第五章 傳輸層

將介紹傳輸層的多工模式與連線管理技術，並讓讀者有開發網路應用程式的基本技巧。

5-1傳輸層簡介

在 OSI 參考模式中，**『傳輸層』(Transport Layer)** 以下層次的功能是在複雜網路中，找到所期望連接的電腦，然後提供一個相互能通訊的連線。也就是說，通訊雙方工作站無論在網路任何地方，經過這些層次的處理後，宛如兩部工作站就旁邊，也好像兩部工作站已可結合成一部，有關它們之間的訊號處理、實體連線問題、尋找工作站位址、等等工作都已完成。因此，有關傳輸層以下的服務稱之為『連線導向服務』(Connection-oriented Service)。從傳輸層以上的層次 (會議層、表現層和應用層) 的功能則著眼於應用程式的觀點，它們提供有關應用軟體所需的程序之間 (Process to Process) 通訊為主，因此稱之為『應用導向服務』(Application-oriented Service)。傳輸層是位於兩種服務之間的介面，是整個通訊協定階層的核心，如下圖 5-1 所示。因此，傳輸層負責提供通訊雙方使用者 (或程序) 之間，以約定的通訊品質來傳送資料。當使用者之間的連線建立後，傳輸層便要提供適當的通訊品質，並且監督資料傳輸的過程，以保證該通訊品質的維持。如無法達到必須通知使用者。

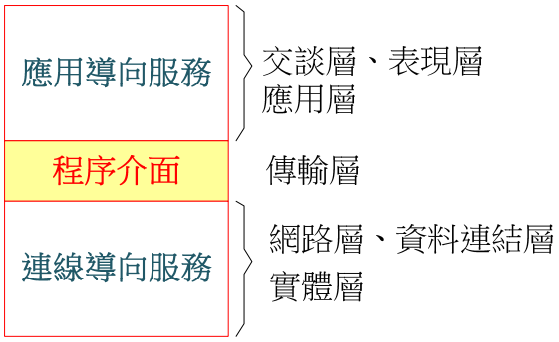


圖 5-1 傳輸層的主要功能

另一觀點而言，傳輸層以下的通訊軟體大多和網路架構比較有關聯，是網路工程師架設網路時較須考慮的重點；而軟體工程師欲開發網路應用程式時，對於傳輸層以後 (交談、表現、及應用層) 的處理，才是它考慮的重點，亦是，開發應用程式只要以傳輸層作為起始點就可以了。如果網路上提供完備的傳輸服務介面，開發網路應用軟體就如同在一般作業系統

上開發程式一樣方便。一般網路作業系統（如 Unix）都會提供有關傳輸層的介面程式（如，socket()），讓使用者直接開發應用程式，而無需考慮目前處於何種網路架構之下。

5-2終端對終端連線

圖 5-2 是整個網路通訊協定大略的連接功能結構。其中實體層的功能是將資料轉換成訊號，然後發送到傳輸媒介上傳輸；鏈路層是將資料訊框包裝發送到網路上，並負責存取網路及控制可能發生的錯誤，因此屬於『節點對節點連接』(Node-to-Node Connection)；網路層是負責在複雜網路上尋找目的工作站的位址，並負責連接到它，稱之為『工作站對工作站連接』（Station-to-Station Connection），因此到網路層為止，我們已經讓兩部工作站連接在一起了。

但在任一部主機電腦上也許會有多個網路程序（Process）正在執行，這些程序（使用者或程式）都必須透過這部工作站和其它工作站內的程序通訊，如何提供程序之間的連線，這便是傳輸層主要的工作。一般傳輸層都會提供許多傳輸埠口，來讓應用程式連接，也使應用程式之間可達到實質的連線，因此稱之為『終端對終端連線』(End-to-End connection)，如圖 5-2 中 a0 與 b0 埠口之間的連線。又圖 5-2 中，工作站 A 透過網路層服務連接到工作站 B，傳輸層提供工作站 A 之程序（IE）和工作站 B 的程序（Web Server）之間的連線，此連線稱之為『程序對程序的連線』(Process-to-Process Connection)。由圖 5-1 中，我們觀察到傳輸層提供連接導向和應用導向程序之間的介面，應用程式只要連接到傳輸層，就可以透過網路連結到對方的應用程式，無須考慮網路架構，才合乎通訊協定的堆疊原理。

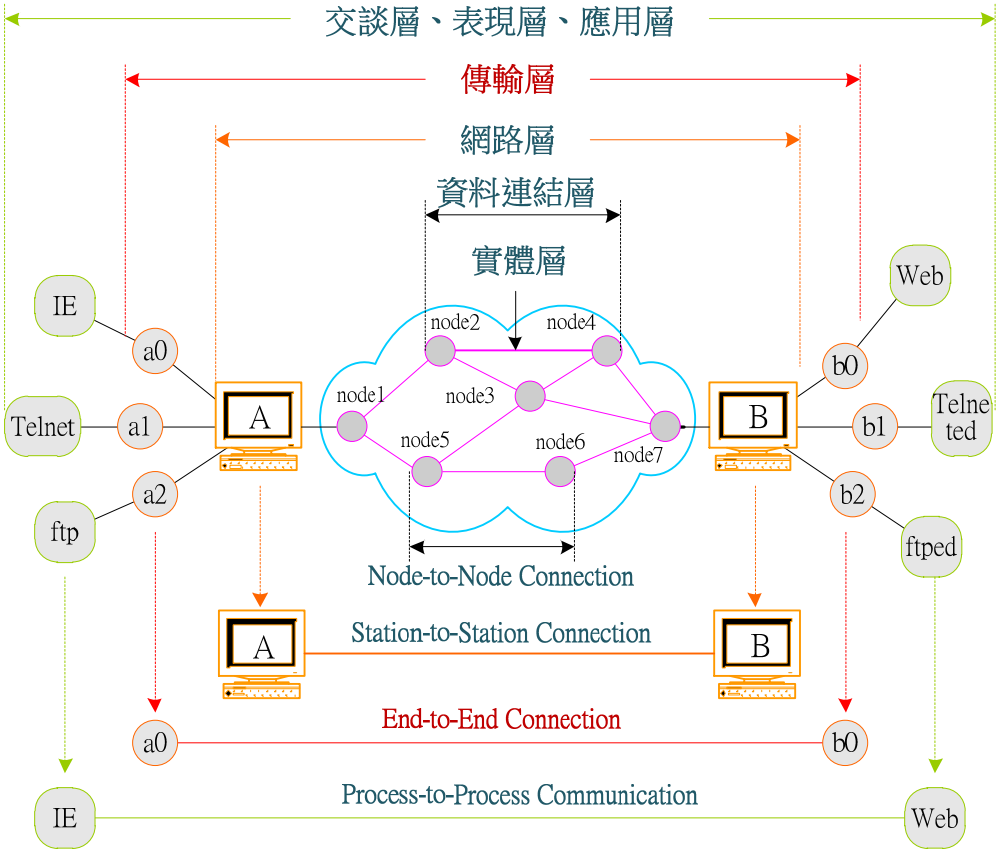


圖 5-2 網路連接功能圖

5-3 傳輸模式型態

傳輸模式是決定網路架構的型態，依照不同的應用環境有下列兩種模式：

5-3-1 單一層次傳輸模式

『**單一層次傳輸模式**』(Single-Layer Transport Mode) 表示已將傳輸層和網路層結合成單一層次，其原因如下：一般在區域網路的應用環境下 (或稱區域網路作業系統，如 Microsoft Network)，網路範圍不是很大；而且主要訴求在於高效率和高可靠度的應用，如辦公室自動化、或工廠自動化等等，此時傳輸層勢必採用連接導向方式，以提高整個網路的效益。再由另一方面來觀察，雖然傳輸層 (連接導向) 可以保障非連接方式網路層的可靠性，但也僅侷限於當網路層發生錯誤時，再由傳輸層負責將它偵測出來，並從事補救的工作 (如，要求對方重送)，對整個網路的傳輸效益而言並沒有提昇。因此，在範圍較小的網路環境裡，也儘量考慮使用連接導向的網路層。既然如此，那麼傳輸層是否有存在的必要呢？亦是值得商確的問題。所以，一般區域網路作業系統將傳輸層和網路層合併，以一個連接導向的網路層來取代即可，如圖 5-3 中 Microsoft Network 的 NetBEUI (NetBIOS Extended User Interface)。同

樣的，NetBEUI 也是應用服務導向的介面程式，具有路徑選擇的功能，亦能提供若干個介面存取點讓程式設計者開發應用程式。然而 NetBEUI 與 LLC （Logical Link Control）之間連線是一對一的關係，LLC 是提供連接導向服務，因此，對整個網路而言，可靠性高，效率也高。

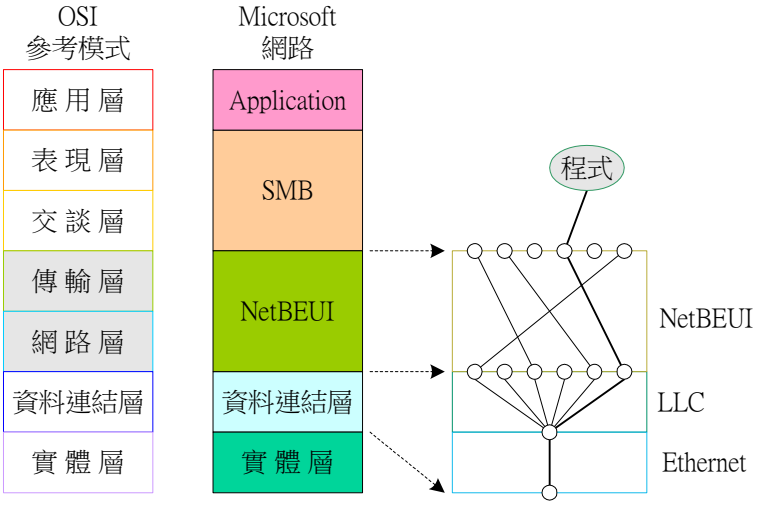


圖 5-3 單一層次傳輸模式

5-3-2 兩層次傳輸模式

在較大的網路系統之下，網路連接所遇到的問題可能更為嚴重。例如，不同網路之間資料傳輸格式的轉換、資料傳遞延遲時間的不同、或網路之間路徑的尋找等等。所以我們勢必另闢一個層次來負責網路之間連線的問題，但我們又希望能提供一個和網路環境無關的介面，讓程式設計者開發網路應用程式。因此，『**兩層次傳輸模式**』（Two-Layer Transport Mode）中的網路層和傳輸層分別負責這兩項工作。

但這兩個層次都有可能提供非連接方式或連接導向方式，我們以目前使用較廣泛的 TCP/IP 網路和 ATM 網路來說明它們之間的關係，如圖 5-4 所示。在大網路環境裡，透過網路之間連接到工作站，可能會跨越許多不同網路架構，考慮到它們之間連結效率的問題，一般都採用電報傳輸模式（如 IP）。有關網路傳輸可能發生的錯誤，例如，位元錯誤、順序錯亂、封包遺失、或封包重複接收等等問題，必須由另一個可靠的傳輸層（如 TCP）來負責。但針對較短的訊息（如 SMTP）、廣播訊號、或需要即時回應的近距離訊息（如 SNMP），則傳輸層也許會採用非連接方式（如 UDP）。

目前大型網路的應用層次漸漸提高，無效率的非連接網路似乎漸漸不能滿足系統的要求。許多大都會網路公司（如 HiNet）希望提供更有效率且傳輸速率更快的網路環境，也紛紛開始架設虛擬電路連接方式的 ATM 網路。也有許多公司開始研發具有虛擬電路連接功能的路由器，主要希望提供一個可靠性較高之連接導向方式的網路連結（網路層功能）。但目前有關網路應用軟體的開發，都還架設在 TCP/IP 網路環境之下，也希望能即時連結到所有 Internet 網路上，因此，才有 IP over ATM 的連結的產生。基本上，IP over ATM 的網路之間連接是屬於虛擬電路（分封交換）方式，但網路之間封包的傳送還是以 IP 封包為主。如圖 5-4 所示，傳輸模式就有 UDP over IP over ATM（傳輸層是非連接方式）和 TCP over IP over ATM（傳輸層是連接導向方式）兩種可靠性較高的連結模式（網路層為連接導向方式）。

		網路層	
		非連接方式	連接導向
傳輸層	非連接方式	UDP over IP	UDP over IP over ATM
	連接導向	TCP over IP	TCP over IP over ATM

圖 5-4 傳輸層和網路層的不同架構組合

5-4 傳輸多工模式

傳輸層和網路層之間的連接方式，也如同所有通訊協定層次關係一樣，層次之間是透過『服務存取點』（Service Access Point，SAP）來銜接。『**多工模式**』（Multiplexing）就是『傳輸服務存取點』（Transport Service Access Point，TSAP）和『網路服務存取點』（Network Service Access Point，NSAP）之間的對應關係，它們之間可能是一對一、多對一、或是一對多的多工連線（如 1-8-4 節說明）。傳輸層為提供應用導向與連結各項服務之間的介面，它的多工關係就顯得特別重要。我們可以說，TSAP 就是應用層程式設計的起始介面，因為它獨立（Independent）於網路架構關係。因此，一般傳輸層必須提供多個連接點，例如，X.25 以 12 位元表示可達 4096 個連接埠；TCP 以 16 位元表示可達 65536 個連接埠。傳輸層的多工方式牽涉到網路型態和網路應用範圍。我們以下列兩大類來介紹。

5-4-1 向上多工模式

傳輸層以下的連結，也許必須經過多層次的子網路連接，相對的，它的連接費用也會隨之提高，因此，對於較廣泛的大型網路（如 Internet）大多採用『**向上多工模式**』（Upward Multiplexing）。也就是說，一個網路連線讓多個應用程式所共用，如圖 5-5 所示。針對每一個傳輸服務存取點的定址方式，就採用『NSAP：TSAP』的對應關係；舉例而言，TCP/IP 網路上是以『IP 位置：傳輸埠口』方式定址，譬如 163.15.2.1:23，則表示 IP 位置為 163.15.2.1；TCP 傳輸埠口為 23。至於 TSAP 執行時間的排程問題（Scheduling），便依照工作站作業系統（如 Unix）的排程管理，例如，分時系統（Time Sharing System）是以時間分割或循環點名法（Round-robin）來輪流處理；而即時系統（Real-time System）是以優先權較高的 TSAP 優先處理。

如圖 5-5 所示，工作站 A、B、C、D 為向上多工模式，亦是一個網路連線提供多個應用程式的銜接。如工作站 D（IE）瀏覽工作站 A（Web Server）上的網頁，而它們之間的連線為 163.15.2.1:80 ←→ 163.15.2.6:1568；另外由工作站 C 遠端簽入工作站 A 是由 138.42.3.78:2458 和 163.15.2.1:23 之間連線所達成。同樣的，工作站 B 和工作站 A 之間的 ftp 是由 163.15.2.1:20 到 144.45.43.2:2345 的連線來傳輸檔案。

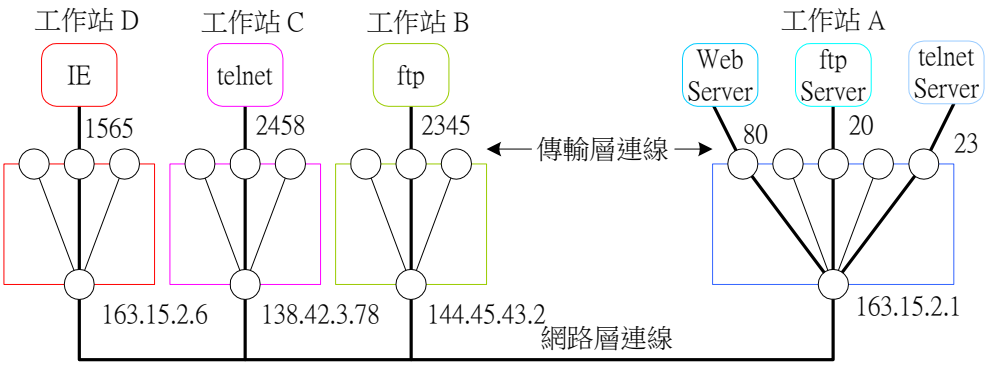


圖 5-5 傳輸層與網路層的向上多工關係

5-4-2 向下多工模式

『**向下多工模式**』（Downward Multiplexing）表示一個傳輸介面連結到多個網路連線上，如此針對大量傳輸資料的應用，可提高其傳輸速率。如圖 5-6 所示，ftp Server（工作站 A）採用三條網路層傳輸連線的向下多工模式。在這種模式下，如果使用 Ethernet 網路，就必須使用 3 只網路卡，每一個網路卡使用一個網路位址。因此，工作站 A 擁有三個網路位址（163.15.2.1、163.15.2.2、163.15.2.3）。理論上，每一只網路卡傳輸速率如果是 100 Mbps

（100BaseTx），則 ftp Server 網站傳輸速率應該為 300 Mbps，但是否可完全達到該速率，這可和網路架構型態有絕對性的關係。如果網路層使用 ATM 網路，則必須連結 3 個虛擬通道（Virtual Channel, VC），再依照各條通道傳輸速率的總和為該網站之速率。

如圖 5-6 所示，工作站（B、C、D）如使用 ftp 程式（ftp_B、ftp_C、ftp_D）連接到工作站 A 的檔案伺服器（FTP Server），則它們之間連線如下：ftp_B 到 FTP Server 為 144.45.43.2 與 163.15.2.1 的連線；ftp_C 到 FTP Server 為 138.42.3.78 與 163.15.2.2 的連線；ftp_D 到 FTP Server 為 163.15.2.6 與 163.15.2.3 的連線。因此，對工作站 A 的檔案伺服器而言，傳輸速率就是三條網路傳輸速率的總和。（但此 FTP Server 可需有特殊處理能力來分辨不同的傳輸才行）

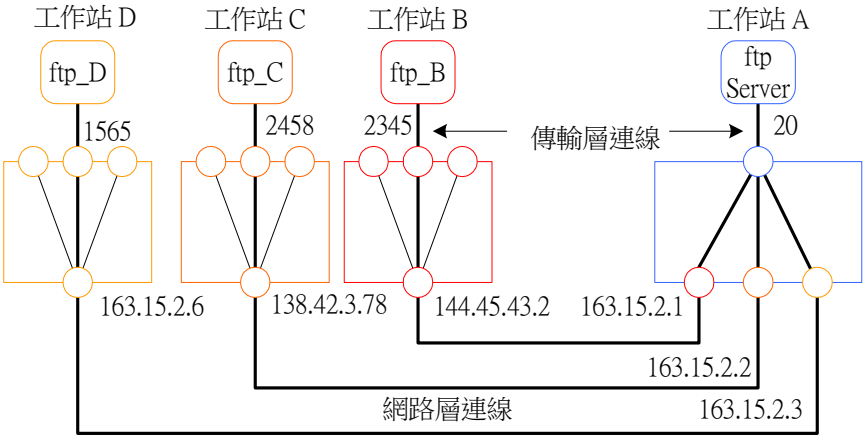


圖 5-6 傳輸層與網路層的向下多工關係

5-5傳輸服務元件

『**傳輸服務元件**』（Transport Service Element）是讓使用者（或應用程式）來存取傳輸服務的介面。傳輸服務與網路服務之間的相異如下：網路服務是用來模式化真實網路所提供的服務，然而傳輸服務是提供一個與實際網路無關的抽象環境，讓使用者來連接。真實網路上會有封包遺失與封包錯誤的現象，因此網路服務通常都不太可靠；但傳輸服務主要訴求在於脫離真實網路環境，並提供一個抽象化的系統介面，讓使用者直接存取而無須理會真實環境。亦即在較不可靠的網路環境上，提供一個可靠的介面服務。至於如何提供可靠的傳輸介面，如錯誤偵出、連線管理、資料遺失、或資料重複等等問題，乃是傳輸層的主要功能。

正因如此，使用者連接傳輸服務介面時，無須考慮網路上可能發生的各種情況，以及網路的架構為何。傳輸服務元件好比是一般作業系統中的系統呼叫（System Call）一樣，其處理方式也如同**『程序與程序間通訊』（Process-to-process communication）**的運作方式。對使用者而言，其『不知道也不需要知道網路的存在，或網路上資源的存取方式』。網路上的傳輸服務元件多半大同小異，我們以柏克萊（Berkeley）Unix 系統的 TCP 協定為例，其服務元件如下：

- **socket()**：產生新的通訊端點。當使用者呼叫 socket() 程式介面時，會產生一個新的端點（或稱傳輸埠口），亦表示已完成建立連線。socket() 參數裡描述所使用的位址格式，和希望連線的服務型式。成功的 socket() 呼叫會回傳一個普通檔案描述子（file identify）；接下來，透過網路存取資源就如同在本身電腦上存取檔案一樣，都是針對這個檔案描述子來做存取的動作。Socket() 呼叫方式和開啟檔案的 open() 介面程式完全一樣。

- **bind()**：銜接一個存取位址（TCP 埠口號碼）。socket() 呼叫成功後，只配置記憶體空間，並未連接到傳輸層的存取點。在 TCP 協定的存取點稱之為埠口（port）；我們可以利用 bind() 服務元件呼叫，將 socket() 所產生的端點銜接到（attach）到某一個空閒的埠口以後就可以利用這個埠口和遠端使用者通訊，通訊模式就用類似管道（pipe）的方式，見圖 5-7 所示。

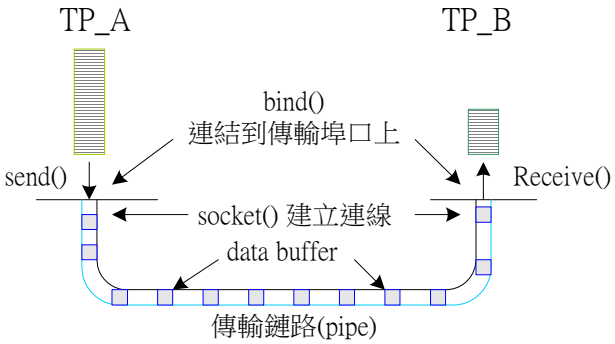


圖 5-7 傳輸鏈路

- **listen()**：宣告願意接受連結。當使用者呼叫 listen() 程式時，傳輸層會配置記憶體空間作為對方要求連線時存放訊息的佇列，並傳回訊息給呼叫端。

- **accept()**：接收呼叫，直到對方嘗試連線到達為止。當 listen() 呼叫後，表示已準備好接收訊息。接收訊息以 accept() 程式呼叫來完成，呼叫後該程序（process）會進入等待狀態，直到對方要求連線訊息到達。
- **connect()**：主動嘗試建立連線。當使用者呼叫 bind() 後，希望主動建立與遠端之間的連線，可呼叫 connect() 程式介面來達成。
- **send()**：透過連結傳送資料。傳輸層之間的連線就像管道（pipe）一樣，當使用者以 send() 傳送資料，就如將資料填入管道內。管道內能夠存放資料多寡取決於 listen() 呼叫時所配置的空間。
- **receive()**：透過連結接收資料。
- **close()**：釋放連結。

使用者可直接透過檔案描述子（socket()）來執行傳送與接收（send()/receive()）的動作，當執行 send() 時，是將資料傳送緩衝器內，而緩衝器以先進先出（First-in First-out, FIFO）的佇列排列，網路層再依序傳送到網路上；接收端收到資料後，也依照 FIFO 佇列的順序排列於接收緩衝器上，等待使用者以 receive() 呼叫來索取。如以抽象資料型態來觀察，雙方通訊的動作就如同管道（pipe）連線一樣，如圖 5-7 所示。又從另一個觀點來看，當傳送端到將資料寫入管道時，並不表示資料已發送到網路上，而如何發送到網路上，是依照作業系統的排序順序來決定；由接收端而言，當執行一個 receive() 呼叫時，是由接收緩衝器上讀取一份資料，但此時緩衝器不一定會有資料可讀取，因此也可能發生等待的現象，到底是否要等待也是由作業系統（或應用程式）來決定。

5-6 傳輸連線管理

傳輸層最主要是提供應用服務的傳輸連線，因此連線管理就顯得格外重要，我們以下列幾種較重要之功能來介紹

5-6-1 定址方式

傳輸層的『**定址方式**』（Addressing）是以指定傳輸埠口方式。傳輸埠口即是連接應用程式的位置，應用程式必須隨時監視該埠口是否有連線要求，並給予適當的服務；也就是說，

我們想要連結遠端電腦上某一個應用程式，只要連接到該埠口位置，便可以得到所需的服務。例如，在 168.4.5.1 電腦上有一個網頁伺服器，其埠口是 80，因此任何遠端的使用者只要連接到 168.4.5.1:80 便可連接到該網頁伺服器。我們以圖 5-8 來說明其運作情形：

- 工作站 B（168.4.5.1）上有一部網頁伺服器，它位於傳輸埠口 80 的位址。網頁伺服器並呼叫 listen() 等待使用者連接。
- 工作站 C（172.6.4.8）上的 IE 程式欲連接到網頁伺服器，它透過 socket() 和 bind() 介面程式銜接到傳輸埠口 1567 的位址。緊接著，IE 發出 connect() 程序呼叫要求連接網頁伺服器，該程序以 172.6.4.8:1567 為原始點，而以 168.4.5.1:80 為目的地。
- 工作站 C（172.6.4.8）和工作站 B（168.4.5.1）的網路層以下的通訊軟體連線成功。
- 工作站 C 之傳輸埠口（172.6.4.8:1567）的連線要求訊號到達對方傳輸層埠口位址（168.4.5.1:80）。
- 網頁伺服器如果同意，則雙方就開始建立起傳輸連結。它們之間連線是 172.6.4.8:1567 與 168.4.5.1:80 兩個傳輸埠口為端點。

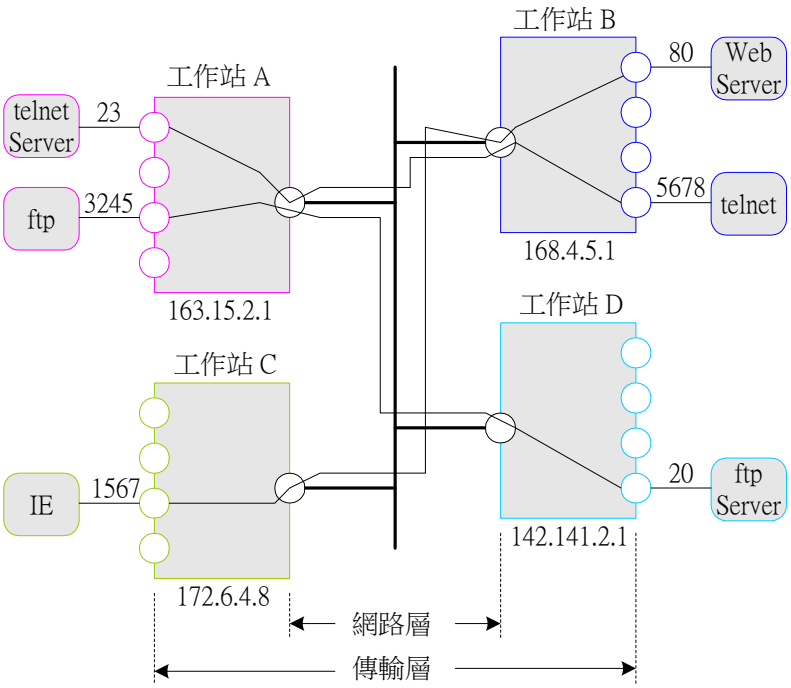


圖 5-8 傳輸埠口定址方式

上述連線步驟中存在一有趣的問題，為什麼連線電腦知道網頁伺服器位於埠口 80？可能的答案是該網頁伺服器已裝設在埠口 80 很久，所以大家都知道。其實我們通常會將常用的伺服器都將其固定在某一個埠口，即所謂的『著名埠口』(Well-know Port)，並將其編列成冊廣播。漸漸地，一般使用者要連結某一部電腦上伺服器，只要知道電腦的 IP 位址就好，而不必知道伺服器到底在哪一個埠口。但對於較不常用伺服器還是要指定埠口位址。

並非每一個伺服器（或稱應用程式）都監視自己的傳輸埠口，對於較不常用伺服器大多不再自行監控自己的埠口，監控的工作由一個稱之為『程序伺服器』(Process Server)來負責。

遠端電腦想要和伺服器連線時，首先以『起始連線協定』(Initial Connection Protocol) 和程序伺服器交談，程序伺服器再將該連線轉接到其所希望之伺服器上。對於傳輸層上每一埠口所連接的伺服器，或者連接在哪一部電腦的埠口，要讓一般使用者都知道，的確是件非常困難的事(尤其是對一些較不常用的伺服器)。所以我們可以在工作站上裝設一個特殊伺服器，用於登錄本工作站所有伺服器的名稱和埠口位址，稱之為『名稱伺服器』(Name Server)。使用者為了尋找出相對應的伺服器的埠口位址，首先和名稱伺服器連線。此時使用者送出一個訊息描述服務的名稱，然後名稱伺服器送回一個埠口位址。使用者得到埠口位址後，再依照埠口位址和所期望連接的伺服器建立連線。

在較大的網路應用環境裡，許多應用伺服器分散在網路各個工作站上，甚至有許多工作站專門處理某一特殊伺服器 (如檔案伺服器)，因此『本地名稱伺服器』(Local Name Server) 已不能滿足環境的需求，『分散式名稱伺服器』(Distributed Name Server) 即因應而生。分散式名稱伺服器不只紀錄有關本身電腦的伺服器埠口位址，也紀錄其他相關工作站上的伺服器，甚至可設定一部主機電腦專門負責該工作。在此模式中，當新的伺服器產生，必須向名稱伺服器註冊，包括它的服務名稱以及自己的埠口位址 (包含主機位址)。名稱伺服器會將這個資料記錄在內部的資料庫裡，因此若有查訊進入，就可以在資料庫內找尋答案。目前名稱伺服器都有監聽的功能，在網路上監視哪個伺服器放在什麼位置，再將其紀錄在資料庫內等待查詢。

如果網路上伺服器過多，而且查詢訊息非常忙碌，並非一部名稱伺服器可以處理時，便有建立多部名稱伺服器的必要。對於網路上多個資源（伺服器）所存放的埠口位址，就有必要以階層式來定址。階層式定址方式就類似檔案目錄一樣，用樹狀結構來命名，儲存和處理

伺服器埠口位址的伺服器稱之為『目錄伺服器』(Directory Server)。每一個目錄伺服器負責被查詢整個資源目錄中的一部份，如果被查詢到非本目錄範圍，則具有轉送的功能。(詳細運作方式請參考拙著『Internet 網路原理與實務』)

5-6-2 建立連線

在鏈路層裡『**建立連線**』(Connection Establishment) 非常的簡單，因為它是兩實體連線之間建立連線。一般兩實體連線距離多半不會太遠，發送連線要求訊號，對方大多能即時接收，如接收不正常也能即時回應給傳送端。但在傳輸層的連線控制就不是那麼容易，如果子網路 (網路層) 使用電報傳輸模式 (Datagram)。要求連線訊號在網路上傳送可能會發生嚴重的延遲，因而產生連線錯誤的情況。

如圖 5-9 所示，工作站 A (TP_A) 發送要求連線的訊號 CR (Connect Request) 後啟動計時器，當時網路嚴重塞車而使連線訊號發生阻塞。傳送端在計時器溢時後未收到對方回應訊號，認為連線訊號已遺失了。傳送端另再發送出傳線要求訊號，而且該訊號比前次訊號更早到達工作站 B (TP_B)。TP_B 連續回應兩個同意連線訊號給 TP_A，TP_A 以為這是接收端重複傳送回應訊號，而不知道已建立兩條連線。例如，使用者和銀行電腦連線，要求銀行將帳戶 A 內扣出 5 萬元轉入帳戶 B。連線訊號發生嚴重的延遲，使用者電腦再次發送連接訊號，銀行電腦接收到兩筆要求連線訊號，這時候如果沒有良好的控制方法，銀行將會轉帳兩次而發生嚴重錯誤而不自覺，使用者電腦也沒有發現異狀。

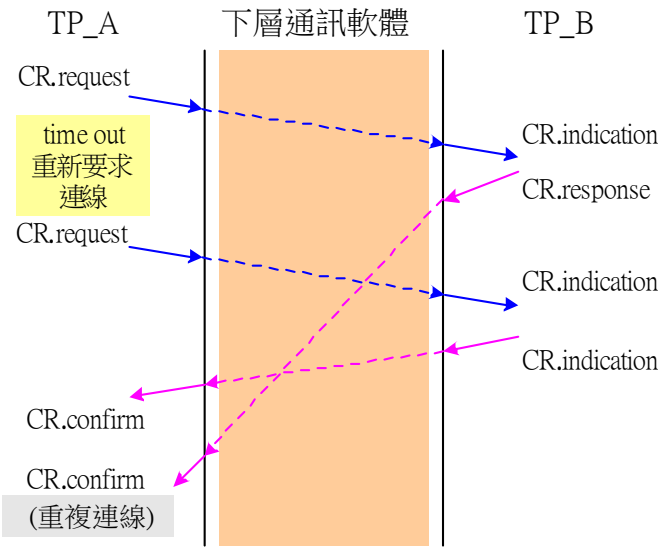


圖 5-9 重複連線錯誤

我們可以發現會發生這個錯誤的主要原因是接收端和傳送端不知道這個連線訊號是重複的，如果我們想出辦法讓每個連線都編有號碼，兩端就較有可能判斷連線是否重複的。為了解決這個問題，Tomlinson（1975）提出**『三向式握手法』（Three-way Handshake）**。其運作的主要原理，就是不管要求訊號或回應訊號都編有序號，尤其回應時必須指明這是回應第幾號要求連線訊息；對於序號的編列不必依照一定的順序，只要能標示出獨立訊息便可以。如圖 5-10 所示，工作站 A（TP_A）送出要求連接訊號（Connect Request，CR）並附帶序列號碼 x(seq=x)；工作站 B(TP_B) 針對 TP_A 的要求而送出同意連線訊號 （Ack(ack=x)），並標示出本回應訊號的序號(seq=y)；TP_A 接到 TP_B 的同意連線訊號 Ack(seq=y, ack=x)，便知道是針對哪一個連線要求的回應，並在傳送一個確認訊號表示連線成功 Ack（seq=x, ack=y）；TP_B 收到確認訊號也知道針對哪一個要求連結訊號所建立的連線已連接成功。

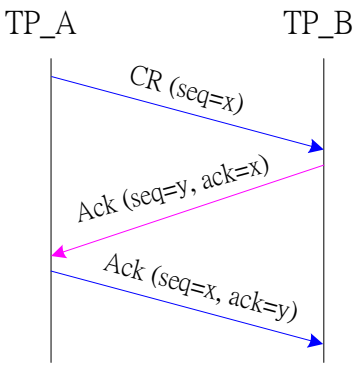
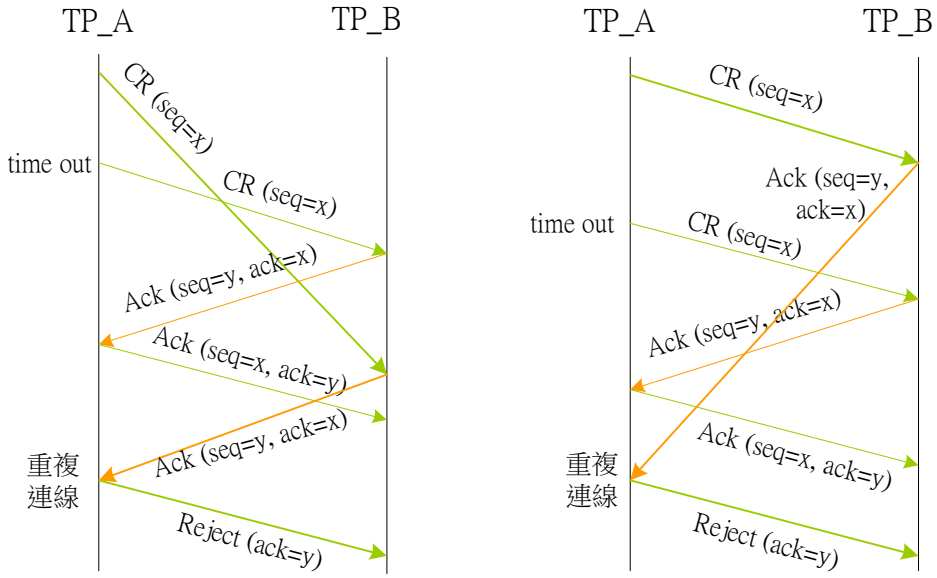


圖 5-10 三向式握手法（正常運作）

如同圖 5-9 要求連線訊號發生嚴重延遲，TP_A 計時器溢時後再發送一次要求連線訊號，造成 TP_B 重複接收連線訊號。如圖 5-11（a）中，當 TP_B 回應第二個連線時，TP_A 由回應序號（seq=y, ack=x）就可判斷出已經重複連線，而拒絕 TP_B 的第二條回應訊號。又如圖 5-11 (b) 所示，TP_B 的回應訊號在溢時後未送達 TP_A，TP_A 認為連線訊號已遺失而無法到達 TP_B；因此，TP_A 再發出同一個連線要求，但 TP_B 的第二次回應訊號比第一次早到 TP_A，TP_A 依照回應訊號的序號（seq=y, ack=x）判斷已重複連線，拒絕第二條連線回應。



(a) 要求連線訊號延遲　　(b) 回應確認訊號延遲

圖 5-11 三向式握手法偵測重複連線情況

5-6-3 釋放連線

『**釋放連線**』(Release Connection) 如同建立連線一樣，看起來非常容易，但實際運作可能會產生許多問題。首先我們如採用非對稱方式，只要有一方要求釋放連線，自己也就立即釋放連線，另一端應該無條件釋放連線。這可能發生下列問題：對方正在傳送資料而被強迫中斷，或對方傳送資料出去還未接收到回應確認訊號便被強迫中斷，它將不知道到底資料是否真正傳送到。就好像打電話一樣，並未詢問對方同意否便立即掛斷電話，也許對方並未將話講完。但如果採用對稱式釋放連線，要求釋放連線必須得到對方同意才能中斷連線。這也會出問題，當送出釋放要求時，如釋放連線遺失再次傳送釋放要求，或對方回應訊號一直到達不了，也有可能造成永遠等待。因此沒有百分之一百安全的方法，我們還是認為使用三向式握手連絡法最安全。

三向式握手法對於釋放連線的操作如圖 5-12 (a) 所示，TP_A 送出要求釋放連線 (Disconnect Request，DR) (DR(seq=x)) 並啟動計時器，等待對方回應。TP_B 收到對方要求終止連線，便立即處理未完成的工作。當工作站 B 準備好可以終止連線後，TP_B 送出釋放連線訊號 (DR(seq=y, ack=x)) 回應 TP_A 並啟動計時器，TP_A 收到 TP_B 的要求斷線訊號，便送終止連線確認訊號 (Ack(seq=x, ack=y)) 回應 TP_B，並立即釋放連線。TP_B 收到確認訊號後也釋放該連線。在所有訊號上都附帶有序號，可判斷是針對哪一個訊號回應。

萬一 TP_A 的回應訊號(Ack(seq=x, ack=y))遺失，當時 TP_A 已經釋放連線。TP_B 在計時器溢時後未收到確認訊號 (Ack)，也自行釋放連線，如圖 5-12 (b) 所示。圖 5-12 (c) 表示 TP_B 回應釋放連線 (DR(seq=y, ack=x)遺失，TP_A 在計時器溢時後未收到回應，便再重送釋放連線要求 (DR(seq=x))。重送訊號後 TP_A 有收到對方回應 (DR(seq=y, ack=x))，雙方正常釋放連線。但如果很不幸，TP_B 回應訊號(DR)和 TP_A 的重送釋放連線訊號(DR) 都遺失。這時候雙方都會在計時器溢時後，再次重送釋放訊號，也許會雙方一直都接收不到，而產生死結現象。因此我們必須設計，在重送幾次後未能收到對方回應，便自動自行釋放連線，如圖 5-12 (d) 所示。

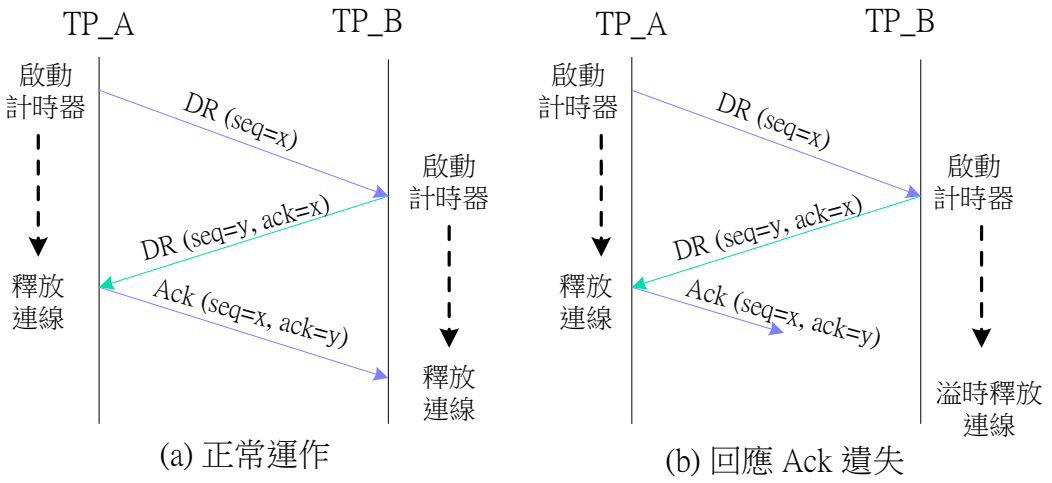


圖 5-12 (a) (b) 三向式握手法釋放連線運作狀況

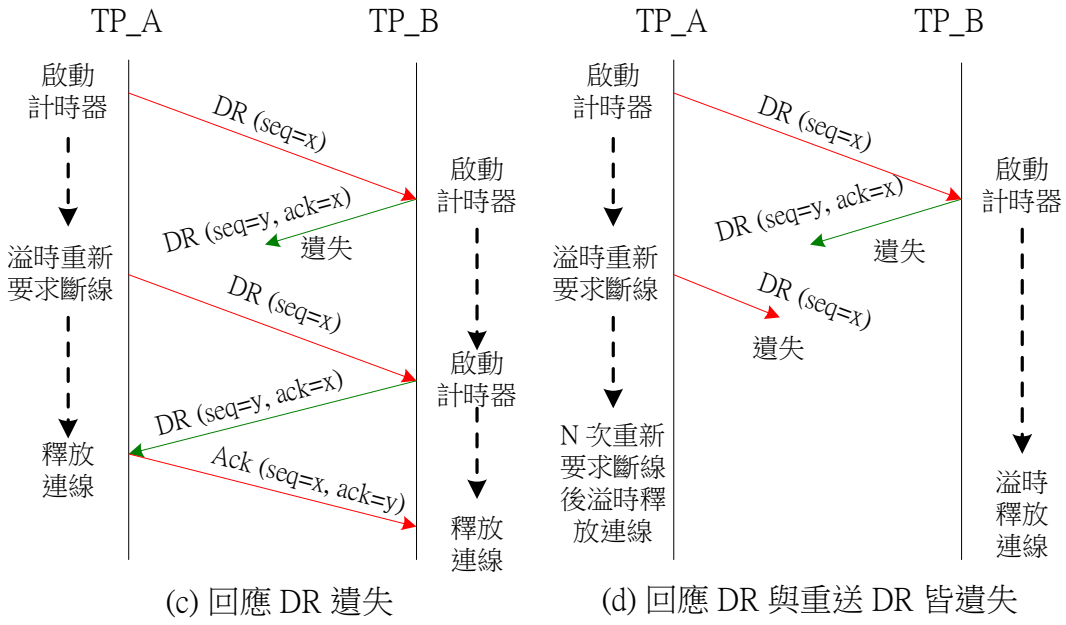


圖 5-12 (c) (d) 三向式握手法釋放連線運作狀況

5-6-4 流量控制

在連線的建立和釋放介紹之後，現在讓我們來探討連線中資料傳送的『**流量控制**』(Flow Control)。傳輸層和鏈路層相同都採用『滑動視窗法』(Sliding Windows)(請參考 3-4 節)，做傳送端和接收端之間資料流動的控制，以保證在不同的傳輸速率下，較慢的接收端不會被較快的傳送端資料所淹沒 (緩衝器超載)。但因網路環境不同，造成傳輸層和鏈路層使用不同的的流量管理方式，兩者最大的不同點就是緩衝器大小的設定。在滑動視窗法中，傳送和接收兩端都必須設有傳送和接收視窗，視窗的大小取決於緩衝器的多寡。在緩衝器的管理上有下列兩大問題：

(A) 緩衝器多寡問題

鏈路層想要傳送資料時，必須將訊框存放於緩衝器上。一直等到對方以搭順風車 (Piggyback) 方式傳回確認訊號 (N(R))，才可以由緩衝器上移除，否則必須準備重送該訊框。鏈路層的資料傳送直接交給實體層轉換成訊號傳輸，不會造成嚴重的延遲，因此緩衝器的多寡問題比較容易控制。但傳輸層是將資料封包交給下一層的子網路負責傳送，如果子網路採用電報傳輸方式 (如 IP)，資料封包在網路上延遲的時間將會很長。可能在傳輸層連續傳送多個資料封包後，還未收到對方搭順風車方式回應確認，因此資料封包留在緩衝器的時間便會加長，造成緩衝器的數量必須增加。我們以圖 5-13 簡單的情況來說明，TP_A 連續傳送 9 個封包給 TP_B，但 Data(2) 所經過的子網路發生嚴重的延遲。雖然其它封包都已到達，但 TP_B 必須等到 Data(2) 到達後，使 Data(2) 到 Data(7) 之間都已按照順序排列，才可將這些資料傳送給上一層，方可清除 Data(2) ~ Data(7) 的緩衝器空間。同樣的情形，TP_A 在接收到 Ack(2) 後才可以清除 Data(2) 以前的緩衝器空間，其餘 (Data(3) 以後) 可能還必須保留，預防對方要求重送。

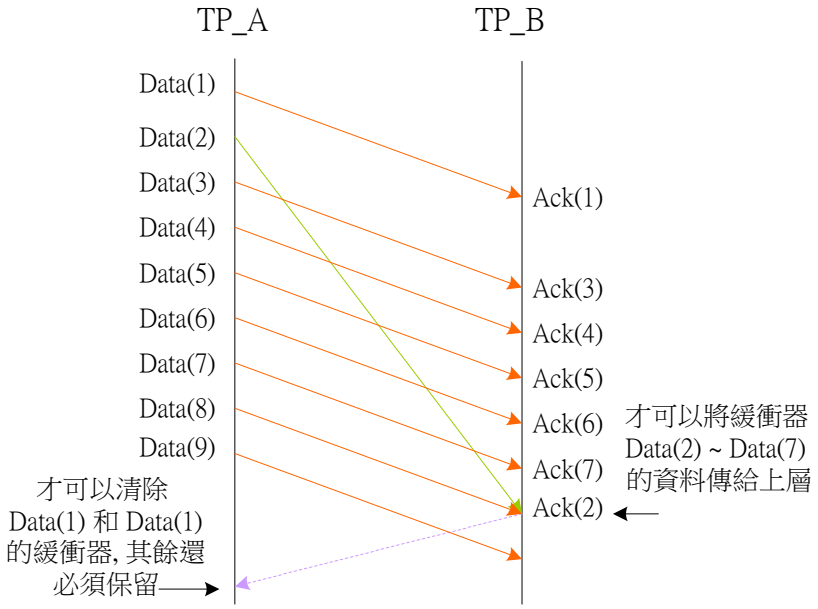


圖 5-13 緩衝器多寡的問題

又鏈路層同一時間的連線數目可能較少，但在傳輸層上每一個應用程式都必須有一條連線（或每一個傳輸埠口），每一條連線都是使用滑動視窗法傳送資料，所以針對每一條連線都必須設有傳送和接收緩衝器。如果預留過多的緩衝器，會造成記憶體浪費；預留太少，容易造成緩衝器溢流。

(B) 緩衝器長度問題

預留一個緩衝器到底要有多大的空間？在鏈路層上我們主要以網路的傳輸效率來考慮一個訊框應該多大比較適合，因此緩衝器的大小比較容易控制，一般都以訊框長度來決定緩衝器空間大小，所以對緩衝器的預留空間不至於產生太大的困擾。但在傳輸層資料封包的大小，主要決定於上一層應用軟體的需求，才能提高效益。例如在做檔案傳輸時，我們就需要較大的資料封包；但如果是遠端簽入（Remote Login）時，為了能夠即時交談，資料封包又希望小一點。當我們針對緩衝器預留太大的空間，對於較小的資料封包會造成嚴重的浪費；預留太小又容易發生容量不足的問題。針對以上兩個問題，我們也有兩個解決方案：

(B-1) 可變長度緩衝器配置法

在傳輸層上，每一個埠口都是銜接特殊應用軟體（伺服器），每一埠口對緩衝器大小的需求大約可以預估得到。這樣預估雖然不是非常準確但也差不多了。例如對方要求檔案傳送的連線，我們就對緩衝器的空間配置大一點；如果是遠端簽入，則配置小一點。每次遠端要求

連線，都依照連線特性配置緩衝器容量，不再配置固定一樣大小的空間，稱之為可變長度緩衝器配置法。

(B-2) 動態緩衝器配置法

雙方建立連線時，接收端並不知道傳送端會傳送多少資料，而以預估方式來配置空間多寡容易產生誤差。如果傳送端能夠告訴接收端有多少資料準備傳送；接收端也能告訴對方還有多少空間可以用來接收資料，這對緩衝器應該配置多寡的問題就可以解決了。我們還是利用滑動視窗法中的『搭順風車』(piggyback) 方法來完成，首先傳送端要求建立連線時，順便附帶要求多少緩衝器的訊息給接收端；接收端回應同意連線時，也順便附帶同意預備多少緩衝器給傳送端存放。例如 TCP 通訊協定的封包裡有一個 window 的欄位，如果要求連線端預估有 30 個封包準備傳送，就將該欄位設定為 30 (window = 30)，被連線端就可依該欄位的值來預備緩衝器的數量，雙方也可以利用回應訊號 (window = 20) 來協商可提供緩衝器的數量。

如果僅在連線建立時，雙方協商要使用多少緩衝器可能還是不能解決問題。例如傳送端要求 20 個緩衝器，也經過接收端同意。但連線後接收端僅能提供 10 個，在這種情況下傳送端是否可以傳送資料？或者在交談式的應用程式，到底要傳送多少資料是應用程式隨著使用者情況隨時改變，因此建立連線時所期望傳送的資料多寡也不能表示爾後連線時所要傳送的數目。如何解決這個問題，一般我們使用的方法是雙方隨時協調緩衝器數量。

傳送端所送出去的每一個資料封包內都用 window 欄位來指明還有多少封包準備傳送 (如，window = 20)；而接收端回應封包裡的 window 也指明預留多少緩衝器準備接收。如果傳送端發現接收端的 window 欄位較小或等於 0 時，則它就必須減緩傳送速度或暫停傳送。

5-6-5 錯誤偵出

資料傳輸方面大略可歸類四種錯誤發生的可能：封包損壞、封包遺失、封包重複接收和封包順序錯亂 (out of order)。後面三種錯誤都可以由滑動視窗法偵測出來，而其自動重複傳送 (ARQ) 可以使用退後-N ARQ (請參考 3-6-2 節) 或選擇重送 ARQ (請參考 3-6-3 節)

方法，一般通訊協定（如，TCP）大都採用選擇重送 ARQ。至於如何偵出封包損壞，也都使用檢查集（Check Sum）方法（請參考 3-5-2 節）。

習 題

1. 請簡述傳輸層的功能。

2. 請敘述單一層次和雙層次的傳輸模式的特點，並舉例說明其功能。

3. 請列舉至少 5 種應用環境其網路層和傳輸層都使用非連接方式，並說明其特點。

4. 何謂向下多工傳輸模式？請說明其特點。

5. 何謂向上多工傳輸模式？請說明其特點。

6. 何謂名稱伺服器（Name Server）？何謂目錄伺服器（Directory Server）？請說明其功能。

7. 請依自己目前網路環境（100MbaseTx），規劃及安裝一部傳輸速率為 500 Mbps 的 Web Server，主機伺服器安裝 Windows 2000。並說明欲達到該速率的瓶頸。