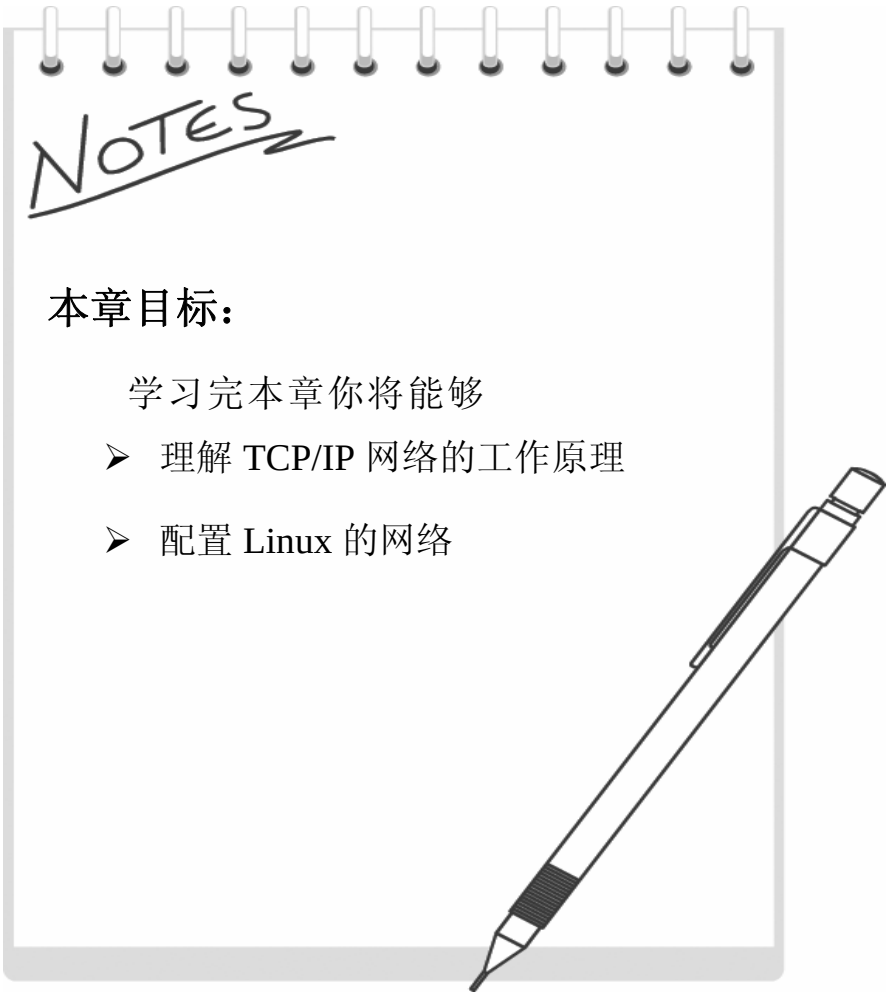
第一章　网络配置

本章导读

Linux 系统主要用作服务器。在互连网时代，要使用计算机就离不开网络。本章将讲解 Linux 系统的网络配置。但是，在开始配置网络之前，需要了解一些基本的网络原理。

本实验参考手册需要配合视频使用，互为补充，如有疑问，及时到 www.koorka.com 答疑。



1. TCP/IP 概述

计算机网络是由地理上分散的、具有独立功能的多台计算机，通过通信设备和线路互相连接起来，在配有相应的网络软件的情况下，实现计算机之间通信和资源共享的系统。计算机网络按其所跨越的地理范围可分为局域网 LAN （ Local Area Network ）和广域网 WAN （ Wide Area Network ）。

在众多的网络中，大家要进行通讯，就需要有一定的规则和相同的语言。OSI 的七成结构规定了网络通讯的规则，而 TCP/IP 相当于网络通讯的一种语言。

1.1 OSI 参考模型

计算机网络是为了实现计算机之间的通信，任何双方要成功地进行通信，必须遵守一定的信息交换规则和约定，这些信息交换规则和约定就称为通信协议（ protocol ）。计算机上的网络接口卡、通信软件、通信设备都是遵循一定的协议设计的，必须符合一定的协议规范。

为了减少协议设计的复杂性，大多数网络都按层或级的方式来组织，每一层都建立在它的下层之上。不同的网络在分层数量和各层的名字、内容与功能上都不尽相同，然而，在所有的网络中，每一层的目的都是向它的上一层提供一定的服务，而把这种服务是如何实现的细节对上层加以屏蔽。

层按功能来划分，每一层都有特定的功能，它一方面利用下一层所提供的功能，另一方面又为其上一层提供服务。通信双方在相同层之间进行通话，通话规则和协定的整体就是该层的协议。每一层都有一个或多个协议，几个层合成一个协议栈（ protocol stack ）。协议的分层模型便于协议软件按模块方式进行设计和实现，这样每层协议的设计、修改、实现和测试都可以独立进行，从而减少复杂性。

OSI 模型有七层，其分层原则如下：

根据不同层次的抽象分层；

每一层应当实现一个定义明确的功能；

每一层功能的选择应当有助于制定网络协议的国际标准；

各层边界的选择应尽量减少跨过接口的通信量；

层数应足够多，以避免不同的功能混杂在同一层中；但也不能太多，否则体系结构会过于庞大。

根据这些原则， ISO 在 1983 年推出的 OSI 参考模型如图 1-1 所示：

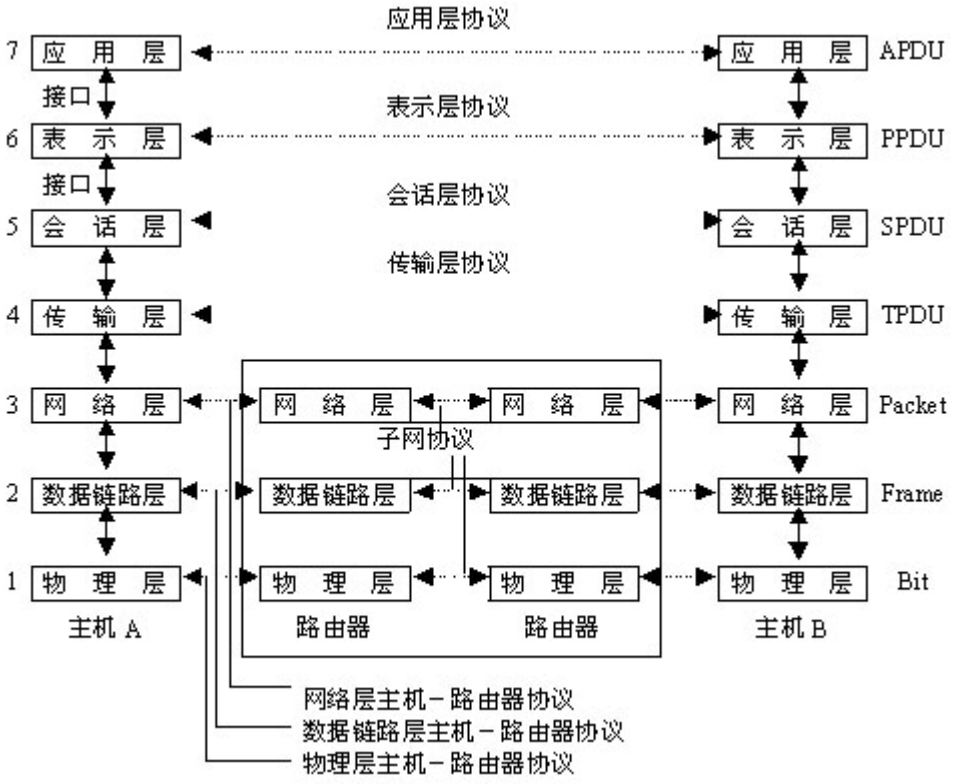


图 1-1

OSI 七层模型是指从物理层到应用层这七层，它不涉及通信的物理介质。随着网络技术的发展，特别是局域网的发展，后来对 OSI 七层模型进行了改进。修订之一就是非正式地增加了一些子层和新层，如增加了第 0 层，使之覆盖了象电缆连接器和光纤这样的硬件细节。本节我们只对 OSI 模型的七层的功能作简单描述。

1.1.1 物理层

物理层（ physical layer ）是 OSI 模型的最低层，它建立在物理通信介质的基础上，作为系统和通信介质的接口，用来实现数据链路实体间透明的比特流传输。在设计上必须保证一方发送出二进制“ 1 ”时，另一方收到的也是“ 1 ”而不是“ 0 ”。

物理层是 OSI 中唯一设计通信介质的一层，它提供与通信介质的连接，描述这种连接的机械、电气、功能和规程特性，以建 立、维护和释放数据链路实体之间的物理连接。物理层向上层提供位信息的正确传送。

物理层协议定义了硬件接口的一系列标准，典型地如用多少伏特电压表示“ 1 ”，多少伏特表示“ 0 ”；一个比特持续多少时间；传输是双向的还是单向的；最初的连接如何建立和完成通信后连接如何终止；一次通信中发送方和接收方如何应答；设备之间连接件的尺寸和接头数；每根线的用途等。

1.1.2 数据链路层

数据链路层（ data link layer ）的主要任务是加强物理层传输原始比特的功能，使之对网络层显现为一条无错链路。它在相邻网络实体之间建立、维持和释放数据链路连接，并传输数据链路数据单元（帧， frame ）。它是将位收集起来，按包处理的第一个层次，它完成发送包前的最后封装，及对到达包进行首次检视。其主要功能为：

数据链路连接的建立与释放：在每次通信前后，双方相互联系以确认一次通信的开始和结束。数据链路层一般提供无应答无连接服务、有应答无连接服务和面向连接的服务等三种类型服务。

数据链路数据单元的构成：在上层交付的数据的基础上加入数据链路协议控制信息，形成数据链路协议数据单元。

数据链路连接的分裂：当数据量很大时，为提高传输速率和效率，将原来在一条物理链路上传输的数据改用多条物理链路来传输（与多路复用相反）。

定界与同步：从物理连接上传输数的比特流中，识别出数据链路数据单元的开始和结束，以及识别出其中的每个字段，以便实现正确的接收和控制。

顺序和流量控制：用以保证发送方发送的数据单元能以相同的顺序传输到接收方，并保持发送速率与接收速率的匹配。

差错的检测与恢复：检测出传输、格式和操作等错误，并对错误进行恢复，如不能恢复则向相关网络实体报告。

1.1.3 网络层

网络层（ network layer ）关系到子网的运行控制，其关键问题之一是确定分组从源端到目的端如何选择路由。本层维护路由表，并确定哪一条路由是最快捷的，及何时使用替代路由。路由既可以选用网络中固定的静态路由表，几乎保持不变，也可以在每一次会话开始时决定（如通过终端协商决定），还可以根据当前网络的负载状况，高度灵活地为每一个分组决定路由。

网络层的另一重要功能是传输和流量控制，它在子网中同时出现过多的分组时，提供有效的流量控制服务来控制网络连接上传输的分组，以免发生信息“堵塞”或“拥挤”现象。

网络层提供两种类型的网络服务，即无连接的服务（数据报服务）和面向连接的服务（虚电路服务）。网络层使较高层与连接系统所用的数据传输和交换技术相独立。

IP 协议工作在本层，它提供“无连接的”或“数据报”服务。

1.1.4 传输层

传输层（ transport layer ）的基本功能是从会话层接收数据，在必要时把它们划分成较小的单元传递给网络层，并确保到达对方的各段信息准确无误。而且，这些任务都必须高效率地完成。

传输层是在网络层的基础上再增添一层软件，使之能屏蔽掉各类通信子网的差异，相用户进程提供一个能满足其要求的服务，其具有一个不变的通用接口，使用户进程只需了解该接口，便可方便地在网络上使用网络资源并进行通信。

通常情况下，会话层每请求建立一个传输连接，传输层就为其创建一个独立的网络连接。如果传输连接需要较高的信息吞吐量，传输层也可以为之创建多个网络连接，让数据在这些网络连接上分流，以提高吞吐量。另一方面，如果创建或维持一个网络连接不合算，传输层可以将几个传输连接复用到一个网络连接上，以降低费用。在任何情况下，都要求传输层能使多路复用对会话层透明。

传输层是真正的从源到目标“端到端”的层，也就是说，源端机上的某程序，利用报文头和控制报文与目标机上的类似程序进行对话。在传输层以下的各项层中，协议是每台机器和它直接相邻的机器间的协议，而不是最终的源端机和目标机之间的协议，在他们中间可能还有多个路由器。图 1-1 说明了这种区别， 1 层 -3 层是链接起来的， 4 层 -7 层是端到端的。

TCP 协议工作在本层，它提供可靠的基于连接的服务。它在两个端点之间提供可靠的数据传送，并提供端到端的差错恢复与流控。

1.1.5 会话层

会话层（ session layer ）允许不同机器上的用户之间建立会话关系，即正式的连接。这种正式的连接使得信息的收发具有高可靠性。会话层的目的就是有效地组织和同步进行合作的会话服务用户之间的对话，并对它们之间的数据交换进行管理。

会话层服务之一是管理对话，它允许信息同时双向传输，或任意时刻只能单向传输。约属于后者，则类似于单线铁路，会话层将记录此时该轮到哪一方了。一种与会话有关的服务是令牌管理（ token management ），令牌可以在会话双方之间交换，只有持有令牌的一方可以执行某种关键操作。

另一种会话服务是同步（ synchronization ）。同步是在连续发送大量信息时，为了使发送的数据更加精细地结构化，在用户发送的数据中设置同步点，以便记录发送过程的状态，并且在错误发生导致会话中断时，会话实体能够从一个同步点恢复会话继续传送，而不必从开头恢复会话。

TCP/IP 协议体系中没有专门的会话层，但是在其传输层协议 TCP 协议实现了本层部分功能。

1.1.6 表示层

表示层（ presentation layer ）完成某些特定的功能，由于这些功能常被请求，因此人们希望找到通用的解决办法，而不 是要让每个用户来实现。值得一提的是，表示层以下的各层只关心可靠的传输比特流，而表示层关心的是所传输的信息的语法和语义。

表示层尚未完整定义和广泛使用，如 TCP/IP 协议体系中就没有定义表示层。表示层完成应用层所用数据所需要的任何转换，以提供标准化的应用接口和公共的通信服务。如数据格式转换、数据压缩 / 解压和数据加密 / 解密可能在表示层进行。

1.1.7 应用层

应用层（ application layer ）包含大量人们普遍需要的协议。本层处理安全问题与资源的可用性。最近几年，应用层协议发展很快，经常用到的应用层协议有：FTP 、 TELNET 、 HTTP 、 SMTP 等。

OSI 模型的各层之间任务明确，它们只与上下相邻层打交道：接受下层提供的服务，向上层提供服务。由于所有的网络协议都是分层的，象堆栈一样，因此经常将协议各层统称协议栈。它们的工作模式一般为：发送时接收上层的数据，将其分割打包，然后交给下层；接收时接收下层的数据包，将其拆包重组，然后交给上层。这样，一个包的传输过程是：发送主机的应用程序将数据传递给网络协议栈实现（网络通信程序），网络协议栈实现将数据层层打包，最后交由物理层在数据链路上发送；接收主机收到数据后，逐层拆包向上传递，直到最后达到应用层，应用程序得到对等方的数据。

1.2 TCP/IP 协议的体系

Internet 采用的是 TCP/IP 协议体系， TCP/IP 协议体系是因其两个著名的协议 TCP 和 IP 而得名的。TCP/IP 协议体系在和 OSI 的竞争中取得了决定性的胜利，得到了广泛的认可，成为了事实上的网络协议体系标准。Linux 系统也是采用 TCP/I 体系结构进行网络通讯。

TCP/IP 协议体系和 OSI 参考模型一样，也是一种分层结构。它是由基于硬件层次上的四个概念性层次构成，即网络接口层、互联网层、传输层和应用层。图 1-2 表示了 TCP/IP 协议体系及其与 OSI 参考模型的对应关系。

从图 1-2 可以看出，对照 OSI 七层协议， TCP/IP 第三层以上是应用层、传输层和网际互联层， TCP/IP 的应用层组合了 OSI 的应用层和表示层，还包括 OSI

会话层的部分功能。但是，这样的对应关系并不是绝对的，它只有参考意义，因为 TCP/IP 各层功能和 OSI 模型的对应层还是有一些区别的。

<table>
<tr><td>OSI 模型</td><td colspan="5">TCP/IP 协议体系</td></tr>
<tr><td>应用层</td><td rowspan="3">应用层</td><td colspan="2" rowspan="3">Telnet
FTP
HTTP
SMTP</td><td colspan="2" rowspan="3">TIME
DNS
SNMP
TFTP</td></tr>
<tr><td>表示层</td></tr>
<tr><td>会晤层</td></tr>
<tr><td>传输层</td><td>传输层</td><td colspan="2">TCP</td><td colspan="2">UDP</td></tr>
<tr><td>网络层</td><td>互联网层</td><td colspan="4">IP ICMP</td></tr>
<tr><td>数据链路层</td><td>网络接口层</td><td>Ethernet</td><td>Token-Ring</td><td>PPP</td><td>Other Media</td></tr>
<tr><td>物理层</td><td>硬件</td><td colspan="4">Hardware</td></tr>
</table>

图 1-2 TCP/IP 协议体系及其与 OSI 模型的对应关系

1.2.1 TCP/IP 分层工作原理

TCP/IP 协议体系和 OSI 模型的分层结构虽然不完全相同，但它们的分层原则是一致的，即都遵循这样的一个思想：分层的协议要被设计成达到这样的效果，即目标机的第 n 层所收到的数据就是源主机的第 n 层所发出的数据。

图 1-3 描述了 TCP/IP 分层工作原理，它表示了两台主机上的应用程序之间传输报文的路径。主机 B 上的第 n 层所收到的正是主机 A 上的第 n 层所发出的对象。

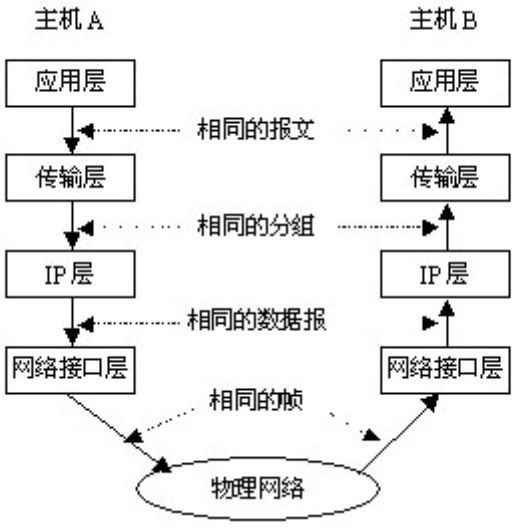


图 1-3

在图 1-3 中我们忽略了一个重要的内容，即没有描述发送方主机上的应用程序与接收主机的应用程序之间通过路由器进行报 文传输的情况。

图 1-4 中描述使用路由器的 TCP/IP 分层工作，图中报文经历了两种结构不同的网络，也使用了两种不同的网络帧，即一个是从主机 A 到路由器 R ，另一个是从路由器 R 到主机 B 。主机 A 发出的帧和路由器 R 接收到的帧相同，但不同于路由器 R 和主机 B 之间传送的帧。与此形成对照的是应用程序层和传输层处理端到端的事务，因此发送方的软件能和最终的接收方的对等层软件进行通信。也就是说，分层原则保证了最终的接收方的传输层所收到的分组与发送方的传输层送出的分组是一样的。

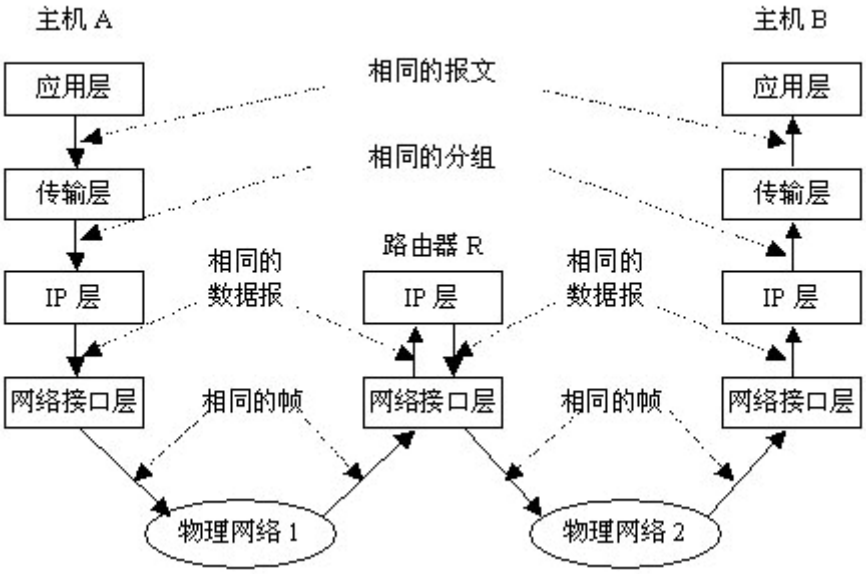


图 1-4

TCP/IP 的概念性层次包含两个重要的分界线，一个是协议地址分界线，以区分高层和低层的寻址，另一个是操作系统分界线，以区分系统与应用程序。图 1-5 描述了 TCP/IP 概念层模型的分界。

	概念性层次	分界线
	应用层	操作系统之外的软件
	传输层	操作系统之外的软件
	IP 层	只用 IP 地址
	网络接口层	用物理地址
	硬件	

图 1-5

在整个层次结构中，通信协议使用了复用和分解的技术。当发送报文时，发送方在报文中加入了报文类型、选用的协议等附加信息。所有的报文以帧的形式在网络中复用传送，形成一个分组流。在接收方收到分组时，参考附加信息对接收到的分组进行分解。图 1-6 给出了一个分解的例子，它描述了网络接口层如何根据帧的报头中的类型字段对接收到的帧进行分解。

当网络接口层完成对帧的分解操作之后，它要把那些包含有 IP 数据报的帧送

给 IP 模块， IP 软件模块分析这些帧以后，参照传输层协议对它们进行进一步的分解操作。图 1-7 描述了 IP 层的分解， IP 层的软件模块检查数据报报头，根据其中的协议类型选择相应的协议进行处理。

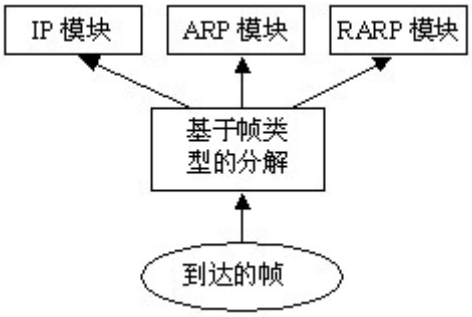


图 1-6

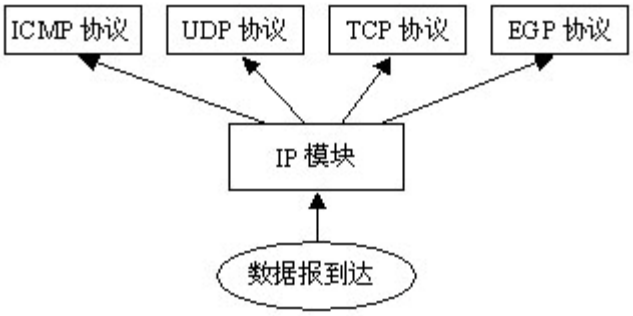


图 1-7

1.2.2 TCP/IP 的几个常用概念

TCP/IP 协议体系及其实现中有很多概念和术语，为了方便大家理解后面的内容，本节集中介绍一些最常用的概念与术语。

1.2.2.1 包（packet）

包（ packet ）是网络上传输的数据片段，也称分组。在计算机网络上，用户数据要按照规定划分为大小适中的若干组，每个组加上包头构成一个包，这个过程称为封装（ encapsulation ）。网络上使用包为单位传输的目的是为了更好地实现资源共享和检错、纠错。包是一种统称，在不同的协议不同的层次，包有不同的名字，如 TCP/IP 协议中，数据链路层的包叫帧（ Frame ）， IP 层的包称为 IP 数据报，TCP 层的包常称为 TCP 报文等。应用程序自己也可以设计自己的包类型，如在自己设计的 socket 程序中使用包。图 1-8 是 IP 数据报格式。

图 1-8

1.2.2.2 网络字节顺序

由于不同体系结构的计算机存储数据的格式和顺序都不一样，要建立一个独立于任何特定厂家的机器结构或网络硬件的互联网，就必须定义一个数据的表示标准。例如，将一台计算机上的一个 32 位的二进制整数发送到另一台计算机，由于不同的机器上存储整数的字节顺序可能不一样，如 Intel 结构的计算机存储整数是低地址低字节数，即最低存储器地址存放整数的低位字节（叫做小端机， Little Endian ），而 Sun Sparc 结构的计算机存储整数是低地址高字节数，即最低存储器地址存放整数的高位字节（叫做大端机， Big Endian ）。因此，直接把数据按照恒定的顺序发送到另一台机器上时可能会改变数字的值。

为了解决字节顺序的问题， TCP/IP 协议定义了一种所有机器在互联网分组的二进制字段中必须使用的网络标准字节顺序（ network standard byte order ）：必须首先发送整数中最高有效字节（同大端机）。因此，网络应用程序都要求遵循一个字节顺序转换规则：“主机字节顺序—— > 网络字节顺序—— > 主机字节顺序”，即发送方将主机字节顺序的整数转换为网络字节顺序然后发送出去，接收方收到数据后将网络字节顺序的整数转换为自己的主机字节顺序然后处理。

1.2.2.3 地址解析协议 ARP

严格地说，地址解析协议 ARP 并不是 TCP/IP 协议体系的一部分，但是在 Internet 和以太网盛行的今天，作为物理地址到 IP 地址转换的一个协议标准，ARP 协议的重要性显得特别突出。因此本节专门介绍 ARP （ Address Resolution Protocol ）协议。

我们前面介绍的 TCP/IP 网络使用 IP 地址进行寻址， IP 地址是一种网际地址，它工作在第三层。对于所有的计算机通信，最终表现为某种形式的链路（如以太网、令牌环、 FDDI 和 PPP ）上将数据从节点移动到另一节点。许多链路能连接多个节点，这就要求在其上发送的所有数据都应标明目的地址，以便正确传递。

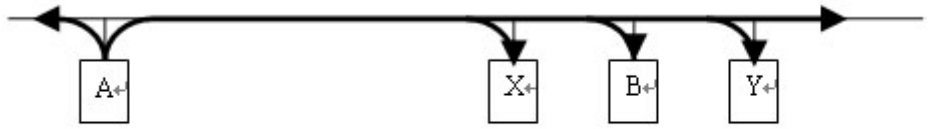
这些地址与 IP 地址无关，它们是完全独立的，处于 IP 地址以外，这就是设备的物理硬件地址。具体到以太网络，其物理硬件地址就是我们常说的 MAC 地址。递交一个网际报文分组，网络软件必须使 IP 地址最终变换成物理硬件地址，使用硬件地址来传输帧。

和 IP 地址不同， MAC 地址是设备内嵌的，它在设备出厂前就定好了。 MAC 地址是一个 6 字节的串，通常写成 16 进制数，以冒号分隔，如我的一台机器的网卡地址就是 00:60:08:C4:51:07 。 MAC 地址由电气和电子工程师学会（ IEEE ）分配，在所有以太网设备中是唯一的。 MAC 地址分两部分，前三个字节是厂商代码，后三个字节是设备编号。厂商必须确保它生产的以太网设备都具有同样的前三个字节（厂商代码），以及不同的后三个字节。

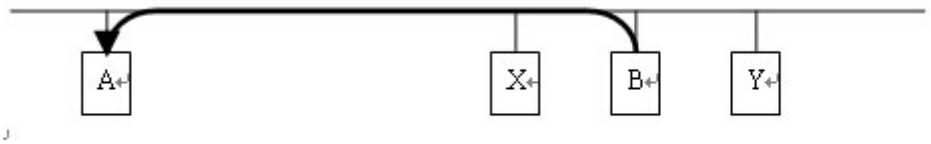
TCP/IP 网络使用 IP 地址寻址， IP 包在 IP 层实现路由选择。但是 IP 包在数据链路层的传输却需要知道设备的物理地址，因此需要一种 IP 地址到物理地址的转换协议。 TCP/IP 协议栈使用一种动态绑定技术，来实现一种维护起来既高效又容易的机制，这就是地址解析协议 ARP 。

ARP 协议是 TCP/IP 协议的设计人员为象以太网这种有广播能力的网络找到的一种创造性地解决地址转换问题的方法。这种办法允许在不重新编译代码、不需维护一个集中式数据库的情况下，在网络中动态增加新机器。这种办法就是动态地址转换。

如图 1-9 所示， ARP 动态地址转换的基本思想很简单：当主机 A 想转换 IP 地址 I B 时，它就广播一个专门的报文分组，要求具有 IP 地址 I B 的主机以其物理地址 P B 做出应答。包括 B 在内的所有主机都收到这个请求，但是只有主机 B 才辨认其 IP 地址，发回一个回答，回答中包含其物理地址。当 A 收到回答时，它便知道 B 的物理硬件地址，并使用这个地址直接把网际报文分组发送给 B 。



(a) 主机 A 把含有 I_B 的 ARP 请求广播给所有计算机



(b) 主机 B 以含有地址对（I_B，P_B）的 ARP 回答作为应答

图 1-9

2. 配置 Linux 系统的 IP 地址

在 RedHat Linux 中，要设置主机的 IP 地址，通用的方法是直接更改脚本。例如，当主机中只有一块网卡时，其设备名为 eth0，此时要给该网卡设置 IP 地址，只需要修改/etc/sysconfig/network-scripts/ifcfg-eth0 文件即可，下面是一个 ifcfg-eth0 文件的示例：

```
DEVICE=eth0
BOOTPROTO=static
BROADCAST=192.168.10.255
IPADDR=192.168.10.1
NETMASK=255.255.255.0
NETWORK=192.168.10.0
ONBOOT=yes
```

本示例表明了如何设置 Linux 主机的 IP 地址。每个字段的含义如下：

DEVICE　　设备名

BOOTPROTO　　使用动态 IP 还是静态 IP。此处为静态，如要使用动态（自动获取 IP），将 static 改为 dhcp 即可

BROADCAST　广播地址

IPADDR　　IP 地址

NETMASK　　子网掩码

NETWORK　　网络号（子网）。

设置完 ifcfg-eth0 文件后，需要执行如下命令才能使设置生效（重启网络服务）：

```
[root@gw root]#/etc/rc.d/init.d/network restart
```

如果主机上有两块网卡，要设置第二块网卡的 IP 时，就不是更改 ifcfg-eth0 文件了，此时第二块网卡的设备名叫 eth1，要修改的文件应该是在同一目录下的 ifcfg-eth1 文件。

那么如何来查看所进行的设置已经生效了呢？可以用 ifconfig 命令，此命令可以用来查看网络设置，也可以用来更改网络设置。执行 ifconfig | more 命令，会出现类似如图 1-10 所示的内容。

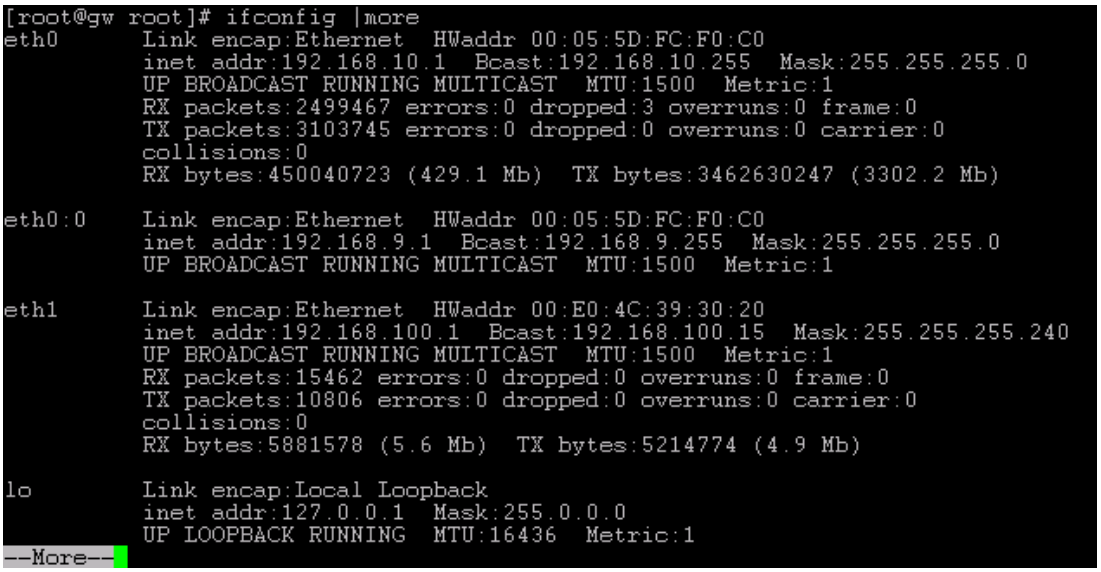


图 1-10

从图 1-10 中可以看出，主机有两块网卡 eth0 和 eth1，但是其中还有一个 eth0:0 是什么意思呢？仔细观察的话会发现它的 MAC 地址（HWaddr）与 eth0 相同，实际上是给 eth0 网卡绑定了两个 IP 地址。可以执行下面的命令来给 eth0 绑定第二个 IP 地址：

[root@koorka root]#ifconfig eth0:0 192.168.9.1 netmask 255.255.255.240 broadcast 192.168.9.15

同理，如果要给 eth0 网卡绑定第三个 IP 地址，只需要把 eth0:0 换成 eth0:1 即可。

当用 ifconfig 给网卡设置或更改 IP 地址时，重启系统就返回到原始设置了，要使设置的 IP 地址永久生效，一个方法就是前面所讲的更改/etc/sysconfig/network-scripts/ifcfg-eth0 文件；另一个方法就是把该命令放在系统的启动脚本/etc/rc.d/rc.local 中，这样，每次开机系统都会执行该文件，不用每次登录进去运行 ifconfig 命令。

ifconfig 命令还可以用来修改网卡的 Mac 地址、启用和禁用网卡，其基本语法如下：

ifconfig 网络端口 IP 地址 hw <HW> MAC 地址 netmask 掩码地址 broadcast 广播地址 [up/down]

3. 设置主机名

在一个网络环境中，主机名是识别某个计算机的唯一标识，而在一个单机环境中，主机名只给出了系统的称呼。

设置主机名的方法很简单，只需要用 hostname 命令即可。如果是在一个网络中，主机名必须是计算机的主机名的全称，如 www.koorka.com；如果不是网络环境，则可以设置任意的主机名和域名，如：mail.koorka.com、www.sme-solution.com。

在 Linux 系统中，要设置主机名称，可使用 hostname 命令：

[root@koorka ~]#hostname www.koorka.com

设置完主机名后，如果主机名称没有通过 DNS 服务器解析，则必须在本机的/etc/hosts 文件中进行解析。它给每个主机赋一个 IP 地址，即使计算机不在网络上，在/etc/hosts 中也会包含用户自己的主机名。

例如，如果主机不在 TCP/IP 网络中，主机名为 www.koorka.com，则在/etc/hosts 中包含下列内容：

```
127.0.0.1               www localhost.localdomain localhost
```

它给主机 localhost.localdomain 赋了一个回送地址 127.0.0.1（如果不在网络上就使用该地址），别名 localhost 也赋给该地址。

如果在 TCP/IP 网络上，真正的 IP 地址和主机名也应该在/etc/hosts 中，例如，如果主机名为 www.koorka.com，IP 地址为 202.127.124.110，则在/etc/hosts 中添加如下行：

```
202.127.124.110    www.koorka.com
```

用上面的方法设置的主机名只是当前生效，如果重新启动系统，则主机名又恢复了原来的设置。想让新修改的主机名永久生效，则必须要修改/etc/sysconfig/network 文件，在该文件中，添加（修改）一行：

```
HOSTNAME=koorka
```

4. 设置缺省网关

设置好 IP 地址以后，如果要访问其它的子网或者 Internet，用户还需要设置路由，在此不做介绍，这里采用设置缺省网关的方法。在 Linux 中，设置缺省网关有两种方法，第一种方法就是直接使用 route 命令，在设置缺省网关之前，我们先用 route –n 命令查看路由表，如图 1-11 所示。

```
[root@pmgw etc]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.5.0     0.0.0.0         255.255.255.240 U     0      0        0 eth1
10.1.1.0        0.0.0.0         255.255.255.0   U     0      0        0 pentava10
192.168.10.0    0.0.0.0         255.255.255.0   U     0      0        0 eth0
127.0.0.0       0.0.0.0         255.0.0.0       U     0      0        0 lo
[root@pmgw etc]#
```

图 1-11

执行如下命令设置网关：

```
[root@pmgw etc]# route add default gw 192.168.10.254
```

设置了网关后的路由表如图 1-12 所示。

```
[root@pmgw etc]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.5.0     0.0.0.0         255.255.255.240 U     0      0        0 eth1
10.1.1.0        0.0.0.0         255.255.255.0   U     0      0        0 pentaval0
192.168.10.0    0.0.0.0         255.255.255.0   U     0      0        0 eth0
127.0.0.0       0.0.0.0         255.0.0.0       U     0      0        0 lo
0.0.0.0         192.168.10.254  0.0.0.0         UG    0      0        0 eth0
[root@pmgw etc]#
```

图 1-12

增加缺省网关的第二种方法是在/etc/sysconfig/network 文件中添加如下字段：

```
GATEWAY=192.168.10.254
```

同样，只要是更改了脚本文件，必须重启网络服务来使设置生效，可执行下面的命令：

```
[root@gw root]#/etc/rc.d/init.d/network restart
```

对于第一种方法，如果不想每次开机都执行 route 命令，则应该把要执行的命令写入/etc/rc.d/rc.local 文件中。

5. 设置 DNS 服务器。

设置 DNS 服务器的方法比较简单，只需修改/etc/resolv.conf 文件即可。下面是一个 resolv.conf 文件的示例：

```
nameserver 192.168.10.1
nameserver 202.106.196.115
```

其中 192.168.10.1 为第一名字服务器，202.106.196.115 为第二名字服务器。

以上示例是当用户的计算机作为一台客户机使用时所进行的主要设置。

下面是配置一台主机的网络的一个实例：

现有一局域网，其网关为 192.168.100.1，DNS 服务器为 192.168.100.1，下面为一台 Linux 客户机（IP 地址为 192.168.100.3，子网掩码为 255.255.255.240），设置网络，使其能通过 192.168.100.1 的网关上网，并把其主机名设置为 myhost。

操作步骤如下：

（1）设置 IP 地址和网关。只要编辑/etc/sysconfig/network-scripts/ifcfg-eth0 文件即可，该文件的内容如下：

```
DEVICE=eth0
BOOTPROTO=static
```

```
IPADDR=192.168.100.3
NETMASK=255.255.255.240
GAYEWAY=192.168.100.1
ONBOOT=yes
```

具体含义请参看基础知识部分。

（2）设置 DNS 服务器。只需编辑/etc/resolv.conf 文件即可，将/etc/resolv.conf 的内容设置如下：

```
nameserver 192.168.100.1
```

（3）设置主机名。只需修改/etc/sysconfig/network 文件，network 文件的内容如下：

```
NETWORKING=yes
HOSTNAME=myhost
```

（4）为了解析本机的 NetBIOS 名，可将/etc/hosts 文件修改为如下内容：

```
127.0.0.1          myhost localhost.localdomain localhost
192.168.100.3       myhost
```

（5）使设置生效。此时需要重启网络服务，可执行下面的命令：

```
[root@gw sysconfig]#/etc/rc.d/init.d/network restart
```

对于主机名，必须重启计算机后才能生效。至此网络设置已经完成，如果外部路由已经开通，则可以 ping 一个外部地址试试看，比如 ping 202.106.196.115。

对于网络设置，除了上面的方法外，我们还有另外两种方法：

1．使用 RedHat 自带的 netconfig 工具

（1）在系统提示符下运行 netconfig 命令，出现如图 1-13 所示的界面。

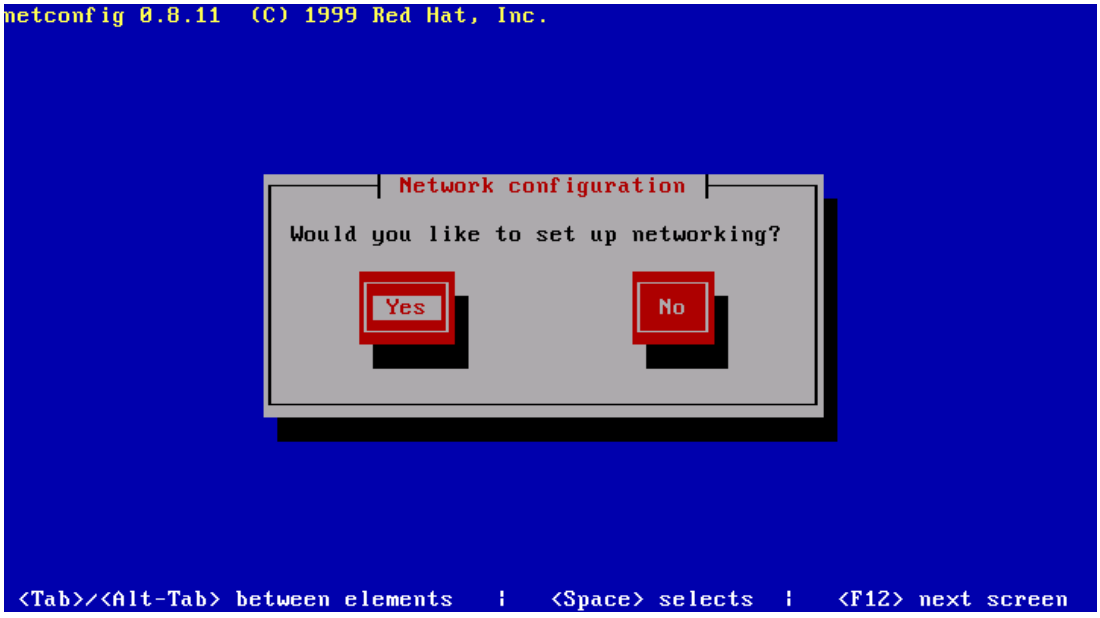


图 1-13

此时选择 Yes 表示要更改设置，按回车键后出现图 1-14 所示的界面。

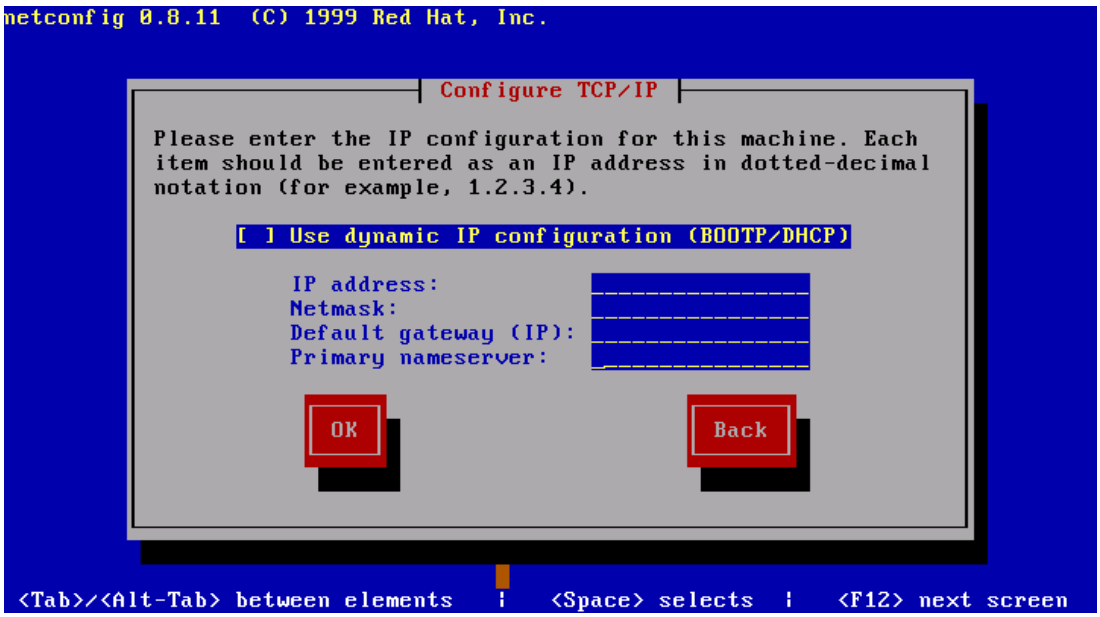


图 1-14

分别在对应的栏目内填入对应的值即可（如图 1-15 所示），如果想要设置成自动获取 IP 地址（通过 DHCP 服务器分配 IP 地址），只需把上面的 Use dynamic IP configuration （BOOTP/DHCP）项选中即可。

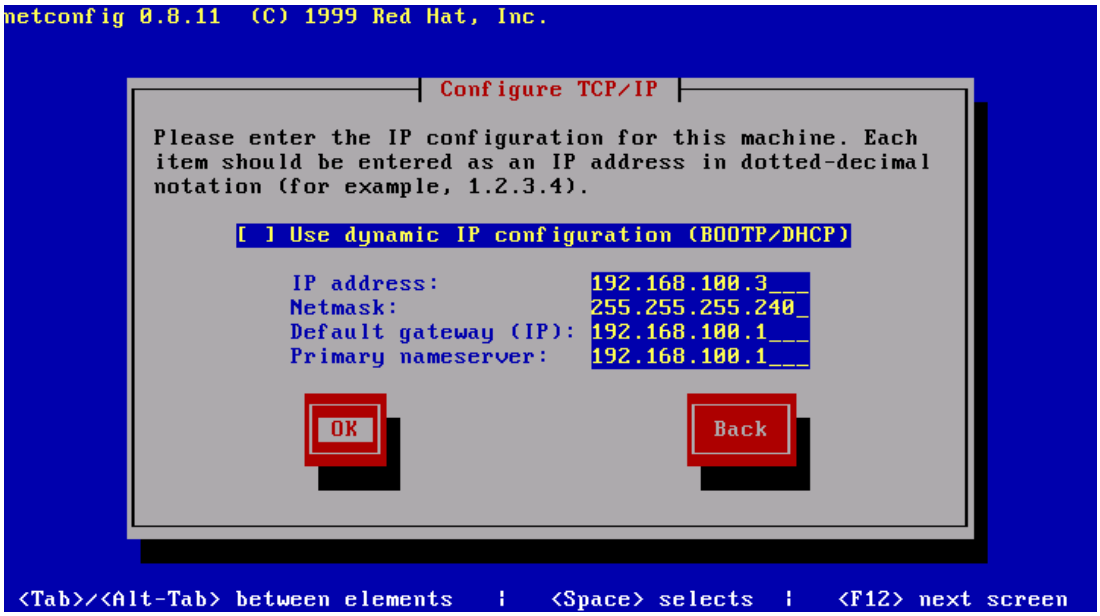


图 1-15

设置完成后，查看/etc/sysconfig/network-scripts/ifcfg-eth0 文件，会发现 netconfig 命令实质上是更改/etc/sysconfig/network-scripts/ifcfg-eth0 文件的内容和/etc/resolv.conf 文件的内容，因此，也需要重新启动服务或者重启计算机来使设置生

效。

（2）重启服务，可以执行下面的命令：

```
[root@gw sysconfig]#/etc/rc.d/init.d/network restart
```

2．用 ifconfig 命令和 route 命令：

（1）添加（更改）IP 地址，执行如下命令：

```
[root@koorka sysconfig]#ifconfig eth0 192.168.100.3 netmask 255.255.255.240 broadcast 192.168.100.14
```

因为使用的是非标准掩码，所以需要加上 broadcast 参数。

（2）设置缺省路由，执行如下命令：

```
[root@koorka sysconfig]#route add default gw 192.168.100.1
```

（3）设置 DNS 服务器，与前面的方法相同，修改/etc/resolv.conf 文件：

```
nameserver  192.168.100.1
```

（4）上面执行的 ifconfig 和 route 命令写入/etc/rc.d/rc.local 中。因为下次再开机时，系统又会恢复以前的设置，从前面所讲的系统的启动步骤一节中了解到/etc/rc.local 是在系统启动的最后一步才执行的，将上面的命令都写入此文件中后，则每次开机系统都会执行此命令。添加完成后/etc/rc.d/rc.local 文件的内容如下：

```
#!/bin/sh
#
# This script will be executed *after* all the other init scripts.
# You can put your own initialization stuff in here if you don't
# want to do the full Sys V style init stuff.

touch /var/lock/subsys/local
ifconfig eth0 192.168.100.3 netmask 255.255.255.240 broadcast 192.168.100.14
route add default gw 192.168.100.1
```