無線行動感測器網路上分散式叢集首選舉演算法

李權修　　陳君福　　劉傳銘

台北科技大學 資訊工程系

{chl,chunfu,cmliu }@mcse.csie.ntut.edu.tw

摘要

近年來，許多關於無線感測器網路的研究陸續被提出，在這些研究當中，一個很重要的議題是在能源有相當限制的感測器上，如何有效率地利用有限能源去聚集資料然後傳送到使用者端。目前已經有很多資料收集的通訊協定被提出來，但是這些協定大部分適用在一個靜態的感測器網路上。然而，在許多的應用中，感測器節點是具備有移動的能力，因此本論文探討在感測器節點可移動的情況下，無線感測器網路如何有效率地收集資料。在這篇論文中，我們將針對無線行動感測器網路中叢集架構的資料收集協定提出兩個針對叢集首選舉的演算法，且將實作出上述演算法，並進一步評估它們的在能源消耗上相關的效能。

關鍵詞：行動感測器網路、叢集、資料收集協定、移動性

一、簡介

近年來，在微電子機械系統上的發展，已經有能力發展出一種低成本、低功率和多功能的感測器節點(sensor node)。這些感測器節點除了可以感測資料外，也都具備有簡易的計算能力和短距離的通訊能力，而且體積都很小。使用大量的感測器節點所組合而成的網路，被稱為無線感測器網路(wireless sensor networks)[1]。在無線感測器網路中，感測器節點是被放置在很接近感測區域的位置或是在感測區域中。通常在距離感測區域的遠處，會有一個基地台(base-station, BS)用來收集這些感測到的資料。圖 1 就是一個簡單的例子，說明感測器節點透過無線通道把資料傳回到基地台，再經由網際網路把資料傳回去給使用者。由於這些感測器都是被隨意放置在被感測區域的任一處，所以必須要有一套負責資料收集的演算法去維護感測器將資料傳送到基地台上。

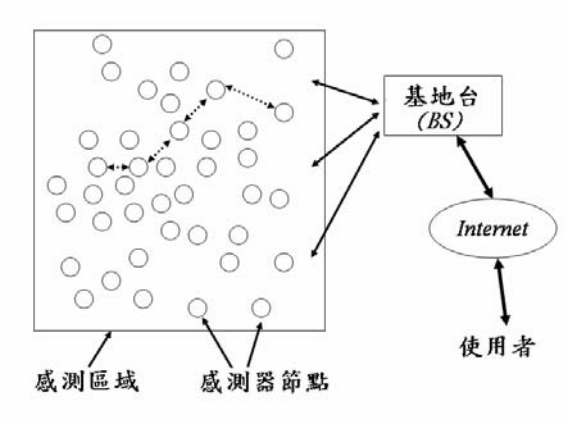

圖 1 無線感測器網路的架構

在無線感測器網路中，許多應用的目的都是要把感測的資料傳送到使用端。把感測器節點所感測的資料傳送到使用端的方式，被稱為資料收集協定。目前在無線感測器網路的資料收集協定上的問題，已經有許多相關的研究和文獻被提出[4][6][7][8][9][11][13][14]。在早期的研究中，都是著重在靜態(static)無線感測器網路上的應用，到了最近對於無線行動感測器網路(wireless mobile sensor networks)上的研究逐漸蓬勃發展起來。因為在許多的應用上，探討行動感測器網路是有其必要性和便利性。行動感測器網路中，感測器節點需要移動的因素可以分為主動和被動兩種模式。例如海洋溫度的監控，負責收集這塊區域的溫度感測器節點是散佈在海上，感測器節點會隨著海洋的潮流漂浮而移動，如此的移動方式則是被動的模式。在戰場上，感測器節點可能需要主動去追蹤敵軍的動態，收集敵軍的資料。如此一來則須具備主動式移動能力的

感測器節點[18]。而針對於上述資料收集的問題，若是使用傳統在靜態無線感測器網路上的資料收集協定，將無法有效地滿足上述所提到的需求。

在本論文中，我們將針對資料收集問題在無線行動感測器網路中提出相對的解決方案。我們將提出一個使用叢集架構(cluster-based)的資料收集協定，將網路上的所有感測器節點形成多個叢集，每個叢集選出一個叢集首 (cluster head)來管理叢集內感測器節點，收集叢集內感測到的資料。在這樣的架構上，我們將叢集的產生分割為兩個步驟：叢集首的選舉(cluster-head election)和叢集的組成(cluster formation)。我們將會提出兩個叢集首選舉的演算法，並且使用三個不同的移動模型(mobility models)去評估我們所提出的資料收集協定。

我們將在第二節敘述相關的背景知識。我們提出本論文所使用的系統架構在第三節。在第四節，我們提出兩個叢集首選舉演算法。我們實作出所提出的所有演算法，並且評估其效能在第五節。第六節將會對本篇論文作出結論。

二、背景知識

在無線感測器網路上，如何把感測的資料收集到使用端是一個很重要的問題。在許多的研究中，感測器節點使用多重跳躍(multi-hop)的通訊方式，以避免為了直接將資料傳送到基地台上所要消耗的大量能源。因為使用多重跳躍的方式，所以全部的資訊將快速累積到使用者端上。若是每一個感測器節點都把每一次接收到的資料，都未經過任何處理後就傳送出去，就會消耗太多的能源在通訊方面。所以每一個感測器節點都應該有能力可以融合或者結合所接收(感測)的資料，以減少傳送到基地台所要消耗的能源。

[10]提出在靜態感測器網路的實際應用上中，使用樹狀架構的演算法就可以完成資料收集的目的，而目前存在的資料收集協定包括有Flooding[13]，Gossiping[6]，Directed Diffusion[9]，SPIN[7]，LEACH[8]，PEGASIS[11]，HIT[4]和Data funneling[14]。雖然上述提出的這些資料收集協定都可用來收集網路中的資料，但這些資料收集協定當初都是考慮在靜態的無線感測器網路中運作，並無法完全的適用於行動感測器網路，在靜態感測器網路中，多數資料收集協定預期網路拓樸不會有大幅變動，每次組織好資料收集的路徑後，便可持續使用幾個回合以上，直到網路拓樸改變後，再重新組織新的資料收集的路徑，但由於行動感測器網路隨時都是處於在移動的狀態，所造成整個網路的拓樸快速變化將使得原先的路徑無法重複使用，導致這些協定會必須花費許多的資源重新組織出一個新的資料收集路徑，這樣的協定因此並不適用於行動感測器網路上。為了要便於維護在行動感測器網路中，快速變化的網路拓樸，所以我們考慮簡化樹狀架構(可由多層的叢集所組織)，改而使用叢集的架構為基礎去發展我們的演算法。根據我們的探討，我們發現其中LEACH是一個可以適用於移動環境的資料收集協定。主要原因是LEACH使用一個叢集為基礎的通訊協定，每一回合所組織的叢集只會適用於該回合的資料收集。

LEACH的資料收集路徑的產生，主是要使用叢集為基礎的架構。在組織叢集主要可以分成兩個步驟：(1)叢集首選舉，(2)叢集的形成。由[8]中，我們可以知道LEACH是使用方程式 1 來決定感測器節點是否要成為叢集首。方程式 1 顯示出在LEACH協定所使用的門檻值。

$$T(n)=\begin{cases}\dfrac{P}{1-P[r \bmod (1/P)]} & if\ n\in G\\ 0 & \text{otherwise}\end{cases} \quad 。 \quad (1)$$

P 表示該回合所需要叢集首的百分比(e.g., P=0.05)；r 代表現在是第幾個回合；G 表示尚未被選為叢集首的感測器節點；n 表示該感測器節點的ID。

舉例而言，考慮有 100 個感測器節點，我們預期每一回合要產生 5 個叢集首。假設在第一回合，只有產生 3 個叢集首，然後第二回合的叢集首將由剩下 97 個感測器節點去產生。套用方程式 1，在第二回合將會產生多於預期 5 個以上的叢集首。除此之外，我們可以利用方程式 1 去推導出在一個回合內，沒有任何叢集首產生的機率為：

$$P_f(r) = \left(1 - \frac{P}{1 - P\left(r \bmod \frac{1}{p}\right)}\right)^{|G|} \text{。} \quad (2)$$

根據我們觀察方程式 2，我們可以得到一個論點，使用如此的門檻值去產生叢集首，將會發生在某些回合中，一個叢集首都沒有的狀態；而且每一回合叢集首的產生是與之前回合所產生的叢集首有密切的關連性。在叢集的形成上，LEACH因為當初只有考慮在靜態的環境，所以產生出來的叢集在資料收集上，將會消耗更多的能源。因為在叢集形成和感測器節點傳送資料到叢集首的時候，感測器節點的位置是不相同的，所以在叢集組織的過程中，所加入的叢集，可能在資料傳送的時候，就不是距離最近的叢集首。所以在行動感測器網路上，若是沒有考慮到感測器節點移動的性質，將會消耗更多的能源。在第四節，我們將會提出兩個叢集首選舉演算法解決上述的叢集首選舉的問題，再使用[12]中所提出的機制CM來組織叢集，叢集首會廣播他們在這回合內的移動方式，各個非叢集首的感測器節點則再和自身的移動方式比較來推知哪個叢集首所組成的叢集最適合加入。

在這篇論文中，我們主要討論的是無線行動感測器網路上的資料收集問題。所以我們考慮感測器節點不同的移動方式對於我們演算法的影響，在無線隨意網路中，已經有許多移動模型被提出，包含隨意行走移動模型(random walk mobility model)[5]，隨意方向移動模型(random direction mobility model)[16]等。在[12]中，作者也提出一個簡易的移動模型SM(simple mobility)，在這模型中感測器節點被限制只能往事先定義好的方向移動。在這篇論文中，我們將會使用上述的移動模型來評估我們的演算法的效能。

三、系統架構

在本論文中，我們將會考慮一個無線行動感測器網路，主要的特性如下：

- BS 是固定的，而且距離所有的感測器節點有一段距離。
- 所有的感測器節點都是同質，並且都有能源的限制。
- 每一個感測器節點都有能力知道自己目前的位置，例如配備有 Location Finding System[2][17]。
- 所有的感測器節點在時間上都是同步的[3]。
- 每一個感測器節點都是有能力移動到任意的位置。

我們使用[8]中提出的無線電模組。E_{elec}=50 *nJ/bit*是電路所消耗的能源，ε_{amp}=100 *pJ/bit/m*^{2}是無線電傳送所需要消耗的能源。在這個無線電模型中，這無線電的傳送功率(transmission power)是可以被任意調整。我們同時也假設d^2是通道傳輸時的能源遺失率(energy loss)。傳送l個位元到d距離遠的地方，傳送和接收端所會消耗能源的模型是如下：

- 傳送端：$E_{Tx}(l,d) = l \times (E_{elec} + \varepsilon_{amp} \times d^2)$
- 接收端：$E_{Rx}(l,d) = E_{elec} \times l$

在資料融合所要消耗的能源，我們使用[8]中，所提出來的數據，E_D=5*nJ/bit*。也就是每次處理 1 位元所要消耗的能源是 5*nJ*。

四、叢集首選舉演算法

在這一節中，我們將會提出兩個叢集首選舉演算法：第一個演算法是使用計數的方式去決定叢集首；第二個演算法是使用感測器節點的位置去決定叢集首。

Cluster-Head Election by Counting

在這一小節中，我們將會描述一個分散式的叢集首選舉演算法 ACE-C (**A**lgorithm of **C**luster-head **E**lection by **C**ounting)。假設在行動感測器網路中，有 N 個移動感測器節點，我們將這些感測器節點從 0 到 N-1 依序編號。每一個感測器節點因此將會得到行動感測器網路中的唯一識別碼(unique ID)。假設我們預期在每一個回合將會產生 C 個叢集首。ACE-C 的流程如下所述。針對每一個感測器節點 v，我們使用 v_{id} 表示為每一個感測器

Algorithm of Cluster-head Election by Counting

Input: C: the number of cluster-heads in a round;

N: the total number of sensor nodes

(1) CH=0 /* number of cluster-heads */

(2) **while** CH < C **do**

(2.1) v_{id} = (v_{id}+1) mod N /* v_{id}: sensor node id*/

(2.2) **if** (v_{id} = 0) **then**

(2.2.1) v is a cluster-head;

(2.2.2) broadcast an advertisement message;

(2.2.3) increases CH by 1

endif

(2.3) wait a period of time to receive the advertisement message;

(2.4) **if** (message is received) **then**

(2.4.1) increases CH by 1

endif

endwhile

END

圖 2 演算法 ACE-C

節點 v 的編號，並且每一個感測器節點 v 都會經由每一個叢集首的廣告訊息(advertisement)得知到目前叢集首的數量。在每一次叢集首選舉步驟的開始，每一個感測器節點都會設定目前叢集首的數量是 0。針對某一個感測器節點 v，它首先會把自己的 v_{id} 加 1，並且取 v_{id} 除以 N 的餘數為 v_r。假設 v_r 等於 0，感測器節點 v 就是叢集首，然後它會發出一個廣告訊息給其他所有的感測器節點。

- 假設 v_r 不等於 0，感測器節點 v 就不是叢集首，然後它會等待接收由其他叢集首所傳送的廣告訊息。在接收到一個廣告訊息後，感測器節點 v 將會把叢集首的數量加 1。直到感測器節點 v 所記錄的叢集首數量為 C，才會停止叢集首選舉的程序。

圖 2 和圖 3 分別顯示這演算法和一個例子。在圖 3 中，有 9 個感測器節點，分別指派從 0 到 8 的編號。假設這預期叢集首的數量在每一個回合是需要 3 個。經過我們的演算法後，感測器節點 6，7，8 將在第一個回合中的前三個迴圈(loop)中會成為

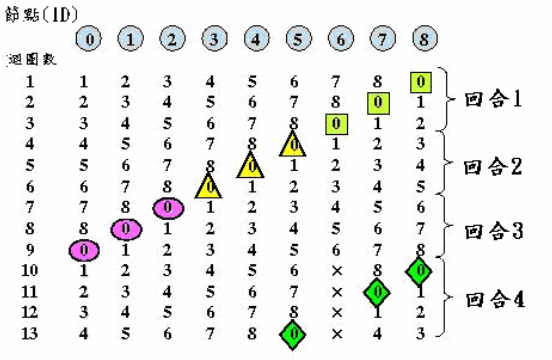

圖 3 使用 ACE-C 中有 9 個感測器節點的例子

叢集首。接下來的三個迴圈將會決定下一個回合的三個叢集首。圖 3 也顯示出當某一個感測器節點消耗了所有能源後的案例。假設感測器節點 6 在第 10 個迴圈成為叢集首。但是因為它的能源耗盡了，在第 12 個迴圈沒有向外廣告任何訊息。其他所有的感測器節點將會等待一個週期後，發現沒有接收到任何的訊息，它們將會假設感測器節點 6 的能源已經消耗殆盡，然後繼續下一個迴圈。在這

個時候，感測器節點 5 將在第 13 個迴圈會成為叢集首。因此，在第四回合中，叢集首是感測器節點 5，7 和 8。

Cluster-Head Election with Location

在這一小節，我們則提出一個分散式的叢集首選舉演算法 ACE-L(**A**lgorithm of **C**luster-head **E**lection with **L**ocation)，ACE-L 是應用在移動感測器節點，主要概念是使用感測器節點的移動性來讓每一個感測器節點輪流成為叢集首以平衡能源的消耗。簡單來說，每一個感測器節點是依據自己的位置來決定是否要成為叢集首。叢集首的產生是必須要由有意當叢集首的感測器節點先競爭無線通道後，才得以決定結果。

假設我們預期在行動感測器網路中，將會產生 C 個叢集首。這演算法的流程如下所描述。在系統運行的開始，我們給定 C 個固定的參考點(reference points)。這 C 個參考點將會影響到每一個叢集首的位置和每一個感測器節點去競爭無線通道的優先權。在每一個回合，當叢集首選舉的步驟開始，每一個感測器節點都是有意願去成為叢集首並且會去計算自己與所有參考點的距離，依據自己與最近的參考點距離去決定競爭無線通道的時間點。當競爭到無線通道後，該感測器節點就成為叢集首。考慮任意一個感測器節點 v。我們定義最接近感測器節點 v 的參考點為主要參考點(main reference point)。考慮到感測器節點的移動性和可以成為叢集首的機率後，我們設定感測器節點的延遲時間為感測器節點 v 目前的位置到主要參考點位置的距離。這延遲時間主要是決定了感測器節點要競爭無線通道的優先權。當感測器節點延遲了一段時間後，就會傳送一個信標(beacon)給所有的感測器節點，主要是用來競爭通道。假設感測器節點 v 接收到一個從其他感測器節點傳送的信標後，會依據該信標的感測器節點是否與自己有相同的主要參考點來考慮是否要繼續執行演算法。如果感測器節點 v 與接收到的信標的感測器節點有相同的主要參考點，就會停止去競爭通道，並且在該回合中將不會去成為叢集首。如果沒有，就繼續延遲一段時間，再去競爭通道。

我們用圖 4 舉出一個簡單的例子來說明，在該例子中，使用我們的演算法將會從 15 個感測器節點中產生 4 個叢集首(C=4)。假設這 15 個感測器節點是 v_1，v_2...，v_{15} 和這 4 個參考點是 rp_1，rp_2，rp_3，rp_4。考慮感測器節點 v_3。感測器節點 v_3 與所有參考點的距離分別是 d_1，d_2，d_3，d_4。我們可以清楚了解 d_1 是最短的距離，所以參考點 rp_1 就是感測器節點 v_3 的主要參考點。以 rp_1 為主要參考點的所有感測器節點中，感測器節點 v_3 將會有最短的延遲時間。因此感測器節點 v_3 將會是參照到參考點 rp_1 的叢集首。以此類推，感測器節點 v_5，v_6 和 v_{10} 將會是參照到參考點 rp_2，rp_3 和 rp_4 的叢集首。

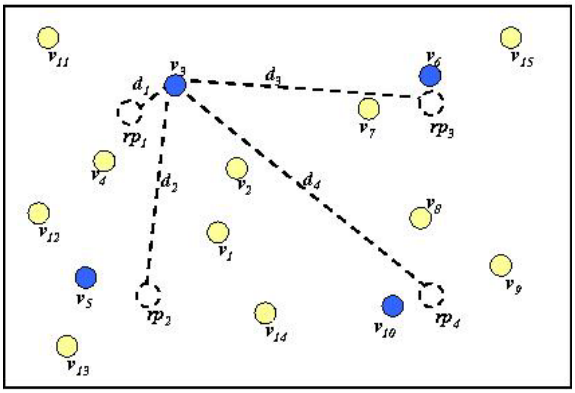

圖 4 使用 ACE-L 中有 15 個感測器節點的例子

圖 5 顯示出 ACE-L 的演算法。在步驟(1)，感測器節點 v 計算與所有參考點的距離，並且選擇距離最接近的參考點為主要參考點。在步驟(2)將會去檢查有沒有任何一個已經產生的叢集首是與自己有相同的主要參考點。假如有的話，這演算法將會停止。如果沒有的話，演算法將會繼續執行步驟(3)。在步驟(3)，感測器節點 v 設定延遲時間為感測器節點 v 與主要參考點之間的距離。在設定了延遲時間後，感測器節點 v 再一次的檢查是否有其他節點成為叢集首，並且是與自己有相同的主要參考點，其方式與步驟(2)相同。假如延遲時間已經超過了，感測器節點 v 將會去競爭通道並且傳送出信標而成為叢集首。演算法 ACE-L 的正確性主要是

Algorithm of Cluster-head Election by Location

Input: *C* reference points, *rp*

(1) compute the distance to all *rp* and
set distance to be the distance to the main *rp*;

(2) **if** receive *chb* from other node *u* **and**
nodes *u* and *v* have the same main *rp* **then**
/* *chb*: cluster-head beacon */
(2.1) cluster-head ← true;
(2.2) exit

(3) **else**
(3.1) *delay time* ← distance divided by 10;
(3.2) **while** (*delay time* decrease one) **do**
(3.2.1) **if** receive *chb* from other node *u* **and**
nodes *u* and *v* have the same main *rp* **then**
(3.2.1.1) cluster-head ←true;
(3.2.1.2) exit
endif
endwhile
(3.3) transmit its cluster-head beacon

endif

End

圖 5 演算法 ACE-L

標而成為叢集首。演算法 ACE-L 的正確性主要是來自於無線通道所具備的傳輸特性。在固定的頻率和相同時間的情況下，只有一個感測器節點可以使用該頻率去傳輸資料(除了使用 CDMA 機制)。因此在每一個時槽(time slot)中，只有一個感測器節點可以傳送叢集首信標。假設現在有 C 個參考點，rp_1，rp_2，…， rp_C。使用上述的特性，假如一個感測器節點使用參考點 rp_i 為主要參考點，當 $i < C$ 並且正在使用無線通道，在這一個時刻，其他所有也是使用 rp_i 為主要參考點的感測器節點將不能傳送任何信標，也就是說將不會有其他的叢集首使用相同的主要參考點。其它使用不同參考點的叢集首產生的方式也是相同的。因此，我們的演算法 ACE-L 將會產生 C 個叢集首，除非所有可以運作的感測器節點是不足 C 個。

叢集首選舉演算法的差異性

在這一小節中，我們將會去比較演算法 ACE-C 和演算法 ACE-L 的差異性。首先，因為演算法 ACE-C 只有使用唯一識別碼去決定是否要成為叢集首，而沒有考慮到位置的因素，所以產生出來的叢集首彼此之間的位置可能會非常的接近。如此的現象可能會產生叢集大小(在一個叢集內的感測器節點的數量)是非常的不平均，會出現某些叢集大小是非常的大，某些叢集的大小是非常的小。這樣的結果會使得叢集大小是比較大的叢集首消耗更多的能源，進而使得整個系統生命週期變短。演算法 ACE-L 使用本身位置距離參考點的距離來考慮是否要去選舉叢集首，因此可以避免上述叢集大小不均的結果。考慮在一個行動感測器網路中有

100 個感測器節點，我們預期每一個回合中，將會產生 5 個叢集首，所以預期每個叢集內將會有 20 個感測器節點。我們應用演算法 ACE-C 和演算法 ACE-L 分別去產生叢集首。圖 6 顯示是這叢集首大小的標準差的分布狀況。在這個例子中，我們是連續執行 200,000 個回合。在我們的例子中，可以清楚的瞭解使用演算法 ACE-L 所產生的叢集大小是比使用演算法 ACE-C 的結果要佳。更進一步的探討，在我們的結果中，演算法 ACE-L 產生的叢集中，有較多比例的叢集大小是接近 18 到 22 個之間。因此我們可以知道使用演算法 ACE-L 可以節省更多的能源消耗，在第五節的實驗模擬中將會有更多的結果讓我們知道這樣的趨勢。

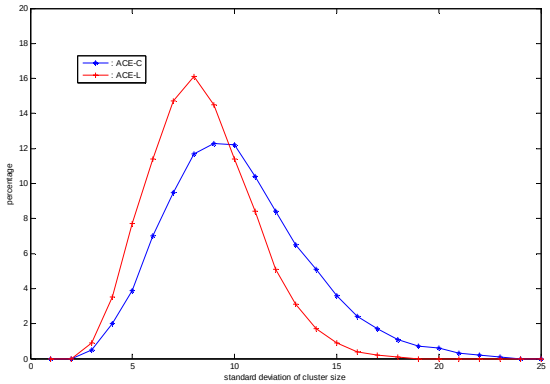

圖 6 使用演算法 ACE-C 和 ACE-L 在 20 萬回合中，所產生的叢集大小之標準差的分布

第二點，演算法 ACE-C 因為沒有考慮到位置的資訊，所以它可以適用在靜態和動態的感測器網路中。與演算法 ACE-C 的對比，演算法 ACE-L 特別是設計在移動式的環境中。當使用演算法 ACE-L 在一個靜態的感測器網路中，某些感測器節點將會快速的消耗能源，因為靜態的參考點將會使得某些感測器節點持續的成為叢集首。因此演算法 ACE-L 是不能適用於靜態的感測器網路中。如果要調整演算法 ACE-L 在一個靜態的感測器網路，一種可能是使用動態的參考點。無論如何，每一個感測器節點都必須要知道在參考點的移動模型。最後一點因為使用參考點，所以移動模型將會影響到 ACE-L 的效果。在本論文中，我們考慮三種移動模型。這些移動模型在演算法 ACE-L 中所產生的影響是比較小。假設有一種移動模型會使所有的感測器節點的分佈是非常的不平均，使用演算法 ACE-L 將不一定是一個好的選擇。反之，演算法 ACE-C 沒有特別對於位置資訊的需求，所以不論任何的移動模型都是可以適用。

五、實驗模擬

在這一小節中，我們將呈現在這論文中，我們提出的所有資料收集協定的模擬，並且和 LEACH 去做比較。目前已經有許多的資料收集協定已經被提出在靜態的感測器網路中，例如 PEGASIS 和 HIT。這些協定使用大量的訊息傳遞來建立一條比較好的鏈結以收集資料；因此這些協定的效能是比起 LEACH 要佳。無論如何，當網路拓樸改變的時候，使用的鏈結也應該要隨之被改變。在一個行動感測器網路中，網路拓樸變化得更為頻繁，PEGASIS 和 HIT 將不能適用在如此的網路型態中。而 LEACH 使用小量的訊息來建立叢集，而非使用需要不停更新的鏈結，所以可以被應用在一個移動環境中，因此我們考慮使用 LEACH 和我們的演算法做比較。

在這實驗中，我們所討論的有效率的使用能源是考慮這系統的所能運轉的回合數有多久。我們首先會實作 CM 這個協定，CM 主要是在組織叢集的步驟，所以在叢集首的選舉步驟，我們是使用與 LEACH 相同的機制。然後，我們把演算法 ACE-C 和演算法 ACE-C 加上 CM 的機制，分別產生 ACE-C 通訊協定和 ACE-L 通訊協定。這些實驗將可以顯示出這三個協定的結果都可以使系統生命週期延長。我們將會先討論這生命週期和比較這三個通訊協定。然後，我們驗證我們所提出的叢集首選舉演算法產生的叢集首數量是比 LEACH 還要平均。

在這些實驗中，我們考慮三個移動模型：SM 模型，RWM 模型和 RDM 模型。這感測器網路的區域是 200m × 200m，有 100 個感測器節點。每

表 1 針對不同的移動模型，來評估所有協定的效能

	第一個節點能源耗盡的回合				最後一個節點能源耗盡的回合			
	LEACH	CM	ACE-C	ACE-L	LEACH	CM	ACE-C	ACE-L
SM	304	309	413	253	590	602	832	984
RWM	384	426	628	646	530	555	780	907
RDM	458	501	712	663	530	560	767	882

一個感測器節點的速度是介於 0 到 1m/s 和初始的能源是 1.0J。我們讓每一個感測器節點在每一個回合中產生 2000-bits 的資料回傳給基地台。每一回合中的時間是 5 秒。在開始，所有的感測器節點是隨意的被放置在感測器網路所定義的區域內。當我們使用到 LEACH 的叢集首選舉的機制時，我們使用初始的機率 *P* 是 5%，所以這每一回合所產生的叢集首數量 5 個。

系統生命週期

我們現在將考慮系統生命週期，也就是整個行動感測器網路可以運轉多少個回合數。我們首先考慮發生第一個感測器節點能源消耗殆盡的回合。假設每一個回合所產生的叢集首都是相等的，能源的消耗就會更佳的平均分散在所有的叢集首上，因此也就是使整個系統的生命週期延長。在實際上，在每一個回合產生相等的叢集首可以使得第一個感測器網路能源消耗殆盡的回合延後。表 1 顯示出使用不同移動模型時，這平均的系統生命週期的回合數。在這個實驗中，我們從 40 次不同的網路拓樸中，分別取出第一個感測器節點和最後一個感測器節點能源消耗殆盡的回合。我們所提出的三個協定的效能都明顯的優於 LEACH，並且延長了整個網路的系統生命週期。在如此移動的環境中，CM 的效能比 LEACH 要來的好，主要是因為 CM 在組織叢集的機制。ACE-C 和 ACE-L 的效能使得系統的生命週期延長了更久的時間，主要是因為這兩個演算法在每一個回合都產生了相等的叢集首。當使用演算法 ACE-L 在 SM 的移動模型中，第一個感測器節點能源消耗殆盡的回合比較早的主要原因是在於 SM 模型的特性。SM 模型因為只有往四個方向移動，並且感測器節點在碰到邊界的時候就會選擇與原本的方向相反來前進，所以會出現在參考點附近移動的感測器節點，容易經常性的成為叢集首，所以第一個感測器節點能源消耗殆盡的回合數會比較早。從表 1 分析第一個感測器節點能源消耗殆盡的回合，我們可以簡單的結論，選舉叢集首的方式與移動模型之間是會互相影響的。

圖 7 顯示出在一個感測器網路的範圍為 200m × 200m 內，當每一個感測器節點的初始能源有 1.0J 時，在 RDM 的移動模型中，系統的生命週期的狀況。從圖上的點，我們所提出的演算法的系統生命週期的時間都比 LEACH 還要久；因此在能源消耗的方面，我們的演算法可以節省更多的能源。我們所提出的叢集首選舉演算法可以比 LEACH 的效能要好很多，主要的原因是我們的演算法避免了每一個回合沒有叢集首產生的狀況，也就避免了感測器節點直接把資訊傳送到基地台端。一般而言，因為每一個回合中，叢集首的分佈很平均，所以 ACE-L 的協定是比 ACE-C 的協定效能要更好。在某些回合中，當使用 ACE-C 的協定，會出現 5 個叢集首是互相接近。這樣的狀況是會消耗大量的能源，主要的原因我們已經在第四節討論過。

在實驗中，我們分別設定 4 個參考點和 5 個參考點。4 個參考點分別是在四個象限的中心點；5 個參考點，是以 4 個參考點為基礎，在放置一個參考點在整個網路的中心點。結果顯示使用 4 個參考點的效能比 5 個參考點要來的佳。接下來，我們使用第三節所引用的無線電模組和資料融合所要消耗的能源模式及數據，使用數學分析的方式去分析為什麼使用 4 個參考點會比使用 5 個參考點的效能佳。

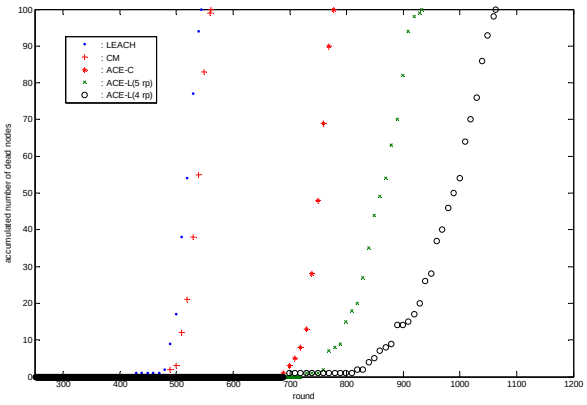

圖 7 行動感測器網路中，有 100 個感測器節點在 200m × 200m 的區域內，使用 RDM 移動模型的系統生命週期

首先我們分別定義幾個符號如下：

- k 為參考點的數量，也就是要產生叢集首的數量。
- N 為感測器節點的數量。
- l 為感測器節點在每一個回合要傳送的資料大小。
- d_{toBS}^2 為叢集首與 BS 的距離。
- d_{toCH}^2 為感測器節點與叢集首的距離。
- E_D 為進行一位元的資料融合所要消耗的能源。

我們計算一個叢集首所要消耗的能源為：

$$\begin{aligned} E_{CH} &= E_{Tx} + E_{Rx} + E_{data-fusion} \\ &= l \times E_{elec} + l \times \varepsilon_{amp} \times d_{toBS}^2 + \left(\frac{N}{k} - 1\right) \times l \times E_{elec} + \left(\frac{N}{k} - 1\right) \times E_D \\ &= l \times \left[E_{elec} + \varepsilon_{amp} \times d_{toBS}^2 + \left(\frac{N}{k} - 1\right)\left(E_{elec} + E_D\right) \right] \end{aligned} \quad (3)$$

我們計算出一個非叢集首所要消耗的能源為：

$$\begin{aligned} E_{non-CH} &= E_{Tx} = l \times E_{elec} + l \times \varepsilon_{amp} \times d_{toCH}^2 \text{ 。} \\ &= l \times \left(E_{elec} + \varepsilon_{amp} \times d_{toCH}^2 \right) \end{aligned} \quad (4)$$

由方程式 3 和 4 推算出一個叢集所要消耗的能源為：

$$\begin{aligned} E_{cluster} &= E_{CH} + \left(\frac{N}{k} - 1\right) E_{non-CH} \\ &= l \times \left\{ \left[E_{elec} + \varepsilon_{amp} \times d_{toBS}^2 + \left(\frac{N}{k} - 1\right)\left(E_{elec} + E_D\right) \right] + \left(\frac{N}{k} - 1\right)\left(E_{elec} + \varepsilon_{amp} \times d_{toCH}^2\right) \right\} \text{。} \end{aligned} \quad (5)$$

因此由方程式 5 來分別計算出 4 個參考點和使用 5 個參考點所要消耗的能源。

首先我們先計算使用 4 個參考點所消耗的能源。依據模擬環境的假設，我們把 4 個參考點的位置依序放置在座標為(50,50)，(50,150)，(150,50)和(150,150)的位置上。我們假設 $d_{toCH}^2 = \frac{100^2}{2\pi}$，依據方程式 5 可推導出：

$E_{cluster\_4} = 0.0127 + 0.0000002 \times d_{toBS}^2$ 。 (6)

利用方程式 6，我們依據每一個參考點距離基地台的距離，可以計算出使用 4 個參考點在一個回合中，平均所要消耗的能源為 $0.0864J$。

同樣地將 5 個參考點定在(50,50)，(50,150)，(150,50)，(150,150)和(100,100) 的位置之上。依據我們所假設 $d_{toCH}^2 = \frac{25^2 \times 3 + 50^2 \times 3}{2\pi}$ (四個象限)和 $d_{toCH}^2 = \frac{50^2}{2\pi}$ (中心點)的條件下，由方程式分別可以得知消耗的能源量：

[1] 四個象限

$E_{cluster\_5} = 0.0097 + 0.0000002 \times d_{toBS}^2$ ， (7)

[2] 中心點

$E_{cluster\_5} = 0.00551 + 0.0000002 \times d_{toBS}^2$ 。 (8)

利用方程式 7 和方程式 8，依據每一個參考點距離基地台的距離，可以計算出使用 5 個參考點在一個回合中，平均所要消耗的能源為 $0.0889J$。因此經由數學的分析，我們可以更清楚的了解到使用 4 個參考點的效能將會比使用 5 個參考點的效能要來的佳。

叢集首產生的數量

在第二節，我們討論到使用 LEACH 協定的叢集首選舉演算法的問題，在每一個回合中所產生叢集首的數量之差異性是非常的大，這可能會出現在某一些回合中，一個叢集首都不會被選舉出來。現在這實驗將顯示出當使用 LEACH 的叢集首選舉演算法，將會出現 12%到 15%的結果是沒有任何一個叢集首被選舉出來。如此的叢集首選舉演算法將會使的能源消耗的效能是非常的差。表 2 顯示出在每一個回合叢集首產生的分佈比率。這結果也顯

表 2 使用不同叢集首選舉演算法所產生叢集首數量的百分比

叢集首數目	0	1	2	3	4	5	6	7	8	超過 9
LEACH (%)	12	3	6	15	14	15	12	10	9	4
ACE-C (%)	0	0	0.2	0	0.1	99.7	0	0	0	0
ACE-L (%)	0	0.6	0.4	0.4	0.7	97.9	0	0	0	0

示出，假設系統預期在每一個回合中會有 *C* 個叢集首被選舉出來，我們的叢集首選舉機制將可以產生出 *C* 個叢集首，除了剩餘的感測器節點的數量是小於 *C* 以外。因為演算法 ACE-C 是使用計數的方式，所以演算法 ACE-C 比演算法 ACE-L 將會有更多的回合數是產生 C 個叢集首。

六、結論

在本論文中，針對於在行動感測器網路，我們提出兩個以叢集為基礎的資料收集協定，來解決資料收集的問題。在我們的以叢集為基礎的資料收集協定可以分為兩個步驟：(1)組織叢集階段，(2)資料傳遞階段。在組織叢集的階段中，我們分為叢集首的選舉步驟和叢集形成步驟。在叢集形成步驟中，我們所提出的 CM 機制可以減少感測器節點在加入叢集的過程中，選擇加入一個不適當叢集的機率；因此可以增加整個行動感測器網路的系統生命週期。在叢集首選舉的步驟中，我們提出了兩個分散式的叢集首選舉演算法，ACE-C 和 ACE-L。我們的叢集首選舉演算法實現了(1)避免一個回合中，沒有產生任何叢集首，(2)在每一個回合產生固定並且是預期的叢集首。在行動感測器網路上，結合 CM 的機制和我們所提出的叢集首選舉演算法，可以得到幾個不同的分散式資料收集的協定。

在本論文中，我們考慮了三個不同的移動模型，RWM，RDM 和 SM。在我們的實驗中，我們可以得知 RWM 和 RDM 是比 SM 還要更接近真實的網路狀況。我們所提出的資料收集協定，是可以應用在上述的移動模型內，並且可以延長網路的系統生命週期時間。

在未來的工作上，我們將繼續研究下述幾點：

1. 找出適當的參考點數量，以求讓演算法達到更佳的效能。
2. 在行動感測器網路中，考慮更多的移動模型所造成的影響。

七、參考文獻

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, no. 8, 2002, pp. 102-114.

[2] N. Bulusu, J. Heidemann, and D. Estrin, "Gps-less low cost outdoor localization for very small devices," *IEEE Personal Communication Magazine*, vol. 7, no. 5, 2000, pp. 28-34.

[3] A. Cerpa, J.Elson, D. Estrin, L. Girad, M. Hamilton, and J. Zhao, "Habitat monitoring: application driver for wireless communication technology," *Proceedings of ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean*, 2001, pp. 3-5.

[4] B. J. Culpepper, L. Dung, and M. Moh, "Design and analysis of hybrid indirect transmissions (HIT) for data gathering in wireless micro sensor network," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 8, no. 1, 2004, pp.61-83.

[5] V. Davies, *Evaluating Mobility Models within an Ad Hoc Network*, Master Thesis, Colorado School of Mines, USA, 2000.

[6] S. Hedetniemi, and A. Liestman. "A survey of

gossiping and broadcasting in communication networks," *Networks*, vol. 18, no. 4, 1988, pp. 319-349.

[7] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Adaptive protocols for information dissemination in wireless sensor networks," *Proceedings of ACM International Conference on Mobile Computing and Networking* (*MobiCom '99*), 1999, pp. 174-185.

[8] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Transactions on Wireless Communications*, vol. 1, no. 4, 2000, pp. 660-670.

[9] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed diffusion: a scalable and robust communication paradigm for sensor networks," *Proceedings of ACM International Conference on Mobile Computing and Networking* (*MobiCom '00*), 2000, pp. 56-67.

[10] P. Levis, S. Madden, D. Gay, J. Polastre, R. Szewczyk, A. Woo, E. Brewer, and D. Culler, "The emergence of networking abstractions and techniques in TinyOS," *Proceedings of the First Symposium on Networked System Design and Implementation (NSDI04)*, San Francisco, California, USA, 2004, pp. 1-14.

[11] S. Lindesy, 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, 2002, pp.924-935.

[12] C.-M. Liu, and C.-H. Lee, "Power-efficient communication protocols for data-gathering on mobile sensor networks," *Proceedings of IEEE Vehicular Technology Conference Fall*, 2004, pp. 4635- 4639.

[13] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu, "The broadcast storm problem in a mobile ad hoc network," *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, Seattle, Washington, 1999, pp. 151-162.

[14] D. Petrovic, R. C. Shah, K. Ramchandran, and J. Rabaey, "Data funneling: routing with aggregation and compression for wireless sensor networks," *Proceedings of the First International Workshop on Sensor Network Protocols and Application*, 2003, pp. 156-162.

[15] P. Levis, S. Madden, D. Gay, J. Polastre, R. Szewczyk, A. Woo, E. Brewer, and D. Culler, "The emergence of networking abstractions and techniques in TinyOS," *Proceedings of the First Symposium on Networked System Design and Implementation (NSDI04)*, San Francisco, California, USA, 2004, pp. 1-14.

[16] E. Royer, P.M. Melliar-Smith, and L. Moser, "An analysis of the optimum node density for ad hoc mobile networks," *Proceedings of the IEEE International Conference on Communications (ICC)*, 2001, pp. 857-861.

[17] A. Savvides, C. Han, and M.B. Srivastava, "Dynamic fine-grained localization in ad hoc network of sensors," *Proceedings of ACM International Conference on Mobile Computing and Networking* (*MOBICOM '01*), 2001, pp. 166-179.

[18] Jonathan Friedman, David Lee, Ilias Tsigkogiannis, Parixit Aghera, Advait Dixit, Sophia Wong, Aman Kansal, William Kaiser, and Mani Srivastava, " RAGOBOT: A new hardware platform for research in Wireless Mobile Sensor Networks" International Conference on Distributed Computing in Sensor Systems(DCOSS '05), pp. 412-412