車載網路之叢集演算法

Clustering Algorithm in VANETs

林易蓁　　高志孝　　羅壽之

東華大學資訊工程學系

Email: sclo@mail.ndhu.edu.tw

摘要—在這篇論文中，我們提出一個以中心位置與行動節點移動性為基底的叢集演算法。優點在於可以使得叢集首與叢集成員之間的平均距離縮短。我們的叢集演算法能夠建立穩定的叢集，減少叢集重建的次數、延長每個行動節點在叢集內的時間與增加叢集成員的數量（可進一步提高資料共享節點的可獲性）。模擬實驗的結果顯示出我們所提出的叢集演算法優於其他三個常被引用的方式：LID、LCC 與 MOBIC。

關鍵詞—車載網路, 叢集化, 叢集首, 中心位置, 移動性

Abstract—This paper presents a new center-position and mobility clustering algorithm (CPM). The strength of our proposed algorithm is to shorten the average distance between cluster heads and cluster members. The proposed clustering algorithm is intended to create stable clusters by reducing the number of re-clustering, prolonging cluster lifetime, and increasing the number of cluster members (which can increase the node availability for efficiently sharing interested information). Simulation results reveal that the proposed clustering algorithm outperforms the other three widely used clustering algorithms which are the Lowest-ID (LID) clustering algorithm, Least Cluster-Changed (LCC) clustering algorithm, and MOBIC.

*Keywords—*Vehicular Ad Hoc Networks, Clustering, Cluster Head, Center-Position, Mobility

一、 簡介

(一) 車載網路下的叢集技術

叢集化的結構已經在過去的研究中用在不同型態的網路，像是蜂巢式網路、無線感測網路(Wireless Sensor Network, WSN)與行動隨意網路(Mobile Ad Hoc Network, MANET)。車載網路 (Vehicular Ad Hoc Network, VANET) 是MANET的子集合，叢集化在MANET也被證實有一定的效能貢獻[1]，我們的研究主要是將MANET上叢集化的技術延伸到車載網路中。

車載網路有別於傳統的MANET，它擁有下列特性：(1)車輛擁有充足的電力，(2)駕駛者可藉由車輛上的裝置像是無線通訊、電子地圖和導航系統等，得到許多外部的資訊（例如車輛本身的位置與路況等），(3)網路拓樸隨著車輛速度與位移的改變而有所變化，(4)路側單元感測系統的存在。

車輛在車載網路下的頻寬限制會因為無線傳輸品質改變而受到影響，藉由階層式叢集方式可以改善通訊的品質，減少行動節點各自的負擔。叢集結構可以把一群車輛（以下稱為行動節點）視為叢集，並且在叢集內部選出一個適合的行動節點負責資料傳輸的工作，同時也是叢集之間的中繼點，負責與其他叢集的通訊。

為了配合車載網路環境的動態特性，以行動節點為基礎的叢集，必須要週期性更新網路拓樸，還有車輛的移動狀態。同時，叢集化的過程也必須避免太多叢集建立的花費。行動節點的移動性會使得網路拓樸變動，進而導致叢集的重新建立與解散。

由於車載網路變動性較大，叢集的重建與叢集首的角色改變是不可避免的。因此，叢集演算法除了叢集的建立之外，應該設計有效管理叢集的機制。

(二) 研究動機

過去 MANET 的研究中，我們發現根據叢集應用的不同而設計的方式也不相同。由於我們的叢集演算法是根據車載網路特性為考量，主要應用在城市街道。所以我們認為有兩個考慮應該要被強調。第一點：由於城市街道，行動節點的密集程度高，所以我們提出以中心位置為叢集首的概念，不但可以使得叢集首與叢集成員之間的距離縮短，還可以使得路由路徑相對的不易斷裂、穩定通訊的品質、增加資料散播的速度與發揮資料分享的優勢。第二點：我們希望在移動性相當的行動節點中，找出相較其他鄰居中移動性最穩定的行動節點，此行動節點即為最適合的叢集首候選者。

二、 相關研究

(一) 叢集與角色分配

叢集結構就是將網路上的行動節點分為不同的虛擬群組，利用行動節點不同的行為與規則去配置角色，而這些所謂的規則就稱為叢集演算法。

叢集中行動節點依照不同的性質與功能，可區分為幾種角色，一般分為叢集首（Cluster Head, CH）、叢集通信閘（Cluster Gateway, CG）、叢集成員（Cluster Member, CM）。叢集首是每個叢集內的掌管者，負責叢集內部傳輸的安排、資料的傳遞與內部的溝通等工作。叢集通信閘，有時也被稱為邊界節點，是非叢集首節點的集合，並且擁有叢集間的連結。所以它負責與鄰居通訊與叢集之間資料的傳遞。叢集成員，有時也會被稱為一般行動節點，它不屬於叢集首行動節點的集合，也沒有任何叢集間的連結。叢集結構如圖 1、叢集結構所示。

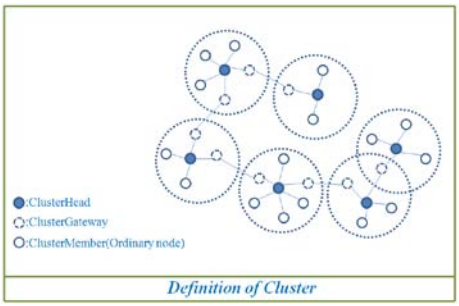

圖 1、叢集結構

(二) 叢集的重要性

傳統行動節點間沒有組織性的交換資料，這樣的方式，隨著行動節點的移動性，伴隨著網路拓樸的快速變化，而使得網路的規模性差且效率低。所以，階層式架構對於 MANET 有很大的優勢，其主要的概念是將網路劃分成許多小型網路。因此，叢集化的觀念對於管理網路拓樸者是很有效率的，並且可提升資源的再利用度進而增加系統的處理速度、有利於網路上資料傳輸的路由建立或使得原本任意配置的行動節點更好管理且更穩定。

(三) 相關研究

目前在 MANET 上的叢集演算法已經被提出很多，隨著車載網路的迅速發展，也有少部分車載網路的叢集演算法。在過去的研究中，我們先把叢集演算法分成兩部分詳細的討論：第一部分著重在叢集建立階段，第二部分著重在叢集維護階段。叢集建立階段主要為叢集初始化的建立，而叢集維護階段在於降低動態環境下當網路拓樸改變時所帶來的影響，包括 CH 的反覆重選與否、叢集數目上升與下降、CH 的存活時間的長短。

然而，現在大部分的演算法都把重心放在叢集建立上，如何利用不同的規則或是量測值來選取最佳的 CH。在叢集維護大部分都是以一些情況來討論，只有少部分的演算法比較有規模的描述叢集維護的程序。

- 叢集建立階段

首先介紹最原始的兩個叢集演算法：第一個是LID（Lowest Identification Clustering）[2]，其概念如同LCA[3]，是以具有最小ID編號的行動節點選為CH。LID的優點就是容易使用ID編號去選出CH，只需要兩次比較就能得到結果。但是缺點是網路拓樸分割後的叢集沒有規則，而且當網路拓樸改變時，會使得叢集不穩定，所以就非常容易使得叢集個數變多。

第二個是最大連結度為基礎的叢集演算法HCC(High Connectivity Clustering)[2]，以具有最多連結度的行動節點選為CH。當連結度相同時，再改由ID小的來決定CH。HCC優點在於當CH負責的行動節點愈多，相對的叢集個數就會減少。而叢集個數減少，相對的就可以使整個網路資訊儲存的維護量降低。不過，當網路拓撲有所改變時，當時的時間與下一秒的時間所負責的叢集成員個數可能有巨大的差異，所以方法的穩定性比LID差。

由於LID或HCC所產生的叢集數量可能會太多，因而造成網路不穩定，所以後來K. Xu等作者提出RCC（Random Competition based Clustering）[4]，以競爭的概念去選定CH，它們修正前面兩個演算法穩定性的問題。

- 叢集維護階段

C. C. Chiang等作者提出了叢集數改變最少的演算法（Least Cluster head Change Clustering, LCC）[5]，解決LID或HCC演算法所產生的叢集穩定性不佳。作者們認為必須要有一個CH重選的機制來維持CH的數量，否則CH數量一直上升會使得叢集的數目上升，進而造成CH的負擔過大。主要改良的叢集維護階段為：(1)當CM移動到新的叢集範圍內時，CH並不需要改變，只有CM需要改變。(2)當一個CM移出它所屬叢集範圍外時，並且無法加入至別的叢集時，則它將形成一個新的叢集，並且變成一個新的CH。(3)當CH從原本的叢集移入新的叢集時，CH將挑戰新的叢集的CH地位，CH或新的叢集的CH將有一方會退休成為CM。其中挑戰方式可依據LID或HCC任一個叢集演算法。(4)當某CH離開後，其成員將依據LID或HCC叢集演算法重新加入其他叢集，或形成新的叢集。

以移動性相關作法裡面，其中MOBIC（Mobility Metrics Clustering）[6]為了改善LID與HCC的效能，考慮了行動節點的相對移動性。這個叢集演算法，其選擇CH的方式是利用兩個度量，相對移動性度量（Relative Mobility metric）還有區域匯集的移動資訊（Aggregate local mobility value）做為CH決策評估因子。如下列所示：

$$M_Y^{rel}(X) = 10\log_{10}\frac{R_x P_{r\,X\to Y}^{new}}{R_x P_{r\,X\to Y}^{old}} \qquad (1)$$

$$M_Y = \mathrm{var}_0\left\{M_Y^{rel}(X_j)\right\}_{j=1}^{m} = E\left[(M_Y^{rel})^2\right] \qquad (2)$$

其中，R_xP_r是從接收行動節點偵測的訊號，也代表傳送行動節點與接收節點之間兩兩的距離。M_Y是計算Y跟其鄰居的移動性，其中X_j表示Y的鄰居，M_Y越小代表Y對於其鄰居的相對移動性較小且越有機會變成CH。

每個行動節點經過訊息的交換後，算出(1)和(2)，有最小區域匯集的移動資訊的行動節點即成為CH。在叢集維護階段，是以LCC演算法的規則來運作。比較不一樣的是，當兩個CH交會的時候，先等待一段時間（Cluster-Contention-Interval, CCI）。若CCI超過一個臨界值後，相對移動度量小的行動節點為CH；否則兩個CHs的狀態不會變。

此方法雖然不用GPS的額外裝置，也不需要行動節點的速度距離等資訊的優點，但其缺點是MOBIC只利用相對移動性去計算行動節點的移動資訊。所以如果從一個移動很快速的行動節點收到的移動資訊，去計算移動後所得的值不會很精準，造成錯誤的估計。

接下來我們說明一些在車載網路環境下的

叢集演算法。

直接傳遞演算法 DPP（Directional Propagation Protocol）[7]，它引用MOBIC[6]所提及之方式來選擇CH。DPP改進MOBIC的部份是一個叢集內同時可以擁有兩個 CH，一個為header，另一個則為trailer。

CBMAC（Cluster-Based Medium Access Control Protocol）[8]中，行動節點分為四個狀態 :Undecided，Member，Gateway 與 Cluster head，一開始所有行動節點都屬於Undecided；其叢集維護階段，當兩個CH交會時，必須靠著下面的方程式來選出一個CH，另外原本的叢集則合併到選定的叢集裡。

$$W_v = w_1 \cdot \Delta_v + w_2 \cdot D_v + w_3 \cdot M_v, \text{where } \{w_1, w_2, w_3\} \in [0,1] \quad (3)$$

其中，$\triangle_V$為行動節點V的連結度，D_V為與每個1-hop鄰居距離的平均值，M_V為與每個1-hop鄰居相對速度的平均值。選出來值最小者做為CH。

(四) 演算法分類

我們將叢集演算法在選叢集首時，所依照的評估因子分成下列幾類，如圖 2 所示：

- ID-Based：

每個行動節點都有一個唯一性的識別碼，此類演算法在 CH 選定的過程依照識別碼來選定，通常選擇最小或最大者當選 CH。

- Degree-Based：

此類演算法在 CH 選定的過程，是以節點的連結度，即行動節點擁有的 1-hop 鄰居的個數，作為評量的基礎。

- Competition-Based：

此類演算法，利用競爭的方式，先競爭到的行動節點即為贏家，贏家就得到當 CH 的權利。

- Mobility Based：

此類演算法，通常是根據行動節點的行為（例如： 行動節點的速度、距離、或是移動的方向等）做為 CH 的選定條件。

- Multiple-metric Based：

此類演算法通常以兩個或以上的評估因子配上不同的權重值，而不是值最小或是最大的就可以當 CH。

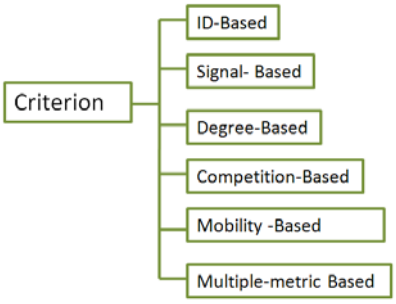

圖 2、叢集演算法分類

三、 叢集演算法

在進入此篇論文的重點之前，我們有兩點假設：(1)每個行動節點都攜帶 GPS，(2)每個行動節點都擁有唯一 ID。

(一) 叢集首選取方式

在我們的叢集演算法中，叢集首選定機制是以中心位置為叢集首的概念，以及在移動性相當的行動節點中，找出相較其他鄰居中移動性最穩定的行動節點。叢集建立階段是以 *RPM* (Relative Position and Mobility Metrics)做為比較條件，其值最小的行動節點即為 CH，步驟如下：

Step 1：假設在時間 *t* 的時候，行動節點 1 與鄰近 *m*-1 個鄰居的位置以下面的式子表示：

$$P_1 = (x_1(t), y_1(t))$$
$$P_2 = (x_2(t), y_2(t))$$
$$P_k = (x_k(t), y_k(t))$$
$$P_m = (x_m(t), y_m(t))$$

Step 2：算出行動節點自己與鄰居節點的虛擬中心點（假設為行動節點 *c*）位置，P_C 以下面的式子來表示：

$$P_c = (x_c(t), y_c(t)) = \left(\frac{\sum_{i=1}^{m} x_i(t)}{m}, \frac{\sum_{i=1}^{m} y_i(t)}{m} \right)$$

求出虛擬中心節點位置之後，計算自己與虛擬

中心節點 c 的距離。任意行動節點 i 與虛擬中心節點 c 的距離如下面的式子表示：

$$Dist_{ic} = |P_i - P_c| = \sqrt{(x_i(t)-x_c(t))^2 + (y_i(t)-y_c(t))^2}$$

Step 3：此外，我們還必須考慮行動節點本身的速度與所有鄰居節點速度的差異。任意行動節點 i 的速度用下列公式計算：

$$V_i = \sqrt{\left((x_i(t)-x_i(t-1))^2 + (y_i(t)-y_i(t-1))^2\right)}$$

相同公式計算所有鄰居節點（包含本身）的速度值，並經過數值大小排序後以 $\{V_1, V_2, \cdots, V_i, \cdots, V_m\}$ 表示。接著找出這列數值的中位數 V_c，如下列公式所示：

$$V_c = median \quad \{ V_1, V_2, \cdots, V_i, \cdots, V_m \}$$

接著計算本身速度與 V_c 的差異如下所示：

$$Rel\_Speed_{ic} = |V_i - V_c|$$

Step 4：我們以調整權重的方法來當作叢集首選定的決策評估因子，α 為比重加權值。正規化之後，所得的 $RPM \in [0,1]$，其公式為下列所示：

$$RPM_i = \alpha \cdot \frac{Dist_{ic}}{Max\{Dist_{jc},\ j \in 1 \sim m\}} + (1-\alpha) \cdot \frac{Rel\_Speed_{ic}}{Max\{Rel\_Speed_{jc},\ j \in 1 \sim m\}}$$

求出 RPM_i 之後，記錄於HELLO訊息中通知所有鄰居節點，同時並收集鄰居節點的*RPM*值。如果本身的*RPM*值在所有鄰居節點中數值最小，則宣告自己為叢集首。

(二) 叢集身份說明

首先，我們先以圖3說明在叢集演算法中會使用到的節點身份：

- 閒置節點（Undecided Node, UN）:當行動節點在叢集建立階段的初始化時，或者，當行動節點都尚未屬於任何叢集，並且也沒有任何的身份時，屬於此種身份狀態。此行動節點在網路拓樸下是沒有傳遞資料的功能，如行動節點 7、14、15 與 16。

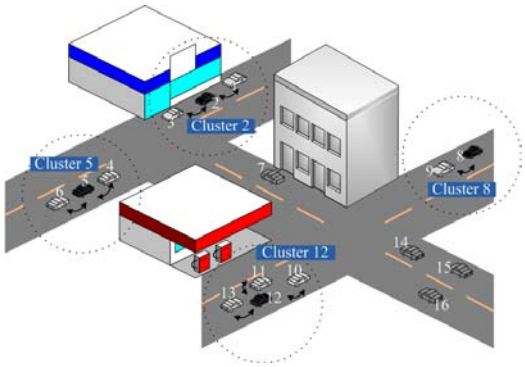

圖 3 叢集首演算法概念圖

- 叢集首節點（Cluster Head Node, CH）：當行動節點在叢集建立階段的叢集首選舉過程完成後，所當選的節點屬於這種身份狀態。此行動節點的工作為負責管理其叢集成員，收集或分享資訊給其叢集成員；其與叢集成員對應為一對多的情形，並且負責與其他叢集首通訊。如行動節點 2、5、8 與 12。
- 叢集成員節點(Cluster Member Node, CM)：當行動節點在叢集建立階段的叢集首選舉過程完成後，除了叢集首之外，並且有加入叢集的行動節點，屬於這種身份狀態。此行動節點的工作為負責向叢集首查詢或取得資訊；其與叢集首的對應為多對一的情形，即每個叢集成員節點最多只能有一個叢集首。如行動節點 1、3、4、6、9、10、11 與 13。

介紹完三種不同的行動節點之後，我們將介紹在叢集演算法裡使用的表示符號：

- *BI*（Broadcast Interval）：每個行動節點廣播 HELLO 訊息的週期時間間隔。
- *CI*（Contention Interval）：兩個 CH 競爭時通訊的時間間隔（設為 *BI* 的倍數）。
- *TI*（Timeout Interval）：每個行動節點清空 neighbor table 中過期(設為 *BI* 的倍數）資訊的週期時間間隔。
- *I*（Node Identification）：唯一性的 ID 編號。

- $|UN_i|$（Number of Undecided Nodes）：行動節點 i 的 1-hop 鄰居裡面 UN 的個數。
- U（Threshold of Undecided Nodes）：UN 進入叢集選定的最低下限。
- CH_i（CH Node i）：行動節點 i 的身份狀態是 CH。
- C_i （CH Identification）：行動節點 i 的 CH 的 ID 編號，也是叢集的 ID 編號。
- CD（Contention Distance）：CH 進入叢集競爭的限制（為一半的傳輸半徑）。
- $|CM_i|$（Number of CMs for Node i）：行動節點 i 本身為 CH，其所擁有的 CM 個數。
- JOIN_INVITED：叢集首選舉的程序後，選定出來的 CH 所發出的邀請訊息。
- JOIN_REPLY：叢集首選舉的程序後，收到選定出來的 CH 所發出的邀請訊息後，回覆同向方向的 CH，請求加入叢集。
- CH_RESIGN：CH 叢集競爭的程序後，所競爭失敗的 CH 所發出的辭職訊息，用來告知原本所擁有的 CM，並且解散其叢集。

(三) 運作方式說明

我們的叢集演算法主要的設計目標在於有效的資料分享及散佈。而以叢集的角度來看，我們的演算法設計重點著重於：(1)使用接近中心點位置的行動節點來當作 CH，縮短與其他行動節點的位置之優點，用以延長 CH 的存活時間。(2)避免無效叢集（指只有 CH 一個行動節點的叢集或是 CH 與一個以下叢集成員的叢集）的存在，以減少叢集的數量。(3)減少叢集的競爭與解散，減少利用度低的行動節點。(4)避免無謂的叢集重選，以減少不必要的花費成本。(5)CH 選定的限制，加強有效叢集的存在，以增加叢集成員的數量。

我們的叢集演算法分成兩個部份，叢集建立階段以及叢集維護階段。我們的演算法是以分散式的方式運作，每個行動節點都需要週期性維護一些區域性資訊。在叢集建立階段，步驟如下：

Step 1：在初始化階段，因為每個行動節點 i 不屬於任何的叢集，所以皆為 UN。

Step 2：每個行動節點 i 隔 BI 時間就週期性的廣播 HELLO 訊息給其 1-hop 鄰居，告知自己的存在；同時地，也交換自己與鄰居的資訊。

Step 3：當行動節點 i 收到廣播訊息之後，則進入選舉判斷的程序。判斷屬於 UN 身份狀態的鄰居個數是否超過臨界值。如果行動節點 i 收到 $|UN_i|$ 個以上同向方向 UN 的廣播訊息，即 $|UN_i| \geq u$ 之後，才開始進入叢集首選舉的程序。每個行動節點開始計算 RPM 值並收集鄰居節點的 RPM 值，如果自己的 RPM 在鄰居中是最小值，則宣告自己為 CH。若有兩個以上的行動節點有相同的 RPM 值，則比較 $Dist_{ic}$ 第一個為最高優先權，Rel_Speed_{ic} 後者為次高優先權者，而小的為 CH；若同時相等，則 ID 小的為 CH。

Step 4：選定之後出來的 CH，設定 C_i 為自己的 ID ，設定完後並且廣播 JOIN_INVITED 訊息給其 1-hop 鄰居。

Step 5：當新加入網路拓樸的 UN 或是在叢集首選舉的程序完畢後的 UN，收到 JOIN_INVITED 訊息後，先檢查是否為同向方向的 JOIN_INVITED 訊息。如果是的話，則回復 JOIN_REPLY 訊息給 CH。接著，變換身份狀態到 CM，然後設定 C_i 為 JOIN_INVITED 訊息內的 ID。

在叢集維護的階段，我們分成三個身份狀態 UN、CH 與 CM 來說明。不同的身份狀態進入不同的狀態，下面將以圖 4 進行說明，其說明如下：

- 閒置節點(UN)：

(1)當收到同向方向的 JOIN_INVITED 訊息，則設定收到的訊息來源 ID 為自己的 C_i，並且加入

其叢集。且轉換身份狀態到 CM。

(2) 當鄰近區域中沒有收到同向方向的 JOIN_INVITED 訊息，則進入選舉判斷的程序。

- 叢集首節點(CH)：

(1)假設有兩個 CH_i 與 CH_j 通訊超過 *CI* 之後，則進入 CH 叢集競爭的程序。進入之後，先判斷是否為同向方向，如果是同向方向的話，再判斷兩 CH 的距離是否小於 *CD*。如果小於 *CD* 內，則判斷叢集的大小，若 $|CM_i| \geq |CM_j|$，則 CH_j 放棄自己的身份狀態狀態，並且廣播 CH_RESIGN 的訊息告知原本隸屬於自己的 CM，然後加入 C_i 並且變換身份到 CM。

(2)當鄰近區域中沒有收到隸屬於自己的 CM 廣播訊息時，則放棄自己叢集首的身份狀態，並且轉換身份狀態到 UN。

- 叢集成員節點(CM)：

(1)當超過 *TI* 時間之後，CM 聽不到自己 CH 所發出的 HELLO 訊息時，則判定 CH 離開，並且轉換自己的身份狀態為 UN。

(2)收到 CH_RESIGN 訊息後，轉換自己的身份狀態為 UN。

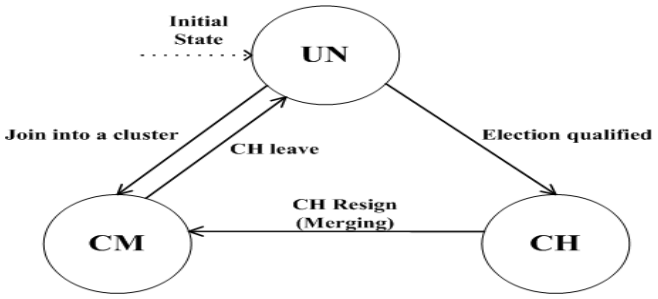

圖 4、身份狀態轉移

四、效能比較

(一) 移動模型

我們所提出的叢集演算法的模擬是採用 VanetMobiSim[9]，其為延伸 CanuMobiSim[10] 的移動模型，它主要包含使用地理資料檔案為主的資料結構 GDF(Geographical Data File compliant data structures)，還有車輛導向的移動模式。獲得網路拓樸資料的方式有四種：(a)允許使用者自行定義道路地圖，(b)載入 GDF，(c)載入 TIGER 地圖[11]，(d)隨機產生 Voronoi graph。另外，還可以定義車道的行進方向、不同的速度限制分級以及號誌燈的開關。

(二) 實驗參數

我們採用道路密集程度較高的華盛頓哥倫比亞特區，地圖大小為 1500m×1500m，地圖來源是從 TIGER 資料庫中取得。在 NS2[12]模擬中產生 200 個行動節點，每個行動節點的無線傳輸範圍 *TR* (Transmission Range)為 100m、150m、200m、250m。每個行動節點廣播 HELLO 訊息的週期 *BI* 設為 0.5 秒，兩個 CH 通訊的時間間隔 *CI* 設為 4×*BI*。每個 CH 進入叢集競爭的條件 *CD* 設為 *TR/2*，每個行動節點清空 neighbor table 過時資料的週期時間間隔 *TI* 設為 2×*BI*。總共實驗模擬的時間為 300 秒。最後，在實驗結果中，跑了十次的模擬實驗，每一次選取隨機產生出的 10 張在華盛頓哥倫比亞特區上不同的車輛移動模型，再將這十次的結果取平均數，以顯示穩定的效能圖。

下面介紹實驗中用來比較效能的評估標準：

- 叢集個數 (Average number of clusters)：計算方式是每 10 秒觀察系統內每一個叢集的叢集個數並進行加總。在叢集演算法中，我們不希望叢集建立後，產生太多的叢集，會使得無效叢集變多，減少叢集化所帶來的系統效能。
- 叢集成員的個數(Average cluster size in numbers of cluster members)：指一個叢集首所掌管的 CMs，計算方式是每 10 秒觀察系統內每一個叢集的成員個數並進行加總，將其除以全部叢集的個數，最後將全部觀察到的數據除以觀察次數。在資料分享的應用方面，我們希望一個叢集內的 CMs 不可以太少，以達到有效率的資料散播。

- 叢集的存活時間(Average Cluster lifetime)：指在實驗模擬時間之內，平均的叢集存活時間。計算方式是將每一個叢集的存活時間進行加總，最後將其除以全部叢集的個數，即 CH 存活的時間。我們希望行動節點在叢集內的時間越長越好，以提升行動節點在網路拓樸下的利用性。
- 節點閒置時間(Average idle time)：指在實驗模擬時間之內，每一個行動節點不在叢集內的時間。計算方式是將每一個行動節點處於閒置狀況的時間進行加總，最後除以實驗模擬時間之內的全部行動節點個數。我們不希望行動節點不在叢集內的時間太長，反而希望每個行動節點能有其功能，讓網路拓樸變得更有系統，以發揮出叢集技術的最大優點。
- 叢集成員駐留時間(Average resident time)：指在實驗模擬時間之內，平均的成員駐留時間。計算方式是將每一個叢集成員的駐留時間進行加總，最後將其除以全部行動節點於叢集的駐留次數。

(三) 實驗結果

圖 5 中我們的效能有很大的改良，因為我們的 CH 選擇條件有加入相同移動方向才加入叢集的限制。限制較為嚴格，所以產生的叢集會比較多。圖 6 中，LID 與 LCC 進入叢集首選舉的條件比較簡單，只憑著 ID 為條件。因此會造成叢集成員的快速加入又離開的現象，所以叢集成員會略差一些。圖 7 中，一開始 CPM 會比 MOBIC 低的原因，是因為我們的演算法在叢集選定條件下比較多條件，所以一開始產生的叢集數量比較少，相對的時間也少。圖 8 中，我們的效能會比 LID、LCC、MOBIC 提升不少，是因為 LID 與 LCC 的叢集選定簡單，因此它們形成叢集的機會相對的高，所以不在叢集內的時間就會低。圖 9 中，由於我們的演算法應用是定位在資料散佈上面，因此我們希望叢集組成的目標可以達到大量資料分享的特點。所以我們把目標放在提升叢集的存活時間、成員數目，以及減少行動節點處於 UN 的時間。而像 MOBIC 這類因為當加入條件比較嚴苛，雖然可以有效提升行動節點駐留時間，但是會導致成員個數過少，且行動節點處於 UN 的時間過多，而不利於資料散佈的應用。

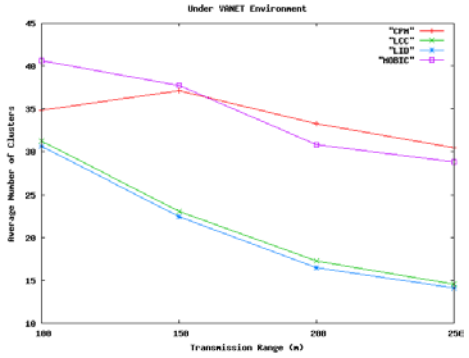

圖 5、平均叢集數

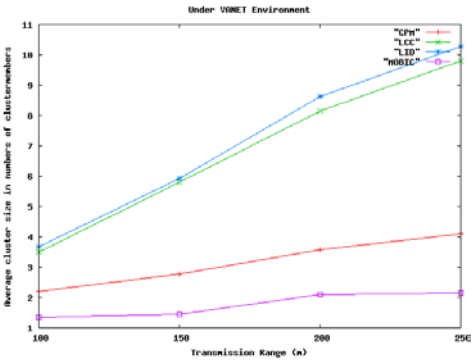

圖 6、平均叢集成員個數

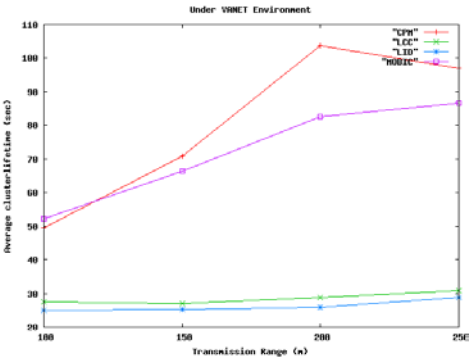

圖 7、平均叢集存活時間

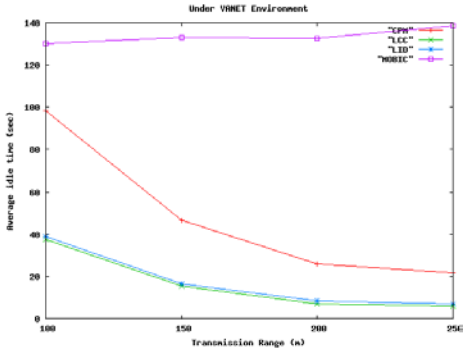

圖 8、平均閒置時間

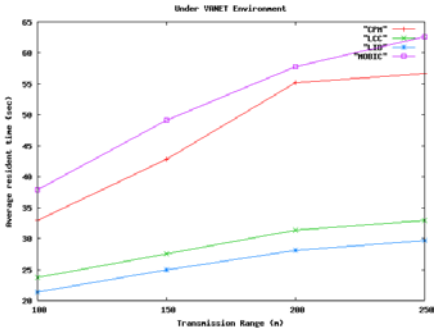

圖 9、平均駐留時間

五、結論與未來展望

在本篇論文中，我們整理了傳統行動隨意網路與車載網路中已被提出來的叢集演算法，並討論它們的優缺點。並且提出一個以中心位置與行動節點的移動性為基底的複合式度量叢集演算法，雖然計算量相較其他演算法來得高，但是換取得是資料分享的路徑較不易斷裂，以及可以使得叢集首與叢集成員之間的距離短，進而穩定通訊的品質，增加資料散播的速度，發揮資料分享的優勢。我們的叢集演算法能夠保證在車載網路下的穩定性，像是減少叢集重建的次數、增加叢集成員的數量、延長每個行動節點在叢集內的時間、減短每個行動節點不在叢集的時間。

在未來的展望裡，我們希望能進一步的考慮下列幾點：探討多個叢集首的演算法、結合路側單元以得到更多駕駛者所想要的資料與訊息。

參考文獻

[1] C. E. Perkins, "*Ad Hoc Networking*," Addison-Wesley, 2001.

[2] M. Gerla and J. T. C. Tsai, "Multiuser, mobile, multimedia radio network," Wireless Network, vol. 1, pp.255–265, Oct. 1995.

[3] A. Ephremides, J. E. Wieselthier, and D. J. Baker, "A design concept for reliable mobile radio networks with frequency hopping signaling," Proceedings of IEEE, vol. 75, pp. 56–73, 1987.

[4] K. Xu, X. Hong, and M. Gerla, "A heterogeneous routing protocol based on a new stable clustering scheme," in Proc. MILCOM, vol. 2, pp. 838-843, Oct. 2002.

[5] C. C. Chiang et al., "Routing in clustered multihop, mobile wireless networks with fading channel," in Proc. IEEE SICON, 1997.

[6] P. Basu, N. Khan, and T. D. C. Little, "A mobility based metric for clustering in mobile ad hoc networks," in Proc. IEEE ICDCSW, pp. 413–418, Apr. 2001.

[7] T. D. C. Little and A. Agarwal, "An information propagation scheme for VANETs," in Proc. IEEE Intelligent Transportation Systems, 2005.

[8] Y. Gunter, B. Wiegel and H. P. Grossmann, "Cluster-based medium access scheme for VANETs," in Proc. IEEE Intelligent Transportation Systems, pp. 343–348, Oct. 2007.

[9] VanetMobiSim, http://vanet.eurecom.fr/

[10] CanuMobiSim, http://canu.informatik.uni-stuttgart.de/mobisim/

[11] TIGER,http://www.census.gov/geo/www/tiger/%2011

[12] NS-2.33, http://www.isi.edu/nsnam/ns