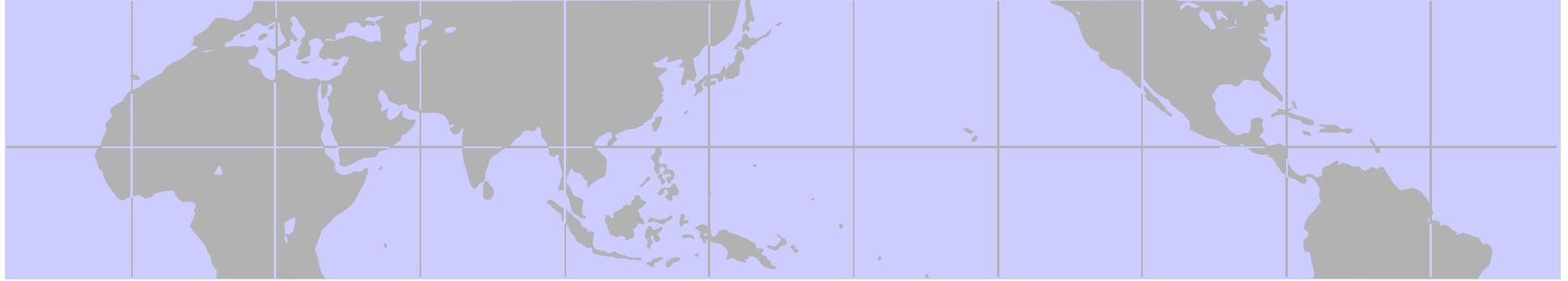


Introduction to Computer Network (電腦網路概論)

Prof. Ruo-Wei Hung (洪若偉)

Department of Computer Science and Information Engineering,
Chaoyang University of Technology,
Wufeng, Taichung 41349, Taiwan
E-mail: rwhung@cyut.edu.tw

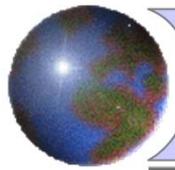




Course Introduction

(課程介紹)





任課教師



洪若偉 (Ruo-Wei Hung)

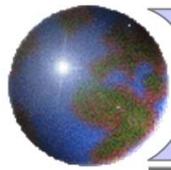
- E-mail: rwhung@cyut.edu.tw
- Web-site: <http://www.cyut.edu.tw/~rwhung>
- Tel: 04-2332-3000 ext. 7758
- Office Hours:

週二 13:30~15:30 (E724研究室)

週四 13:30~15:30 (E724研究室)

■ Personal Information

- ▶ 一妻一子一女
- ▶ 一屋一田四車(含腳踏車、機車)一貸款一基金

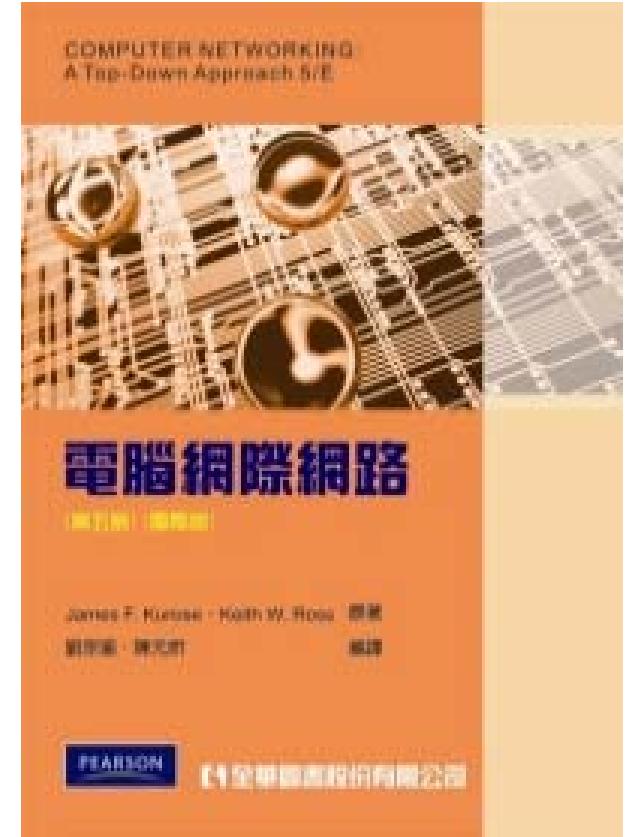


教科書



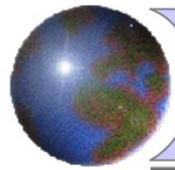
■ 教科書(Textbook) :

1. 劉宗瑜、陳元忻譯, “電腦網際網路 5/e”, 全華圖書, 2010年. (ISBN: 978-986-154-977-4)
2. Kurose and Ross, “Computer Networking—A Top-Down Approach, 5/e”, Pearson Addison-Wesley, 2010. (歐亞書局代理)
3. 投影片 (Slides)



■ 參考書目(Reference Books):

1. 陳湘陽, “網路概論 2/e”, 博碩, 2009. (ISBN: 978-986-201-187-4)
2. 陳惠貞/周念湘, “網路概論”, 基峰, 2007. (ISBN: 978-986-181-201-4)



上課方式

1. 投影片(挖洞填空)+題目講解 (課堂上課)。
2. 隨時隨堂考。(不能補考)





Chapter 1: Computer Networks and the Internet (電腦網路與網際網路)

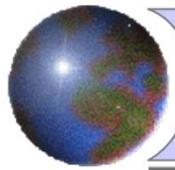
Chapter I: Network Media and Devices (網路媒介與設備)

Chapter 2: Application Layer (應用層)

Chapter 3: Transport Layer (傳輸層)

Chapter 4: Network Layer (網路層)

Chapter 5: Link Layer and Local Area Networks (連結層與區域網路)



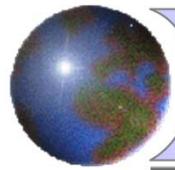
How to Solve a Problem or Study a New Topic?

■ 2W-1H-1E

- ▶ What? (Network 是什麼?)
- ▶ Why? (為何需要 Network?)
- ▶ How? (如何設計、安裝、使用 Network?)
- ▶ Extension! (如何應用 Network?)

■ 5W-2H思考法

■ 九宮格思考法



Why do We Need to Study Network ?



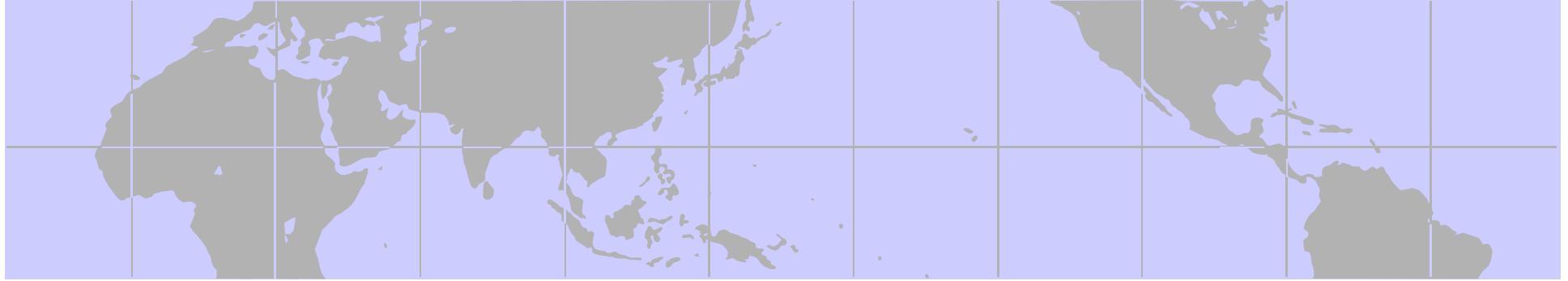
■ 重要性

- ▶ 網路 (Network) 是終端設備透過通訊媒介及通訊設備來溝通的一種現代化電腦應用。
- ▶ 電腦網路需要硬體及軟體互相搭配。
- ▶ 電腦網路需要硬體基礎設施，搭配軟體以達到終端設備通訊的目的。
- ▶ 電腦網路的應用無遠弗界。
- ▶ 電腦網路已成為人類不可或缺的生活必須品。

■ 課程目標

- ▶ 學習電腦網路系統的運作。
- ▶ 學習電腦網路系統的組成。
- ▶ 學習電腦網路系統的設計。
- ▶ 學習電腦網路系統的應用。

End of Introduction



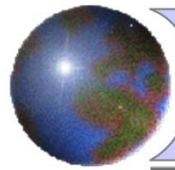
Chapter 1



Computer Networks and the Internet

(電腦網路與網際網路)





Topic Overview

1.1 什麼是網際網路 (Internet) ? (網路協定–protocol)

1.2 網路的邊際(Network Edge)

1.3 網路的核心 (Network Core)

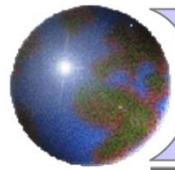
1.4 網路的效能 (Network Performance)

1.5 協定分層及其服務模型 (Protocol Layers and Their Service Models)

***1.6** 面臨攻擊的網路 (Networks Under Attack)

***1.7** 網路的歷史 (History of Networks)

[*] Option



第一章 導覽流程

1.1 什麼是網際網路 (Internet) ?

1.2 網路的邊際 (Network Edge)

- ▶ 終端系統、連線網路、連結

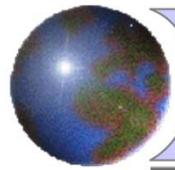
1.3 網路的核心 (Network Core)

- ▶ 電路交換、封包交換、網際網路結構

1.4 網路的效能 (Network Performance)

- ▶ 封包交換網路的延遲、封包遺失和產出率

1.5 協定分層及其服務模型 (Protocol Layers and Their Service Models)



1.1 什麼是網際網路 (Internet) ?



■ 基本元素的描述

- ▶ Interconnected networks
- ▶ 幾百萬台連接的計算裝置：主機 (host) = 終端系統 (end system)
 - 執行網路的應用程式
- ▶ 通訊連結 (link , wired/wireless)
 - 雙絞線、銅線、光纖、無線電、衛星
 - 資料傳輸速率 = 頻寬 (bandwidth, bps)
- ▶ 路由器 (router) / 交換器 (switch)：傳送封包 (packet, 小段的資料)



■ 基本元素的描述 (Cont.)

▶ host/end system

- : PC
- : 伺服器
- : 行動裝置
- : 行動電話

▶ communication links

- : 存取點
- : —— wired links

▶ router/switch

- : 路由器

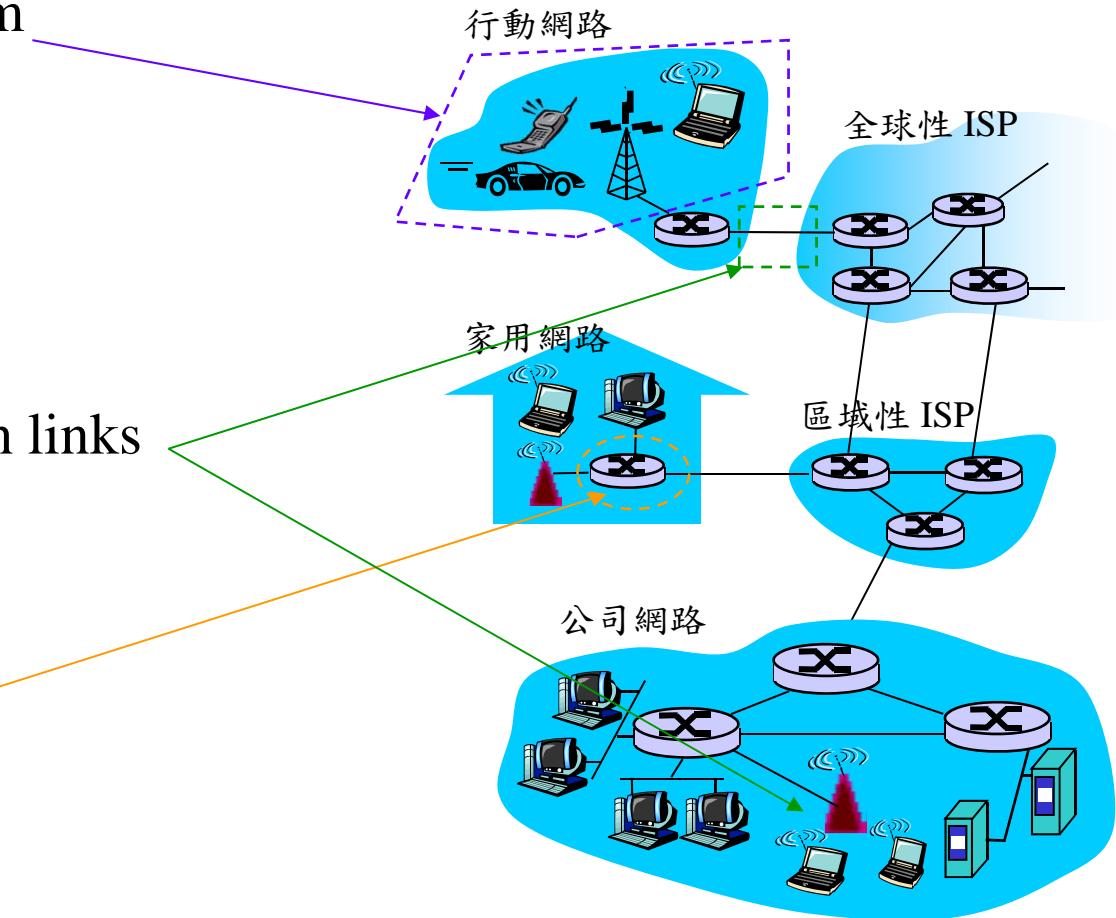
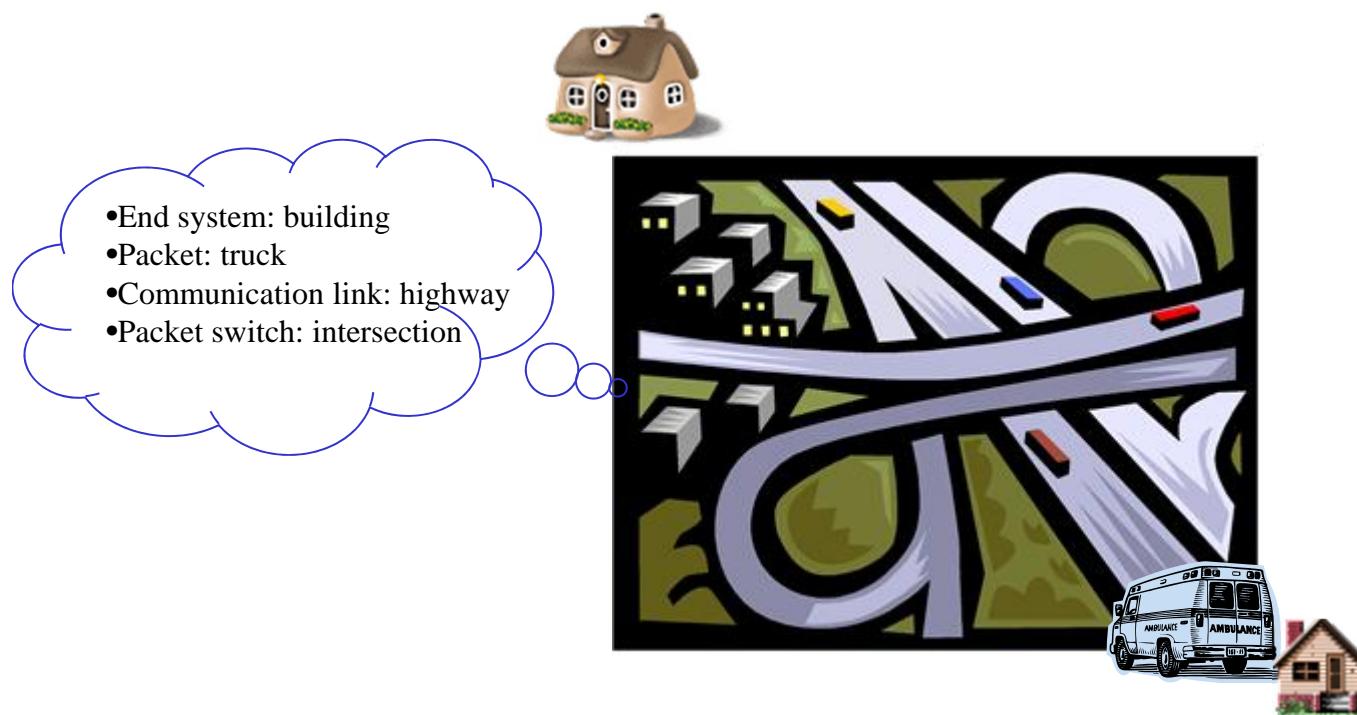


圖1.1 網際網路的元素

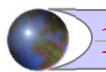


■ 基本元素的描述 (Cont.)

◆ 封包交換網路 vs. 運輸網路：



Ex 1.1



■ 網際網路的應用(p. 1-10)

- ▶ IP 相簿 (photo sharing; <http://www.ceiva.com/>)



- ▶ 網路電話 (skype; <http://www.skype.com>) 





■ 什麼是網際網路：從元件來看

▶ 協定(protocol)控制訊息的傳送、接收

- Handshaking
- 所有人必須對所有協定的運作方式有所共識

○ 例如：TCP、IP、HTTP、網際網路電話協定、乙太網路協定

▶ 網際網路：“網路組成的網路” (interconnected networks)

○ 不嚴謹的階層性

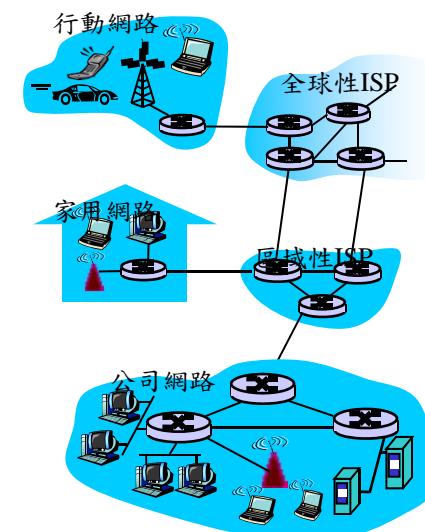
○ 公眾網際網路(public)與私有內部網路(intra-net)

▶ 網際網路標準

○ RFC : Request for comments (<http://www.ietf.org/rfc.html>)

○ IETF : 網際網路工程工作小組 (Internet Engineering Task Force; <http://www.ietf.org>)

Hung2 ○ Other standards: IEEE org.



投影片 8

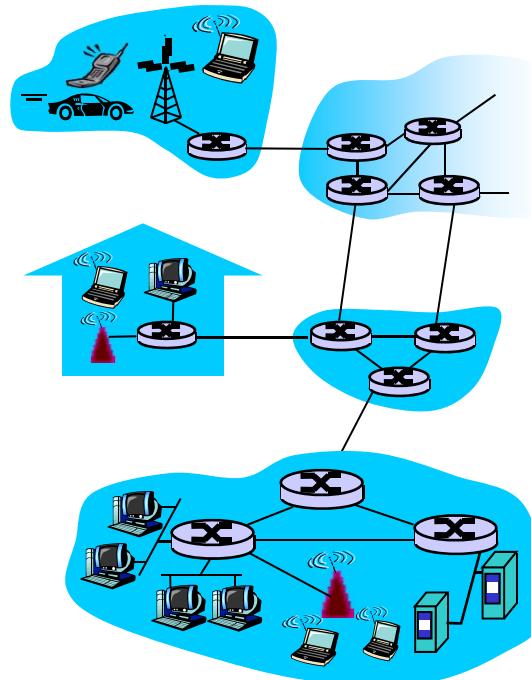
Hung2

Institute of Electrical and Electronics Engineers
Ruo-Wei Hung, 2013/8/25



■ 什麼是網際網路：從服務來看

- ▶ 通訊的架構允許分散式的應用程式
 - 網頁、網路電話、電子郵件、遊戲、電子商務、檔案分享
- ▶ 提供給應用程式的通訊服務
 - 可靠的將資料由來源送達目的地 (TCP – reliability)
 - “盡其可能” 的資料交付 (UDP – un-reliability)





■ 什麼是協定(protocol)

► 人們的協定

- “請問現在幾點？”
 - ... 傳送特定的信息
 - ... 當收到訊息或其它事件時，採取特定的動作

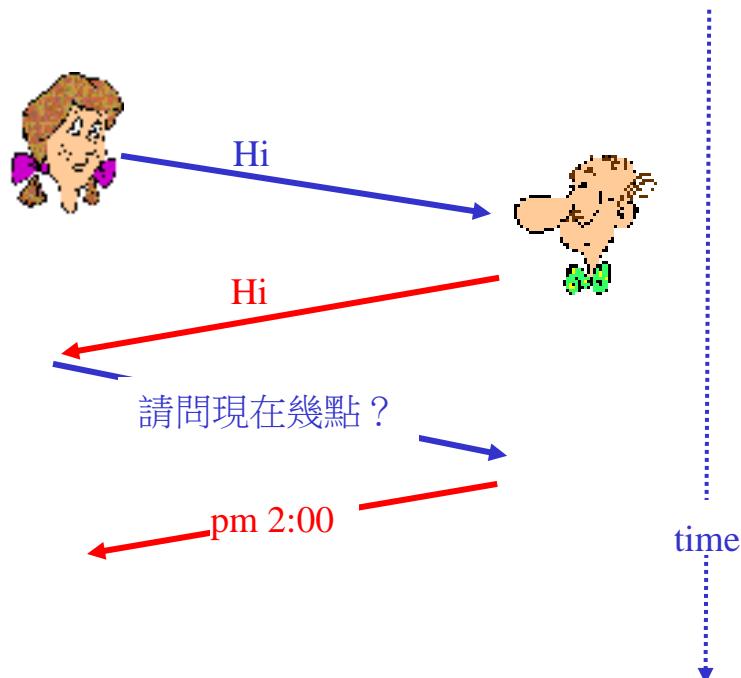
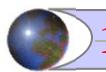


圖1.2(a) 人類的協定



■ 什麼是協定(Cont.)

► 電腦網路的協定

- 機器而非人類
- 網際網路上的所有通訊活動都由協定來管理

1. 協定(protocol)定義兩個以上通訊實體間交換訊息的格式與順序，以及傳送與接收訊息或發生其它事件時所採取的動作。
2. 不同的協定用來完成不同的通訊工作。

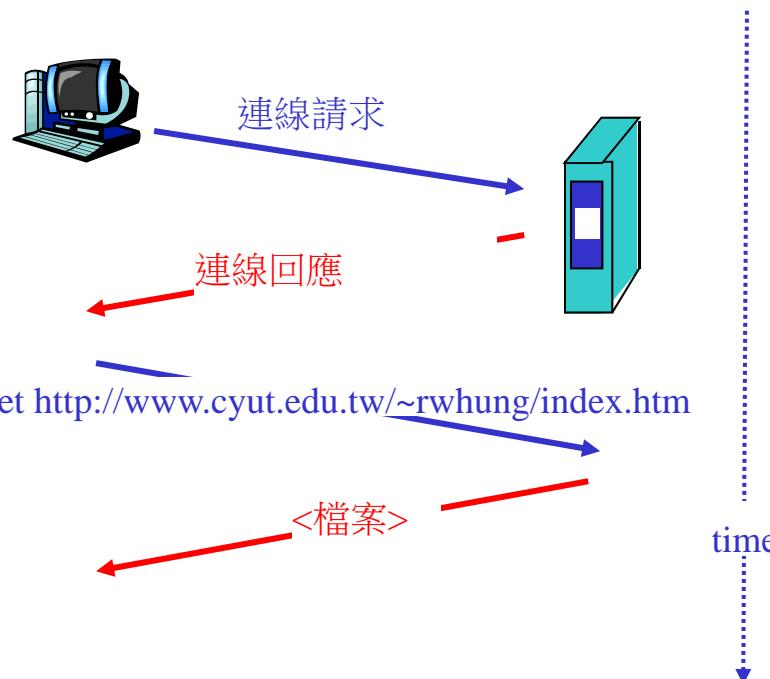
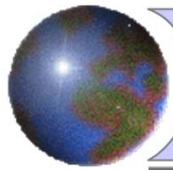
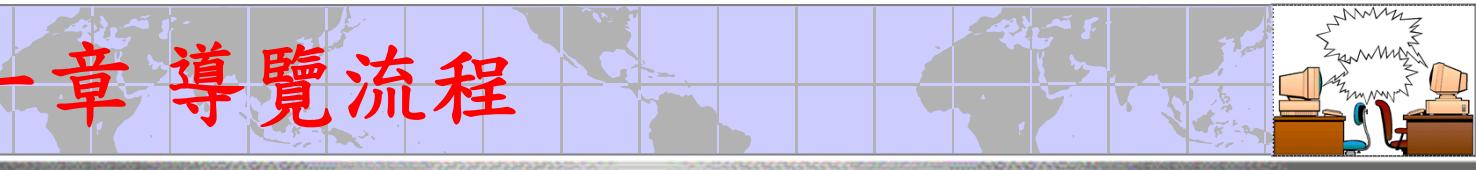


圖1.2(b) 計算機網路的協定





第一章 導覽流程



1.1 什麼是網際網路 (Internet) ?

1.2 網路的邊際 (Network Edge)

- ▶ 終端系統、連線網路、連結

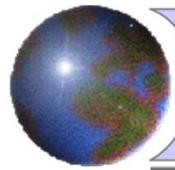
1.3 網路的核心 (Network Core)

- ▶ 電路交換、封包交換、網際網路結構

1.4 網路的效能 (Network Performance)

- ▶ 封包交換網路的延遲、封包遺失和產出率

1.5 協定分層及其服務模型 (Protocol Layers and Their Service Models)



1.2 網路的邊際 (Network Edge)



■ 細看網路結構

► 網路的邊際 (edge)

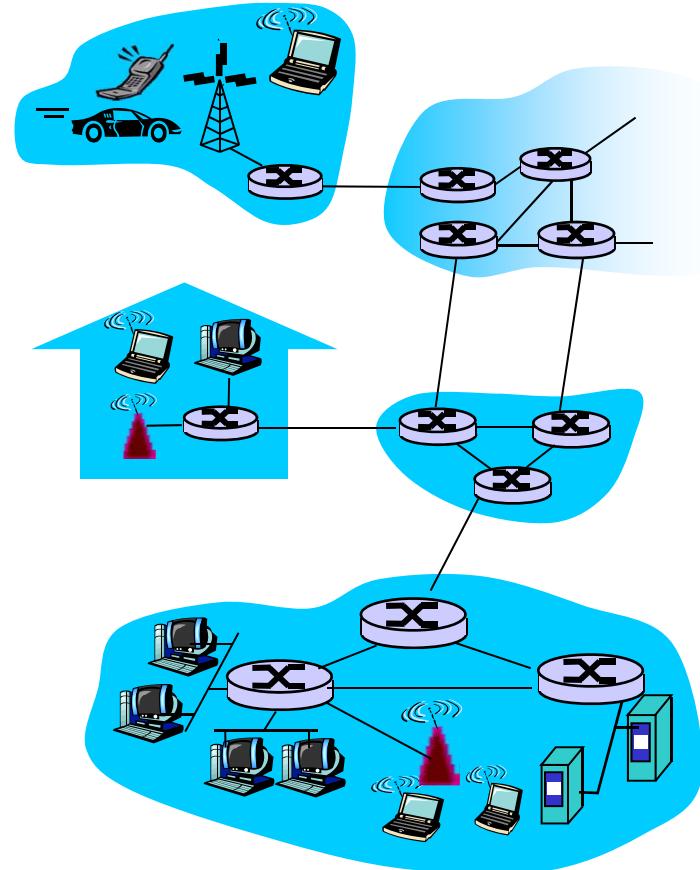
- 應用程式與主機 (1.2節)
- 網路應用 (p.1-10)

► 連線網路與實體媒介

- 有線、無線通訊連結 (Chapter 6)

► 網路的核心 (core)

- 互相連接的路由器/交換器 (1.3節)
- 網路的網路





■ 網路的邊際

► 終端系統 (主機)：

- 執行應用程式 -

例如：網頁、電子郵件

- 位於“網路的邊緣”

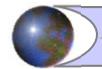
► 基本的終端系統互動模型

- 用戶端/伺服器模型 (client-server model)

- 用戶端主機從隨時開啟的伺服器端要求及接收服務
- 例如：網頁瀏覽器/伺服器、電子郵件用戶端/伺服器
- 分散式應用程式 (distributed application)

- P2P對等式模型 (peer-peer model)

- 使用最少的(或不使用)專用伺服器
- 例如：網際網路電話、BitTorrent、eMule、Limewire



■ 網路的邊際 (Cont.)

► 基本的終端系統互動運作模型 (Cont.)

- 用戶端/伺服器模型
- P2P對等式模型

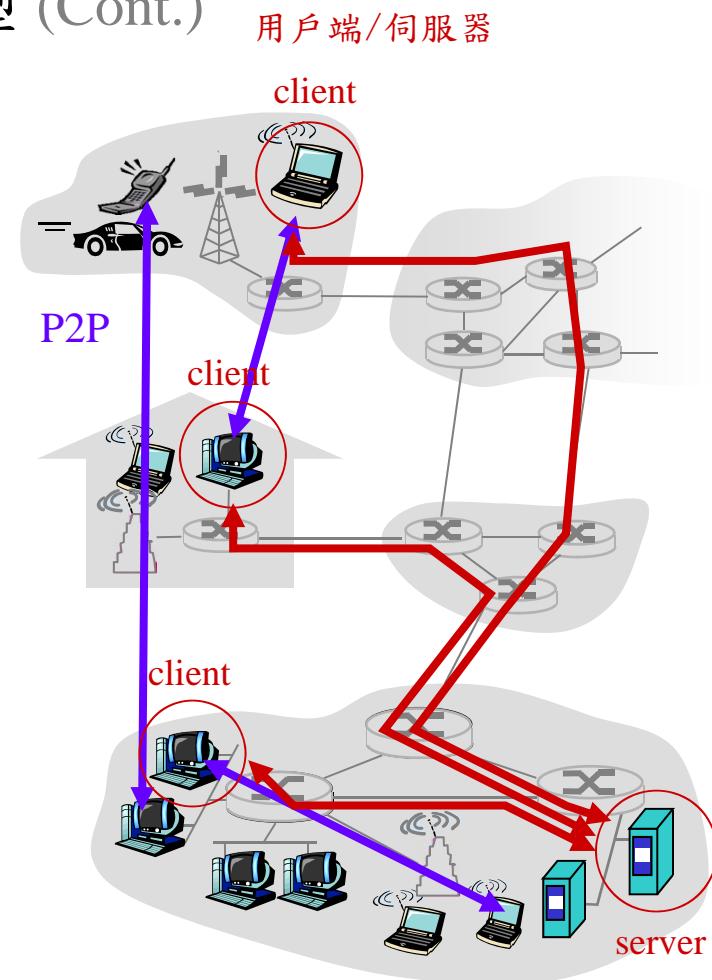


圖1.3 終端系統互動運作模型

Ex 1.3



■ 網路的邊際 (Cont.)

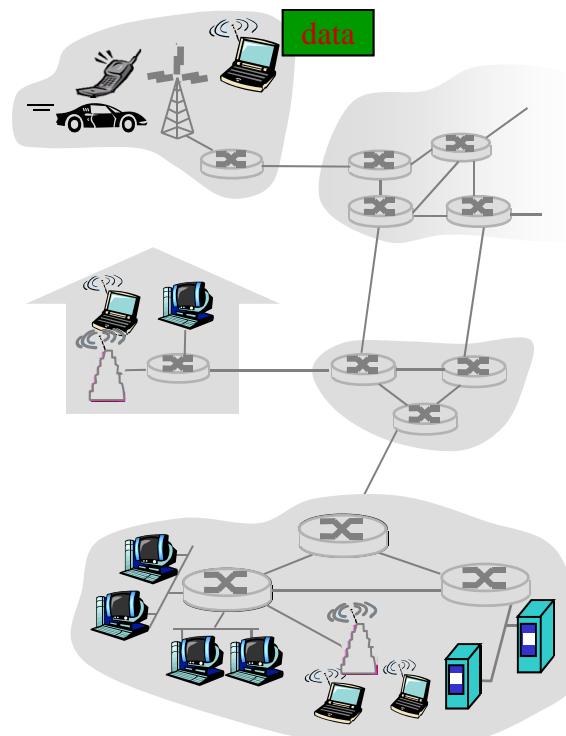
► 終端系統間的資料傳輸服務

- 可靠的資料傳輸服務

- 不允許資料傳輸有錯誤

- 盡其可能的資料傳輸服務

- 盡其可能的將資料放入網路來傳輸，但允許資料傳輸有錯誤





■ 網路的邊際 (Cont.)

► 終端系統間的資料傳輸服務 (Cont.)

○ 可靠的資料傳輸服務

- 交握程序(handshaking)：設定(準備)，在資料的傳輸之前
 - » Hello、hello 的回應－人類的回應
 - » 在兩個通訊的主機之間設定“狀態”
- TCP – Transmission Control Protocol (Chapter 3)
 - » 網際網路的可靠資料轉換服務
 - » RFC 793
 - » 可靠的、按順序的、位元組串流資料傳輸
遺失：確認及重傳
 - » 流量控制：
傳送端的流量不超過接收端
 - » 壓塞控制：(congestion)
當網路壅塞時，傳送端會降低傳送速率



■ 網路的邊際 (Cont.)

► 終端系統間的資料傳輸服務 (Cont.)

○ 盡其可能的資料傳輸服務

– UDP – User Datagram Protocol (Chapter 3)

- » 網際網路的不需可靠資料傳輸服務
- » RFC 768
- » 非預接式
- » 不可靠的資料傳輸
- » 不提供流量控制
- » 不提供壅塞控制



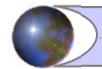
■ 網路的邊際 (Cont.)

► 終端系統間的資料傳輸服務 (Cont.)

- 使用 TCP 的應用程式：
 - HTTP (網頁)、FTP (檔案傳輸)、Telnet (遠端登入)、SMTP (電子郵件)
- 使用 UDP 的應用程式：
 - 串流媒體、遠端會議、DNS (網域名稱系統)、網路電話



Ex 1.4



■ 連線網路與實體媒介

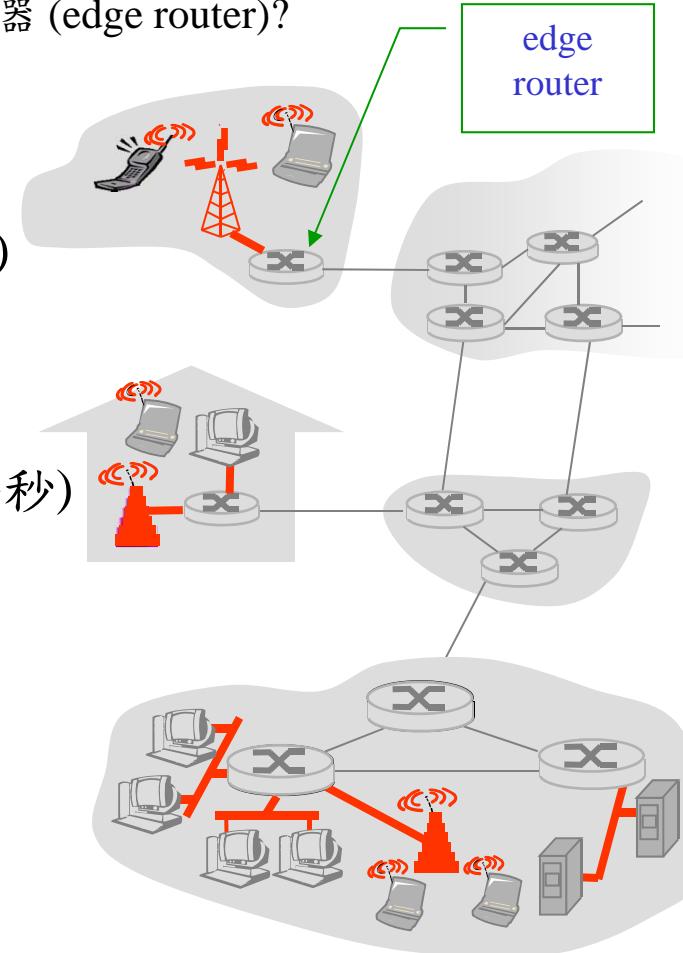
► What? 如何連結終端系統以及邊緣路由器 (edge router)?

► 類別

- 家用連線網路
- 機構的連線網路 (學校、公司)
- 無線連線網路

► 附註

- 連線網路的頻寬 (bps; 位元/每秒)
- 分享或專屬的





■ 連線網路

▶ 家用連線網路

- 家中的終端系統連接到邊際路由器
- 連線方式
 - 點對點的連線
 - 纜線數據機的連線

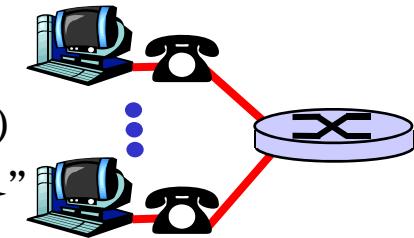


■ 連線網路 (Cont.)

▶ 家用連線網路 (Cont.)

○ 點對點的連線

- 經由數據機撥接 (modem)
 - » 至 56Kbps，直接存取路由器(通常較少)
 - » 無法與電話同時使用：不是“隨時服務”
- DSL: 數位用戶專線 (digital subscriber line)
 - » 建構來源：電話公司 (典型的)
 - » 至 1 Mbps 上傳速率 (目前通常 < 256 Kbps)
 - » 至 8 Mbps 下載速率 (目前通常 < 1 Mbps)
 - » 提供中央辦公室電話的實體線路
 - ADSL (非對稱式 asymmetric 數位用戶專線)
 - » 下載頻寬: 0.5~8 Mbps
 - » 上傳頻寬: ~ 1 Mbps
- VDSL (非常高速 DSL)



上傳與下載速率
不同



■ 連線網路 (Cont.)

▶ 家用連線網路 (Cont.)

○ 點對點的連線 (Cont.)

- DSL: 數位用戶專線 (digital subscriber line) (Cont.)

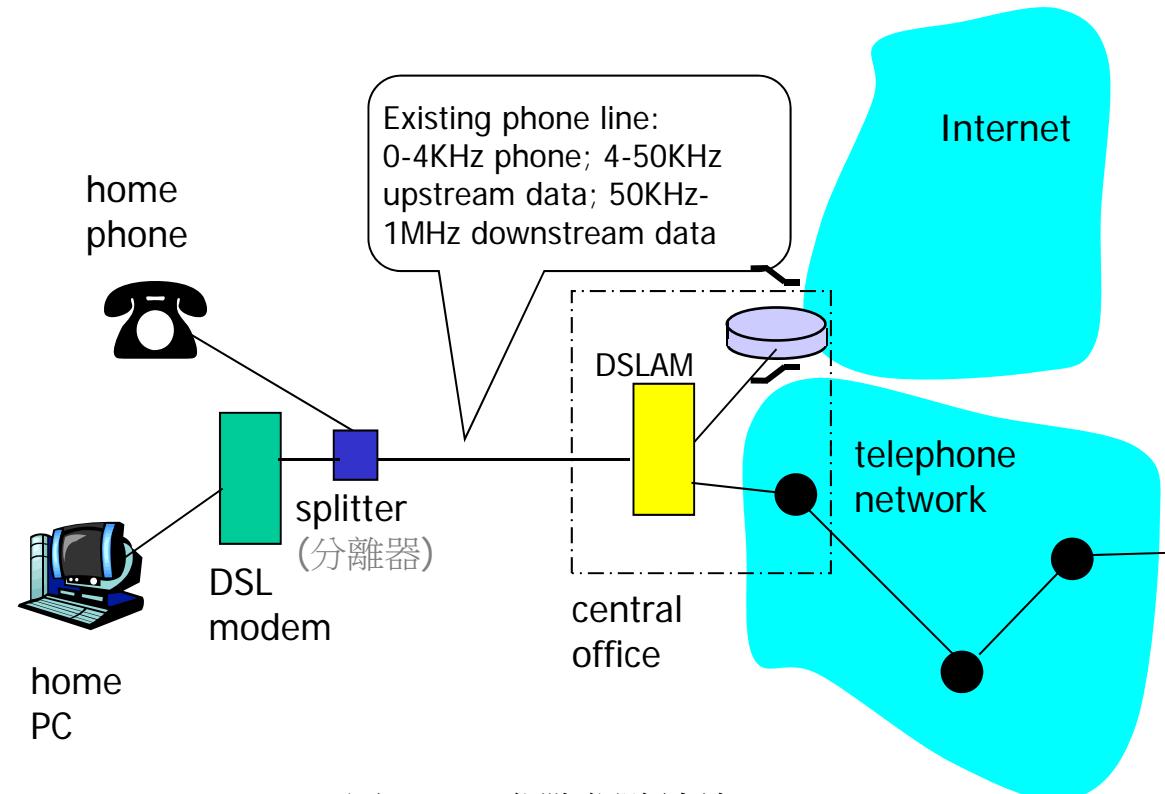


圖1.6 DSL網際網路連線



■ 連線網路 (Cont.)

▶ 家用連線網路 (Cont.)

○ 纜線數據機的連線

- 藉由電纜與光纖的網路將家庭用戶連接到 ISP (Internet Service Provider) 路由器
 - » 家庭用戶分享到路由器的存取線路
- 建構來源：有線電視公司
- HFC (Hybrid Fiber Coaxial Cable)：混合光纖/同軸電纜線
 - » 非對稱的：至 30 Mbps 下載速率，2 Mbps 上傳速率



■ 連線網路 (Cont.)

▶ 家用連線網路 (Cont.)

○ 纜線數據機的連線 (Cont.)

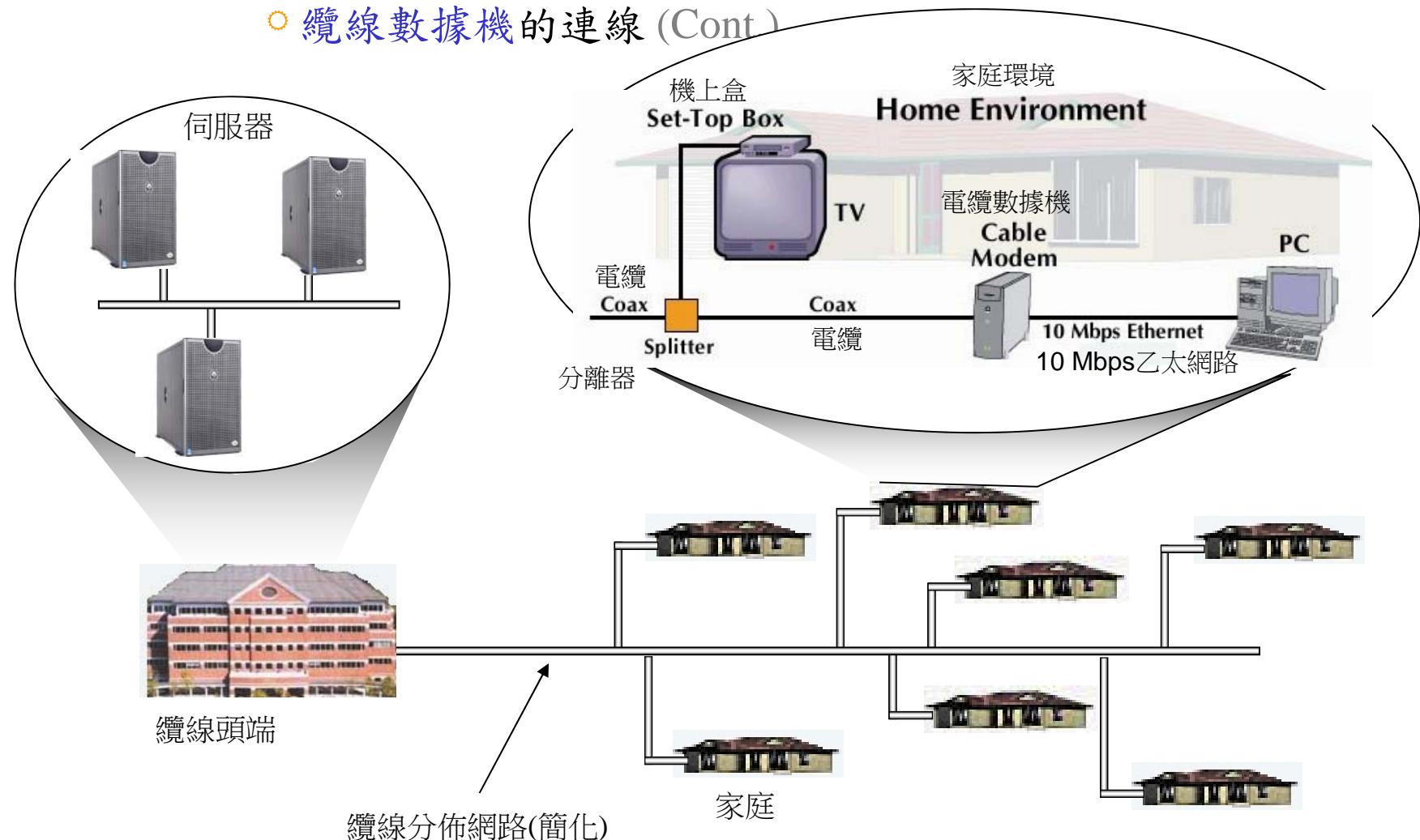


圖1.7 纜線數據機的連線



■ 連線網路 (Cont.)

▶ 家用連線網路 (Cont.)

○ 光纖到府(Fiber-To-The-Home)的連線

- 光纖速度遠快於雙絞線及同軸電纜
- 光纖直接從機房鋪設到住家
- 兩種彼此競爭的光學傳播網路架構
 - » Passive Optical Network (PON, 被動)
 - » Active Optical Network (AON, 主動)
- 遠快於網際網路速率，且也可提供電視及電話服務
- 目前提供的可能用戶速率
 - » 上傳: 2 ~ 10M bps
 - » 下載: 10 ~ 20M bps



■ 連線網路 (Cont.)

▶ 家用連線網路 (Cont.)

○ 光纖到府(Fiber-To-The-Home)的連線 (Cont.)

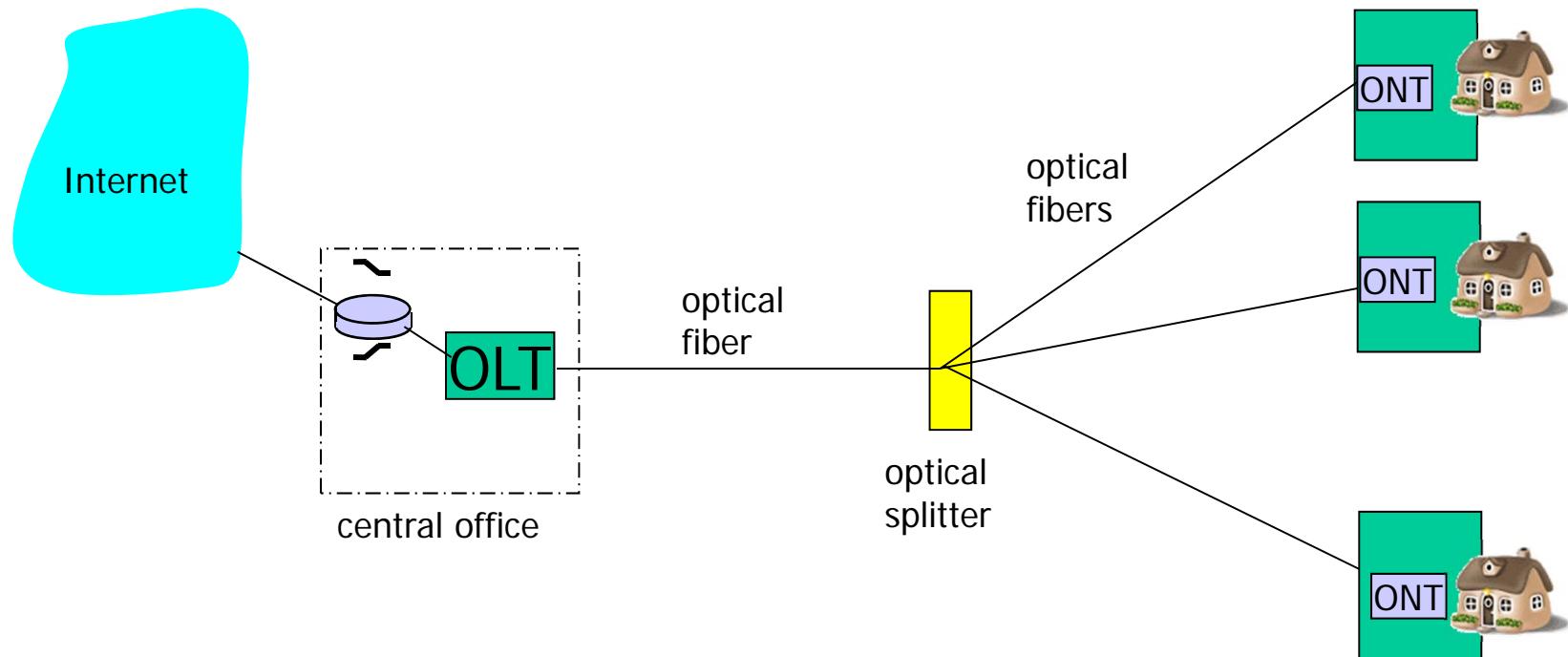


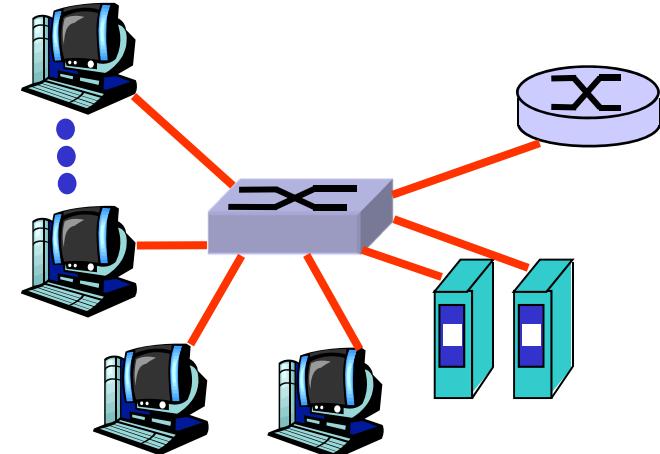
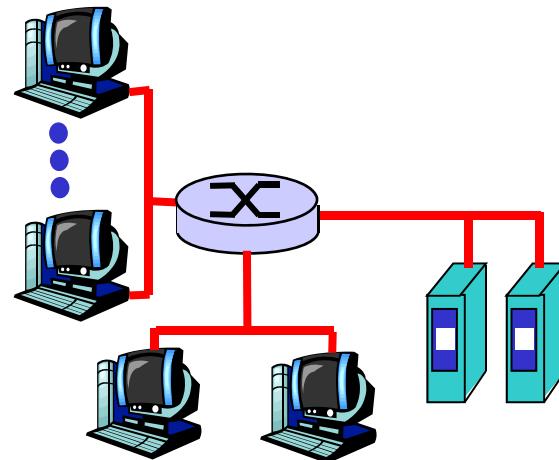
圖1.8 PON FTTH網際網路連線



■ 連線網路 (Cont.)

► 機構的連線網路

- 公司/學校採區域網路 (Local Area Network, LAN) 連結終端系統到邊緣路由器
- 最常見的區域網路
 - 乙太網路 (Ethernet)
 - 10 Mbps、100 Mbps、1 Gbps、10 Gbps 乙太網路
 - 目前的形式：終端系統連接到交換式乙太網路（星狀）
- LANs：第五章、第I章(本教材)





■ 連線網路 (Cont.)

► 機構的連線網路 (Cont.)

○ 典型的乙太網路連線

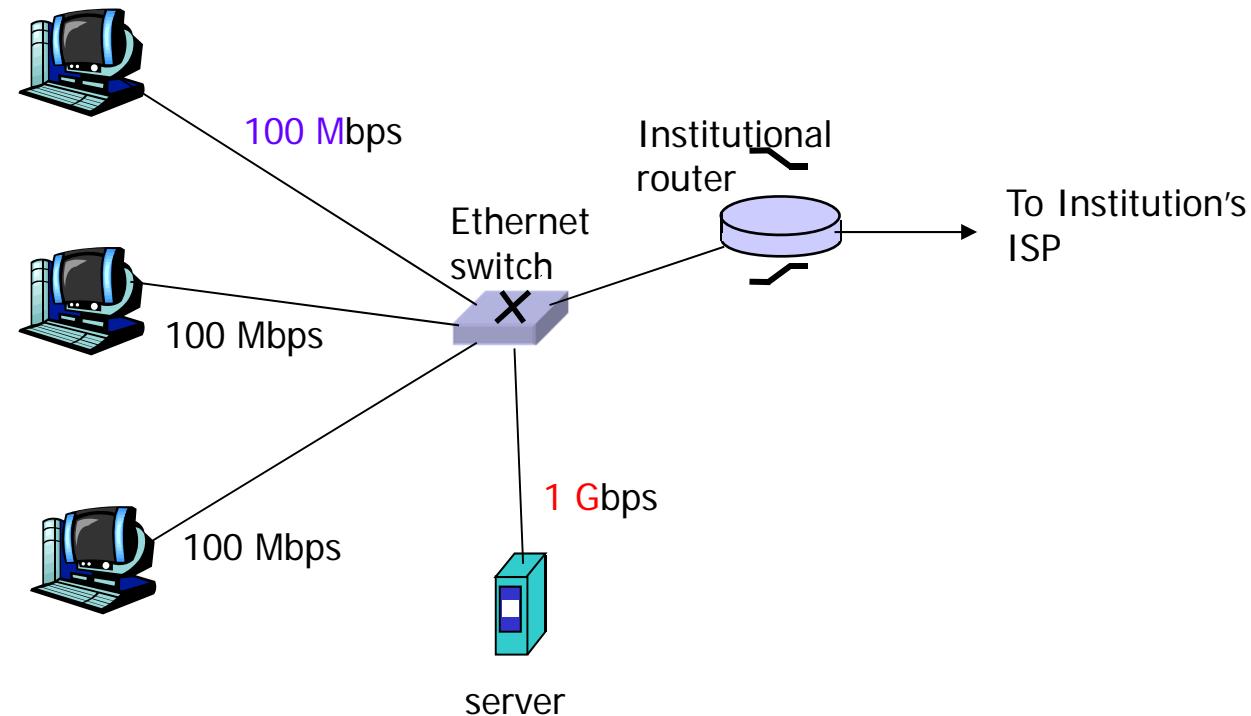


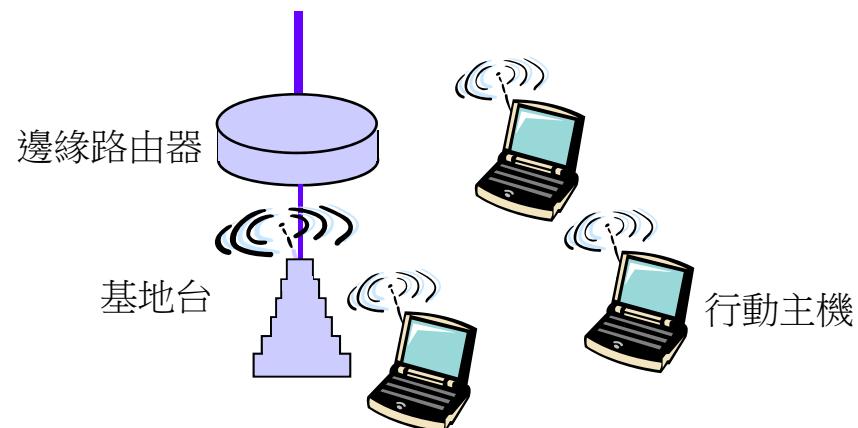
圖1.9 乙太網路網際網路連線



■ 連線網路 (Cont.)

▶ 無線連線網路 (Wireless)

- 分享的無線連線網路連接終端系統到路由器
 - 經由基地台又稱為「存取連結點」
 - Ad hoc
- 無線的LAN：
 - 802.11b/g (WiFi) : 11 or 54 Mbps
- 廣域無線連線網路
 - 由電信公司提供
 - ~1Mbps 透過手機系統 (EVDO, HSDPA)
 - 下一個會是 (?) : WiMAX (10's Mbps) 涵蓋廣域





■ 連線網路 (Cont.)

► 典型家庭網路元件：

- DSL或纜線數據機
- 路由器/防火牆/NAT
- 乙太網路
- 無線存取連結點 (基地台，一般的router均提供此無線連結點)

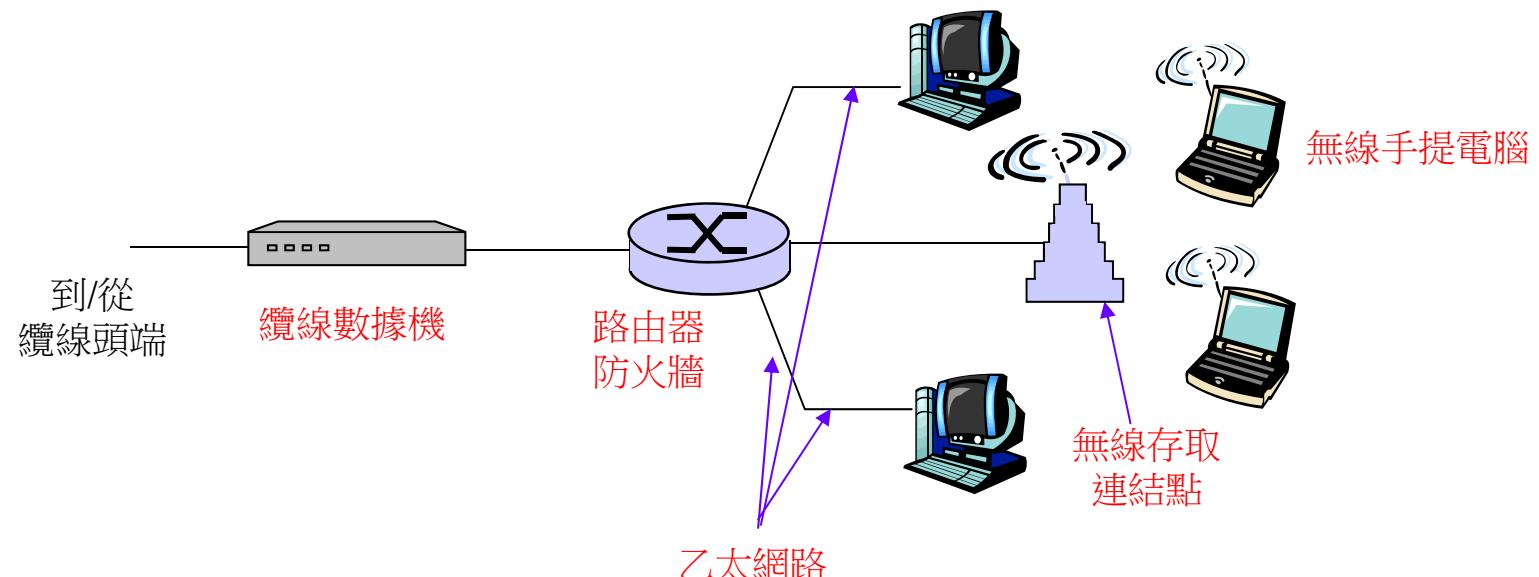


圖1.6 典型家庭網路的示意圖





■ 實體媒介 (Physical media)

▶ 基本術語

- 位元：在多組傳送器/接收器間傳輸
- 實體連結：位在傳送器以及接收器之間

▶ 實體媒介分類

- 引導式媒介(guided media) : (wired link)
 - 信號延著固體媒介傳輸：銅線、光纖、同軸纜線
- 非引導式媒介(unguided media) : (wireless)
 - 信號自由地傳輸，例如：無線電

▶ 詳細內容 (本節將簡述)

- 第I章 (本教材)



■ 實體媒介 (Cont.)

▶ 雙絞銅蕊線 (Twisted Pair)

- 兩條絕緣的銅線
- 第三類：傳統電話線、10 Mbps 乙太網路
- 第五類：100 Mbps 乙太網路
- 第七類：...





■ 實體媒介 (Cont.)

▶ 同軸電纜：(Cable)

- 兩條同軸的銅製導體
- 基頻：
 - 電纜上的單一頻道
 - 傳統的乙太網路
- 寬頻：
 - 電纜上的多頻道
 - HFC

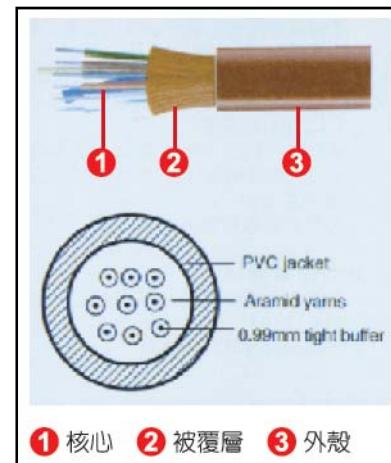




■ 實體媒介 (Cont.)

▶ 光纖：(Fiber Optic)

- 玻璃纖維，可傳導光的脈波，每個脈波代表一個位元
- 高速運轉：
 - 高速的點對點傳輸 (例如：10~100 Gbps)
- 低錯誤率：中繼器(repeater)置放距離遠，免於電磁波干擾

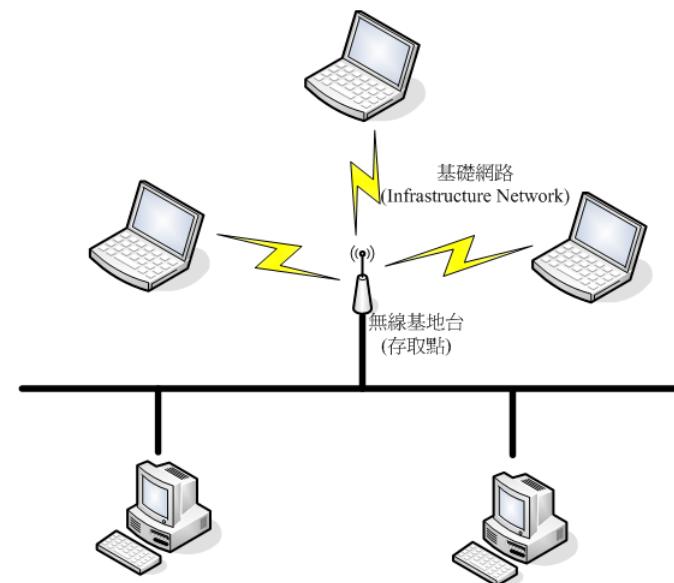




■ 實體媒介 (Cont.)

► 無線電：(Radio)

- 利用電磁頻譜傳送訊號
- 沒有實體的“線路”
- 雙向的
- 傳輸環境的影響：
 - 反射
 - 物件的障礙
 - 干擾



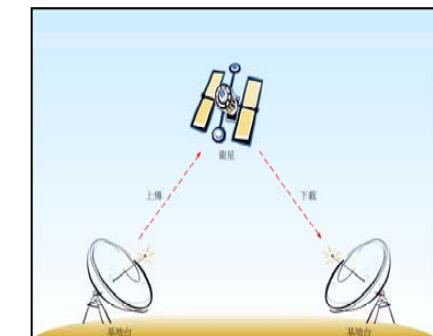


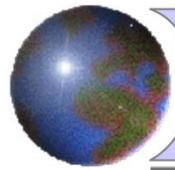
■ 實體媒介 (Cont.)

▶ 無線電：(Cont.)

○ 無線電連結型態：

- 地表微波
 - » 例如：至 45 Mbps 頻道
- LAN (例如：WiFi)
 - » 11 Mbps、54 Mbps
- 廣域 (例如：手機)
 - » 3G 手機：~1 Mbps
- 衛星
 - » Kbps 至 45 Mbps 頻道(或多個小頻道)
 - » 270 msec 端點至端點延遲
 - » 同步衛星 vs. 低軌道衛星





第一章 導覽流程



1.1 什麼是網際網路 (Internet) ?

1.2 網路的邊際 (Network Edge)

▶ 終端系統、連線網路、連結

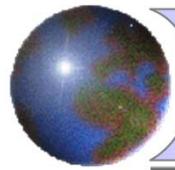
1.3 網路的核心 (Network Core)

▶ 電路交換、封包交換、網際網路結構

1.4 網路的效能 (Network Performance)

▶ 封包交換網路的延遲、封包遺失和產出率

1.5 協定分層及其服務模型 (Protocol Layers and Their Service Models)



1.3 網路的核心 (Network Core)

■ 網路的核心(中心, core)

- ▶ 網狀互相連結的路由器
- ▶ 基本的問題：
資料以何種方式經由網路傳輸？
 - 電路交換(circuit switch)：
每通電話均有專用電路：電話網路
 - 封包交換(packet switch)：
以不連接的「小段資料」經由網路傳送資料

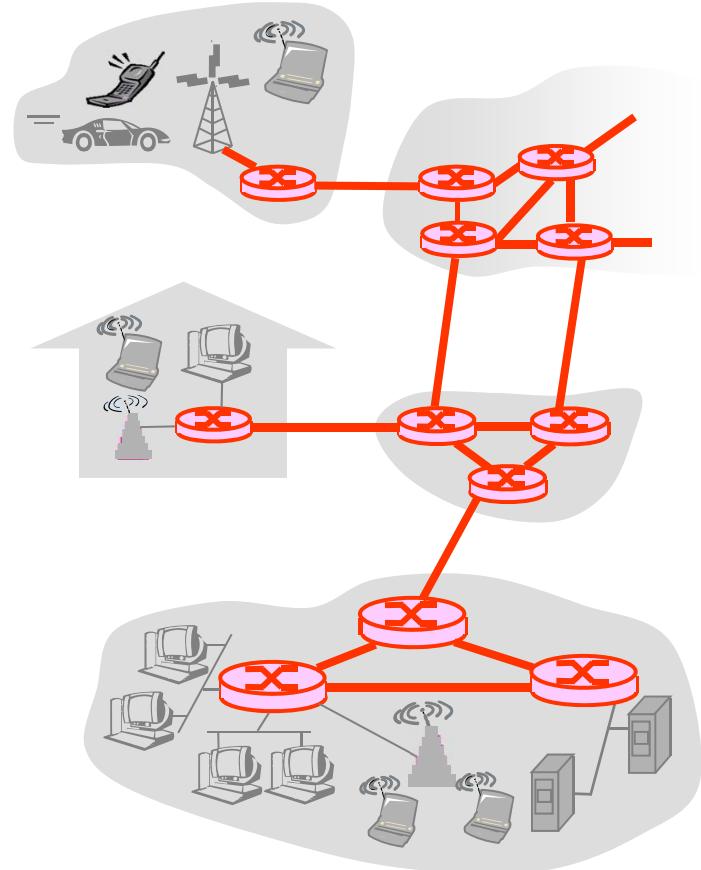
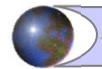


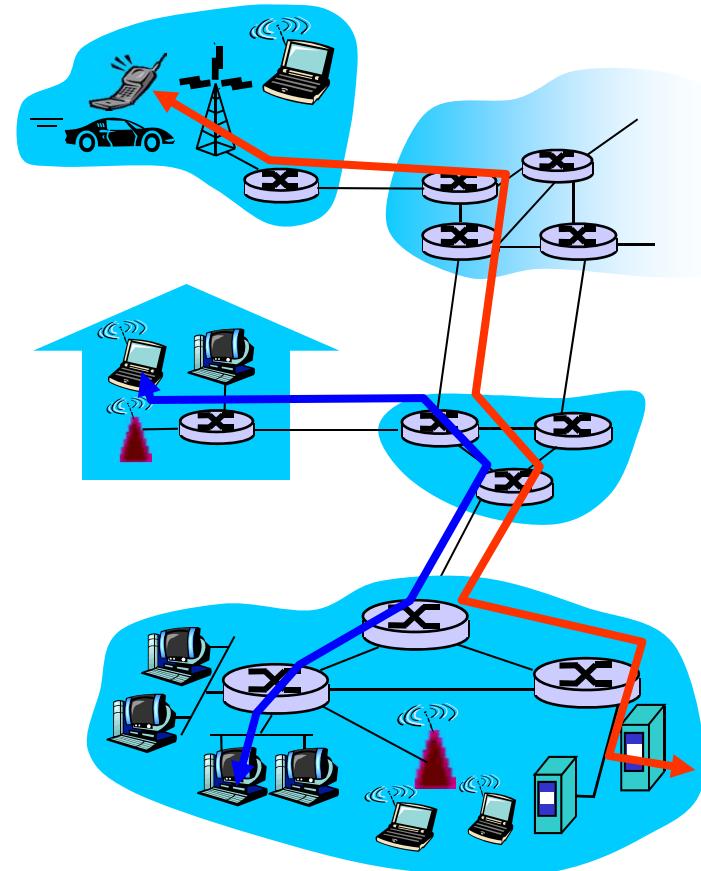
圖1.11 網路核心(中心)



■ 電路交換 (circuit switch)

► 意義：

- 為通訊保留端點至端點的資源
 - 緩衝區、連線傳輸速率等
- 建立實體連線，並保證連結頻寬，交換容量
- 專屬的資源：不需分享
- 如同電路的（保證的）效能
- 需要建立連線
 - 點對點連線





■ 電路交換 (Cont.)

► 特性：

- 網路資源(例如：頻寬)被切分為片段(frame)
- 片段會被分派給通話(通訊)
- 假如擁有此資源的通話沒有在使用，則資源片段會閒置
(非分享的)

► 將連結頻寬(bandwidth)切分為「片段」的技術

- 分頻多工 (Frequency-Division Multiplexing, FDM)
- 分時多工 (Time-Division Multiplexing, TDM)



■ 電路交換 (Cont.)

► 將連結頻寬切分為「片段」的技術 (Cont.)

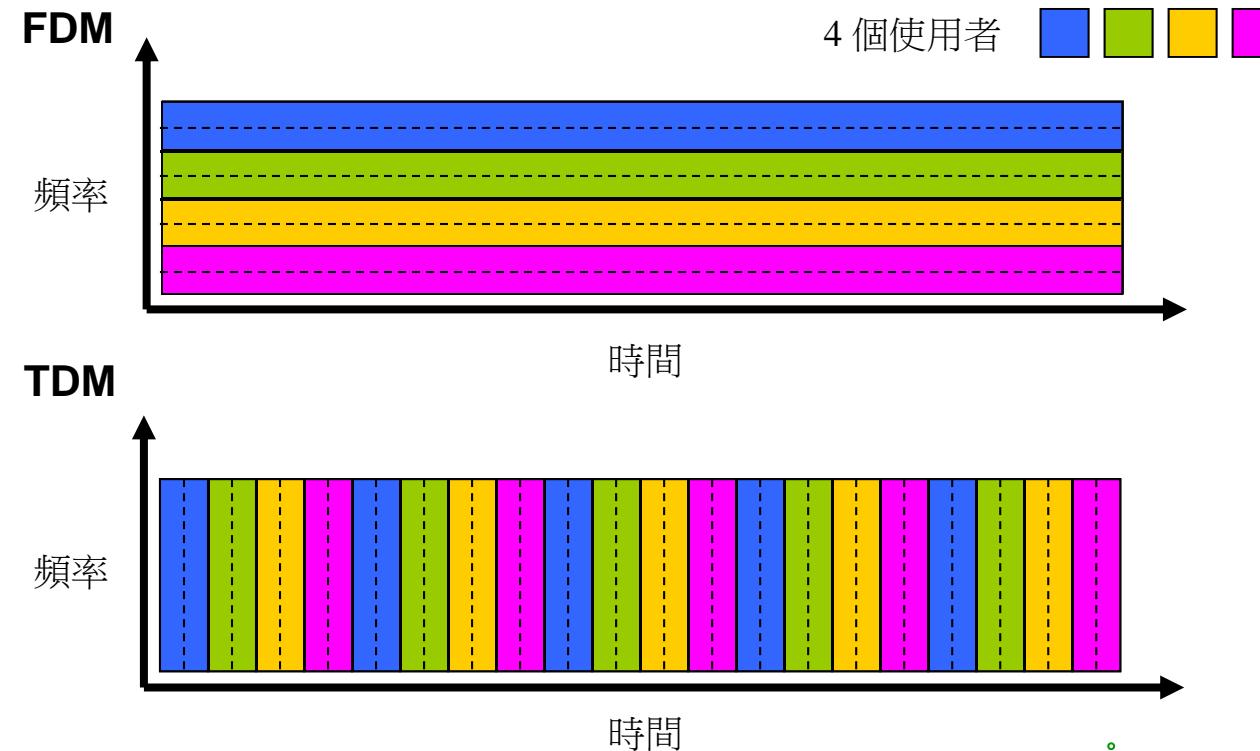


圖1.13 分頻與分時技術

FM廣播採用何種?

■ 電路交換 (Cont.)

► 範例

- 在電路交換網路中，從主機A傳送640,000位元的檔案到主機B時，需要多少的時間?
 - 所有的連結都為 1.536 Mbps
 - 所有連結都使用具有24個時槽的 TDM
 - 需要500毫秒(1毫秒 = 10^{-3} 秒)的時間來建立一條端點到端點的線路

Ans.

- ① 每道線路的傳輸速率 = $1.536 \text{ Mbps} / 24 = 64 \text{ Kbps}$
- ② 需要 $(640,000 \text{ bits}) / (64 \text{ Kbps}) = 10 \text{ sec}$ 來傳送該檔案
- ③ 總共時間 = 傳送時間 + 建立通道時間 = $10 \text{ sec} + 0.5 \text{ sec} = 10.5 \text{ sec}$





■ 封包交換 (packet switch)

► 意義：

- 每一個端點至端點的資料串流都會被切分為封包 (packet)
- 使用者 A、B 的封包分享網路資源
- 每個封包使用完整的連結頻寬
- 資源在需要時被使用

頻寬被切分為
片段專屬的分
派資源保留

► 資源的競爭

- 要求的資源總量可能會超過可用的
- 壓塞(congestion)：封包佇列，等待可用的連結
- 儲存並轉送(store & forward)：封包一次移動一站
- 節點在傳送之前會接收完整的封包



■ 封包交換 (Cont.)

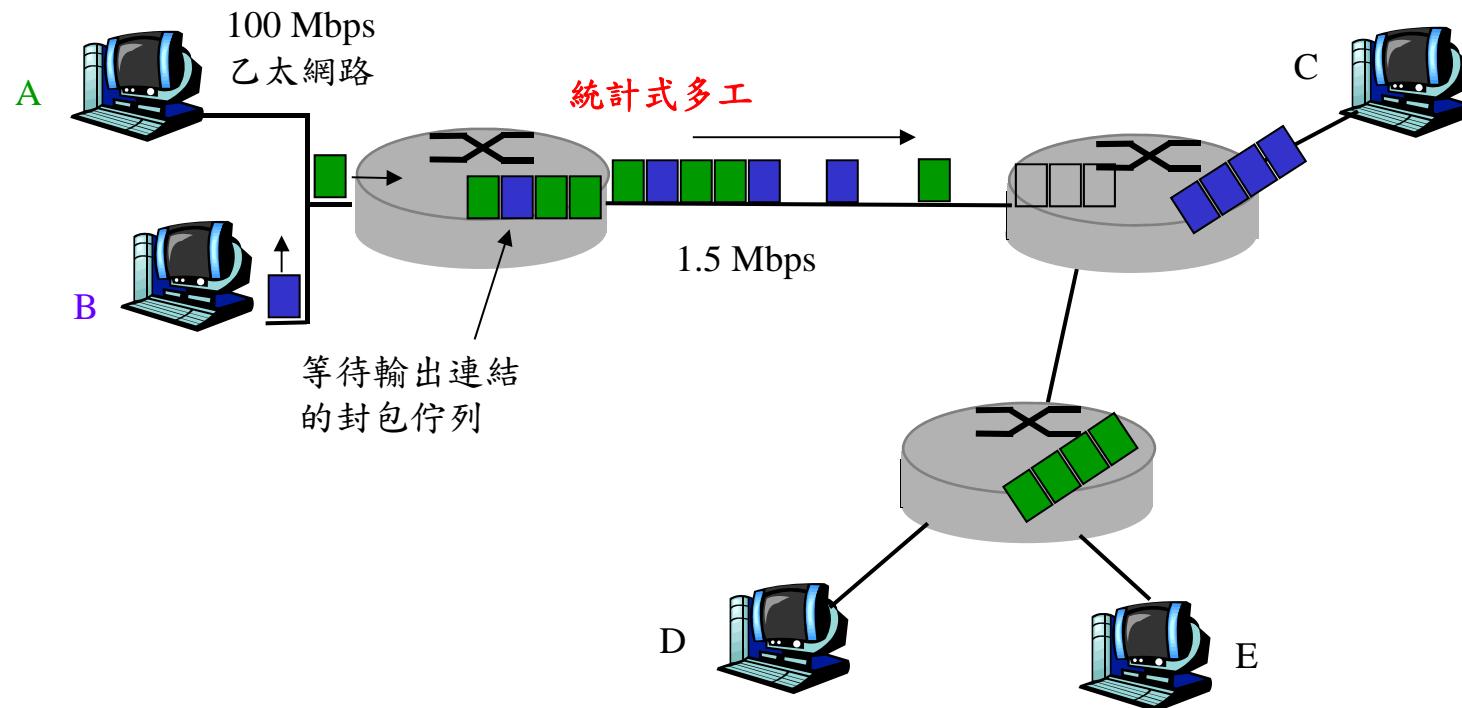


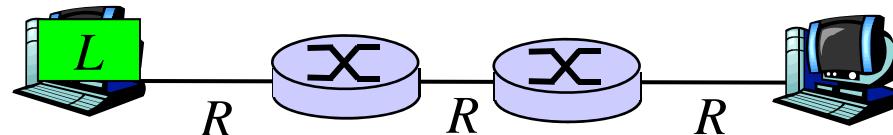
圖 1.14 封包交換

- A 與 B 的封包次序並沒有固定的模式，按需求分配的頻寬共享方式 \Rightarrow 統計式多工
- TDM：在循環TDM時框時，每一個主機會得到同一個時槽



■ 封包交換 (Cont.)

► 儲存並轉送 (store and forward)



- 在 R bps 的連結上，需要 L/R 秒來傳遞(推出)長度為 L 位元的封包
- 儲存並轉送：
 - 在封包傳送到下一個連結之前，它必須完全到達路由器
- 延遲 = $3L/R$ (假設沒有傳遞延遲)
- 例題：
 - $L = 7.5$ Mbits
 - $R = 1.5$ Mbps
 - 傳輸延遲 = 15 sec



■ 封包交換 vs. 電路交換

► 封包交換允許較多的使用者使用網路

► 範例

- 1 Mbps 的連結

- 每一個使用者：

- 活動時的速率 100 Kbps

- 只有 10% 的時間在活動

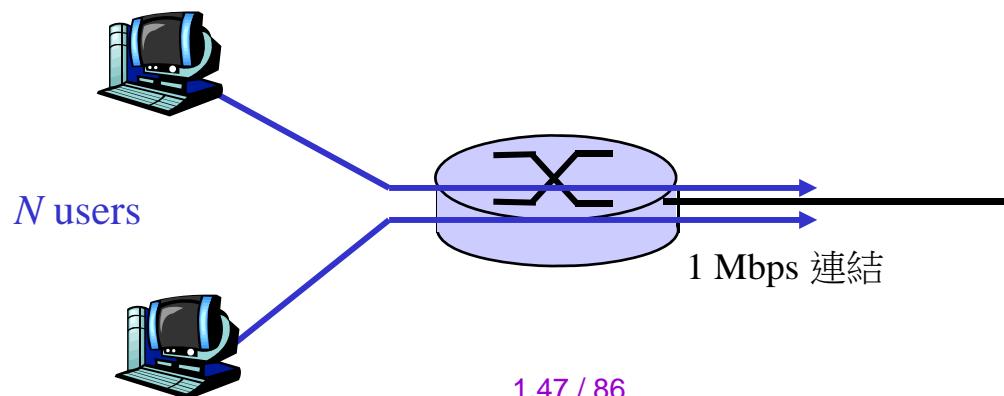
⇒ 電路交換：

- 10 個使用者

⇒ 封包交換：

- 35 個使用者，有大於 10 人同時活動的機率小於 0.0004

習題
P6





■ 封包交換 vs. 電路交換 (Cont.)

▶ 封包交換的問題

- 極度壅塞時
 - 封包延遲及遺失
 - 需要可靠資料傳輸以及壅塞控制的協定

▶ 封包交換 vs. 電路交換

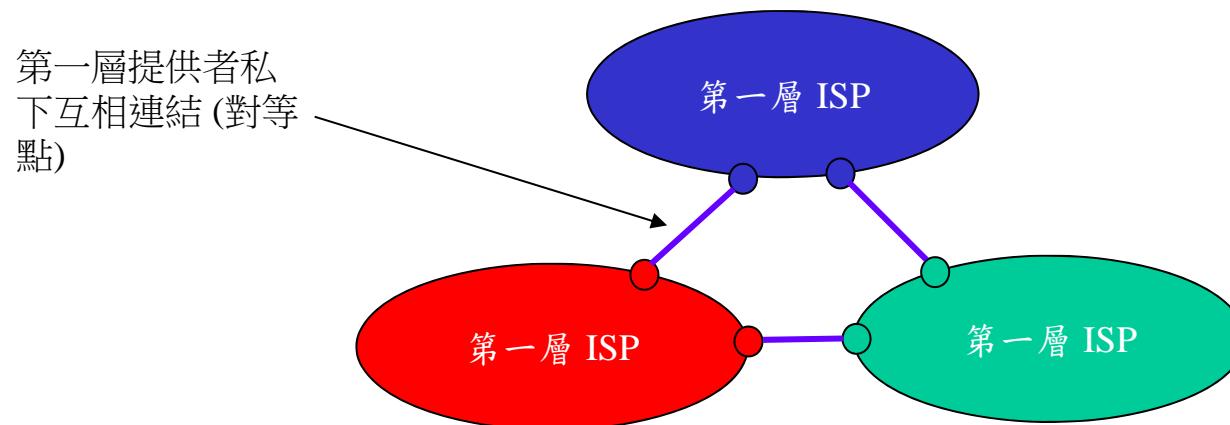
- 依需要的分派(封包交換) vs. 保留的資源(電路交換)

▶ 網際網路 – 封包交換 \Leftrightarrow 電話系統 – 電路交換



■ 網路們的網路

- ▶ 粗略的階層性
- ▶ 在中央：“第一層” ISP (例如：MCI、Sprint、AT&T、Cable and Wireless)，涵蓋本國/國際範圍
- ▶ 平等地彼此對待

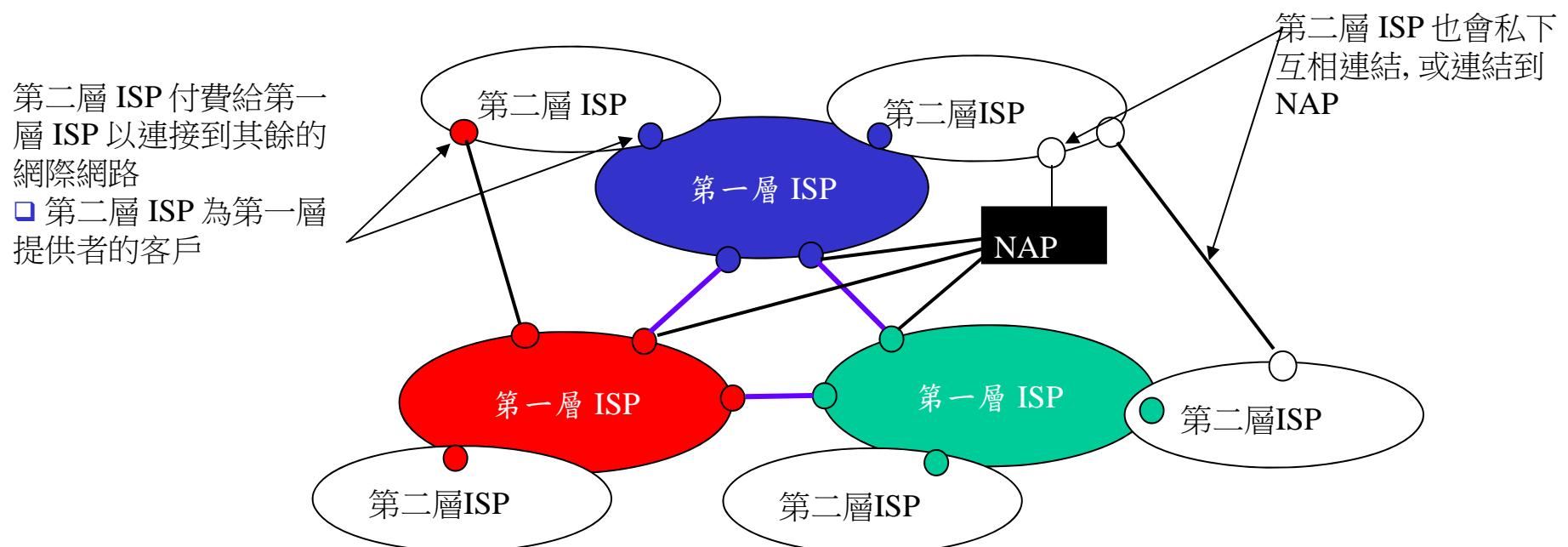




■ 網路們的網路 (Cont.)

► “第二層” ISP：較小的（通常是區域性的）ISP

- 連接到一或多個第一層 ISP，也可能是其它的第二層 ISP





■ 網路們的網路 (Cont.)

► “第三層” ISP 以及 本地 ISP

- 最後一站（“存取”）網路（最接近終端系統）

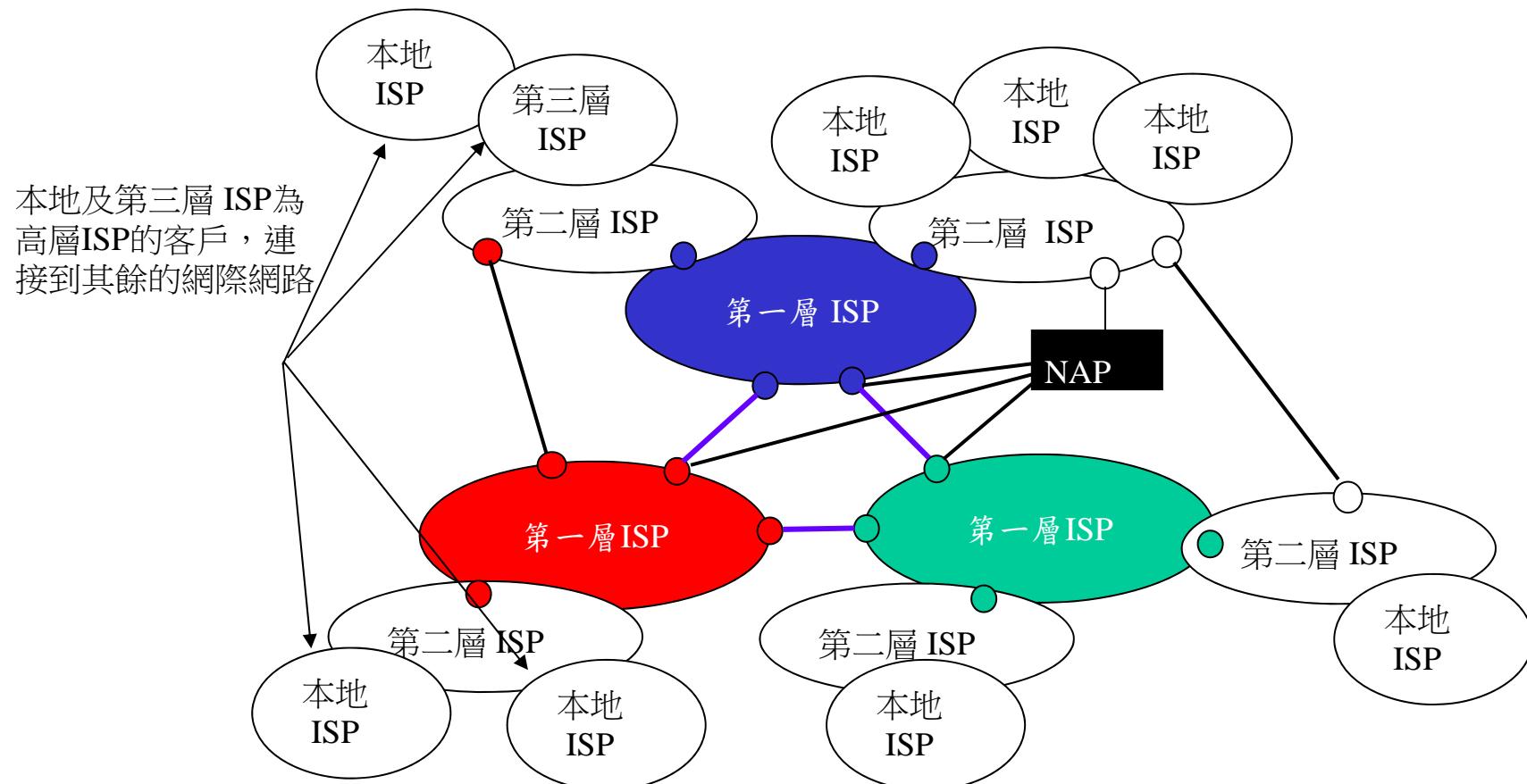
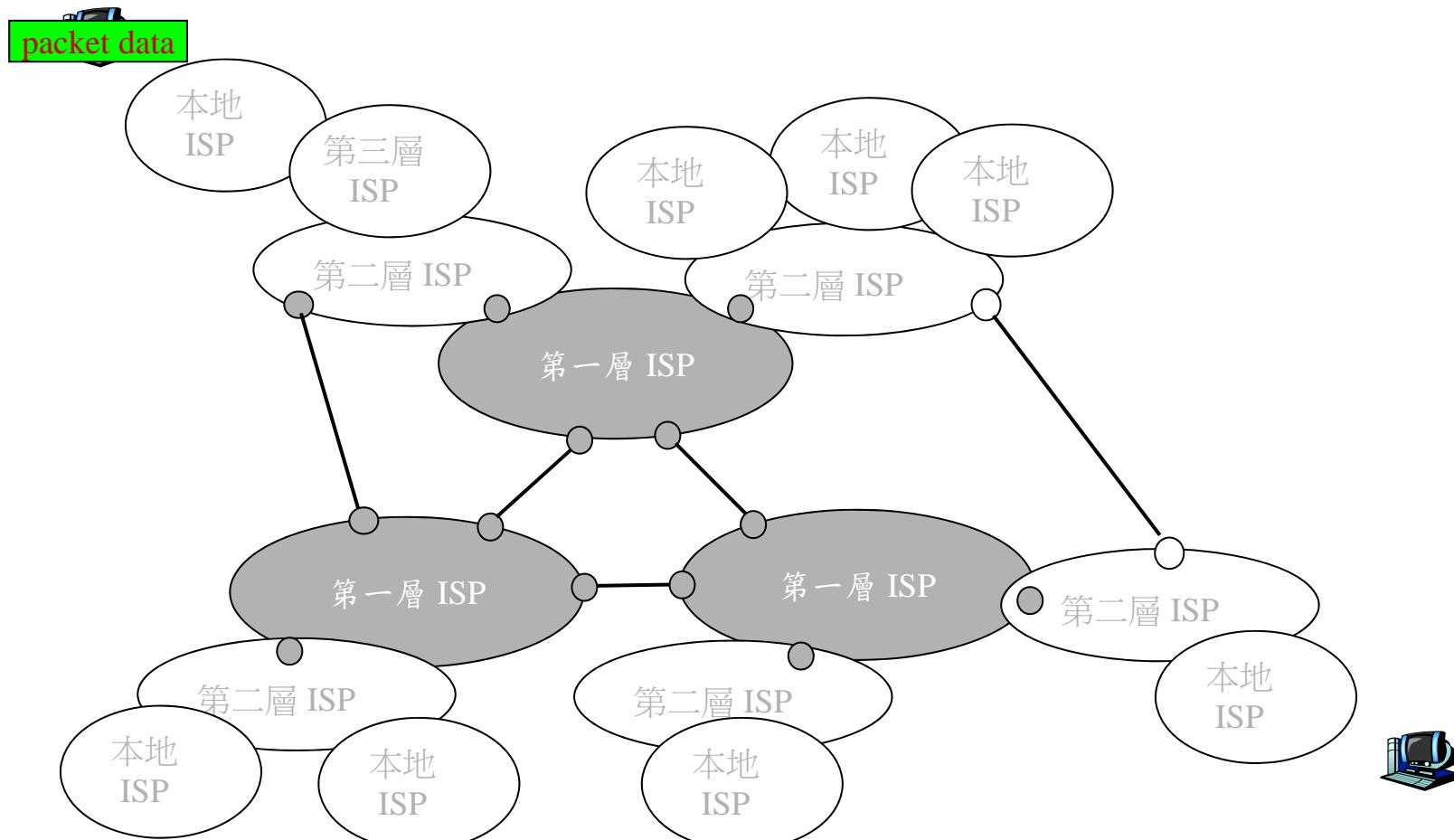


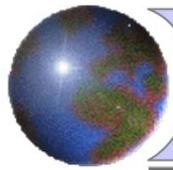
圖 1.15 ISP 的相互連接



■ 網路們的網路 (Cont.)

► 一個封包傳送經過許多網路





第一章 導覽流程

1.1 什麼是網際網路 (Internet) ?

1.2 網路的邊際 (Network Edge)

▶ 終端系統、連線網路、連結

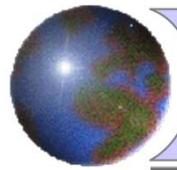
1.3 網路的核心 (Network Core)

▶ 電路交換、封包交換、網際網路結構

1.4 網路的效能 (Network Performance)

▶ 封包交換網路的延遲、封包遺失和產出率

1.5 協定分層及其服務模型 (Protocol Layers and Their Service Models)



1.4 網路的效能 (Network Performance)



■ 封包交換網路的效能

- ▶ 延遲時間 (delay time)
- ▶ 封包遺失 (packet loss)
- ▶ 產出率 (throughput)



■ 封包延遲與遺失如何發生

► 路由器緩衝區內的封包佇列

- 封包抵達連結的速率 > 輸出連結的容量
- 封包在佇列中等待下一輪

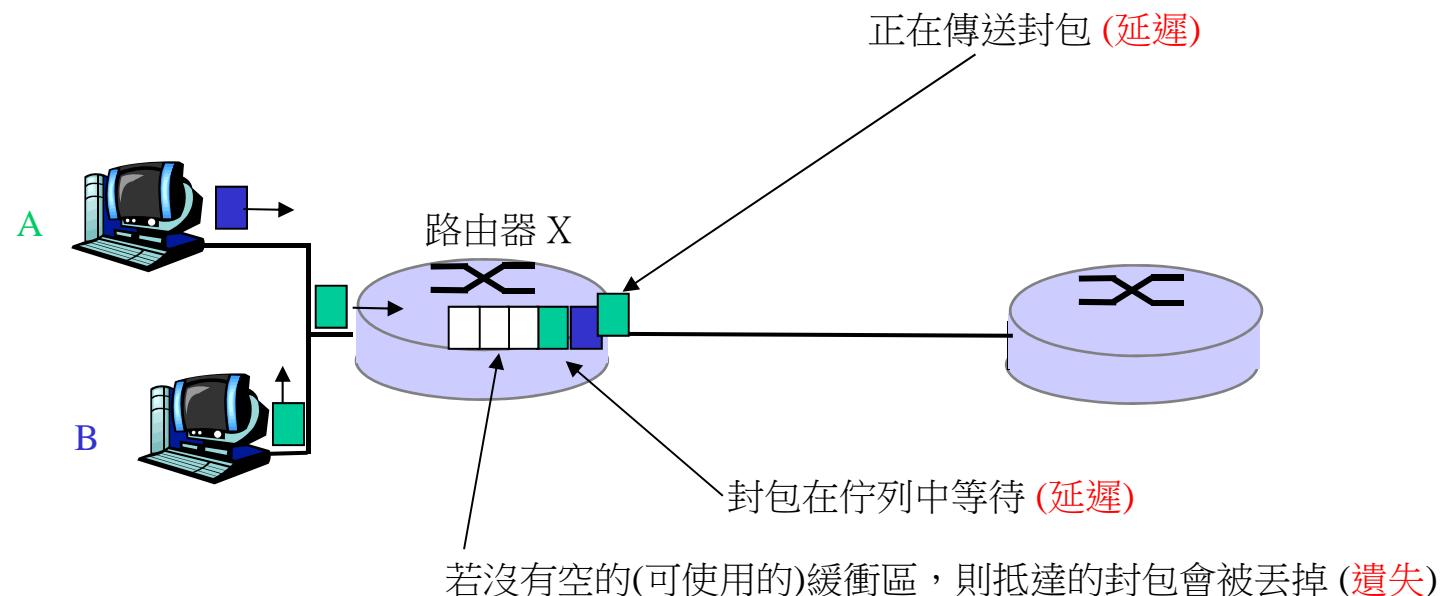


圖1.16 路由器X的節點延遲與封包遺失



■ 封包延遲的四種來源

▶ 節點處理 (processing delay)

- 檢查位元錯誤
- 決定輸出連結

▶ 節點佇列 (queuing delay)

- 等待傳送到輸出連結的時間
- 由路由器的壅塞等級決定

▶ 傳輸延遲 (transmission delay)

- R = 連結頻寬 (bps) 。 ○ ○
- L = 封包長度 (位元)
- 傳送位元的連結的時間 = L/R
- 將全部封包位元全部推入(傳送到)連結線所需的時間

eg. 10 Mbps 乙
太網路的傳輸
速率 $R=10$
 Mbits/sec

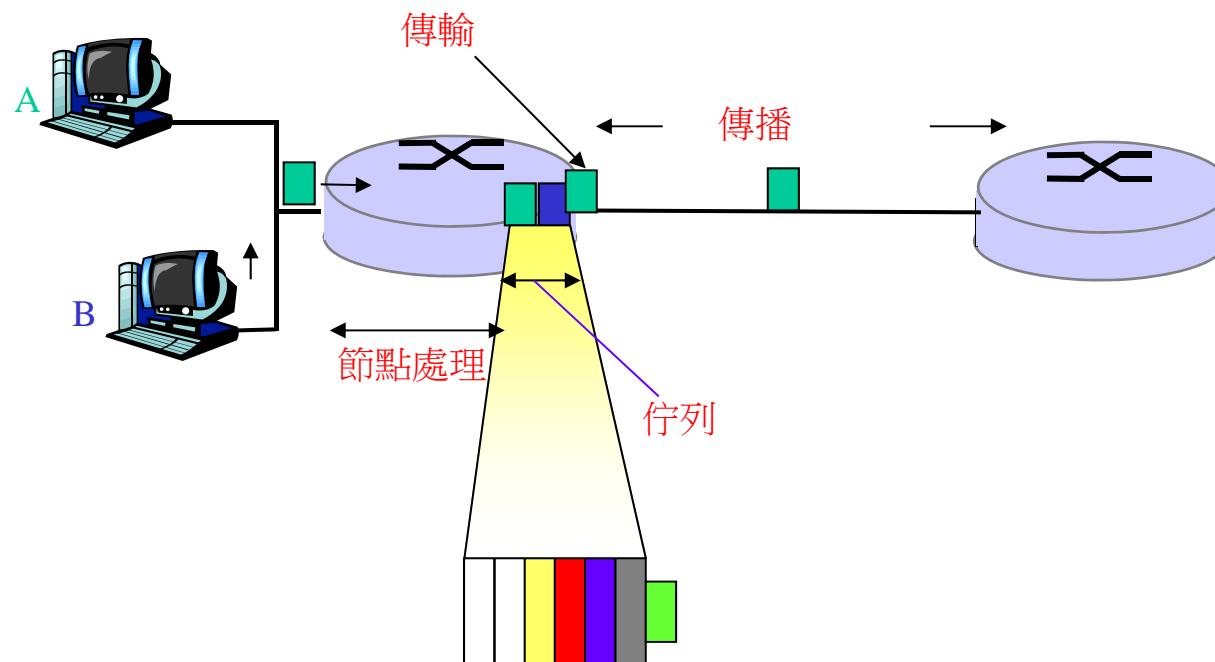
▶ 傳播延遲 (propagation delay)

- d = 實體媒介的長度
- s = 媒介的傳遞速率 ($2 \cdot 10^8 \sim 3 \cdot 10^8 \text{ m/sec}$)
- 傳播延遲 = d/s

⇒ 注意： s 和 R 是非常不同的量！



■ 封包延遲的四種來源 (Cont.)





■ 封包延遲的四種來源 (Cont.)

▶ 車隊的比喻

- 車輛以 100 km/hr 前進
- 收費站以 12 秒服務一台車 (傳送時間)
- 車~位元；車隊~封包
- Q: 車隊在第二個收費站前排好需要多少時間？

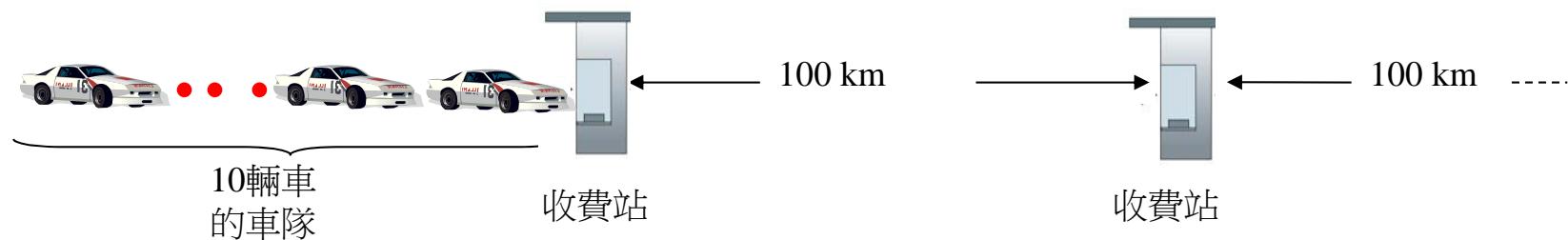


圖1.17 封包延遲 – 以車隊通過收費站為比喻



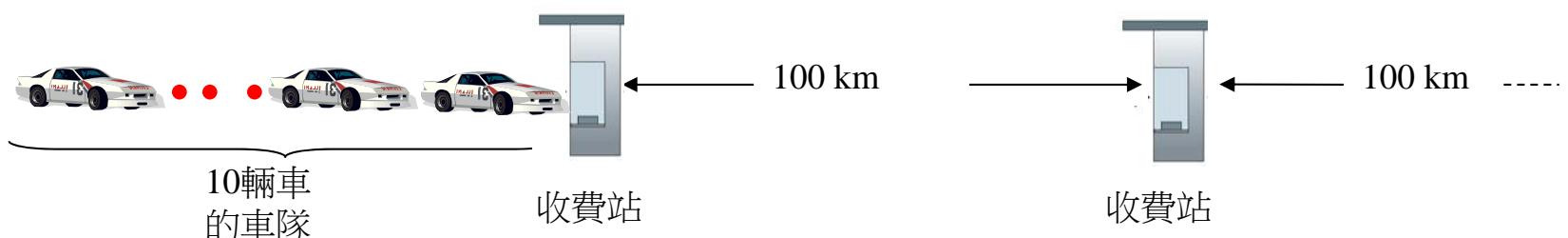
■ 封包延遲的四種來源 (Cont.)

▶ 車隊的比喻 (Cont.)

○ Q: 車隊在第二個收費站前排好需要多少時間？

○ Ans:

- 整個車隊 “通過” 收費站所需的時間 = $12 * 10 = 120 \text{ sec} = 2 \text{ min}$ (佇列時間+處理時間+傳輸時間)
- 最後一台車從第一個收費站到第二個收費站所需的時間 = $100 \text{ km} / (100\text{km/hr}) = 1 \text{ hr} = 60 \text{ min}$ (傳播時間)
- Ans: 62分鐘

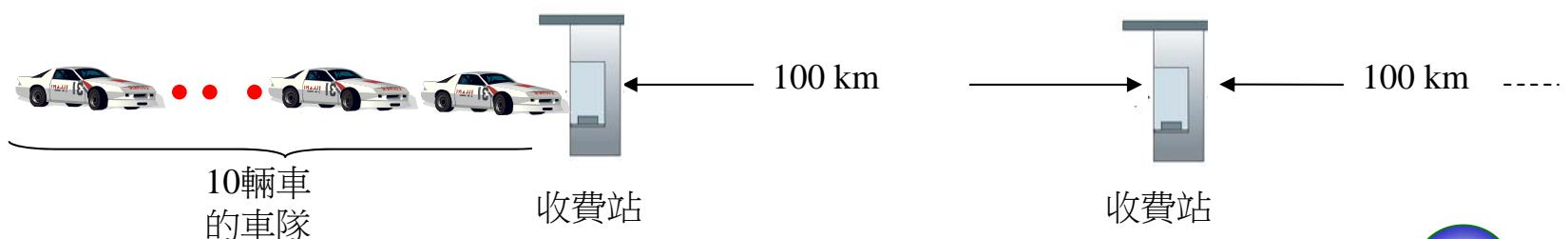




■ 封包延遲的四種來源 (Cont.)

▶ 車隊的比喻 (Cont.)

- 車輛現在以 1000 km/hr 的速率 “傳播(遞)”
收費站以 1 分鐘服務 1 台車
- Q: 在所有的車輛被第一個收費站服務完畢之前，是否有車輛會抵達第二個收費站嗎？
- Ans:
 - 是的！在 7 分鐘後，第 1 台車會到達第二個收費站，此時仍有三台車在第一個收費站
 - 在封包從第一個路由器完全傳送出去之前，它的第 1 個位元可能已抵達第二個路由器！





■ 封包延遲的四種來源 (Cont.)

► 節點延遲 d_{nodal}

$$\circ d_{\text{nodal}} = d_{\text{proc}} + d_{\text{queue}} + d_{\text{trans}} + d_{\text{prop}}$$

– d_{proc} = 處理延遲

» 通常只有幾百萬分之一秒或更少

– d_{queue} = 個列延遲

» 取決於壅塞程度

– d_{trans} = 傳輸延遲

» $= L/R$ ，對低速連結來說，其值會很大

– d_{prop} = 傳播延遲

» 從幾百萬分之一秒到幾個毫秒

⇒ 傳輸延遲 vs. 傳播延遲

http://media.pearsoncmg.com/aw/aw_kurose_network_2/applets/transmission/delay.html

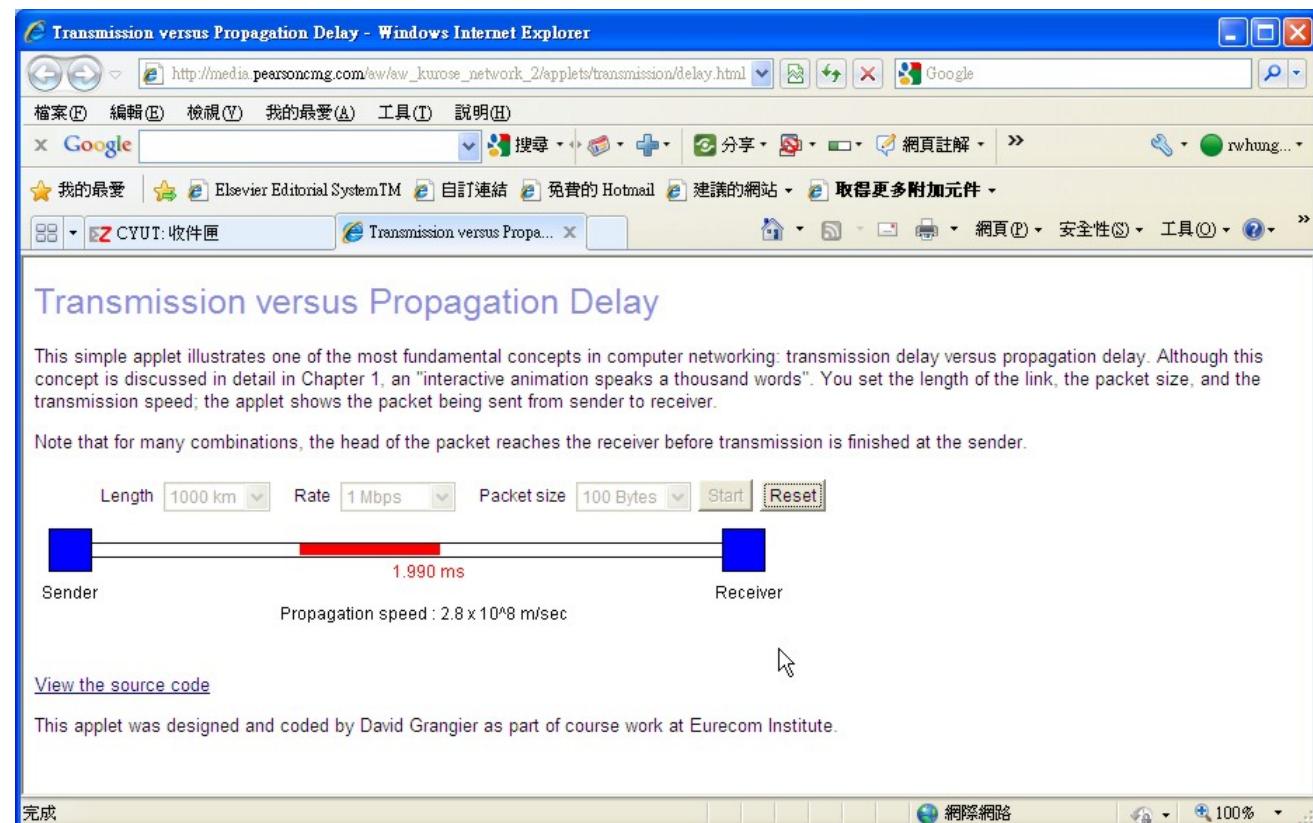


■ 封包延遲的四種來源 (Cont.)

► 節點延遲 d_{nodal} (Cont.)

⇒ 傳輸延遲 vs. 傳播延遲

http://media.pearsoncmg.com/aw/aw_kurose_network_2/applets/transmission/delay.html





■ 封包延遲的四種來源 (Cont.)

▶ 節點延遲 d_{nodal} (Cont.)

⇒ 傳輸延遲 vs. 傳播延遲 (Conti.)

- ① 傳輸延遲：與 routers 間的距離無關，而與封包長度和連結的傳輸速率有關
- ② 傳播延遲：與 routers 間的距離有關，而與封包長度和連結的傳輸速率無關



■ 封包延遲的四種來源 (Cont.)

► 候列延遲 d_{queue}

○ $d_{\text{queue}} = \text{候列延遲}$

– 取決於壅塞程度

○ 相關參數

– R = 連結頻寬 (bps)

– L = 平均封包長度 (位元/封包)

– a = 平均封包抵達率 (抵達封包數/sec)

○ 流量強度 (traffic intensity) = La/R

– $La/R \sim 0$: 平均候列延遲很小

– $La/R \rightarrow 1$: 延遲會變大

– $La/R > 1$: 抵達的“工作”超過可被服務的，平均延遲為無限大！

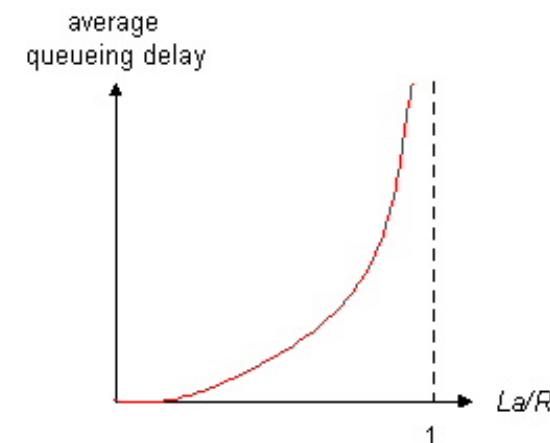
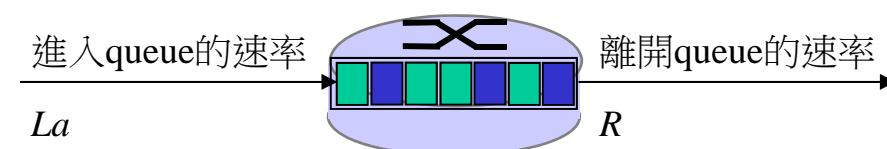


圖1.18 平均候列延遲與流量強度的關係

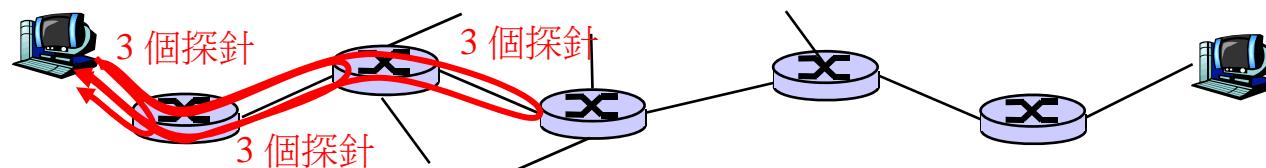




■ “真實的” 網際網路延遲與路徑

► *Traceroute* 程式

- 提供延遲測量，從來源端到路由器，沿著端點至端點的網際網路路徑到達目的端
- 原理：對到達目的端路徑上的所有路由器 i
 - 傳送三個探針封包，它們會抵達路由器 i ，位在通往目的端的路徑上
 - 路由器 i 會回傳封包給傳送端
 - 傳送端會紀錄傳送及回傳的時間間隔





■ “真實的” 網際網路延遲與路徑 (Cont.)

► Traceroute 程式

- traceroute: gaia.cs.umass.edu 至 www.eurecom.fr

從gaia.cs.umass.edu到cs-gw.cs.umass.edu，三次的延遲測量

```
1 cs-gw (128.119.240.254) 1 ms 1 ms 2 ms
2 border1-rt-fa5-1-0.gw.umass.edu (128.119.3.145) 1 ms 1 ms 2 ms
3 cht-vbns.gw.umass.edu (128.119.3.130) 6 ms 5 ms 5 ms
4 jn1-at1-0-0-19.wor.vbns.net (204.147.132.129) 16 ms 11 ms 13 ms
5 jn1-so7-0-0-0.wae.vbns.net (204.147.136.136) 21 ms 18 ms 18 ms
6 abilene-vbns.abilene.ucaid.edu (198.32.11.9) 22 ms 18 ms 22 ms
7 nycm-wash.abilene.ucaid.edu (198.32.8.46) 22 ms 22 ms 22 ms
8 62.40.103.253 (62.40.103.253) 104 ms 109 ms 106 ms
9 de2-1.de1.de.geant.net (62.40.96.129) 109 ms 102 ms 104 ms
10 de.fr1.fr.geant.net (62.40.96.50) 113 ms 121 ms 114 ms
11 renater-gw.fr1.fr.geant.net (62.40.103.54) 112 ms 114 ms 112 ms
12 nio-n2.cssi.renater.fr (193.51.206.13) 111 ms 114 ms 116 ms
13 nice.cssi.renater.fr (195.220.98.102) 123 ms 125 ms 124 ms
14 r3t2-nice.cssi.renater.fr (195.220.98.110) 126 ms 126 ms 124 ms
15 eurecom-valbonne.r3t2.ft.net (193.48.50.54) 135 ms 128 ms 133 ms
16 194.214.211.25 (194.214.211.25) 126 ms 128 ms 126 ms
17 * * *
18 * * * }* 意指沒有回應(探針遺失，路由器沒有回應)
19 fantasia.eurecom.fr (193.55.113.142) 132 ms 128 ms 136 ms
```



■ “真實的” 網際網路延遲與路徑 (Cont.)

► *Traceroute* 程式 (Conti.)

- www.traceroute.org ⇒ Taiwan ⇒ HiNet (source)

The screenshot shows a Windows Internet Explorer window titled "HiNet Traceroute - Windows Internet Explorer". The address bar contains the URL <http://traceroute.hinet.net/cgi-bin/tracert.pl>. The main content area displays the "HiNet Traceroute" page. At the top, there is a search bar with the placeholder "請輸入 IP 或 Hostname" and a text input field containing "www.cyut.edu.tw", followed by two buttons: "送出" (Send) and "清除" (Clear). Below this, a large bold text "107,822 Visitors" is displayed, with the subtitle "Since August 27, 2006". A horizontal line separates this from the "Traceroute Result:" section. In the "Traceroute Result:" section, the text "Type escape sequence to abort." is shown, followed by the command "Tracing the route to www.cyut.edu.tw (163.17.1.15)". Below this, a list of 10 traceroute hops is displayed:

```
1 TPDB-3516.hinet.net (210.65.161.22) 0 msec 0 msec 0 msec
2 TPDT-3012.hinet.net (220.128.2.146) 4 msec 0 msec 4 msec
3 TCHN-3112.hinet.net (220.128.2.9) 4 msec 4 msec 4 msec
4 TCHN-4901.hinet.net (220.128.17.137) 4 msec 4 msec 4 msec
5 TCHN-4935.hinet.net (211.22.189.229) 4 msec 4 msec 4 msec
6 ch-c12r1.router.hinet.net (211.22.189.221) 4 msec 4 msec 4 msec
7 140.128.251.13 4 msec 4 msec 4 msec
8 163.17.7.190 4 msec 4 msec 4 msec
9 163.17.7.252 4 msec 4 msec 4 msec
10 www.cyut.edu.tw (163.17.1.15) 4 msec 4 msec 4 msec
```



■ “真實的” 網際網路延遲與路徑 (Cont.)

► Traceroute 程式 (Conti.)

- Traceroute應用程式: PingPlotter (作者推薦的圖形介面- free)
- <http://www.pingplotter.com/download.html>

The screenshot shows the 'Ping Plotter Download' page from the website <http://www.pingplotter.com/download.html>. The page is displayed in Microsoft Internet Explorer. The left sidebar has a 'Download' link highlighted. The main content area describes the software's availability in three editions: Pro, Standard, and Freeware, and provides links to purchase or download. It also includes sections for 'New to PingPlotter?' and 'Current Releases'.

PingPlotter is available in three editions - Pro, Standard and Freeware. See our feature comparison for a list of difference between these.

If you're new to PingPlotter, or even an experienced user, please check out the [PingPlotter Getting Started Guide](#) to help you get up to speed quickly in the use PingPlotter after you have it downloaded.

PingPlotter runs on the Microsoft Windows platform, including Windows 98, ME, NT4, 2000, XP, 2003, Vista, 2008 and Windows 7. See our [system requirements page](#) for more details.

The Standard and Pro downloads allow you to evaluate the complete feature set for 30 days to see which edition is right for you.

New to PingPlotter?
PingPlotter Standard is recommended!

Current Releases

Product	Download Link	Additional Info	Purchase
PingPlotter Standard Version 3.30.4s, released July 27th, 2010	Download 2,307 KB via: HTTP or FTP	Release notes Product information	Buy Now
PingPlotter Pro Version 3.30.4p, released July 27th, 2010	Download 3,248 KB via: HTTP or FTP	Release notes Product information	Buy Now

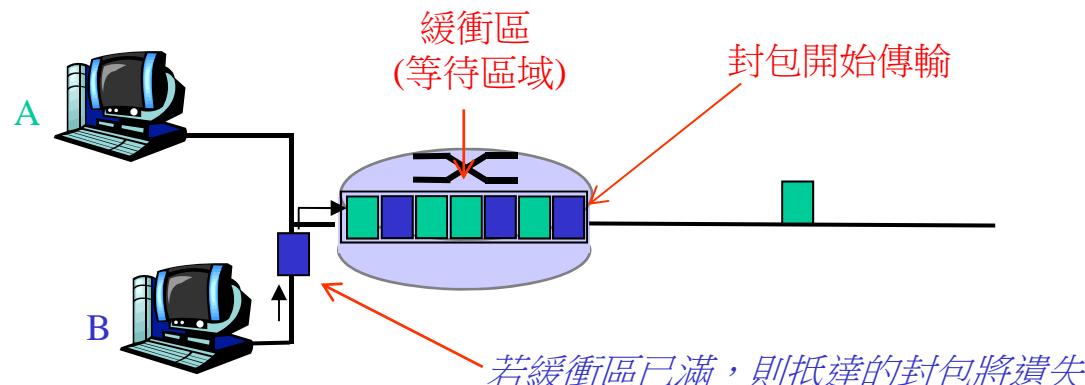
http://www.pingplotter.com/downloads/pngplt_std.exe



■ 封包遺失 (packet loss)

▶ 發生的原因

- 連結(link)之前的佇列 (router緩衝區) 具有有限的容量
- 若佇列已滿，則抵達的封包會被丟棄 (遺失)
- 遺失的封包可能會被前一個節點或來源端系統重新傳送，
也可能完全不被重新傳送

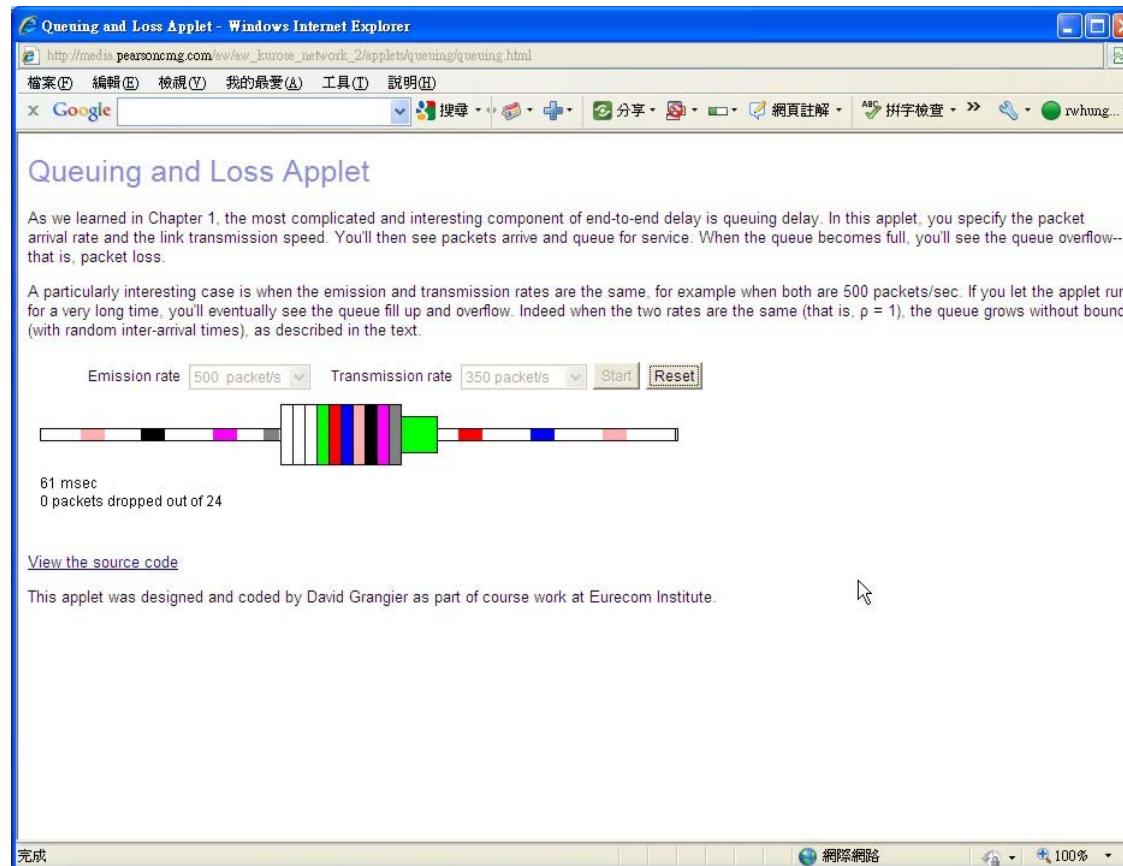




■ 併列延遲和封包遺失

► Java Applet

○ http://media.pearsoncmg.com/aw/aw_kurose_network_2/applets/queuing/queuing.html





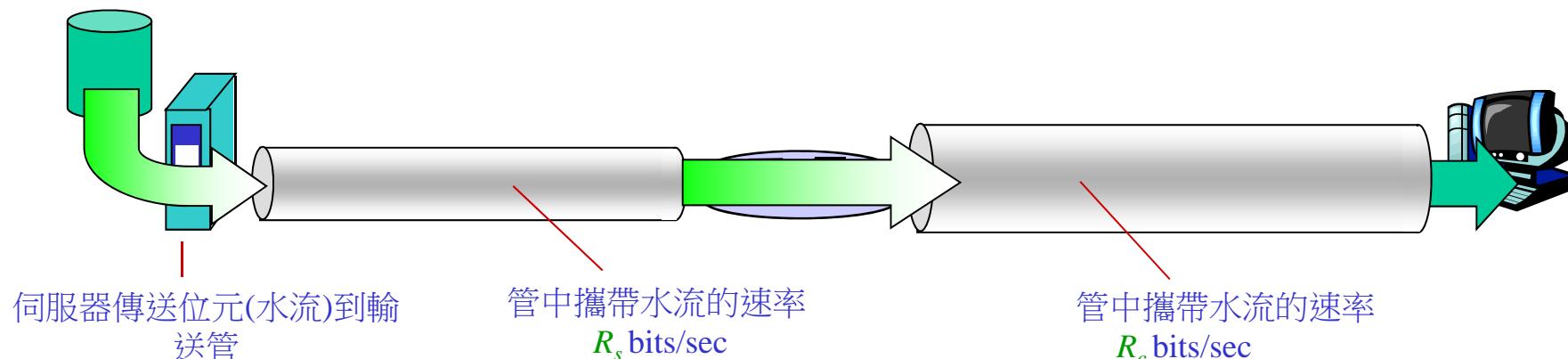
■ 產出率 (throughput)

▶ 定義

- 在傳送端/接收端之間的位元轉輸速率(位元/每秒)

▶ 種類

- 瞬間產出率：任一片刻的位元速率
- 平均產出率：某一段時間的平均位元速率

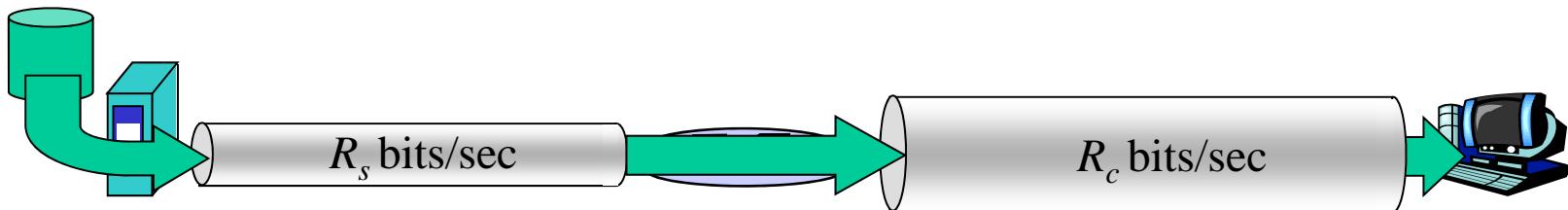




■ 產出率(Cont.)

► 討論

- $R_s < R_c$ 終端到終端的平均產出率為何？
- $R_s > R_c$ 終端到終端的平均產出率為何？



- 瓶頸連結

- 連結在終端到終端的路徑限制了終端到終端的產出率
 - $\min\{R_s, R_c\}$



■ 產出率 (Cont.)

► 端點到端點產出率

- 每條連接終端到終端的產出率為 $= \min(R_c, R_s, R/10)$
- 現實的情況： R_c 或 R_s 通常為瓶頸連結

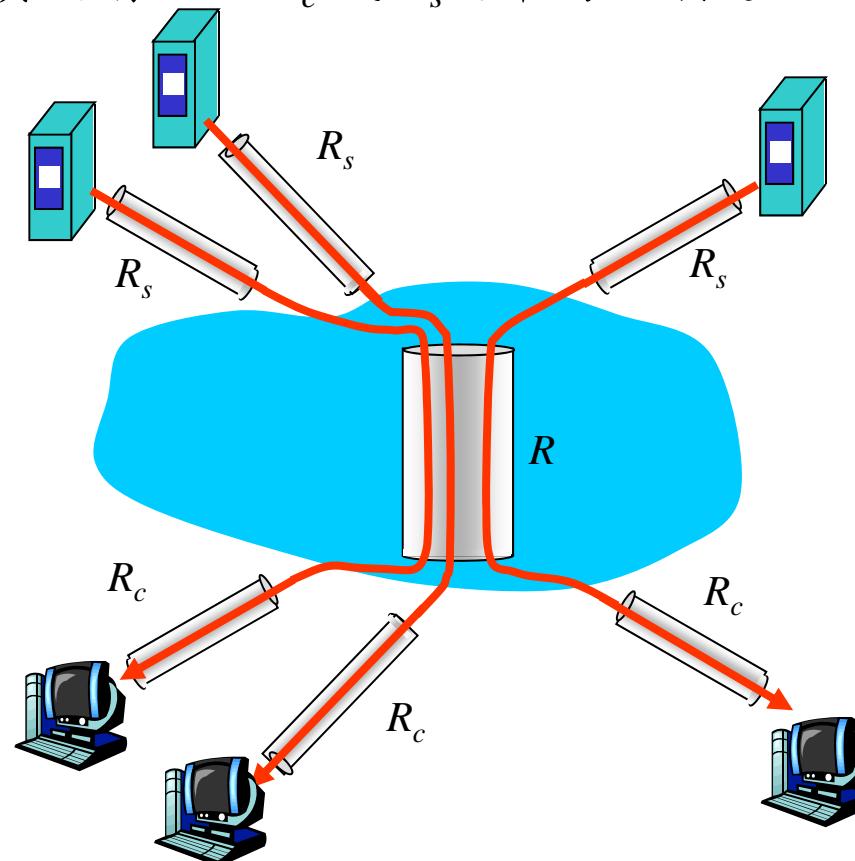
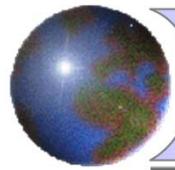
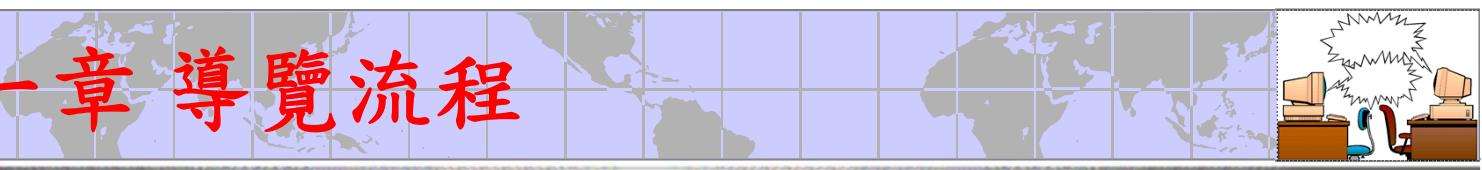


圖1.20 端點到端點產出率 – 10條連接(完全)均分主幹的瓶頸連結 R bits/sec

Exercise
1.4
Exercise
1.3



第一章 導覽流程



1.1 什麼是網際網路 (Internet) ?

1.2 網路的邊際 (Network Edge)

▶ 終端系統、連線網路、連結

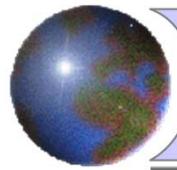
1.3 網路的核心 (Network Core)

▶ 電路交換、封包交換、網際網路結構

1.4 網路的效能 (Network Performance)

▶ 封包交換網路的延遲、封包遺失和產出率

1.5 協定分層及其服務模型 (Protocol Layers and Their Service Models)



1.5 協定分層及其服務模型 (Protocol Layers and Their Service Models)



■ 協定“分層”

► 網路是很複雜的，它包含許多“部份”：

- 主機
- 路由器
- 各種媒介的連結
- 應用程式
- 協定
- 硬體/軟體

► 如何讓硬體/軟體在網路中得以溝通(通訊)?



■ 分層式架構

► 比喻 - 飛機旅行的組織



圖1.21 搭乘飛機旅行

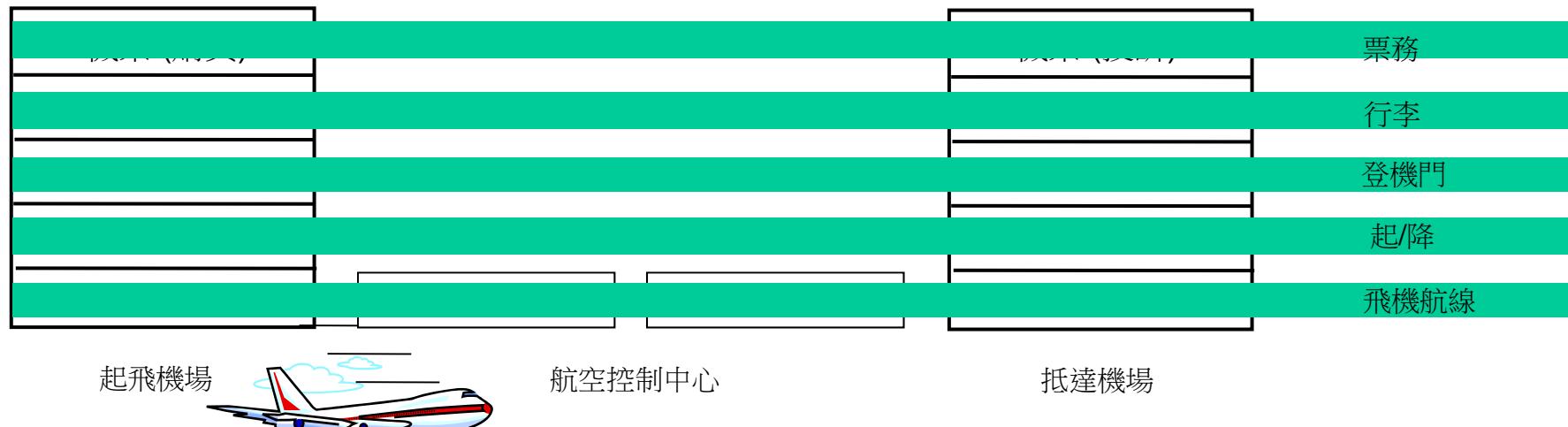
⇒ 一系列的步驟



■ 分層式架構 (Cont.)

► 比喻 - 飛機旅行的組織 (Cont.)

- 航空公司功能的分層



► 分層(layering)：每一個分層實作一個服務

- 每一個層級內執行某些內部動作
- 每一個層級使用其下一層所提供的服務



■ 分層式架構 (Cont.)

► 為什麼要分層？

- 處理複雜的系統
 - 模組化
 - 容易修改分層所提供的服務實作方式
 - 當某一層級修改其功能，其它層級可以維持不變 (eg. 改變登機門的程序並不會影響到系統的其它部份)
- 分層被視為有害嗎？



■ 網際網路協定堆疊 (protocol stack)

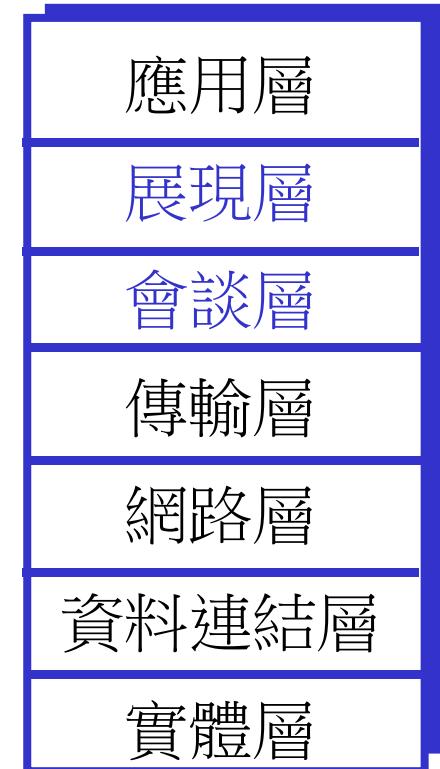
- ▶ 應用層：支援網路應用程式
 - FTP、SMTP、HTTP
- ▶ 傳輸層：應用程式端點到端點間的傳輸
 - TCP、UDP
- ▶ 網路層：從來源端到目的端的資料段路由
 - IP、路由協定
- ▶ 連結層：在相鄰的網路元件間傳輸資料
 - PPP、乙太網路
- ▶ 實體層：“電路中的”位元





■ ISO/OSI 參考模型 (reference model)

- ▶ 展現層：允許應用程式解讀資料的意義，例如：資料加密、資料壓縮、不同電腦之間的轉換
- ▶ 會談層：同步化、檢查點、回復資料的交換
- ▶ 網際網路缺少這兩層！
 - 這些服務需不需要？如果需要，必須要在應用程式中加入這個功能
 - 需要嗎？



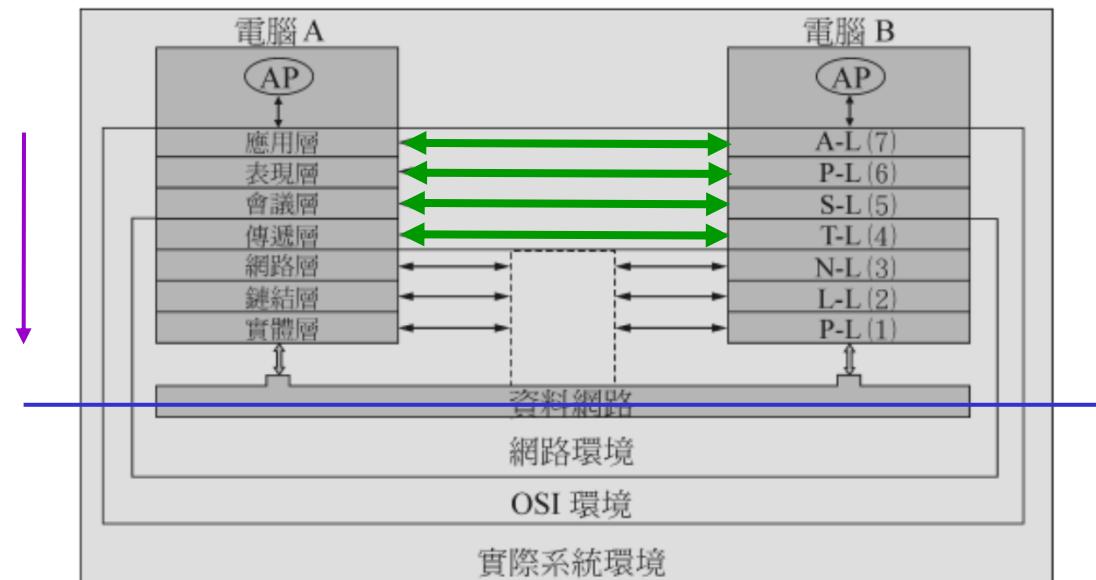
投影片 80

Hung1

ISO = International Organization for Standardization;
OSI = Open Systems Interconnection.
Ruo-Wei Hung, 2011/10/1



■ ISO/OSI 參考模型 (Cont.)

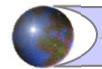




■ ISO/OSI 參考模型 (Cont.)

► ISO封包





■ 封裝 (Encapsulation)

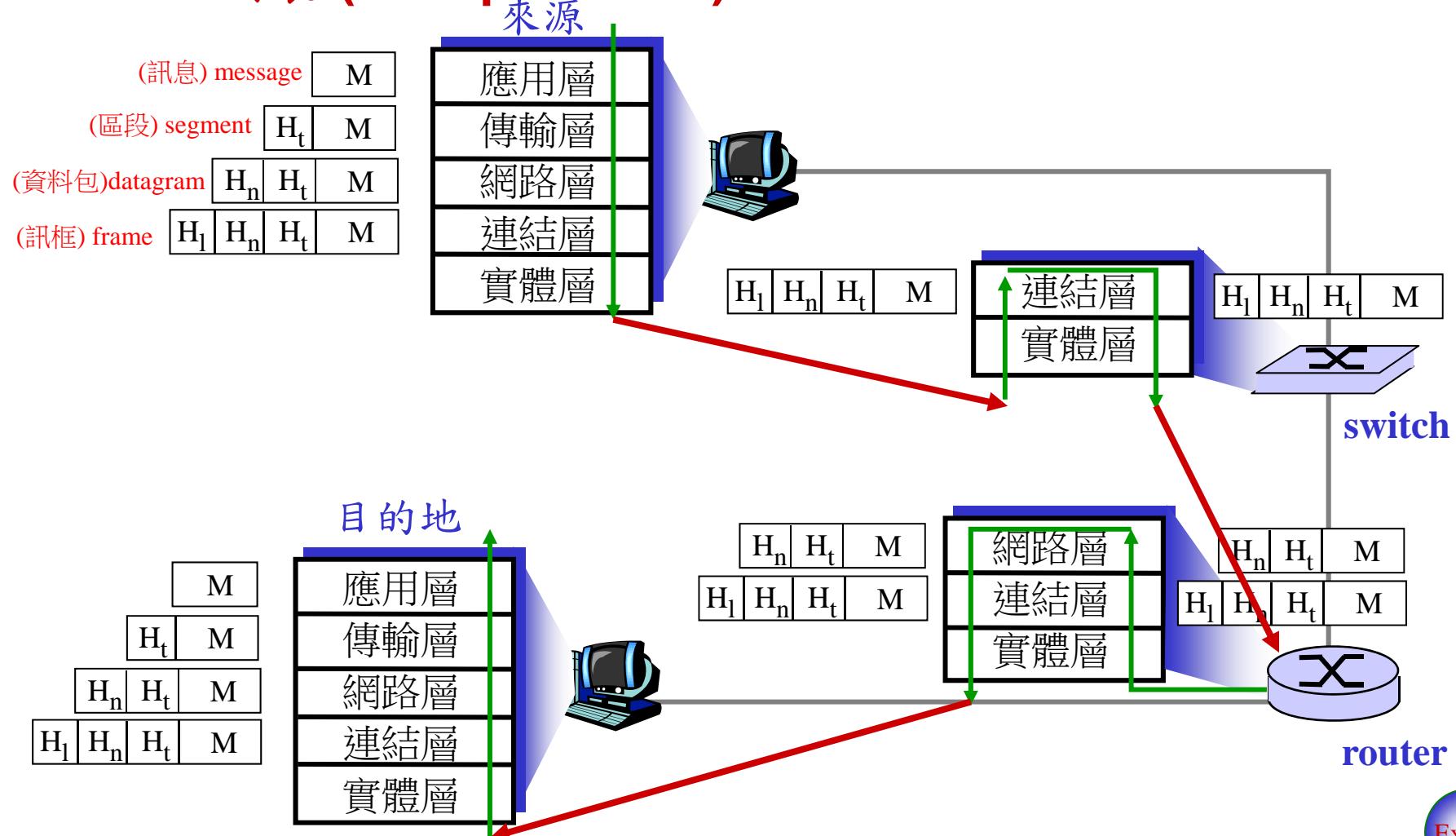


圖1.24 Protocol layering – data encapsulation



■ 網際網路分層模式的各層簡易功能表

網際網路分層層級	網際網路層級名稱	功能說明
第5層	應用層 (Application layer)	應用層提供了使用者網路上的服務，如網頁存取、檔案傳輸、電子郵件、網路電話、...等
第4層	傳輸層 (Transport layer)	傳輸層主要是確保資料在應用層與網路層之間的傳輸品質，即正確、沒有遺失、沒有重複 (TCP vs. UDP)
第3層	網路層 (Network layer)	網路層管理節點到另一節點的路徑，其負責建立、維護及終止兩個節點間的連結路徑，使資料傳輸依其規劃的路徑傳送。因此，其必須有IP定址的功能 (IP)
第2層	連結層 (Link layer)	連結層負責確保實體層連結的資料之正確性，包括資料傳輸的錯誤偵測/更正。由連結層建立一條端點間的可靠通訊通道，使網路層得以正確地存取實體層的資料
第1層	實體層 (Physical layer)	實體層負責資料位元在實體傳輸媒體上的傳輸，使電器訊號可在兩個裝置間交換，主要包括網路的電器規格，如電壓、電流準位、連接器種類、...等

Exercise
1.5



總結

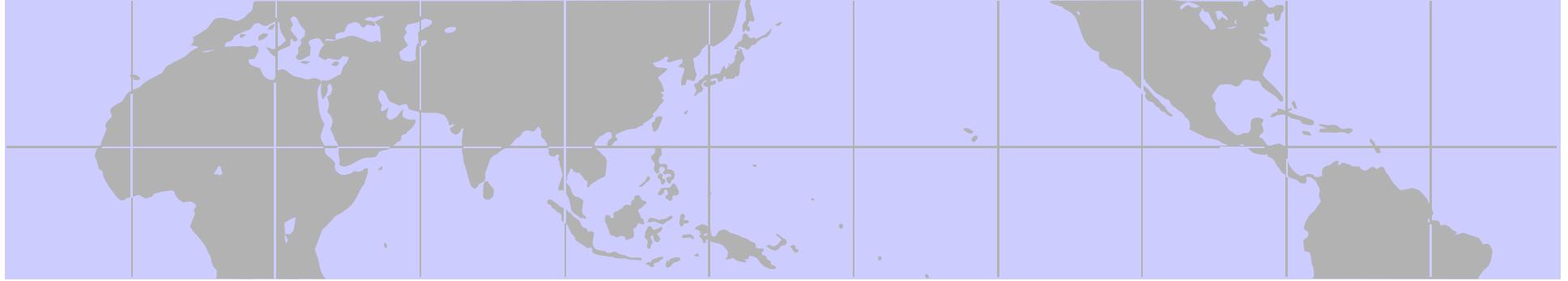
■ 涵蓋大量的題材

- ▶ 網際網路概觀
- ▶ 什麼是協定？
- ▶ 網路的非核心部份、核心、存取網路
 - 封包交換 vs. 電路交換
 - 網際網路結構
- ▶ 效能：遺失、延遲、產出率
- ▶ 分層與服務模型

■ 你現在擁有

- ▶ 內容、概觀、網路的“感覺”
- ▶ 接下來有更深入的細節！

End of Chapter 1

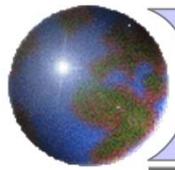


Chapter 2

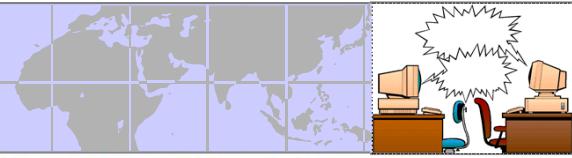
Application Layer

(應用層)





Topic Overview



2.1 網路應用的原理

2.2 Web 和 HTTP

2.3 FTP

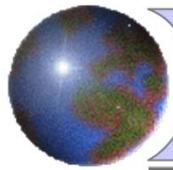
2.4 電子郵件 – SMTP、POP3、IMAP

2.5 DNS

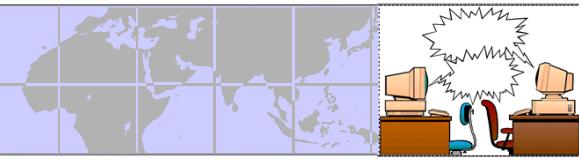
2.6 點對點應用

2.7 使用TCP的Socket程式設計

2.8 使用UDP的Socket程式設計



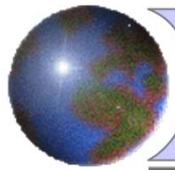
Topic Overview (Cont.)



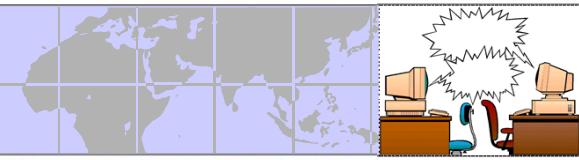
■ 目標 (Goal)

- ▶ 網路應用協定的概念及實作
 - 傳輸層服務模型
 - 用戶端 - 伺服端模式
 - 點對點架構
- ▶ 藉著驗證常用的應用層協定來學習協定
 - HTTP
 - FTP
 - SMTP / POP3 / IMAP
 - DNS
- ▶ 撰寫網際網路應用程式 (HW – 5%)
 - socket API



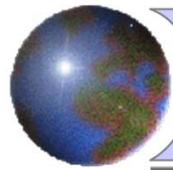


Topic Overview (Cont.)



■ 一些網路應用程式

- ▶ 電子郵件
- ▶ Web
- ▶ 即時訊息
- ▶ 遠端登錄
- ▶ 點對點檔案分享
- ▶ 多使用者網路遊戲
- ▶ 串流式儲存視訊片段
- ▶ 影像分享 IP
- ▶ 即時視訊會議
- ▶ 網格運算



Topic Overview (Cont.)

■ 產生一個網路應用程式

▶ 撰寫一個程式

- 在不同的終端系統上執行且透過網路通訊
- 例如：Web 伺服器軟體與瀏覽器軟體通訊

▶ 為網路核心裝置撰寫小軟體

- 網路核心裝置不執行使用者應用程式
- 終端系統的應用程式允許快速應用的建置與傳遞

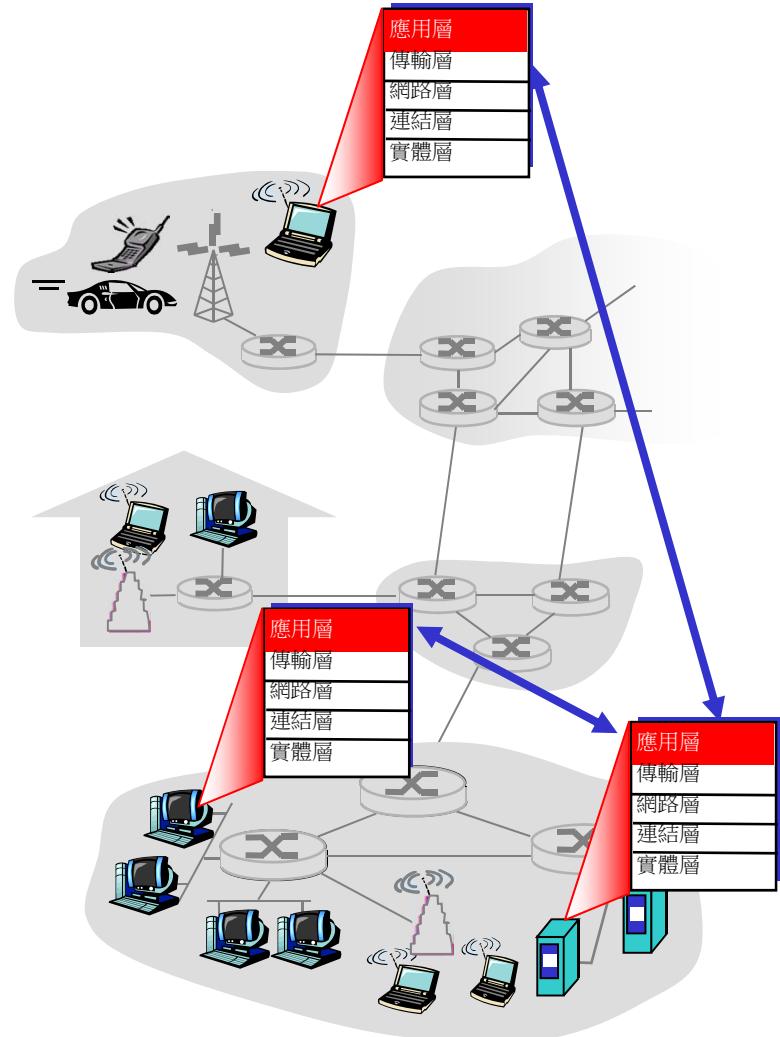
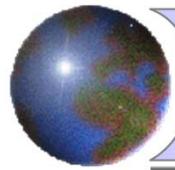


圖2.1 應用程式在終端系統間所進行的應用層通訊



第二章 導覽流程



2.1 網路應用的原理

2.2 Web 和 HTTP

2.3 FTP

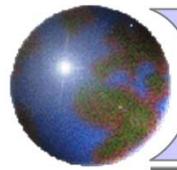
2.4 電子郵件 – SMTP、POP3、IMAP

2.5 DNS

2.6 點對點應用

2.7 使用TCP的Socket程式設計

2.8 使用UDP的Socket程式設計

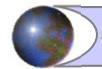


2.1.1 網路應用程式架構



■ 應用程式架構 (application architecture)

- ▶ 用戶端-伺服端架構 (client-server)
- ▶ 點對點架構 (P2P – peer to peer)
- ▶ 用戶端-伺服端和點對點二者混合式的架構 (hybrid)



■ 用戶端-伺服端架構

▶ 伺服器：(server)

- 隨時服務
- 固定 IP 位址
- 以伺服器群(server farm)
延伸擴充性

▶ 用戶端：(client)

- 與伺服器通訊
- 也許是間斷地連結
- 也許有動態 IP 位址
- 彼此間不會直接通訊

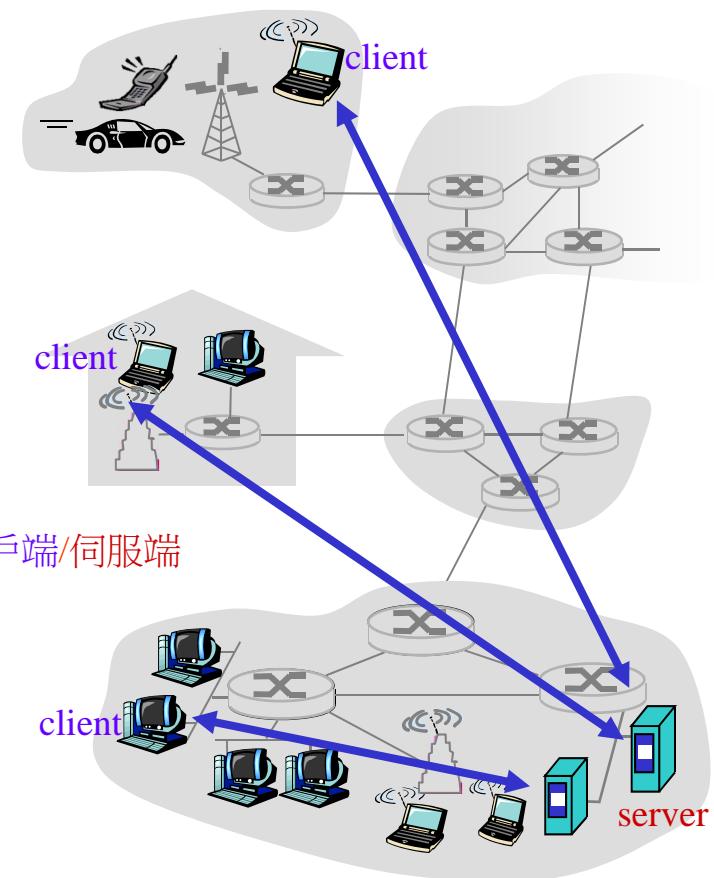
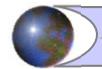


圖2.2(a) 用戶端/伺服端架構



■ 點對點架構

- ▶ 沒有隨時服務的伺服器
 - ▶ 任意的終端系統直接通訊
 - ▶ 對等點間斷地連結和
更換IP位址
 - ▶ 例如：P2P檔案傳輸
- ⇒ 很高的擴張性但是難以管理

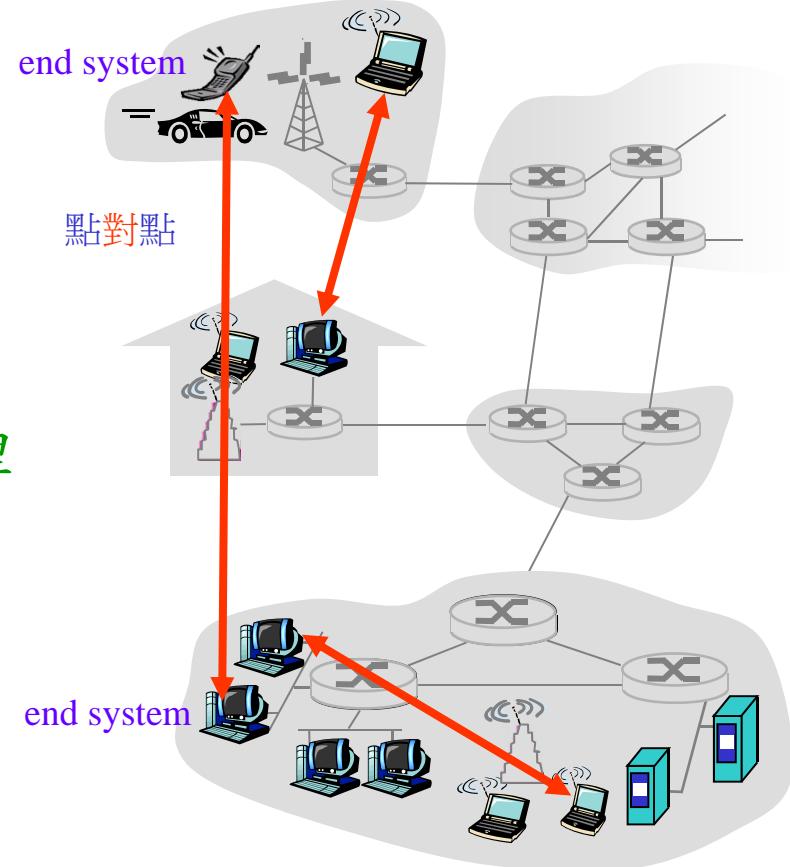


圖2.2(b) 點對點架構



■ 點對點架構 (Cont.)

► P2P應用面臨的挑戰 (p. 2-5)

- ① 對ISP友善 – ISP提供的上傳與下載頻寬一般都不一致 (why?)，然而，P2P應用的上傳與下載一般都是同等對待
- ② 安全性 – P2P應用有高度分散與開放的本質，故比client-server架構更有安全性的問題
- ③ 誘因 – 如何說服P2P使用者願意貢獻其頻寬、儲存空間及運算資源給P2P應用程式使用



■ 用戶端-伺服端和點對點二者混合式的架構

▶ 網際網路電話

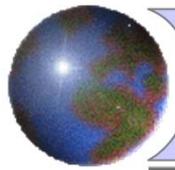
- P2P網路電話的應用
- 中介伺服器：找到遠端連線位址的部份
- 用戶端-用戶端的連接：直接的（不通過伺服器）
- See Sect. 2.6.3

▶ 即時訊息(MSN/LINE)

- 兩個使用者的交談是點對點的
- 存在的偵測/尋找是中央式的
- 當使用者上線時，會將它的IP位址跟中央伺服器註冊
- 使用者與中央伺服器連繫以取得好友的IP位址



Ex 2.1



2.1.2 行程通訊 (process communicating)



■ 行程

- ▶ 在主機內執行的程式 (執行中的程式)
- ▶ 在同一個主機中，兩個行程以行程內通訊機制來互相通訊 (被作業系統管制)
- ▶ 在不同的主機間，行程透過交換訊息來通訊

■ 用戶端行程 vs. 伺服端行程

- ▶ 用戶端行程：啟動通訊的行程 \Rightarrow 主動
- ▶ 伺服端行程：等待被連接的行程 \Rightarrow 被動

\Rightarrow 在點對點架構中，應用程式擁有用戶端行程以及伺服端行程



■ Sockets (門/插槽/插座)

- ▶ 行程 傳送/接收 訊息 給/從 它的 socket
- ▶ socket 好比是「門」
 - 傳送端行程將訊息推出它的門
 - 傳送端行程依靠門另一端的運輸架構，將訊息帶到接收端行程的socket

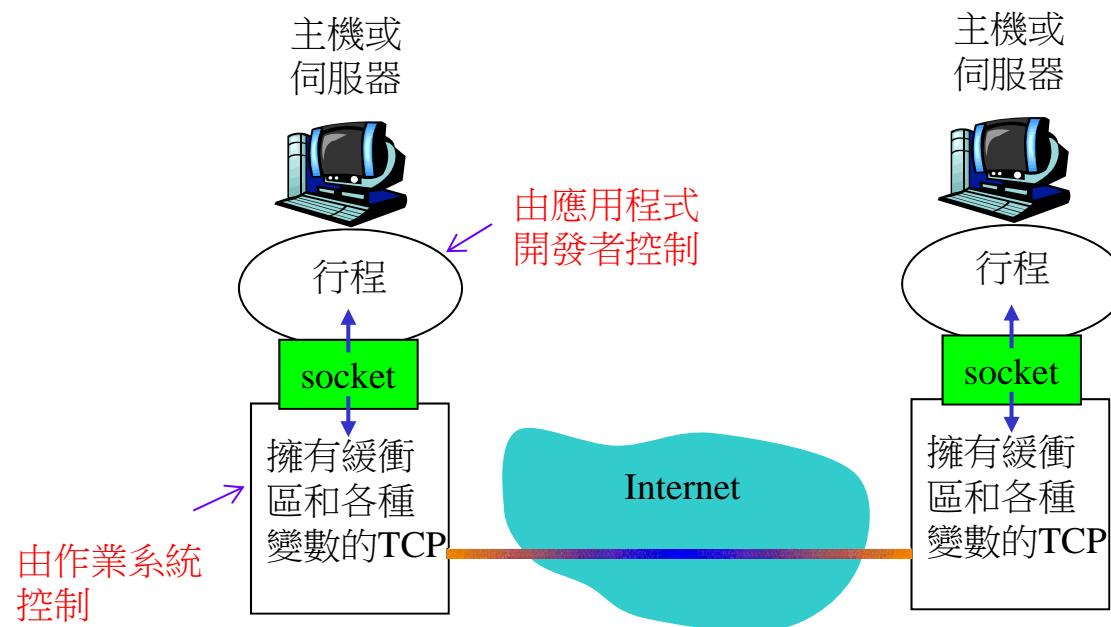


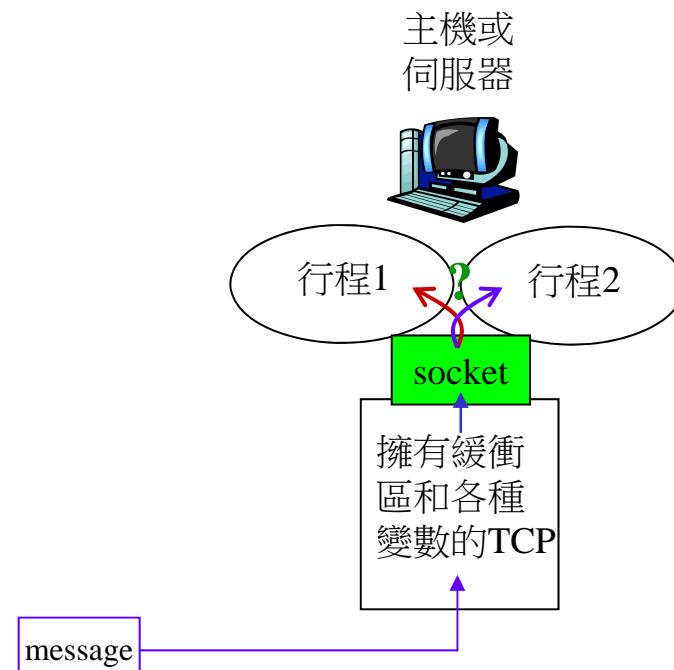
圖2.3 應用程式、socket、及底層的傳輸協定的關係

Ex 2.2



■ 行程定址

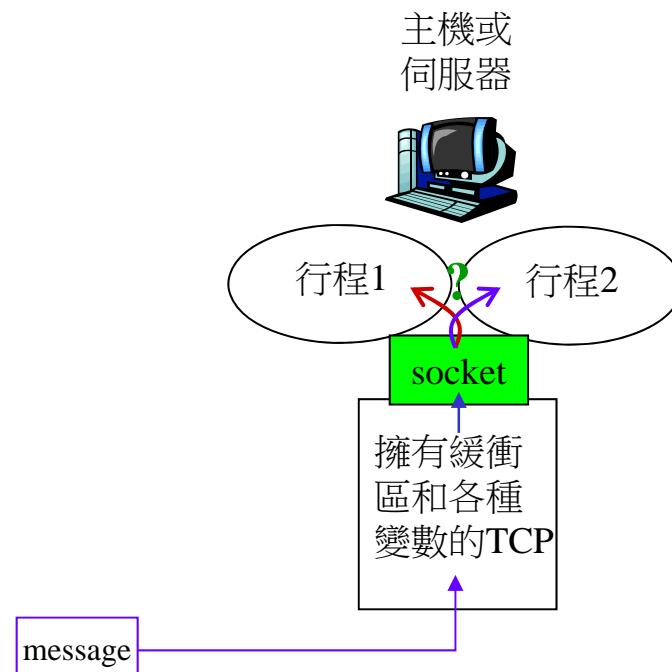
- ▶ 接收到信息，此過程需要經過識別
- ▶ 一台主機有一個唯一的32位元 IP位址 (IPv4)
- ▶ Q：在行程執行的主機上，IP位址足夠用來識別行程嗎？





■ 行程定址 (Cont.)

- ▶ Q：在行程執行的主機上，IP位址足夠用來識別行程嗎？
- ▶ A：不行，可能有許多行程在同一台主機上執行

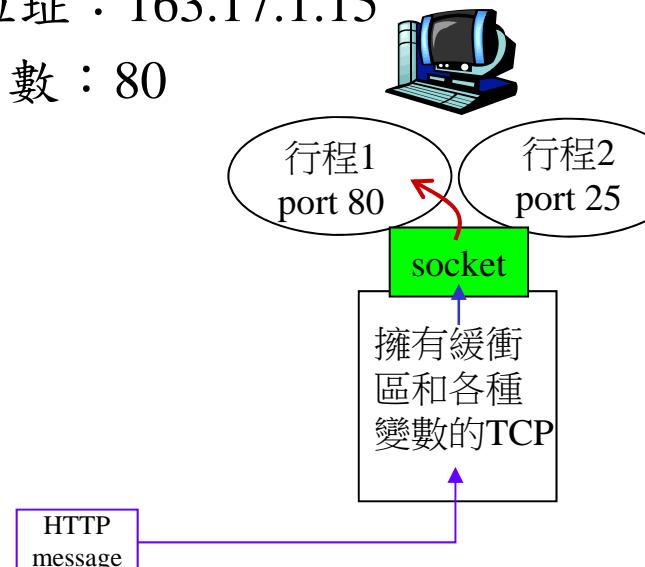




■ 行程定址 (Cont.)

- ▶ 識別碼包含了 IP位址 以及 埠號(port number)，後者屬於主機上的行程
- ▶ 埠號的範例：
 - HTTP 伺服器：80
 - Mail 伺服器：25
- ▶ 寄送HTTP信息到www.cyut.edu.tw網頁伺服器：
 - IP 位址：163.17.1.15
 - Port 數：80

public port number of application:
<http://www.iana.org>



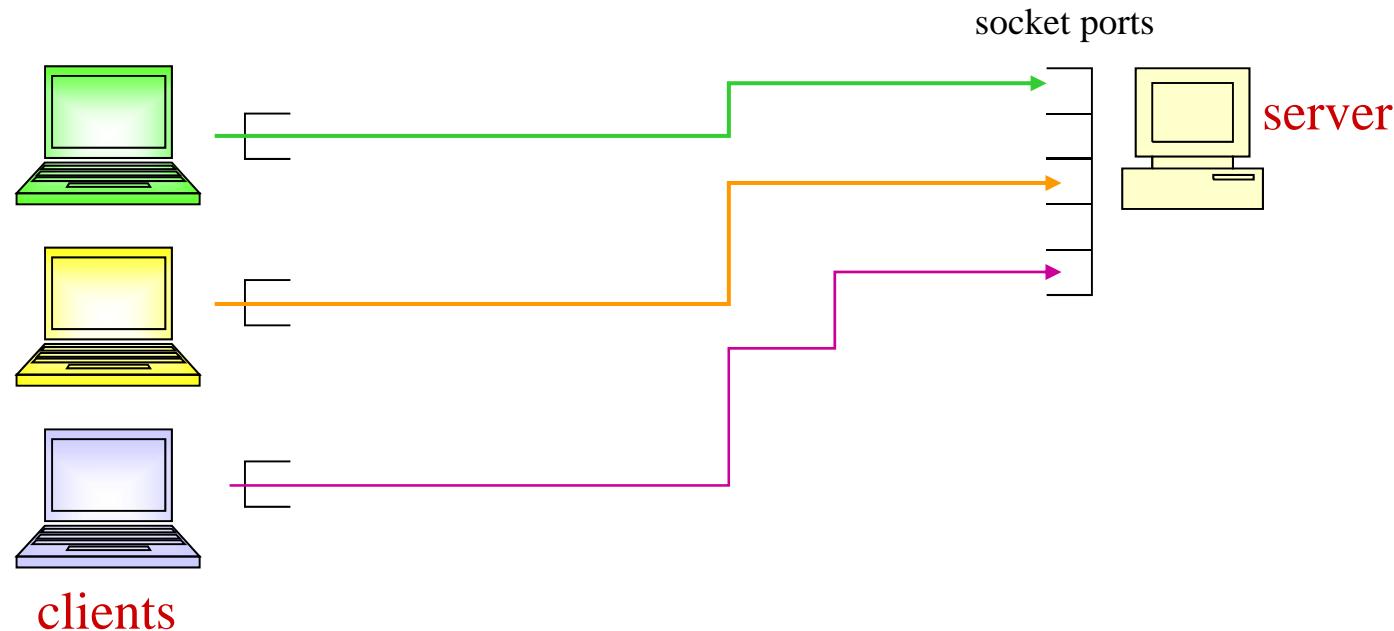
Ex 2.3





■ Socket 再論

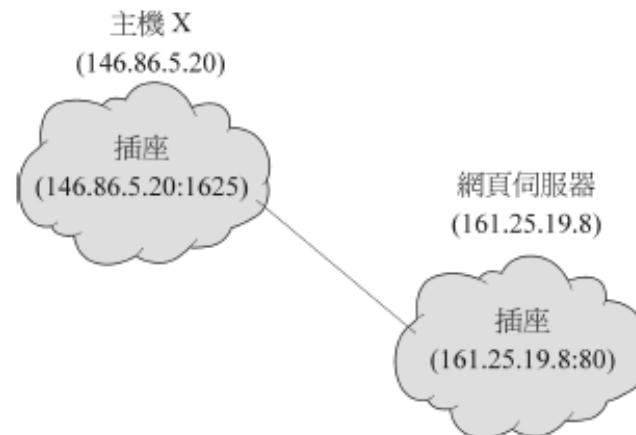
► 一台伺服主機執行多個伺服程式

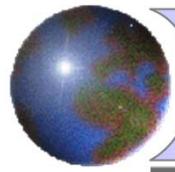




■ Socket 再論 (Cont.)

- ▶ Socket(門/插座)可由一個 IP位址 和一個 埠號(port number) 所組成
- ▶ 一組行程使用一對插座(雙方各一個)在網路上通訊





2.1.3 應用程式可使用的傳輸服務



■ 應用層協定的定義 (ref. 圖 2.8)

- ▶ 所交換訊息的類型
 - 例如：請求訊息和回應訊息。
- ▶ 各種訊息類型的語法：
 - 例如：訊息中的欄位以及如何描述這些欄位。
- ▶ 訊息的語義
 - 意指欄位內的資訊所代表的意義。
- ▶ 決定行程何時傳送訊息和回應訊息，以及如何傳送訊息和回應訊息的規則。

■ 公眾領域的協定：

- ▶ 定義在RFC中
- ▶ 允許可互通性
- ▶ 例如：HTTP、SMTP

■ 私有的協定：

- ▶ 例如：網際網路電話 (Skype)。

有些私有的協定，
在普遍後也可能變成公眾領域的協定



■ 應用程式需要哪些傳輸服務？

- ▶ 傳輸層協定能夠提供哪些服務給呼叫它的應用程式？
- ▶ 資料遺失 (reliable data transfer)
 - 某些應用程式 (例如：語音) 可以忍受一些遺失
 - 其它的應用程式 (例如：檔案傳輸、遠端終端機存取) 要求 100% 可靠的資料傳輸
- ▶ 時序 (timing)
 - 某些應用程式 (例如：網際網路電話、互動遊戲) 需要低的延遲以確保效能
- ▶ 頻寬 (bandwidth)
 - 某些應用程式 (例如：多媒體) 有最小頻寬的需求以確保其效能
 - 其它的應用程式 (彈性使用的應用程式) 使用它們可以取得的任何頻寬



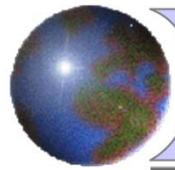


■ 常見網路應用程式的傳輸服務需求

網路應用	資料遺失	頻寬	對時序的敏感性
檔案傳輸	不容忍遺失	彈性使用	無
電子郵件	不容忍遺失	彈性使用	無
網頁文件	不容忍遺失	彈性使用	無
網路電話	可容忍遺失	音訊: 數Kbps ~ 1 Mbps 視訊: 10 Kbps ~ 5 Mbps	有, 數百毫秒(ms)
視訊會議			
預儲影音	可容忍遺失	同上	有, 幾秒鐘
互動遊戲	可容忍遺失	幾 Kbps ~ 10Kbps	有, 數百毫秒
即時訊息	不容忍遺失	彈性使用	可有可無

圖2.4 某些網路應用的需求





2.1.4 網際網路傳輸層協定的服務



■ TCP 服務

- ▶ 連線導向服務 在用戶端和伺服端行程之間需要準備設定
- ▶ 可靠的傳輸服務 在傳送端和接收端行程之間
- ▶ 流量控制 傳送端傳輸資料量不會超過接收端可接收的資料量
- ▶ 壓塞控制 當超過負荷時，將傳送端減速
- ▶ 沒有提供 時序、最小頻寬的保證

■ UDP 服務

- ▶ 傳送端和接收端行程之間 **不可靠的資料傳輸**
- ▶ **不提供**：連接設定、可靠性、流量控制、時序、或頻寬保證

Q：為何會有 UDP ? 。○ ○

當網路應用不需要可靠的資料傳輸時，採用此種協定服務將不需作 handshaking，如此可加速資料傳輸的速度及減少網路資料流量。



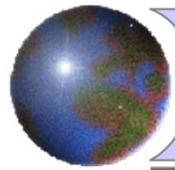


■ 網際網路應用：應用程式 vs. 其下之傳輸層協定

網路應用	應用層協定	底層傳輸層協定
電子郵件	SMTP [RFC 2821]	TCP
遠端終端機存取	Telnet [RFC 854]	TCP
網頁	HTTP [RFC 2616]	TCP
檔案傳輸	FTP [RFC 959]	TCP
串流多媒體	HTTP (例如 YouTube) RTP	TCP 或 UDP
網際網路電話	通常是私有協定 (例如 Skype)	一般是 UDP

圖2.5 某些網路應用、它們的應用層協定、及其使用的底層傳輸層協定





第二章 導覽流程



2.1 網路應用的原理

2.2 Web 和 HTTP

2.3 FTP

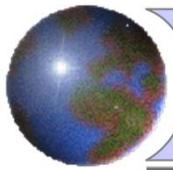
2.4 電子郵件 – SMTP、POP3、IMAP

2.5 DNS

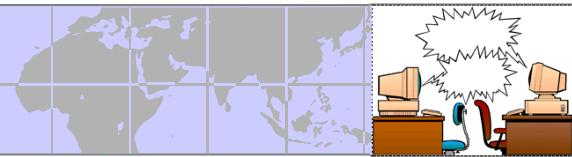
2.6 點對點應用

2.7 使用TCP的Socket程式設計

2.8 使用UDP的Socket程式設計



2.2 Web(網頁) 和 HTTP



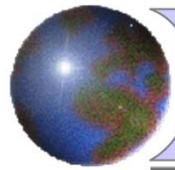
■ 基本術語 (terminology)

- ▶ WWW – Word Wide Web (全球資訊網)
 - ▶ 網頁([web page](#)) 由 物件(objects) 所組成
 - ▶ 物件可能是 HTML檔案、JPEG影像、Java applet、聲音檔、...
 - ▶ 網頁由 基本的HTML檔案 和數個 [參考物件](#) 所組成
 - ▶ 每個物件由一個 URL (uniform resource locator)來定址
 - ▶ 範例 URL：(注意其分隔符號 ‘/’)

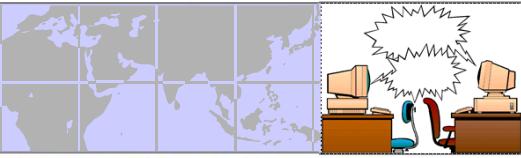
<http://www.cyut.edu.tw/~rwhung/Course/Network.jpg>

主機名稱

路徑名稱



2.2.1 HTTP 概觀



■ HTTP

- ▶ 超文件傳輸協定(hypertext transfer protocol)
- ▶ Web的應用層協定
- ▶ 用戶端/伺服端模型
 - 用戶端：瀏覽器，用來請求、接收及「顯示」Web物件
 - 伺服端：Web 伺服器，依請求回應並傳送物件
- ▶ HTTP 1.0 : RFC 1945
- ▶ HTTP 1.1 : RFC 2068

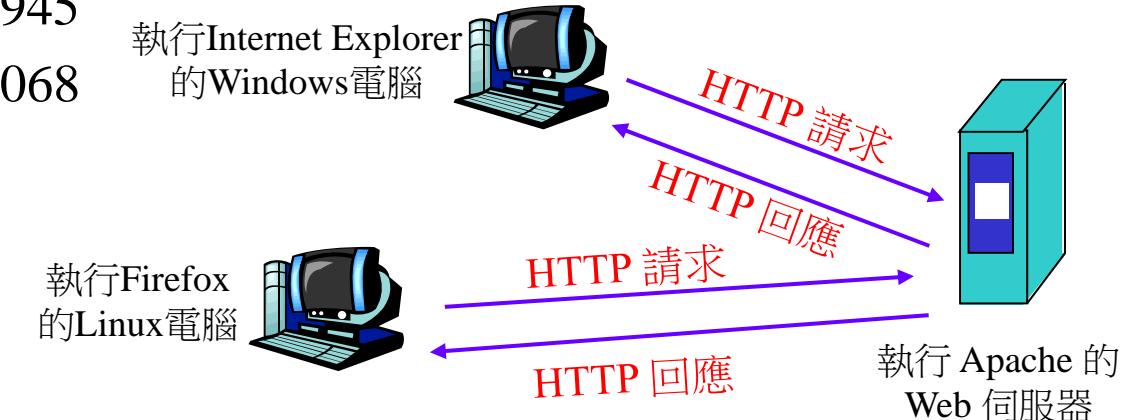


圖2.6 HTTP的請求-回應行為

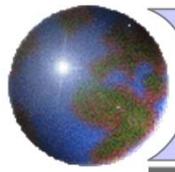


■ 使用 TCP：

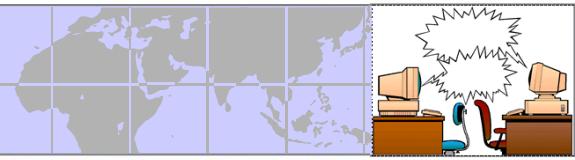
- ▶ 用戶端先建立與伺服端的 TCP連線(產生socket)，port 80
- ▶ 伺服端接受來自用戶端的 TCP連線
- ▶ HTTP 訊息(應用層協定訊息)在瀏覽器(HTTP用戶端)和 Web 伺服器(HTTP伺服端)之間交換
- ▶ TCP 連線結束

■ HTTP 是 “無狀態的協定” (stateless protocol)

- ▶ HTTP伺服端不會儲存/維護用戶端過去的請求資訊



2.2.2 HTTP 連線



■ HTTP 連線方式

▶ 非永久性 HTTP

- 一個TCP連線上最多傳送一個物件
- HTTP/1.0 使用非永久性 HTTP

three-way
handshake Hung1

▶ 永久性 HTTP

- 在客戶端和伺服端之間的單一TCP連線上，可以傳送多個物件
- HTTP/1.1 預設使用永久性連線

Hung1

TCP uses a three-way handshake to open a connection:

- (1) ACTIVE OPEN: Client sends a segment with:
 - SYN bit set *
 - port number of client
 - initial sequence number (ISN) of client
- (2) PASSIVE OPEN: Server responds with a segment with:
 - SYN bit set *
 - initial sequence number of server
 - ACK for ISN of client
- (3) Client acknowledges by sending a segment with:
 - ACK ISN of server

Ruo-Wei Hung, 2013/8/29



■ 非永久性 HTTP :

▶ 用戶端瀏覽程式輸入

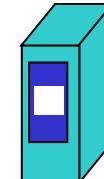
<http://www.cyut.edu.tw/~rwhung/index.htm>

(內含10個.jpg物件)

client



server



1a. HTTP用戶端利用埠號80

建立與www.cyut.edu.tw伺服器的TCP連線

1b.位於 www.cyut.edu.tw的 HTTP 伺服器主機
等待埠號80的 TCP 連線。
「接受」連線並通知用戶端

2. HTTP用戶端透過TCP連線的socket，
傳送一個HTTP請求訊息給伺服器。

這個訊息指出用互端需要物件檔案/~rwhung/index.htm

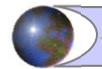
3. HTTP伺服器收到請求訊息，產生
回應訊息，內含所請求的物件，
並將此訊息送到它的 socket

4. HTTP 伺服器關閉 TCP 連線

5. HTTP 用戶端收到回應訊息，其中包含 html 檔，
顯示html。解析 html 檔, 找到 10 個參考的 jpeg 物件

6. 對 10 個 jpeg 物件，每個都重複步驟 1-5





■ 回應時間模型：

- ▶ RTT (round-trip time, 來回時間) 的定義：一個小封包由用戶端傳送到伺服端，然後再回來所經歷的時間。
- ▶ 回應時間：
 - 一個 RTT：初始化 TCP 連線 (建立 sender 與 receiver 間的 "管子")
 - 一個 RTT：HTTP 請求以及回傳的 HTTP 回應訊息的前幾個位元組
 - 檔案傳輸時間
 - 總計 = 2RTT + 檔案傳輸時間

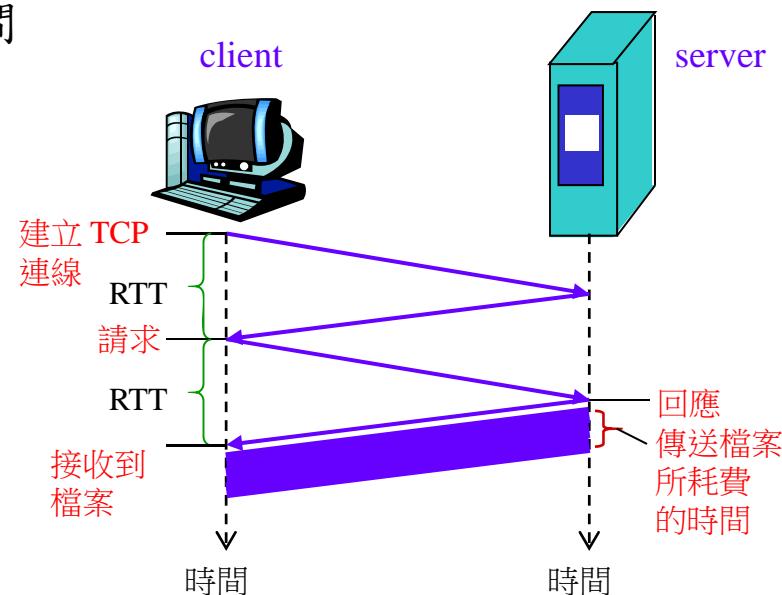


圖2.7 請求/接收HTML檔案所需時間的簡單計算

Exercise
2.3



■ 永久性 HTTP :

▶ 非永久性 HTTP 問題

- 每個物件都需要 2 個 RTT
- 每個 TCP 連線都造成 OS(作業系統) 的額外負擔
- 瀏覽器通常會開啟平行的 TCP 連線以取得參考物件

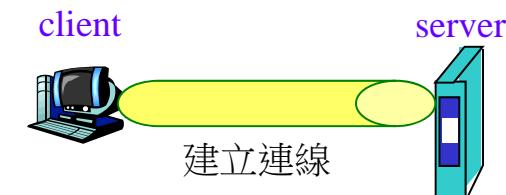
▶ 永久性 HTTP

- 在傳送回應之後，伺服器會持續保留該條連線
- 在相同的用戶端/伺服端之間，後續的 HTTP 訊息會經由此開啟的連線傳送



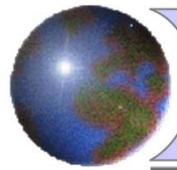
○ 類別

- 無管線的永久性：
 - » 每個參考物件需要一個 RTT
- 管線化的永久性：
 - » 所有的參考物件可能只需要一次RTT
 - » 不需等待尚未處理完的請求得到回應，而送出下一個物件
 - » HTTP/1.1 的 預設模式

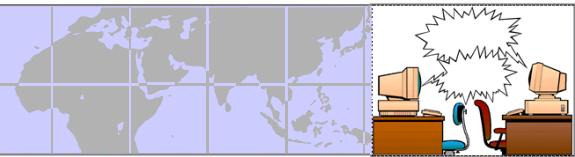


同一個TCP
連線可傳送
多個物件



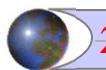


2.2.3 HTTP 訊息格式



■ 兩種 HTTP 訊息

- ▶ 請求 (request message)
 - ▶ 回應 (response message)
- ⇒ ASCII (人類可閱讀的格式)



■ HTTP 請求訊息：

請求列

(GET, POST,
HEAD 命令)

GET /~rwhung/index.htm HTTP/1.1

Host: www.cyut.edu.tw

User-agent: Mozilla/4.0

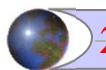
Connection: close

Accept-language:fr

非永久性連線

歸位+換行 (cr+lf)
表示訊息的尾端

(extra carriage return, line feed)



■ HTTP 請求訊息：一般格式

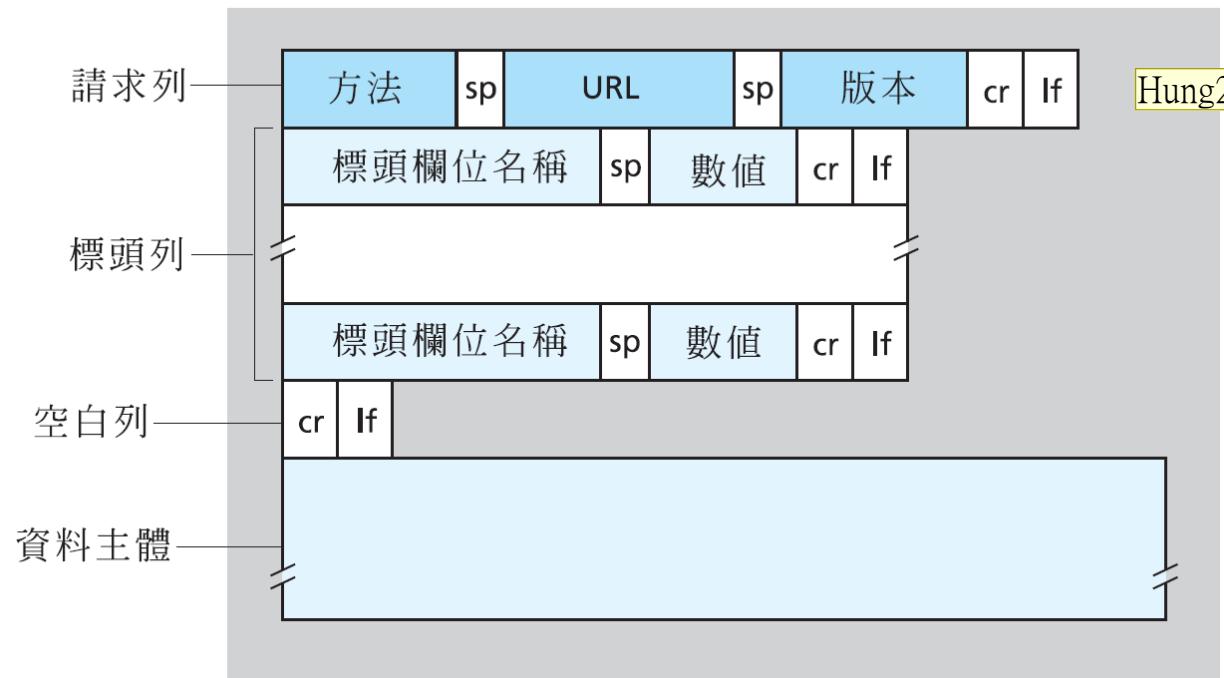


圖2.8 HTTP 請求訊息的一般格式

投影片 34

Hung2

sp = space

cr = carriage return

lf = line feed

Ruo-Wei Hung, 2013/8/29



■ HTTP 請求訊息 (Cont.)

▶ 上傳表單

- POST
- 資料主體為「輸入表單的資料」

▶ GET

- 資料主體為空

■ HTTP 回應訊息：

狀態列
(協定
狀態碼
狀態訊息)

HTTP/1.1 200 OK

Connection close

Date: Thu, 06 Aug 1998 12:00:15 GMT

Server: Apache/1.3.0 (Unix)

Last-Modified: Mon, 22 Jun 1998

Content-Length: 6821

Content-Type: text/html

標頭列

資料, 例如,
請求的
HTML 檔案

data data data data data ...



■ HTTP 回應訊息：(Cont.)

▶ HTTP 回應狀態碼

(伺服端→用戶端回應訊息的第一行)

▶ 幾個狀態碼的例子：

- 200 OK

- 請求成功，請求的物件包含在此訊息的後方。

- 301 Moved Permanently

- 請求的物件已經搬移；新的位置指定在此訊息的後方 (Location:)。

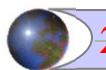
- 400 Bad Request

- 伺服器無法了解請求訊息。

- 404 Not Found

- 請求的文件不存在於該伺服器。

- 505 HTTP Version Not Supported



■ 自己試驗 HTTP (用戶端) :

Hung3

▶ Telnet 登入你喜歡的Web伺服器：

- telnet cis.poly.edu 80

▶ 輸入一個 GET HTTP 請求：

- GET /~ross/ HTTP/1.1
- Host: cis.poly.edu

藉著輸入這個訊息 (按下歸位鍵兩次)
你傳送了這個最小的 (但卻是完整的)
GET 請求到 HTTP 伺服器

▶ 觀察 HTTP 伺服器傳送的回應訊息！

```
HTTP/1.1 200 OK
Date: Tue, 13 Jul 2010 12:17:49 GMT
Server: Apache/1.2.5
Last-Modified: Tue, 22 Jun 2010 19:20:37 GMT
ETag: "2b3e-258f-4c210d05"
Content-Length: 9615
Accept-Ranges: bytes
Content-Type: text/html

<html
    xmlns:m="http://schemas.microsoft.com/office/2004/12/omml"><head><meta http-equiv="Content-Type" content="text/html; charset=windows-1252"><link rel="Edit-Tim
e-Data href="index_files/editdata.mso"><title>Keith Ross's Homepage</title><link
rel="themeData" href="index_files/themedata.thmx"><link rel="colorSchemeMapping" hr
ef="index_files/colorschememapping.xml"><style>
<!--
p.MsoNormal, li.MsoNormal
<mso-style-parent:""; margin-bottom:.0001pt;
font-size:12.0pt;
font-fam
```

投影片 38

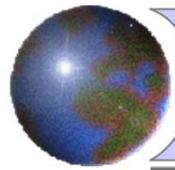
Hung3

Windows 7使用telnet:

(1)控制台 (2)程式集 (3)解除安裝程式 (4)點選左側"開啟or關閉Windows功能" (5)對"Telnet用戶端"打勾

==> "命令提示字元"視窗程式即可輸入 "telnet" 指令。

Ruo-Wei Hung, 2013/8/29



2.2.4 使用者與伺服器的互動：cookies



■ Cookies

- ▶ HTTP伺服器 → 無狀態的 = 不用記錄使用者的存取資訊
- ▶ 網頁伺服器如何辨識不同使用者？→ HTTP的 *cookie*
 - RFC 2965 – 讓網站能夠追蹤使用者，以限制使用者的存取
- ▶ Cookie 技術的四項元素：
 - HTTP 回應訊息的 cookie 標頭列
 - HTTP 請求訊息的 cookie 標頭列
 - 保留在使用者終端系統上的一份 cookie 檔案，並且由使用者瀏覽器管理
 - Web 網站的後端資料庫 (back-end database)



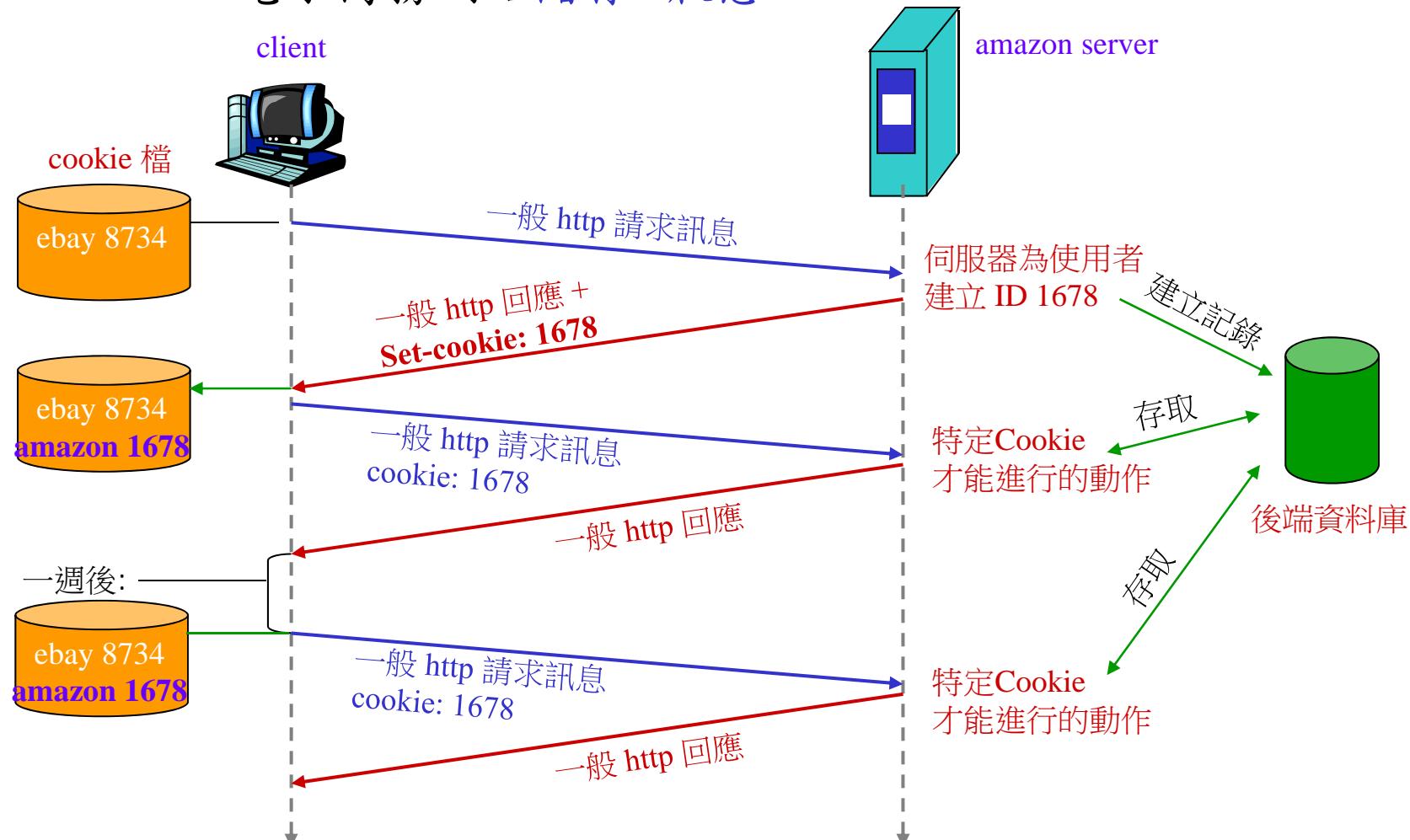
■ Cookie 範例：

- ▶ Royce 都是使用同一台PC來存取網際網路
- ▶ 第一次拜訪某個電子商務網站
- ▶ 當初始的HTTP請求到達網站時，電子商務網站會產生：
 - 唯一的識別號碼
 - 後端資料庫上為此識別碼建立一筆紀錄



■ Cookie 範例：(Cont.)

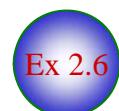
► 電子商務網站儲存“狀態”



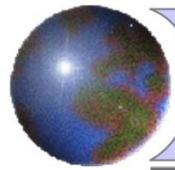


■ cookies 能做什麼？

- ▶ 授權
- ▶ 購物車
- ▶ 推薦
- ▶ 使用者會談狀態 (web e-mail)



Ex 2.6



2.2.5 網頁快取：代理伺服器(proxy)



■ 目標

- ▶ 滿足用戶端的請求，且不牽涉到來源伺服器

■ 設定/使用 proxy

- ▶ 使用者設定瀏覽器：Web 經由快取伺服器存取
- ▶ 瀏覽器傳送所有的 HTTP 請求到快取伺服器
 - 在快取伺服器中的物件：快取伺服器會回傳物件
 - 否則，快取伺服器會向來源伺服器請求物件，接著將物件回傳給用戶端

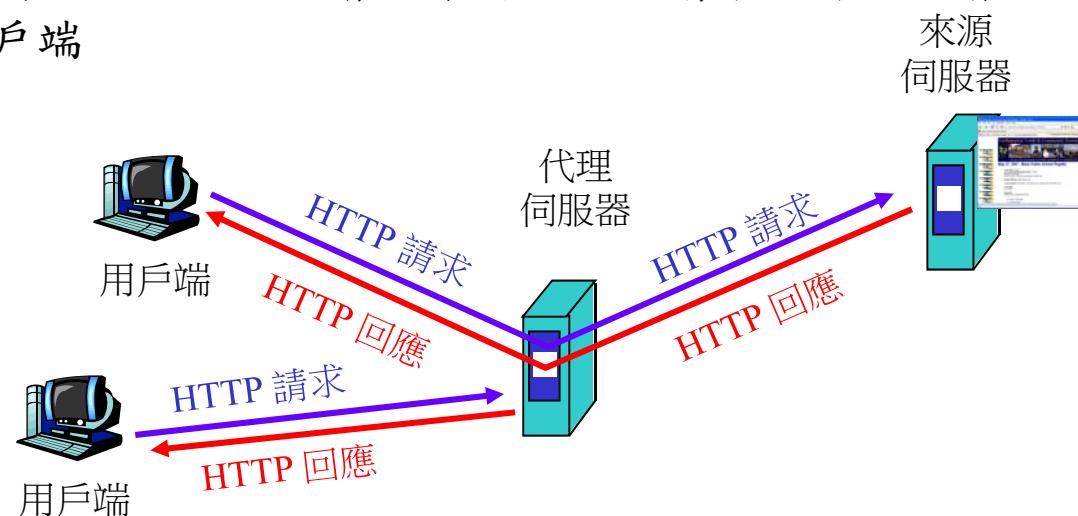


圖2.11 透過proxy請求物件



■ 網頁快取 (Cont.)

- ▶ 快取伺服器同時扮演了用戶端和伺服端的角色
- ▶ 通常ISP會安裝快取伺服器 (大學、公司、家用型 ISP)
- ▶ 為什麼要使用 Web 快取?
 - 降低用戶端發出請求後的回應時間
 - 減少機構連結到網際網路的流量
 - 具有高密度快取伺服器的網際網路可使得「差的」節目提供者能夠有效率地提供節目 (而 P2P 分享也是一樣)





■ 網頁的範例

▶ 假設

- 平均物件大小 = 1M bits
- 機構的瀏覽器到來源伺服器的平均請求速率 = 15 Objs/sec
- 從機構的路由器到任何來源伺服器再回來的延遲時間 = 2 sec

▶ 結果

- LAN的利用率 (流量強度, see 圖1.18)
$$= \frac{(15 \text{ Objs/sec}) \cdot (1\text{M bits/Obj})}{(100 \text{ Mbps})}$$
$$= 15\%$$
- 存取連結的利用率 (流量強度)
$$= \frac{(15 \text{ Objs/sec}) \cdot (1\text{Mbits/Obj})}{(15 \text{ Mbps})}$$
$$= 100\%$$
- 全部的延遲 =
網際網路延遲 + 存取延遲 + LAN 延遲
$$= 2 \text{ 秒} + \text{幾分鐘} + \text{幾毫秒}$$

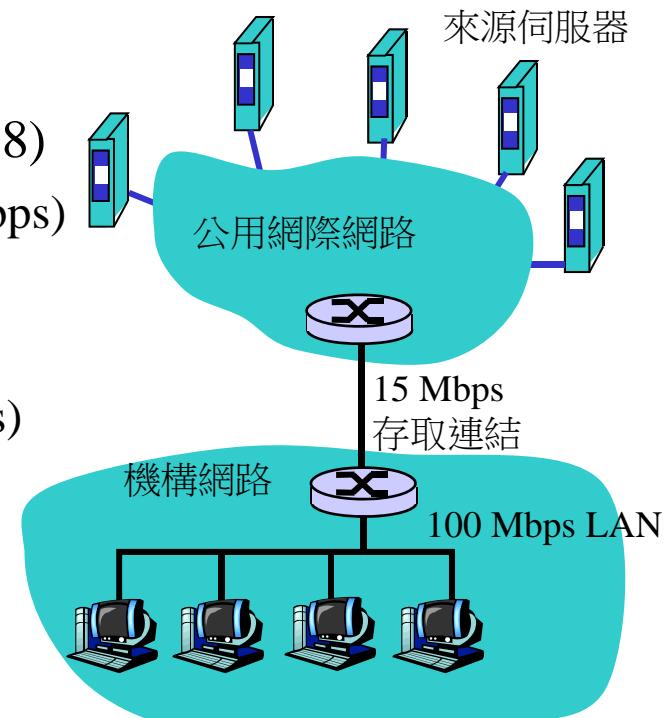
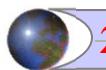


圖2.12 機構網路與網際網路間的頻頸





■ 網頁的範例 (Cont.)

► 可能的解決方案

- 增加存取連結的頻寬，比如說 100 Mbps

- 結果

- LAN的利用率 = 15%
- 存取連結的利用率 = 15%
- 全部的延遲
 - = 網際網路延遲 + 存取延遲 + LAN 延遲
 - = 2 秒 + 數毫秒 + 數毫秒
- 升級通常是相當昂貴的

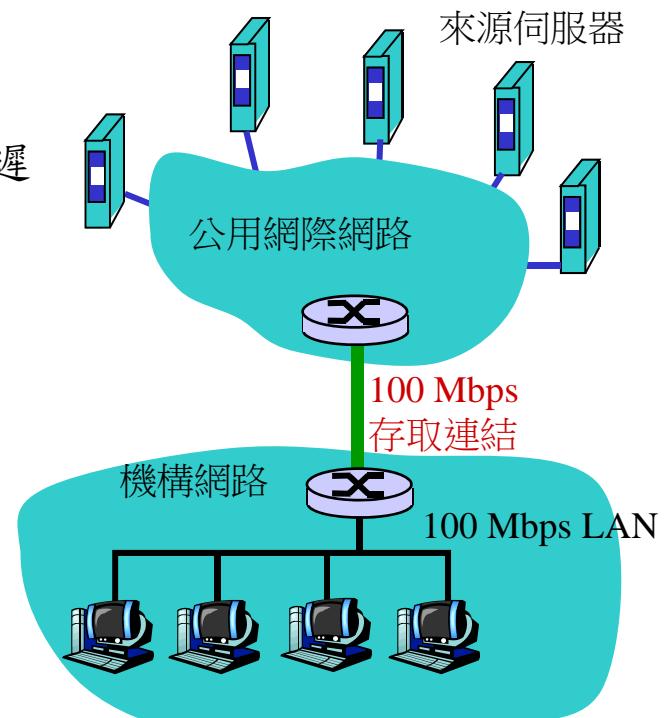


圖2.12 機構網路與網際網路間的頻頸





■ 網頁的範例 (Cont.)

► 可能的解決方案 (Cont.)

- 安裝 快取伺服器 (假設命中率為 0.4)

- 結果

- 40% 的請求會被立即滿足
- 60% 的請求被來源伺服器滿足
- 存取連結的利用率減為 60%，
變成幾乎可以忽略的延遲
(比如說 10 毫秒 = 0.01 秒)
- 總計的平均延遲
 $= \underline{\text{網際網路延遲 + 存取延遲 + LAN 延遲}}$
 $= \underline{0.6 * (2.01) \text{ 秒}} + 0.4 * \text{毫秒} < 1.4 \text{ 秒}$

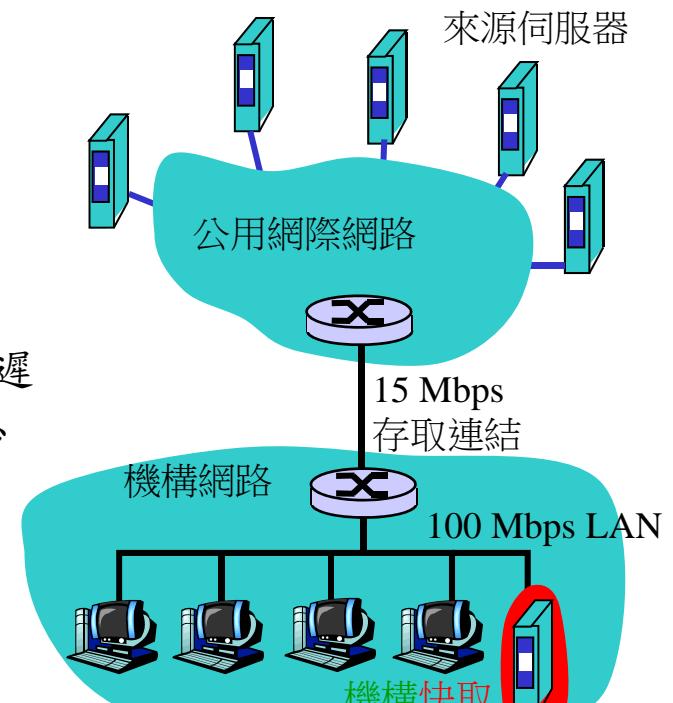
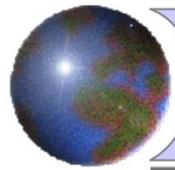


圖2.12 機構網路與網際網路間的頻頸

Ex 2.8





第二章 導覽流程



2.1 網路應用的原理

2.2 Web 和 HTTP

2.3 FTP

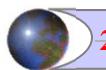
2.4 電子郵件 – SMTP、POP3、IMAP

2.5 DNS

2.6 點對點檔案分享

2.7 使用TCP的Socket程式設計

2.8 使用UDP的Socket程式設計



■ FTP (File Transfer Protocol)

- ▶ 檔案傳輸協定
- ▶ 傳輸檔案 到/從 遠端主機 (上傳/下載)
- ▶ 用戶端/伺服端 模型
 - 用戶端: 開始傳輸的那一邊 (不管是 到/從 遠端)
 - 伺服端: 遠端主機
- ▶ ftp : RFC 959
- ▶ ftp 伺服器 : 埠號 21

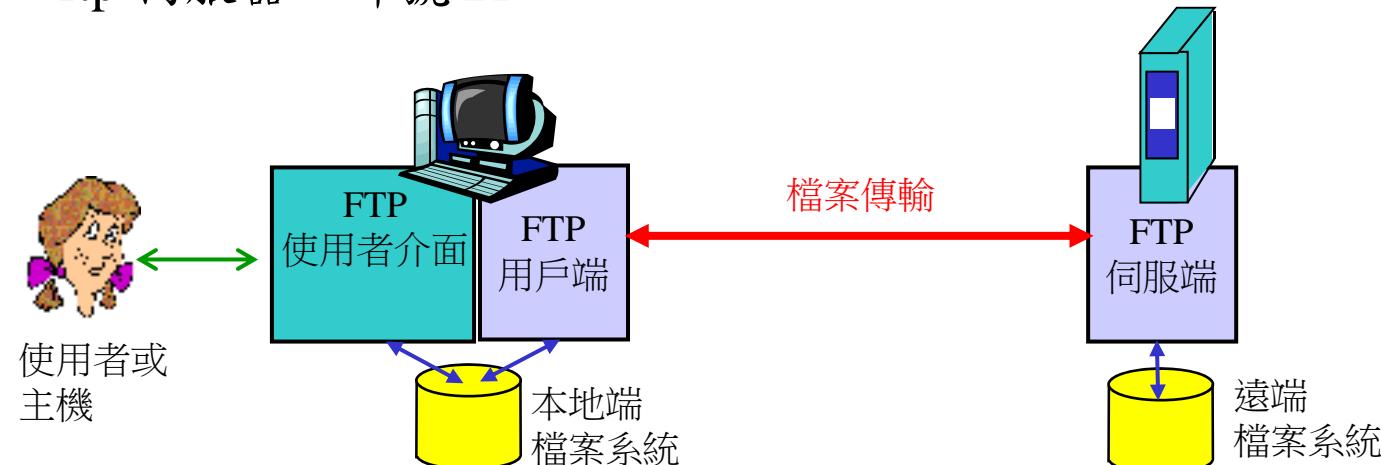
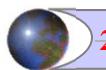


圖2.14 FTP 會在本地端與遠端檔案系統間搬移檔案



■ FTP 運作原理

► 分離的控制、資料連結

- FTP 用戶端以埠號21聯繫 FTP伺服端，指定 TCP 為傳輸協定
- 用戶端經由控制連線取得授權
- 用互端經由控制連線傳送命令來瀏覽遠端目錄
- 當伺服器收到檔案傳輸的命令時，伺服器會為用戶端開啟兩個 TCP的資料連線
- 在傳輸一個檔案之後，伺服器會關閉該連線
- 伺服端開啟第二個TCP資料連線以傳送另一個檔案
- 控制連線： “頻帶外” (out-of-band)
- FTP 伺服器會維護 “狀態”：目前的目錄、之前的授權

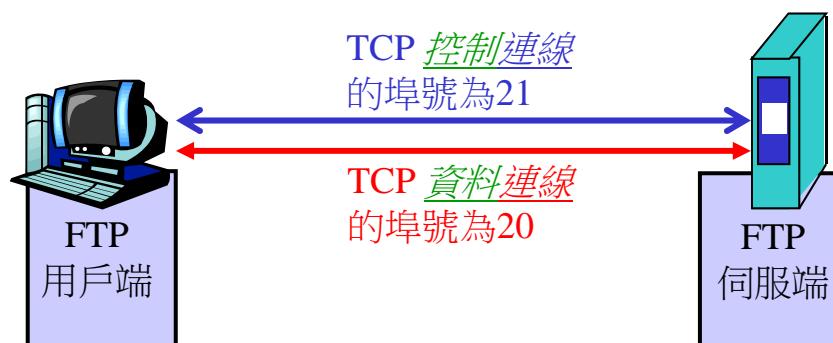


圖2.15 FTP 的控制與資料連線





■ FTP 指令、回覆

▶ 範例命令

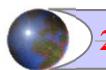
(經由控制連結頻道，以 ASCII 文字傳送)

- **USER** username
- **PASS** password
- **LIST** 回傳目前目錄下的檔案清單
- **RETR** filename 取得 (意指get) 檔案
- **STOR** filename 將檔案儲存 (意指put) 到遠端主機

▶ 範例回覆碼

(狀態碼和資料 (類似 HTTP))

- 331 使用者名稱ok，需要密碼
- 125 確定資料連結已經打開；轉換開始
- 425 資料連結無法打開
- 452 錯誤訊息寫入檔案



■ FTP 指令、回覆 (Cont.)

命令提示字元 - ftp home.cyut.edu.tw

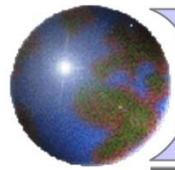
```
Microsoft Windows XP [版本 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Administrator>ftp home.cyut.edu.tw
Connected to home.cyut.edu.tw.
220 (vsFTPd 2.0.5)
User (home.cyut.edu.tw:(none)): s9427001
331 Please specify the password.
Password:
230 Login successful.
ftp> help
Commands may be abbreviated. Commands are:

!          delete      literal      prompt      send
?          debug       ls           put         status
append     dir        mdelete    pwd         trace
ascii      disconnect  mdir        quit        type
bell       get        mget        quote      user
binary     glob       mkdir      recv        verbose
bye        hash       mls         remotehelp
cd         help       mput      rename
close     lcd        open       rmdir
```

>ftp ftp.isu.edu.tw
>帳號 = ftp

Ex 2.9



第二章 導覽流程



2.1 網路應用的原理

2.2 Web 和 HTTP

2.3 FTP

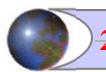
2.4 電子郵件 – SMTP、POP3、IMAP

2.5 DNS

2.6 點對點應用

2.7 使用TCP的Socket程式設計

2.8 使用UDP的Socket程式設計



■ 電子郵件系統的三個主要元件

- ▶ 使用者代理程式 (user agents)
- ▶ 郵件伺服器 (mail servers)
- ▶ 簡單郵件傳輸協定：SMTP (Simple Mail Transfer Protocol)

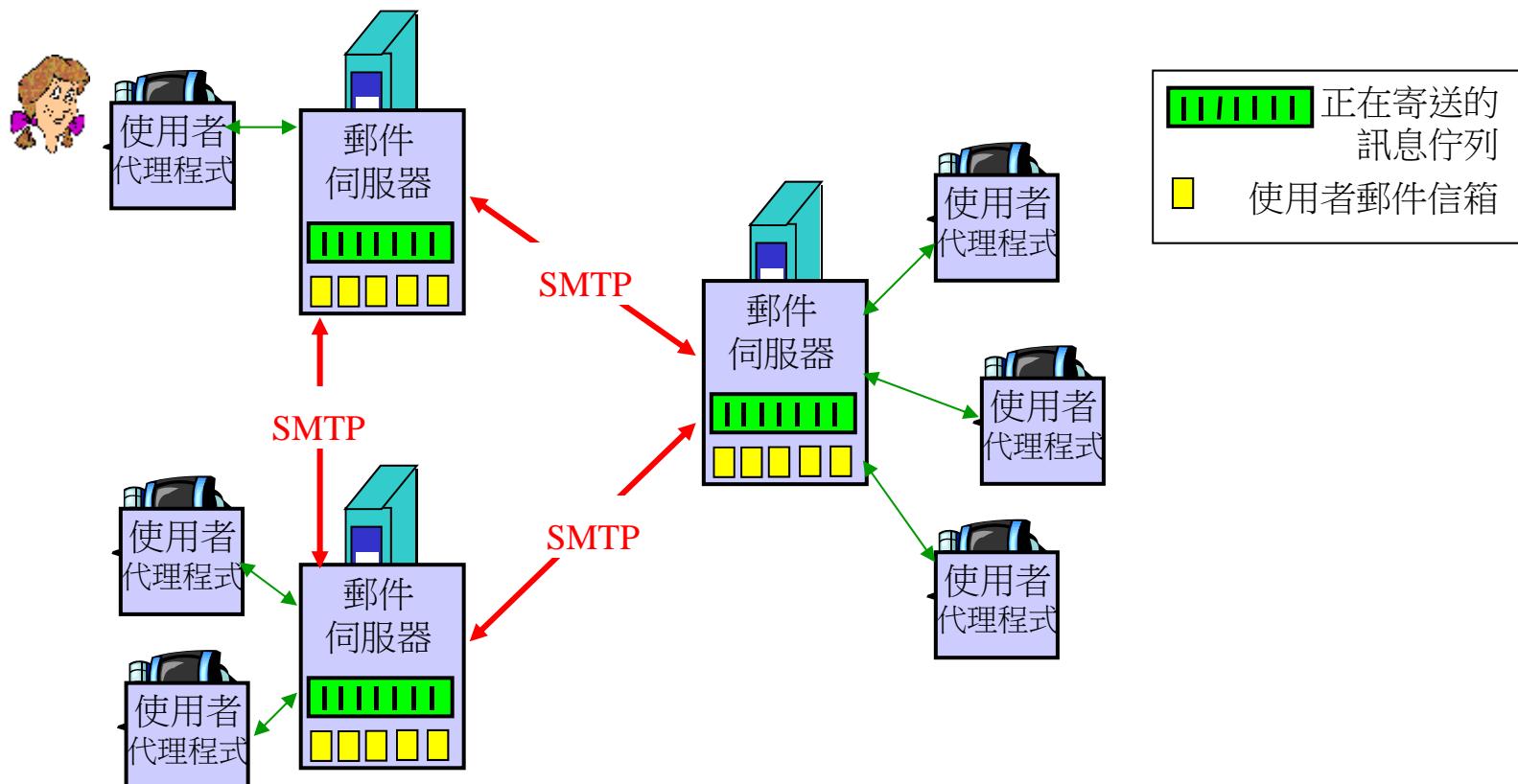


圖2.16 網際網路電子郵件系統的三個主要元件

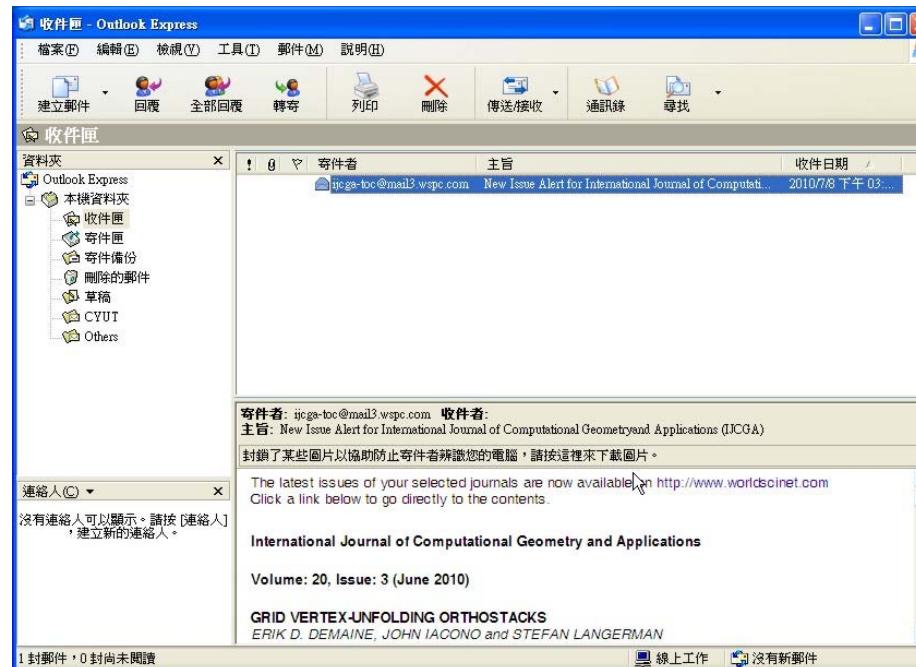


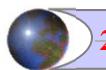


■ 電子郵件三個主要元件 (Cont.)

► 使用者代理程式 (user agents)

- 又稱 “郵件檢示器” (mail reader)
- 撰寫、編輯、讀取郵件訊息
- 例如：Eudora、Outlook、elm、Netscape Messenger
- 寄送、收取伺服器上的訊息

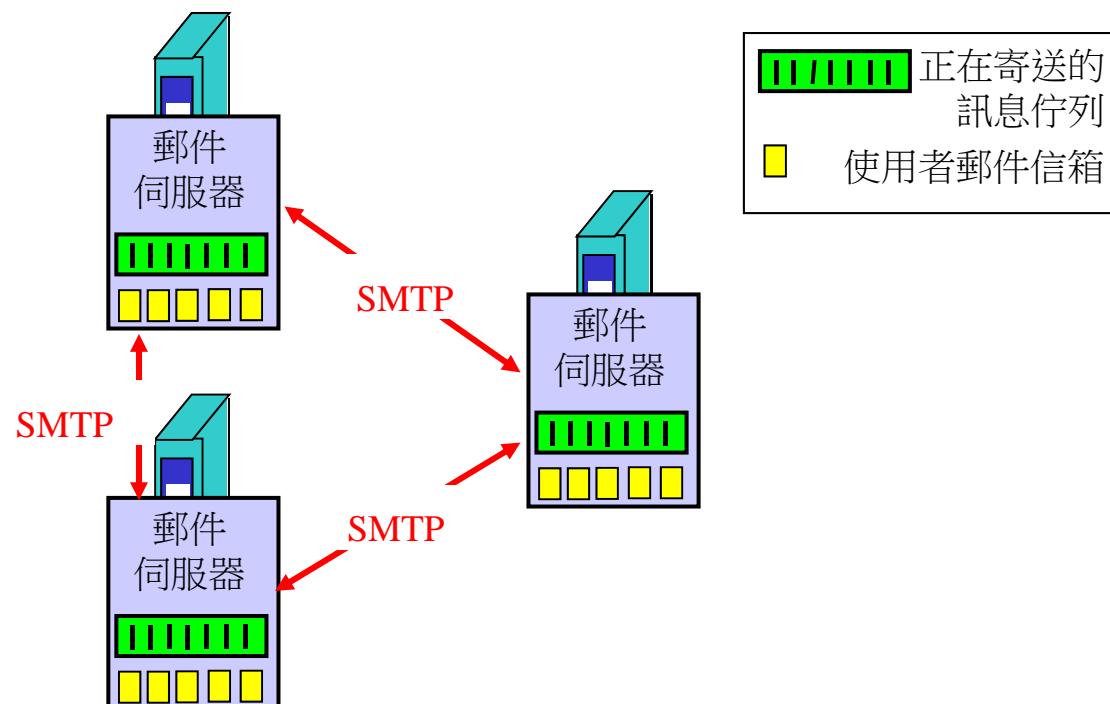




■ 電子郵件三個主要元件 (Cont.)

► 郵件伺服器 (mail servers)

- 郵件信箱(mail box) 保存了使用者收到的信件
- 訊息佇列(message queue) 內含要出去的(將要寄出的)郵件訊息
- SMTP 協定 在郵件伺服器之間傳送電子郵件訊息
- 用戶端：傳送電子郵件的伺服器
- 伺服端：接收電子郵件的伺服器





■ 電子郵件三個主要元件 (Cont.)

► 簡單郵件傳輸協定：SMTP (Simple Mail Transfer Protocol)

- 定義在 RFC 2821
- 使用 TCP，埠號 25，從用戶端到伺服端可靠地傳輸電子郵件訊息
- 直接傳輸：從傳送伺服器到接收伺服器（不會使用中介伺服器來傳送郵件）
- 傳輸的三個階段
 - 交握 (handshake, 自我介紹)
 - 傳送訊息
 - 關閉
- 指令/回應 的互動
 - 指令：ASCII 文字
 - 回應：狀態碼和資訊
- 訊息必須是 7 個位元的 ASCII 碼





■ 傳送電子郵件範例

► 情境 – Alice 傳送電子郵件給 Bob

► 過程 –

- ① Alice 執行使用者代理程式，編寫一份訊息寄給 bob@cyut.edu.tw
- ② Alice 的使用者代理程式傳送訊息到她的郵件伺服器，此訊息被放置在訊息佇列中
- ③ Client side of SMTP用戶端開啟 TCP連線，連結到Bob的郵件伺服器
- ④ SMTP 透過TCP連線傳送 Alice 的訊息
- ⑤ Bob 的郵件伺服器將訊息放置在Bob的郵件信箱中
- ⑥ Bob 執行他的使用者代理器來閱讀訊息

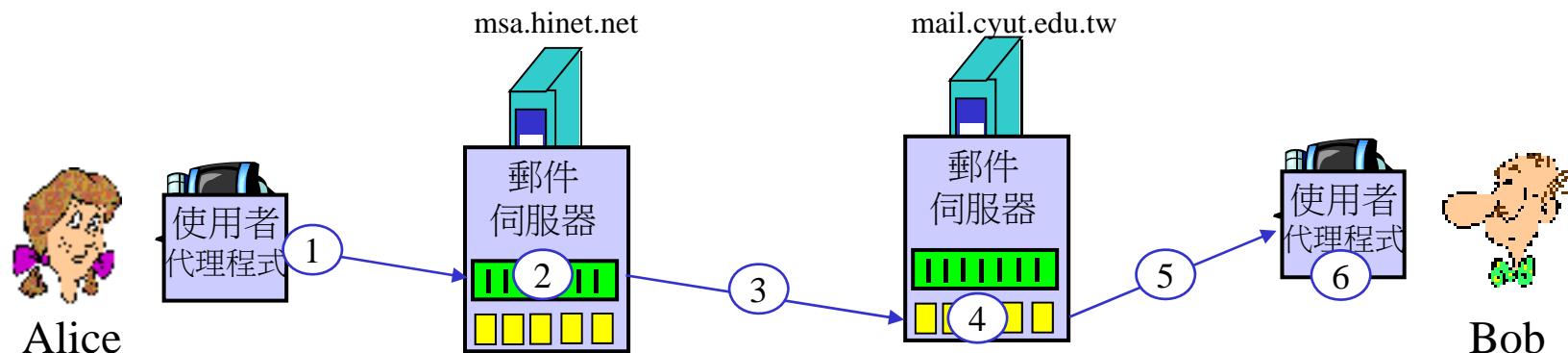


圖2.17 Alice 傳送一封電子郵件給 Bob 的過程



■ SMTP 互動範例

► SMTP客戶端(Client)與SMTP伺服端(Server)之間訊息交換記錄

S: 220 cyut.edu.tw

C: HELO msa.hinet.net

S: 250 Hello msa.hinet.net, pleased to meet you

C: MAIL FROM: <alice@msa.hinet.net >

S: 250 alice@msa.hinet.net... Sender ok

C: RCPT TO: <bob@cyut.edu.tw >

S: 250 bob@cyut.edu.tw ... Recipient ok

C: DATA

S: 354 Enter mail, end with "." on a line by itself

C: Do you like ketchup?

C: How about pickles?

C: .

S: 250 Message accepted for delivery

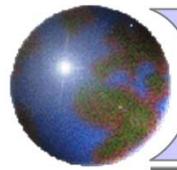
C: QUIT

S: 221 cyut.edu.tw closing connection



■ SMTP 結語

- ▶ SMTP 使用 **永久性連線**
- ▶ SMTP 要求訊息 (標頭及主體) 必須是7位元的**ASCII**格式
- ▶ SMTP 伺服器用 CRLF.CRLF 來決定訊息的尾端
- ▶ 與 HTTP 的比較：
 - HTTP：取回 (pull)
 - SMTP：送出 (push)
 - 都有 ASCII 指令/回應的互動、狀態碼
 - HTTP：每個物件都封裝在它自己的回應訊息中
 - SMTP：多個物件以一個多重訊息來傳送



2.4.3 郵件訊息格式與 MIME



■ 郵件訊息格式

- ▶ SMTP：交換郵件訊息的協定
- ▶ RFC 822 – 文字訊息的格式標準：

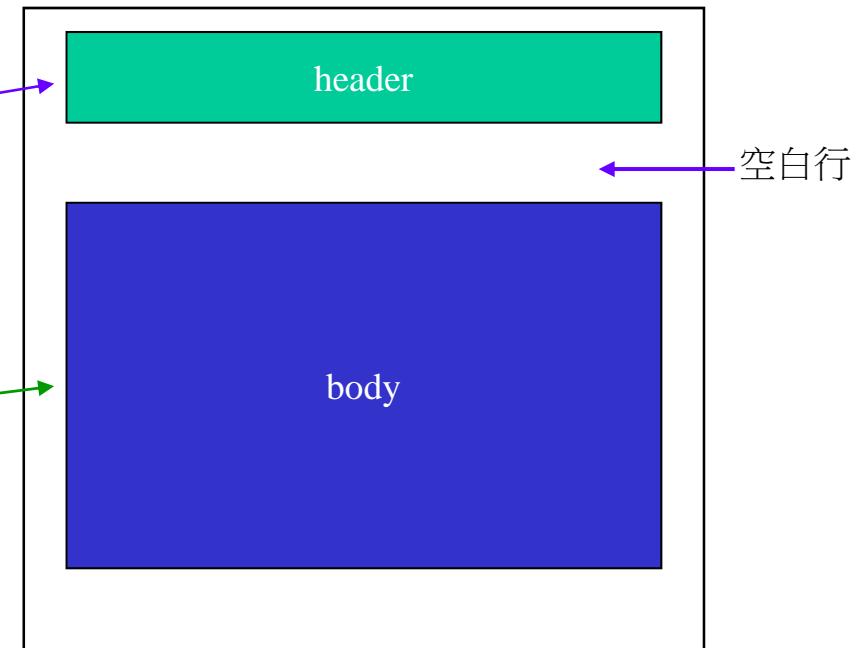
- 標頭檔連結，例如：

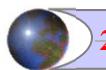
- To :
 - From :
 - Subject :

⇒ 與 SMTP 命令 不同！

- 本體

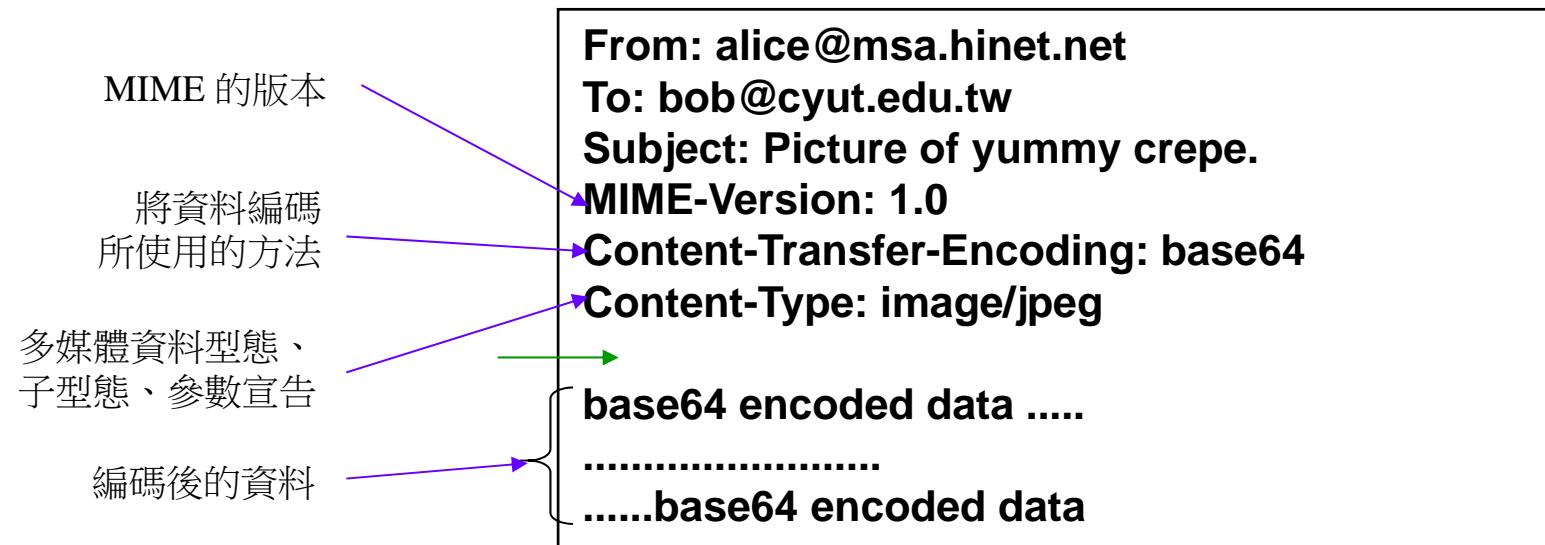
- “訊息” ，僅 ASCII 字元

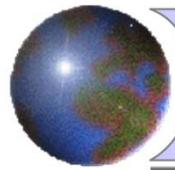




■ 郵件訊息格式 – 多媒體延伸

- ▶ **MIME**: multipurpose internet mail extensions, multimedia mail extension, RFC 2045, 2056
- ▶ 訊息標頭中額外的文字列宣告了 MIME 的content type





2.4.4 郵件存取協定 (Mail access protocol)

■ 郵件傳輸系統協定

- ▶ SMTP：傳送/儲存到接收端的郵件伺服器
- ▶ 郵件存取協定：從伺服器取得電子郵件
 - POP : Post Office Protocol [RFC 1939]
 - 授權階段 (代理器 \leftrightarrow 伺服器) 與存取程序階段
 - IMAP : Internet Mail Access Protocol [RFC 1730]
 - 更多功能 (更複雜)
 - 在伺服器上操作儲存的訊息 (Webmail)
 - HTTP: Hotmail 、 Yahoo! Mail 等等

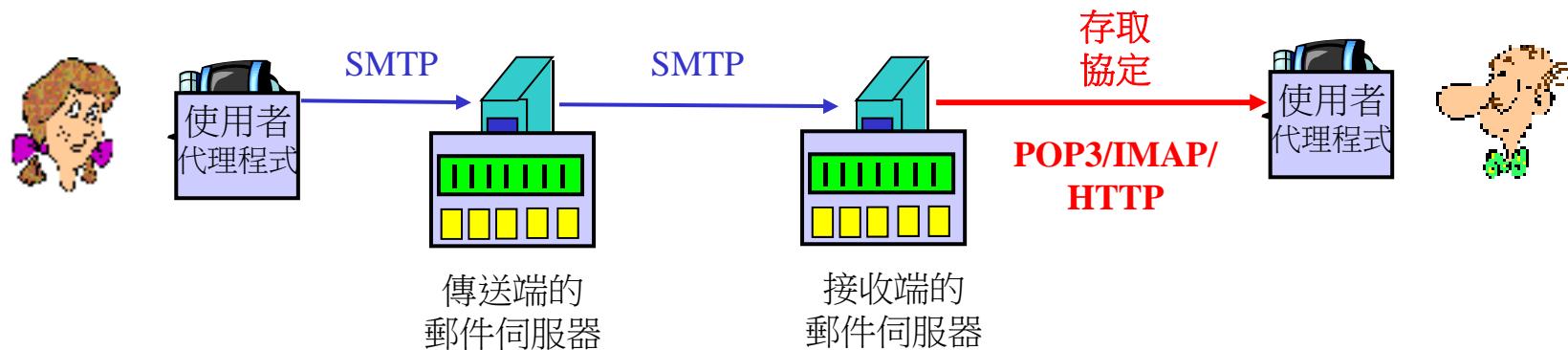


圖2.18 電子郵件協定與它們的通訊實體

■ 郵件存取協定

► *POP3* : Post Office Protocol ver. 3 [RFC 1939]

- 授權階段

- 用戶端指令 :
 - » user : 宣告使用者名稱
 - » pass : 密碼
- 伺服器回應
 - » +OK
 - » -ERR

S: +OK POP3 server ready
 C: user bob
 S: +OK
 C: pass hungry
 S: +OK user successfully logged on

- 存取程序階段 (用戶端)

- list : 將訊息編號列表
- retr : 以編號收取訊息
- dele : 刪除
- quit

C: list
 S: 1 498
 S: 2 912
 S: .
 C: retr 1
 S: <message 1 contents>
 S: .
 C: dele 1
 C: retr 2
 S: <message 1 contents>
 S: .
 C: dele 2
 C: quit
 S: +OK POP3 server signing off



■ 郵件存取協定 (Cont.)

► POP3 (Cont.)

- “下載並刪除” 模式 – 假如Bob更換了用戶端，他將無法重新讀取郵件 (下載郵件並從mail server上刪除此郵件)
- “下載並保留” 模式 – 在不同的用戶端保留副本 (下載郵件但不從mail server上刪除此郵件)
- 在會談期間，POP3為無狀態的 (mail server不會保有使用者的狀態資訊)
- 不允許使用者在mail server上建立資料夾

► IMAP

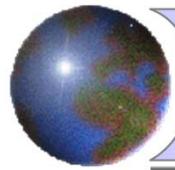
- 較POP3複雜及功能更強
- 允許使用者在mail server上建立郵件資料夾
- 使用者可於所建立的郵件資料夾間搬移電子郵件
- 在會談期間，IMAP server會保有使用者的狀態資訊(eg.郵件資料夾名稱/哪筆訊息屬於哪個資料夾)



■ 郵件存取協定 (Cont.)

► HTTP

○ Web mail



第二章 導覽流程



2.1 網路應用的原理

2.2 Web 和 HTTP

2.3 FTP

2.4 電子郵件 – SMTP、POP3、IMAP

2.5 DNS

2.6 點對點應用

2.7 使用TCP的Socket程式設計

2.8 使用UDP的Socket程式設計



■ DNS (Domain Name System)：網域(領域)名稱系統

► What ?

- 人類：擁有許多識別碼來辨識身份
 - 身份證號碼、名字、護照號碼
- 網際網路主機、路由器：
 - *IP 位址* (32 位元, IPv4) – 使用位址資料段 → 機器所使用的
 - *領域名稱* – 例如，www.cyut.edu.tw → 人類所使用的

Q: IP位址以及領域名稱之間的對應？(eg. www.cyut.edu.tw
→ 163.17.1.15?)

A: 領域名稱系統 – 轉換領域名稱成IP位址的系統



■ DNS – 網域名稱系統 (Cont.)

► 基礎

Hung4

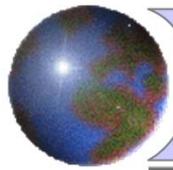
- 分散式的資料庫 (*distributed database*) – 以許多階層式的DNS伺服器(name servers) 實作
- 應用層協定 (*application layer*) – 允許主機、路由器、DNS伺服器連接，以解析(*resolve*)網域名稱 (網域名稱→IP位址 的轉換)。DNS是讓主機可以查詢此分散式資料庫的應用層協定

投影片 69

Hung4

分散式: 非由一台電腦完成解譯, 而是由多台DNS伺服器共同完成一解譯工作。

Ruo-Wei Hung, 2013/8/29



2.5.1 DNS 服務



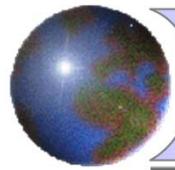
■ DNS 所提供的服務

▶ 基本服務

- 主機(網域)名稱到IP位址的轉換

▶ 額外服務

- 主機別名 (host aliasing)
 - 正規主機名稱 vs. 別名
 - 正規主機名稱 = www.cyut.edu.tw
 - 別名 = cyut university of technology
 - 應用程式可以呼叫 DNS，提供別名，以取得正規主機名稱/IP位址
- 郵件伺服器別名 (CYUT mail server: cyut.edu.tw = mail.cyut.edu.tw)
- 負載分配 (load distribution)
 - 複製的Web伺服器：一組IP位址對應到一個正規主機名稱
 - 忙碌的網站www.nhk.com → 98.129.126.138 / 98.129.126.139
 - 1st query www.nhk.com → 98.129.126.138
 - 2nd query www.nhk.com → 98.129.126.139



2.5.2 DNS 的運作



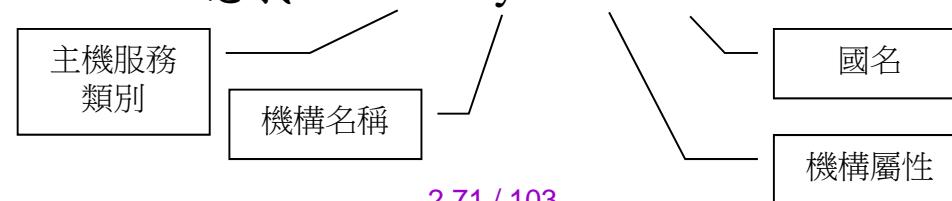
■ DNS 的運作

▶ 集中化的 DNS

- 只用一台 DNS 伺服器 – DNS table (兩欄位 = domain name, IP address)
 - 問題：
 - 單點故障
 - 大量的網路流量
 - 遠距離集中式資料庫
 - 維護
- ⇒集中化的 DNS 不具擴充性!

▶ 分散式、階層式 DNS 資料庫

- 目前網際網路所採用
- 沒有任何單一的DNS伺服器擁有網際網路上所有主機的對映資訊
- 解析網域名稱時，需透過多台DNS伺服器合作，才可完成
- domain name 意義 www.cyut.edu.tw





■ DNS 的運作 – 分散式、階層式 DNS 資料庫

► 階層式 DNS 伺服器類別

- 根(root)名稱伺服器
- 高層網域 DNS (TLD) 伺服器 (Top Level Domain)
- 官方DNS 伺服器

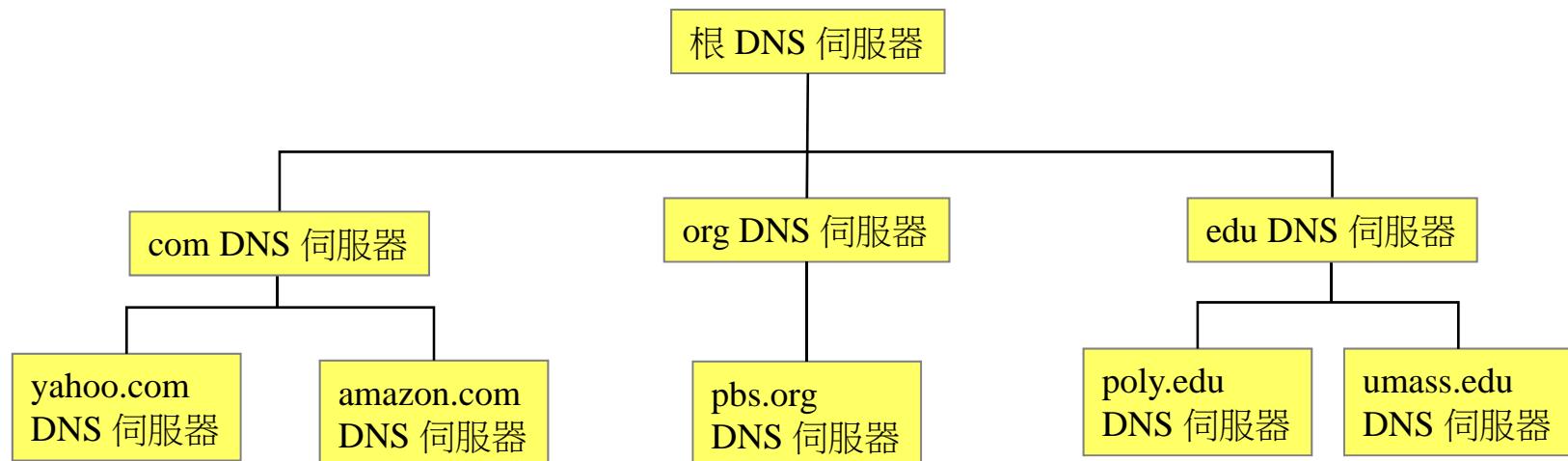
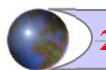


圖2.19 部份的DNS伺服器階層架構

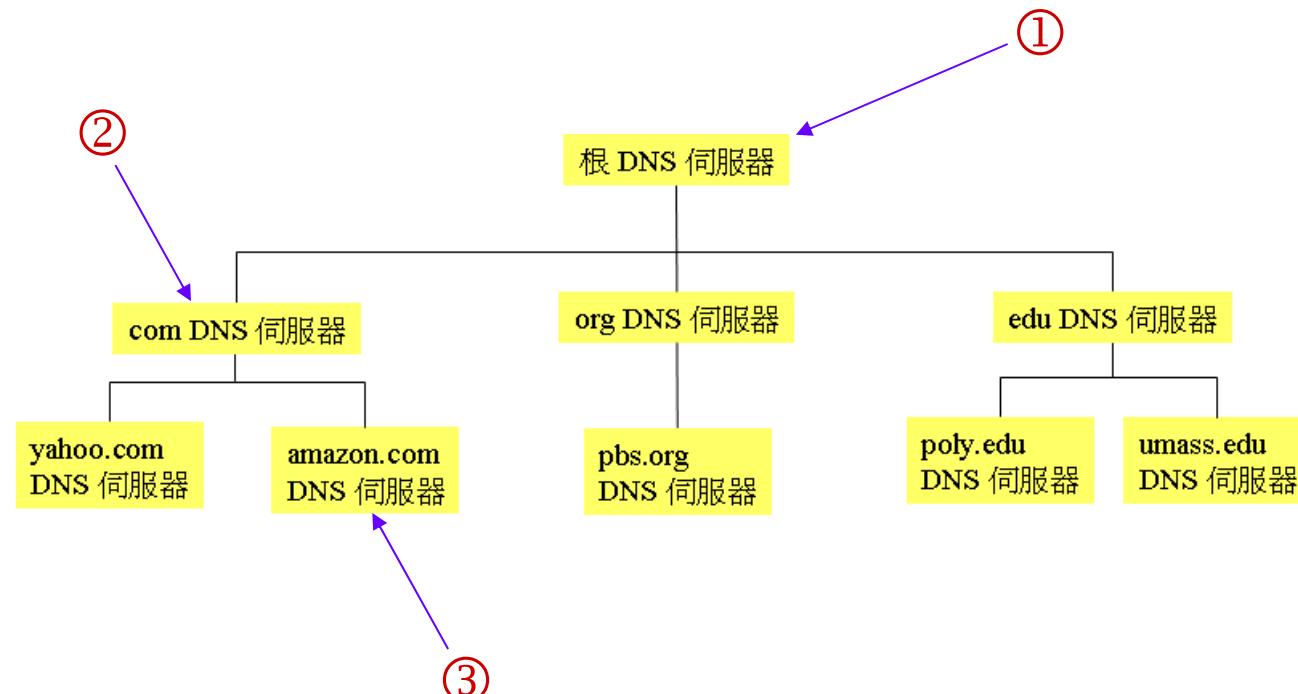




■ DNS 的運作 – 分散式、階層式 DNS 資料庫 (Cont.)

▶ 用戶端想要知道 www.amazon.com 的 IP 位址

- ① 用戶端向根伺服器查詢，尋找 com DNS 伺服器(IP位址)
- ② 用戶端向 com DNS 伺服器查詢，得到 amazon.com DNS 伺服器
- ③ 用戶端向 amazon.com DNS 伺服器查詢，得到 www.amazon.com 的 IP 位址





■ DNS 的運作 – 分散式、階層式 DNS 資料庫 (Cont.)

► DNS : 根名稱伺服器

- 全世界共有 13 個名稱伺服器

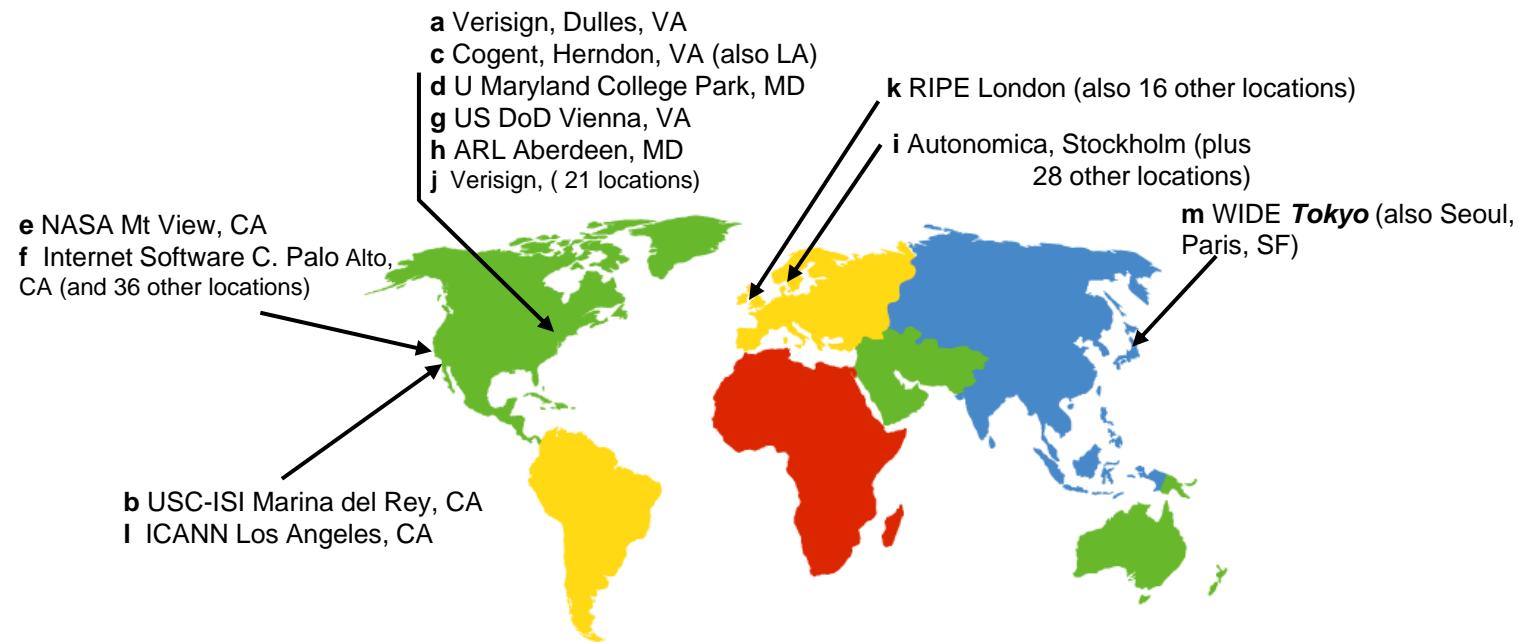
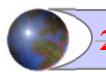


圖2.20 2009年的DNS根伺服器 (名稱, 組織, 地點)



■ DNS 的運作 – 分散式、階層式 DNS 資料庫 (Cont.)

▶ DNS : 高層網域 DNS (TLD) 伺服器

- 負責 com、org、net、edu、etc、以及全部的高層國家領域 uk、fr、ca、jp、cn、hk、tw
- Network solutions 公司負責維護 com TLD 的伺服器
- Educause 公司負責 edu TLD

▶ DNS : 官方 DNS 伺服器

- 組織/公司內的 DNS 伺服器，替組織內的伺服器提供管控的主機名稱到IP的對應 (例如，web 以及 mail)
- 可以是組織本身或服務業者來維護的



■ DNS 的運作 – 分散式、階層式 DNS 資料庫 (Cont.)

► DNS：區域名稱伺服器 (local DNS server)

- 不全然屬於階層架構中
- 每個 ISP (家用型 ISP、公司、大學) 都有一個
 - 也稱為 “預設名稱伺服器”
- 當主機做了一個DNS查詢，此查詢會被傳送到它的區域DNS伺服器
 - 如同代理伺服器(proxy)，將查詢傳入階層架構中



■ DNS 名稱解析範例

- ▶ 情境 – csie.cyut.edu.tw 的主機想要查詢 www.ccu.edu.tw 的IP位址
 - 循環式查詢 (iterative query)

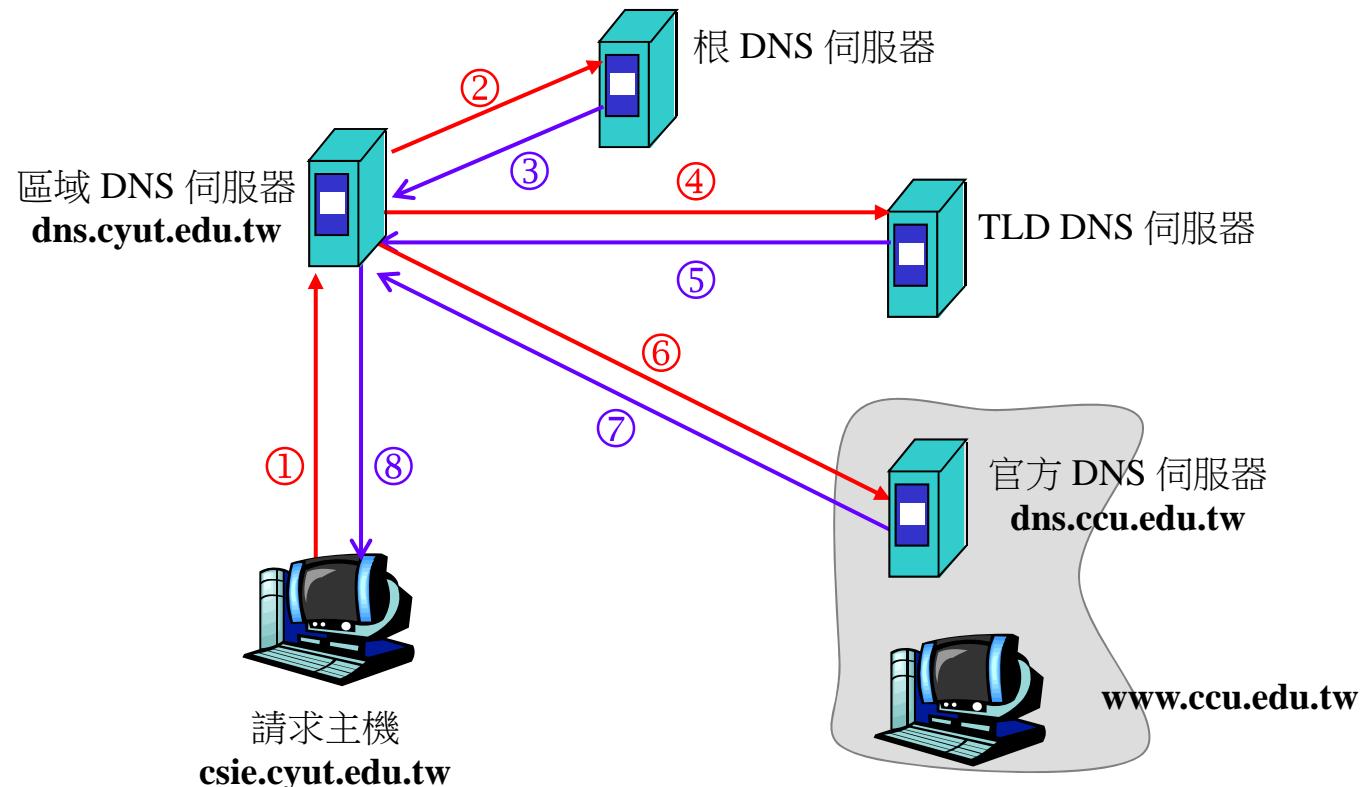
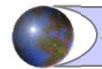


圖2.21 各種 DNS 伺服器的互動 (iterative query)



■ DNS 名稱解析範例 (Cont.)

► 情境 – csie.cyut.edu.tw 的主機想要查詢 www.ccu.edu.tw 的位址
(Cont.)

- 遞迴式查詢 (recursive query) – 加重被聯繫的名稱伺服器的名稱解析負擔

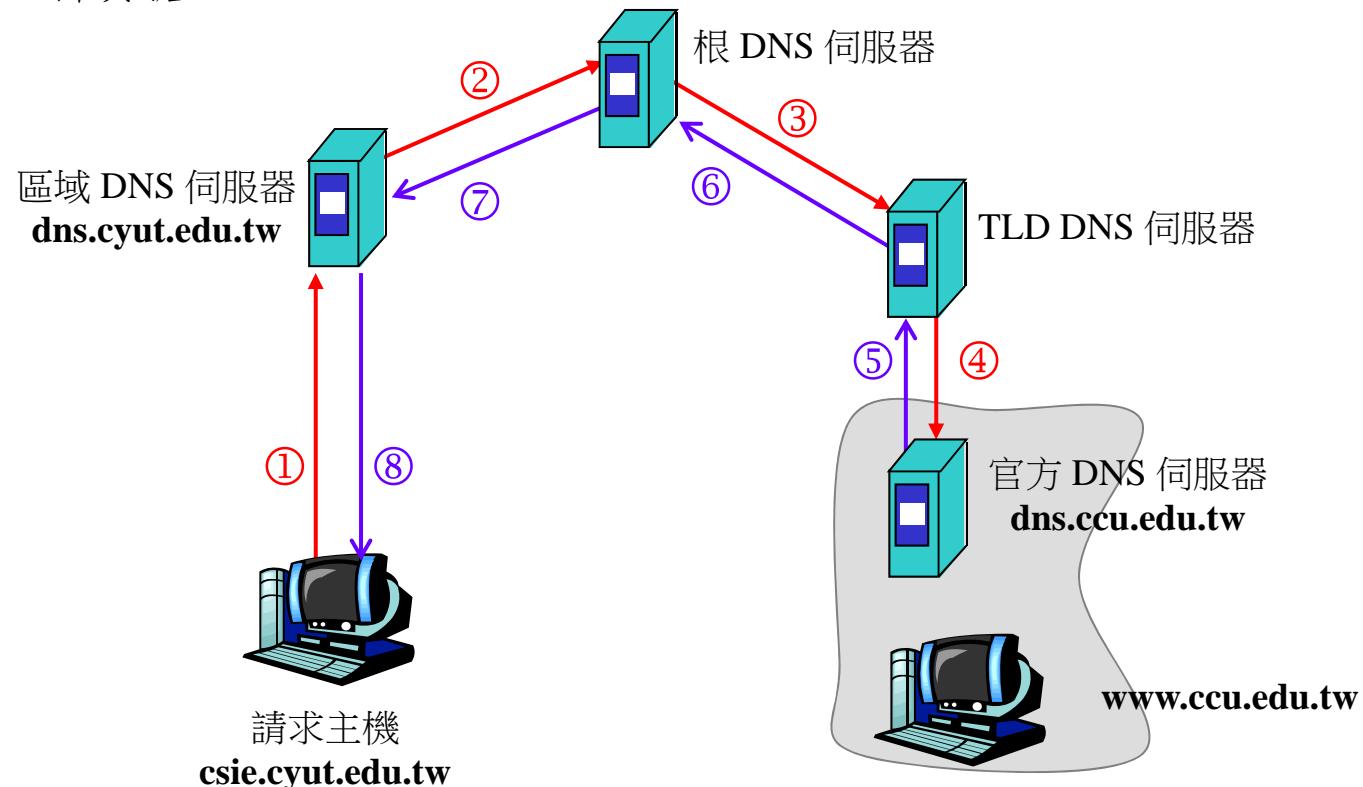


圖2.22 DNS伺服器的遞迴式查詢 (recursive query)





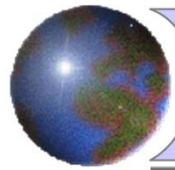
■ 在DNS中新增紀錄

- ▶ 情境 – 新成立的公司 “Royce Corp.”
 - 向 註冊服務商 註冊名稱 royce.com.tw (例如，HiNet)
 - 需要提供該服務商你的(主要及次要)官方DNS伺服器的名稱與IP位址
 - 服務商在 com TLD 伺服器中輸入兩筆 RR：
 - » (royce.com.tw, dns.royce.com.tw, NS)
 - » (dns.royce.com.tw, 163.13.15.2, A)
 - 建立 官方DNS伺服器 中放入 Type A 的 www.royce.com.tw 紀錄 以 及 Type MX 的 royce.com 紀錄
Hung5
⇒人們要如何得到你的網站的IP位址呢？

投影片 79

Hung5

MX = Message eXchange
Mail transfer 必須為之(設定MX)。
Ruo-Wei Hung, 2013/8/29



第二章 導覽流程



2.1 網路應用的原理

2.2 Web 和 HTTP

2.3 FTP

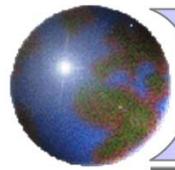
2.4 電子郵件 – SMTP、POP3、IMAP

2.5 DNS

2.6 點對點應用

2.7 使用TCP的Socket程式設計

2.8 使用UDP的Socket程式設計



2.6.1 點對點檔案分享 (P2P File Sharing)

Hung6

■ 點對點檔案分享 (peer-to-peer file sharing)

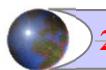
► 範例情境

- Alice 在她的筆記型電腦上執行對等式的用戶端應用程式
- 間歇地連接到網際網路，每次連接都得到一個新的IP位址(Dynamic)
- 尋找 “Hey Jude”檔案
- 應用程式會顯示其它擁有Hey Jude 的副本的對等點
- Alice 選擇了其中一個對等點 – Bob
- 檔案由Bob的電腦複製到Alice的筆記型電腦：HTTP
- 當Alice從Bob下載檔案的同時，可能有其他的使用者從Alice下載
- Alice的對等點同時是Web用戶端以及暫時的Web伺服端
⇒所有的對等點都是伺服器 = **高擴充性!** (圖2.24)

投影片 81

Hung6

Peer: 同儕/對等點。
Ruo-Wei Hung, 2013/8/29



■ 點對點檔案分享 (Cont.)

► 範例情境 (Cont.)

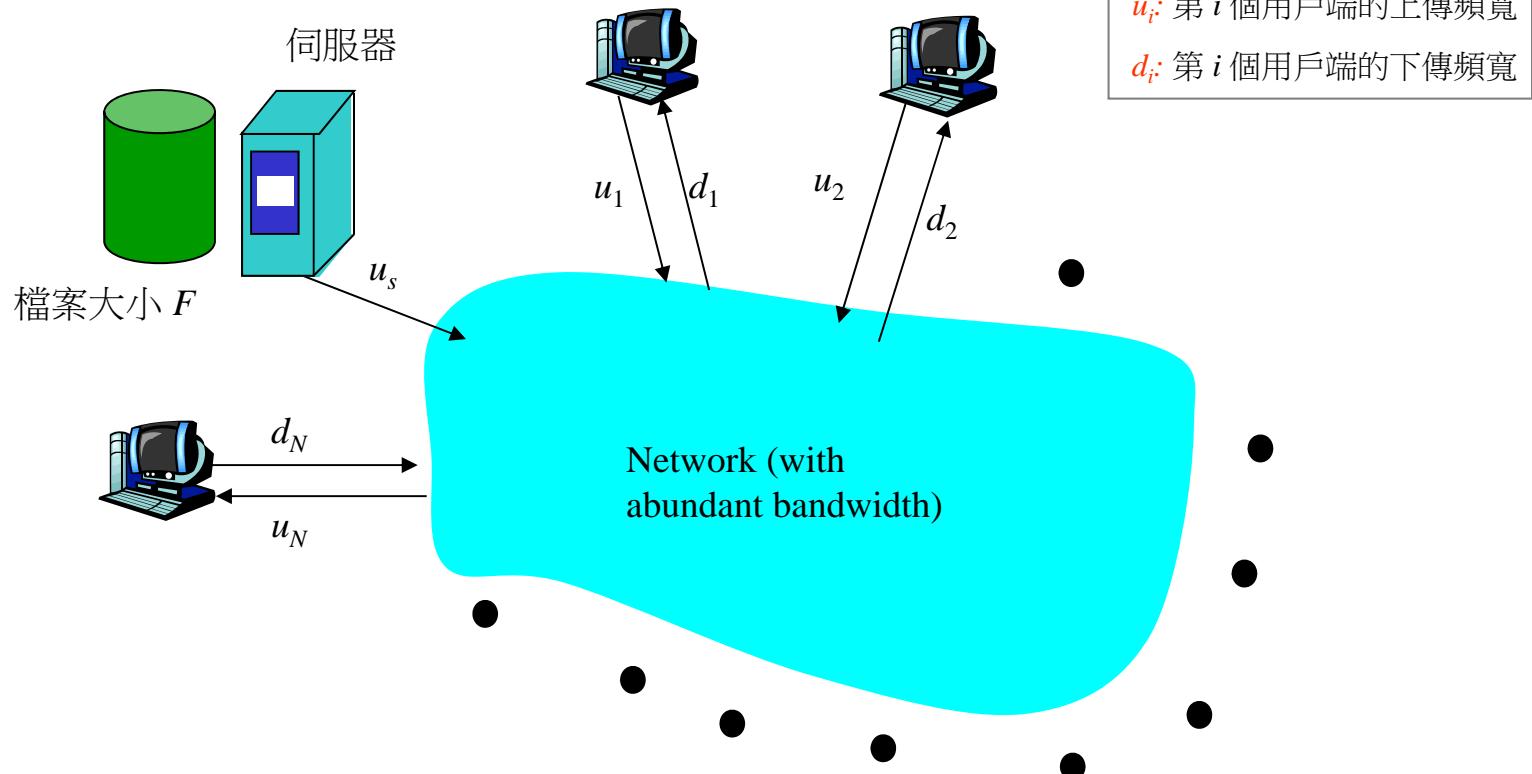


圖2.24 P2P 檔案散佈



■ 集中式P2P索引

① 當對等點連結時，它會告知中央(目錄)伺服器：

- IP 位址
- (分享的)內容

② Alice 尋找 “Hey Jude” 檔案

③ Alice 向 Bob 請求檔案
(file transfer)

► 問題

- 單點(中央伺服器)故障
- 效能瓶頸(中央伺服器)
- 版權侵犯：訴訟“目標”是非常明顯的
- ⇒ 檔案傳輸是非集中式的，
但內容尋找是高度的集中式

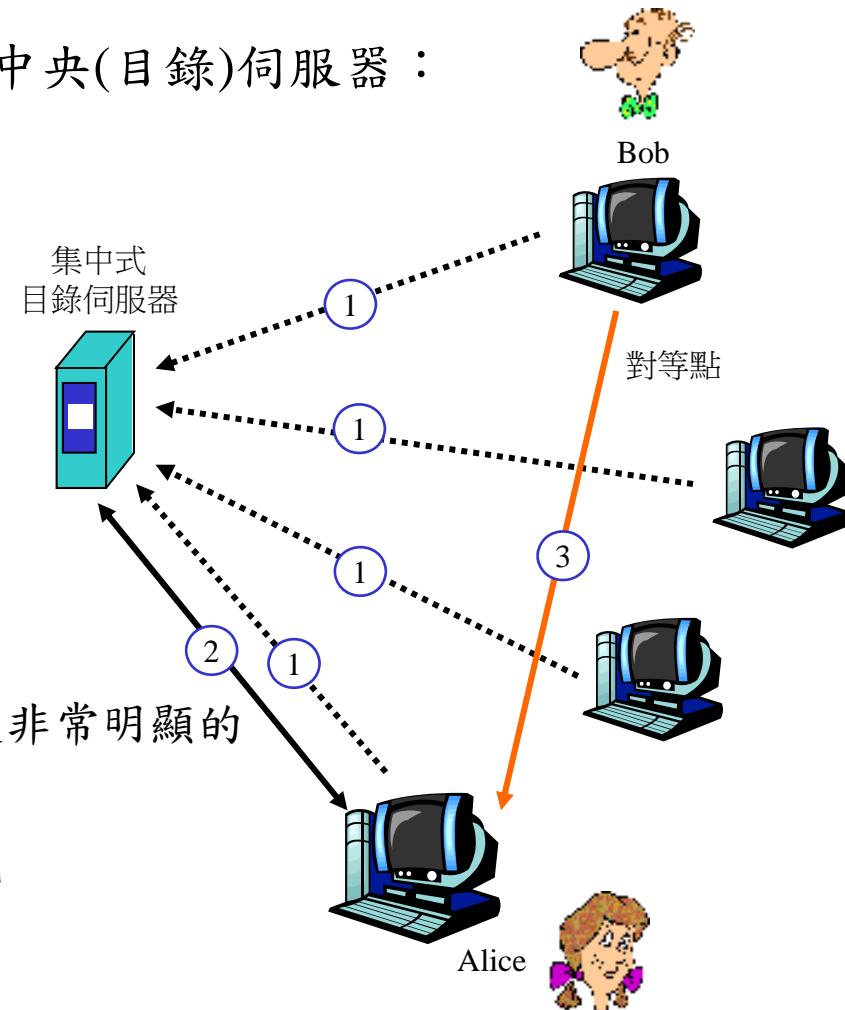
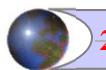


圖2.a 集中式P2P索引





■ 漫出式P2P查尋 (query flooding)

▶ 基本概念

- 完全分散式 – 非集中式伺服器
- 每個對等點都會對其提供分享的檔案建立索引
- 每個對等點不會對其它檔案建立索引

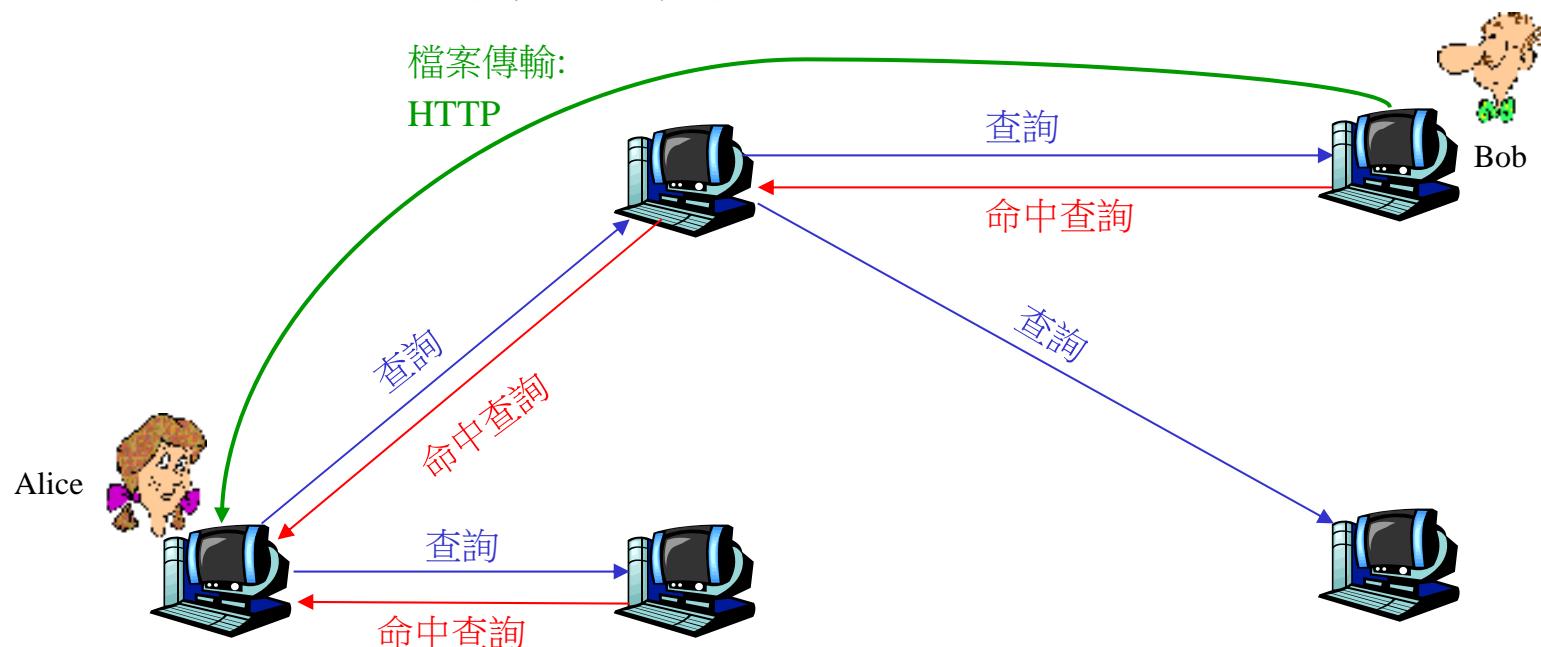
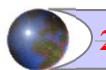


圖2.b 漫出查尋





■ 階層式重疊 (hierarchical query)

▶ 基本概念

- 集中式索引與查詢漫出的整合
- 每個對等點為一個群組領導點或是被指定給一個群組領導點
- 群組領導點會追蹤它的子對等點的內容

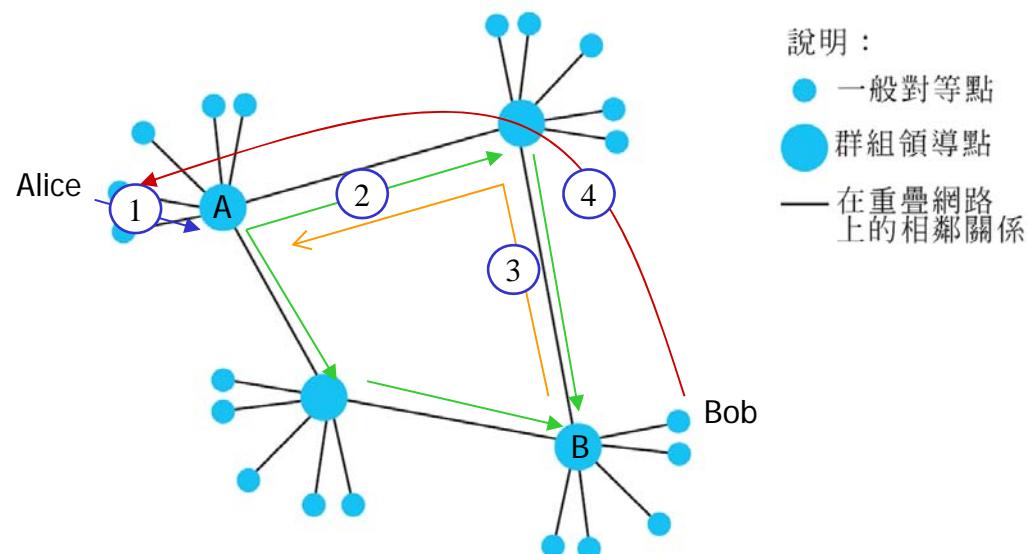
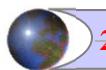


圖2.c 階層式重疊

Ex 2.13



■ 比較客戶端-伺服端與P2P架構

► 散佈時間

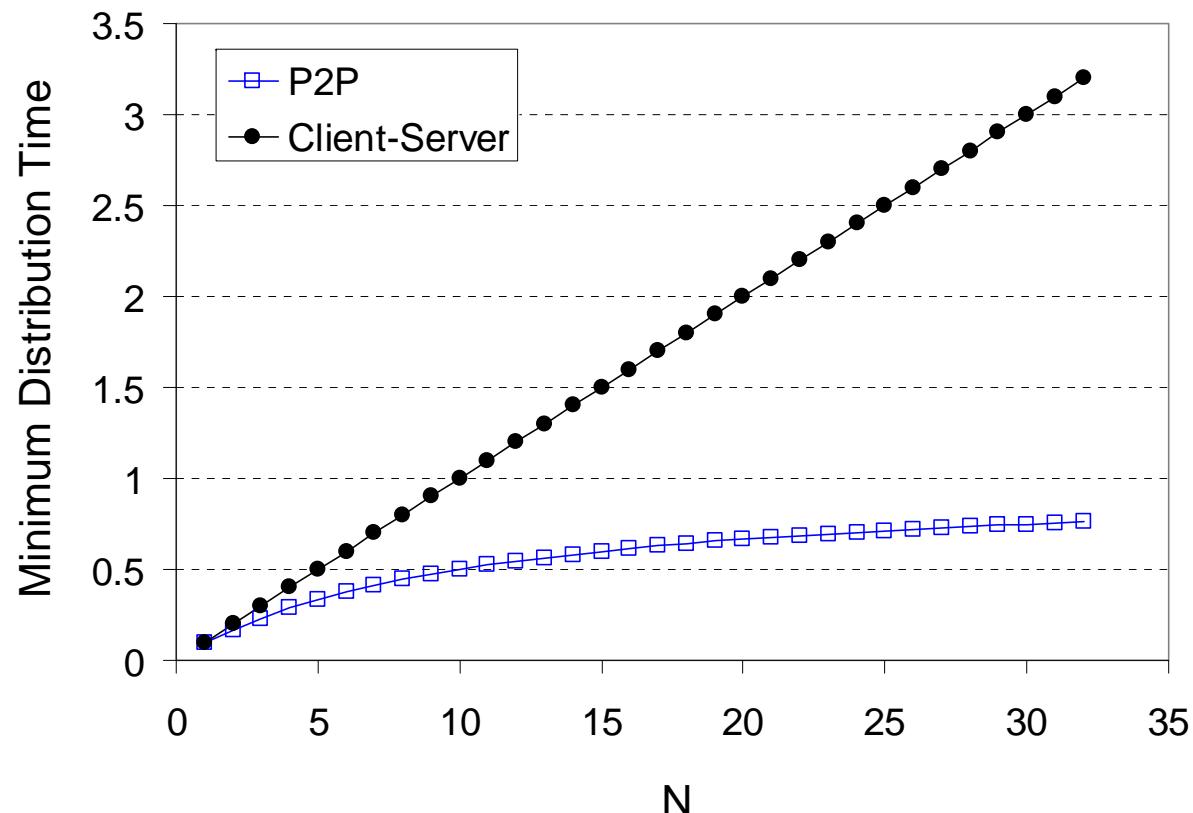
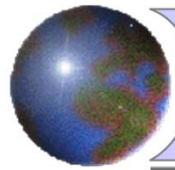


圖2.25 P2P架構與客戶端-伺服端架構的散佈時間比較



2.6.2 分散式雜湊表(DHT)



■ DHT

- ▶ DHT = distributed P2P database, distributed hash table
- ▶ P2P資料庫的運作
 - 資料庫欄位 = (*key, value*) pairs
 - eg. *key*: 身份證號碼; *value*: 姓名
 - 對等點使用 *key* 來對資料庫(DB)查尋，而 DB 回應該 *key* 對應的 *value*
 - Center database: 將 DB 放於一台電腦中 (what happen?)
 - P2P database
 - 將(*key, value*)配對分散在所有參與的對等點(peer)中
 - 每個對等點都只會持有所有(*key, value*)配對的一小部份
 - 對等點可查尋或插入(*key, value*)到此分散式對等資料庫中
 - 查尋作法: 循環式DHT



■ Circular DHT

- ① 給每個peer一個唯一識別碼(id)
- ② P2P資料庫中的對等點形成環狀(ring)
- ③ 每個對等點只知道前及後(immediate successor and predecessor)的對等點

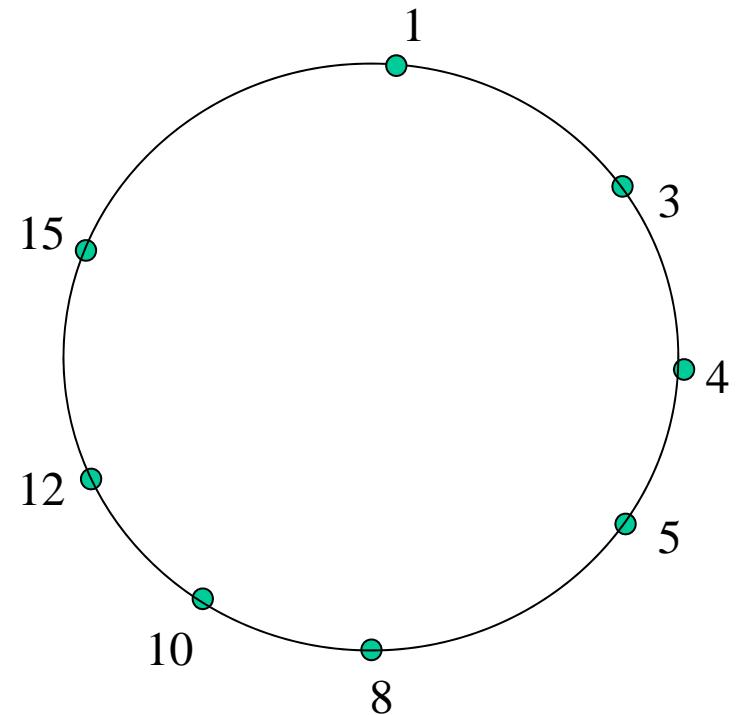
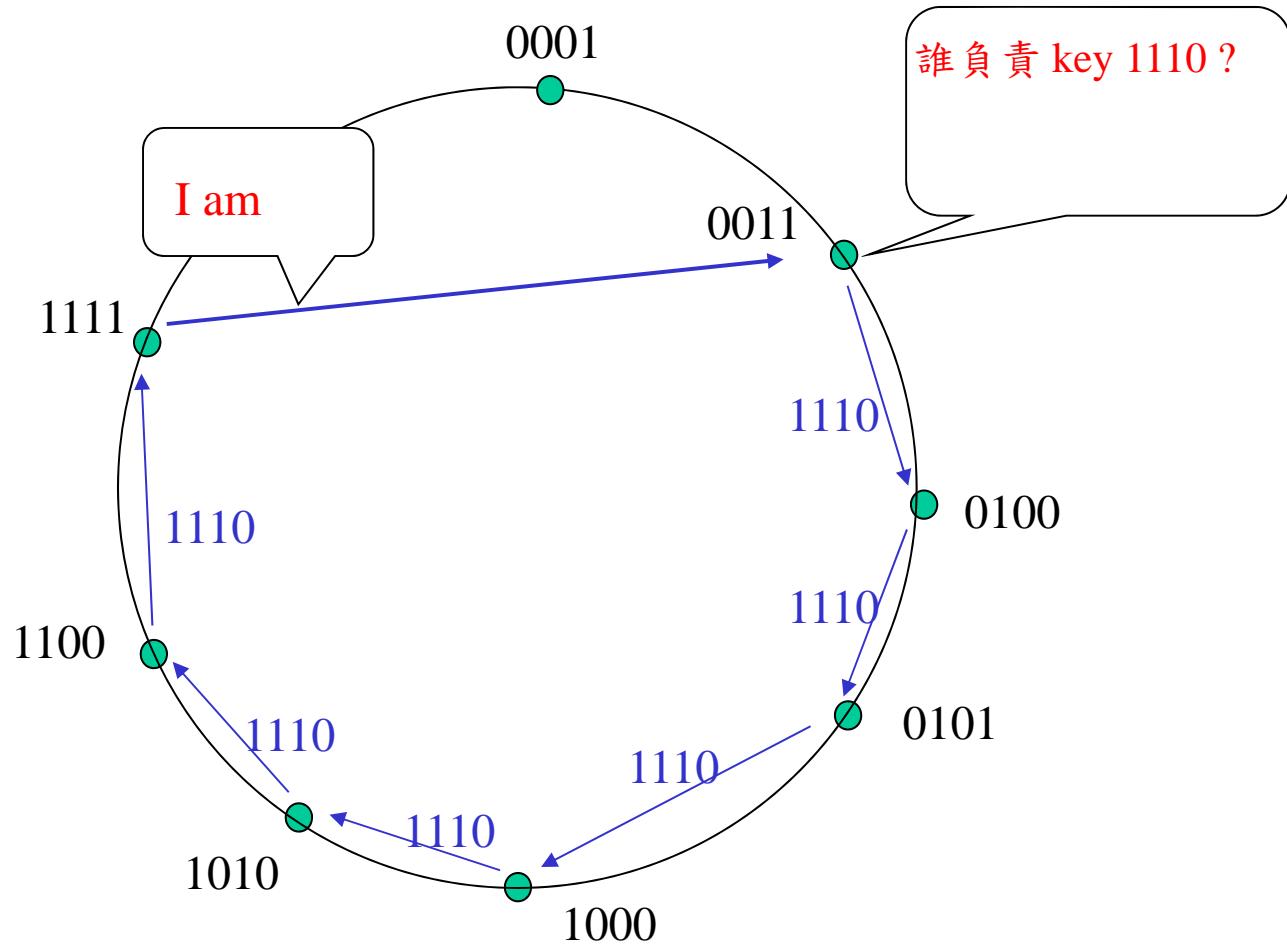


圖2.27(a) 環狀DHT



■ Circular DHT (Cont.)

④ 對等點3查尋key=1110由哪一對等點所控制擁有？





■ Circular DHT with Shortcuts

④ 對等點3查尋key=1110由哪一對等點所控制擁有？

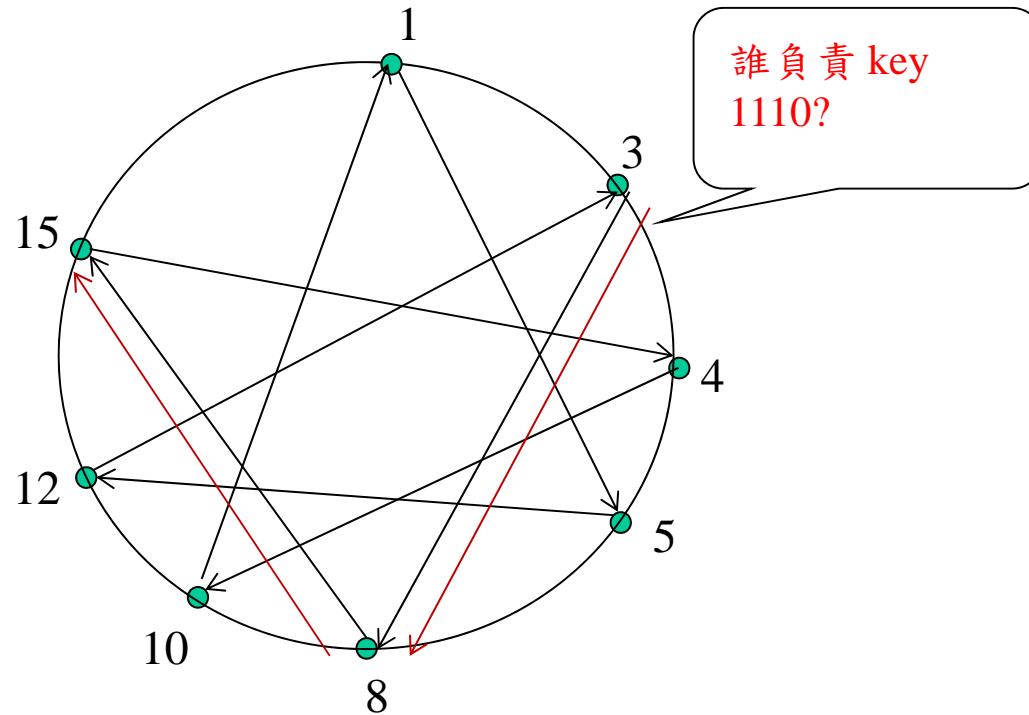
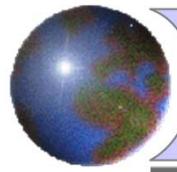


圖2.27(b) 包含捷徑的環狀DHT

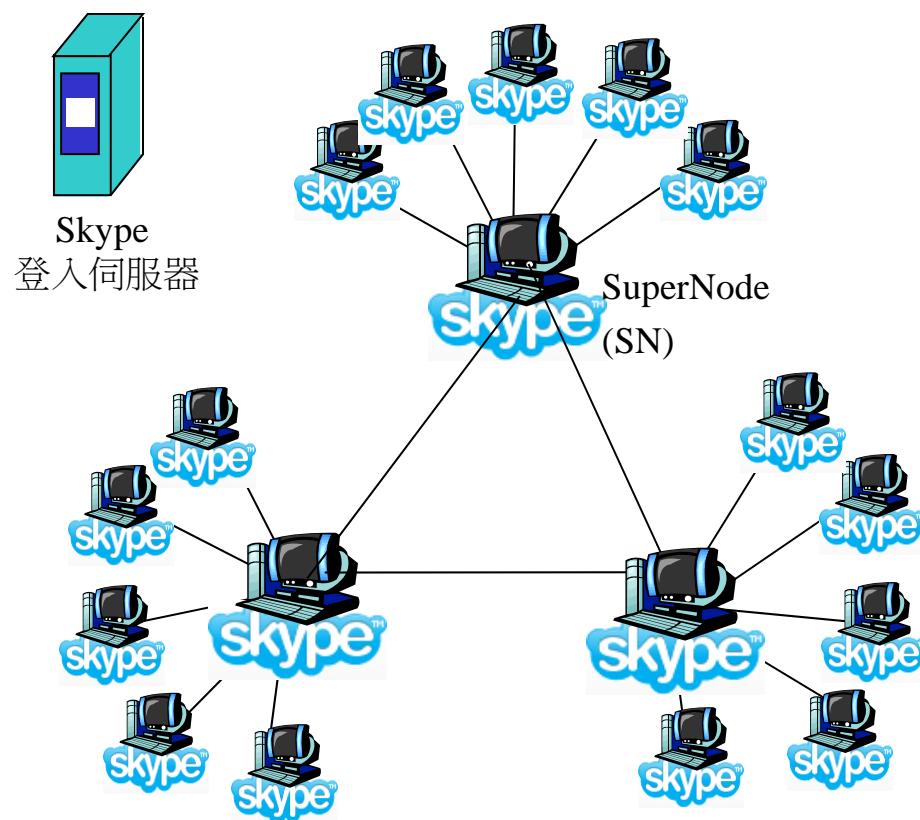


2.6.3 網際網路電話 (Skype)



The Skype logo consists of a red square icon followed by the word "Skype" in a red, lowercase, sans-serif font.

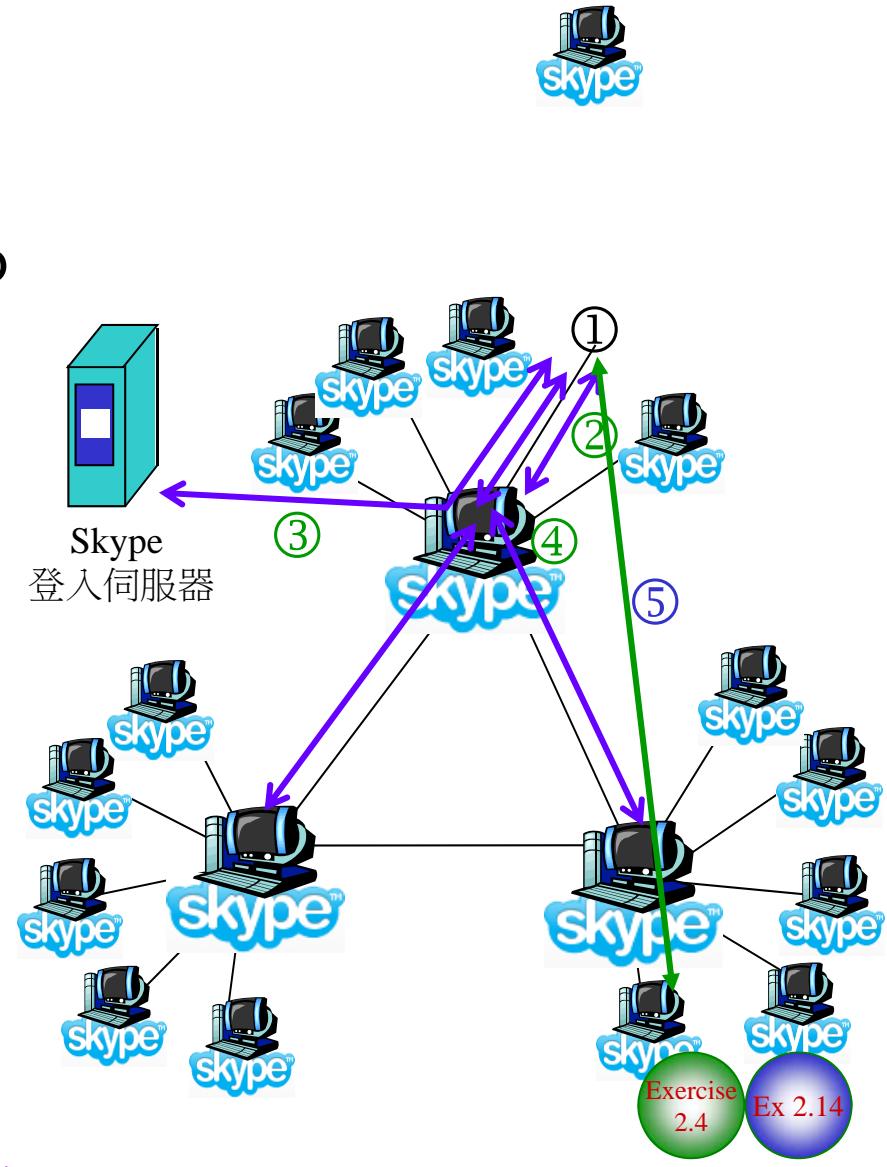
- ▶ P2P (pc到pc、pc到話機、話機到pc) Voice-Over-IP (VoIP) 應用
 - ▶ 階層式重疊 Skype 用戶端(SC)

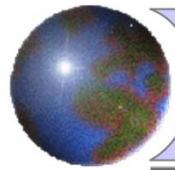




■ Skype : 打電話

- ① 開始使用 Skype
- ② SC 註冊到 SN
- ③ SC 登入Skype伺服器
- ④ 打電話： SC 連接SN 打給 ID
 SN 連接其他 SNs
 (未知協定、可能的漫出)
 找到電話的位址；
 回傳位址到 SC
- ⑤ SC 直接連接電話、經由TCP





第二章 導覽流程



2.1 網路應用的原理

2.2 Web 和 HTTP

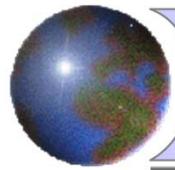
2.3 FTP

2.4 電子郵件 – SMTP、POP3、IMAP

2.5 DNS

2.6 點對點應用

2.7/2.8 使用 TCP/UDP 的 Socket 程式設計



2.7/2.8 Socket Programming with TCP/UDP

■ Socket 程式設計

► *socket*

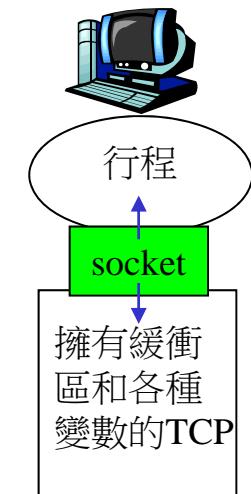
- 在應用程式行程與端點至端點傳輸協定 (UCP或TCP) 之間的門(插槽)
- 可以同時傳送及接收
- 訊息到/從另一個應用程式的行程

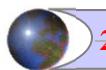
► *Socket API* (application program interface)

- 在BSD4.1 *UNIX*，1981被介紹 (vs. *Windows socket*)
- 由應用程式明顯地製造、使用、發佈
- 用戶端/伺服端模式
- 兩種型態的傳輸服務透過socket API：
 - 可靠的，位元組串流導向 (TCP)
 - 不可靠的資料段 (UDP)



主機或
伺服器





■ 使用 TCP 的 Socket 程式設計

- **TCP 服務**：使用位元組的可靠傳輸，從一個行程到另一個行程

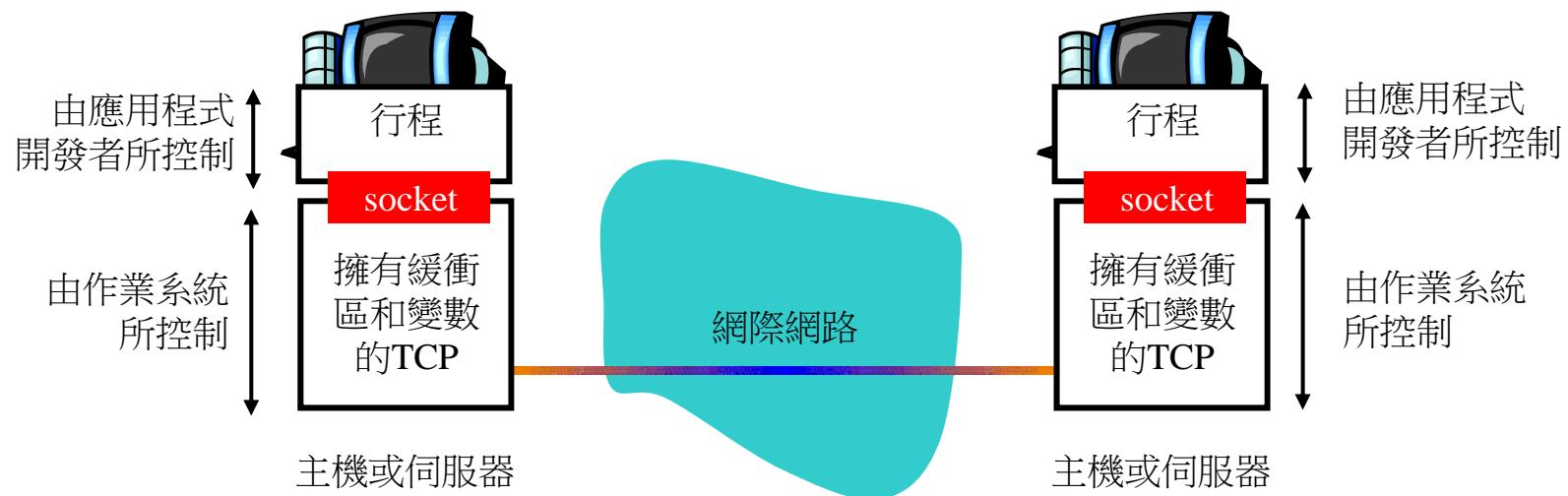


圖2.28 透過TCP socket 的行程通訊



■ 使用 TCP 的 Socket 程式設計 (Cont.)

- ▶ 用戶端/伺服端 socket 的互動: TCP

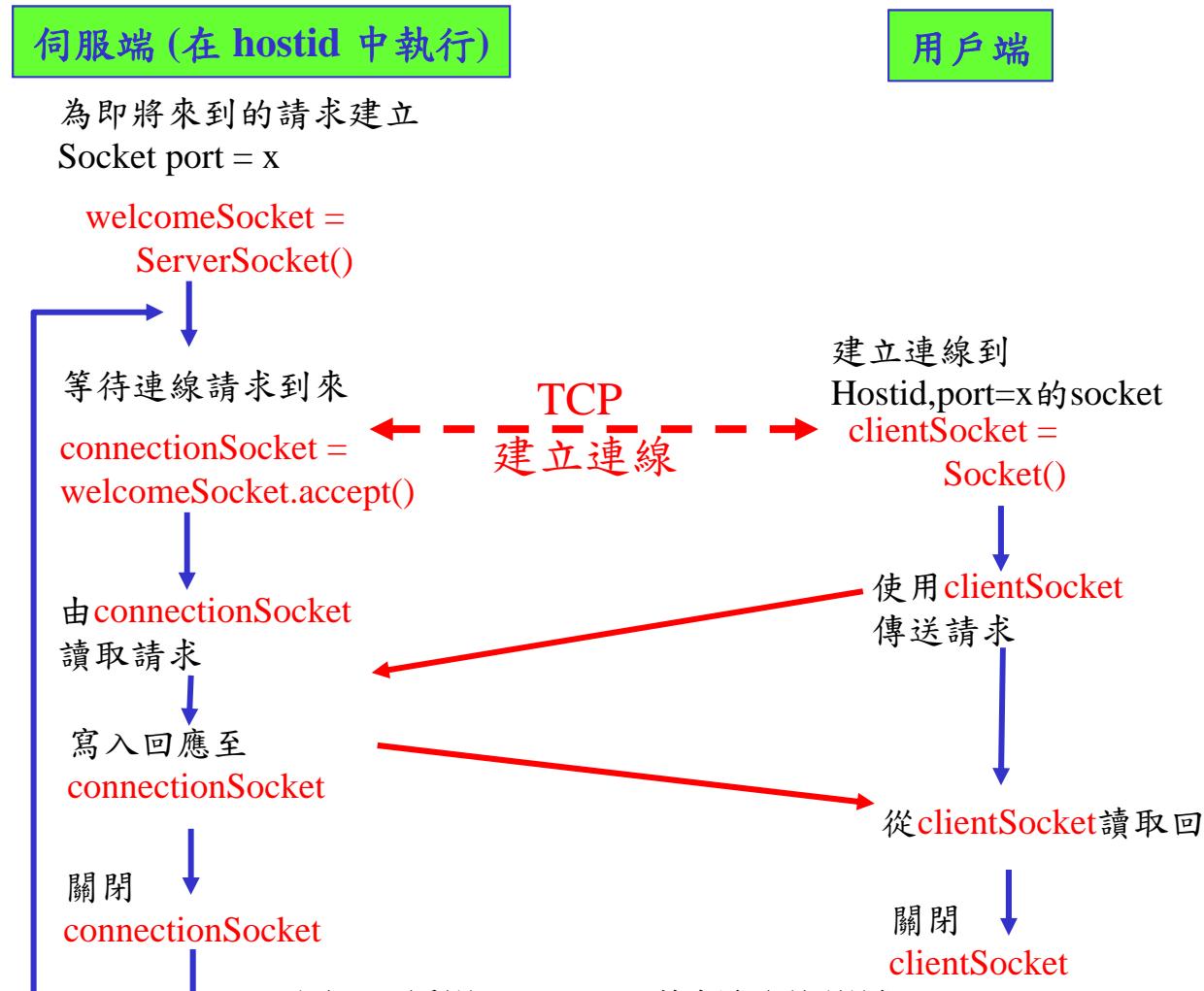


圖2.30 透過TCP socket 的行程通訊過程

2.96 / 103



■ 使用 UDP 的 Socket 程式設計 (Cont.)

- ▶ 用戶端/伺服端 socket 的互動: UDP

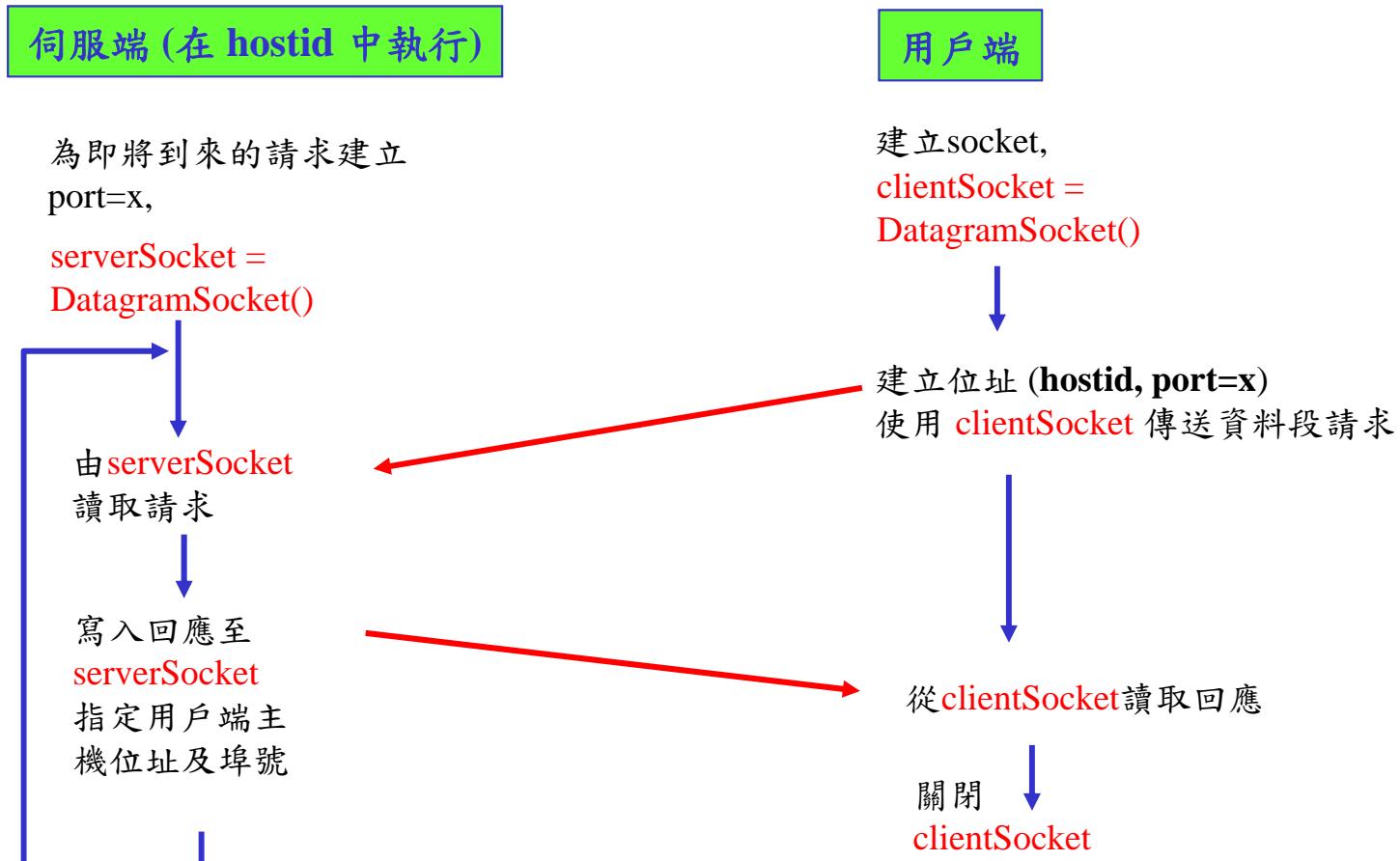
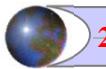


圖2.32 透過UDP socket 的行程通訊過程



■ Berkeley Sockets 程式設計

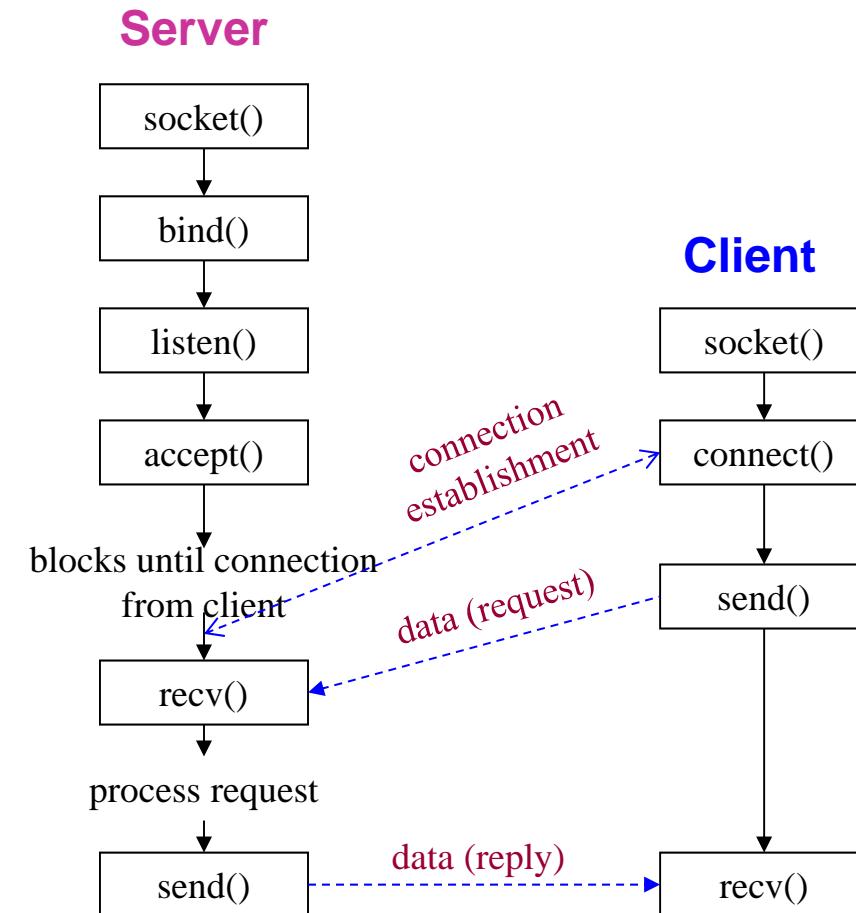
► Berkeley Sockets system calls

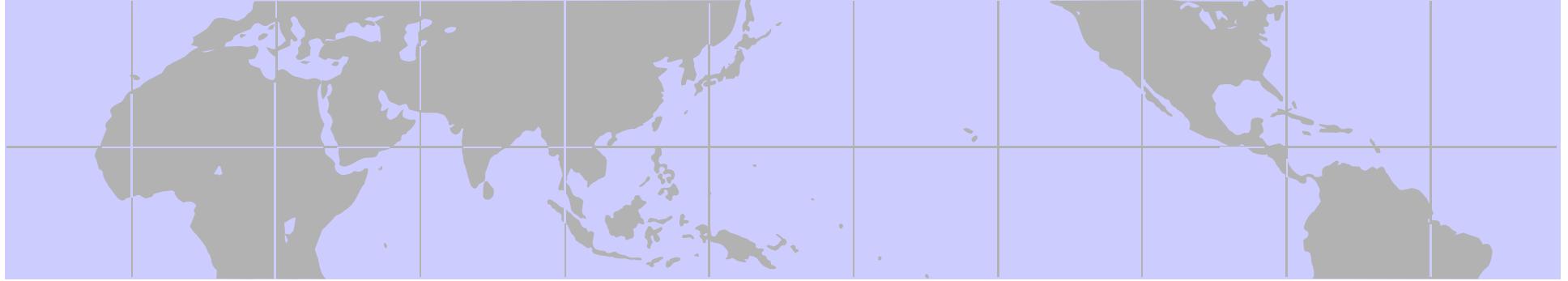
- `socket()` : create endpoint
- `bind()` : bind address
- `listen()` : specify queue
- `accept()` : wait for connection
- `connect()` : connect to server
- transfer data functions: `read()`, `write()`, `recv()`, `send()`
- datagrams functions : `recvfrom()`, `sendto()`
- terminate: `close()`, `shutdown()`



■ Berkeley Sockets 程式設計 (Cont.)

- connection-oriented protocol (TCP)



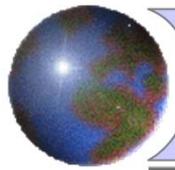


Chapter 3

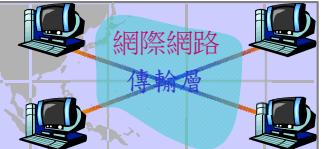
Transport Layer

(傳輸層)





Topic Overview



3.1 傳輸層服務

3.2 多工和解多工

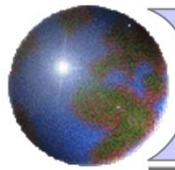
3.3 非預接式傳輸 – UDP

3.4 可靠資料傳輸的原理

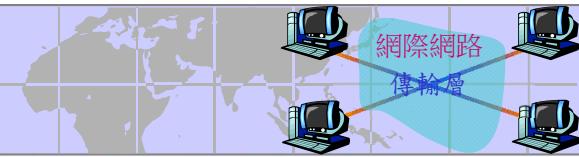
3.5 連線導向傳輸 – TCP

3.6 壓塞控制的原則

3.7 TCP 壓塞控制

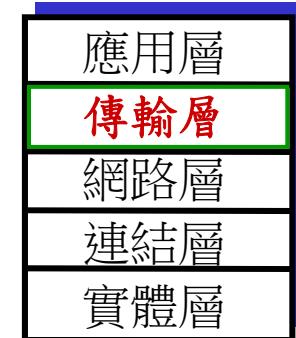


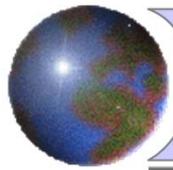
Topic Overview (Cont.)



■ 目標 (Goal)

- ▶ 了解傳輸層服務背後的原則
 - 多工/解多工
 - 可靠的資料傳輸 (reliable data transfer)
 - 流量控制 (flow control)
 - 壓塞控制 (congestion control)
- ▶ 學習有關網際網路上的傳輸層協定
 - UDP：非預接式傳輸
 - TCP：連線導向的傳輸
 - TCP：流量控制、壅塞控制





第三章 導覽流程



3.1 傳輸層服務

3.2 多工和解多工

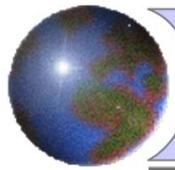
3.3 非預接式(無連線)傳輸 – UDP

3.4 可靠資料傳輸的原理

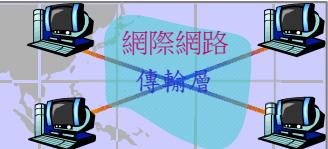
3.5 連線導向傳輸 – TCP

3.6 壓塞控制的原理

3.7 TCP 壓塞控制



3.1 傳輸層服務



■ 傳輸層服務及協定

- ▶ 提供不同主機上執行應用程式之間的邏輯通訊
 - 邏輯通訊 – 從應用程式的觀點來看，兩台執行行程的主機就像是直接連線；實際上，它們可能透過許多路由器及連結線進行連線
- ▶ 在終端(主機)系統間執行的傳輸協定(transport layer)
 - 傳送端： 將應用程式的訊息分割成資料分段、傳送到網路層
 - 接收端： 將資料分段重組成訊息、傳給應用層
- ▶ 應用層可用的傳輸協定超過一個
 - 網際網路： TCP 及 UDP
- ▶ 傳輸層協定的製作範圍
 - 製作在終端系統(*network edge*)
 - 網路中的路由器(*network core*)沒有製作此層，它們只製作到網路層



■ 傳輸層的邏輯通訊 (logical communication)

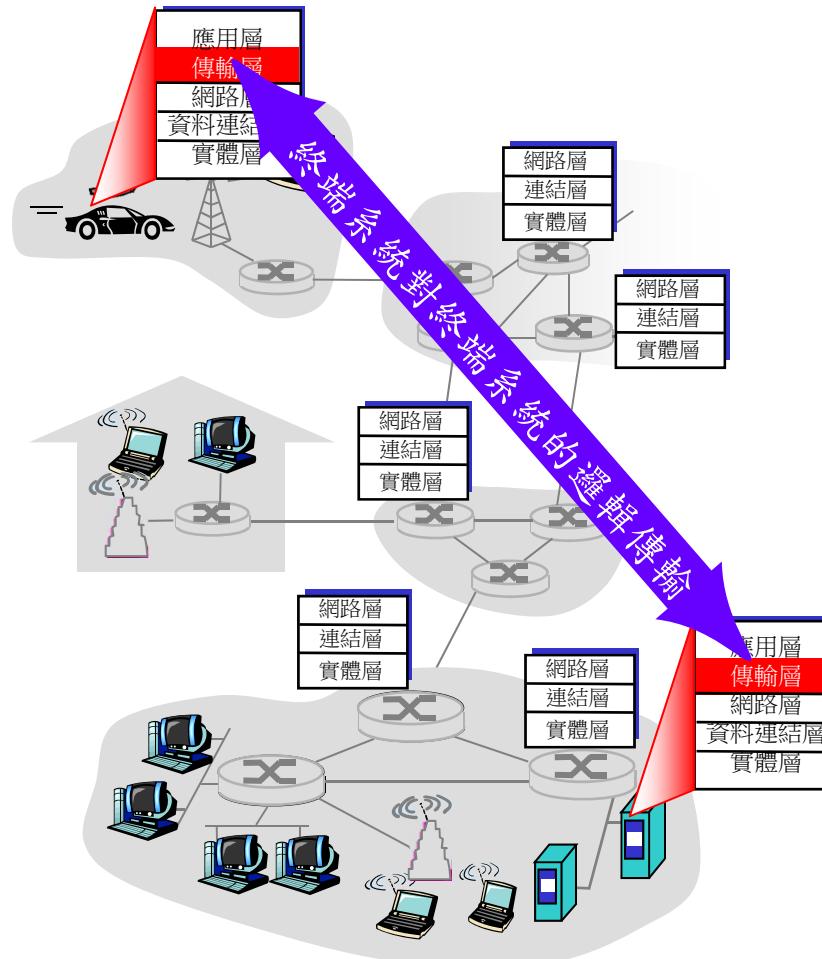
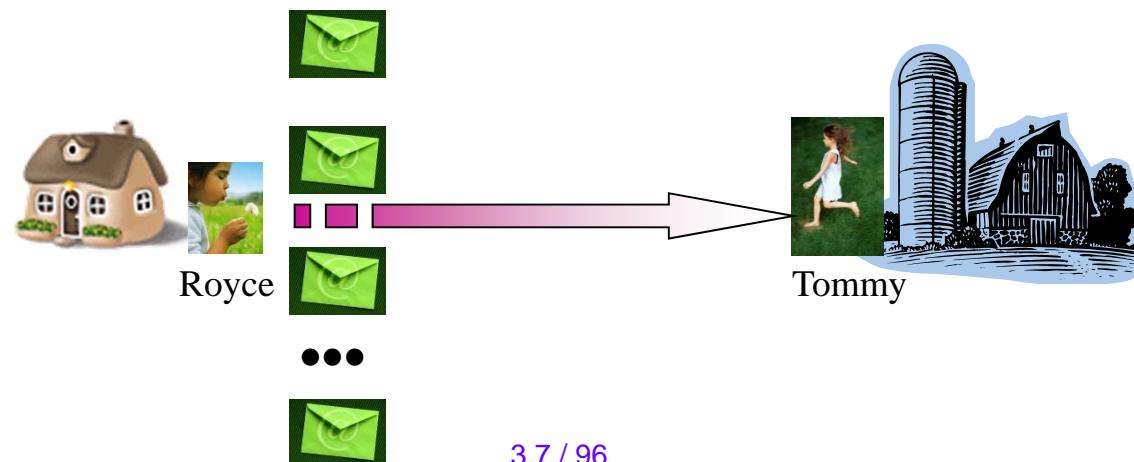


圖3.1 傳輸層提供了應用程式行程間的邏輯通訊而非實體的通訊



■ 傳輸層 VS. 網路層

- ▶ 網路層：提供主機之間的邏輯通訊
- ▶ 傳輸層：提供行程之間的邏輯通訊
 - 依賴、增強、網路層服務
- ▶ 家庭的比喻：12 個小孩傳送信件給12個小孩
 - 行程 = 小孩
 - 應用程式訊息 = 信封中的信
 - 主機 = 房子
 - 傳輸層協定 = Royce 以及 Tommy (雙方收送信的代表)
 - 網路層協定 = 郵政服務





■ 網際網路傳輸層協定

► 可靠的、有序的資料遞送 (TCP)

- 連線建立 (connection creation)
- 壓塞控制 (congestion control)
- 流量控制 (flow control)

► 不可靠的、無序的資料遞送：UDP

- “盡全力”的 IP 的精簡延伸

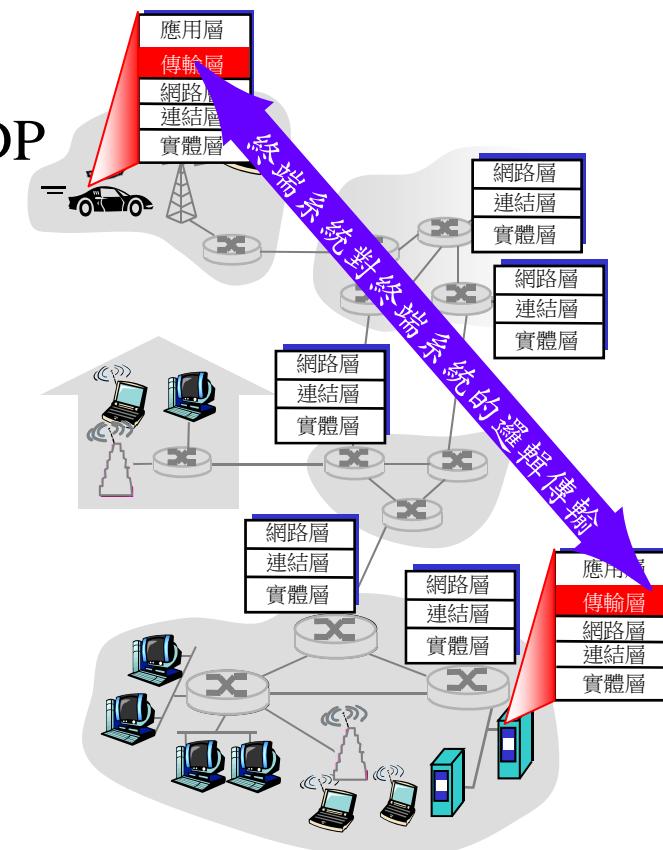
► 不提供的服務：

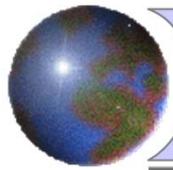
- 延遲保證
- 頻寬保證
- 安全性保證

【註】

① TCP/UDP 封包名稱 = segmentation
(區段)

② 網路層封包名稱 = datagram (資料包)





第三章 導覽流程



3.1 傳輸層服務

3.2 多工和解多工

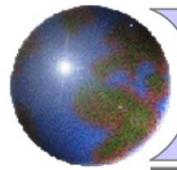
3.3 非預接式傳輸 – UDP

3.4 可靠資料傳輸的原理

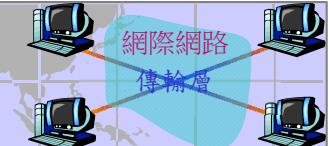
3.5 連線導向傳輸 – TCP

3.6 壓塞控制的原理

3.7 TCP 壓塞控制



3.2 多工和解多工



■ 多工(multiplexing)和解多工(Demultiplexing)

傳送端主機的多工：

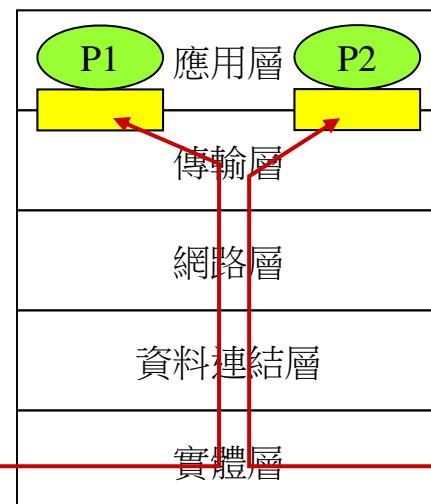
從不同的socket收集資料片段、
用標頭(稍後將用在解多工)
將每個資料片段封裝成資料區段

接收端主機的解多工：

將收到的資料分段傳送給正確的socket

= socket

= 行程



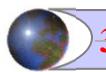
主機 1 (傳送端)

主機 2 (接收端)

主機 3 (傳送端)

圖3.2 傳輸層的多工與解多工





■ 解多工如何運作？

- ▶ 主機收到網路層的 IP 資料包 (IP datagram)
 - 每一個IP資料包都擁有來源端IP位址以及目的端IP位址
 - 每一個IP資料包載送 1 個傳輸層資料區段 (*transport-layer segment*)
 - 每一個傳輸層資料區段都擁有來源端埠號以及目的端埠號
- ▶ 主機使用 IP 位址以及埠號(port number) 將資料區段送到正確的socket

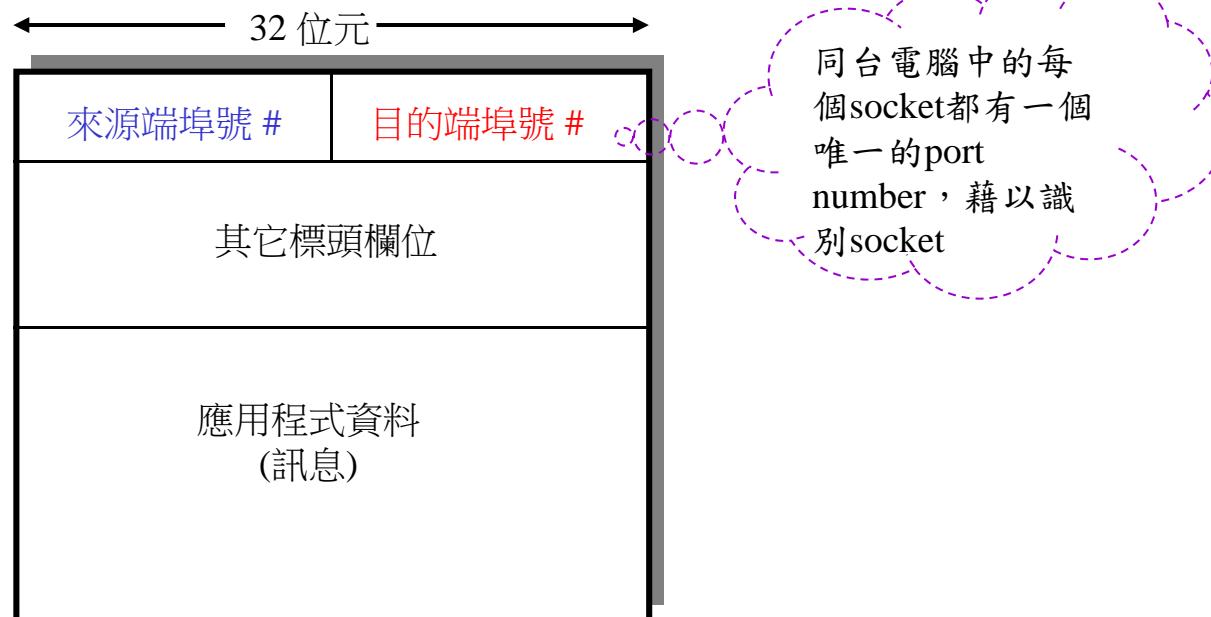


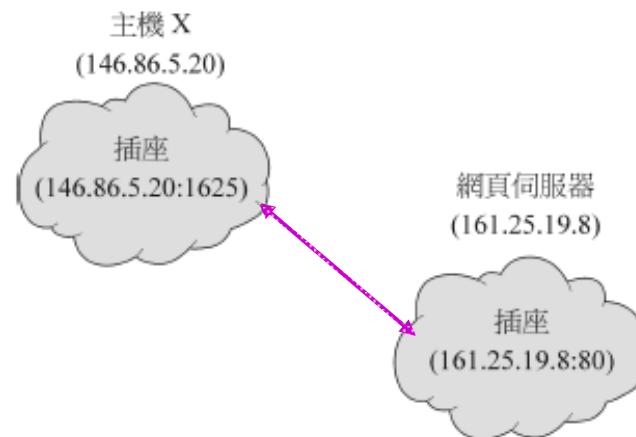
圖3.3 傳輸層資料區段(segment)格式





■ Socket 行程的基本通訊

- ▶ Socket 可由一個 **IP 位址** 和一個 **埠號** (port number) 所組成
- ▶ 一組行程使用一對插座 (socket，雙方各一個) 在網路上通訊





■ 無連線(connectionless)的解多工 – UDP

- ▶ 以埠號產生 socket :

```
DatagramSocket mySocket1 = new DatagramSocket(12534);
```

```
DatagramSocket mySocket2 = new DatagramSocket(12535);
```

- ▶ 以兩組資料識別 UDP socket :

(目的端 IP 位址、目的端埠號)

- ▶ 當主機收到 UDP 資料區段(UDP segment)時：

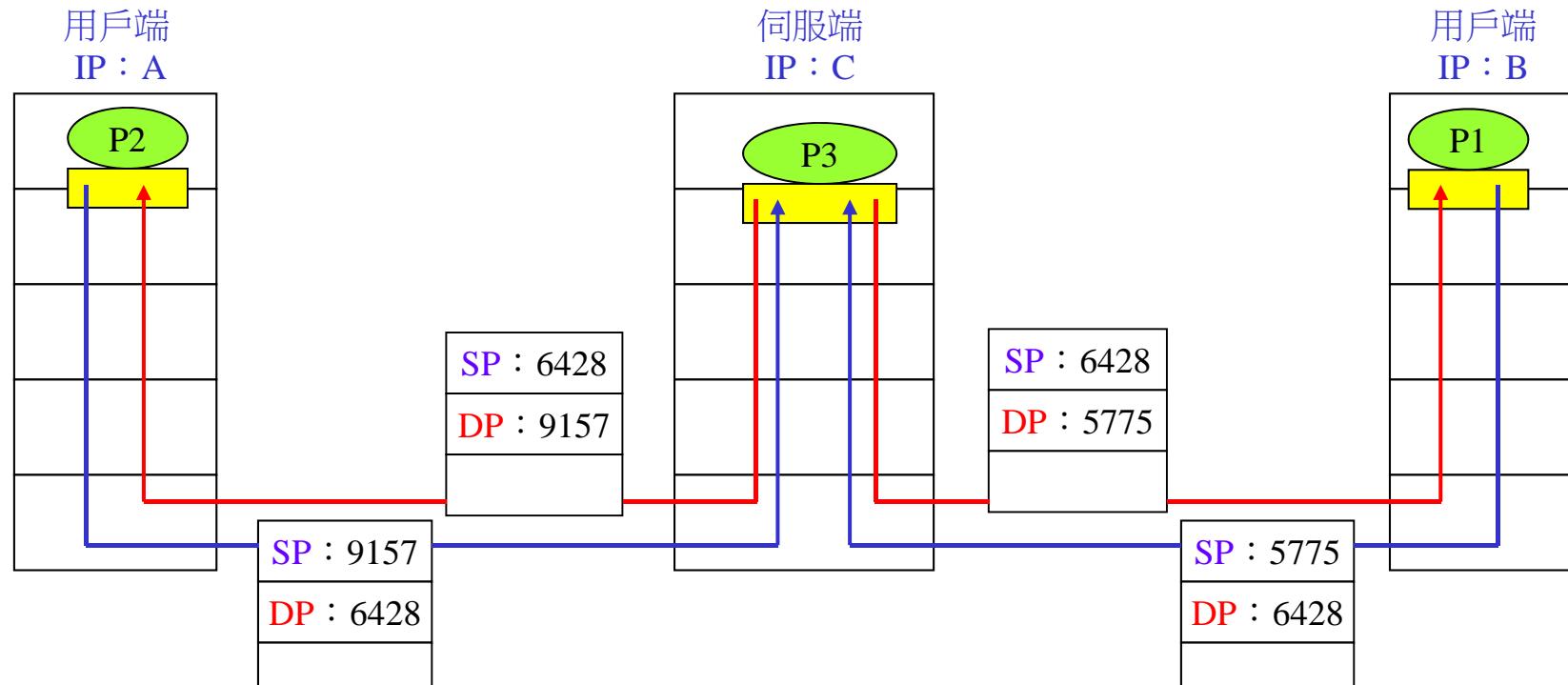
- 確認資料區段中的目的端埠號
 - 以此埠號將 UDP 資料區段傳送到 socket

- ▶ 具有不同來源端 IP 位址的IP 資料包(網路層封包)和/或 來源端埠號會被送到同一個 socket (若目的端埠號相同)



■ 無連線(connectionless)的解多工 (Cont.)

```
DatagramSocket serverSocket = new DatagramSocket(6428);
```



SP = Source Port

DP = Destination Port

SP 提供 “回傳埠號”

圖3.4 來源端與目的端埠號的交換



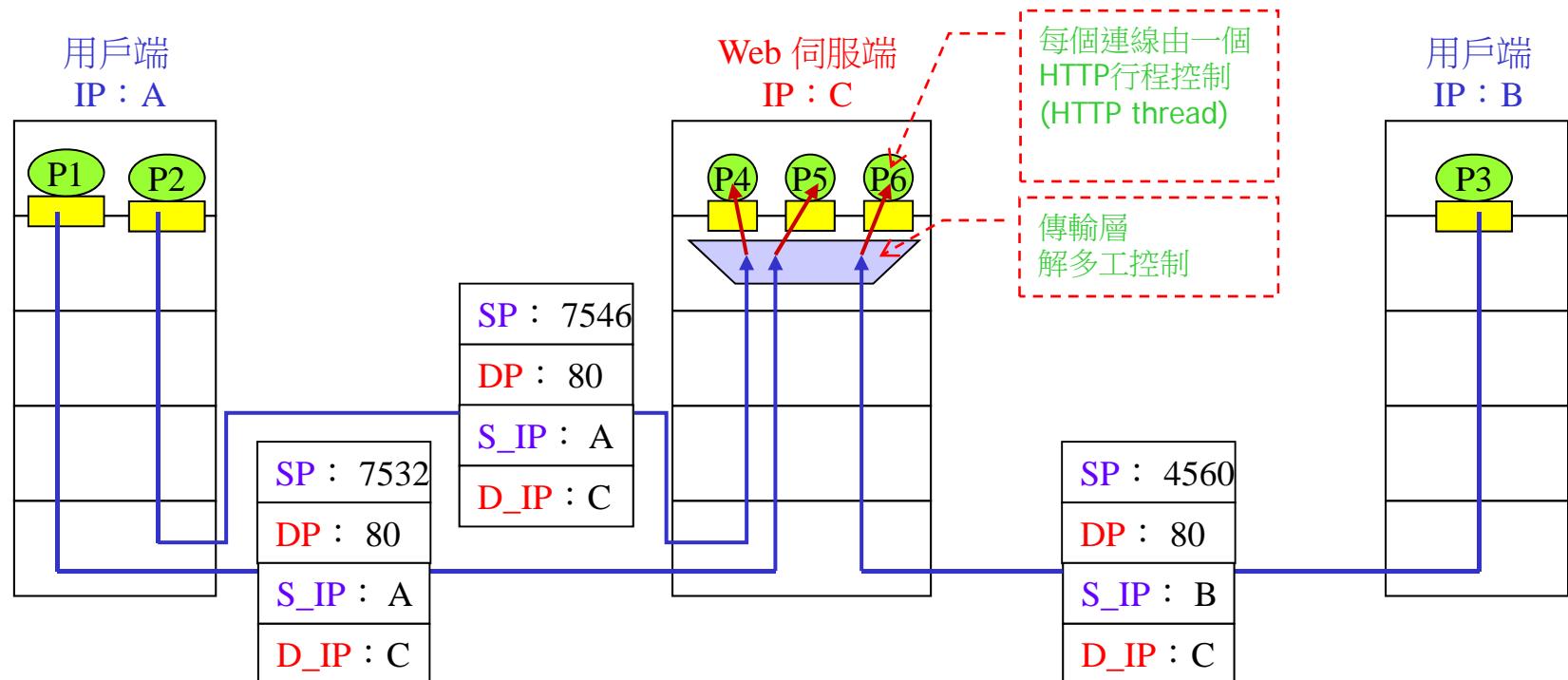


■ 連線(connection)的解多工 – TCP

- ▶ TCP socket 以 四組資料加以識別：
 - 來源端 IP 位址
 - 來源端埠號
 - 目的端 IP 位址
 - 目的端埠號
- ▶ 接收端主機使用全部的四個數值將資料區段送到適當的 socket
- ▶ 伺服端主機可能同時支援許多 TCP sockets：
 - 每個 socket 以它自己的四組資料加以識別
- ▶ Web 伺服器針對連結到它的每一個用戶端都有不同的 socket
 - 非永久性 HTTP 針對每一次的請求都有不同的 socket



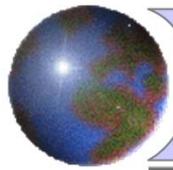
■ 連線(connection)的解多工 (Cont.)



SP = Source Port
DP = Destination Port
S_IP = Source IP addr.
D_IP = Destination IP addr.

圖3.5 兩個用戶端，使用相同的目的端埠號(80)來與相同的Web伺服器進行通訊





第三章 導覽流程



3.1 傳輸層服務

3.2 多工和解多工

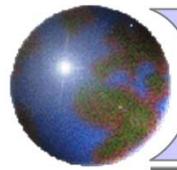
3.3 非預接式(無連線)傳輸 – UDP

3.4 可靠資料傳輸的原理

3.5 連線導向傳輸 – TCP

3.6 壓塞控制的原理

3.7 TCP 壓塞控制



3.3 非預接式傳輸 – UDP



■ UDP (User Datagram Protocol) : 使用者資料包協定

- ▶ RFC 768
- ▶ 實際的、精簡的網際網路傳輸協定
- ▶ “盡全力”的服務、UDP 資料區段可能：
 - 遺失 ⇐ 不保證資料會到達目的地
 - 不按順序傳送給應用程式
- ▶ 非預接式服務
 - 在 UDP 傳送端和接收端之間 沒有交握程序 (*non-handshaking*)
 - 每一個 UDP 資料區段的處理和其它資料區段是獨立的

為何會使用 UDP ?

- (1) 不需建立連線 (建立連線會增加延遲)
- (2) 簡單：在傳送端和接收端不需維持連線狀態
- (3) 較小的封包標頭
- (4) 沒有壅塞控制：UDP 可以盡可能地快速傳送資料
⇒ DNS 採用此種傳輸方式





■ UDP (Cont.)

▶ 常見的網際網路應用及其下層的傳輸協定

應用	應用層協定	底層傳輸層協定
電子郵件	SMTP	TCP
遠端終端機連線	Telnet	TCP
網頁	HTTP	TCP
檔案傳輸	FTP	TCP
遠端檔案伺服器	NFS	一般是 UDP
串流多媒體	私人協定	UDP 或 TCP
網際網路電話	私人協定	UDP 或 TCP
網路管理	SNMP	一般是 UDP
繞徑協定	RIP	一般是 UDP
網域名稱系統	DNS	一般是 UDP

圖3.6 常見的網際網路應用及其下層的傳輸協定



■ UDP (Cont.)

► 通常用在串流的多媒體應用程式

- 可以容忍資料遺失
- 易受速率影響

► 其它使用 UDP 的有：

- DNS
- SNMP (Simple Network Management Protocol)

► 使用 *UDP* 的可靠傳輸？ \Rightarrow 在應用層加入可靠性的機制

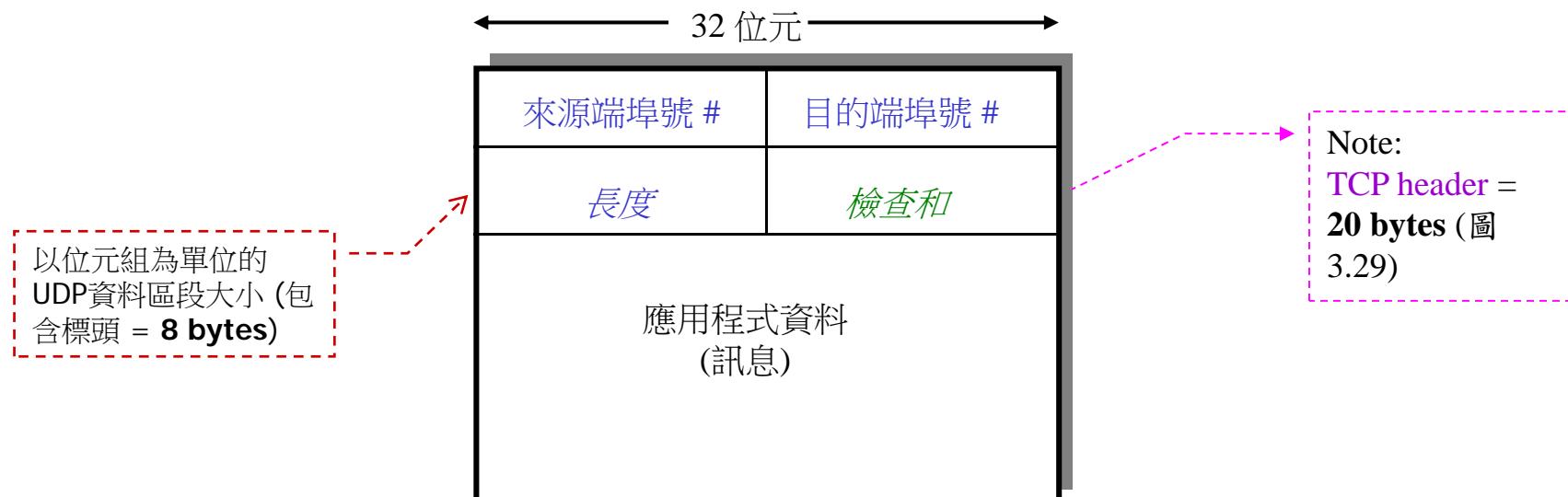


圖3.7 UDP 資料區段(segment)格式



■ UDP (Cont.)

► UDP 檢查和 (check sum)

- 目標： 偵測傳送的資料區段中的 “錯誤”（例如，偵測被翻轉的位元）
- 作法：

傳送端：

- 將資料區段的內容視為一列16位元的整數
- 檢查和：資料區段內容的加法(1的補數和)
- 傳送端將檢查和的值放入UDP的檢查和欄位

Ex:

0110011001100000
0101010101010101
1000111100001100

$$\begin{array}{r} 0110011001100000 \\ 0101010101010101 \\ +) \underline{1000111100001100} \\ 1010010101100001 \\ \downarrow \\ 0100101011000010 \end{array}$$

↓
1's補數: 0→1&1→0

check sum = 1011010100111101



接收端：

- 計算收到的資料區段的資料和(不包括檢查和欄位)
- 確認計算出來的資料和與檢查和欄位的加法結果是否為1111...11

► NO – 偵測到錯誤

► YES – 沒有偵測到錯誤。但是仍然可能有錯誤？後面有更多介紹

⇒ 資料和+檢查和 = 11111...11

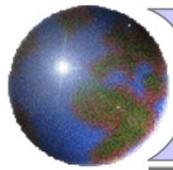
例題：

UDP的資料區段(segment)格式如圖3.7所示。請說明傳送端(來源端)與接收端(目的端)如何偵測傳送的資料區段中的錯誤？

Ex 3.4

Exercise
3.2

Ex 3.3



第三章 導覽流程



3.1 傳輸層服務

3.2 多工和解多工

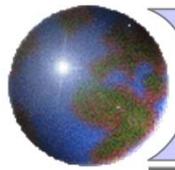
3.3 非預接式傳輸 – UDP

3.4 可靠資料傳輸的原理

3.5 連線導向傳輸 – TCP

3.6 壓塞控制的原理

3.7 TCP 壓塞控制



3.4 可靠資料傳輸的原理



■ 實作可靠資料傳輸 (reliable data transfer) 的分層

- ▶ 應用層、傳輸層、資料連結層
- ▶ 可以列在網路問題中的前十大清單
- ▶ 不可靠通道(資料會遺失及錯誤)的特性決定了可靠資料傳輸協定 (rdt, reliable data transfer) 的複雜性

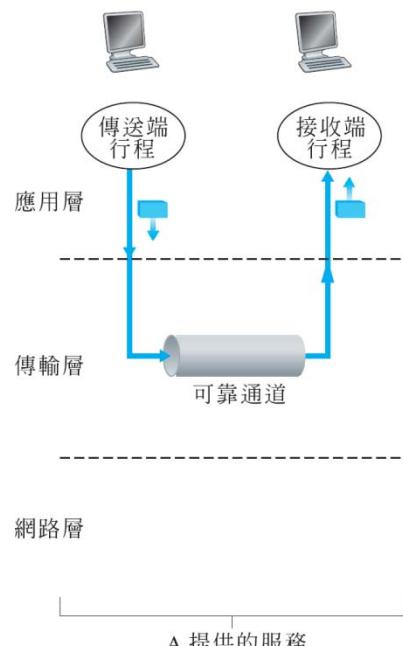
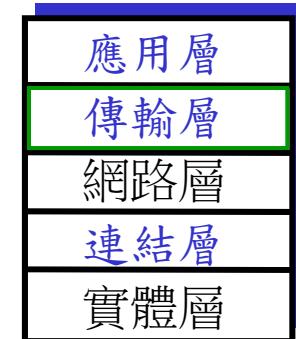


圖3.8 可靠的資料傳輸 – 服務模型與服務實作

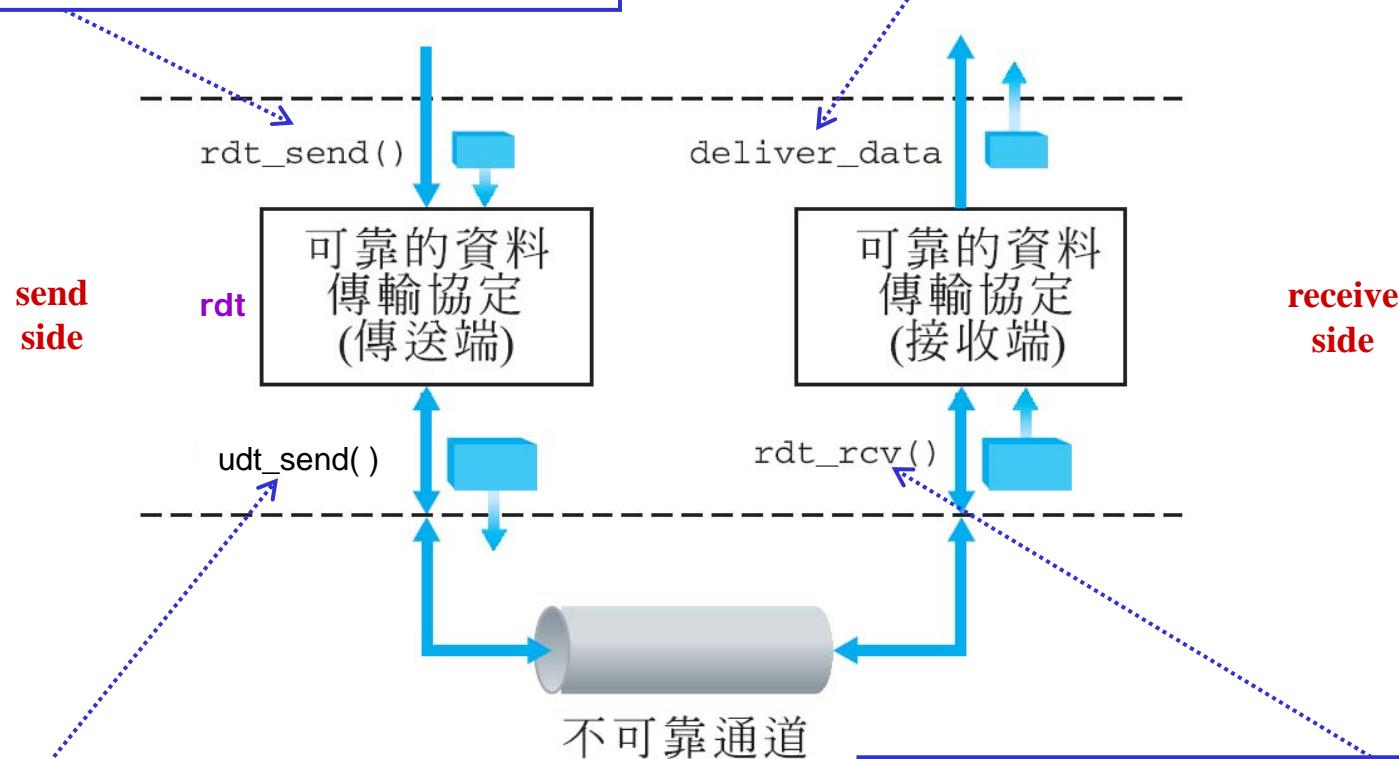




■ 建立可靠的資料傳輸

① **rdt_send()**：被上層呼叫（例如，應用層）。
將資料傳遞給接收端的上層協定

④ **deliver_data()**：被 rdt 呼叫。
將資料傳送到上層

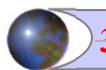


② **udt_send()**：被 rdt 呼叫。
經由不可靠的通道將封包
傳送給接收端
(un-reliable data transfer)

③ **rdt_rcv()**：當封包抵達接收端的通道時被呼叫

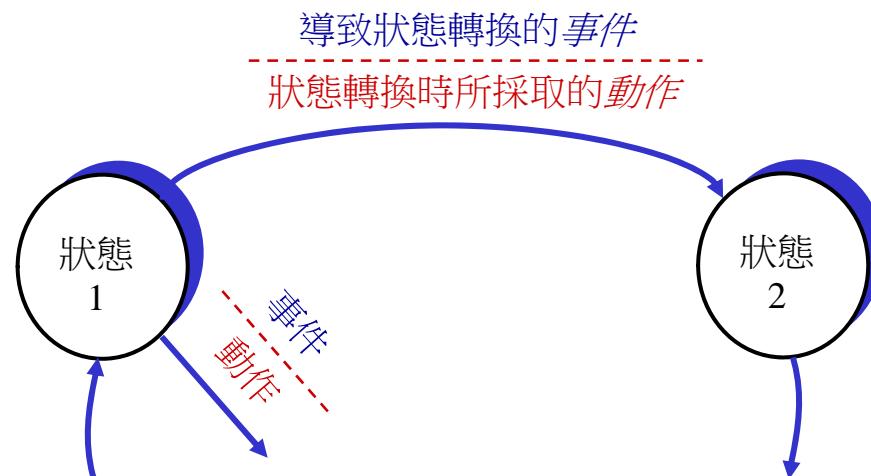


Ex 3.5



■ 有限狀態機 (FSM)

- ▶ Finite-State Machine
- ▶ 說明傳送端、接收端可靠資料傳輸協定的狀態/事件轉移



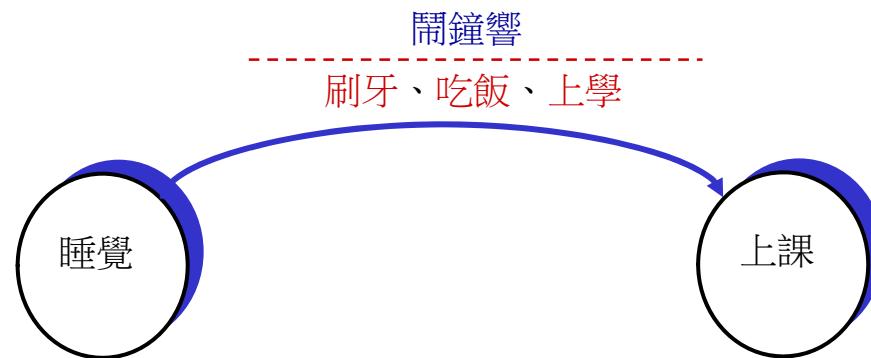
狀態：

在這個“狀態”時，下一個
狀態將唯一地被下一個事件
所決定



■ FSM (Cont.)

► 例子





■ Rdt1.0 – 使用可靠通道的可靠傳輸

- ▶ 底層的通道是完全可靠的
 - 沒有位元錯誤
 - 沒有資料遺失
- ▶ 傳送端和接收端擁有各自的 FSM：
 - 傳送端將資料送入底層的通道
 - 接收端從底層的通道接收資料

next two slides

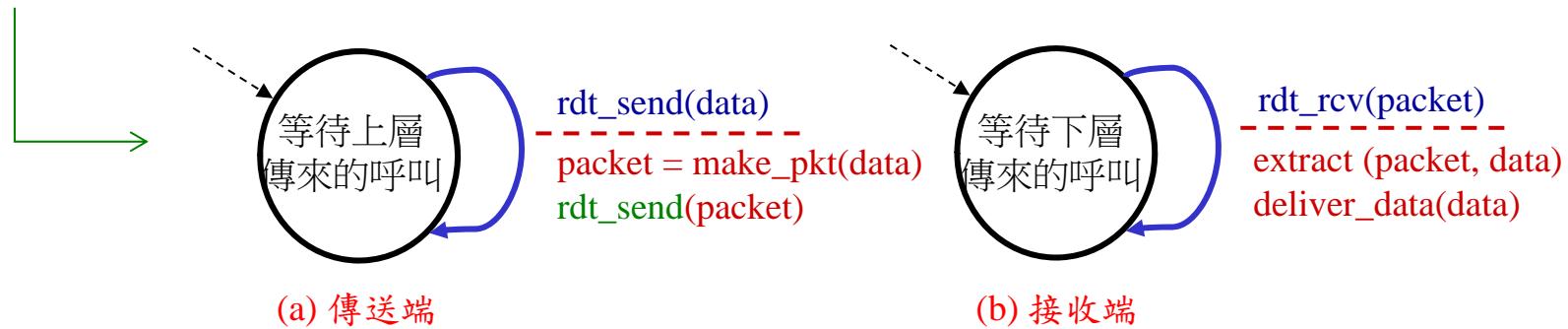
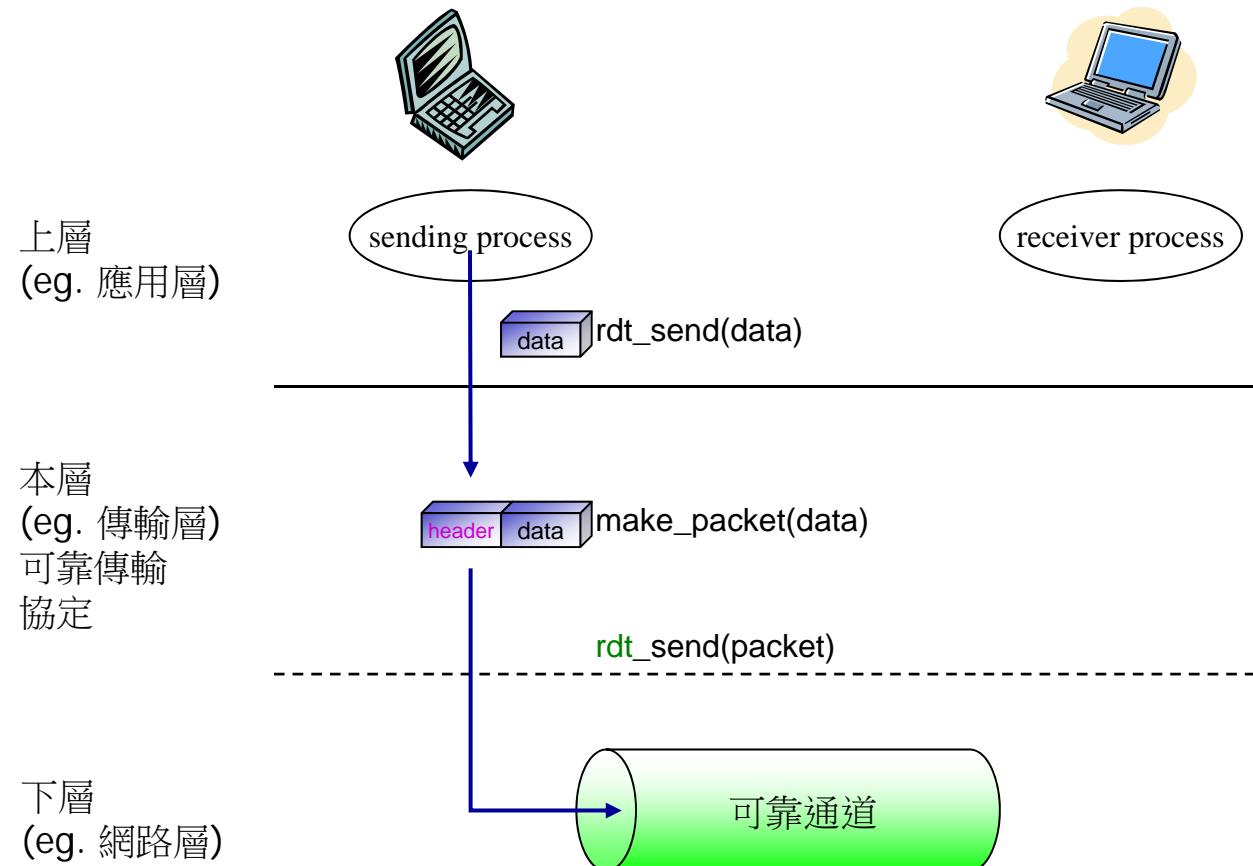


圖3.9 使用絕對可靠通道的協定



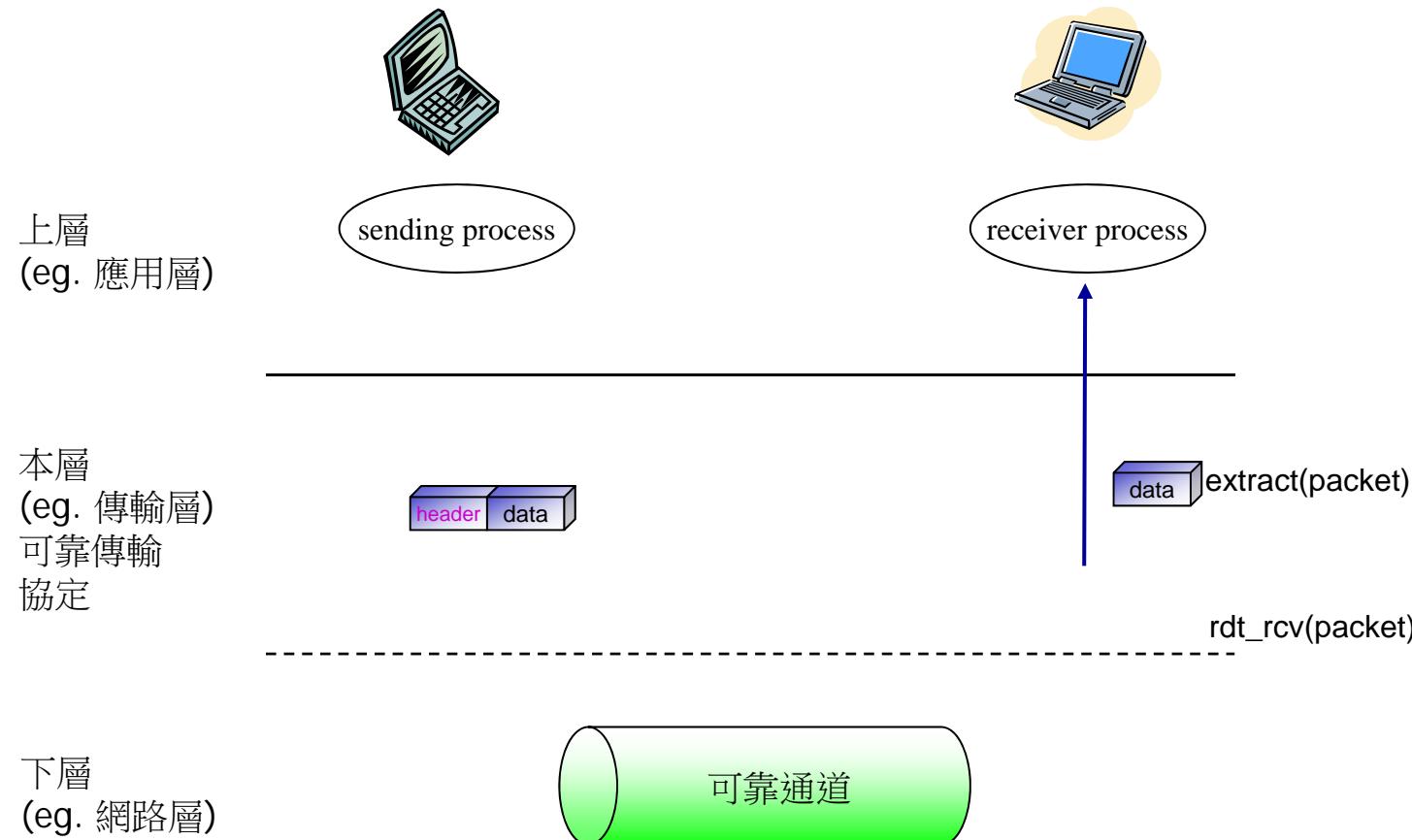


■ Rdt1.0 – 使用可靠通道的可靠傳輸 (Cont.)





■ Rdt1.0 – 使用可靠通道的可靠傳輸 (Cont.)





■ Rdt2.0 – 可能產生位元錯誤的通道

- ▶ 底層的通道可能會將封包中的位元翻轉 (位元錯誤)
 - 偵測位元錯誤的檢查和
- ▶ 問題：如何回覆錯誤？
 - 確認 (ACKs)：接收端明確地告訴傳送端封包的傳送 OK (正確)
 - 否定確認 (NAKs)：接收端明確地告訴傳送端封包的傳送有問題 (不正確)
 - 當收到NAK時，傳送端會重傳封包
- ▶ rdt2.0 的新機制 (優於rdt1.0)：具ARQ協定 (Auto Repeat reQuest)
 - 錯誤偵測
 - 接收端回饋：控制訊息 (ACK、NAK) 接收端 → 傳送端



■ Rdt2.0 (Cont.)

► FSM 說明

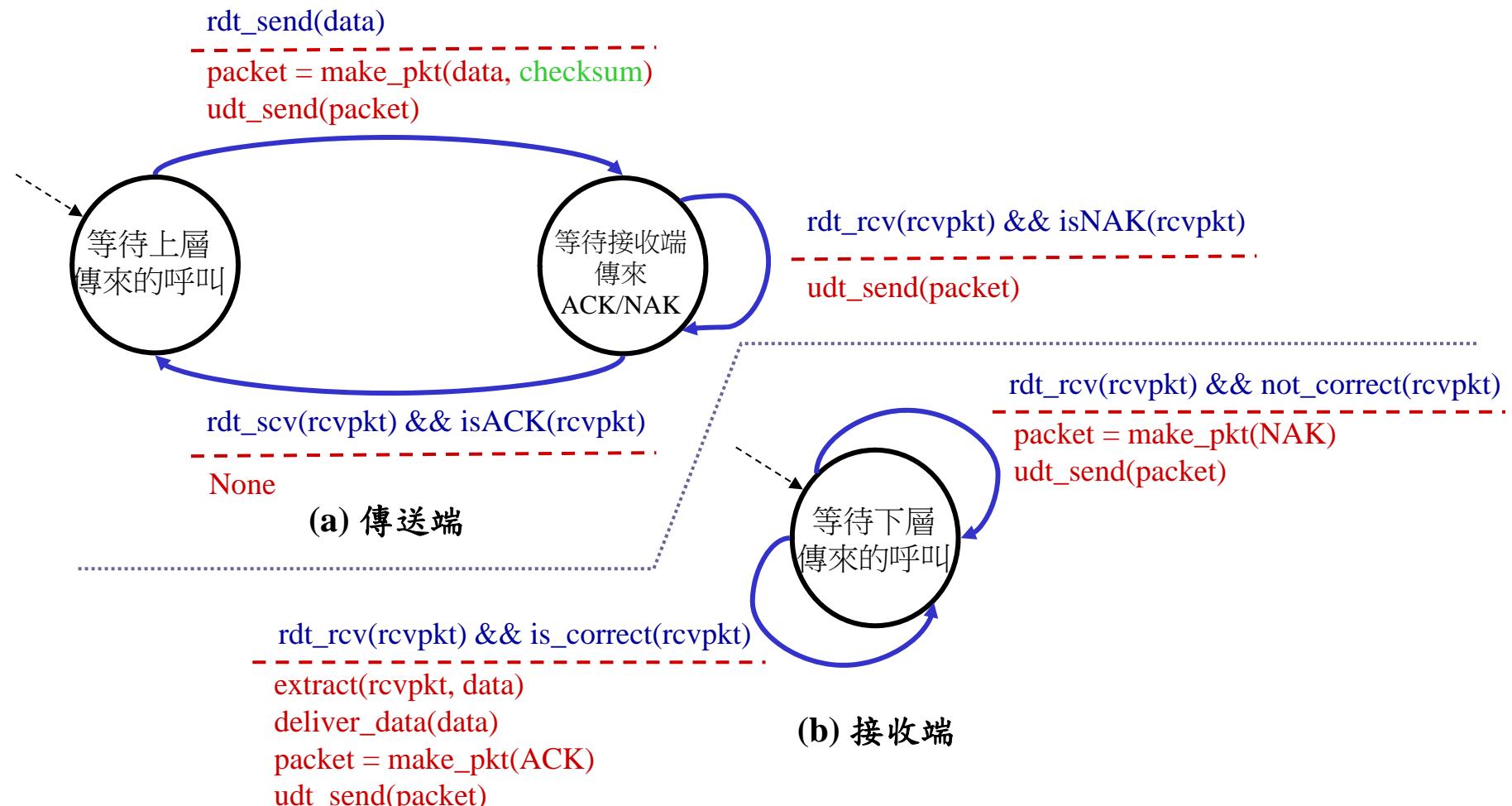
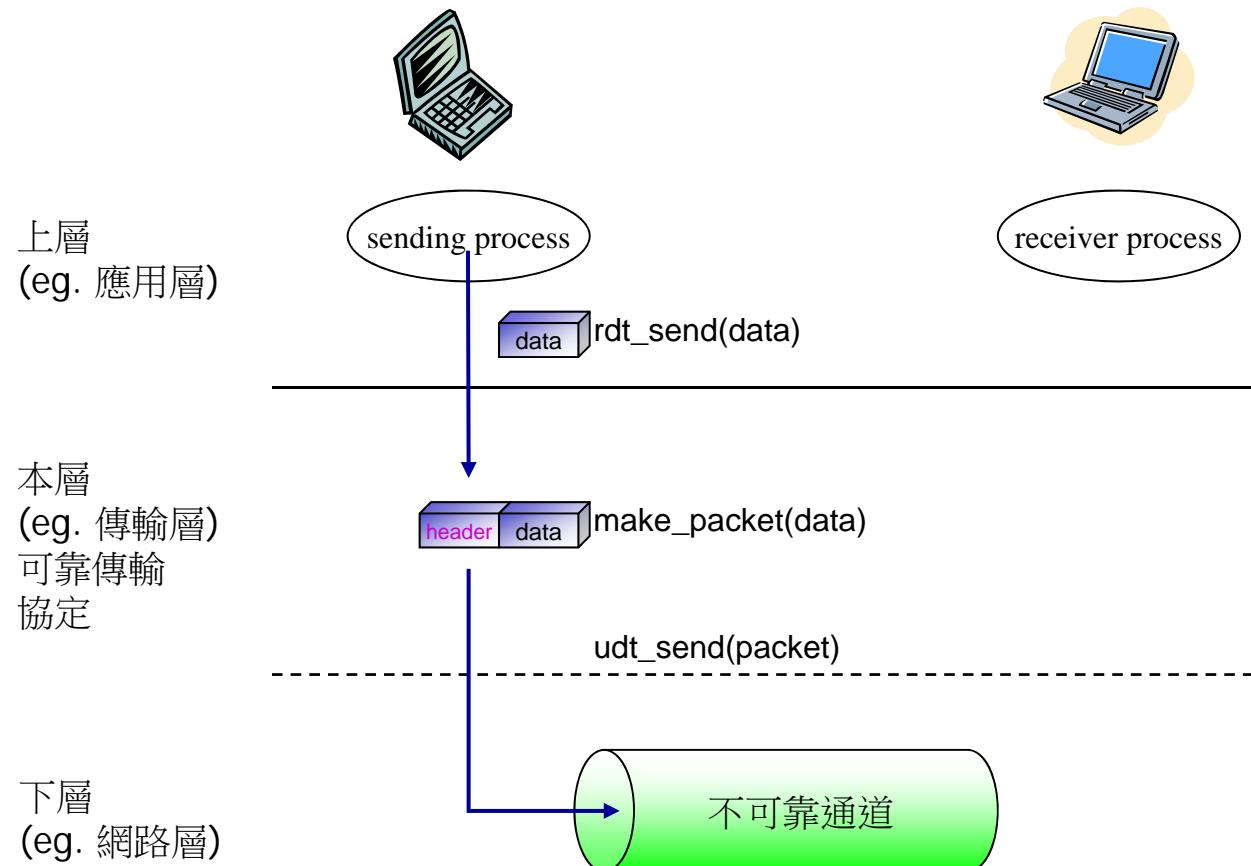


圖3.10 使用會產生位元錯誤通道的協定



■ Rdt2.0 (Cont.)

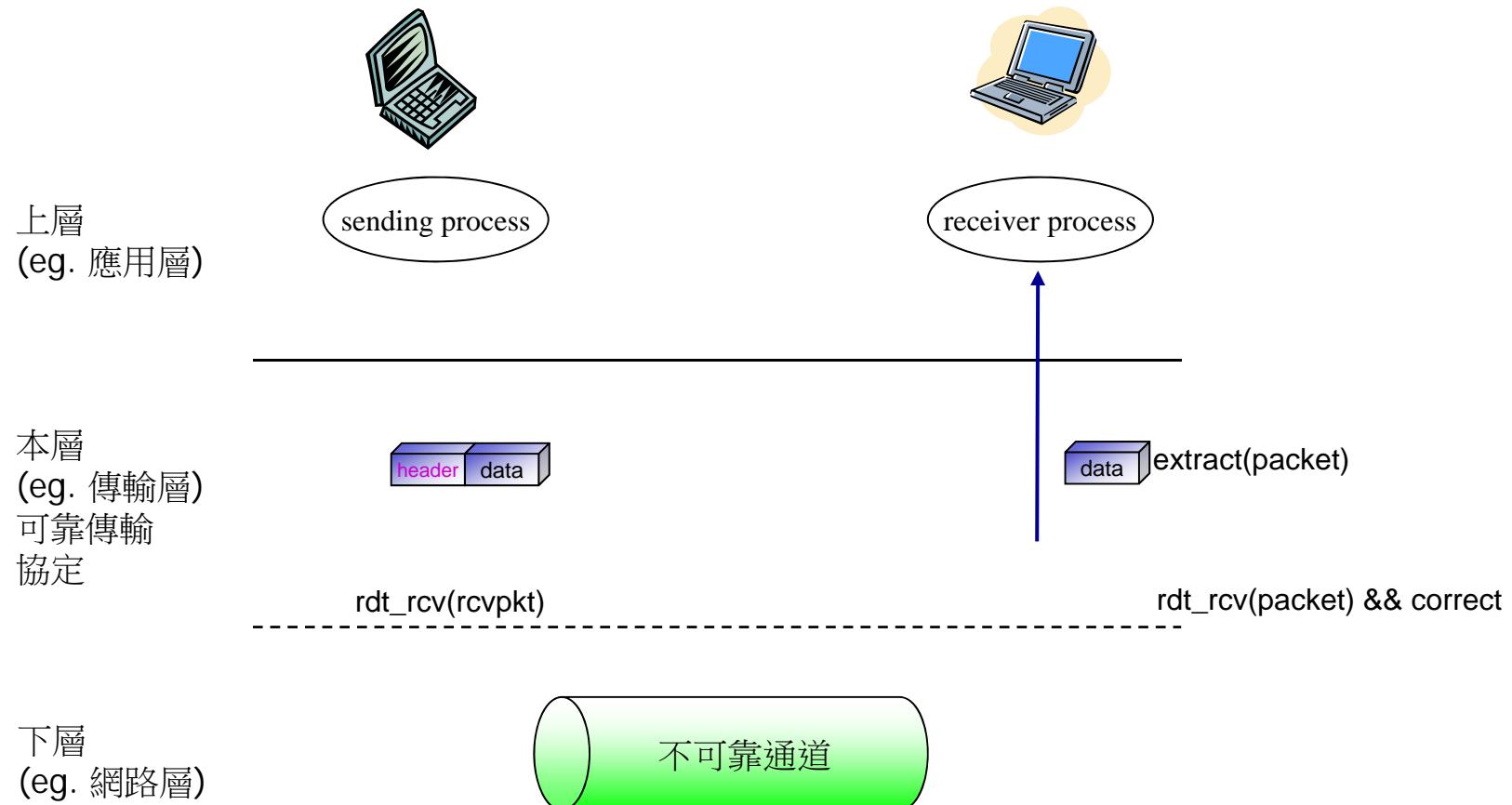
► 沒有錯誤時的運作：





■ Rdt2.0 (Cont.)

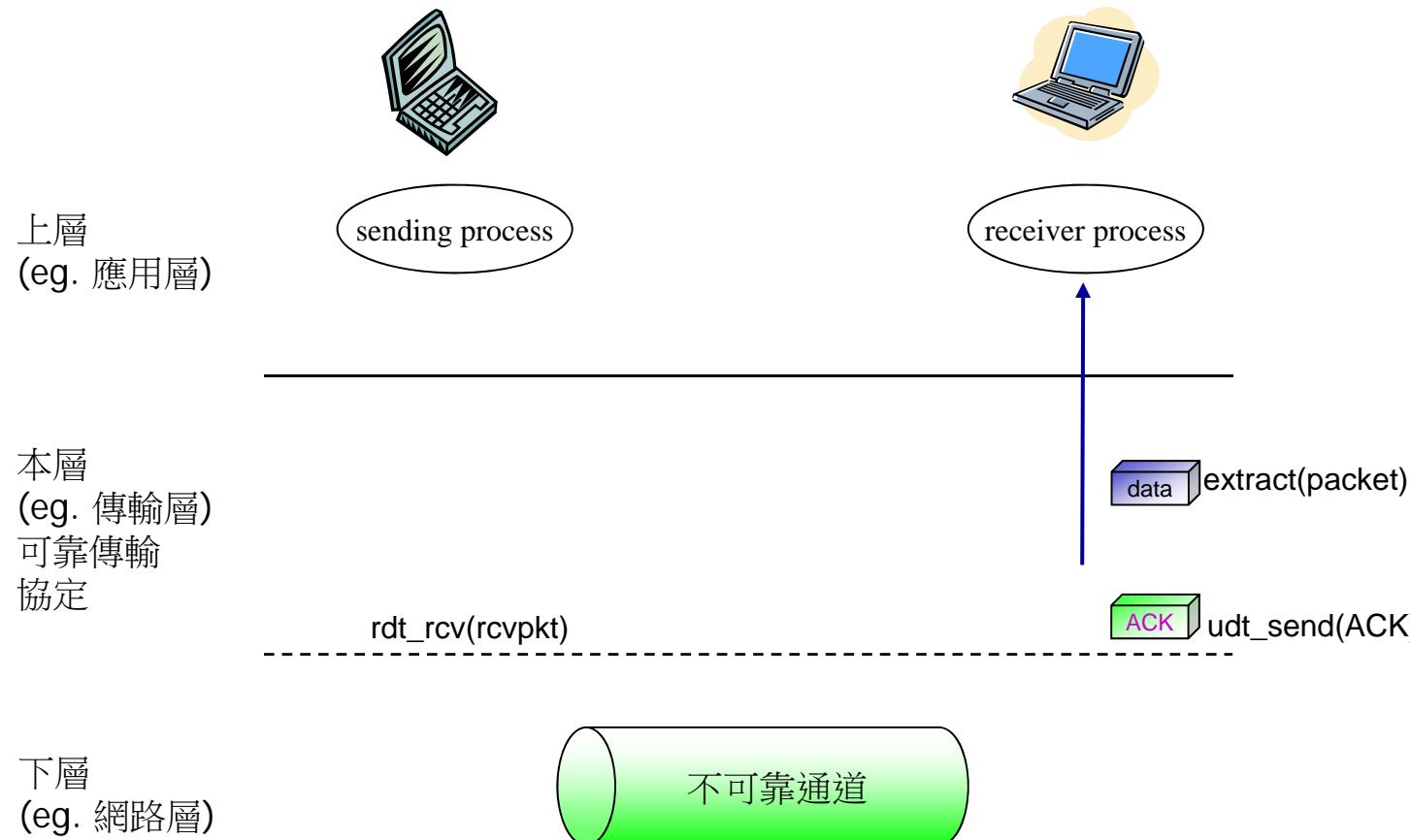
► 沒有錯誤時的運作：(Cont.)





■ Rdt2.0 (Cont.)

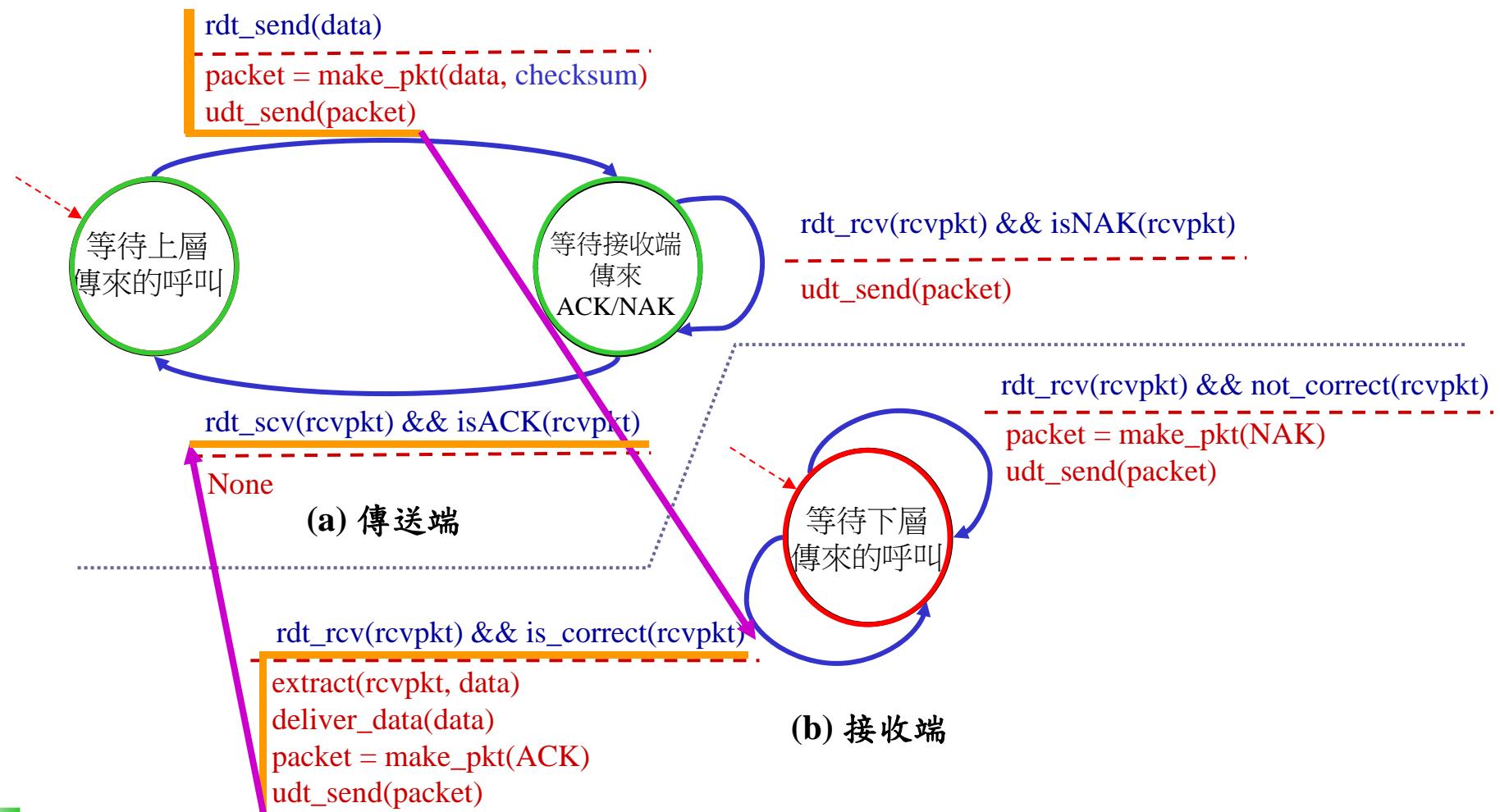
► 沒有錯誤時的運作：(Cont.)





■ Rdt2.0 (Cont.)

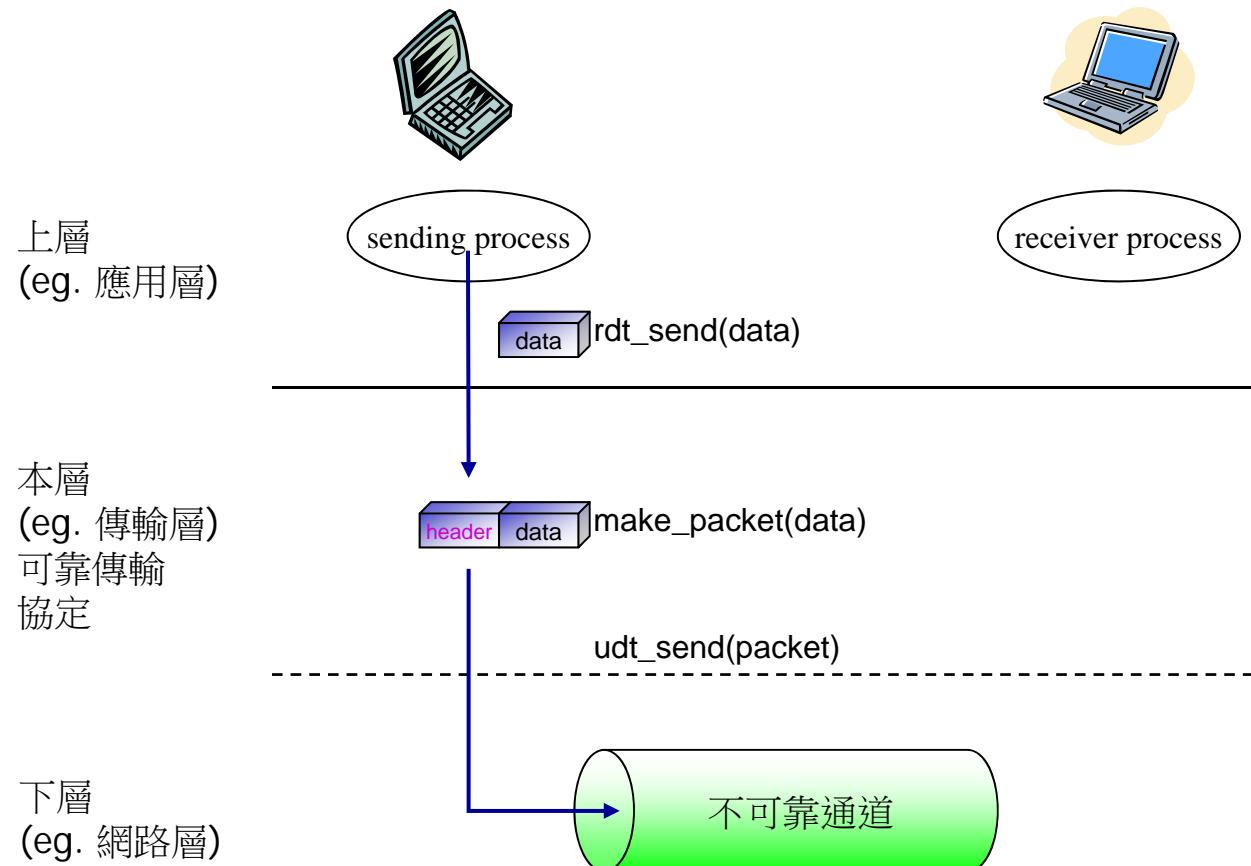
► 沒有錯誤時的運作：(Cont.)





■ Rdt2.0 (Cont.)

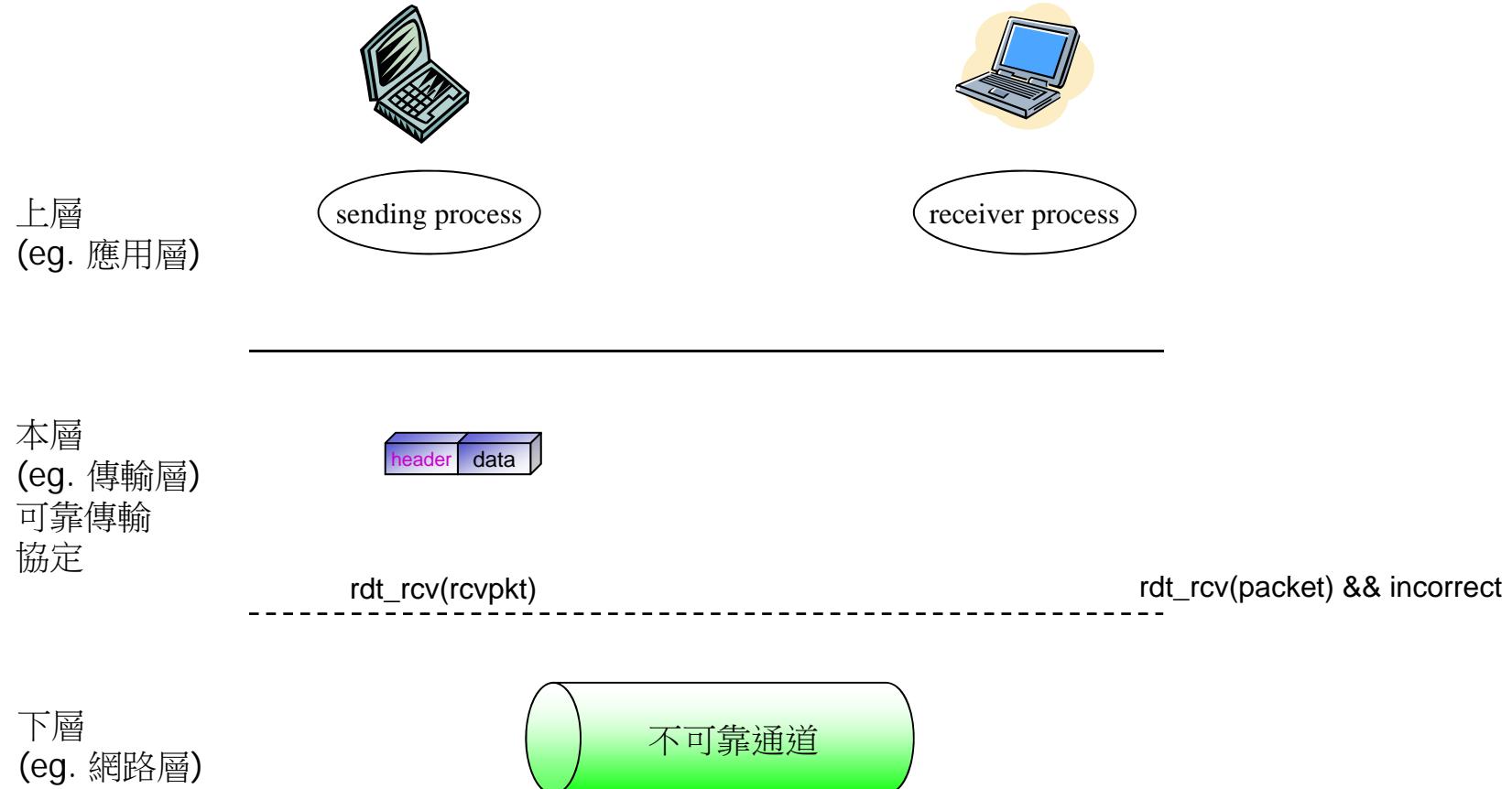
► 發生錯誤時的運作：





■ Rdt2.0 (Cont.)

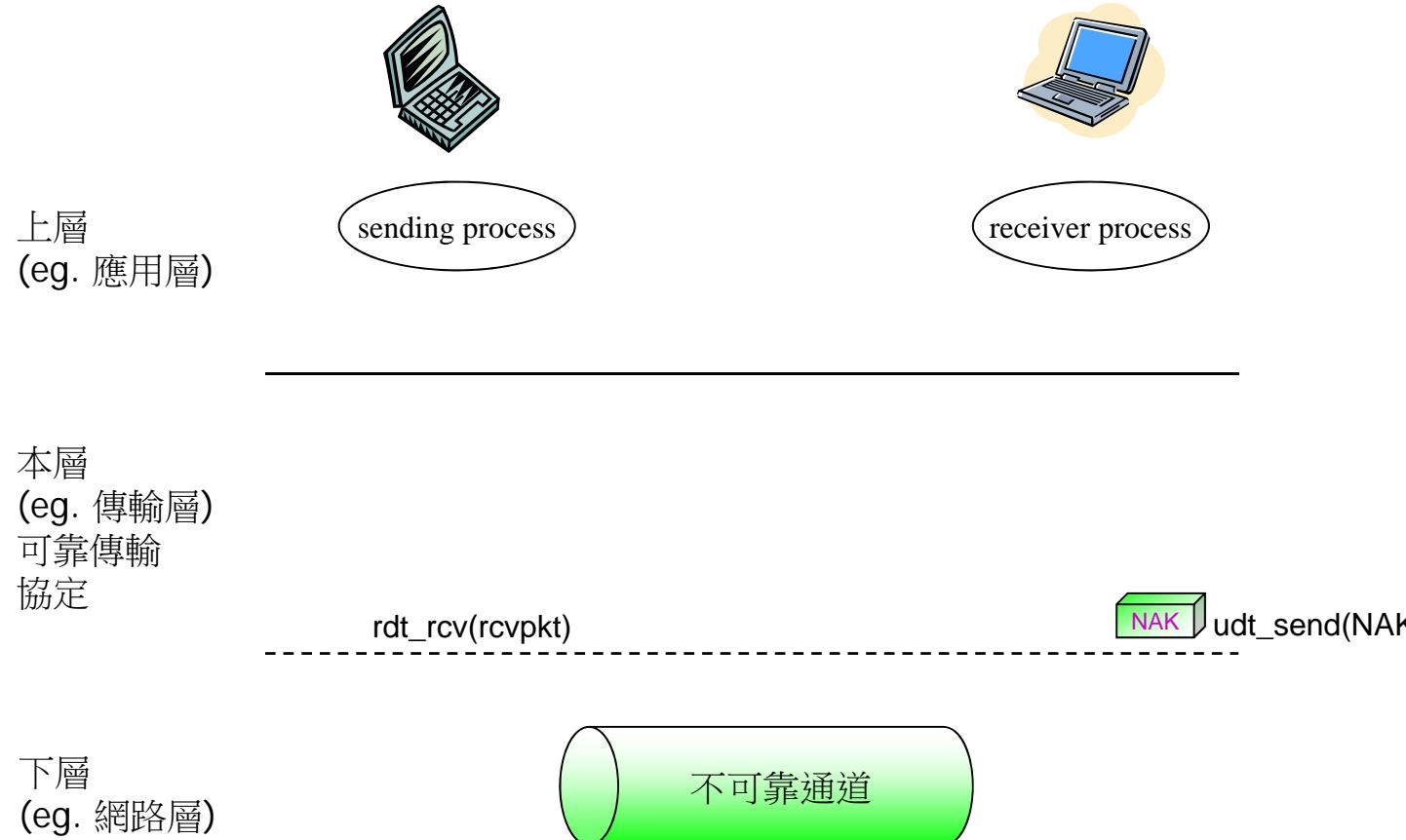
▶ 發生錯誤時的運作：(Cont.)





■ Rdt2.0 (Cont.)

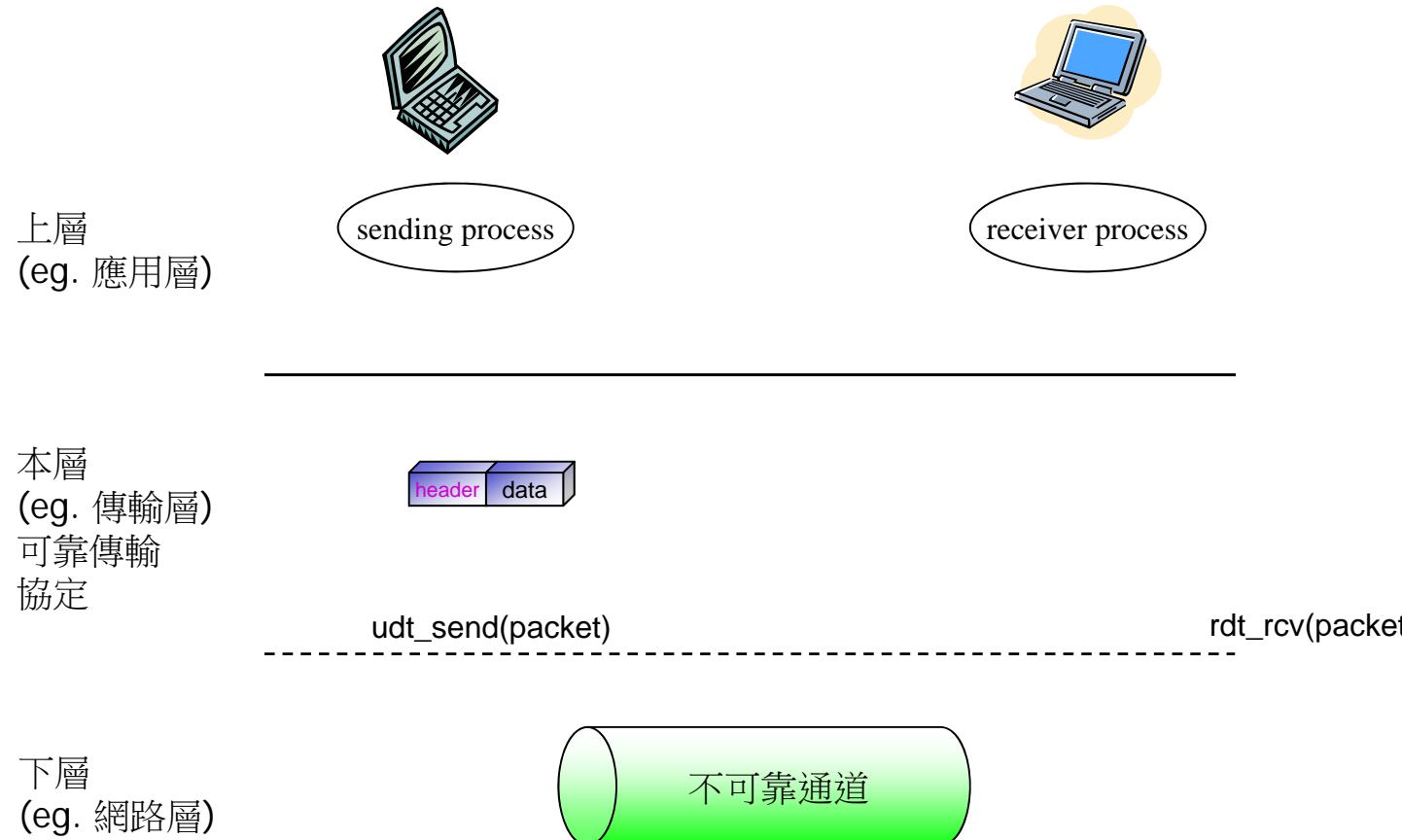
▶ 發生錯誤時的運作：(Cont.)





■ Rdt2.0 (Cont.)

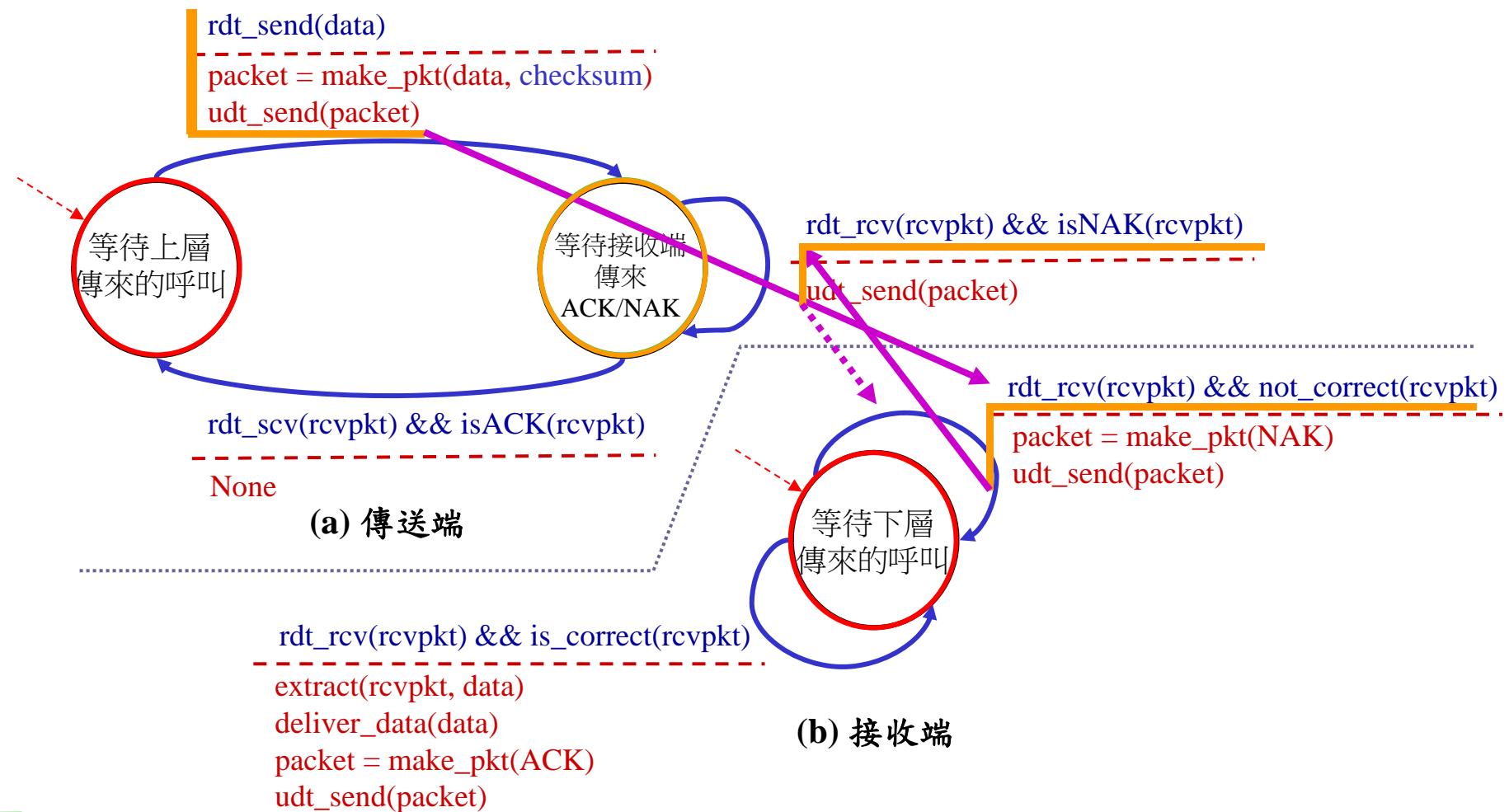
▶ 發生錯誤時的運作：(Cont.)





■ Rdt2.0 (Cont.)

▶ 發生錯誤時的運作：(Cont.)





■ Rdt2.0 (Cont.)

► 致命的缺點：

○ 假如 ACK/NAK 損毀了會如何？

- 傳送端不知道接收端發生了什麼事！
- 沒辦法直接重傳：可能會重複

○ 重複的處理：

- 假如 ACK/NAK 損壞了，傳送端會重新傳送目前的封包
- 傳送端會在每個封包加上 *序號(sequence number)*
- 接收端根據序號刪掉(不往上传) 重複的封包

○ Rdt2.0 為 停止並等待 (stop-and-wait) 協定

- 傳送端傳送一個封包，並等待接收端的回應
- 沒有效率
- 使用 Rdt3.0 改進

⇒ Rdt2.1 & Rdt2.2 (自我學習)

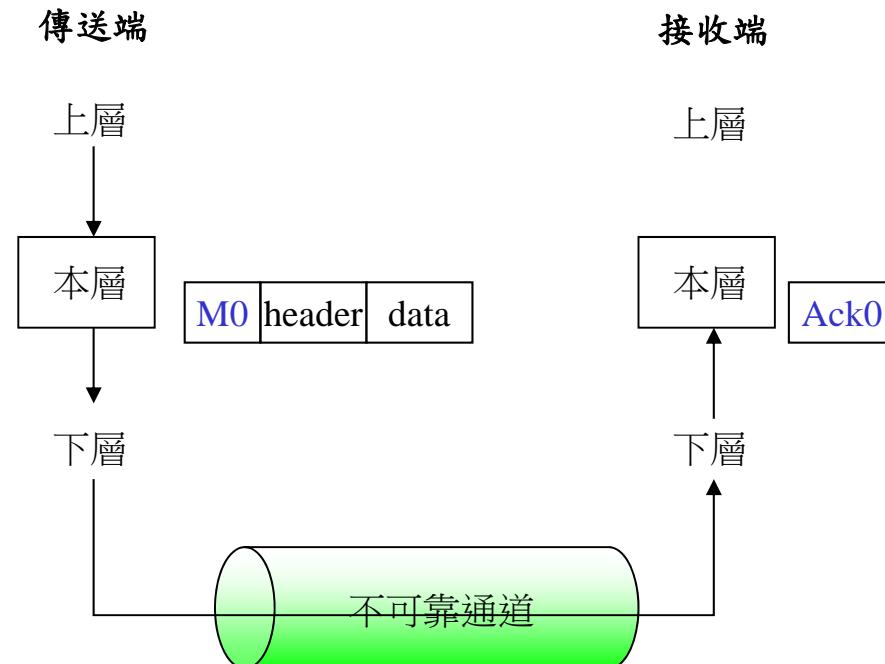


■ Rdt3.0 – 使用會發生錯誤及遺失封包的通道

- ▶ 假設：底層的通道(channel)也可能遺失封包(資料或 ACK)
 - 檢查和 (checksum, 檢查位元錯誤)
 - 封包序號 (2個封包序號, M0、M1)
 - ACK (2個ACK, ACK0、ACK1)
 - 重傳 (使用Timer設定重傳時間)
- ▶ 方法：傳送端等待ACK“合理的”時間
 - 假如在這段時間內沒有收到 ACK，則重傳
 - 假如封包(或 ACK)只是延遲了(沒有遺失)：(過早逾時)
 - 重傳會導致重複，但是封包序號的使用能夠處理這個情況
 - 接收端必須指定確認的封包序號
 - 需要倒數計時器(*countdown timer*)

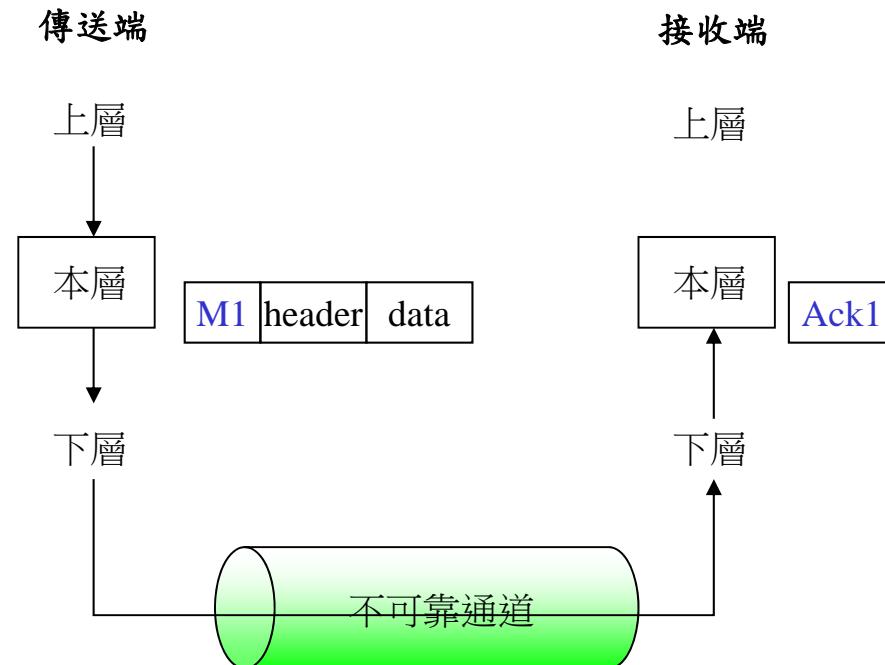


■ Rdt3.0 (alternating-bit protocol)



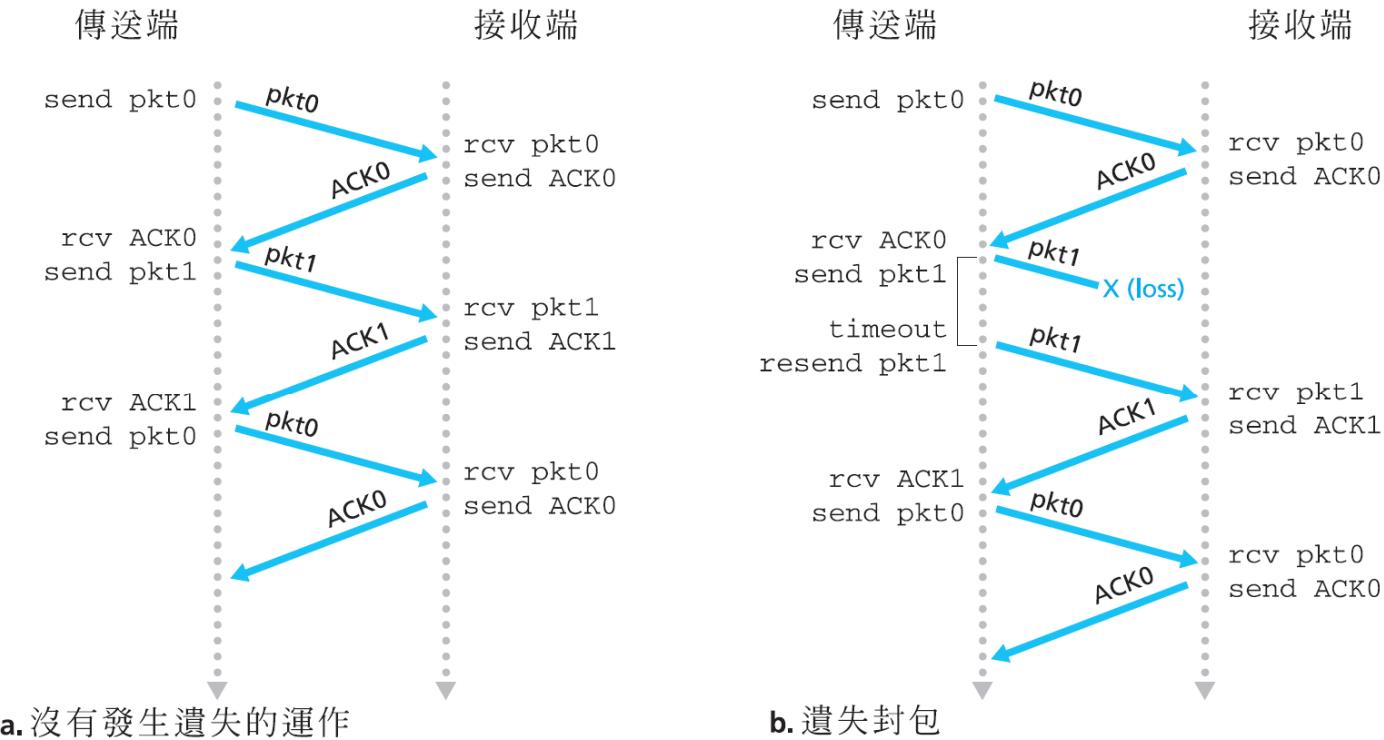


■ Rdt3.0 (alternating-bit protocol)





■ Rdt3.0 (Cont.)

圖3.16 Rdt3.0 的位元變換協定 (alternating-bit protocol, M0/M1交換)



■ Rdt3.0 (Cont.)

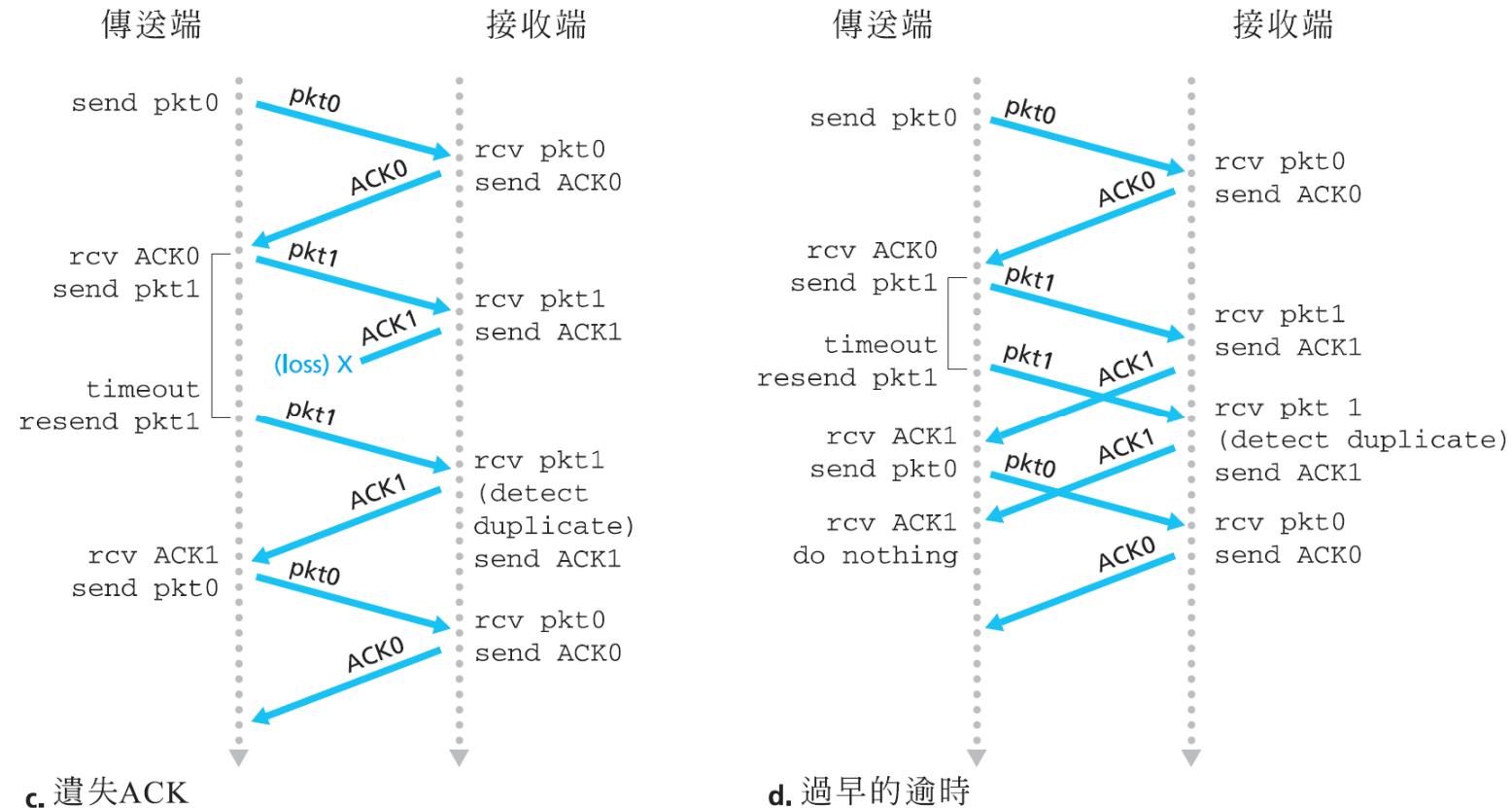


圖3.16 Rdt3.0 的位元轉換協定 (Cont.)

Exercise
3.6

Exercise
3.5



■ Rdt3.0 (Cont.)

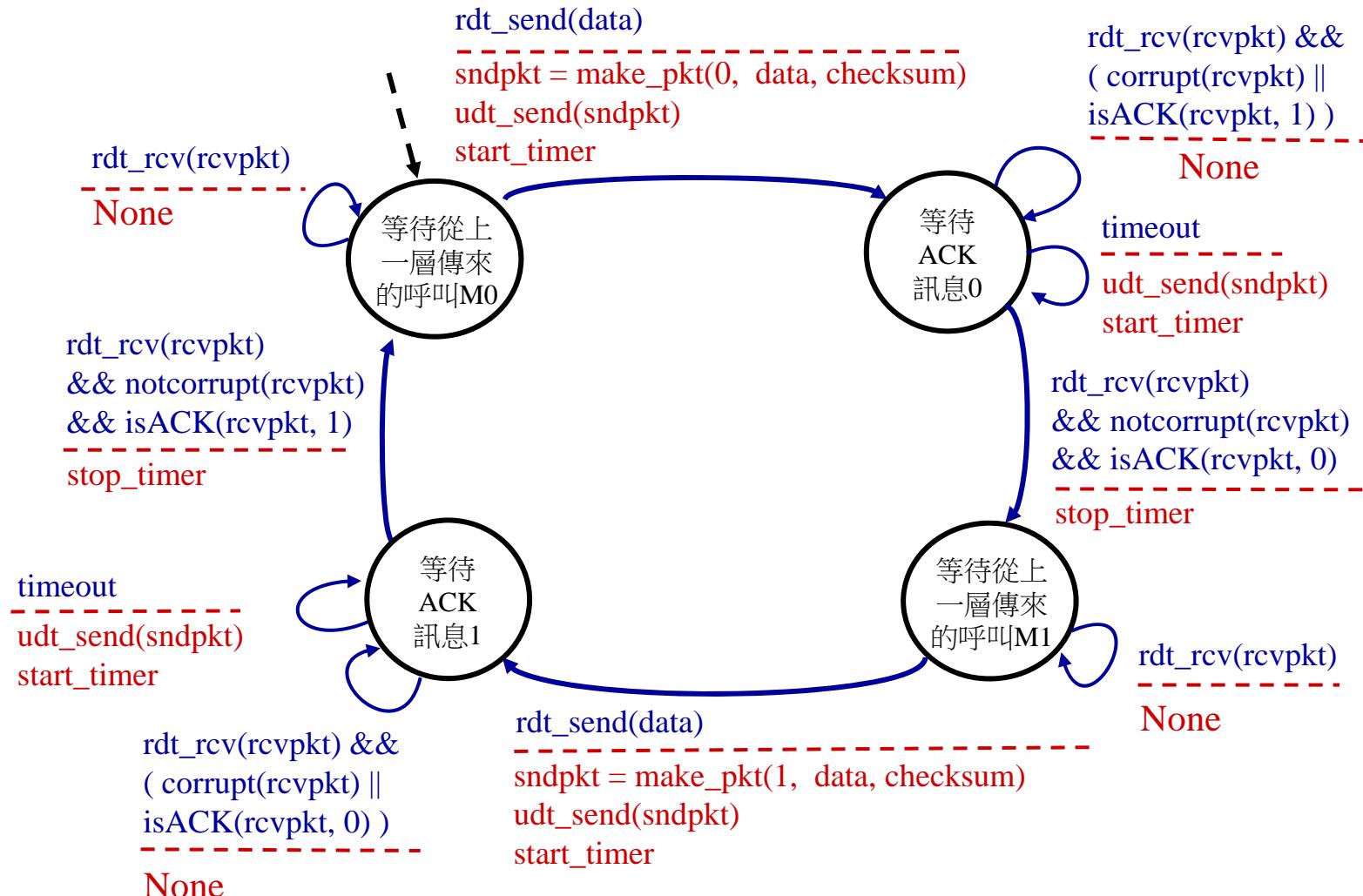


圖3.15 Rdt3.0 傳送端的FSM





■ Rdt3.0 (Cont.)

▶ Rdt3.0 的效能

- Rdt3.0：停止並等待的機制

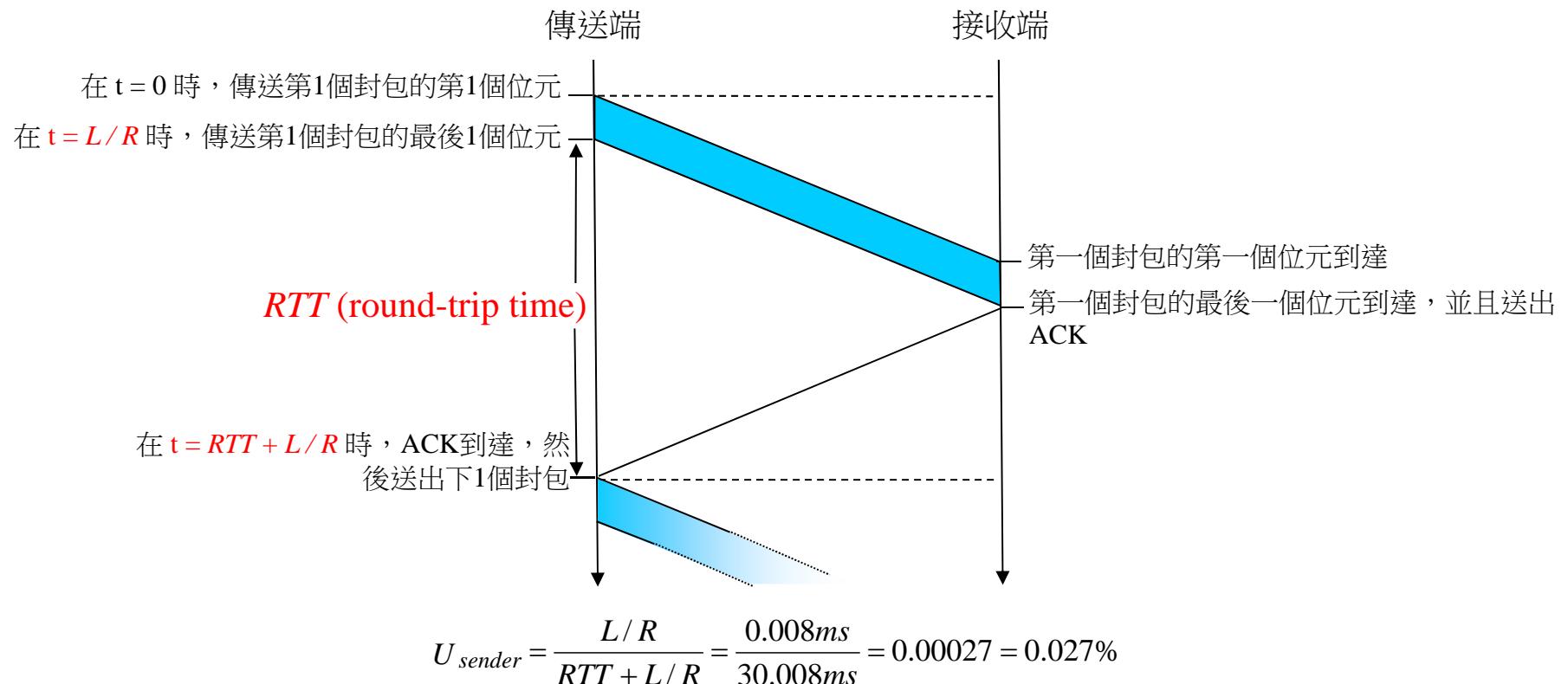


圖3.18(a) Rdt3.0 的停止並等待協定





■ Rdt3.0 (Cont.)

▶ Rdt3.0 的效能 (Cont.)

- Rdt3.0 能夠運作，但是效能很糟
- 範例：1 Gbps 的連結線、15 毫秒(ms) 終端對終端傳遞延遲、1KB 的封包傳送

$$- \text{傳送時間 } d_{trans} = \frac{L(\text{封包長度位元})}{R(\text{傳送速率}, bps)} = \frac{8000 \text{ bits / packet}}{10^9 \text{ bits / sec}} = 8 \mu\text{s}(\text{微秒}) = 0.008 \text{ ms}(\text{毫秒})$$

- 傳送端(通道)使用率(utilization) (傳送端將位元傳入通道的時間比例)

$$U_{sender} = \frac{L/R}{RTT + L/R} = \frac{0.008ms}{30.008ms} = 0.00027 = 0.027\%$$

- 傳送端處於忙碌的時間只有0.027% ⇐ 效能極差
- 網路協定限制了實體資源的使用！

- 解決方案

- 傳送端不使用 stop-and-wait 運作，而允許同時送出多個封包，不需每個封包一一等待確認(ACK)
- Pipeline (3.4.2節)





■ 管線化協定

- ▶ 管線化 (pipelining)：傳送端允許多個、“傳送中的”、還沒有被確認的封包
 - 序號的範圍必須增加
 - 傳送端 和/或 接收端需要暫存器
- ▶ 兩種管線化協定的一般性型態
 - 回溯N (Go-Back N, GBN)
 - 選擇性重複 (Selective Repeat, SR)

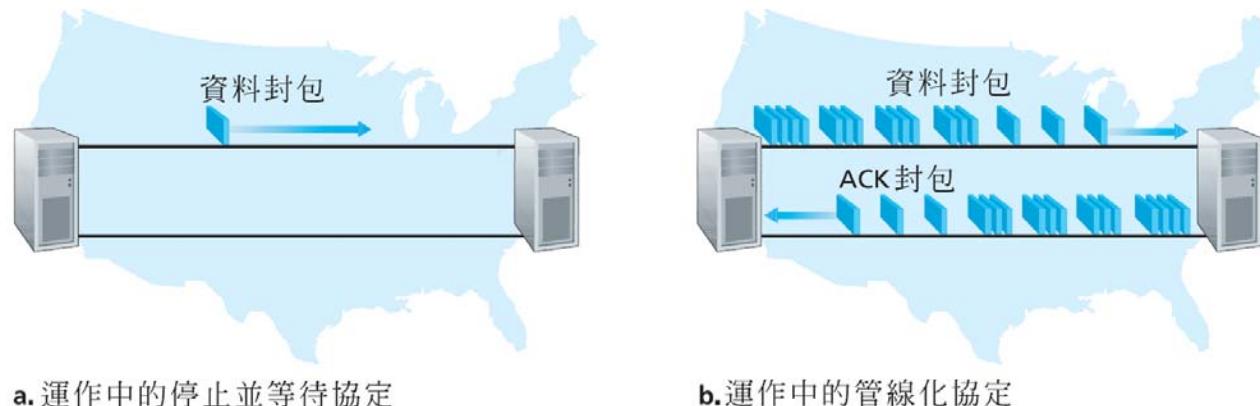


圖3.17 (a)停止並等待協定 (b)管線化協定



■ 管線化 – 增加使用率

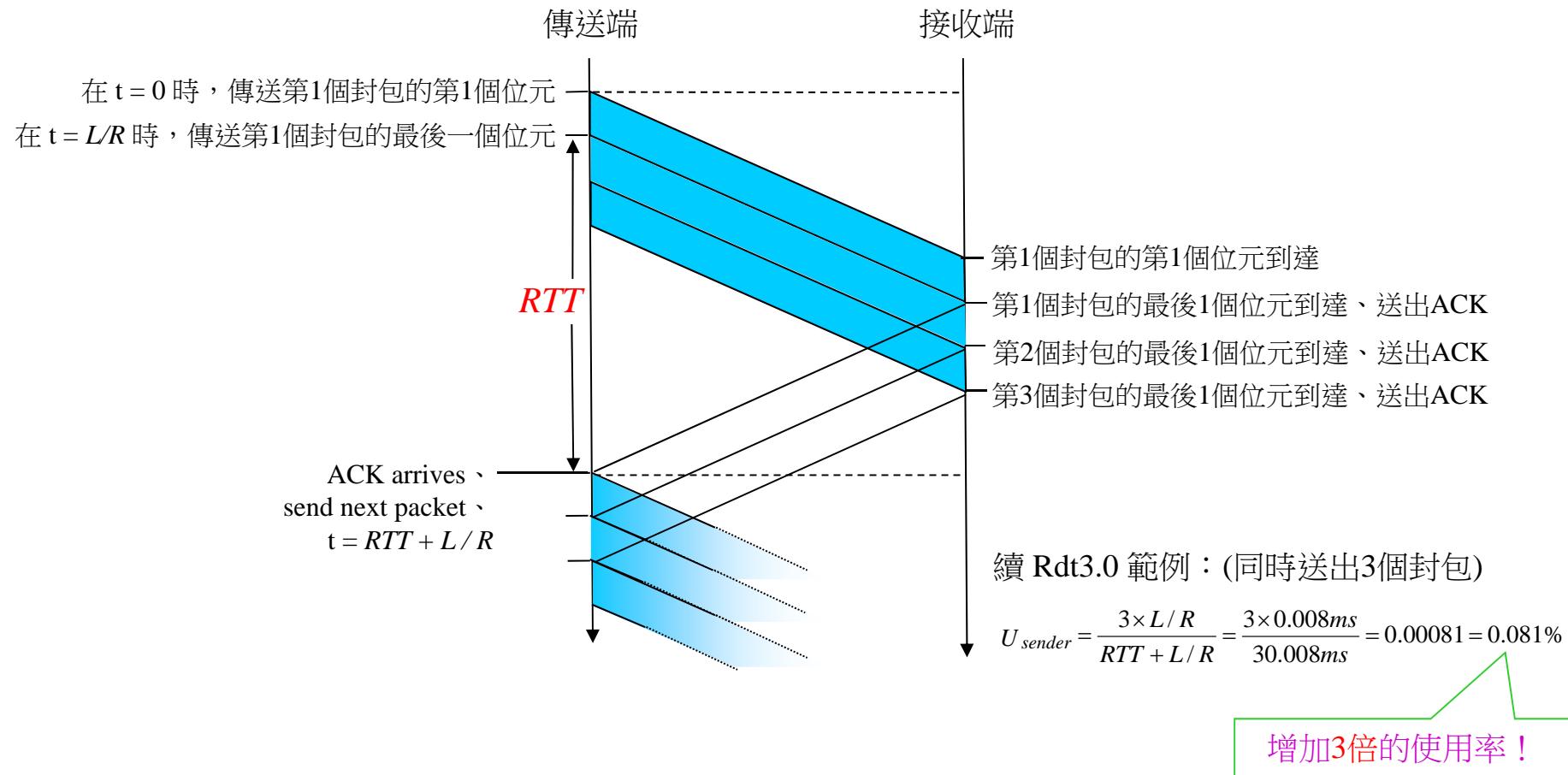


圖3.18(b) 管線化傳輸協定





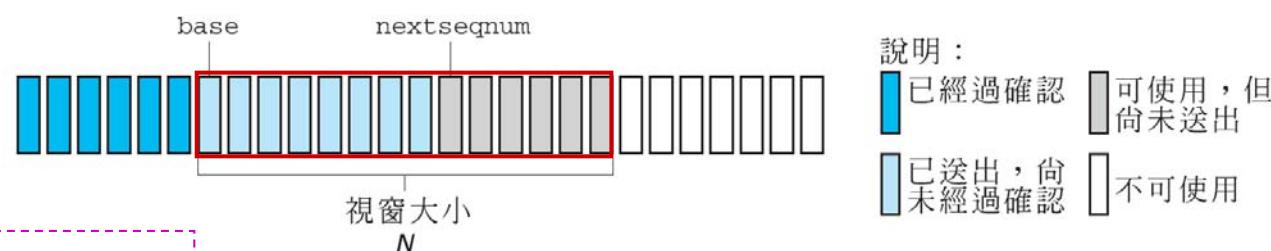
■ 回溯N (GBN)

▶ 概念

- 允許傳送端同時傳送多個封包而不需等待確認
 - 封包標頭的 k -位元序號(封包序號)
- 但管線中未被確認的封包數量不得超過某個可容許的整數 N
- 又稱為 *sliding-window protocol* (滑動視窗協定)

▶ 傳送端製作

- **base**：最久未被確認的封包序號
- **nextseqnum**：下一個要被傳送的封包序號
- 當 $\text{nextseqnum} - \text{base} + 1 \leq N$ ，允許傳送封包
(傳送後 $\text{nextseqnum}++$)；否則，不允許傳送封包
- 當收到確認 base 封包 (ACK base)， $\text{base}++$



GBN Applet:

http://media.pearsoncmg.com/aw/aw_kurose_network_4/applets/go-back-n/index.html

圖3.19 回溯N協定中，傳送端的封包序號觀點



■ 回溯N (Cont.)

► GBN 傳送端的 FSM

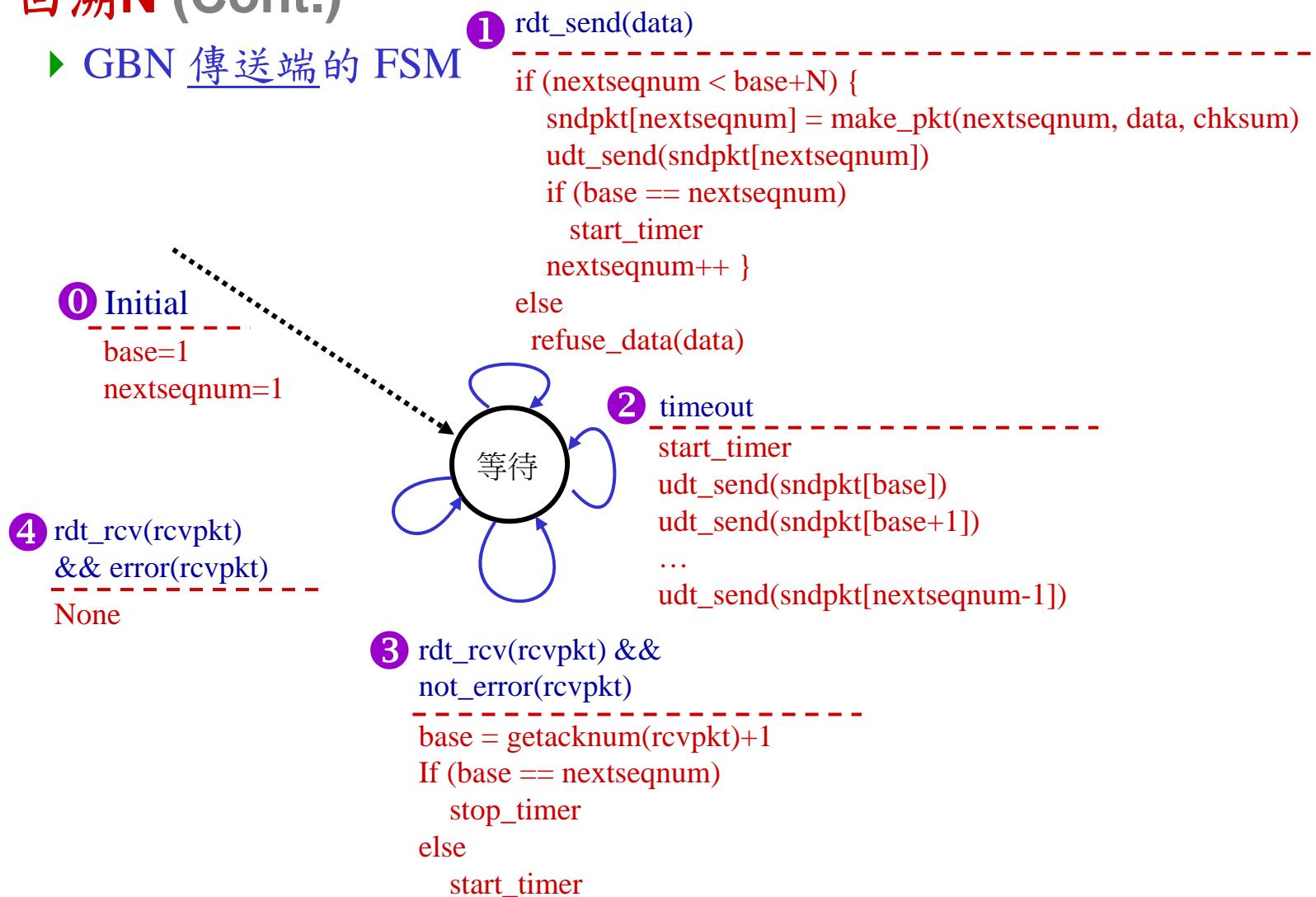


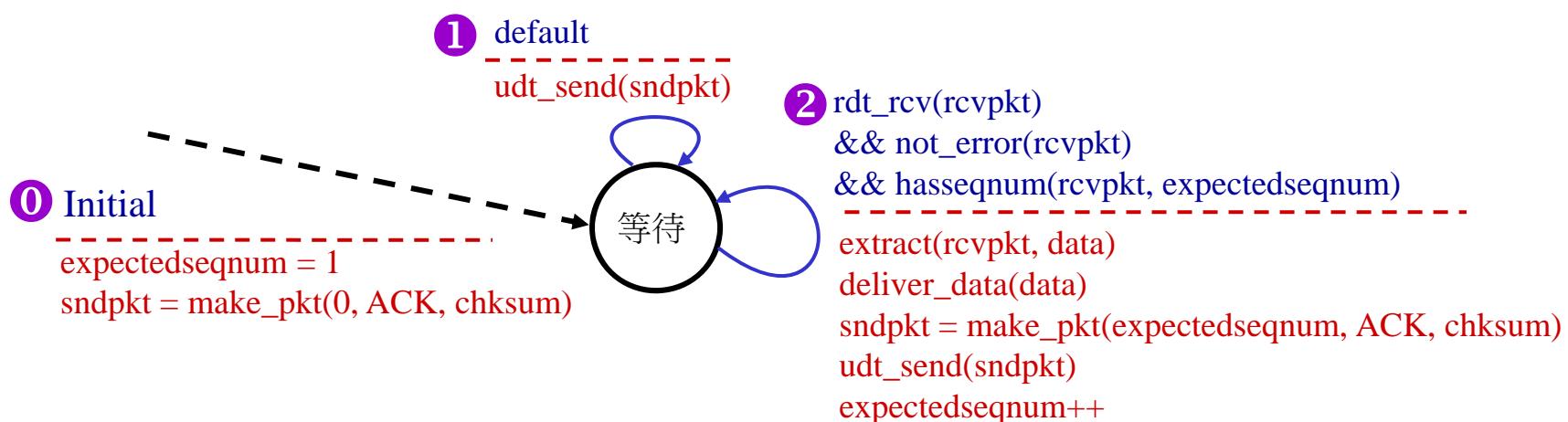
圖3.20 回溯N協定中，傳送端的FSM圖



■ 回溯N (Cont.)

► GBN 接收端的 FSM

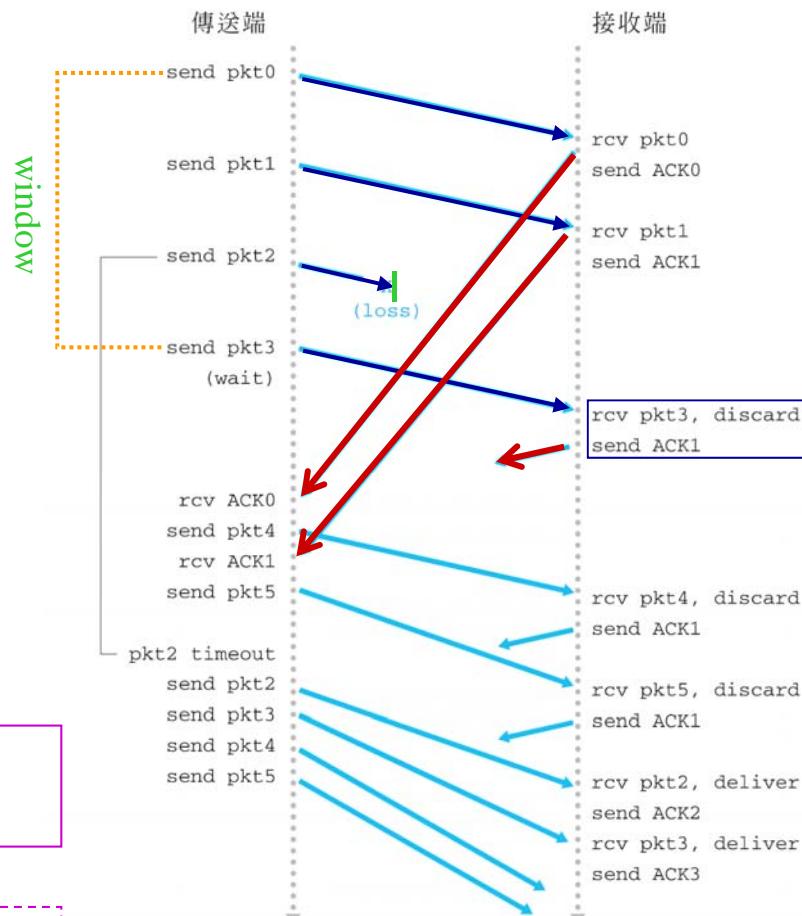
- 只使用ACK：只為接收順序正確的封包傳送 ACK
 - 可能會產生重複的ACK
 - 只需要記住 `expectedseqnum` (期望收到的下一個封包序號)
- 順序不正確的封包：
 - 刪除 (不會暫存) → 接收端沒有暫存器！
 - 重新回應最高的順序正確封包

圖3.21 回溯N協定中，接收端的FSM圖



■ 回溯N(Cont.)

► GBN的運作



TCP採用GBN技術

GBN Applet:

http://media.pearsoncmg.com/aw/aw_kurose_network_4/applets/go-back-n/index.html

圖3.22 回溯N協定的運作



■ 選擇性重複協定 (SR)

▶ GBN的問題

- 單一封包的錯誤，將導致其後的所有封包均需重傳 (重新傳送大量封包)

▶ 選擇性重複

- 傳送端 只重傳沒有收到 ACK 的封包
 - 傳送端針對每一個未確認的封包需要一個計時器
- 接收端分別確認所有正確接收的封包
 - 依需要暫存封包、最終會 依序傳送到上一層
- 傳送端視窗
 - N 個連續的序號
 - 再次、用來限制傳送出去的、未確認的封包序號

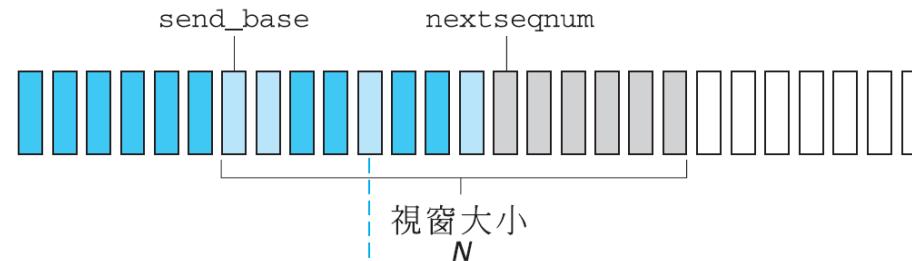
SR Applet:

http://media.pearsoncmg.com/aw/aw_kurose_network_4/applets/SR/index.html



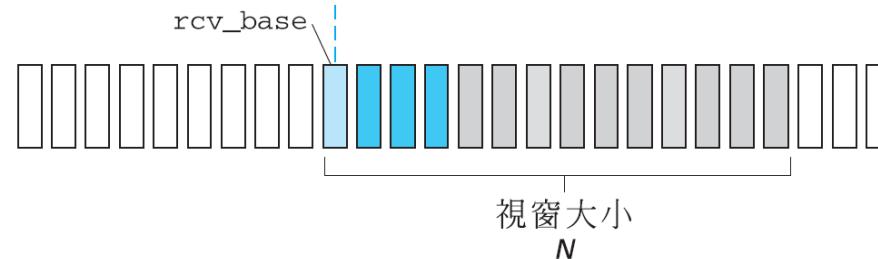
■ 選擇性重複協定(Cont.)

► SR的傳送端、接收端視窗



a. 傳送端對序號的觀點

說明：
■ 已經過確認 ■ 可使用，但尚未送出
■ 已送出，尚未確認 ■ 不可使用



b. 接收端對序號的觀點

說明：
■ 未按順序收到 ■ 可接收的(位於緩衝區)
■ 尚未經過確定 ■ 期望收到，但尚未收到 ■ 不可使用

圖3.23 選擇性重複協定中，傳送端與接收端的封包序號觀點



■ 選擇性重複協定 (Cont.)

► 選擇性重複的運作

傳送端

- 來自上層的資料：
 - ▶ 假如下一個可用的序號在視窗內，則傳送封包
- $\text{timeout}(n)$ ：
 - ▶ 重送封包 n ；重新啟動計時器
 - ▶ 每個封包都有timer
- $\text{ACK}(n)$ 在 $[\text{sendbase}, \text{sendbase}+N]$ 中：
 - ▶ 將封包 n 標示為已收到的
 - ▶ 假如 n 為未確認的封包中最小的，將視窗的 base 往前移到下一個未回應的序號

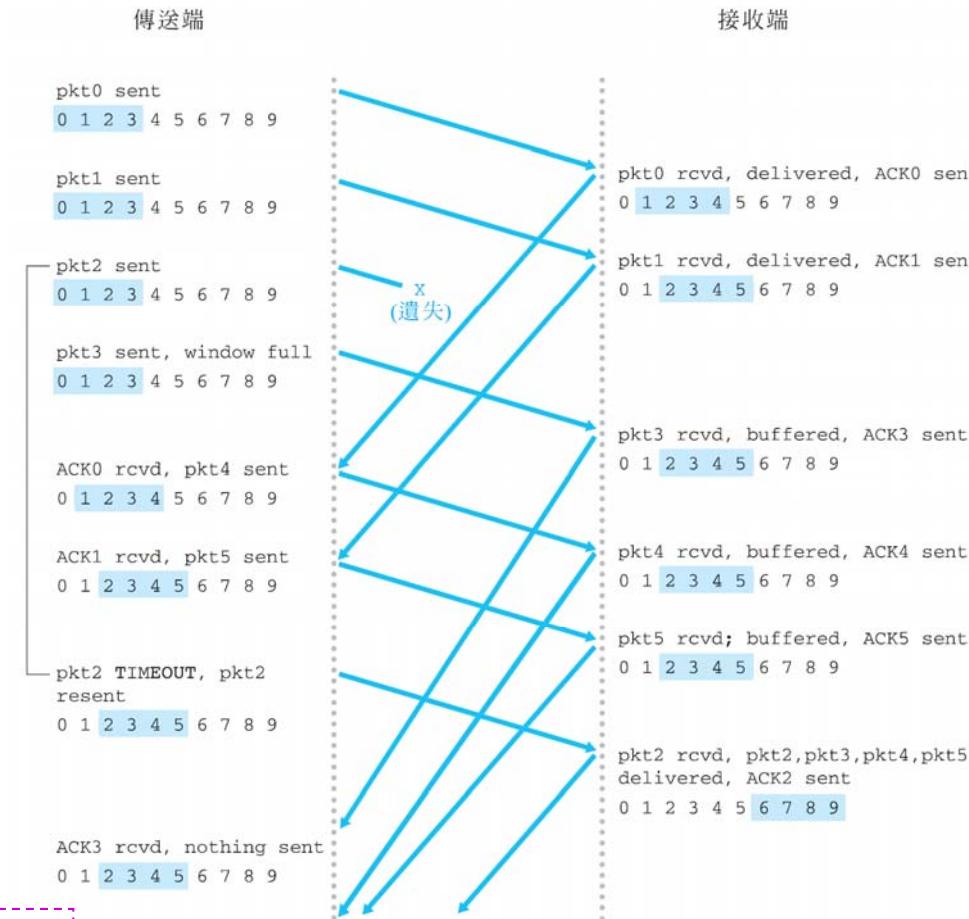
接收端

- 封包 n 在 $[\text{rcvbase}, \text{rcvbase}+N-1]$ 中
 - ▶ 傳送 $\text{ACK}(n)$
 - ▶ 不正確的順序：暫存區
 - ▶ 正確順序：遞送(也遞送暫存區內順序錯誤的封包)、將視窗前進到下一個未接收的封包
- 封包 n 在 $[\text{rcvbase}-N, \text{rcvbase}-1]$ 中
 - ▶ $\text{ACK}(n)$
 - ▶ 否則：
 - ▶ 忽略該封包



■ 選擇性重複協定 (Cont.)

► 選擇性重複的運作 (Cont.)



SR Applet:

http://media.pearsoncmg.com/aw/aw_kurose_network_4/applets/SR/index.html

圖3.26 選擇性重複協定的運作



■ 選擇性重複協定(Cont.)

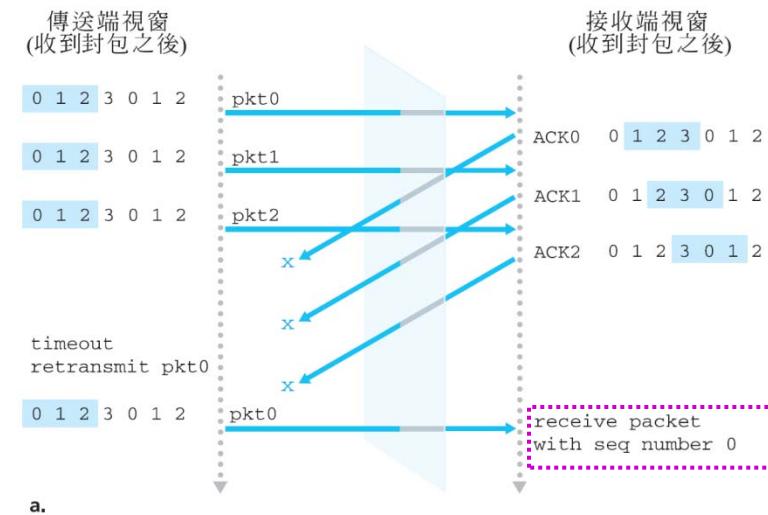
► 選擇性重複的困境

○範例：

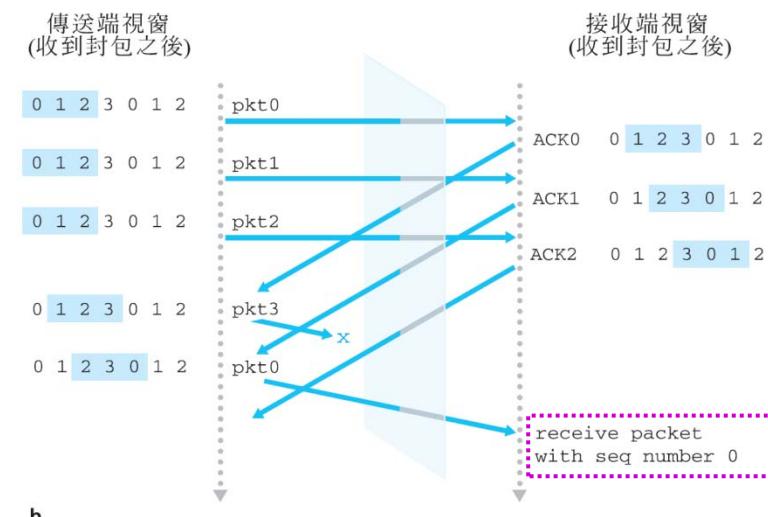
- 序號：0、1、2、3
- 視窗大小=3

○接收端無法分辨(a)(b)兩種情況的差別！

○不正確地重新傳送重複的資料
如同(a)



a.



b.

圖3.27 選擇性重複協定中，接收端無法判斷是新的封包還是重傳的封包



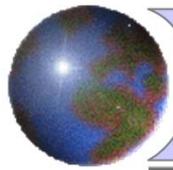


■ 可靠資料傳輸機制的總整理

► 表3.1 可靠資料傳輸機制及其用途的總整理

機制	用途、註解
Checksum	用來偵測傳輸封包的位元錯誤。
Timer	用來進行封包逾時/重送，可能因為該封包(或其ACK)遺失在通道中。
Sequence number	用來對從傳送端流向接收端的資料封包進行循序編號。收到的封包序列號碼若不連續，則接收端可偵測出封包的遺失。
Acknowledge	接收端用此訊息封包告知傳送端某個(組)封包已被正確接收。
Negative acknowledge	接收端用此訊息封包告知傳送端某個封包並未被正確接收。
Sliding-window, pipelining	傳送端可能會被限制只能傳送序號在某個範圍的封包。藉由容許多個正在傳輸但未被確認的封包傳送端的通道使用率將比停止並等待操作模式增加。





第三章 導覽流程



3.1 傳輸層服務

3.2 多工和解多工

3.3 非預接式傳輸 – UDP

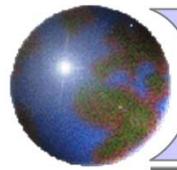
3.4 可靠資料傳輸的原理

3.5 連線導向傳輸 – TCP

▶ 區段結構、可靠的資料傳輸、流量控制、連線管理

3.6 壓塞控制的原理

3.7 TCP 壓塞控制



3.5 連線導向傳輸 – TCP



■ TCP綜觀

- ▶ TCP定義於 RFCs 793、1122、1323、2018、2581
- ▶ TCP具備的功能
 - 點對點：
 - 一個傳送端、一個接收端
 - 可靠的、有順序的位元組串流：
 - 沒有“訊息界線”
 - 管線化：
 - TCP壅塞控制和流量控制，可設定視窗大小
 - 傳送端和接收端暫存器(緩衝區, buffer)
 - 全雙工資料傳輸：(full-duplex service)
 - 同一個連結中，雙向的資料流
 - MSS：最大資料區段大小 (maximum segment size)
 - 連線導向：
 - 交握程序 (控制訊息的交換) 在資料開始交換之前，設定傳送端和接收端的狀態
 - 流量控制：
 - 傳送端傳送的資料量不會超過接收端可接收的資料量



■ TCP綜觀 (Cont.)

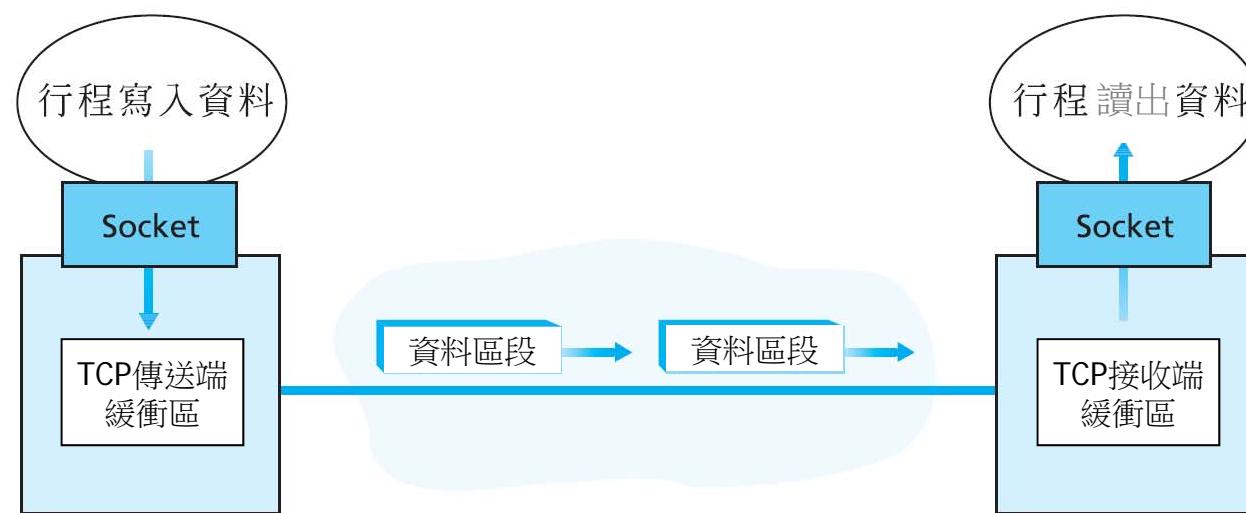


圖3.28 TCP 的傳送端與接收端的緩衝區 (buffer)





■ TCP 資料區段結構 (segment structure)

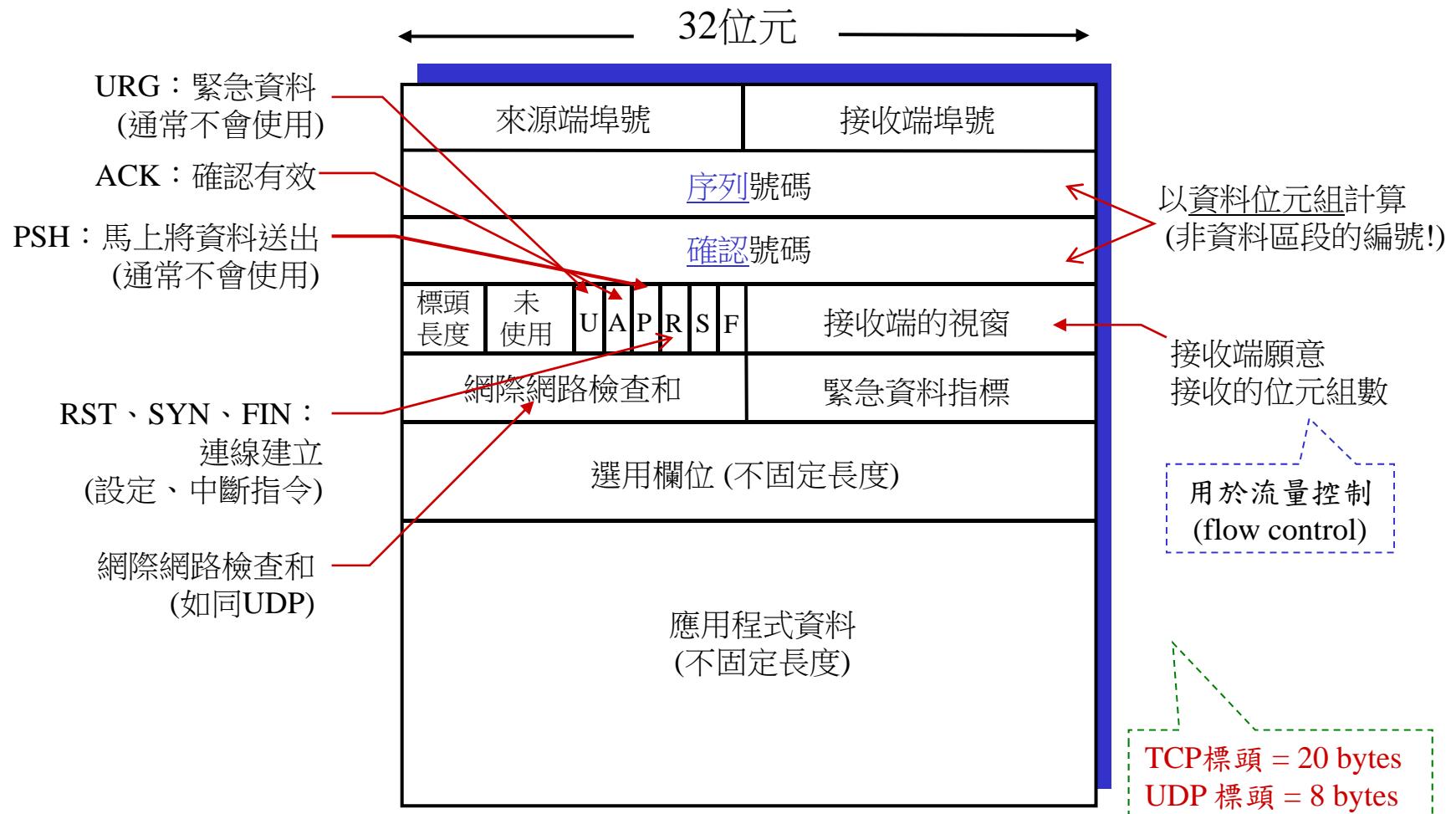


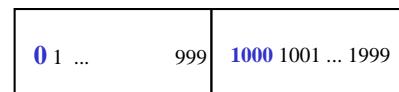
圖3.29 TCP 區段的結構 (segment structure)



■ TCP序列號碼與確認號碼

► 序列號碼：

- 資料區段中，第一個位元的位元組串流 “編號” (非封包序號)
- 例如：主機A將一資料串流(500,000 bytes的檔案)透過TCP傳送給主機B，其中TCP的MSS=1,000 bytes
 - 此資料串流的第1個byte被編號為0
 - 第1個TCP區段的序列號碼 = 0
 - 第2個TCP區段的序列號碼 = 1000



– See 圖 3.30

► 確認號碼：

- 主機A期待從主機B收到的下一個位元組序列號碼(A到B的資料區段) \Rightarrow 主機正在等待的下一個資料位元組的序號
- 累積式確認

$\Rightarrow Q$: 接收端如何處理順序不正確的資料區段

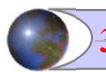
Ans: TCP 規格中未限制、取決於程式開發者



■ TCP序列號碼與確認號碼 (Cont.)

▶ 確認號碼 : (Cont.)

- 主機A傳送資料給主機B
- 主機A已從主機B收到序號從0到535的所有位元組
- 主機A正要傳送一筆TCP區段給主機B \Rightarrow 主機A傳送的TCP區段中的「確認號碼」為536



■ TCP序列號碼與確認號碼 (Cont.)

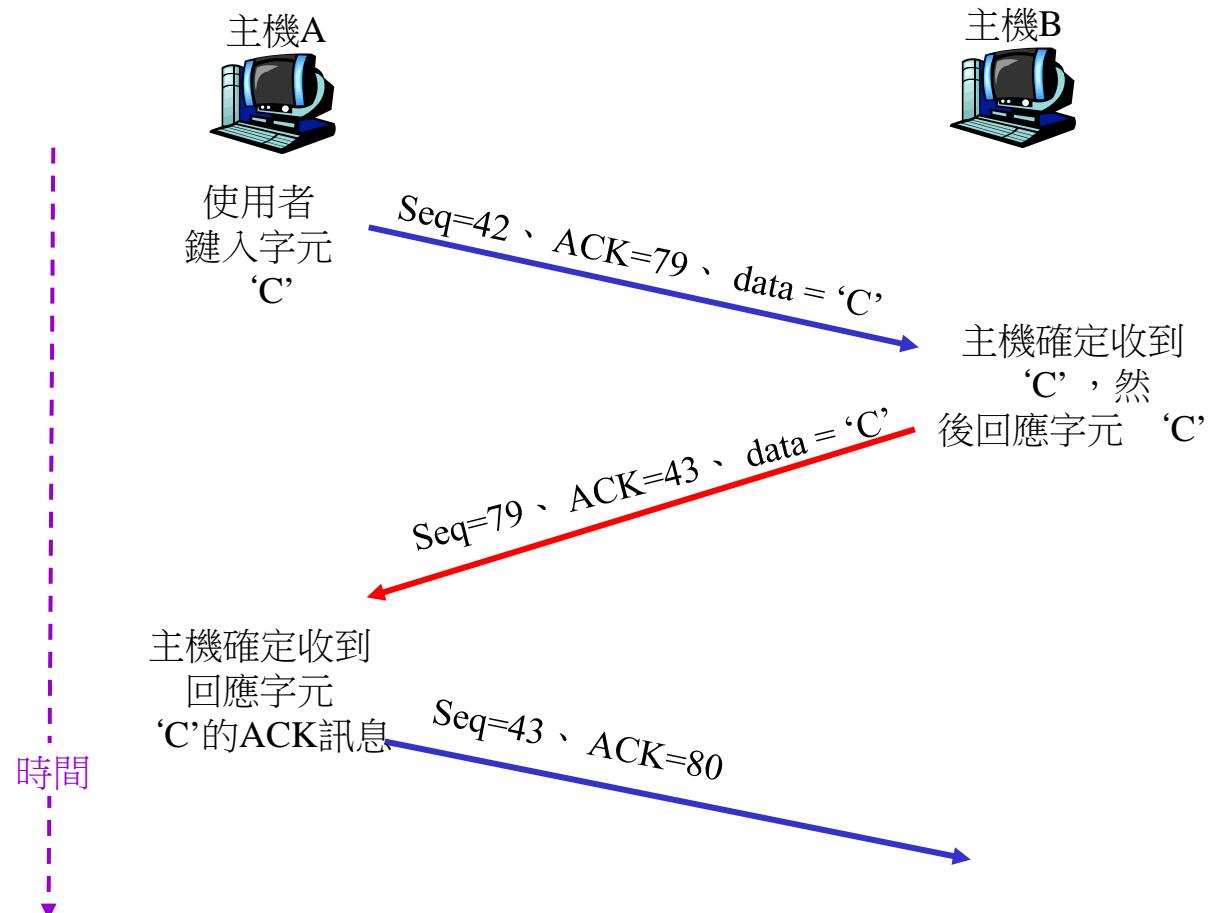


圖3.31 簡單的TCP Telnet範例

Ex 3.12
Ex 3.11



■ TCP 來回傳遞時間以及逾時

► Q: 如何設定 TCP 的逾時值?

- 比 RTT 長
 - 但是 RTT 是不固定的
- 太短：過早逾時
 - 不需要重新傳送
- 太長：太晚對資料區段遺失作出反應

► Q: 如何估計來回傳遞時間 (RTT)? (EstimatedRTT)

- 樣本RTT (SampleRTT)：測量資料區段傳送出去到收到確認所需的时间
 - 忽略重傳
- 樣本RTT會有所變動、我們想要讓預估的RTT“更平滑”
 - 將好幾個最近的測量值做平均、而非目前的樣本RTT



■ TCP 來回傳遞時間以及逾時 (Cont.)

► TCP更新估算RTT值的公式

- $\text{EstimatedRTT} = (1 - \alpha) \cdot \text{EstimatedRTT} + \alpha \cdot \text{SampleRTT}$
 - 指數加權移動平均值
 - 過去樣本的影響將以指數速率減少
 - 建議值： $\alpha = 0.125$



■ TCP 來回傳遞時間以及逾時 (Cont.)

► 範例 RTT 估計：

RTT: gaia.cs.umass.edu to fantasia.eurecom.fr

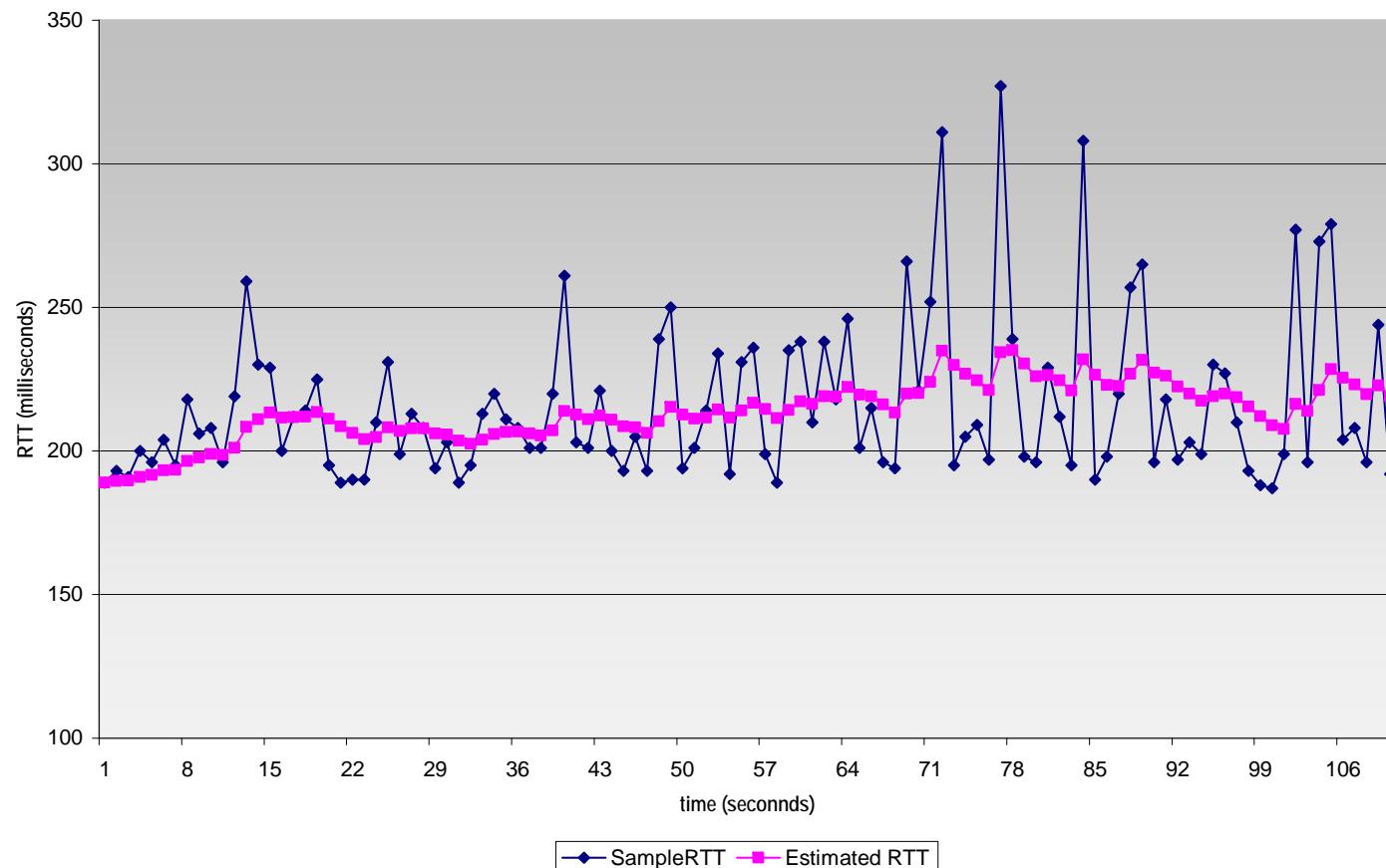


圖3.32 RTT取樣與RTT估算值



■ TCP 來回傳遞時間以及逾時 (Cont.)

► 設定逾時間隔

- 首先估計 SampleRTT 與 EstimatedRTT 的差距：

$$\text{DevRTT} = (1 - \beta) \cdot \text{DevRTT} + \beta \cdot |\text{SampleRTT} - \text{EstimatedRTT}|$$

(β 的建議值=0.25)

- 接著設定逾時間隔：

$$\text{TimeoutInterval} = \text{EstimatedRTT} + 4 \cdot \text{DevRTT}$$



■ TCP 可靠的資料傳輸

▶ What?

- TCP 在 IP 的不可靠服務上建立 rdt 服務

▶ 簡化的TCP傳送端：

- 忽略重複的ack
- 忽略流量控制、壅塞控制



■ TCP 可靠的資料傳輸 (Cont.)

► 簡化的TCP傳送端：(Cont.)

- 從應用程式收到資料：
 - 產生含有序號的資料區段
 - 序號是資料區段中、第一個資料位元組的位元組串流編號
- 假如計時器尚未執行、啟動計時器 (將計時器想成與最久的未確認資料區段有關)
- 逾時時間：TimeOutInterval
- 逾時：
 - 重新傳送導致逾時的資料區段
 - 重新啟動計時器
- 收到Ack：
(假如確認為之前未確認的資料區段)
 - 更新已確認的狀態
 - 假如還有未確認的資料區段、重新啟動計時器



■ TCP 可靠的資料傳輸 (Cont.)

► 簡化的TCP傳送端：(Cont.)

```
NextSeqNum = InitialSeqNum
SendBase = InitialSeqNum

loop (forever) {
    switch(event)

    event : data received from application above
        create TCP segment with sequence number NextSeqNum
        if (timer currently not running)
            start timer
        pass segment to IP
        NextSeqNum = NextSeqNum + length(data)

    event : timer timeout
        retransmit not-yet-acknowledged segment with
            smallest sequence number
        start timer

    event : ACK received 、 with ACK field value of y
        if (y > SendBase) {
            SendBase = y
            if (there are currently not-yet-acknowledged segments)
                start timer
        }

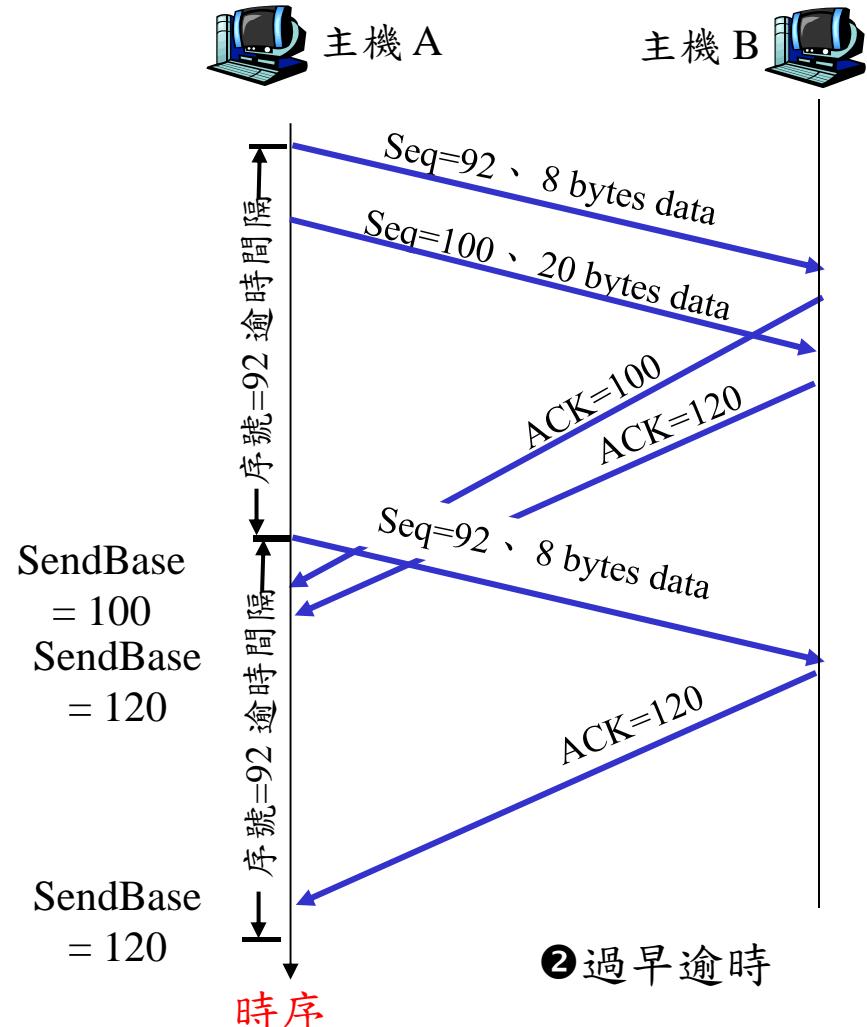
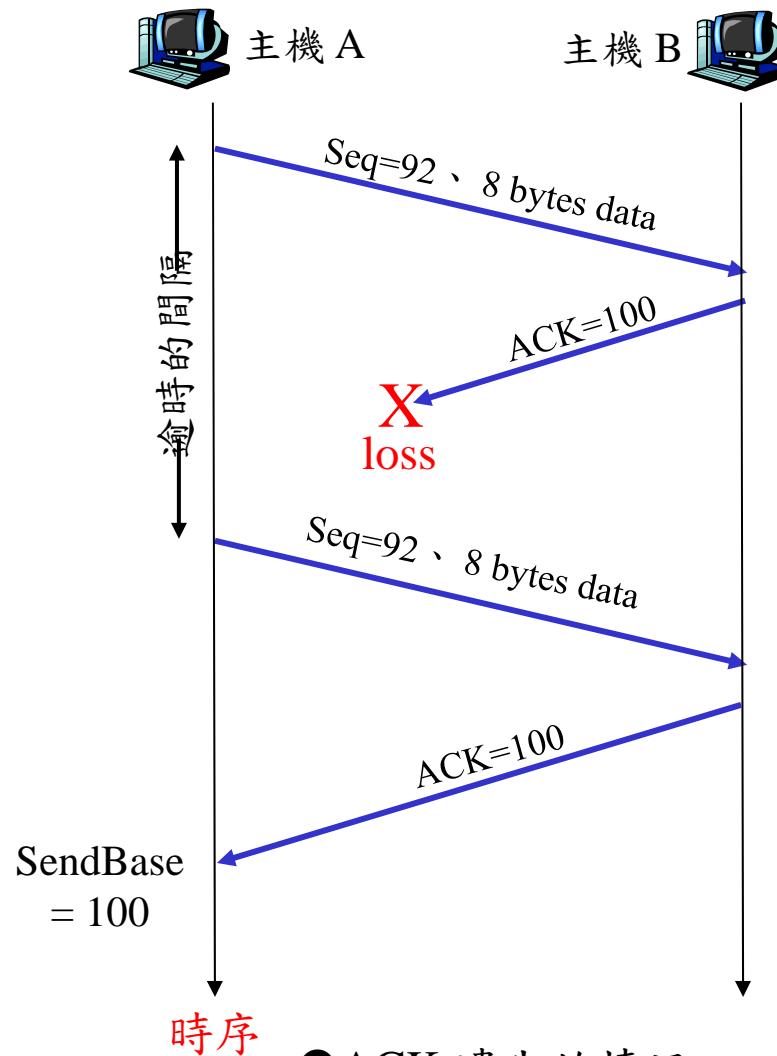
    } /* end of loop forever */
}
```

圖3.33 簡化的TCP傳送端



■ TCP 可靠的資料傳輸 (Cont.)

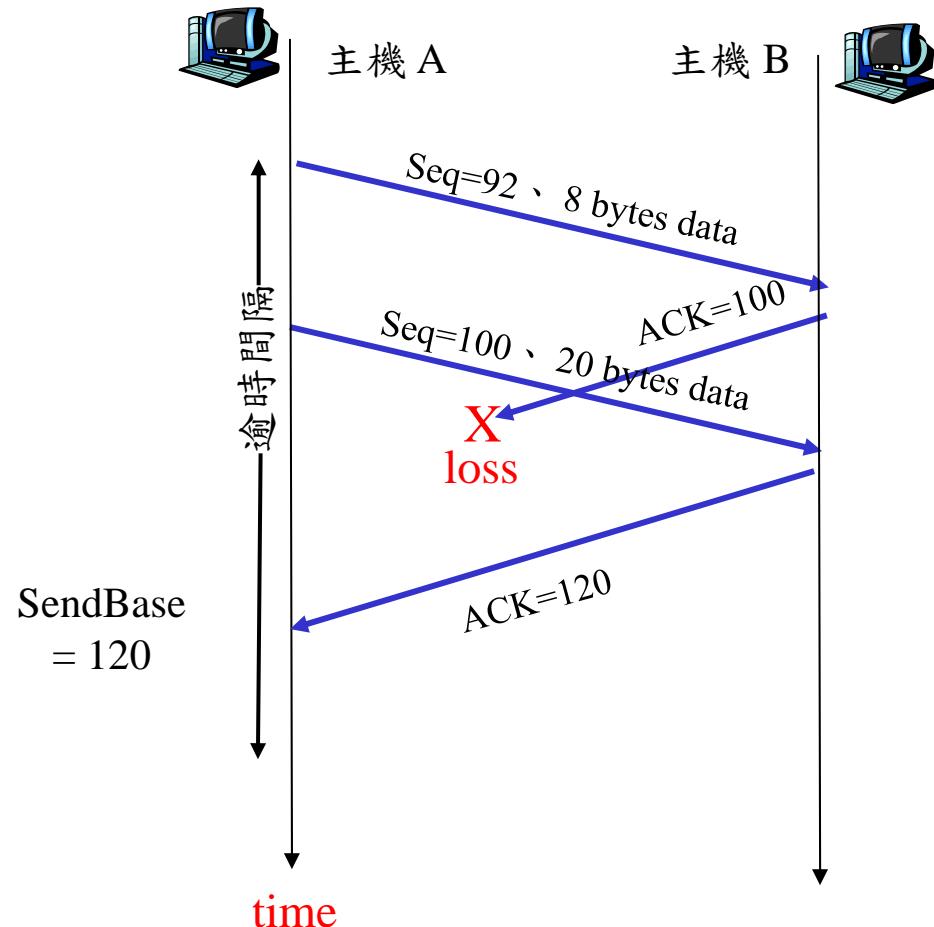
► TCP重新傳送的情況：





■ TCP 可靠的資料傳輸 (Cont.)

► TCP重新傳送的情況：(more)



③ 累積式 ACK 的情況



■ TCP流量控制

▶ 運作原理

- TCP連線的接收端有一個接收緩衝區 (圖3.38)
- 應用程式的行程也許會以較慢的速度從緩衝區讀取資料
- 流量控制 -
 - 傳送端不會傳送太多太快的資料超過接收端的緩衝區
 - 調整傳送端的速度與接收端應用程式能負擔的速度相符



圖3.38 接收端的接收視窗(RcvWindow)與接收緩衝區(RcvBuffer)



■ TCP流量控制的運作

- ▶ 假設：TCP接收端會將順序不正確的資料區段捨棄
- ▶ 緩衝區內的剩餘空間 RcvWindow
 - = $RcvBuffer - (LastByteRcvd - LastByteRead)$
- ▶ 接收端將 RcvWindow 值包含在資料區段裡，以告知剩餘的空間
- ▶ 傳送端限制未確認的資料大小在 RcvWindow 值之下
 - 保證接收端緩衝區不會溢出



Flow control Applet:

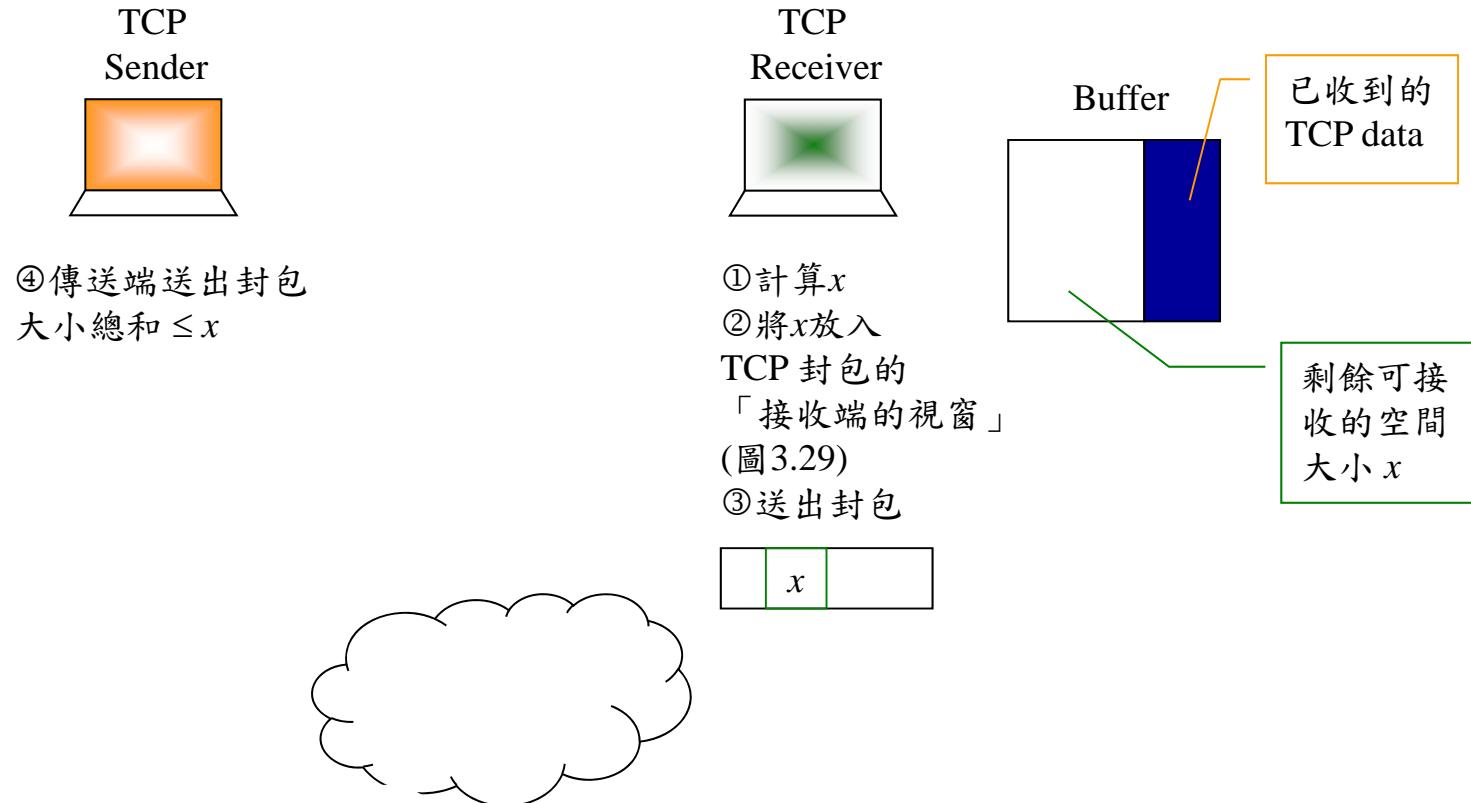
http://media.pearsoncmg.com/aw/aw_kurose_network_4/applets/flow/FlowControl.htm

Ex 3.13



■ TCP流量控制的運作(Cont.)

► 示意圖





■ TCP建立連線

- ▶ TCP 傳送端、接收端在交換資料區段之前，會先建立 “連線”
 - 將 TCP 變數初始化：
 - 序號
 - 緩衝區、流量控制資訊 (例如 RcvWindow)
 - 用戶端：開始連線者
 - `Socket clientSocket = new Socket("hostname", "port number");`
 - 伺服端：被用戶端聯繫
 - `Socket connectionSocket = welcomeSocket.accept();`
 - ▶ 三路交握 (three-way handshaking)
 - 步驟 1：用戶端主機傳送 TCP SYN 資料區段到伺服器
 - 指定初始的序號
 - 沒有資料傳遞
 - 步驟 2：伺服端主機收到 SYN，以 SYNACK 資料區段回應
 - 伺服端配置緩衝區
 - 指定伺服端的初始序號
 - 步驟 3：用戶端收到 SYNACK，回應 ACK 資料區段，可能含有資料
- ⇒ 圖3.39

■ TCP關閉連線

用戶端關閉 socket : clientSocket.close();

- ▶ 步驟 1：用戶端 終端系統傳送 TCP FIN控制區段到伺服端
- ▶ 步驟 2：伺服端 接收到FIN、以 ACK 回應。關閉連線、傳送 FIN
- ▶ 步驟 3：用戶端 收到 FIN、回應 ACK 訊息
 - 進入 “等待計時” – 對接收到的 FIN 做確認的回應
- ▶ 步驟 4：伺服端 收到ACK。連線關閉

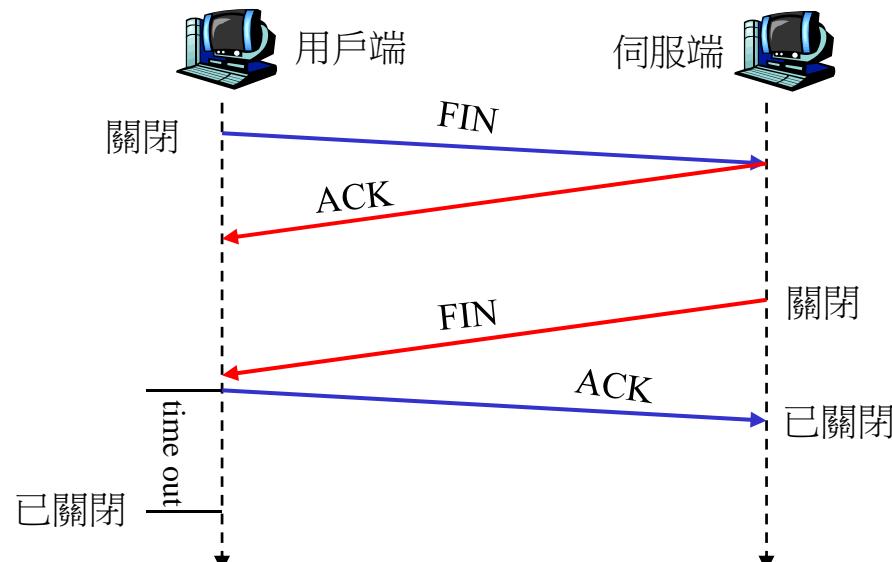


圖3.40 關閉TCP連線





■ TCP的生命週期

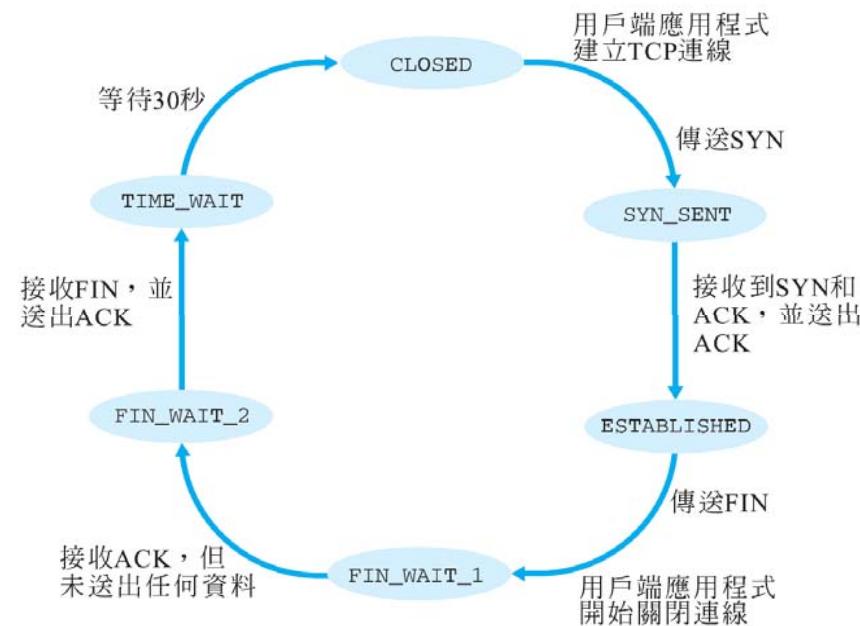


圖3.41 TCP用戶端生命週期

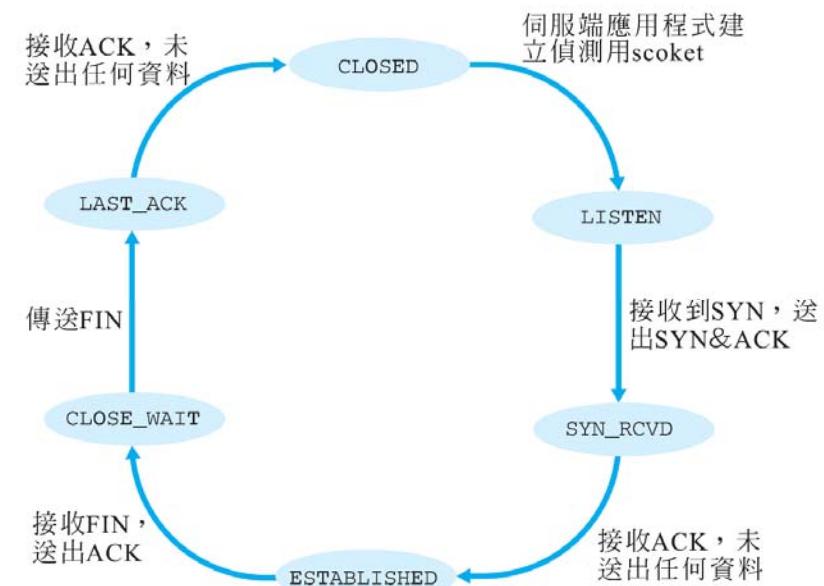
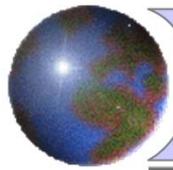


圖3.42 TCP伺服端生命週期



第三章 導覽流程



3.1 傳輸層服務

3.2 多工和解多工

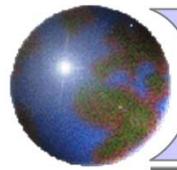
3.3 非預接式傳輸 – UDP

3.4 可靠資料傳輸的原理

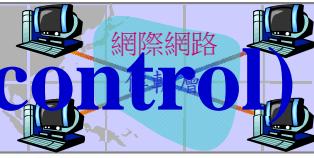
3.5 連線導向傳輸 – TCP

3.6 壓塞控制的原理

3.7 TCP 壓塞控制

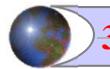


3.6 壓塞控制的原理 (Congestion control)

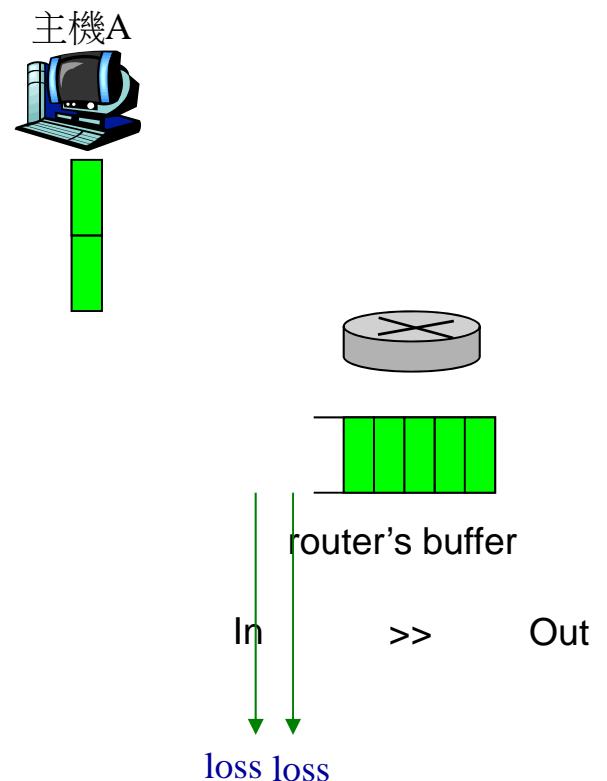


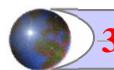
■ 網路壅塞 (congestion)

- ▶ 非正式地：“太多的來源端傳送太多的資料，對網路來說太快，超過能處理的速度”
- ▶ 與流量控制不同！
- ▶ 表現形式：
 - 封包遺失(路由器緩衝區溢出)
 - 長的延遲(在路由器緩衝區佇列中等待)
- ▶ 前十大的網路問題之一



■ 網路壅塞 (Cont.)





■ 壓塞的原因和代價 – 情境 1

- ▶ 兩個傳送端、兩個接收端；一個路由器具無限的緩衝區；沒有重傳機制

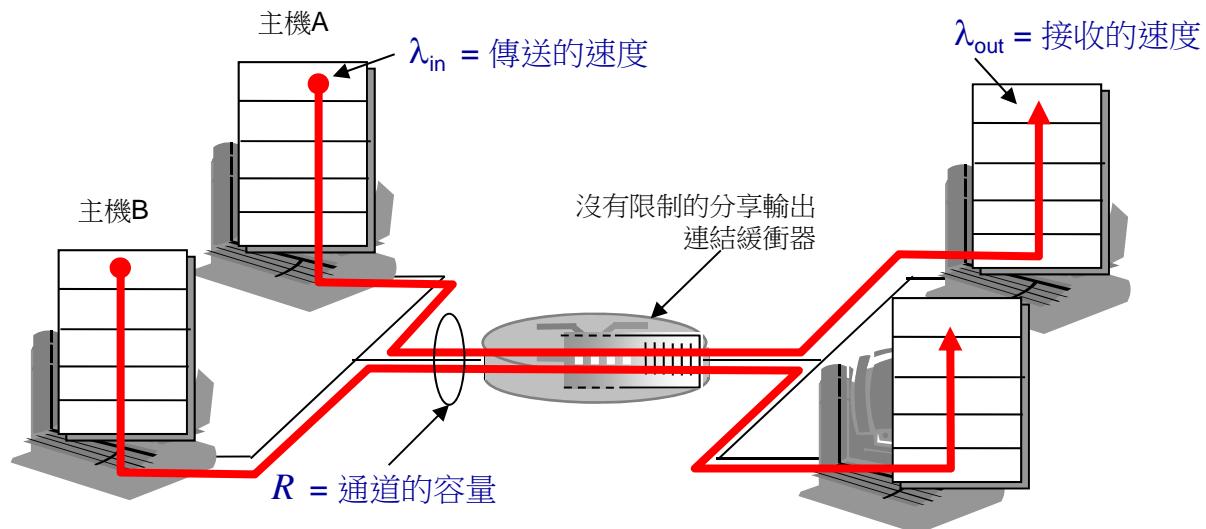


圖3.43 壓塞情境 1



■ 壓塞的原因和代價 – 情境 1 (Cont.)

- ▶ 兩個傳送端、兩個接收端；一個路由器具無限的緩衝區；沒有重傳機制

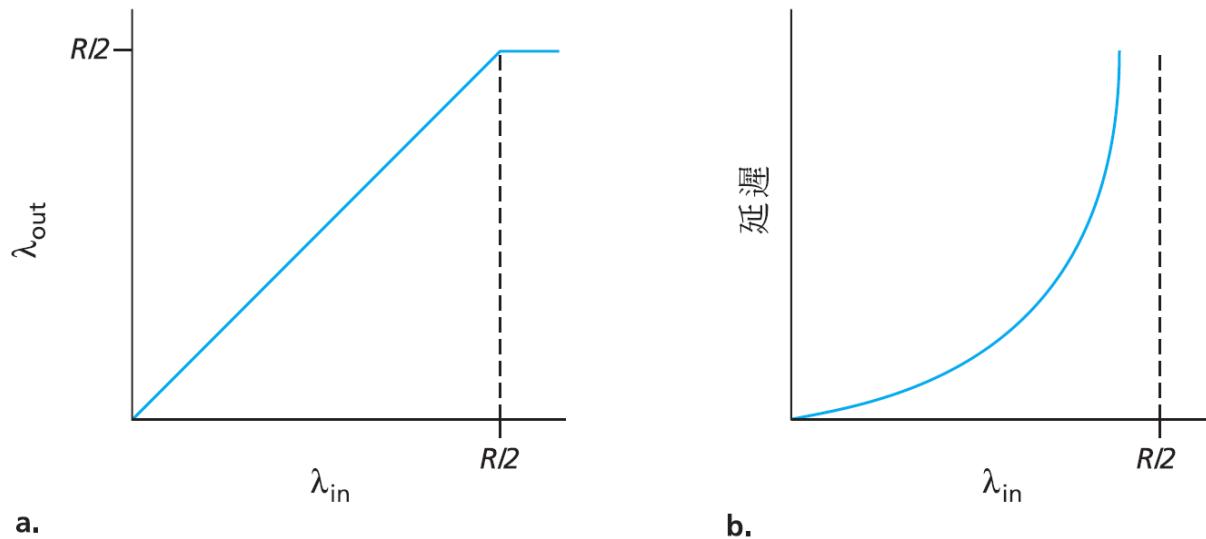
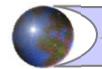


圖3.44 壓塞情境1 – 產出量與延遲時間相對於主機傳送速度的函數圖形

- 當壅塞時會有很長的延遲
- 最大的可達成流通量 = $R/2$



■ 壓塞的原因和代價 – 情境 2

- ▶ 兩個傳送端、兩個接收端；一個路由器具有有限的緩衝區；沒有重傳機制

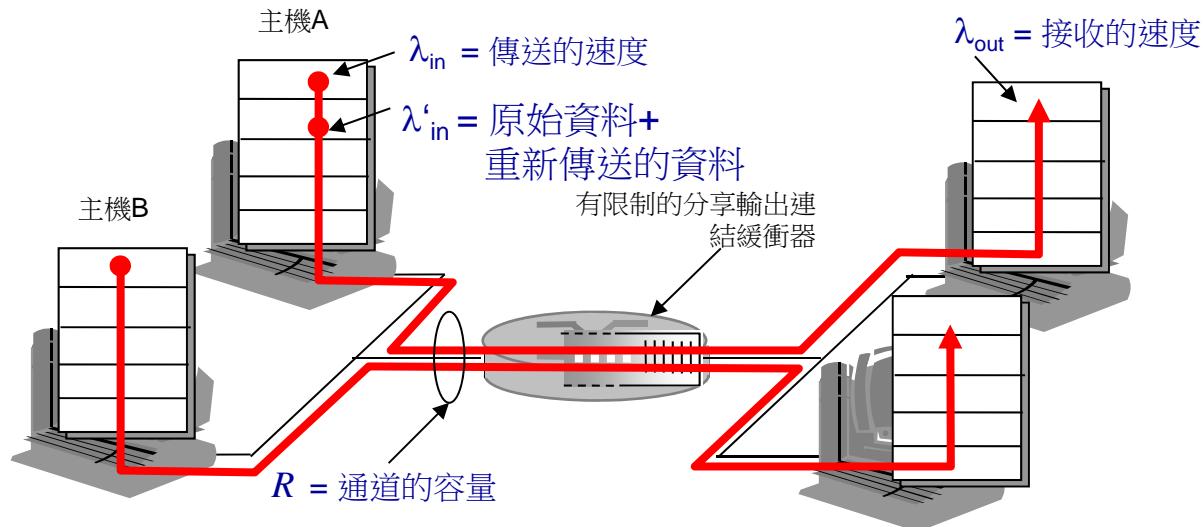


圖3.45 壓塞情境 2



■ 壓塞的原因和代價 – 情境 2 (Cont.)

- ▶ 兩個傳送端、兩個接收端；一個路由器具無限的緩衝區；沒有重傳機制

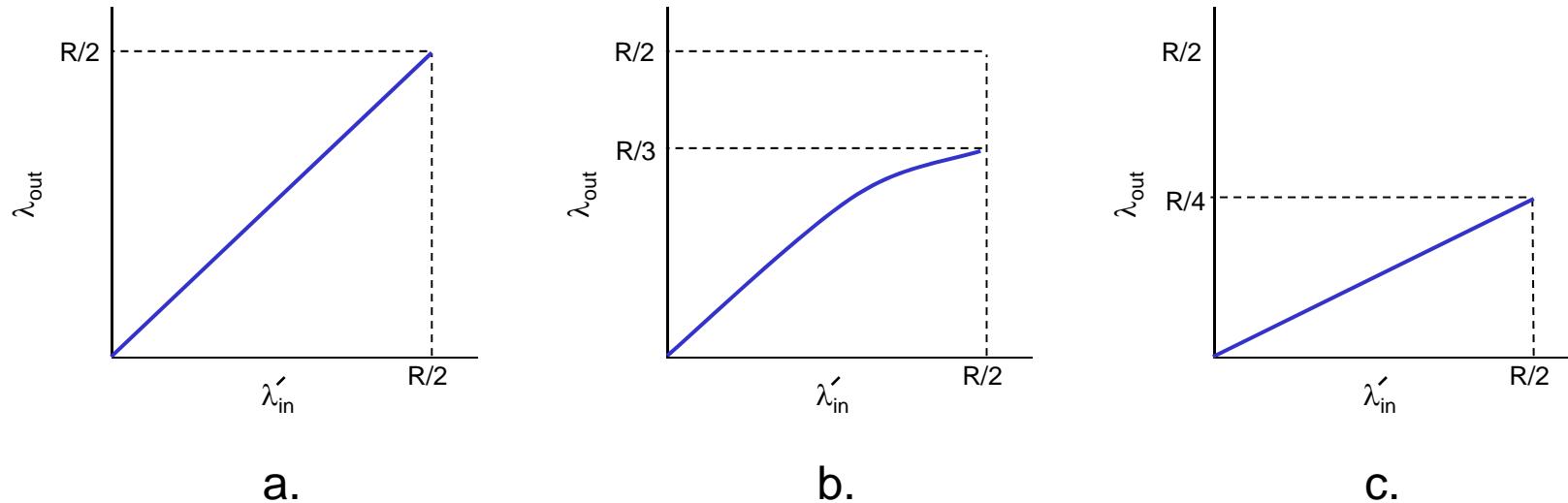


圖3.46 壓塞情境 2 – 使用有限緩衝區效能

壅塞的“代價”：

- 對給定的“實際產量”(goodput)，會有更多的工作
(重新傳輸)
- 不需要的重新傳輸：連結必須負擔多個封包的副本



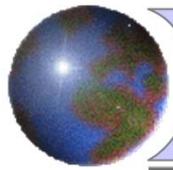
■ 壓塞控制的方法

▶ 端點對端點壅塞控制：

- 網路層並沒有提供明顯的協助
- 根據終端系統觀察到的遺失及延遲來判斷壅塞
- TCP採用的方法

▶ 網路協助的壅塞控制：

- 路由器提供協助給終端系統
 - 以一個位元來表示壅塞 (SNA、DECbit、TCP/IP ECN、ATM)
 - 限制傳送端應該傳送的明確速率



第三章 導覽流程



3.1 傳輸層服務

3.2 多工和解多工

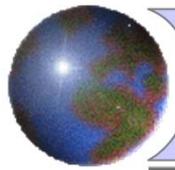
3.3 非預接式傳輸 – UDP

3.4 可靠資料傳輸的原理

3.5 連線導向傳輸 – TCP

3.6 壓塞控制的原理

3.7 TCP 壓塞控制



3.7 TCP 壓塞控制 (TCP Congestion control)



■ TCP 壓塞控制

▶ 原理

- 採用 端點對端點壅塞控制
- 讓傳送端根據某些原則來偵測網路壅塞的程度
- 若網路壅塞則傳送端限制自己將資料流量送入連線的速率

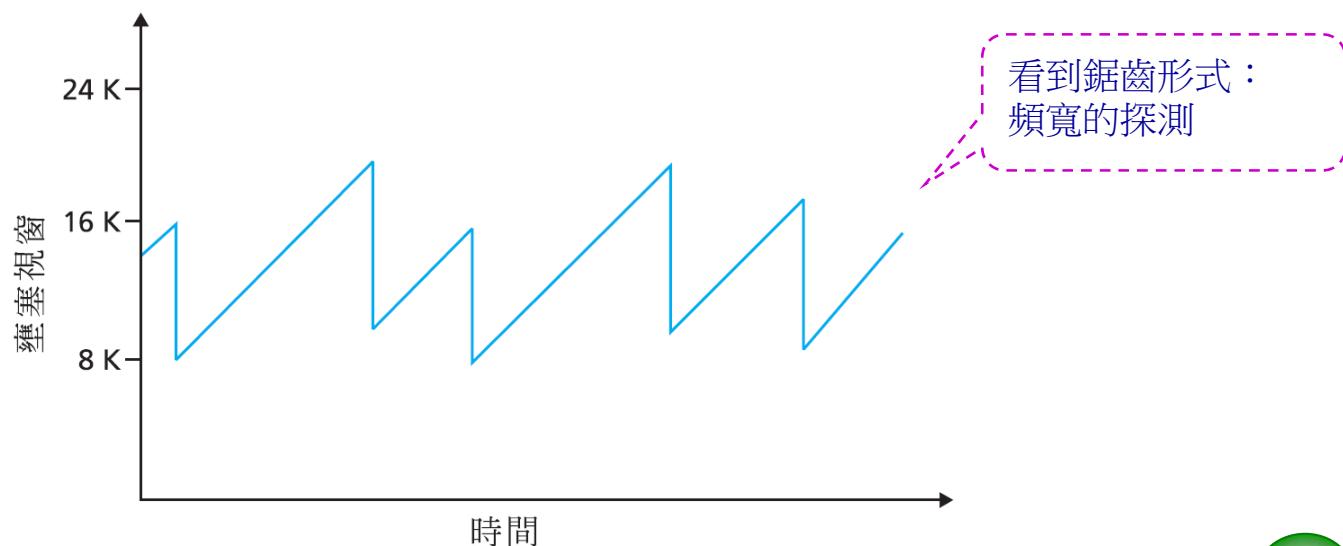
▶ 方法

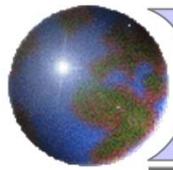
- 累積遞增、倍數遞減 (only describe in this course)
- 低速啟動
- 回應逾時事件



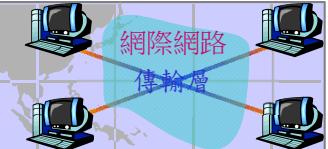
■ TCP 壓塞控制 – 累積遞增、倍數遞減

- ▶ 增加傳送速率(視窗大小)，以探測可用的頻寬(直到發生遺失的狀況)
- ▶ 累積遞增：每個 RTT，將 CongWin (congestion window, 壓塞窗格大小) 加 1，直到發生遺失
- ▶ 倍數遞減：在發生遺失之後，將 CongWin 減為一半





第三章 總結



■ 傳輸層服務的原則：

- ▶ 多工、解多工
- ▶ 可靠的資料傳輸
- ▶ 流量控制
- ▶ 壓塞控制

■ 網際網路上的例證和實作

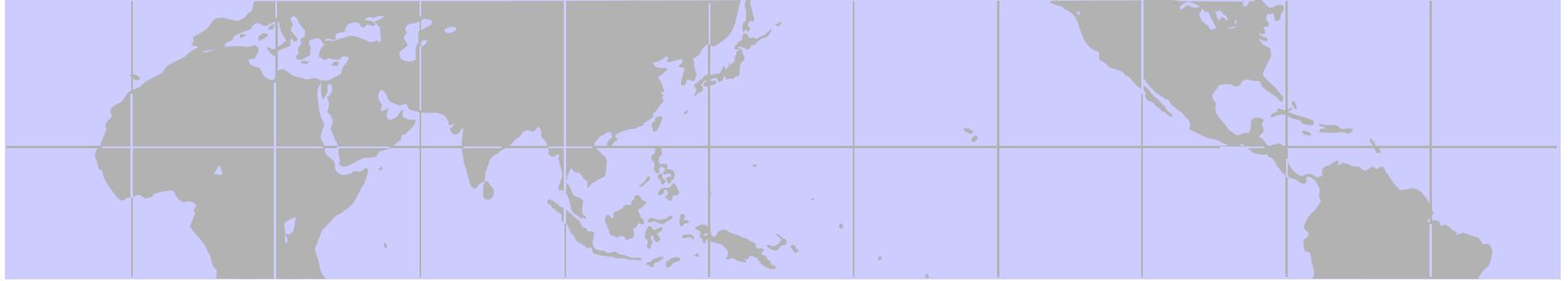
- ▶ UDP
- ▶ TCP

⇒ 接下來：

離開網路 “邊緣” (應用層、傳輸層)

進入網路 “核心” (網路層、連結層) ← 第4、5章

End of Chapter 3

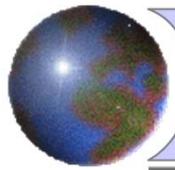


Chapter 4

Network Layer

(網路層)





Topic Overview



4.1 簡介

4.2 虛擬線路網路和資料包網路

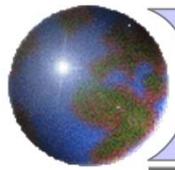
4.3 路由器的內部

4.4 網際網路IP協定 – 網際網路的轉送(forward)及定址

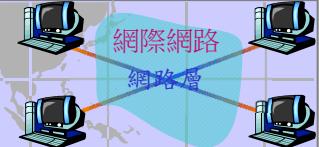
4.5 路由演算法 (routing algorithms)

4.6 網際網路的路由

4.7 廣播與群播路由 (broadcast and multicast)

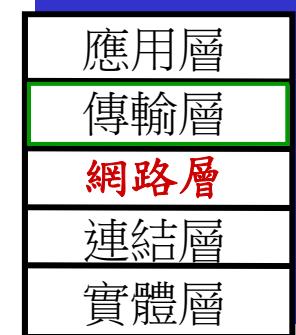


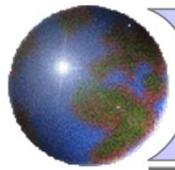
Topic Overview (Cont.)



■ 目標 (Goal)

- ▶ 了解網路層服務背後(隱藏)的原理：
 - 網路層服務模型
 - 轉送(forward) vs. 路由(route)
 - 路由器(router)如何運作
 - 路由演算法 (路徑的選擇)
 - 進階課題： IPv6
- ▶ 網際網路上的例證、實作





第四章 導覽流程



4.1 簡介

4.2 虛擬線路網路和資料包網路

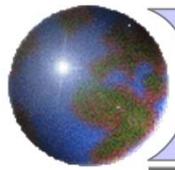
4.3 路由器的內部

4.4 網際網路IP協定 – 網際網路的轉送(forward)及定址

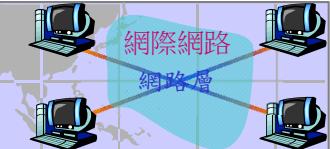
4.5 路由演算法 (routing algorithms)

4.6 網際網路的路由

4.7 廣播與群播路由 (broadcast and multicast)

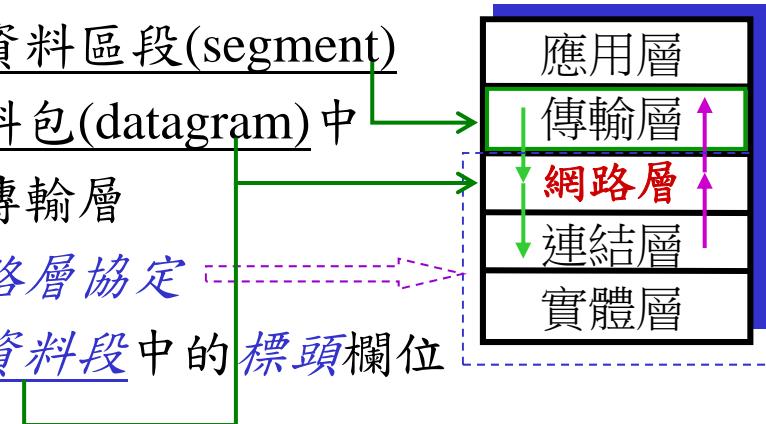


4.1 簡介



■ 網路層

- ▶ 從傳送主機到接收主機傳輸的資料區段(segment)
- ▶ 在傳送端將資料區段封裝在資料包(datagram)中
- ▶ 在接收端，將資料區段傳送到傳輸層
- ▶ 每一台**主機**、**路由器**中都有網路層協定
- ▶ 路由器會檢查經過它的所有IP 資料段中的標頭欄位





■ 網路層的通訊

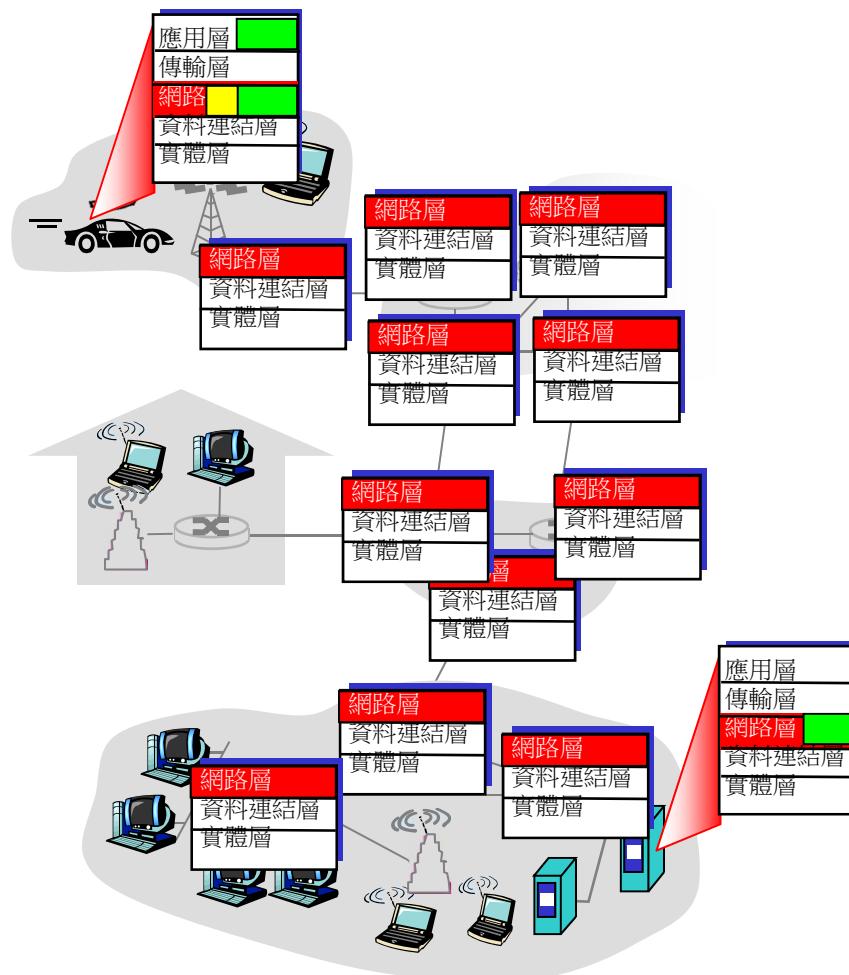


圖4.1 網路層



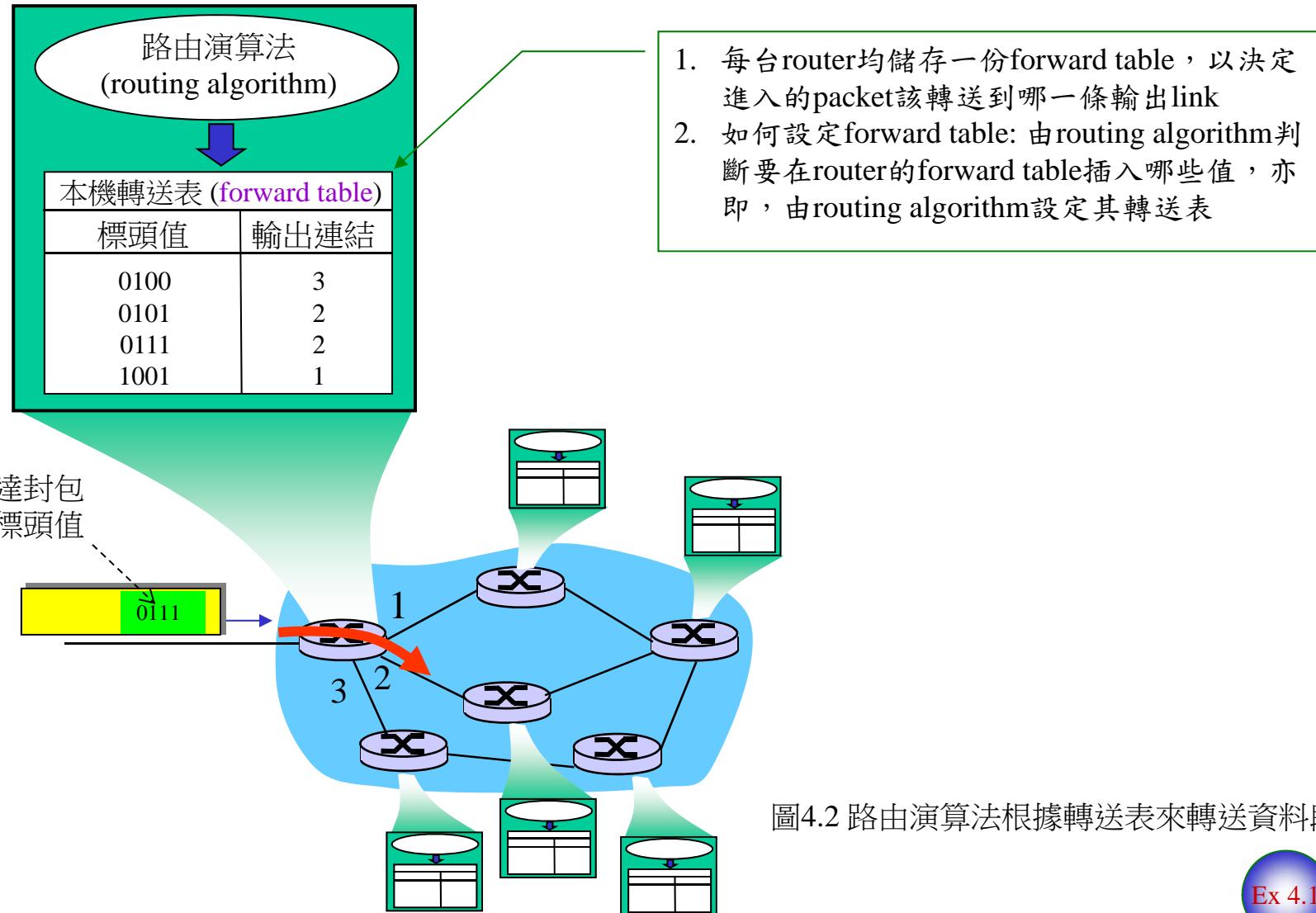
■ 兩個主要的網路層功能

- ▶ 轉送(forward)：將封包從路由器的輸入轉送到適當的路由器輸出
- ▶ 路由(繞送、route)：決定封包從來源端到目的端的路由(路徑)
 - 路由演算法 (routing algorithm)

比喻：

- ▶ 路由：計畫從來源到目的旅行的過程
- ▶ 轉送：經過單一交流道的過程

■ 兩個主要的網路層功能 (Cont.)





■ 建立連線 (connection setup)

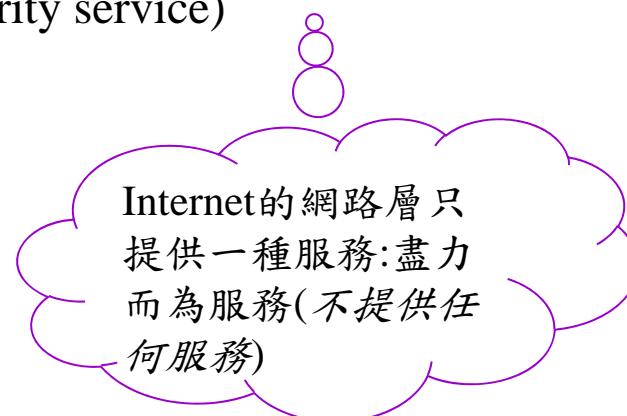
- ▶ 在某些網路架構中網路層的第三個重要功能
 - ATM、訊框轉播(frame relay)、X.25
- ▶ 在網路層資料包傳送之前，兩個主機以及介於其間的所有路由器需先建立虛擬連線 (VC , *Virtual connection*)
 - 路由器介入
 - 此程序稱為「建立連線」
- ▶ 網路層及傳輸層的連線服務：
 - 網路層：兩個主機之間 (在VCs案例中也許可以介入路由器)
 - 傳輸層：兩個行程之間



■ 網路服務模型 (network service model)

Q：有哪些服務模型提供給從傳送端到接收端傳輸資料段的通道？
(網路層可能提供的服務)

- ▶ 個別資料區段的服務範例：(p. 4-7)
 - 傳送保證 (guaranteed delivery)
 - 小於 40 毫秒的傳送保證 (guaranteed delivery with bounded delay)
- ▶ 資料區段 流量 的服務範例：(p. 4-7)
 - 依序封包傳送 (in-order packet delivery)
 - 流量的最低頻寬保證 (guaranteed minimal bandwidth)
 - 限制兩個封包之間間隔的改變 (guaranteed maximum jitter)
 - 安全性服務 (security service)

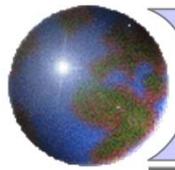




■ 網路服務模型 (Cont.)

表4.1 網際網路、ATM CBR、ATM ABR 服務模型

網路架構	服務模型	保證？				壅塞指示
		頻寬	遺失	排序	時序	
網際網路	盡全力服務	無	無	無	無	無 (藉由遺失來推論)
ATM	CBR	常數速率	有	有	有	不發生壅塞
ATM	VBR	保證速率	有	有	有	不發生壅塞
ATM	ABR	保證最小速率	無	有	無	有
ATM	UBR	無	無	有	無	無



第四章 導覽流程



4.1 簡介

4.2 虛擬線路網路和資料包網路

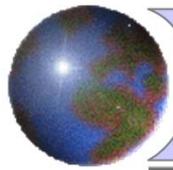
4.3 路由器的內部

4.4 網際網路IP協定 – 網際網路的轉送(forward)及定址

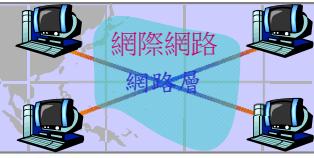
4.5 路由演算法 (routing algorithms)

4.6 網際網路的路由

4.7 廣播與群播路由 (broadcast and multicast)



4.2 虛擬線路網路和資料包網路

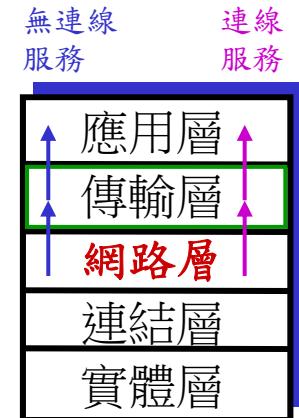


■ 傳輸層

- ▶ 可提供應用程式行程間的可靠連線服務(TCP)及不可靠無連線服務(UDP)

■ 網路層

- ▶ 可提供給傳輸層來源端和目的端主機
 - 可靠的連線服務 – **虛擬線路網路** (virtual-circuit network)
 - ATM、frame relay
 - 不可靠的無連線服務 – **資料包網路** (datagram network)
 - Internet
- ▶ 與傳輸層服務的差異
 - 服務：主機對主機
 - **沒有選擇性**：單一網路只提供其中一個 (如：Internet只提供不可靠的無連線服務而無可靠的連線服務)
 - 實作：在核心(*core-router*)及終端系統(*end system*)





■ 虛擬線路 (Virtual Circuit)

▶ 意義

- 來源端到目的端的路徑，如同電話線路
- 效能良好
- 沿著來源端到目的端路徑的網路行為

▶ 使用

- 在資料開始傳送前，必須為每個連結建立連線、拆除
- 每個封包會承載一個 VC 識別碼(並非目的端主機位址)
- 在來源端到目的端的路徑上，每個路由器都必須維護每個經過的連線的「狀態」
- 連結(link)和路由器(router)的資源(頻寬、緩衝區)可能會配置給 VC (專用資源=預期的伺服器)

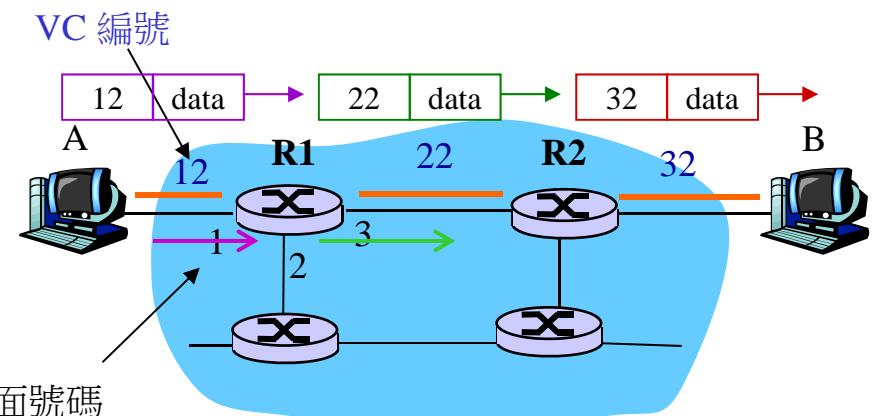


■ 虛擬線路VC (Cont.)

► VC 實作

- 一條 VC 包含了：
 - 一條來源端至目的端間的路徑
 - VC 編號，路徑上的每一個連結都有一個編號
 - 路徑上的每一個路由器轉送表中的紀錄
- 一個屬於虛擬線路的封包會載送一個 VC 編號
- 在每一個連結，VC 編號必須改變
 - 新的 VC 編號從轉送表中得來

■ 虛擬線路VC (Cont.)



R1路由器的轉送表 (forwarding table)

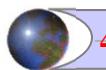
進入介面	進入VC編號	離開介面	離開VC編號
1	12	3	22
2	63	1	18
3	7	2	17
1	97	3	87
...

router 中的
forward table 會
隨VC建立及移
除而有所變化

圖4.3 簡單的虛擬線路網路

- ①情境說明:
主機A要求網路在自己與主機B間建立一條VC
- ②網路選擇:
VC路徑=A-R1-R2-B
連結的VC編號依序為12、22、32





■ 虛擬線路 – 訊號協定 (signal protocol)

- ▶ 用來設定、維護及拆除 VC
- ▶ 用在 ATM、訊框轉播 (frame relay)、X.25
- ▶ 並沒有用在現今的網際網路(internet)中

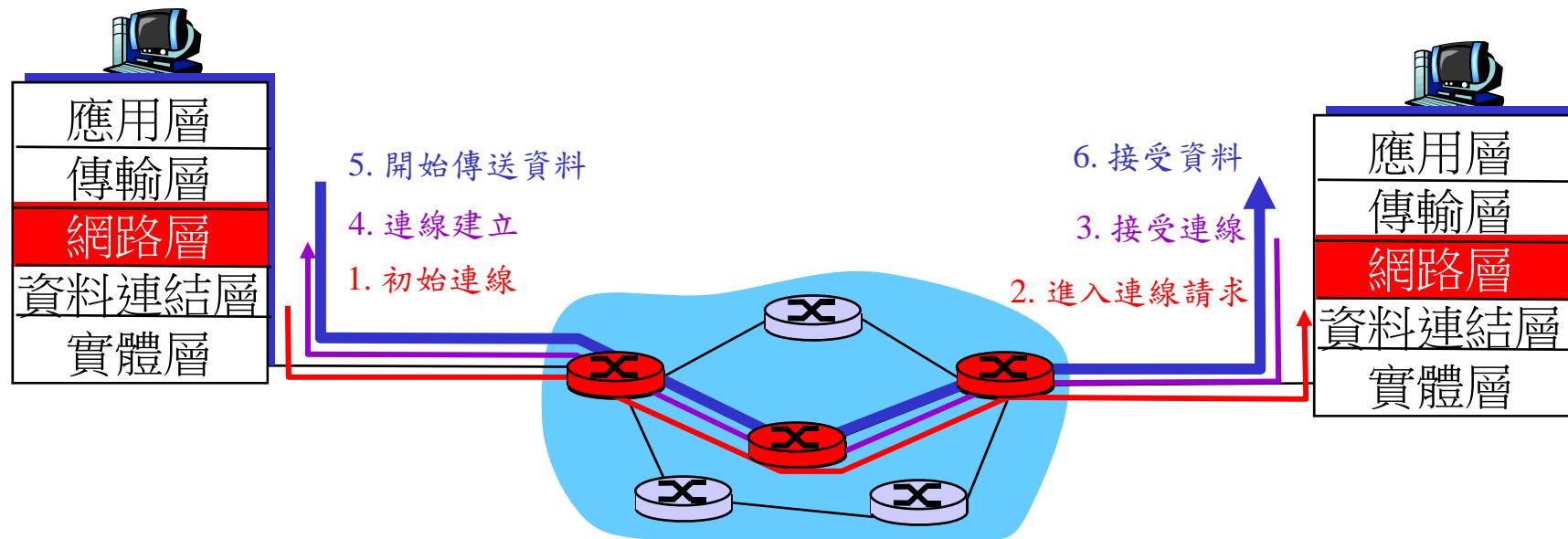


圖4.4 建立虛擬線路





■ 資料(封)包網路 (Datagram network)

► 意義

- 在網路層 不需要建立連線
- 路由器：沒有終端對終端的連線狀態
 - 沒有網路層的「連線」觀念
- 使用目的端主機位址轉送封包
 - 在同一對來源-目的端之間的封包可能會使用不同的路徑

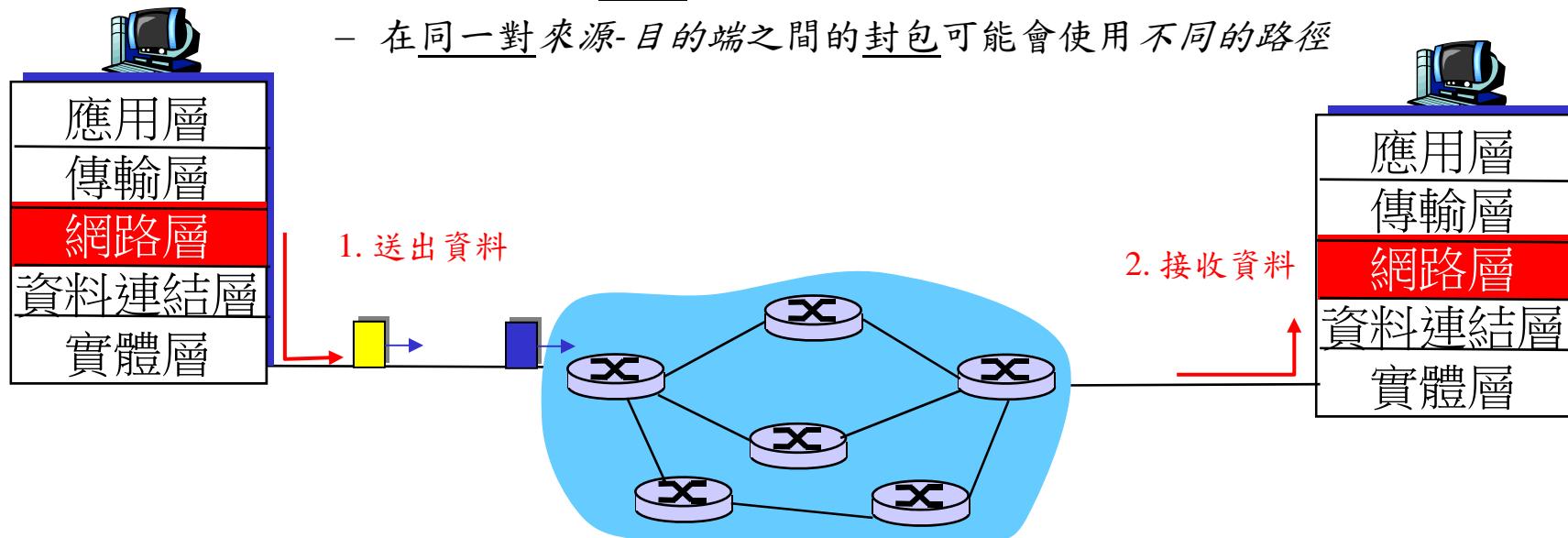


圖4.5 資料包網路





■ 資料包網路 (Cont.)

▶ 轉送表 (forwarding table)

- 40億個可能的紀錄 (32位元IP位址)
- 路由器不需儲存40億個項目

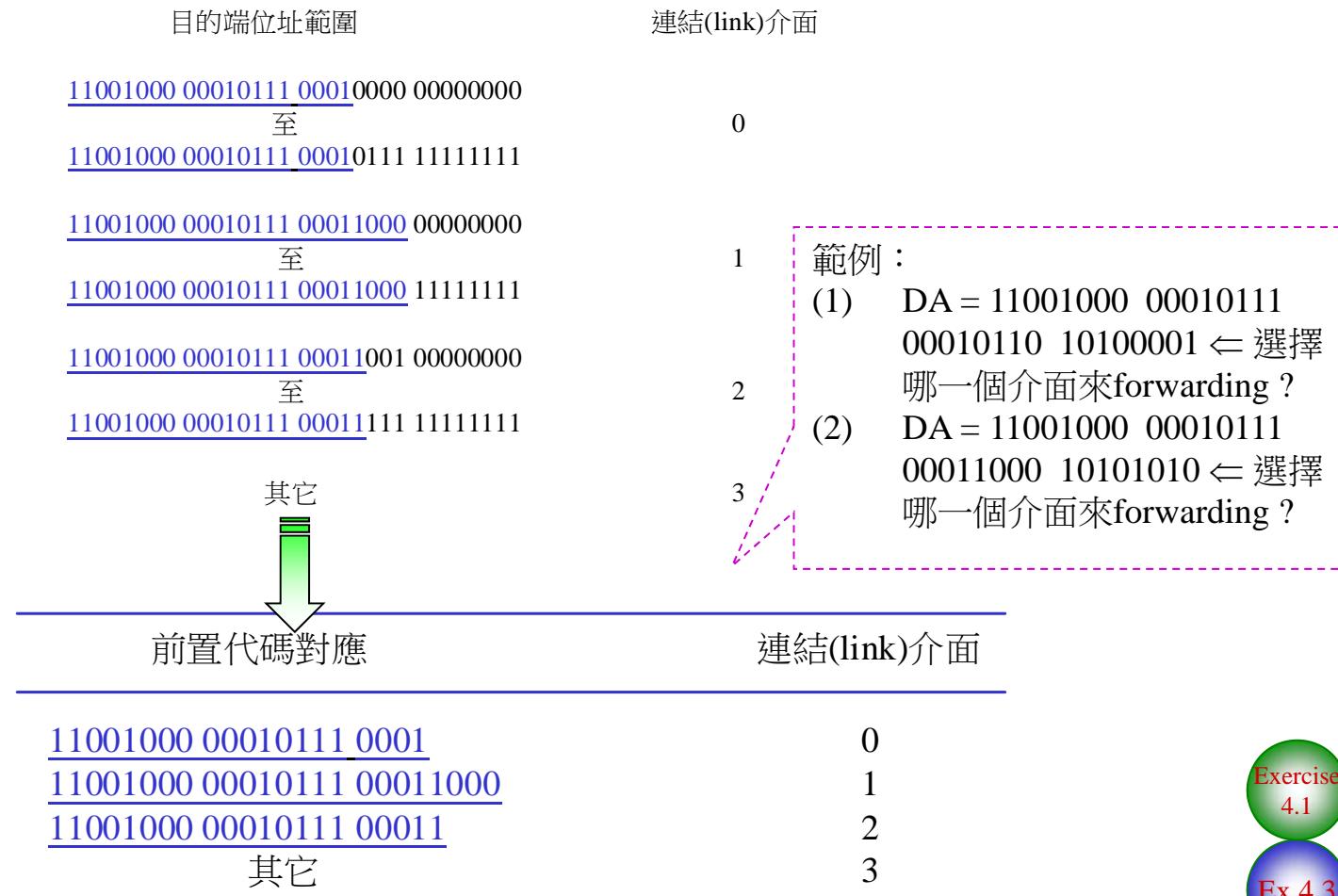
目的端位址範圍	連結(link)介面
<u>11001000 00010111 00010000 00000000</u> 至 <u>11001000 00010111 00010111 11111111</u>	0
<u>11001000 00010111 00011000 00000000</u> 至 <u>11001000 00010111 00011000 11111111</u>	1
<u>11001000 00010111 00011001 00000000</u> 至 <u>11001000 00010111 00011111 11111111</u>	2
其它	3



■ 資料包網路 (Cont.)

▶ 轉送表 (Cont.)

○ 最長前置代碼對應 (longest prefix matching)





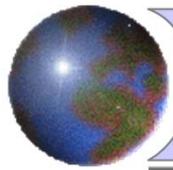
■ 虛擬線路網路 vs. 資料(封)包網路 (Ref Sect. 1.3)

▶ 資料包網路 – 網際網路

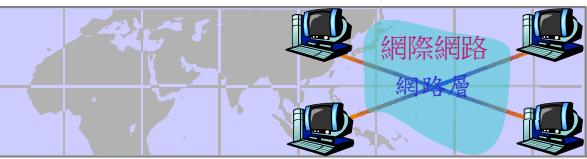
- 電腦之間的資料交換
 - “彈性的” 服務，沒有嚴格的時限要求
- “聰明的” 終端系統（電腦）
 - 能夠適應、執行控制、錯誤復原
 - 網路內部是簡單的、邊緣是複雜的
- 許多連結型態
 - 不同的特質
 - 統一的服務困難

▶ 虛擬線路網路 – ATM (Asynchronous Transfer Mode)

- 來自電話系統
- 人類的對話：
 - 嚴格的時限、可靠性的要求
 - 保證服務的需求
- “愚笨的” 終端系統
 - 電話
 - 網路內部是複雜的、邊緣是簡單的



第四章 導覽流程



4.1 簡介

4.2 虛擬線路網路和資料包網路

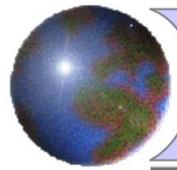
4.3 路由器的內部

4.4 網際網路IP協定 – 網際網路的轉送(forward)及定址

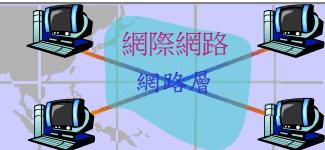
4.5 路由演算法 (routing algorithms)

4.6 網際網路的路由

4.7 廣播與群播路由 (broadcast and multicast)



4.3 路由器的內部 (Router)



■ 路由器架構綜觀

▶ 四項元件

- 輸入埠 (input port)
- 交換結構 (switch fabric)
- 輸出埠 (output port)
- 路由處理器 (routing processor)

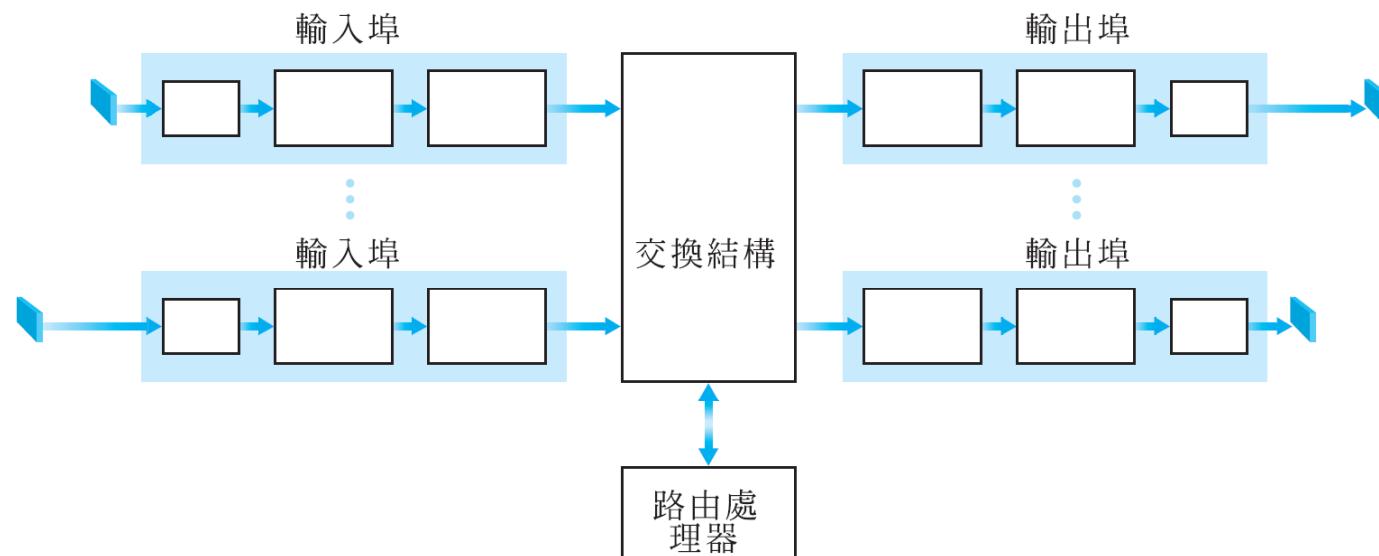
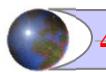


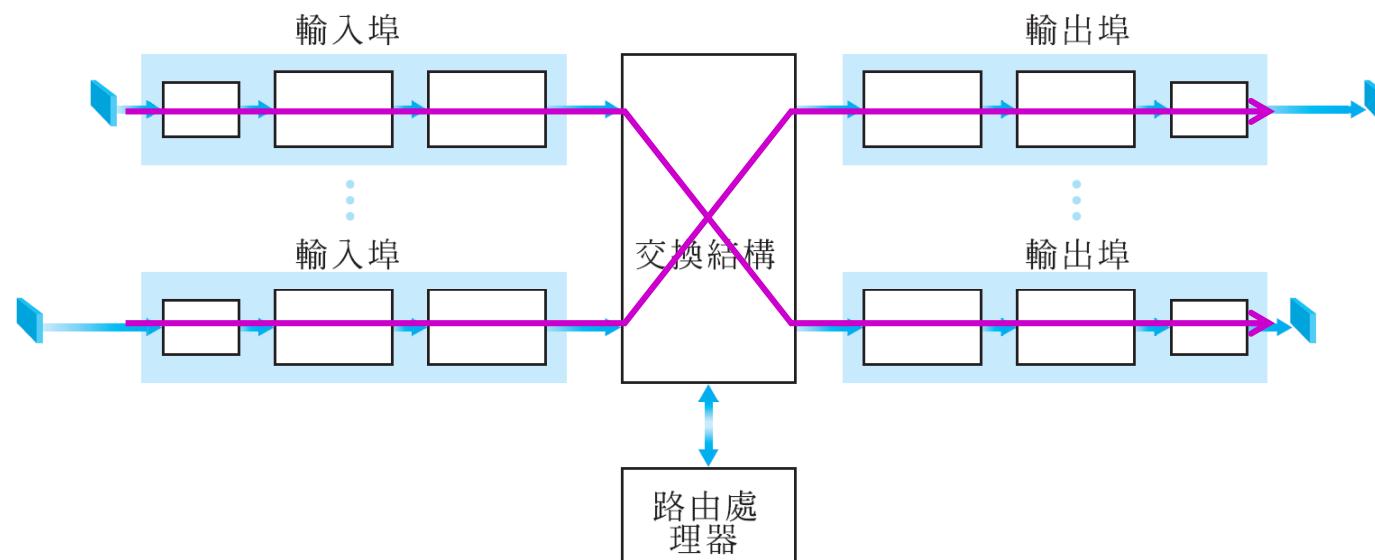
圖4.6 路由器架構





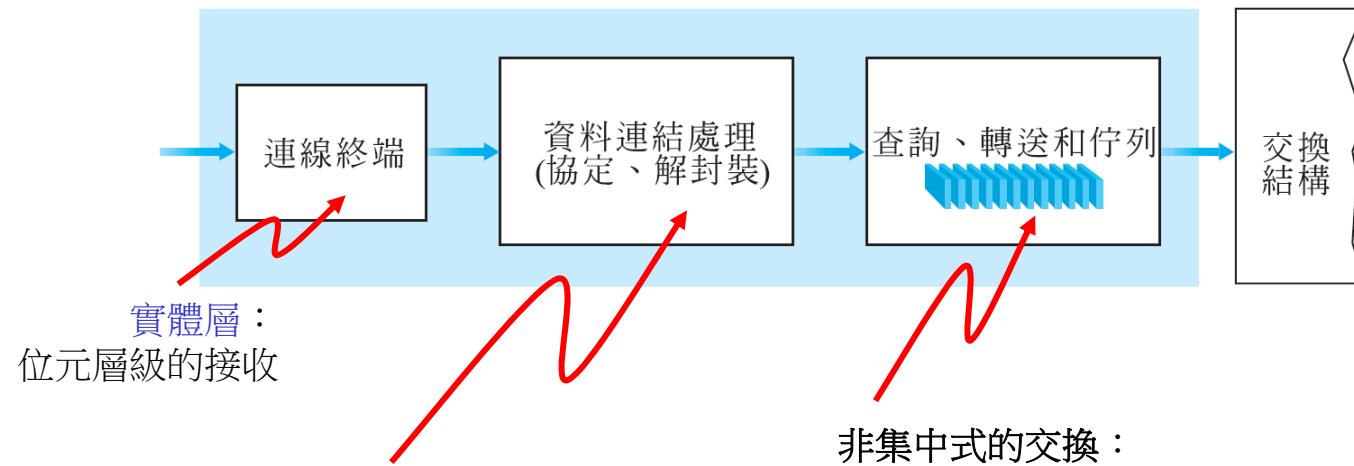
■ 路由器主要功能

- ▶ 執行路由演算法/協定 (RIP、OSPF、BGP)
- ▶ 將資料封包從輸入轉送到輸出連結 (input link → output link)





■ 輸入埠功能



- 紿定一個資料封包目的端，使用輸入埠記憶體中的轉送表來尋找輸出埠
- 目標：以「連線速度」完成輸入埠的處理過程
- 排隊：假如資料封包的抵達速率快於轉送進入交換結構的速率

輸入埠的功能：

- ① 執行實體層功能，以作為實體連結進入路由器的終端。
- ② 執行連結層功能，以和輸入連結遠端的連結層功能進行互動操作。
- ③ 執行查詢和轉送的功能，以讓轉送進入路由器交換結構的封包，會出現在適當的輸出埠中。

圖4.7 路由器輸入埠處理

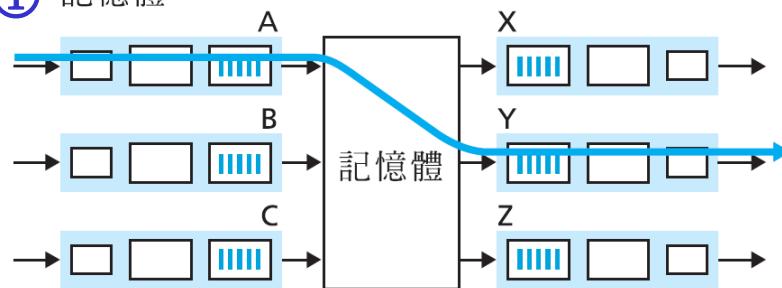




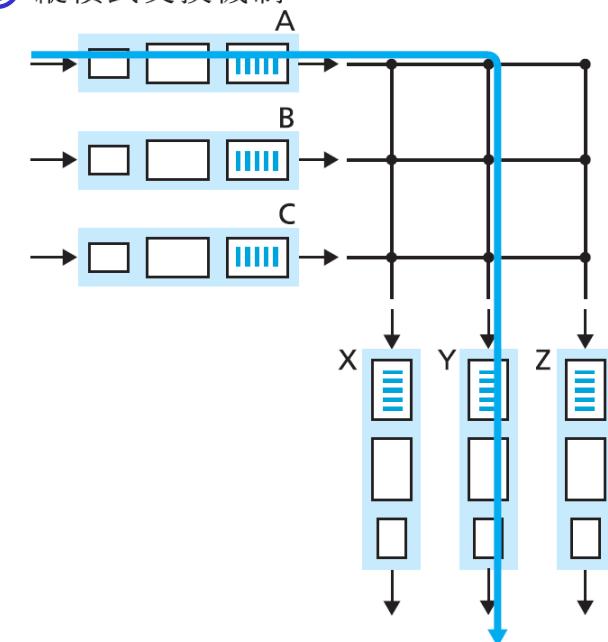
■ 交換結構 (switch fabric)

- ▶ 將路由器的輸入埠連接到輸出埠
- ▶ 三種交換結構 – 透過①記憶體②匯流排③互連網路進行交換

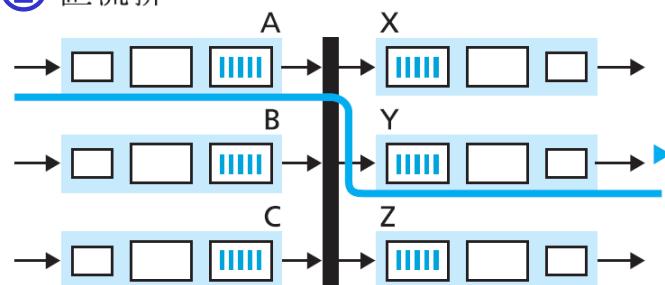
① 記憶體



③ 縱橫式交換機制



② 匯流排



說明：

輸入埠

輸出埠

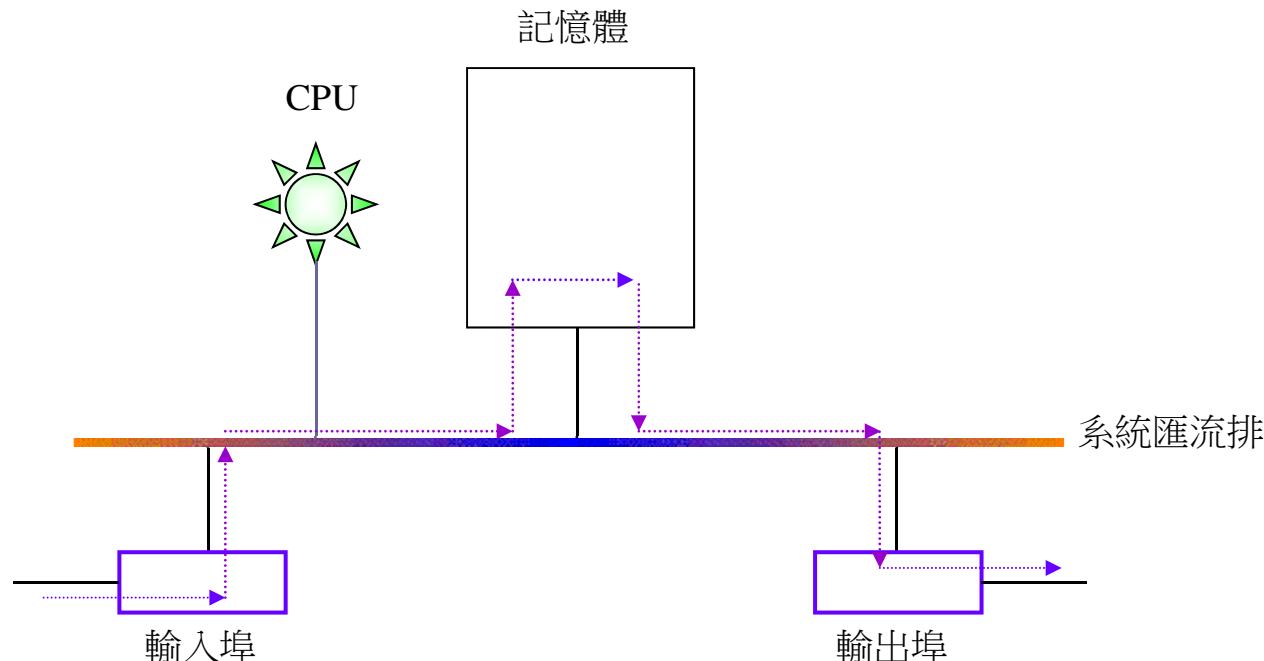
圖4.8 三種路由器交換技術



■ 交換結構 (Cont.)

▶ 三種交換結構 – (Cont.)

- ① 經由記憶體的交換機制 (第一代的路由器)
 - 傳統的電腦，在CPU的直接控制下完成交換動作
 - 封包被複製到系統記憶體，CPU以決定該送往那個輸出埠
 - 速度會被記憶體頻寬所限制 (一個資料段會經過兩次匯流排)



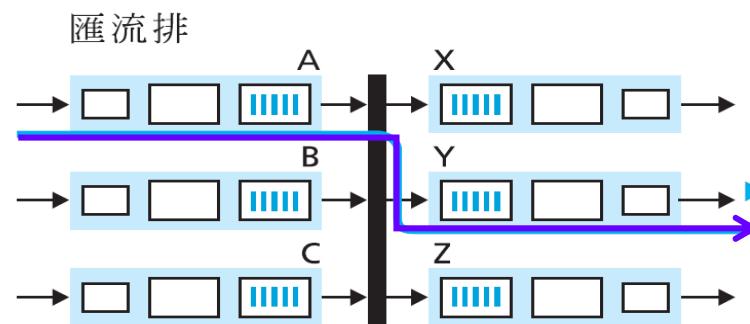


■ 交換結構 (Cont.)

▶ 三種交換結構 – (Cont.)

② 經由匯流排的交換機制

- 資料封包經由共享的匯流排從輸入埠記憶體送到輸出埠記憶體
- 匯流排的競爭：交換速度受限於匯流排頻寬
- 不需經由routing processor的介入
- 若bus忙碌，則封包需在輸入埠中排隊
- 32 Gbps 匯流排的Cisco 5600：對存取路由器及企業路由器是足夠的

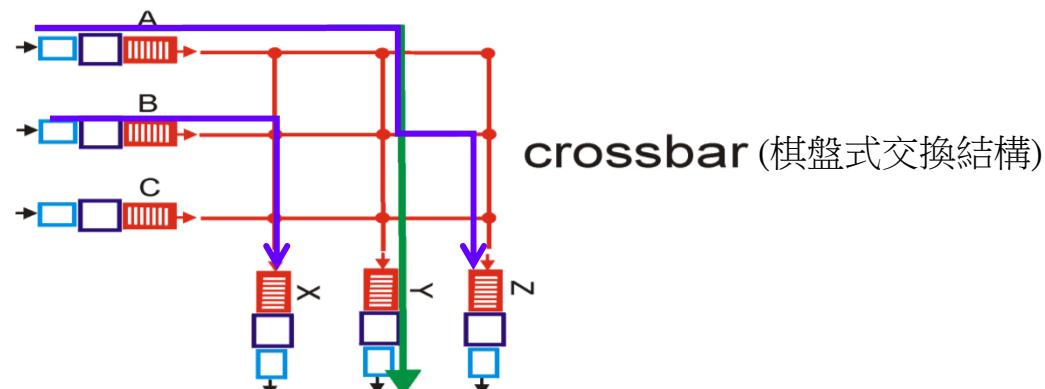




■ 交換結構 (Cont.)

▶ 三種交換結構 – (Cont.)

- ③ 經由互連網路的交換機制
 - 克服單一共用匯流排頻寬限制的方法
 - Banyan 網路、其他的互連網路初始的建立，是將處理器連接到多處裡器
 - 更進一步的設計：將資料封包分割成固定長度的封包單位，並將封包單位送入交換結構中
 - Cisco 12000：經由互連網路可達 60Gbps 的交換速率





■ 輸出埠功能

▶ 概念

- 需要緩衝區 – 當資料封包抵達結構的速率大於傳送速率
- 排程原則選擇佇列中的資料封包加以傳送
- 處理過程與輸入埠相反

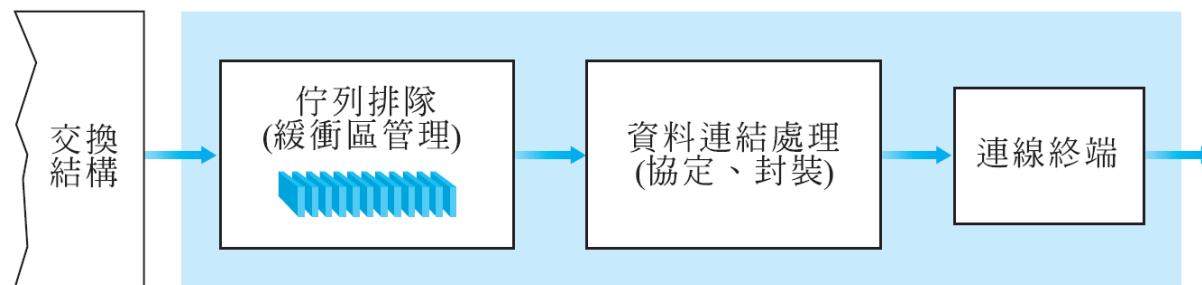


圖4.9 路由器輸出埠處理



■ 輸出埠功能 (Cont.)

▶ 輸出埠佇列 (queuing)

- 當經由交換結構抵達的速率超過輸出連結的速度
- 佇列排隊 (延遲) 以及因為輸出埠的緩衝區溢出導致的遺失！

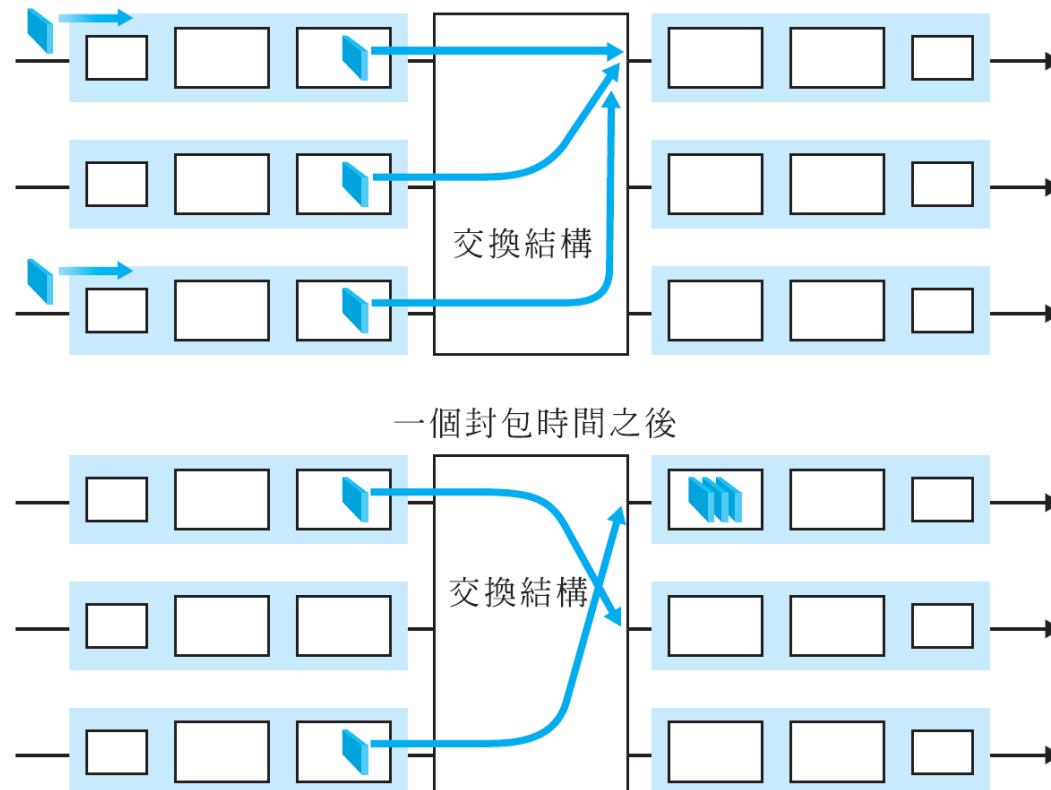


圖4.10 路由器輸出埠佇列



■ 輸入埠佇列

- Hung1 ▶ 交換結構速度慢於輸入埠的加總 \Rightarrow 可能在輸入佇列發生排隊的情形
- ▶ 連線前端 (HOL) 阻塞：在佇列最前端排隊的資料封包會阻止佇列中的其它資料封包往前移動
 - ▶ 佇列延遲以及導因於輸入緩衝區溢出的遺失！

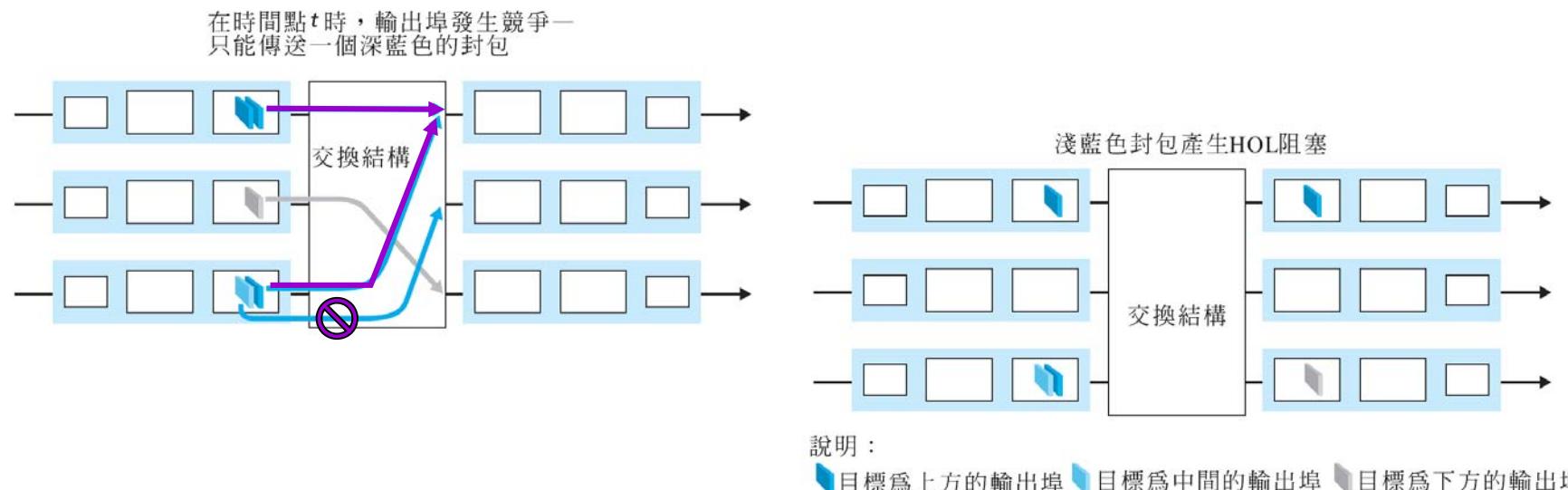


圖4.11 路由器輸入埠佇列交換結構的HOL攔阻

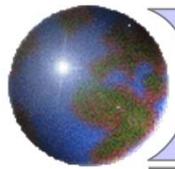


投影片 32

Hung1

交換結構從輸入埠將封包移至輸出埠的速度

Ruo-Wei Hung, 2011/8/7



第四章 導覽流程



4.1 簡介

4.2 虛擬線路網路和資料包網路

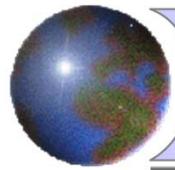
4.3 路由器的內部

4.4 網際網路IP協定- 網際網路的轉送(forward)及定址

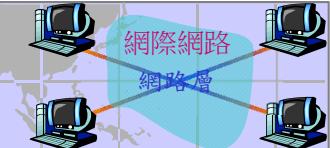
4.5 路由演算法 (routing algorithms)

4.6 網際網路的路由

4.7 廣播與群播路由 (broadcast and multicast)



4.4 網際網路IP協定



■ 網際網路的網路層

► 主機、路由器的網路層功能：

- ① 路由協定 (routing protocol)
- ② IP協定 (InternetProtocol protocol)
- ③ ICMP協定 (ICMP protocol)



圖4.12 網際網路中網路層的功能協定



■ IPv4 資料包格式 (p.4-29)

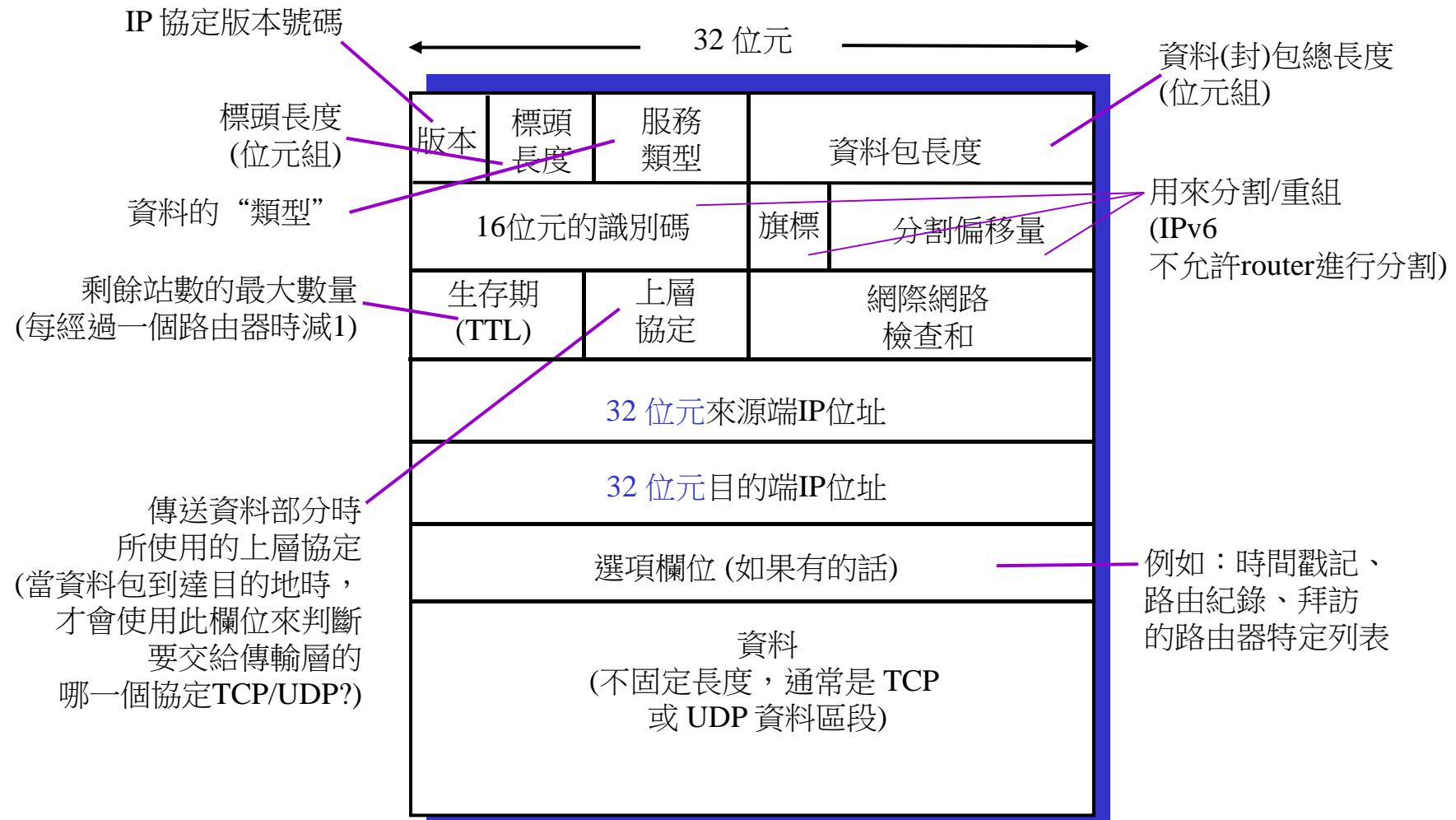


圖4.13 IPv4 資料包格式



■ TCP/IP有多少資源負擔?

- ▶ 20 位元組 TCP header
- ▶ 20 位元組 IP header (標頭)
- ▶ = 40 位元組 + 應用層的訊息負擔

■ UDP/IP有多少資源負擔?



■ IP 分割(fragmentation)與重組(reassembly)

- ▶ 為何需要將IP資料包(datagram)分割成較小的資料分段(fragment) ?

- 網路的連結(link – 資料連結層)具有 *MTU* (maximum transfer units, 最大的可傳輸大小) – 最大的連結層訊框(frame)
 - 不同的連結層型態，不同的MTU
- 過大的IP 資料包在網路內將被切分(分割)
 - 一個資料包(datagram)變成好幾個較小的資料分段
 - 只在目的端被“重組”
 - IP 標頭位元用來識別、排序相關的分割

eg., 乙太網路的
MTU=1500
bytes



■ IP 分割與重組 (Cont.)

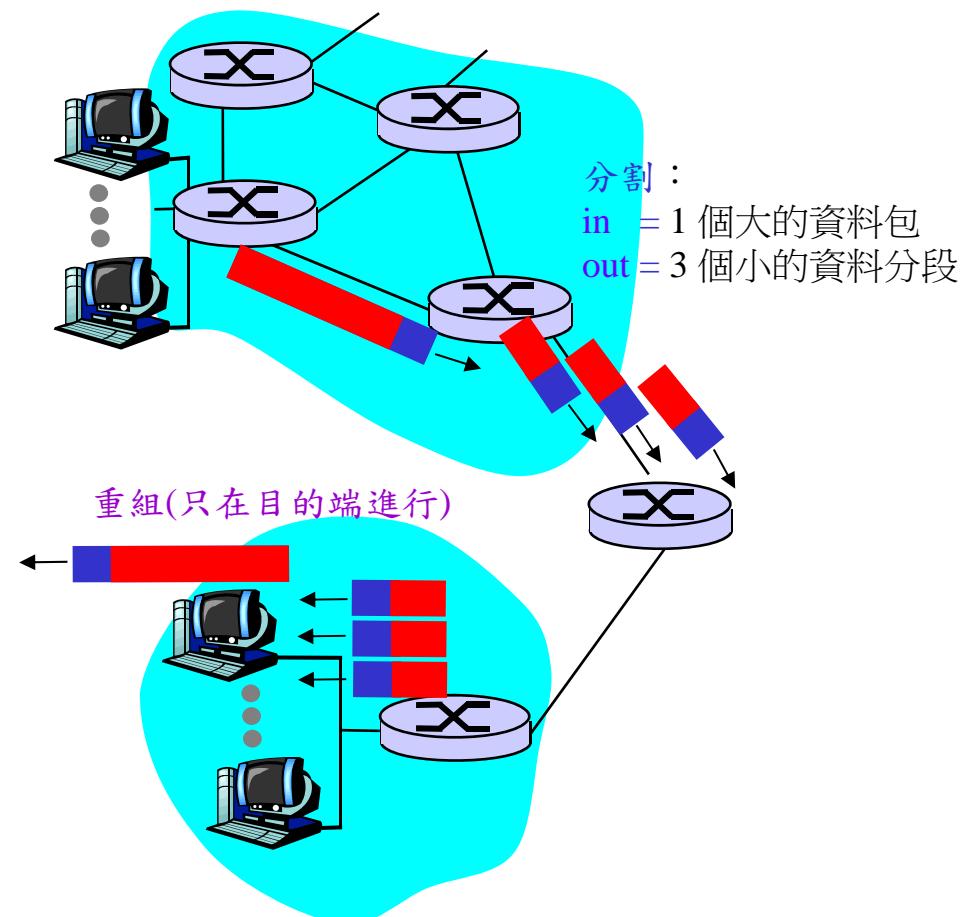


圖4.14 IP資料包的分割與重組

IP Fragmentation Applet:

http://media.pearsoncmg.com/aw/aw_kurose_network_2/applets/ip/ipfragmentation.html

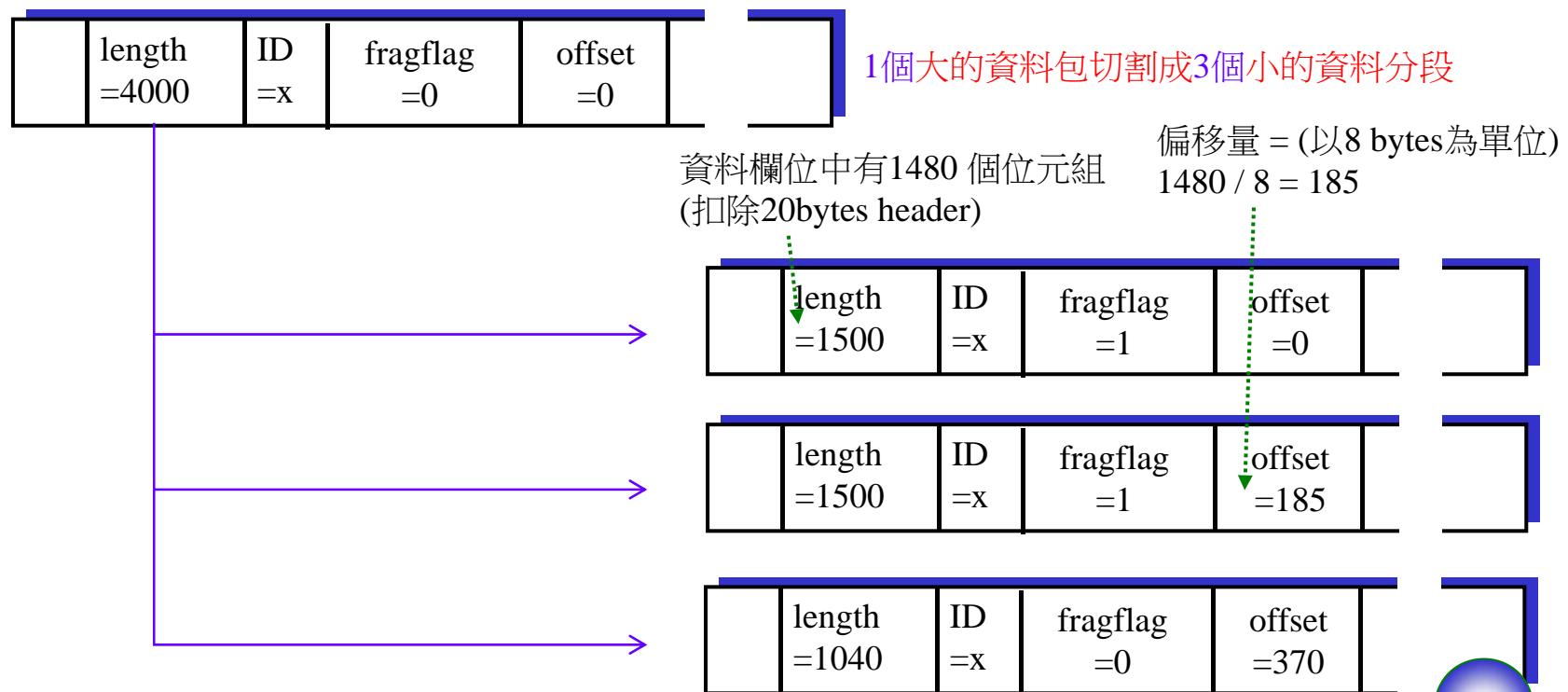


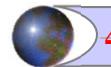
■ IP 分割與重組 (Cont.)

► 範例：

- 4000 位元組的資料包
- MTU = 1500 位元組

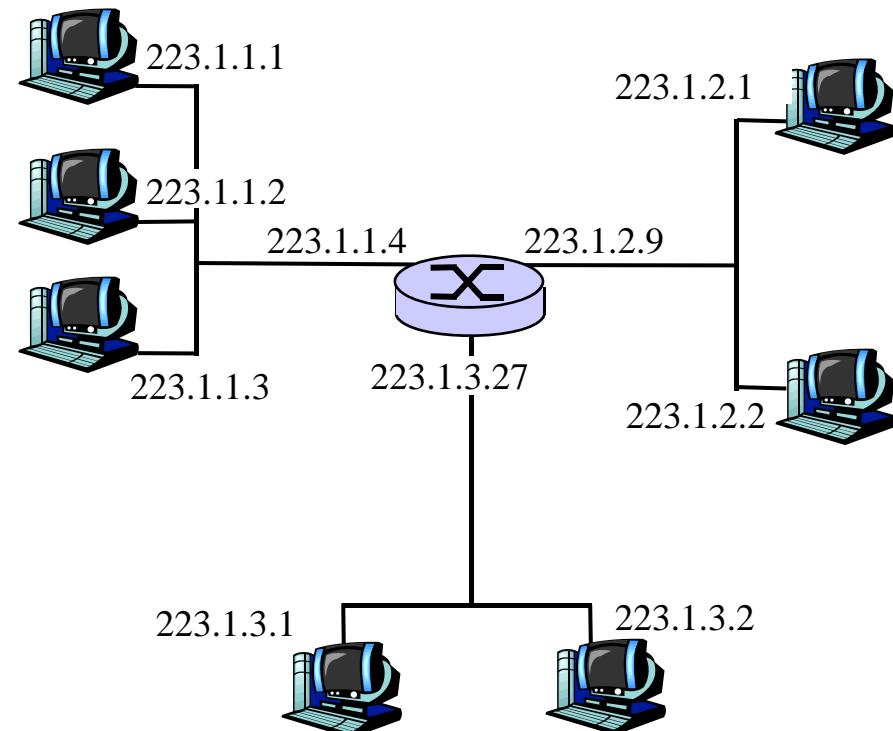
表4.2 IP資料包的分段 (p. 4-33)





■ IPv4 定址(addressing) – 簡介

- ▶ IP 位址：32位元的識別碼，做為主機、路由器的介面
- ▶ 介面：在主機/路由器和實體連結的交界處
 - 路由器通常擁有多個介面
 - 主機通常擁有一個介面
 - IP 位址代表每個介面



$223.1.1.1 = 11011111\ 00000001\ 00000001\ 00000001$

 \underline{223} \underline{1} \underline{1} \underline{1}

圖4.15(a) 介面位址



■ 子網路 (Sub-net)

► IP 位址：

- 子網路部分 (高階位元)
- 主機部分 (低階位元)

► 什麼是子網路？

- IP 具有相同子網路部分的裝置介面
- 可以在沒有路由器的狀況下，實體地互相連結

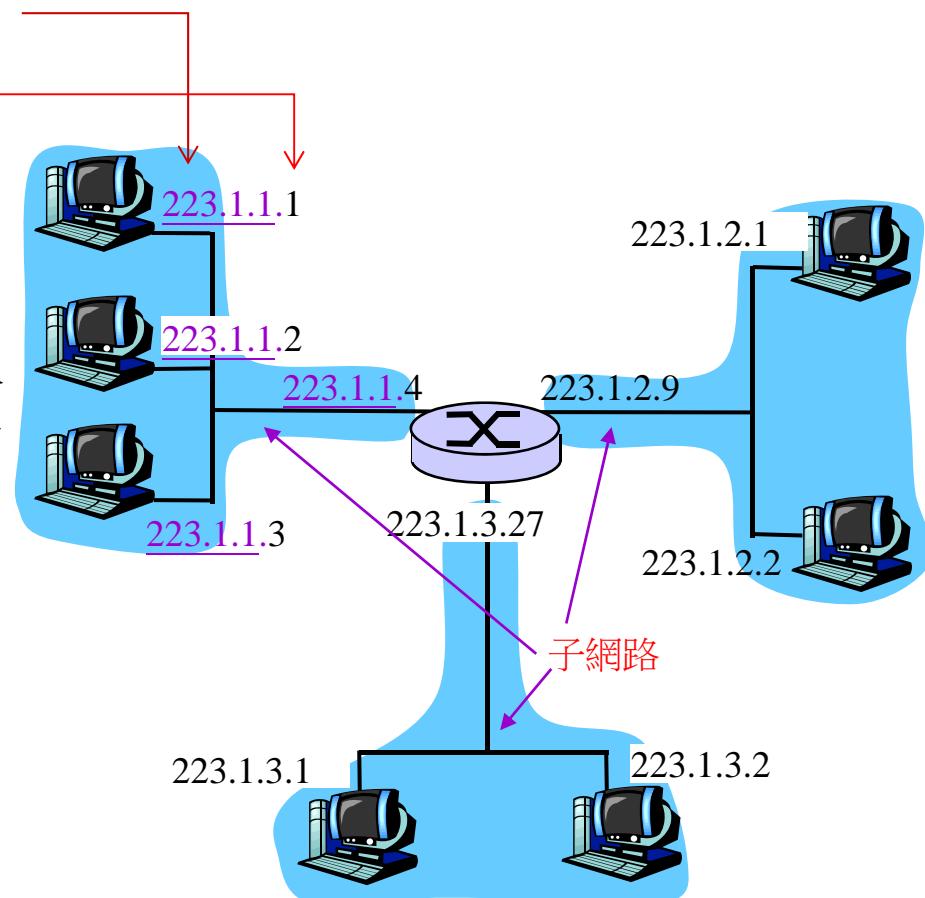


圖4.15(b) 包含 3 個子網路的網路

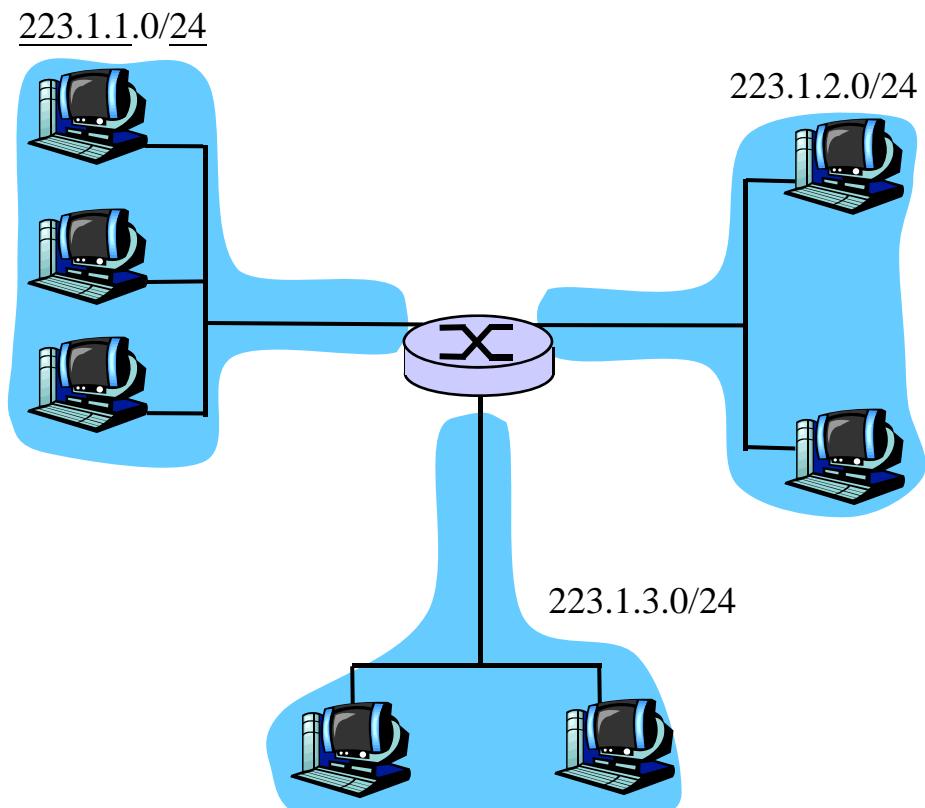




■ 子網路 (Cont.)

► 表示方法：

- 要決定子網路，將每一個介面和它的主機或路由器分開，建立分離的網路。每一個分離的網路稱為一個子網路。
- /24: 最左邊的24 bits 定義子網路的位址



子網路遮罩： /24

圖4.16 子網路位址





■ 子網路 (Cont.)

► 有幾個子網路？

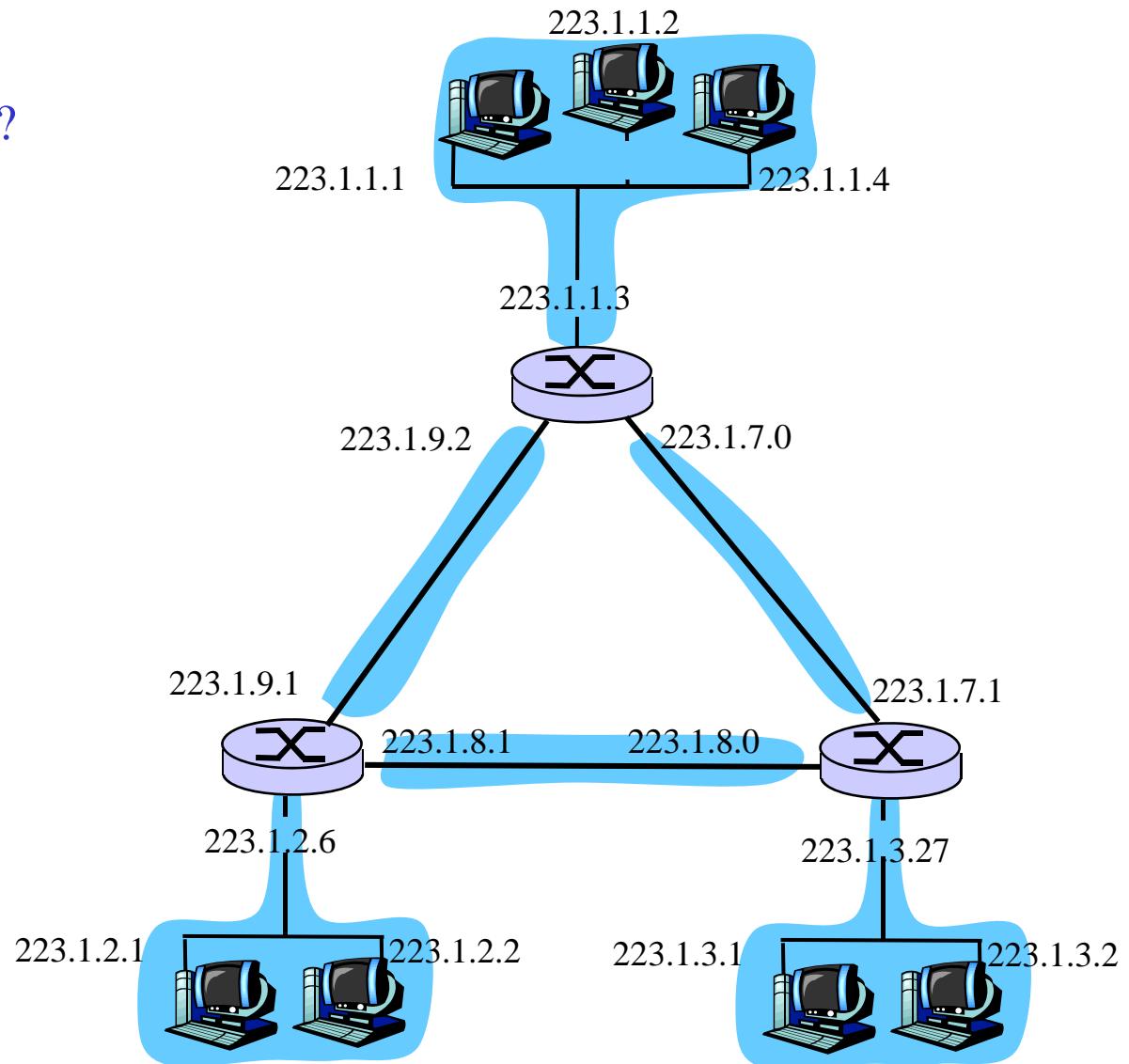


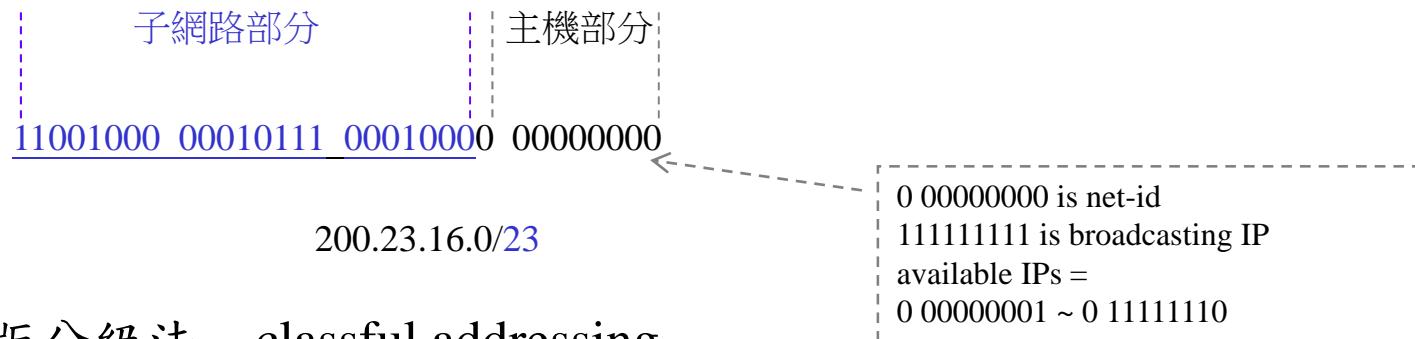
圖4.17 連接到6個子網路的三台路由器



■ IP 位址分配策略：CIDR

Hung2 ▶ CIDR：無分級跨網域路由法 (Classless InterDomain Routing)

- 位址內任意長度的子網路部分
- 位址格式： $a.b.c.d/x$ ，其中 x 是位址的子網路部份的位元數



▶ 舊版分級法 – classful addressing

- 8、16、24位元的A、B、C級子網路
- C級子網路的網路遮罩=/24，共可容納 $2^8 - 2$ 個IP位址 (0=net-id, broadcast=255)
- A、B級子網路，共可容納?個IP位址

▶ 廣播的IP位址(broadcasting) = **255.255.255.255**



投影片 44

Hung2

CIDR: 唸做"cider"
Ruo-Wei Hung, 2011/8/7



■ 如何取得/設定主機IP位址？

- ▶ 手動設定在系統管理的檔案中
 - MS Windows : [控制台]→[網路連線]→[設定]→tcp/ip→[內容]
 - UNIX : /etc/rc.config
- ▶ DHCP：動態主機配置協定 (Dynamic Host Configuration Protocol)
 - 動態地由DHCP伺服器取得位址
 - “隨插即用”的協定(plug-and-play protocol)
 - 取得臨時IP位址





■ DHCP (動態主機配置協定)

▶ 目標

- 當主機連接網際網路時，主機會從網際網路伺服器以**動態方式**獲得它的IP位址
- 在使用時會更新它的IP位址
- 允許網路位址動態地重複使用（當連結“上”時會鎖住這個臨時網路位址）
- 提供使用者使用手機來連接網際網路

▶ DHCP 運作

- 主機廣播 “DHCP搜尋訊息”（廣播的IP位址=255.255.255.255）
- DHCP 伺服器會回應 “DHCP提供訊息”
- 主機請求IP位址： “DHCP請求訊息”
- DHCP 伺服器傳送位址： “DHCP ack” 訊息



■ DHCP (Cont.)

► DHCP 用 戶 端 - 同 服 端 情 境

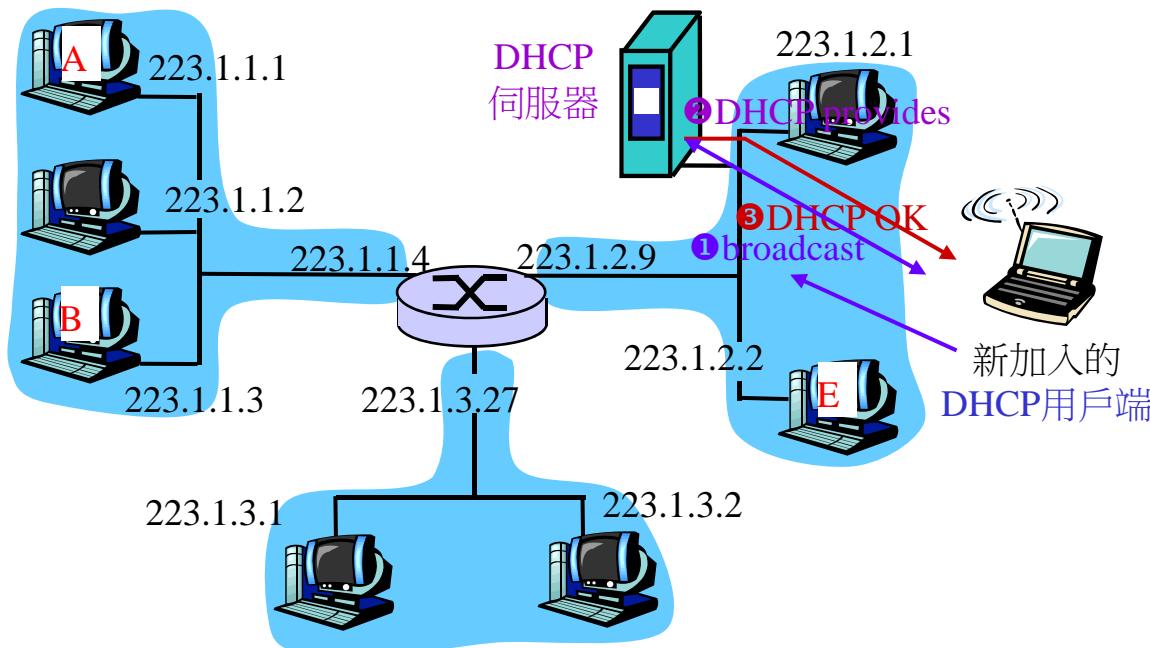
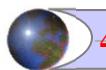


圖4.20 DHCP用 戶 端 - 同 服 端 的 運 作 流 程





■ DHCP (Cont.)

► DHCP 用 戶 端 - 同 服 端 情 境 (Cont.)

DHCP 同服端：223.1.2.5

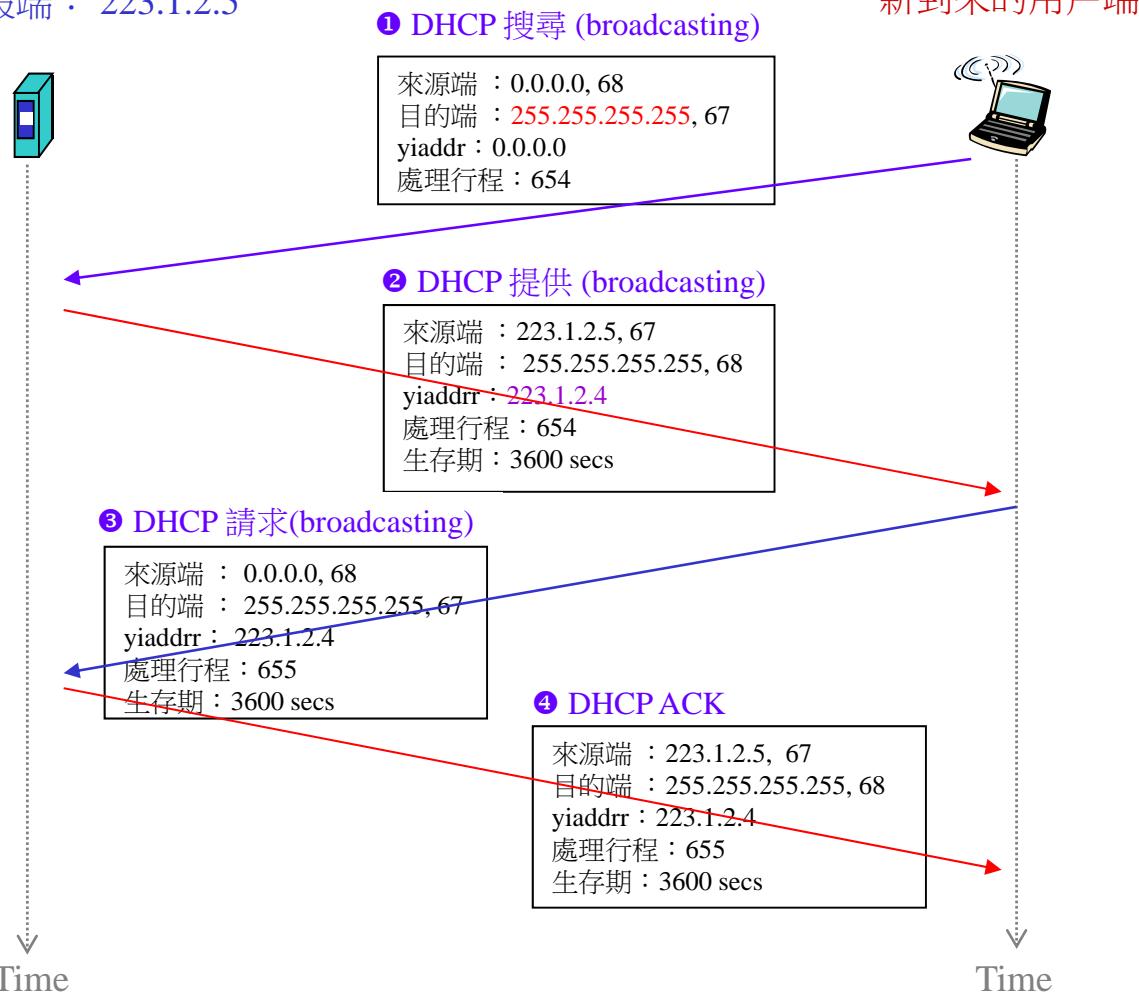
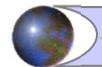


圖4.21 DHCP用 戶 端 - 同 服 端 的 互 動

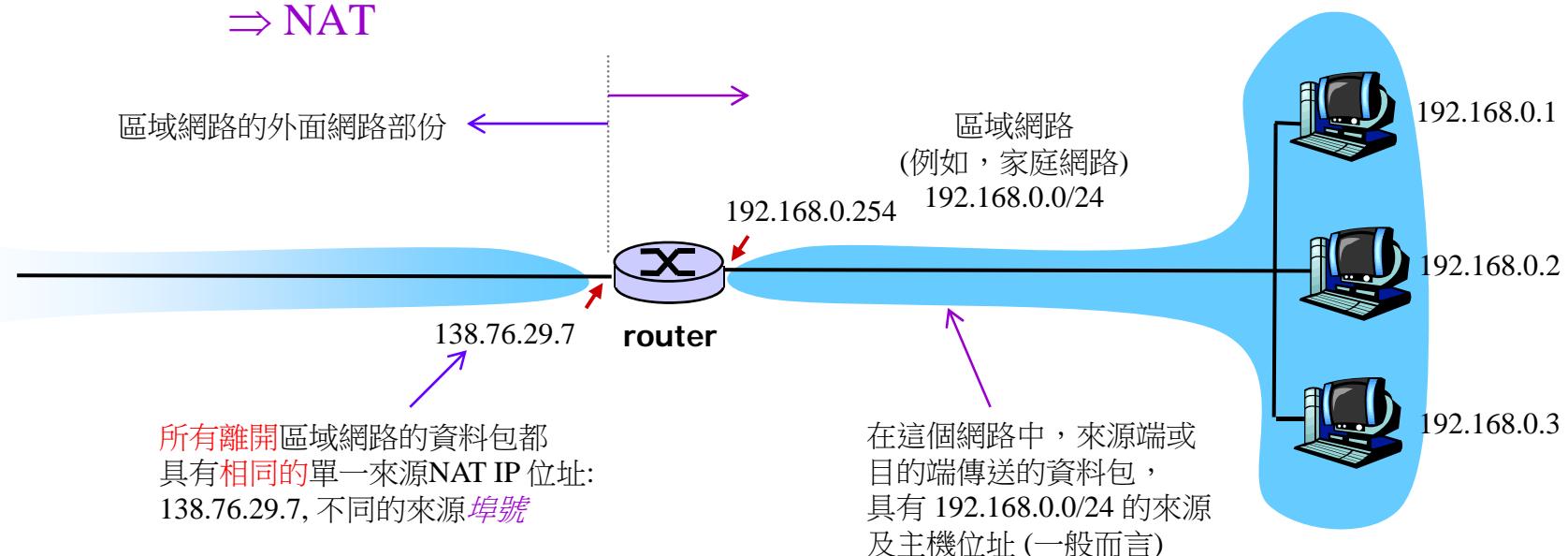




■ NAT – 網路位址轉譯 (Network Address Translation)

► 動機

- 整個區域網路對外界而言，通常只有一個IP位址
- 同一區域網路內的主機，可由router來設定其IP位址
- 所有區域網路內的主機送出的封包均為同一IP位址
- 外界送入的封包也是同一IP位址
- Router如何將外界送入的封包，正確傳送給區域網路內的主機？
⇒ NAT

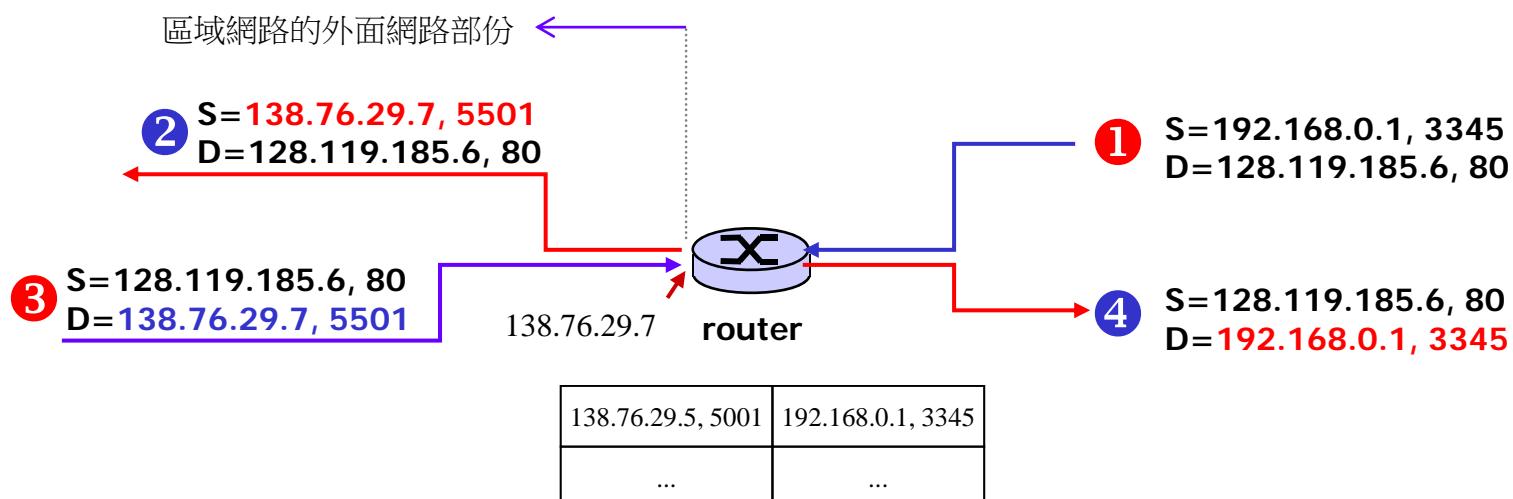




■ NAT (Cont.)

► 實作 - NAT 路由器必須：

- 送出資料(封)包時：更換每一個送出的資料包 (來源IP位址、埠號) 成為 (NAT IP 位址、新埠號)
... 遠端的主機(區網外)會使用 (NAT IP 位址、新埠號) 做為目的端位址，來做回應
- 記憶 (在 NAT 轉譯表中) 每一個 (來源端 IP 位址、埠號) 到 (NAT IP 位址、新埠號) 的轉譯對應
- 進入資料包時：更換每個進入資料封包目的欄位中的 (NAT IP 位址、新埠號) 為儲存在 NAT 表中的對應 (來源端 IP 位址、埠號)





■ NAT (Cont.)

► 實作

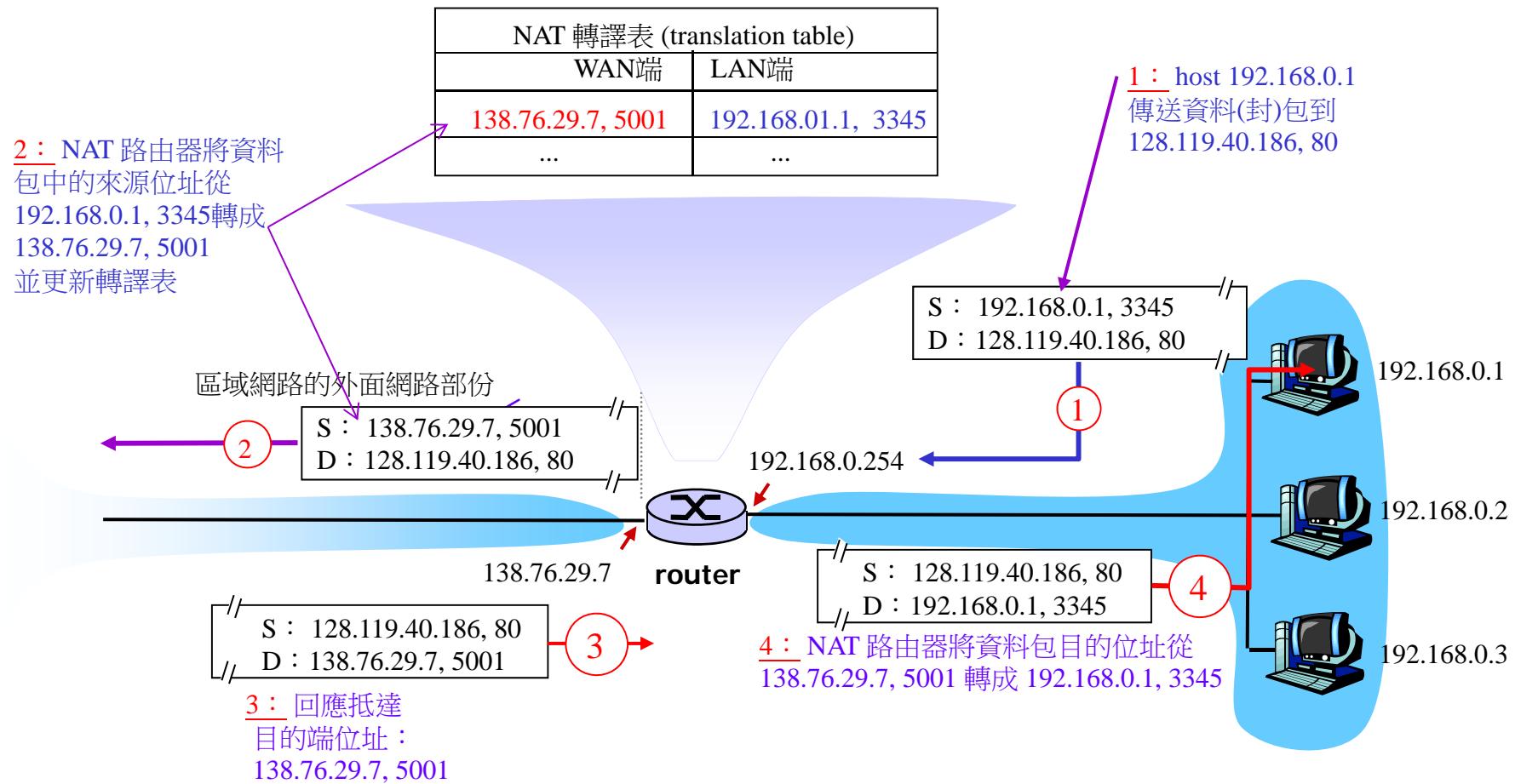
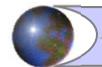


圖4.22 NAT網路位址轉譯

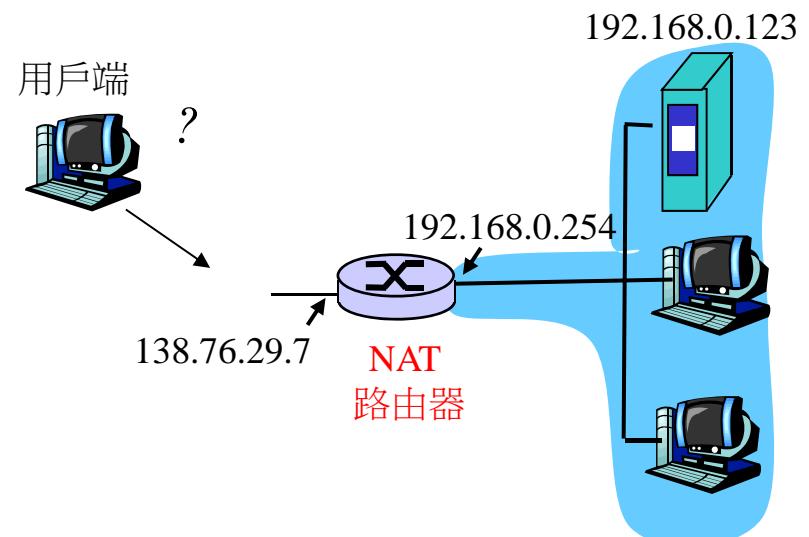




■ NAT的穿越問題 (traversal)

► What ?

- 網外用戶端要連接到伺服器的 192.168.0.123位址
 - 伺服器位址 192.168.0.123 區域連線到 LAN
(用戶端不能使用它的目標位址)
 - 只能有一個外部可見的NATTed 位址： 138.76.29.7

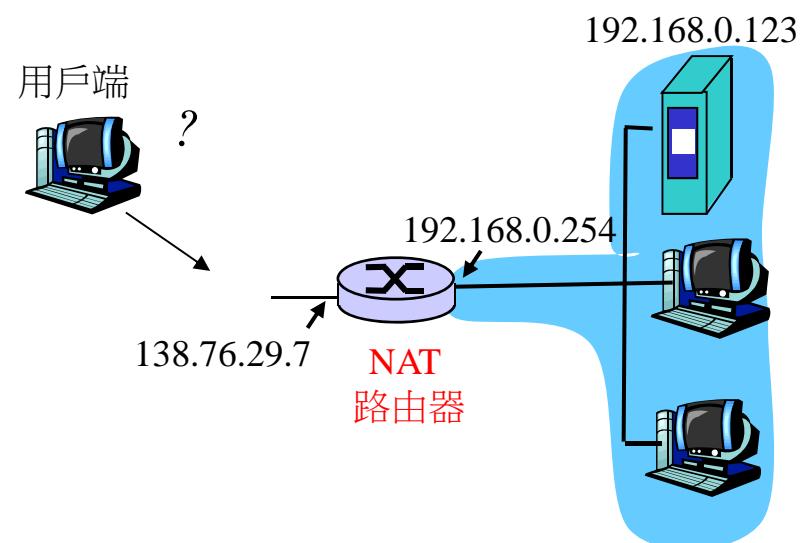


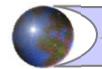


■ NAT的穿越問題 (Cont.)

► How ?

- 解1：靜態配置NAT給予埠號並請求連結向前進入到伺服器
 - 例如：(138.76.29.7、埠號2500) 永遠向前到 (192.168.0.123、埠號25000)





■ NAT的穿越問題 (Cont.)

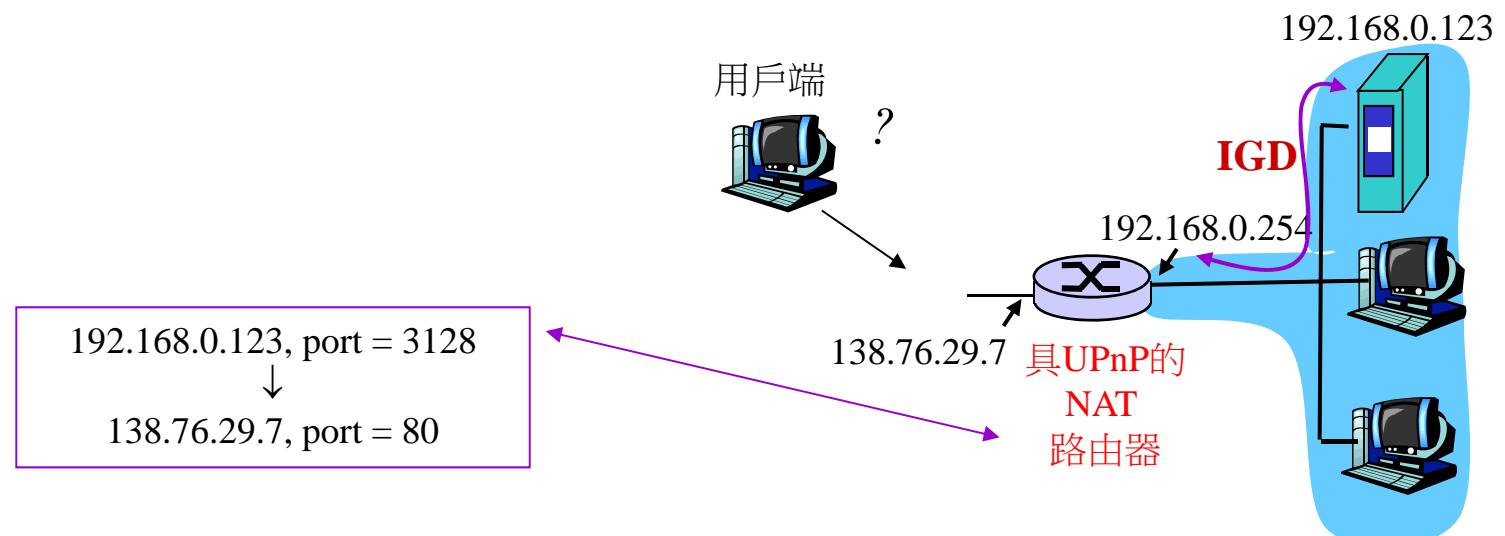
► How ? (Cont.)

Hung3

○ 解2：通用型隨插即用 (UPnP) 網際網路閘道裝置 (IGD) 協定

允許NATted 主機到：

- 學習公用IP位址(138.76.29.7)
- 列舉現有的對映埠號
- 增加/移除對映的埠號(租借時間)
- i.e. 自動配置靜態NAT的對映埠號



投影片 54

Hung3

Universal Plug and Play
Ruo-Wei Hung, 2011/8/8



■ ICMP – 網際網路訊息控制協定

- ▶ 主機和路由器間互相交換網路層資訊的協定
 - 錯誤回報：無法到達主機、網路、埠號、協定
 - 回應請求/回覆 (由ping所使用)
- ▶ ICMP通常被視為IP的一部份
 - ICMP 訊息在IP封包中載送
- ▶ ICMP 訊息
 - 類型、代碼、加上導致錯誤的 IP 資料封包的前八個位元組

類型	代碼	說明
0	0	回應訊息 (ping)
3	0	目的端網路無法連上
3	1	目的端主機無法連上
3	2	目的端協定無法連上
3	3	目的端埠無法連上
3	6	目的端網路未知
3	7	目的端主機未知
4	0	來源抑制訊息 (壅塞控制 – 未使用)
8	0	回應請求 (ping)
9	0	路由器通告
10	0	路由器搜尋中
11	0	TTL 過期
12	0	IP標頭損毀

圖4.23 ICMP 訊息類型

投影片 55

Hung4

Internet Control Message Protocol

Ruo-Wei Hung, 2011/8/8



■ Traceroute 與 ICMP

- ▶ 來源端傳送一系列的 UDP 資料區段給目的端

Hung5

- 第一個具有 TTL = 1
- 第二個具有 TTL = 2、依此類推
- 不可能的埠號

- ▶ 當第 n 個資料段抵達第 n 個路由器：

- 路由器刪除資料段
- 接著傳送一個 ICMP 訊息給來源端 (類型 11、代碼 0)
- 訊息中包含了路由器名稱以及 IP 位址

- ▶ 當 ICMP 訊息抵達時，來源端會計算 RTT

- ▶ Traceroute 會重複這個步驟三次

- ▶ 停止的標準

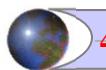
- UDP 資料區段最終會到達目的主機
- 目的端會回傳 ICMP “主機無法到達” 封包 (類型 3、代碼 3)
- 當來源端收到這個 ICMP 時，則會停止

投影片 56

Hung5

Time To Live

Ruo-Wei Hung, 2011/8/8



■ IPv6

▶ 第6版IP協定

▶ 動機

- 32位元的位址空間 (IPv4) 很快就要使用殆盡了
- 標頭格式可以幫助處理/轉送更快速
- 標頭的改變可以幫助QoS (Quality of Service)

▶ IPv6資料封包格式

- 固定長度的**40位元組**標頭
- 不允許切割 (4.4.1節)



圖4.24 IPv6 資料包格式



■ 從 IPv4 轉換成 IPv6

► 動機

- 並不是所有的路由器可以同時升級
- 無法使用“揭旗日”的方式，將IPv4全部轉成IPv6
- 假如網路的運作混和了IPv4和IPv6的路由器會如何？←混亂

► 建立通道

- 在 IPv4 資料包的承載資料中，封裝完整的IPv6資料包、送往 IPv6 路由器

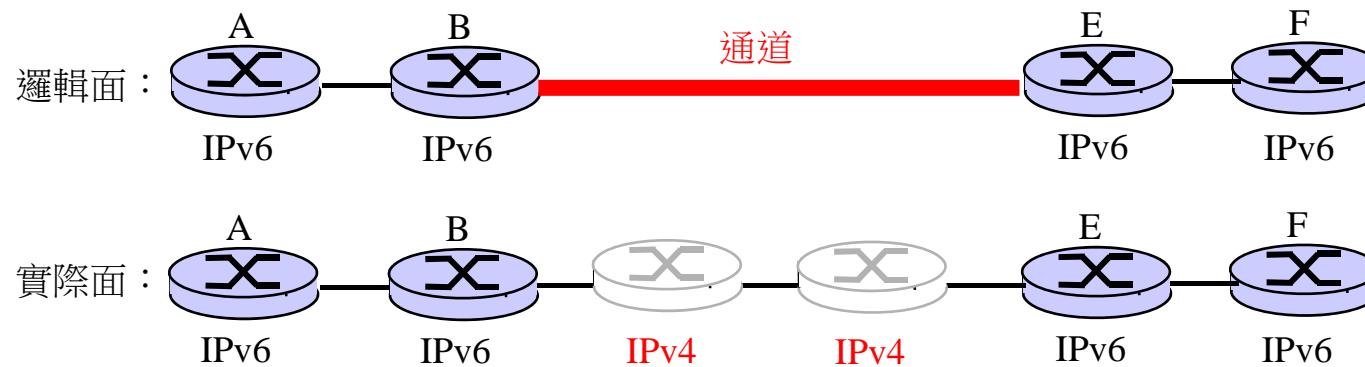
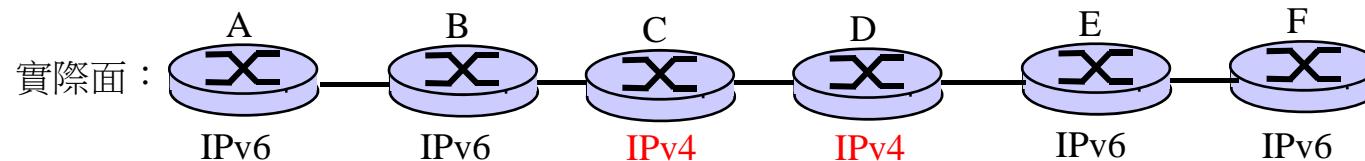


圖4.26 IPv6 融入IPv4 的通道建立



■ 從 IPv4 轉換成 IPv6 (Cont.)

► 建立通道 (Cont.)



資料流：X
來源端：A
目的端：F
資料

來源端：B
目的端：E
資料流：X
來源端：A
目的端：F
資料

來源端：B
目的端：E
資料流：X
來源端：A
目的端：F
資料

資料流：X
來源端：A
目的端：F
資料

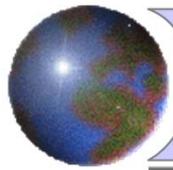
A-到-B :
IPv6

B-到-C :
IPv6 封裝在
IPv4裡面

D-到-E :
IPv6 封裝在
IPv4 裡面

E-到-F :
IPv6

Exercise
4.6 Exercise
4.4
Exercise
4.5 Exercise
4.3



第四章 導覽流程



4.1 簡介

4.2 虛擬線路網路和資料包網路

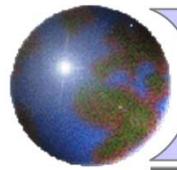
4.3 路由器的內部

4.4 網際網路IP協定 – 網際網路的轉送(forward)及定址

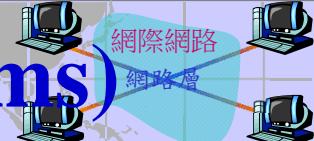
4.5 路由演算法 (routing algorithms)

4.6 網際網路的路由

4.7 廣播與群播路由 (broadcast and multicast)



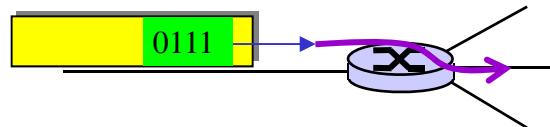
4.5 路由演算法 (routing algorithms)



■ 路由(routing) vs. 轉送(forward)

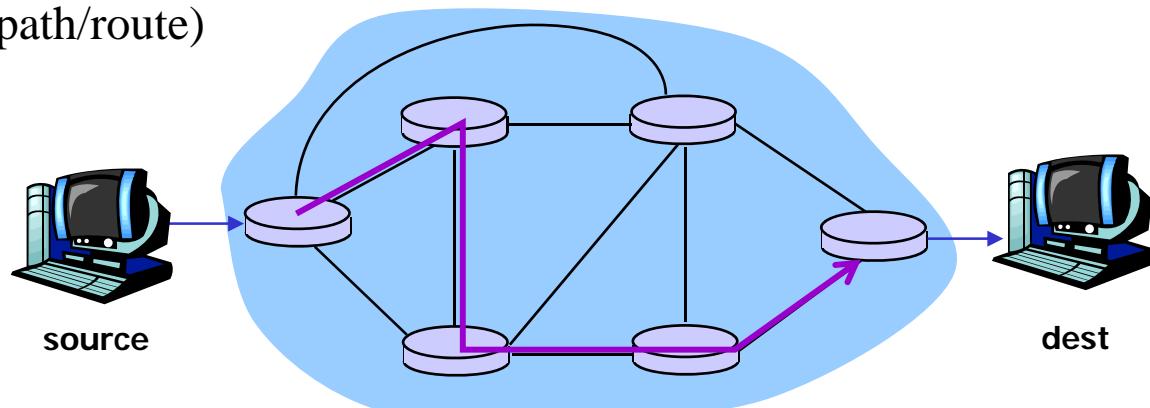
▶ 轉送

- 封包到達router時，router會查詢其轉送表(forwarding table)並決定將該封包送網哪一個連結介面(link interface)



▶ 路由(routing)

- 網路層必需決定封包從傳送端router到接收端router所採取的路徑(path)
- routing(繞徑)的工作在於判斷傳送端到接收端，通過路由器網路的 **良好路徑**(path/route)
Hung6



投影片 61

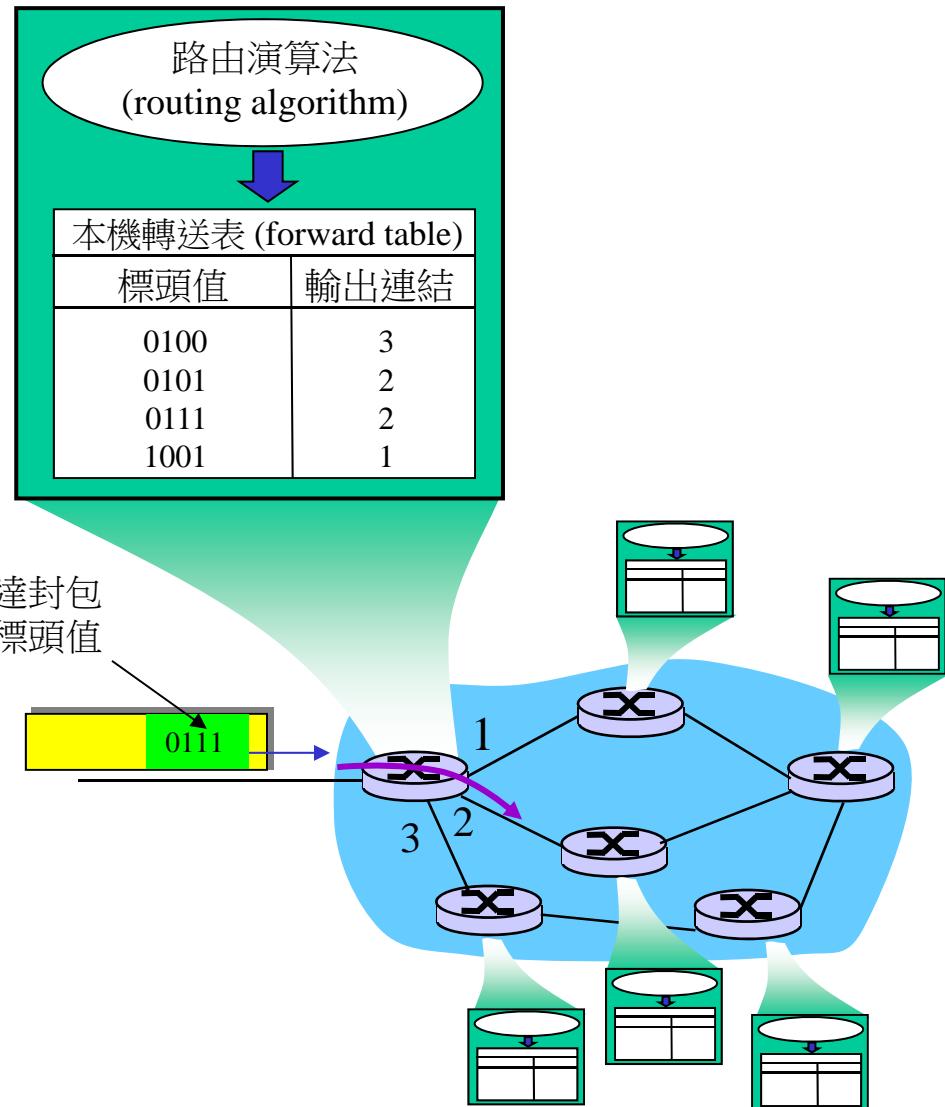
Hung6

良好路徑 = 成本最小路徑

Ruo-Wei Hung, 2011/8/8



■ 路由和轉送的交互作用





■ 抽象化圖形 (abstract graph)

► 圖形： $G = (N, E)$

- N = 一組路由器的集合 (node set) = { u, v, w, x, y, z }
- E = 一組連結的集合 (edge set) = { $(u, v), (u, x), (v, x), (v, w), (x, w), (x, y), (w, y), (w, z), (y, z)$ }

► [註] 抽象化圖形在其它的網路應用上也是很有用的

- 例如：P2P中， N 為對等點的集合、而 E 為TCP連線的集合

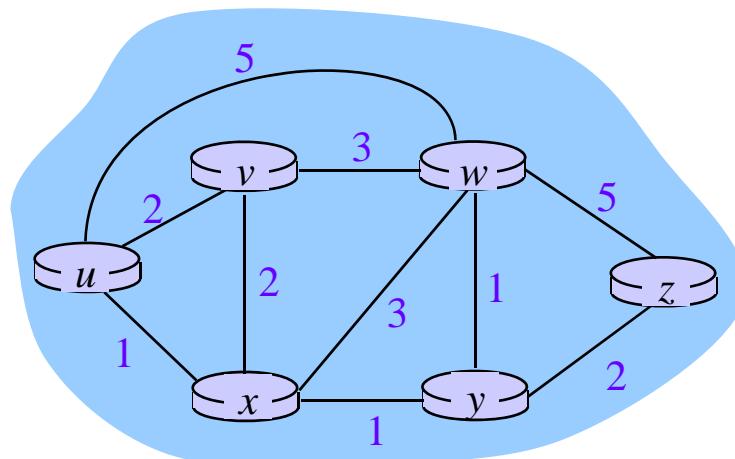


圖4.27 電腦網路的抽象化圖形模型



■ 抽象化圖形 (Cont.)

▶ 成本 (cost, weight)

- $c(x, y) =$ 連結 (x, y) 的成本(權重)
 - 例如， $c(w, z) = 5$
- 成本可能永遠是 1、或是相反地與頻寬相關、或是相反地與壅塞程度相關
- 路徑 $(x_1, x_2, x_3, \dots, x_p)$ 的成本 $= c(x_1, x_2) + c(x_2, x_3) + \dots + c(x_{p-1}, x_p)$

▶ 問題： u 和 z 之間的最小成本路徑為何？

▶ 路由演算法：尋找最小成本路徑的演算法

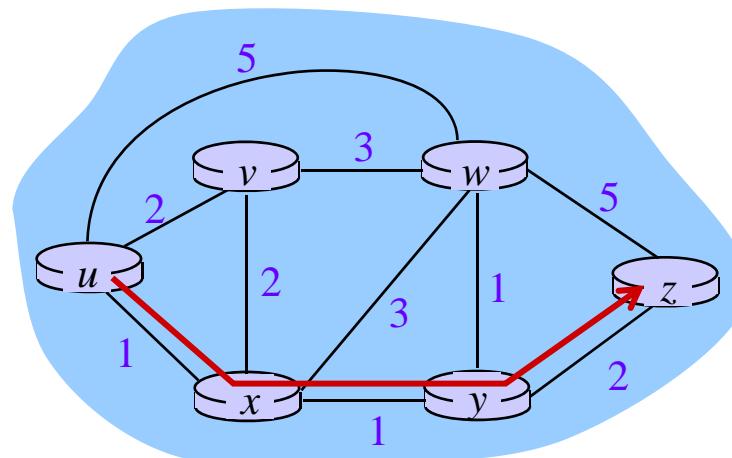


圖4.27 電腦網路的抽象化圖形模型



■ 路由演算法的分類

► 整體或分散式資訊? (*global vs. distributed-decentralized*)

○ 整體的 (*global*)

- 所有的路由器都擁有完整的拓樸(topology)及連結成本(link cost)資訊
- “連結狀態” 演算法 (link-state, LS)
- 最短路徑計算在某台電腦或多台固定電腦中執行 – centralized

○ 分散式 (*distributed*)

- 路由器只知道實體連結的鄰居以及到鄰居的連結成本
- 最短路徑經由分散反覆地計算，以及與鄰居交換資訊
- “距離向量” 演算法 (distance-vector, DV)

► 靜態的或動態的? (*static vs. dynamic*)

○ 靜態的 (*static*)

- 路徑隨著時間緩慢地改變

○ 動態的 (*dynamic*)

- 較快速地改變路徑
- 週期性的更新
- 回應連結成本的改變



■ Dijkstra 演算法

- ▶ 已知所有節點的網路拓樸、連結成本
 - 經由「連結狀態廣播」完成 (broadcasting algorithm)
 - 每個節點都擁有一樣的資訊
- ▶ 從一個節點(來源端)開始 → 到所有其它節點，計算最小成本路徑 (single source shortest paths)
 - 紿定此節點的轉送表
- ▶ 循環式的(iterative)：在 k 次循環以後，會知道 k 個目的節點的最小成本路徑
- ▶ 符號：
 - $c(x, y)$ ：從節點 x 到 y 的連結成本； $=\infty$ 假如不是直接連結(非鄰居)
 - $D(v)$ ：目前從來源端到目的端 v 的最小成本路徑的成本值
 - S ：一組已計算最小成本路徑的節點集合



■ Dijkstra 演算法 (Cont.)

// Initialization

- 1 $S = \{u\}$; //起始節點 u
- 2 **for** all nodes v **do**
- 3 **if** v adjacent to u then $D(v) = c(u, v)$;
- 4 **else** $D(v) = \infty$;
- 5 **Repeat**
- 6 find w not in S such that $D(w)$ is a minimum;
- 7 add w to S ;
- 8 **for** all v adjacent to w and not in S **do**
- 9 update $D(v)$ by the following:
10 $D(v) = \min(D(v), D(w) + c(w, v))$;
 // new cost to v is either old cost to v or known
 // shortest path cost to w plus cost from w to v
- 11 **Until** all nodes in S ;



■ Dijkstra 演算法 (Cont.)

► 範例

① Initially,

$$S = \{u\};$$

w^*	v	x	w	y	z
$D(w^*)$	2	1	5	∞	∞

② Minimum $D(\cdot) \Rightarrow x$

$$S = \{u, \textcolor{blue}{x}\};$$

w^*	v	x	w	y	z
$D(w^*)$	2	1	4	2	∞

③ Minimum $D(\cdot) \Rightarrow v$

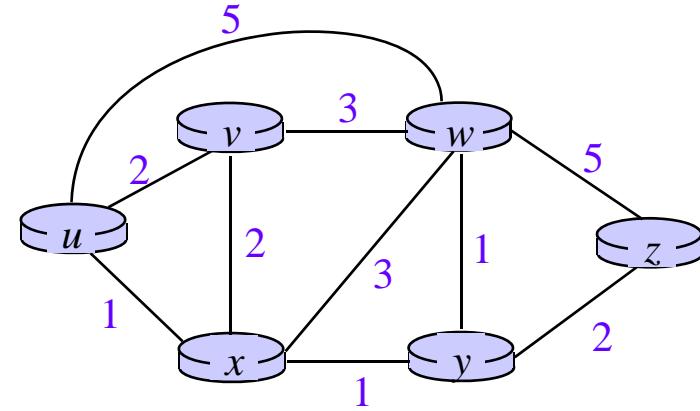
$$S = \{u, x, \textcolor{blue}{v}\};$$

w^*	v	x	w	y	z
$D(w^*)$	2	1	4	2	∞

④ Minimum $D(\cdot) \Rightarrow y$

$$S = \{u, x, v, \textcolor{blue}{y}\};$$

w^*	v	x	w	y	z
$D(w^*)$	2	1	3	2	4



⑤ Minimum $D(\cdot) \Rightarrow w$

$$S = \{u, x, v, y, \textcolor{blue}{w}\};$$

w^*	v	x	w	y	z
$D(w^*)$	2	1	3	2	4

⑥ Minimum $D(\cdot) \Rightarrow z$

$$S = \{u, x, v, y, w, \textcolor{blue}{z}\};$$

w^*	v	x	w	y	z
$D(w^*)$	2	1	3	2	4



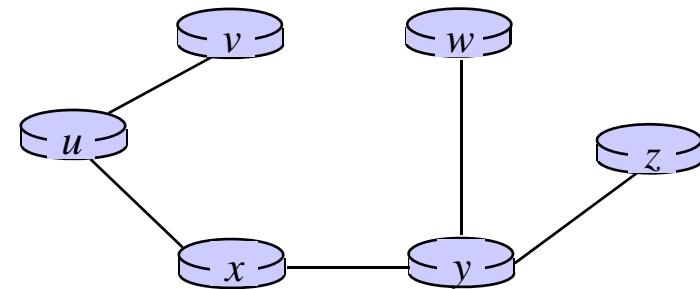


■ Dijkstra 演算法 (Cont.)

► 範例 (Cont.)

○ 結果

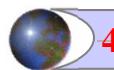
– 從 u 開始的最短路徑樹：



– u 的轉送表：

目的	連結
v	(u, v)
x	(u, x)
y	(u, x)
w	(u, x)
z	(u, x)





■ Dijkstra 演算法 (Cont.)

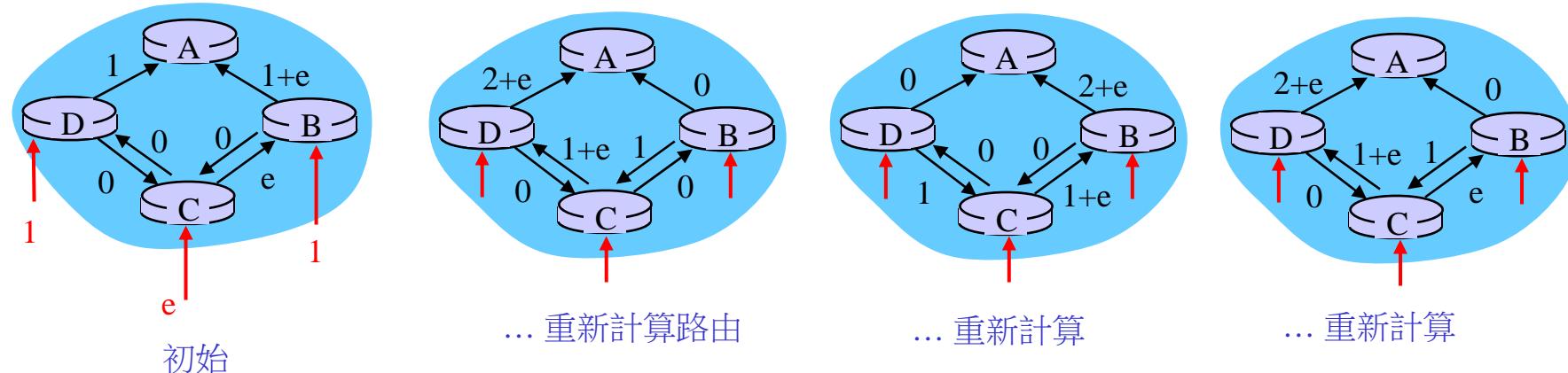
► 討論

○ 演算法複雜度： n 個節點

- 每個迴圈：需要確認不在 S 中的每個節點 w
- $n(n+1)/2$ 個比較： $O(n^2)$
- 可能更有效率的實作： $O(n \log n)$

○ 可能的問題 – 震盪現象 (圖4.29)

- 連結成本 = 該連結的載送負荷
- 連結的成本可能是不對稱的 ($\rightarrow \neq \leftarrow$)





■ 距離向量演算法 (Distance-Vector)

► 概念

○ 分散式

- 各個節點會從一個或多個直接相鄰的節點中取得一些資訊，進行運算，然後，將結果散佈給其相鄰節點

○ 循環式

- 持續運作，直到沒有資訊可以交換

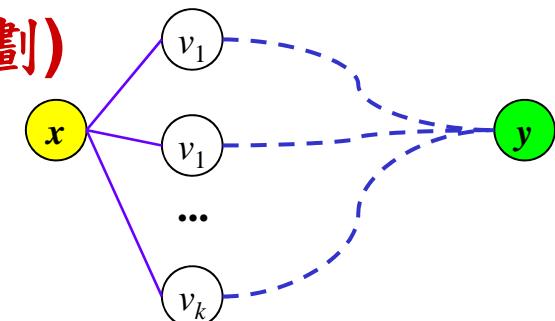
○ 非同步

- 不需所有節點同時運作

■ Bellman-Ford 方程式 (動態程式規劃)

$$\boxed{d_x(y) = \min_v \{ c(x, v) + d_v(y) \}}$$

- $d_x(y)$ = 從 x 到 y 之最小成本路徑的成本
- \min_v 符號作用於 x 的所有相鄰節點 v

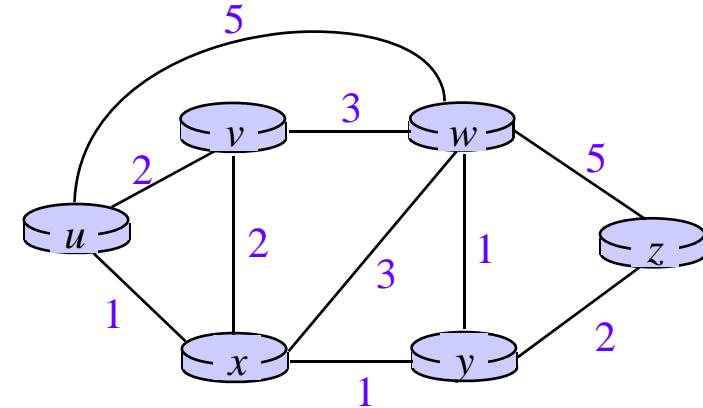




■ Bellman-Ford 方程式

► 範例

- $d_v(z) = 5$ 、 $d_x(z) = 3$ 、 $d_w(z) = 3$
- $d_u(z) = \min \{ c(u, v) + d_v(z), c(u, x) + d_x(z), c(u, w) + d_w(z) \}$
 $= \min \{ 2 + 5, 1 + 3, 5 + 3 \} = 4$



- 可達成最小路徑的節點是最短路徑上的下一站 \Rightarrow 轉送表



■ 距離向量演算法

► 符號

- $D_x(y)$ = 從 x 到 y 的預估最小成本
- 節點 x 知道到每一個鄰居 v 的成本： $c(x, v)$
- 節點 x 維護 距離向量： $\mathbf{D}_x = [D_x(y) : y \in N]$
- 節點 x 也維護它的鄰居的距離向量
 - 對每一個鄰居 v ， x 維護
 $\mathbf{D}_v = [D_v(y) : y \in N]$

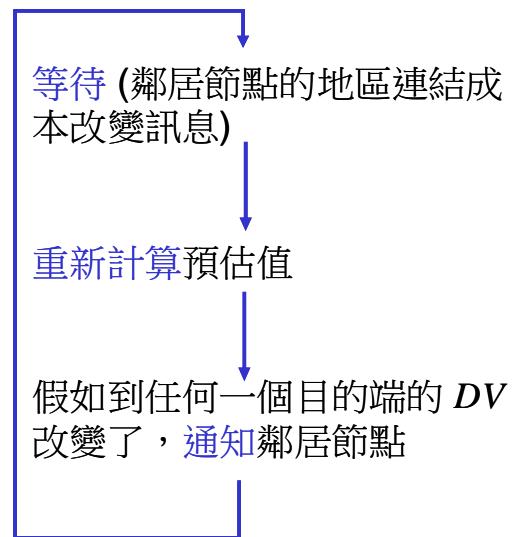


■ 距離向量演算法 (Cont.)

► 基本想法

- 每一個節點定期地傳送它自己的距離向量預估值給鄰居節點
- 當一個節點 x 收到來自鄰居的新 D_v 預估值，它會使用 Bellman-Ford 方程式更新它自己的 D_v 向量：
$$D_x(y) = \min_v \{ c(x, v) + D_v(y) \}, \forall y \in N$$
- 在一般的自然狀況下，預估的 $D_x(y)$ 會涵蓋真正的最小成本 $d_x(y)$
- 每一個節點的運作：

► DV演算法





■ 距離向量演算法 (Cont.)

► 範例

節點 *x*

$$\begin{aligned} D_x(y) &= \min\{c(x, y) + D_y(y), \\ c(x, z) + D_z(y)\} = \min\{2+0, \\ 7+1\} = 2 \end{aligned}$$

目的節點

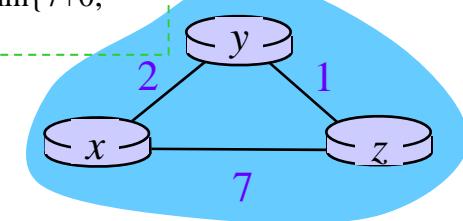
	<i>x</i>	<i>y</i>	<i>z</i>
<i>x</i>	0	2	7
<i>y</i>	∞	∞	∞
<i>z</i>	∞	∞	∞

來源節點

目的節點

	<i>x</i>	<i>y</i>	<i>z</i>
<i>x</i>	0	2	3
<i>y</i>	2	0	1
<i>z</i>	7	1	0

$$\begin{aligned} D_x(z) &= \min\{c(x, z) + D_z(z), \\ c(x, y) + D_y(z)\} = \min\{7+0, \\ 2+1\} = 3 \end{aligned}$$



目的節點

	<i>x</i>	<i>y</i>	<i>z</i>
<i>x</i>	0	2	3
<i>y</i>	2	0	1
<i>z</i>	3	1	0

來源節點

節點 *y*

目的節點

	<i>x</i>	<i>y</i>	<i>z</i>
<i>x</i>	∞	∞	∞
<i>y</i>	2	0	1
<i>z</i>	∞	∞	∞

來源節點

目的節點

	<i>x</i>	<i>y</i>	<i>z</i>
<i>x</i>	0	2	7
<i>y</i>	2	0	1
<i>z</i>	7	1	0

來源節點

節點 *z*

目的節點

	<i>x</i>	<i>y</i>	<i>z</i>
<i>x</i>	∞	∞	∞
<i>y</i>	∞	∞	∞
<i>z</i>	7	1	0

來源節點

目的節點

	<i>x</i>	<i>y</i>	<i>z</i>
<i>x</i>	0	2	7
<i>y</i>	2	0	1
<i>z</i>	3	1	0

來源節點



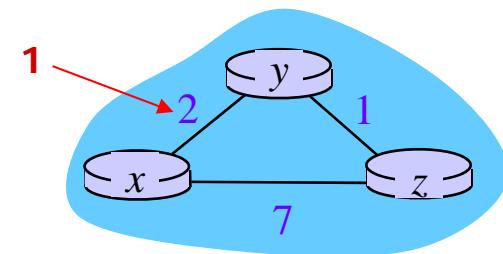


■ 距離向量演算法 (Cont.)

► 連結成本改變

- 節點偵測到區域連結成本的改變
- 更新路由資訊，重新計算距離向量
- 假如 DV 變更，通知鄰居節點

- 連結成本改變情境
 - 在時間點 t_0 ， y 偵測到連結成本的變更，它會更新距離向量，並通知相鄰節點
 - 在時間點 t_1 ， z 接收來自 y 的更新訊息，更新自己的表格。它會計算出到 x 的新最小成本，並傳送給相鄰節點
 - 在時間點 t_2 ， y 會接收到 z 的更新訊息，並且更新自己的距離表格。因為 y 的最小成本並沒有改變，所以 y 不會傳送任何訊息給 z





■ LS 與 DV 演算法的比較

▶ 訊息複雜度

- LS : n 個節點、 E 條連結 $\Rightarrow O(nE)$ 個傳送出去的訊息
- DV : 只在鄰居之間交換 \Rightarrow 收斂時間不固定

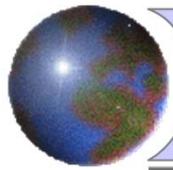
▶ 收斂速度

- LS : $O(n^2)$ 演算法需要 $O(nE)$ 個訊息 \Rightarrow 可能有擺動
- DV : 收斂速度不固定 \Rightarrow 可能產生路由迴圈、計算到無窮大的問題

▶ 強韌性：假如路由器故障會發生什麼問題？

- LS：
 - 節點可能會廣播不正確的連結成本
 - 每個節點只計算它自己的轉送表
- DV：
 - DV 節點可能會廣播不正確的路徑成本
 - 每個節點的轉送表也被其他人所使用 \Rightarrow 將錯誤傳播到網路中

▶ 目前唯二被實際使用在網際網路中的路由演算法



第四章 導覽流程



4.1 簡介

4.2 虛擬線路網路和資料包網路

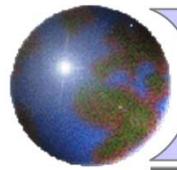
4.3 路由器的內部

4.4 網際網路IP協定 – 網際網路的轉送(forward)及定址

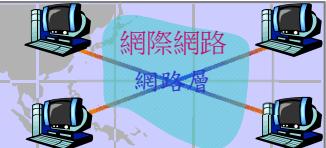
4.5 路由演算法 (routing algorithms)

4.6 網際網路的路由

4.7 廣播與群播路由 (broadcast and multicast)



4.6 網際網路的路由



■ 路由資訊協定 (Routing Information Protocol, RIP)

- ▶ 距離向量演算法
- ▶ 包含在1982年的 BSD-UNIX Distribution 中
- ▶ 距離計量：hop數目（最大 = 15 個 hops）
 - hop count = cost

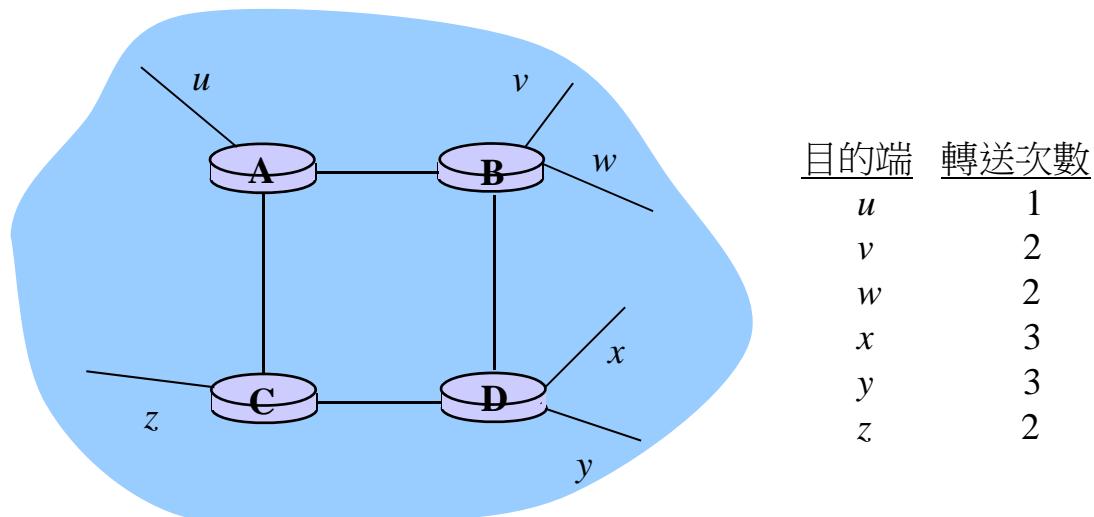


圖4.34 從來源路由器A到各個子網路的hop數(站數)



■ RIP 廣播

- ▶ 距離向量：每隔30秒使用回應訊息(也稱做通告)、在相鄰節點間交換
- ▶ 每個通告：自治網路系統內最多達25個目的端子網路的清單

■ RIP 範例

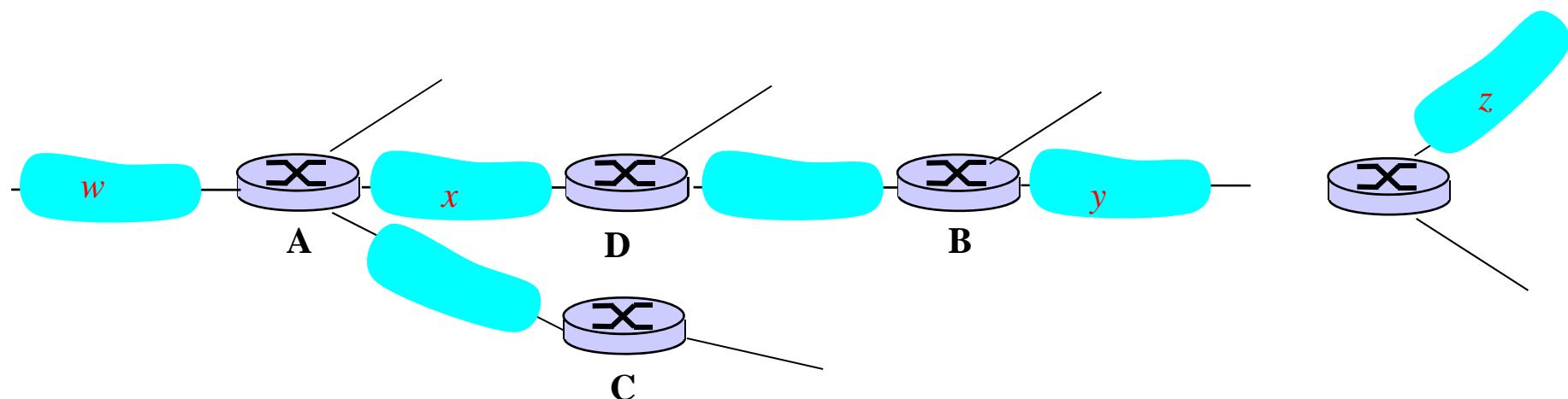
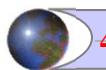


圖4.35 RIP 範例



■ RIP 範例 (Cont.)

- D 中的路由表(routing table)

目的端網路	下一個路由器	到目的端所需的轉送次數
w	A	2
y	B	2
z	B	7
x	--	1
...

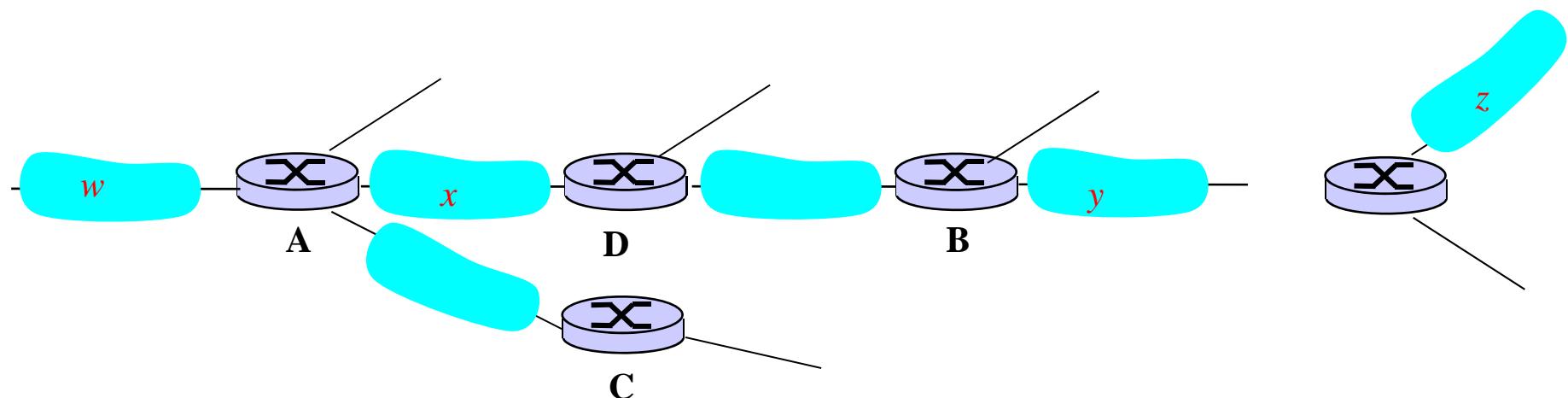


圖4.35 RIP 範例



■ RIP 範例 (Cont.)

► D 中的路由表

目的端網路	下一個路由器	到目的端所需的轉送次數
w	A	2
y	B	2
z		
x		
...		

從A到D的通告：

目的端	下一個路由器	轉送次數
w	--	1
x	--	1
z	C	4
...

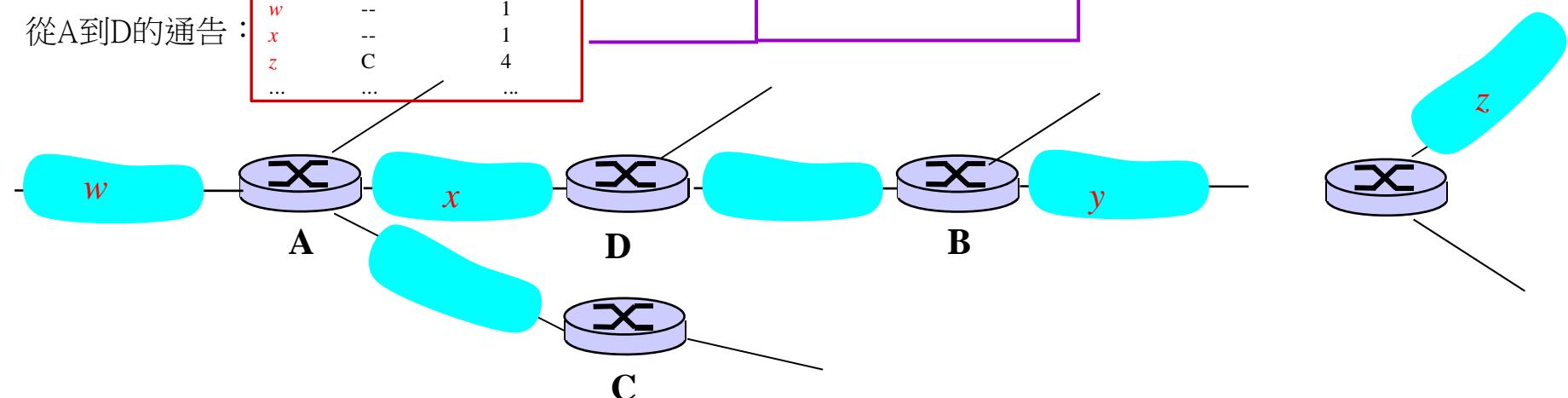
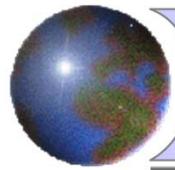


圖4.35 RIP 範例

Ex 4.12



第四章 導覽流程



4.1 簡介

4.2 虛擬線路網路和資料包網路

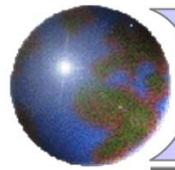
4.3 路由器的內部

4.4 網際網路IP協定 – 網際網路的轉送(forward)及定址

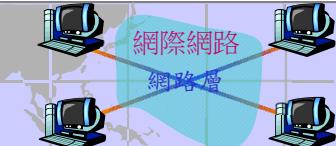
4.5 路由演算法 (routing algorithms)

4.6 網際網路的路由

4.7 廣播與群播路由 (broadcast and multicast)

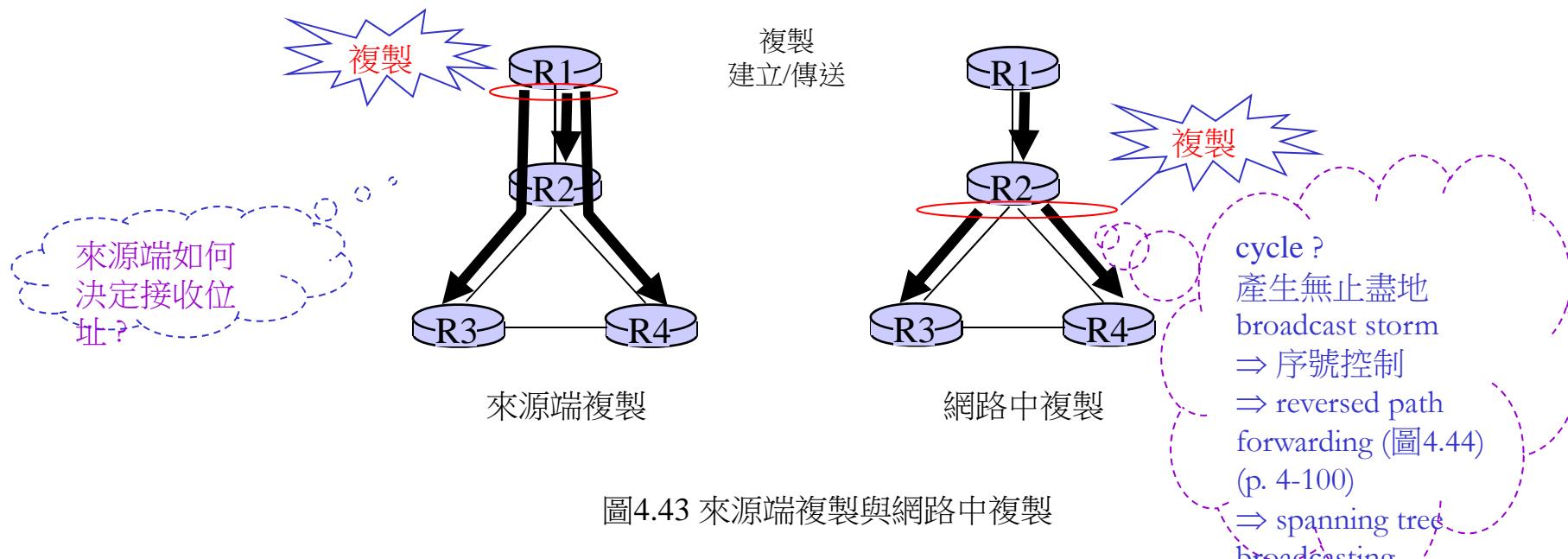


4.7 廣播與群播路由



■ 廣播路由(broadcasting)

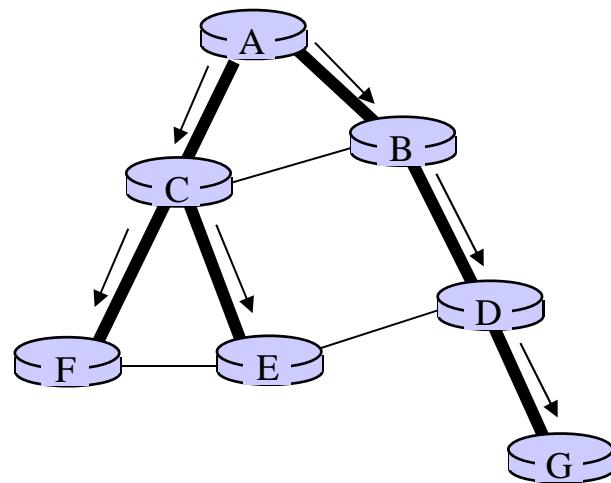
- ▶ 從來源端傳送封包到所有其它的節點
- ▶ 廣播方式
 - 來源端複製
 - 網路中複製 ✓



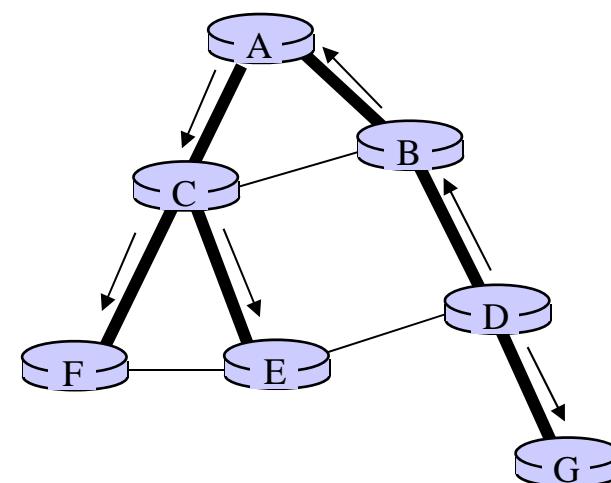


■ 擴張樹廣播

- ▶ 首先，建立一個擴張樹 (spanning tree)
- ▶ 節點沿著擴張樹轉送複本



(a) 從A點開始廣播

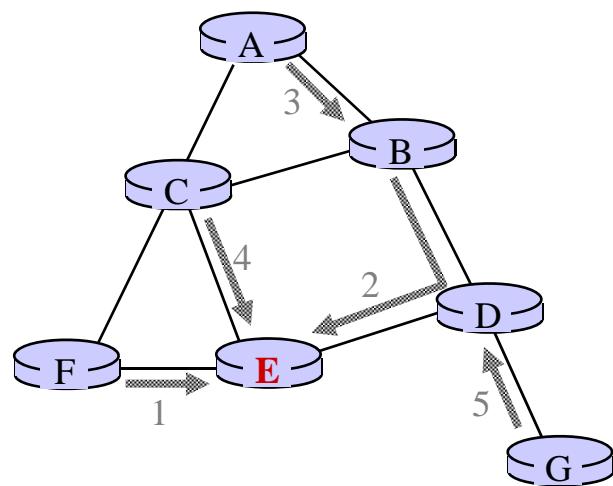


(b) 從D點開始廣播

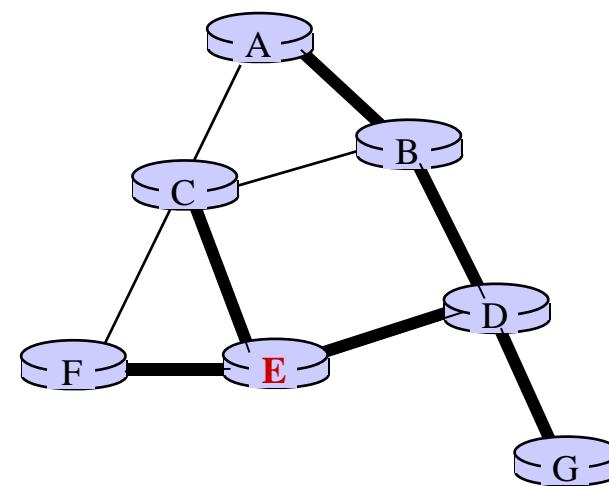
圖4.45 沿著擴張樹進行廣播

■ 中心式擴張樹的建構

- ▶ 選擇中心節點
- ▶ 每個節點會將加入訊息以單點傳播方式傳給中心節點
 - 訊息會被轉送，直到它抵達一個已經屬於擴張樹的節點
 - 節點沿著擴張樹轉送複本



(a) 擴展樹建立步驟



(b) 建立的擴展樹

圖4.46 擴張樹的中心式建構 (E為中心節點)

Ex 4.13



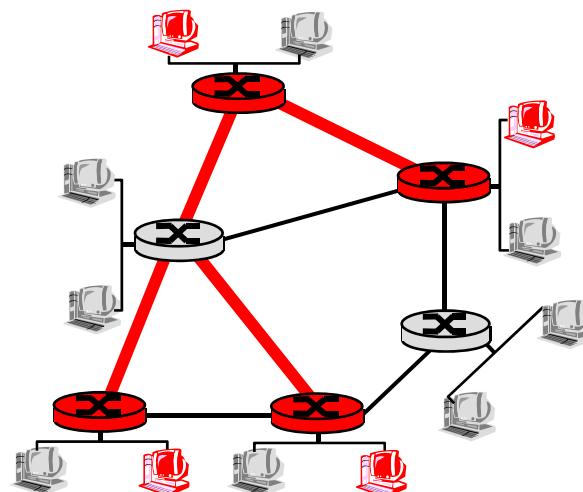
■ 群播

▶ What ?

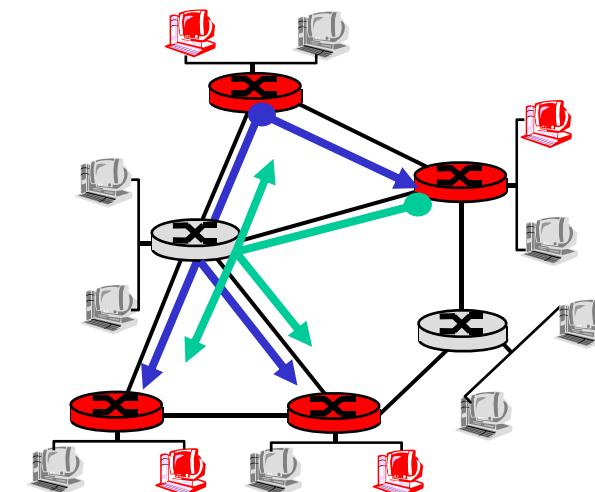
- 將資料封包傳送到 **特定群組** 的節點(路由器)

▶ **群播路由的目標**：找到一棵樹（或是多棵樹）將地區群播群組成員的路由器連結起來

- 樹：不會使用到路由器間的所有路徑
- 以來源端為基礎：從每個來源端到接收端是一個不同的樹
- 分享樹：所有的群組成員都使用相同的樹



分享樹



以來源為基礎的樹



■ 建立群播樹的方法

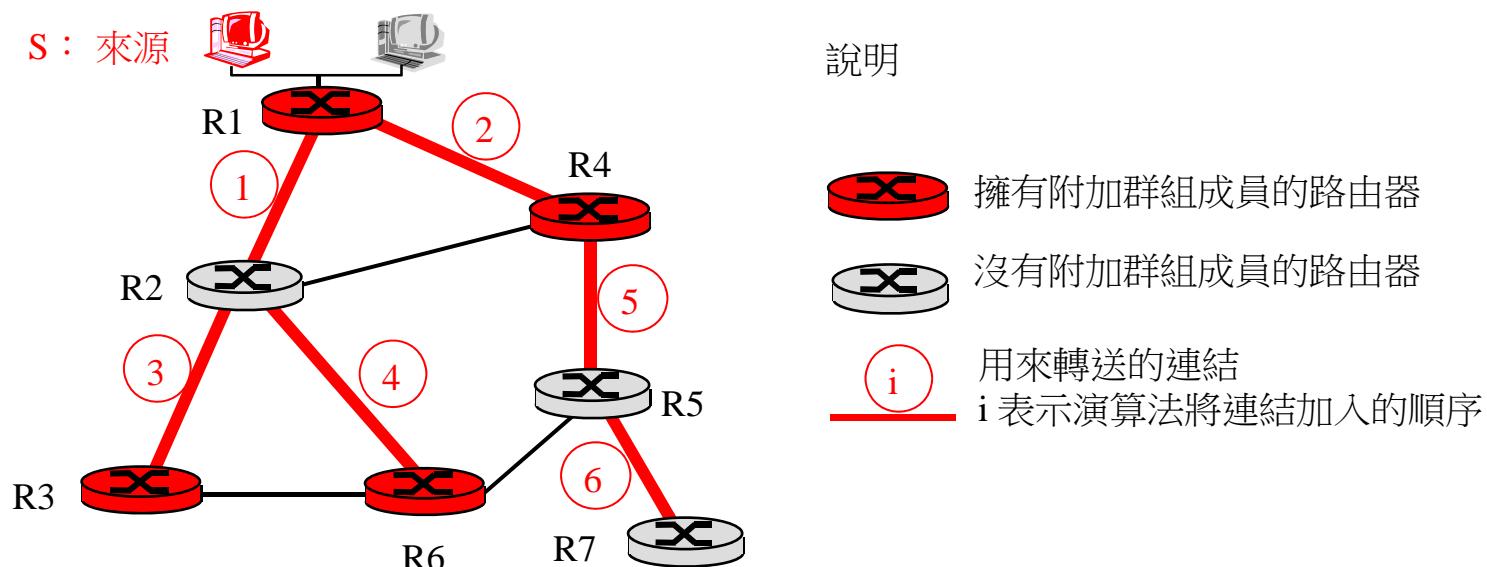
- ▶ 使用來源為基礎的樹：一棵樹一個來源端
 - 最短路徑樹
 - 反向路徑傳送
- ▶ 群組分享樹：群組只使用一棵樹
 - 最小擴張 (Steiner)
 - 以中心為基礎的樹



■ 建立群播樹的方法 (Cont.)

► 最短路徑樹：

- 群播轉送樹：從來源端到所有接收端的最短路徑路由所產生的樹
- Dijkstra 演算法





■ 建立群播樹的方法 (Cont.)

► 分享樹：Steiner 樹

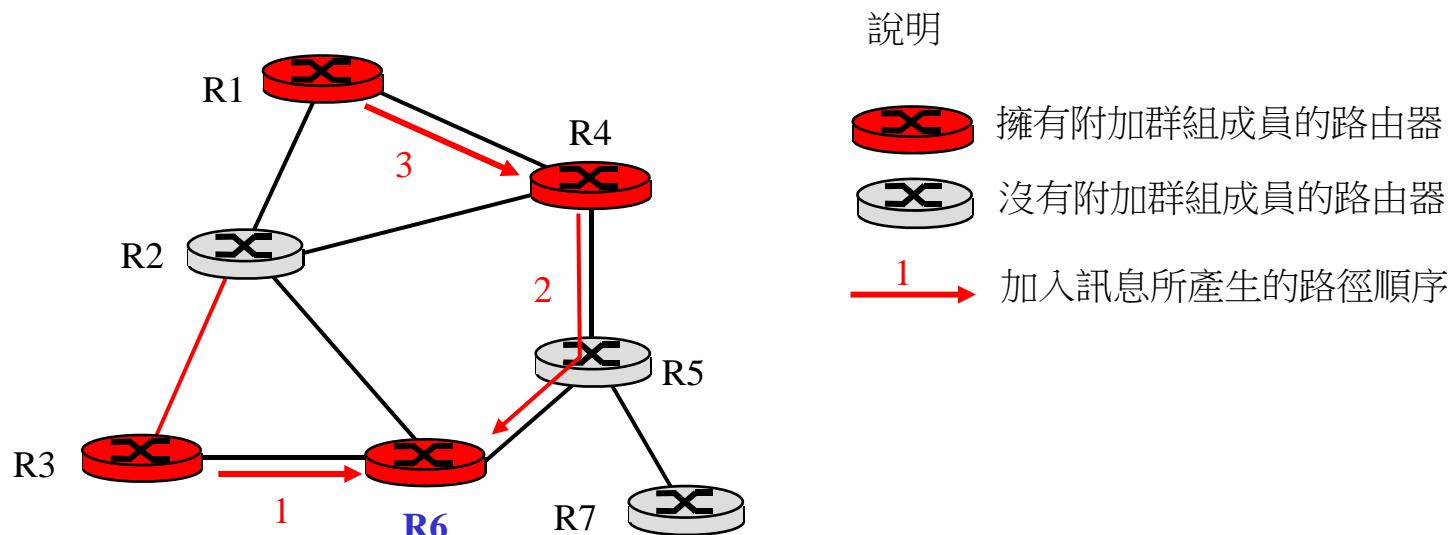
- Steiner 樹：最小成本樹、連接全部擁有群組成員的路由器
- 為NP完全問題
- 極佳的試探存在
- 實際不使用：
 - 計算複雜度
 - 需要整個網路的資訊
 - 龐大的：每當有路由器加入/離開時需要重新執行



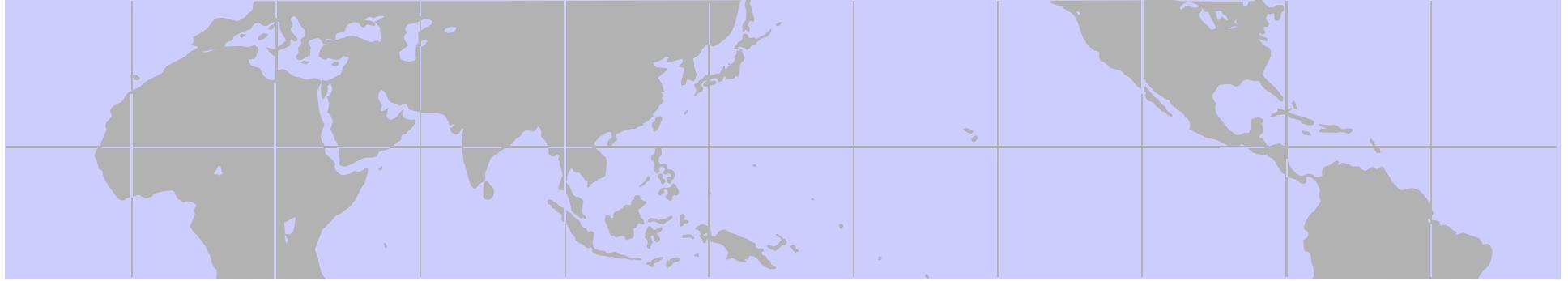
■ 建立群播樹的方法 (Cont.)

▶ 以中心為基礎的樹

- 大家一起分享的單一傳遞樹 (ref. 圖 4.46)
- 在樹中有一個路由器被標記為“中心”
- 加入：
 - 邊緣路由器傳送單點傳播的加入訊息給中心路由器
 - 加入訊息由中間的路由器「處理」並轉送到中心
 - 加入訊息會觸碰到此中心現存樹的分枝、或是抵達中心
 - 加入訊息所使用的路徑會變成這個路由器所使用的樹的分枝
- 假設我們選擇 R6 做為中心：



End of Chapter 4

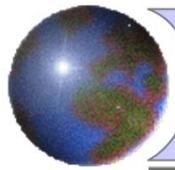


Chapter 5

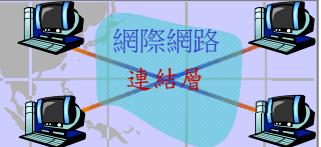
Link Layer

(連結層)





Topic Overview



5.1 連結層簡介與服務

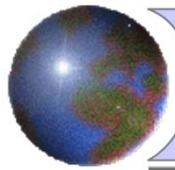
5.2 錯誤偵測和更正技術

5.3 多重存取協定

5.4 連結層定址

5.5 乙太網路

5.6 集線器和交換器



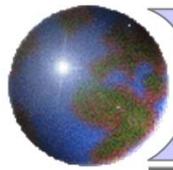
Topic Overview (Cont.)



■ 目標 (Goal)

- ▶ 了解資料連結層背後服務的原理：
 - 發現錯誤並修正
 - 分配一個廣播通道：多重存取
 - 連結層的位址
 - 可靠的資料轉換與流量控制：是！
- ▶ 立即性與實作不同的連結層之技術





第五章 導覽流程



5.1 連結層簡介與服務

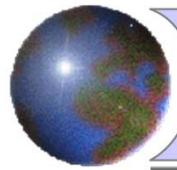
5.2 錯誤偵測和更正技術

5.3 多重存取協定

5.4 連結層定址

5.5 乙太網路

5.6 集線器和交換器

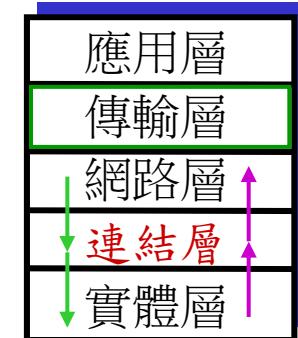


5.1 連結層簡介與服務



■ 基本術語

- ▶ 主機和路由器皆稱為 **節點**(node)
- ▶ 沿著通訊路徑，連結相鄰節點的通訊通道，稱為 **連結**(link)
 - 有線連結 (wired link)
 - 無線連結 (wireless link)
 - 區域網路 (LAN)
- ▶ 連結層的封包稱為 訊框(frame) – 將資料(封)包(datagram)封裝



■ 網路層 vs. 連結層

- ▶ 網路層的責任是規劃來源封包到目的端主機的路徑
- ▶ 連結層的責任為 – 經由連結，將資料封包從一個節點傳輸到相鄰的節點(路徑上，相鄰節點間的可靠/不可靠資料傳輸)



■ 連結層的主要工作

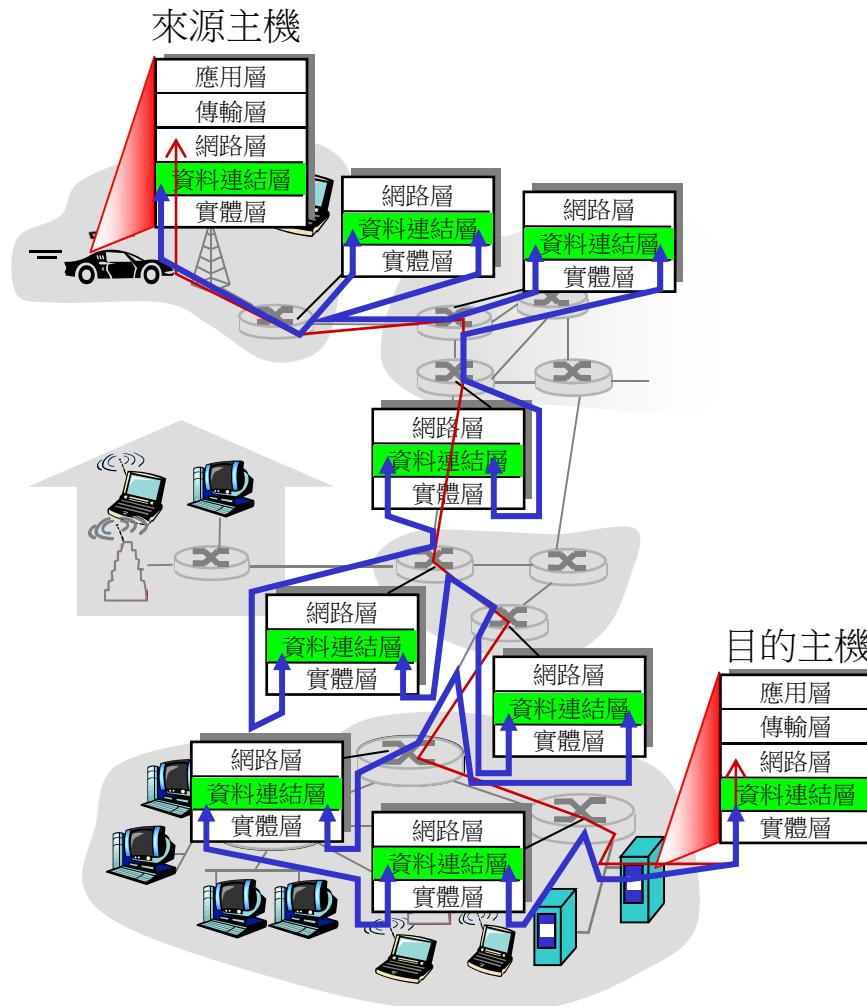


圖5.1 連結層



■ 連結層的主要工作 (Cont.)

- ▶ 資料(封)包在不同的連結上經由不同的連結協定傳輸：
 - 例如，第一個連結為乙太網路、中間連結為訊框傳送、最後一個連結為 802.11
- ▶ 每一種連結協定提供不同的服務
 - 例如，在連結上可能提供或不提供可靠資料傳輸(rdt)
- ▶ 運輸的比喻
 - 從高雄到台北的行程
 - 小型巴士：高雄到 台中
 - 飛機：台中 到 新竹
 - 火車：新竹 到 台北
 - 旅客 = 資料包 (datagram)
 - 每段旅程 = 通訊連結 (communication link)
 - 運輸模式 = 連結層協定 (link layer protocol)
 - 旅行社 = 路由演算法 (routing algorithm, 規劃行程)



■ 連結層服務

► 訊框化、連結存取 : (framing & link access)

- 將資料包(data gram – 網路層封包)封裝成訊框(frame – 連結層封包)
 - 加入標頭及標尾
- 通道存取 (假如媒介共享)
- “MAC” 位址在訊框標頭中，用來識別來源端、目的端
 - MAC (media access control)位址與 IP 位址不同

► 相鄰節點間的可靠傳輸 (reliable delivery)

- 我們已經學習過了 (第三章)
 - 在位元錯誤率低的連結上很少使用 (光纖、某些雙絞線)
 - 無線連結：錯誤率高
- ⇒ 有線連結一般採用不可靠傳輸



■ 連結層服務 (Cont.)

▶ 流量控制 : (flow control)

- 調整相鄰傳送端和接收端節點之間的步調

▶ 錯誤偵測 : (error detection)

- 因訊號衰減或雜訊所產生的錯誤

- 接收端偵測到錯誤的存在：

- 通知傳送端重新傳送訊息或是將訊框丟掉

▶ 錯誤更正 : (error correction)

- 不憑藉重新傳送、接收端辨識並更正位元錯誤

▶ 半雙工和全雙工 (half-duplex & full-duplex)

- 在半雙工中，連結兩端的節點可以傳送、但無法同時

-
- 與傳輸層的服務類似
 - 傳輸層提供端點間兩行程的通訊區段的服務
 - 連結層提供節點間訊框的服務



■ 連結層的實作在那裡？

- ▶ 在任意一個與每個主機中 – 每台網路上的設備
- ▶ 連結層的實作在 “網路卡” 中 (又稱網路介面卡, NIC)
 - 乙太網路卡、PCMCI 卡、802.11 卡
 - 執行連結實體層
- ▶ 隨著進入主機系統匯流排
- ▶ 結合硬體、軟體與介面卡
- ▶ 連結層部份功能實作在 S/W 、
部份功能實作在 H/W

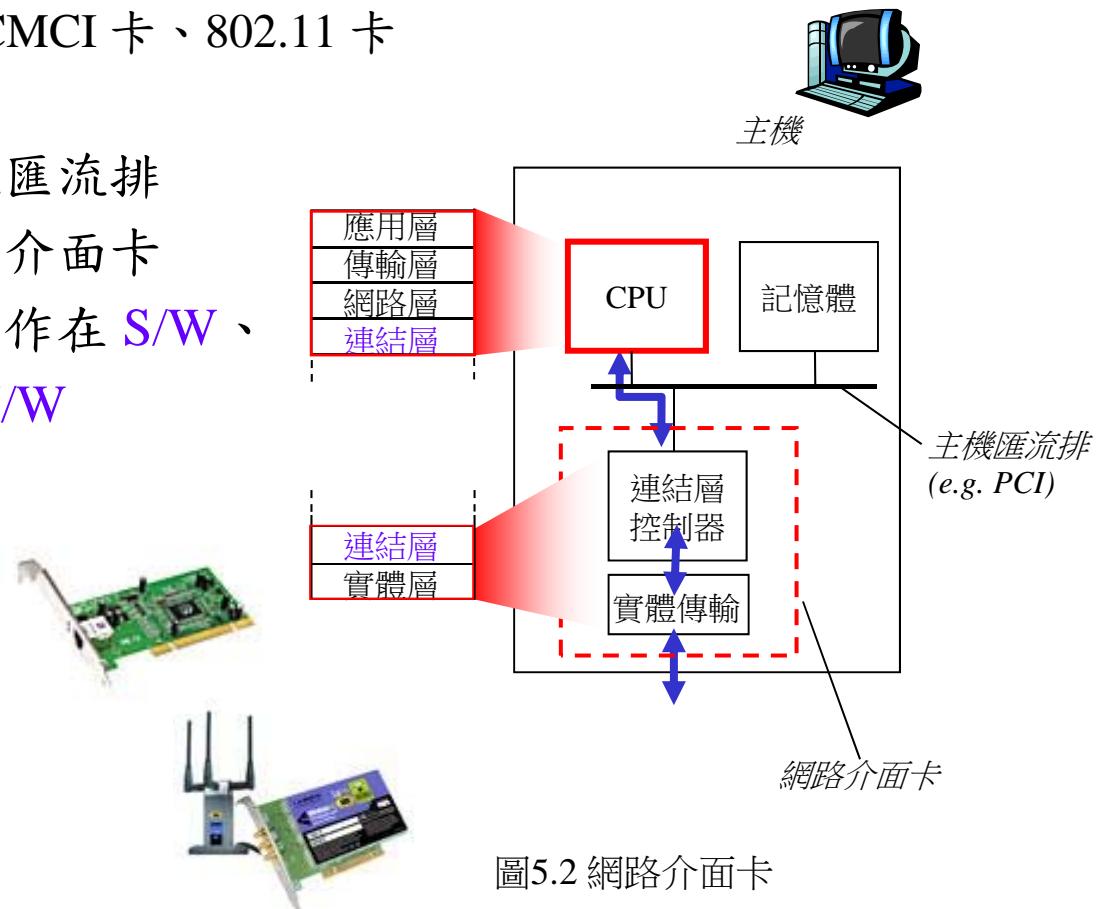


圖5.2 網路介面卡



■ 網路卡通訊

▶ 傳送端：

- 將資料包(datagram)封裝在訊框(frame)中
- 加入錯誤確認位元、rdt、流量控制等等

▶ 接收端

- 尋找錯誤、rdt、流量控制等等
- 取出資料包，將它傳送到接收端節點

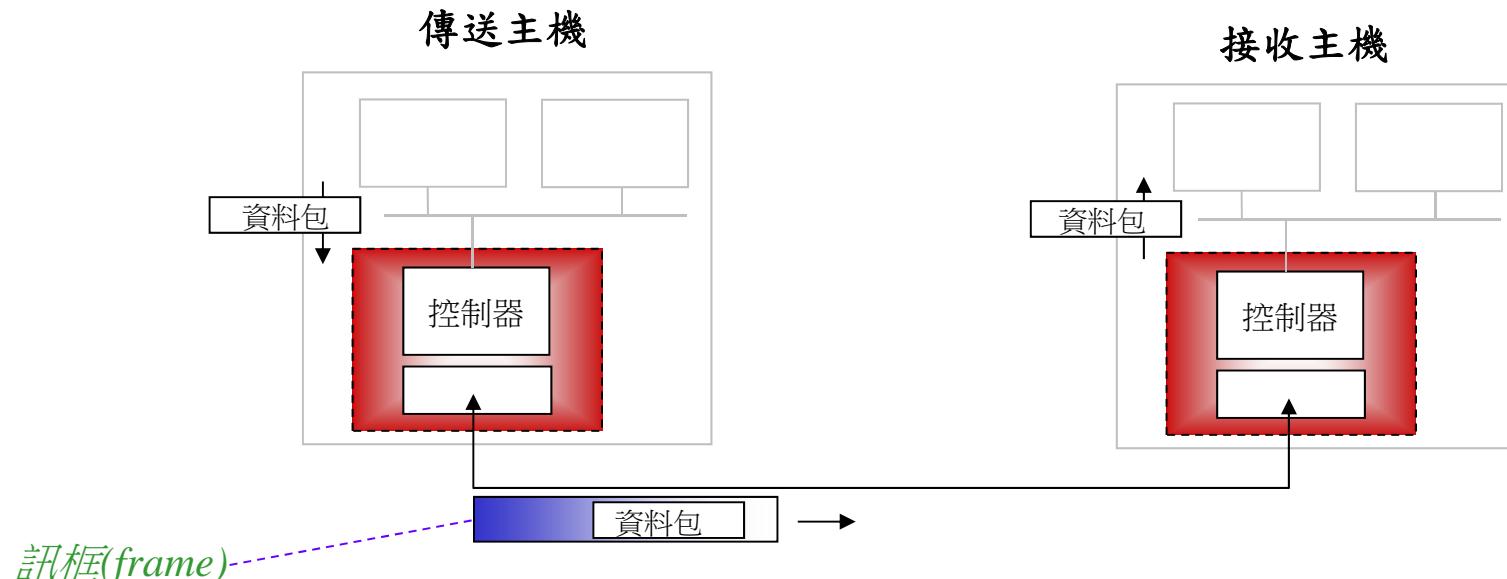
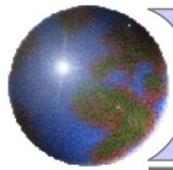
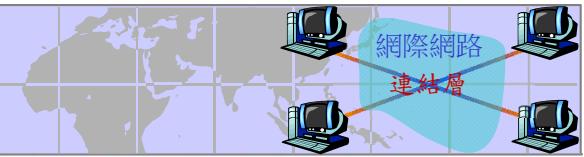


圖5.3 網路卡通訊





第五章 導覽流程



5.1 連結層簡介與服務

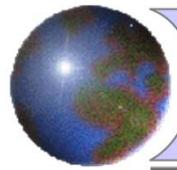
5.2 錯誤偵測和更正技術

5.3 多重存取協定

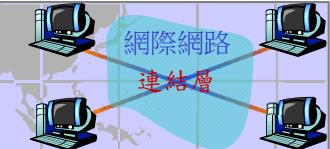
5.4 連結層定址

5.5 乙太網路

5.6 集線器和交換器



5.2 錯誤偵測和更正技術



■ 錯誤偵測

- ▶ 訊框封包 (frame) 內容
 - EDC = 錯誤偵測及更正位元 (冗餘)
 - D = 被錯誤檢查所保護的資料、可能包含標頭欄位
- ▶ 錯誤偵測 並不是 100% 可靠的
 - 協定可能會遺漏某些錯誤 (不過這很少發生)
 - 更大的 EDC 欄位能夠得到更佳的偵測與更正

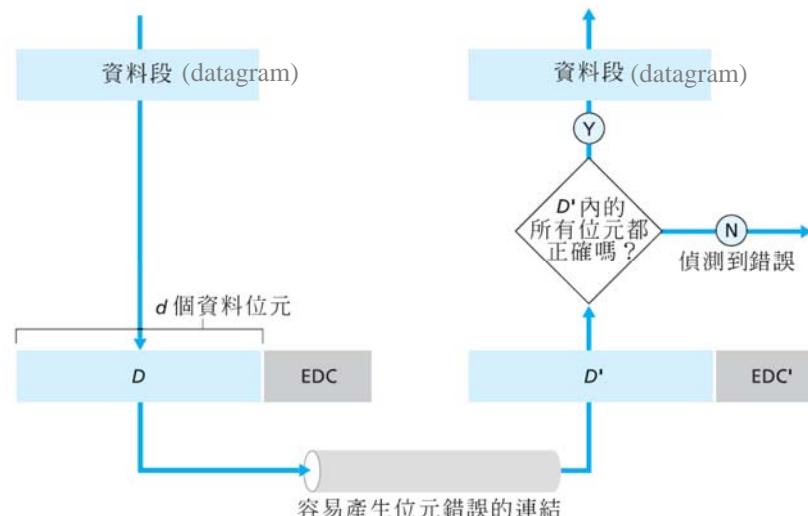


圖5.4 錯誤偵測與更正的情景



Hung3

■ 同位檢查

▶ 單一的同位元 (偶同位/奇同位) - 以偶同位為例

- EDC 為 1 bit
- D 內的 1 的個數 + EDC = 偶數
- 偵測單一位元錯誤
- 無法更正錯誤位元

▶ 二維同位元

- 偵測並更正單一位元錯誤

將 D 切成
 i 列 j 行

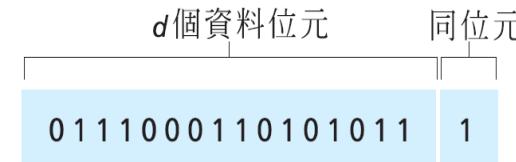
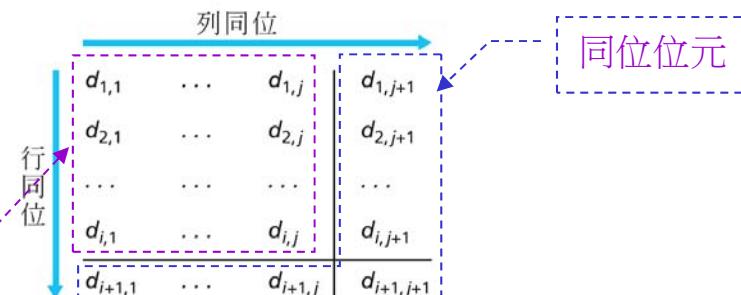


圖 5.5 單一位元的偶同位



沒有錯誤 可更正的單一位元錯誤

1	0	1	0	1	1
1	1	1	1	0	0
0	1	1	1	0	1
0	0	1	0	1	0

1	0	1	0	1	1
1	0	1	1	0	0
0	1	1	1	0	1
0	0	1	0	1	0

同位錯誤

同位錯誤

同位錯誤

Exercise
5.1

Ex 5.2

圖 5.6 二維的偶同位



投影片 14

Hung3

最簡單的錯誤偵測方法

Ruo-Wei Hung, 2011/8/8



■ 網際網路檢查和

▶ 目標

- 偵測在傳送的資料區段中的“錯誤”(例如、翻轉的位元)(注意：
大都用在傳輸層)
- See 第3.3.2節

▶ 傳送端

- 將資料區段的內容視為一連串16位元的整數
- 檢查和：資料區段內容相加(1的補數和)
- 傳送端會將檢查和的值放入UDP檢查和欄位中

▶ 接收端

- 計算收到的資料區段的檢查和
- 確認計算出來的檢查和是否和檢查和欄位中的相等：
 - NO - 偵測到錯誤
 - YES - 沒有偵測到錯誤。但是仍然可能有錯誤?

投影片 15

Hung1

CheckSum較常用於傳輸層
Ruo-Wei Hung, 2011/8/8



■ 網際網路檢查和 (Cont.)

► 作法：

傳送端：

- 將資料分段的內容視為一列16位元的整數
- 檢查和：資料分段內容的加法 (1的補數和)
- 傳送端將檢查和的值放入 UDP的檢查和欄位

Ex:

0110011001100000
0101010101010101
1000111100001100

$$\begin{array}{r} 0110011001100000 \\ 0101010101010101 \\ +) \underline{1000111100001100} \\ \hline 10100101011000001 \\ \hline \end{array}$$

↑
0100101011000010

↓
1's補數: 0→1&1→0

check sum = 1011010100111101

接收端：

- 計算收到的資料分段的檢查和
- 確認計算出來的檢查和是否和檢查和欄位中的相等：
 - ▶ NO – 偵測到錯誤
 - ▶ YES – 沒有偵測到錯誤。但是仍然可能有錯誤？後面有更多介紹

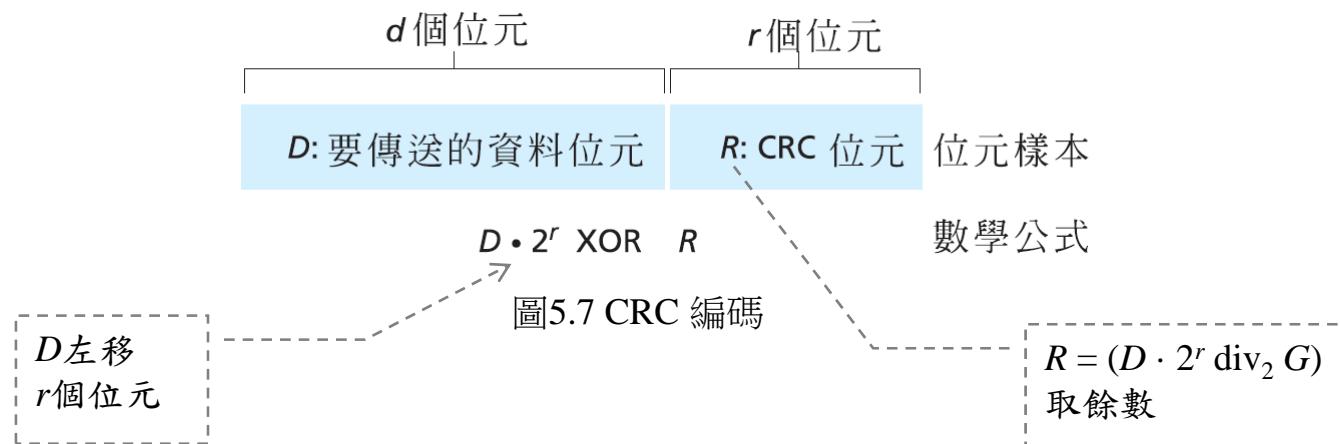
⇒ 資料+檢查和 = 111111...11



Hung2

■ 循環冗餘檢查 (CRC, cyclic redundancy check)

- ▶ 將資料位元 D 視為二進位數字
- ▶ 選擇 $r+1$ 個位元為樣本 (產生器) G
 - 傳送端/接收端協調出來的資訊，第一個bit為1
- ▶ 運作：傳送端計算 r 個 CRC 位元， R ，使得
 - $\langle D, R \rangle$ 正好能被 G 整除 (2模數除法，其內+和-法均為XOR)
 - 接收端知道 G ：將 $\langle D, R \rangle$ (2模數除法, div_2)除以 $G \Rightarrow$ 假如餘數不為0：
偵測到錯誤
 - 可以偵測出任何少於 $r+1$ 個位元的叢發錯誤
- ▶ 實務上經常使用在 ATM、HDCL



投影片 17

Hung2

CRC較常用於網路卡的連結層

Ruo-Wei Hung, 2011/8/8



■ 循環冗餘檢查 (Cont.)

▶ 作法：

傳送端：

- 將資料位元 D 視為二進位數字
- 選擇 $r+1$ 個位元為樣本 (產生器) G
- 計算 r 個 CRC 位元， R ，使得
$$R = (D \cdot 2^r \text{ div}_2 G) \text{ 取餘數}$$
- 將 $\langle D, R \rangle$ 傳送到接收端

接收端：

- 擁有 $r+1$ 個位元為樣本 (產生器) G
- 計算收到的資料封包 $\langle D', R' \rangle$
 - ▶ 將 $\langle D', R' \rangle$ (2模數除法) 除以 G
 - ▶ 若餘數不為0：偵測到錯誤
 - ▶ 否則， $D = D'$





■ 循環冗餘檢查 (Cont.)

► CRC 範例

- $D = 101110$
- $G = 1001, r = 3$
- 傳送端計算 $R = (D \cdot 2^r \text{ div}_2 G)$ 取餘數
 - $D \cdot 2^r = 101110\ 000$
 - $\text{rem}(D \cdot 2^r \text{ div}_2 G) = R = 011$
- $\langle D, R \rangle = 101110\ 011$
- 若接收端接收 $101111\ 011 = \langle D', R' \rangle$
 - $\text{rem}(101111011 \text{ div}_2 1001) = 1 \neq 0$
 - ⇒ 錯誤發生

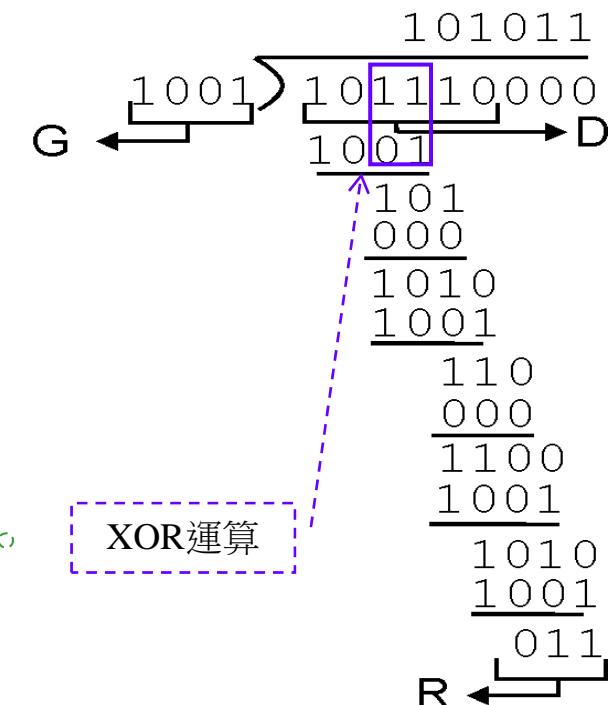
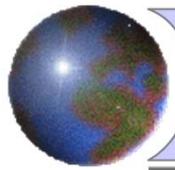


圖5.8 簡單的CRC運算

Exercise
5.3

Ex 5.3





第五章 導覽流程



5.1 連結層簡介與服務

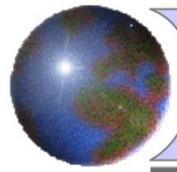
5.2 錯誤偵測和更正技術

5.3 多重存取協定

5.4 連結層定址

5.5 乙太網路

5.6 集線器和交換器



5.3 多重存取協定



■ 多重存取連結與協定

► 兩種型態的“連結”：

- 點對點

- 撥號存取的PPP
- 乙太網路交換器和集線器之間的點對點連結



- 廣播 (分享的線路或媒介)

- 傳統的乙太網路
- 上傳 HFC
- 802.11 無線區網





■ 多重存取連結與協定 (Cont.)

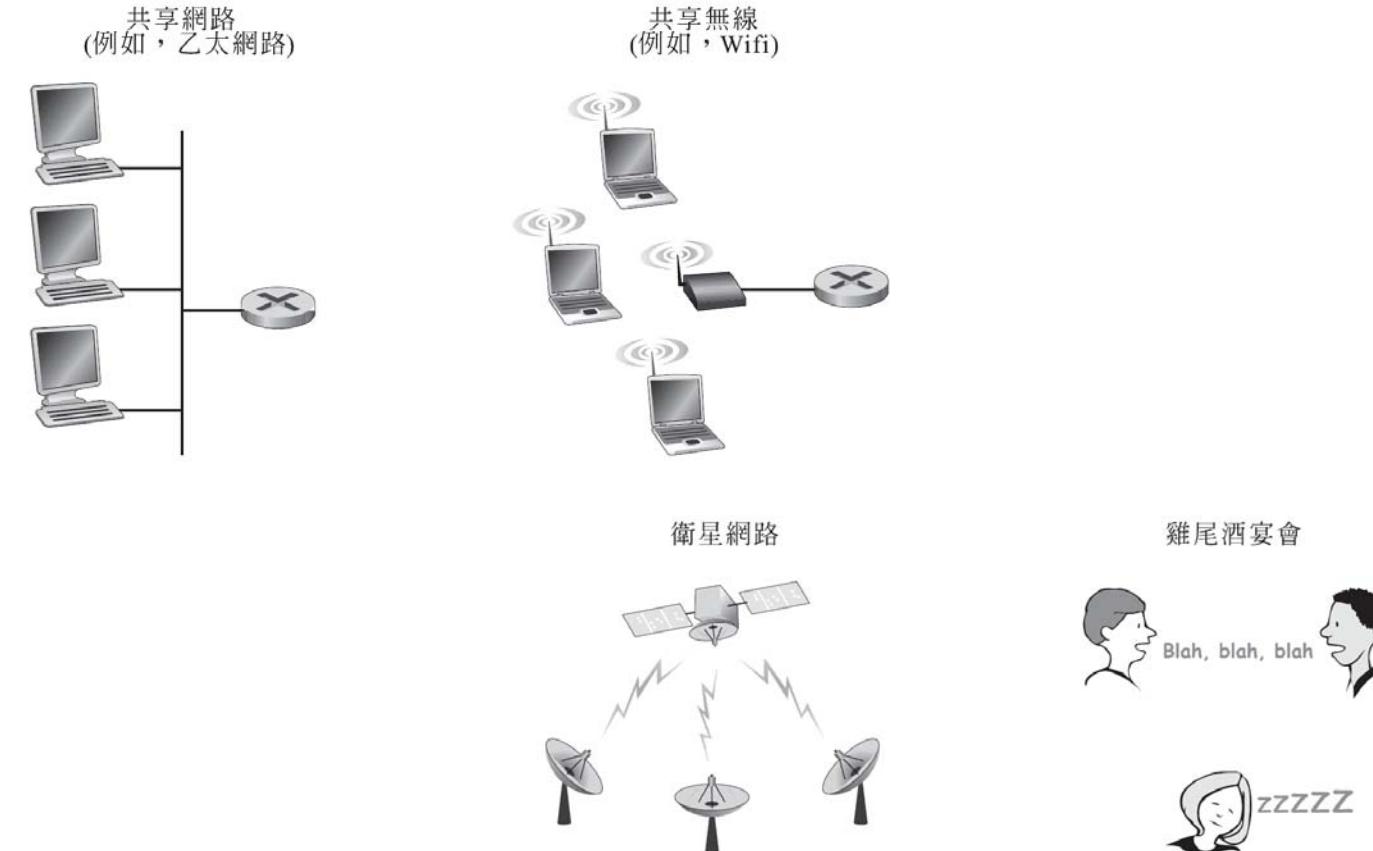


圖5.9 各種多重存取通道



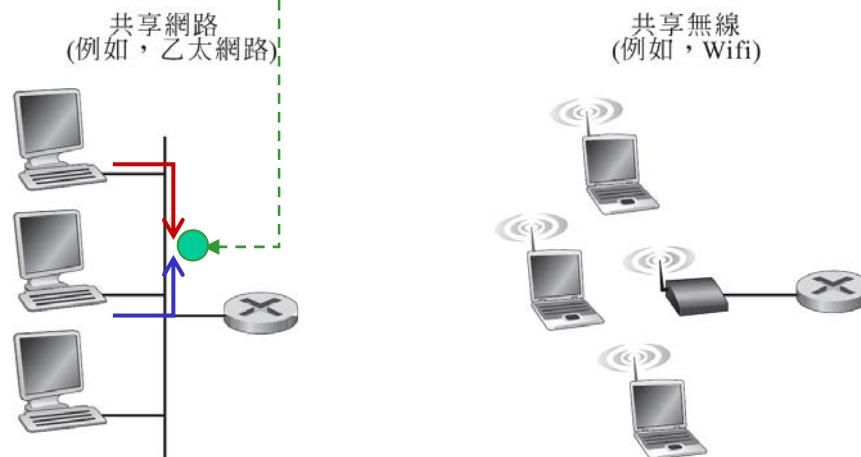
■ 多重存取協定 (multiple access protocol)

▶ 多重存取

- 單一的分享廣播通道
- 同時兩個或更多節點的傳送 – 干擾
 - 碰撞 (collision) – 假如節點在同一個時間收到兩個或更多個信號

▶ 多重存取協定

- 決定節點如何分享通道的分散式演算法。例如，決定節點何時能夠傳送
- 必須使用通道本身來傳送有關通道分享的通訊!
 - 沒有不同頻帶的通道做為協調的功能





■ 理想的多重存取協定

► 速率為 $R \text{ bps}$ 的廣播通道

- 當只一個節點要傳送資料時，它可以擁有 $R \text{ bps}$ 傳輸率
- 當 n 個節點要傳送資料時，每一個節點以平均速率 R/n 來傳送資料
- 完全的非集中式：(協定是分散式的)
 - 沒有用來協調傳輸的特殊節點
 - 沒有時脈和時槽的同步
- 協定要簡單



■ 多重存取協定的分類

▶ 通道分割 (channel partitioning protocol)

- 將共享的通道分割成「小塊」(時槽、頻率、碼)
- 將這些「小塊」分配為每個節點專用的
- ref. 1.3 節

▶ 隨機存取 (random access protocol)

- 不分割通道，允許碰撞
- 從碰撞「復原」

▶ 輪流存取 (polling protocol)

- 節點可以輪流，但是要傳送較多資料的節點可以傳送較長的時間
- 節點輪流使用完整的通道



■ 通道分割協定

► TDM：時間分割多重存取 (time-division)

- 每一個「回合」中，存取通道
- 每一節點在每一回合中，得到固定長度的時槽（長度 \geq 封包傳送時間）
- 沒有使用的時槽(slot)會變成閒置的
- 範例：6個節點站的 LAN，1、3、4 有封包；時槽 2、5、6 為閒置的

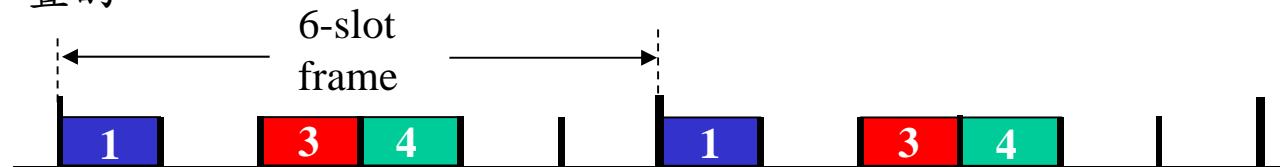


圖5.10(a) 一個6個節點的TDM例子



■ 通道分割協定 (Cont.)

► FDM：頻率分割多重存取 (frequency-division)

- 通道頻譜被分割成頻帶
- 每一節點站分配到固定的頻帶
- 沒有用到的傳輸頻帶會閒置
- 範例：6個節點站的 LAN，1、3、4 有封包，頻帶 2、5、6 為閒置的

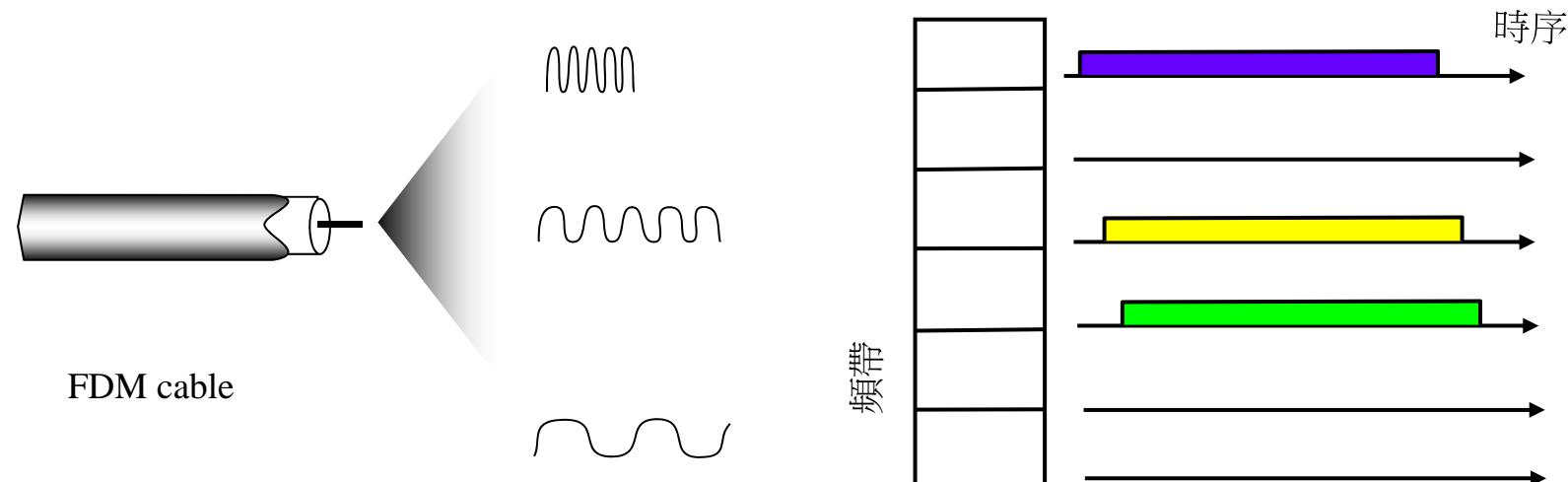
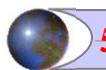


圖5.10(b) 一個6個節點的FDM例子



■ 隨機存取協定

- ▶ 當節點有封包要傳送時 –
 - 以全部的通道資料速率 R 傳送
 - 節點間沒有做前置的協調
- ▶ 兩個以上的節點同時傳送資料時 –
 - “碰撞” (collision)
- ▶ 隨機存取協定說明：
 - 要如何偵測碰撞
 - 要如何從碰撞中復原 (例如，經由等待一段隨機延遲時間來重傳)
- ▶ 隨機存取協定的範例：
 - 分槽式 ALOHA
 - ALOHA
 - CSMA、CSMA/CD、CSMA/CA



■ 隨機存取協定 (Cont.)

► 分槽式(slotted) ALOHA

○ 假設

- 所有的訊框(frame)都是同樣大小
- 時間被切分為等大小的時槽(time-slot)，長度可傳送1個訊框
- 節點只有在時槽開始時，才會傳送訊框
- 節點為同步的
- 如果兩個以上的節點在時槽內傳送，所有的節點都會偵測到碰撞

○ 運作

- 當節點要傳送新的訊框時，它會在下一個時槽傳送
- 如果沒有碰撞，節點會在下一個時槽傳送新的訊框
- 假如產生碰撞，節點會依照機率 p ，在接下來的每個時槽中重新傳送訊框，直到成功為止



■ 隨機存取協定 (Cont.)

► 分槽式 ALOHA (Cont.)

○ 優點

- 單一的活動節點，能以通道的全部速率連續傳送
- 高度的非集中式：只有節點中的時槽需要同步
- 簡單

○ 缺點

- 碰撞，浪費時槽
- 閒置的時槽
- 節點需要在比傳輸封包少的時間內偵測碰撞
- 時脈同步

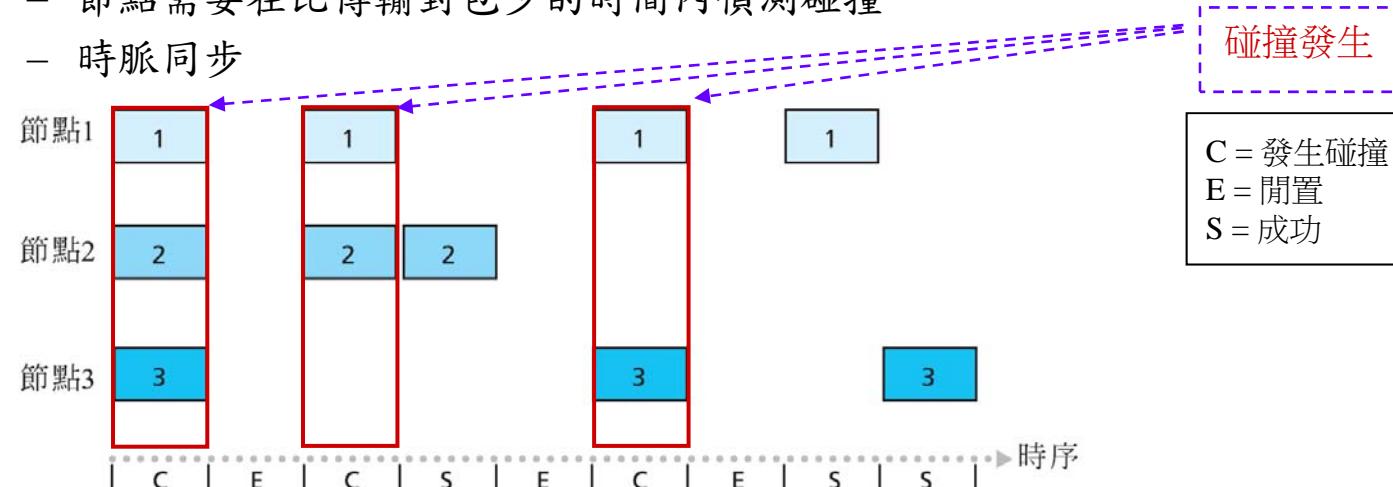


圖5.11 分槽式 ALOHA的例子



■ 隨機存取協定 (Cont.)

► 分槽式 ALOHA (Cont.)

○ 效率

- 當有大量節點，每個節點有大量的訊框要傳送時，長時間下成功時槽所佔的比例
- 假設有 n 個節點有很多訊框要傳送，每個時槽傳送的機率為 p
- 節點 1 在一個時槽中成功的機率為 $= p(1-p)^{n-1}$
(節點 1 進行傳送的機率 = p ；其它節點不進行傳送的機率 = $(1-p)^{n-1}$)
- 任意節點傳送成功的機率為 $= n \cdot p(1-p)^{n-1}$
- 當 n 趨近於無線大，我們得到 $n \cdot p(1-p)^{n-1}$ 的極限值為 $1/e = 0.37$
⇒ 在最佳狀況下，分槽式 ALOHA 通道成功用來傳輸的時間只有 37%



■ 隨機存取協定 (Cont.)

Hung4 ► CSMA (*carrier sense multiple access*)

- 原理

- 傳送之前先聆聽 (carrier sensing)
- 假如通道感測到閒置 → 傳送整個訊框
- 假如通道感測到忙碌 → 延後傳送

- 人類的比方

- 不要打斷別人的談話

投影片 32

Hung4

載波感測多重存取
Ruo-Wei Hung, 2011/8/8

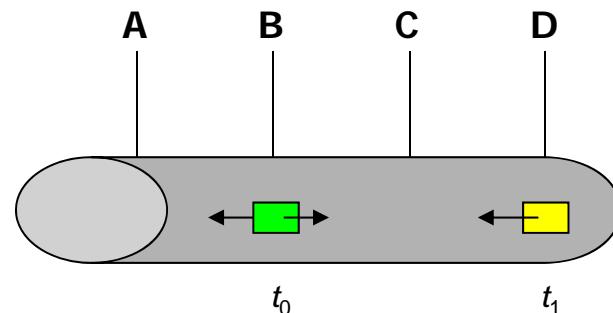


■ 隨機存取協定 (Cont.)

► CSMA (Cont.)

○ CSMA 碰撞 (collision)

- 碰撞還是會發生
 - » 傳遞延遲使得兩個節點可能無法聽到其他人的傳送
- 碰撞時，傳送整個封包時間的浪費
- 距離和傳遞延遲決定了碰撞的機率



*t₁*時, B的封包還未抵達D, 因此, D會感測到"通道閒置"而送出封包 \Rightarrow 碰撞發生

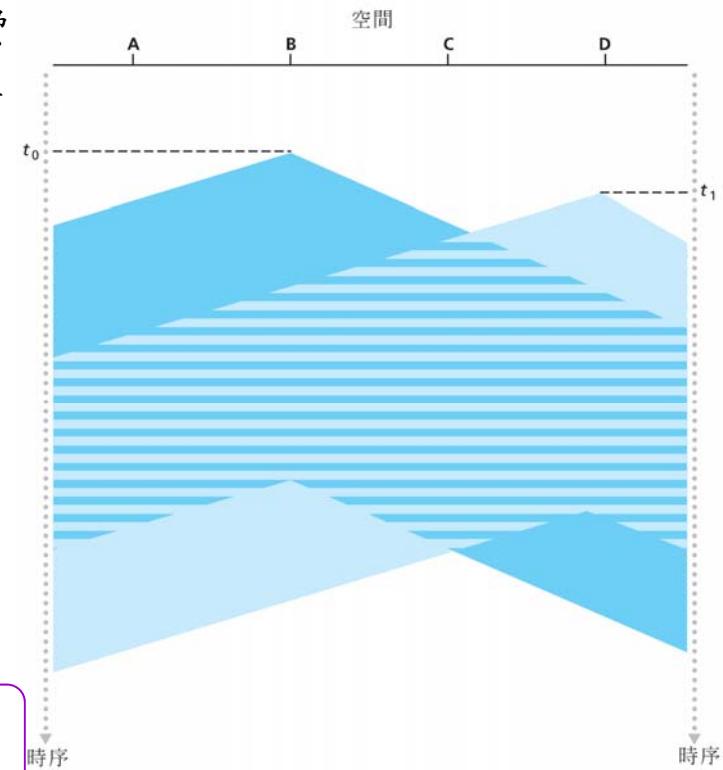


圖5.13 兩個傳輸時，發生碰撞的CSMA節點的時空示意圖



■ 隨機存取協定 (Cont.)

► CSMA (Cont.)

◦ CSMA/CD (CSMA with collision detection)

- 碰撞偵測：(collision detection)
 - » 在區域網路中是簡單的：測量訊號強度，比較傳送和接收訊號
 - » 在無線區域網路中是困難的：當傳送時，接收端關閉
- EtherNet(乙太網路) 採用CSMA/CD

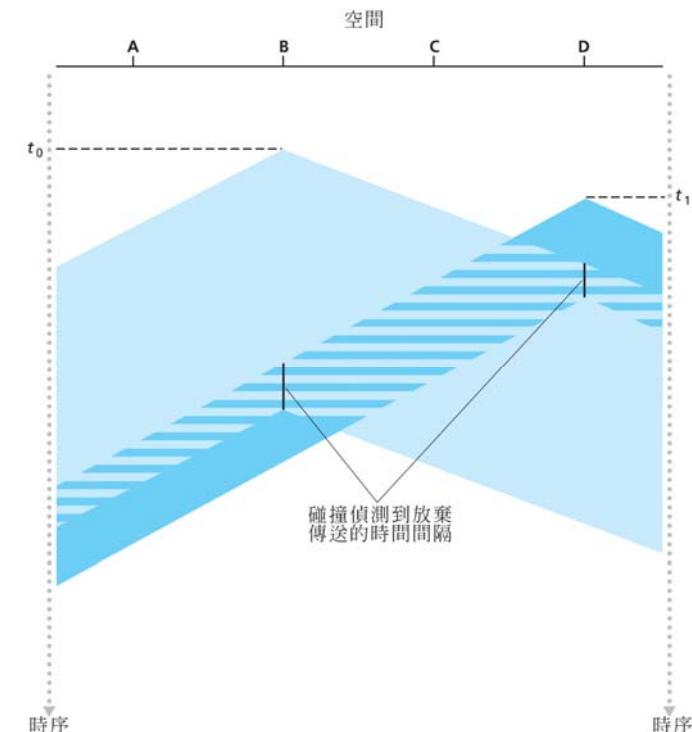
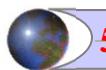


圖5.14 使用碰撞偵測的CSMA

R.-W. Hung ©2013



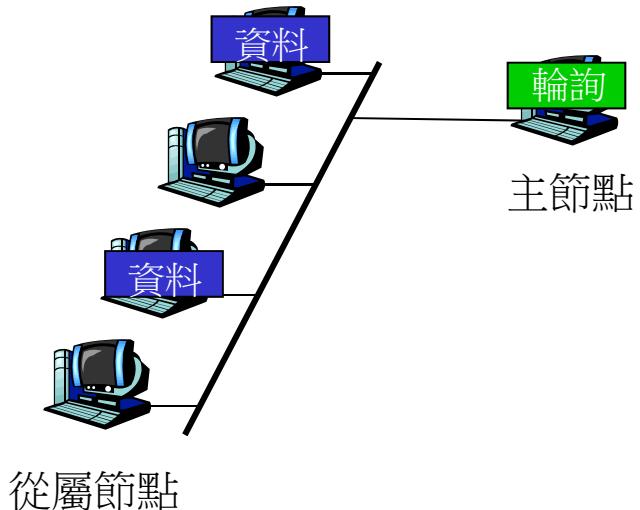
■ 輪流存取協定

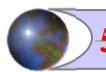
▶ 輪詢 (Polling)

- 主節點輪流邀請從屬節點傳送資料

- 問題：

- 輪詢的額外負擔
- 延遲
- 單點故障 (主節點)

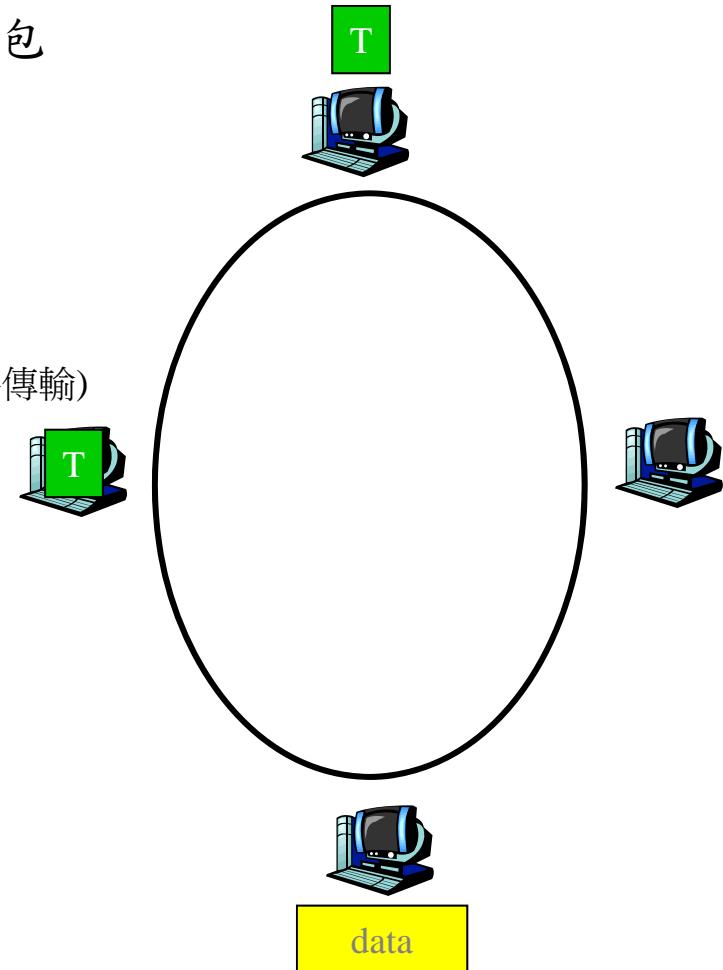
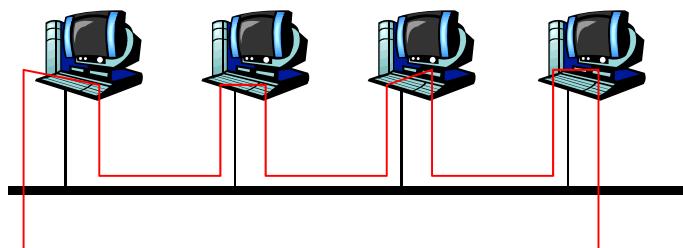




■ 輪流存取協定 (Cont.)

▶ 記號(令牌/權杖)傳遞 (*Token passing*)

- 控制記號(token)依序從一個節點傳到下一個節點
- 當節點擁有token時，才可傳送封包
- 問題：
 - 記號的額外負擔
 - 延遲
 - 單點故障 (token 遺失)
- 邏輯連接成環狀 (ring) ?
 - (無資料傳輸)
 - OK





■ 多重存取協定總結

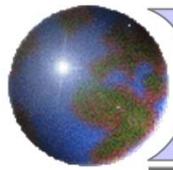
- ▶ 通道分割 – 以時間、頻率或碼
 - 時間分割、頻率分割
- ▶ 存機存取 (動態的)
 - ALOHA、S-ALOHA、CSMA、CSMA/CD
 - 載波感測 – 在某些技術中是簡單的 (有線)、其他的有困難 (無線)
 - CSMA/CD 使用在乙太網路
 - CSMA/CA 使用在 802.11
- ▶ 輪流
 - 從主節點輪詢、記號傳遞
 - 藍芽 h、FDDI、IBM Token Ring



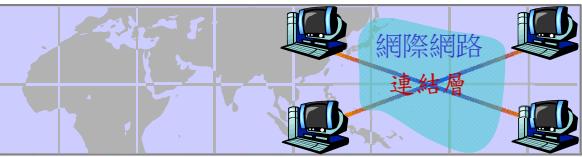


■ 區域網路的技術

- ▶ 到目前介紹的連結層：
 - 服務、錯誤偵測/更正、多重存取
- ▶ 接下來：區域網路的技術
 - 定址 (addressing)
 - 乙太網路 (Ethernet)
 - 集線器、交換器 (hub / switch)
 - PPP (point-to-point protocol)



第五章 導覽流程



5.1 連結層簡介與服務

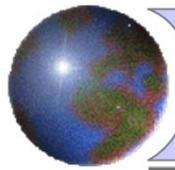
5.2 錯誤偵測和更正技術

5.3 多重存取協定

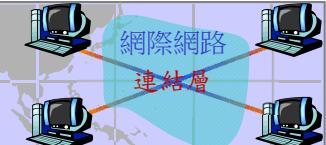
5.4 連結層定址

5.5 乙太網路

5.6 集線器和交換器



5.4 連結層定址



■ MAC 位址 (media access control)

- ▶ 32 位元的 IP 位址 (IPv4) \Rightarrow IPv6 ?
 - 網路層位址
 - 用來接收資料包(datagram)，到目的IP子網路
- ▶ MAC (或是區域網路、實體、乙太網路) 位址 (LAN 位址)
 - 連結層位址
 - 用來接收訊框(frame) – 從一個介面到另一個實體連結介面 (同一個網路)
 - 48 位元 MAC 位址 (大多數的 LAN)
 - 燒錄在網路卡(NIC)的ROM中，有時候是在軟體裡 (virtual machine)



■ MAC 位址 (LAN 位址)

- ▶ 每一張網路卡都具有 唯一的MAC位址 (48 bits)

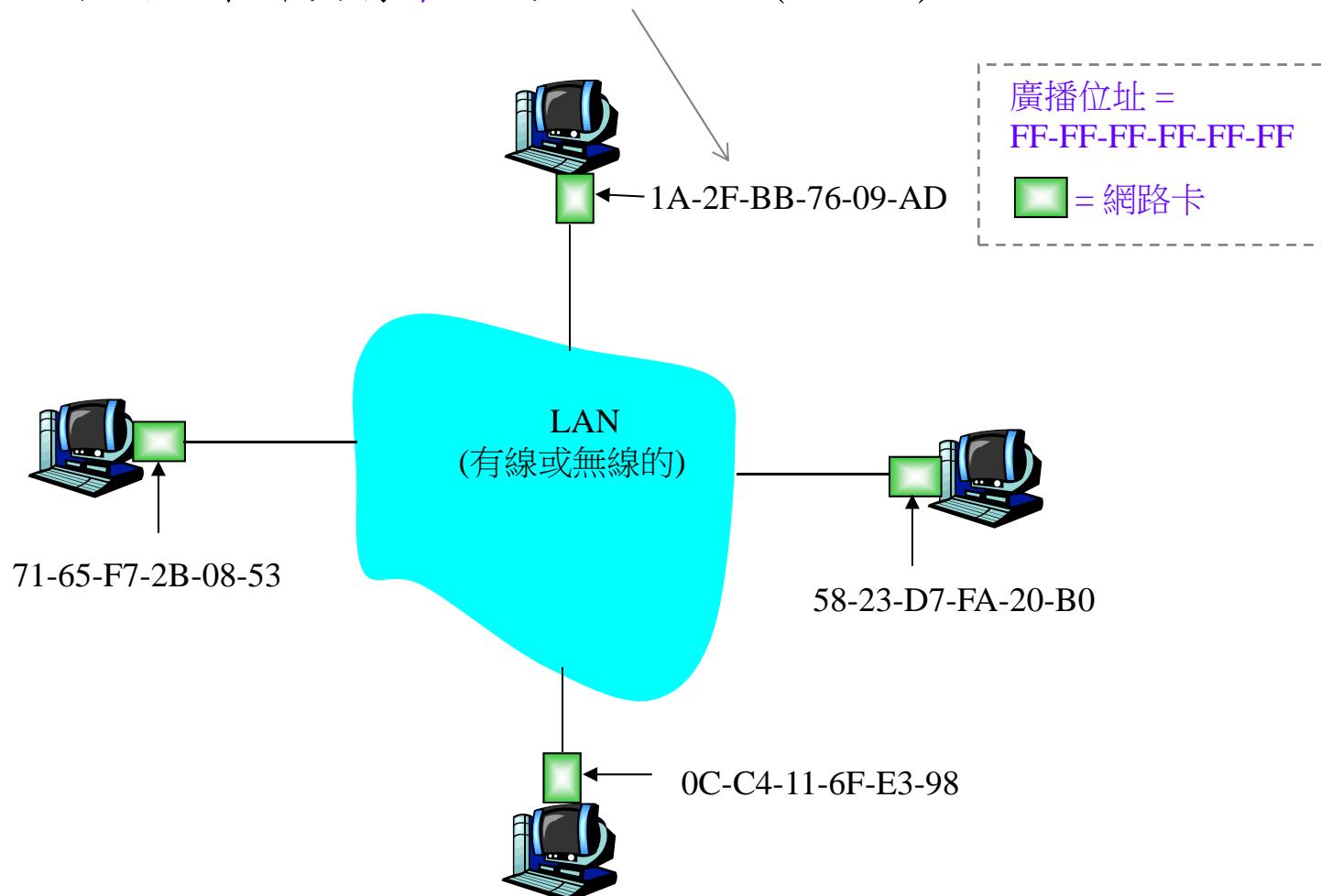


圖5.16 每個連接到LAN的網路卡都有獨一無二的MAC位址



■ MAC 位址 (Cont.)

▶ MAC 位址的分配

- 由 IEEE (Institute of Electrical and Electronics Engineers, www.ieee.org) 機構管理
- 網路卡製造商向 IEEE 購買部分的MAC位址 (保證唯一性)

▶ MAC 位址 vs. IP 位址

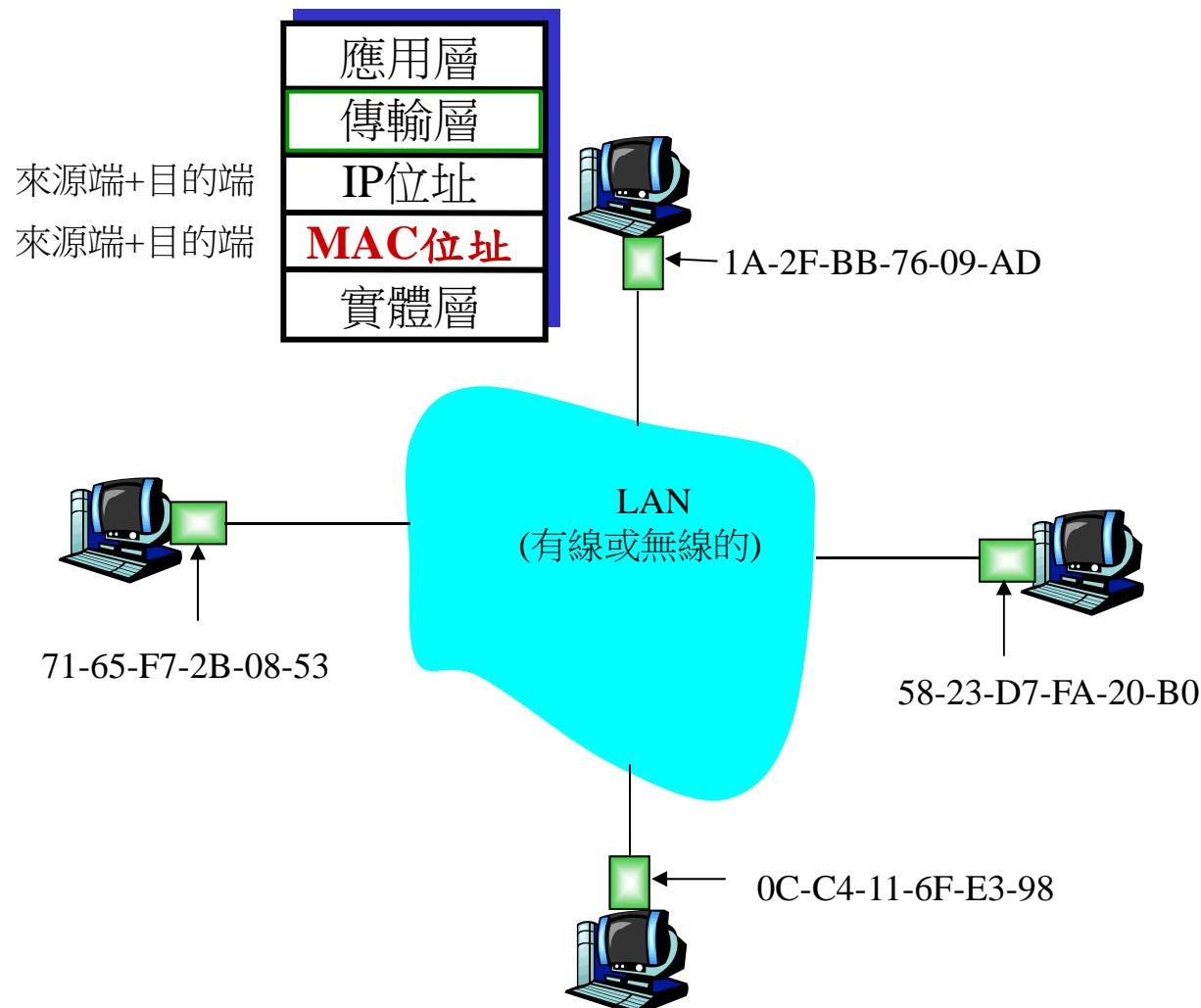
- MAC 位址：如同身份證號碼
- IP 位址：如同郵寄地址

▶ MAC 位址是可攜性

- 可以將網路卡從一個LAN帶到另一個LAN
- IP 位址 不具可攜性 – 與節點連接的IP子網路相關



■ IP與MAC位址



Ex 5.5



■ ARP

Hung5

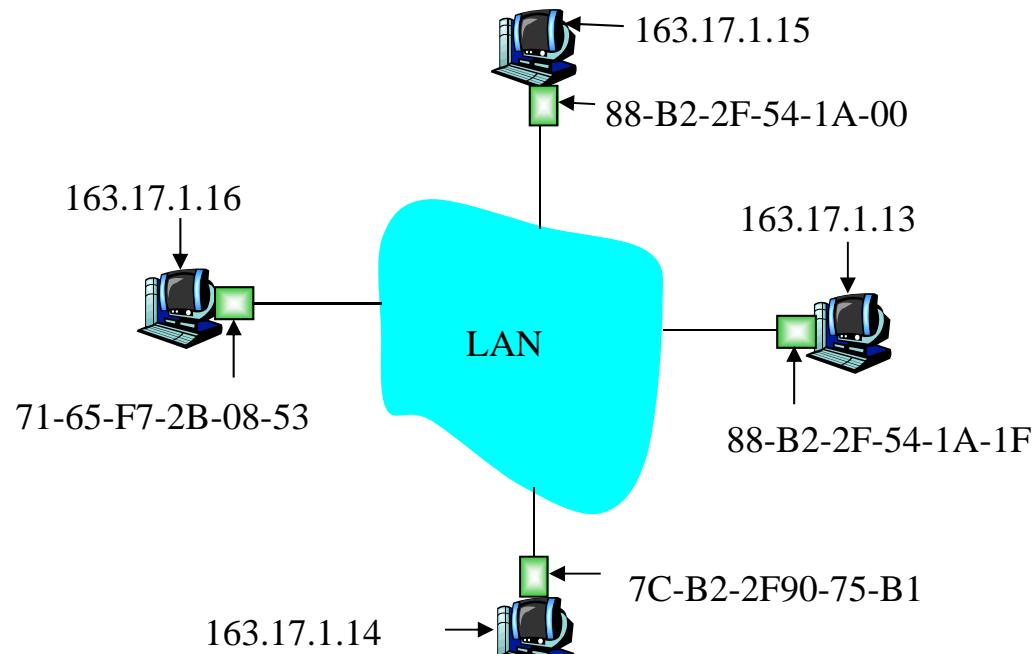
► ARP (Address Resolution Protocol, 位址解析協定) [RFC 826]

► Q：當我們知道節點 B 的 IP 位址時，要怎麼決定 B 的 MAC 位址？

LAN 中部
份的資料
(非全部)

- 區域網路上的每一個 IP 節點（主機、路由器）都有 ARP 表
- ARP 表格：某些區域網路節點的 **IP/MAC 位址對應**
<IP 位址; MAC 位址; TTL>

Hung6 – TTL (存活期)：位址對應會被遺忘的時間 (通常 20 分鐘)



投影片 44

Hung5 ARP只用於區網, 而DNS可解析internet所有IP位址
Ruo-Wei Hung, 2011/8/9

Hung6 Time To Live
Ruo-Wei Hung, 2011/8/9



■ ARP (Cont.)

► ARP 表

IP 位址	MAC 位址	TTL
163.17.1.14	7C-B2-2F90-75-B1	13:46:20
163.17.1.15	88-B2-2F-54-1A-00	12:43:00
163.17.1.16	71-65-F7-2B-08-53	13:50:00

圖5.18 節點163.17.1.13 中可能的 ARP 表格





■ ARP (Cont.)

► ARP 協定

○ 情境一：A 想要傳送封包給同一LAN中的 B，B 的 MAC 位址不在 A 的 ARP 表中

○ 過程步驟

① A 廣播 ARP 查詢封包，內含 B 的 IP 位址

» 目的端 MAC 位址 = FF-FF-FF-FF-FF-FF (廣播的MAC位址)

» LAN 上的所有機器都會收到 ARP 查詢

② B 收到 ARP 查詢 封包，將它(B)的 MAC 位址回應給 A

» 訊框傳送給 A 的 MAC 位址 (單點傳播)

③ A 在它的ARP表格中儲存B的 IP-到-MAC 的位址對應，直到資訊變舊(逾時)

○ ARP 為 “隨插即用” 的：

- 節點在沒有網路管理者介入的狀況下建立它們的ARP表格



■ ARP (Cont.)

▶ ARP 協定 (Cont.)

- 情境二：A 想要傳送封包給不同LAN中的 B，B 的 MAC 位址不在 A 的 ARP 表格中（經由 R 從 A 到 B 傳送資料封包，另假設 A 知道 B 的 IP 位址）

- 路由器 R 中有兩個 ARP 表（對每個 IP 介面均保存一個 ARP 表）

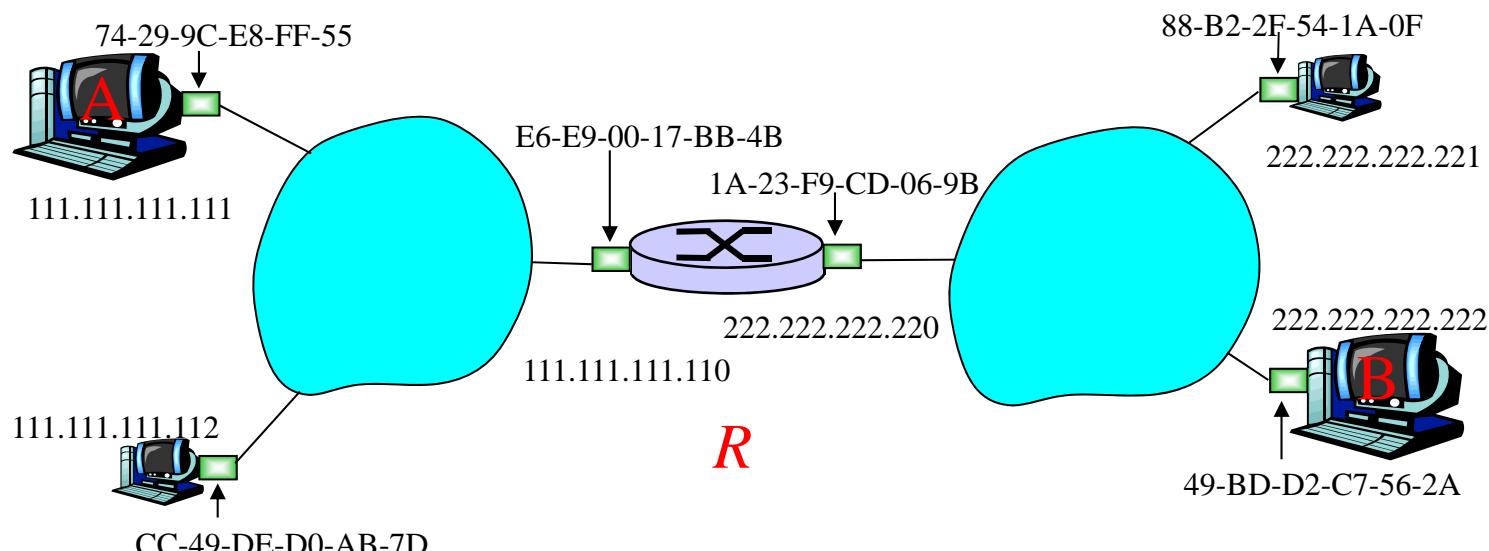


圖5.19 由路由器相連的兩個子網路



■ ARP (Cont.)

▶ ARP 協定 (Cont.)

○ 情境二 (Cont.)

○ 過程步驟

S=A; D=B
S=A; D=R

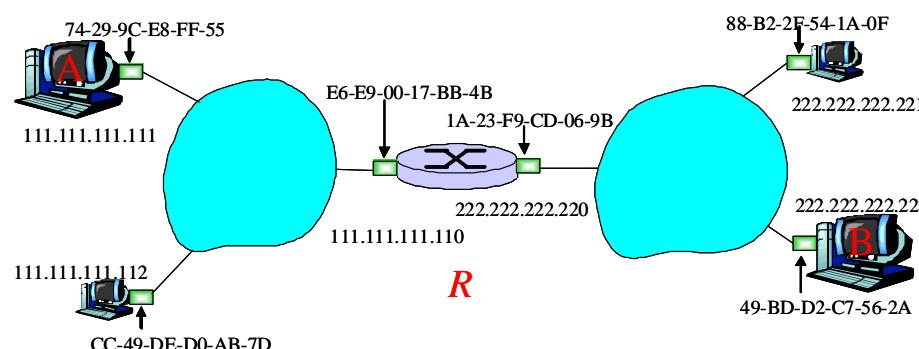


- ① A 建立來源端為 A，目的端為 B 的資料段
- ② A 使用 ARP表格，取得 R 中 $IP = 111.111.111.110$ 的 MAC 位址
- ③ A 使用 R 的 MAC 位址做為目的端、建立連結層訊框、訊框內包含了 A 到 B 的 IP 資料段
- ④ A 的網路卡傳送訊框
- ⑤ R 的網路卡接收訊框

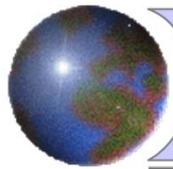
S=A; D=B
S=R; D=B



- ⑥ R 從乙太網路訊框移除 IP 資料段，並看到它的目的端是 B
- ⑦ R 使用 ARP表 ($IP = 222.222.222.220$) 取得 B 的 MAC 位址
- ⑧ R 建立包含 A 到 B 的 IP 資料段的訊框、並傳送給 B



Exercise 5.6
Ex 5.6



第五章 導覽流程



5.1 連結層簡介與服務

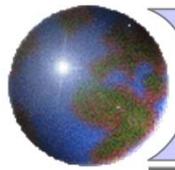
5.2 錯誤偵測和更正技術

5.3 多重存取協定

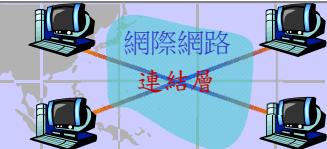
5.4 連結層定址

5.5 乙太網路 (Ethernet)

5.6 集線器和交換器



5.5 乙太網路



■ 乙太網路 (Ethernet)

► “主要的” 有線區域網路技術

- 便宜 – 100 Mbps 網卡只要 \$20 !
- 第一個廣泛使用的區域網路技術
- 比記號環區域網路以及 ATM 簡單、便宜
- 速度不斷地提升：10 Mbps ~ 10 Gbps

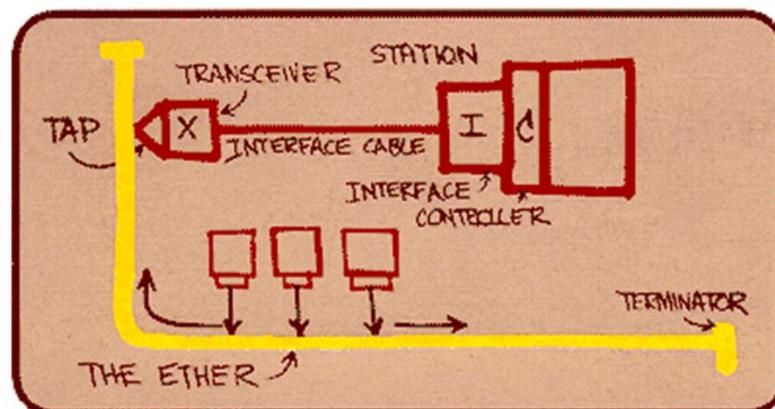


圖5.20 Metcalfe's 的原始乙太網路設計圖

■ 星狀拓樸 (star topology)

- ▶ 在1990年代中期，匯流排拓樸 (bus topology) 是非常普及的
 - 所有節點在相同碰撞範圍內 (可以碰撞其它節點)
- ▶ 現今流行：星狀拓樸
 - 在中央的是交換器(switch)作動
 - Hung7 ○ 每一個 “spoke” 均執行(分離)乙太網路協定 (每一個節點不會發生碰撞)

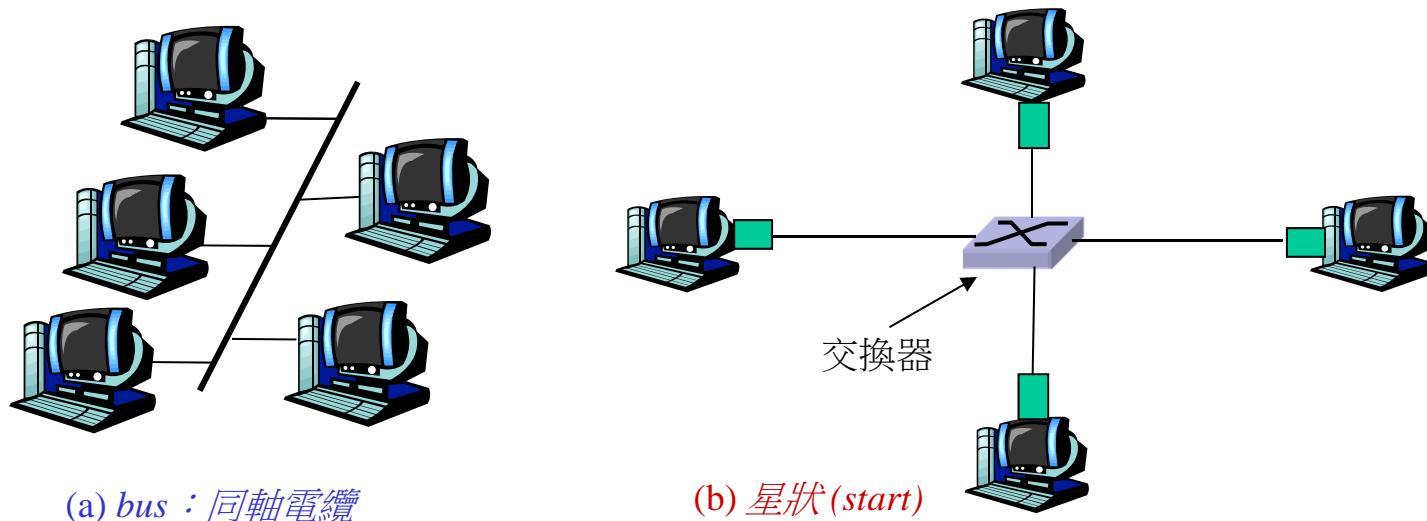


圖5.21 乙太網路 (a) 汇流排架構 (b) 星狀架構

投影片 51

Hung7

中心通訊設備若使用hub(集線器), 則會發生碰撞
Ruo-Wei Hung, 2011/8/9



■ 乙太網路訊框結構 (frame structure)

- ▶ 傳送端網路卡將 IP 資料包(datagram) (或是其他的網路層協定封包) 封裝在乙太網路訊框中
- ▶ 前置位元 (8 bytes)
 - 一個位元組10101011之後有7個位元組的10101010
 - 用來同步傳送端和接收端的時脈速率

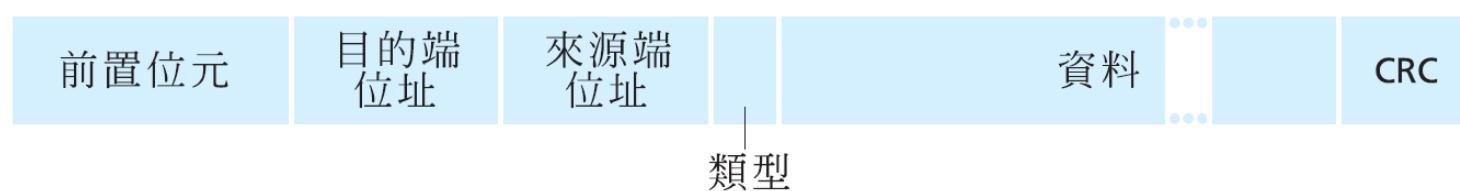


圖5.22 乙太網路訊框結構



■ 乙太網路訊框結構 (Cont.)

▶ (MAC)位址：6 個位元組

- 假如網路卡收到與目的端位址相符的訊框或是廣播位址訊框(例如 ARP封包)，它會將訊框中的資料傳送到網路層協定
- 否則，網路卡會刪除訊框

▶ 類型：2個位元組

- 表示上一層的協定 (通常是 IP 或是其他可能支援的協定，如 Novell IPX 和 AppleTalk)

▶ CRC：4個位元組

- 接收端會檢查此一欄位，如果錯誤就會丟棄此一訊框





■ 乙太網路 – 不可靠(*un-reliability*)的非預接式 (*connectionless*)

Hung8

- ▶ 非預接式：傳送端和接收端NICs之間沒有交握程序
- ▶ 不可靠的：接收端NIC不會傳送 ACK 或是 NAK 到傳送端NIC
 - 傳送到網路層的資料包串流可能會出現間斷(gaps, 資訊電報遺失)
 - 假如應用程式使用 TCP，則間斷會被填滿
 - 否則，應用程式會看到間斷

Hung9

- ▶ 乙太網路MAC 協定：unslotted CSMA/CD

投影片 54

Hung8 NIC = Network Interface Card
Ruo-Wei Hung, 2011/8/9

Hung9 CSMA/CD = Carrier Sense Multiple Access with Collision Detection
Ruo-Wei Hung, 2011/8/9



■ 乙太網路 – CSMA/CD 演算法

► 運作步驟

- ① NIC從網路層接收資料包並建立訊框 (datagram → frame)
- ② 假如NIC感測通道是閒置的，則它會開始傳送訊框。假如它感測到通道是忙碌的，則它會等待通道閒置、接著再開始傳送
- ③ 假如NIC將整個訊框傳送出去而沒有偵測到其它傳送，則NIC完成這個訊框的傳送！
- ④ 假如NIC在傳送的同時，偵測到其它傳送，則它會停止傳送並且傳送出一個擁擠訊號 (jam signal)
- ⑤ 在④之停止傳送之後，NIC會進入指數退回 (exponential backoff) 階段：在第 n 次碰撞之後，NIC會從 $\{0, 1, 2, \dots, 2^m - 1\}$ 中隨機選擇 k ($m = \min\{n, 10\}$)。然後，NIC會等待 $k \times 512$ 個位元時間後，再回到步驟 ②





■ 乙太網路標準

▶ 定義在 IEEE 802.3

- 定義連結層與實體層

▶ 許多不同的乙太網路標準

- 一般 MAC 協定與訊框格式
- 不同 速度：2 Mbps、10 Mbps、100 Mbps、1Gbps、10G bps
- 不同 實體層的媒介：光纖、電纜

Hung10

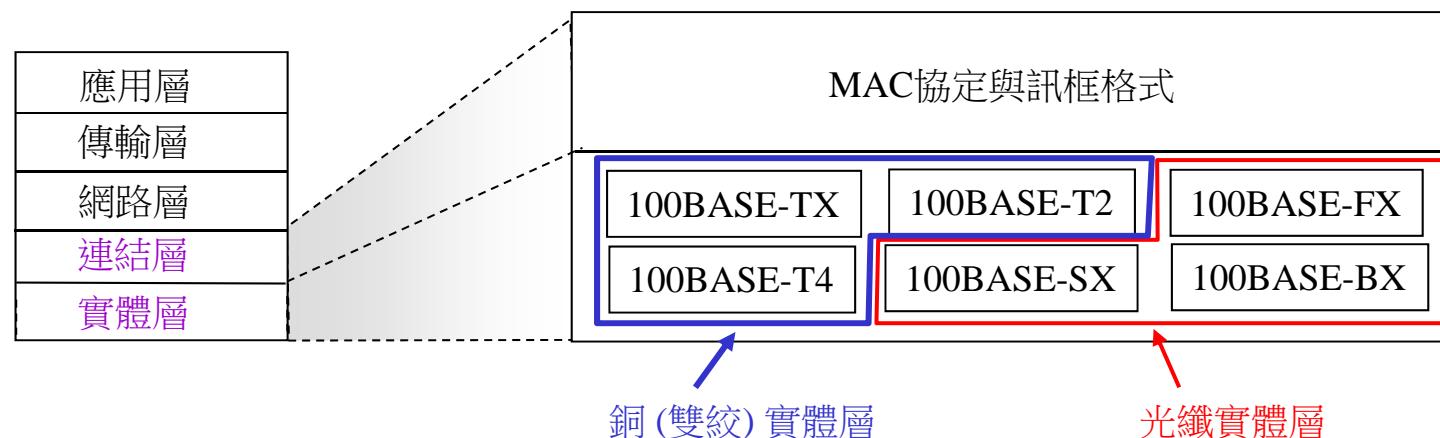


圖5.25 100Mbps 的乙太網路標準 – 共通的連結層、不同的實體層

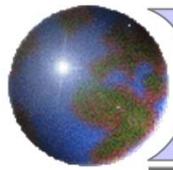
投影片 56

Hung10

BASE: 基頻乙太網路, LAN只有載送乙太網路的資料流.

T: Twisted, 雙絞銅芯線.

Ruo-Wei Hung, 2011/8/9



第五章 導覽流程



5.1 連結層簡介與服務

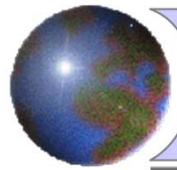
5.2 錯誤偵測和更正技術

5.3 多重存取協定

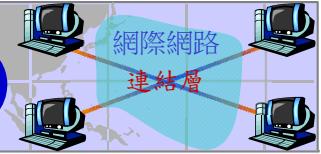
5.4 連結層定址

5.5 乙太網路 (Ethernet)

5.6 集線器(hub)和交換器(switch)



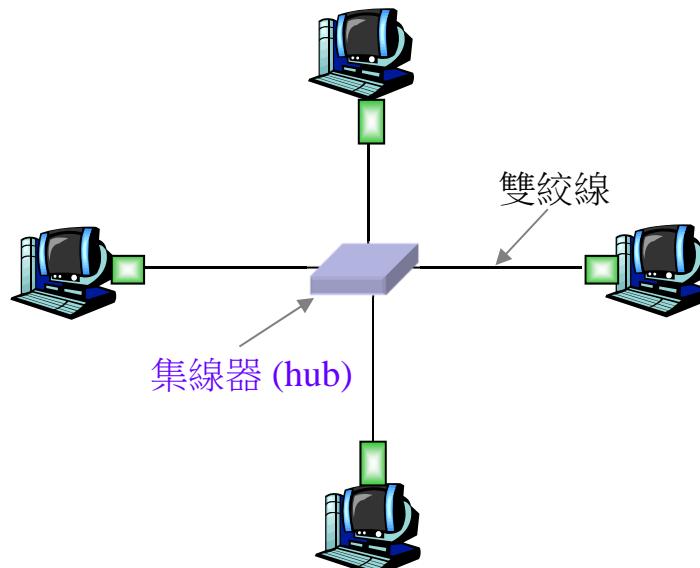
5.6 集線器(hub)和交換器(switch)



■ 集線器 (hub)

▶ 實體層的中繼器(repeater)

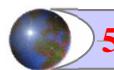
- 從一個連結(link)進入的位元會以同樣的速率被送出到所有其它的連結
- 一個連接到集線器的節點可以碰撞另一個節點
- 沒有訊框緩衝區
- 集線器 沒有 CSMA/CD：由主機的NICs偵測碰撞





■ 交換器 (switch)

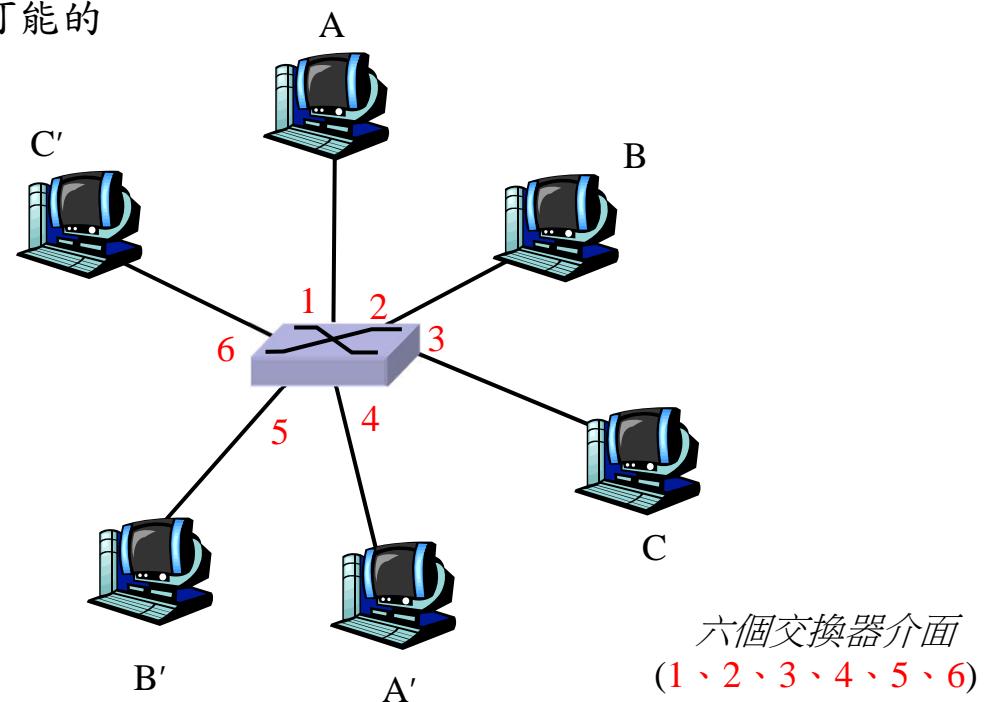
- ▶ 連結層裝置：比集線器聰明，扮演作動的角色
 - 儲存及轉送乙太網路訊框
 - 當訊框在此裝置轉送時，其檢驗進入訊框MAC位址、並選擇性地轉送到一個或更多個連接介面
 - 使用CSMA/CD來存取區段(segment)
- ▶ 透明的
 - 主機感覺不到交換器的存在
- ▶ 隨插即用、自我學習
 - 交換器不需要設定



■ 交換器 (Cont.)

▶ 允許多個同時傳送

- 主機有專用且直接的連結線到交換器
- 交換器有緩衝區，可以緩衝封包
- 乙太網路協定使用在每一個進入的連結、但是沒有碰撞 - 全雙工
 - 每個連結都擁有它的碰撞範圍
- 交換器：A-到-A' 與 B-到-B' 可同時傳送，且不會發生碰撞
 - 使用 hub 則是不可能的

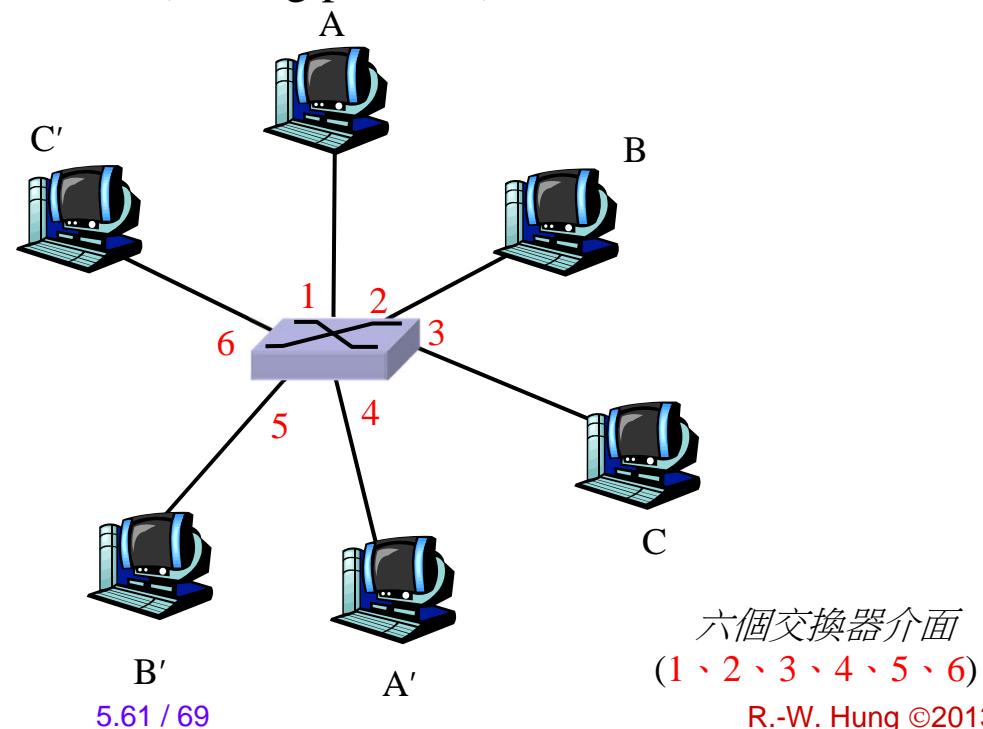


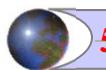


■ 交換器 (Cont.)

► 交換器表 (switch table)

- Q : 交換器如何知道 A' 可通過到達介面 4、B' 可通過到達介面 5？
- A : 每一個交換器都有一個 **交換表**，其欄位如下：
 - <主機MAC位址; 到達主機的介面; 時間標籤>
 - 類似路由器表(routing table)
- Q : 如何建立、維持在交換表中的通道？
- A : 一些像是路由器的協定 (routing protocol)

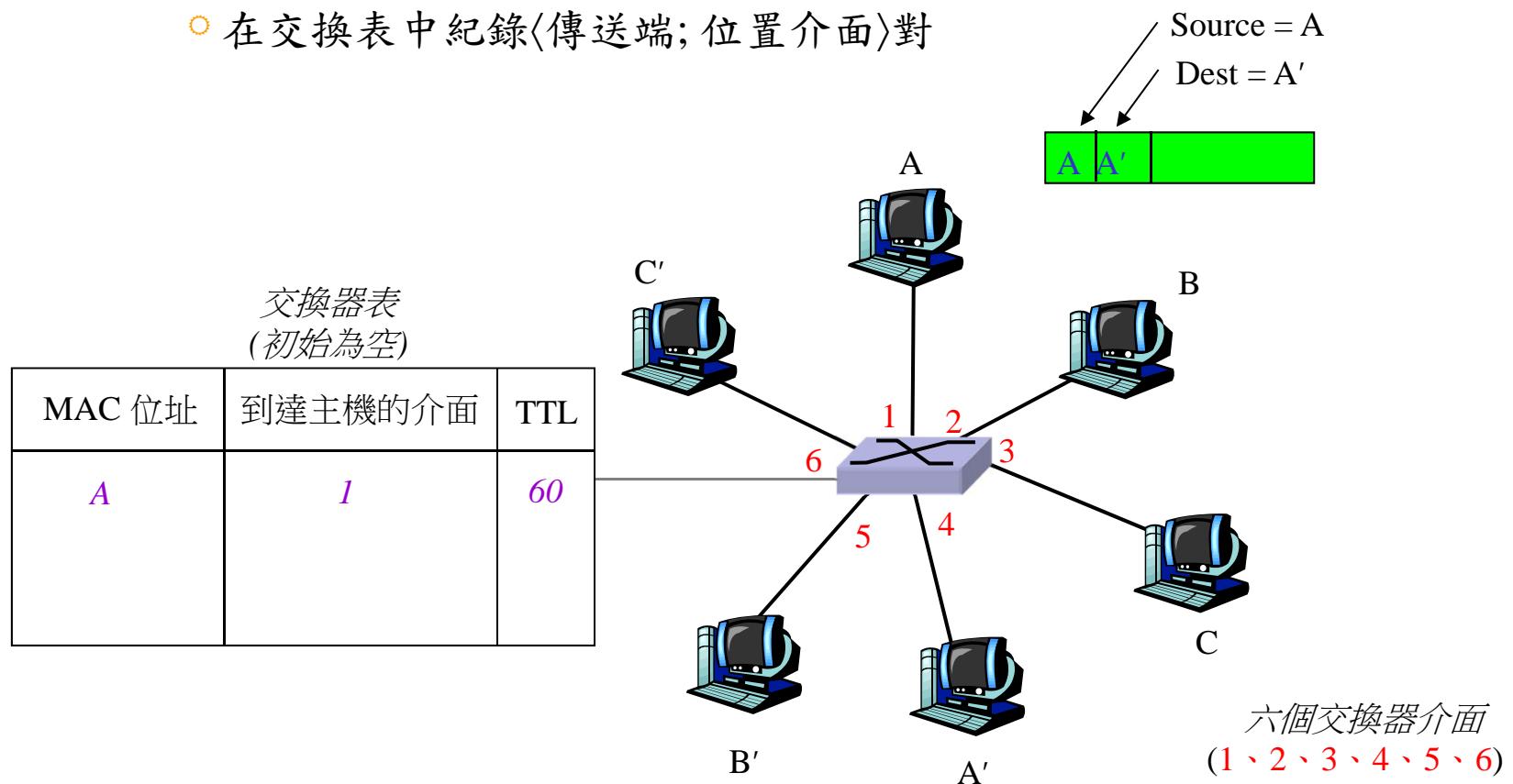




■ 交換器 (Cont.)

► 交換器表的自我學習(self-learning)

- 交換器會學習哪一個主機經由哪一個介面可以到達
- 當訊框抵達時，交換器會“學習”傳送端的位置：進入的LAN區段(LAN segment)
- 在交換表中紀錄<傳送端；位置介面>對

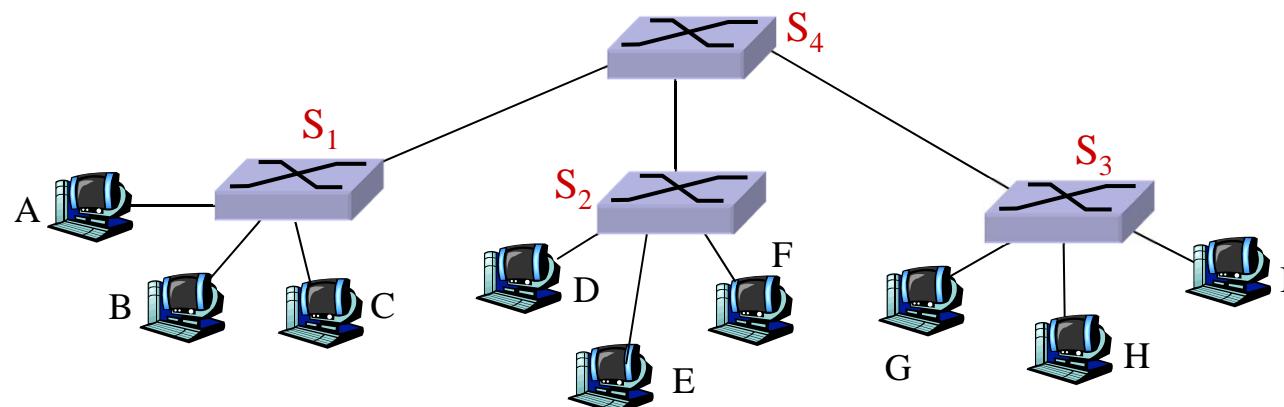




■ 交換器 (Cont.)

▶ 以交換器互相連結

- 交換器之間可以連接在一起
- Q：從 A 傳送封包到 F，如何知道 S_1 把預定的訊框轉送到 F via S_4 與 S_2 ？
- A：自我學習！(How ? Thanking ...)





■ 機構網路

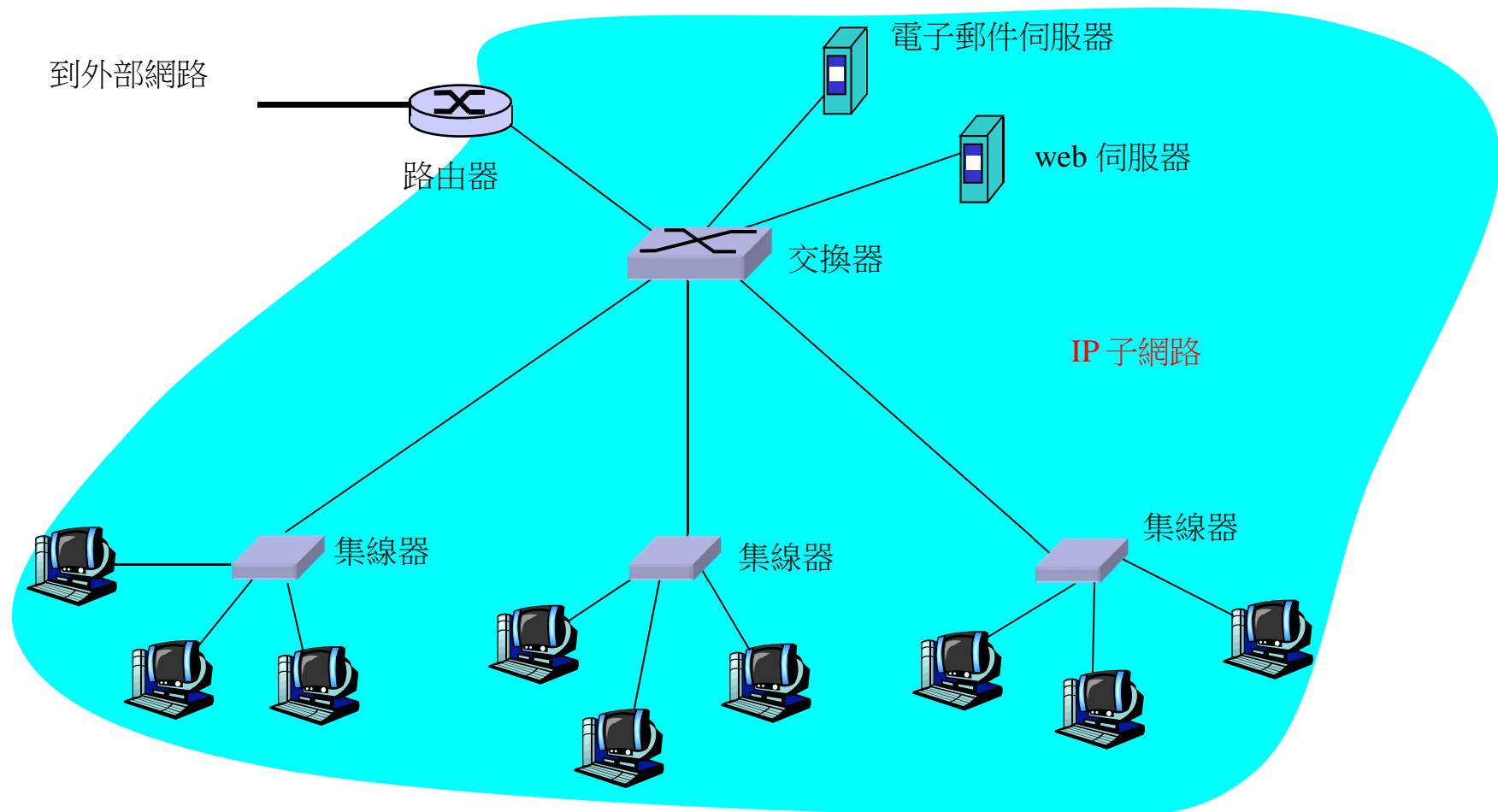
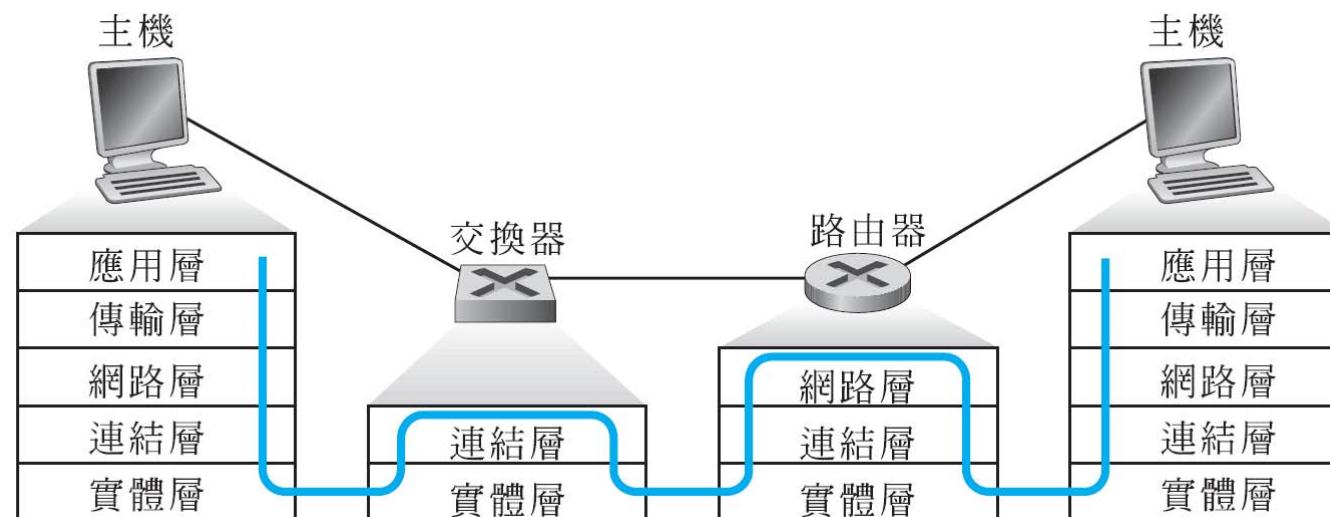


圖5.26 同時使用集線器、乙太網路交換器、及路由器的機構網路



■ 交換器 vs. 路由器

- ▶ 都是儲存並轉送的裝置
- ▶ 路由器：網路層裝置（檢驗網路層標頭）
- ▶ 交換器：連結層裝置
- ▶ 路由器維護路由表，實作路由演算法
- ▶ 交換器維護交換表，實作過濾、學習演算法

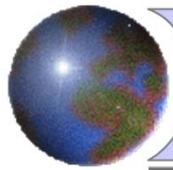




■ 比較總表

	集線器 hub	路由器 router	交換器 switch
流量隔離	無	有	有
隨插即用	有	無	有
最佳化路由	無	有	無
直接傳送	有	無	有





第五章 總結



■ (資料)連結層服務的原則

- ▶ 錯誤偵測、更正
- ▶ 分享寬頻通道：多種存取
- ▶ 連結層位址

■ 各種連結層技術的例證和實作

- ▶ 乙太網路
- ▶ 交換器區域網路
- ▶ PPP ×
- ▶ 連結層的虛擬網路 ×

End of Chapter 5

課程結束

謝謝同學本學期的配合

