Delphi 网络高级编程

鲍敏 吴昊 等 编著

人 民 邮 电 出 版 社

第 7 章　UDP 协议相关网络应用

UDP 是无连接协议，与 TCP 操作不同，计算机间并不需要建立连接，也就是说无需传递握手信号，也不用知道对方主机是否在网上，只要知道对方主机的 IP 地址就可以传递数据。

UDP 协议的另一个特点就是它的每一个应用程序可同时作为客户方和服务器方。由于 UDP 协议并不需要建立一个明确的连接，因此建立 UDP 应用要比建立 TCP 应用简单得多。在 TCP 应用中，作为服务方的 Socket 必须明确地设置成监听模式（Listen），而其他 Socket 则必须使用 Connect 方法来初始化一个连接。在使用 UDP 协议时就可以省略这些过程（但是在建立的时候需要 Bind 一个端口）。

UDP 协议的母体和 TCP 协议一样，也是 IP 协议，它的数据流结构依次包括 IP 头、UDP 头和数据 3 部分。在 IP 头里包括发送方和接收方的 IP 地址、端口和使用的协议类型。UDP 头里包括发送方和接收方的端口。

7.1　发送 UDP 数据包

7.1.1　使用 NMUDP 组件发送 UDP 数据包

NMUDP 组件封装了 UDP 协议的实现，并给出了几个属性和方法用于对整个 UDP 组件的动作进行操控。以下程序使用了 NMUDP 组件进行文本数据的传送。程序代码参见光盘目录 ch7\UDP Package Send。

程序的窗体类以及 uses 字段的内容如下：

```
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, NMUDP, ComCtrls;

type
  TForm1 = class(TForm)
    NMUDP1: TNMUDP;
    btnSend: TButton;
    Memo1: TMemo;
    StatusBar1: TStatusBar;
    GroupBox1: TGroupBox;
    btnLoad: TButton;
    OpenDialog1: TOpenDialog;
    SaveDialog1: TSaveDialog;
    btnSave: TButton;
    Label1: TLabel;
    Edit1: TEdit;              //UDP 包发送的目标地址
```

```
    Label2: TLabel;
    Edit2: TEdit;              //发送的目标端口
    Label3: TLabel;
    Edit3: TEdit;              //本地监听端口
    btnSet: TButton;
   procedure NMUDP1DataReceived(Sender: TComponent; NumberBytes: Integer;
     FromIP: String; Port: Integer);
   procedure btnSendClick(Sender: TObject);
   procedure FormCreate(Sender: TObject);
   procedure btnLoadClick(Sender: TObject);
   procedure btnSaveClick(Sender: TObject);
   procedure btnSetClick(Sender: TObject);
   procedure NMUDP1DataSend(Sender: TObject);
   procedure NMUDP1Status(Sender: TComponent; status: String);
  private
   { Private declarations }
  public
   { Public declarations }
  end;

var
  Form1: TForm1;
```

程序的设计界面如图 7-1 所示。

图 7-1　UDP 程序设计界面

在程序主窗体启动的时候进行程序的初始化工作，代码如下：

```
procedure TForm1.FormCreate(Sender: TObject);
begin
    {初始化界面}
    Edit1.Text:='';
    Edit2.Text:='';
    Edit3.Text:='';
    Memo1.Text:='';
```

```
    Memo1.ScrollBars:=ssBoth;
    StatusBar1.SimpleText:='Load Text File to Send';
    {设置 UDP 组件基本属性}
   NMUDP1.ReportLevel := Status_Basic;
end;
```

在程序实际工作之前需要对 NMUDP 组件进行一些设置工作，指定远程主机的 IP 地址和端口，并且指定本机监听的 UDP 端口，这样程序才能做到即能发送信息也能接收信息。具体实现代码如下：

```
procedure TForm1.btnSetClick(Sender: TObject);
begin
    {设置地址信息}
    NMUDP1.LocalPort:=StrToInt(Edit3.Text);
    NMUDP1.RemoteHost:=Edit1.Text;
    NMUDP1.RemotePort:=StrToInt(Edit2.Text);
end;
```

使用 btnLoad 按钮进行读取文本文件的工作，实现代码如下：

```
procedure TForm1.btnLoadClick(Sender: TObject);
var
    Ext:string;
begin
   if  OpenDialog1.Execute then
        begin
            Ext:=ExtractFileExt(OpenDialog1.FileName);
          if  Ext<>'.txt'  then
                begin
                   ShowMessage('Only  Txt  File  Accepted');
                     exit;
                end;
            Memo1.Lines.LoadFromFile(OpenDialog1.FileName);
        end;
end;
```

使用 btnSave 按钮保存接收到的文本文件，实现代码如下：

```
procedure TForm1.btnSaveClick(Sender: TObject);
begin
   if  SaveDialog1.Execute  then
       Memo1.Lines.SaveToFile(SaveDialog1.FileName);
end;
```

程序关键的地方在于发送文件，在 btnSend 按钮中实现，代码如下：

```
procedure TForm1.btnSendClick(Sender: TObject);
var
  txt:pchar;
```

```
begin
  {检查信息是否完整}
  if Memo1.Text='' then
     begin
          StatusBar1.SimpleText:='Nothing to send';
           exit;
     end;
  {一般一个包只能发送 1024 或者 2048 个字符}
  GetMem(txt,Length(Memo1.Text)+1);
  ZeroMemory(txt,Length(Memo1.Text)+1);
  txt:=PChar(Memo1.Text);
  {发送之前必须设置地址信息}
  NMUDP1.RemoteHost := Edit1.Text;
  NMUDP1.RemotePort := StrToInt(Edit2.Text);
  {发送 UDP 包}
  NMUDP1.SendBuffer(txt^,Length(Memo1.Text));
end;
```

如果 NMUDP 组件监听的端口有信息到达，会触发 OnDataReceived 事件，代码如下：

```
procedure TForm1.NMUDP1DataReceived(Sender: TComponent;
  NumberBytes: Integer; FromIP: String; Port: Integer);
var
  TXT: PChar;
  Count: Integer;
begin
  StatusBar1.SimpleText:='Receiving data';
  GetMem(txt,NUmberBytes+1);
  ZeroMemory(TXT,NumberBytes+1);
  NMUDP1.ReadBuffer(TXT^, Count);
  Memo1.Lines.Add(TXT);
  FreeMem(Txt);
  StatusBar1.SimpleText:='data Received ';
end;
```

在一些辅助性的信息提示事件中，更新状态条上信息显示代码如下：

```
procedure TForm1.NMUDP1DataSend(Sender: TObject);
begin
    StatusBar1.SimpleText:='Data  Send';
end;

procedure TForm1.NMUDP1Status(Sender: TComponent; status: String);
begin
     StatusBar1.SimpleText:=Status;
end;
```

程序的运行结果如图 7-2 所示，使用本地的端口接收本地发送的信息。Memo 组件中的第一行数据“ Data ”是原始数据，第二行中的“ Data ”是接收到的数据。

图 7-2　发送和接收 UDP 数据

7.1.2　使用 Winsock 函数发送 UDP 数据

在一些简单的场合，可以使用 Winsock 函数实现发送 UDP 数据。这样可以避免将整个 NMUDP 组件包含在程序中。在这个程序中主要实现 UDP 数据包的发送。程序代码参见光盘目录 ch7\Winsock UDP。

程序的窗体类以及 uses 字段的内容如下，注意需要包含 Winsock 库函数的支持：

```
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, Winsock, ComCtrls;

type
  TForm1 = class(TForm)
    Label1: TLabel;
    Edit1: TEdit;                    //发送的目标地址
    Label2: TLabel;
    Edit2: TEdit;                    //发送的目标端口
    Label3: TLabel;
    Memo1: TMemo;
    btnSend: TButton;
    StatusBar1: TStatusBar;
    procedure btnSendClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
    procedure SendUDP(IP:string;Port:integer;Content:string);
  end;

var
```

```
  Form1: TForm1;
```

在程序主窗体启动的时候进行 Winsock 的初始化工作：

```
procedure TForm1.FormCreate(Sender: TObject);
var
    WSAData:TWSAData;
begin
    {初始化}
    Edit1.Text:='';
    Edit2.Text:='';
    Memo1.Text:='';
    {Winsock 版本协商}
   if (WSAStartup(MAKEWORD(2,0),WSAData)<>0) then
    begin
        {初始化失败}
        StatusBar1.SimpleText:='Init Failed';
        exit;
    end
    else
        StatusBar1.SimpleText:='Init Success';
end;
```

另外实现一个发送 UDP 数据的函数，在这个函数中，首先是创建一个 SOCK_DGRAM 类型的 Socket 对象，然后使用 sendto 库函数向指定的 IP 地址的特定端口发送数据，实现代码如下：

```
procedure TForm1.SendUDP(IP: string; Port: integer; Content: string);
var
  addr : sockaddr_in;
  fd : integer;
   ErrCode:integer;
   pContent:PChar;
   Re:integer;
begin
   {创建一个 UDP socket}
  fd := socket(AF_INET,SOCK_DGRAM,0);
  if (fd = INVALID_SOCKET) then
   begin
      ErrCode:=WSAGetLastError();
     StatusBar1.SimpleText:='socket error,Error Code:'
                                +inttostr(ErrCode);
      exit;
   end;
   {准备内存空间}
   ZeroMemory(@addr,sizeof(sockaddr_in));
  addr.sin_family := AF_INET;
   {填充 IP 信息}
   addr.sin_addr.S_addr  := inet_addr(PChar(IP));
  addr.sin_port := Port;
```

```
  {准备发送的内容}
  GetMem(pContent,Length(Content)+1);
  ZeroMemory(pContent,Length(Content)+1);
  StrPCopy(pContent,Content);
  {发送信息}
  Re:=sendto(fd,
              pContent^,          //content
              Length(Content),  //length of content
             0,                  //flag set to zero MSG_DONTROUTE
              addr,               //addr to send
             sizeof(addr));     //size of addr
  {检测发送是否成功}
 if Re=SOCKET_ERROR then
    begin
        ErrCode:=WSAGetLastError();
        StatusBar1.SimpleText:='Error Code:'+
                                    Inttostr(ErrCode);
        FreeMem(pContent);
        Exit;
    end;
  {释放内存}
  FreeMem(pContent);
  {关闭 socket}
  closesocket(fd);
end;
```

在 btnSend **按钮中调用该函数，在调用函数之前首先进行一些基本的判断工作，代码如下：**

```
procedure TForm1.btnSendClick(Sender: TObject);
begin
    {检查信息是否完整}
   if Edit1.Text='' then
        begin
            ShowMessage('Set Target IP');
             exit;
        end;
   if Edit2.Text='' then
        begin
            ShowMessage('Set Target Port');
             exit;
        end;
    Try
        SendUDP(Edit1.Text,
                StrToInt(Edit2.Text),
                Memo1.Text);
    Except
       ShowMessage('Error on Send');
        Exit;
    end;
```

```
end;
```

在程序关闭的时候，还需要注销对 Winsock 的调用，具体代码如下：

```
procedure TForm1.FormDestroy(Sender: TObject);
begin
        {释放 Winsock}
        WSACleanUP();
end;
```

7.2　利用 UDP 协议进行网络广播

在网络上广播的信息理论上是所有机器都可以接收的。在正常的情况下，网络中总是在进行着广播信息的传送，例如某主机需要通过广播来表明自己目前使用的 IP 地址的唯一性等。在正常情况下，广播信息是不可以跨路由传送的。因为如果对这种信息如果不加以限制的话，将会引起信息风暴。路由器会主动丢弃这种 IP 包。因此本节的程序一般只能在同一个网关后面使用。

UDP 协议本身的最大特点就是本身是无连接的，只管信息的发送，不管信息接收。利用这种特性可以非常方便地实现网络广播。本节就利用 UDP 协议的这个特点实现一个利用广播机制工作的在线多人交谈工具。程序代码参见光盘目录 ch7\BroadCast。

程序的窗体类以及 uses 字段的内容如下：

```
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
   NMUDP, StdCtrls;

type
  TForm1 = class(TForm)
     NMUDPSend: TNMUDP;
     Button1: TButton;
     Memo1: TMemo;
     Label1: TLabel;
     Label2: TLabel;
     Edit2: TEdit;                    //指定发送的信息内容
     btnSay: TButton;
     GroupBox1: TGroupBox;
     Edit1: TEdit;                    //指定发送地址
     tbnSet: TButton;
     Label4: TLabel;
     Label5: TLabel;
     Edit3: TEdit;                    //指定发送的目标端口
     Label3: TLabel;
     Edit4: TEdit;                    //指定本地监听端口
     Label6: TLabel;
    procedure btnSayClick(Sender: TObject);
```

```
    procedure tbnSetClick(Sender: TObject);
    procedure NMUDPSendStatus(Sender: TComponent; status: String);
    procedure FormCreate(Sender: TObject);
    procedure NMUDPSendDataReceived(Sender: TComponent;
      NumberBytes: Integer; FromIP: String; Port: Integer);
    procedure NMUDPSendBufferInvalid(var handled: Boolean;
      var Buff: array of Char; var length: Integer);
    procedure NMUDPSendDataSend(Sender: TObject);
    procedure NMUDPSendInvalidHost(var handled: Boolean);
    procedure NMUDPSendStreamInvalid(var handled: Boolean;
      Stream: TStream);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
```

程序界面如图 7-3 所示。

图 7-3　广播程序的设计界面

在这个程序中，通过一些 Edit 组件让使用者设置好本局域网的广播地址。另外就是设置广播使用的端口号码。端口号码非常重要，这就类似于聊天室里的房间，使用不同的端口可以接收不同的广播信息。另外就是设置本地监听的端口号码。如果设置错误，就会接收不到广播的信号。

在程序主窗体启动的时候进行程序的初始化工作，代码如下：

```
procedure TForm1.FormCreate(Sender: TObject);
begin
    //set view
     Memo1.Clear;
    //IP info
     //broadcast
     Edit1.Text:='10.13.101.255';
     Edit3.Text:='6666';
```

```
    Edit4.Text:='6666';
    //init
   // this dont work at all
    NMUDPSend.ReportLevel := Status_Basic;
    NMUDPSend.LocalPort:=StrToInt(Edit4.Text);
    NMUDPSend.RemoteHost:=Edit1.Text;
    NMUDPSend.RemotePort:=StrToInt(Edit3.Text);
end;
```

在程序的使用过程中，最重要的就是进行设置工作，代码如下：

```
procedure TForm1.tbnSetClick(Sender: TObject);
begin
      {选择聊天频道}
      NMUDPSend.RemoteHost:=edit1.Text;
      NMUDPSend.RemotePort:=StrToInt(Edit3.Text);
      NMUDPSend.LocalPort:=StrToInt(Edit4.Text);
end;
```

设置完成之后就可以进行信息的发送工作了，具体代码如下：

```
procedure TForm1.btnSayClick(Sender: TObject);
var
   data:TMemoryStream;
   buf:Pchar;
   size:integer;
begin
      {申请内存空间}
      size:=length(Edit2.Text);
      GetMem(buf,size+1);
      FillChar(buf^,size,#0);
      {填充数据}
      StrPCopy(buf,Edit2.Text);
      {填充流}
      data:=TMemoryStream.Create;
      data.Write(buf^,size);
      {设置目标 IP}
      NMUDPSend.RemoteHost:=Edit1.Text;
      NMUDPSend.RemotePort:=StrToInt(Edit3.Text);
      {发送流信息}
      NMUDPSend.SendStream(data);
      {释放内存}
      FreeMem(buf);
      data.Free;
end;
```

如果发送信息成功，会触发 OnDataSend 事件，代码如下：

```
procedure TForm1.NMUDPSendDataSend(Sender: TObject);
begin
```

```
    Self.Caption:='Data  send'
end;
```

当有信息到达本地端口的时候，NMUDP **组件会触发** OnDataReceived **事件，可以进行信息的读取工作，实现代码如下：**

```
procedure TForm1.NMUDPSendDataReceived(Sender: TComponent;
  NumberBytes: Integer; FromIP: String; Port: Integer);
var
    data:TMemoryStream;
    buf:pchar;
begin
      {更新显示}
      Self.Caption:='Receive  data';
      {准备流}
      data:=TMemoryStream.Create;
      {读取流信息}
      NMUDPSend.ReadStream(data);
      {准备内存空间}
      getmem(buf,NumberBytes+1);
      FillChar(buf^,NumberBytes+1,0);
      data.Read(buf^,NumberBytes);
      {显示接收的内容}
      memo1.Lines.Add(FromIP+' : '+buf);
      {清空所有申请的内存}
      data.Free;
      freemem(buf);
end;
```

在 NMUDP **组件的使用过程中，还有很多的中间状态通过事件提示，具体代码如下：**

```
procedure TForm1.NMUDPSendBufferInvalid(var handled: Boolean;
  var Buff: array of Char; var length: Integer);
begin
    {使用的内存空间有问题}
    ShowMessage('Buffer  Invalid');
end;

procedure TForm1.NMUDPSendInvalidHost(var handled: Boolean);
begin
    {不是合理的主机名}
    ShowMessage('Host  unKnow');
end;

procedure TForm1.NMUDPSendStreamInvalid(var handled: Boolean;
  Stream: TStream);
begin
    {使用的流有问题}
    ShowMessage('Message  Stream  InValid');
end;
```

```
procedure TForm1.NMUDPSendStatus(Sender: TComponent; status: String);
begin
     //show  status
      caption:=status;
end;
```

程序运行的结果如图 7-4 所示。

图 7-4　UDP 在线交谈程序

> **可以将这个交谈工具和前面使用 TCP 协议实现的在线交谈工具进行一个比较。在使用 TCP 协议的时候，需要一个服务器来中转所有的客户端程序的信息，每个人也可以非常清楚地知道有多少人在线交谈。而在这个程序中，所有的程序自己管理信息的发送和接收，对于信息的接收方到底有多少是完全不了解的，如果中途由于网络原因导致信息遗失也是无法弥补的。另外，使用这种机制其安全性也是无法保障的，如果有人发现了广播的端口并进行监听，其他人根本就无法发现他的存在。大家在选择使用何种协议的时候可以综合考虑。**

关于广播使用的地址，在 IP 协议中指定了如下三类广播地址：

- **本地局域网内广播，地址是全 1，即 “ 255.255.255.255 ”；**
- **对直接相连的局域网发广播，地址是 A、B 或者 C 类地址主机部分全是 1；**
- **子网广播地址，其地址是子网地址主机部分全是 1。**

不过，Berkeley 在 BSD UNIX 实现 TCP/IP 的时候，使用的是非标准广播地址，也称为 Berkeley 广播地址，它是用全 0 代替全 1。

第 8 章　ICMP 协议相关网络应用

ICMP（Internet Control Message Protocl，网际控制报文协议）是和 IP 协议同一层次的协议，对 Internet 以及各种 IP 网络的正常运转起到至关重要的作用。Ping 就是最常用的 ICMP 应用，该指令向对方主机的 IP 堆栈发出一个 ICMP 请求报文，并等待该主机返回 ICMP 应答报文，从而检测该主机的状态。当然，有的主机使用防火墙关闭 ICMP，在这种情况下，Ping 指令不会获得该主机的应答报文。

Windows 系统的 ICMP 工作（需要用 RAW SOCKET），需要通过 ICMP.dll 动态链接库实现（在 Windows 2000 中开始支持 RAW SOCKET）。

8.1　ping 指令程序实现

ping 是最常用的网络状态判断指令，几乎所有支持网络的操作系统（无论是 Unix 还是 Windows）都包含该指令。

在使用 ping 指令的时候，需要知道对方主机的 IP 地址。如果在 ping 指令的返回字符中有“Reply from”字样，说明对方主机在网上，如果出现“Request timeout”字样，说明对方主机不在网上，如图 8-1 所示。

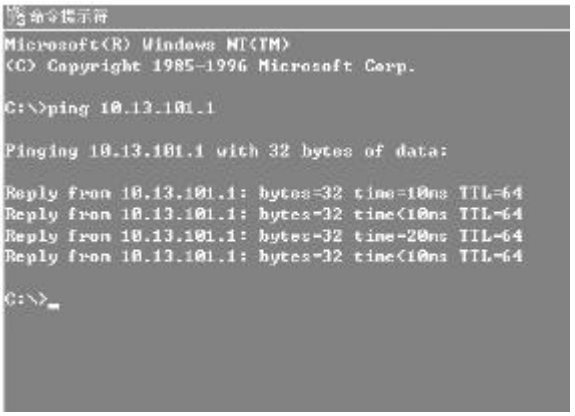

图 8-1　使用 ping 来判断网络情况

以下介绍 ping 的程序实现。这里的实现全部都建立在系统自带的 ICMP.dll 动态链接库基础之上。程序代码参见光盘目录 ch8\ping。

程序的窗口类的内容如下：

```
type
  TFormPing = class(TForm)
    EditAddr: TEdit97;              //指定需要检测的主机地址
```

```
    Label1: TLabel;
    OutlookBtnPing: TOutlookBtn;
    Label2: TLabel;
    MemoResult: TMemo;          //显示检测的结果
   procedure OutlookBtnPingClick(Sender: TObject);
   procedure FormCreate(Sender: TObject);
  private
   { Private declarations }
  public
   { Public declarations }
  end;

var
  FormPing: TFormPing;
   hICMP:THandle;                    //打开的 icmp 句柄
```

程序的界面如图 8-2 所示。

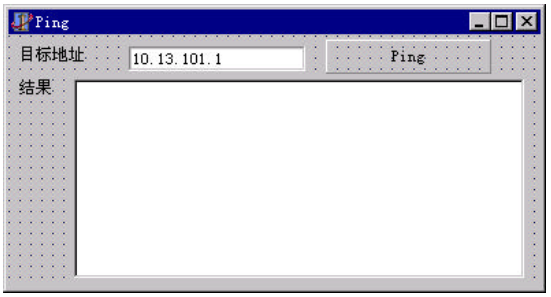

图 8-2　ping 程序的设计界面

程序窗口类非常简单，首先是用一个 Edit 组件让使用者指定需要检测的主机 IP 地址，有一个按钮进行 ping 的操作，在窗体的下部放置一个 Memo 组件，用于显示最终的指令运行结果。在程序中还有一个全局变量 hICMP 用于储存打开的 ICMP 句柄。

为了使用 ICMP.dll 动态链接库的内部函数，必须自己定义一些常量、结构，并且需要自己声明所有的函数。Delphi 没有带支持 ICMP 协议的库文件，这些常数、结构和函数都可以在相应的 C 头文件中找到。

使用 ICMP 需要以下的常数、结构和函数的定义（包括 uses 字段的内容，因为在一些定义的结构中使用了 Winsock 或者是 Windows 库中的变量类型），具体代码如下：

```
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  OutlookBtn, StdCtrls, TB97, Winsock;

type

DWORD=LongWord;          //定义 ICMP 数据类型
THandle=LongWord;
PIPOptionInformation = ^TIPOptionInformation;
TIPOptionInformation =
   record
```

```
            TTL: Byte;
            TOS: Byte;
            Flags: Byte;
            OptionsSize: Byte;
            OptionsData: PChar;
    end;

  PIcmpEchoReply = ^TIcmpEchoReply;
  TIcmpEchoReply =
    record
        Address: DWORD;
        Status: DWORD;
        RTT: DWORD;
        DataSize:Word;
        Reserved: Word;
        Data: Pointer;
        Options: TIPOptionInformation;
    end;
  {声明 ICMP 相关函数}
  function IcmpCreateFile():THandle;stdcall external 'ICMP.dll';
  function IcmpCloseHandle(Handle:THandle):Boolean;stdcall external 'ICMP.dll';
  function IcmpSendEcho(Handle:THandle;DestAddr:DWORD;
        RequestData: Pointer;RequestSize: Word;RequestOptions: PIPOptionInformation;
        ReplyBuffer: Pointer;ReplySize: DWORD;Timeout: DWORD):
        DWORD;stdcall external 'ICMP.dll';
  {两个辅助函数}
  procedure ValidCheck();
  procedure FreeWinsock();
  {关键的 ping 实现函数}
  function Ping(IPAddr:String;TimeOut:Word):String;
  {自定义的几个常量}
  Const
  { Exception Message }
  SInitFailed   = 'Winsock version error';
  SInvalidAddr  = 'Invalid IP Address';
  SNoResponse   = 'No Response';
  STimeOut      = 'Request TimeOut';
```

其中 ValidCheck **函数用于检测使用的** Winsock **库函数的版本。并且在这个函数中进行** ICMP **句柄的创建工作。在这个函数中，如果内部的函数操作没有返回需要的结果，就会主动触发一个异常，提示使用者程序初始化失败。具体代码如下：**

```
  procedure ValidCheck();
  var
     WSAData:TWSAData;
  begin
       {初始化 Winsock dll}
      if (WSAStartup(MAKEWORD(2,0),WSAData)<>0) then
          raise Exception.Create(SInitFailed);
       {创建 icmp 句柄}
```

```
        hIcmp:=IcmpCreateFile();
        if hICMP=INVALID_HANDLE_VALUE then
            raise Exception.Create('Create ICMP Failed');
end;
```

在 FreeWinsock **函数中并不仅仅是释放使用的** Winsock **引用，还需要关闭打开的** ICMP **句柄，释放占用的系统资源。具体代码如下：**

```
procedure FreeWinsock();
begin
        IcmpCloseHandle(hIcmp);          //关闭 icmp 句柄
        WSACleanUP;
end;
```

在 ping **函数中，就需要进行** ping **功能的具体操作了。这个函数需要两个执行参数，一个是目标主机的** IP **地址，另外一个是超时的设置。因为目标主机很可能没有开机或者网络出现故障信息包无法正常传递，因此设置一个合理的超时时间可以避免程序始终处于悬挂等待状态。在这个函数中，需要是进行** ICMP **请求报文的准备工作和** ICMP **应答报文的处理。通过对应答报文的处理可以了解很多的网络信息。**

```
function Ping(IPAddr:String;TimeOut:Word):String;
var
        IPOpt:TIPOptionInformation;          // 需要发送的 IP Options 数据包
        FIPAddress:DWORD;
        pReqData,pRevData:PChar;
        pIPE:PIcmpEchoReply;                 // ICMP 回应报文
        FSize: DWORD;
        MyString:string;
        FTimeOut:DWORD;
        BufferSize:DWORD;
        temp:Integer;
        pIPAddr:Pchar;
begin

          {获取需要检测的 IP}
          GetMem(pIPAddr,Length(IPAddr)+1);
          ZeroMemory(pIPAddr,Length(IPAddr)+1);
          StrPCopy(pIPAddr,IPAddr);

          {获取实际地址}
          FIPAddress := inet_addr(pIPAddr);

          {释放中间变量}
          FreeMem(pIPAddr);

          {检测是否是合法的地址}
          if FIPAddress=INADDR_NONE then
           begin
                result:=SInvalidAddr;//Exit
```