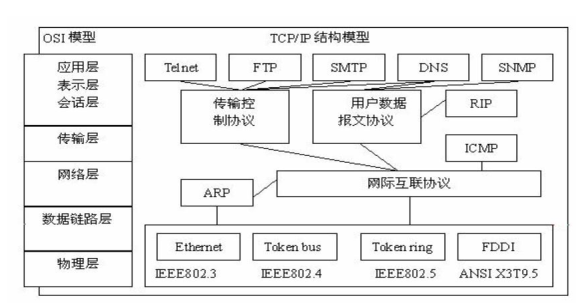
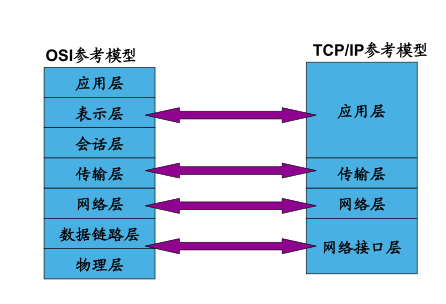
# linux网络程序应用

## 网络模型

计算机网络是指由通信线路互相连接的许多自主工作的计算机构成的集合体，各个部件之间以何种规则进行通信，就是网络模型研究的问题。网络模型一般是指OSI七层参考模型和TCP/IP四层参考模型。这两个模型在网络中应用最为广泛。OSI（Open System Interconnect），即[开放式系统](http://baike.baidu.com/item/%E5%BC%80%E6%94%BE%E5%BC%8F%E7%B3%BB%E7%BB%9F)互联。 一般都叫OSI参考模型，是ISO（[国际标准化组织](http://baike.baidu.com/item/%E5%9B%BD%E9%99%85%E6%A0%87%E5%87%86%E5%8C%96%E7%BB%84%E7%BB%87)）组织在1985年研究的[网络互联](http://baike.baidu.com/item/%E7%BD%91%E7%BB%9C%E4%BA%92%E8%81%94)模型。该[体系结构](http://baike.baidu.com/item/%E4%BD%93%E7%B3%BB%E7%BB%93%E6%9E%84)标准定义了网络互连的七层框架（[物理层](http://baike.baidu.com/item/%E7%89%A9%E7%90%86%E5%B1%82)、[数据链路层](http://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E9%93%BE%E8%B7%AF%E5%B1%82)、[网络层](http://baike.baidu.com/item/%E7%BD%91%E7%BB%9C%E5%B1%82)、[传输层](http://baike.baidu.com/item/%E4%BC%A0%E8%BE%93%E5%B1%82)、[会话层](http://baike.baidu.com/item/%E4%BC%9A%E8%AF%9D%E5%B1%82)、[表示层](http://baike.baidu.com/item/%E8%A1%A8%E7%A4%BA%E5%B1%82)和[应用层](http://baike.baidu.com/item/%E5%BA%94%E7%94%A8%E5%B1%82)），即ISO[开放系统互连参考模型](http://baike.baidu.com/item/%E5%BC%80%E6%94%BE%E7%B3%BB%E7%BB%9F%E4%BA%92%E8%BF%9E%E5%8F%82%E8%80%83%E6%A8%A1%E5%9E%8B)。在这一框架下进一步详细规定了每一层的功能，以实现[开放系统](http://baike.baidu.com/item/%E5%BC%80%E6%94%BE%E7%B3%BB%E7%BB%9F)环境中的互连性、[互操作性](http://baike.baidu.com/item/%E4%BA%92%E6%93%8D%E4%BD%9C%E6%80%A7)和应用的可移植性。其中　**OSI参考模型并没有提供一个可以实现的方法，而是描述了一些概念，用来协调进程间通信标准的制定。**即OSI参考模型并不是一个标准，而是一个在制定标准时所使用的概念性框架。TCP/IP参考模型是[计算机网络](http://baike.baidu.com/item/%E8%AE%A1%E7%AE%97%E6%9C%BA%E7%BD%91%E7%BB%9C/18763)的祖父[ARPANET](http://baike.baidu.com/item/ARPANET)和其后继的因特网使用的[参考模型](http://baike.baidu.com/item/%E5%8F%82%E8%80%83%E6%A8%A1%E5%9E%8B)。ARPANET是由[美国国防部](http://baike.baidu.com/item/%E7%BE%8E%E5%9B%BD%E5%9B%BD%E9%98%B2%E9%83%A8)DoD（U.S.Department of Defense）赞助的研究网络。逐渐地它通过租用的[电话线](http://baike.baidu.com/item/%E7%94%B5%E8%AF%9D%E7%BA%BF)连结了数百所大学和政府部门。当[无线网络](http://baike.baidu.com/item/%E6%97%A0%E7%BA%BF%E7%BD%91%E7%BB%9C)和[卫星](http://baike.baidu.com/item/%E5%8D%AB%E6%98%9F/21511)出现以后，现有的协议在和它们相连的时候出现了问题，所以需要一种新的参考[体系结构](http://baike.baidu.com/item/%E4%BD%93%E7%B3%BB%E7%BB%93%E6%9E%84)。这个体系结构在它的两个主要协议出现以后，被称为TCP/IP参考模型（TCP/IP reference model）。



由图可见，TCP/IP模型包括4层：

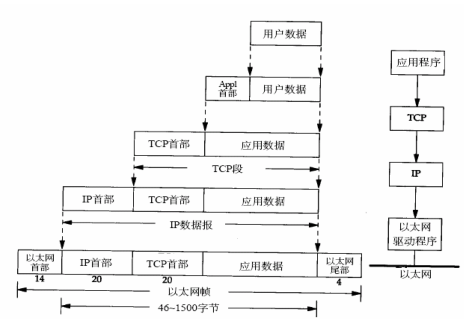
　　网络接口层--对应OSI参考模型的物理层和数据链路层；

　　网络层--对应OSI参考模型的网络层；

　　运输层--对应OSI参考模型的运输层；

　　应用层--对应OSI参考模型的5、6、7层。

### 数据封装



### TCP/IP协议族

TCP/IP实际上是一个协同工作的通信家族，为网络数据通信提供通路。大体分为三部分：

1. Internet协议（IP）
2. 传输控制协议（tcp）和用户数据报协议（UDP）
3. 处于TCP和UDP之上的一组应用协议。

#### 网络层

第一部分称为网络层。主要包括Internet协议（IP）、网际控制报文协议（ICMP）和地址解析协议（ARP）:

Internet协议（IP）

该协议被设计成互联网分组交换通信网，以形成一个网际通信环境。他负责在源主机和目的主机之间传输来自较高层软件的称为数据报文的数据块，它在源和目的地之间提供非连接型传递服务。

网际控制报文协议（ICMP）

它实际上不是IP层部分，但直接同IP层一起工作，报告网络上的某些出错情况。允许网际路由器传输差错信息或者测试报文。

地址解析协议（ARP）

ARP实际上不是网络层部分，它处于IP和数据链路层之间，它是在32位IP地址和48位物理地址之间执行翻译的协议。

#### 传输层协议

第二部分包括是传输层协议，包括数据传输控制协议和用户数据报文协议。

#### 传输控制协议（TCP）

该协议对建立网络上用户进程之间的对话负责，它确保进程之间的可靠通信，所提高的功能如下：

1. 监听输入对话建立请求
2. 请求另一网络站点对话
3. 可靠的发送和接收数据
4. 适度的关闭对话

用户数据报文协议（UDP）

UDP提供不可靠的非连接性传输层服务，它允许在源和目的之间传输数据，而不必在传送数据之前建立对话。它主要用于那些非连接性的应用程序，如：视频点播。

第三部分主要包括Telent，文件传送协议（FTP和TFTP），简单文件传送协议（SMTP）和域名服务器（DNS）等协议。

#### IP协议

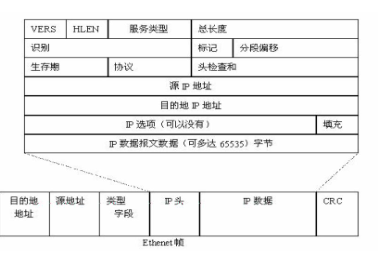
IP主要由以下四个主要功能

1. 数据传送
2. 寻址
3. 路由选择
4. 数据报文的分段

IP的主要目的是为数据输入/输出网络提供基本算法，为高层协议提供无连接的传送服务。这意味着在IP将数据递交给接收站点以前不在传输站点和接收站点之间建立对话。它只是封装和传递数据，但不向发送者或接收者报告包的状态，不处理所遇到的故障。

IP包由IP协议头和协议数据两部分构成。

IP协议头



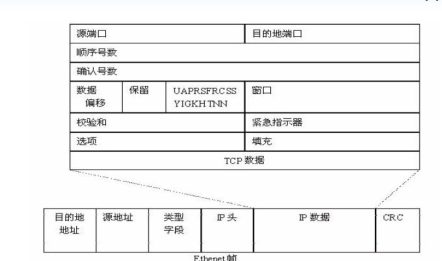
#### TCP协议

TCP是重要的传输层协议，目的是允许数据同网络上的其他节点进行可靠的交换。它能提供端口编号的译码，以识别主机的应用程序，而且完成数据的可靠传输。

•TCP协议具有严格的内装差错检验算法确保数据的完整性。

• TCP是面向字节的顺序协议，这意味着包内的每个字节被分配一个顺序编号，并分配给每包一个顺序编号。

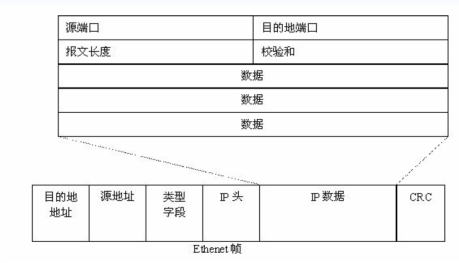
TCP协议头



#### UDP协议

UDP也是传输层协议，它是无连接的，不可靠的传输服务。当接收数据时它不向发送方提供确认信息，它不提供输入包的顺序，如果出现丢失包或重份包的情况，也不会向发送方发出差错报文。由于它执行功能时具有较低的开销，因而执行速度比TCP快。

UDP协议头



## Linux网络编程

### Socket

linux中的网络编程通过Socket（套接字）接口实现，Socket是一种文件描述符。套接字，是支持[TCP/IP](http://baike.baidu.com/item/TCP%2FIP)的[网络通信](http://baike.baidu.com/item/%E7%BD%91%E7%BB%9C%E9%80%9A%E4%BF%A1)的基本操作单元，可以看做是不同主机之间的进程进行双向通信的端点，简单的说就是通信的两方的一种约定，用套接字中的相关函数来完成通信过程。它是网络通信过程中端点的抽象表示，包含进行网络通信必需的五种信息：连接使用的协议，本地主机的IP地址，本地进程的协议端口，远地主机的IP地址，远地进程的协议端口。

常用的TCP/IP协议的3种套接字类型如下所示。

流套接字（SOCK\_STREAM）：

流套接字用于提供面向连接、可靠的数据传输服务。该服务将保证数据能够实现无差错、无重复发送，并按顺序接收。流套接字之所以能够实现可靠的数据服务，原因在于其使用了传输控制协议，即TCP（The Transmission Control Protocol）协议。

数据报套接字（SOCK\_DGRAM）：

数据报套接字提供了一种[无连接](http://baike.baidu.com/item/%E6%97%A0%E8%BF%9E%E6%8E%A5)的服务。该服务并不能保证数据传输的可靠性，数据有可能在传输过程中丢失或出现数据重复，且无法保证顺序地接收到数据。数据报套接字使用UDP（User Datagram Protocol）协议进行数据的传输。由于数据报套接字不能保证数据传输的可靠性，对于有可能出现的数据丢失情况，需要在程序中做相应的处理。

原始套接字（SOCK\_RAW）：

原始套接字(SOCKET\_RAW)允许对较低层次的协议直接访问，比如IP、 ICMP协议，它常用于检验新的协议实现，或者访问现有服务中配置的新设备，因为RAW SOCKET可以自如地控制Windows下的多种协议，能够对网络底层的传输机制进行控制，所以可以应用原始套接字来操纵网络层和传输层应用。比如，我们可以通过RAW SOCKET来接收发向本机的ICMP、IGMP协议包，或者接收TCP/IP栈不能够处理的IP包，也可以用来发送一些自定包头或自定协议的IP包。网络监听技术很大程度上依赖于SOCKET\_RAW

原始套接字与标准套接字（标准套接字指的是前面介绍的流套接字和数据包套接字）的区别在于：原始套接字可以读写[内核](http://baike.baidu.com/item/%E5%86%85%E6%A0%B8)没有处理的IP数据包，而流套接字只能读取TCP协议的数据，数据报套接字只能读取UDP协议的数据。因此，如果要访问其他协议发送数据必须使用原始套接字。

地址结构

struct sockaddr

{

u\_short sa\_family;

char sa\_data[14];

}

Sa\_family:

地址族，采用“AF\_xxx”的形式，如：AF\_INET。

Sa\_data：14字节的特定。

struct sockaddr\_in

{

short int sin\_family; /\* Internet 地址族 \*/

unsigned short int sin\_port; /\* 端 端 口号 号 \*/

struct in\_addr sin\_addr; /\* IP 地址 \*/

unsigned char sin\_zero[8]; /\* 填0 \*/

}

编程中一般并不直接针对sockaddr数据结构操作，而是使用与sockaddr等价的的sockaddr\_in数据结。

struct in\_addr

{

unsigned long s\_addr;

}

S\_addr: 32位的地址。

地址转换

IP地址通常由数字加点(192.168.0.1)的形式表示，而在struct in\_addr中使用的是IP地址是由32位的整数表示的， 为了转换我们可以使用下面两个函 数 ：

int inet\_aton(const char \*cp,struct in\_addr \*inp)

char \*inet\_ntoa(struct in\_addr in)

函数里面a代表ascii ，n代表network.第一个函数表示将a.b.c.d形式的IP转换为32 位的IP,存储在 inp指针里面。第二个是将32 位IP转换为a.b.c.d的格式。

字节序转换

不同类型的CPU 对变量的字节存储顺序可能不同：有的系统是高位在前，低位在后，而有的系统是低位在前，高位在后，而网络传输的数据顺序是一定要统一的。所以当内部字节存储顺序和网络字节顺序不同时，就一定要进行转换。网络字节顺序采用big endian排序方式。

htons

把unsigned short类型从主机序转换到网络序

htonl

把unsigned long类型从主机序转换到网络序

ntohs

把unsigned short类型从网络序转换到主机序

ntohl

把unsigned long类型从网络序转换到主机  
IP与主机名

在网络上标识一台机器可以用IP，也可以使用主机名。

struct hostent \*gethostbyname(const char \*hostname)

struct hostent

{

char \*h\_name; /\* 主机 主机 的 正 式名称 名称 \*/

char \*h\_aliases; /\* 主机 主机 的 别名 名 \*/

int h\_addrtype; /\* 主机 主机 的 地址 类型 型 AF\_INET\*/

int h\_length; /\* 主机 主机 的 地址 长度 度 \*/

char \*\*h\_addr\_list; /\* 主机 主机的 的IP地址 地址 列表 表 \*/

}

#define h\_addr h\_addr\_list[0] /\* 主机 主机 的 第一个 个IP 地址\* /

函数

进行Socket编程的常用函数有：

socket :创建一个socket

bind:用于绑定IP地址和端口号到socket.

connect:该函数用于绑定之后的client端，与服务器建立连接。

listen:设置能处理的最大连接请求，listen()并未开始接受连线，只是设置socket为listen模式。

accept:用来接收socket链接。

send：发送数据

recv：接收数据

### 基于TCP协议通信编程

**服务器端流程**

1. 创建一个socket，用函数socket()

2. 绑定IP地址、端口等信息到socket上，用函数bind()

3. 设置允许的最大连接数，用函数listen()

4. 接收客户端上来的连接，用函数accept()

5. 收发数据，用函数send()和recv()，或者read() 和write()

6. 关闭网络连接

**客户端流程**

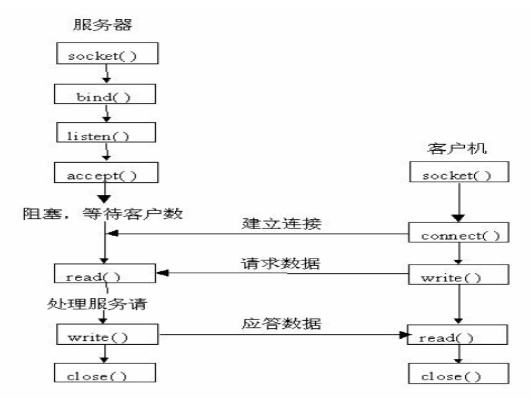
1. 创建一个socket，用函数socket()

2. 设置要连接的对方的的IP地址和地址和端口等属性

3. 连接服务器，用函数connect()

4. 收发数据，用函数send()和recv()，或者read() 和write()

5. 关闭网络



### 基于UDP协议通信编程

**服务器端**

1. 创建一个socket，用函数socket()

2. 绑定IP地址、端口等信息到socket上，用函数bind()

3. 循环接收数据，用函数recvfrom()

4. 关闭网络连接

**客户端**

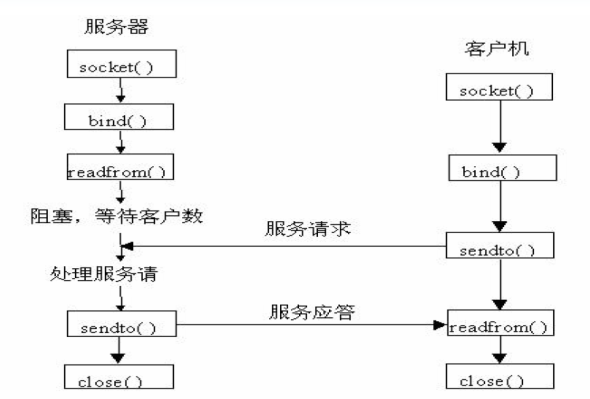
1. 创建一个socket，用函数socket()

2. 绑定IP地址、端口等信息到socket上，用函数bind()

3. 设置对方的的IP地址和端口等属性

4. 发送数据，用函数sendto()

5. 关闭网络连接



### 服务器模型

在网络程序里面,一般来说都是许多客户对应一个服务器，为了处理客户的请求, 对服务端的程序就提出了特殊的要求。目前最常用的服务器模型有：

• 循环服务器:服务器在同一个时刻只可以响应一个客户端的请求

• 并发服务器:服务器在同一个时刻可以响应多个客户端的请求

一般来说，UDP使用的是循环服务器，TCP使用的是并发服务器。

