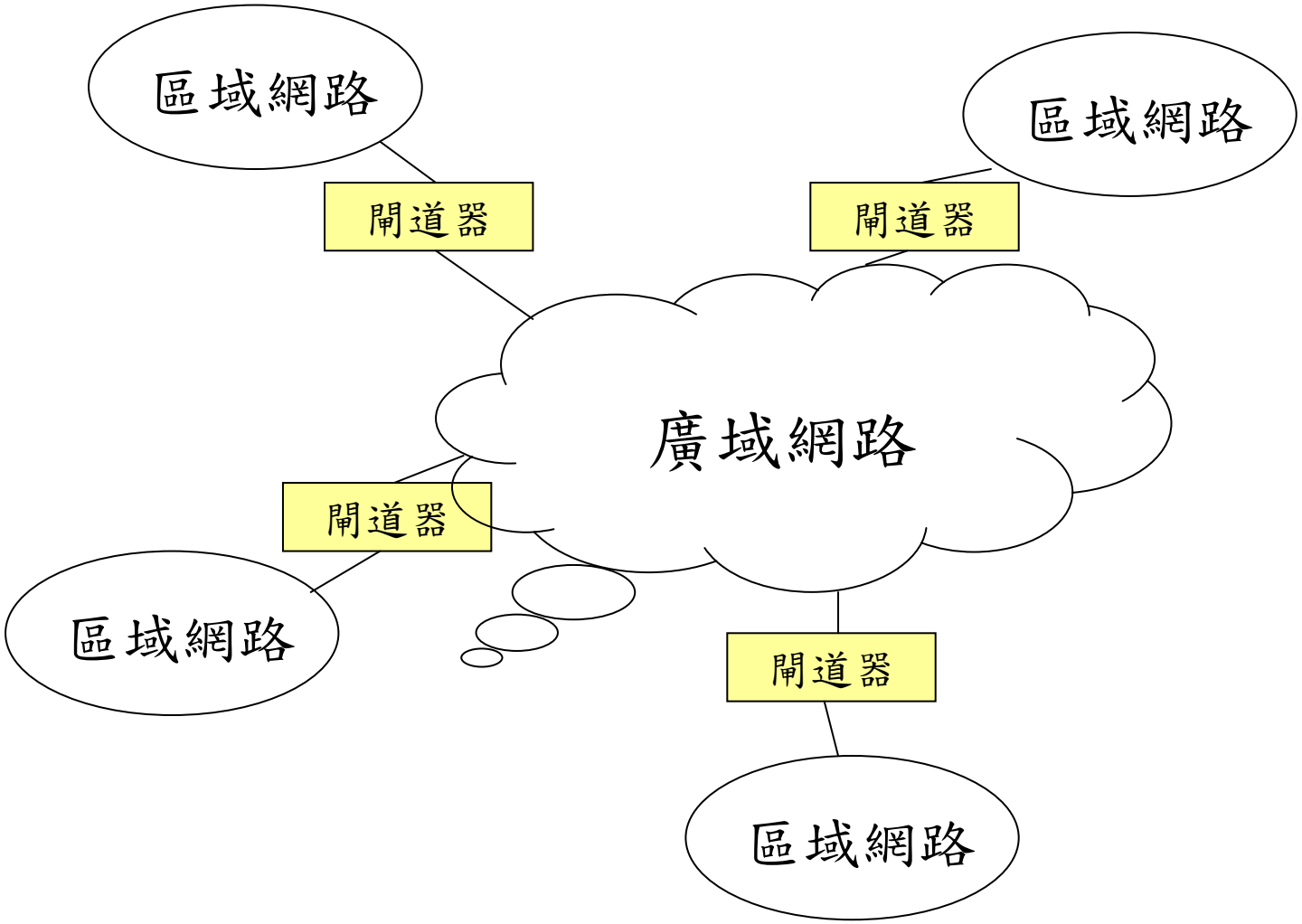
教育部顧問室通訊科技教育改進計畫專案獎助

第五章
網路路由基本概念

區域網路與廣域網路

- 與區域網路比較，廣域網路通常涵蓋較大的地理區域，而其主要的目的是要做為區域網路與區域網路間互連的橋樑。
- 在廣域網路的架構中，我們可將經由網路閘道器（Gateway）所接上來的各個區域網路視為子網路（Sub networks）。

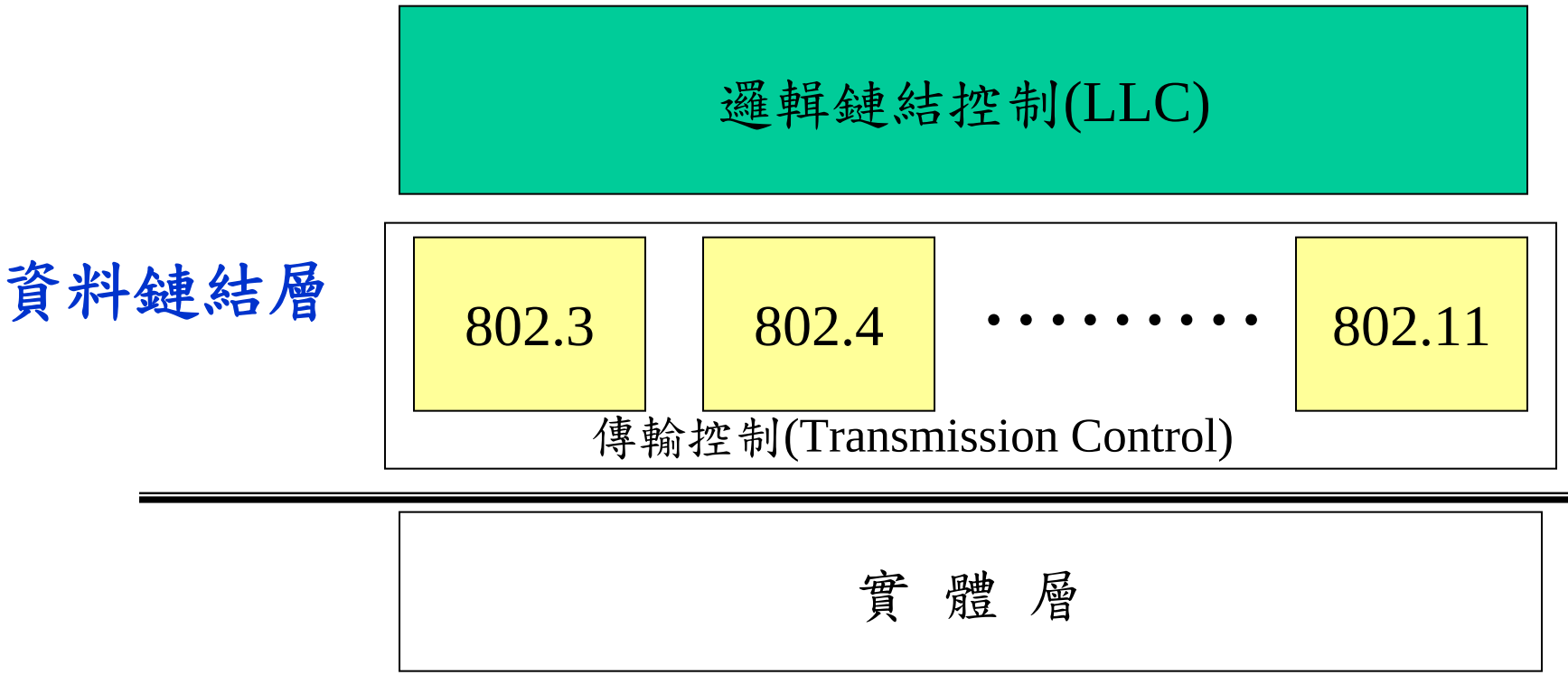
區域網路與廣域網路



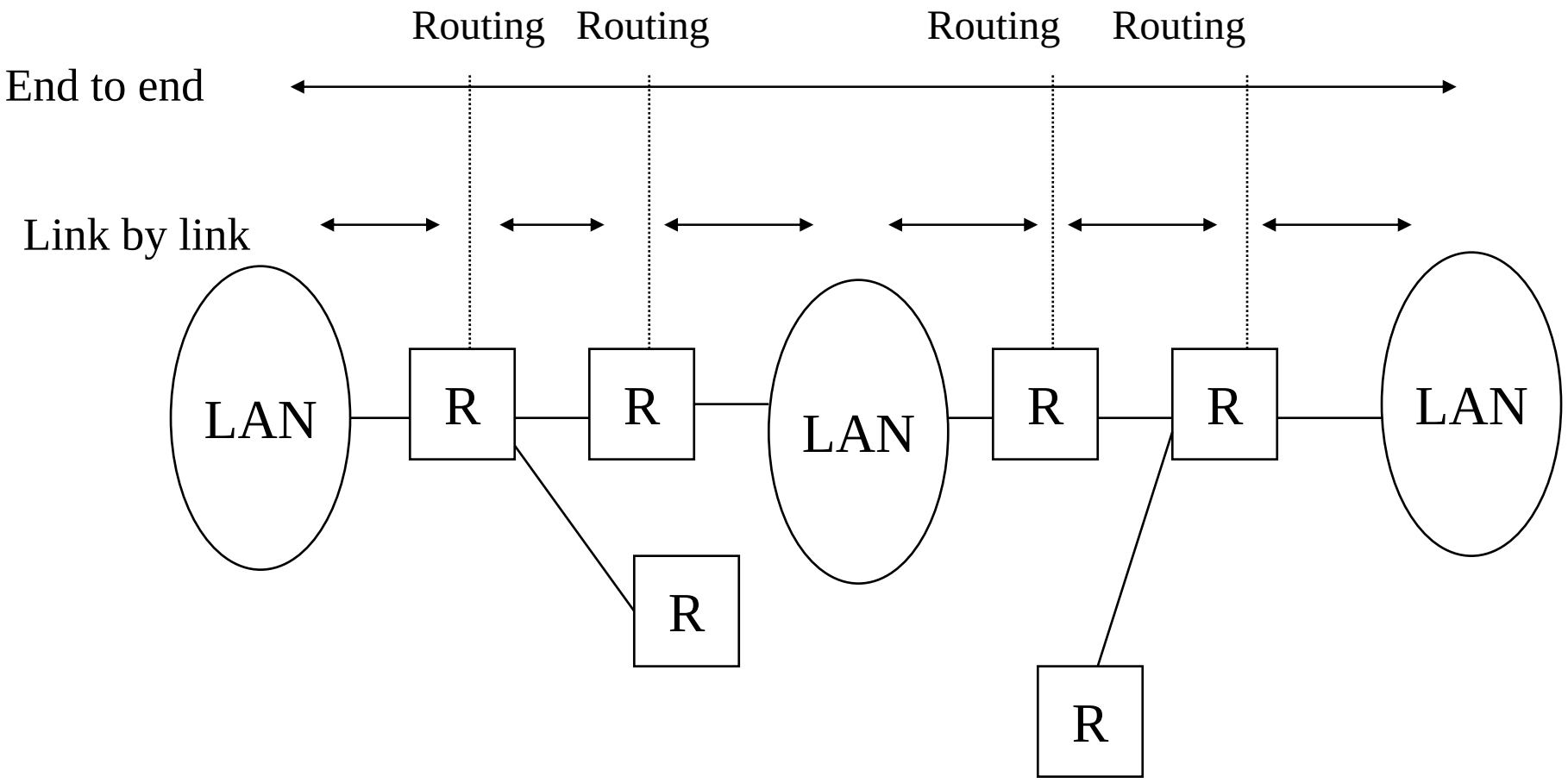
區域網路與廣域網路

- 區域網路互連
  - 區域網路有一項很重要的特性，便是在實體層（Physical Layer）的技術上具備廣播的特性，不管是環狀、無線、或匯流排狀的區域網路拓撲，基本上，網路上的任一台電腦或終端設備所送出之資料，在同一網段範圍內的設備均可收到。
  - 區域網路較屬於鏈路間（Link by link）的問題；而廣域網路已將許多區域網路互連在一起，因此所需解決之問題較屬於選擇好的路由(routing)，將資料封包由起始點到終點（End-to-End）的網路層問題。

鏈結層架構

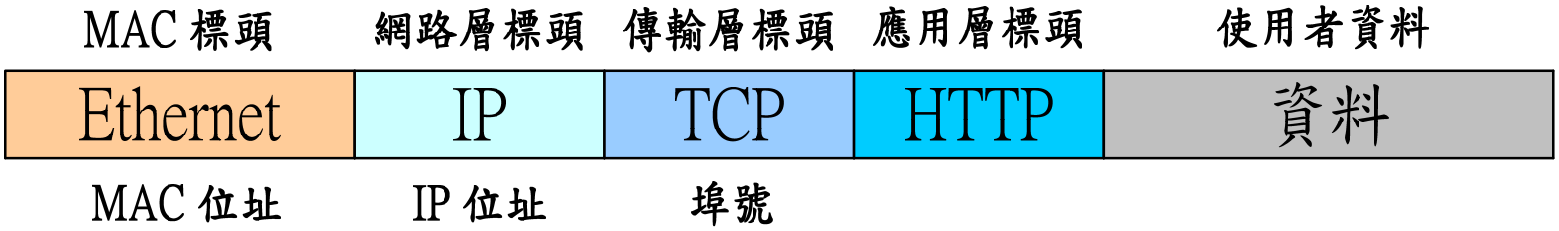

資料鏈結層
邏輯鏈結控制(LLC)
802.3
802.4
802.11
傳輸控制(Transmission Control)
實 體 層

Layer 2 v.s. Layer 3



網路網路4-層協定堆疊

5 - 7	應用層	SMTP, HTTP, DNS, FTP, TELNET, TFTP	用戶/伺服 或 點對點
4	傳輸層	TCP, UDP	行程對行程
3	網路層	IP	主機對主機
1 - 2	實體層&資料鏈結層	PPP, SLIP	在鏈結之上



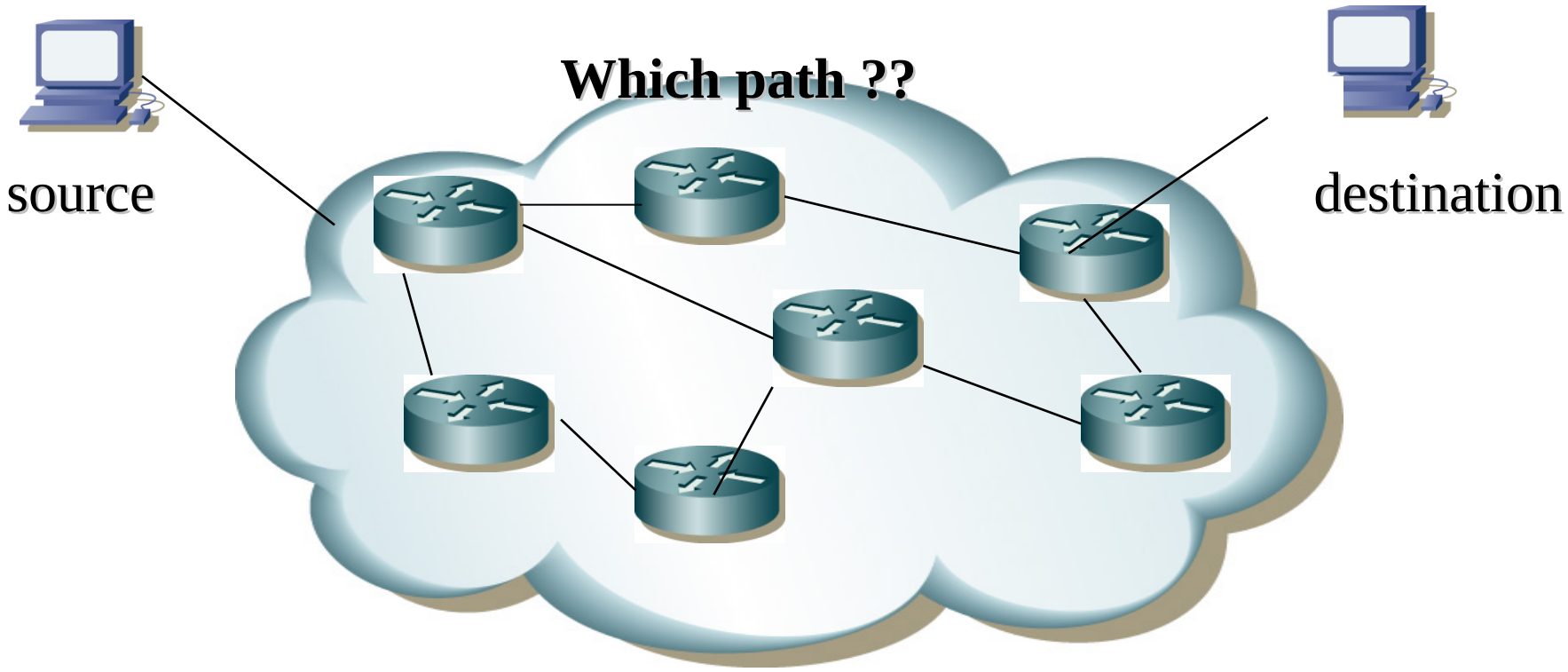
RFC 的標準

層		RFCs	名稱	內容
5 - 7	應用層	318, 435, 495 114, 354, 959 1866, 2068,1045, 2069	TELNET FTP HTTP	遠端登入協定 檔案傳輸協定 超文件傳輸協定
4	傳輸層	675, 761,793, 896 768	TCP UDP	傳輸控制協定 使用者資料包協定
3	網路層	760, 791,815, 1154 777, 792 1245-1247	IP ICMP OSPF	網路協定 網路控制訊息協定 開放式最短路徑優先路由協定
2	資料鏈結層	1661-1663 1055	PPP SLIP	點對點協定 串列線路網路協定

路由選擇功能

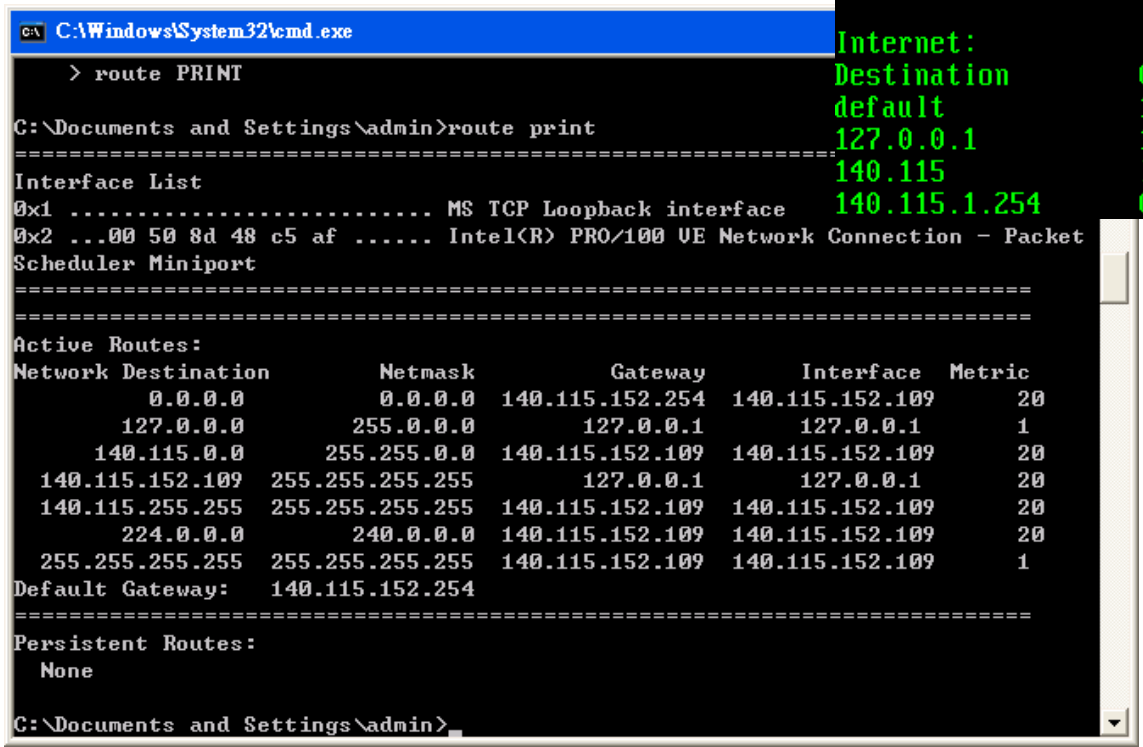
- 網際網路是由許多網路所構成的互連網路，從某一起始點到某一終點的路徑可以有很多選擇
- 路由選擇的主要任務就是要從中選擇一條最恰當的路徑
  - 選擇最佳路由路徑（optimal routing paths）能力
  - 有效的封包傳送能力

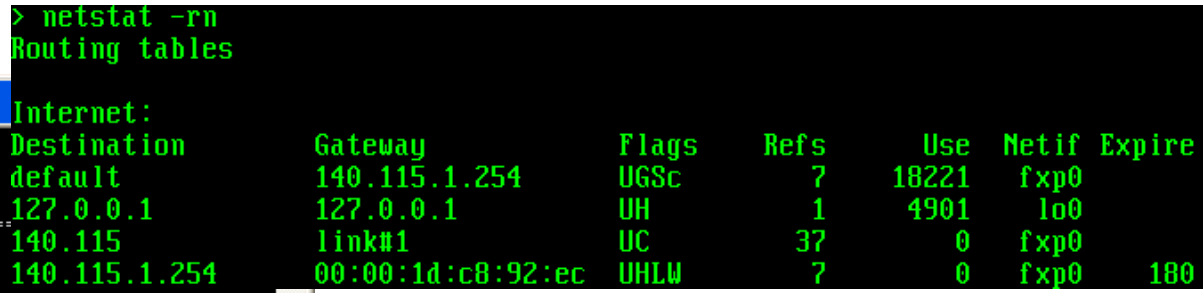
What is routing



路由選擇功能

- 網路上的路由器（Router）利用選徑演算法得到其到各節點之最短路徑後，這些資訊通常會存放在路由表（Routing Table）中。在實務上，要瞭解資料封包的傳送路徑，一般使用者可以在Windows系統中，下route print顯示出其路由表





路由選擇功能

- 路由演算法常見的衡量標準有：
  - 頻寬：鏈結的資料容納量
  - 延遲：將封包沿各連結來源移動到目的地所需的時間
  - Load：路由器忙碌與否
  - 可靠度：通常是指各網路連結的錯誤率
  - hop count：封包在送達目的地之前，需經過的路由器數目
  - 時脈數：使用 IBM PC 時脈頻率（約為 55 毫秒）計算之資料連結上的延遲
  - 成本(距離)：由網路管理員指定，通常是根據線路成本或其他估算而定

路由演算法的好壞考量因素

- 最佳化：根據指標的權值來計算最佳路徑的能力
- 簡單：如果路由演算法的設計越簡單，則執行效能通常越好，而且不管是硬體或是軟體的成本與製作上都會容易許多
- 穩定：在出現不正常或不可預測的事件時，仍然能保持繼續正常運作
- 收斂快：路由演算法收斂速度要盡量快，convergence是所有網路設備達到具有相同的網路資訊所需的時間

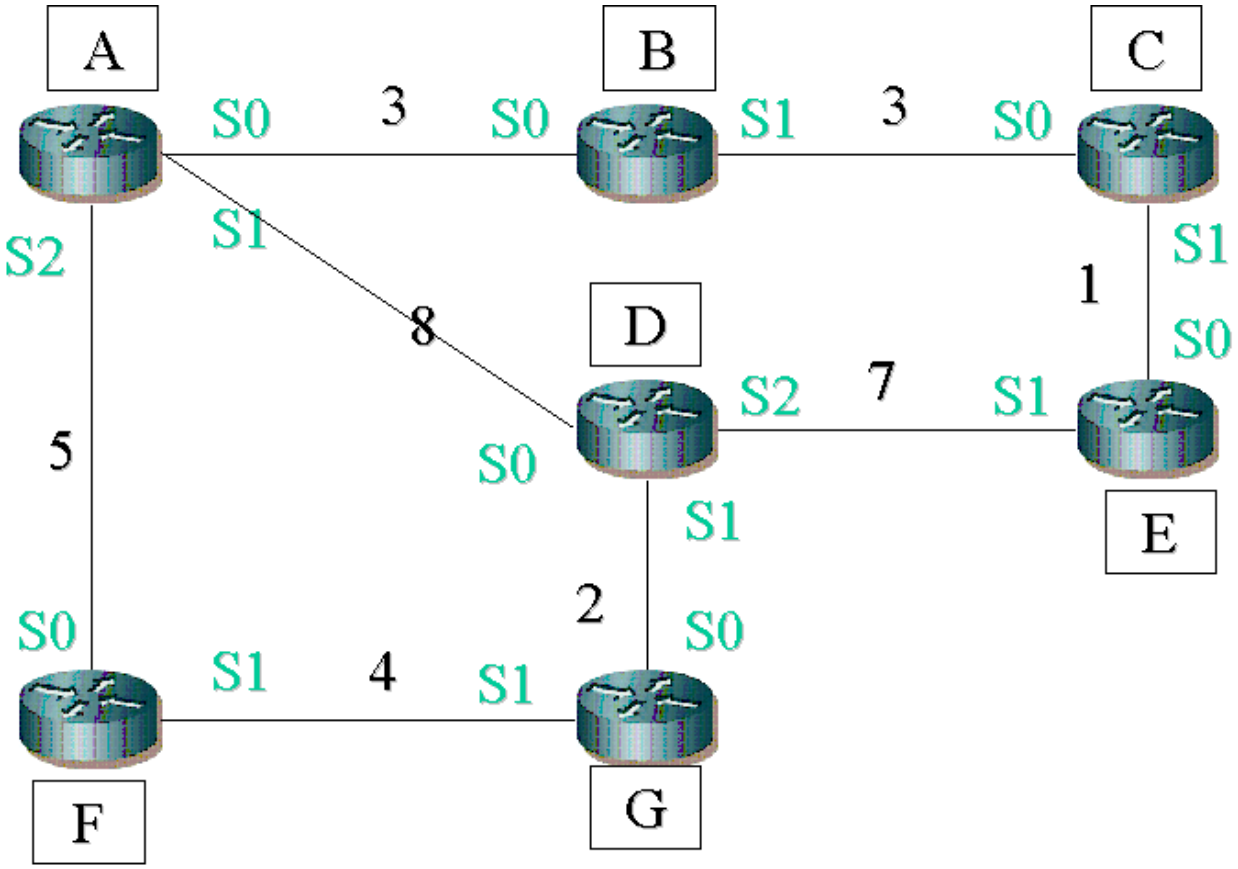
距離向量（Distance）與鏈結狀態（Link state）路由演算法

- 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

- 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

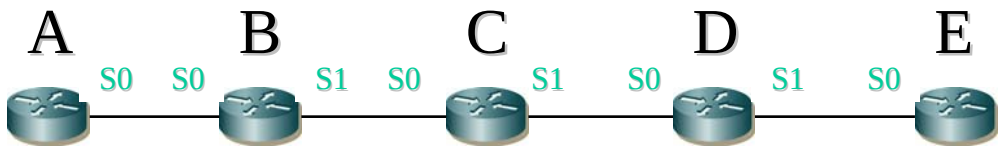
From F to	Interface	Metric
F	local	0
A	S0	1
G	S1	1
B	S0	2
D	S0	2
C	S0	3
E	S1	3

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

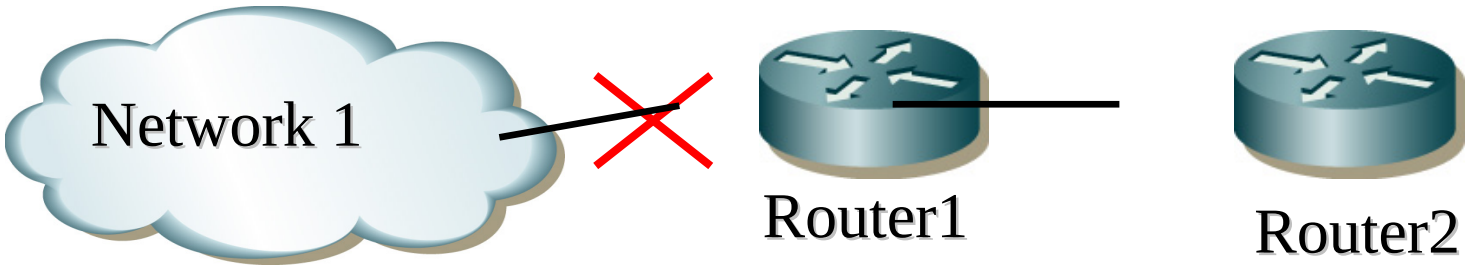
然而過一段時間，C 必須再次跟鄰居交換訊息，發現到 A 這個 entry 必須更新，於是 C 就更新自己的路由表，如下表

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

- 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

其他路由演算法

- Optimal routing
  - Traffic may be splitted at some strategic points to “smooth” delay and to increase network throughput
- Hot potato (deflection) routing
  - To minimize buffer overflow and to reduce the packet loss

內部與外部路由

- Routing時，每一節點都需要將其所獲得的資訊以氾濫（Flooding）的方式傳送出去，而傳送出去後到每一節點都獲得此訊息時，該筆訊息便不會在網路上繼續漫延，稱為收斂
- 若網路的規模太大，則將影響收斂的時間，且非常沒有效率

→ 分層次(內部/外部)選擇路由

內部與外部路由

- 將網路分割成較小規模的網路，稱為自治系統（Autonomous System，簡稱AS）
- AS為受同一個權責單位管理的網路，包含一個或多個路由器，路由資訊可以在同一個或不同的自治系統間散播；一個自治系統可以是一個校園網路，也可以是ISP業者的網路，而且一自治系統也可以被另一自治系統所涵蓋
- 單一自治系統內部所散播路由資訊的協定稱為內部路由協定（Interior Gateway Protocol，簡稱IGP）
- 跨自治系統散播路由資訊的協定稱為外部路由協定（Exterior Gateway Protocol，簡稱EGP）

內部與外部路由

- 內部路由協定主要是以最短路徑為主要訴求
- 外部路由協定雖然也考慮最短路徑的原則，但也需要有策略方面的考慮

內部與外部路由