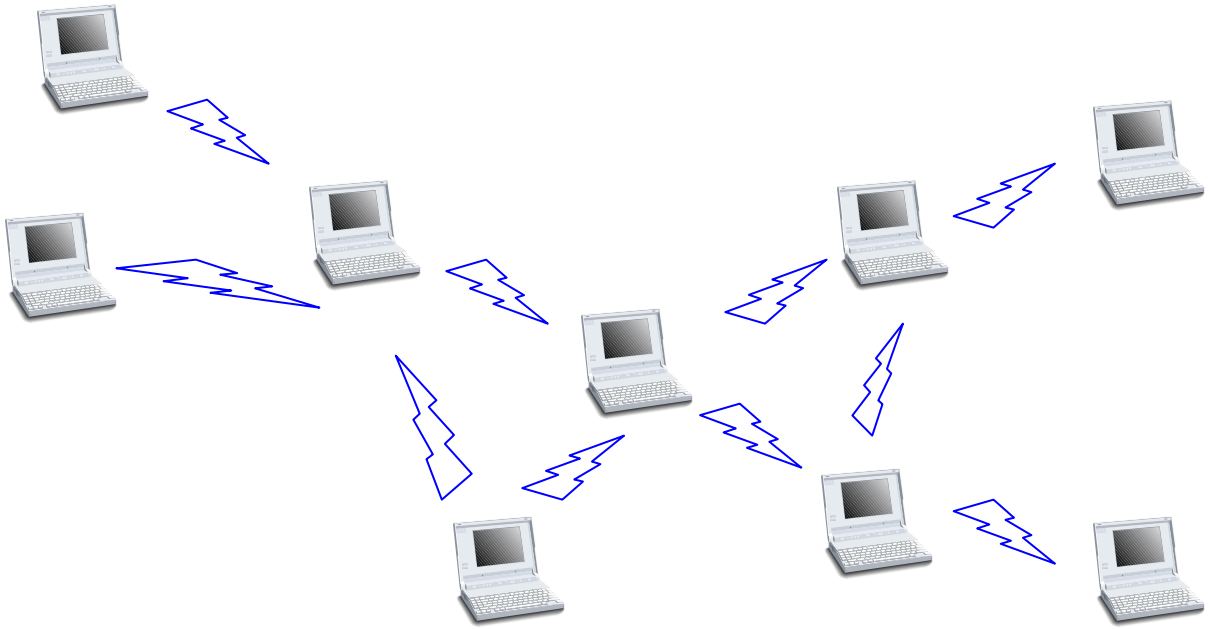
Chapter 8
無線隨意網路路由

簡介(一)

- 無線隨意網路：MANET = Mobile Ad Hoc Networks
  - a set of mobile hosts, each with a transceiver
  - no base stations; no fixed network infrastructure
  - multi-hop communication
  - needs a routing protocol which can handle changing topology



簡介(二)

- 無線網路不同於有線網路的特性：
  - 目的位址（Destination Address）不等於目的位置（Destination Location）。
  - 無線訊號強度會因處於不同環境而有不同程度之衰減 。
  - 無線網路具有動態網路拓撲之結構 。
  - 同屬一個無線網路集合將會共用同一個傳輸媒介 。
  - 資料傳輸過程未受保護，故易受外界干擾且保密安全性低。
  - 省電之課題
  - 具有分散式執行功能。
  - 無迴路（Loop-free）結構。
  - 具有多重中繼段（Multi-hop）路由。
  - 對動態鏈結作快速地適應。
  - 不可靠的廣播機制。

簡介(三)

- 無線網路路由通訊協定在功能與設計之考慮：
  - 路徑的建立是主動或被動 。
  - 是否需週期性更新路徑 。
  - 是否需維護多條路由 。
  - 網路架構為叢集式或平面式。
  - 是否提供安全機制。
  - 是否支援群播功能 。
  - 建立路徑所需花費之時間 。
  - 工作站電源管理的考量 。
  - 通訊頻寬的需求 。
  - 所需之參考資訊表格數量 。

簡介(四)

- 兩種路由方法
- 表格驅動式路由 (Table Driven Routing)
  - 主動式 Proactive 路由
- 需求式路由 (On-demand Routing)
  - 回應式 Reactive 路由

簡介(五)

- 表格驅動式路由 (Table Driven Routing)：每一個節點（工作站）都會各自維護自己的路由表（Routing Table），再利用各個節點週期性地廣播自己的路由資訊，來彼此交換與更新訊息並藉此得以主動發現所需之封包傳送路徑。
  - 循序的目的地距離向量路由協定（Destination-sequenced Distance-vector Routing，DSDV）。
  - 群首閘道交換路由協定（Clusterhead Gateway Switch Routing，CGSR）。
  - 無線路由協定（Wireless Routing Protocol，WRP）。

簡介(五)

- 需求式路由 (On-demand Routing)：當有某一節點需要一條路徑以傳送封包給目的節點時才開始建構。而各節點本身並不會週期性與主動性地傳送路由更新資訊給網路上其他節點 。
  - 輕量移動路由協定（Lightweight Mobile Routing，LMR）。
  - 暫時性順序路徑演算法 （Temporally-ordered Routing Algorithm，TORA）。
  - 動態來源端路由協定 （Dynamic Source Routing，DSR）。
  - 無基礎式需求距離向量路由協定 （Ad Hoc On-demand Distance Vector Routing Protocol，AODV） 。
  - 關聯性基礎路由協定 (Associativity-based Routing， ABR) 。
  - 輔助定位路由通訊協定 (Location-aided Routing ，LAR) 。

表格驅動式路由演算法

- 表格驅動式路由協定有兩類主要的演算法：
  - 距離向量路由（Distance-vector Routing，DVR）。
  - 鏈路狀態路由（Link-state Routing，LSR）。

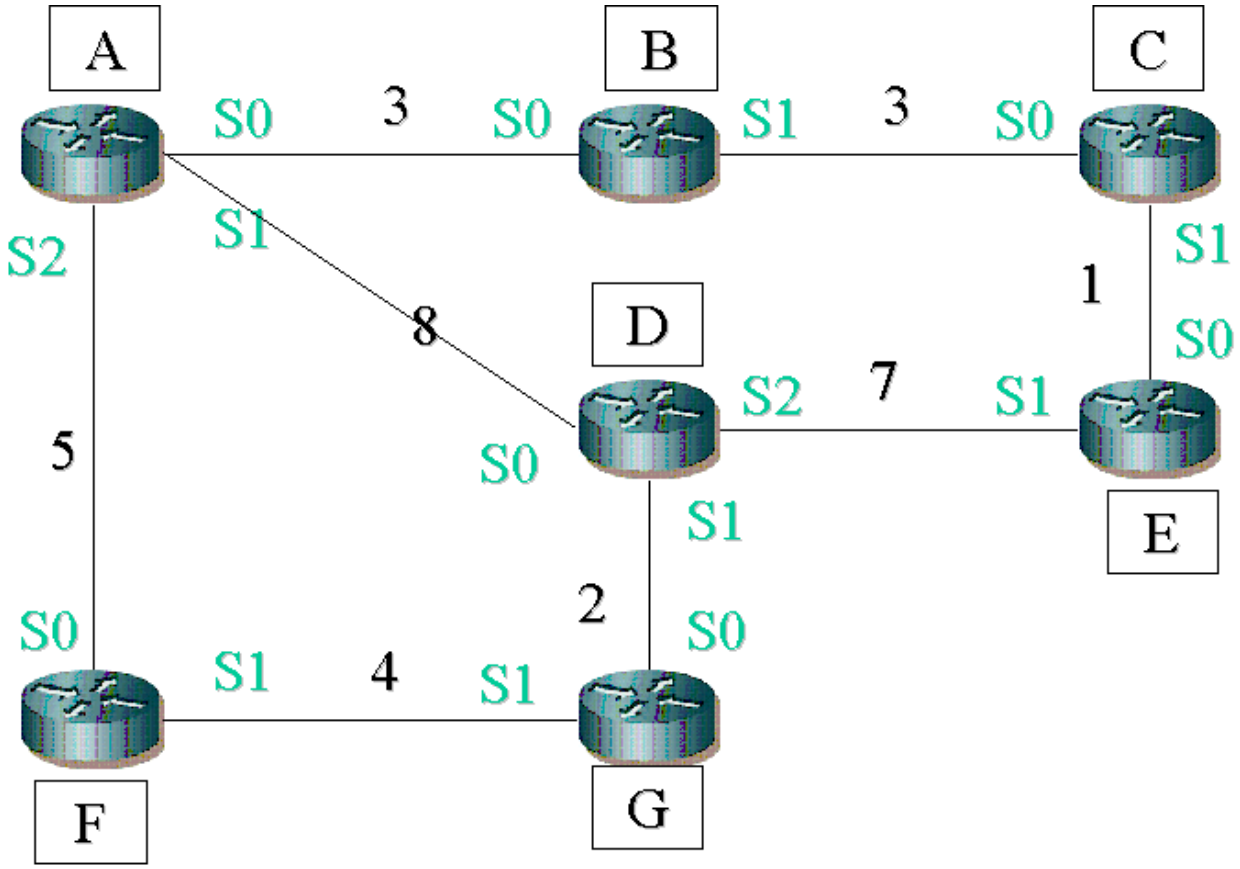
距離向量（Distance）與鏈結狀態（Link state）路由演算法

- 距離向量與鏈結狀態演算法最主要的差別：
  - 距離向量演算法需藉由其鄰近節點所告知之〝距離〞資訊做為其是否選擇該節點為路徑的依據
  - 鏈結狀態演算法則是每一節點經由鄰近節點的告知，獲得所有網路〝鏈結狀態〞的資訊（亦即擁有網路的全貌）後，再做為選徑的依據

距離向量演算法 (DVR)

- 使用距離向量的路由器都會自行維護一個路由表，他會記錄到每一個已知目的地的最佳距離到自己的路由表中
- 最典型的代表就是Bellman-Ford Routing Algorithm
- 當機器開機時，路由器中只有自己本身的路由資訊，然後會和鄰近的路由器以廣播的方式，彼此交換自己的路由表內的資訊，藉由這樣的交換訊息，很快的就可以得知整個網路的連結狀態

Example



Example

From A to	Interface	Metric
A	local	0

From B to	Interface	Metric
B	local	0

From C to	Interface	Metric
C	local	0

From D to	Interface	Metric
D	local	0

From E to	Interface	Metric
E	local	0

From F to	Interface	Metric
F	local	0

From G to	Interface	Metric
G	local	0

From A to	Interface	Metric
A	local	0
B	S0	1
D	S1	1
F	S2	1

From B to	Interface	Metric
B	local	0
A	S0	1
C	S1	1

From C to	Interface	Metric
C	local	0
B	S0	1
E	S1	1

From D to	Interface	Metric
D	local	0
A	S0	1
E	S2	1
G	S1	1

From E to	Interface	Metric
E	local	0
C	S0	1
D	S1	1

From F to	Interface	Metric
F	local	0
A	S0	1
G	S1	1

From G to	Interface	Metric
G	local	0
D	S0	1
F	S1	1

From A to	Interface	Metric
A	local	0
B	S0	1
D	S1	1
F	S2	1
C	S0	2
E	S1	2
G	S1	2

From B to	Interface	Metric
B	local	0
A	S0	1
C	S1	1
D	S0	2
F	S0	2
E	S1	2

From C to	Interface	Metric
C	local	0
B	S0	1
E	S1	1
A	S0	2
D	S1	2

From D to	Interface	Metric
D	local	0
A	S0	1
E	S2	1
G	S1	1
B	S0	2
F	S0	2
C	S2	2

From E to	Interface	Metric
E	local	0
C	S0	1
D	S1	1
B	S0	2
A	S1	2
G	S1	2

From F to	Interface	Metric
F	local	0
A	S0	1
G	S1	1
B	S0	2
D	S0	2

From G to	Interface	Metric
G	local	0
D	S0	1
F	S1	1
A	S0	2
E	S0	2

From A to	Interface	Metric
A	local	0
B	S0	1
D	S1	1
F	S2	1
C	S0	2
E	S1	2
G	S1	2

From B to	Interface	Metric
B	local	0
A	S0	1
C	S1	1
D	S0	2
F	S0	2
E	S1	2
G	S0	3

From C to	Interface	Metric
C	local	0
B	S0	1
E	S1	1
A	S0	2
D	S1	2
F	S0	3
G	S0	3

From D to	Interface	Metric
D	local	0
A	S0	1
E	S2	1
G	S1	1
B	S0	2
F	S0	2
C	S2	2

From E to	Interface	Metric
E	local	0
C	S0	1
D	S1	1
B	S0	2
A	S1	2
G	S1	2
F	S1	3

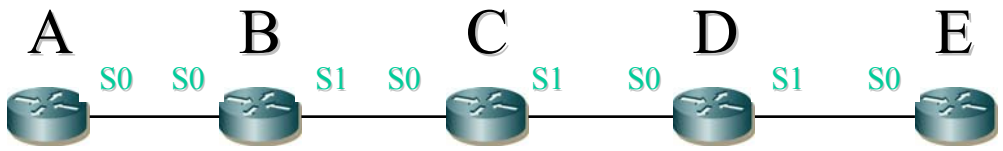
From F to	Interface	Metric
F	local	0
A	S0	1
G	S1	1
B	S0	2
D	S0	2
C	S0	3
E	S1	3

From G to	Interface	Metric
G	local	0
D	S0	1
F	S1	1
A	S0	2
E	S0	2
B	S0	3
C	S0	3

距離向量演算法-無限計數

- 因為每一網路節點不需掌握網路整體拓撲，因此在選徑時就有可能會有形成迴圈的現象，甚至有可能會使得路徑的選擇產生無限計數（Count to infinity）的問題

經一段時間後，路由器各自之完整的路由表如下：



From A to	Interface	Metric
A	local	0
B	S0	1
C	S0	2
D	S0	3
E	S0	4

From B to	Interface	Metric
A	S0	1
B	local	0
C	S1	1
D	S1	2
E	S1	3

From C to	Interface	Metric
A	S0	2
B	S0	1
C	local	0
D	S1	1
E	S1	2

From D to	Interface	Metric
A	S0	3
B	S0	2
C	S0	1
D	local	0
E	S1	1

From E to	Interface	Metric
A	S0	4
B	S0	3
C	S0	2
D	S0	1
E	local	0

假設A-B之間的鏈路斷了，B無法到達A，但是B卻從C得知有一條到A的路徑，於是B就更新自己的路由表，如下表

From B to	Interface	Metric
A	S1	3
B	local	0
C	S1	1
D	S1	2
E	S1	3

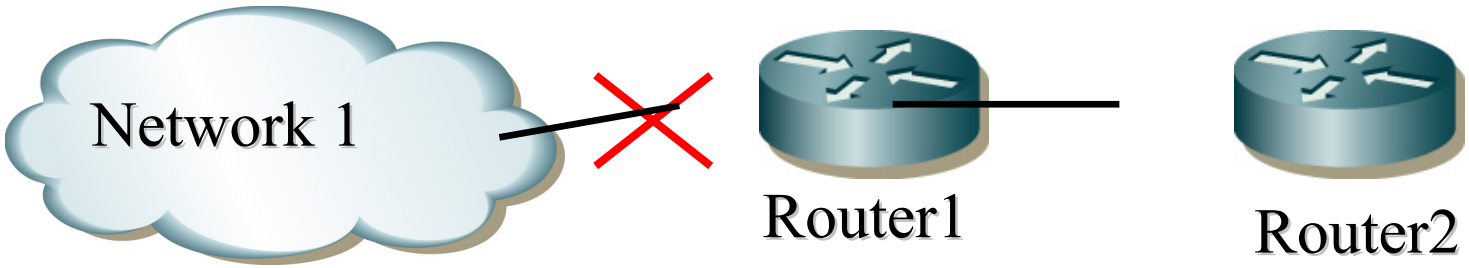
然而過一段時間，C 必須再次跟鄰居交換訊息，發現到 A 這個 entry 必須更新，於是 C 就更新自己的路由表，如下表

From C to	Interface	Metric
A	S0	4
B	S0	1
C	local	0
D	S1	1
E	S1	2

距離向量演算法

- 解決無限計數的方法：
  - **Maximum hop count**：限制其跳躍數
  - **Split horizon**：這個方法的原則是不回送資料，所以當A-B之間斷了，C不會再把A的訊息告訴B，因為到A的訊息是由B告訴C的
  - **Hold Down**： 利用Hold down timer，若hold down timer過期前的任何時間裡，從不同的路由器接到更新資訊，但指標較差，則不理會更新資訊。在hold down timer時效範圍內，不理會指標較差的更新資訊，則能有更多時間將中斷的改變資訊傳播到整個網路上。

Split horizon



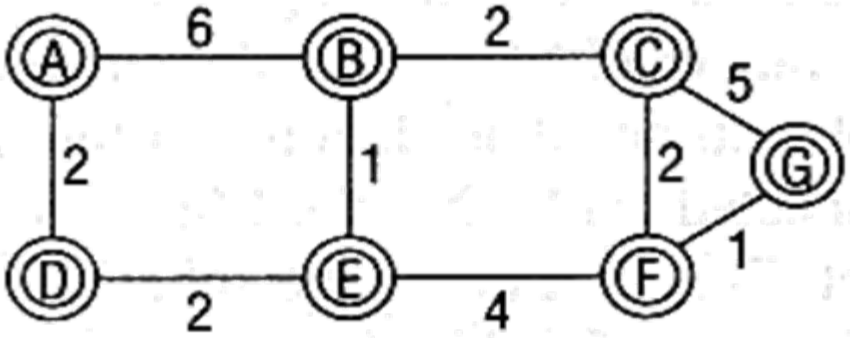
Router2不能告訴Router1有關Network 1的資訊，因爲那是Router 1告訴Router 2的。

鏈結狀態路由演算法

- 鏈結狀態演算法在一般圖形理論中也稱為最短路徑（Shortest Path First， *SPF*）演算法，它保持一份網路整體拓撲的資料庫。 再利用圖形理論中的最短路徑演算法找出該節點到各點之最短路徑
- 圖形理論中，最短路徑的演算法有許多種，但其中以Dijkstra演算法最有名，而Dijkstra演算法也被引用為OSPF（Open Shortest Path First）演算法中

鏈結狀態路由演算法：Dijkstra's Algorithm

Initial State: each host only knows its direct neighbors



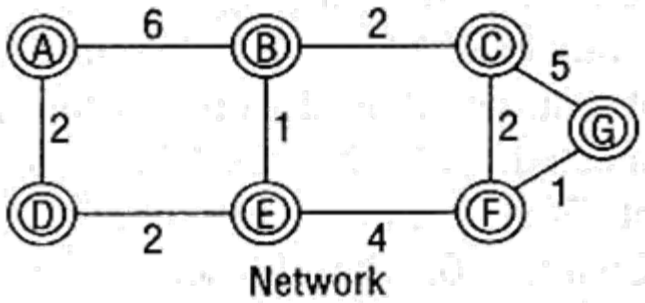
Network

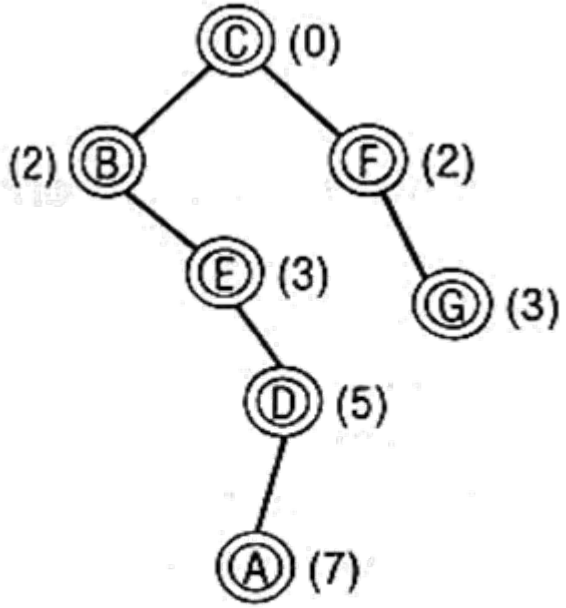
A	B	C	D	E	F	G
B/6 D/2	A/6 C/2 E/1	B/2 F/2 G/5	A/2 E/2	B/1 D/2 F/4	C/2 E/4 G/1	C/5 F/1

Link State Database

Figure 9.13

Evolution of States in C





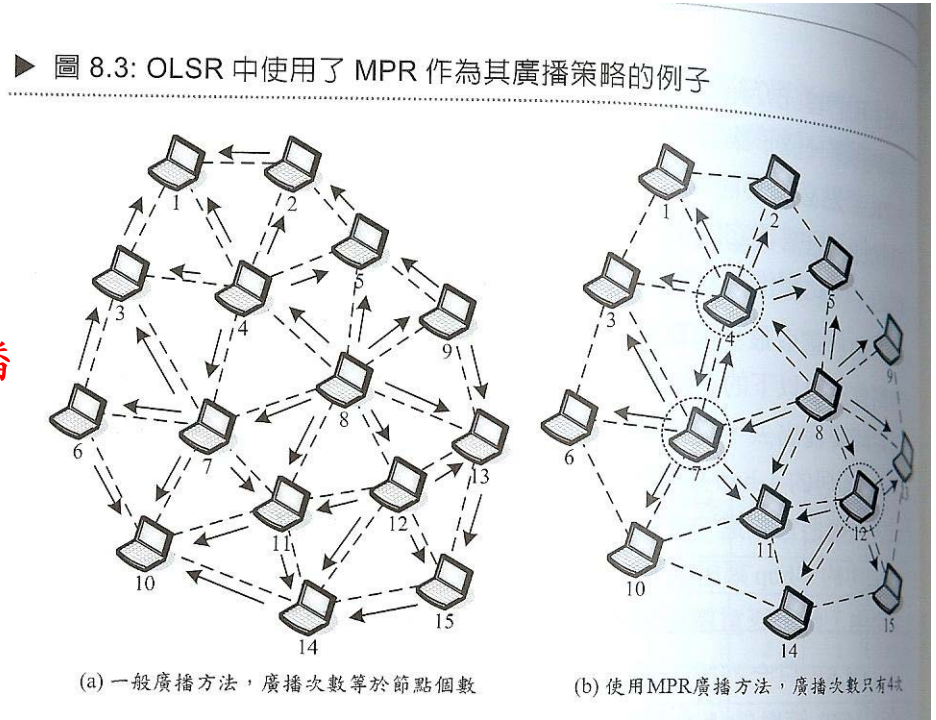
7. Place A in path.
   Examine A's LSP.
   No nodes left.
   Terminate.

- Comments: This is a centralized algorithm, not appropriate.

鏈結狀態路由演算法

- Optimal Link State Routing (OLSR)：
  - OSPF最佳化版本
  - 使用多重傳遞節點(MPR) 減少鏈結狀態更新的封包數量
  - MPR:挑選1-hop鄰居，使得能夠完全覆蓋全部2-hop鄰居

15次廣播



4次廣播

鏈結狀態路由演算法

- 鏈結狀態演算法有下列優點：
  - 無迴圈（Loopless）：路徑的決定是在擁有整個網路資訊後才決定，因此不會找出有迴圈的路徑
  - 多路徑（Multiple paths）到目的地：既然每一節點能獲知網路全貌，因此每一節點可依需要找出到目的地且不共用鏈結之多條路徑，而找出多條路徑的目的，可以是為了做路徑保護（備用路徑），或是分擔訊務使用

表格驅動式路由協定：
循序的目的地距離向量路由協定 (DSDV)

- DSDV：針對DVR進行修正而得。
  - DSDV多了目的地循序號碼（Destination-sequenced Number）的記錄。利用目的地循序號碼來判斷路徑是否需要更新，避免因產生迴路效應而使路徑發生錯誤 。

表格驅動式路由協定：
循序的目的地距離向量路由協定 (DSDV)

目的地 Destination	下一個節點 Next Hop	路徑節點數 Metric	循序號碼 Sequence Number	第一次相連時間 Install

DSDV通訊協定中每一個節點內都必須存有一個路由表

- 目的地(Destination)：可到達之節點。
- 下一個節點(Next Hop)：在到達某一目的節點之路徑上的第一個相鄰節點。
- 所需節點數(Metric)：到達目的節點中間所會經過的節點個數。
- 循序號碼(Sequence Number)：每個節點進行週期性資訊時所附上之號碼。初始時之編號是隨機產生，而後每次進行更新時號碼會加二。
- 第一次相連時間(Install)：各個節點第一次與其他節點進行通訊時所記錄之時間。

表格驅動式路由協定：
循序的目的地距離向量路由協定 (DSDV)

- 當網路拓撲結構有改變的時候，皆透過週期性的傳送廣播封包來告知新的路由資訊 。
- 這些路由資訊中不會改變的部分包含每一個節點第一次與其他節點通訊之時間 (Install)。
- 當某一節點在進行路由表更新時，對於具有相同目的節點之數條路徑：
  - 此節點通常會選擇具有最新循序號碼之路徑。
  - 若是遇到具有相同循序號碼之路徑，則將會選擇具有較少中繼節點數者。

表格驅動式路由協定：循序的目的地距離向量路由協定 (DSDV)

- 在動態網路中，擁有鏈結（Links）的節點可能會移動。
- 其鄰近之節點若是在經過一段時間內沒再收到此一節點的更新訊息時，其鄰近節點會認為此一節點已經離開，則這些鄰近節點會將各自的路由表中凡與此移動節點有關之路徑的路徑節點數設成無限大，並建立一個新的循序號碼將其廣播出去。
- 一般節點所發出的循序號碼都會被設定為偶數，而擁有無限大路徑節點數的循序號碼會被設定為奇數。

表格驅動式路由協定：循序的目的地距離向量路由協定 (DSDV)

- 基本上每一個節點都會同時維護兩個路由表:
  - 一個是自己建立路徑時所需之路由表。
  - 另一個則是要廣播路由更新資訊給相鄰節點用的路由表。
- 每一個節點為了確保自己可以收到最好的路徑，都會保留之前的路徑資訊。
- 在DSDV通訊協定中，採用具較少中繼節點數者來更新自己路由表之方式來防止路徑迴圈發生。
- 路由表中的節點都會是以最短路徑（Shortest Path）的方式到達目的端節點。

表格驅動式路由協定：
循序的目的地距離向量路由協定 (DSDV)

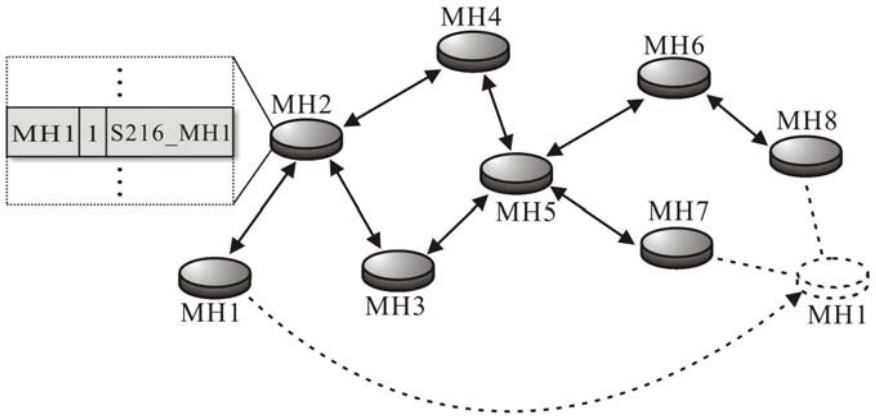
- 在DSDV通訊協定中，為防止各節點內之路由表不會常常被更改，而使網路較容易穩定下來 之作法：
  - 阻尼變動（Damping Fluctuations）：是描述如何去設置一個時間表，以防止路由表不斷的被改變。
    - 讓某些節點在接收到類似這樣的訊息封包後延遲將它送出。
    - 此延遲之時間最好大於兩個平均散佈之時間。
    - 所謂的平均散佈的時間為網路內每個節點傳送訊息給其它結點所需之時間平均。

表格驅動式路由協定：
循序的目的地距離向量路由協定 (DSDV)

- Example: 節點MH1在無線網路當中移動之情形。

目的地 Destination	下一個節點 Next Hop	路徑節點數 Metric	循序號碼 Sequence number	第一次相連時間 Install
MH_1	MH_2	2	$S216\_MH_1$	$T001\_MH_4$
MH_2	MH_2	1	$S328\_MH_2$	$T001\_MH_4$
MH_3	MH_2	2	$S546\_MH_3$	$T001\_MH_4$
MH_4	MH_4	0	$S602\_MH_4$	$T001\_MH_4$
MH_5	MH_5	1	$S312\_HM_5$	$T002\_MH_4$
MH_6	MH_5	2	$S056\_MH_6$	$T001\_MH_4$
MH_7	MH_5	2	$S128\_MH_7$	$T002\_MH_4$
MH_8	MH_5	3	$S056\_MH_8$	$T002\_MH_4$

MH4初始的路由表

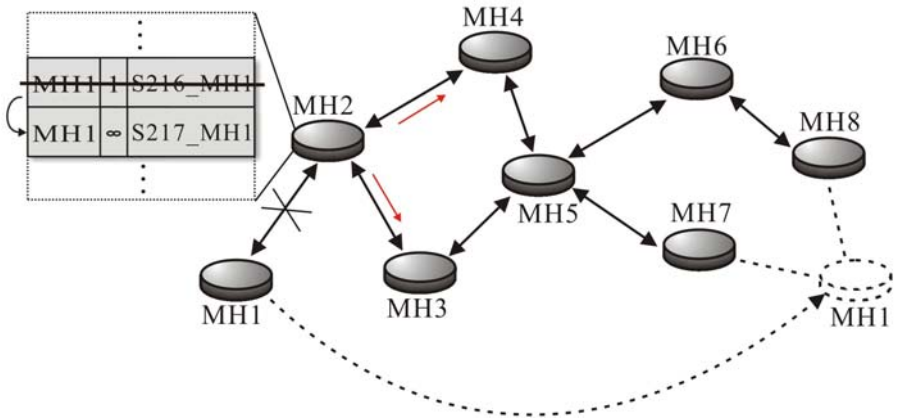


節點MH1在Ad Hoc網路中移動之情形

表格驅動式路由協定：
循序的目的地距離向量路由協定 (DSDV)

目的地 Destination	路徑節點數 Metric	循序號碼 Sequence number
MH_1	2	$S216\_MH_1$
MH_2	1	$S328\_MH_2$
MH_3	2	$S546\_MH_3$
MH_4	0	$S602\_MH_4$
MH_5	1	$S312\_HM_5$
MH_6	2	$S056\_MH_6$
MH_7	2	$S128\_MH_7$
MH_8	3	$S056\_MH_8$

MH4一開始所送出的更新路由資訊



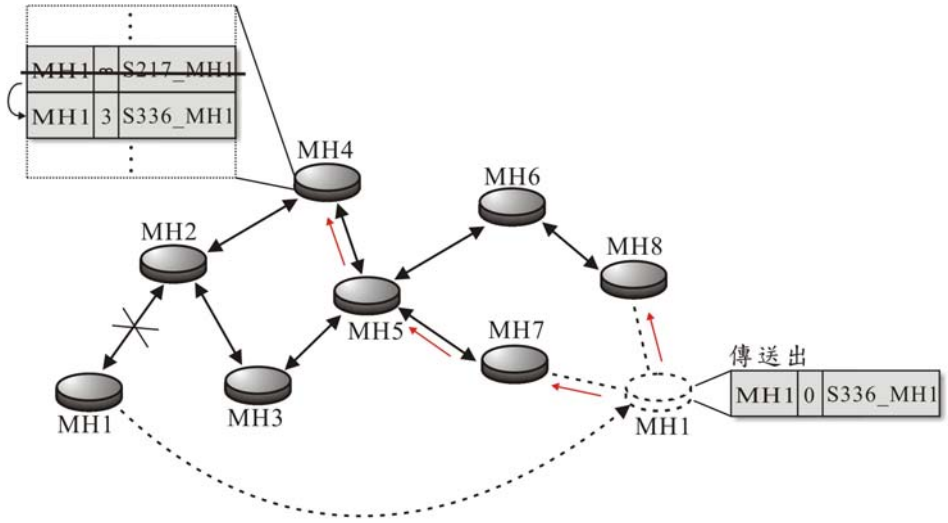
節點MH1移動至節點MH7與MH8附近

- MH2發現MH1離開後，其將自行更改到MH1的路徑節點數為無限大，並且更改其循序號碼為奇數。同時MH2會將這個消息廣播出去讓整個網路節點都知道MH2現在與MH1已經斷訊。

表格驅動式路由協定：
循序的目的地距離向量路由協定 (DSDV)

目的地 Destination	下一個節點 Next Hop	路徑節點數 Metric	循序號碼 Sequence number	第一次相連時間 Install
MH_1	MH_2	∞	$S217\_MH_1$	$T001\_MH_4$
MH_2	MH_2	1	$S328\_MH_2$	$T001\_MH_4$
MH_3	MH_2	2	$S546\_MH_3$	$T001\_MH_4$
MH_4	MH_4	0	$S602\_MH_4$	$T001\_MH_4$
MH_5	MH_5	1	$S312\_HM_5$	$T002\_MH_4$
MH_6	MH_5	2	$S056\_MH_6$	$T001\_MH_4$
MH_7	MH_5	2	$S128\_MH_7$	$T002\_MH_4$
MH_8	MH_5	3	$S056\_MH_8$	$T002\_MH_4$

節點MH4收到MH2發出與MH1中斷連線之封包後的更新路由表



節點MH1移動至MH7與MH8附近

表格驅動式路由協定：
循序的目的地距離向量路由協定 (DSDV)

目的地 Destination	下一個節點 Next Hop	路徑節點數 Metric	循序號碼 Sequence number	第一次相連時間 Install
MH_1	MH_5	3	$S336\_MH_1$	$T510\_MH_4$
MH_2	MH_2	1	$S448\_MH_2$	$T001\_MH_4$
MH_3	MH_2	2	$S666\_MH_3$	$T001\_MH_4$
MH_4	MH_4	0	$S722\_MH_4$	$T001\_MH_4$
MH_5	MH_5	1	$S432\_MH_5$	$T002\_MH_4$
MH_6	MH_5	2	$S176\_MH_6$	$T001\_MH_4$
MH_7	MH_5	2	$S238\_MH_7$	$T002\_MH_4$
MH_8	MH_5	3	$S176\_MH_8$	$T002\_MH_4$

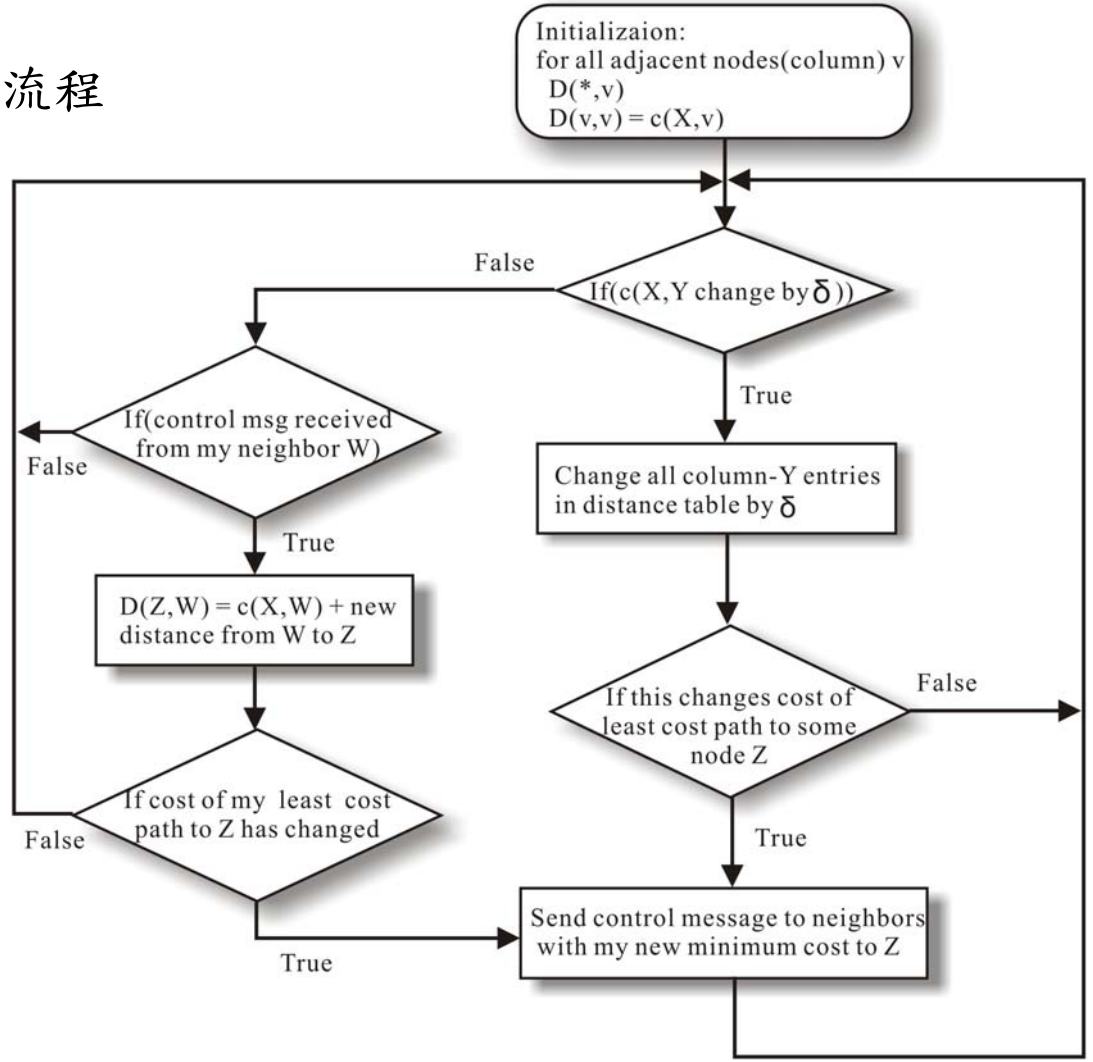
節點MH4再次更新後的路由表

目的地 Destination	路徑節點數 Metric	循序號碼 Sequence number
MH_4	0	$S722\_MH_4$
MH_1	3	$S336\_MH_1$
MH_2	1	$S448\_MH_2$
MH_3	2	$S666\_MH_3$
MH_5	1	$S432\_MH_5$
MH_6	2	$S176\_MH_6$
MH_7	2	$S238\_MH_7$
MH_8	3	$S176\_MH_8$

節點MH4在MH1移動至MH7跟MH8附近後所送出的更新路由資訊

表格驅動式路由協定：循序的目的地距離向量路由協定 (DSDV)

DSDV協定的處理流程



表格驅動式路由協定：群首閘道交換路由協定 (CGSR)

- 以DSDV為基礎並結合叢集路由機制設計而成的。
- 在多重中繼段和移動式無線網路中，讓節點加入某一叢集中，並受到此叢集之群首（Clusterhead）的控制。
- 在叢集式架構中會使用分散式演算法（Distributed Algorithm），來選出某一個節點作為某一叢集之群首。
- 在一群首通訊範圍內的所有節點均可加入此一叢集，而成為此叢集之成員。
- 在叢集中之所有節點均可以與群首直接進行通訊。
- 一個叢集中之節點可分為三種角色：
  - 群首（Clusterhead）。
  - 閘道（Gateway）。
  - 一般節點（Node）。

表格驅動式路由協定：群首閘道交換路由協定 (CGSR)

- **演算法**：
  - 一個叢集的複雜度（Complexity）主要取決於「群首選擇」的方法：
  - 一般有兩種分散式叢集演算法可採用：
    - 最低識別碼叢集演算法（Lowest-ID Clustering Algorithm）。
    - 最高連通性叢集演算法（Highest-connectivity（Degree）Clustering Algorithm）。
  - 無線網路中，維護一個叢集最重要的準則是穩定性，倘若群首太常變更將不利於此叢集之效能。
  - 討論另一種演算法：LCC叢集演算法（Least Cluster Change Clustering Algorithm）。
  - 此演算法只有在兩種情況下才會導致群首的變更：
    - 當某一個群首進入另外一個群首的通訊範圍時。
    - 當某一個群首脫離所有叢集的通訊範圍時。

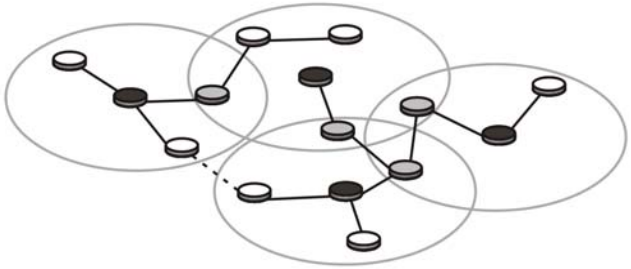
表格驅動式路由協定：群首閘道交換路由協定 (CGSR)

- LCC演算法：
  - 先採用「最低識別碼」（Lowest-ID）或「最高聯通性」（Highest-connectivity）叢集演算法，來創造一個初始的叢集。
  - 當某一個原屬於叢集A的非群首節點移動到叢集B時，則叢集A和B中之群首並不會改變。（只是叢集中之成員會變少或變多）
  - 若某一個非群首節點離開它的叢集，且沒有進入到其它叢集時（即指未進入其他叢集之通訊範圍內），則它將會成為一個新的群首，並形成一個新的叢集。
  - 當叢集A之群首節點離開叢集A並進入到叢集B時，則此一進入之群首節點將會和原本叢集B之群首節點互相競爭。再根據「最低識別碼」或「最高聯通性」叢集演算法（或其他較好的優先權方案）來競爭，勝出者將成為此叢集B的新群首。
  - 當叢集A之群首節點離開叢集A時，則此叢集A將會根據「最低識別碼」或「最高聯通性」叢集演算法，重新計算以產生新的群首。

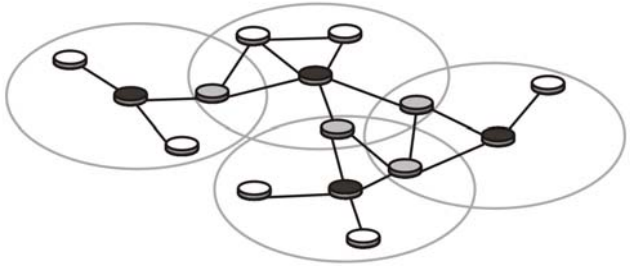
表格驅動式路由協定：群首閘道交換路由協定 (CGSR)

- 某兩個節點彼此停留在彼此的通訊範圍內且有穩定的訊號能源，則此兩個節點間會形成鏈結（Link）。
- 每個叢集都會有屬於自己的傳輸代碼（Transmission Code），且每一個節點都會記錄著自己所屬叢集的傳輸代碼。
- 兩個節點會因為彼此的傳輸代碼不同，而無法互相傳輸。
- 定義上述兩個使用不同代碼的節點，彼此間之鏈結為偽鏈結（Pseudo Links），並且從連線狀態中移除。
- 群首將會與叢集中的每一個成員進行鏈結 。

表格驅動式路由協定：
群首閘道交換路由協定 (CGSR)



在形成叢集之前的各節點鏈結情形



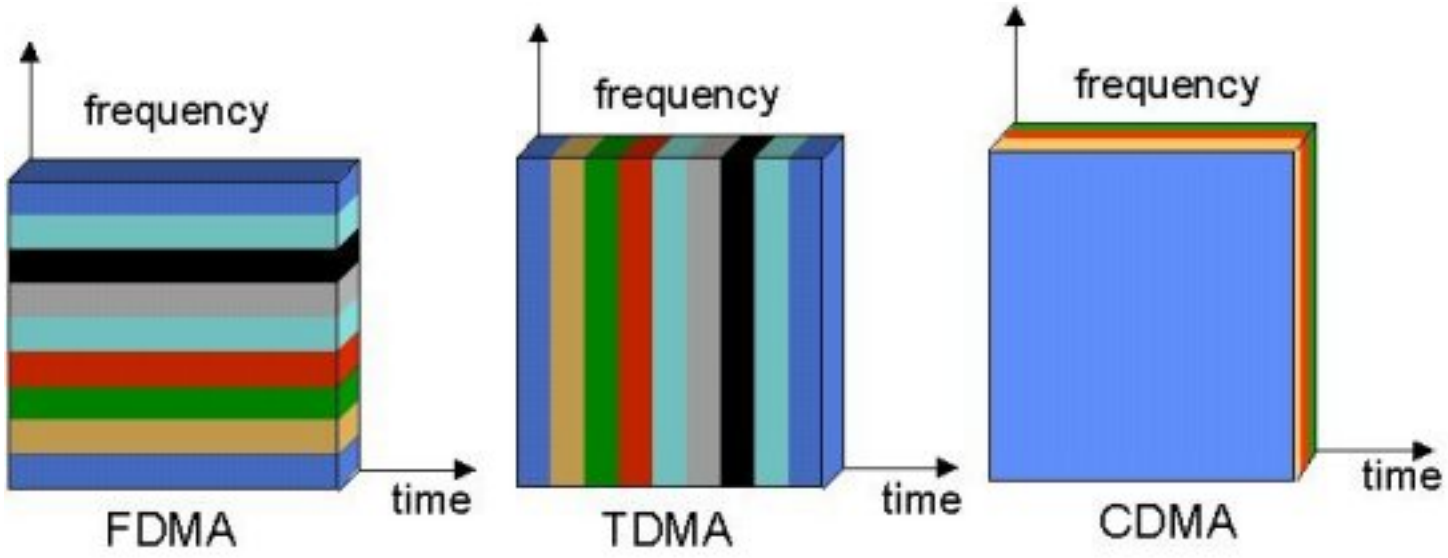
形成叢集後的各節點鏈結情況（黑點代表群首，灰點代表閘道節點，白點代表一般節點）

表格驅動式路由協定：群首閘道交換路由協定 (CGSR)

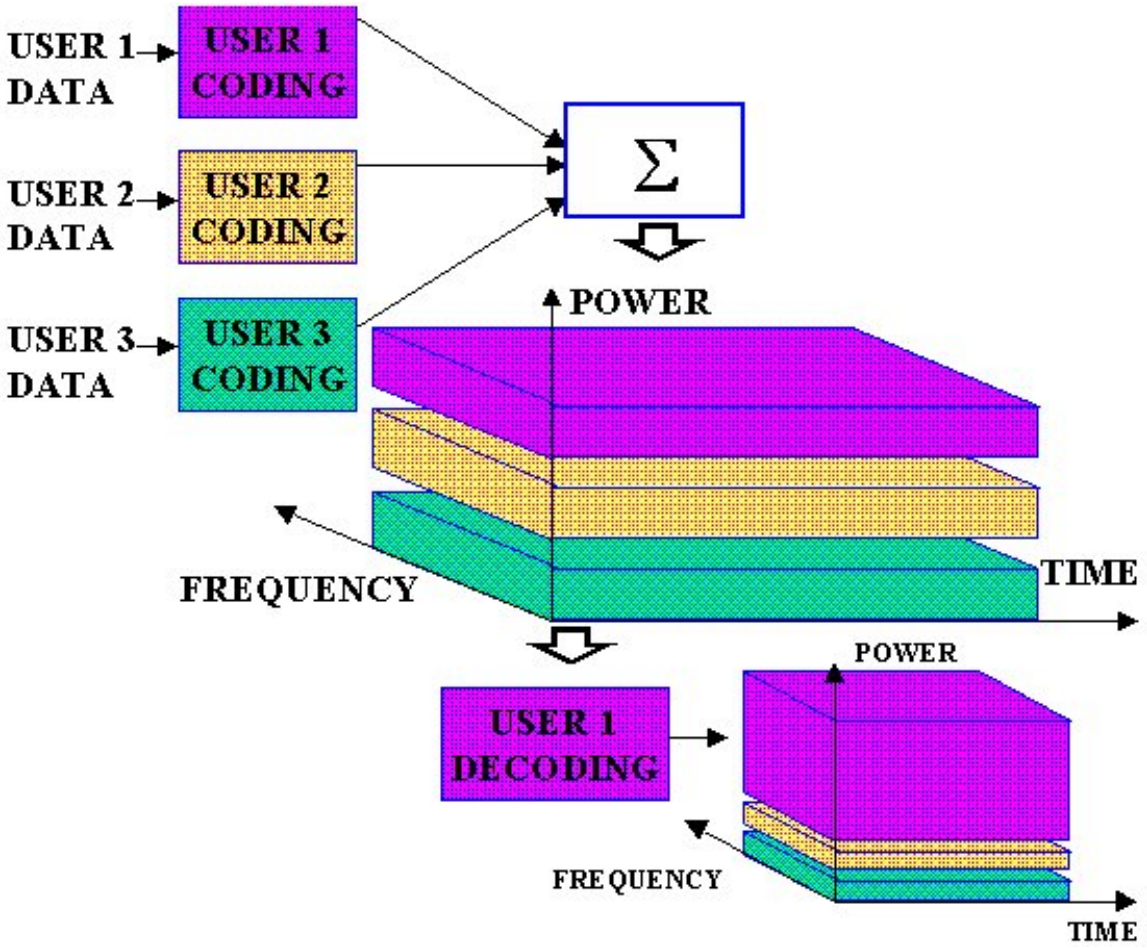
- 頻道存取演算法 ：
  - 相互交叉的叢集中，可藉由使用「分碼多重存取」（CDMA）之方式來增加相互交叉的叢集彼此間平行傳輸（Parallel Transmission）的效能。
  - 在叢集內部各節間點的運作，一般可使用「群首控制符記（Token）」的「輪詢協定」（即Polling）來依序分配頻道給其中某一個節點使用，以避免各節點間因傳輸競爭而產生的碰撞情況。
  - 安排給予群首傳輸優先權，使群首具有更多的傳送機會。
  - 每一個叢集中只有得到允許符記的節點可以使用被分配的傳輸代碼（即CDMA方法中的編碼）來存取頻道。
  - 群首需具有重新發出允許符記之能力，原因：
    - 在一些情況中允許符記可能會遺失。

CDMA

- In Code Division Multiple Access (CDMA), every communicator will be allocated the entire spectrum all of the time. CDMA uses codes to identify connections.

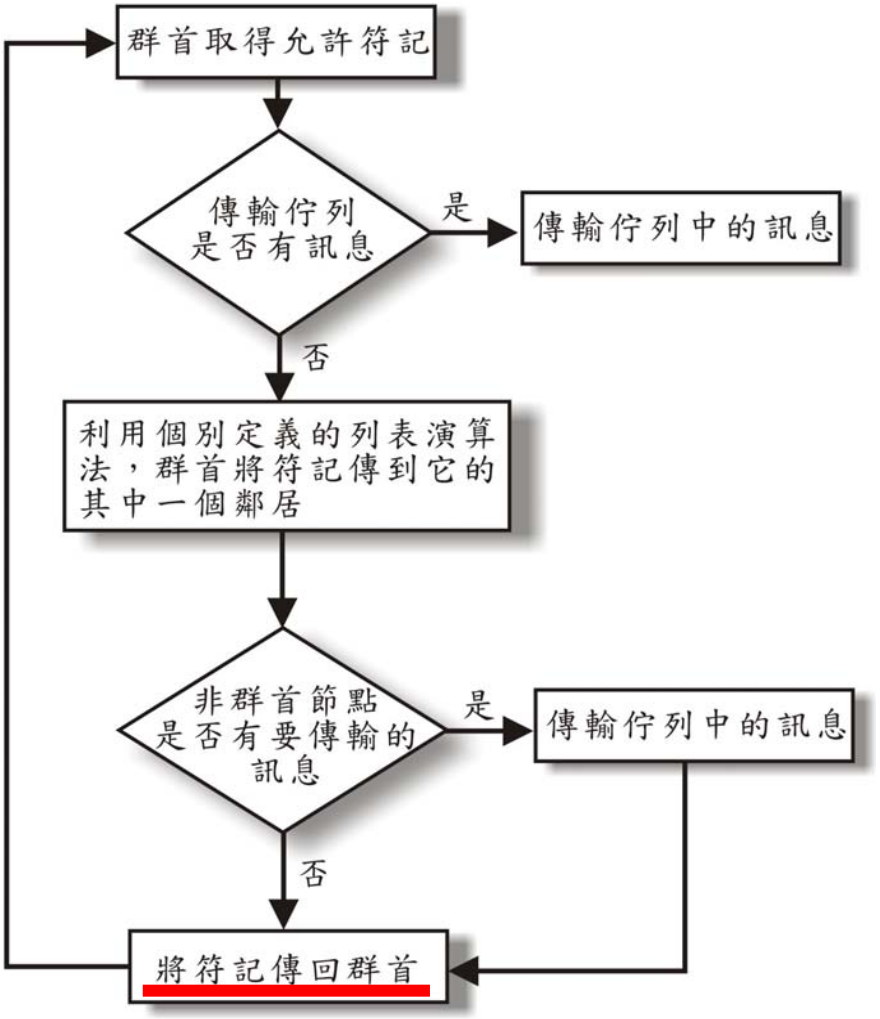


CDMA **(Cont.)**


USER 1 DATA
USER 1 CODING
USER 2 DATA
USER 2 CODING
USER 3 DATA
USER 3 CODING
Σ
POWER
FREQUENCY
TIME
USER 1 DECODING
POWER
FREQUENCY
TIME

表格驅動式路由協定：群首閘道交換路由協定 (CGSR)

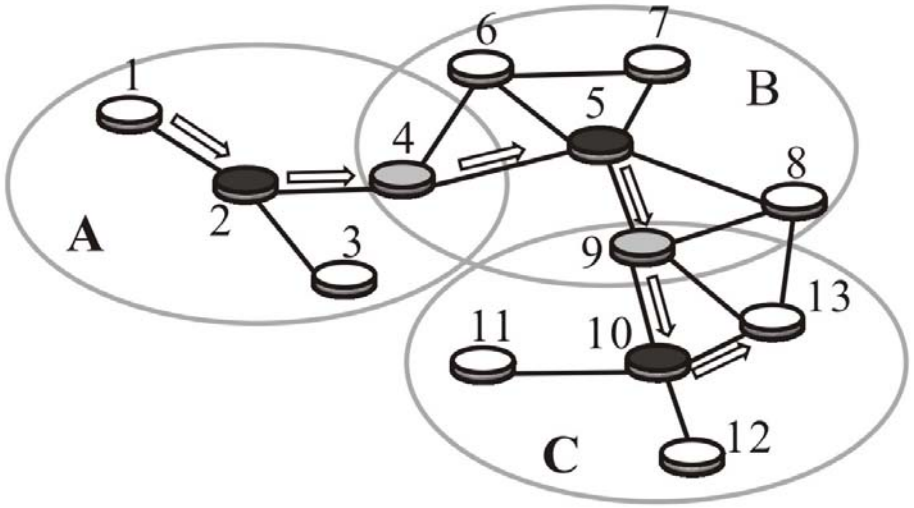
採用輪詢協定的頻道存取演算法



表格驅動式路由協定：群首閘道交換路由協定 (CGSR)

- CGSR路徑選擇：
  - 封包的傳送路徑只由群首節點與閘道節點交替建構而成。
    - C1, G1, C2, G2,… ,Cn
  - 最後到達目的地叢集的群首節點，再由目的地群首節點將封包轉送至目的地節點。
  - 其優點在於群首節點會有較多的傳送機會，而閘道節點則是唯一可在各叢集之間轉送封包的節點。
  - 每個節點有兩種數據結構：
    - 叢集成員表：
      - 描述每個目標節點所在叢集的群首。
      - 每個節點會使用DSDV協定週期性的與鄰近節點交換叢集成員表以更新表項內容，並且使用循序號碼來避免老舊路徑資訊的傳播。
    - 路由表：
      - 記錄了位於通往目的地群首路徑上的下一個節點。(群首或閘道)

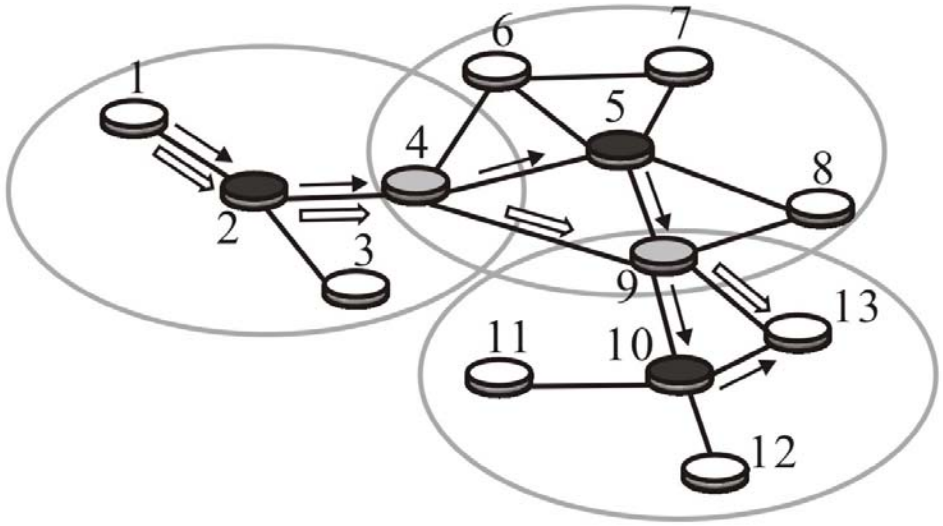
表格驅動式路由協定：
群首閘道交換路由協定 (CGSR)



CGSR的運作示意圖：由節點1傳送封包到節點13之範例

表格驅動式路由協定：
群首閘道交換路由協定 (CGSR)

- DSDV與CGSR的比較：
- 最主要的不同之處在於CGSR可以避免閘道對閘道的傳送。雖然增加路徑長度



- 不同方法會走不同路線，黑箭頭表CGSR方法，白箭頭表DSDV方法
- 模擬結果CGSR比DSDV有較少的平均延遲時間

需求式路由協定：
輕量移動路由協定 (LMR)

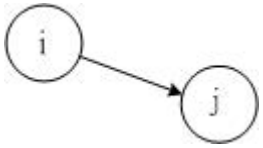
- 在被動式建立路由演算法中，LMR會考慮以下幾點：
  - 有需要時，才建立路徑。
  - 在網路拓撲發生改變前，此演算法仍可以被使用 。
  - 已存在的路徑被毀壞或改變時，則需迅速地反應以重建新的路徑。
- LMR乃是利用一種詢問與回應（Query-reply Process）的處理機制，藉由簡短的控制封包來建立路由之集合，以維護多路徑之結果。
- 當網路拓撲改變或存在之路徑損壞時，詢問與回應的機制會主動地刪除無效的路徑並快速地重建新的路由集合。
- LMR的另一個新的特點就是來源端的初始化（Source-initiated）。
- 來源端的初始化是指給定一個目的地端（DEST），而此演算法只會針對那些要求路由的來源端（Source）進行路徑的維護工作，並不會同時對整個網路路由都進行維護。

需求式路由協定：
輕量移動路由協定 (LMR)

- 路由的建立與維護 ：
- LMR的執行可以被分成三個基本階段：
  - 路由建立階段：
    - 建立一個初始的路由集合 。
  - 路由維護階段：
    - 當路徑有遭受毀壞，且又有資料傳輸需要使用該路徑時，即立即重新建立一條新的路徑以維護整個拓撲。
  - 路由消失階段。

需求式路由協定：
輕量移動路由協定 (LMR)

- LMR定義與格式 ：
  - 網路拓撲：
    - 假設圖形G=（N,L）為包含N個有限節點與L條無方向性鏈結之集合。在此圖形G中，任兩個相鄰之節點可以相互溝通。
    - 節點j與節點i之間存在一個鏈結$l_{i,j}$，而此鏈結可以是有方向性或沒有。
    - 若此鏈結之方向為從節點i到節點j，則節點i為節點j之「上游」（Upstream，UP）且稱節點i為節點j的「上游相鄰節點」。
    - 反之，節點j則為節點i之「下游」（Downstream，DN）且稱節點j為節點i的「下游相鄰節點」。



需求式路由協定：
輕量移動路由協定 (LMR)

- 在鏈結層上的通訊協定有以下幾項假設：
  - 每個節點i在所有時間點上都知道其所有的相鄰節點。
  - 傳輸的封包皆會被正確的接收。
  - 資料封包和控制封包使用不同的通道。
- LMR具有三個不同型態的控制封包：
  - 詢問（Query，QRY）。
  - 回應（Reply，RPY）。
  - 錯誤詢問（Failure-query，FQ）。

詢問（QRY）	SID	DID	SEQ	XID
回應（RPY）	DID		XID	
錯誤詢問（FQ）	DID		XID	

控制封包欄位

需求式路由協定：
輕量移動路由協定 (LMR)

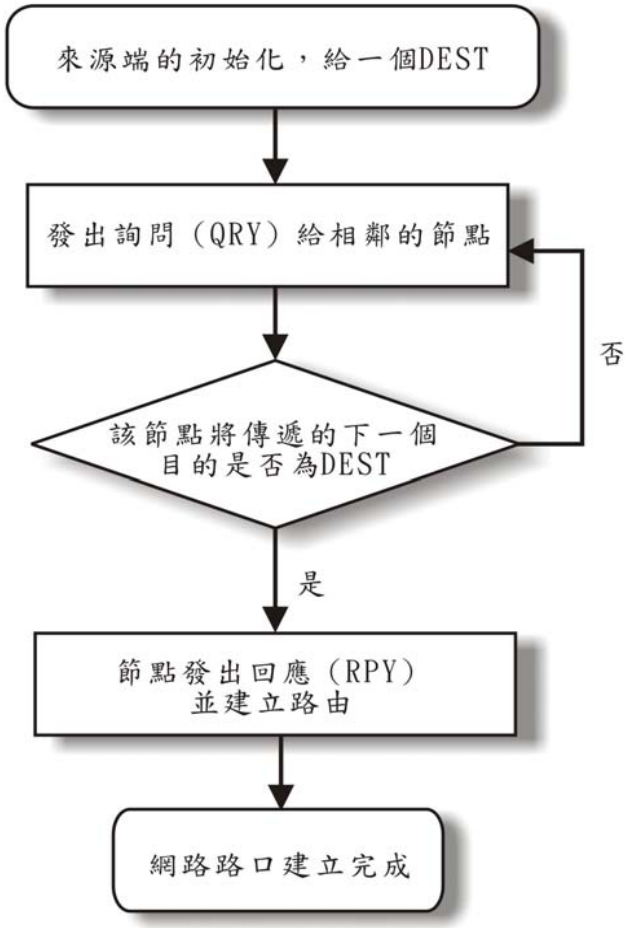
- 欄位說明：
  - 來源節點識別碼（Source Node Identifier，SID）。
  - 目的節點識別碼（Destination Node Identifier，DID）。
  - 循序計數器（Sequence Counter，SEQ）。
  - 交換識別碼（Transmitting Node Identifier，XID）。
  - SID是用來識別發送QRY的來源節點，而DID則是接收此QRY的目的端。
  - 在LMR中，這目的節點即是指DEST。不同來源節點對不同目的節點都會擁有各自的SEQ序列號碼。如此，此三個欄位（SID，DID，SEQ）即形成一個用來區分不同QRY的唯一識別區。
  - XID欄位用來識別最近廣播的封包和得到更新QRY傳播的節點。XID是會改變的，而此改變會隨著QRY所傳播的節點不同而不同。

需求式路由協定：
輕量移動路由協定 (LMR)

- 對於每個鄰居節點j，節點i包含著一個鏈結的狀態表（Link State）LS_j。而對於一個主動鏈結會具有以下六個狀態中之一種：
  - 未指定（Unassigned，UN）：未指定方向之鏈結。
  - 上游（Upstream，UP）：指向上游節點之鏈結。
  - 下游（Downtream，DN）：指向下游節點之鏈結。
  - 阻擋向下游（Downstream-blocked，DN-B）：阻擋指向下游節點之鏈結。
  - 未指派等待（Unassigned-waiting，UN-W）：等待被指定方向之鏈結。
  - 等待廣播（Awaiting-broadcast，A-BR）：等待接受廣播之鏈結
- 若某一節點i並未與其相鄰節點j鏈結，則此相鄰節點j將會標記NULL以表示此鏈結為被動鏈結。
- 在鏈結的標記方面，若此鏈結被標記為UN、UN-W或是A-BR則代表此鏈結為無方向性的。否則，此鏈結則為有方向性。

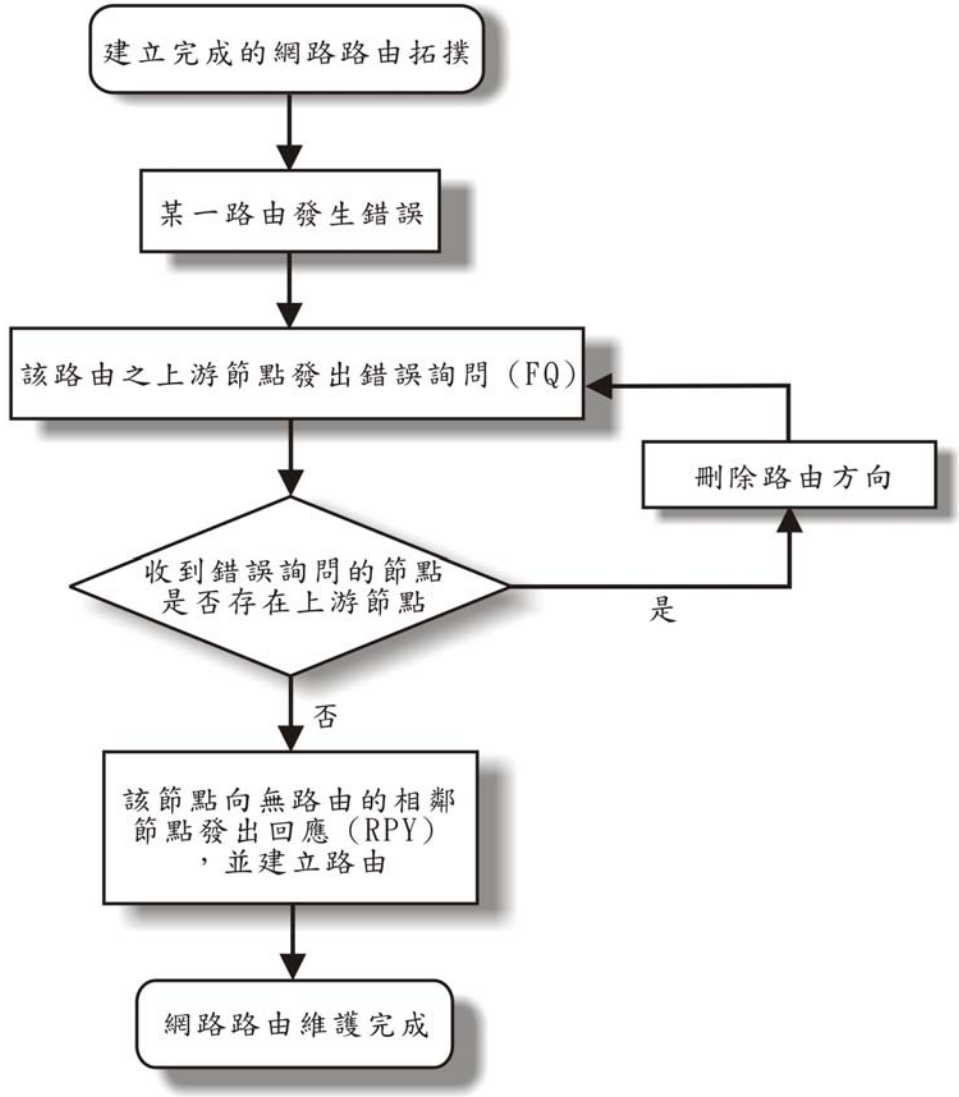
需求式路由協定：
輕量移動路由協定 (LMR)

- LMR演算法流程：
  - 詢問與回應（QRY-RPY）機制：
    - 由SRC送QRY給其鄰點，那些沒有路徑之節點在接收到一個QRY，將會重新廣播或轉遞QRY。當某一節點有路徑則會廣播一個RPY，此RPY將朝著發QRY的來源節點方向反向泛濫，路由因此而被建立。



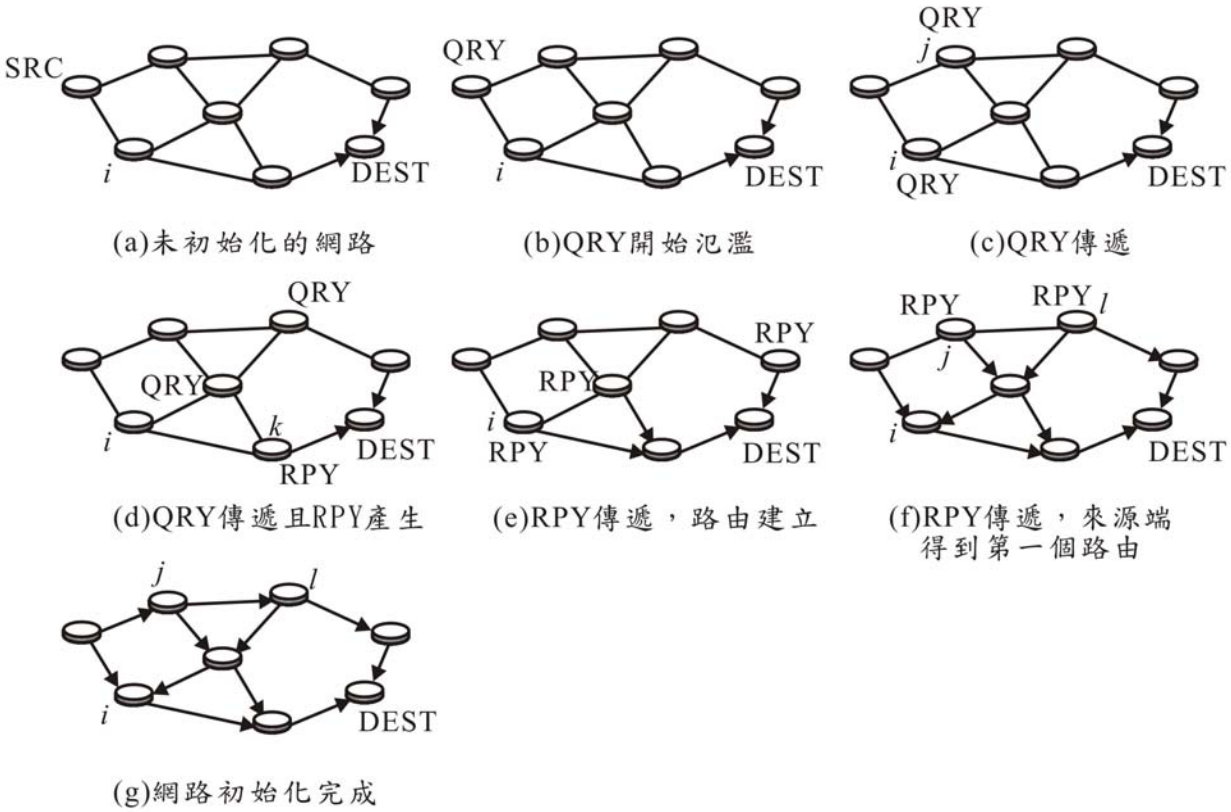
需求式路由協定：
輕量移動路由協定 (LMR)

- 錯誤詢問與回應（FQ-RPY）機制：
  - 假如發生鏈結錯誤的節點沒有其他節點之路徑通過它到DEST，且其亦有上游節點，則朝著它的上游節點發一個FQ。



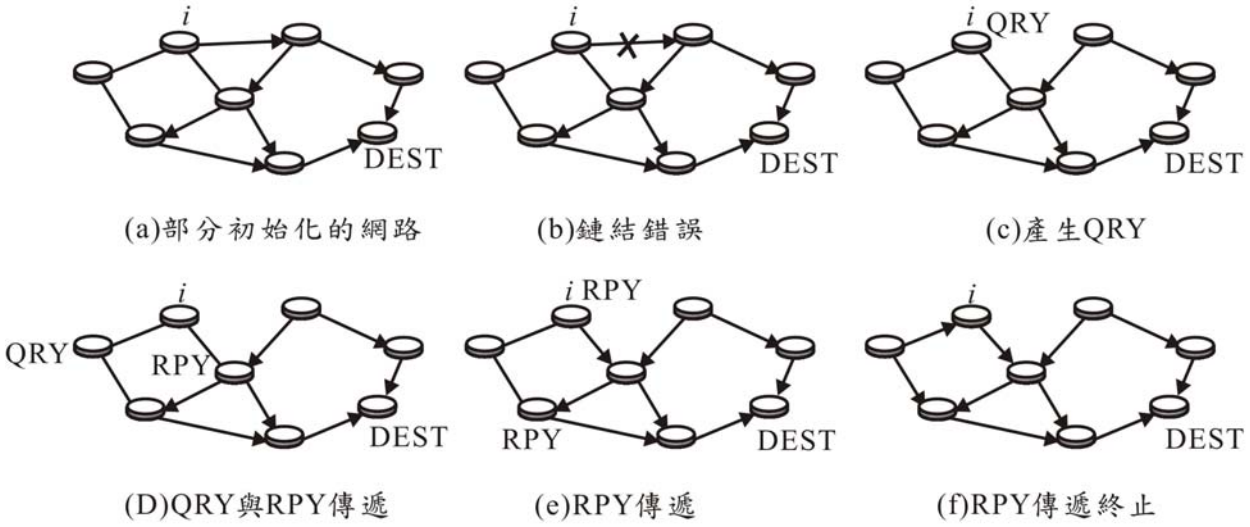
需求式路由協定：
輕量移動路由協定 (LMR)

- 建立、維護與消失路由的方式：
- 在網路開始的時候，除了鄰近DEST的節點外，其餘節點的鏈結都是無方向性。
  - 建立路由階段：（QRY-RPY機制）



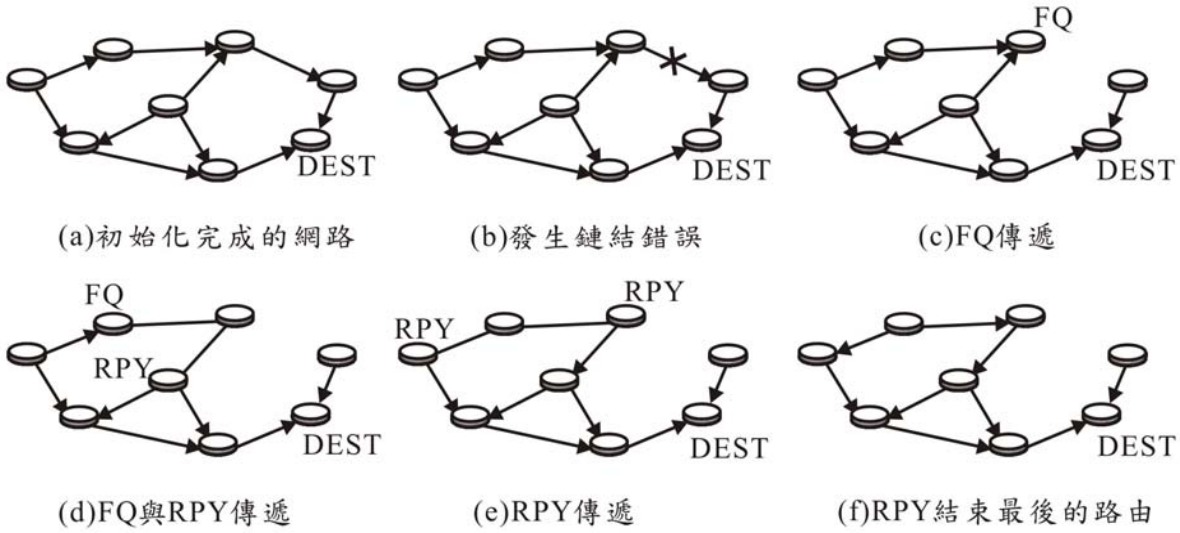
需求式路由協定：
輕量移動路由協定 (LMR)

- 維護與消失路由階段(1)：在下游鏈結錯誤後路由重建階段（QRY-RPY機制）。



需求式路由協定：
輕量移動路由協定 (LMR)

- 維護與消失路由階段(2)：無效路由刪除，路由重建階段（FQ-RPY機制）。

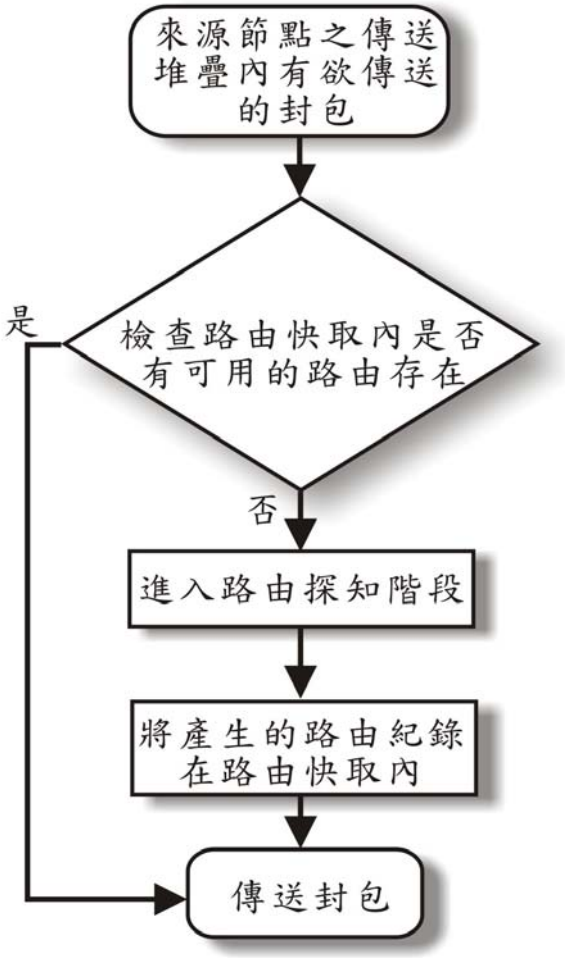


需求式路由協定：
動態來源路由協定 (DSR)

- DSR是由來源節點進行主動尋路的一種無線網路路由技巧。
- 任一來源節點皆可透過對自身路由快取的維護，或是藉由路徑探知而動態產生路徑，並以此決定使用較適合之路徑。
- DSR相較於其他主動式路由協定：
  - DSR不需額外產生一些與路由相關的週期性資訊，也就是沒有任何的週期性路由資訊在節點之間氾濫，這將會節省許多頻寬上的浪費。
  - 來源節點決定了所有封包在傳遞時所需經過的中間節點順序，此路由的資訊將會被放在封包的頭端（Header）部份，而此封包將會依此順序來傳遞至目的節點。
- DSR包含了兩個主要的階段：
  - 路徑探知階段。
  - 路徑維護階段。
- DSR允許網路內的任一個節點能同時擁有多條路徑。

需求式路由協定：
動態來源路由協定 (DSR)

- 在無基礎式網路內DSR路由協定的運作流程圖 。
- 假設這些節點皆能夠轉送來自網路中任一節點所產生之封包，且在網路剛形成之初，所有節點內之路由快取皆是空的。

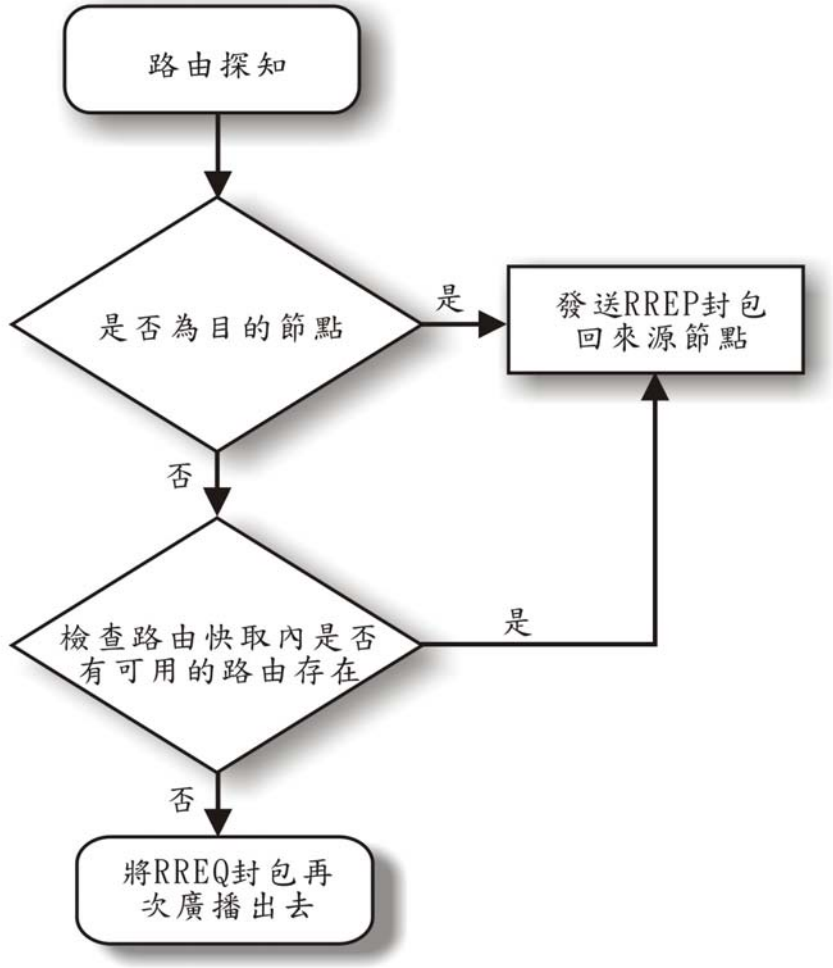


需求式路由協定：
動態來源路由協定 (DSR)

- DSR路由協定 ：
  - 當網路處於初始狀態，或某一來源節點在其路由快取中搜尋不到任何與目的節點有關之路徑時，則此來源節點將進入路由探知階段。
  - 在路由探知階段，來源節點會產生一個「路由請求」（Route Request，RREQ）之封包，再透過廣播方式將此請求封包送至其鄰近之節點。
  - 在RREQ封包的表頭檔裡，存在有一個名為「路由記錄」之欄位。當接收到此RREQ封包之節點不是目的節點，且在此欄位中亦無自己位址之記錄時，此節點將會把自身的位址依序填入此記錄中，再將此RREQ封包繼續廣播出去，直到此封包達到目的節點為止。
  - 當目的節點接收到一個RREQ封包時，此目的節點將會產生一個名為「路由回應」（Route Reply，RREP）之封包。其中包含了此RREQ封包中所收集的路由記錄。
  - 在廣播的過程中，當任何中間節點在接收到RREQ封包時，若當時該節點之路由快取內已擁有有關到目的節點的路徑資訊，則此中間節點將會把此路徑資訊直接附在RREP封包內，再由此節點直接送回給來源節點，而不再繼續廣播RREQ封包。

需求式路由協定：
動態來源路由協定 (DSR)

路由探知階段之流程圖



需求式路由協定：
動態來源路由協定 (DSR)

Route Discovery

Route Reply

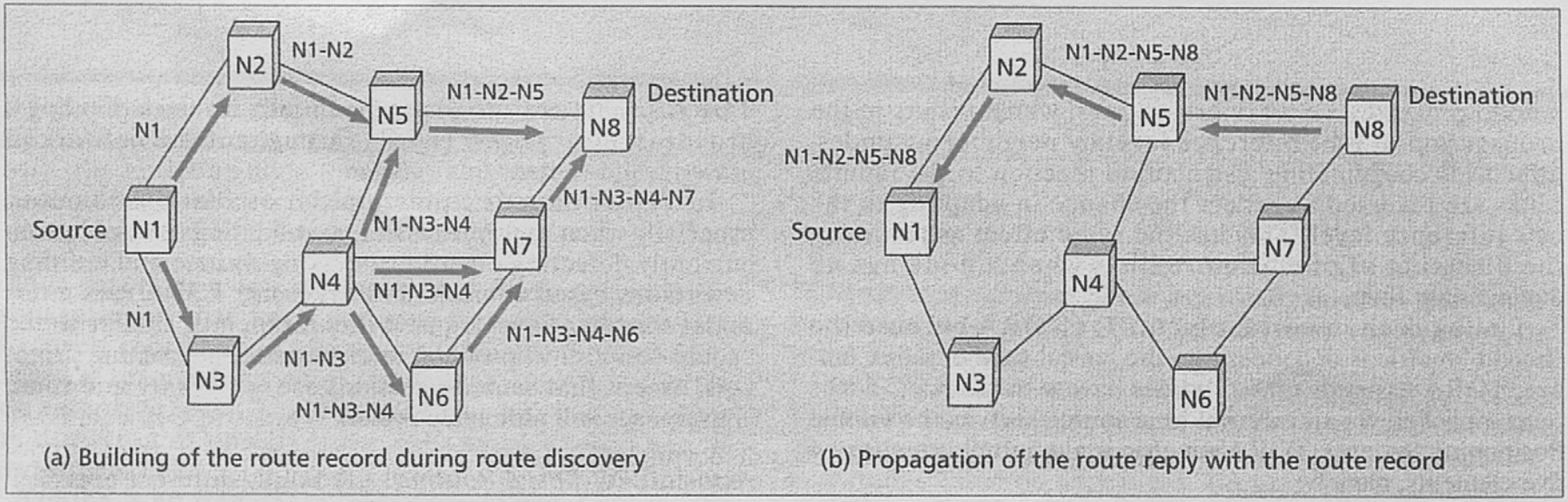
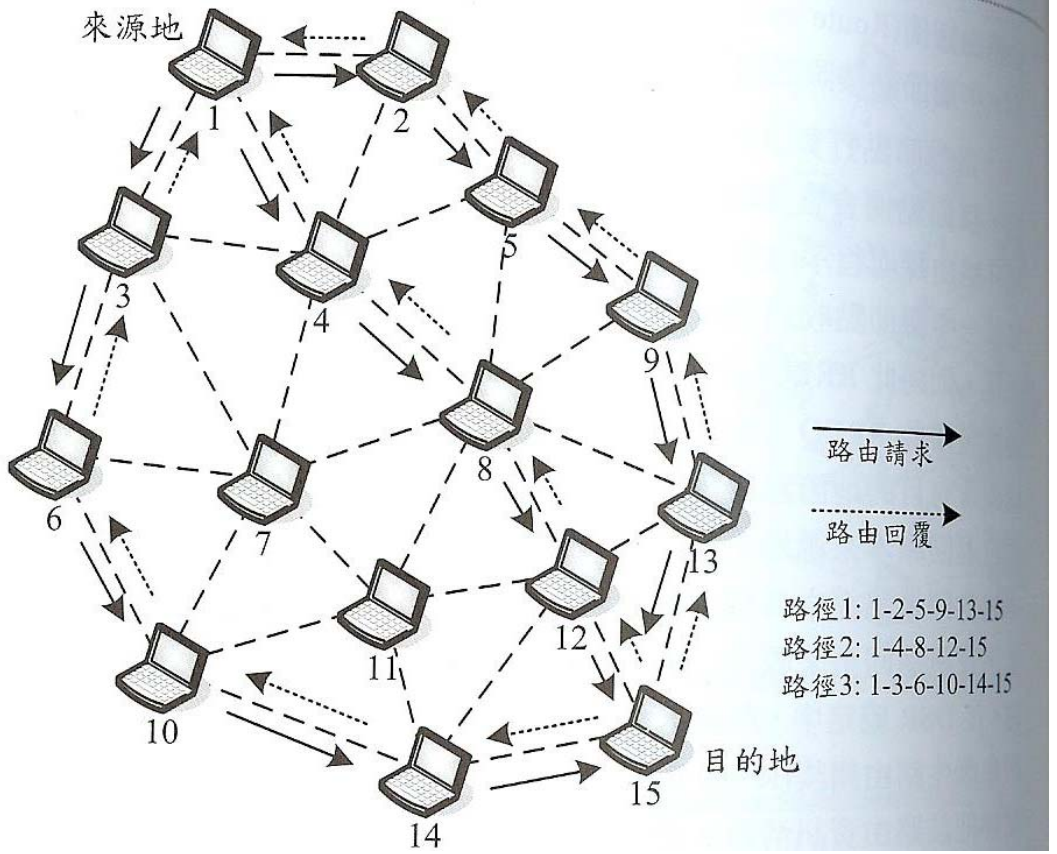


Figure 4. *Creation of the route record in DSR.*

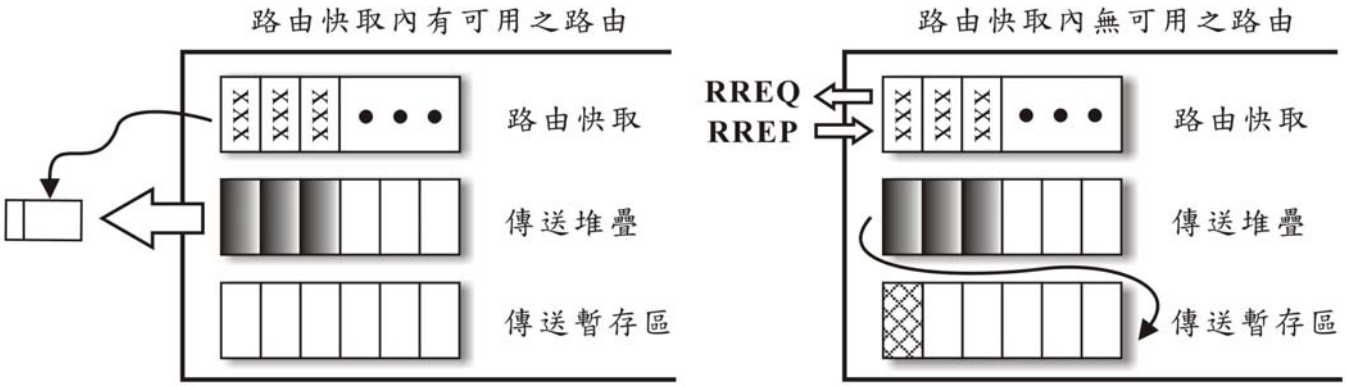
需求式路由協定：
動態來源路由協定 (DSR)

▶ 圖 8.4: DSR 的路徑探索例子



需求式路由協定：
動態來源路由協定 (DSR)

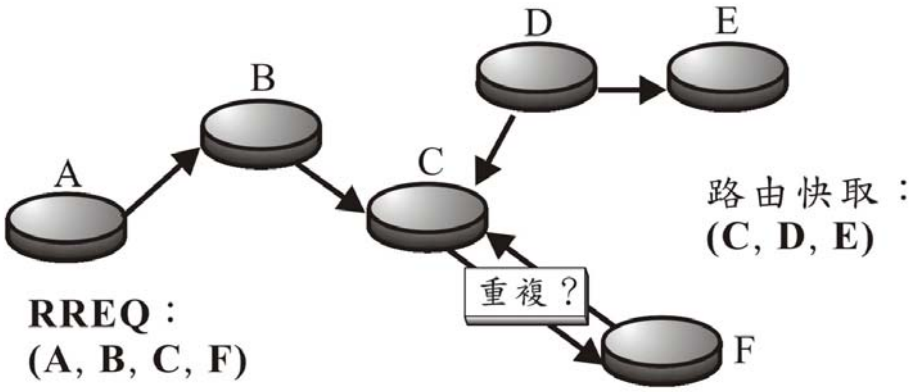
- 節點都會維護一個所謂的「傳送暫存區」（Send Buffer）。
- 當節點想要傳送封包時，若其無法在路由快取內馬上找到一條適合的路徑時，此封包將會先被存於傳送暫存區中以等待新的路由探知結果。
- 為避免此暫存區因儲存過多暫存性封包而發生溢位之情況，節點本身必須採用先進先出〈First In First Out，簡稱FIFO〉或其他之演算法來對存於此暫存區之封包進行篩選。



節點內所維護的傳送暫存區

需求式路由協定：
動態來源路由協定 (DSR)

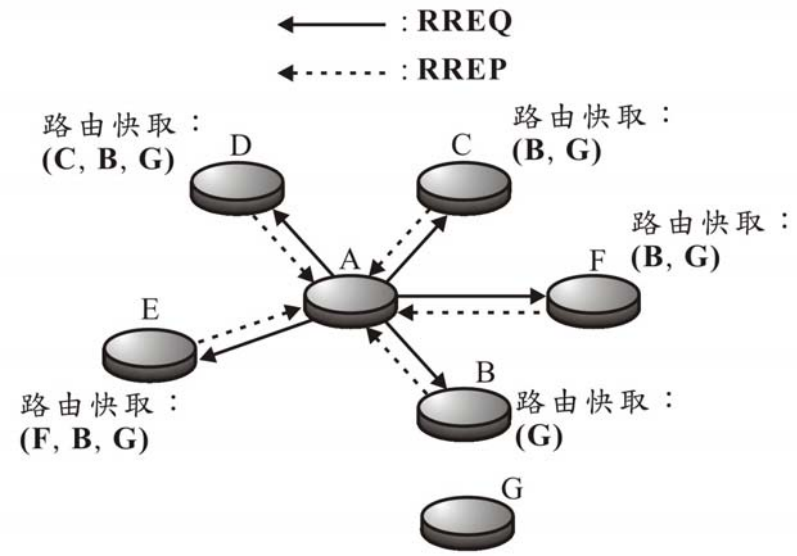
- Example 1：在路由探知階段，來源節點A欲尋找至目的節點E之路徑而廣播其RREQ封包卻發生重複路徑之情況 。



- F發現重複路徑之發生 故其將不會發出RREP

需求式路由協定：
動態來源路由協定 (DSR)

- Example 2：在某些網路環境下，DSR路由協定會產生一種名為「路由回應風暴」（RREP Storms）之問題。
  - 當某一個節點A把對某一目的節點G之RREQ封包廣播出去時，若是其數個鄰近節點（如：節點B, C, D, E, F）之路由快取中皆存有對此目的節點G之路徑，而不約而同地發出其各自的RREP封包。如此一來，除了會大量浪費網路頻寬之外，更可能產生網路內RREP封包間的碰撞而使節點A無法正確收到任何回應之封包，如下圖所示。 Random Delay to wait the data sent from A



需求式路由協定：
無基礎式需求距離向量路由協定 (AODV)

- AODV是以循序的目的地距離向量路由協定（DSDV）與動態來源端路由協定（DSR）為基礎發展而得。
- AODV同時採用了
  - DSDV中的逐跳路由（Hop by Hop）、序列號碼（Sequence Number）
  - DSR中的需求式路由機制（即指路由的探知與維護方式）
- AODV與DSDV不同之處在於，AODV會因應當時所需之路徑來進行操作而減少了廣播路由尋找封包的次數。而與DSR相比，AODV的封包並不需要像DSR包含該條路由所有節點的資訊，以減低網路頻寬的浪費。

需求式路由協定：
無基礎式需求距離向量路由協定 (AODV)

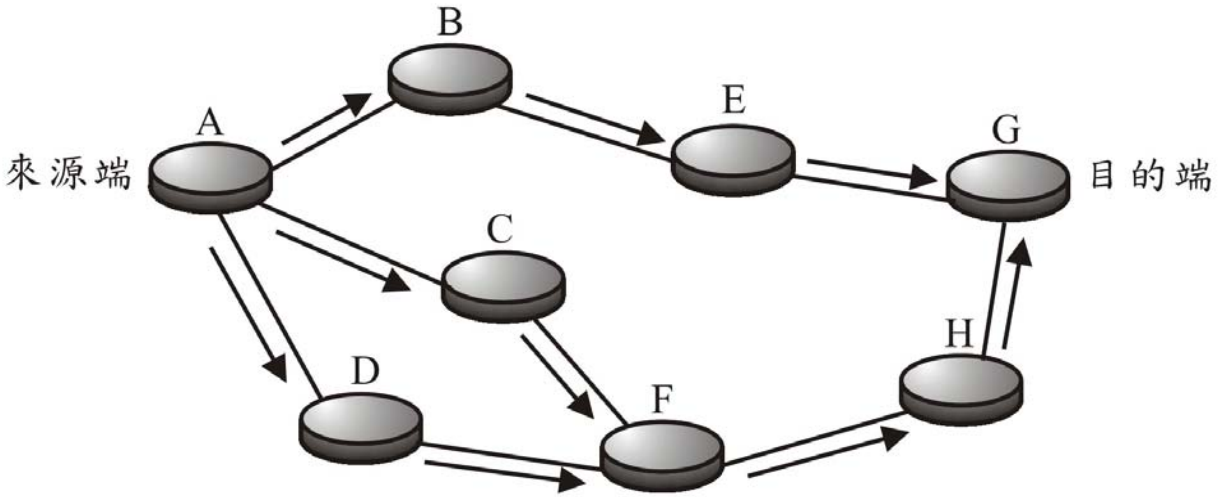
- AODV的運作 ：
  - AODV是利用封包中網際網路表頭（IP Header）部份的資訊來處理整個網路的路由運作。
  - 在AODV架構下所有封包皆是利用網際網路中UDP/IP協定，並使用埠號（Port Number）：654與255.255.255.255的網際網路位址（IP Address）對整個網路進行封包的廣播。
- AODV中的序列號碼 ：
  - 在每個節點所記錄的路由表中，除了有目的節點的IP位址之外，還多記錄了目的節點的路由序列號碼，而這組序列號碼即稱之為「目的地序列號碼」（Destination Sequence Number），用來維持AODV路由資訊的更新。
  - AODV定義之序列號碼範圍為32位元的正整數，即0至223－1。因此，當序列號碼到達其上限值時，如果再增加則數值會重回為0。

需求式路由協定：
無基礎式需求距離向量路由協定 (AODV)

- AODV節點中之路由表：
  - 在AODV中，每個節點內部都會擁有一個路由表來維持其路徑資訊。而此路由表中所記錄的除了上述序列號碼之外，也記錄了一些有用的資訊，如下所述：
    - 目的節點（Destination）位址：記錄著目的節點的IP位址。
    - 下一個中繼節點（Next Hop）：記錄位於通往目的節點之路徑上的下一個中繼節點。
    - 中繼節點個數（Number of Hops）：記錄本節點到目的節點間所需經過的中繼節點個數。
    - 目的節點序列號碼（Sequence Number for the Destination）：記錄該目的節點之序列號碼。
    - 路徑有效時間（Expiration Time for the Route Table Entry）：記錄該路徑資訊存在路由表中的有效時間。
    - 活躍的鄰居節點（Active Neighbor）：在一節點的路由表中，每條路徑的上游節點。

需求式路由協定：
無基礎式需求距離向量路由協定 (AODV)

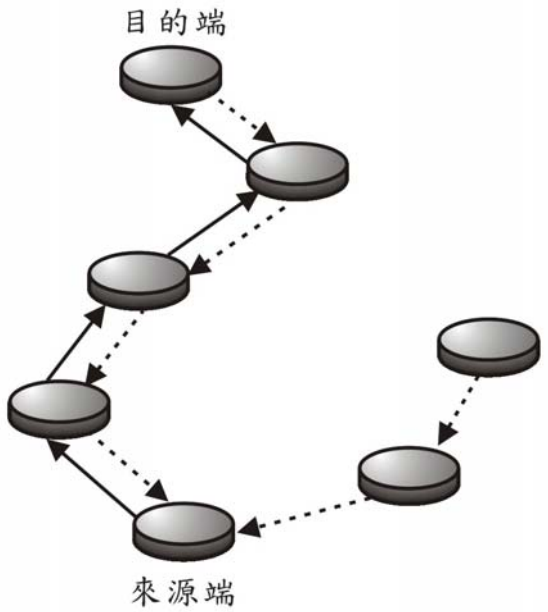
- AODV協定中定義了四種控制封包格式：
- 路徑請求（Route Request，簡稱RREQ）封包 ：
  - 當某個來源節點欲傳送資料至另一個目的節點，如果此來源節點在本身之路由表中找不到可以到達該目的節點之路徑，或是此路徑資訊已經過期而被註記為無效時。這時來源節點將會廣播「路徑請求」（RREQ）封包以搜尋能夠到達該目的節點之新路徑，如下圖所示。



需求式路由協定：
無基礎式需求距離向量路由協定 (AODV)

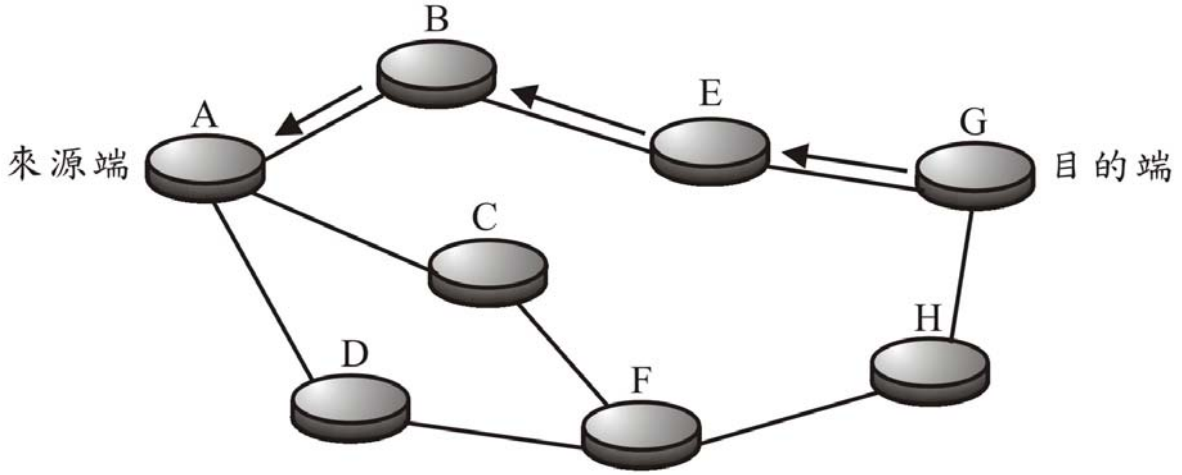
- 路徑回覆Route Reply（RREP）封包 ：
  - 在整個網路中，收到RREQ封包之節點會根據RREQ封包上之發起者IP位址與UDP/IP表頭內容資訊搜尋本身之路由表，然後對發起此RREQ封包之發起者進行逆向路徑之更新或建立以利將來RREP封包的回送。
  - 同樣地，收到RREP封包之節點，也會建立或更新前往目的節點之路徑，以確保路徑資訊的最新狀態。等回送的RREP封包被發起者收到後，一條具有雙向路徑資訊的通道就被建立完成如下圖所示。

建立從來源節點至目的節點之雙向路徑，而右邊兩個節點則因無法找到目的節點，因而丟棄RREQ封包，只保留對來源節點的反向路徑。



需求式路由協定：
無基礎式需求距離向量路由協定 (AODV)

- 在AODV協定中，會發出RREP封包之節點有兩種：
  - 一種是RREQ封包中所記錄之目的節點。
  - 另一種則是介於來源節點與目的節點之間的中繼節點。



RREP封包回送至發起者之範例

需求式路由協定：
無基礎式需求距離向量路由協定 (AODV)

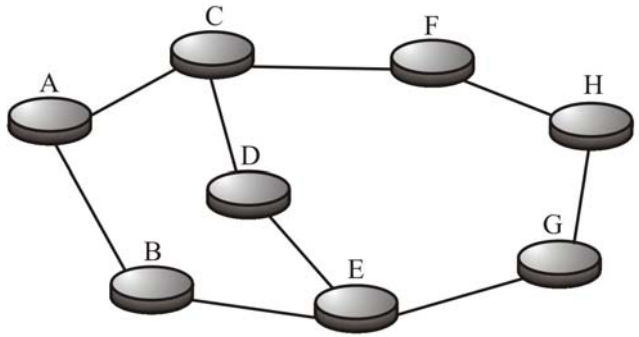
- 路徑錯誤Route Errors（RERR）封包 ：
  - 一般在AODV協定下，某一節點要處理路徑錯誤（Route Error）、鏈結毀損（Link Breakage）或是路徑刪除（Route Deletion）時將會依照下列幾個步驟：
    - 首先，先把該路徑設為無效。
    - 列出因此受到影響之所有目的節點。
    - 檢查是否有鄰近之節點亦會受到影響。
    - 送出Route Errors（RERR）封包給這些受影響之鄰近節點。

需求式路由協定：
無基礎式需求距離向量路由協定 (AODV)

- 一個節點會發出RERR封包之情況，可分為以下三種：
  - 當某一節點在傳輸資料時，發現原本可用之路徑（Active Route）發生鏈結毀損（Link Breakage），並嘗試自行修復路徑（Route Repair）無效時。
  - 當某一節點收到一個已經具有預設目的節點之資料封包時，卻發現本身並無此目的節點之有效的路徑。
  - 當某一節點收到一個RERR封包，且其上所通知之路徑與本節點記錄中之一條或多條有效路徑有關時。

需求式路由協定：
無基礎式需求距離向量路由協定 (AODV)

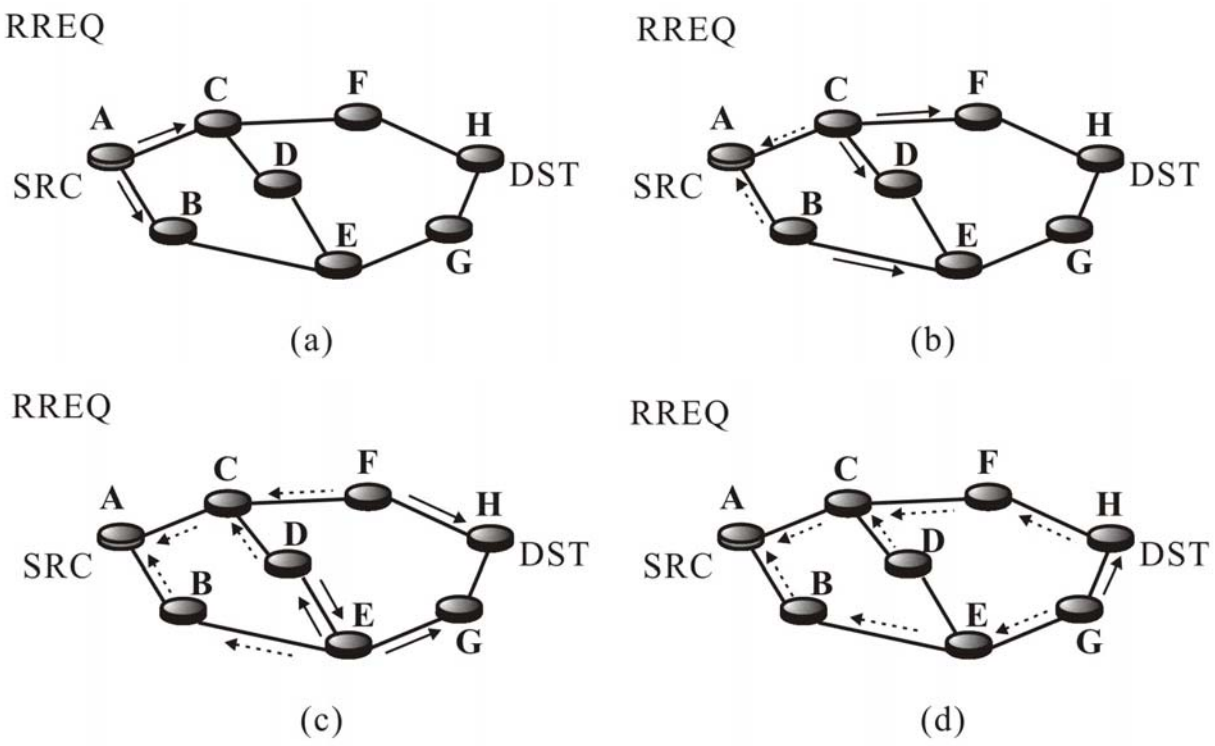
- Example 1：根據AODV協定與時間點描述之方式來示範節點A如何傳送資料封包到節點H。
  - 此網路中具有八個節點，而各節點間之連線則代表各節點存在路由表中已知之路徑。
  - 假設節點A為來源節點，節點H為目的節點。



網路拓撲示意圖

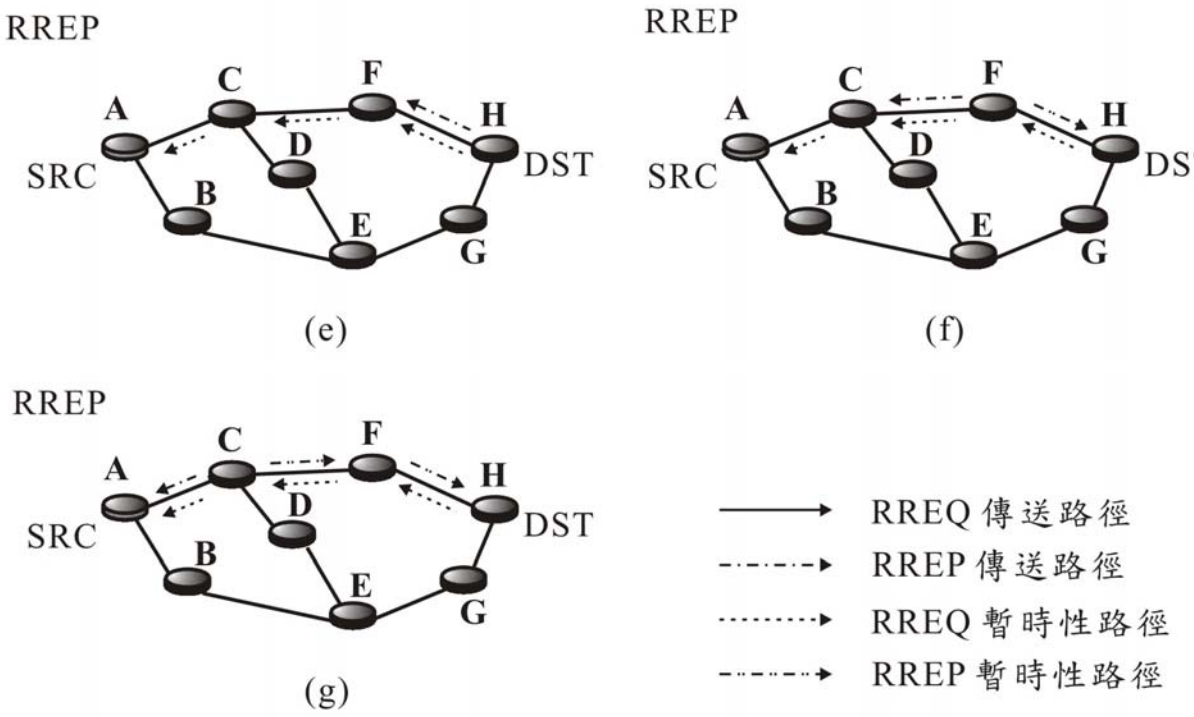
需求式路由協定：
無基礎式需求距離向量路由協定 (AODV)

- AODV運作流程（1）：



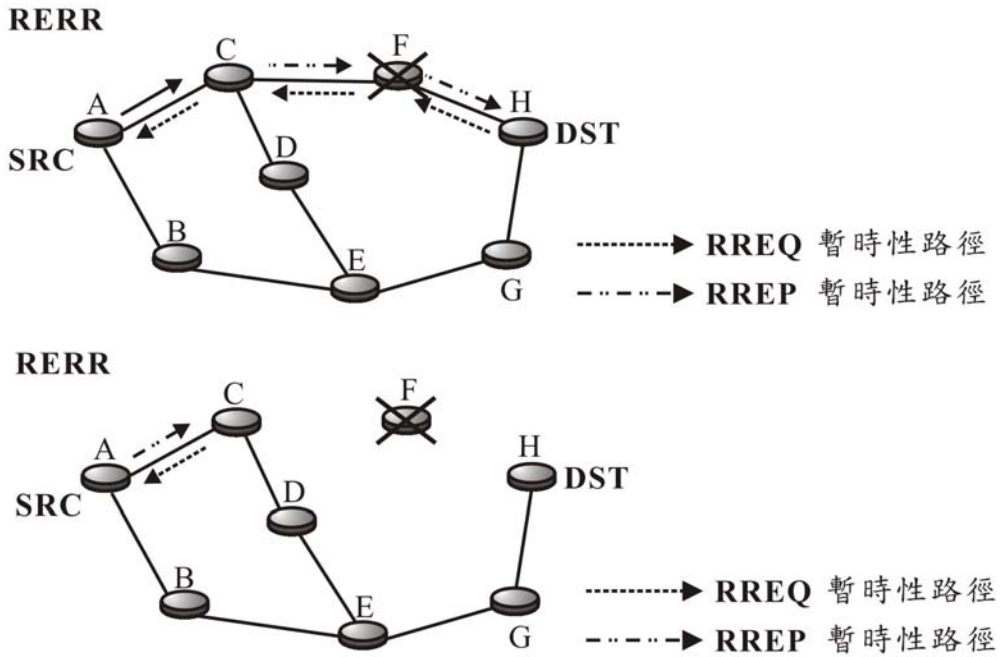
需求式路由協定：
無基礎式需求距離向量路由協定 (AODV)

- AODV運作流程（2）：



需求式路由協定：
無基礎式需求距離向量路由協定 (AODV)

- Example 2：節點F突然毀損或離開而發生了所謂的路徑錯誤（Route Error）之情況 。



節點C開始進行區域修復（Local Repair）

需求式路由協定：
關聯性基礎路由協定 (ABR)

- ABR是一個介於廣播（Broadcast）與點對點（Point-to-point）的折衷路由方法。
- 在ABR協定下，每個節點不需擁有路由資訊，而路徑之選擇乃建立於所謂的關聯性（Associativity）狀態。
- 此關聯性狀態常常會暗示節點是否正處於一個穩定之狀態，因此路徑不需要時常被要求重新更新。
- 在一般傳統的路由協定中，此路由準則之考量如下：
  - 對於連結（Link）變化的快速適應性。
  - 考量到達目的節點所需之中繼節點（Hop）。
  - 傳送延遲（Propagation Delay）。
  - 迴路的避免。
  - 連結的能力。

需求式路由協定：關聯性基礎路由協定 (ABR)

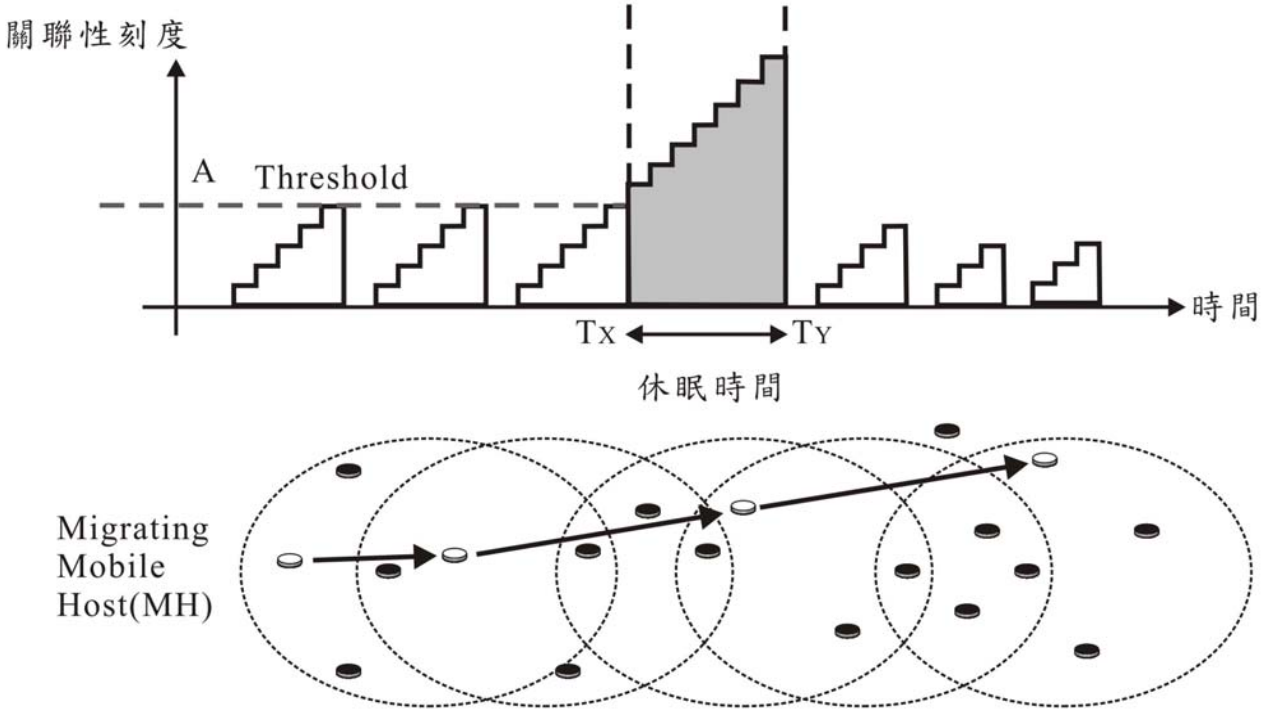
- 在ABR協定中，另一套新的路由準則被引用，而這些考量與節點之關聯穩定度（Association Stability）皆有關系，其描述如下：
  - 路徑壽命（Route Longevity）。
  - 中間節點（Intermediate Node，IN）的轉送負載（Relaying Load）。
  - 被選擇的路徑中，其具有連結能力的資訊。
  - 關聯性刻度（Associaticity Ticks）。

需求式路由協定：
關聯性基礎路由協定 (ABR)

- 關聯性刻度表示一個移動式主機（Mobile Host，簡稱MH）在時間、空間上其連線的穩定度。
- 當一個MH移動時，其與鄰近節點之間的關聯就會發生變化。而在此變化發生之期間，即可用關聯性刻度來表示。
- 在無線網路環境中，每一個移動式節點會周期性的產生一個訊標（Beacon）。藉由此訊標節點可以更新自己與其鄰近節點間的關聯性刻度。網路中當某一個鄰近節點具有較高的關聯性刻度時，則我們會認為此節點正處於穩定之狀態。

需求式路由協定：
關聯性基礎路由協定 (ABR)

- 下圖即表示一個MH在Ad Hoc無線網路環境中移動時，與其鄰居間在時間與空間上之關聯性刻度。



需求式路由協定：
關聯性基礎路由協定 (ABR)

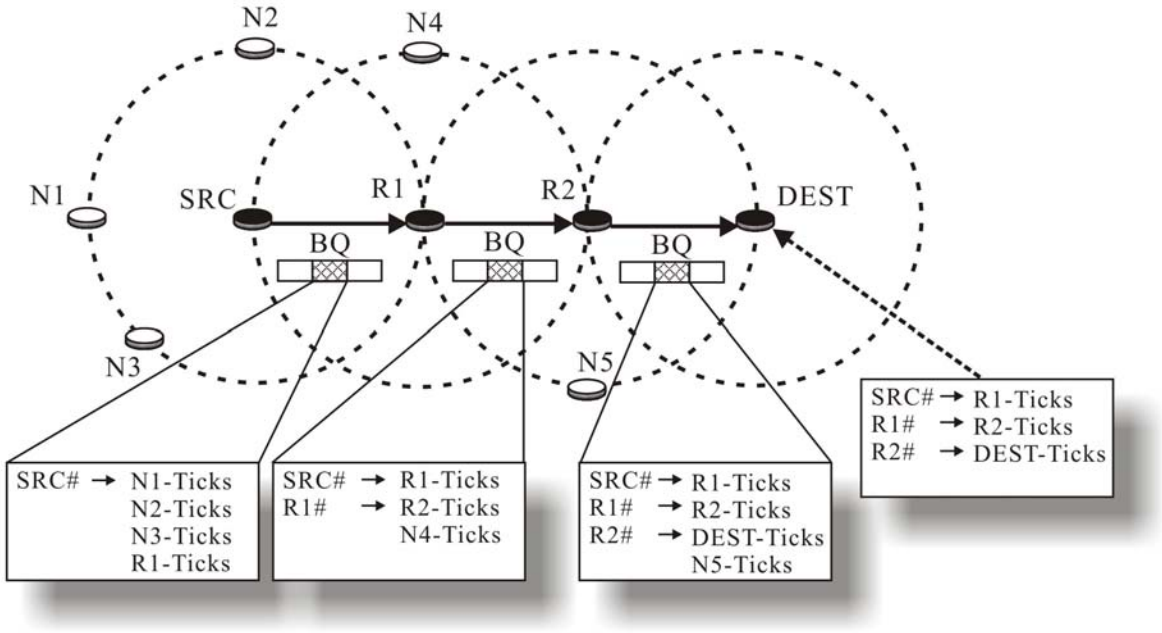
- ABR協定概述 ：
  - ABR協定包含了三個重要的階段：
    - 路徑探知階段（Route Discovery）。
    - 路徑重建階段（Route Reconstruction，RRC）。
    - 路徑刪除階段（Route Delete，RD）。
  - 當來源節點（Source，SRC）需要一條路徑以傳送資料封包時，此節點即會進入路徑探知階段。
  - 若是因SRC、目的節點（Destination，DEST）或者是路徑中某中間節點之移動而造成此路徑變更時，則此時SRC將會進入路徑重建階段。
  - 若是此路徑已不再被需要時，則SRC將會進入路徑刪除階段，以刪除此一路徑。

需求式路由協定：關聯性基礎路由協定 (ABR)

- ABR路徑探知階段 ：
  - ABR路徑探知階段是由廣播要求（Broadcast Query，BQ）與等待回應（Wait Reply，REPLY）兩個所形成的循環階段。
  - 當SRC需要路徑來傳資料封包到某個DEST時，此時SRC會在網路中廣播一個BQ封包，以搜尋並建立到DEST之路徑。
  - 右圖為BQ控制封包格式，其中包括：
    - TYPE：封包種類。
    - SRC ID：來源節點ID。
    - DEST ID：目的節點ID。
    - LIVE：此封包所經過的中繼節點數。
    - IN IDs：中繼節點ID。
    - METRICS：關聯性刻度參數。
    - SEQ NO.：封包序列號碼。
    - CRC：封包錯誤檢查碼。

需求式路由協定：
關聯性基礎路由協定 (ABR)

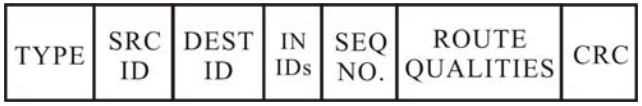
- 所有接收到此BQ封包之INs將先確認，是否在此之前已經處理過相同之封包，若已經處理過便丟棄此封包。
- INs本身亦會檢查自己是否為DEST。若不是，則將自己的MH位址填入此封包內，再將此封包繼續以廣播之方式傳送給其鄰近節點。
- 節點本身之關聯性刻度與其他路由準則參數也會被包在此封包之內。



BQ封包上關聯性刻度資訊的變化之過程

需求式路由協定：
關聯性基礎路由協定 (ABR)

- 當DEST接收到一個BQ封包，從此封包中便可得到一條從SRC至DEST之路徑。
- 在經歷一段收集時期後DEST將會收到許多BQ封包，亦因此而獲得許多可用之路徑。
- 透過最佳路徑演算法，DEST可以在眾多的路徑中選出一條最佳之路徑（即最穩定之路徑），再透過REPLY封包將此路徑之資訊回傳給SRC。
- 在ABR協中，最佳路徑的意思並非為最短路徑。此演算法利用由BQ封包所蒐集到INs間之關聯性刻度而得知其路徑之穩定性，故可能會選擇**較穩定但較長之路徑**。



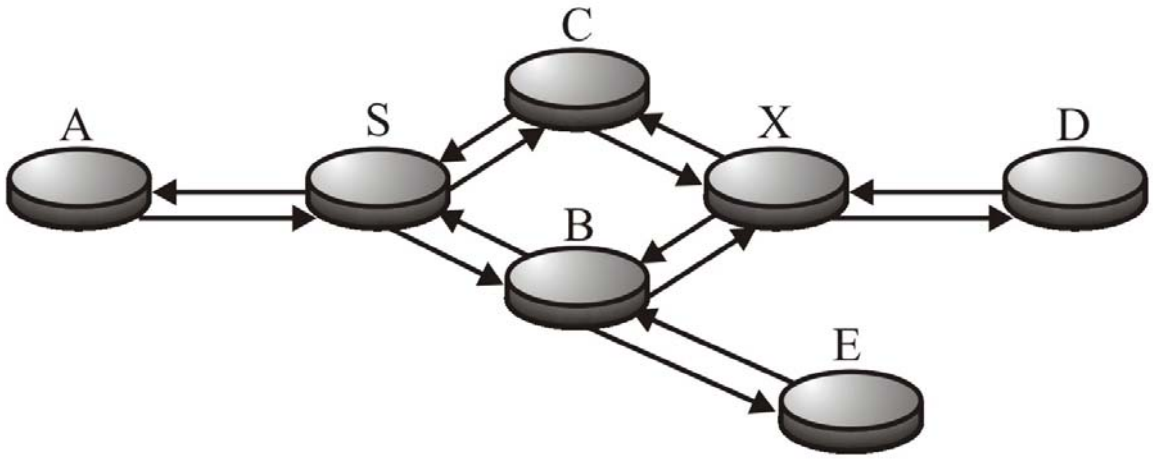
REPLY控制封包格式

需求式路由協定：
輔助定位路由通訊協定 (LAR)

- LAR協定之運作方式類似動態來源端路由（DSR）協定，其乃是利用位置資訊來改善通訊路由協定之效能。
- 在移動的無基礎式無線網路中（Ad-hoc Network），LAR協定會利用全球定位系統（Global Positioning System, GPS）提供網路節點在二維平面上之位置資訊進而得知一個可移動主機之位置，然後使用位置資訊去減少節點在二維平面上路由的額外花費。
- 利用訊息泛濫來尋找路由 ：
  - 類似於DSR，LAR使用泛濫之方式來找尋路徑。
  - 當節點S欲找尋一路徑到達節點D，則節點S將會廣播一個路徑要求訊息封包給它所有之鄰近節點。
  - 網路中任何一個節點在收到這個路徑要求訊息封包時，會先比較此封包中所載之目的節點是否為自己。
  - 若為自己，表示已找到目的節點。反之，此節點將會繼續轉送此訊息封包。

需求式路由協定：
輔助定位路由通訊協定 (LAR)

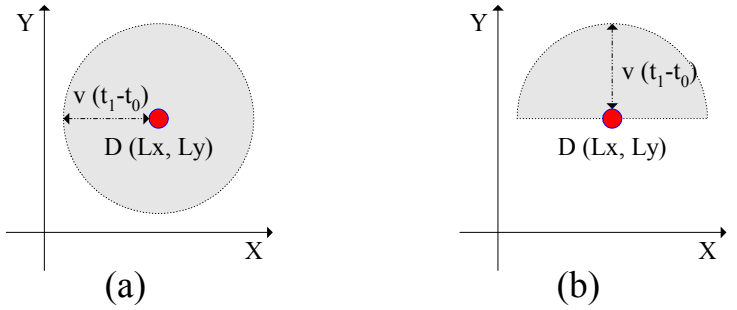
- 在訊息封包氾濫的過程中，為了避免在網路中傳送多餘的要求訊息，節點在轉送訊息封包給其鄰近節點時，只會廣播一次（節點會使用自身的序列計數器來檢測重複接收之訊息封包）。
- 若經歷一段時間後目的節點沒有收到任何路徑要求訊息或來源節點沒有收到任何路徑回應（可能是封包遺失或損壞等），都會造成此次的廣播封包逾時（Timeout）而無效。



LAR使用訊息泛濫之方式來找尋路徑

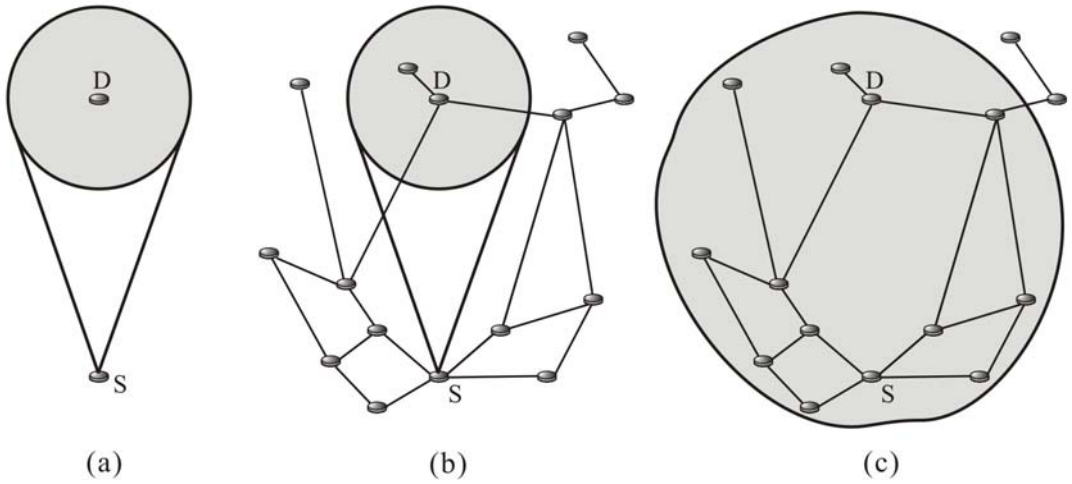
需求式路由協定：
輔助定位路由通訊協定 (LAR)

- 預期區域與要求區域（Expected Zone and Request Zone）：
  - 所謂的「**預期區域**」是指：某一節點S欲找尋一條路徑至某一節點D，若在時間為t_0時節點S知道節點D在位置L處。而當時間點為t_1時，節點D之移動平均速度為v，故節點S可假定節點D在時間點為t_1時之位址是在以L為中心點，v（t_1-t_0）為半徑之圓的範圍內，如下圖（a）所示。
  - 圖中灰色所涵蓋之範圍即為預期區域。然而，若節點D之實際移動速率大於平均速率，則在時間為t_1時，此目的節點D將可能會位於預期區域之外。
  - 另一方面，若節點S若能擁有較多關於可移動目的節點之資訊，則將可以產生出一個較很小的預期區域。例如節點S知道目的節點D欲往北移動，則此預期區域將會是圖（a）所示之圓的一半，如圖（b）所示。



需求式路由協定：
輔助定位路由通訊協定 (LAR)

- 所謂的「**要求區域**」是指：某一節點S欲找尋一條路徑至某一節點D，則此節點S會先設定一個要求區域，如下圖所示。

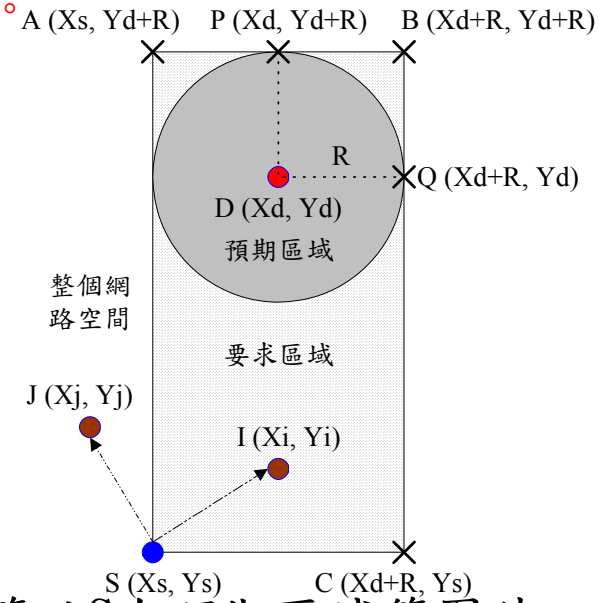


需求式路由協定：
輔助定位路由通訊協定 (LAR)

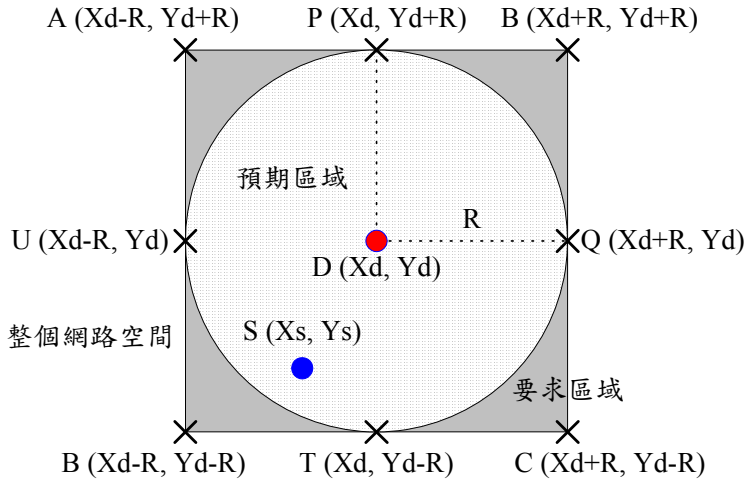
- 若來源節點S不在預期區域內，則從節點S到節點D之路徑上將會擁有預期區域外之節點。故此類節點所形成之額外的範圍將會被含括在要求區域中，如前圖（c）所示。以此類推節點S與節點D兩者會逐漸被至於要求區域

需求式路由協定：
輔助定位路由通訊協定 (LAR)

- 演算法 ：
  - 第一類LAR協定演算法：
    - 此方法使用一個矩形的要求區域來尋找路徑，如下圖所示。
    - 在第一類演算法中，首先需先定義要求區域為一個最小之矩形且其邊需個別平行於X軸與Y軸。而在此矩形中將包含著目前的節點S與預期區域。



（a）來源節點S在預期區域範圍外



（b）來源節點S在預期區域範圍內

需求式路由協定：
輔助定位路由通訊協定 (LAR)

- 一開始，來源節點S（Xs, Ys）會先計算出要求區域之四個角落。
- 在路徑要求訊息封包氾濫時，若有一非位於矩形內之節點收到此一訊息封包，則其將會丟棄此封包。否則，會將其再轉送至鄰近節點以尋找路徑。(限制廣播)

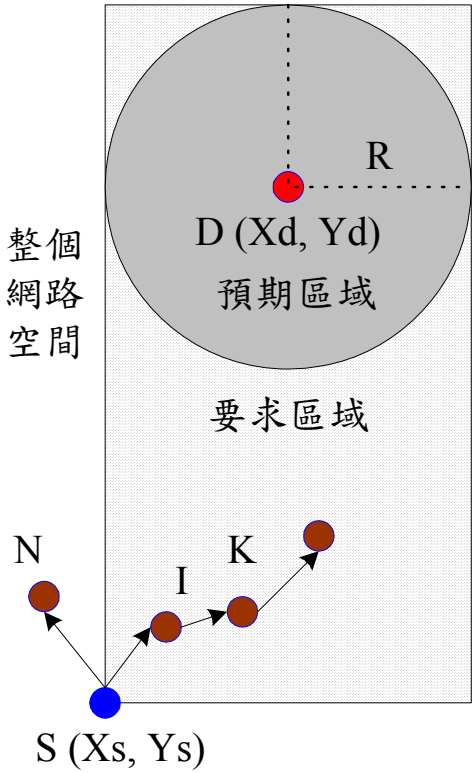
- 第二類LAR協定演算法：
  - 在第一類演算法中，來源節點會明確地在訊息封包中指出要求區域。
  - 在此類演算法中，要求區域已不再被需要，而是需要考慮與目的節點之幾何距離。
  - 來源節點所發出之路經要求訊息封包中將會包含兩筆資訊：
    - 在尋路之時，假設來源節點S知道目的節點D之位置為（X_d, Y_d）。則節點S會計算（X_d, Y_d）到它自己之距離（以DISTs表示），而此距離值將會被包含在節點S所發出之訊息封包中。
    - 此外，此訊息封包內亦會包含目的節點D之位置座標（X_d, Y_d）。

需求式路由協定：
輔助定位路由通訊協定 (LAR)

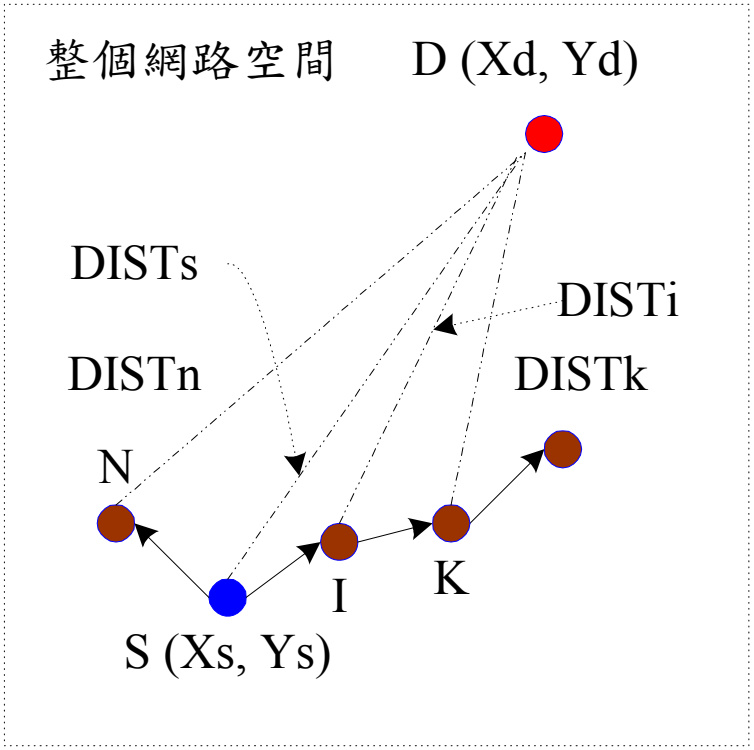
- 當某一節點I收到從節點S送出之訊息封包，此節點I將會計算（X_d, Y_d）到它自己之距離（以DISTi表示），同時再考慮：
  - 某一參數δ，若（DISTs＋$\delta \geq$DISTi），則節點I將會繼續廣播此訊息封包到鄰近節點，且此訊息封包包含了DISTi和（X_d, Y_d）之資訊。
  - 反之，若（DISTs＋$\delta \leq$DISTi），則節點I將丟棄此訊息封包。
- 當某一節點J收到從節點I送出之訊息封包：倘若節點J先前已接收過此相同之封包，則其會丟棄此封包。否則節點J將會計算（X_d, Y_d）到它自己之距離（以DISTj表示）。同時再考慮：
  - 若（DISTi＋$\delta \geq$DISTj），則節點J將會繼續廣播此訊息封包到鄰近節點，且此訊息封包中之DISTi值將被DISTj值所取代。
  - 反之，若（DISTi＋$\delta \leq$DISTj），則節點J將丟棄此訊息封包。

需求式路由協定：
輔助定位路由通訊協定 (LAR)

- 第一、二類LAR協定演算法之比較:



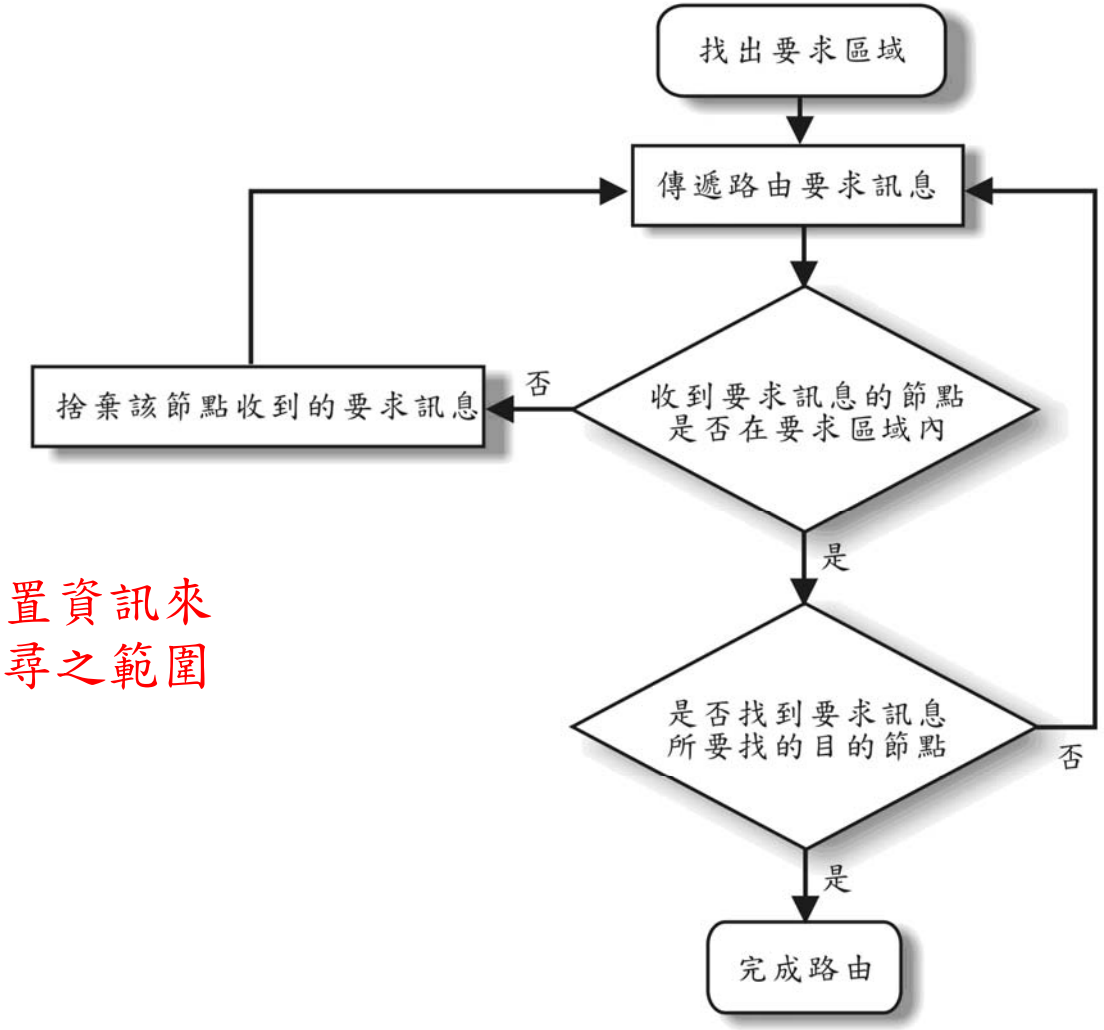
（a）第一類LAR協定演算法



（b）第二類LAR協定演算法

需求式路由協定：
輔助定位路由通訊協定 (LAR)

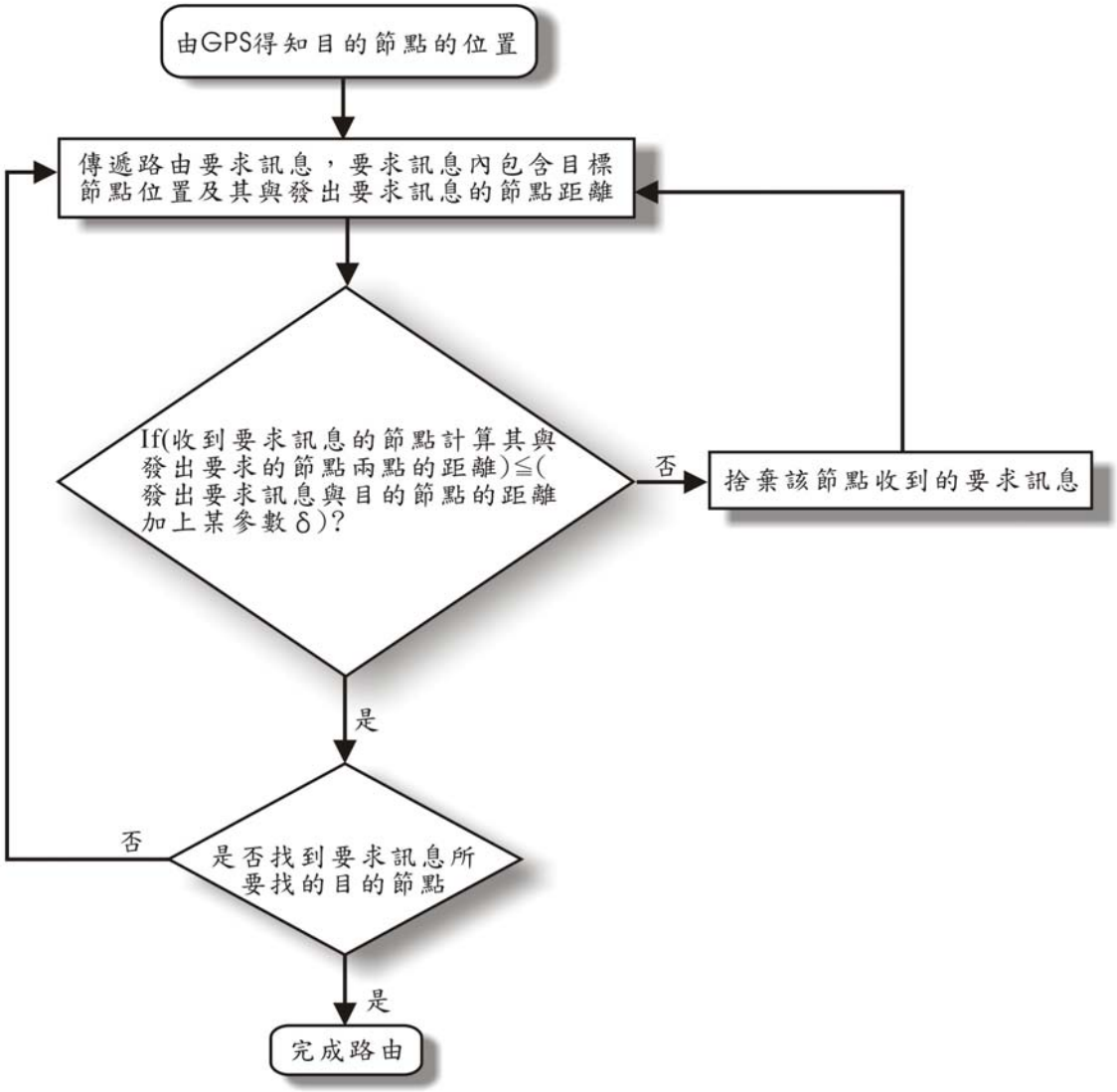
- 演算法流程 1：



LAR協定：利用位置資訊來
減少尋路時所需搜尋之範圍

需求式路由協定：
輔助定位路由通訊協定 (LAR)

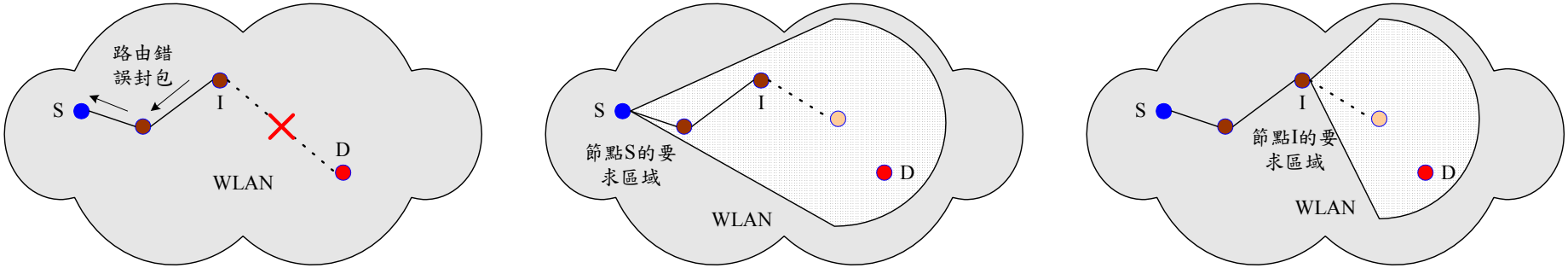
- 演算法流程 2：



LAR協定：藉由全球定位系統來尋找路徑

需求式路由協定：
輔助定位路由通訊協定 (LAR)

- 區域搜尋（Local Search）：
  - 在LAR協定中，任何中繼節點（如節點I）偵測到路徑發生錯誤或損壞時，此節點I會送出一個路徑錯誤封包給來源節點S，如下圖所示。
  - 當來源節點S收到此封包，其將會重新進行另一個新路徑的初始化程序，以找尋另一條可到達目的節點D之新路徑。
  - 然而，因從來源節點S至節點I之間之路徑並無損壞，故初始化之尋路範圍將僅被侷限於節點I與目的節點D之間。



ZRP
(Zone Routing Protocol)

- The Zone Routing Protocol (ZRP) for Ad Hoc Networks
- Cornell University
- Z.J. Haas and M.R. Pearlman
- draft-ietf-manet-zone-zrp-01.txt, 1998

ZRP Outline

- Hybrid of table-driven and on-demand!!
- From each node, there is a concept of "zone".
  - Within each zone, the routing is performed in a table-driven manner (proactive).
  - However, a node does not try to keep global routing information.
- For inter-zone routing, on-demand routing is used.
  - This is similar to DSR.

ZRP Example

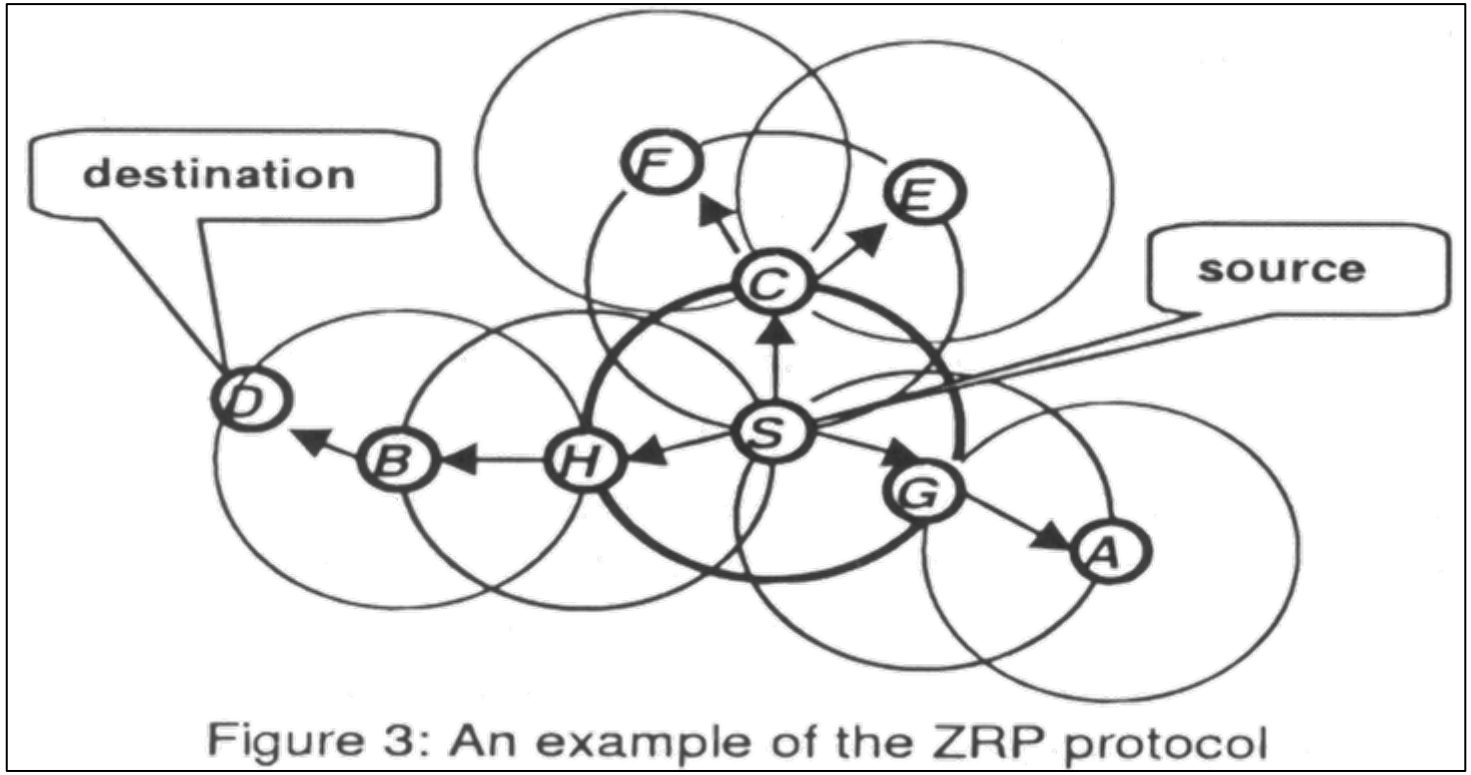
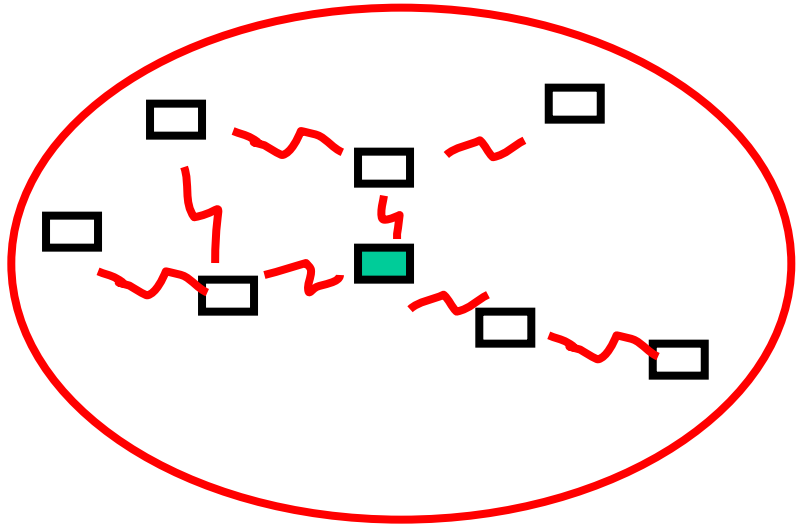


Figure 3: An example of the ZRP protocol

Route Discovery

- By an operation called "boardercast":
  - sending the route-request to boarder nodes



結論與比較

- 集中式與分散式操作。
- 主動式與需求式路由方式。
- 週期性更新。
- 多路徑維護。
- 單向/雙向路徑。
- 平面/叢集。
- 分組轉送機制。
- 路徑度量選擇。
- 特殊節點。
- 特殊硬體需求。
- 支援群播功能 。
- 服務品質（QoS）。

結論與比較

	DSDV	CGSR	WRP	LMR	TORA	DSR	AODV	ABR	LAR
分散式操作	是	是	是	是	是	是	是	是	是
主動需求	主動	主動	主動	需求	需求	需求	需求	需求	需求
週期性更新	是	是	是	是	否	否	否	否	是
維護多條路徑	否	是	否	是	是	是	否	否	是
單向/雙向路徑	雙向	雙向	雙向	單向	雙向	單向	雙向	雙向	雙向
省能源	否	否	否	否	否	否	否	否	否
平面群集	平面	群集	平面	平面	平面	平面	平面	平面	平面
分組轉送機製	逐跳	逐跳	逐跳	逐跳	逐跳	源路由	逐跳	逐跳	逐跳
提供安全機製	無	無	無	無	無	無	無	無	無
路由度量選擇	最短路徑	最短路徑	最短路徑	最短路徑	最短路徑	最短路徑	最短路徑	最短路徑	最短路徑
存在特殊節點	否	有	否	否	否	否	否	否	否
特殊硬體需求	否	否	否	否	GPS	否	否	否	否
支援群撥功能	否	否	否	否	否	否	否	否	否
Qos	否	否	否	否	否	否	否	否	否