第七章 傳輸層通訊協定

7-1 傳輸層簡介

在未探討傳輸層的功能之前，我們先以圖 7-1 來說明，Internet 網路如何建構雙方的通訊連線。

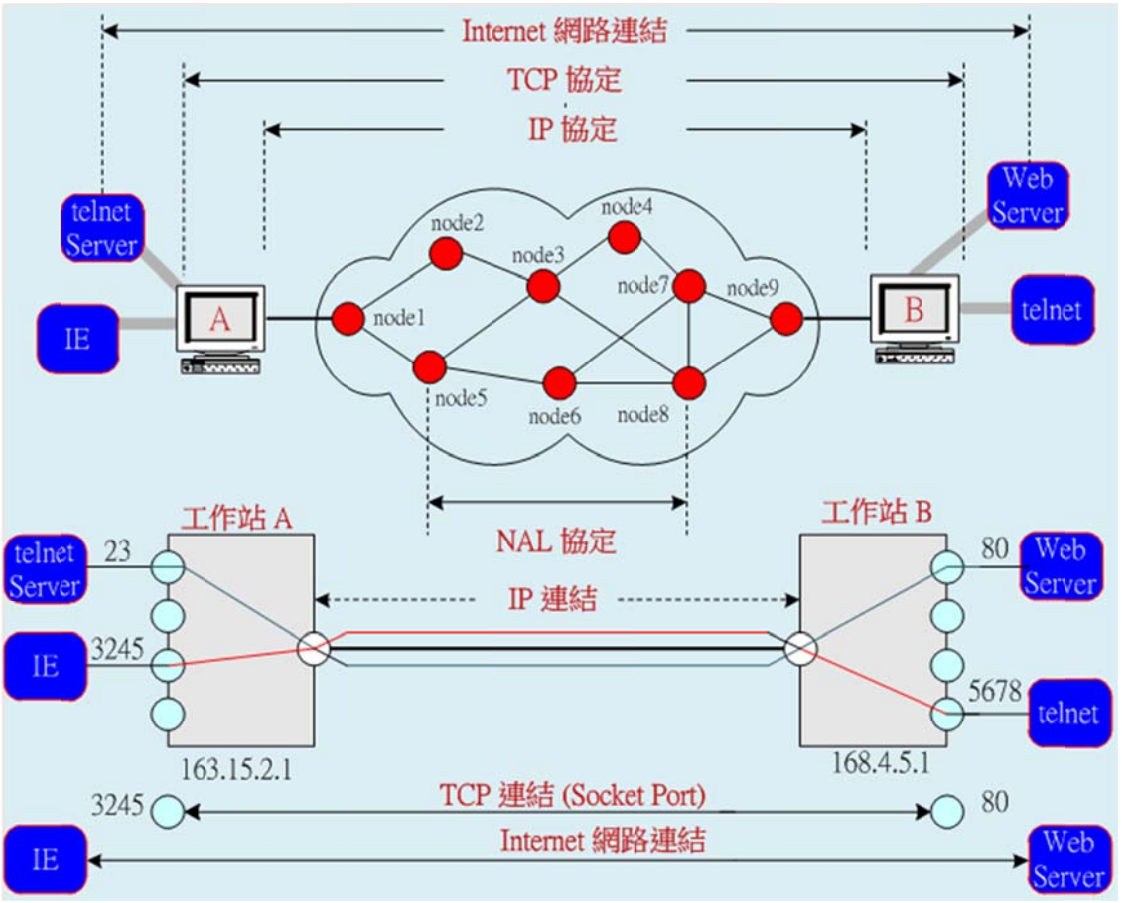

圖 7-1 傳輸層連結功能

首先觀察網路媒介存取層所扮演的角色，它是提供如何將一連串的資料組合成一個訊框，再將其發送到網路上，並且定義接收端如何來判斷一個框的開始和結束的基本技術。這些技術包含資料在傳輸媒介上的訊號方式，以及定義電腦之間在一個共用傳輸媒介上存取的共同協定（如，CSMA/CD 協定），基本上，是屬於同一網路內的通訊方式，因此稱之為**『節點對節點連接』（Node-to-Node Connection）**。

另外，網際層是負責在複雜的網路上尋找目的工作站的位址，然後建立雙方電腦之間的連線。在一條通訊連線之中也許會跨越多個不同型態的網路（譬如，Ethernet 網路），才能到達目的位址，並不在侷限於某一網路之內，因此稱之為『**Station-to-Station Connection**』（**工作站對工作站連線**）。

由上面的敘述我們可以知道，在 Internet 網路裡，到達網際層的連結功能時，通訊雙方的工作站已能夠連結到對方，但連結之後如何使雙方通訊，就是屬於傳輸層的功能。傳輸層提供工作站中的應用程式的通訊介面，稱之為『**主機對主機連線**』（**Host-to-Host Connection**）或稱為『**端點對端點連線**』（**End-to-End Connection**）。

最重要的是，傳輸層的通訊介面將與網路型態無關（Independent），應用程式透過該通訊介面，便可以和對方通訊，而不用理會所經過網路是何種型態，或是到底跨越了多少個網路銜接，因此稱之為『**應用通訊**』（**Application Communication**）。

如圖 7-1，工作站 A 和 B 兩者也許分處 Internet 網路的任何角落，經過網際層連接後，兩部工作站就如同在同一桌面上一樣。網際層的功能也和電話接通一樣，當兩具電話接通後，兩具電話不論在何地區，都像撥接到相鄰電話一樣。當然，工作站之間也許會經過許多區間網路的連接，工作站與工作站的連接就是透過這些區間網路之間的網際層共同來達成，但各區間網路之內連接就屬於網際存取層的功能。而工作站上也許會有許多應用程式同時在執行，傳輸層則提供通訊介面讓這些應用程式連接，例如，工作站 A（163.15.2.1）的 IE 程式連接到第 3245 傳輸埠口，該埠口連接到工作站 B（168.4.5.1）的第 80 傳輸埠口，而 80 埠口試銜接到 Web 伺服器上。由此可見 163.15.2.1:3245 與 168.4.5.1:80 之間連線為『**端點對端點連線**』，而 IE 與 Web Server 之間為『**應用對應用連線**』（**Application-to-Application Connection**）。同樣的 163.15.2.1:23（工作站 A）與 168.4.5.1:5678（工作站 B）也是一條傳輸層連線，而 Telnet 和 Telnet Server 之間則是屬於應用層連線（第十三章介紹）。

7-1-1 傳輸層通訊協定

基本上，傳輸層提供應用程式的連接介面，對一般程式設計師而言（開發電子商務軟體）就顯得特別重要，而網際層和媒介存取層之建設則是比較屬於網路工程師的工作。因此傳輸層介於雙方的連線重點，它的連接功能對整個網路的通訊效益，就有絕對性的影響。傳輸層提供終端對終端連線，相當於 OSI 參考模式的第四層。在 Internet 網路上所訴求的遠距離的連線，因此在網際層方面為了能符合各種不同環境的需求，而採用非連接方式，也就是『**電報傳輸**』（**Datagram**），通訊

過程中也許會出現錯誤或遺失某些資料，這時就必須仰賴傳輸層去偵測及處理資料保全的工作。譬如，網際層也許會有遺失某些封包而不知曉，傳輸層必須有能力去偵測出，並要求對方重送；或者網際層通訊當中，可能會發生資料錯誤，這也必須由傳輸層檢測出來。因此，在 Internet 網路上必須有一個可靠的通訊協定來補救網際層可能出現的錯誤，一般都採用**『傳輸控制協定』（Transmission Control Protocol, TCP）**，也就有所謂**『TCP/IP』**的組合，另外 Internet 網路也提供非連接方式的連接，兩個主要通訊軟體說明如下：

- **『傳輸控制協定』（Transmission Control Protocol, TCP）：**提供連接導向（Connection -oriented）程序（Process）之間的可靠性（Reliable）連線服務。TCP 提供標準通訊介面，可讓異質電腦（Hetergeneous）之間連線。
- **『使用者電報傳輸協定』（User Datagram Protocol, UDP）：**提供非連接服務（Connectionless）的使用者（或程序）之間連線。類似 TCP 服務但為不可靠性（Unreliable）連線。

7-1-2 傳輸層訊框包裝

傳輸層的 TCP/UDP 封包經過 IP 封包包裝，又再經過 Ethernet 訊框包裝後，才發送到網路上，其包裝結構如圖 7-2 所示。

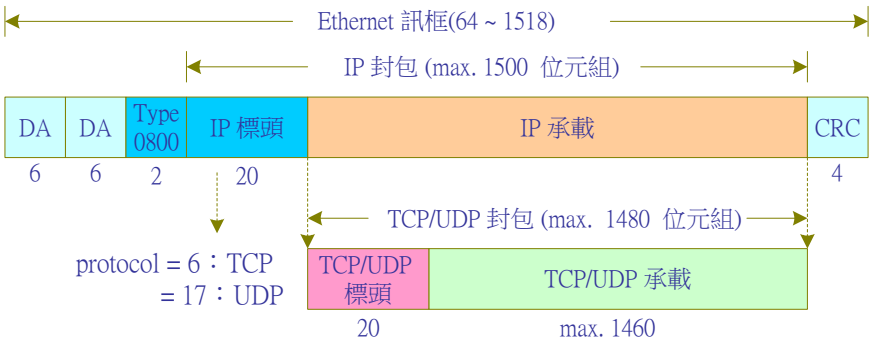

圖 7-2 傳輸層訊框包裝

在 Ethernet 框標頭中，型態（Type）為 0x0800 表示該訊框是承載 IP 封包，又在 IP 封包標頭中，如 Protocol 欄位為 6 表示本封包是承載 TCP 訊息；如 Protocol = 17 則表示承載 UDP 訊息。一般傳送 TCP/IP 訊息，IP 標頭大多不會攜帶特殊訊息（如來源路徑選擇），也沒有選項欄位（Option），因此 IP 標頭佔用 20 位元組長度。TCP 標頭如沒有特殊訊息，也是佔用 20 位元組長度。而一般 UDP 標頭也都是佔用 20 位元組。

由圖 7-2 可以看出，在 Ethernet 網路環境下，TCP/UDP 所能承載的訊息長度最高為 1460 位元組。一般應用上，如果資料長度大於 1460 位元組，則會選用 TCP 協定傳送，但如果小於 1460 位元組，則儘量使用 UDP 協定傳送，以提高傳輸效率。在 Internet 環境下，TCP 協定大多使用於大量資料的傳輸協定上，例如，FTP 或 HTTP 協定。另一方面，UDP 協定都使用在較少資料量的傳輸協定，例如，DNS 或 SNMP 協定。

7-2 TCP 協定

『傳輸控制協定』(Transmission Control Protocol, TCP) 和 IP 兩者似乎是連結在一起的同一名稱 (TCP/IP)，兩者的功能確實是相輔相成。IP 的功能是無論兩部工作站在無遠弗屆的任一個角落，都能將它們連結在一起。TCP 提供網路的服務接點讓應用程式使用，也就是說，提供端點對端點 (End-to-End) 的連線。主機電腦上有多個應用程式都必須透過網路提供或使用網路服務，TCP 就提供多點服務的連線 (虛擬鏈路的多工功能) 讓各種應用程式可同時連結到網路上。TCP 和 IP 的關係宛如電話系統中的電話號碼和分機號碼。當我們撥接電話時，將依照電話號碼的位址在廣泛的電話大海之中找到對方，並和其連接完成 (IP 功能，各地區的交換機就如網路閘門)。這並不能表示我們已連絡上受話的對方，但最起碼我們也連線到對方的電話機上 (IP 已連結到主機上)。欲找到受話的人也許可用人工呼叫，但也可以再撥分機號碼 (TCP 的埠口號碼)，這表示在主機號碼上再加入分機號碼來表示通訊的個人 (TCP 的點對點連線)。人與人之間的對話就像網路上應用程式之間的通訊。

TCP 和 IP 另一個相輔相成的功能是 IP 提供非連接的不可靠傳輸，對於有關可靠傳輸的處理程序就必須仰賴 TCP 來完成。換言之，IP 傳送當中，也許會發生封包損壞、封包遺失、封包重複或次序錯亂等現象，這些情況都必須由 TCP 來負責檢測出，並要求對方重送、重整封包順序等工作。因此，TCP 必須提供連接導向的連線，才能使整個網路通訊達到可靠性的傳輸。本節將依此介紹 TCP 之特性。

7-2-1 TCP 封包格式

圖 7-3 為 TCP 的封包格式，各欄位功能如下：

- **來源埠口 (Source Port)：** 來源之 TCP 埠口。
- **目的地埠口 (Destination Port)：** 目的地之 TCP 埠口。

- **順序編號（Sequence Number）：**該封包的順序編號。
- **確認號碼（Acknowledge Number）：**回應封包的確認號碼，也是期望傳送端下次發送封包的序號，其表示該確認號碼以前的封包都以正常接收。
- **資料偏移量（Data Offset）：**因為 TCP 的 Option 欄位長度並非固定，Data Offset 用來表示傳輸資料（Data）是在整個封包之區段起始位址。
- **位元碼（Code bits）：**（6 位元）（URG, ACK, PSH, TST, SYN, FIN）此欄位作控制訊息傳遞之用。而且目前有關 TCP/IP 網路上的特殊處理工作（如防火牆等等）都是利用這些控制碼來運作。其中：

  (1) URG（Urgent）：表示該封包為緊急資料，並使 Urgent Point 欄位有效。

  (2) ACK（Acknowledge）：本封包有回應確認功能，其確認 Acknowledge Number 欄位中所指定的順序號碼。

  (3) PSH（Push）：請求對方立即傳送 Send Buffer 中的封包。

  (4) RST（Reset）：要求對方立即結束連線（強迫性），且發送者已斷線。

  (5) SYN（Synchronous）：通知對方要求建立連線（TCP 連線）。

  (6) FIN（Finish）：通知對方，資料已傳輸完畢，是否同意斷線。發送者還在連線中等待對方回應。

- **視窗（Window）：**此欄位是用來控制封包流量，告訴對方目前本身還有多少緩衝器（Receive Buffer）可以接收封包（滑動視窗法之特性）。如果 Window = 0 表示緩衝器已滿暫停傳送資料。Window 大小的單位是以位元組表示（Byte）。
- **檢查集（Checksum）：**此欄位為 16 bits 長的檢查碼，接收方可依此 Checksum 來確定所收封包（資料極表頭）是否正確。
- **緊急指標（Urgent Point）：**當 URG = 1 時，其代表緊急資料是在資料區的什麼位址。
- **任選欄（Option）：**目前此欄位只應用於表示接收端能夠接收最大資料區段的大小。如果不使用此欄位，則可以使用任意的資料區段大小。

- **填補欄位（Padding）：**將 Option 欄位補足 32 位元的整數倍。

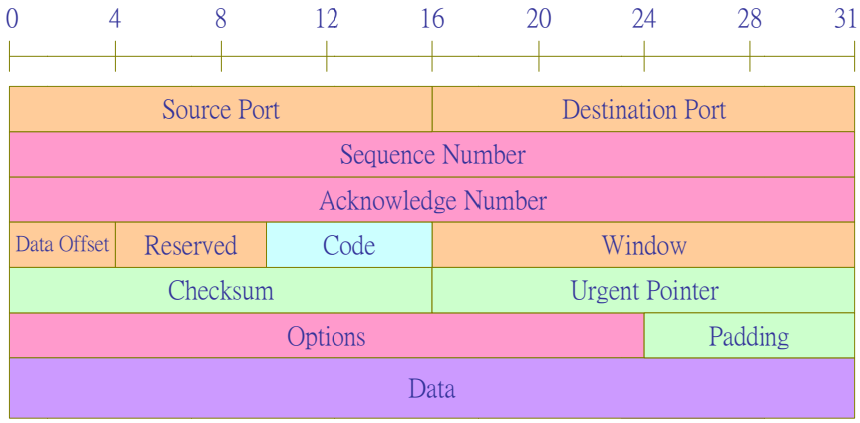

圖 7-3 TCP 封包格式

圖 7-4 為執行 telnet 163.15.2.62 命令時（利用 Microsoft 網路監視器，附錄 A），所擷取的封包視窗，由視窗中各欄位的說明，而將它填入封包欄位上，如圖 7-5 所示，我們可以比較圖 7-3 和 7-5，便可了解 TCP 封包包裝的型態。又 Code（或 Flags）欄位為 0x02，此欄位是 6 個位元（000010），第二個位元為 1（SYN = 1），表示要求對方同意連接 TCP 連線的意思。

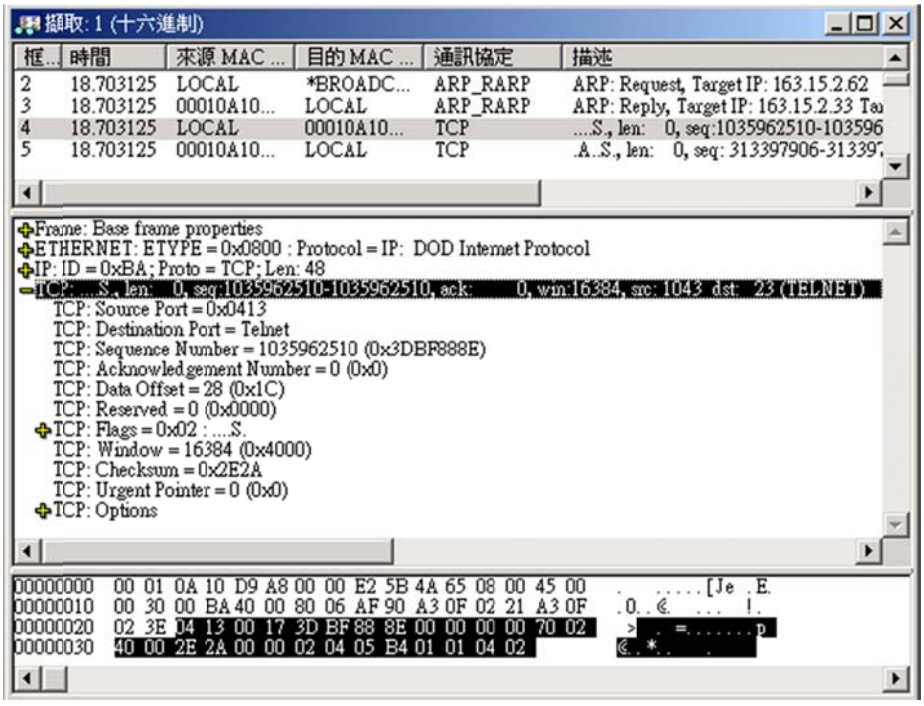

圖 7-4 TCP 標頭擷取視窗

1043 (source Port)			23 (destination port)
1035962510 (sequence number)			
0 (acknowledge number)			
28 (DO)	0	0x02 (code)	16384 (window)
0x2E2A (checksum)			0 (urgent point)
0x020405B4 (options)			
01010402 (options)			

圖 7-5 TCP 封包標頭擷取範例

7-2-2 TCP 傳輸埠口

『TCP 埠口』(TCP Port) 是 TCP 連線中虛擬鏈路的邏輯編號，讓應用層應用軟體可透過埠口位置銜接上網路。TCP 埠口是利用 16 位元表示，則理論上可以提供 65536 (= 216) 個連接埠。TCP 埠口和 IP 之連線是多工處理的關係 (如圖 7-6 所示)，至於虛擬鏈路之間如何選擇處理次序，可依照作業系統的程序排程 (Scheduling) 來管理，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，或即時系統 (Real-time system) 則以優先權較高的連線優先處理。對於每一個連接埠口都給予唯一的編號 (如 2020)，如圖 7-6 裡程式 A 連接之位址為 163.15.2.1:2020，表示在 IP 位址為 163.15.2.1(宛如電話號碼) 的主機之第 2020 連接埠口 (宛如分機號碼)。程式 1 的連結位址為 163.15.4.5:8080，程式設計者或網路使用者只要記住這個位址，就可以連絡到程式 1，而不必知道 (也很難知道) 連線之間所經過何種網路，或它在全世界的哪一個角落。

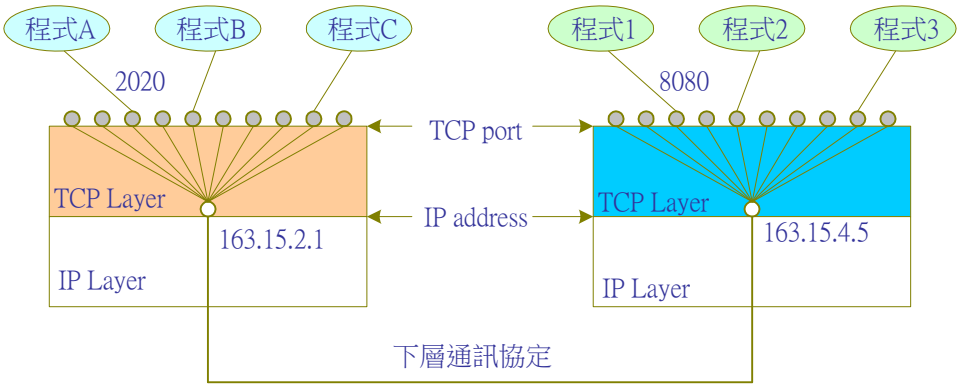

圖 7-6 TCP 與 IP 之連線關係

TCP port 的編號範圍為 0 ~ 65535 (216)，我們採用兩種配置方式：**『固定配置埠』(Static Allocated Port)** 和 **『動態配置埠』(Dynamically Allocated Port)**。將 0 ~ 1023 之位置配置給固定的應用程式 (或常用伺服器) 使用，使一般用戶連接時不必特別去記憶連接埠的位置，稱之為固定

配置埠。例如，當使用者連接到某一主機（163.15.2.1）上的 Web Server（80），只要連接上 http://163.15.2.1，便自動連結到 Web Server，而不用輸入 http://163.15.2.1:80。我們節錄一些較常用的著名埠（Well-known）位置如表 7-1。動態配置埠是當應用程式需要連線時，才由系統動態地配置出來，其埠號碼範圍為 1024 ~ 65535，但某些位置如果被特別指定連線使用（如 8080），就不可以再重複配置。

表 7-1 著名服務的埠號碼（節錄）

埠號碼	服務名稱	傳輸協定	應用服務說明
0	保留		
1	tcpmux	TCP	TCP port 服務多處理
7	Echo	TCP/UDP	回應測試
11	systat	TCP	系統狀態顯示
15	netstat	TCP	網路狀態顯示
20	ftp-data	TCP	FTP 資料傳輸埠
21	ftp	TCP	FTP 控制埠
23	telnet	TCP	Telnet 遠端登入埠
25	smtp	TCP	Simple Mail Transfer Protocol
37	time	TCP/UDP	Time Server
42	name	UDP	Name Server
43	whois	TCP	是誰、nickname（別名）
53	domain	TCP/UDP	Domain Name Server（DNS）
79	finger	TCP	尋找使用者
80	http	TCP	Web Server
109	pop-2	TCP	Post Office Protocol 2
110	pop-3	TCP	Post Office Protocol 3
119	nntp	TCP	Network News Transfer Protocol
123	ntp	TCP	Network Time Protocol

7-3 TCP 連線管理

TCP 是連接導向的傳輸方式，對於連線管理也顯得特別重要，而所發生的問題也較大，這必須經過特殊處理才不至於發生錯誤現象。譬如，當 TCP 建立連線時，它的連線訊號必須經由 IP 封包傳送，而 IP 封包也許會經由多個實體網路跨接，並不能保證連接訊號是否可安全到達目的地，甚至傳送訊號會在各子網路上發生嚴重的延遲，此時，TCP 傳送端也許會誤判該訊號遺失，再發出重複的要求連線訊號。這種情況如圖 7-7 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。

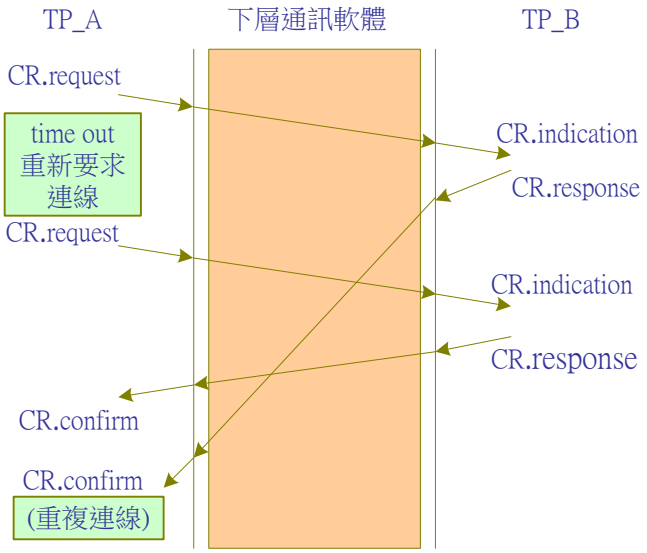

圖 7-7 重複連線錯誤

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 7-8 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連線已連接成功。

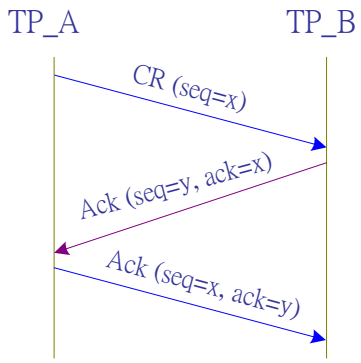

圖 7-8 三向握手式連絡法之訊號方式

因此，TCP 採用三向握手式連絡法 (Three-way handshake) 來實現連線處理，其中會用到封包內二個序號：Sequence Number (seq) 及 Acknowledge Number (ack)，以及 Code (或稱 Flags) 欄位中四個旗標：ACK、SYN、FIN 和 RST。以下分別介紹各種連線情況的處理方式 (注：旗標都用大寫字母表示，而控制欄位都用小寫表示)：

7-3-1 TCP 建立連線

圖 7-9 為三向握手式連絡法的連線建立圖，所謂三向握手式是表示有三個訊號來建立連線：(1) SYN (A→B) 表示 A 向 B 要求連線；(2) SYN&ACK (B→A) 為 B 回應給 A，表示同意或不同意連線要求；(3) ACK (A→B) 表示 A 確認收到 B 的回應。另外， seq 表示要求連線的封包號碼；ack = seq +1 表示確認 seq 封包，並要求傳遞下一個封包序號 (滑動視窗法確認方式)。

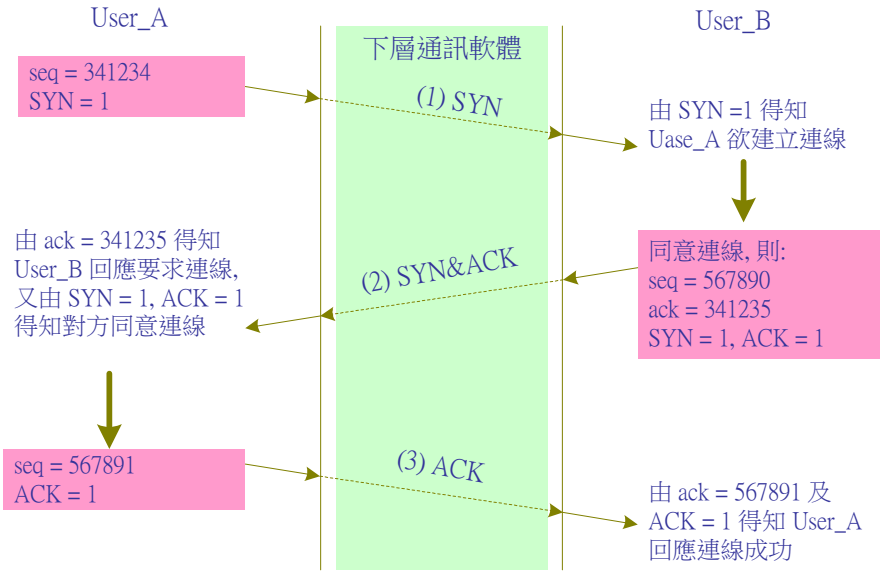

圖 7-9 TCP 建立連線運作程序

7-3-2 TCP 資料傳送

TCP 連線中的資料流量控制是採用滑動視窗法（Sliding Window），而錯誤偵測技巧是採用檢查集（Checksum）方法。以下介紹資料的傳輸方式，至於滑動視窗法，我們在下一節（7-4 節）會專門介紹。TCP 封包中 Sequence Number（seq） 表示該封包的順序號碼、Acknowledge Number（ack）為期望對方發送的封包順序號碼，也確認該序號以前的封包都已正常接收。傳送開始的 Sequence Number 是由亂數（Random Number）產生。圖 7-10 為 User_A 傳送資料給 User_B 的資料流動控制程序。其中 data_length 表示每次封包內所傳送的資料位元組數量。我們要特別強調封包序號的計算是以資料位元組（byte）為單位，而不是以傳送次數為單位，因此，回應確認序號是 ack = seq + data_length。由圖中可以看出，傳送資料的控制訊號是 PSH 旗標和所傳遞資料的順序號碼 seq，而回應確認訊號是 ACK 旗標和 ack 欄位的順序號碼。PSH 旗標表示希望接收端儘速將資料傳給上層通訊協定，並回應確認訊號給傳送端。

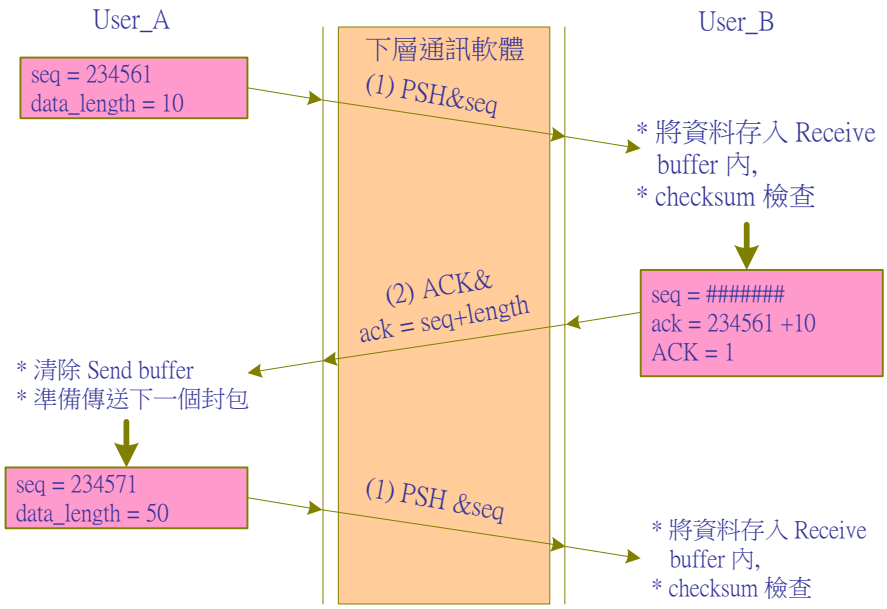

圖 7-10 TCP 資料傳送運作方式

7-3-3 TCP 連線終止

TCP 連線是連接導向且雙向傳輸模式，對於連線終止的處理比較麻煩。一端傳輸完畢要求結束連線，但另一端可能還有要傳送資料，因此雙方必須協議好才可以終止連線。圖 7-11 為連線終止的運作程序，首先 User_A 傳送準備終止連線（FIN = 1）（(1) FIN & seq）訊號給 User_B。當

User_B 收到要求終止連線訊號後，立即回應接收到該訊號((2) ACK & ack = seq + 1)，此時 User_B 必須詢問上層通訊軟體是否還有資料要繼續通訊，如上層回應同意終止連線，則 User_B 再告訴 User_A 確實可以終止連線 ((3) FIN & seq & ack)。User_A 再次收到 User_B 要求終止連線後，並由 ack 序號得知此訊號是上一次要求對方的回應，便知道上一次要求終止已得到對方同意，因此，再發送同意終止連線訊號給 Uase_B ((4) ACK & ack = seq + 1)，同時也釋放該連線。User_B 收到該同意終止訊號，並由 ack 欄位和 ACK = 1，得知 A 已釋放連線，並且自己也釋放連線。為了防止封包遺失或溢時傳送，雙方請求或回應封包都以封包序號 (seq 及 ack) 作為確認基準。

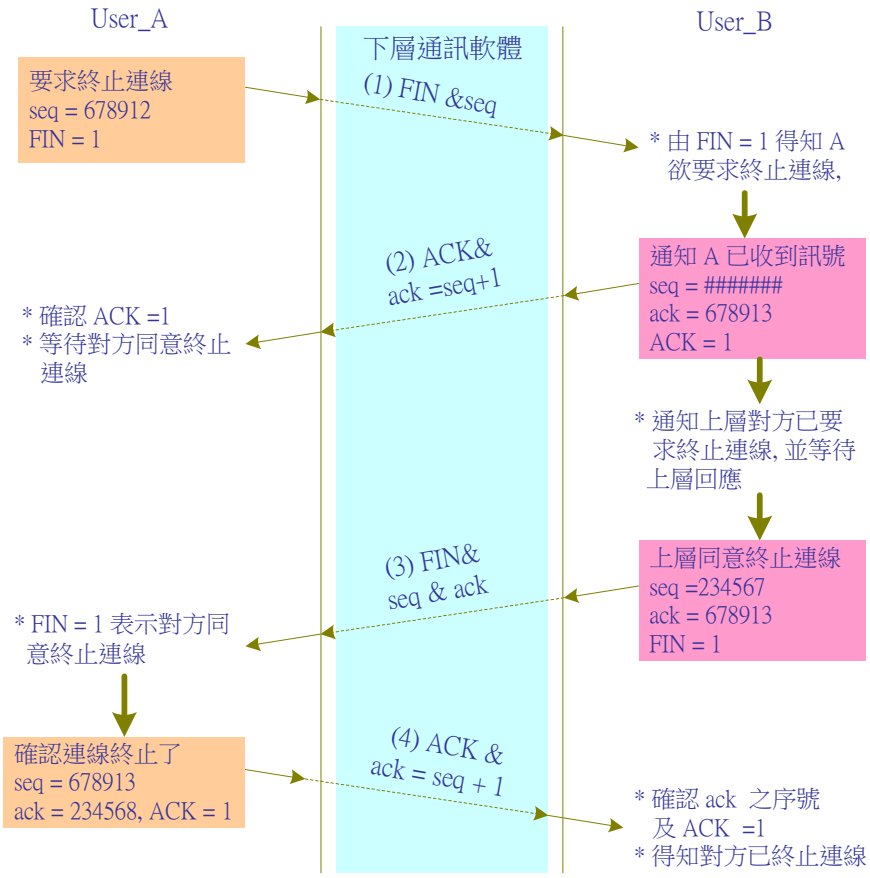

圖 7-11 TCP 終止連線運作程序

7-3-4 TCP 連線重新啟動

如果連線雙方有一端不正常中斷，而使另一方無法傳送給對方時，則可發出重置訊號，要求對方重新連線。如圖 7-12 所示，TCP_B 已中斷 (中斷原因不詳)，而 TCP_A 端不知道仍舊繼續傳送資料，該封包到達 TCP_B 的下層通訊軟體 (IP 層) 會因為到達不了該埠口，而回應 ICMP Destination Unreachable 之 Code = 3 (無法到達連接埠) 給 TCP_A。此時 TCP_A 會送出連線重新啟動訊息 (SYN & RST)，TCP_B 便可回應 (SYN & ACK) 重新啟動連線 (以原來之傳輸埠口號碼)。也有可能主機 B (TCP_B) 已經關機，而 TCP_A 接收不到回應訊號，便自行中斷連線。

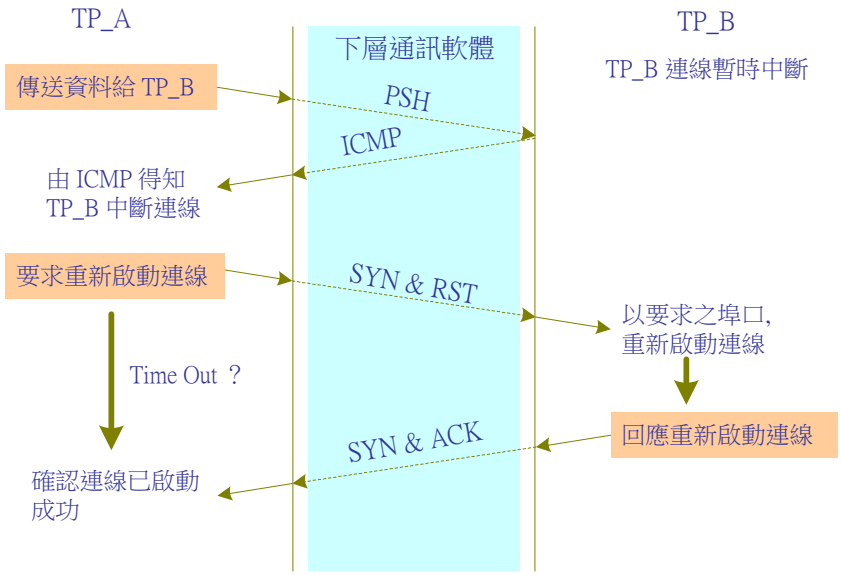

圖 7-12 連線重新啟動

7-3-5 TCP 連線識別

在 Internet 網路上針對任何一個傳輸埠口，一般都允許同時連接多個傳輸線，如圖 7-13 中，一個 Web Server 也許會有多個客戶端（IE）同時要求傳送資料，但 Web Server 如何來分辨每一傳輸線？其實 TCP 識別連線非常簡單，它是利用 IP 位址 ＋TCP 傳輸埠口來分辨，每一條連線識別如下：

【IP 位址:TCP 埠口】←→【IP 位址:TCP 埠口】

如圖 7-13 (b) 中就有四條連線分別為：

【148.34.12.6:80】←→【165.4.3.45:2356】

【148.34.12.6:80】←→【166.45.21.3:4521】

【148.34.12.6:80】←→【136.7.34.2:2678】

【148.34.12.6:80】←→【136.7.34.2:2478】

在同一部主機上（IP 位址）不可能會有同一傳輸埠口（TCP 埠口）連結到不同的應用程式上的情形發生，因此，上述的連線辨識也不會有重複現象發生。一般 Web Server 的傳輸埠口都會放置在 80 埠口，又在客戶端啟動 TCP 連線時，主機會找出一個空閒的傳輸埠口來使用，用戶取用埠口也都會限制在 1024 埠口以後的位址。如同一主機執行相同的應用程式（如 IE）兩次以上，它所取用的傳輸埠口也不可能一樣，因此用 IP 位址 ＋TCP 埠口來表示連線位址，也不會有重複

現象發生。如圖 7-13 中，136.7.34.2 主機執行兩次 IE 程式，並連結到同一部 Web Server，還是可以區分出來各自的連線。

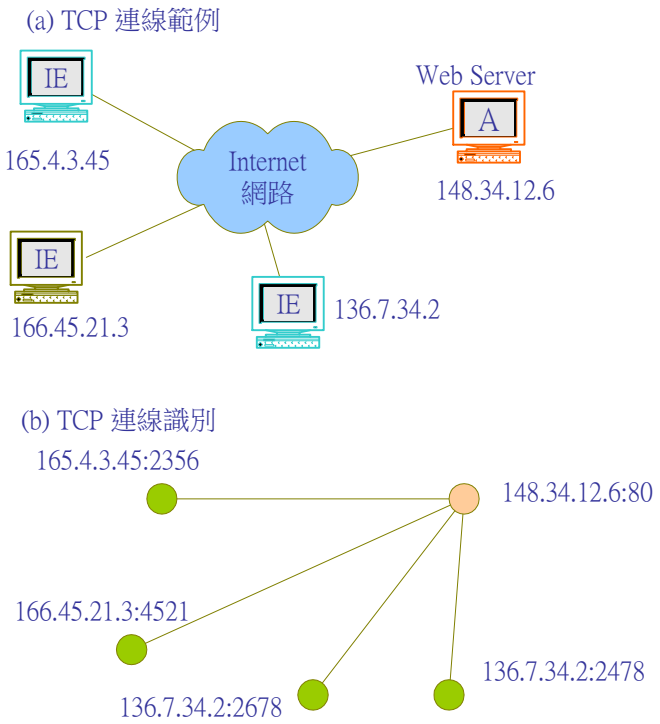

圖 7-13 TCP 連線範例與識別

7-4 TCP 流量控制

TCP 主要的功能是希望在不可靠的網路裡，建構一個可靠的傳輸環境。依照我們的了解 Internet 網路為了能連結不同網路型態，又希望跨接各偏遠地區的網路，而採用在網路層上以變異性較高的 IP 協定。IP 協定最主要的特點是連結性較高，但相對應的可靠性也較低，當傳送端發送封包後，也許此封包會在傳送中遺失，或是被傳送端重複傳送，或連續的封包到達目的地後，次序已發生錯亂，甚至封包在傳送中已被外來訊號干擾，而發生資料錯誤，等等這些問題，基本上 IP 協定並沒有能力去檢定這些可能發生的錯誤，這都必須由 TCP 協定來偵測出，並從事補救的措施，由此可以看出 TCP 流量控制就顯得非常重要。

何謂『**流量控制**』(**Flow Control**)？是在已建立完成的連線上從事資料傳輸的控制方法，並且以何種方法可將傳輸連線的使用率提昇到最高？流量控制除了管理資料在通訊連線上傳輸的順序外，還牽涉到傳送端和接收端緩衝器多寡的管理問題。當電腦欲傳送資料時，將資料以每次傳送封包大小，依序填入『**傳送緩衝器**』(**Send Buffer**)內。當傳送端將資料封包傳送出去後，可能必須經過一段時間後才能到達接收端，接收端是否正常收到資料？如何回應給傳送端？如果發生問題

怎樣重送？傳送端必須確認該訊號已正常傳送完畢，才可將資料封包由傳送緩衝器上刪除，因此在大量連續傳送資料中傳送緩衝器就必須很大。在接收端也必須有一個叫『**接收緩衝器**』(**Received Buffer**)，接收到資料後判斷資料正常便填入緩衝器內，但因為通訊連線也許很長，每一筆資料封包到達的時間不一定相同，到達的資料封包不一定按照原來封包順序，也可能在一串的封包內其中有幾個封包發生錯誤。因此有幾個封包是重新傳送？有幾個封包是重複傳送？接收緩衝器必須將接收到的封包按序號排列，才可傳送給上層通訊軟體，所以接收緩衝器有可能要很大。當然由技術層面來講，我們希望流量控制的技術儘可能減少緩衝器的需求。可是不論傳送緩衝器或接收緩衝器都不可能無限的擴充，如何重複使用緩衝器以減低緩衝器需求也是流量控制的重要項目之一。

7-4-1 停止與等待法

我們以圖 7-14 來說明最簡單的流量控制法，此方法又稱為『**停止與等待**』(**Stop-and-Wait**)。其表示傳送端（TP_A）將資料送出後必須等待接收端（TP_B）回應，再決定是否繼續傳送下一筆資料。如果回應接收正常（Ack）便傳送下一筆資料；如果溢時（Time out）未收到回應或收到回應不確認訊息（Nak），則需重新傳送該筆資料。

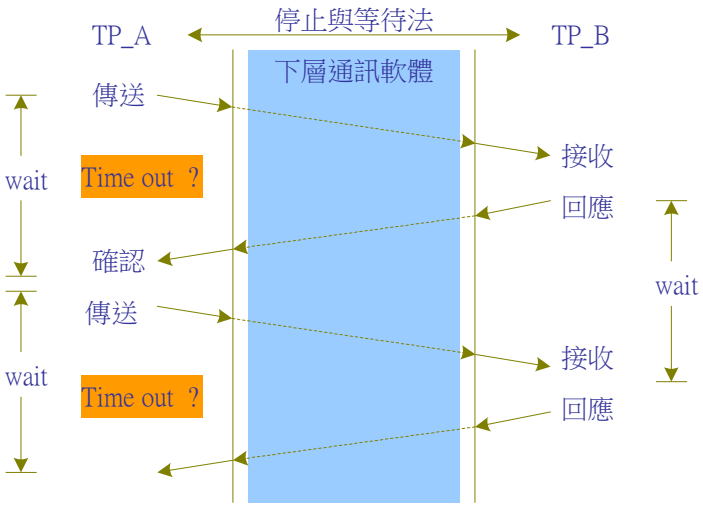

圖 7-14 停止與等待流量控制

由圖 7-14 中我們可以發現停止與等待法，通訊中兩端花費太多的等待時間 (大部分時間都在等待對方回應)，但其優勢在於處理方法簡單，所以對近距離（回應時間較短）或要求快速反應的設備大部分採用此方法。而且由於傳送和接收雙方只要一個緩衝器便可，一般適用於主機電腦內，或電腦機房內各處理設備之間資料的傳輸，對於遠距離 Internet 網路的傳輸便不適合。以下我們接介紹較適合於網路上使用的滑動視窗法。

7-4-2 滑動視窗法

『滑動視窗法』(Sliding Window) 的功能是當建立連線後，我們希望能充分利用這條連線來傳輸資料 (連接導向服務)，如果採用停止與等待方法，通訊連線大部分時間都是空閒著，傳輸效率太差。滑動視窗法可連續傳送多筆資料，接收端只要回應目前已正確接收到第幾筆資料，一次確認多筆資料，網路上可連續傳送多筆資料，提高網路的使用率。

(A) 運作原理

TCP 連線是屬於全雙工傳輸方式，也就是說，雙方建立 TCP 連線後，這條連線允許雙方同時傳送資料。我們利用圖 7-15 來說明滑動視窗法如何來控制雙向傳輸的基本原理。首先假設上層通訊軟體有一筆資料 (1Mbytes) 希望借由 TCP_A 傳送給 TCP_B 的上層通訊軟體，此時 TCP_A 會依照**『傳送緩衝器』(Send Buffer)** 的空間大小 (假設 300 Kbyte) 填入該筆資料，再依序由緩衝器上取出每一傳送封包長度(一般皆為 1460 Bytes)的資料，組裝後傳送給下層通訊軟體(IP 層)，並在每一封包上給予兩個序號：(如圖 7-3 中 TCP 封包的 Sequence Number 與 Acknowledge Number 欄位)

- **seq (Sequence Number) :** 表示目前傳送封包的序號。
- **ack (acknowledge Number) :** 表示期望對方下次傳送給我之封包序號，也就是說下次對方傳給我的封包的 seq。也表示該序號以前的資料都已正常接收。

TCP 封包順序號碼的編碼如下：當 TCP 準備傳送資料時，便由亂數中取一個 32 位元數字當作起始的順序號碼 (Sequence Number)(或 seq)，而確認序號 (Acknowledge Number)(或 ack) 不用填入任何值。接收端收到該封包後，計算封包內資料的位元組長度 (Length) (或 len)，再以收到的順序號碼 (seq) 加上資料長度 (len) 後 (以 Bytes 為單位)，填入確認序號欄位 (ack = seq + len)，並回應給傳送端。傳送端由對方回應的確認號碼得知是否成功傳送封包。

我們用圖 7-15 來說明滑動視窗的運作程序，假設每個封包長度為 10 個位元組(Bytes)，TP_A 連續傳送 3 個封包給 TP_B，而順序號碼為 seq =20、30、40。又每一封包上皆是 ack = 20，也表示之前 TCP_B 送給 TCP_A 的順序號碼 20 以前的資料都已正常接收，期望下一次 TCP_B 送來的順序序號為 20。

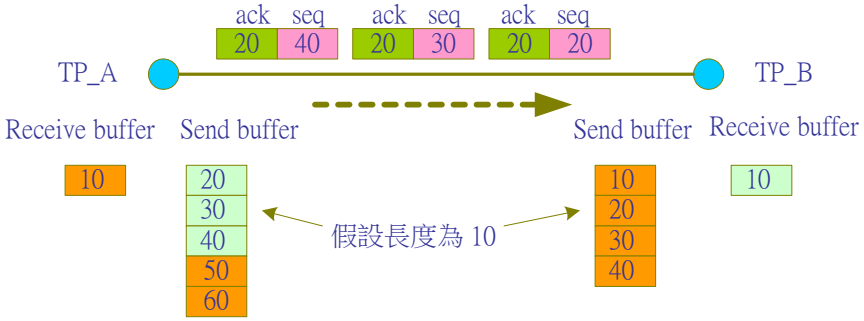

圖 7-15 滑動視窗法基本原理之一

如果 TCP_B 接收到上列資料（圖 7-15）後處理如下：(1) 由 ack = 20，得知先前所送順序號碼 20 以前的資料對方都正常接收，並清除該資料所佔用的緩衝器空間；(2) 又再利用檢查集（Checksum）錯誤檢出法檢查封包資料（seq = 20、30、40）有沒有損壞，如正常便依序填入接收緩衝器（Receive Buffer）內，再依照最後序號 40 加上該筆資料長度（40 + 10），而以 ack = 50，表示前面的資料都已接收正常；(3) 有可能此時 TCP_B 也有資料要傳送給 TCP_A，因此，就將 ack（50） 訊號附加在傳送封包上。如圖 7-16，TCP_B 連續傳送兩個封包給 TCP_A，其中封包訊號為（seq = 20、ack = 50）與（seq = 30、ack = 50）。TCP_A 接收到 ack = 50，便知道序號 50 以前的資料都傳送正常，便清除它們的緩衝器，並將這兩個封包存入接收緩衝器上。此時 TP_A 的傳送緩衝器就空閒起來，便可再填入欲傳送的資料（50 ~ 99），並等待下次傳送。

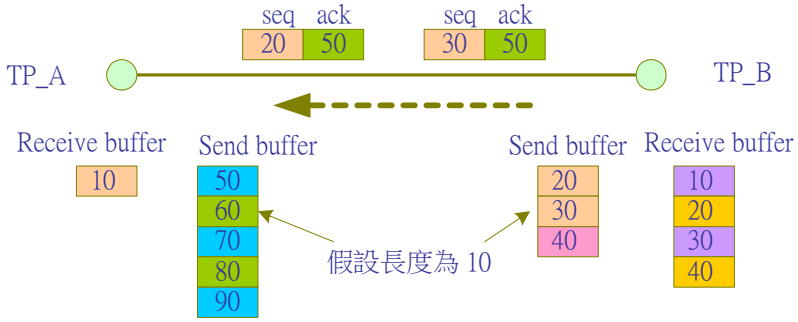

圖 7-16 滑動視窗法基本原理之二

(B) 傳送視窗與接收視窗

由圖 7-15 和 7-16 可以觀察到，傳送緩衝器如果以串列排列，前端緩衝器不停的傳送到連線上，而緩衝器的後端必需不停的上層通訊軟體填入封包，如此該串列緩衝器將無限的往前延伸，而緩衝器也無法重複使用。我們可以用另一種環狀序列來排列緩衝器，如圖 7-17 所示，環狀序列同樣有兩個指標，一者為前端指標，是由上層通訊軟體填入封包的指標；另一者為後端指標，是由緩衝器上取出封包發送到傳輸線上的指標。如果後端不停的將封包發送到傳輸線上，而前端又不停的

填入封包，則整個環狀序列將不停的滑動，因此稱之為『滑動視窗法』(Sliding Window)，傳送緩衝器也稱為『傳送視窗』(Send Window)。當然接收緩衝器也是同樣道理，前端不停的由傳輸線上接收封包；而後端也不停的傳送給上層通訊軟體，也稱之為『接收視窗』(Receive Window)。

如圖 7-17 所示 (假設每個封包長度為 10 Bytes，實務上長短是不定的)，當 TCP (TCP_A 或 TCP_B) 欲傳送封包時，便將傳送視窗的後端指標號碼填入 seq 序號，表示目前傳送資料的順序號碼，而將接收視窗的前端指標填入 ack，表示期望對方下一次傳送的序號，並確認序號以前的資料都正常接收。

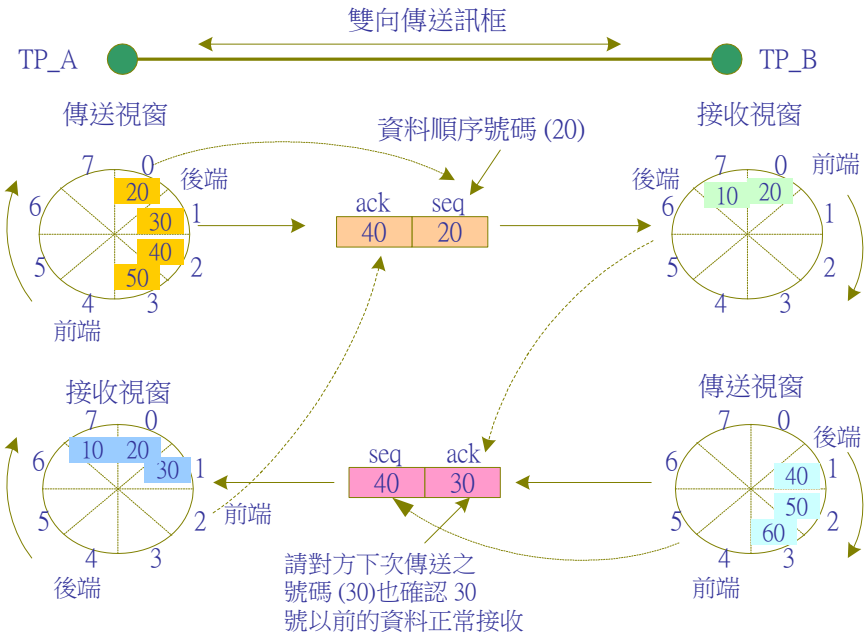

圖 7-17 傳送視窗與接收視窗

(C) 全雙工運作範例

我們用圖 7-18 來介紹一個全雙工傳輸的 TCP 連線，假設工作站 A (136.7.34.2) 上執行 IE 程式，其利用 TP_A (埠口 2478) 連結到工作站 B (148.34.2.6) 的 Web Server，該 Web Server 連結到 TP_B (埠口 80)，因此雙方所建立的連線為【136.7.34.2:2478 ←→148.34.2.6:80】。如圖中，IE 程式依序將欲傳送的資料填入 TP_A 的傳送緩衝器上，而 TP_A 也依序將資料發送到網路上。譬如，TCP_A 連續發送 4 個封包給 TCP_B (20、30、40、50)，每個封包資料長度為 10 Bytes，TCP_B 接收後填入接收緩衝器，也依序傳送給 Web Server。相同的，TCP_B 由 Web Server 上拿取欲傳送的資料填入傳送緩衝器上，也依序傳送給 TCP_A，TCP_A 接收後填入接收緩衝器，再由 IE 程式取走，它們之間就是利用傳送資料時，順便攜帶 ack 訊號來確認已收到的資料，此方法又稱為『搭順風車』(Piggyback)。

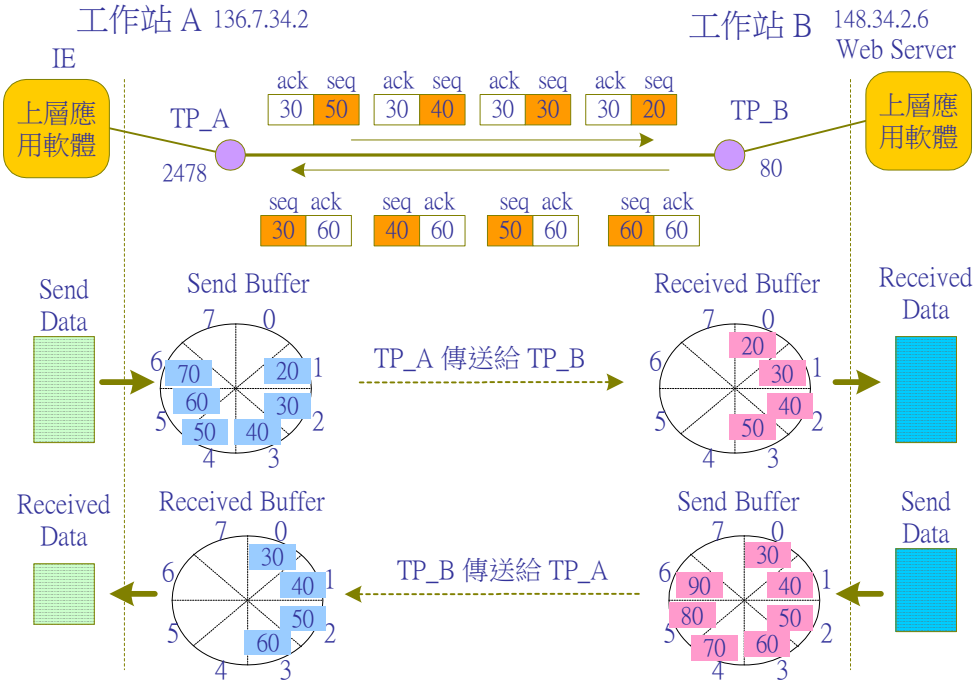

圖 7-18 滑動視窗法之運作

(D) PUSH 旗號

並非所有情況下都是雙方同時傳送資料，多數的時候是單方向傳送資料，此時對方如何傳送確認的訊號？一般發送端都會將 Code PUSH 設定為 1（PSH = 1），表示希望對方接收到後，即時傳送給上層通訊軟體，並回應確認訊號。一般情況下，接收端都會視其情形來回應，如果有要傳送資料它便以搭順風車方式回應，否則另發送確認訊號（ACK & ack）給傳送端。至於會不會即時回應或傳送給上層軟體，這要看接收端的工作負荷情形，因此，在傳送端都設定有一計時器，如溢時未收到回應，將再重送一次。接收端由順序號碼可判斷是否重複接收。

(E) 錯誤控制

TCP 錯誤檢出方法是利用檢查集（Checksum）方法。傳送端發送封包之前，將封包標頭和所承載的資料以檢查集計算，得到一個檢查集數再填入封包標頭的檢查集欄位。接收端收到封包後，再以相同的方法計算所得的檢查集數是否和檢查集欄位的值相同，如果相同便回應正確(Code ACK = 1)，否則回應錯誤（Code ACK = 0）要求對方重送。

7-4-3 視窗大小

視窗大小（Window Size）影響滑動視窗法的傳送速度，也代表傳送緩衝器和接收緩衝器的長度。視窗愈大表示可連續傳送的資料較多，對方也可一次確認較多的資料正常接收，但相對應的必

需較長的緩衝器之記憶體空間。如果連續傳送較長的資料，而對方緩衝器不足於存放時，將會產生資料遺失的現象，因此，在雙方傳送資料之前，必需協議出緩衝器大小，也就是視窗的大小。

一般都是由接收端決定通訊的視窗大小，如圖 7-19 所示，當 TCP_A 要求建立連線時，希望的視窗大小為 4096(以 Bytes 為單位)，並將 4096 的數值填入封包之視窗(Window)欄位裡(win = 4096)。TCP_B 接收到後檢查自己的緩衝器之記憶體後，發現只能提供 2048 Bytes 空間，因此將該訊息回應給 TCP_A (win = 2048)。TCP_A 再回應確認以 2048 的視窗大小傳輸資料，雙方的協議便成功。

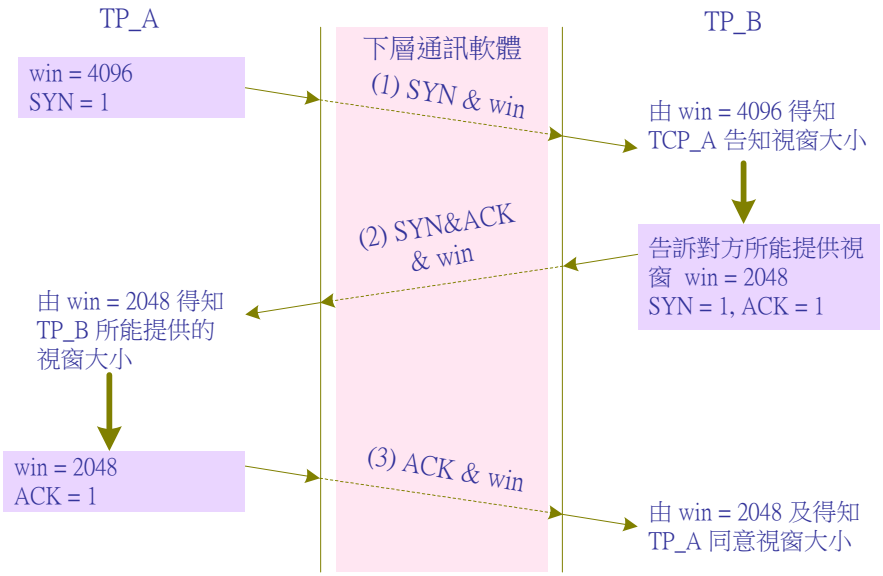

圖 7-19 TCP 建立連線時協商視窗大小

但當接收端接收資料而來不及傳送給上層通訊軟體時，可能會快速耗損接收緩衝器，發生的原因可能是發送端傳輸過快，或上層通訊軟體處理速度過慢，造成滑動視窗滑動過慢。為了不讓傳送的資料因緩衝器爆滿而遺失，接收端必需隨時告知傳送端還有多少緩衝器可以接收資料，通知的方法是接收到回應確認時，將緩衝器空閒長度填入視窗 (Window) 欄位內，傳送端便可以知道對方還有多少空間可以接收資料，如同在圖 7-19 中加入 win 欄位訊號。如果傳送端發現對方視窗欄位的數值快速減少時，便必需減慢傳輸速度。甚至傳送端收到 Window = 0 的訊息，表示對方緩衝器已爆滿，而必須暫停傳送資料。因此，接收端常用此方法來要求傳送端暫停傳送資料，但也不一定是緩衝器爆滿。

7-5 UDP 通訊協定

Internet 網路除了提供可靠性服務的 TCP 連接外，也提供非連接方式傳輸稱之為**『使用者電報傳輸協定』(User Datagram Protocol, UDP)**。UDP 傳輸協定比 TCP 簡單，沒有連線要求、連線終止、以及流量控制的管理程序。它的優點是傳輸速率較快，主要應用於較少量、即時性傳輸，而對資料正確性的要求較不高 (如語音或視訊) 的環境下使用。而其缺點則是無法提供正確性較高的資料傳輸。採用 UDP 傳輸可能會有資料重覆、資料未依序到達、資料遺失等等問題，必須由使用者自行解決。但從一方面來思考，Internet 網路上有許多應用系統，它們之間的傳輸量很低，而且需要即時反映訊息，如果採用 TCP 連線反而會浪費許多連結時間，而影響傳輸效率，在這種情況之下使用 UDP 的效率相對應較高，譬如 DNS 伺服系統或 SNMP 協定。因此，可以做一個簡單的結論，再傳輸量比較少或需要及時反映的環境下，使用 UDP 協定傳輸會優於 TCP 協定。但在許多情況下，使用者很難去決定到底應該使用何種協定來傳輸目前的資料，因此，在許多系統在同一傳輸埠口上，提供有 TCP 和 UDP 兩種協定讓使用使者連接，如果使用者的資料不需要分割，也就是說，一個 UDP 封包可以承載的話，那就使用 UDP 協定傳輸，如果需要多筆封包傳輸，則使用 TCP 協定傳輸。

至於一個傳輸層封包可以承載多少訊息，如果以圖 7-2 IP 封包格式計算，一個 IP 封包最大的長度為 65536 Bytes，扣除 IP 封包標頭長度(20 Bytes)，再扣除 UDP 封包標頭長度(20 Bytes)，則 UDP 所能承載的資料長度為 65496 Bytes，但這是很難達成的網路傳輸環境。一般我們為了 IP 封包在傳輸中避免有再被分段的情形，網路通訊之前都會協議出**『最大傳輸單位』(Maximum Transmission Unit, MTU)**，如果以連結的子網路都是 Ethernet 網路為例 MTU = 1518 Bytes，而 IP 封包最大為 1500 Bytes，則 UDP 封包最大為 1480 Bytes，也表示所傳輸的資料少於 1460 Bytes，則可使用 UDP 協定傳輸，否則必需用 TCP 協定傳輸。也可說是，到底傳輸量少於多少位元組可以用 UDP 協定傳輸，這必需看使用者連結網路的環境，並沒有一個固定的標準。

UDP 封包與 IP 封包之包裝方式如圖 7-2 所示，圖 7-20 為 UDP 封包格式，因其為非連接方式，所以沒有順序號碼、確認號碼和其它控制欄位，而各欄位功能如下：

- **來源埠口 (Source Port)**：發送端之傳輸埠口。
- **目的埠口 (Destination Port)**：接收端之傳輸埠口。
- **長度 (Length)**：該封包所承載資料 (Data) 的長度。
- **檢查集 (Checksum)**：該封包之錯誤檢查的檢查集。

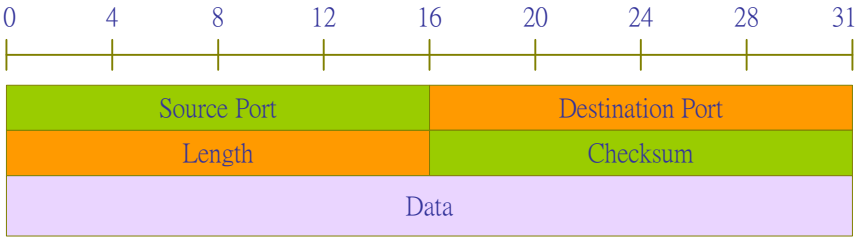

圖 7-20 UDP 之封包格式

至於『檢查集』欄位的產生就較為複雜，因此，有些應用環境為了提高效率，而將此欄位填入 0，而不使用錯誤檢查的功能。檢查集所檢查的範圍除了 UDP 標頭和所承載的資料外，還包含一些 IP 標頭的欄位，我們將所檢查的欄位組成一個稱之為『虛擬標頭』(Pseudo Header)，其內容如下：

- **IP Source Address**：(4 Bytes) IP 標頭之來源 IP 位址。
- **IP Destination Address**：(4 Bytes) IP 標頭之目的 IP 位址。
- **Protocol**：(1 Byte) IP 標頭之協定號碼欄位。
- **Length**：(2 Bytes) UDP 標頭之長度欄位。
- **Padding**：(1 Bytes) 補滿虛擬標頭成為偶數位元組長度，以方便計算 Checksum。

虛擬標頭的檢查方法是傳送端欲發送資料之前，首先建構虛擬標頭，再計算出檢查集的檢查碼，將其填入 UDP 的檢查集欄位，並捨棄虛擬標頭而不將其傳送過去。接收端收到 UDP 封包後，也再建立虛擬標頭來計算檢查碼，如果所計算出來的檢查碼和檢查集欄位的值相同，便判斷該 UDP 封包沒有發生錯誤。使用虛擬標頭檢查可視為 UDP 封包的雙重保全機制，如果封包在傳遞中發生錯誤，而下層通訊沒有檢查出來，虛擬標頭可以做第二道防線的檢查。

7-6 傳輸層程式介面

既然程式設計師可以透過傳輸埠口連結到應用程式，而不用理會網路所連結型態，這也表示傳輸層介面是和網際層的關係是獨立性的，因此，傳輸層必須提供程式介面讓使用者銜接。使用者也可透過程式介面來開發網路應用程式，在網路系統上有兩個介面標準：Socket 和 TLI (Transport Layer Interface)，但 Internet 網路的應用層協定大多建立在 Socket 介面上，我們在第十章有詳細介紹 Socket 開發程式的範例，至於 TLI 介面在 Internet 網路上較少應用，讀者如欲更瞭解 TLI 的運作原理，請參考有關書籍 (或附錄上參考書目)。以下就針對這兩個標準作簡單的介紹。

7-6-1 Socket 程式介面

Socket 是一種『應用程式介面』(Application Program Interface, API)，它是由一組稱為『Socket Library』的庫存函數所構成，它提供開發各式各樣應用程式的工具，程式設計師只要透過介面程式來編寫應用程式，就不需要去考慮網路是如何實現出來的。就如一般作業系統上的『系統呼叫』(System Call)程式一樣。程式設計者只要呼叫 open() 介面程式，而給予適當的參數(parameter)，它便會指定到磁碟機或其他週邊設備上，亦可存取到所需的資料，而無需考慮到磁碟機和其他週邊設備如何驅動馬達和磁軌。

『插座』(Socket) 的概念是希望提供一標準介面，來銜接一般應用程式，也就是說，提供一個網路和程式之間的標準介面，只要應用程式合乎 Socket 的介面標準(也就是系統呼叫的參數)，便可安然無恙的連接到網路，而不用理會網路實際的連接型態。這種觀念就好像我們一般電話『插座』一樣，只要合乎插座內的配線裝置，就可以連結到電話系統，甚至可撥接電話到任何地方，而不用理會電話系統實體上是如何安裝及跨接。

Socket 為了提高程式設計的方便性，將各種功能呼叫程式製作成庫存函數，並可載入系統核心，讓使用者編寫網路應用程式，就如一般檔案系統一樣，以下簡略介紹一般較常用的 Socket 功能呼叫：

(1) socket()：開啟 Socket 通訊服務點。

(2) bind()：對 socket() 定址，連結至相對應的 TCP 或 UDP 埠口。

(3) listen()：設定 socket 為等待狀態，等待 Client 端要求連線。

(4) connect()：要求和對方通訊端（socket）連線。

(5) accept()：接受對方（socket）連線要求。

(6) write()：將資料寫入連線中的 socket，傳送到通訊對方。

(7) read()：由連線的 socket() 中讀取資料。

(8) close()：釋放 socket，中斷連線。

圖 7-21 是利用 Socket 功能呼叫所開發之檔案伺服器的主從式架構，此型態是連接導向方式，一般都必須使用 TCP 連線。首先 Server 端利用 socket()、bind() 連結到通訊傳輸埠口（TCP 埠

口），而以 listen() 進入聆聽狀態，並以 accept() 等待對方連線要求。Client 端可以用 connect() 要求連線，連線後雙方以 read() 或 write() 來接收或傳送資料，並以 close() 功能呼叫來結束雙方的通訊。

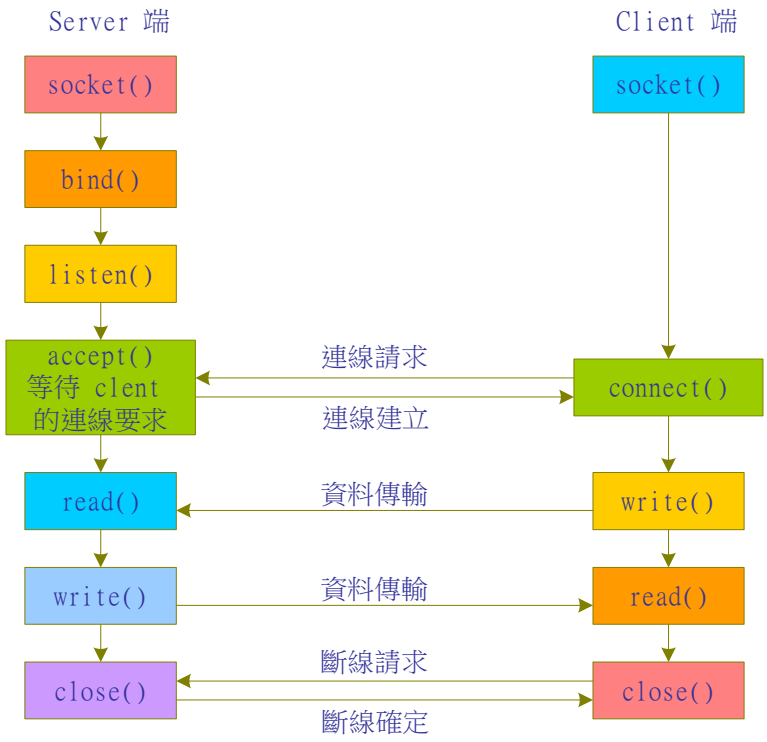

圖 7-21 利用 Socket 發展程式範例

7-6-2 TLI 程式介面

『傳輸層介面』（Transport Layer Interface, TLI）最早是在 Unix System V 3.0 版本中出現，而是作為 OSI 通訊協定的傳輸層介面標準。TLI 介面和 Socket 介面最大的不同點是，Socket Library 中所提供的功能呼叫，如 read()、write() 都屬於 Unix 的**『系統呼叫』（System Call）**，而是屬於作業系統的一部份，也就是說，Socket 介面是包含在 Unix 的核心程式之內。而 TLI 僅是一個介面程式，使用者必須將 TLI Library 和所發展的程式連結（Link）在一起，才可以產生執行檔。TLI 介面在執行時，必須透過 Unix 的 Stream I/O 介面（請參考 Unix 作業系統）才可以和傳輸層作溝通。

雖然 Socket 和 TLI 介面在實現方面有所不同，但它們的目的都是一致的，也是網路應用程式和網路下層之間的標準介面，讓使用者不用理會網路實際的連結型態，便可開發網路應用程式。TLI Library 也提供一系列的標準功能呼叫，以下我們列出一些較常用的介面程式：

(1) t_open()：開啟 TLI 通訊端點。

(2) t_bind()：對通訊端點定址到 TCP 或 UDP 傳輸埠口。

(3) t-alloc()：配置記憶體佔用空間。

(4) t_listen()：設定通訊端點為聆聽（Listen）狀態。

(5) t_connect()：要求對方連線。

(6) t_accept()：接受對方連線要求。

(7) t_snd()：連接導向的傳送資料。

(8) t_rcv()：連接導向的接收資料。

(9) t_close()：結束通訊連線。

圖 7-22 為利用 TLI 發展一個檔案伺服器的主從式架構，此型態也是連接導向方式（TCP 連線）。首先，Server 端利用 t_open() 和 t_bind() 開啟一個通訊端點，並連結到 TCP 傳輸埠口上，再載入記憶體空間內（t_alloc()）執行，緊接著，呼叫 t_listen() 來等待 Client 端連線要求。Client 端也連結到通訊端點後，以 t_connect() 向 Server 端要求連線。Server 端收到連線訊號後，以 accept() 呼叫得知連線對方的位址。爾後雙方以 t_snd() 與 t_rcv() 互相傳送與接收資料。

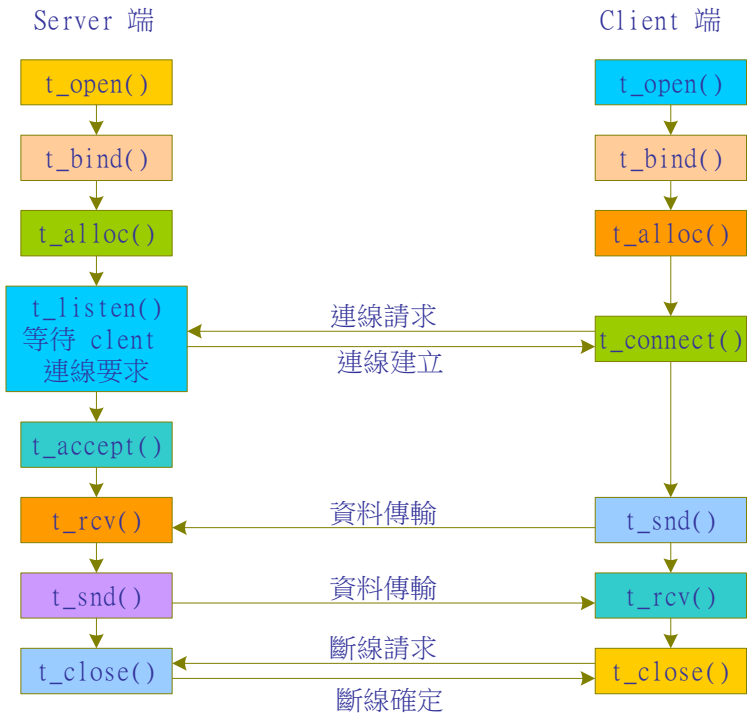

圖 7-22 利用 TLI 發展程式範例

習題

1. 在 Internet 網路上，一般傳輸層有哪兩個通訊協定？請分別敘述其功能。
2. 傳輸層中有 TCP 和 UDP 協定，我們分別選用這兩個通訊協定來傳輸資料的依據為何？
3. 為何『**TCP/IP**』兩個通訊協定必需結合在一起？請敘述其原因。
4. 請執行 telnet 命令，來擷取三向式連絡法的建立連線封包，並請繪圖說明其動作順序。
5. 請執行 ftp 命令來擷取 TCP 封包，並說明封包標頭各欄位表達的功能。
6. 請執行 mail 命令，來擷取 UDP 封包，並說明封包標頭各欄位表達的功能。
7. 何謂『**TCP 埠口**』(**TCP Port**)？並請說明 TCP 和 IP 的多工關係。
8. TCP 有『**動態配置**』和『**固定配置**』兩種方法來指定傳輸埠口位址，請說明兩者在使用上有何不同？
9. 何謂『**三向握手式連絡法**』(**Three-way Handshake**)？
10. 請說明 TCP 建立連線的運作程序。
11. 請說明 TCP 資料傳送的運作程序。
12. 請說明 TCP 連線終止的運作程序。
13. 請說明 TCP 連線如何重新啟動？並請說明重新啟動的時機。
14. 請說明在同一伺服器可能同時接受許多客戶端的連線，它如何來分辨這些連線？並請舉例說明之。
15. 何謂『**流量控制**』(**Flow Control**)？
16. 請說明『**停止與等待**』(**Stop-and-Wait**)流量控制法的運作程序。它有何優缺點？
17. 請簡略說明『**滑動視窗法**』(**Sliding Window**)的運作程序。
18. 請說明 TCP 封包標頭中 Sequence Number 和 Acknowledge Number 兩個欄位，在滑動視窗法的流量控制中扮演何種功能？

19. 如果雙方資料在傳送當中，接收端想請傳送端暫停傳送，應如何處置？

20. 在 TCP 連線當中，雙方如何來協議緩衝器大小的問題？

21. 請執行 ftp 中的 get 命令來擷取封包（請參考附錄 A），並繪圖說明流量控制的情形。

22. 請說明在 Ethernet 網路上，UDP 最大封包大小為何？如何計算出來？

23. 何謂 UDP 的『**虛擬標頭**』（**Pseudo Header**）？它的功能為何？

24. 一般傳輸介面程式有哪兩種？請分別敘述其特性。