在連線機率已知的 Delay-Tolerance Network 的繞徑方法

連懷恩　　陳建　　黃嘉淵

交通大學資訊工程系　　交通大學資訊工程系　　工業技術研究院 資訊與通訊研究所

dep.cis93@nctu.edu.tw　　chienchen@cs.nctu.edu.tw　　riceiwang@itri.org.tw

摘要

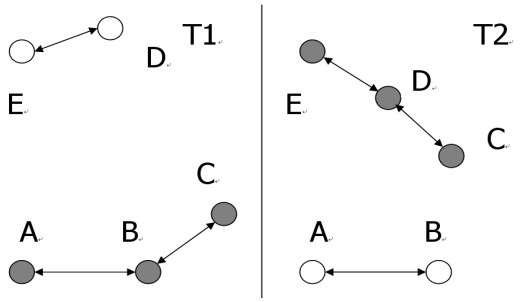
在連線狀況可預測的 delay-tolerance network 裡，可以很容易的尋找 delay 最短的封包傳遞路徑。可是在預測不夠精準、或連線品質不穩的情況下，找到的路徑往往無法保證符合需要的傳輸成功率。本篇即是在假設點與點之間連線成功機率已知的情況下，提出一個傳輸成功率符合要求，同時 delay 最短的繞徑演算法。我們的繞徑方法是，將每個時間區段的網路連線狀況用矩陣表示，利用類似矩陣相乘的方法，得出傳輸成功率符合要求的最短 delay 路徑。根據模擬實驗結果顯示，此繞徑法可以保證所找到的路徑，其傳輸成功率一定符合要求、同時比只考慮機率的繞徑法有 delay 更短的優勢。而事實上，這也是少見，能夠同時考慮兩種異質性變數的繞徑演算法。

關鍵詞：delay-tolerance network、routing、probability、delay

一、背景介紹

(一)何謂 delay-tolerance network

在一般的 wireless ad-hoc network 裡，routing protocol 都是假設 network 是連接的，即使 node 的移動會產生或中斷一些 link，但大部分的 node 都是 connected 的；但越來越多新的應用，由於 node 在稀疏網路中移動、環境改變、頻譜共用、或 sensor node 休眠節能等等的因素，使得 network 不總是 connected 的，node 與 node 之間的連線狀態會隨著時間改變，產生新的連線或中斷連線。如圖(一)所示，在這種情況下，node 往往會有一個 queue 來暫存封包，以備未來有新連線出現時可以將封包繼續 forward 下去。類似這樣，network 連線的狀況不穩定，每一次的 packet delivery 往往會有很長的 delay，這樣的 wireless ad-hoc network 就稱為 delay-tolerance networks [4] 、highly-partitioned networks、或 disruption- tolerant networks。而為了盡量提升 delay-tolerance network 的性能表現與提高繞徑成功率，除了有 queue 可以暫存封包外，其他常見的配套做法還包括適合各種特定情況的 routing protocol、網路編碼 [10],[6]、或根據網路連線拓撲的缺口，安置可自主移動的傳輸中繼站[3]等。



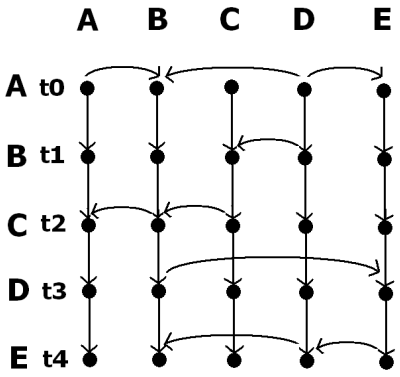
(圖一) 雖然 A 點和 E 點從未相遇，但 packet 卻可以從 A 傳遞到 E

(二)研究動機與目標

在 delay-tolerance network 裡，常常可以藉由物體的運動模式或統計過去的資料，來預測未來 network 的連線狀況，以此決定 packet routing 的路徑 [7]。例如人造衛星的軌道和運動速率不變，可以依此推斷出它進入傳輸範圍的時間；又如裝設在客運和公車上的 vehicular networks，因為班次和路線固定，所以從總站出發的班車可以預測，何時可

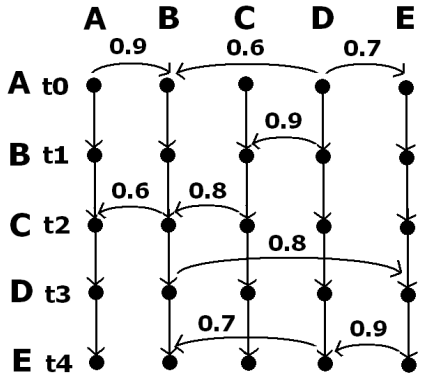
將更新的資訊散佈到所有營運中的其他班車。

一般來說，每個 delay-tolerance network 對未來連線狀況的預測，都可以用圖形的方式表示。如圖(二)所示，每個點都代表 network 中的每個 node，不同的橫排表示不同的時間區段，橫向的連線代表在該時間區段內 node 的連線，縱向的連線則代表每個 node 皆有 queue 可以暫存 packet 到下一個時間區段。之前被研究過的的繞徑方法，就是設定橫向連線的 weight 為 0 或很小的數，縱向連線的 weight 為 1，然後用 shortest path algorithm 去找出起點到終點間的最短 delay 路徑 [8]。



(圖二) 基本的 delay tolerance network 連線預測圖

但在大多數的狀況下，去預測未來連線的發生時間，往往是有誤差的；以公車上的 vehicular networks 為例，班車的位置會受到路況影響，連帶使得某一時間的預期連線機率不總是等於 1，而可能會是一個小於 1 的機率值。另外，有許多 delay-tolerance network 是處在嚴苛的環境或充滿不確定性的狀況下，往往無法確保它連線的品質，可以說，這種不確定性正是 delay-tolerance network 的特性之一。假設我們可以從統計或是其他的機率模型，估算連線之間的成功率，在加入了連線的機率值後，可以看到如圖(三)所示，每個橫向連線都加入了 0~1 之間的機率值。因此，我們的繞徑問題，就從單純只有 delay 的 weighted graph，變成同時要考慮 delay 和 probability 的 weighted graph。此時單純用 time domain 的 shortest delay path 找到的路徑，其成功傳遞封包到目的地的機率很可能會小於期望值。碰到這種情況時，傳統的方法是要找出所有可能的第二短路徑，從中找出最快然後又符合期望的機率值的路徑，若是找不到的話再依序找第三快、第四快，每次都要重新尋找路徑，代價非常高。另外，因為 delay-tolerance network 的 delay 很高，不像一般的 wireless network，如果傳輸失敗的話還可以馬上要求重傳，所以一開始就決定一條成功率夠大的路徑是非常重要的。這篇研究的目標，就集中在符合期望的傳輸成功率之下(例如大於等於 0.85)，找出 delay-tolerance network routing 的最快路徑。在這裡我們提出了以矩陣反覆相乘法來作為 delay-tolerance network 的繞徑演算法，它可以在網路連線機率已知的情況下，找出傳輸成功率符合要求、同時 delay 最短的路徑。由於它可以在 source node 時就決定出一條路徑，delay 的時間長短和可靠度都是可預期的，這使得 node 和 node 之間可以傳送大量連續性的資料，而不必等待遠方傳回的傳輸成功與否的回報，這是以往網路品質常常變動的 delay-tolerance network 很難做到的。此外，在好的網路連線狀況下，相較於過去在 stochastic delay-tolerance network 裡常用的 epidemic 或 broadcast routing [2],[9],[5]，在我們的矩陣反覆相乘法裡，點到點間的資料傳遞通常只需要一個封包就能達成，不但降低了 node 對 buffer 大小的需求，也使得 network traffic 的 overhead 下降。



(圖三) 加入機率值的連線預測圖

二、矩陣反覆相乘法

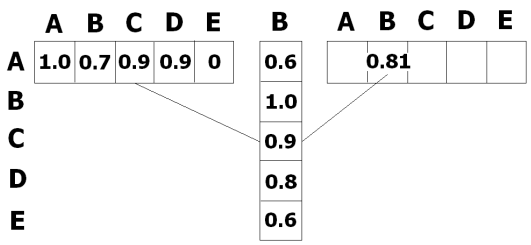
（一）基本概念

假設 network 中有 N 個 node，把剛剛的 network 連線狀況預測圖化成 matrix 的形式，每個時間區段、即每個橫排的連線狀況都建成一個 N x N 矩陣，如圖(四)，而一個擁有 N 個 node、T 個時間區段的預測圖就可以化成 T 個 N x N 的矩陣。其中矩陣中的每個元素，代表在該時間區段內，node 與 node 之間成功連線的機率。

接下來從 t0 矩陣開始，沿著時間軸的順序把矩陣和矩陣相乘起來，matrix(t0) * matrix(t1) * matrix(t2) *…..。和一般的矩陣乘法，每個元素是由 N 個 pair 相乘再加起來不同的是，這裡是取 N 個相乘的 pair 中最大的值，如此一來，每次的矩陣相乘皆代表從 t=0 開始，經過了一個新的時間區段後，node 與 node 間累積的連線成功機率。如圖(五)的例子，經過一個時間區段後，考慮 5 種從 A 到 B 的方式，分別為 AA*AB=0.6、AB*BB=0.7、AC*CB=0.81、AD*DB=0.72、AE*EB=0，其中最大機率的為 AC*CB 的 0.81。如此一直反覆做矩陣相乘，直到代表起點到目的地 node 的元素值大於所要求的成功傳輸機率值，而此時找到的傳輸路徑即為耗時最短、且成功機率符合要求的傳輸路徑。

	A	B	C	D	E
A	1.0	0.9	0	0	0
B	0	1.0	0	0	0
C	0	0	1.0	0	0
D	0	0.6	0	1.0	0.7
E	0	0	0	0	1.0

(圖四) 圖三在時間區段 t0 時的矩陣形式



(圖五) 從 5 個相乘的 pair 中取最大的機率值

（二）演算法詳述

整個演算法可以分成兩大部分：矩陣相乘迴圈和路徑回溯：先執行矩陣相乘迴圈，一直到目標元素的機率值大於或等於要求，接下來執行路徑回溯，找出完整的路徑資訊，如圖(六)。

```
t=0 , Floyd-Warshall 產生 Result(t)。
矩陣相乘迴圈 , While ( 目標元素機率值 < Threshold )。
                Floyd-Warshall 產生 Matrix(t+1)。
                Result(t+1) = Result(t) x Matrix(t+1)。
                t=t+1。
路徑回溯。
```

(圖六) 整個演算法流程的虛擬碼

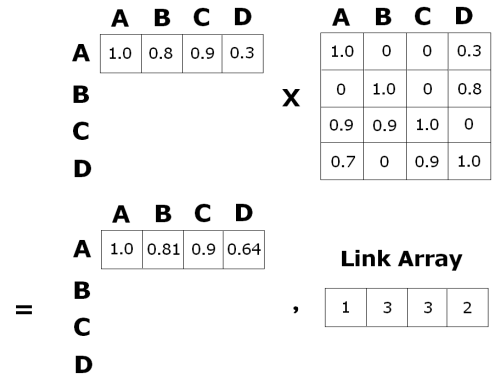
在矩陣相乘迴圈裡，每執行一次迴圈，就要找出下個時間區段的矩陣並和目前的矩陣相乘。由於在同一個時間區段裡，node 和 node 之間連線的方式不只一種，這裡需要機率最大的連線，所以每個時間區段都要做一次 all pair shortest path 來找出該時間區段的矩陣表示式。在這裡使用修改過的 Floyd-Warshall 演算法，把原本找最短路徑改成找最大機率路徑。就機率而言，從 A 點經 B 點再到 C 點的路徑機率值為 P(A→B)*P(B→C)。所以，修改過的 Floyd-Warshall recursion 如下：

$$\mathrm{p}_{ij}^{(k)} \begin{cases} p_{ij} & ,\text{if } k = 0 \\ \max(p_{ij}^{(k-1)}, p_{ik}^{(k-1)} * p_{kj}^{(k-1)}) & ,\text{if } k >= 1 \end{cases} \quad (1)$$

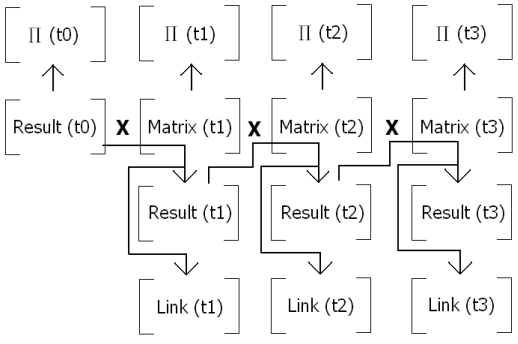
將 node 與 node 之間的最大機率連線機率值，填入對應的矩陣元素中，成為該時間區段的矩陣表示式。此外，在進行 Floyd-Warshall 演算法的同時，還會同時建立一個 predecessor matrix Π，用來記錄該矩陣每個元素的最大機率連線的詳細走法。

在用 Floyd-Warshall 找出每個時間區段的矩陣表示法後，接下來就要將這些矩陣，依著時間順序一一相乘。每次矩陣相乘的結果、也就是記錄目前累積結果的矩陣稱為 Result 矩陣，而相乘的流程也就是 Result(t) 矩陣相乘下個時間區段的矩陣 Matrix(t+1)，得到新的 Result(t+1)矩陣，代表從 t=0

到目前為止，起點到各個 node 的累積連線成功率。這裡要注意的是，因為要找的僅為起點到目的地的路徑，所以每次的矩陣相乘並不用做完所有的 N x N 個元素。例如起點若為 node A，那每次矩陣相乘都只需要將 Result(t)矩陣的第一個 row(代表起點為 A)，相乘 Matrix(t+1)矩陣的每個 column，取得新的 Result(t+1)矩陣的第一個 row。而每次矩陣相乘時，也會建立一個 1 x N 的 Link Array，紀錄新的 result(t+1)矩陣的每個元素，其機率值來源是 N 組相乘的 pair 的哪一組。這樣的矩陣相乘迴圈會一直反覆執行到目標元素機率值大於或等於要求為止。圖(七)為執行一次矩陣相乘的例子，假設起點為 A 點，Result(t) x Matrix(t+1) = Result(t+1)，及該次矩陣相乘所對應的 Link Array(t+1)，其中的 Link Array(1,3,3,2)則代表 Result(t+1)=(AA*AA, AC*CB, AC*CC, AB*BD)。圖(八)則為反覆執行迴圈的流程圖。



(圖七) 執行一次矩陣相乘的例子



(圖八) 矩陣相乘迴圈的展開圖

矩陣相乘迴圈結束後接著執行路徑回溯。回溯時，會交替的檢查Π matrix 和 link array，找出目的地元素機率值的來源。假設之前執行迴圈時從 t0 相乘到 t3，那回溯時就依序檢查Π(t3), link(t3), Π(t2), link(t2), Π(t1), link(t1), Π(t0)，將所經過的 node 填入一個 stack 中，填完後再一個一個 pop 出來，就可以得到最後的路徑。

(三) 演算法的時間複雜度

整個演算法的時間複雜度為 $O((N^3 + N^2)T + N)$，其中 N^3 為執行一次 Floyd-Warshall 演算法，N^2 為執行一次矩陣相乘，N 則為執行路徑回溯，T 則代表矩陣相乘迴圈執行的次數，相當於抵達目的地時的 delay 大小。顯然的，T 會受到連線的品質和網路連線的密度影響。

(四)多重路徑的延伸

雖然在能取得可靠的機率值前提下，矩陣反覆相乘法可以提供相當程度的傳輸可靠度，但在每次封包發送時皆採用單一路徑，若路徑上任何一個 node 發生能源耗盡、故障或是 buffer 已滿的情況，仍然會使得整個傳輸失敗。而在 delay-tolerance network 裡，常常會讓一次的傳輸，發送數個內容相同的封包來提高傳輸成功率。因此在有多餘可用頻寬的狀況下，一樣可以使用矩陣反覆相乘法，來為一次傳輸計算出多條有效的傳輸路徑。很直覺的方法是，在計算出第一條路徑後，把連線預測圖中，所有走過的橫向連線和 node 刪去，依此再用矩陣反覆相乘法計算第二適合的路徑…依此類推。如此一來，即可找到數條 disjoint 的最佳路徑，來進一步增加整體的可靠度。

三、實作與模擬

(一) 參與模擬的各種繞徑方法

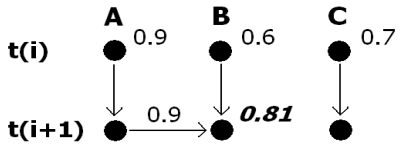
在模擬中，除了本文介紹的矩陣反覆相乘法之外，另外實做了兩組對照組，用來比較過往只單純考慮 delay 或只考慮機率，即只考慮預測圖裡縱向或橫向的 weight，其結果和同時考慮兩者的矩陣反

覆相乘法有何不同。為了在相同的條件狀況下比較，設定每次都只傳一個 packet，每個時間區段都只能有一個 node 暫存 packet。若 packet 抵達目的地時機率值不足則直接記錄下抵達時的機率值，不考慮重傳或其他補償機制。以下是各種方法的介紹：

(a)僅考慮 delay。在只考慮 delay 的情況下，使用 dalay dijkstra 演算法，一個一個時間區段檢查所有的路徑，若有路徑到達目的地即停止，則該路徑將會是 delay 最短的路徑。

(b)僅考慮 probability。在這裡並無法使用考慮整個連線預測圖的 probability dijkstra，因為 dijkstra 的使用前提是，已經拜訪過的 node，它目前的機率值即是最後的最大機率值，而這個前提在 delay-tolerance network 中並無法成立。如圖(九)，在 t(i)的時候，從起點到 B 和 C 的最大機率分別是是 0.6、0.7，這時顯然 C 點在矩陣相乘時會佔優勢，容易被選進路徑中；但經過了一個時間區段後，增加了一條 A 到 B 的連線，使得起點到 B 的最大機率提升為 0.81(矩陣法都只取最大值)，大於 C 的 0.7，這時很多原本以 C 為中繼點的路徑全部需要更新，破壞了原本 dijkstra 的運算規則。所以在這種情況下，該時間區段若無法直接找到抵達目的地的有效路徑，則暫存住 packet 等待下一個時間區段，顯然是比較明智的選擇，多一次無謂的傳輸反而是更降低整體的傳輸成功率。所以在只考慮 probability 的情況裡，source node 會一直暫存住 packet，直到在同一個時間區段裡，能找到傳輸機率值大於要求的路徑為止。

(c)同時考慮 delay 和 probability，即本文所提出的矩陣反覆相乘法，在一開始就決定整條 routing 的路徑。

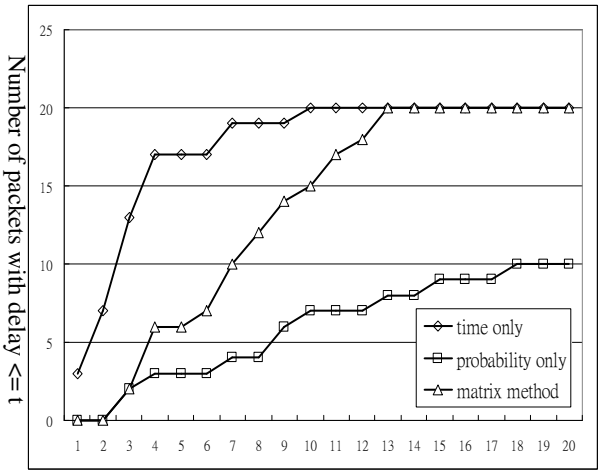


(圖九) 各個點的最大連線機率會一直更新，破壞原本 dijkstra 的運算規則

(二) 模擬的環境設定

我們用 C 語言寫出三組演算法，並做一個資料產生器來產生初始資料：設定有 32 個 node，20 個時間區段(t0~t19)，平均每個時間區段有大約 30 個單向連線。考慮到環境影響與每個 node 的能源狀況不同，每個 node 的傳輸範圍都是不固定的，即 A 到 B 與 B 到 A 的連線不一定同時存在。要求的傳輸成功率為 0.85，點到點之間的連線機率為 0.85~1 之間的 uniform distribution。也就是說，在我們的模擬環境中，一條路徑要被判定為成功，必須要同時滿足路徑機率>=0.85 及 delay<=19 的條件才行。

(三)模擬結果



(圖十) 所有抵達目的地的 packet，其 delay 對封包數的累積分布函數圖

圖(十)代表各種繞徑方法的 delay 大小比較，其中橫軸為 delay 的大小，縱軸為傳輸時間小於此 delay 的累積封包數。曲線越陡，代表越多封包越快抵達目的地，即 delay 越短。在這裡可以發現，delay 最短的是僅考慮 delay 的繞徑法，矩陣法次之，delay 最長的是僅考慮 probability 的繞徑法。其中僅考慮 probability 的繞徑法，由於一直找不到能在同一個時間區段符合要求傳輸機率的路徑，使得在預測時間結束前，仍有一半左右的封包沒有被發送出去。

(表一) 不管路徑成功與否，在給定的時間範圍內(t<=19)，所有能抵達目的地的路徑，其 delay 和 probability 的統計

	僅考慮 dalay	僅考慮 probability	矩陣法
抵達目的地的平均 delay	2.45	8.1	6.65
抵達目的地的平均 probability	0.698	0.916	0.885

表(一)統計所有在 20 個時間區段內抵達目的地的封包，並計算出它們的平均 delay 和路徑機率值。平均而言，delay 的長短仍是以僅考慮 delay 的繞徑法最佳、矩陣法次之，僅考慮 probability 的繞徑法最差。而抵達時的平均路徑機率，矩陣法和僅考慮 probability 的繞徑法皆能符合要求，其中又以僅考慮 probability 的方法，抵達時的平均路徑機率最高。注意到矩陣法的平均路徑機率值 0.885 僅比要求的機率值 0.85 高出 0.035，證明矩陣法的確能在要求的機率值和 delay 之間取得良好的平衡。

(表二)各種方法找到機率和 delay 同時符合要求的路徑的成功率

	僅考慮 dalay	僅考慮 probability	矩陣法
成功率	5%	45%	100%

如表(二)所示，若同時考慮進 delay 和 probability 的限制條件，矩陣法的成功率是 100%，或是說，只要符合限制條件的路徑存在，矩陣法就一定能找到。而僅考慮 probability 的繞徑法，因為 delay 有時會超出預測時間範圍，成功率大概只有一半；最糟的是僅考慮 delay 的繞徑法，找到的路徑，其機率值多半不符合要求，成功率只有 5%。

由模擬的結果可以發現，就 delay 長短而言，僅考慮 delay 的方法表現最好，僅考慮 probability 方法的最差。但是由於僅考慮 delay 的方法，抵達目的地時的路徑機率太低，達成率僅有 5%，所以說明了矩陣反覆相乘法，是在兼顧目標達成率與 delay 的狀況下，三種方法中最理想的方法。

四、結論

在現實生活中，對 delay-tolerance network 未來連線狀況的預測，往往很難達到 100%的可靠。現有在 delay-tolerance network 上的繞徑演算法，只能同時滿足 delay 或傳輸成功率的需求，因此會有 delay 太高或是傳輸成功率太低的缺點。在這裡，我們提出了以矩陣反覆相乘法來作為 delay-tolerance network 的繞徑演算法，它可以在網路連線機率已知的情況下，找出一條傳輸成功率符合要求、同時 delay 最短的路徑。而我們的模擬實驗結果也證明，它的確達成了我們一開始設定的目標，並在 delay 和和連線成功率間取得良好的平衡。也由於矩陣反覆相乘法是在一開始就決定封包傳遞的路徑，同一個封包大都也只需保留一份 copy，使得這個方法特別適合應用在有大量傳輸需求的 delay-tolerance network 上。當然，如何在各種 scenario 下，找到適合的機率統計或預估模型[1],[3]，是這個方法可以維持高可靠度的前提，也是未來我們努力的方向。

五、參考文獻

[1] A. Lindgren, et al " Probabilistic routing in intermittently connected networks. Mobile Computing and Communications Review, 7(3), July 2003.

[2] A. Vahdat and D. Becker, "Epidemic routing for partially connected ad hoc networks," Tech. Rep. CS-200006, Department of Computer Science, Duke University, Durham, NC, 2000.

[3] B. Burns, et al, "MV Routing and capacity building in disruption tolerant networks", IEEE INFOCOM 2005, Miami, FL, March, 2005.

[4] Delay-tolerant networking research group. http://dtnrg.org

[5] F. Tchakountio and R. Ramanathan, “Tracking highly mobile endpoints”, ACM Workshop on Wireless Mobile Multimedia (WoWMoM), July 2001, Rome, Italy.

[6] K. Harras, K. Almeroth and E. Belding-Royer, Delay Tolerant Mobile Networks (DTMNs): Controlled Flooding in Sparse Mobile Networks, IFIP 2005.

[7] Sushant Jain, Kevin Fall, Rabin Patra, “Routing in a Delay Tolerant Network”.

[8] Shashidhar Merugu, Mostafa Ammar, Ellen Zegura, “Routing in Space and Time in Networks with Predictable Mobility”.

[9] T. Small and Z. J. Haas, “The Shared Wireless Infostation Model - A New Ad Hoc Networking Paradigm (or Where there is a Whale, there is a Way)”, Mobihoc 2003, June 1-3, 2003.

[10] Yong Wang, et al, Erasure-Coding Based Routing for Opportunistic Networks. DTN workshop05.