Socket Programming

何謂 socket

從網路的角度來看，socket 就是通訊連結的端點；從程式設計者的角度來看，socket 提供了一個良好的介面，使程式設計者不需知道下層網路協定運作的細節便可以撰寫網路通訊程式。

Windows Socket 是以實作於 Berkeley Software Distribution(BSD, release 4.3)中的 UNIX sockets 為基礎所發展出來的一套 API，其不僅支援 TCP/IP，對於 XeroxR Network System (XNS)，Digital Equipment Corporation's DECNet protocol，NovellR Corporation's Internet Packet Exchange/Sequenced Packed Exchange(IPX/SPX) 亦可以支援。此外也提供機器碼的相容性，因此在不同的作業系統上移植時並不需要做任何修改。

Windows Socket API 是一套動態連結函式庫(DLL)，即程式在編譯時期並不會和這些函式庫連結，而是等到執行期間才會呼叫這函式。

Sockets 的分類

在 TCP/IP 架構下，sockets 可分為下面兩類：

（1）Datagram sockets(connectionless)

資料在 datagram sockets 間是利用 UDP 封包傳送，因此接收端 socket 可能會收到次序錯誤的資料，且其中部分資料亦可能會遺失。

（2）Stream sockets(connection-oriented)

資料在 stream sockets 間是利用 TCP 封包來傳送，因此接收端 socket 可以收到順序無誤、無重覆、正確的資料。此外 TCP 傳送時是採資料流的方式，因在傳送時會所有資料會視情況被分割在數個 TCP 封包中。

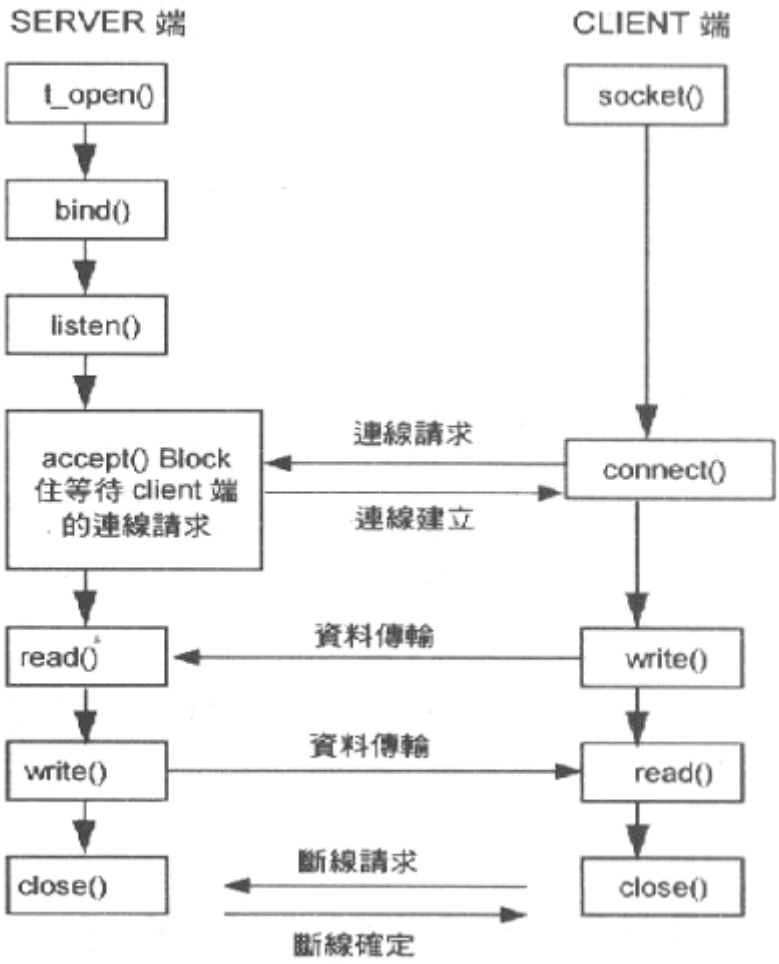
主從式架構模型(Client/Server model)

每個網路應用程式都有一個通訊端點，一種端點是用戶端，另一種是伺服器。根據定義，用戶端會先送出第一個封包，由一個伺服器接收。在初步接觸後，用戶端和伺服器均能開始收送資料。

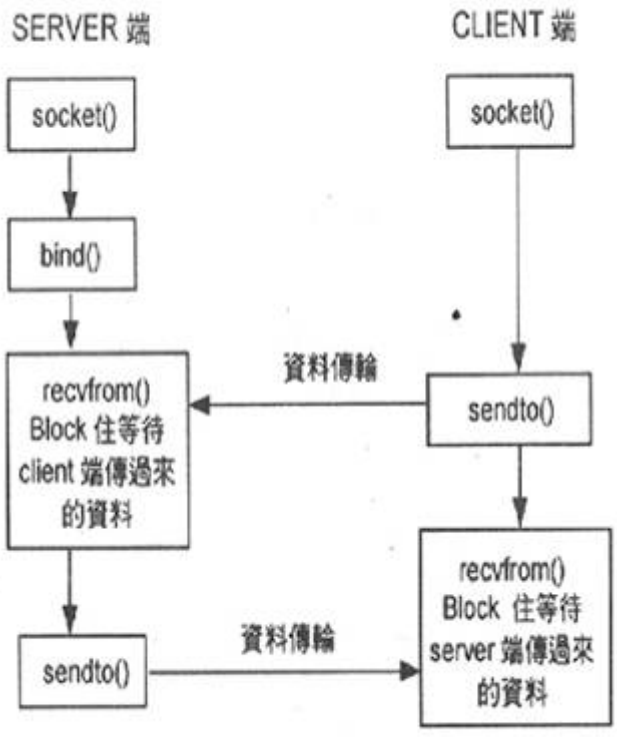
依據 socket 所提供的服務來將它分類，然而在用戶端和伺服器上的這兩個 sockets 必須是同一類才能互相通訊，也就是說，他們必須都是 stream(TCP)或都是 datagram(UDP)。用戶端的應用程式必須要能找到並識別伺服器的 socket，而伺服器會將它的 socket 命名以讓用戶端識別，就 TCP/IP 而言，一個 socket name 包括了 IP 位址、連結埠編號、以及協定本身。用戶端可用 Windows Sockets 的名稱伺服函式來查到標準伺服器的連結埠編號，而如果知道伺服器的主機名，則可以 Windows Sockets 的主機名稱分析函式，來查得伺服器的 IP。當用戶端 socket 成功地聯繫上伺服器端之 socket 後，這兩者便形成一個“結合”(association)。在此時，每個 socket 都可以由它的名字及對方的名字所形成的組合加以識別。這個結合包括五個要素：所用的協定、用戶端 IP 位址、用戶端連結埠號碼、伺服器端 IP 位址、伺服器端連結埠號碼。這個“結合”的觀念並不只是 Windows Sockets 程式設計的基礎，它也是一般網路通訊的重要觀念。在結合中的資訊可識別及引導封包通過網路，從這一端的程式傳至另一端。

所有的網路應用程式皆可分為五個步驟：

- 開啟一個 socket
- 為 socket 命名
- 與另一個 socket 結合
- 在 sockets 間收送資料
- 關閉 socket



TCP Socket Connection



UDP Socket Connectoin

圖一 Socket 程式簡圖

開啟一個 Socket

Socket 是通訊的端點，好比是電腦的網路介面卡，使得網路應用程式可以像介面卡插在主機板上一樣，插入網路中。一般說來你只會有一片網路卡在電腦中，但是你可以有許多 sockets，而且它們也可以同時使用一片網路卡。用戶端與伺服器端都需要一個 socket 以存取網路資料，使用 socket()函式呼叫就可開啟一個 socket。(如圖一所示)

為 Socket 命名

伺服器端的程式必須為它的 socket 命名，這樣用戶端才能找到並正確地辨識出它的 socket，如果伺服器沒有替它的 socket 命名，則協定堆疊會拒絕用戶端要通訊的請求。要幫 socket 取名必須設定三個參數：協定、連結埠號碼、及位址，而用戶端就要用這些值來和伺服器建立連結。要為 socket 命名，伺服器必須為 socket 位址結構設初始值並

呼叫 bind()函式，以指定本身連結埠號碼和 IP 位址，完成命名的工作。

與另一個 Socket 結合

假設我們在用戶端與伺服器端均開啟一個 socket，並至少為伺服器端的 socket 命名。接下來伺服器要準備接收封包，而用戶端要準備發送封包，當此準備工作完成後，此兩端的 sockets 就叫建立一個"結合"(association)。如何為結合此兩端的 socket 做準備呢?在 WinSock API 中提供了幾個函式來完成此動作。在伺服器端則以呼叫 listen()來準備接受用戶端送來的連結要求，如果收到連線要求，則開啟另一個新的 socket 來和用戶端進行連線(使用 accept()函式)；而在用戶端則是呼叫 connect()函式與伺服器端的 socket 完成結合。

在 Sockets 間收送資料

此時我們已經在用戶端和伺服器的 sockets 間建立了結合，也就是說，我們已經可以開始收送資料了。如何收送資料呢?在一個已連結的 socket 上收送資料可呼叫 recv()與 send()來完成；而在一個無連結的 socket 上收送資料可呼叫 recvfrom()與 sendto()來完成。

關閉 Socket

當用戶端完成收送資料且往後並不會在使用時，必須關閉 socket，對 TCP socket 而言，關閉 socket 除了將 socket 的資源還給協定堆疊，此外並嘗試將以建立的連結關閉。但對 UDP socket 而言，則是單純地將資源還給協定堆疊。關閉 socket 可呼叫 closesocket()來完成。

參考文獻

1. "INTERNETWORKING WITH TCP/IP VOL.3 CLIENT-SERVER PROGRAMMING AND APPLICATIONS Windows Sockets Version, " Douglas E. Comer and David L. Stevens, Prentice-Hall International, Inc.

2. “Windows Sockets 網路程式設計經典”，李孟書、黃鶴超譯，美商愛迪生衛斯理、碁峰資訊股份有限公司合作出版。

3. “WinSock 網路程式設計之鑰”，黃俊堯、黃耀文、許景華、陳孝忠著，資訊人文化事業