网络编程

一、网络协议概述

网络协议是网络上所有设备（网络服务器、计算机及交换机、路由器、防火墙等）之间通信规则的集合，它规定了通信时信息必须采用的格式和这些格式的意义。大多数网络都采用分层的体系结构，每一层都建立在它的下层之上，向它的上一层提供一定的服务，而把如何实现这一服务的细节对上一层加以屏蔽。一台设备上的第 n 层与另一台设备上的第 n 层进行通信的规则就是第 n 层协议。在网络的各层中存在着许多协议，接收方和发送方同层的协议必须一致，否则一方将无法识别另一方发出的信息。网络协议使网络上各种设备能够相互交换信息。

1.1 层次结构

由于网络节点之间联系的复杂性，在制定协议时，通常把复杂成分分解成一些简单成分，然后再将它们复合起来。最常用的复合技术就是层次方式，网络协议的层次结构如下：

（1）结构中的每一层都规定有明确的服务及接口标准。

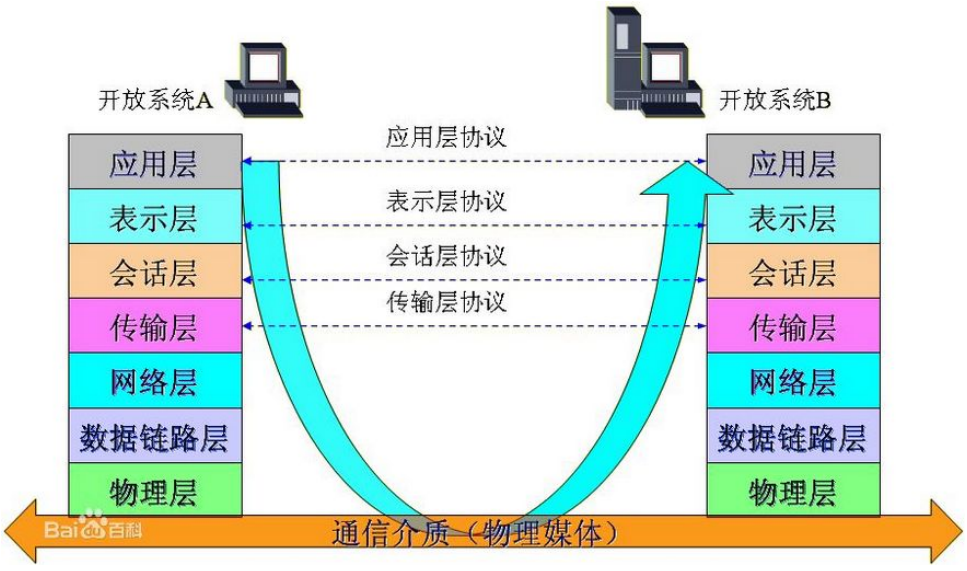
（2）把用户的应用程序作为最高层

（3）除了最高层外，中间的每一层都向上一层提供服务，同时又是下一层的用户。

（4）把物理通信线路作为最低层，它使用从最高层传送来的参数，是提供服务的基础。

1.2 层次划分

为了使不同计算机厂家生产的计算机能够相互通信，以便在更大的范围内建立计算机网络，国际标准化组织（ISO）在 1978 年提出了“开放系统互联参考模型”，即著名的 OSI/RM 模型（Open System Interconnection/Reference Model）。它将计算机网络体系结构的通信协议划分为七层，自下而上依次为：物理层（Physics Layer）、数据链路层（Data Link Layer）、网络层（Network Layer）、传输层（Transport Layer）、会话层（Session Layer）、表示层（Presentation Layer）、应用层（Application Layer）。



OSI 协议参考模型的这7个层虽然规定得非常细致和完善，但在实际中却得不到广泛的应用，其重要的原因之一就在于它过于复杂。但它仍是此后很多协议模型的基础，这种分层

架构的思想在很多领域都得到了广泛的应用。

与此相区别的TCP/IP 协议模型从一开始就遵循简单明确的设计思路，它将TCP/IP 的7层协议模型简化为4 层，从而更有利于实现和使用。

二、 TCP/IP 协议概述

在 linux 系统中网络体系结构中采用的是 tcp/ip 协议，网络传输的协议比较多，而 tcp 协议和 ip 协议，是网络通信时，经常使用的两种协议，我们为了方便描述，一般用 tcp/ip 来描述整体的网络协议。

TCP（传输控制协议）和 IP（网际协议）Transmission Control Protocol/Internet Protocol 的简写，中译名为传输控制协议/因特网互联协议，又名网络通讯协议，是 Internet 最基本的协议、Internet 国际互联网络的基础，由网络层的 IP 协议和传输层的 TCP 协议组成。TCP/IP 定义了电子设备如何连入因特网，以及数据如何在它们之间传输的标准。协议采用了 4 层的层级结构，每一层都呼叫它的下一层所提供的网络来完成自己的需求。

2.1 TCP/IP 协议特点

（1）TCP/IP 协议不依赖于任何特定的计算机硬件或操作系统，提供开放的协议标准，即使不考虑 Internet，TCP/IP 协议也获得了广泛的支持。所以 TCP/IP 协议成为一种联合各种硬件和软件的实用系统。

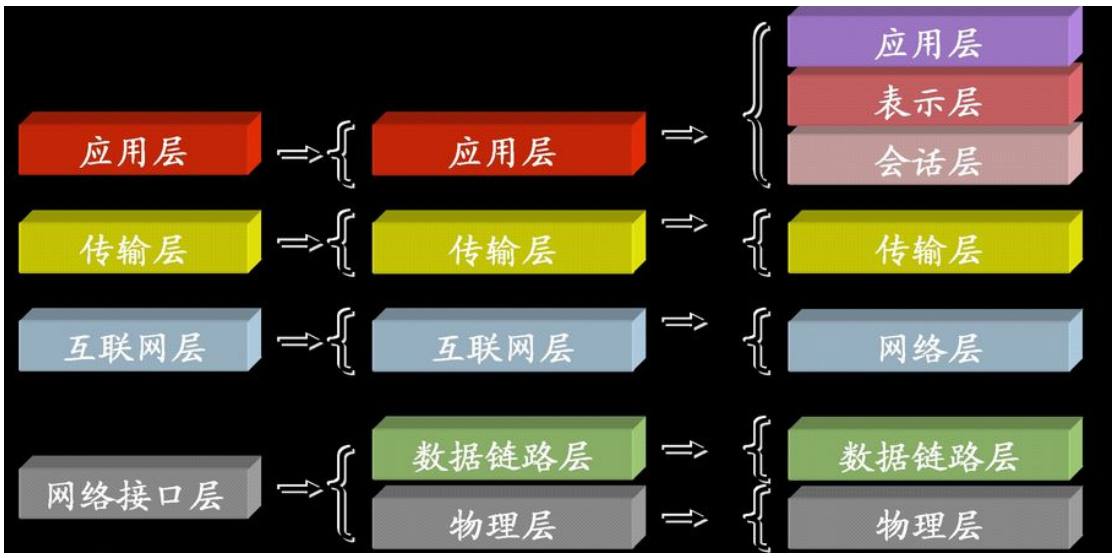
（2）TCP/IP 协议并不依赖于特定的网络传输硬件，所以 TCP/IP 协议能够集成各种各样的网络。用户能够使用以太网（Ethernet）、令牌环网（Token Ring Network）、拨号线路（Dial-up line）、X.25 网以及所有的网络传输硬件。

（3）统一的网络地址分配方案，使得整个 TCP/IP 设备在网中都具有唯一的地址

（4）标准化的高层协议，可以提供多种可靠的用户服务。

2.2 TCP/IP 层次模型

TCP/IP 的协议参考模型和OSI 协议参考模型的对应关系如下图所示。下面分别对者TCP/IP 的4 层模型进行简要介绍



从协议分层模型方面来讲，TCP/IP 由四个层次组成：网络接口层、网络层、传输层、应用层。

（1）网络接口层:

物理层(网络接口层)是定义物理介质的各种特性：机械特性、电子特性、功能特性、规程特性等，数据链路层是负责接收 IP 数据包并通过网络发送，或者从网络上接收物理帧，抽

出 IP 数据包，交给 IP 层。----------网卡驱动中主要实现网络接口层定义的内容。

（2）网络层：

负责将数据帧封装成 IP 数据报，并运行必要的路由算法。

（3）传输层：

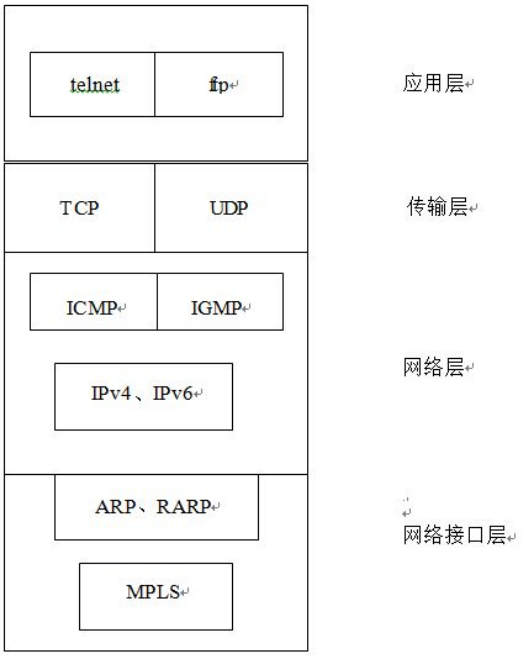
负责端对端之间的通信会话连接与建立。传输协议的选择根据数据传输方式而定。传输层协议主要有 tcp 协议和 udp 协议。

（4）应用层：

负责应用程序的网络访问，这里通过端口号来识别各个不同的进程。

2.3 协议族

虽然 TCP/IP 名称只包含了两个协议，但实际上，TCP/IP 是一个庞大的协议族，它包括了各个层次上的众多协议，下图列举了各层中一些重要的协议，并给出了各个协议在不同层次中所处的位置如下。



ARP：用于获得同一物理网络中的硬件主机地址。

MPLS：多协议标签协议，是很有发展前景的下一代网络协议。

IP：负责在主机和网络之间寻址和路由数据包。

ICMP：用于发送报告有关数据包的传送错误的协议。

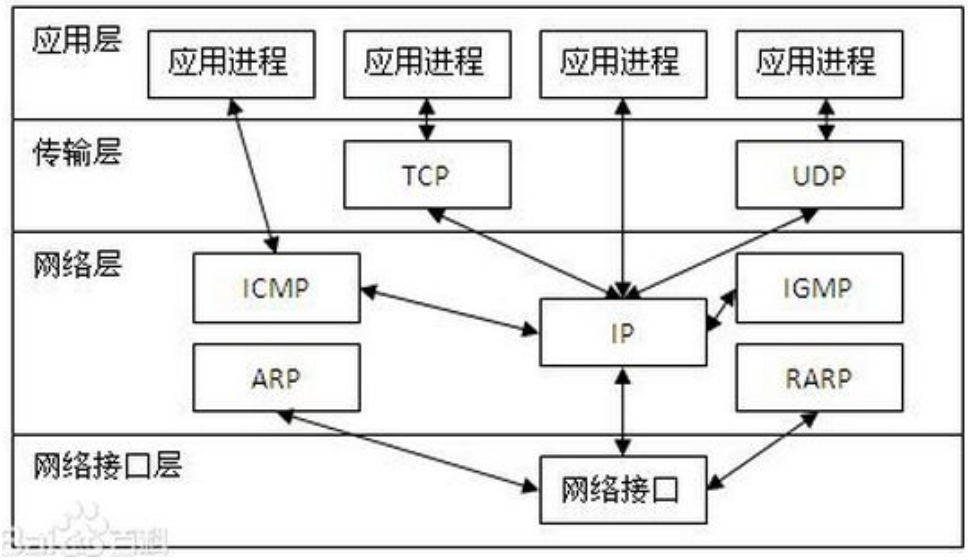
IGMP：被 IP 主机用来向本地多路广播路由器报告主机组成员的协议。

TCP：为应用程序提供可靠的通信连接。适合于一次传输大批数据的情况。并适用于要求得到响应的应用程序。

UDP：提供了无连接通信，且不对传送包进行可靠的保证。适合于一次传输少量数据，可靠性则由应用层来负责。

udp 和 tcp 特征刚好是相反，对方的优点就是自己的缺点。

2.4 TCP/IP 协议模块关系



2.5 协议的选择

协议的选择应该考虑到以下 3 个方面。

（1）对数据可靠性的要求。

对数据要求高可靠性的应用需选择 TCP 协议，如验证、密码字段的传送都是不允许出错的，而对数据的可靠性要求不那么高的应用可选择 UDP 传送，如语言、视频。

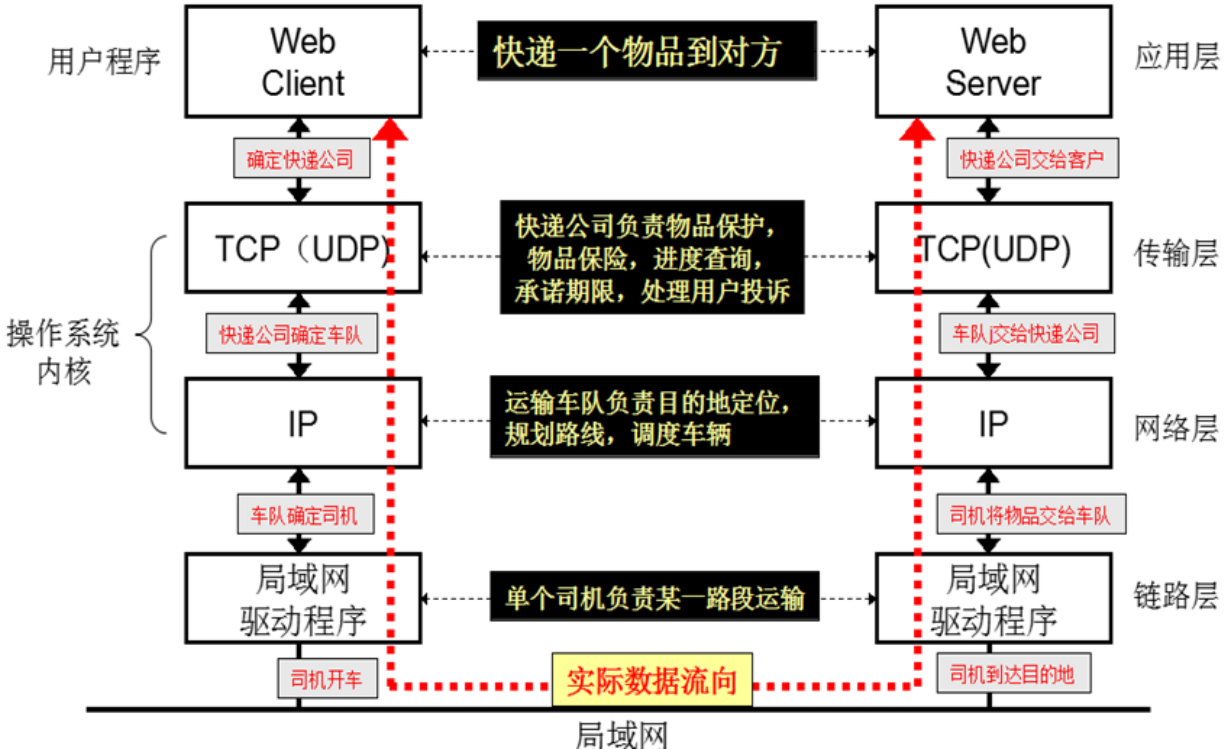
（2）应用的实时性。

TCP 协议在传送过程中要使用三次握手、重传确认等手段来保证数据传输的可靠性。使用 TCP 协议会有较大的时延，因此不适合对实时性要求较高的应用，如 VOIP、视频监控等。相反，UDP 协议则在这些应用中能发挥很好的作用。

（3）网络的可靠性。

由于 TCP 协议的提出主要是解决网络的可靠性问题，它通过各种机制来减少错误发生的概率。因此，在网络状况不是很好的情况下需选用 TCP 协议（如在广域网等情况），但是若在网络状况很好的情况下（如局域网等）就不需要再采用 TCP 协议，而建议选择 UDP 协议来减少网络负荷。

Internet协议介绍:



三、IP 地址

3.1 IP 地址简介

IP 地址是指互联网协议地址（英语：Internet Protocol Address，又译为网际协议地址），是 IP Address 的缩写。IP 地址是 IP 协议提供的一种统一的地址格式，**它为互联网上的每一个网络和每一台主机分配一个逻辑地址，以此来屏蔽物理地址的差异。**

IP 是英文 Internet Protocol 的缩写，意思是“网络之间互连的协议”，也就是为计算机网络相互连接进行通信而设计的协议。在因特网中，它是能使连接到网上的所有计算机网络实现相互通信的一套规则，规定了计算机在因特网上进行通信时应当遵守的规则。任何厂家生产的计算机系统，只要遵守 IP 协议就可以与因特网互连互通。正是因为有了 IP 协议，因特网才得以迅速发展成为世界上最大的、开放的计算机通信网络。因此，IP 协议也可以叫做“因特网协议”。

IP 地址被用来给 Internet 上的电脑一个编号。大家日常见到的情况是每台联网的 PC 上都需要有 IP 地址，才能正常通信。我们可以把“个人电脑”比作“一台电话”，那么“IP 地址”就相当于“电话号码”，而 Internet 中的路由器，就相当于电信局的“程控式交换机”。

IP 地址是一个 32 位的二进制数，通常被分割为 4 个“8 位二进制数”（也就是 4 个字节）。IP 地址通常用“点分十进制”表示成（a.b.c.d）的形式，其中，a,b,c,d 都是 0~255 之间的十进制整数。例：点分十进 IP 地址（100.4.5.6），实际上是 32 位二进制数（01100100.00000100.00000101.00000110）。

IP 地址（英语：Internet Protocol Address）是一种在 Internet 上的给主机编址的方式，也称为网际协议地址。常见的 IP 地址，分为 IPv4 与 IPv6 两大类。

IPV4 就是有 4 段数字，每一段最大不超过 255。由于互联网的蓬勃发展，IP 位址的需求量愈来愈大，使得 IP 位址的发放愈趋严格，各项资料显示全球 IPv4 位址可能在 2005 至 2010 年间全部发完(实际情况是在 2011 年 2 月 3 日 IPv4 位地址分配完毕)。

地址空间的不足必将妨碍互联网的进一步发展。为了扩大地址空间，拟通过 IPv6 重新定义地址空间。IPv6 采用 128 位（16 字节）地址长度。在 IPv6 的设计过程中除了一劳永逸地解决了地址短缺问题以外，还考虑了在 IPv4 中解决不好的其它问题。

3.2 IP 地址类型

公有地址 Public IP

公有地址（Public address）由 Inter NIC（Internet Network Information Center 因特网信息中心）负责。这些 IP 地址分配给注册并向 Inter NIC 提出申请的组织机构。通过它直接访问因特网。

私有地址 Privat IP

私有地址（Private address）属于非注册地址，专门为组织机构内部使用。

以下列出留用的内部私有地址

A 类 10.0.0.0--10.255.255.255

B 类 172.16.0.0--172.31.255.255

C 类 192.168.0.0--192.168.255.255

不能直接连上Internet的IP,主要用于局域网络内的主机联机规划. 可以利用NAT (Network Address Transfer) 主机，透过 IP 伪装来使私有 IP 的计算机连上 Internet。

3.3 IP 地址分配

地域划分：

TCP/IP 协议需要针对不同的网络进行不同的设置，且每个节点一般需要一个“IP 地址”、一个“子网掩码”、一个“默认网关”。不过，可以通过动态主机配置协议（DHCP），给客户端自动分配一个 IP 地址，避免了出错，也简化了 TCP/IP 协议的设置。

IP 地址现由因特网名字与号码指派公司 ICANN（Internet Corporation for Assigned Names and Numbers）分配。

InterNIC：负责美国及其他地区；

ENIC：负责欧洲地区；

APNIC（Asia Pacific Network Information Center）： 我国用户可向 APNIC 申请（要缴费）

PS：1998 年，APNIC 的总部从东京搬迁到澳大利亚布里斯班。

负责 A 类 IP 地址分配的机构是 ENIC

负责北美 B 类 IP 地址分配的机构是 InterNIC

负责亚太 B 类 IP 地址分配的机构是 APNIC

字段划分：

IP 地址可以视为网络标识号码与主机标识号码两部分，因此 IP 地址可分两部分组成，一部分为网络地址，另一部分为主机地址。

将 IP 地址分成了网络号和主机号两部分，设计者就必须决定每部分包含多少位。网络号的位数直接决定了可以分配的网络数;主机号的位数则决定了网络中最大的主机数。然而，由于整个互联网所包含的网络规模可能比较大，也可能比较小，设计者最后聪明的选择了一种灵活的方案：将 IP 地址空间划分成不同的类别，每一类具有不同的网络号位数和主机号位数。

IP 地址分为 A、B、C、D、E 5 类，它们适用的类型分别为：大型网络、中型网络、小型网络、多目地址、备用。常用的是 B 和 C 两类。

其中 A、B、C 3 类（如下表格）由 InternetNIC 在全球范围内统一分配，D、E 类为特殊地址。

类别	最大网络数	IP 地址范围	最大主机数	私有 IP 地址范围
A	126（2^7-2）	0.0.0.0-127.255.255.255	16777214	10.0.0.0-10.255.255.255
B	16384(2^14)	128.0.0.0-191.255.255.255	65534	172.16.0.0-172.31.255.255
C	2097152(2^21)	192.0.0.0-223.255.255.255	254	192.168.0.0-192.168.255.255

A 类 IP 地址：

一个 A 类 IP 地址是指， 在 IP 地址的四段号码中，第一段号码为网络号码，剩下的三段号码为本地计算机的号码。如果用二进制表示 IP 地址的话，A 类 IP 地址就由 1 字节的网络地址和 3 字节主机地址组成。A 类 IP 地址中网络的标识长度为 8 位，主机标识的长度为 24 位，A 类网络地址数量较少，有 126 个网络，每个网络可以容纳主机数达 1600 多万台。

A 类 IP 地址 地址范围 1.0.0.0 到 127.255.255.255[1]（二进制表示为：00000001 00000000 00000000 00000000 - 01111110 11111111 11111111 11111111）。最后一个是广播地址。

A 类 IP 地址的子网掩码为 255.0.0.0，每个网络支持的最大主机数为 256 的 3 次方-2=16777214 台。

B 类 IP 地址：

一个 B 类 IP 地址是指，在 IP 地址的四段号码中，前两段号码为网络号码。如果用二进制表示 IP 地址的话，B 类 IP 地址就由 2 字节的网络地址和 2 字节主机地址组成。B 类 IP 地址中网络的标识长度为 16 位，主机标识的长度为 16 位，B 类网络地址适用于中等规模的网络，有 16384 个网络，每个网络所能容纳的计算机数为 6 万多台。

B 类 IP 地址地址范围 128.0.0.0-191.255.255.255 （二进制表示为：10000000 00000000 00000000 00000000----10111111 11111111 11111111 11111111）。 最后一个是广播地址。

B 类 IP 地址的子网掩码为 255.255.0.0，每个网络支持的最大主机数为 256 的 2 次方-2=65534 台。

C 类 IP 地址：

一个 C 类 IP 地址是指，在 IP 地址的四段号码中，前三段号码为网络号码，剩下的一段号码为本地计算机的号码。如果用二进制表示 IP 地址的话，C 类 IP 地址就由 3 字节的网络地址和 1 字节主机地址组成。C 类 IP 地址中网络的标识长度为 24 位，主机标识的长度为 8 位，C 类网络地址数量较多，有 209 万余个网络。适用于小规模的局域网络，每个网络最多只能包含 254 台计算机。

C 类 IP 地址范围 192.0.0.0-223.255.255.255 （二进制表示为: 11000000 00000000 00000000 00000000 - 11011111 11111111 11111111 11111111）。

C 类 IP 地址的子网掩码为 255.255.255.0，每个网络支持的最大主机数为 256-2=254 台

D 类 IP 地址：

D 类 IP 地址在历史上被叫做多播地址(multicast address)，即组播地址。在以太网中，多播地址命名了一组应该在这个网络中应用接收到一个分组的站点。多播地址的最高位必须是“1110”，范围从 224.0.0.0 到 239.255.255.255。

3.4 特殊的网址（了解）

1. IP 地址中的每一个字节都为 1 的 IP 地址（“255．255．255．255”）是当前子网的广播地址；

2. IP 地址中凡是以“11110”开头的 E 类 IP 地址都保留用于将来和实验使用。

3. IP 地址中不能以十进制“127”作为开头，该类地址中数字 127．0．0．1 到 127.255.255.255 用于回路测试，如：127.0.0.1 可以代表本机 IP 地址，用“http://127.0.0.1”就可以测试本机中配置的 Web 服务器。

三、端口号

端口有什么用呢？我们知道，一台拥有 IP 地址的主机可以提供许多服务，比如 Web 服务、FTP 服务、SMTP 服务等，这些服务都是通过 1 个 IP 地址来实现。那么，主机是怎样区分不同的网络服务呢？显然不能只靠 IP 地址，因为 IP 地址与网络服务的关系是一对多的关系。实际上是通过“IP 地址+端口号”来区分不同的服务的。

服务器一般都是通过端口号来识别的。例如，对于每个 TCP/IP 实现来说，FTP 服务器的 TCP 端口号都是 21，每个 Telnet 服务器的 TCP 端口号都是 23，每个 TFTP(简单文件传送协议)服务器的 UDP 端口号都是 69。任何 TCP/IP 实现所提供的服务都用知名的 1～1023 之间的端口号。这些知名端口号由 Internet 号分配机构（InternetAssignedNumbersAuthority,IANA）来管理。

TCP 与 UDP 段结构中端口地址都是 16 位，可以在 0---65535 范围内的端口号。对于这 65536 个端口号有以下的使用规定：

（1）任何知名 TCP/IP 实现所提供的服务都用 1---1023 之间的端口号，是由 IANA 来管理的；端口号小于 256 的定义为常用端口，服务器一般都是通过常用端口号来识别的。256～1023 之间的端口号通常都是由 Unix 系统占用，以提供一些特定的 Unix 服务，也就是说，提供一些只有 Unix 系统才有的、而其他操作系统可能不提供的服务。

（2）大多数 TCP/IP 实现给临时端口号分配 1024---5000 之间的端口号。大于 5000 的端口号是为其他服务器预留的。

（3）客户端只需保证该端口号在本机上是惟一的就可以了。客户端口号因存在时间很短暂又称临时端口号；

下列函数可以实现IPv4和IPv6的地址和主机名之间的转化。
gethostbyname()：将主机名转化为IP地址
gethostbyaddr()：将IP地址转化为主机名

```
struct hostent
{
    char  *h_name;          /*正式主机名*/
    char  **h_aliases;      /*主机别名*/
    int   h_addrtype;       /*地址类型*/
    int   h_length;         /*地址字节长度*/
    char  **h_addr_list;    /*指向IPv4或IPv6的地址指针数组*/
}
```

getaddrinfo()：自动识别IPv4地址和IPv6地址。

```
struct addrinfo
{
    int ai_flags;           /*AI_PASSIVE, AI_CANONNAME*/
    int ai_family;          /*地址族*/
    int ai_socktype;        /*socket类型*/
    int ai_protocol;        /*协议类型*/
    size_t ai_addrlen;      /*地址字节长度*/
    char *ai_canonname;     /*主机名*/
    struct sockaddr *ai_addr;  /*socket结构体*/
    struct addrinfo *ai_next;  /*下一个指针链表*/
}
```

将主机名转化为IP地址

所需头文件	#include <netdb.h>
函数原型	struct hostent *gethostbyname(const char *hostname)
函数传入值	hostname：主机名
函数返回值	成功：hostent 类型指针
	出错：-1

所需头文件	#include <netdb.h>
函数原型	int getaddrinfo(const char *node, const char *service, const struct addrinfo *hints, struct addrinfo **result)
函数传入值	node：网络地址或者网络主机名
	service：服务名或十进制的端口号字符串
	hints：服务线索
	result：返回结果
函数返回值	成功：0
	出错：-1

四、 socket 编程

1 socket 定义

socket 概述： 套接字------是一套网络编程的接口函数集的统称。

在 Linux 中的网络编程是通过 socket 接口来进行的。人们常说的 socket 接口是一种特殊的 I/O，它也是一种文件描述符。每一个 socket 都用一个结构描述{协议，本地地址、本地端口}来表示；一个完整的套接字则用一个相关描述{协议，本地地址、本地端口、远程地址、 远程端口}。socket 也有一个类似于打开文件的函数调用，该函数返回一个整型的 socket 描述符，随后的连接建立、数据传输等操作都是通过 socket 来实现的。

可以将一个 socket 理解为一个打开的网络。它类型于我们对文件操作中打开文件时返回的文件描述符，类似于，当我们对一个文件操作时，使用文件描述符来操作，当我们要进行一个网络通信时，使用 socket 来操作。

2 socket 类型

常见的 socket 有 3 种类型如下。

（1）流式 socket（SOCK_STREAM）

流式套接字提供可靠的、面向连接的通信流；它使用 TCP 协议，从而保证了数据传输的正确性和顺序性。

（2）数据报 socket（SOCK_DGRAM）

数据报套接字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，并且不保证是可靠、无差错的。它使用数据报协议 UDP。

（3）原始 socket（一般很少用）

原始套接字允许对底层协议如 IP 或 ICMP 进行直接访问，它功能强大但使用较为不便，主要用于一些协议的开发。

3 地址及数据顺序处理

当我们进行网络传输时，ip 地址，我们经常使用的是点分式表示的 ip,而为了方便网络传输，在传输时，使用的是经过转化的 ip 地址。

3.1 地址结构相关处理

（1）数据结构介绍-----描述网络主机信息的。

下面首先介绍两个重要的数据类型：sockaddr 和 sockaddr_in，这两个结构类型都是用来保存 socket 信息的，如下所示：

```
struct sockaddr {
    unsigned short    sa_family;    /*地址族*/    ipv4 ipv6
    char sa_data[14];                /*14 节的协议地址，包含该 socket 的 IP 地址和端口号。*/
};
struct sockaddr_in {
    __kernel_sa_family_t sin_family;        /* 地址族 2 个字节 Address family        */
    __be16               sin_port;          /* 端口号 2 个字节，区分应用 Port number
      */
    struct in_addr    sin_addr;             /* IP 地址 4 个字节 Internet address          */

    /*填充 0 以保持与 struct sockaddr 同样大小       Pad to size of `struct sockaddr'. */
```

```
    unsigned char           __pad[__SOCK_SIZE__ - sizeof(short int) -
                    sizeof(unsigned short int) - sizeof(struct in_addr)];
};
```

这两个数据类型是等效的，可以相互转化，通常 sockaddr_in 数据类型使用更为方便。在建立 socketadd 或 sockaddr_in 后，就可以对该 socket 进行适当的操作了。

```
struct in_addr {
    __be32    s_addr;        //网络字节序的二进制表示的 ip 地址。
};
```

数据类型	说明	定义所在的头文件
int8_t	带符号的 8 位整数	<sys/types.h>
uint8_t	无符号的 8 位整数	<sys/types.h>
int16_t	带符号的 16 位整数	<sys/types.h>
uint16_t	无符号的 16 位整数	<sys/types.h>
int32_t	带符号的 32 位整数	<sys/types.h>
uint32_t	无符号的 32 位整数	<sys/types.h>
sa_family_t	套接口地址结构的地址族	<sys/socket.h>
socklen_t	套接口地址结构的长度，一般为 uint32_t	<sys/socket.h>
in_port_t	TCP 或 UDP 端口号，一般为 uint16_t	<netinet/in.h>
in_addr_t	IPv4 地址，一般为 uint32_t	<netinet/in.h>

（2）结构字段，

下面列出了该结构 sa_family 字段可选的常见值。

结构定义头文件	#include <netinet/in.h>
sa_family	**AF_INET：IPv4 协议**
	AF_INET6：IPv6 协议
	AF_LOCAL：UNIX 域协议
	AF_LINK：链路地址协议
	AF_KEY：密钥套接字（socket）

sockaddr_in 其他字段的含义非常清楚，具体的设置涉及其他函数，在后面会有详细的讲解。

3.2 数据存储优先顺序转化

（1）函数说明。

计算机数据存储有两种字节优先顺序：高位字节优先（称为大端模式）和低位字节优先（称为小端模式，PC 机通常采用小端模式）。Internet 上数据以高位字节优先顺序在网络上传输，因此在有些情况下，需要对这两个字节存储优先顺序进行相互转化。

这里用到了 4 个函数：htons()、ntohs()、htonl()和 ntohl()。这 4 个地址分别实现网络字节序和主机字节序的转化，这里的 h 代表 host，n 代表 network，s 代表 short，l 代表 long。通常 16 位的 IP 端口号用 s 代表，而 IP 地址用 l 来代表。

（2）函数格式说明。

htons 等函数语法要点

所需头文件	#include <arpa/inet.h>
函数原型	uint16_t htons(unit16_t host16bit) uint32_t htonl(unit32_t host32bit) uint16_t ntohs(unit16_t net16bit)

- **htons()：主机字节序到网络字节序的转化(16位)**
- **ntohs()：网络字节序到主机字节序的转化(16位)**
- **htonl()：主机字节序到网络字节序的转化(32位)**
- **ntohl()：网络字节序到主机字节序的转化(32位)**

	uint32_t ntohs(unit32_t net32bit)
函数传入值	host16bit：主机字节序的 16 位数据
	host32bit：主机字节序的 32 位数据
	net16bit： 网络字节序的 16 位数据
	net32bit： 网络字节序的 32 位数据
函数返回值	成功：返回要转换的字节序
	出错：−1

3.3 ip 地址格式转化

通常用户在表达地址时采用的是点分十进制表示的数值（或者是以冒号分开的十进制 IPv6 地址），而在通常使用的 socket 编程中所使用的则是二进制值，这就需要将这两个数值进行转换。这里在 IPv4 中用到的函数有 inet_aton()、inet_addr()和 inet_ntoa()，而 IPv4 和 IPv6 兼容的函数有 inet_pton()和 inet_ntop()。

- **int inet_aton(const char *straddr,struct in_addr *addrptr);**//将 straddr 表示的点分 ip 字符串转换为网络字节序的二进制整数。成功返 0，失败返回-1，转换后保存在 addrptr 中。
- **char * inet_ntoa(struct in_addr inaddr);** //将二进制数值转换为点分 ip 字符串返回。
- **in_addr_t inet_addr(const char *straddr);**//返回整数表示的 ip.出错返回 INADDR_NONE;
- int inet_pton(int af, const char *src, void *dst); //将点分十进制地址映射为二进制地址

所需头文件	#include <arpa/inet.h>	
函数原型	int inet_pton(int af, const char *src, void *dst);	
函数传入值	af	AF_INET：IPv4 协议
		AF_INET6：IPv6 协议
	strptr：要转化的值	
	addrptr：转化后的地址	
函数返回值	成功：0	
	出错：−1	

- const char *inet_ntop(int af, const void *src,char *dst, socklen_t cnt); //二进制地址映射为点分十进制地址

所需头文件	#include <arpa/inet.h>	
函数原型	int inet_ntop(int family, void *addrptr, char *strptr, size_t len)	
函数传入值	family	AF_INET：IPv4 协议
		AF_INET6：IPv6 协议
函数传入值	addrptr：转化后的地址	
	strptr：要转化的值	
	len：转化后值的大小	
函数返回值	成功：0	
	出错：−1	

五、 socket 函数 API

1 函数说明

进行 socket 编程的基本函数有 socket、bind、listen、accept、send、sendto、recv、recvfrom 这几个，也可以使用 read 和 write 进行收发操作，其中对于客户端和服务器端以及 TCP 和 UDP 的操作流程都有所区别，这里先对每个函数进行一定的说明，再给出不同情况下使用

的流程图。

socket：该函数用于建立一个 socket 套接口，可指定 socket 类型等信息。在建立了 socket 连接之后，可对 socketadd 或 sockaddr_in 进行初始化，以保存所建立的 socket 信息。

bind：该函数用户于将套接字和端口号之间产生绑定关系，用于告诉传输层绑定的端口号有数据时，提交给当前套接字。若绑定其他地址则不能成功。另外，它主要用于 TCP 的连接，如果不进行绑定操作，tcp 传输时，无法对端口进行监听。而在 UDP 的连接中则无必要。

connect：该函数在 TCP 中是用于 bind 的之后的 client 端，用于与服务器端建立连接，而在 UDP 中由于没有了 bind 函数，因此用 connect 有点类似 bind 函数的作用。

accept:服务器用于接收连接，建立连接。

listen:用于服务器端，指定能够监听的最大的请求数。

send 和 **recv**：这两个函数用于接收和发送数据，可以用在 TCP 中，也可以用在 UDP 中。当用在 UDP 时，可以在 connect 函数建立连接之后再用。

sendto 和 **recvfrom**：这两个函数的作用与 send 和 recv 函数类似，也可以用在 TCP 和 UDP 中。当用在 TCP 时，后面的几个与地址有关参数不起作用，函数作用等同于 send 和 recv；当用在 UDP 时，可以用在之前没有使用 connect 的情况时，这两个函数可以自动寻找指定地址并进行连接。

2 socket

socket:该函数用于建立一个 socket 套接口,可指定 socket 类型等信息。在建立了 socket 连接之后，可对 socketadd 或 sockaddr_in 进行初始化，以保存所建立的 socket 信息。

所需头文件	#include <sys/socket.h>	
函数原型	int socket(int domain, int type, int protocol);	
函数传入值	domain： 协议族	AF_INET：IPv4 协议
		AF_INET6：IPv6 协议
		AF_LOCAL：UNIX 域协议
		AF_ROUTE：路由套接字（socket）
		AF_KEY：密钥套接字（socket）
	type： 套接字类型	SOCK_STREAM：字节流套接字 socket，tcp 方式
		SOCK_DGRAM：数据报套接字 socket ，udp 方式
		SOCK_RAW：原始套接字 socket（一般不用）
	protocol：0（原始套接字除外）	
函数返回值	成功：非负套接字描述符	
	出错：–1	

3 bind

该函数用户于将套接字和端口号之间产生绑定关系,用于告诉传输层绑定的端口号有数据时，提交给当前套接字。若绑定其他地址则不能成功。另外，它主要用于 TCP 的连接，如果不进行绑定操作，tcp 传输时，无法对端口进行监听。而在 UDP 的连接中则无必要。

所需头文件	#include <sys/socket.h>
函数原型	int bind(int sockfd, const struct sockaddr *my_addr, socklen_t addrlen);
函数传入值	sockfd：socket 函数返回的套接字描述符
	my_addr：本地地址
	addrlen：地址长度
函数返回值	成功：0

	出错：−1

端口号和地址在 my_addr 中给出了，若不指定地址，则内核随意分配一个临时端口给该应用程序。

4 listen

用于服务器端，指定能够监听的最大的请求数。

所需头文件	#include <sys/socket.h>
函数原型	int listen(int sockfd, int backlog);
函数传入值	sockfd：套接字描述符
	backlog：请求队列中允许的最大请求数，大多数系统缺省值为 5，如果超出这个数目，客户端会接收到 ECONNREFUSED 拒绝连接的错误。
函数返回值	成功：0
	出错：−1

5 accept

服务器用于接收连接，建立连接。

所需头文件	#include <sys/socket.h>
函数原型	int accept(int sockfd, struct sockaddr *addr, socklen_t *addrlen);
函数传入值	sockfd：套接字描述符
	addr：用于保存客户端地址
	addrlen：地址长度
函数返回值	成功：返回新的套接字描述符，用于和客户端进行通讯
	出错：−1

6 connect

该函数在 TCP 中是用于 bind 的之后的 client 端，用于与服务器端建立连接，而在 UDP 中由于没有了 bind 函数，因此用 connect 有点类似 bind 函数的作用。

accept:服务器用于接收连接，建立连接。

所需头文件	#include <sys/socket.h>
函数原型	int connect(int sockfd, const struct sockaddr *serv_addr, socklen_t addrlen);
函数传入值	sockfd：套接字描述符
	serv_addr：存放服务器端地址
	addrlen：地址长度
函数返回值	成功：0
	出错：−1

7 send

sendto 和 **recvfrom**：这两个函数的作用与 send 和 recv 函数类似，也可以用在 TCP 和 UDP 中。当用在 TCP 时，后面的几个与地址有关参数不起作用，函数作用等同于 send 和 recv；当用在 UDP 时，可以用在之前没有使用 connect 的情况时，这两个函数可以自动寻找指定地址并进行连接。

所需头文件	#include <sys/socket.h>
函数原型	ssize_t send(int s, const void *buf, size_t len, int flags);

函数传入值	s：本地套接字描述符
	buf：指向要发送数据的指针
	len：数据长度
	flags：一般为 0
函数返回值	成功：发送的字节数
	出错：−1

8 recv

所需头文件	#include <sys/socket.h>
函数原型	ssize_t recv(int s, void *buf, size_t len, int flags);
函数传入值	s：本地套接字描述符
	buf：存放接收数据的缓冲区
	len：数据长度
	flags：一般为 0
函数返回值	成功：接收的字节数
	出错：−1

9 sendto

所需头文件	#include <sys/socket.h>
函数原型	ssize_t sendto(int s, const void *buf, size_t len, int flags, const struct sockaddr *to, socklen_t tolen);
函数传入值	s：本地套接字描述符
	buf：指向要发送数据的指针
	len：数据长度
	flags：一般为 0
	to：目地机的 IP 地址和端口号信息
	tolen：地址长度
函数返回值	成功：发送的字节数
	出错：−1

10 recvfrom

所需头文件	#include <sys/socket.h>
函数原型	ssize_t recvfrom(int s, void *buf, size_t len, int flags, struct sockaddr *from, socklen_t *fromlen);
函数传入值	s：本地套接字描述符
	buf：存放接收数据的缓冲区
	len：数据长度
	flags：一般为 0
	from：源主机的 IP 地址和端口号信息
	fromlen：地址长度
函数返回值	成功：接收的字节数
	出错：−1

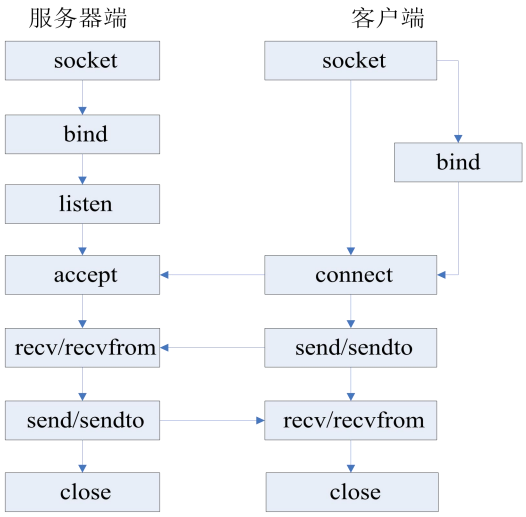
六、 TCP socket 编程

使用 tcp 编程流程：

tcp 协议通信双方不具有对等的关系。并且它们之间通信时，都是由客户端发起。不管采用哪一种方式进行网络通信，必须有一方是服务器，另一方是客户端。

具体实现：

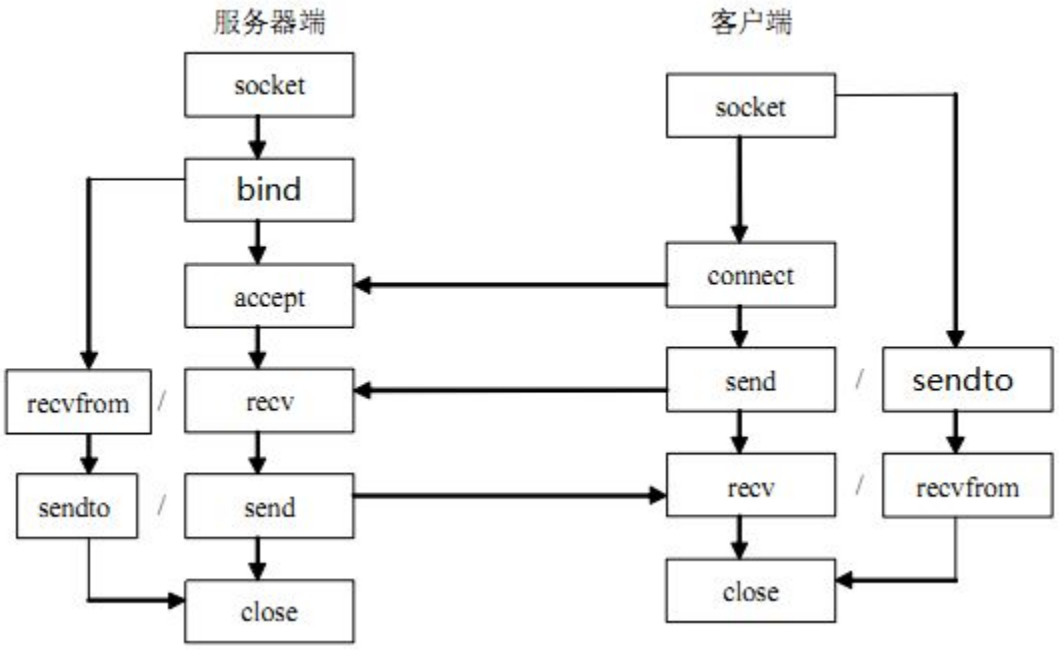
使用 tcp 建立服务器，等待客户端进行连接，接收客户端发送过来的信息。并打印。同时在收到信息时，对信息进行处理并返回给客户端。

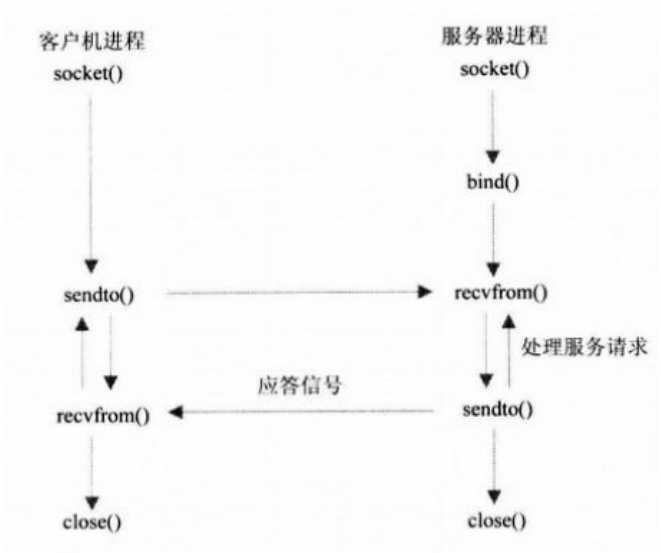


UDP socket 编程

使用 udp 编程流程：

通信双方具有对等的关系，谁都可以先发出数据。




客户机进程
socket()
sendto()
recvfrom()
close()
服务器进程
socket()
bind()
recvfrom()
处理服务请求
sendto()
close()
应答信号