第十八讲 网络编程技术

吕松茂
lsmtech@163.com

内容提要

- WinSock基础
- 网络聊天程序的实现

内容提要

- **WinSock基础**
  - 网络聊天程序的实现

一、WinSock基础

1.1 概述

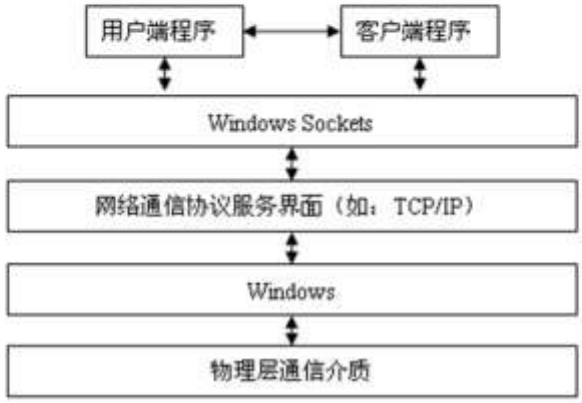
WinSock是网络编程接口，而不是协议，它构成了Windows平台下进行网络编程的基础。Delphi中各种网络组件的强大功能，都是建立在WinSock API基础之上的。

一、WinSock基础

Socket 是建立在传输层协议(主要是TCP 和UDP)上的一种套接字规范，它定义两台计算机间进行通信的规范(即一种编程规范)，如果说两台计算机是利用一个“通道”进行通信，那么这个“通道”的两端就是两个套接字。套接字屏蔽了底层通信软件和具体操作系统的差异，使得任何两台安装了TCP 协议软件和实现了套接字规范的计算机之间的通信成为可能。

一、WinSock基础

在Delphi 中，其底层的Socket 也应该是Windows的Socket。Socket减轻了编写计算机间通信软件的难度。Delphi中对Windows Socket进行了有效的封装，使用户可以很方便地编写网络通信程序。



1.2 TCP、UDP和IP协议

对应于OSI七层模型，TCP和UDP协议是位于传输层的协议，而IP则位于网络层。如图所示：

一、WinSock基础

OSI模型和网际协议族

TCP是传输控制协议，它是一种面向连接的协议，它向用户提供可靠的全双工的字节流。TCP关心确认、超时和重传等具体细节。大多数Internet应用程序使用TCP，因为它是一种精致的、可靠的字节流协议。

UDP是用户数据报协议，它是一种无连接协议。UDP是一种简单的，不可靠的数据报协议，与TCP不同，UDP不能保证每一个UDP数据报可以到达目的。

IP是网际协议。Ipv4(我们通常就称之为IP)自80年代早期以来一直是网际协议族的主力协议，它使用32位地址，为TCP、UDP、ICMP和IGMP提供递送分组的服务。90年代中期，又设计出了用以替代Ipv4的Ipv6，它使用128位的大地址。

TCP/IP协议核心与应用程序关系图

1.3 套接口（Socket）和Winsock API

套接口（Socket）最初是由加利福尼亚大学Berkeley学院为UNIX操作系统开发的网络通信编程接口。

一、WinSock基础

90年代初，由Microsoft、Sun Microsystems等几家公司共同参与制定了一套标准，即Windows Sockets规范。1993年，他们制定了Windows Sockets1.1规范，定义了16位Windows平台下的网络标准编程接口。

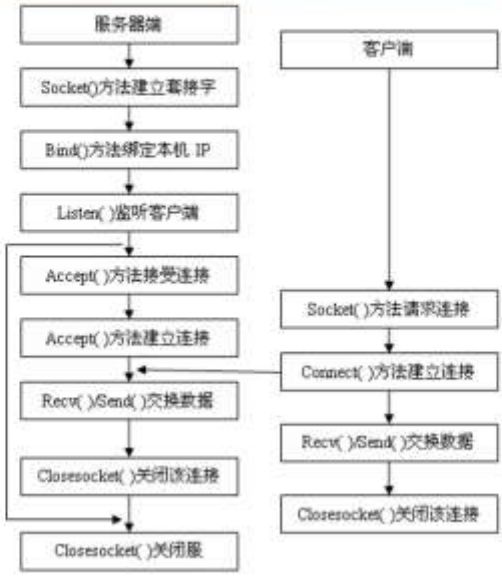
对应于OSI七层参考模型，WinSock API是位于会话层和传输层之间的，它提供了一种可为指定的传输协议(如TCP、UDP)打开、进行和关闭会话的能力。Windows Sockets规范已经成为在Windows平台上开发网络应用程序的已接受标准，它也为想开发网络程序的程序员提供了巨大的方便。

1.4 面向连接和无连接

面向连接发服务中，进行数据交换之前，通信双方必须建立一条用以进行通讯的路径。这样既确定了通讯双方之间的联系，又可以保证双方都处于活动状态，可以彼此响应。多数情况下，面向连接的协议可以保证传输数据的可靠性。

TCP协议即是一种面向连接的协议。应用程序利用TCP进行通讯时，发起方和接受方之间会建立一个虚拟连接，通过这一连接，双方可以把数据当作一个双向的字节流来进行交换。

面向连接的套接字的系统调用时序



无连接服务不管目的方是否处于待接收状态，源方只管将信息发送给目的方，也不管目的方是否已经接受到了该信息，以及接收到的信息是否无误。

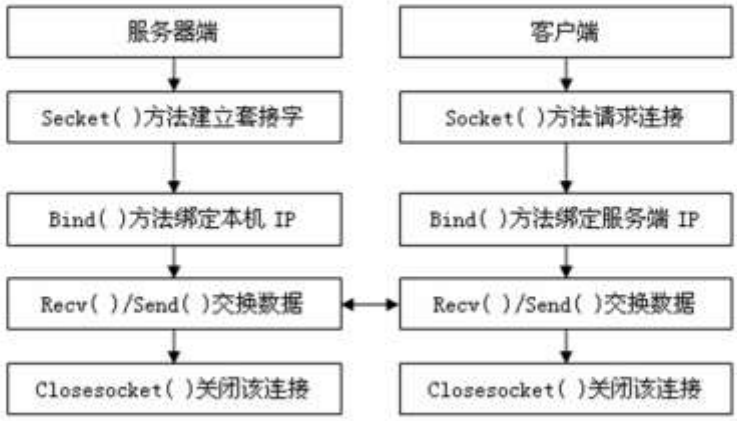
UDP协议即是一种无连接协议，数据传输方法采用的是数据报。

面向连接服务和无连接服务各有优缺点。

面向连接的服务能够保证通讯双方传递数据的正确性，但它却要为此进行额外的校验，同时，通讯双方建立通信信道也需要许多系统开销。

无连接服务最大的优点就是速度快，因为它不用去验证数据完整性，也不为数据是否已经被接收操心。

无连接协议的套接字调用时序



1.5 客户/服务器模式

当今网络应用中，通信双方最常见的交互模式便是客户/服务器模式。在这种模式中，客户向服务器发出服务请求，服务器收到请求后为客户提供相应的服务。

客户/服务器模式通常采用监听/连接方式实现。一个服务器端的应用程序通常在一个端口监听对服务的请求，也就是说，服务进程一直处于休眠状态，直到一个客户对这个服务提出了连接请求。此时，服务进行被“唤醒”并且为客户提供服务，即对客户的请求作出适当的反应。

1.6 套接口类型

在使用TCP/IP协议时，可选的套接口类型有三种：流式套接口、数据报套接口及原始套接口。

流式套接口定义了一种可靠的面向连接的服务，实现了无差错无重复的顺序数据传输。对于建立在这种类型套接口上的套接字来说，数据是可以双向传输的字节流，无长度限制。

数据报套接口定义了一种无连接的服务，数据通过相互独立的报文进行传输，是无序的，并且不保证可靠、无差错。即一个建立在数据报套接口上的套接字所接收的信息有可能重复，或者和发出时的顺序不同。

一、WinSock基础

原始套接口允许对底层协议如IP或ICMP直接访问，主要用于新网络协议实现的测试。目前，只有WinSock 2提供了对它的支持。

1.7 使用面向连接的协议时套接口的调用

客户/服务器模式的网络应用程序通常使用面向连接的协议。

采用面向连接的协议（如TCP）时，服务器处理请求往往比较复杂，不是一来一去的简单请求应答所能解决的，往往要经过反复的交互。大多数TCP服务器是并发服务器。

无连接客户端更为简单，只需要调用socket（）建立一个套接字c，就可以用sendto（）和recvfrom（）函数进行与服务器端的数据交换了。在完成会话后，调用closesocket（）函数关闭套接字c。

内容提要

- WinSock基础
- 网络聊天程序的实现

二、网络聊天程序的实现

2.1 使用TCP协议

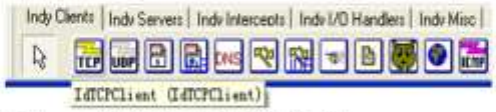
由于网络的不稳定及经常丢失数等原因，在Internet上实现两台机器间的联系需要使用TCP。基于TCP/IP协议的传输是面向连接的点到点的传输。在聊天时，用户首先必须设置本地机器的端口号，打开本地TCP服务器，然后向远程主机发送连接请求，建立虚路连接，才能开始数据的传送与接受。

二、网络聊天程序的实现

TIdTCPClient、TIdTCPServer组件的用法

1. TIdTCPClient组件

TIdTCPClient组件位于Indy Clients组件板如图所示，TIdTCPClient组件客户方通信管理，封装了完整的Socket。TIdTCPClient组件是许多Indy客户端组件的基类 。



(1)TIdTCPClient组件主要属性介绍

❁BoundIP属性：用来指定客户端连接的本地IP地址

二、网络聊天程序的实现

- BoundPort属性：用来指定客户端连接的本地IP地址，在Connect方法中指定绑定端口号
- Host属性：用于标识远程计算机地址,可以是IP地址，也可以是计算机名
- Port属性：用于表示远程计算机端口，和Host属性一起用于指明远程计算机地址
- ReadTimeout属性：用来指明读数据时的连接超时毫秒数
- RecvBufferSize属性：指明了用来接收数据的缓冲区字节大小，默认值为8192。
- SendBufferSize属性：指明了写数据到连接缓冲区时允许的最大字节数，默认值为32768
- LocalName属性：指明本地计算机名称。

二、网络聊天程序的实现

(2)TIdTCPClient组件的过程和方法介绍

- Connect

  形式：procedure Connect(const ATimeout:integer);

  含义：用来向服务器请求建立一个连接

- ConnectAndGetAll

  形式：function ConnectAndGetAll():sting;

  含义：用于连接到Host属性和Port属性指定的远程计算机并从服务器中读取所有数据

- Connected

  形式：function Connected(): Boolean;

  含义：用来指明到对方计算机的连接是否已经建立

二、网络聊天程序的实现

- OpenWriteBuffer
  形式：procedure OpenWriteBuffer(const AThreshhold:Integer);
  含义：打开写缓冲区
- CloseWriteBuffer
  形式：procedure CloseWriteBuffer();
  含义：关闭写缓冲区
- Disconnect
  形式：procedure Disconnect();
  含义：用于断开与对方计算机的连接

二、网络聊天程序的实现

❁FlushWriteBuffer

形式：procedure FlushWriteBuffer(const AByteCount: Integer);

含义：发送自调用OpenWriteBuffer以来的所有缓冲区数据到对方计算机，并随后清空缓冲区

❁ReadBuffer

形式：procedure ReadBuffer(var ABuffer; constAByteCount:Longint);

含义：从Indy的接收缓冲区中读取的字节数

二、网络聊天程序的实现

- ReadStream

  形式：procedure ReadStream (AStream:TStream; AByteCount:LongInt;const AreadUntil Disconnect :Boolean);

  含义：从Indy缓冲中读取数据并存储在AStream参数指定的目的流中

- WriteBuffer

  形式：procedure WriteBuffer(const Abuffer; AByteCount:LongInt;const AwriteNow:Boolean)

  含义：用于向对方计算机发送数据，数据被写往Indy缓冲区或者直接发送给对方

- WriteStream

  形式：Procedure WriteStream(AStream:Tstream; const AAll:Boolean;const AWriteByteCount: Boolean; const ASize:Integer);

  含义：用于发送一个对象流

二、网络聊天程序的实现

(3)TIdTCPClient组件的事件介绍

❁OnStatus事件：是一个TidSatusEvent类型的时间句柄，在当前连接状态改变时被触发。

二、网络聊天程序的实现

2. TIdTCPServer组件

TIdTCPServer组件位于Indy Servers组件板如图所示，TIdTCPServer组件封装了一个完整的多线程TCP服务器。TIdTCPServer使用一个或多个线程来监听客户连接。通过TidThreadMgr对象的关联，为每一个连接到服务器的客户链接分配一个独立的线程。TIdTCPServer允许配置服务器监听线程，包括默认端口、监听队列、最大连接数等。

二、网络聊天程序的实现

TIdTCPServer组件实现了两套机制来链接线程提供服务。第一种方法利用响应事件句柄的方法来处理客户链接。第二种方法使用TidCommandHandler对象来辨认合法的服务命令，提供一些方法和属性来处理参数，执行动作，表述正确或者错误的响应。

(1)TIdTCPServer组件的属性介绍

- Active属性：用于指出TIdTCPServer的当前状态
- Bindings属性：为TCP服务器提供默认端口号，并被TidListenerThread 对象用来获取对Socket句柄的访问和由TCP/IP协议栈提供的底层方法

二、网络聊天程序的实现

- DefaultPort属性：指明了其监听新的客户端链接请求的端口号
- Greeting属性：当客户端链接请求被监听线程接收时，Greeting中包含了被发往客户端的欢迎信息
- ListenQueue属性：用来为监听线程指明可允许的、未处理的链接请求数目
- Threads属性：包含了在监听线程中创建的线程列表
- LocalName属性：标识了用户计算机系统的主机名

二、网络聊天程序的实现

(2)TIdTCPServer组件的过程和方法介绍

❁BeginWork

形式：Procedure BeginWork (AWorkMode:TWorkMode;const Size:Integer);

方法：用于触发OnBeginWork事件，同时维护读/写堵塞操作的数量，以及初始读写操作的大小信息

❁DoWork

形式：procedure DoWork (AWorkMode:TWorkMode;const ACount:Integer);

方法：用于触发OnWork事件。在调用DoWork过程之前必须调用BeginWork过程，否则DoWork过程将不产生任何效果。

二、网络聊天程序的实现

- EndWork

形式：procedure EndWork(AWorkMode:TWorkMode);

方法：EndWork过程用于触发OnEndWork事件。EndWork上可以嵌套调用，但是on EndWork事件仅在第一次调用时触发

二、网络聊天程序的实现

(3)TIdTCPCServer组件的事件句柄介绍

- OnConnect事件：在客户线程试图连接到TCP服务器时触发
- OnDisconnect事件：在客户线程试图断开与TCP服务器时触发
- OnExecute事件：当客户线程试图执行TidPeerThread.Run方法时触发
- OnStatus事件：在当前连接的状态改变时被触发

2.2 Delphi编写WinSock程序步骤

1. TClientSocket 控件和TCerverSocket 控件的属性设置

- 设置ServerSocket 控件的Port 属性
- 设置Active 属性为True
- 设置ClientServer 控件中Port 属性应设置成和服务器端的Port 属性值一致
- 设置Host 属性

二、网络聊天程序的实现

2．建立与远程计算机的连接

- 要在远程计算机系统之间进行数据传输，首先必须在通信的两台主机之间建立连接。
- 服务器端的 ServerSocket 控件调用Open 方法初始化 Socket 连接，同时也就设置了Active 属性为True，将 ServerSocket 控件设置成监听模式，随时侦测是否有连接请求。
- 如果服务器接受了客户程序的连接请求，则触发 OnAccept 事件
- 在客户端程序中，ClientSocket 控件设置Port、Host 等必须的属性，然后设置Active属性为True，提出连接请求。

二、网络聊天程序的实现

3. 计算机之间数据传输

❁一旦服务器端接受了客户机方面的连接请求，客户机就可以发送数据。这时在客户机和服务器之间就拥有了一个Socket，通过此Socket 实现双方通信。

❁客户机端用如下形式发送数据：

```
ClientSocket1.socket.sendtext('string you want to send');
```

❁服务器端采用如下形式接收数据：

```
ServerSocket1.socket.recievetext( str: string);
```

❁此函数返回接收到的字符串的长度，将字符串存储在变量str中。上述是数据传输的最简单的例子，还可以采用Socket 属性所提供的其他方法来实现复杂的数据传输。

二、网络聊天程序的实现

4. 关闭与远程计算机的连接

- 数据传输完毕，最后要关闭Socket 连接，方法如下。
- 关闭某一个Socket 连接：ClientSocket1.close 或 ClientSocket1.active:=false;
- 关闭所有的Socket 连接：ServerSocket1.close 或 ServerSocket1.active:=false;

总结

- WinSock基础
- 网络聊天程序的实现

谢谢大家