具效率的無線感測網路物件偵測策略

Efficient Strategy of Object Detecting in Wireless Sensor Network

黃書彥

國立中興大學資訊管理學系碩士班

yellowbooky@hotmail.com

曹世昌

亞洲大學資訊科學與應用學系教授

sctsaur@asia.edu.tw

林詠章

國立中興大學資訊管理學系副教授

iclin@nchu.edu.tw

摘要

隨者科技的快速發展，無線感測網路可大量應用於環境的偵測，軍事用途以及住家監測等等，但由於其體積、無線感測節點的處理能力、記憶體與電力均受到限制，因此，對於無線感測網路的處理需求、儲存空間及電力的消耗都要盡可能降低。本篇主要在探討無線感測網路中物件的追蹤，追蹤包含了兩個動作：查詢以及更新，在本篇文章中，將應用樹狀結構來實做查詢和更新，以及更進一步的利用指標來減少物件的更新成本。

關鍵詞：無線感測網路、更新、查詢、追蹤。

1. 前言

無線感測網路中，每個感測節點都具有於其他節點的溝通能力，資料處理能力以及感應環境周圍的能力，無線感測網路可能由數十個到數百個，甚至是數千個節點所構成，每個節點能夠把所感應到的資料，傳回給一個主要節點，我們通常稱它為 sink，並且由這個主要節點統整，並整理出所需要的資訊，本篇所要探討的物件追蹤，包含了更新和查詢兩個動作，更新是指當某一物件 O 從某個節點 A 的感測範圍，移到另一個節點 B 的感測範圍時，這兩個節點以及相關的節點要去做更新節點內資料表格的動作；而追蹤是指，當一個人想要得知物件 O 在哪個節點位置附近時，會從 sink 這個主要節點發出一個查詢的封包，接著這個封包會傳到物件 O 所在區域的節點 C 上，並且此節點 C 會做一個

回傳的動作，並且把物件 O 的位置加入這個封包，之後傳回給 sink，所以這個人就能夠得知此物件 O 位於哪個節點附近。兩個或多個節點可以同時去感應物件，可以更為準確的去追蹤一個物件[7]。

找出 K 個最接近鄰居的查詢是一種新的概念[11]，一開始使用者要在空間中隨意找尋一個有興趣的點，叫做「查詢點」，這篇一樣是使用者從 sink 發出一個請求的封包，接著就能使用以全局樹狀架構為基礎的 GRT 演算法、以區域樹狀架構為基礎的 KBT 演算法或沒有基礎架構的 IKNN 演算法來找出最接近查詢點的附近的 K 個節點，並且取得這 K 個點的資料以及其位置。

Tsai 等學者於 2007 年提出了一種新的物件追蹤方法[6]，此方法 Face-track 類似螞蟻留氣味，把螞蟻比喻為一物件，螞蟻爬過的地方就會殘留著氣味，此時追蹤者就可以沿著螞蟻的氣味前進，進而到達螞蟻所在的地方；此篇使用了一種 Gabriel Graph (GG)去建構無線感測網路的基礎架構以及維持此基礎架構[8, 9]，在這個基礎架構之下，追蹤者只需要一開始用廣播的方式發送一查詢的封包，接著只需要跟著每個已經記錄著物件位置的信號點前進，並且使用 Face-track shortening 和 Loop face-track removing 的方式適時修正追蹤物件位置的策略，使得能夠更快的追蹤到物件以及達到更省電的目的。

目前有些樹狀結構被應用在無線感測網路的基礎架構上[2, 3, 5]，在[5]裡面，使用了Drain-And-Balance(DAB)的樹狀結構，在無線感測網路當中去追蹤物件，這個方法的查詢封包不需要總是洪水氾濫至整個無線感測網路中，而且物件的更新封包也並不需要總是傳遞到sink，只需要記錄在樹狀結構上其他的感測節點，有效率地改善查詢和更新這兩動作的成本。在[2]裡面，物件位置更新這個動作，可看成是[5]的物件位置更新動作的延伸，並且使用了Deviation-Avoidance Tree (DAT)和Zone-based Deviation-Avoidance Tree (Z-DAT)這兩個樹狀結構去減少了物件更新的成本，之後更再一步的使用了Query Cost Reduction (QCR)這個演算法去減少總成本，本篇便是基於這個基礎架構下，再使用指標來減少物件更新之成本。在[3]，此作者提出了一種概念，就是在一個無線感測網路中，不只有一點sink，把sink沿著整個基礎架構的密度去增加，也就是說每個sink有自己的樹狀結構，多個sink

時就有多個樹狀結構，而且每個單獨sink所構成的的樹狀結構可能會有所重疊，這樣子雖然能夠增加查詢的速度以及更好的負載平衡，但是顯然增加了不少更新的成本以及多點sink的高額架構成本，此降低的查詢成本並不符合多個sink和多個樹狀基礎架構重疊上的巨大成本。

在PCS系統中[1]，當手機使用者位置更新時，例如使用者從其所在的LA移動到另一個LA，就不用像傳統的二層架構，每次都必須更新HLR，只需要檢測其所在LA的VLR是否仍然相同，相同的話更新VLR的項目，不同的話則從原本的VLR做個指標，指向新的VLR，並且取消舊VLR的項目並增加新VLR的項目，這樣子可以節省的更新成本以及比率會很高，其作法只需要藉由指標以及虛擬層的判斷，而更新VLR的資料和指標即可，這樣子就不用每次都去更新HLR。而本篇便是基於此架構下的指標，來減少物件移動時的更新成本。

2. 物件查詢和更新

這個小節將介紹此篇文章的假設和定義，以及此無線感測網路的查詢和更新物件的方法。

2.1. 假設和定義

本篇的無線感測網路是屬於非叢集式(Non-cluster based)的無線感測網路，也就是說在整個無線感測網路之中，只有 sink 的存在，並沒有區分為許多叢集，當使用者需要得知某個物件 O 的所在位置，便從 sink 這個節點發出一個查詢的封包，接著這個封包可能用洪水氾濫法傳送至整個無線感測網路，也可能使用一種邏輯架構樹去傳送至無線感測網路，此時某個感測到此物件 O 的感測節點 S，收到此封包後將會回傳此封包至 sink 這個節點；並且假設當物件 O 位於無線感測網路中的任一個地方時，所有感測到此物件 O 的感測節點，將只有收到最強訊號的感測節點會回傳查詢物件 O 這個封包[10]；此時無線感測網路每個節點的感測範圍將可被切割成類似 Voronoi graph 的架構[4]，比如說某個物件 O 位在節點 S 的感測範圍多邊型中，除了節點本身 S 之外，沒有其他的感測節點會比節點本身 S 更靠近物件 O，當有查詢物件 O 的封包到達節點 S 時，只有節點 S 會回傳此封包到 sink。

2.2. 無線感測網路架構

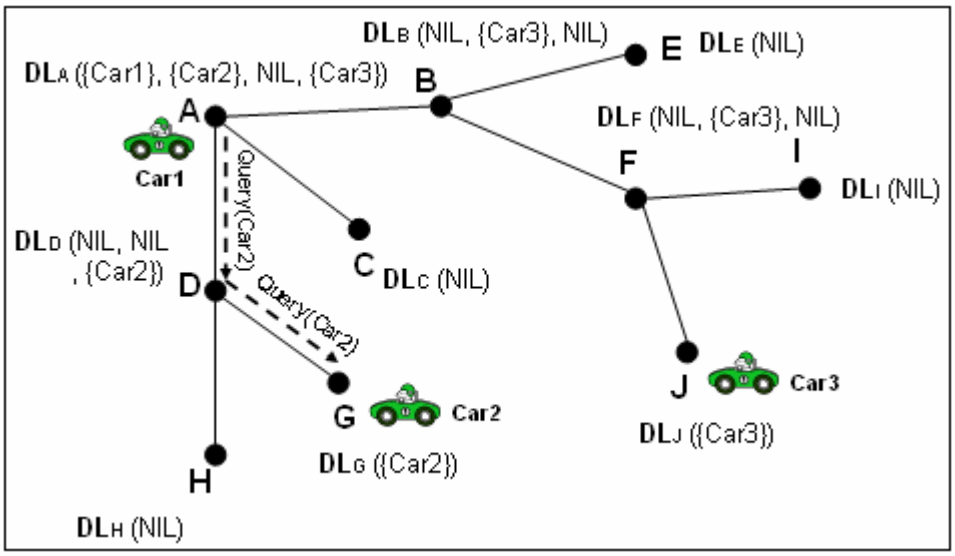

圖 1. (a) 樹狀基礎架構下的物件查詢。

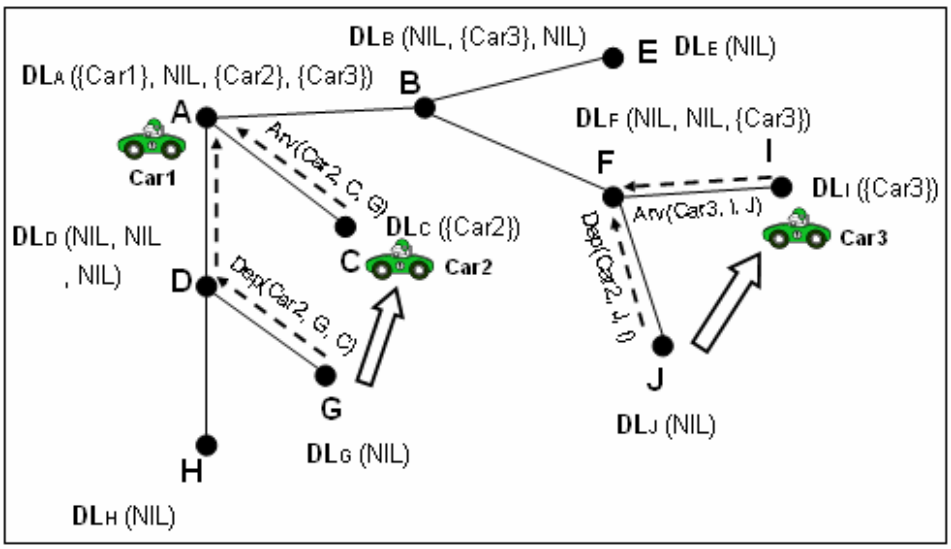

圖 1. (b) 感測節點表格更新情況。

當物件抵達或離開一個節點 S 的感測多邊型範圍後，一個偵測事件將被產生[2], [5]，圖 1.(a)表示物件查詢策略，每個感測節點裡面都有一個表格，存有自己所感測到物件的資訊以及其鄰近節點 (一次跳躍)的物件資訊，從 A 點這個 sink 的表格來看，存有自己感測到的物件 Car3，以及三個鄰近節點(一次跳躍)B、C 和 D 裡表格的物件資訊；從 D 點來看，由於 D 點的感測

範圍並沒有物件的資訊，所以表格裡第一個欄位並沒有物件的資訊，但其鄰近子節點 H 以及 G 中，G 有偵測到物件 Car2，故 D 表格的第三個欄位有 Car2 的資訊；而從 H 點來看，由於 H 點並沒有任何的子節點，H 點本身也沒有偵測到有物件在感測範圍內，故 D 點的表格只存有 NIL 的資訊，其它感測節點的資料結構則依此類推，要查詢的時後，則從 sink 發出一個查詢的封包，並依照 sink 裡紀錄物件資訊的表格，去對子樹發送封包，當物件所在的感測節點 G 收到此 Query 封包後，將會回傳此封包至 sink 節點。

圖 1.(b)表示物件更新策略，當 Car2 從 G 點移動到 C 點時，會從 G 點發出一個 Dep 的封包，主要是要告知 G 的祖先節點其 Car2 已經從 G 移動到 C，並且依序更新收到此封包節點的表格，此 Dep 封包會傳遞到 G 和 C 的共同祖先 A 而停止；同時 C 點亦會發出一個 Arv 的封包至 C 的父節點，此 Arv 封包仍會傳遞至 G 和 C 的共同祖先 A 而停止，並且依序更新收到 Arv 封包節點的表格；Car3 的移動方式亦同。

我們使用了 DAT 演算法去架構樹狀結構[2]，來建構出無線感測網路的基礎架構，當使用 DAT 演算法去建構出一個樹狀基礎架構的無線感測網路後，我們更進一步使用指標[1]來改善其每次物件從節點 1 移動到節點 2 時，都必須將更新的封包訊息送至節點 1 和節點 2 的最小共同祖先節點的溝通成本。

3. 提出方案

參考圖 2，我們假設一種情況，就是當 Car1 頻繁的在 K 和 G 之間移動時，每次物件的更新訊息都必須被傳送至 sink，當有多個物件時，而且每個物件都頻繁的在 K 和 G 之間移動，此時 sink 點的更新成本會非常大。我們在非叢集式的無線感測網路下應用 DAT 演算法去建立樹狀架構(圖 2、圖 3)，更進一步應用了指標的概念在此物件更新的策略[1]，來減少物件的更新比率和成本。

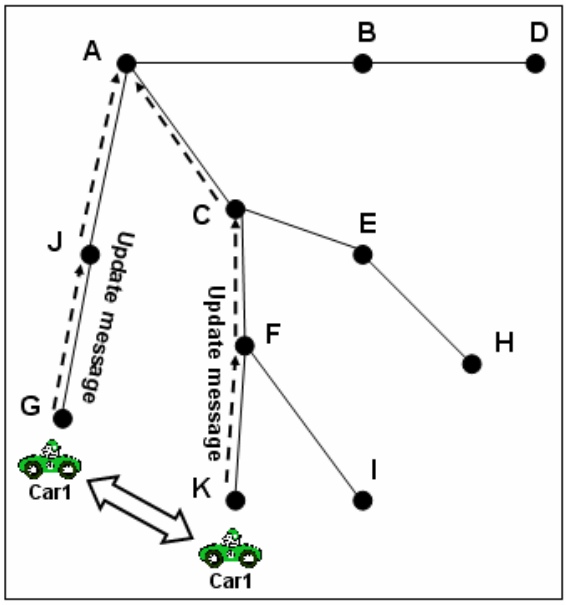

圖 2. 當 Car1 在 G 和 K 之間頻繁移動時的情況。

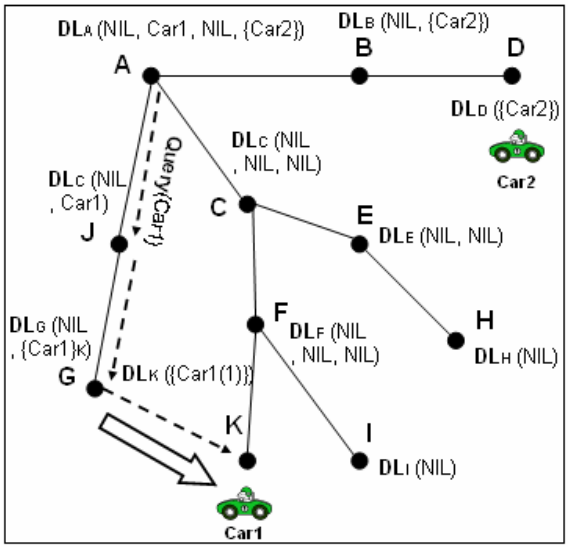

圖 3. 當 Car1 從 G 移動到 K 時，G 點的表格多了 Car1 資訊的指標指向 K，而 K 的表格也更新了 Car1 的資訊。

3.1. 使用指標的物件更新策略

1. 當一物件 O 從節點 X 移動到節點 Y 時，會檢查 X 是否為源於 Y 的子節點，或者 Y 是否為源於 X 的子節點，亦或者兩者都不是。
2. 如果 X 為 Y 的子節點時，則 X 更新表格第一個欄位為 NIL，Y 更新表格第一個欄位為 Car1，Y 表格裡記載 X 物件資訊的欄位更新為 NIL。
3. 如果 Y 是 X 的子節點時，則更新 X 表格第一個欄位為 NIL，X 表格裡記載 Y 物件資訊的欄位更新為 Car1，而 Y 更新第一個欄位則更新為 Car1。
4. 若皆非以上兩種情況，X 不會立刻傳遞 Dep(O, X, Y)封包至其父節點，而 Y 亦不會立刻傳送 Arv(O, Y, X)封包至其父節點，X 會在其表格中增加一個指向 Y 的指標，並且 Y 表格的第一個欄位將會更新為 Car1(1)，(1)是代表 X 有一個指標指向 Y，此時 Y 點會記錄 X 有個指標指向 Y，以及紀錄初始點 X(注意，之後的點都會記錄此 X，以便之後要沿原路傳回 Dep 封包)；注意，如果物件從 Y 再回到 X，則 X 會取消指向 Y 的指標，Y 則會更新 Y 表格第一個欄位為 NIL，代表 X 此時沒有個指向 Y 的指標。
5. 之後當 Car1 繼續任意移動，當從 Y 移動到 A，再移動到 B 時(A 和 B 可為父子節點或不是)，會繼續使用指標來指向

物件所在節點的位置，當到 B 時指標增加到 4(也就是{4})，會開始更新的動作，不再使用指標來指向物件，也就是會從 A 點發出 Dep(O, X, B)的封包，此 Dep(O, X, B)的封包會循原本的指標沿原路傳回 X 和 B 的共同祖先節點，收到此 Dep(O, X, B)封包的節點會依序更新所存的表格資訊；而 B 也會發出 Arv(O, B, X)的封包至 B 的父節點，此 Arv(O, B, X)的封包會沿著原本的樹狀結構傳回 X 和 B 的共同祖先節點，收到此 Arv(O, B, X)封包的節點也會依序更新所存的表格資訊。

圖 3 介紹了一個簡單的範例，當 Car1 從 G 移動到 K 時，並不會立即發出 Dep 和 Arv 的更新封包到 G 和 K 的共同祖先節點 A，這時候只需要將 G 的表格更新成 DLG (NIL, {Car1}K)，多了個 Car1 資訊的指標，這個指標指向 K，而 K 也會將其內的表格更新為 DLK ({Car1})，當使用者發出一個欲查詢 Car1 的查詢封包 Query(Car1)時，這個封包會依序由每個節點內表格的資訊依序往子節點傳遞，當此封包傳遞至 G 時，此時 G 點有個指標指向 K，此封包便繼續往 K 點傳送(注意，G 點和 K 點在 DAT 演算法下並沒有連接)，之後此查詢封包將會沿著原路回到 sink。

3.2. 使用指標的物件查詢策略

1. 當某個使用者從 sink 發出查詢的封包時，sink 會從其表格內找出使用者欲查詢的物件資訊位於 sink 的哪個子樹，之後 sink 便把封包往此子節點 X 傳遞。
2. 當 X 收到此封包後，一樣會從依其欄位找出其物件位於 X 的哪個子節點之後，之後依序往下傳遞此查詢封包，當傳遞到樹狀結構的最後一個子節點 Y 還沒有發現此物件時，代表此節點有個指標指向其他節點 Z。注意，Z 並不屬於 Y 的子節點，之後繼續把此查詢封包往 Z 傳遞，當送到某個節點後，而此節點的感測範圍擁有此物件時，此節點將會沿著原本此節點的樹狀結構回傳此查詢封包至 sink(注意，此查詢封包回傳時並不會沿著傳遞過來的路線回去)。

在這種基礎樹狀架構下，當同時有許多物件頻繁地在 K 和 G 之間移動時(圖 2)，使用指標能夠有效地減少物件更新的成本，以圖 3 來看，當 Car1 從 G 移動到 K 時，以原本的方法[2]去對物件做更新，

總共需要 6 次更新(A、J、G、C、F 和 K) ，但是以指標的方法去做更新的話，只需要做 2 次更新(G 的指標和 K)，尤其當物件數一多，能夠減少的更新成本會降低許多；而查詢的話更只需要多一次跳躍就行，雖然查詢成本增加了，但是降低的更新成本更有顯著的增加。

圖 4 我們說明一個應用指標在物件更新和查詢的實例，Car1 從從 G 點開始，經過 K 和 F 到達 E 時，各感測節點的表格及指標變化，一開始 Car1 在 G，此時 G 的表格為 DLG({Car1})，當 Car1 移動到 K，此時 G 表格更新為 DLG(NIL, {Car1}K)，K 的表格為 DLK({Car1(1)})，當 Car1 繼續從 K 移動到 F，K 表格更新為 DLK(NIL, {Car1(1)}F)，F 的表格更新 DLF({Car1(2)}, NIL, NIL)，之後 Car1 繼續往 E 移動，此時 F 表格更新為 DLF(NIL, NIL, NIL, {Car1(2)}E)，注意，F 表格增加的最後一個欄位為指向 E 的指標，之後 E 點依此類推。

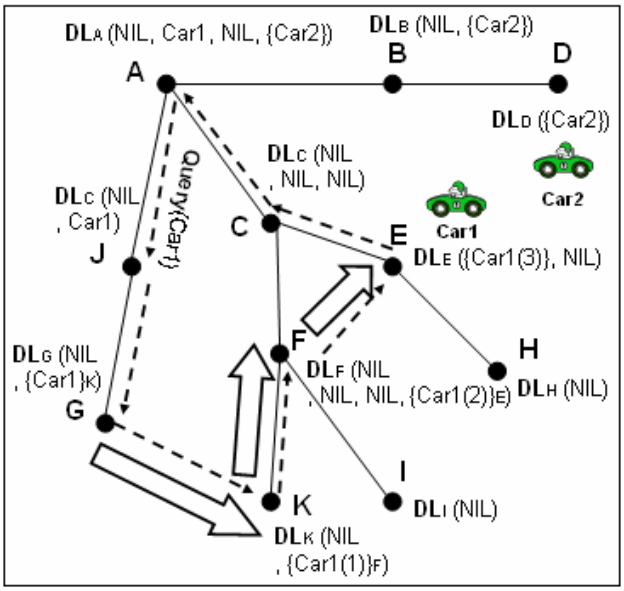

圖 4. Car1 從 G 點開始，經過 K 和 F 到達 E 時，各感測節點的表格及指標變化，虛線代表查詢封包 Query(Car1)的傳遞路線。

當使用者從 A 這個 sink 發出一個查詢的封包 Query(Car1)時，此封包的傳送路徑如圖 4，而回送路徑則沿著 E 原本的樹狀結構往 E 的父節點傳送；依本例來看，當 Car1 到達 E 時，如果 Car1 從 E 再度回到 F 時，E 表格會更新為 DLE(NIL, NIL)，F 表格會更新為 DLF({Car1(2)}, NIL, NIL)，若 Car1 從 E 移動到 B 的話，指標已經增加到 4，代表需要更新動作，此時 Dep(Car1, G, B)封包會沿著原本指標延源路往回傳遞到 G，G 再傳遞 Dep(Car1, G, B)封包回 A，此時所有收到此封包的感測節點會進行更新表格的動作，而 B 亦會依其

原本的樹狀結構傳送回 Arv(Car1, B, G)的封包，收到此封包的感測節點也會依序更新的動作。

4. 績效評估

在這一小節，我們將會評估此指標對於物件更新和查詢成本的影響，我們假設當更新感測節點表格時，新增指標和更改欄位的成本我們把它當作一樣，方便做估算，並且分析說明當物件頻繁的移動時，使用指標改善的更新成本會非常的大。

從圖 4 來看，當 Car1 從 G 移動到 E 時，我們的方法總共做了 6 次更新成本，但是如果是原本方法時[2]，總共需要 11 次更新成本，而查詢動作我們的方法總共需要 7 次跳躍，原本方法需要 4 次跳躍，通常更新感測節點的表格會比感測節點單純的遞送封包還需要消耗掉更多的能源，這會使得感測節點的能量耗損更快，而且當物件不斷的在 G 和 K 之間來回時，我們的方法只需要更新指標和復原指標即可，但是原本的方法則需要不斷的傳送 Dep 和 Arv 的封包往 A 點，此時更新成本會非常之大，造成感測節點的能源耗損的更快速，由於感測節點通常是用電池當作能量來源，因此能夠達到省電的效果是個必要考量。

假如我們的方法指標到達 4 時(圖 4，從 E 到 B)，會開始做更新的動作，此時會發出 Dep(Car1, G, B)的封包沿著 E、F、K 到 G 回去，並從 G 點回到 A 點，也會從 B 點發出 Arv(Car1, B, G)的封包，收到這些封包的感測節點，會開始更新的動作，我們的方法從 G 跑到 B，總共需要 13 次更新成本，但是原本的方法則需要 15 次更新成本；另一個情況，當 Car1 不從 E 點移動到 B，而移動到 H 時，總共需要 14 次更新成本，原本的方法需要 13 次更新成本，這是我們考慮當指標達到 4 的時候，需要再重新做更新成本的總和，我們的方法特別在物件橫跨過多個子樹時，其所降低的更新成本會很大，不用每次都需要不斷的往樹狀結構的前端送更新封包，因此當物件橫跨多個子樹前進時，明顯地會比原本的方法還要好，何況物件可能會常常在位於兩個不同子樹的感測節點間移動(圖 2)，所以指標能減少許多的更新成本。

5. 結論

在這篇文章裡面，我們提出了一種用於 PCS 方法的指標，來減少物件無謂的更新成本，並且在物件頻繁的在相鄰子樹之間移動時，有效的減少了兩個子樹的更新成本，雖然查詢時需要付出一點封包跳躍次數增加的代價，但是減少的更新成本更多，不但達到負載平衡，更能有效地降低感測節點電池能源的消耗量，我們使用指標的好處如下：

1. 每個節點的表格只需要增加一個指標的欄位即可，容易實做。
2. 當物件頻繁的在某兩個感測節點來回移動時，指標更可有效地節省能源消耗。
3. 物件的更新比率以及更新成本有效地降低。
4. 對於物件的移動能夠更有效地管理及感測。

由於此無線感測網路架構類似於分散式資料庫，需要一層一層查詢下去，因此對於許多階層式無線感測網路能夠以指標方式有效地減少更新的比率和成本。

7. 參考文獻

[1] C. C. Chang and I. C. Lin, "The Strategy of Reducing the Location Update Traffic Using Forwarding Pointers in Virtual Layer Architecture, " Computer Standards & Interfaces, Vol. 25, No. 5, September 2003, Pages 501-513.

[2] C. Y. Lin, W. C. Peng, and Y. C. Tseng, "Efficient In-Network Moving Object Tracking in Wireless Sensor Networks," IEEE Transactions on Mobile Computing, Vol. 5, No. 8, AUGUST 2006.

[3] C. Y. Lin, Y. C. Tseng, and T. H. Lai, "Message-Efficient In-Network Location Management in a Multi-sink Wireless Sensor Network," Proceedings of the IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing Vol. 1, 05-07 June 2006, pp. 496 – 505.

[4] F. Aurenhammer, "Voronoi Diagrams—A Survey of a Fundamental Geometric Data Structure," ACM Computing Surveys, Vol. 23, No. 3, pp. 345-405, Sept. 1991.

[5] H.T. Kung and D. Vlah, "Efficient Location Tracking Using Sensor Networks," Proceedings of IEEE Wireless Communication and Networking Conference, 2003.

[6] H. W. Tsai, C. P. Chu, and T. S. Chen, "Mobile Object Tracking in Wireless Sensor Networks," Vol. 30, No. 8, 8 June 2007, pp. 1811-1825.

[7] M. Ding, D. Chen, A. Thaeler, and

X. Cheng, “Fault-Tolerant Target Detection in Sensor Networks,” Proceedings of the IEEE on Wireless Communications and Networking Conference (WCNC 2005), Vol. 4, New Orleans, LA, USA, March 2005, pp. 2362–2368.

[8] Q. Huang, S. Bhattacharya, C. Lu, and G. Roman, “FAR: Face-aware Routing for Mobicast in Large-scale Sensor Networks,” ACM Transactions on Sensor Networks Vol. 1, No. 2, 2005, pp. 240–271.

[9] Q. Huang, C. Lu, and G.-C. Roman, “Reliable Mobicast via Face-Aware Routing,” Proceedings of the IEEE Conference on Computer Communications (INFOCOM’ 04), pp. 2108-2118, Hong Kong, China, March 2004.

[10] W. P. Chen, J. C. Hou, and L. Sha, “Dynamic Clustering for Acoustic Target Tracking in Wireless Sensor Networks,” IEEE Transactions on Mobile Computing, Vol. 3, No. 3, pp. 258-271, July, 2004

[11] Y. Xu, T. Y. Fu, W. C. Lee, and Y. C. Tseng, “Processing k Nearest Neighbor Queries in Location-aware Sensor Networks,” Signal Processing, Vol. 87, No. 12, December 2007, pp. 2861-2881.