多路徑合併路由選擇：無線感測網路中兩個省電的策略

Merged Multi-path Routing：Two Energy-Efficient Strategies in Wireless Sensor Networks

柯志鴻
長榮大學資訊管理研究所
kech@mail.cju.edu.tw

洪明祁
長榮大學資訊管理研究所
r26921083@mail92.cju.edu.tw

蘇溢芳
成功大學資訊工程研究所
emily@dblab.csie.ncku.edu.tw

劉裕秋
長榮大學資訊管理研究所
r26931151@mail93.cju.edu.tw

摘要

微型機電製造與無線通訊技術的快速進步，促成了無線感測網路的發展。無線感測網路是由數百或更多小型感測器所組成，主要的功能是去偵測環境，以及處理和儲存偵測到的資料，進而透過無線傳輸的方式傳送資料。目前無線感測網路運作時遇到的最大瓶頸，主要是受限於感測器的電量。從許多的研究發現，傳送資料所消耗的電量佔運作時整體的總耗電量相當大的比重。因此，本研究試圖在一個 source 多個 sinks 的環境中，去減少重複資料被傳送的情形，來達到省電的目的。我們提出兩個方法分別為 ODF 和 FCMN，主要是利用逐步比較感測器節點與其鄰居節點間，何者距離 sinks 較近的作法，去尋找合適的鄰居節點來接續資料的傳送。而其中 FCMN 更是具有低計算複雜度，且在實驗分析中也證明了利用這個方法所建立的路徑，比過去學者所提的方法更為有效率，能夠以較少的耗電量完成資料的傳送。

關鍵詞：無線感測網路、感測器、路由路徑、查詢

Abstract

Recent advances in the technologies for electronics and wireless communications have encouraged the development of wireless sensor networks. The wireless sensor network is composed of a great deal of sensor devices which are equipped with functions like sensing environment, data processing, and wireless communicating with each other. Currently, the power of batteries is one of major concerns for wireless sensor networks. Moreover, many papers have exhibited that data transmission always consumes most of whole energy consumption. In this paper we study how to reduce the energy consumption by avoiding transmitting same data in the environment with single source and multiple sinks. We proposed two algorithms, ODF (Overall Decrease First) and FCMN (The Full Cover with Minimum Neighbors), to find the nodes closer to sinks by comparing the distances between sinks and nodes step by step. The FCMN algorithm was designed with low complexity and justified in data transmission with lower energy usage than past works by simulation results.

Keywords: Wireless Sensor Network, Sensor, Routing Path, Query

1. 緒論

由於微型機電製造技術與無線通訊技術日益進步，使用者對無線設備的接受度大為提高，而感測器的應用也隨處可見，使得無線感測網路 (Wireless Sensor Network) 的發展受到許多研究機構和團體的重視，紛紛投入相關的研發工作。無線感測網路是由數百或更多的感測器 (sensors) 所組成，感測器主要能夠偵測環境變動，以及處理和儲存偵測到的資料，並以無線傳輸的方式傳送相關資

料給使用者。另外感測器還具備自我組織 (Self-Organization) 及通訊的能力，且能隨意的佈置。基本上，無線感測網路的擴充性高，感測器可以順利在不容易進入的地形中佈署，使用者能夠在遠處監控即可得到所需的資料，大幅降低人力佈置的危險和成本。

1.1. 無線感測網路相關應用

由於無線感測網路受重視之程度與日俱增，相關的應用也逐一而生，舉凡軍事佈署、醫療救援、生態環境、商業上皆有其應用，如此廣泛的應用範圍也是許多研究人員積極投入研發工作的原因之一。在軍事佈署的應用上，可以將感測器佈置在前線的戰場或不容易進入觀察的地形中，再從後方的接收器觀察敵軍的動態，作為戰略上的參考情報；醫療救援的應用方面舉例來說，可把感測器佈置在整棟醫療中心或放在病人的身上，觀察病人的活動情形以及身體內部細微的反應變化，幫助家屬和醫護人員了解病情，也減少醫護人員繁忙工作及醫療上的疏失；另外佈置在火災或其他災難現場的感測器，消防人員可以透過接收器正確地得知需要緊急救難的地點，提高救援效率以及降低不必要的救援傷害；生態環境的應用例子，如生物學家可以在廣大的區域佈置感測器，監控生物的活動習性或是監測自然環境的變動，做為災難發生的警報器，以便提早進行災難的預防工作；商業及其他的應用上舉例來說，可以監控車流量改善交通問題，提供便民服務或降低商業的運輸成本。

雖然無線感測網路有許多不同的應用，然而其運作的基本原則卻是相同的，即是在想要偵測的區域中佈置感測器，而使用者透過操作介面(例如，筆記型電腦或 PDA 等)發出查詢(Query) 給最近的感測器(稱之為 Sink)，然後 sink 便將此查詢廣播(broadcast) 出去，直到查詢傳送至可偵測到相關資料的感測器(稱之為 Source) 為止。之後，source 便將所偵測到的相關資料沿著一或多條路由路徑(routing path)回傳至 sink 給使用者。

1.2. 研究動機

無線設備的能源通常來自電池，而感測器也不例外。感測器微小的體積裝載電池提供能源，在電池容量的技術尚未有重大突破前，無線感測網路運作的主要瓶頸仍在電池的電量。因此，如何節省感測器的耗電量以便延長網路運作的生命週期，是一個相當重要的議題。

關於如何節省感測器耗電量這項議題，本論文從資料傳輸路徑的方向切入研究，主要探討 source 如何找出有效率的路徑回傳資料給 sink，以便減緩感測器的電量快速消耗。本研究藉由查詢屬性的相關程度，發現可以藉由減少重覆資料的傳送，來降低感測器傳送及接收資料的電量耗費，以便達到省電的目的。

1.3. 論文架構

本論文其餘的內容如下：第二章針對研究問題加以分析並且陳述相關的研究；本研究的環境及假設在第三章說明；第四章則描述所提出的演算法，並做進一步的討論；第五章針對所提的方法做效能的評估；結論和未來的研究方向在第六章。

2. 問題分析與相關研究

對於建立資料傳輸路徑這方面的問題，在這一節中我們將從『各類 sensors 的個數』、『查詢資料的相關性』及『查詢的函數』等三項因素去分析可研究的方向，並且也將說明本論文研究的部份及相關研究。

2.1. 問題分析

首先，就『各類 sensors 的個數』這項因素而言，所指的是 sinks 與 sources 兩類感測器個數的多寡。根據各類應用環境中 sinks 與 sources 的數目，我們可將其分為一個(single node)或多個(multiple nodes)兩種情況，如此便能產生圖 2-1 的四種環境，圖中的實心圓代表 sinks，空心圓代表 sources，其間的實線代表虛擬的路徑。

1. Single sink to Single source (SS)：此類環境如圖 2-1(a)，僅有單一 sink 發出查詢，且僅有一個 source 能偵測到查詢的相關資料。
2. Single sink to Multiple sources (SM)：此類環境如圖 2-1(b)，僅單一 sink 發出查詢，且有多個 sources 能偵測到查詢的相關資料，有許多學者提出這類環境的解決方案 [1, 3, 6, 7, 8]。

3. Multiple sinks to Single source (MS) ： 此類環境如圖 2-1(c)，有多個 sinks 發出查詢，且僅有一個 source 會偵測到相關的資料 [5]。
4. Multiple sinks to Multiple sources (MM)：此類環境如圖 2-1(d)，有多個 sinks 發出查詢，且同時有多個 sources 都能偵測到相關的資料。這類的環境最為複雜，許多研究學者提出 Multiple sinks to Single source 或 Single sink to Multiple sources 的解決方法，且試圖將所提的方法應用在其上 [1, 4, 5]。然而，實際上是無法完全的適用。

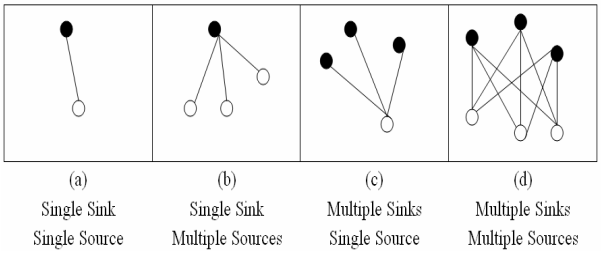


圖 2-1：各類 sensors 具有不同個數之環境

至於『查詢資料的相關性』這項因素，主要是討論使用者查詢所要求的資料相關程度，它可以根據查詢中的敘述語(Predicate) 判斷各查詢所要求資料的相關程度。我們依其相關性，大致上可作以下的三種分類：

1. 不同的查詢(Different Query)類型：此類型的查詢依其查詢屬性的相同與否，又可分為：
   (1) 屬性不相同：舉例來說，Query 1: 60 < light < 80，而 Query2: 30 < temperature < 40，兩個查詢的屬性 light 和 temperature 並不相同，因此可視為不同的查詢類型。
   (2) 屬性相同：當兩個查詢敘述中的屬性相同，但其要求的範圍不同時，即屬此類型。舉例來說，Query 1: 60 <light < 80，Query 2: 40 < light < 50，雖然屬性的部分 light 相同，但是查詢的範圍沒有交集。
2. 相同的查詢(Same Query)類型：此類型中的查詢屬性相同而且查詢範圍一樣，舉例來說，Query1: 60 < light < 80，而 Query2: 60 < light < 80，兩個查詢同樣都是針對 light 而且範圍都在 60 到 80 之間。
3. 相交的查詢(Overlapping Query)類型： 此類型查詢的屬性相同且查詢範圍不相同卻有交集。舉例來說，Query 1: 60 < light < 80，Query 2: 40 < light < 70，兩個同樣針對 light 屬性作查詢，而查詢範圍相交於 60 到 70 之間。

在『查詢的函數』這項因素上，我們主要是針對查詢的敘述中是否有聚合函數(Aggregation Function)去做進一步的分析。聚合函數是指 SQL 語法中含有 SUM、MIN、MAX、COUNT 和 AVERAGE 等五種函數，含有聚合函數的查詢舉例如下：

```
Select      Count(*)
From sensor  AS  s
Where       s.light > 60
```

上面的查詢主要的意思是“計算感測器的亮度大於 60(cd/m2) 的有幾個”，利用 COUNT 只需要累加個數，而不需要回傳所有感測器的亮度資料。利用聚合函數建立資料回傳路徑的研究有[1, 2, 6]，大都討論 multiple sources 建立資料回傳路徑時，如何將路徑合併。其中 [1]提出的方法，需要配合傳輸的資料量進行計算，傳送資料量大的 source 將以最短的路徑到達 sink，以便減少節點傳送資料。傳送資料量小的路經必須配合資料量大的路徑，所以通常會拉長與 sink 的傳輸距離。而[2]提出的方法是越早合併越省電，在 [6] 中也有提到其他的聚合函數，如 MEDIAN、DISTINCT 和 HISTOGRAM，但是經過實驗分析後發現，能夠節省的耗電量不夠顯著。

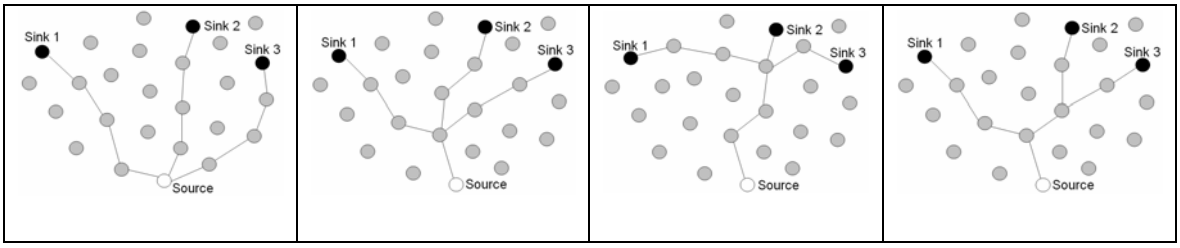
經由上述的分析，可以將問題歸納成二十四種情況，如表 2-1 所示。但是實際上只有十六種情況可能發生，在表中以“✓”代表可能發生的情形，反之“✗”代表不可能發生的情形。不可能發生的情形乃是因在 single sink 的應用環境下，只有單一的查詢，因而不會發生 Same Query 及 Overlapping Query 的情況。

表 2-1：建立資料傳輸路徑問題的分類

	Single Sink Single Source	Single Sink Multiple Sources	Multiple Sinks Single Source	Multiple Sinks Multiple Sources
Different Query + Aggregation	✓	✓	✓	✓
Different Query + No Aggregation	✓	✓	✓	✓
Same Query + Aggregation	x	x	✓	✓
Same Query + No Aggregation	x	x	✓	✓
Overlapping Query + Aggregation	x	x	✓	✓
Overlapping Query + No Aggregation	x	x	✓	✓

2.2. 問題描述

過去許多研究 [1, 2, 3, 6]集中於探討，當一個 sink 發出查詢向多個 sources 要求回傳資料時，如何選擇資料回傳的路徑以便減少回傳的資料量。而本論文所要探討的是另一種環境的問題，考慮當一個 source 要回傳相同資料給多個 sinks 時，資料傳送路徑該如何建立，才能有效降低系統整體的耗電量。最基本的路徑選擇，即是為 source 與每一個 sink 各自找出一條傳輸路徑。舉例來說，如圖 2-2(a) source 分別與三個 sinks 建立傳輸資料的路徑，即使節點之間傳輸的資料是相同的。倘若傳送的是一個單位的資料量，則圖 2-2(a) 建置的路徑總共花費十二個單位的傳輸資料量。反觀圖 2-2(b) 建立合併的路徑，僅花費十個單位的傳輸資料量。由此我們可以知道路徑合併確實可以減少傳送的資料量，達到省電的目的。然而過度的合併路徑相對的增加 sink 接收資料時間的延遲，如圖 2-2(c)所示，總共花費九個單位的資料傳輸量，但是 sink 1 和 sink 3 接收到資料的時間卻比各自建一條傳輸路徑更久。圖 2-2(a)以四個 hop counts 的時間將資料傳送各個 sinks，圖 2-2(c)將資料傳送傳給三個 sinks 卻分別花費六個、四個和五個 hop counts 的時間。由此例可發現路徑何時合併這個問題的重要性及複雜度，本論文主要便是針對多個 sinks 向同一 source 要求相同資料時，討論資料傳送路徑在何處應該合併，而在何處資料應該如何分開傳送才能夠達到真正省電的目的且不增加傳輸資料的延遲。在圖 2-2 的例子中，最佳的傳送路徑應該如圖 2-2(d)所示，總共花費九個傳輸資料量，而且皆以四個 hop counts 的時間即可將資料傳給三個 sinks。



(a) (b) (c) (d)

圖 2-2：各種多路徑傳輸的示意圖

2.3. 相關研究

關於多個 sinks 向單一 source 要求資料的傳送路徑問題，曾有[5]提出 SEAD 的方法。基本上，SEAD 的方法適合在 mobile sink 的環境下運作，主要是以節點之間的距離和封包流量比率(packet traffic rate)作為計算的參數，建立 D-Tree (傳輸資料的樹)。SEAD 方法把 sink 加入移動的變數，因此 SEAD 的方法中有一部分是 sink 移動的管理。在 SEAD 的方法中主要是讓移動的 sink 進行尋找鄰近不會移動的節點作為替代 sink，稱之為存取點(Access node)。藉由存取點發出 join query 與已經存在的傳送路徑樹建立分支，而 D-Tree 的建立分為 Gate Replica Search 和 Replica Placement 兩個部份。Gate replica search 主要是找出樹上作為分支的位置，儲存 source 的資料備份，後來的存取點只需要直接與 gate replica 建立分支；Replica Placement 的部份則是判斷 gate replica 與存取點之間需不需加入分支點，若不需要新增分支點(Non-replica mode)，則存取點直接與 gate replica

建立傳輸資料的分支。若是需要加入分支點(Junction replica mode)，則 gate replica 與 junction replica 先建立分支，存取點再與 junction replica 建立分支。就 D-tree 建立的部分而言，與本研究所提出的建置路徑概念是類似的，但是利用存取點去接收資料的時間會有延遲的狀況，因為存取點是直接與已經存在的 gate replica 建立分支，所以可能造成存取點與 source 之間不是最短路徑，在後面的實驗章節將會加以比較討論。

3. 環境與假設

此章節將對無線感測網路的環境作介紹並陳述研究的假設，另外也定義了演算法中所要用到的名詞、符號及資料結構：

1. 感測節點(sensor node)
   - 感測節點皆配備電池作電力供應，並且能感測和傳送環境中的資料。
   - 感測節點有固定的位置。
   - 每一個感測節點都有一個 node ID。
   - 當兩個感測節點能夠直接互相通訊，不需藉由其他感測節點代傳訊息時，此兩個感測節點互稱為對方的鄰居節點(neighbor node)。
   - 每一個感測節點知道它的周圍有哪些鄰居節點。
2. 查詢(query)
   - sink 利用 flooding 的方式傳送查詢至 source 要求資料。
   - sink 發出的是 continuous query，此類的 query 將會要求 source 在某一段時間內依某個頻率(e.g. 一分鐘 一次)持續回傳資料給 sink。
   - source 接收到查詢的要求，會將使用者所要的資料依所選取的傳送路徑回傳至 sink 給使用者。

為了後續演算法內容的描述方便，在此我們先定義相關的符號：

- Qi：ID 為 i 的 query type。
- Nj：ID 為 j 的感測節點(簡稱為節點)。
- Sk：ID 為 k 的 sink。
- S(Qi)：發出 Qi 至同一 source 要求相同資料的所有 sinks 所構成的集合。也就是說，S(Qi)={Sk | Sk 是發出查詢 Qi 至 source 取得資料的 sink}。
- B(Nj)：Nj 的鄰居節點所構成的集合，i.e., B(Nj)={Nk| Nk 是 Nj 的鄰居節點}。
- H(Nj, Qi)：所有發出 Qi 的 sinks，它們在 Nj 的 hop-count value(HCU) 為分量(component)所構成向量的集合。也就是說，H(Nj, Qi) = {(C1, C2, …, Cm) | Ck 代表第 k 個傳送 Qi 的 sink 在 Nj 的 HCU 值，m 為發出 Qi 的 sinks 數，且 1≦k≦m，k 是整數}。此處所提到的 HCU，我們將在稍後作詳細的定義。
- T(Nj, Qi)：在 Nj 的鄰居節點中，被選擇來接續回傳查詢 Qi 的 result 的所有鄰居節點所構成的集合。換言之，T(Nj, Qi) = {Nk | Nk 是被 Nj 選擇來接續傳送 Qi 的 result 的節點，此處 Nk 屬於 B(Nj)}。

在此，我們利用例子來說明上述的四個符號：S(Qi)，B(Nj)，H(Nj, Qi)，T(Nj, Qi)。圖 3-1 是一個無線感測網路的簡單示意圖，在圖中，黑色實心的節點代表 sink，空心的節點代表 source，其餘灰色的節點代表一般的 sensor nodes。各節點間若有灰色虛線連接，代表節點間可以彼此通訊；反之，即表示節點間無法直接進行通訊，而須經由其他節點的轉傳來進行間接通訊。在圖 3-1 中，假設 sinks X 和 Y 發出 Q1，則可以得知 S(Q1)={X, Y}。另外，以 source 節點 S 為例，有 4 個鄰居節點 A、B、C 和 D，所以 B(S) = {A, B, C, D}。

接下來，在舉例說明符號 H(Nj, Qi) 前，我們先對 hop-count value(HCU)的意義以及如何得知其數值加以描述。當一個 sink，如圖 3-1 中的節點 X，發出查詢去要求資料時，將會有一個相對應的 HCU 值被記錄在每一個接收到此查詢訊息的感測節點中。[2]中有描述各感測節點如何計算得知本身應記錄的 HCU 值，在底下我們說明它的計算規則。

首先，發出查詢的 sink 本身(也就是圖 3-1 中

的節點 X)紀錄 HCU 為 0。接著，針對那些可以『直接』接收到從 sink 所送出查詢的 sensor nodes，也就是感測節點 X 的鄰居節點 E、F 和 G，它們所應記錄相的 HCU 便是感測節點 X 的 HCU 值加 1，也就是 1 (= 0+1)。依循這樣的原則，所有其他的節點便可以根據它們是從那一個感測節點接收到此查詢訊息，而能計算出應記錄的 HCU 值。然而，值得更進一步說明的是，任一個節點 N 可能接收到兩個或兩個以上的鄰居節點轉傳查詢訊息過來，並且這幾個鄰居節點的 HCUs 值可能都不相同。此時，節點 N 所應記錄的 HCU 值，就是這些鄰居節點 HCUs 的最小值再加上 1。如圖 3-1 中的

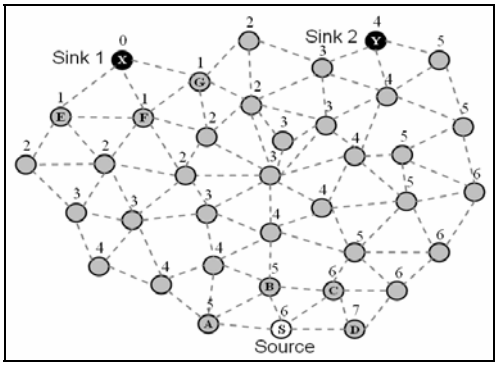


圖 3-1：感測節點 X 發出查詢時，各節點記錄的 HCU 資訊

節點 S，同時會接收到節點 A、B、C 和 D 轉傳查詢訊息過來，而這些節點的 HCUs 分別是 5、5、6 及 7，因此節點 S 應該記錄的 HCU 為 MIN(5, 5, 6, 7)+1 = 6。就 HCU 值實際所代表的意義上而言，一個節點 S 所記錄的 HCU 值，即表示發出此查詢的 sink，中間至少必須經過 (HCU-1) 個感測節點的轉傳，才能將查詢訊息傳送至節點 S。

接下來以圖 3-2 為例，假設二個 sinks X 及 Y 都發出查詢 Q1 去 source S 要求資料，網路中的任意一個節點將會有兩個 HCU 值，分別相對應於發出查詢的兩個 sinks。舉例而言，圖中的節點 A，它有兩個 HCU 值，5 和 6，其中的 5 是對應於 sink 1 的 HCU 值，而 6 則是對應於 sink 2 的 HCU 值。我們將這兩個 HCU 值 model 成一個向量(vector)的兩個分量(components)，便形成向量(5, 6)，我們將此一向量稱為 Hop-Count Vector (HCV)，而此一向量所構成的集合，則用符號 H(A, Q1) 表示之。因此 H(A, Q1)= {(5, 6)}。

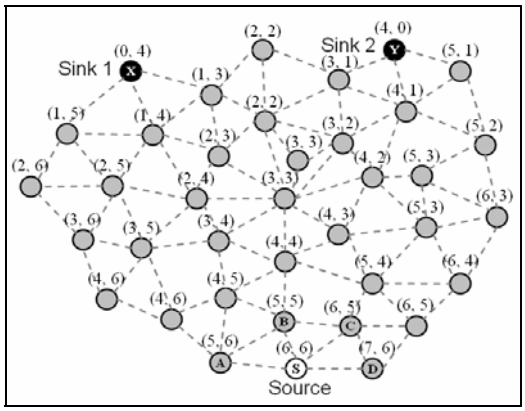


圖 3-2：二個 sinks 時，節點的 HCV 值

接下來，我們說明符號 T(Nj, Qi)的意義，仍然以圖 3-2 為例。當 sink 1 及 sink 2 的查詢都送達 source S 後，source S 必須設法建立傳送路徑將查詢的 result 回傳給 sink 1 及 sink 2。很明顯地，source S 只能將 result 傳送給它的鄰居節點，之後再由鄰居節點接續轉傳出去。而 S 可以從鄰居節點的集合 B(S) = {A, B, C, D}中挑選某些節點，使其成為轉傳的節點。被 source S 挑選出來的節點，我們便將它們放在集合 T(S, Qi)中。假設所選取的鄰居節點為 B，則 T(Nj, Qi) = { B }。接著，再從節點 B 接續選取下一個轉傳的節點，一直持續到至每一個 sink 的傳送路徑都被建立為止。

最後，我們說明感測節點記錄各種資訊所使用到的資料結構。

- HCU table ：用來記錄感測節點相對應於各 sinks 的 HCUs，欄位說明如下：

  Sid： 發出查詢的 sink 編號。

  Value： 相對於 sink 的 HCU。

Sid	Value

圖 3-3：HCU table

- NHCU (Neighbor's HCU) table ：用來記錄鄰居的 HCUs。當 sink 發出查詢後，每個感測節點在轉傳查詢時，必須記錄是從哪一個鄰居節點收到查詢，以及此鄰居節點的 ID 及 HCU 值。整個表格的欄位意義說明如下：

  Bid： 鄰居節點的 ID。

  Sid： 發出查詢的 sink 編號。

Qid： 編號為 i 的查詢。

Value： 鄰居節點的 HCU。

Bid	Sid	Qid	Value

圖 3-4：NHCU table

4. 演算法

當多個需求相同資料的 sinks 向同一 source 發出查詢時，建立傳送路徑最簡單且基本的作法，即是為 source 至每一個 sink 個別建立一條獨立的傳送路徑，如圖 2-2(a)。很明顯地，這樣的作法是非常耗電且沒有效率的。本研究試圖利用感測節點的 HCV 資訊，根據某些規則去建置資料的傳送路徑。在建置路徑的過程中，儘可能將相同的資料僅沿著一條路徑傳送來減少耗電量，然而在實際的情況中，並無法僅僅利用唯一一條路徑就能傳送同一份資料給分散在不同地方的所有 sinks。因此，我們提出兩個演算法來達到下列幾項目的：(1) 儘可能讓相同的資料保持在一條路徑上傳送；(2) 設計判斷規則去找出在那些節點時，資料就應該從一條傳送路徑重覆至多條路徑上去傳送；(3) 當資料的傳送必須從一條路徑變為多條時，決定出多少條才是最佳的；(4) 決定出總耗電量最低的傳輸路徑。

4.1. The Overall Decrease First(ODF) Algorithm

所提的第一個演算法稱之為 Overall Decrease First (ODF)，其基本想法是，由 source 開始，依序將感測節點與其鄰居節點的 HCV 作量化的比較，根據比較的結果去決定出那些鄰居節點適合成為傳送路徑上的節點。接著，再以被選擇的節點依相同的原則反覆去決定接續傳送的節點，直到 source 和各 sinks 之間的路徑都被建立為止。演算法中一個重要的步驟即是對節點的 HCV 進行量化的比較，藉以選取接續傳送的節點，所以接下來我們先說明如何進行量化的比較以及選取的規則，再描述 ODF 演算法的步驟。

4.1.1 The Rules of Sensor Selection (RSS)

當多個 sinks 發出查詢去同一 source 要求相同資料時，網路中每個感測節點都將記錄相對於這個查詢的 HCV 資訊，其中的各個分量值即是相對應於各個 sink 的 HCU。經由我們進一步的分析後發現，在任一感測節點與其鄰居節點所記錄的 HCVs 中，相對應於同一 sink 的分量值，其差值必定是 -1, 0, 或 1。我們由圖 4-1 舉例來說明，圖上的數字代表由 sink 1 發出查詢時，每個節點所記錄的 HCU 資訊。圖 4-1 節點 B 的鄰居節點集合為 B(B)={A, C, H, I, S}，它們的 HCU 分別是 5、6、4、4 和 6，而節點 B 的 HCU 為 5，可以容易地計算出節點 B 與五個鄰居節點的 HCU 差值確實是 -1, 0, 或 1。在多個 sinks 的環境中，相對應於同一 sink 的分量，其差值仍然保証為 -1, 0, 或 1。

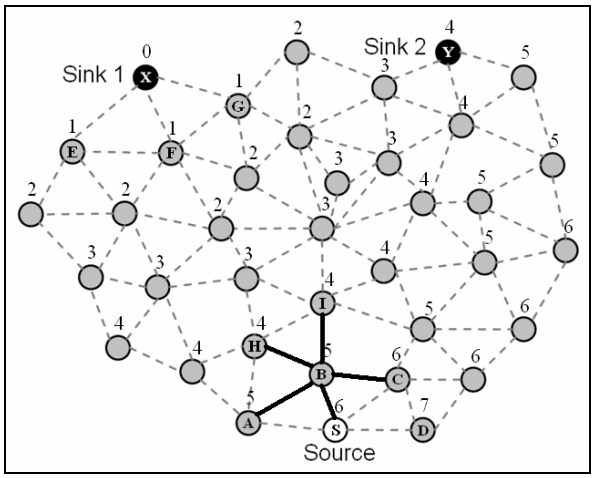


圖 4-1：節點 B 與鄰居節點的 HCU 差值

接著，我們說明上述三種差值所代表的意義。在此，將使用符號 T(i, H(Nj, Qm)) 去代表感測節點 Nj 所記錄的 HCV 向量中，發出查詢 Qm 的第 i 個 sink 的 HCU。也就是說，如果感測節點 Nj 所記錄的 HCV 向量是 (C1, C2, …. Cn)，便可以得到 T(i, H(Nj, Qm)) = Ci，這裡 Ci 代表發出查詢 Qm 的第 i 個 sink 的 HCU。假設以感測節點 Nj 為例，其鄰居節點為 Nk，將兩者 HCVs 中同一個分量進行比較，將會是下列三種情況之一：

1. T(i, H(Nj, Qm)) > T(i, H(Nk, Qm))：事實上，此時 T(i, H(Nj, Qm)) - T(i, H(Nk, Qm)) = 1，如圖 4-1 中的感測節點 B 和它的鄰居節點 H 和 I 。這種情況代表鄰居節點 Nk 比感測節點 Nj 更接近第 i 個發出 Qm 的 sink。
2. T(i, H(Nj, Qm)) = T(i, H(Nk, Qm))：此時 T(i, H(Nj, Qm)) - T(i, H(Nk, Qm)) = 0，如圖 4-1 中的感測節點 B 和它的鄰居節點 A。這種情況

代表鄰居節點 Nk 和感測節點 Nj 跟第 i 個發出 Qm 的 sink 的距離相等。

3. T(i, H(Nj, Qm)) < T(i, H(Nk, Qm))：實際上，此時 T(i, H(Nj, Qm)) - T(i, H(Nk, Qm)) = -1，如圖 4-1 中的感測節點 B 和它的鄰居節點 C 和 S 。這種情況代表鄰居節點 Nk 比感測節點 Nj 離第 i 個發出 Qm 的 sink 更遠。

上述的三種情況只是針對相鄰兩節點的 HCVs 中單一分量的值去進行比較，若同時考量相鄰兩節點的 HCVs 中所有分量的比較，情況將變得相當複雜。為了後面內容的說明方便，我們將用比較簡單的符號來表示感測節點的 HCV 向量。我們以 X=(X1, X2, …, Xk)表示發出查詢 Qm 的感測節點 Nj 所記錄的 HCV 向量。因此，X= H(Nj, Qm)，且 Xi = T(i, H(Nj, Qm))。另外，令 Y=(Y1, Y2, …, Yk) 表示感測節點 Nj 的鄰居節點 Nk 所記錄的 HCV 向量，所以可以得到，Y = H(Nj, Qm)，且 Yi = T(i, H(Nk, Qm))。如此一來，上面三種情況便可以寫成 (I) Xi > Yi；(II) Xi = Yi； (III) Xi < Yi。若寫成通式，便是 Xi？Yi ，其中“？”可能是“ >”、“ = ”或“ < ”。

R1.	X1 > Y1, X2 > Y2
R2.	X1 < Y1, X2 < Y2
R3.	X 1 = Y1, X2 = Y2
R4.	X 1 > Y1, X2 < Y2
R5.	X 1 < Y1, X2 >Y2
R6.	X 1 > Y1, X2 = Y2
R7.	X 1 = Y1, X2 > Y2
R8.	X 1 < Y1, X2 = Y2
R9.	X 1 = Y1, X2 < Y2

圖 4-2：具有兩個分量的 HCVs 比較結果

現在我們針對每個 HCV 向量中有兩個分量的情況加以說明。令 X=(X1, X2) 且 Y=(Y1, Y2)，因為兩個 HCVs 的單一分量的比較結果就有上述的三種情形，所以在兩個分量的情形下，將會產生如圖 4-2 的九種可能比較結果。

若再進一步針對這九種可能的比較結果進行分類，且以向量 Y(鄰居節點的 HCV) 相對於向量 X (感測節點 Nj 的 HCV) 的角度去看，將可以得到下列的六大類：

Class 1. 兩個分量都變小：如圖中的 R1。此類的情況若發生，代表鄰居節點 Nk 與發出查詢的兩個 sinks 的距離(hop count)都比感測節點 Nj 還近。

Class 2. 一個分量變小，且另一個分量相等：如圖中的 R6 及 R7。此類的情況若發生，代表鄰居節點 Nk 與發出查詢的兩個 sinks 之中的一個距離比感測節點 Nj 還近，而與另一個 sink 的距離則和感測節點 Nj 一樣。

Class 3. 一個分量變小，且另一個分量變大：如圖中的 R4 及 R5。此類的情況若發生，代表鄰居節點 Nk 與發出查詢的兩個 sinks 之中的一個距離比感測節點 Nj 還近，而與另一個 sink 的距離則比感測節點 Nj 還遠。

Class 4. 兩個分量都相等：如圖中的 R3。此類的情況若發生，代表鄰居節點 Nk 與發出查詢的兩個 sinks 的距離和感測節點 Nj 都一樣。

Class 5. 一個分量變大，且另一個分量相等：如圖中的 R8 及 R9。此類的情況若發生，代表鄰居節點 Nk 與發出查詢的兩個 sinks 之中的一個距離比感測節點 Nj 還遠，而與另一個 sink 的距離則和感測節點 Nj 一樣。

Class 6. 兩個分量都變大：如圖中的 R2。此類的情況若發生，代表鄰居節點 Nk 與發出查詢的兩個 sinks 的距離都比感測節點 Nj 還遠。

由以上的分類及說明，接下來將介紹選擇資料傳送節點的規則，我們將其稱之為 the Rules of Sensor Selection (RSS)。事實上，每一次選擇傳送節點的基本原則就是：『選取較為接近 sink 的節點』。若以量化的角度看，即是選擇 HCU 值能減少的鄰居節點為下一個轉傳資料的節點。接著，我們定出多個 sinks 時的 RSS 選取通則如下：

- 第一優先選擇：屬於 Class 1 的鄰居節點，亦即滿足圖 4-2 的 R1 者。
- 第二優先選擇：屬於 Class 2 的鄰居節點，亦

即滿足圖 4-2 的 R6 或 R7 者。

- 第三優先選擇：屬於 Class 3 的鄰居節點，亦即滿足圖 4-2 的 R4 或 R5 者。
- 不宜列入選擇：屬於 Class 4、Class 5 及 Class 6 的鄰居節點，亦即滿足圖 4-2 的 R2, R3, R8 及 R9，不宜被選擇。

上述的通則中，屬於 Class 3 的節點之所以仍被列入選擇，主要乃是因其存在部分的分量變小，代表仍有利於將資料往更接近部份 sinks 的地方傳送。

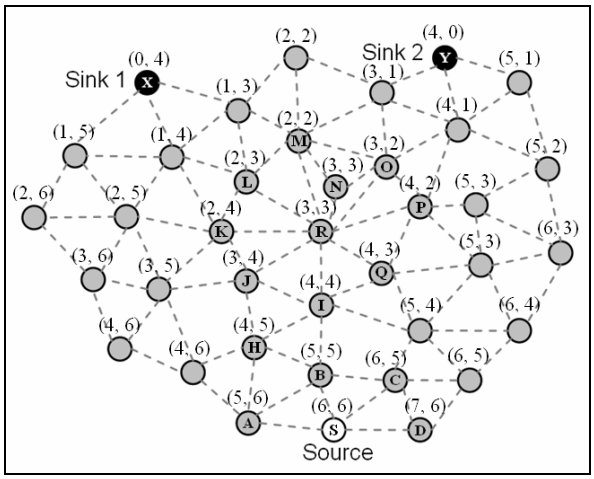


圖 4-3：感測節點 R 與九個鄰居節點

接下來我們以圖 4-3 為例來說明 RSS 的選取規則。在圖 4-3 中，假設由 X (i.e. sink 1)和 Y (i.e. sink 2)發出查詢 Q1，則每一個節點紀錄的 HCVs 資訊如圖 4-3 所示。其中節點 R 的 HCV 為 H(R, Q1)={(3, 3)}，其鄰居節點的 HCVs 分別為 H(I, Q1)={(4, 4)}、H(J, Q1)={(3, 4)}、H(K, Q1)={(2, 4)}、H(L, Q1)={(2, 3)}、H(M, Q1)={(2, 2)}、H(N, Q1)={(3, 3)}、H(O, Q1)={(3, 2)} 、H(P, Q1)={(4, 2)} 和 H(Q, Q1)={(4, 3)}。

我們把以上九個鄰居節點的 HCVs 依圖 4-2 的九種比較情況加以整理可以得到表 4-1。依 RSS 的選取規則，屬於 Class 1 的鄰居節點 M 具有第一優先被選擇權，而第二優先被選擇權則為屬於 Class 2 的 L 及 O 節點，第三優先被選擇權才是屬於 Class 3 的 K 及 P 節點，其餘節點不予選取。

表 4-1：節點 R 與鄰居節點的 HCV 比較表

H(R, Q1)={(3, 3)}, X1 = 3, X2 = 3		
Class 1	R1. X1 > Y1, X2 > Y2	H(M, Q1) = {(2, 2)}
Class 2	R6. X 1 > Y1, X2 = Y2	H(L, Q1) = {(2, 3)}
	R7. X 1 = Y1, X2 > Y2	H(O, Q1) = {(3, 2)}
Class 3	R4. X 1 > Y1, X2 < Y2	H(K, Q1) = {(2, 4)}
	R5. X 1 < Y1, X2 >Y2	H(P, Q1) = {(4, 2)}
Class 4	R3. X 1 = Y1, X2 = Y2	H(N, Q1) = {(3, 3)}
Class 5	R8. X 1 < Y1, X2 = Y2	H(Q, Q1) = {(4, 3)}
	R9. X 1 = Y1, X2 < Y2	H(J, Q1) = {(3, 4)}
Class 6	R2. X1 < Y1, X2 < Y2	H(I, Q1) = {(4, 4)}

4.1.2. ODF 的步驟

本小節描述 ODF 演算法的設計理念，同時也舉例說明採用 ODF 演算法如何去決定出傳送路徑的各節點。基本上，ODF 演算法的主要設計理念是，從 source 節點開始，找尋比 source 節點更為接近 sinks 的鄰居節點為下一個傳送節點。接著，重覆相同的作法，繼續為每一個已被選取的感測節點找尋下一個傳送節點，直到 source 至每一個 sink 的資料傳送路徑都被建立為止。

ODF 演算法的流程如下：

(1). 每個感測節點記錄自己和所有鄰居節點的 HCVs。

(2). 以 source 作為『參考點』。

(3). 為『參考點』選擇鄰居節點中適合傳送資料者：

(3-1). 比較『參考點』與所有鄰居節點的 HCVs 關係是屬於那一 Class。

(3-2). 依 RSS 的優先順序挑選一個或多個鄰居節點。

(3-3). 設定被選取的鄰居節點為新的『參考點』。

(4). 若尚有 sinks 的路徑未被建立，則跳回 (3) 繼續執行；否則，演算法結束。

在整個演算法的流程中，有三個步驟需要加以詳細說明，分述如下。

Step (1)： 每個感測節點記錄自己和所有鄰居節點的 HCVs。

當查詢由 sink 發出後，每個接收到查詢的感

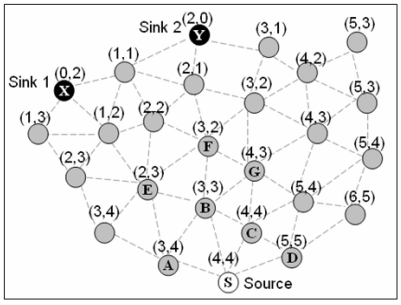
測節點會計算出相對此查詢的 HCU 值，並將其記錄在 HCU table 中，而且也會將鄰居節點的 HCUs 記錄在 NHCU table 中。圖 4-4 的表格是感測節點 S 所記錄的資訊，利用所記錄的 HCUs 可以求得 HCVs，如圖 4-5 所示。

Step (3-1)：比較『參考點』與所有鄰居節點的 HCVs 關係是屬於那一 Class。

在此步驟中，我們將針對感測節點所記錄的 HCVs 資料進行量化的比較，藉以判斷出其鄰居節點是屬於那一個 Class。若延續圖 4-5 的例子，H(A, Q1) = (3, 4) 且 H(S, Q1) = (4, 4)，所以鄰居節點 A 屬於 Class 2。同理可推得，鄰居節點 B, C, D 分別是屬於 Class 1，Class 4，Class 4。

Step (3-2)：依 RSS 的優先順序挑選一個或多個鄰居節點。

經由前一步驟的執行，已經可以求得每一個鄰居節點所屬的 Class，接著便依 RSS 中各 Class 的選擇優先順序，去挑選出最適合當作下一個傳送節點的鄰居節點。接續前面步驟所用的例子，我們可以發現，鄰居節點 B 屬於 Class 1，將是具有被最先挑選的順序(如圖 4-6 所示)。因此，鄰居節點 B 被選為下一個傳送節點；也就是說，節點 S 將把資料送至節點 B。至於選取多個鄰居節點的情況，我們將在後面的章節去說明。



HCU Table

Sid	Qcount
1	4
2	4

NHCU Table

Bid	Sid	Qid	Value
A	1	1	3
B	1	1	3
C	1	1	4
D	1	1	5
A	2	1	4
B	2	1	3
C	2	1	4
D	2	1	5

圖 4-4：感測節點 S 所記錄的 HCUs

H(S, Q1)=(4, 4)

H(A, Q1)=(3, 4)

H(B, Q1)=(3, 3)

H(C, Q1)=(4, 4)

H(D, Q1)=(5, 5)

圖 4-5：節點 S 記錄的 HCVs

值得一提的是，在選擇下一個傳送節點時，有可能發生兩個(含)以上具有相同優先權的鄰居節點，例如，在圖 4-4 的例子中，若感測節點 B 繼續要選取下一個傳送節點時，便會遇到此一問題，因為 HCVs 值為 H(E, Q1)={(2, 3)} 及 H(F, Q1)={(3, 2)}的兩個鄰居節點，都具有所有鄰居節點中最高的優先被選擇權。事實上，此一問題值得更多的討論，我們將它放在下一小節說明。

HCV	Priority
H(A, Q1)=(3, 4)	Priority 2
H(B, Q1)=(3, 3)	Priority 1
H(C, Q1)=(4, 4)	Priority 4
H(D, Q1)=(5, 5)	Priority 4

圖 4-6：利用 RSS 去決定節點的選取順序

4.1.3. ODF 的討論

本小節將討論執行 ODF 演算法時，若遇到相同優先權的鄰居節點將如何選取，以及何時必須選擇多個鄰居節點，並且我們也針對 ODF 進行複雜度的分析。

4.1.3.1. 相同優先權時的選取策略

當有兩個(含)以上的鄰居節點具有相同的優先被選取權時，為了節省電力的消耗，原則上將避免同時選取具有相同優先權的所有鄰居節點。然而，即使是具有相同的優先權(i.e. 屬於相同的 Class)，仍須進一步檢查是否屬於圖 4-2 中相同的比較結果，才能夠決定所應採取的作法。所以，底下我們將分兩種情況去說明所要採取的作法。第一種是具有相同的優先權，且屬於圖 4-2 中相同的比較結果；第二種則是具有相同的優先權，但屬於圖 4-2 中不同的比較結果。

- 相同的比較結果

在此種情況下，同時選擇多個具有相同比較結果的鄰居節點來傳送資料，所達成的果

效是和選擇一個時相同。所以，原則上是僅選取一個鄰居節點當下一個傳送節點。我們考量到所有的鄰居節點不見得會具有相同的剩餘電量，而且每個鄰居節點傳送每筆資料所耗費的電量不見得相同(電量耗費通常與節點間的距離有關)，因此所採用的選取策略是利用下列的計算式 wj 去找出具有最大值的鄰居節點為被選取的節點。之所以找出計算式 wj 最大值的原因是，通常剩餘電量愈大者較適合來負責傳送資料；而感測半徑愈大者，它每傳送一筆資料所耗費的電量就愈大，因此愈不適合被選取。

$$w_j = \frac{\text{剩餘電量}}{\text{感測半徑}} \quad (1)$$

- 不同的比較結果

至於針對同時有多個鄰居節點具有 RSS 中相同的優先權，但是卻是屬於圖 4-2 中不同的比較結果，所採用的策略是同時選取多個鄰居節點為下一步的傳送節點，這也就造成傳送路徑由一條分為多條。之所以採取選擇多個鄰居節點的原因，主要是想讓每一個 sink 在每一步的節點選取時，都能有被選取的節點比『參考點』更為接近 sink。以圖 4-4 為例，當從節點 B 要去選取鄰居節點時，節點 E 及 F 的 HCVs 為 H(E, Q1)={(2, 3)} 及 H(F, Q1)={(3, 2)}，因此這兩個節點在 RSS 中的選取優先順序是相同的，此時我們將同時選擇這兩個鄰居節點。因為針對 sink 1 而言，節點 E 能夠比節點 B 更為接近它；但是，此節點並沒有比節點 B 愈來愈接近 sink 2 (因為兩者 HCVs 的第二個分量皆為 3)。然而，我們卻可以發現節點 F 能夠比節點 B 更為接近 sink 2，但是卻又不能比節點 B 更為接近 sink 1。為了在選取節點時，讓每一個 sink 都能有節點比『參考點』更為接近 sink，就必須同時選取這兩個節點。這樣的策略主要是發展自一個重要的原則：『能夠僅用單一條路徑傳送就儘量保持用一條，否則就從一條路徑變為多條路徑來達到快速有效率的傳送』。在此還有一點必須加以說明：需要選取多少個鄰居節點？作法上是持續選取節點直到滿足『所有的 sinks 至少都有一個被選取的節點是比「參考點」更接近它們』為止。因此，實際上每一步驟被選取的節點數將會小於或等於 sinks 數。如圖 4-4 由節點 S 去選取鄰居節點，僅需選取一個鄰居節點 B 即可。

4.1.3.2. ODF 的複雜度分析

在ODF演算法中，有一個重要的步驟是去比較『參考點』與所有鄰居節點的 HCVs關係是屬於那一Class。而要判斷是屬於那一Class，就必須對HCVs中各分量去作比較。參考圖 4-2 可以發現，當sinks數目是二個時，針對每一個sink而言皆有“ > ”、“ = ”和“ < ”3 種情況，所以比較的規則就有 3*3 = 9 條。一旦sinks數目是三個時，比較的規則便有 3*3*3 = 27 條。可以推算知，當發出查詢的sinks數目為k時，將會產生 3^k個比較規則。若假設每一個感測節點平均有n個鄰居節點，則ODF演算法中為了判斷所有鄰居節點是屬於那一個Class，其計算複雜度便是 $O(n*3^k)$，很明顯地是一個沒有效率的作法。因此，我們設計了第二個演算法。

4.1.3.3. 為何僅比較相同分量

上述的討論皆僅針對相鄰兩節點的 HCVs 中，同一分量的值去進行比較，之所以沒有針對不同分量進行比較，例如將圖 4-4 中節點 S 的 HCV 的第二個分量 4，與鄰居節點 A 的 HCV 的第一個分量 3 作比較，主要乃是因為不同分量的值所代表的是距離不同 sinks 的 HCUs，所以即使將此二個不同分量的值進行比較，其比較的結果並不代表有任何特別的意義存在，而且也不能被用於選取傳送節點的判斷上，所以我們並沒有作 HCVs 中不同分量的比較，而僅比較 HCVs 中相同的分量。

4.2. The Full Cover with Minimum Neighbors (FCMN) Algorithm

由於前面所提的 ODF 演算法隨著發出查詢的 sinks 數增加，其計算複雜度也隨著大幅增加，所以在實際選取節點上是較沒有效率的。故此，我

們提出第二個演算法，the Full Cover with Minimum Neighbors (FCMN) Algorithm。這個演算法主要是依據鄰居節點的 HCVs 中分量值減少的個數，去決定該選取那些節點。

4.2.1 FCMN 的步驟

FCMN 演算法的設計理念和 ODF 演算法類似，主要仍是為傳送路徑逐步地選取節點。然而，FCMN 卻在每一步驟依 HCVs 中分量值減少個數的多寡去從鄰居節點中選取最少的節點數。

FCMN 演算法的流程如下：

(1). 每個感測節點記錄自己和所有鄰居節點的 HCVs。

(2). 以 source 作為『參考點』。

(3). 為『參考點』選擇鄰居節點中適合傳送資料者：

(3-1). 比較『參考點』與鄰居節點 HCVs，計算有多少分量值減少。

(3-2). 依分量值減少個數的多寡，依序選取一個或多個鄰居節點。

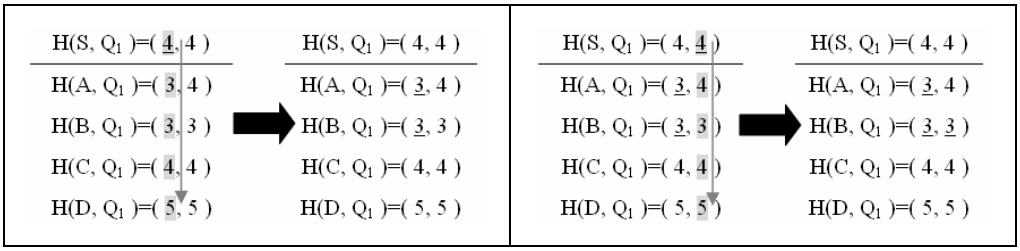
(3-3). 設定被選取的鄰居節點為新的『參考點』。

(4). 若尚有 sinks 的路徑未被建立，則跳回 (3) 繼續執行；否則，演算法結束。

FCMN 和 ODF 演算法主要不同點共有二處，詳細說明如下。

Step (3-1)：比較『參考點』與鄰居節點 HCVs，計算有多少分量值減少。

在此步驟中，我們將『參考點』與其鄰居節點進行 HCVs 的比較，找出 HCV 中有任何分量變小的鄰居節點。我們仍然以圖 4-4 為例，節點 S 與它的四個鄰居節點進行 HCV 的比較，如圖 4-7 所示。在圖所 4-7(a)中，首先進行第一個分量值的比對(灰色網底的部份)，發現鄰居節點 A 及 B 的 HCVs 的第一個分量比節點 S 的小，我們以畫底線作記號，如圖 4-7(b) 所示。接著進行第二個分量的比較，只得到鄰居節點 B 的第二分量值有變小，如圖 4-7(d)所示。因此，總括來說，節點 S 的四個鄰居節點中，只發現 A 及 B 有變小的情形。



(a) (b) (c) (d)

圖 4-7：找出 HCV 中有任何分量變小的鄰居節點

Step (3-2)：依分量值減少個數的多寡，依序選取一個或多個鄰居節點。

針對前一步驟所找出並標記的鄰居節點，此步驟主要是從這些被標記的節點中，決定出最後被選取來傳送資料的節點。我們所採行的原則是：優先選取 HCV 中最多分量減少的鄰居節點。因為每一個分量的減少代表此鄰居節點比『參考點』更接近某一個 sink，所以更多分量的減少，代表著此鄰居節點比『參考點』同時更接近多個 sinks。因此，選取這個鄰居節點，將能同時滿足想要將資料傳送得更接近多個 sinks 的目的。所以，我們將從分量減少最多至分量減少最少的鄰居節點中依序選取節點，而選取動作的終止條件即為，所有的 sinks 都至少有一個被選取的鄰居節點是比『參考點』更接近它們。若仍以圖 4-4 為例，上一步驟被標記 HCV 有分量減少的鄰居節點是 A 與 B，其中分量減少最多的是節點 B，它共有 2 個分量減少，而例子中也剛好只有 2 個 sinks，所以，結果只需選取一個節點 B 就可以了。在後面討論的例子中，我們將會舉出需要選取多個鄰居節點的情況。

4.2.2. FCMN 的討論

接下來我們舉出其他的例子，主要是增加節

點的個數以及將發出查詢的 sink 數目增加至四個。此例中，感測器的佈置採取六角形的擺放，位置不在網路邊緣的節點都有六個鄰居，如圖 4-8 所示，節點內的編號代表感測節點的 ID。

下面的討論主要分為兩個部份，(1) 針對 FCMN 如何能比 ODF 減少計算的複雜度加以說明；(2) 針對同時有多個鄰居節點其 HCVs 分量值減少數目相同時，提出合適的選取方案。

4.2.2.1. FCMN 與 ODF 計算複雜度的比較

在圖 4-8 中，假設目前是以編號 12 的感測節點為『參考點』去選取鄰居節點，若採用FCMN演算法，我們將可以得到圖 4-9。圖 4-9 左邊列出編號 12 的感測節點及它的所有鄰居節點，而圖 4-9 右邊則是將HCV中沒有任何分量減少的鄰居節點畫上直線表示刪除的作用。如圖中所示，三個鄰居 N6、N13 和N20 已被刪除。但是若是採用ODF演算法，就必須對所有的鄰居一一檢查，看它們是屬於那一Class。即使是HCV中分量沒有減少的鄰居節點，仍須進行檢查。而且每個節點檢查所花的成本，將隨著HCV的分量個數呈現指數性成長。假設sink的數目為k，每個感測節點的鄰居節點數目為n，ODF演算法必須對每一個鄰居節點(Bi) 去檢查應屬於那一條比較規則(Ri)，顯示在圖 4-10(a)。其比較規則的數量為m (=3^k) 個，以ODF演算法去從鄰居節點中挑選，則須執行n*3^k次。若採用FCMN演算法，將僅對每一個鄰居節點的HCV中的分量值進行檢查，去挑出有任何分量值變小的鄰居節點，因為必須對每一個分量作檢查，所以共需執行n*k次，如圖 4-10(b)所示。因此可看出FCMN比ODF有效率。

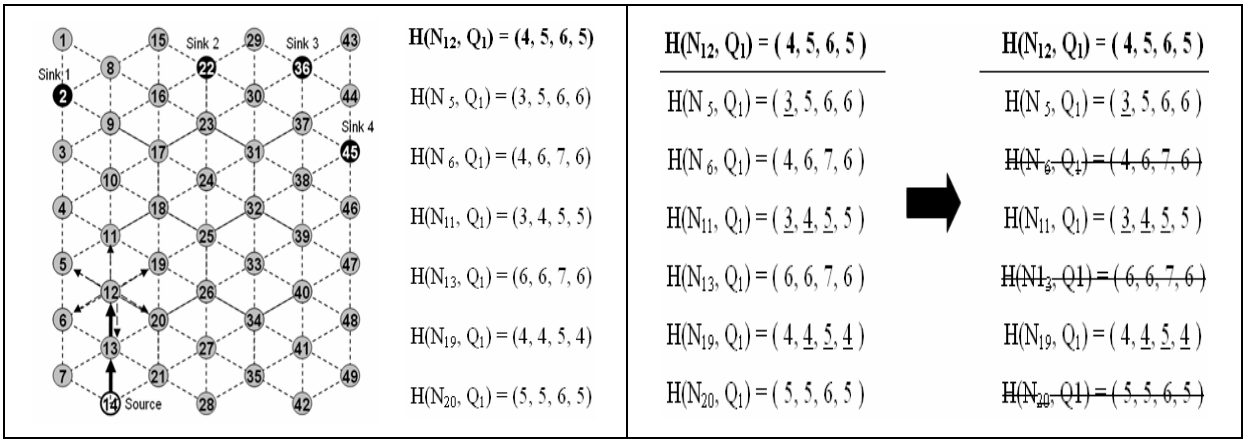
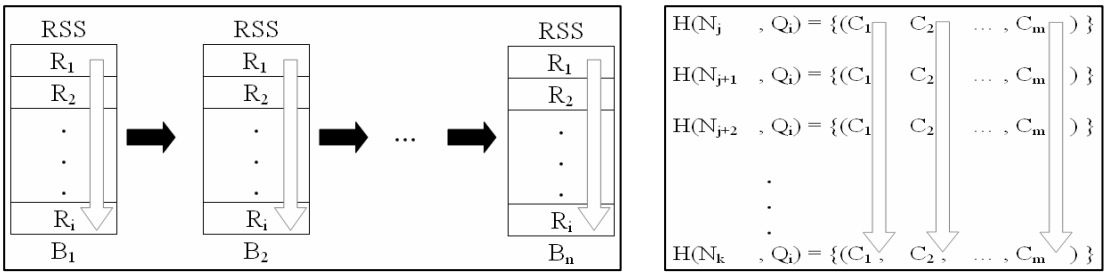


圖 4-8：四個 sinks 發出查詢的範例

圖 4-9：FCMN 演算法選取節點示意圖



(a) (b)

圖 4-10：兩個演算法進行鄰居節點選取的比較方式

4.2.2.2. 分量值減少數目相同時如何選取

在這一小節中，我們將針對同時有多個鄰居節點的 HCVs 中，具有相同數目的分量值減少的情況加以討論。在底下的內容中，主要先根據分量值減少的位置進分類，之後再說明各類的情況如何選取節點。在此假設 HCVs 中有相同數目的分量值減少的鄰居節點為 Nj 及 Nk，令其 HCVs 分別為 H(Nj, Qi) = (C1, …, Ci , …, Ci+j, …Cm)和 H(Nk, Qi) =

(C1, …, Ci , …, Ci+j, …Cm)。為了說明方便，我們使用了一個名詞，cover set，它的定義說明如下。由『參考點』去選取鄰居節點時，主要是選取 HCV 分量值比『參考點』的 HCV 分量值減少的鄰居節點。當某個分量值減少，意謂著此鄰居節點比『參考點』更為接近某個 sink。為了記錄每個鄰居節點比『參考點』更為接近的 sinks，我們將這些 sinks 所構成的集合稱之為此鄰居節點的 cover set。以圖 4-8 為例，若以 N12 為『參考點』，則其鄰居節點 N5, N11, N19 的 cover sets 將分別是 {sink1}, {sink1, sink2, sink3} 及 {sink2, sink3, sink4}。因此，我們可以發現，當兩個鄰居節點的 HCVs 中，分量值減少的數目相同時，也就表示兩個鄰居節點 cover sets 的 size (指 cover set 中 sinks 的個數)是相同的。此處所舉例的 N11 及 N19，它們 cover sets 的 size 都是 3。底下，我們針對 cover sets 的 size 相同的鄰居節點，依其 cover sets 是否有交集分成三種情況去討論：

- Cover sets 完全相同

若鄰居節點 Nj 的分量值減少的位置在 Ci 到 Ci+j，而鄰居節點 Nk 的分量值減少的位置也在 Ci 到 Ci+j，此時兩個鄰居節點的 cover sets 完全相同。如圖 4-11 的例子，矩形框出來的分量位置 C2 和 C3 是兩個鄰居節點相同的分量值減少的部份。

$$H(N_j, Q_i) = (C_1, \boxed{C_2, C_3,} C_4)$$
$$H(N_k, Q_i) = (C_1, \boxed{C_2, C_3,} C_4)$$

圖 4-11：分量值減少的位置完全相同

在此種情況下，選取一個或二個鄰居節點，能夠更為接近的 sinks (i.e., C2 和 C3 所對應的 sinks) 都是相同的。若以系統整體的耗費成本考量，將僅選取一個鄰居節點當下一個傳送節點。當然，若再考量這二個鄰居節點可能具有不相同的剩餘電量，而且每個鄰居節點傳送每筆資料所耗費的電量也可能不相同，我們便可以採用和 ODF 演算法相同的計算式 wj 去找出具有最大值的鄰居節點為被選取的節點。

- Cover sets 完全不相同

若鄰居節點 Nj 的分量值減少的位置在 C1 到 Ci，而鄰居節點 Nk 的分量值減少的位置在 C(i+j) 到 Cm，此時兩個鄰居節點的 cover sets 完全不相同。如圖 4-12 的例子，兩個圓形所涵蓋的分量值減少的位置(C1, C2) 與(C3, C4) 是沒有任何相同的。遇到此類情況所採取的策略是，同時選取這些鄰居節點為資料傳送節點。因為這些鄰居節點的被選取，將會造成同時可以有多個不同的 sinks 都有節點更為接近它們。就 Nj 的被選取而言，可以讓分量 C1 與 C2 所對應的 sinks 有節點更為接近，而 Nk 的被選取，則造成分量 C3 與 C4 所對應的 sinks 有節點更為接近。很明顯地，這兩個節點的被選取都是必要的。

$$H(N_j, Q_i) = (\underline{C_1}, \underline{C_2}, C_3, C_4)$$
$$H(N_k, Q_i) = (C_1, C_2, \underline{C_3}, \underline{C_4})$$

圖 4-12：分量值減少的位置完全不相同

- Cover sets 有部份交集

當鄰居節點 Nj 的分量值減少的位置在 C1 到 C(i+j)，而鄰居節點 Nk 的分量值減少的位置在 Ci 到 Cm，此時兩個鄰居有相交的分量位置在 Ci 到 C(i+j)，亦即它們的 cover sets 有發生部份交集的情況。如圖 4-13 的例子，矩形框出來的分量位置 C2 和 C3 是兩個鄰居節點 cover sets 交集的部份，而圓形所涵蓋的分量值位置 C1 和 C4 則不屬於 cover sets 交集的部份。

$$H(N_j, Q_i) = (\underline{C_1}, \boxed{\underline{C_2}, \underline{C_3},} C_4)$$
$$H(N_k, Q_i) = (C_1, \boxed{\underline{C_2}, \underline{C_3},} \underline{C_4})$$

圖 4-13：分量值減少的位置部份重疊

對於兩個鄰居節點的 cover sets 有部份交集的情況，我們所採取的策略是，優先選取 cover sets 中不屬於交集部份的分量值較大的鄰居節點。在圖 4-9 中，兩個鄰居節點 N11 和 N19 是屬於此情況的例子，我們用圖 4-14 表示。其中它們的第二和第三個分量值是相同且相交的部份，所以我們僅比較它們不屬於交集的部份，也就是 N11 的第一個分量值 3 及 N19 的第四個分量值 4，得到的結果是選擇鄰居節點 N19。

$H(N_{11}, Q_1) = (3, 4, 5, 5)$

$H(N_{19}, Q_1) = (4, 4, 5, 4)$

圖 4-14：分量值減少的位置有交集

分量值大者代表距離 sink 較長，我們便優先選取一個鄰居節點可以更接近它。若不屬於交集的部份，其分量值一樣大，則一樣採用和 ODF 演算法相同的計算式 wj 去找出具有最大值的鄰居節點為被選取的節點。最後，我們以圖 4-8 為例，說明當鄰居節點的 cover sets 有部份交集時，不同的選取方式所產生的影響。延續上面的討論，參考圖 4-9，N12 的鄰居節點 N11 和 N19 發生 cover sets 有部份交集的情況，原則上我們選擇了 N19。因為圖 4-8 的例子中共有四個 sinks，而 N19 的 cover set 只包含三個 sinks，並不包括第一個分量所對應的 sink，所以我們必須再選擇其它的鄰居節點，才能達到每個 sink 都有節點更為接近它們。事實上，演算法的終止條件即是設法找到最少數目的鄰居節點，且它們 cover sets 的聯集，剛好等於所有 sinks 所構成的集合。

$H(N_{12}, Q_1) = (4, 5, 6, 5)$

$H(N_5, Q_1) = (3, 5, 6, 6)$

~~$H(N_6, Q_1) = (4, 6, 7, 6)$~~

$H(N_{11}, Q_1) = (3, 4, 5, 5)$ ⇒ $H(N_5, Q_1) = (3, 5, 6, 6)$

~~$H(N_{13}, Q_1) = (6, 6, 7, 6)$~~ $H(N_{11}, Q_1) = (3, 4, 5, 5)$

$H(N_{19}, Q_1) = (4, 4, 5, 4)$

~~$H(N_{20}, Q_1) = (5, 5, 6, 5)$~~

圖 4-15：針對尚未在 cover set 的 sink 去選取合適節點

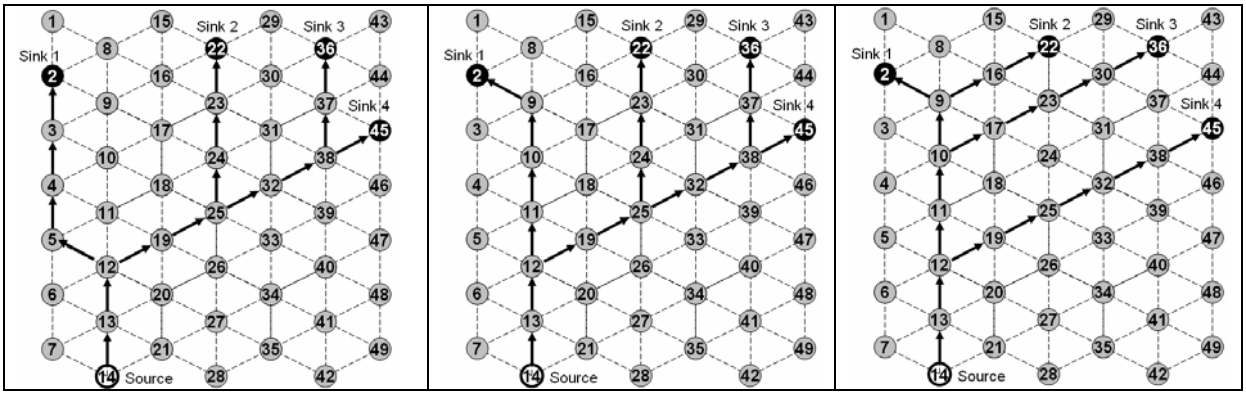


圖 4-16：不同選取方式所產生的路徑

參考圖 4-9，我們得知剩下能被選擇的鄰居節點是 N5 及 N11。因為目前主要是為了選取第一個分量有減少的鄰居節點，所以完全不用理會其他分量是否有減少。在此一原則下，我們得到如圖 4-15 右邊的兩個節點，仍然是 N5 及 N11。接下來，可以用不同的原則去選取。第一種原則是單純選擇僅是第一個分量減少的鄰居節點，也就是選擇 N5。

此原則主要是希望能夠簡單地補足之前選擇的 N19 不足的部份就夠了，也就是說，使得所選擇的 N5 的 cover set 剛好是四個 sinks 和 N19 的 cover set 的差集 (i.e. sink 1)。若依此種原則，圖 4-8 的例子將能夠建立出如圖 4-16 最左邊的圖，其中粗黑線條者為傳送路徑。另一種原則是選擇較多分量減少的鄰居節點，也就是選擇 N11。雖然依這原則被選出來的節點可能和之前已被選取的節點(如 N19)的 cover set 有交集，但這種原則主要的考量是，如果其他被選擇的鄰居節點(如 N19)，若發生沒電或受到外力影響導致故障，產生節點無法運作的狀況，可以利用此節點輔助部份原本需要 N19 傳送資料的 sinks。依此種原則，所建立的傳送路徑如圖 4-16 中間的圖。

最後要討論的是，當兩個鄰居節點 cover sets 的 size 是相同的，但其 cover sets 僅有部份是相交時，我們若不是依前面內容所說的，優先選取 cover

sets 中不屬於交集部份的分量值較大的鄰居節點，而是選取分量值較小的鄰居節點，將會造成什麼樣的影響。仍然以圖 4-8 為例，參考圖 4-9 可以得知是選擇 N11 而不是 N19。因為 N11 的 cover set 只有{sink1, sink2, sink3}，而沒有包含{sink4}，接著，我們必須再從 N5 和 N19 中選取一個節點，而其中第四個分量值有減少的只剩 N19，因此就選取 N19，便得到圖 4-16 最右圖。

若進一步比較圖 4-16 中不同傳送路徑的圖，可以發現最左圖花費 16 個 hops 去達成資料傳送的目的，而中間和最右圖則需花費 17 個 hops 傳送資料。

5. 實驗模擬

本章將針對我們所提出的 FCMN 演算法進行實驗模擬，透過我們自行撰寫之 Java 程式，以評估 FCMN 演算法可以達到多大的效益，此效益主要是評估可以節省多少的能源消耗。尤其是對於許多網路環境的因素，如 sink 的個數、鄰居節點的個數以及 sink 分布的位置，我們都將依序分析 FCMN 演算法在各種不同環境的效能。本章節所有的結果都是經由重覆執行一百次所得到的平均值。之所以沒有評估 ODF 演算法，乃是其複雜度較高，我們暫不在目前的實驗中比較。

表 5-1：實驗的參數及其意義

符號	意義	參數值域	預設值
R	通訊距離	75 公尺	75 公尺
Etrn	傳送資料耗電量	12 mA	12 mA
Erec	接收資料耗電量	1.8 mA	1.8 mA
N	感測器佈置數目	500, 1000, 1500, 2000 個	1000 個
S	發出查詢的 sink 數目	2、4、6、8 個	6 個
B	鄰居節點數目	4、6 個	6 個
A	sink 分佈的範圍	∠90°, ∠180°, ∠270°, ∠360°	∠360°

5.1. 模擬架構

首先介紹實驗進行時所使用的相關參數及其意義，如表 5-1 所示[9]。在各個實驗中，若沒有特別提及的參數，其預設值如下：通訊距離 R = 75m、傳送一筆資料的耗電量 Etrn = 12 mA、接收一筆資料的耗電量 Erec = 1.8 mA、佈置的感測器數目 N = 1000、發出查詢的 sink 個數 S = 6 以及每個感測節點的鄰居節點數目 B = 6。另外，我們也針對了發出查詢的所有 sinks 其分佈範圍的影響進行探討，若將整個區域視為含蓋∠360°的圓形區塊，當分佈的範圍 A=∠360°時，代表發出查詢的所有 sinks 是以 source 為中心，隨機散佈在整個區域，而當分佈的範圍 A=∠180°時，代表發出查詢的所有 sinks 只能以 source 為中心，隨機散佈在半個圓形區塊，其餘角度則以此類推。

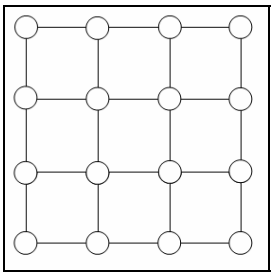


圖 5-1：棋盤式網路圖

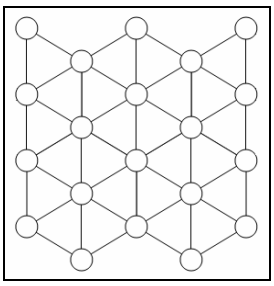


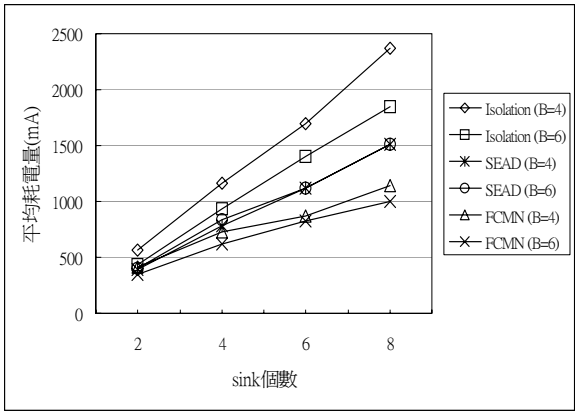
圖 5-2：蜂巢式網路圖

關於感測網路中節點的佈置方式，我們採取規則性的佈置作法，分別是棋盤式佈置的網路圖和蜂巢式佈置的網路圖。棋盤式的佈置方式，每一個節點可以直接和上、下、左、右四個方位的節點通訊，如圖 5-1，所以鄰居數目為四個。而在蜂巢式的佈置方式中，每一個節點將可以直接和六個鄰居節點通訊，如圖 5-2 所示。針對兩種佈置方式，本研究都將進行實驗評估。在後面的實驗評估中，我們將本論文所提出的 FCMN 演算法與其他兩個方法，Isolation 與 SEAD，進行比較。此處的 Isolation 方法指的是，source 與每一個 sink 各別建立一條獨立的資料傳送路徑，而 SEAD 方法則是前面相關研究章節中的[5]所提出的。

5.2. Sink 個數對耗電量的影響

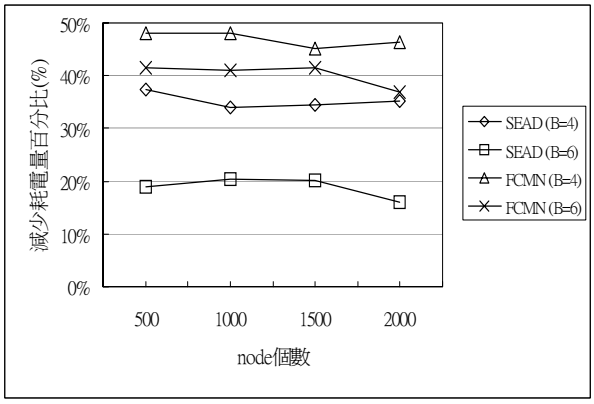
首先，第一個實驗結果顯示在圖 5-3，呈現出在不同數目的鄰居節點下，三種方法的平均耗電量。可以明顯推知，FCMN 和 SEAD 的耗電量都比 Isolation 方法低，所降低的耗電量百分比顯示在圖 5-4。可以看到 FCMN 在鄰居節點數為 4，且系

統的總節點數為 500 時，可以比 Isolation 方法降低耗電量達 48%，而 FCMN 較差的情況是發生在鄰居節點為 6，且系統的總節點數為 2000 時，但也比 Isolation 方法降低了 37% 的耗電量，由此可以確認我們所提的 FCMN 方法的優越性。其中 SEAD 比 Isolation 減少的耗電量不如 FCMN，主要原因是 SEAD 建立的路徑可能不是最短路徑；而且在 SEAD 方法中，也可能因為 access node 距離 source 較近，所以不與 gate replica 建立路徑，造成損失合併路徑的機會而無法減少耗電量。



(N = 1000, A = 360)

圖 5-3：不同數目的鄰居節點對耗電量的影響



(S = 6, A = 360)

圖 5-4：與 Isolation 方法相比減少的耗電量百分比

另外，由圖 5-3 中也發現在發出查詢的 sinks 數目愈多時，FCMN 及 SEAD 比 Isolation 減少的耗電量愈大。由此可知，sinks 數目愈大方，產生的路徑合併效益也就愈大。

6. 結論與未來展望

本論文中，我們介紹了『各類 Sensor 的個數』、『查詢資料的相關性』及『查詢的函數』三項因素對資料傳輸上的影響，並針對 Multiple sinks 發出 Same Query type 向 One source 要相同資料的情況提出路徑合併的方法，本論文中提出兩個解決方法，主要是利用逐步比較『參考點』與鄰居節點間，何者距離 sinks 最近的方法，找出合適的鄰居節點來建立資料傳送路徑，而其中 FCMN 更是具有低計算複雜度，最後透過實驗分析，驗證我們提出的 FCMN 所建立的路徑，而且加入許多網路環境改變的因素，如 sink 的個數、鄰居節點的個數以及 sink 分布的位置，確實 FCMN 能夠以較少的耗電量完成資料傳送，且比過去學者所提的方法更為有效率，而且我們除出的兩個方法都有一個特性，即 source 到每一個 sink 之間的路徑是最短路徑。

參考文獻

[1] B. J. Bonfils and P. Bonnet, “ Adaptive and Decentralized Operator Placement for In-Network Query Processing, ” *In Proceedings of the 2nd International Workshop on Information Processing in Sensor Networks(IPSN'03)*, Palo Alto, CA, pp.47-62, Apr. 2003.

[2] C. Intanagonwiwat, D. Estrin, R. Govindan, and J. Heidemann, “ Impact of Network Density on Data Aggregation in Wireless Sensor Networks, ” *In Proceedings of the 22nd International Conference on Distribut- ed Computing Systems (ICDCS)*, Vienna, Austria, pp.457-458, Jul. 2002.

[3] V. Chowdhary and H. Gupta, “ Communication-Efficient Implement- ation of Join in Sensor Networks, ” *In Proceedings of the International Conference on Database Systems for Advanced Applications (DASFAA'05)*, Beijing, China, pp.447-460, Apr. 2005.

[4] G. Jiang and G. Cybenko, “ Query Routing Optimization in Sensor Communication Networks, ” *In Proceedings of the 4lst IEEE Conference on Decision and Control*, Las Vegas, USA, pp.1999-2005, Dec. 2002.

[5] H. S. Kim, T. F. Abdelzaher, and W. H. Kwon, “ Minimum-energy asynchrono- us dissemination to mobile sinks in wireless sensor networks, ” *ACM Conference on Embedded Networked Sensor Systems (SenSys 2003)*, Los Angeles, USA, pp.193-204, Nov. 2003.

[6] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, “ TAG: a Tiny AGgregation service for ad-hoc sensor networks, ” *In Proceedings of the 5th symposium on Operating systems design and implementation (OSDI '02)*, Boston MA, pp.131-146, Dec. 2002.

[7] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, “ The Design of an Acquisitional Query Processor For Sensor Networks, ” *ACM SIGNOD*, San Diego, CA, pp.491-502, Jun. 2003.

[8] S. Madden, R. Szewczyk, M. J. Franklin and D. Culler, “Supporting Aggregate Queries Over Ad-Hoc Wireless Sensor Network,” *In Proceedings of the 4th IEEE Workshop on Mobile Computing Systems and Applications*, Callicoon, NY, USA, pp.49-58, Jun. 2002.

[9] Y. Yao and J. Gehrke, “Query Processing for Sensor Networks,” *In Proceedings of the 1st Biennial Conference on Innovation Data System Reach (CIDR)*, Asilomar, CA, USA, pp.21-32, Jan. 2003.