A Fault-Tolerance Embedded Stable Routing Algorithm based on Signal Strength in Ad Hoc Wireless Networks

***I-Shyan Hwang, * Yao-Chang Hsieh and **Chiung-Ying Wang**

***Department of Computer Engineering and Science**
Yuan-Ze University, Chung-Li, Taiwan, 32026
****Department of Information Management**
Transworld Institute of Technology, Douliu, Taiwain 640
E-mail: *ishwang@saturn.yzu.edu.tw, *s887452@mail.yzu.edu.tw, **ann@tit.edu.tw

摘要

隨意式無線網路(*Ad Hoc* wireless networks)是一種新的網路型態，和一般無線網路環境比較，它不需要有基地台(base station)的架構，而是透過網路上的某些移動台(mobile host)傳遞資料。每個移動台就好像動態的(dynamic)基地台一般，提供路由(routing)的機制。由於具有移動性(mobility)的特性，從來源端到目的端的傳輸路徑，會因為某個移動台移動而使得整條路徑斷線。此時，來源端或中間節點必須花費額外的時間找出一條新的路徑來傳遞資料。為了補救此狀況發生，我們提出一個以訊號強度為感測且具有容錯(fault tolerance)機制的穩定繞送演算法。依接收端所感測的訊號強度(signal strength)，定出兩個高、低門檻值(threshold)，來作為判定穩定鍊結的依據。當鍊結的訊號強度大於高門檻值時，表示兩節點非常接近，會造成此路徑的跳躍數(number of hops)增加；反之，當鍊結的訊號強度小於低門檻值時，表示接收端正處於傳輸範圍的臨界點，增加路徑斷線的機率。因此，我們定義所謂的穩定鍊結為接收的訊號強度介於兩門檻值之間且瞬間訊號強度變化趨近於零，利用這些穩定的鍊結從來源端傳遞資料到目的端。除了使用第一條最穩定路徑(stable routing path)外，我們使用容錯機制來找出其他可用的穩定路徑來增加繞送穩定度。最後，撰寫模擬程式驗證以及比較其他演算法。

關鍵字：隨意式無線網路(Ad Hoc wireless network)、移動性(mobility)、訊號強度(signal strength)、穩定路徑(stable routing path)、容錯(fault tolerance)。

1. 簡介

在現有的無線網路(wireless network)中，資料的傳遞都是要透過有線的網路作主幹(backbone)，例如移動台(mobile)、個人數位助理（PDA）透過基地台或衛星（satellite）再與外界網路、其他移動台或個人數位助理作連繫。但是，若發生了一些很特殊的狀況，例如戰爭、大地震等災難，使得房屋、電線倒塌、基地台損毀，這些要透過固網（fixed network）或基地台的網路架構會變的不可靠，並且導致對外通訊中斷。因此，若能在此特殊地區使用筆記型電腦、個人數位助理等移動台，在不以有線網路為主幹的狀態之下，自己形成一個區域網路，然後再與外界做聯繫，便可以應付這些特殊狀況。這種不需透過有線網路或基地台，而藉由中間的移動台傳遞資料，使自己和遠端的移動台形成一個區域網路的無線環境，我們稱為隨意式無線網路（*Ad Hoc* wireless Network)。

如[1]、[2]、[3]，鑑於隨意式網路沒有基地台可以傳遞資料，每一個移動台是透過無線電波傳送資料，傳輸時要考慮到頻寬的問題，傳輸功率若太大，也就是傳輸範圍太大，會使得頻寬無法得到最佳的再利用（frequency reuse）[17]，因此每一移動台的傳輸距離不能太長。如果要傳遞的目的端離來源端很遠時，來源端就必須尋找一些移動台來幫忙傳遞資料，資料必須經由許多移動台一步一步的傳遞，這些移動台間所組合的路徑也就是從來源端到目的端的一條傳遞路徑。可動性（mobility）的特徵使得個人電腦不只侷限於固定地方，我們可以拿著筆記型電腦四處走動，但這會造成管理上的問題，當某移動台移動了，遠離了上一個移動台的傳輸範圍，就會收不到上一個移動台送來的資料封包，使得傳遞的路徑中斷而無法再往下傳遞給下一個移動台，所以，資料無法傳給目的端移動台。

會造成這斷線問題產生也是因為傳輸範圍的限制所造成的，當一移動台移動了，且遠離了上一個移動台的傳輸範圍，即無法收到封包，也就使得路徑斷線。所以，要解決此問題的發生也就要針對其特性（可動性與傳輸範圍的限制）而提出方法。在一般繞送演算法，如在[5]中提到的最短路徑演算法中，一路徑中，每個移動台要傳遞到的下一個移動台，一定是在他傳輸範圍內最遠的移動台，如此才可能使得路徑中要傳遞的移動台最少，一旦路徑上的移動台少，從來源端到目的端的封包處理時間也就相對的少了許多，在單一的端點對端點（end to end）的傳遞時間會是最快。但是，最短路徑演算法最大的缺點就是一移動台選出的下一個移動台是在他傳輸範圍內最遠的移動台，也就是在可傳輸範圍的邊緣了。因此，一移動台很容易就遠離上一個移

動台的傳輸範圍，造成收不到訊號而斷線。

一條傳遞的路徑若是斷線，會付出很大的的代價，如：end-to-end TCP problem、重新繞送的時間、封包流失。所以，我們需要一個穩定繞送演算法，找出一條路徑來繞送資料，並動態地處理動態的網路拓樸。在本文中，我們提出一個穩定繞送演算法，根據接收的訊號強度為測量標準，當一移動台收到另一鄰近移動台傳來的封包，該移動台利用測量裝置[9]，得到兩移動台間的訊號強度大小，並根據最近的兩次的訊號強度得到的瞬間訊號強度變化量，以此作為判斷穩定鍊結的考量。選擇一條最穩定的路徑，不但可以使資料一直以此路徑傳送資料，不需一直做路徑的維護而浪費成本。此外由於網路拓樸一直在變化，穩定的路徑也可能會變的不穩固，當其中的一個鍊結快失去聯繫時，我們提出一套新的方法，就是希望能預測它並在斷線前找出一條新的路徑來替代的容錯機制(fault-tolerance)。因此本篇就是要設計一套演算法，利用訊號強度尋找穩定、穩固的路徑繞送，且當此穩定路徑快斷線時，立即找另一條路徑替代的容錯機制。

本篇論文共五章。第二章描述既有的相關研究。第三章敘述我們的繞送演算法、容錯理論。第四章規劃模擬環境、參數的設定及討論模擬的結果。第五章敘述論文的結論和未來展望。

2. 相關研究

每個移動台都可以移動，所以管理上是一個很大的問題，例如路徑的尋找，要哪一些移動台當作路由器（router），要如何找出一條穩定又快速的路徑傳遞封包到目的端移動台？在[10]，[11]中，當路徑斷線後又該如何發現、如何處理？在[15]中，無線的網路環境，要依靠著空氣介質當傳媒，頻寬又是很短缺可貴的，要如何合理的配置，才能作最大的利用，又不會發生碰撞（collision）？當有很多移動台都要傳遞資料時，又該怎樣的排程(scheduling)才合理？在[12]，[13]中，在有限的電池容量，移動台要處理本機上的工作，又要發射功率傳遞資料，要如何有效率的利用才不會浪費電池呢？這些種種議題都很重要，也有很多的學者專家已在研究探討，而本篇論文主要是針對穩定的路徑繞送做探討。

2.1 隨意式無線網路下的繞送方式

在[1]中，隨意式網路中的每個移動台都可視為路由器，因為每一移動台都肩負著傳遞封包的責任。如圖1，節點C 不在節點A 的傳輸範圍（以節點A 為中心劃的圓，即為A 的傳輸範圍），節點A 亦不在節點C 的傳輸範圍裡。假如節點A 要和節點C 交換封包，一定要節點B 當作他們的橋樑、路由器，因為節點 B 都在節點A、C 的傳輸範圍內。因此，資料封包在路徑的傳遞路徑為：節點A 先傳遞給節點B，B 再傳給節點C。

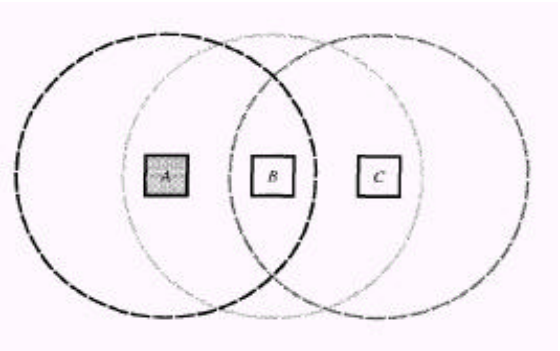


圖1：A 與C 的通訊需透過B 傳遞

2.2 隨意式無線網路下的繞送演算法

在[2]，[16]中，繞送的策略可分為靜態繞送（Static）和動態繞送（Dynamic or adaptive)，集中式管理（Centralized）和分散式管理（Distributed)。

在靜態繞送中，事先就算出一條路徑，之後，若有資料要傳輸，就照著這路徑一步一步地送達目的端；但是網路上的狀態是無時無刻的變化，這最佳路徑可能因為一時的擁塞導致它已不再是最佳路徑、甚至線路中斷而無法傳遞，若是我們再繼續走這路徑，我們將不會得到最好的效益。動態繞送可以解決這問題，它可以依據最新的資訊而動態地選擇目前最好的一條路徑。

靜態繞送與動態繞送，也可以說分別是Table-driven 繞送與On-demand 繞送的機制。在Table-driven 演算法中，先算出到網路上所有節點的路徑，之後，若有資料要傳輸，參考路由表（routing table）中的資訊，找出一條路徑來傳送，照著路徑上的節點依序傳遞到目的端節點。但是，表中記載的路由資訊並非及時的，而是前一個時間點所收集到的資訊，所以，很有可能在下一個時間點，路徑已經斷掉、不存在了。因為隨意式網路的網路拓樸（topology）一直在變化，為了使路由表與目前的網路拓樸更接近，每隔一段時間，每一個節點都要將封包廣播出去，將自己附近的資訊告知網路上其他節點，以讓其他節點更新路由表，如此將會使得網路上充斥著封包增加網路上的負載（overhead)。因此，維護路由表的成本相當高。On-demand 可以解決這些問題，每個節點想傳遞資料，才根據最新資訊尋找一條路徑，因此它有最新的資訊而動態地選擇目前最好的路徑；也不用每隔一段時間將封包廣播出去增加網路負載。但是，相對的，它也有一個缺點，臨時地搜尋路徑，會浪費一些時間。但與其使用一個過時的資訊，增加網路負載，不如使用最新的資訊來繞送資料。

隨意式網路的網路環境中，是不可能有集中

式管理的節點，因為移動性，此刻，此節點在這裡，下一刻他可能移動到另一個地方，其他節點如何找到他、向他詢問繞送資訊？又如果這個管理繞送的節點當機了，則整個網路將全部癱瘓。因此網路上的每個節點都該自己決定自己到其他節點的最佳路徑。

因此結合兩項優點（分散式管理與On-demand 繞送演算法），隨意式網路的繞送演算法要架構在動態分散式繞送演算法（Dynamic Distributed Routing algorithm）。而我們的繞送演算法就是架構於此。

2.3 TCP 上的繞送問題

在TCP/IP 的架構中[10]、[11]、[14]，TCP layer 所負責的是來源端和目的端兩點的資料處理，負責網路中通訊的最後處理，它要將一個個打散的封包重新組合，讓資料回復從來源端傳出前的原貌。所以，若是資料封包正確無誤的話，目的端會發出回應封包（Acknowledgment）告訴來源端已經收到正確資料。

此外，在 TCP 上的擁塞控制(Congestion Control)中，來源端若是經過一特定時間（Time Out）沒有收到目的端回送一個回應封包，告知已收到正確的封包，來源端會以為網路上發生了擁塞現象，而將傳送資料的速度放慢，降低資料量，使網路上的擁塞消失。這在有線網路架構下的假設是非常合理的，因為發生 Time out 的原因幾乎都是由於網路擁塞。但是，在無線網路架構下，卻似乎有一點不大合適，因為在無線網路上會發生 Time out 的原因卻不只是因為網路發生擁塞，還有因為無線網路中的可移動性或傳輸介質的不穩定性等原因造成的資料流失，所以若要將現行有線網路上的 TCP-Congesting Control 運用到無線網路，有一些機制是需要修改的。另外，發生斷線時，中間節點要如何發現，要如何將訊息傳回來源端，都引發很大探討，而且處理斷線問題也要花費相當多的時間。

在本篇論文中，我們發展一套穩定演算法以減少斷線的機率，讓 TCP-Congesting Control 盡量不發生，並減少斷線處理的時間。

2.4 以功率為考量的繞送演算法

在[5]中，SSA 演算法為 On-demand 的演算法，以訊號強度決定節點是否處於穩定狀態，尋找一條可以維持很久而不斷線的路徑。演算法分為路徑尋找（set-up）和路徑維護（maintenance），其主要的想法如下：

A． 路徑尋找：

1. 來源端節點廣播一封包去尋找目的端節點。
2. 當中間節點收到此封包，假如偵測兩節點間的訊號強度夠大的話，而且此封包以前沒收過，他就繼續將此封包廣播出去；否則就將此封包丟棄，不傳出去。
3. 這個封包會紀錄著它經過哪些節點。
4. 當目的端節點收到第一個抵達的封包時，馬上將它照著來時的路徑回傳來源端節點。這封包記錄的路徑將是最短的穩定路徑，也不會發生擁塞。

B． 路徑維護：

當一中間節點發現他到下一個節點的連線中斷時，馬上送一個錯誤訊息給來源端節點，通知路徑斷線。當來源端節點收到後，作以下動作：

1. 發送 route-search 封包，尋找新的路徑，
2. 送出訊息，告知所有中間節點別再使用已斷線的路徑。

此SSA 方法，若是收到的封包訊號強度小於一門檻值時，便捨棄此封包，不繼續廣播出去，如此的結果可能導致兩點間有路徑存在，但卻找不到路徑的問題發生。另外，SSA 沒有做路徑崩壞預測，所以，當路徑崩壞時，依然要花額外時間搜尋新的路徑。

在[6]中，S_{nn} 定義為移動台 n 到移動台 m 兩點間的信號強度。當 S_{nm} 小於某個門檻值 S_t，會因為訊號太弱，節點 n 無法和節點 m 聯繫。因為通訊是雙方面的，一定要 S_{nn} 和 S_{mn} 都大於 S_t，兩節點方能聯繫。為了能得到自己與相鄰節點的訊號強度大小、變化量，每個節點每隔一段時間要發送一個封包給相鄰節點，所以，每個節點會收到所有相鄰節點送來的封包，然後記錄每個封包的訊號強度大小。

聯繫關係（affinity）為兩節點間穩定度的關係，是該篇論文最重要的參數，是尋找穩定路徑且減少擁塞發生的關鍵。兩節點 n 和 m 的 Affinity a_{nm} 定義如下：

a_{nm} = high, if $\triangle S_{nm(ave)}$ is positive; or,
$(S_t - S_{nm(current)}) / \triangle S_{nm(ave)}$, if $\triangle S_{nm(ave)}$ is negative. （公式1）

其中，$\triangle S_{nm(ave)}$是最近幾次的平均訊號變化量，$S_{nm(current)}$是最近一次測得的訊號強度。假如，$\triangle S_{nm(ave)}$為正，表示兩節點正在接近，affinity 不必量化，只需設為 high，表示有好的聯繫關係。

演算法如下：

來源端節點廣播一個封包到目的端節點，目的端節點收到後，再照原路徑回傳給來源端節點，來源端節點若收到第一個符合以下條件的封包，就以該封包記載的路徑傳遞資料。

條件：

令 $\eta^p_{sd} = \min[\forall_{i,j}\ \eta^p_{ij}]$ {find the stability of path k}， $D_{sd} / C < f * \eta^p_{sd}$ （公式2）

i、j 為P 路徑上的任兩相鄰節點，η_{ij} 定義為 $\min[a_{ij}, a_{ji}]$。令該路徑的最小 affinity η^p_{ij}

為 η^{p}_{sd}，若是 η^{p}_{sd} 乘以一個常數 f（correction factor）大於（要傳的資料量(D_{sd})除於可用頻寬(C)），則此條路徑就是穩定的路徑。

此演算法雖然利用頻寬作為考量，以能傳輸所有資料量的路徑為優先考量，但是網路拓樸一直在變化，當資料傳輸到一半時，便很可能會斷線導致傳輸中斷。因此，若是能做路徑崩壞預測，在斷線前，及時切換到新的路徑，便可在穩定、不斷線的路徑上將資料傳完。該演算法每隔一段時間要廣播封包給相鄰節點，以測量訊號強度和變化量，此動作將造成網路的嚴重負載。

在[7]中，在TCP 的協定上，作者模擬證明一條路徑的跳躍數(Hop Count)將對於吞吐量有多大的影響。所以，在繞送的演算法，不能只單單考慮聯繫關係，還要考慮跳躍數。如圖2，TCP 的封包吞吐量（throughput）會隨著跳躍數的增加而遞減。

判定路徑演算法為：令 B_i 代表在路徑 i 上要傳遞的資料量（單位為位元組)。B_i 的公式如下：

$$B_i = t_i * a_{min}(i) \qquad \text{（公式3）}$$

t_i 為這條路徑的吞吐量，$a_{min}(i)$ 為此路徑的最小聯繫關係值，目的端節點選擇一條具有最大的 B 值，也就是評估一條在斷線前可傳輸的資料量。在路徑維護部分，若是一路徑斷線，來源端會被通知路徑崩毀，並重新尋找一條路徑從來源端到斷裂的節點，假如有找到就使用此路徑，但是，若在經過幾次收尋而一無所獲時，來源端就重新廣播尋找到目的端節點的路徑。

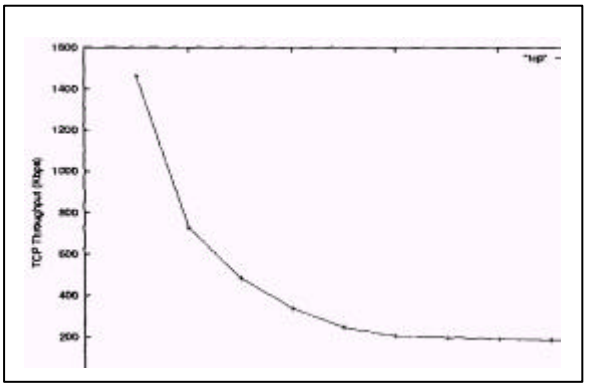


圖 2：一路徑的跳躍數與 TCP 吞吐量的關係圖

在[9]中，提到群組（cluster）形成的演算法，名為 ASAP algorithm。分為兩種 pilot 封包，一為 init pilot，另一為 normal pilot。

群組形成：

每個節點以最大的功率送出 init pilot 封包，將自己的資訊傳播給所有相鄰的節點，收到的節點便可得知是誰傳來的（ID）還有兩點的距離。接下來，使用 LID 演算法去形成群組並找出群組標頭（cluster-head)。

另外，該篇也說明，群組中的群組標頭與一般節點的功率控制（power control)：

1. **群組標頭的功率控制：**
   群組標頭若要動態的調整群組大小，可以動態調整傳送 pilot 封包的功率，若要使得群組變小，即降低傳送 pilot 封包的功率；反之，欲使得群組變大，即增加傳送 pilot 封包的功率。並且在傳輸資料給其他節點時，調大發射功率，讓群組內較遠的節點能收到封包。
2. **一般節點的功率控制：**
   每個節點隨時注意群組標頭送來的 pilot 封包，當節點收到後，判定兩點之間的傳輸功率程度（transmission power level)。當一般節點要與群組標頭聯繫時，就以最近測到的傳輸功率來傳送封包，節省能源的浪費。

在[18]中，所提出的穩定繞送演算法是以能源消耗作為繞送時的參考，找出能源消耗最小的路徑，所以此繞送演算法所找出的路徑具有節省能源消耗的特性。在路徑維護機制中包含了 Route-Based Reroute 以及 Local-Based Reroute 兩個機制。Route-Based Reroute 機制的主要運作方式，是藉由偵測出路徑能源變動次數大於路徑長度的b倍時啟動 Reroute 機制；在 Local-Based Reroute 機制中，定義不穩定的鍊結為，第一種狀態為移動造成鍊結的不穩定，即 $n_{I,j} > 0$ 且 $n_{I,j} + P_{i,j} \geq Pm_{i,j}$ 時，其中，$n_{I,j} = P_{i,j}' - P_{i,j}$，$P_{i,j}'$表示目前收集到路徑區段由主機 i 到主機 j 的能源位階等級，$P_{i,j}$ 表示前一個由主機 i 到主機 j 的能源位階等級，$Pm_{i,j}$ 為節點 i、j 可以提供最大傳輸的能源位階；第二種狀態為該主機能源無法負荷造成路徑的不穩定，即 $n_{I,j} = 0$ 且 $P_{i,j} = Pm_{ij}$ 時。若以上兩者中的一狀態發生，便啟動 Reroute 機制。藉由這兩個機制，可以降低資料傳輸路徑中斷的機率，並且使路徑具有較高的穩定性。

3． 系統設計與架構

此章節，我們提出以訊號強度為量測標準的穩定繞逕演算法，與路徑維護的容錯觀念。

3.1 系統假設

為了能測量到鄰近移動台的訊號強度，每個移動台都必須配置「訊號測量、調整裝置」(power measuring/adjusting circuitry) [9]。當一移動台廣播一封包，鄰近的移動台收到此封包時，經由「訊號測量、調整裝置」測量後，即可得到發送者與接收者之間的訊號強度，經由訊號強度大小我們可得知兩移動台的距離是相近還是遠離。訊號強度比較大，兩點距離較近；反之，訊號強度比較小，兩點距離較遠。

以訊號強度為量測標準，定義兩訊號高、低門檻值（threshold）S_{high} 與 S_{low}，和危險門檻值(dangerous threshold) S_{danger}，如圖3。

1. 穩定狀態（Stable state）：當一移動台接收到另一移動台送來的封包，由封包測出的兩移動台間的訊號強度介於 S_{high} 與 S_{low} 之間。
2. 危險區域（Dangerous zone）：當兩移動台間的訊號強度小於危險門檻值（S_{danger}）。

當兩移動台間的訊號大於高門檻值（S_{high}）時，雖然屬於穩定狀態，但是卻會造成一條路徑的跳躍數（number of hops）變多，使得兩端點間延遲時間(end to end delay time)變長。最後我們假設每一移動台的最大發射功率（maximum transmitting power）都是一樣的。

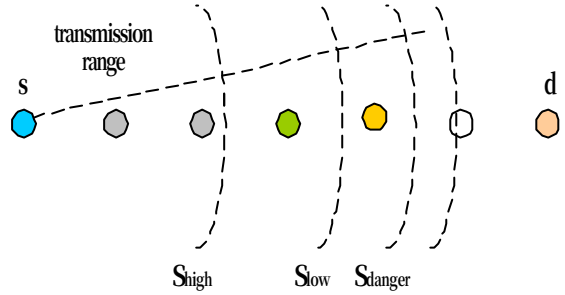


圖3：定義兩高、低門檻值 S_{high} 與 S_{low} 與危險門檻值（S_{danger}）

3.2 以訊號強度為基礎的穩定繞送演算法

為了得到一條穩定、不易斷線的路徑傳遞資料，我們提出一個穩定繞送演算法。我們希望能找到一條穩定的路徑：

（1）. 路徑的所有鍊結的訊號強度皆在兩個訊號門檻值之間，

（2）. 且所有鍊結瞬間訊號變化量趨近於零。

所有鍊結的訊號強度介於兩個訊號門檻值的原因是：當鍊結的訊號強度大於高門檻值時，表示兩節點非常接近，會造成此路徑的跳躍數(number of hops)增加；反之，當鍊結的訊號強度小於低門檻值時，表示接收端正處於傳輸範圍的臨界點，容易造成路徑斷線的機率。

瞬間訊號變化量趨近於零表示這鍊結是屬於穩定的狀態，不容易斷線；反之，若是因為若瞬間訊號變化量大於零或小於零表示，這鍊結有一個或兩個節點都在移動，甚至可能是一直維持直線移動，那麼將很可能在短時間內斷線。接著我們敘述穩定繞送演算法的作法：為了找出一條穩定路徑，來源端節點要先廣播一「路徑尋找封包」(Route-Request packet）尋找目的端節點，目的端節點收到封包後再照原路回傳給來源端節點，來源端節點等待一 Time-Out 時間後，從所有回傳封包中記載的資訊，決定哪一條路徑是最穩定的路徑。尋找路徑的過程如圖4。

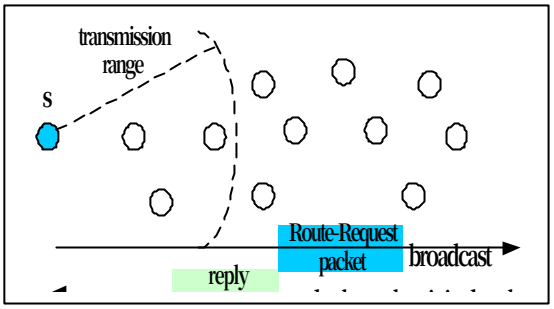


圖4：廣播到目的端 d，d 收到後，照原路傳回來源端 s

Step 1. 來源端節點廣播「路徑尋找封包」，尋找目的端節點。
Step 2. 當目的端節點收到「路徑尋找封包」，依原來路徑回送給來源端節點。
Step 3. 來源端節點等待 time-out 時間，收集由目的端節點回傳的「路徑尋找封包」。
Step 4. If (每條鍊結的訊號強度皆在兩門檻值 S_{high} 與 S_{low} 之間且 ΔP 趨近於0) 的路徑存在，到 step 5， else if (每條鍊結的訊號強度皆在兩門檻值 S_{high} 與 S_{low} 之間) 的路徑存在，到 step 5， else if (每條鍊結的訊號強度大於低門檻值 S_{low}) 的路徑存在，到 step 5， else if (鍊結訊號強度小於低門檻值 S_{low}，且它的 ΔP 要大於0) 的路徑存在，到 step 5，
Step 5. 從以上封包中的 number of hops 欄位找出最少跳躍數的路徑作為最穩定路徑。
Step 6. 以 step 5 得到的穩定路徑傳送資料。

表1：穩定繞送演算法的路徑選擇

在廣播到目的端節點時，若移動台收到此封包時，需將自己的相關資訊，例如自己的 ID、自己與上一個移動台的訊號強度等放入封包，然後再繼續廣播出去。最後，當目的端節點收到由不同路徑傳遞來的「路徑尋找封包」，即可從封包裡的資訊得知它是經過哪些節點傳遞過來，得此資訊之後，再沿著原來路徑送回來源端移動台（並非廣播）。在該路徑上的每個移動台收到此封包時，也是根據自己目前的狀態而更改封包裡的資訊，例如自己與上一個移動台的訊號強度、並依據上一次和這一次的兩點間訊號強度，所得的瞬間訊號強度大小(ΔP)等資訊放入封包。最後在 Time-Out 時間後，來源端移動台從各路徑

收到「路徑尋找封包」，決定哪一條路徑最為穩定且快速，演算法如表1。接下來，立即由此路徑傳送資料封包到目的端節點。

「路徑尋找封包」的封包格式如下（如圖5）：

1. Source ID: 來源端移動台ID。
2. Destination ID: 目的端移動台ID。
3. Number of hops: 來源端到目的端的跳躍次數。
4. Every node's ID: 此路徑上經過的移動台ID。
5. Link's signal strength: 路徑上每一段鍊結（link）的訊號強度。
6. Link's signal variation: 路徑上每一段鍊結的瞬間訊號強度變化量。

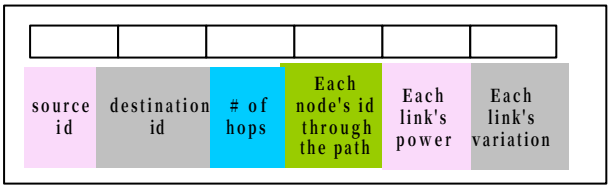


圖5：「路徑尋找封包」的格式

接下來細說在尋找路徑時，各移動台所扮演的角色和其工作，將所有移動台分為三種角色：來源端節點（source node）、中間節點（intermediate node）和目的端節點（destination node）個別討論，並分兩階段討論（來源端至目的端、目的端至來源端）。

A. 來源端到目的端：

1. 來源端節點 s：

首先，來源端節點先廣播「路徑尋找封包」，以尋找一條穩定的路徑到達目的端節點，然後設定一 Time-Out 時間，等待從目的端節點送回的「路徑尋找封包」。在初始化時，封包中的number of hops 欄位設為 0，Source ID 欄位和 Destination ID 欄位分別設為自己的ID和目的端節點的ID。

2. 中間節點 m：

非來源端且非目的端節點即為中間節點。當中間節點收到前一個節點 n 傳來的「路徑尋找封包」，經由「訊號測量、調整裝置」，可以測量出前一節點與自己之間的訊號強度 P_{nm}，並將 P_{nm} 加到 Link's signal strength 欄位、自己的ID 加到 Every node's ID 欄位、number of hops 欄位的值加1。再將此封包廣播出去。

3. 目的端節點 d：

當一節點檢查上一個節點 n 送來的封包，若封包的 Destination ID 欄位等於他自己的ID，此節點即為目的端節點。一樣地，目的端節點將 P_{nd} 加到封包中 Link's signal strength 欄位，並在 number of hops 欄位的值加1。這時候的封包已經記錄著此封包由來源端到目的端節點經過哪些節點、總共幾次的跳躍數和每條鍊結的訊號強度大小。從封包裡Every node's ID 欄位，得知它是經過哪些節點傳遞過來，沿著這些節點反方向送回來源端節點（非廣播式回送）。

B. 目的端到來源端節點：

回傳給來源端節點時，為了得到最新資訊和兩相鄰近節點的瞬間訊號變化量，每個中間節點要再將本身最新資訊寫入封包。

1. 目的端節點 d：

沿著原路徑將封包回送到來源端節點。

2. 中間節點 m：

當中間節點收到前一個節點 n 傳來的「路徑尋找封包」，得到上一節點 n 與自己之間的訊號強度 P_{nm}，將原來 Link's signal strength 欄位的 P_{nm} 修改成目前最新的 P_{nm}，並將 ΔP_{nm} 加到 Link's signal variation 欄位，修改封包後，再將此封包傳到下一個節點。ΔP_{nm} 為節點 n 與自己(m)間的瞬間變化量（公式如下）。

$$\Delta P_{nm} = (P_{nm(current)} - P_{nm(previous)}) / \Delta t \quad \text{（公式4）}$$

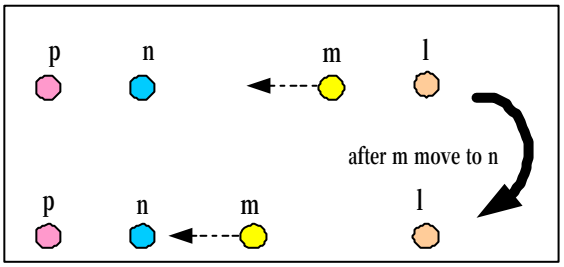
$P_{nm(current)}$ ：這一次節點 n 與 m 間測到的訊號強度。

$P_{nm(previous)}$ ：上一次節點 n 與 m 間測到的訊號強度。

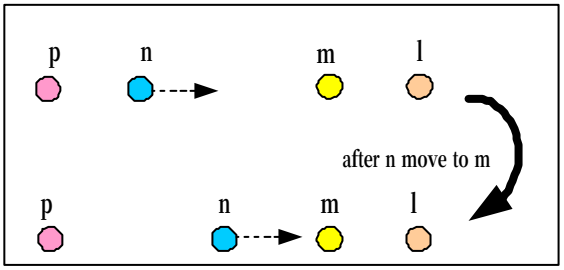
a） 若 ΔP_{nm} 趨近於0，表示兩節點幾乎都沒移動（no mobility），屬於穩定狀態。

b） 若 ΔP_{nm} 大於0，表示兩節點正在靠近中，如圖6，但相對的，節點 n 會遠離 n 的另一邊節點（節點 p）或 m 會遠離 m 的另一邊節點（節點 l）。因此，此情況也不算是穩定狀態。

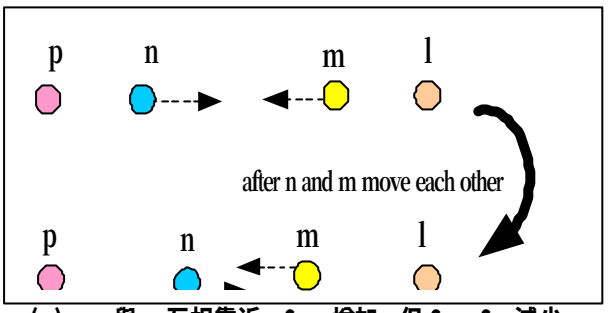
c） 若 ΔP_{nm} 小於0，表示兩節點正在遠離中，兩點當然不屬於穩定狀態。



（a）： m 靠近 n，S_{nn} 增加，但 S_{nl} 減少。



（b）： n 靠近 m，S_{nn} 增加，但 S_{pn} 減少。



(c): n 與 m 互相靠近，S_{nm} 增加，但 S_{pn}，S_{ml} 減少。

圖 6 ：ΔP_{nm} 大於 0 的關係圖

3. 來源端節點 s：

a) 從 time-out 時間內收集到的封包，決定哪一個封包記錄的路徑在未來會最穩定。尋找的方式如表 1。

找到穩定的路徑後，就依此路徑傳送資料到目的端節點。

3.3 容錯機制

當來源端節點開始以一條穩定的路徑傳遞資料時，雖然它是目前最穩定的路徑，但因為移動性的關係，經過一段時間後可能造成兩相鄰節點從可以聯繫到可能無法聯繫（斷線）。因此，在隨意式無線網路架構裡，一條穩定的條路徑不大可能一直永遠保持聯繫。

所以，找到一條穩定路徑後，它也不可能一直為我們傳遞資料。我們必須一直觀察它的狀態，隨時觀察它是不是可能快斷線了。若是快斷線的話，就該馬上再找一條穩定的路徑取代原來路徑，由新的穩定路徑傳遞資料，這就是容錯（fault tolerance）的觀念，也就是穩定路徑的維護措施（route maintenance），讓資料一直保持在穩定的路徑上傳遞，寄望能保持零斷線的機率。因此，每一個移動台 m 在傳遞的過程，皆要密切注意上一個節點 n 傳來的資料封包，測量訊號強度大小。

若符合以下兩項條件，表示節點 m，n 可能快斷線了，將會開始實行容錯機制（fault tolerance mechanism），尋找新的穩定路徑。

1) 若節點 m，n 間的訊號強度在危險區域（dangerous zone），（也就是 P_{nm} 小於危險門檻值 S_{danger}），

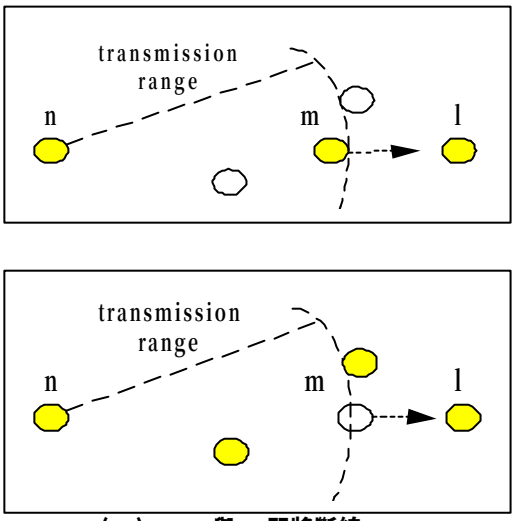
2) 節點 m，n 是逐漸遠離的（$\Delta P_{nm} < 0$）。

容錯機制分為兩部分：區域容錯機制（local fault tolerance）和全域容錯機制（global fault tolerance）。當上述兩條件成立，兩節點可能即將斷線。節點 m 將發送「路徑重找封包」（reroute-request packet），給上一位節點 n 和來源端節點 s，要他們分別尋找區域新穩定路徑（local new stable path），和全域新穩定路徑（global new stable path）。若有找到新的區域穩定路徑，則採用它；若無，則採用新的全域穩定路徑。

1) 區域容錯機制：

何謂區域穩定路徑？為了避免大規模的替換整條路徑，將可能會斷線的這一鍊結（由 2 個節點組成的鍊結），替換成另一條路徑，這一小段路徑即為區域穩定路徑。

如圖 7 所示，節點 n、m、l 依序是來源端 s 到目的端 d 的中間節點，假設節點 m 往 n 的反方向移動（逐漸遠離），且已經在危險區域，節點 n 就必須找另一條替代路徑到達 l，因為 L_{nm}（節點 n，m 的鍊結）隨時都可能會斷。所以，節點 n 廣播「路徑尋找封包」尋找 l，尋找路徑的穩定繞送演算法如上述 3-2 節的方式一樣，但尋找的路徑跳躍數不得超過 3，因為原本的路徑只有 2 個跳躍數，若是找到一條要 5、6 個跳躍數的路徑，將會增加節點處理時間，也就增加傳輸時間、降低效能。



(a): m 與 n 即將斷線。

(b): n 到 l 的中間節點由 m 換成兩個深色節點。

圖 7 ：節點 n 找到替代路徑到 l，跳躍數為 3

若找到區域替代路徑，節點 n 便將此資訊放入「路徑尋獲封包」（reroute-got packet），並發送給來源端節點 s，節點 s 收到後，將原來即將崩壞的路徑替換成新的區域路徑，並以新的替代路徑傳遞資料。

2) 全域容錯機制：

在尋找區域替代路徑時，節點 s 也同時在尋找全域替代路徑，因為同時尋找新的路徑才不會浪費時間，導致原路徑斷線。來源端節點 s 廣播「路徑尋找封包」尋找目的端節點 d，尋找路徑的演算法如上述 3-2 節的方式一樣。

在尋找的過程中，若收到節點 n 送來的「路

徑尋獲封包」(n 找到區域路徑)，來源端節點 s 馬上改用新的區域路徑傳遞資料；若沒收到，則一直等到一個 Time-Out 時間，收集目的端節點 d 回傳的「路徑尋找封包」，再判定哪一條路徑最為穩定（也就是新的全域路徑），以新的替代路徑傳遞資料。

4． 效能評估

利用程式模擬實際狀況，並套用多組參數。先找出兩參數（訊號強度的高、低門檻值 S_{high} 和 S_{low}）的最佳值，再套用到我們提的穩定繞送演算法，與最短路徑演算法做斷線機率、吞吐量、跳躍數等比較。之後，再找出危險門檻值(S_{danger})的最佳值，應用在容錯機制中，比較有容錯機制或沒有容錯機制的穩定繞送演算法，將會對斷線機率、吞吐量有何改善，並對網路上封包負載有何影響。

4.1 模擬網路平台

模擬的環境如下：

1. 移動台的活動空間設定為 1500*1500 平方單位的空間。
2. 活動空間內的移動台個數，設定 100 台、200 台，並隨機分佈在各地方。
3. 每一移動台可能會動也可能不動，動與不動由隨機亂數來決定。
4. 移動台的移動速度設定為多種速度，10、20、30 (單位距離/單位時間)。
5. 移動台的移動方向，為 uniform distribution，每一次要移動前，都機率的選擇一個角度。若是移動到某個邊界時，無法再往該方向移動，並停在邊界，下一次的移動方向則不選擇此邊界，而是選擇其他角度方向。
6. 所有移動台的最大傳輸功率一樣（傳輸範圍一樣遠)，假設訊號強度與距離平方成反比(距離愈遠，接收到的訊號愈小)。最大傳輸範圍設定為 300、500 距離單位。
7. 模擬時間：300 單位時間。

4.2 模擬過程之討論與成果

首先，要找出兩個高、低門檻值 S_{high} 和 S_{low} 的最佳值，先將訊號強度轉換成距離單位，我們套用訊號強度與距離平方成反比公式，將 $(S_{high})^{-1/2}$ 與 $(S_{low})^{-1/2}$ 換成 SD_1 與 SD_2，且 $1 < SD_1 < SD_2 <$ (傳輸距離的最大值)。

將 SD_1 與 SD_2 組合套用，找出一組最佳的 SD_1 與 SD_2，會讓路徑有最少的斷線率。如圖 8 所示，X 軸套用表 2 的資料測試模擬，其中 X 軸每個數列中的值表示 SD_2 與 SD_1 (例如 90%~80%，表示 SD_2 =90% 傳輸距離與 SD_1 =80% 傳輸距離)。Y 軸則是在模擬的 300 單位時間內路徑重建的次數。右上角圖例的 N 代表節點數、T_R 表傳輸半徑、V 表節點移動速率。由圖 8 中，我們看出「SD_2 = (70% 傳輸範圍)」在各組環境的路徑重建的次數比較少。在圖 8 中，將各環境的 Y 軸加總，便得到圖 9，其中每組門檻值的 Y 軸為各環境的路徑重建次數加總。我們可以看出其中又以「SD_1 = 30%，SD_2 = 70% 傳輸範圍」在「SD_2 = (70% 傳輸範圍)」中的平均路徑跳躍數最少，因此此組「SD_1 = 30%，SD_2 = 70% 傳輸範圍」就是我們最佳的訊號高低門檻值。接下來，我們以此組訊號高低門檻值套用在穩定繞送演算法。

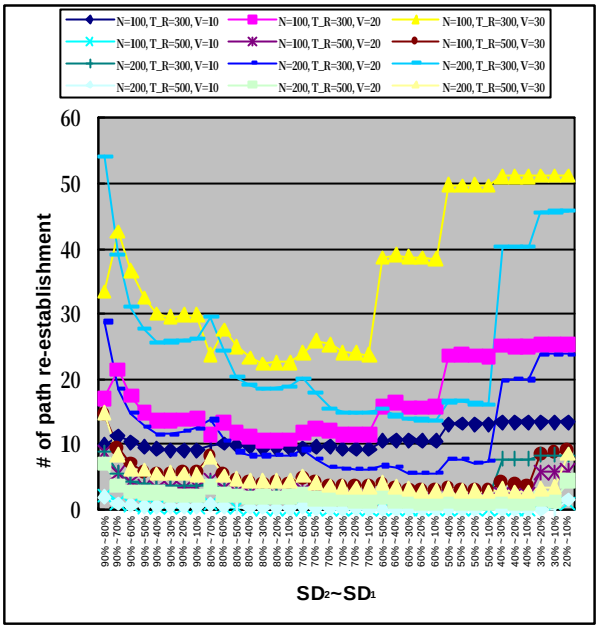


圖 8：兩訊號門檻值與斷線機率的關係圖

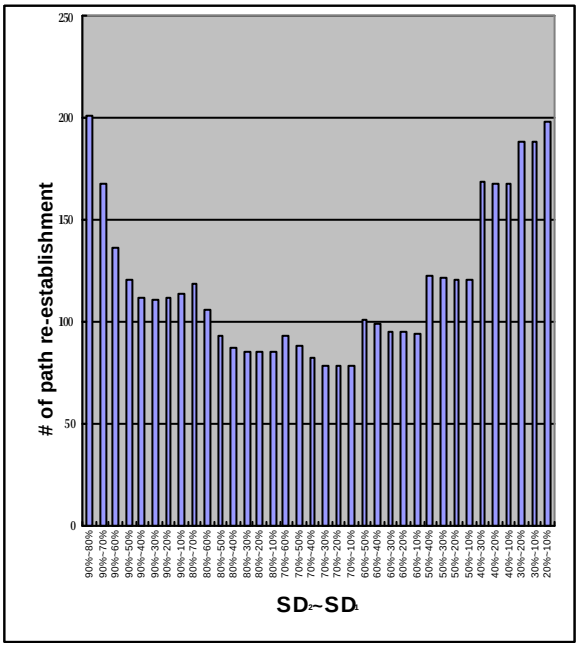


圖 9：兩訊號門檻值與斷線機率的關係圖(所有環境的 Y 軸加總)

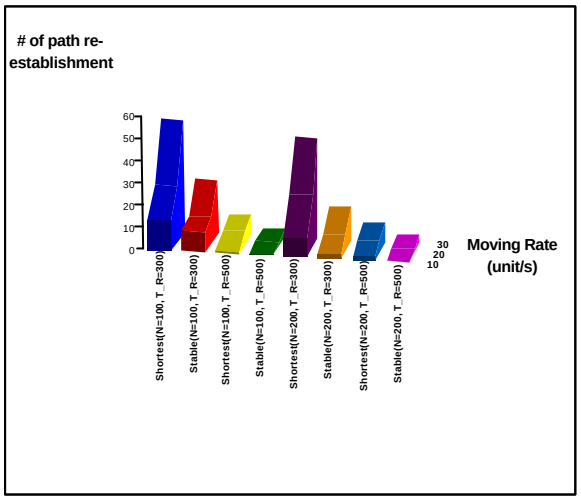


圖 10：穩定繞送演算法與最短路徑演算法的斷線機率比較

在圖 10，X 軸為移動速率，Y 軸為各種環境組合，單數值皆為 Shortest Path Algorithm，偶數為我們的穩定演算法。節點移動速率越大，Z 軸的斷線後重新繞送的次數越多；網路上的節點數越多，斷線後重新繞送的次數越少；傳輸範圍越大，斷線後重新繞送的次數越少。另外，我們提出的穩定繞送演算法和最短路徑演算法的比較下，可以明顯看出不管在何種環境下，我們穩定繞送演算法的斷線後重新繞送次數都比較少。因此證明我們提出的穩定繞送演算法有較低的路徑重建次數，也就是有較低的斷線機率。

在圖 11 中，節點移動速率、網路上的節點數與路徑跳躍數沒有明顯的關係。但傳輸範圍，就有較明顯的區別，傳輸範圍越大，路徑跳躍數越少。另外，和最短路徑演算法的比較下，我們穩定繞送演算法的路徑跳躍數比較多，平均多了一個跳躍數，原因是為了找出比較穩定的路徑，所以路徑上的鍊結數(跳躍數)會比較多。若是只考慮路徑的跳躍數，跳躍數越多，吞吐量就會越差，但是在考慮斷線機率的話，結果可能就不一樣。因為，當拓樸快速的改變，最短路徑演算法下的路徑會一直斷線，若處於一直斷線的狀態，則要花相當多的時間再重新尋找路徑，而且斷線亦會造成封包流失。所以，圖 12 中，在各種不同環境中，最短路徑演算法因為斷線率太大的關係，以致於吞吐量皆比我們的穩定繞送演算法較差，尤其是節點移動速率大的情況更是明顯。在最短路徑演算法中，移動速率越大，斷線率明顯增加、吞吐量也明顯往下下滑。另外，亦看出節點數越多、傳輸範圍越大，斷線率越低，吞吐量也越好。

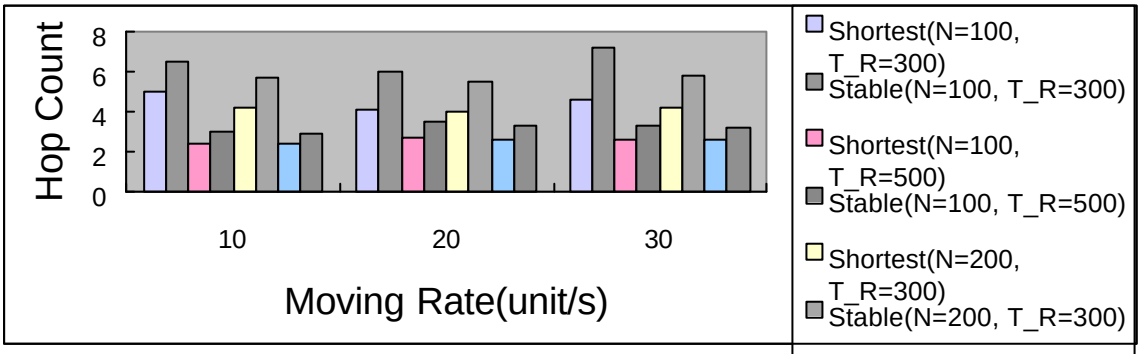


圖 11：穩定繞送演算法與最短路徑演算法的跳躍數比較

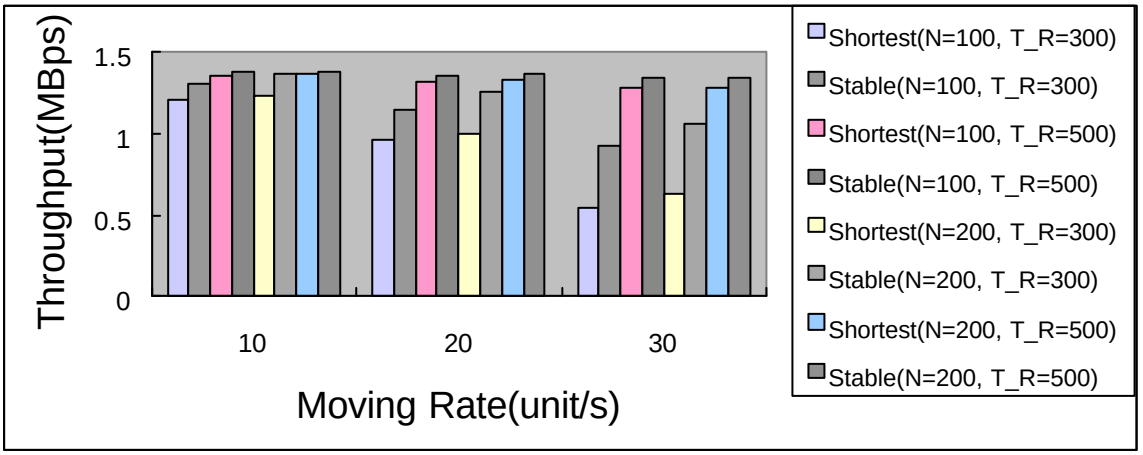


圖 12：穩定繞送演算法與最短路徑演算法的吞吐量比較

接著，我們討論具有容錯機制的穩定繞送演算法，找出啟動容錯機制的最佳危險門檻值 S_{danger}，先將 $(S_{danger})^{-1/2}$ 換成距離單位 SD_{danger}。令 SD_{danger} 套用表 2 的值，然後找出最少斷線率的 SD_{danger} 值。

SDdanger
100%傳輸範圍
95%傳輸範圍
90%傳輸範圍
85%傳輸範圍
80%傳輸範圍
75%傳輸範圍

表 3：SD_{danger} 的設定值

如圖 13 所示，X 軸為 SD_{danger} 值，其中「90%」表 90%傳輸距離，Y 軸表示在模擬 300 單位時間內的斷線後路徑重建次數。套用六種不同值來評估時，100%、95%、90%傳輸範圍所造成的曲線是往下降(路徑重建次數呈直線降低)，90%、85%、80%、75%傳輸範圍所造成的曲線是已呈現水平(路徑重建次數已不再明顯下降)，因此 SD_{danger} =90%、85%、80%、75%傳輸範圍，都可以有不錯的斷線率，而且斷線率也差不多。那麼要選哪一個 SD_{danger} 值呢？我們就考慮降低斷線率的所付出成本代價(Cost)—啟動容錯機制的 Local Reroute、Global Reroute 次數。所以，從這四組找出最少的 Local Reroute、Global Reroute 次數，也就是網路上的封包負載越少，所花費的成本越低。

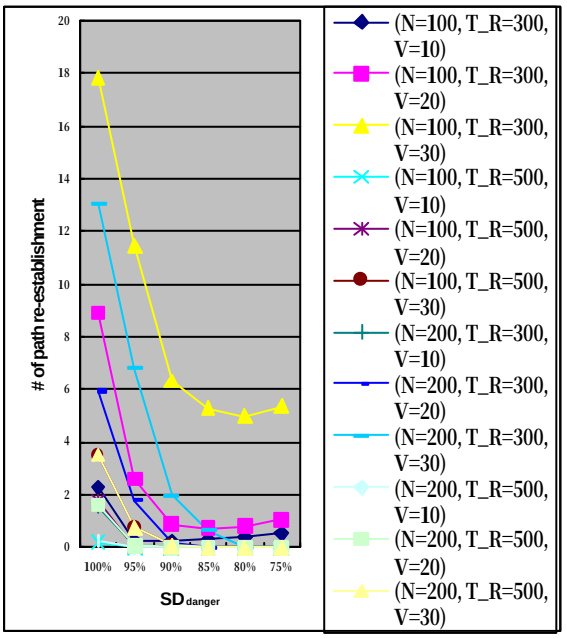


圖 13：危險門檻值與斷線率的關係圖

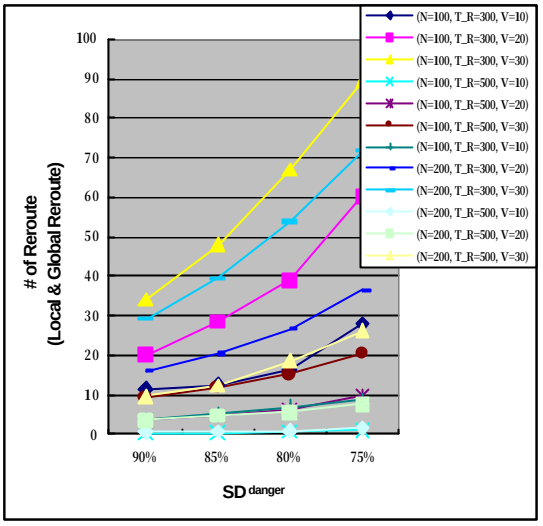


圖 14：危險門檻值與啟動容錯機制(斷線前的路徑繞送)的關係圖

如圖 14 所示，X 軸為 SD_{danger} 值，Y 軸則是在模擬的 300 單位時間內啟動容錯機制的次數(斷線前的路徑繞送次數)。從圖中看出，在容錯機制啟動次數，「90% 傳輸範圍」的表現最少、最佳。因此 SD_{danger} 將設成「90% 傳輸範圍」，並套用在「具有容錯機制的穩定繞送演算法」中。

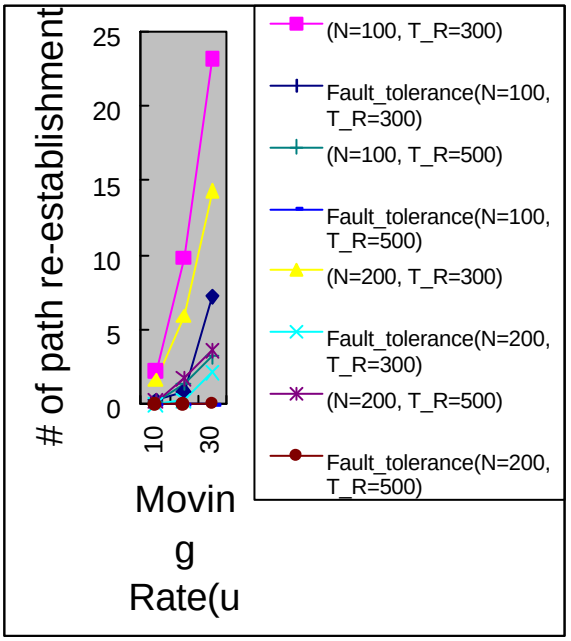


圖 15：在穩定繞送演算法，有無容錯機制的斷線機率關係圖

在圖 15，右上角圖示中，有「Fault-tolerance」表具有容錯機制的穩定繞送演算法，沒有「Fault-tolerance」表不

加上容錯機制的穩定繞送演算法。節點移動速率越大，斷線後路徑重建的次數越多；網路上的節點數越多，斷線後路徑重建的次數越少；傳輸範圍越大，斷線後路徑重建的次數越少。另外，我們提出的穩定繞送演算法在有無容錯機制下的比較，可以看出不管在何種環境下，有容錯機制下的斷線後路徑重建次數都比較少，趨近於零，因為都在斷線前就啟動容錯機制（Local and Global Reroute）了。因此證明我們提出「具有容錯機制的穩定繞送演算法」有很低的斷線後繞送次數（很低的斷線機率)。

雖然在斷線率上有較好的表現，卻需要付出一些代價的。圖 16 中，Y 軸為有容錯機制與無容錯機制的網路封包負載比值，明顯看出，有容錯機制與沒有容錯機制的比值多了 1 倍。這是因為容錯機制裡，Local Reroute 和 Global Reroute 所發出的廣播封包造成的。另外，移動速率越大(網路拓撲改變越大)，啟動容錯機制次數越多，兩者比值越大；傳輸範圍越小，比值越大。如圖 17 中，在各種不同環境中的吞吐量，有容錯機制的穩定繞送演算法都比沒有容錯機制的還好，吞吐量或相等或較好，移動速率越大(網路拓撲改變越大) 吞吐量越好，我們證明具有容錯機制的穩定繞送演算法有最佳的吞吐量。

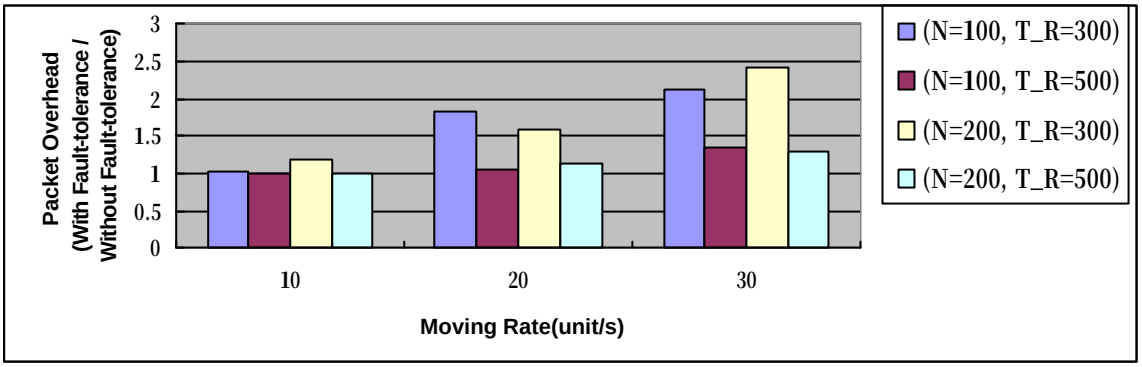


圖 16：在穩定繞送演算法，有無容錯機制的網路封包負載比值關係圖

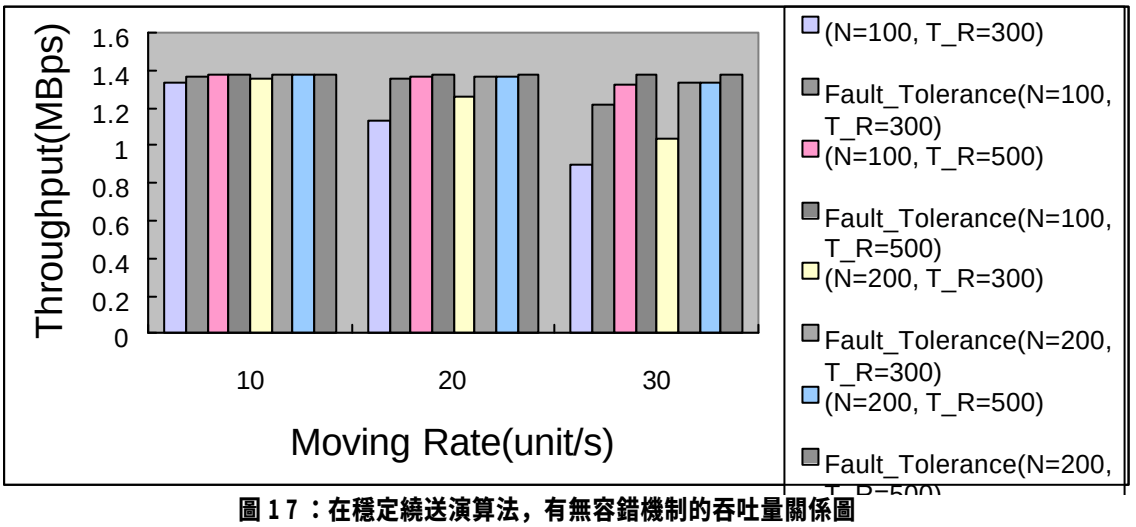


圖 17：在穩定繞送演算法，有無容錯機制的吞吐量關係圖

5 結論與未來展望

在動態的隨意式無線網路環境裡，網路拓樸一直在變化，如何在動態的網路環境提供穩定的路徑繞送是本篇論文的重點。為了達到我們的目的，我們使用訊號強度作為量測兩節點間的穩定程度，找出路徑所有鍊結的訊號強度皆在兩個訊號門檻值之間，且鍊結的瞬間訊號變化量趨近於零，以此為基準即可找出一條穩定、不易斷線的路徑。另外，我們預測路徑的斷線，當路徑的一鍊結訊號強度小於危險門檻值且鍊結的瞬間訊號變化量小於零，我們判定此路徑可能會斷線，並將資料切換到另一條替換路徑傳送的容錯觀念。如此可以大大減少資料傳到一半時斷線的機率，以致於封包的流失、浪費時間等待中間節點通知來源端路徑中斷、浪費時間再重新找一條路徑。和最短路徑演算法比較的模擬結果雖然發現「具有容錯機制的穩定繞送演算法」有較高的跳躍數，但卻具有較低的斷線率和吞吐量（因為最短路徑演算法的斷線率太高)。因此我們提出的演算法有較好的效能，另外，因為斷線率高、吞吐量低會使得同樣的資料量需要更久的時間傳遞，一路徑的頻寬（bandwidth）也會因為傳輸的時間增加而佔住頻寬，浪費頻寬資源。

由模擬的結果，我們的機制在節點快速移動的環境中，斷線率、吞吐量有更佳的表現(相對於最短路徑演算法)。所以，我們的演算法非常適合在網路拓撲快速改變(節點快速移動)的環境，此點在Ad Hoc的移動性特質來看是非常重要的。在考慮吞吐量的模擬時，我們並未考慮碰撞(collision)、信號干擾(signal interference)、及容錯機制造成的封包負載。加上這些變因才能使得網路環境更實際，因此，在未來工作裡，我們會朝著這方面改善。在本文中並未探討網路頻寬的問題，加上頻寬的考量才能更符合目前實際的網路架構，也就是在我們的機制中加上頻寬的考量--要如何在有限頻寬中有效的利用頻寬；有效、快速的預留頻寬給穩定路徑；公平的分配頻寬給使用者；保留一條路徑給容錯機制：在找到一條穩定路徑後，隨時留意網路動態並保留另一條路徑提供容錯機制使用，以免當容錯機制要啟動時，找不到可利用的頻寬當為替代路徑。我們提出的機制在QoS的貢獻上，提供使用者一個穩定、不易斷線的路徑利用，可以使資料一直穩定的傳遞，不會有資料中斷或爆衝(burst)、資料流不穩定(jitter)，提供資料的穩定控制(flow control)，所以，此機制很適合應用在語音、視訊的資料傳輸。在QoS的參數中，我們考慮了路徑的穩定繞送，但QoS還有其他可以探討的參數，我們希望可以再結合其他QoS參數使Ad Hoc更成熟穩定。

參考文獻

[1] D.B. Johnson, "Routing in ad hoc networks of mobile hosts", Mobile Computing Systems and Applications, 1994. Proceedings, Workshop on, 1995, Page(s): 158 –163.

[2] E.M. Royer and Chai-Keong Toh "A review of current routing protocols for ad hoc mobile wireless networks", IEEE Personal Communications Volume: 62, April 1999, Page(s): 46 -55.

[3] M.S. Corson, J.P. Macker and Cirincione, G.H. "Internet-based mobile ad hoc networking", IEEE Internet Computing Volume: 34, July-Aug. 1999, Page(s): 63 –70.

[4] C. Elliott, and B. Heile, "Self-organizing, self-healing wireless networks", Aerospace Conference Proceedings, 2000 IEEE Volume: 1, 2000, Page(s): 149 -156 vol.1.

[5] R. Dube, C.D. Rais, Kuang-Yeh Wang and Tripathi, S.K. "Signal stability-based adaptive routing (SSA) for ad hoc mobile networks", IEEE Personal Communications Volume: 41, Feb. 1997, Page(s): 36 –45.

[6] K. Paul, S. Bandyopadhyay, A. Mukherjee, and D. Saha, "Communication-aware mobile hosts in ad-hoc wireless network", Personal Wireless Communication, 1999 IEEE International Conference on, 1999, Page(s): 83 –87.

[7] S. Agarwal, A. Ahuja, J.P. Singh and R. Shorey, "Route-lifetime assessment based routing (RABR) protocol for mobile ad-hoc networks", Communications, 2000. ICC 2000. 2000 IEEE International Conference on Volume: 3, 2000, Page(s): 1697 -1701 vol.3.

[8] I. Stojmenovic and Xu Lin "Power-aware localized routing in wireless networks", Parallel and Distributed Processing Symposium, 2000. IPDPS 2000. Proceedings. 14th International, 2000, Page(s): 371 –376.

[9] Taek Jin Kwon; M. Gerla, "Clustering with power control", Military Communications Conference Proceedings, 1999. MILCOM 1999. IEEE Volume: 2, 1999, Page(s): 1424 -1428 vol.2.

[10] G. Holland and N. Vaidya, "Impact of routing and link layers on TCP performance in mobile ad hoc networks", Wireless Communications and Networking Conference, 1999. WCNC. 1999 IEEE, 1999, Page(s): 1323 -1327 vol.3.

[11] R. Prakash and M. Sahasrabudhe, "Modifications to TCP for improved performance and reliable end-to-end communication in wireless networks", Wireless Communications and Networking Conference, 1999. WCNC. 1999 IEEE, 1999, Page(s): 938 -942 vol.2.

[12] Jae-Hwan Chang and L. Tassiulas, "Energy conserving routing in wireless ad-hoc networks", INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE Volume: 1, 2000, Page(s): 22 -31 vol.1.

[13] Jung-Hee Ryu and Dong-Ho Cho "A new routing scheme concerning power-saving in mobile ad-hoc networks", Communications, 2000. ICC 2000. 2000 IEEE International Conference on Volume: 3, 2000, Page(s): 1719 -1722 vol.3.

[14] Andrew S. Tanenbaum, "Computer Networks", Prentice Hall, pp. 374-393.

[15] Chunhung Richard Lin; Jain-Shing Liu "QoS routing in ad hoc wireless networks", Selected Areas in Communications, IEEE Journal on Volume: 178, Aug. 1999, Page(s): 1426 –1438.

[16] Sung-Ju Lee, Mario Gerla and Chai-Keong Toh, " A simulation Study of Table-Driven and On-Demand Routing Protocols for Mobile Ad Hoc Networks", IEEE Network, vol. 13 4, July-Aug. 1999, pp. 48-54.

[17] David J. Goodman, "Wireless Personal Communications Systems", Addison-Wesley, Pp.356-360

[18] Tein-Yew Chung and Zheng-Ying Liu, "Stabilization Strategy and Routing for Ad-Hoc Wireless Networks", Institute of Electrical and Computer Engineering and Science Yuan-Ze University of Technology, 2000.