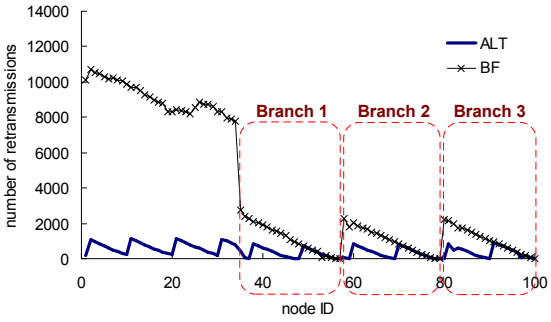
Fig. 11 compares the number of MAC-layer retransmissions incurred by ALT and BF in the three long-thin topologies. Without lock gates, we observe that the length of a branch directly affects the number of sensor readings generated, and hence negatively affects the number of packet retransmissions due to contention, such that the unbalanced topology (containing the longest branch) incurs more retransmissions. The branch node with ID 34 suffers a steep jump of the number of retransmissions in comparison to its neighboring upstream nodes due to the heavy contention caused by the traffic coming from upstream nodes. In contrast, using ALT, lock gates dynamically adjust the cluster sizes according to the traffic loads, so that the



(a) the balanced topology

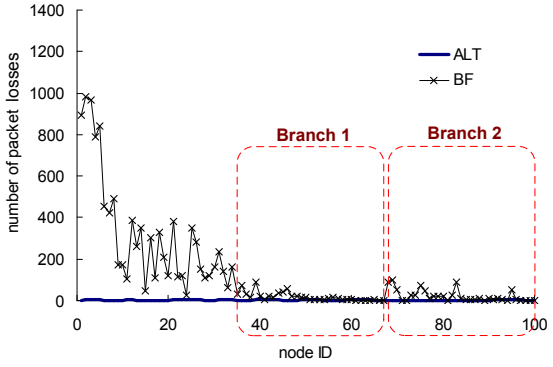


(b) the unbalanced topology

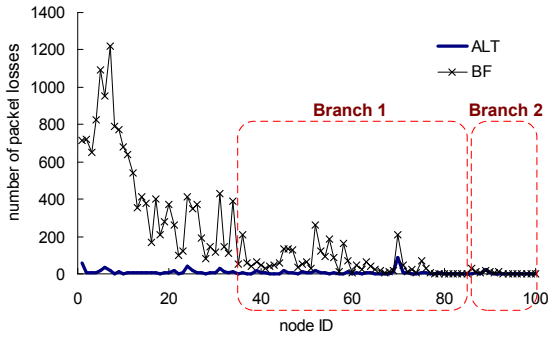


(c) the cross topology

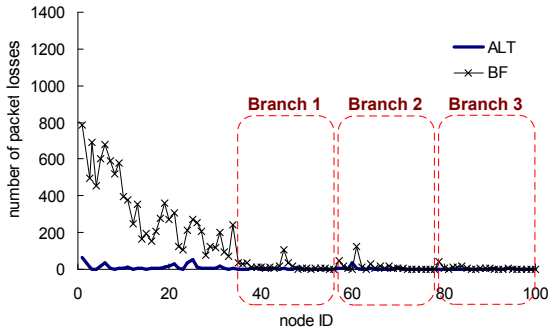
Fig. 11. Comparison on the numbers of packet retransmissions under ALT and BF.



(a) the balanced topology



(b) the unbalanced topology



(c) the cross topology

Fig. 12. Comparison on the numbers of packet losses under ALT and BF.

number of retransmissions is not only reduced but also not affected by the topology. In Fig. 11, the numbers of retransmissions incurred by ALT in all three topologies exhibit very similar saw-tooth curves with valleys occurring at the lock gate nodes due to the fact that lock gates collect the packets within the corresponding clusters and reduce the number of transmissions.

We now compare the number of packet losses at each sensor node in the three network topologies. As shown in Fig. 12, using BF, sensor nodes closer to the sink node incur a higher number of packet losses because these sensor nodes have to relay more sensor readings from upstream nodes. In particular, the unbalanced topology suffers more packet losses than the other two topologies due to the existence of the longest branch and hence the highest contention. The branch node with ID 34 incurs more packet losses than its upstream neighboring nodes due to higher contention. In contrast, using ALT, the numbers of packet losses are quiet small in all three topologies, which indicates that lock gates significantly reduce the number of transmissions and mitigate the contention.

5.4. Average Packet Latency

In the same experiments described in Section 5.3, we also measure the average latency per successfully delivered packet from each sensor node to the sink node, as shown in Fig. 13. Using BF, sensor nodes simply relay the sensor readings toward the sink node, and the average latency is directly influenced by how far a sensor node is located away from the sink node. As we can observe, nodes in the unbalanced topology suffer from higher latency due to the longest branch. Using ALT, since the lock gates have to keep the received sensor readings locally while waiting to receive enough number of sensor readings to fill up a packet of maximum payload size, the packet latency is higher than using BF. According to the analysis in Section 4.5, we calculate the difference between the average packet latency using ALT and that using BF. Since the cluster size in these experiments is about 10, we have $K = 10$. In addition, the payload size of a sensor reading is 3 bytes and each sensor node generates one sensor reading every 5 seconds, so we have $L = 3$ and $\lambda_j = 0.6$ for each sensor node s_j, $j = 1..K$. By Eq. (4), we can derive that

$$T_i = \frac{L_{\max}}{\delta \times \sum_{\alpha=1}^{K} \lambda_\alpha} = \frac{117}{1 \times 10 \times 0.6} = 19.5.$$

Using Eq. (5), for each $j = 1..K$, we can obtain that

$$n_j = \left\lceil \frac{\lambda_j}{\sum_{\alpha=1}^{K} \lambda_\alpha} \times \frac{L_{\max}}{L \times \delta} \right\rceil = \left\lceil \frac{0.6}{10 \times 0.6} \times \frac{117}{3 \times 1} \right\rceil = 4.$$

The value of τ can be measured by the average packet latency using BF. Thus, from Fig. 13, we can obtain that $\tau \approx 0.29$. Therefore, according to Eq. (3), we calculate the expected extra packet latency using ALT by

$$T_i - \frac{L}{2K} \sum_{\alpha=1}^{K} \frac{n_\alpha + 1}{\lambda_\alpha} - \frac{K-1}{2}\tau$$
$$= 19.5 - \frac{3}{2 \times 10} \times (10 \times \frac{4+1}{0.6}) - (\frac{10-1}{2} \times 0.29)$$
$$= 5.695.$$

Table IV gives the average packet latency of each sensor node using ALT and BF (which are derived from Fig. 13). It can be observed the difference between the average packet latency using ALT and that using BF is 5.17, which is close to the analyzed result (*i.e.*, 5.695). The slight inaccuracy is due to the fact that not all clusters have 10 sensor nodes in our experiments. The above observation validates the soundness of our analysis in Section 4.5.

Table IV. Average packet latency of each sensor node using ALT and BF (unit: seconds).

scheme	balanced	unbalanced	cross	average
ALT	5.58	5.26	5.54	5.46
BF	0.29	0.29	0.28	0.29
difference	5.29	4.97	5.26	5.17

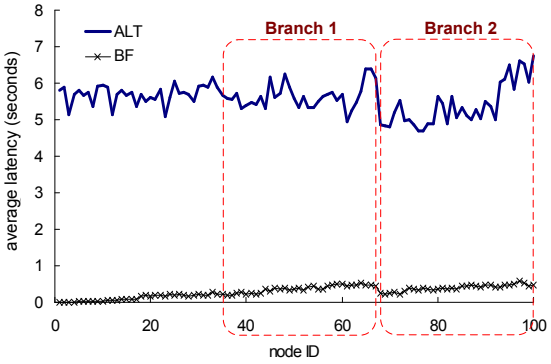
From Fig. 13, it can be observed the average packet latency using ALT exhibits saw-tooth curves in all three topologies due to the fact that every lock gate transmits a collected packet after T_i seconds as calculated by Eq. (1) while sensor readings arrive at the corresponding lock gates at different moments in time. Although BF incurs low latency, BF suffers from large numbers of retransmissions and packet losses. It is clear to conclude that ALT balances between the latency and the congestion in long-thin WSNs.

6. Conclusions

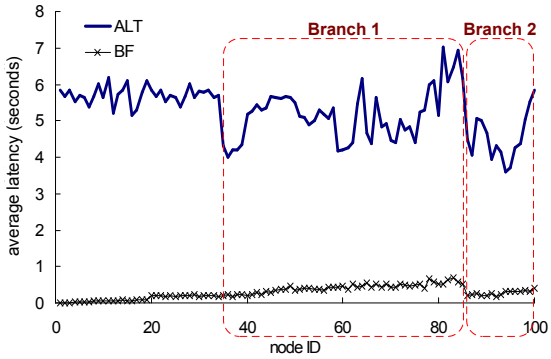
Many realistic WSN applications dictate the deployment of long-thin topology which demands new data collection schemes. This paper describes the ALT lock gate designation scheme which (1) designates multiple lock gates within a long-thin WSN to regulate data collection and (2) adapts the designation of lock gates dynamically in response to changing sensor reading generation rates of sensor nodes. ALT balances between the responsiveness and the congestion of data collection, and mitigates the funneling effect by regulating (collected) data that could be transmitted downstream and spatially separating areas where packets are transmitted. Using the Jennic JN5139 wireless micro-controllers, we evaluate the performance of ALT via experiments with prototyped long-thin WSNs. Experimental results demonstrate the merits of ALT and reveal the impact of lock gate designation on MAC-layer behaviors and latency.

References

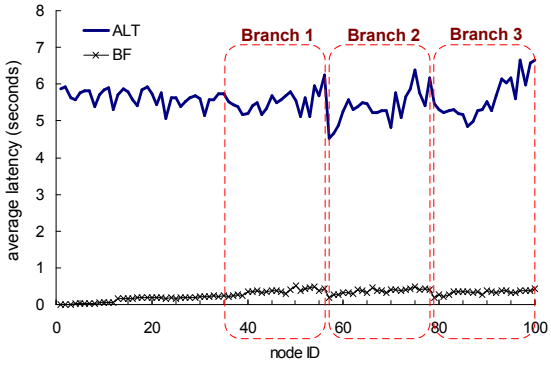
1. R. Szewczyk, E. Osterweil, J. Polastre, M. Hamilton, A. Mainwaring, and D. Estrin, "Habitat monitoring with sensor networks," *Communications of the ACM*, vol. 47, no. 6, pp. 34–40, June 2004.
2. S. Helal, W. Mann, H. El-Zabadani, J. King, Y. Kaddoura, and E. Jansen, "The gator tech smart house: a programmable pervasive space," *IEEE Computer*, vol. 38, no. 3, pp. 50–60, March 2005.



(a) the balanced topology



(b) the unbalanced topology



(c) the cross topology

Fig. 13. Comparison on the average packet latency under ALT and BF.

3. Y. C. Tseng, Y. C. Wang, K. Y. Cheng, and Y. Y. Hsieh, "iMouse: an integrated mobile surveillance and wireless sensor system," *IEEE Computer*, vol. 40, no. 6, pp. 60–66, June 2007.
4. M. S. Pan, H. W. Fang, Y. C. Liu, and Y. C. Tseng, "Address assignment and routing schemes for ZigBee-based long-thin wireless sensor networks," in *IEEE Vehicular Technology Conference*, May 2008, pp. 173–177.
5. G. S. Ahn, S. G. Hong, E. Miluzzo, A. T. Campbell, and F. Cuomo, "Funneling-MAC: a localized, sink-oriented MAC for boosting fidelity in sensor networks," in *ACM International Conference on Embedded Networked Sensor Systems*, October 2006, pp. 293–306.
6. I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–114, August 2002.
7. Jennic wireless microcontrollers. [Online]. Available: http://www.jennic.com/
8. D. De, "A distributed algorithm for localization error detection-correction, use in in-network faulty reading detection: applicability in long-thin wireless sensor networks," in *IEEE Wireless Communications and Networking Conference*, April 2009.
9. C. Hua and T. S. P. Yum, "Optimal routing and data aggregation for maximizing lifetime of wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 16, no. 4, pp. 892–903, August 2008.
10. K. Kalpakis, K. Dasgupta, and P. Namjoshi, "Efficient algorithms for maximum lifetime data gathering and aggregation in wireless sensor networks," *Computer Networks*, vol. 42, no. 6, pp. 697–716, August 2003.
11. B. Krishnamachari, D. Estrin, and S. Wicker, "The impact of data aggregation in wireless sensor networks," in *IEEE International Conference on Distributed Computing Systems Workshops*, July 2002, pp. 575–578.
12. C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, "Directed diffusion for wireless sensor networking," *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, pp. 2–16, February 2003.
13. S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TAG: a tiny aggregation service for ad-hoc sensor networks," *ACM SIGOPS Operating Systems Review*, vol. 36, pp. 131–146, December 2002.
14. A. F. Harris III, R. Kravets, and I. Gupta, "Building trees based on aggregation efficiency in sensor networks," *ACM Ad Hoc Networks*, vol. 5, no. 8, pp. 1317–1328, November 2007.
15. P. von Rickenbach and R. Wattenhofer, "Gathering correlated data in sensor networks," in *ACM Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, October 2004, pp. 60–66.
16. S. Pattem, B. Krishnamachari, and R. Govindan, "The impact of spatial correlation on routing with compression in wireless sensor networks," *ACM Transactions on Sensor Networks*, vol. 4, no. 4, pp. 1–33, August 2008.
17. O. Younis, M. Krunz, and S. Ramasubramanian, "Node clustering in wireless sensor networks: recent developments and deployment challenges," *IEEE Network*, vol. 20, no. 3, pp. 20–25, May/June 2006.
18. S. Basagni, "Distributed clustering for ad hoc networks," in *IEEE International Symposium on Parallel Architectures, Algorithms and Networks*, June 1999, pp. 310–315.
19. F. Kuhn, T. Moscibroda, and R. Wattenhofer, "Initializing newly deployed ad hoc and sensor networks," in *ACM International Conference on Mobile Computing and Networking*, September 2004, pp. 260–274.
20. W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Transactions on Wireless Communications*, vol. 1, no. 4, pp. 660–670, October 2002.
21. O. Younis and S. Fahmy, "HEED: a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks," *IEEE Transactions on Mobile Computing*, vol. 3, no. 4, pp. 366–379, October–December 2004.
22. Y. Zhu, R. Vedantham, S. J. Park, and R. Sivakumar, "A scalable correlation aware aggregation strategy for wireless sensor networks," in *IEEE International Conference on Wireless Internet*, July 2005, pp. 122–129.
23. S. Lindsey, C. Raghavendra, and K. M. Sivalingam, "Data gathering algorithms in sensor networks using energy metrics," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 9, pp. 924–935, September 2002.

24. K. Du, J. Wu, and D. Zhou, "Chain-based protocols for data broadcasting and gathering in sensor networks," *Proc. IEEE International Symposium on Parallel and Distributed Processing*, April 2003.
25. Y. P. Chen, A. L. Liestman, and J. Liu, "A hierarchical energy-efficient framework for data aggregation in wireless sensor networks," *IEEE Transactions on Vehicular Technology*, vol. 55, no. 3, pp. 789–796, May 2006.
26. D. Ganesan, D. Estrin, and J. Heidemann, "Dimensions: why do we need a new data handling architecture for sensor networks?" *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 1, pp. 143–148, January 2003.
27. D. Ganesan, B. Greenstein, D. Estrin, J. Heidemann, and R. Govindan, "Multiresolution storage and search in sensor networks," *ACM Transactions on Storage*, vol. 1, no. 3, pp. 277–315, August 2005.
28. Y. C. Wang, Y. Y. Hsieh, and Y. C. Tseng, "Multiresolution spatial and temporal coding in a wireless sensor network for long-term monitoring applications," *IEEE Transactions on Computers*, vol. 58, no. 6, pp. 827–838, June 2009.
29. K. W. Fan, S. Liu, and P. Sinha, "Structure-free data aggregation in sensor networks," *IEEE Transactions on Mobile Computing*, vol. 6, no. 8, pp. 929–942, August 2007.
30. B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, "Span: an energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks," *ACM Wireless Networks*, vol. 8, no. 5, pp. 481–494, September 2002.
31. M. Cardei and D. Z. Du, "Improving wireless sensor network lifetime through power aware organization," *ACM Wireless Networks*, vol. 11, no. 3, pp. 333–340, May 2005.
32. Y. Zou and K. Chakrabarty, "A distributed coverage- and connectivity-centric technique for selecting active nodes in wireless sensor networks," *IEEE Transactions on Computers*, vol. 54, no. 8, pp. 978–991, August 2005.
33. G. Xing, X. Wang, Y. Zhang, C. Lu, R. Pless, and C. Gill, "Integrated coverage and connectivity configuration for energy conservation in sensor networks," *ACM Transactions on Sensor Networks*, vol. 1, no. 1, pp. 36–72, August 2005.
34. Q. Zhao and M. Gurusamy, "Lifetime maximization for connected target coverage in wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. 16, no. 6, pp. 1378–1391, December 2008.
35. LAN/MAN Standards Committee of the IEEE Computer Society, "IEEE Std 802.15.4-2003, Wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (LR-WPANs)," *IEEE*, 2003.

附錄三

Dynamic Water Gate Assignment Scheme for Data Aggregation in Long-Thin Sensor Networks

Y.-C. Wang, C.-H. Chuang, Y.-C. Tseng, and C.-C. Shen

International Computer Symposium (ICS)

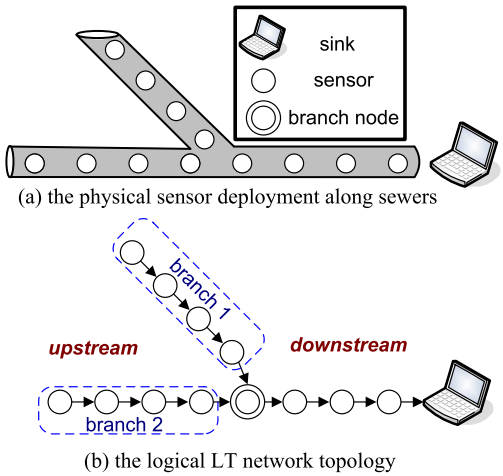
Dynamic Water Gate Assignment Scheme for Data Aggregation in Long-Thin Sensor Networks

You-Chiun Wang, Che-Hsi Chuang, Yu-Chee Tseng
Department of Computer Science
National Chiao-Tung University
Hsinchu 30010, Taiwan
E-mail: {wangyc, chuangch, yctseng}@cs.nctu.edu.tw

Chien-Chung Shen
Department of Computer and Information Sciences
University of Delaware
Newark, DE 19716, USA
E-mail: cshen@cis.udel.edu

***Abstract*—The *long-thin (LT)* network topology is promoted for many deployments of *wireless sensor networks (WSNs)* due to the application requirements. An LT topology consists of long branches of sensors, where each sensor has only one potential parent toward the sink. Data aggregation may help reduce excessive packet contention, but constraints imposed by the maximum payload size of each packet severely limit the amount of sensing data that can be aggregated along a long branch of sensors. Therefore, the paper suggests that multiple aggregation nodes, called *water gates*, should be designated along a branch to aggregate sensing data sent from their upstream sensors. Then, we develop a dynamic water gate assignment scheme that reduces the response time while avoids network congestion for data collection in LT WSNs, which can accommodate the time-varying sensing data generating rates of sensors. A testbed of 100 sensors is deployed and experimental results demonstrate the effectiveness of our scheme.**

***Keywords*-data aggregation; long-thin network; pervasive computing; wireless sensor network.**



(a) the physical sensor deployment along sewers

(b) the logical LT network topology

Figure 1: An example of the LT WSN deployed along sewers.

I. Introduction

Wireless sensor networks (WSNs) facilitate pervasive monitoring of the physical environments to enable many military and civil applications [1]–[4]. Sensors may be deployed to form networks with arbitrary topologies [5], [6] or regular topologies [7], [8]. In recent research, the *long-thin (LT)* network topology has been promoted for many WSN applications where the sensor deployment is subject to application requirements [9]. Practical examples include a CO_2 monitoring system inside tunnels, a surveillance system of moving cars along streets, and a monitoring system of water quality within sewers. Fig. 1(a) gives the physical sensor deployment along sewers, and Fig. 1(b) shows its logical LT network topology. Specifically, an LT topology consists of a bunch of long *branches* and each branch may have tens or even hundreds of sensors. For each sensor along a branch, there is only one potential parent node toward the sink. Branches are connected at *branch nodes*, which are denoted by double circles in Fig. 1(b).

By viewing an LT WSN with n sensors as a shortest-path tree rooted at the sink, let d_i denote the depth of sensor s_i from the sink, $i = 1..n$. Considering that each sensor transmits one packet to the sink without data aggregation, the WSN requires $\sum_{i=1}^{n} d_i$ transmissions to collect all sensing data. Fig. 1(b) gives an example, where the WSN requires 62 transmissions without data aggregation. Then, let us consider the following aggregation scheme: The leaf sensors begin transmitting their sensing data first. Each intermediate sensor waits to aggregate its own sensing data with the (aggregated) data sent from its child(ren), and then forwards the aggregated packet to the sink. Assume that successively aggregated sensing data can be loaded into one large packet, this scheme requires only n transmissions. Fig. 1(b) shows an example, where the WSN requires 12 transmissions using this aggregation scheme. Nevertheless, constrained by the maximum payload size $L_{\max}$ of each packet and the compression ratio δ ($0 < \delta \leq 1$), the above scheme is not feasible for LT WSNs because each sensor can only aggregate up to $\lfloor \frac{L_{\max}}{\delta} \rfloor$ bytes of sensing data, while the total size of sensing data generated by the sensors along a long branch always exceed this bound.

To comply with the maximum payload size, this paper suggests that along a branch, multiple aggregation nodes (called *water gates*) should be designated, where each water

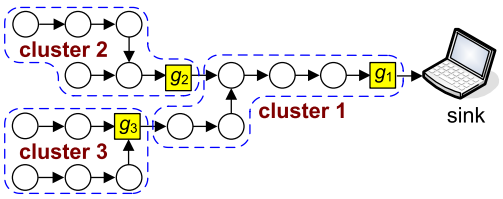


Figure 2: An LT WSN with three clusters.

gate aggregates its own sensing data with all of the sensing data from its upstream sensors (up to the immediate, upstream water gate(s) or the end of the branch) subject to the $\lfloor \frac{L_{\max}}{\delta} \rfloor$ bound. Fig. 2 shows an example. Considering that $L_{\max}$ is 3 bytes, δ is 0.5, and the size of sensing data is 1 byte, we designate one water gate every 6 sensors so that water gate g_1 aggregates its own sensing data with those from its 5 upstream nodes into one aggregated packet. Similarly, this can be done by water gates g_2 and g_3. Such an aggregated packet (of payload size $L_{\max}$) is said to be completely *filled up* with sensing data and is forwarded to the sink without being aggregated with any other sensing data along the branch. Besides, water gates help reduce contention among traffics by spatially separating areas where packets are transmitted. In Fig. 2, since water gates g_2 and g_3 hold their transmissions until enough sensing data from their upstream sensors have been collected, the sensors in cluster 1 can transmit their sensing data to g_1 with almost no interference from other sensors in clusters 2 and 3.

However, because sensors may generate sensing data at different *rates*, each water gate should wait for a different amount of time to completely fill up an aggregated packet of size $L_{\max}$. Let λ_j denote the sensing data generating rate (or *data rate* for short) of sensor s_j and $c(g_i)$ denote the *cluster* of nodes containing water gate g_i and its upstream sensors. Assuming that each λ_j is a constant and the WSN operates at a steady state, Eq. (1) indicates the amount of time t_i taken by water gate g_i to completely fill up an aggregated packet of size $L_{\max}$:

$$t_i = \frac{L_{\max}}{\delta \times \sum_{s_j \in c(g_i)} \lambda_j}. \quad (1)$$

When t_i is large, the sink will wait for a longer time to receive an aggregated packet from g_i, which increases the network response time. Large t_i implies either the size of $c(g_i)$ is small or the total data rate of the sensor nodes in $c(g_i)$ is low. On the contrary, small t_i implies either the size of $c(g_i)$ is large or the total data rate of the sensors in $c(g_i)$ is high. In this case, too many packet transmissions will congest the network.

Therefore, this paper proposes a *dynamic Water gate Assignment scheme for Data Aggregation (WADA)* to facilitate effective data aggregation in LT WSNs. Given time thresholds t_{lower} and t_{upper} specified by the WSN application to reduce the network response time while avoid the congestion caused by transmitted packets, WADA designates water gates in an LT WSN such that for each water gate g_i, we have $t_{lower} \leq t_i \leq t_{upper}$. Moreover, when data rate λ_i varies over time, WADA can adaptively adjust the locations of water gates to balance response time and congestion.

In the literature, the subject of data aggregation in WSNs has been extensively studied. However, most studies adopt either a tree-based or a cluster-based structure to aggregate data in WSNs, and none of them consider the LT topology. Tree-based aggregation schemes construct the shortest-path routing trees, and focus on how to choose a good routing metric to facilitate data aggregation. For example, the studies of [10], [11] propose data-centric schemes to select an appropriate path to reduce energy consumption. The study in [12] constructs an aggregation tree based on sensors' energy consumption. Each sensor predicts the energy consumption of its potential parents and selects the one that can remain the most energy as its parent. In contrast, there exists at most one route from a sensor to the sink in LT WSNs (*i.e.*, each sensor has at most one potential parent node toward the sink) so that existing solutions may not be directly applied.

In comparison, cluster-based aggregation schemes group sensors into clusters and conduct data aggregation within each cluster. For example, in LEACH [13] and PEGASIS [14], sensors relay the sensing data to their cluster heads, which are assumed to be able to directly communicate with the sink. However, this assumption is not valid in LT WSNs. HEED [15] groups sensors into clusters such that sensors within a cluster are single-hop away from the cluster head. However, in an LT WSN, sensors within a cluster are located along a long branch that are multi-hop away from their cluster head. SCT [16] proposes a ring-sector division clustering scheme, where sensors in the same section are grouped into one cluster. Explicitly, this scheme cannot be used in LT WSNs. To the best of our knowledge, the proposed WADA scheme is the first effort that addresses efficient data aggregation in LT WSNs.

We organize the rest of this paper as follows: Section II proposes our WADA scheme. Section III reports our prototyping experience and experimental results. Section IV concludes the paper.

II. The Proposed WADA Scheme

An LT WSN can be modeled as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ includes the sink and all sensors, and $\mathcal{E}$ includes all of the communication links. We represent the topology of $\mathcal{G}$ as a tree rooted at the sink. Each sensor s_j has a data rate of λ_j that may vary over time, $j = 1..n$. A sensor is called a *branch node* if it has more than one child in $\mathcal{G}$. Fig. 2 shows an example, where there are three clusters and sensors g_1, g_2, and g_3 are water gates.

Given an LT WSN, WADA arbitrarily groups sensors into non-overlapping clusters and assigns their water gates. Note that the sensor nearest to the sink is always assigned as

a water gate. On generating the sensing data, each regular sensor will send the data to its water gate. Each water gate g_i then collects the sensing data from the sensors within its cluster $c(g_i)$. After collecting sufficient sensing data that can be aggregated to fill up one packet of payload size $L_{\max}$, g_i sends the aggregated packet to the sink. To reduce the latency of waiting to aggregate sensing data to fill up one packet, g_i dynamically adjusts its cluster based on the duration t_i that it took to generate the last aggregated packet (by Eq. (1)). When $t_i < t_{lower}$, the total data rate within this cluster (*i.e.*, $\sum_{s_j \in c(g_i)} \lambda_j$) becomes too high, and the aggregated packets are sent to the sink more often. In this case, g_i *shrinks* its cluster by excluding some sensors to reduce the total data rate. On the other hand, when $t_i > t_{upper}$, the total data rate within this cluster becomes too low, and the monitoring quality degrades. Therefore, g_i *expands* its cluster by including more sensors to reduce the latency of generating aggregated packets. Note that within each cluster $c(g_i)$, the sensing data sent from each sensor $s_j \in c(g_i)$ may be relayed to g_i in a pipeline manner subject to the contention of wireless transmissions. Thus, right before g_i sends out each aggregated packet to the sink, the ratio of sensing data received from sensor s_j within this packet is approximately equal to $\frac{\lambda_j}{\sum_{s_k \in c(g_i)} \lambda_k}$. That is, WADA ensures that the amount of reported sensing data from each sensor can be fairly proportional to the data rate of that sensor.

We then define the terms used in the paper. A water gate g_k is called a *child water gate* of another water gate g_i if g_k is an immediate upstream water gate of g_i. In this case, g_i is the *parent water gate* of g_k. Fig. 2 gives an example, where g_2 is a child water gate of g_1 and g_1 is the parent water gate of g_2. A water gate may have several child water gates but has only one parent water gate. Besides, if a water gate has no child water gate, it is called a *leaf water gate*; otherwise, it is called an *intermediate water gate*.

From an initial water gate designation, each water gate executes WADA asynchronously but will coordinates with its parent and child water gates. Specifically, each water gate g_i measures its current t_i value, moves one of its child water gates downstream or upstream by one hop if needed, and calculates its new t_i value. This process is repeated until g_i calculates that $t_{lower} \leq t_i \leq t_{upper}$. In particular, for each intermediate water gate g_i, two cases are considered:

- **Case of $t_i < t_{lower}$:** Water gate g_i shrinks its cluster by querying each of its child water gates g_k for its t_k value. When g_k is also adjusting its own child water gates, it replies to g_i that itself is *busy*; otherwise, g_k replies to g_i for its t_k value. When g_i finds that all of its child water gates are busy, it waits for a Δ_t time and will try again. Otherwise, g_i sends a *pull* message to the non-busy child water gate g_k with the maximum t_k value and whose parent node, say, s_j is not a branch node. On receiving the pull message, g_k asks s_j to become a new water gate and clears itself as a water gate. Therefore, the old cluster $c(g_k)$ is replaced by a new cluster $c(s_j)$. Here, we use the term "move" to represent the above operation. However, when g_i cannot find such a child water gate (which means that the parent nodes of all its non-busy child water gates are branch nodes), the non-busy child water gate g_k with the maximum t_k value is asked to move one-hop downstream. These operations are repeated until g_i finds that $t_i \geq t_{lower}$. Fig. 3(a) shows an example, where g_1 wants to adjust its child water gates. Since g_2's parent node is a branch node, g_3 is asked to move downstream. However, if the parent nodes of all child water gates are branch nodes, as shown in Fig. 3(b), g_1 asks g_2 to move to branch node b (assuming $t_2 > t_3$).

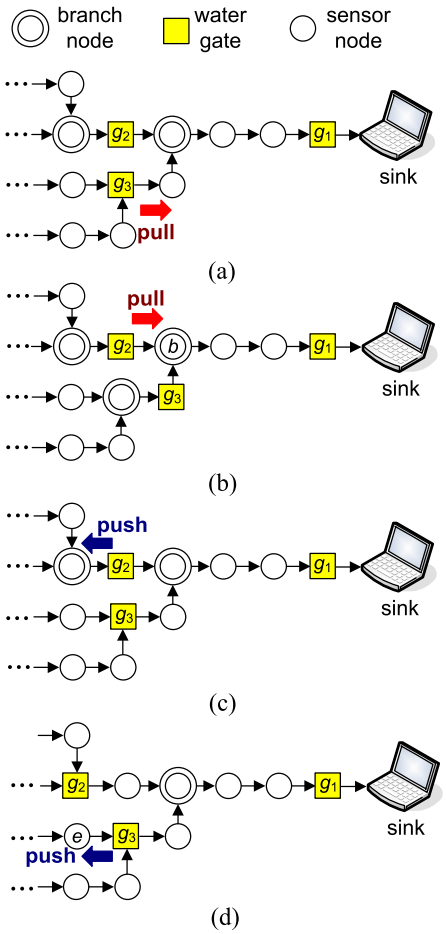


Figure 3: Four examples to show how to move child water gates: (a) g_1 pulls g_3 downstream due to $t_1 < t_{lower}$, (b) g_1 pulls g_2 to the branch node b due to $t_1 < t_{lower}$, (c) g_1 pushes g_2 upstream due to $t_1 > t_{upper}$, and (d) g_1 pushes g_3 to node s due to $t_1 > t_{upper}$.

To avoid excluding too many sensors when shrinking a cluster, which may make its t_i value increase drastically, WADA avoids moving a child water gate whose

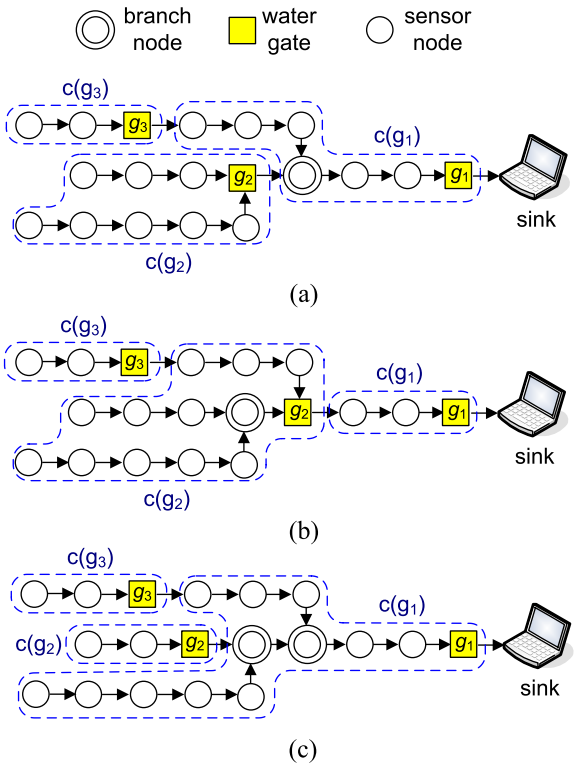


Figure 4: Bad examples of moving child water gates.

parent node is a branch node. Fig. 4(a) shows a bad example, where $t_2 > t_3$. If g_1 simply moves its child water gate with the maximum t_i value, g_2 will be asked to move downstream, as shown in Fig. 4(b). In this case, the size of cluster $c(g_1)$ drastically decreases from 7 to 3. Moreover, the size of cluster $c(g_2)$ drastically increases from 9 to 13. Instead, WADA moves g_3 one-hop downstream.

- **Case of $t_i > t_{upper}$:** Water gate g_i expands its cluster by querying each child water gate g_k for its t_k value. Then, g_i sends a *push* message to ask the non-busy child water gate g_k that is not a branch node and has the minimum t_k value to move one-hop upstream. When g_i cannot find such a child water gate (which means that all of its non-busy child water gates are branch nodes), the non-busy child water gate g_k with the minimum t_k value is asked to move upstream. These operations are repeated until g_i calculates that $t_i \leq t_{upper}$. Fig. 3(c) gives an example, where g_1 wants to adjust its child water gates. Since g_2 is not a branch node, it will be moved upstream. When all child water gates are branch nodes, as shown in Fig. 3(d), WADA asks g_3 to move to node e (assuming that $t_3 < t_2$).

To avoid including too many sensors when expanding a cluster, which may make its t_i value decrease drastically, WADA avoids moving a child water gate that is a branch node. Fig. 4(a) shows a bad example, where $t_2 < t_3$. If g_1 simply moves its child water gate with the minimum t_i value, g_2 will be asked to move upstream, as shown in Fig. 4(c). In this case, the size of cluster $c(g_1)$ drastically increases from 7 to 13. Moreover, the size of cluster $c(g_2)$ drastically decreases from 9 to 3. Instead, WADA moves g_3 one-hop upstream.

When moving a leaf water gate g_i, two cases should be considered:

- If g_i is asked to move downstream but finds that $t_i < t_{lower}$, it selects one leaf node, say, s_j in its cluster and designates s_j as a new water gate. Thus, the number of water gates will increase by one.
- If g_i is a leaf node but is still asked to move upstream, g_i clears itself as a water gate. Thus, the number of water gates will decrease by one.

To avoid oscillating water gates, each water gate g_i should maintain a list recording its past locations on $\mathcal{G}$. When g_i finds that it has moved between two adjacent *oscillating nodes* more than α times and its parent water gate still asks it to move to one of the oscillating nodes, it enters the *oscillating state*. Then, g_i notifies its parent water gate of stopping moving it in that direction. Water gate g_i will exit the oscillating state if its parent water gate asks it to move to one non-oscillating node or a predefined timer expires.

III. Experimental Results

We deploy 100 sensors to collect the environmental data, as shown in Fig. 5. Each sensor has a Jennic chip [17], which is a low power, wireless microcontroller supporting the ZigBee protocol [18]. Each sensor has a communication distance of 30 cm but two adjacent sensors are placed with a distance of 15 cm to ensure the network connectivity. We consider two LT topologies: x-topology and y-topology. As shown in Fig. 6, x-topology has four branches, where three branches have 22 nodes while one branch has 33 nodes; y-topology has three branches, where branch 1 has 15 node, branch 2 has 33 nodes, and branch 3 has 51 nodes. We compare WADA against the *Direct Relay (DR) scheme* and the *Static Water gate Assignment (SWA) scheme*. DR does not adopt any data aggregation and each sensor simply relays the sensing data from its upstream nodes to the sink. In SWA, each branch node is assigned as a water gate and we do not adjust the assignment of water gates during the experiments.

The packet size is 15 bytes, which contains a header of 12 bytes and a payload of 3 bytes. Each sensor reports its sensing data every $\Delta_s \in [7, 13]$ seconds and Δ_s will change every 30 seconds. We remove the headers of all received packets and concatenate their payloads into one single packet, so we have $\delta = 1$. In our experiments, we set $L_{\max}$ to 118 bytes, which is the maximum payload size defined in ZigBee. The experiment time is 10 minutes. In the experiments of executing WADA, the measurement

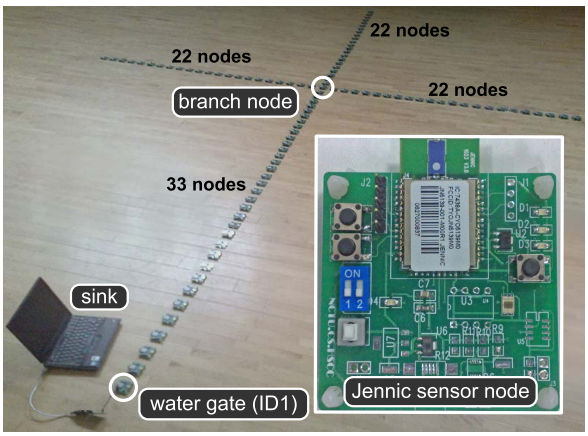
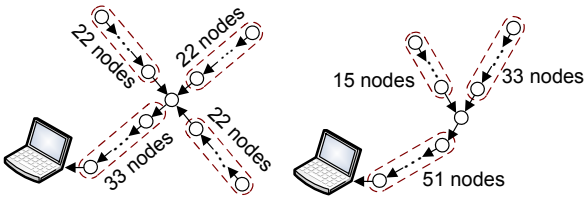


Figure 5: The snapshot of our 100-node LT WSN prototype.



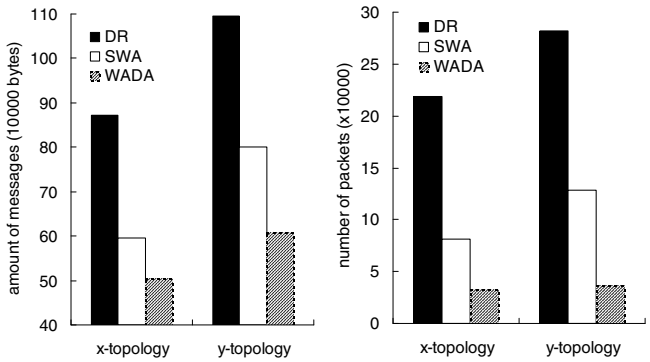
(a) x-topology (b) y-topology

Figure 6: Two LT network topologies used in our experiments.

of messages sent by sensors includes all of the control messages such as query, reply, push, and pull used to adjust the assignment of water gates. Besides, we set Δ_t = 2 seconds, t_{lower} = 24 seconds, and t_{upper} = 26 seconds.

Fig. 7(a) shows the total amount of messages sent by sensors under different network topologies. All schemes incur higher amount of messages under the y-topology because this topology has the longest branch (with 51 nodes). DR incurs the highest amount of messages because it does not adopt any data aggregation. By dynamically adjusting the locations of water gates based on the network condition, WADA incurs the smallest amount of messages compared with DR and SWA. Specifically, under x-topology, WADA reduces 42.3% and 15.7% of message transmissions compared with DR and SWA, respectively. Under y-topology, WADA reduces 44.7% and 24.3% of message transmissions compared with DR and SWA, respectively. The above results verify the effectiveness of WADA.

Fig. 7(b) shows the total number of packets sent by sensors under different network topologies. When sensors transmit more packets, the network could be seriously congested. DR makes the sensors transmit the most number of packets, because it does not aggregate any sensing data. WADA incurs the smallest number of packets among all schemes because it adaptively clusters sensors and aggregates their packets accordingly. Under x-topology, WADA reduces 84.9% and 59.5% of packets compared with DR



(a) amount of messages (b) number of packets

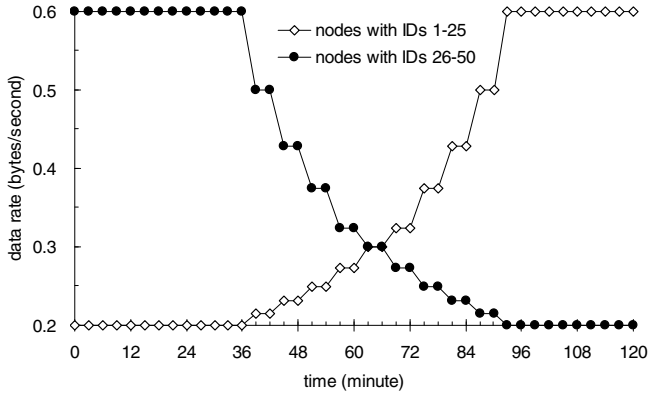
Figure 7: Comparison of data transmissions under x-topology and y-topology.

and SWA, respectively. Under y-topology, WADA reduces 87.3% and 72.2% of packets compared with DR and SWA, respectively. These results demonstrate that WADA significantly reduces the number of packets sent by sensors, which can greatly alleviate the network congestion.

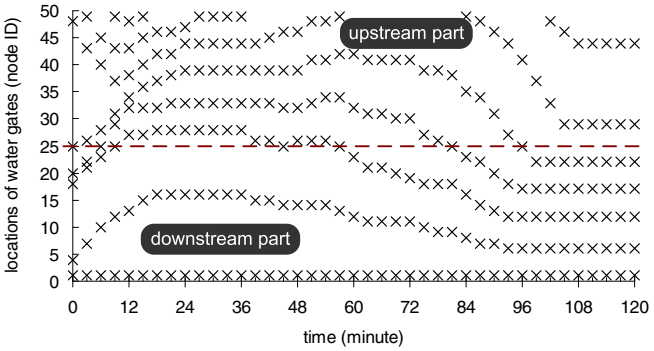
To demonstrate the adaptability of WADA to varying data rates, we deploy a sink (of ID 0) and a line of 50 sensor nodes (of IDs 1 to 50), where the node with ID 1 is the most downstream sensor. The experiment time is 120 minutes and we measure the number of water gates and their locations over time. The λ value of sensors with IDs 1 to 25 increases while that of sensors with IDs 26 to 50 decreases over time, as shown in Fig. 8(a). Here, we use the terms "downstream part" and "upstream part" to represent the sensors with IDs 1 to 25 and with IDs 26 to 50, respectively. Fig. 8(b) shows the locations of water gates over time. We observe that before the 60th minute, most water gates are located at the upstream part because sensors in the upstream part have a higher data rate. Therefore, WADA shrinks the sizes of clusters in the upstream part and thus assigns more water gates. After the two data rates cross around the 66th minute, the above behavior reverses. Most water gates move to the downstream part since sensors in the downstream part now have a higher data rate.

IV. Conclusions

A large number of WSN applications dictate the deployment of the LT topology that demands new data aggregation solutions. This paper proposes the WADA scheme that assigns multiple water gates to regulate data aggregation and adapts the designation of water gates dynamically in response to changing data rates of sensors. WADA not only reduces the network response time but also avoids potential network congestion for data collection in LT WSNs. By using the sensors nodes equipped with Jennic wireless microcontrollers, we measure the performance of WADA through several experiments of our prototyped LT WSNs. Experimental results verify that WADA significantly reduces



(a) data rates



(b) the locations of water gates

Figure 8: The locations of water gates (marked by 'x') under different data rates.

the amount of data transmissions of sensors and adaptively adjusts the assignment of water gates according to various network conditions.

ACKNOWLEDGEMENT

Y.-C. Tseng's research is co-sponsored by MoE ATU Plan, by NSC grants 97-3114-E-009-001, 97-2221-E-009-142-MY3, 98-2219-E-009-019, and 98-2219-E-009-005, by MOEA 98-EC-17-A-02-S2-0048, and 98-EC-17-A-19-S2-0052, by ITRI, Taiwan, by III, Taiwan, and by Intel.

REFERENCES

[1] S. Helal, W. Mann, H. El-Zabadani, J. King, Y. Kaddoura, and E. Jansen, "The gator tech smart house: a programmable pervasive space," *IEEE Computer*, vol. 38, no. 3, pp. 50–60, 2005.

[2] Y. C. Tseng, Y. C. Wang, K. Y. Cheng, and Y. Y. Hsieh, "iMouse: an integrated mobile surveillance and wireless sensor system," *IEEE Computer*, vol. 40, no. 6, pp. 60–66, 2007.

[3] L. W. Yeh, Y. C. Wang, and Y. C. Tseng, "iPower: an energy conservation system for intelligent buildings by wireless sensor networks," *International Journal of Sensor Networks*, vol. 5, no. 1, pp. 1–10, 2009.

[4] S. C. Hu, Y. C. Wang, C. Y. Huang, and Y. C. Tseng, "A vehicular wireless sensor network for co_2 monitoring," in *IEEE Conference on Sensors*, 2009, pp. 1498–1501.

[5] A. Cerpa and D. Estrin, "ASCENT: adaptive self-configuring sensor networks topologies," in *IEEE INFOCOM*, 2002, pp. 1278–1287.

[6] H. Zhang and J. C. Hou, "Maintaining sensing coverage and connectivity in large sensor networks," *International Journal of Wireless Ad Hoc and Sensor Networks*, vol. 1, no. 1–2, pp. 89–124, 2005.

[7] Y. C. Wang, C. C. Hu, and Y. C. Tseng, "Efficient placement and dispatch of sensors in a wireless sensor network," *IEEE Transactions on Mobile Computing*, vol. 7, no. 2, pp. 262–274, 2008.

[8] Y. C. Wang and Y. C. Tseng, "Distributed deployment schemes for mobile wireless sensor networks to ensure multilevel coverage," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 9, pp. 1280–1294, 2008.

[9] M. S. Pan, H. W. Fang, Y. C. Liu, and Y. C. Tseng, "Address assignment and routing schemes for ZigBee-based long-thin wireless sensor networks," in *IEEE Vehicular Technology Conference*, 2008, pp. 173–177.

[10] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: a scalable and robust communication paradigm for sensor networks," in *ACM International Conference on Mobile Computing and Networking*, 2000, pp. 56–67.

[11] B. Krishnamachari, D. Estrin, and S. Wicker, "The impact of data aggregation in wireless sensor networks," in *IEEE International Conference on Distributed Computing Systems Workshops*, 2002, pp. 575–578.

[12] A. F. Harris III, R. Kravets, and I. Gupta, "Building trees based on aggregation efficiency in sensor networks," *ACM Ad Hoc Networks*, vol. 5, no. 8, pp. 1317–1328, 2007.

[13] M. J. Handy, M. Haase, and D. Timmermann, "Low energy adaptive clustering hierarchy with deterministic cluster-head selection," in *IEEE International Workshop on Mobile and Wireless Communications Network*, 2002, pp. 368–372.

[14] S. Lindsey and C. S. Raghavendra, "PEGASIS: power-efficient gathering in sensor information systems," in *IEEE Aerospace Conference*, 2002, pp. 1125–1130.

[15] O. Younis and S. Fahmy, "Distributed clustering in ad-hoc sensor networks: a hybrid, energy-efficient approach," in *IEEE INFOCOM*, 2004, pp. 629–640.

[16] Y. Zhu, R. Vedantham, S. J. Park, and R. Sivakumar, "A scalable correlation aware aggregation strategy forwireless sensor networks," in *IEEE International Conference on Wireless Internet*, 2005, pp. 122–129.

[17] Jennic microcontrollers. [Online]. Available: http://www.jennic.com/

[18] LAN/MAN Standards Committee of the IEEE Computer Society, "IEEE Std 802.15.4-2003, Wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (LR-WPANs)," *IEEE*, 2003.

附錄四

Address Assignment and Routing Schemes for ZigBee-Based Long-Thin Wireless Sensor Networks

M.-S. Pan, H.-W. Fang, Y.-C. Liu, and Y.-C. Tseng

IEEE 67th Vehicular Technology Conference

Address Assignment and Routing Schemes for ZigBee-Based Long-Thin Wireless Sensor Networks

Meng-Shiuan Pan*[1], Hua-Wei Fang†‡[2], Yung-Chih Liu†[3], and Yu-Chee Tseng*[4]
*Department of Computer Science
National Chiao Tung University, Hsin-Chu, Taiwan
† Networks and Multimedia Institute
Institute for Information Industry, Taipei, Taiwan
‡ Department of Computer Science and Information Engineering
National Taiwan University, Taipei, Taiwan
Email:{[1]mspan, [4]yctseng}@cs.nctu.edu.tw, [2]d95030@csie.ntu.edu.tw, [3]ulysses@nmi.iii.org.tw

***Abstract*—Wireless sensor networks (WSNs)** **have been extensively researched recently. This paper makes two contributions to this field. First, we promote a new concept of *long-thin (LT) topology* for WSNs, where a network may have a number of *linear paths* of nodes as backbones connecting to each other. These backbones are to extend the network to the intended coverage areas. At the first glance, a LT WSN only seems to be a special case of numerous WSN topologies. However, we observe, from real deployment experiments, that such a topology is quite general in many applications and deployments. The second contribution is that we show that the *address assignment* and thus the *tree routing scheme* defined in the original ZigBee specification may work poorly, if not fail, in a LT topology. We thus propose simple, yet efficient, address assignment and routing schemes for a LT WSN. Simulation results and prototyping experiences are also reported.**

***Index Terms*—address assignment, long-thin network, routing protocol, wireless sensor network, ZigBee.**

I. Introduction

The rapid progress of wireless communication and embedded micro-sensing MEMS technologies has made wireless sensor networks (WSN) possible. A WSN usually needs to configure itself automatically and support ad hoc routing. A lot of research works have been dedicated to WSNs, including power management [15], routing and transportation [3], coverage issue [7], and localization [1]. In the application side, habitat monitoring is explored in [4], wildfire monitoring is addressed in [6], and navigation is studied in [14].

Recently, several WSN platforms have been developed, such as MICA [11] and Dust Network [5]. For interoperability among different systems, standards such as ZigBee [17] have been developed. In the ZigBee protocol stack, physical and MAC layer protocols are adopted from the IEEE 802.15.4 standard [8]. 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 device* by associating with the coordinator or a router. In ZigBee, a device is said to join a network successfully if it can obtain a 16-bit network address from the coordinator or a router. ZigBee specifies a distributed address assignment scheme, which allows a parent device to locally compute addresses for child devices. While the assignment scheme has low complexity, it also prohibits the network from scaling up and thus cannot be used in LT networks.

In this paper, we discuss the LT network, which is considered as a specific but common network topology in many surveillance applications, such as gas leakage detection of fuel pipes, carbon dioxide concentration monitoring in tunnels, stage measurements in sewers, street lights monitoring in highway systems, flood protection of rivers, and vibration detection of bridges. In such a network, nodes may form several long backbones and these backbones are to extend the network to the intended coverage areas. A backbone is a linear path which may contain tens or hundreds of ZigBee routers. So the network can be scaled up with limited hardware cost.

In this work, we propose address assignment and routing schemes for ZigBee-based LT WSNs. To assign addresses to nodes, we design rules to divide nodes into clusters. Each node belongs to one cluster and each cluster has a unique *cluster ID*. All nodes in a cluster will have the same cluster ID, but different *node IDs*. The structure of ZigBee network address is divided into two parts: one is *cluster ID* and the other is *node ID*. Following the same ZigBee design philosophy, the proposed scheme is simple and has low complexity. Existing works [2][12][13][16] have discussed address assignment for WSNs, but they are not designed for ZigBee or LT WSNs. To the best of our knowledge, this is the first work addressing this issue. Moreover, similar to the ZigBee tree routing protocol, the proposed routing protocol can also utilize nodes' network addresses to facilitate routing. In addition, routing can take advantage of shortcuts for better efficiency, so our scheme does not restrict nodes to relay packets only to their parent or child nodes as ZigBee does.

The rest of this paper is organized as follows. Preliminaries

are given in Section II. Section III presents our algorithms. Section IV presents some performance evaluations. Finally, Section V concludes this paper.

II. Preliminaries

A. ZigBee Address Assignment

In ZigBee, network addresses are assigned to devices by a distributed address assignment scheme. Before forming a network, the coordinator determines the maximum number of children of a router (Cm), the maximum number of child routers of a router (Rm), and the depth of the network (Lm). Note that a child of a router can be a router or an end device, so $Cm \geq Rm$. The coordinator and routers can each have at most Rm child routers and at least $Cm - Rm$ child end devices. Devices' addresses are assigned in a top-down manner. For the coordinator, the whole address space is logically partitioned into $Rm + 1$ blocks. The first Rm blocks are to be assigned to the coordinator's child routers and the last block is reserved for the coordinator's own child end devices. From Cm, Rm, and Lm, each router computes a parameter called $Cskip$ to derive the starting addresses of its children's address pools. The $Cskip$ for the coordinator or a router in depth d is defined as:

$$Cskip(d) = \begin{cases} 1 + Cm \times (Lm - d - 1) & \text{if } Rm = 1 \\ \frac{1 + Cm - Rm - CmRm^{Lm-d-1}}{1 - Rm} & \text{otherwise.} \end{cases} \quad (1)$$

The coordinator is said to be at depth $d = 0$, and d is increased by one after each level. Address assignment begins from the ZigBee coordinator by assigning address 0 to itself. If a parent node at depth d has an address A_{parent}, the n-th child router is assigned to address $A_{parent} + (n-1) \times Cskip(d) + 1$ and n-th child end device is assigned to address $A_{parent} + Rm \times Cskip(d) + n$. An example of the address assignment is shown in Fig. 1. The $Cskip$ of the coordinator is obtained from Eq. (1) by setting $d = 0$, $Cm = 5$, $Rm = 4$, and $Lm = 2$. Then the child routers of the coordinator will be assigned to addresses $0 + (1-1) \times 6 + 1 = 1$, $0 + (2-1) \times 6 + 1 = 7$, $0 + (3-1) \times 6 + 1 = 13$, and etc. The address of the only child end device of the coordinator is $0 + 4 \times 6 + 1 = 25$. Note that the length of a network address is 16 bits; thus, the maximum address capacity is $2^{16} = 65536$. Obviously, the above assignment is much suitable for regular networks, but not for LT WSNs. For example, when setting $Cm = Rm = 2$, the depth of the network can only be 15. Also, when there is a LT backbone, the address space is not well utilized.

B. ZigBee Tree Routing Protocol

In a ZigBee network, the coordinator and routers can directly transmit packets along the tree without using any route discovery. When a device receives a packet, it first checks if it is the destination or one of its child end devices is the destination. If so, this device will accept the packet or forward this packet to the designated child. Otherwise, it forwards the packet to its parent. Assume that the depth of this device is d and its address is A. This packet is for one of its descendants if the destination address A_{dest} satisfies $A < A_{dest} < A + Cskip(d-1)$, and this packet will be relayed to the child router with address

$$A_r = A + 1 + \left\lfloor \frac{A_{dest} - (A+1)}{Cskip(d)} \right\rfloor \times Cskip(d).$$

If the destination is not a descendant of this device, this packet will be forwarded to its parent. In the ZigBee tree routing, each node can only choose its parent or child as the next node. This strategy may cause long delay in LT networks.

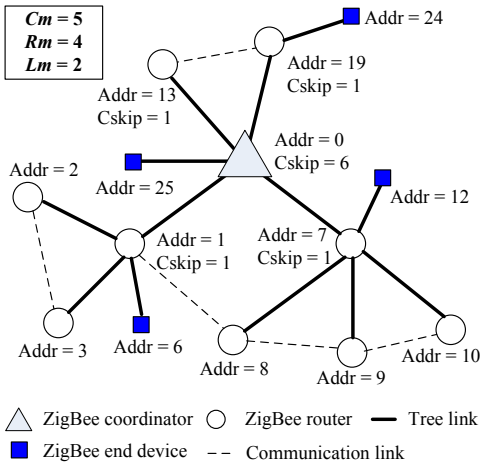


Fig. 1. A ZigBee address assignment example.

III. The Proposed Schemes

Assuming that all nodes are router-capable devices, we show how to form a LT WSN (as in Fig. 2(a)). Nodes are divided into multiple *clusters*, each as a line segment. For each cluster, we define two special nodes, named *cluster head* and *bridge*. The cluster head (resp., the bridge) is the node that has the smallest (resp., largest) hop count to the coordinator. As a special case, the coordinator, is also considered as a cluster head. The other nodes are *network nodes* (refer to Fig. 2(b)). A cluster C is a *child* cluster of a cluster C' if the cluster head of C is connected to the bridge of C'. Reversely, C' is C's *parent* cluster. Note that a cluster must have a linear path as its subgraph. But it may have other extra links beside the linear path. For example, in Fig. 2(b), there are two extra radio links (A, $A2$) and ($A1$, $A3$) in A's cluster. To be compliant with ZigBee, we divide the ZigBee 16-bit network address into two parts, an m-bit *cluster ID* and a $(16-m)$-bit *node ID*. The network address of a node v is thus expressed as (C_v, N_v), where C_v and N_v are v's cluster ID and node ID, respectively.

A. Network Planning

Before deploying a network, the network manager should carefully plan the placement of cluster head, bridge, and network nodes. There are some basic principles:

1) The network contains a number of linear paths.
2) For each cluster, the first and the last nodes are pre-assigned (manually) as cluster head and bridge, respectively.

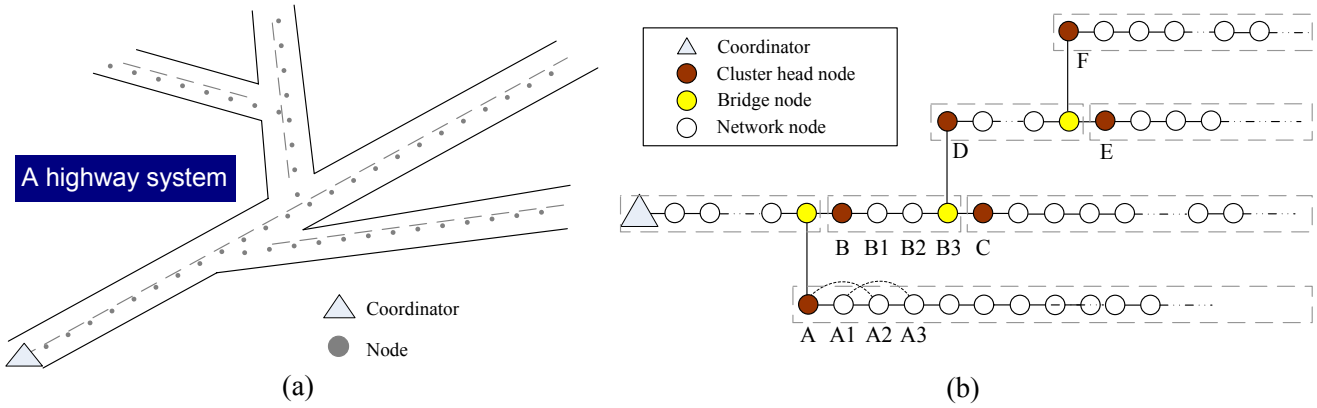


Fig. 2. (a) A LT WSN. (b) Role assignment.

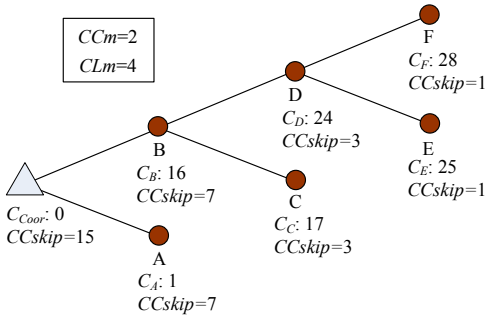


Fig. 3. The logical network of Fig. 2(b).

3) A cluster head that is not the coordinator should have a link to the bridge of its parent cluster.
4) Conversely, the bridge of a cluster which has child clusters should have a link to the cluster head of each child cluster.
5) A cluster does not cross other clusters and does not have links with other clusters except those locations nearby the cluster head and bridge areas.

After planing the placements, the network manager can construct a *logical network* G_L, in which each cluster is converted into a single node and the parent-child relationships of clusters are converted into edges. For example, Fig. 3 is the logical network of Fig. 2(b). Then the network manager can determine two parameters: the maximum number of children CCm of a node in G_L and the depth CLm of G_L.

B. Address Assignment

Network addresses are assigned in two stages. With the above planning, the network manager first mutually assigns a cluster ID to each node. Then node IDs are assigned in a distributed manner after network deployment.

The network manager assigns cluster IDs as follows. If $CCm = 1$, there is no branch in the network and the cluster ID assignment is a trivial case. If $CCm \geq 2$, the cluster IDs are assigned following the style of ZigBee in a recursive manner. The nodes in the coordinator's cluster have a cluster ID of 0. For each node at depth d in G_L, if its cluster ID is C, then its i-th child cluster is assigned a cluster ID of $C + (i-1) \times CCskip(d) + 1$, where

$$CCskip(d) = \frac{1 - CCm^{CLm-d}}{1 - CCm}. \quad (2)$$

Fig. 3 shows an example of cluster ID assignment of Fig. 2(b). Note that, in our scheme, for a cluster head in G_L with depth d, we also set each node in this cluster has a *logical depth* d.

After cluster ID assignment and node deployment, each node periodically broadcasts HELLO packets including its IEEE 64-bit MAC address, 16-bit network address (with its cluster ID but node ID is initiated to *NULL*), and role. Each node maintains a neighbor table to record its neighbors' information. Then the node ID assignment is started by the coordinator broadcasting its beacon with its node ID setting to 0. When a node without a node ID receives a beacon, it will send an *Association_Request* to the beacon sender. If there are multiple beacons, the node with the strongest signal strength will be selected. When the beacon sender, say, v with a network address (C_v, N_v), receives the association request(s), it will do the following steps.

1) If v is *not* a bridge node, it sets a parameter $N = N_v + 1$. Then it sorts request senders according to the received signal strength of their request packets in an descending order into a list L. Then v sequentially examines each node $u \in L$ by the following rules:
   - If $C_u \neq C_v$, v skips u and continues to examine the next node in L.
   - Otherwise, v assigns $N_u = N$ and increments N by 1. Then v replies an *Association_Response* to u with its address. If L is not empty, v loops back and continues to examine the next node in L.

   After finishing the above iteration, v further selects a node u, from the accepted ones, using the following rules: i) If there is a bridge node in the accepted ones, v selects the bridge node. ii) Otherwise, v selects the last node in L. Then v delegates u as the next beacon sender by sending a command *next_beacon_sender*(u) to u.

TABLE I
PART OF THE RESULTING NETWORK ADDRESSES IN FIG. 2.

Node	Network address		Node	Network address	
	Cluster ID	Node ID		Cluster ID	Node ID
A	00001	00000000000	B	10000	00000000000
A1	00001	00000000001	B1	10000	00000000001
A2	00001	00000000010	B2	10000	00000000010
A3	00001	00000000011	B3	10000	00000000011

2) If v is a bridge node, it only accepts the requests from cluster heads of its child cluster. When deciding to accept a node u, v replies an *Association_Response* to u with $N_u = 0$ and a *next_beacon_sender*(u).

For each node u that receives a *next_beacon_sender*(u) in the above steps, it will use the MLME-START primitive defined in IEEE 802.15.4 to start its beacons. Then the same procedure is repeated. Note that we allow a beacon sender to accept multiple children so as to reduce the communication cost of address assignment. Table I shows parts of the address assignment results in Fig. 2(b).

C. Routing Rules

Routing in our LT WSN can be purely based on the above address assignment results. Through HELLO packets, a node can collect its neighbors' network addresses. Suppose that a node v at logical depth d receives a packet with a destination address (C_{dest}, N_{dest}). If v is the destination, it simply accepts this packet. Otherwise, v performs the following procedures.

1) If the destination is a neighbor of v, v sends this packet to the destination directly.
2) If $C_{dest} = C_v$, the destination is within the same cluster. Node v can find an ancestor or a descendant in its neighbor table, say, u such that $C_u = C_{dest}$ and the value of $|N_u - N_{dest}|$ is minimized, and forward this packet to u.
3) If C_{dest} is a descendant cluster of C_v, i.e., $C_v < C_{dest} \leq C_v + (CCm - 1) \times CCskip(d) + 1$, then v checks if it has a neighbor u which satisfies $C_u \leq C_{dest} \leq C_u + (CCm - 1) \times CCskip(d + 1) + 1$. If such a u exists, then v forwards the packet to u. In case that there are multiple candidates, the one with the smallest $|N_u - N_{dest}|$ is selected. Otherwise, v finds a neighbor u which is located in the same cluster and has the maximum N_u and forwards the packet to u.
4) For all other cases, C_{dest} must be an ancestor cluster of C_v or not within the same logical subtree. Then v checks if it has a neighbor u which satisfies $C_u < C_v \leq C_u + (CCm - 1) \times CCskip(d - 1) + 1$. If such a u exists, v forwards the packet to u. Note that the above condition confines that C_u is the parent cluster of C_v. Otherwise, v finds a neighbor u which is located in the same cluster and has the minimum N_u and forwards the packet to u.

Note that the above design tries to make a balance between efficiency and simplicity. It basically follows the ZigBee tree-like routing. However, making shortcut along the linear paths of the LT WSN is possible due to the existence of neighbor tables and our design of hierarchical network addresses. Therefore, unlike the original ZigBee tree routing, nodes are not restricted to relay packets only to their parents or children.

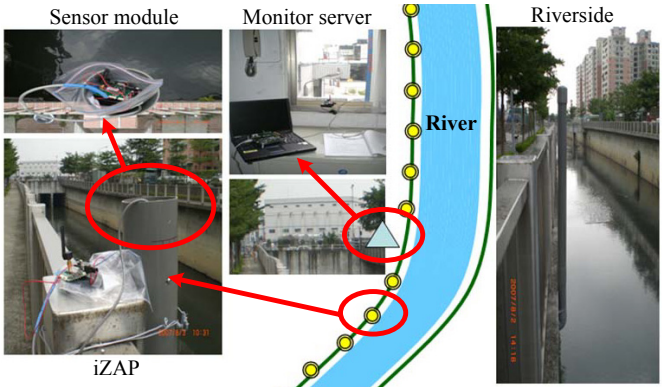


Fig. 4. The field deployment of a LT network.

IV. PERFORMANCE EVALUATIONS

A. Prototyping Experience

We implement a ZigBee-based wireless sensor node, named III ZigBee Advanced Platform (iZAP). The RF module is the Jennic JN5121 with chip antenna [10]. iZAP provides extra I/O pins, which can connect to sensor modules. In the iZAP, we also implement an external watchdog timer (WDT), which can be used to reset the device when an abnormal event occurs. There are three LEDs and an RS232 connector on the device. More details of iZAP can be found in [9].

A LT WSN is deployed to monitor the water level of a river in Taipei County, Taiwan as shown in Fig. 4. There are 41 nodes in the network. The distance between two nodes is 100 meters and the network depth is 20. There are three clusters in the network and the cluster of the coordinator has only one member (the coordinator itself). Nodes report the sensed readings to the coordinator every minute. The coordinator passes the received sensory readings to a monitor server (as shown in Fig. 5), which can show the statuses of the environment and nodes. The system manager can also use the monitor server to issue commands to the network nodes.

We record the average report latencies of nodes every one hour for two days. Nodes use either the proposed routing protocol or the ZigBee tree routing protocol to report sensory data. When using the ZigBee protocol, we mutually assign addresses to nodes and restrict that each node can only report data to its parent node. Fig. 6 shows the experimental results. We can observe that, in average, our protocol can perform better than ZigBee. This result also indicates that, when using our protocol, the stability of the network can be better.

B. Simulation Results

We simulate the proposed routing protocol in a large scale LT network, whose layout is the same as the one in Fig. 2(b). The distance between two adjacent nodes is 20 m. The simulation program randomly generates source-destination pairs and

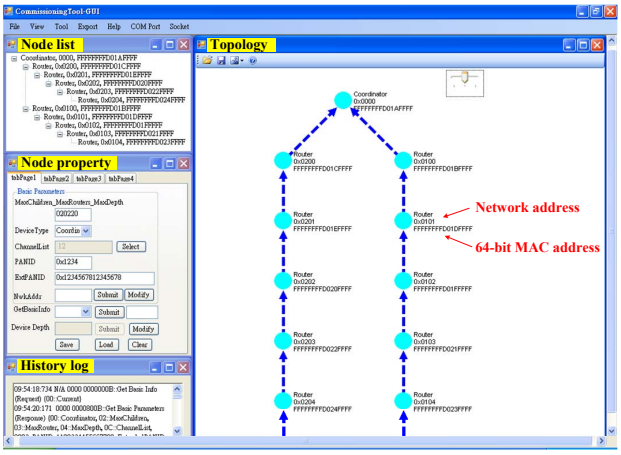


Fig. 5. The graphical user interface of the monitor server.

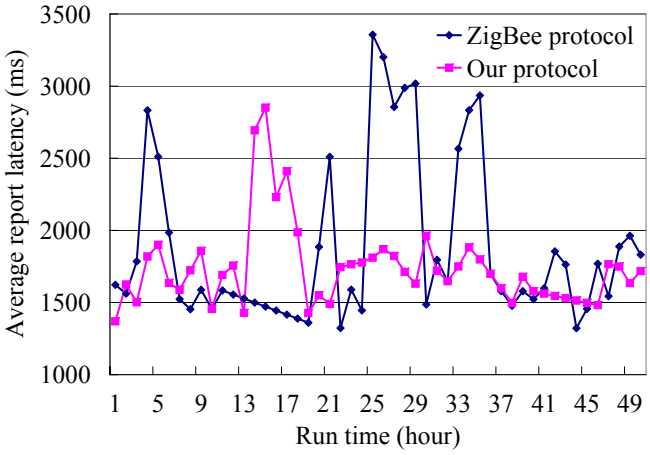


Fig. 6. Comparison on the measured report latency.

records the averaged hop counts from sources to destinations. Fig. 7 shows the simulation results. We can observe that, when transmission ranges become large, the average number of hop count decreases. This result indicates that shortcuts can effectively reduce the number of transmissions.

V. Conclusions

We have proposed address assignment and routing schemes for ZigBee-based LT WSNs. The proposed address assignment scheme divides nodes into several clusters and then assigns each node a cluster ID and a node ID as its network address. The routing protocol uses addresses of nodes to find routing paths and allows nodes to utilize shortcuts. We verify both our schemes by real implementation and simulations. In the future, we plan to discuss address assignment and routing schemes for other specific but common topologies such as hypercube networks.

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.

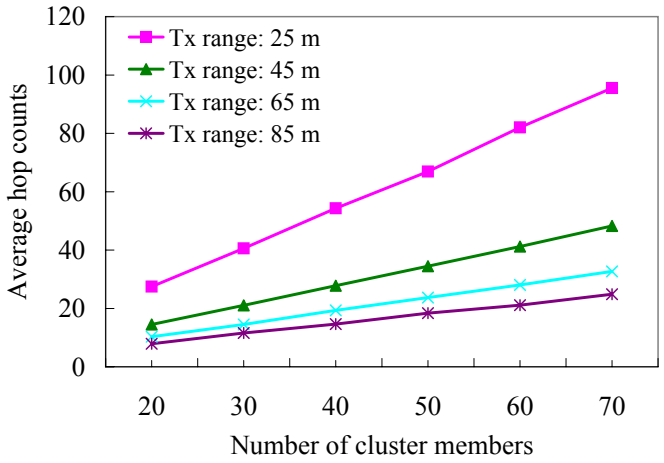


Fig. 7. Simulation on averaged number of hop counts when transmission ranges of nodes and the number of cluster members are varied.

References

[1] A. A. Ahmed, H. Shi, and Y. Shang. SHARP: A new approach to relative localization in wireless sensor networks. In *Proc. of Int'l Conference on Distributed Computing Systems Workshops (ICDCSW)*, 2005.

[2] M. Ali and Z. A. Uzmi. An energy-efficient node address naming scheme for wireless sensor networks. In *Proc. of IEEE Int'l Networking and Communications Conference (INCC)*, 2004.

[3] D. Braginsky and D. Estrin. Rumor routing algorithm for sensor networks. In *Proc. of ACM Int'l Workshop on Wireless Sensor Networks and Applications (WSNA)*, 2002.

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

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

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

[7] C.-F. Huang, Y.-C. Tseng, and L.-C. Lo. The coverage problem in three-dimensional wireless sensor networks. In *Proc. of IEEE Global Telecommunications Conference (Globecom)*, 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] III ZigBee Advanced Platform (iZAP). http://zigbee.iii.org.tw/.

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

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

[12] E. Ould-Ahmed-Vall, D. M. Blough, B. S. Heck, and G. F. Riley. Distributed unique global ID assignment for sensor networks. In *Proc. of IEEE Mobile Adhoc and Sensor Systems Conference (MASS)*, 2005.

[13] C. Schurgers, G. Kulkarni, and M. B. Srivastava. Distributed on-demand address assignment in wireless sensor networks. *IEEE Trans. Parallel Distributed System*, 13(10):1056–1065, 2002.

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

[15] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient MAC protocol for wireless sensor networks. In *Proc. of IEEE INFOCOM*, 2002.

[16] H. Zhou, M. W. Mutka, and L. M. Ni. Reactive ID assignment for sensor networks. In *Proc. of IEEE Mobile Adhoc and Sensor Systems Conference (MASS)*, 2005.

[17] ZigBee-2006 specification, ZigBee document 064112, 2006.