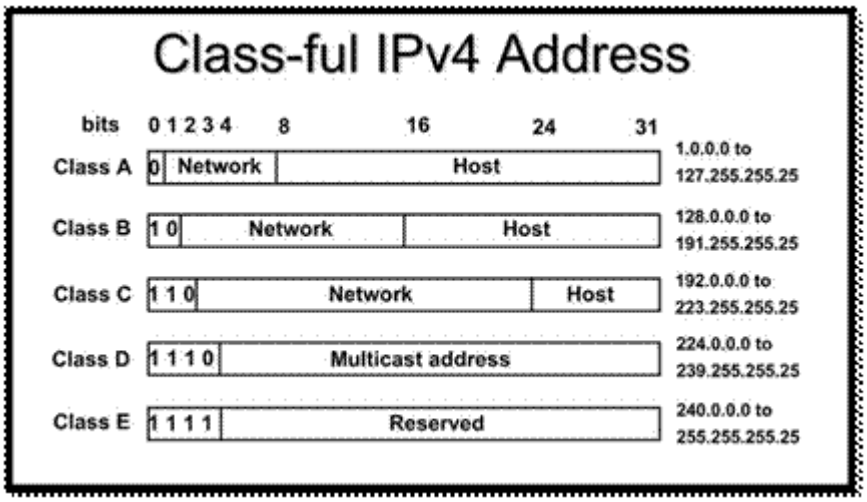
作業系統原理
小專題：Socket in IPv6
B9742203 鄭功蔚

什麼是IP？
IP就像是網路世界在使用的門牌號碼，當郵差要送信件時（封包），則必須要有地址才可以找到收件人。

IPv4到IPv6的演進：
1981年9月 RFC791制定了Version 4 (IP 4)的版本，從誕生迄今大約經過了20年的時間。在這段期間電腦的技術有很大的進步，出現 在這段期間電腦的技術有很大的進步，出現了各式各樣在當初設計IPv4時所未被設想到的使用型態。網路的急速普及化，也使得現 的使用型態 網路的急速普及化也使得現在的網路規模遠超過設計當初的預期。以下是當初制定的IPv4所分配的class

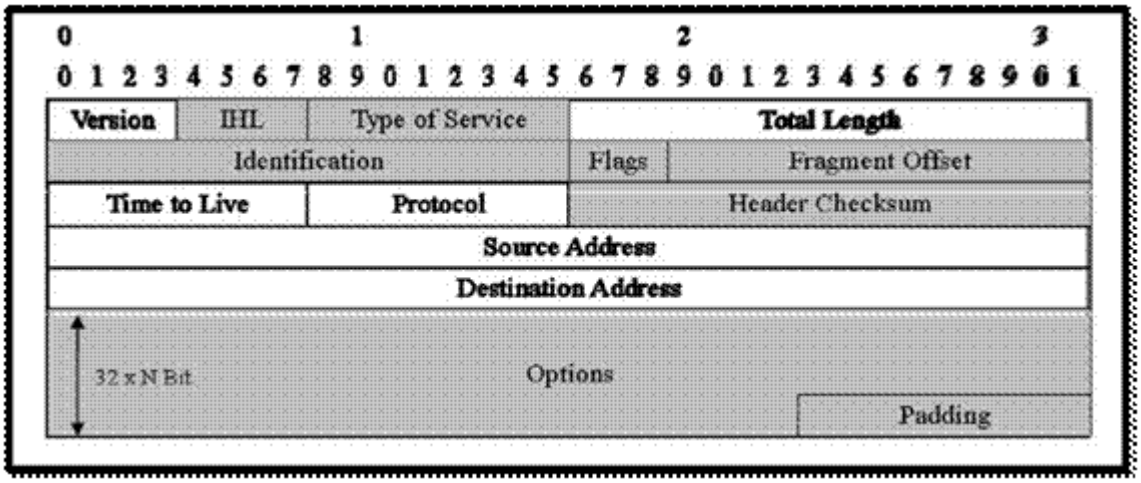


IPv4的網路位址只有32位元，可以表示達4294967296的網路位址。雖然說四十二億個位址，全世界每人分配一個可能都還足夠，但是在實際的狀況裡，網路位址的分配並非如此簡單。好比說一個人不見得只會用到一組IP，若一個人同時用電腦，也同時使用手機上網，這樣的話實體IP上就會不負使用了。雖然當時出了許許多多的解決方案像是分成子網路（Subnetting），不分級網域間路由(Classless Inter Domain Routing, CIDR)，網路位址轉譯(Network Address Translation, NAT)，位址重複使用一動態主機設定協定(Dynamic Host Configuration Protocol, DHCP)等方法，但終究是個治標不治本的藥，而IPv6也就從此誕生了。

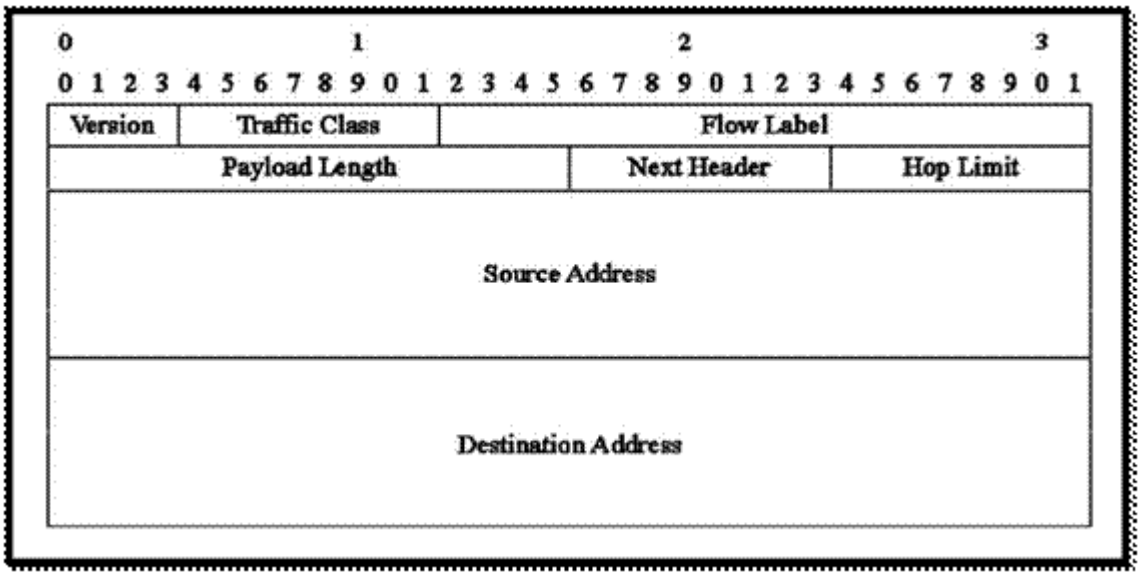
IPv6簡介：
IETF於一九九五年起開始進行新一代的網際網路通信協定之推動，成立Ipng及Ngtrans兩個工作小組，著手進行所謂IPv6之訂定與相關測試活動之推動，以尋求解決近年來IPv4位址即將不足，與改善現有通信協定不夠完善等問題。初期由於IPv6標準制定未臻成熟，且因為有路由選擇機制(Classless

Inter-Domain Routing；CIDR)，及網路位址轉換機制(Network Address Translation；NAT)的輔助，IPv4位址尚未有立即不足之憂慮，故未見立即之市場效益。經過幾年的努力，IPv6標準於一九九八年成為IETF正式之RFC 2460。

下圖為IPv4與IPv6之header樣試圖，當中灰色的部分為IPv6取消或變更的部分



IPv4 Header



IPv6 Header

Version (4 bits)	表示Internet Protocol 的版本號碼。IPv6 即為 表示Internet Protocol 的版本號碼 IPv6 即為0110。
Traffic Class (8 Bits)	表示封包的類別或優先度 這個欄位與IP 4 表示封包的類別或優先度。這個欄位與 IPv4之 " Service Type " 提供相同的功能。
Flow Label (20 Bit) Flow Label (20 Bit)	顯示封包所屬的Flow編號。在不支援Flow Label 欄位的機能的主機或路由器上，會使用其預設值0。

Payload Length (16 Bit)	以無號整數表示在IPv6基本標頭之後剩下的封包長度，以Byte為單位計算。
Next Header (8 bits)	下一個標頭的種類
Hop Limit (8 bits)	以無號數表示IPv6 封包被捨棄之前最多可經過的 以無號數表示IPv6 封包被捨棄之前最多可經過的節點數。
Source Address (128 bits)	封包來源的IPv6 位址。
Destination Address (128 bits)	封包目的地的IPv6 位址。一般來說，會設定為最終目的地的位址，但若延伸標頭中有Routing Header存在時，則不設定最終目的地，而是設定於Source Routing List所記錄的下一個Route Interface的位址。

IPv6優點：

1.長度(Length)：由於IPv6表頭長度固定，IPv4的整體長度由IPv6的封包承載長度(Payload Length)取代。

2. 協定型式(Protocol Type)：協定型式欄位重新命名成下一表頭(Next Header)，用以反映IP封包新的組織架構。此外，除了原先的使用者數據電報協定(UDP)和傳輸控制協定(TCP)協定型式外，亦可增加延伸表頭(Extension Header)。

3. 存活時間(Time To Live)：此欄位變更成跳躍點限制(Hop Limit)，以符合路由器實際的處理狀況。

最後，增加二個新的欄位，用以支援及時(Real Time)訊務之需求，為優先順序(Traffic Class)和訊流標記(Flow Label)。雖然表頭的整體長度是增加的，欄位的數目卻相對減少了。此外，選項機制(Option)是完全地被修正。選項欄位是由延伸表頭來取代且置放於IPv6表頭和轉送層(Transport Layer)PDU之間。

實做：
在Visual Studio2005以IPv6建立socket。

程式碼
server端

```
#include <winsock2.h>
#include <stdio.h>
#include <ws2tcpip.h>
#pragma  comment(lib,  "Wsock32.lib")            //連結Wsock32.lib資料庫
void main()
{
    int FromLen,retval;
    char *Port = "5001";                  //宣告指定的PORT號
    char Buffer[1024] = "";               //宣告Buffer的大小
    ADDRINFO Hints, *AddrInfo;              //定義Hints與指標AddrInfo結構
    char *Address = NULL;                 //宣告Address為空值
    SOCKADDR_STORAGE From;                    //定義From為一個SOCKADDR_STORAGE的架構
    SOCKET ServSock,msgsock;                //宣告SOCKET ServSock與msgsock

    WORD wVersionRequested;                 //存取能使用Socket最高的版本.2
    wVersionRequested = MAKEWORD( 2, 2 );
```

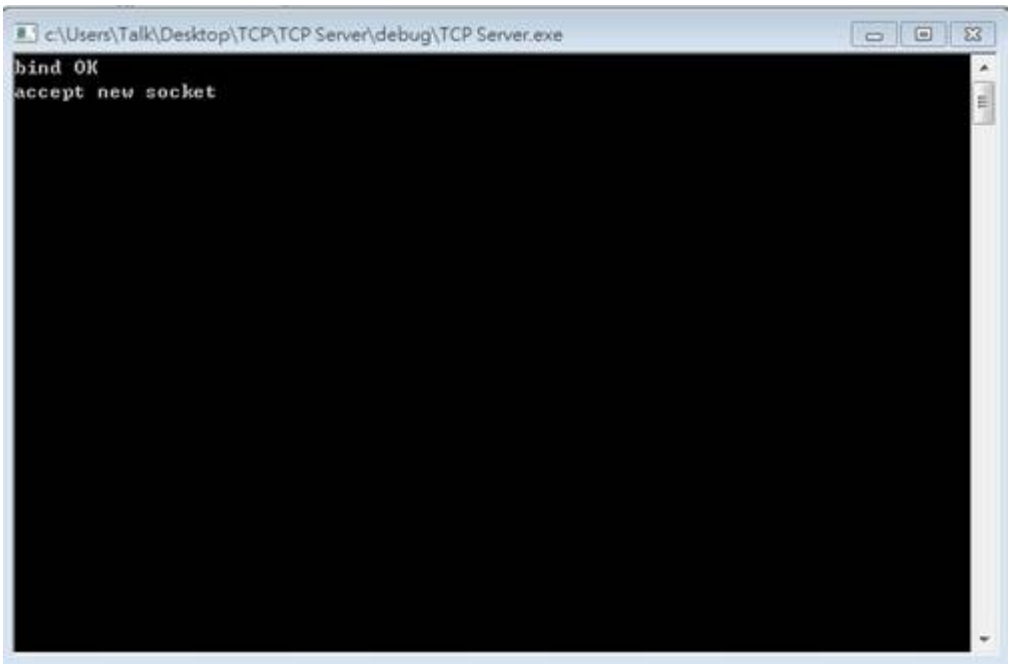
```
    WSADATA wsaData;                        //資料初始化到WSAStartup
    WSAStartup( wVersionRequested, &wsaData ); //執行指定的Winsock DLL

    memset(&Hints,0,sizeof(Hints));          //清除所使用的記憶體區塊
    Hints.ai_family = PF_INET6;              //設定使用IPv6成員
    Hints.ai_socktype = SOCK_STREAM;         //TCP使用SOCK_STREAM
    Hints.ai_flags = AI_NUMERICHOST | AI_PASSIVE;
                            //將flag 設定為AI_PASSIVE 表示在bind()函數調用中使用返回位址的結構
    getaddrinfo(Address,Port,&Hints,&AddrInfo);
                            //取得到位址與PORT號並轉換ANSI host名稱為一個可用的位址
    ServSock=socket(PF_INET6,SOCK_STREAM,0);  //宣告SerSock 參數IPv6，TCP，這裡使用預設
    if(bind(ServSock,AddrInfo->ai_addr,AddrInfo->ai_addrlen) >= 0 );
                            //綁定ServSock IP位址，IP長度
    printf("bind OK\n");                    //顯示ServSock bind成功
    listen(ServSock,5);                     //設可偵測個listen
    FromLen=sizeof(From);                   //宣告sockaddr長度

    while(1)
    {
      msgsock=accept(ServSock,(LPSOCKADDR) & From,&FromLen);
            //msgsock 為接受(ServSock的SOCKET,自己的位置Form,Form的長度)
      printf("accept new socket\n");        //顯示接受新的SOCKET
      retval = recv(msgsock,Buffer,1000,0); //接受msgsock,資料為Buffer裡面的內容,長度,0
      printf("Server收到:%s\n",Buffer);     //顯示Server收到Buffer的資料
      retval = send(msgsock,Buffer,sizeof(Buffer),0);
                            //回傳msgsock 資料為Buffer,Buffer長度
    }

    closesocket(msgsock);                   //關閉msgsock SOCKET
    WSACleanup();                           //清空回傳值
}
```

client

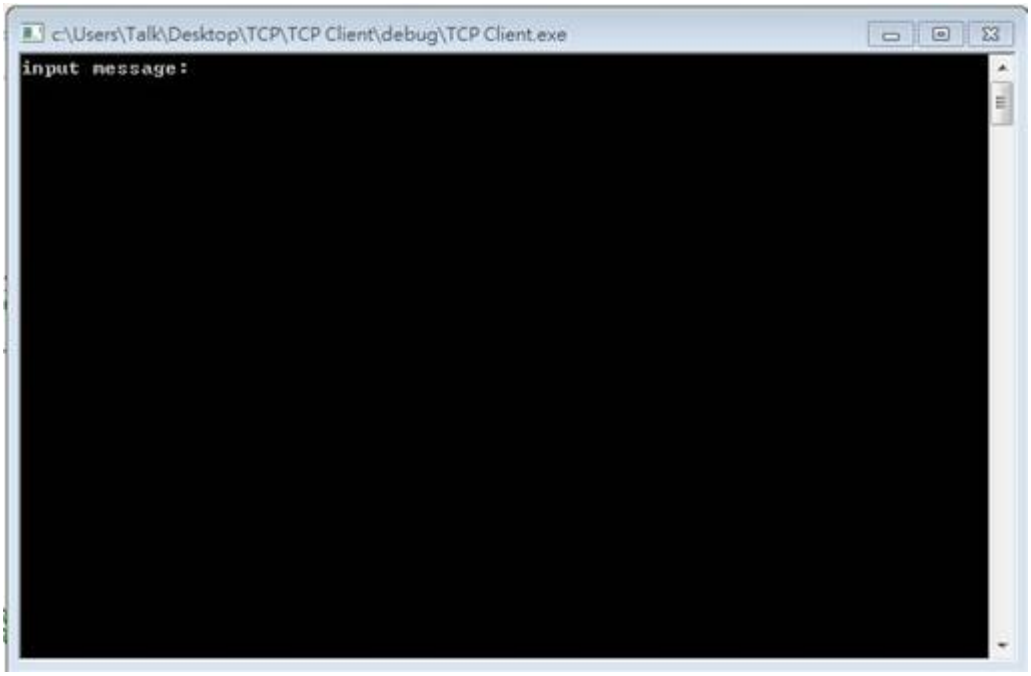
```
#include <winsock2.h>
#include <stdio.h>
#include <ws2tcpip.h>
#pragma  comment(lib,  "Wsock32.lib")   //連結使用Wsock32.lib資料庫
void main()
{
    char *Server ="::1";                //目的地IPv6位址
    char *Port="5001";                  //目的地port
    ADDRINFO Hints, *AddrInfo;          //定義Hints 與指向AddrInfo架構

    WORD wVersionRequested;             /
                        *訪問者能使用Socket最高的版本。高位指定次要版本; 低位指定主要版本。*/
    wVersionRequested = MAKEWORD( 2, 2 );
            /* Use the MAKEWORD(lowbyte, highbyte) macro declared in Windef.h */
    WSADATA wsaData;
            /*這個結構被用於存放初始化數據到WSAStartup*/
    WSAStartup( wVersionRequested, &wsaData ); //啟動使用的Winsock DLL

    SOCKET sockClient;
    memset(&Hints,0,sizeof(Hints));          //清除記憶體區塊
    Hints.ai_family=PF_INET6;                //設定使用IPv6 成員
    Hints.ai_socktype=SOCK_STREAM;           //設定使用TCP SOCK_STREAM
    getaddrinfo(Server,Port ,&Hints,&AddrInfo );//取得到伺服器位址與port號

    char recvbuffer[1024]="";                //接收資料的長度
    char buffer[1024]="";                    //發送資料的長度
```

```
    while (1)
    {
        sockClient=socket(PF_INET6,SOCK_STREAM ,0);
                    //宣告sockClient 參數IPv6，TCP，協定(寫入while可讓程式一直和server持續進行)
        connect(sockClient,AddrInfo->ai_addr, AddrInfo->ai_addrlen);
                   //連結到sockClient的位置,位置長度(寫入while可讓程式一直和server持續進行)
        printf("input message:");            //顯示input message:
        scanf_s("%s",buffer);                //輸入的字串填入buffer
        send(sockClient,buffer,sizeof buffer,0);
             //(send) 發送sockClient的資料為buffer,長度,sizeof buffer,flags設為
        recv(sockClient,recvbuffer,sizeof recvbuffer,0);
                   //(recv) 接收sockClient的資料為recvbuffer,長度,sizeof buffer,flags設為
        printf("%s\n",recvbuffer);           //顯示接收到字串
    }
    closesocket(sockClient);                 //關閉sockClient 的SOCKET
}
```

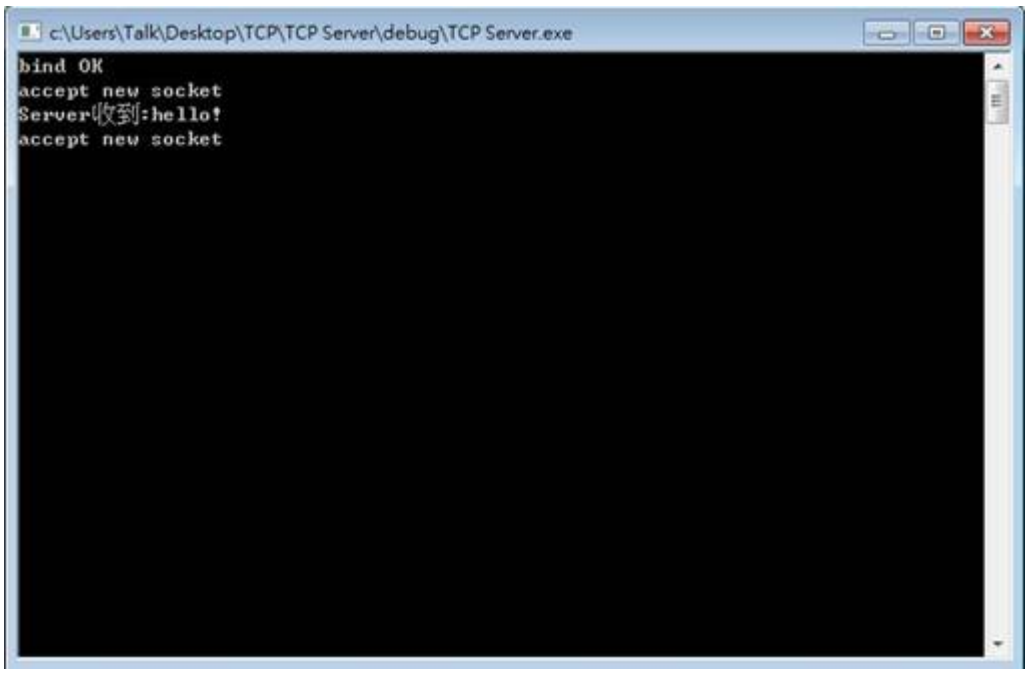
執行結果：



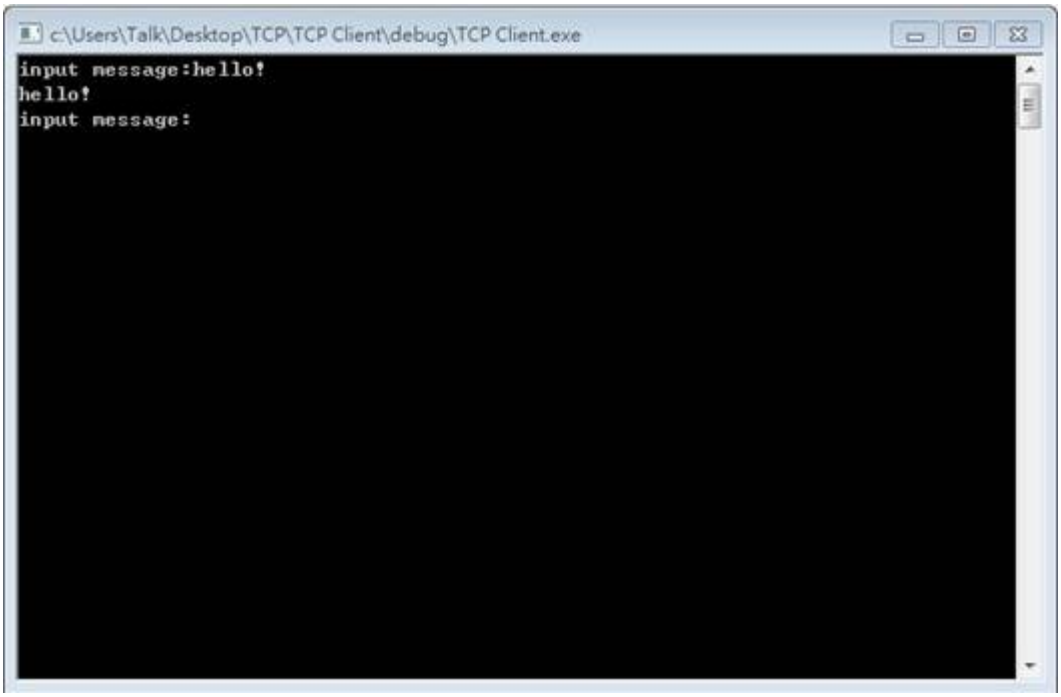
Server端



Client端



Server端 接收資料



Client端 傳送資料

參考資料：

1.旗標出版社 IPv6 新世代網際網路協定暨整合技術

2. http://kksc.myweb.hinet.net/IPV6.htm

3. http://zh.wikipedia.org/zh-tw/IPv4