这章介绍IP协议和它们在Java中的实现.IP代表'互联网协议',它是互联网的基础协议,像胶水一样将网络都连接在一起.

2.1 IP

如RFC 791所说,'互联网协议被设计使用在包-交换的计算机通信网络的互相连接的系统中'.互联网仅仅是一个大量的这样的系统的通信,通过IP协议,各种类型的包-交换网络,包括以太网和令牌环网,电话线路和卫星通信电路.

IP是协议族最基本的元素统称为TCP/IP,由子协议组成,比如 ARP-地址解析协议, RARP-反向地址解析协议, ICMP-互联网控制消息协议, BOOTP-引导协议, IGMP-因特网组管理协议, UDP-用户数据报, 和TCP-传输控制协议.这本书处理TCP和UDP;其它提到的协议可通过各种方式支持TCP和UDP,通常不是网络程序员所要关注的.

IP由(1)主机寻址系统, (2)IP数据包格式定义, 和(3)适当的协议-关于传输和接受包的规则组成.

目前存在两种版本的IP: IPv4,第一个公开的可用的协议版本, IPv6, 在写此书的时候使用是有限的,提供了地址空间的大量扩展,而且有大量的改进和新特性.

2.2 网络寻址

2.2.1 网络接口

一个网络主机通过一个或多个网络接口连接到网络: 这些是硬件设备,通常表现为控制卡(controller cards)(网络接口控制器或者NICs).每个物理网络接口可以有一个或多个IP地址,在随后的章节讨论.这样,每个网络主机至少有一个IP地址.这个主题在将2.3节进一步讨论.

2.2.2 IP地址

一个互联网主机通过固定长度的'IP地址'标识.这是一个数字由一个'网络'或者'子网'部分(在互联网中唯一标识子网)和一个'主机'部分(在子网络中唯一标识主机)组成.

在IPv4中一个IP地址是一个32位数字,写成4个8位片段的'点分'形式,比如, 192.168.1.24 或 127.0.0.1.

在IPv6中一个IP地址是一个128位数字,写成冒号分隔的8个比特一段的每段4位十六进制数,比如 0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:1,为了方便,有许多4位0的两个邻近的冒号有必要表示成:给定的地址可以简写成: ::1.

2.2.3 域名

数字IP寻址系统通过一个被称为域名系统或者DNS的按照字母顺序的命名系统补全,复制主机名到该域中,提供了在IP地址和主机名之间的映射,这个过程被称为'解析'.

2.2.4 端口

每个互联网主机支持大量的IP '端口',在这台主机上代表了独立的服务,通过一个16位'端口数'标识,范围在1-65535.这些端口中有许多是被预分配的,'众所周知的端口'范围在1-1023,'注册的端口'在范围1024-49151(0x0400-0xbfff).在"众所周知的端口"的服务在某些操作系统需要特殊的权限,比如,在Unix-style系统需要超级用户权限.

一个指定的TCP或者UDP服务器可通过{IP地址,端口}对被寻址.这个对被称为一个"套接字地址(socket

address)".

2.2.5 套接字

在主机中一个通信端点通过一个称为套接字的抽象概念表示.一个套接字在它的本机中与一个IP地址和一个端口关联.在Java中,一个套接字代表Socket,ServerSocket,DatagramSocket或者MulticastSocket其中的一个实例.

2.2.6 网络地址的类

在Java中,一个IP地址通过java.net.InetAddress代表.一个IP端口用一个范围在1-65535之间的整型代表,大部分通常是1024或者在这之上.一个IP套接字地址在Java中既可以通过一个{IP地址,端口}对表示,也可以通过JDK 1.4 SocketAddress类,分装了{IP地址,端口}对.

各种Java网络地址类的目的和使用展现在表格2.1.

表格 2.1 网络地址类

名称	描述
InetAddress	代表一个IP地址或者一个解析的主机名：用于远程地址.如果主机名解析失败,这个对象不能被构造.
InetSocketAddress extends SocketAddress	代表一个IP套接字地址,也就是说,一个{IP地址,端口}或者{主机名,端口}对.对于后者({主机名,端口})尝试去解析主机名当构造对象的时候,但是对象仍然可用在一些情况中比如通过一个代理连接如果解析失败.可以只使用一个{端口}构造,在'通配符'本地IP地址使用的情况下,意味着'所有的本地接口'.
NetworkInterface	代表一个本地网络接口,由一个接口名称(比如, 'leo')和一系列与接口关联的IP地址组成.在组播中用来标识本地接口.

从JDK 1.4,InetAddress类是抽象的,有两个衍生的类：用于IPv4的Inet4Adress和用于IPv6的Inet6Address.实际上你不需要去关心这些衍生类的存在.你不能构造它们：你通过InetAddress的静态方法获取这些类的实例,通常你最好假设它们是InetAddress的实例.这些衍生类之间唯一的区别从程序员的角度看就是Inet6Adress.isIpV6CompatibleAddress方法,如果'地址是一个IP4v兼容IPv6地址'则返回true;如果地址是一个IPv4地址,返回false.这是一个需要关心的很少见的Java程序.

2.2.7 特殊的IP地址

除了IP地址属于它的网络接口,一个互联网主机也有两个特别的IP地址,只在这台主机中可用,如表格2.2所示.

表格 2.2 特殊的IP地址

名称	IPv4	IPv6	描述
回路地址(loopback)	127.0.0.1	::1	用来标识服务本机在这样的情况下：主机的外部DNS或者IP地址不可用或者uninteresting,比如,在一个只用来打算在一个单独的主机中通信的系统.
通配符	0.0.0.0	::0	当创建的套接字去表明它们应该绑定到'所有的本地IP地址'而不是一个指定的时候使用.这是一个正常的情况.在Java中,它可以通过一个缺失的或者null的InetAddress指明.

InetAddress类有大量询问地址属性的方法.这些方法总结在表格2.3.

表格 2.3 InetAddress方法

名称	如果为'true'的意思
isAnyLocalAddress	通配符地址：见表格 2.2.
isLinkLocalAddress	链接本地单播地址.在IPv4中未定义.在IPv6中是一个以FE:8o开头的地址.
isLoopback	回路地址：见表格 2.2.
isMCGlobal	全局范围的组播地址.
isMCLinkLocal	链接-本地范围内的组播地址
isMCNodeLocal	节点-本地范围内的组播地址
isMCOrgLocal	组织-本地范围内的组播地址
isMCSiteLocal	站点-本地范围内的组播地址
isMulticastAddress	组播地址.在IPv4中这个一个范围在224.0.0.0到239.255.255.255内的地址;在IPv6中是以FF开始的地址.
isSiteLocal	站点-本地单播地址;在IPv4中未定义;在IPv6中这个一个以FE:CO开始的地址.

a.IPv6情况下参数在FRC 2.73中的规范.

isMCGlobal,isMCLinkLocal等方法返回关于组播地址范围内的信息将在II.1.4中讨论.

2.3 多宿主

一个多宿主主机是一个有多个IP地址的主机.这样的主机通常位于在两个IP子网的之间的网关,通常有多个物理网络接口.实际上只有在这样的主机上,编程者需要关心特殊的本地IP地址和网络接口.

网络接口实际上在JDK 1.4(引入了NetworkInterface类)之前是不可见的.从JDK 1.4开始,一个主机的网络接口可从下面的方法获得:

```
class NetworkInterface {
    static Enumeration getNetworkInterfaces() throws SocketException;
    Enumeration getInetAddresses();
}
```

getNetworkInterfaces返回一个NetworkInterface的枚举,getInetAddresses返回一个InetAddress的枚举,代表所有的或者绑定到一个单独的网络接口的可用的IP地址的子网.如果没有安全管理,列表是完整的;另外,任何InetAddress的访问通过安全管理的checkConnect方法不存在列表中的将会被拒绝.

因此可访问的被主机支持的IP地址,可用通过示例 2.1的代码顺序检索.

```
// Enumerate network interfaces(JDK >= 1.4)

Enumeration interfaces = NetworkInterface.getNetworkInterfaces();
while(interfaces.hasMoreElements()) {
    NetworkInterface inft = (NetworkInterface) interfaces.nextElement();

    // Enumerate IentAddress of this network interface
    Enumeration addresses = inft.getInetAddresses();
    while(addresses.hasMoreElement()) {
        InetAddress address = (InetAddress) addresses.nextElement();
        // ...
    }
```

```
}
```

Example 2.1 枚举本地网络接口

2.4 IPv6

Java一直支持IPv4,IP协议最初的版本.IPv6是IP的下一个版本,打算去提高IPv4的许多方面,包括效率,扩展性;32位IPv4地址空间;服务质量支持;传输级别的鉴定和隐私.

从JDK 1.4,Java也支持IPv6,若主机平台也这样,它对编程者的完全透明的.你的已经存在的Java网络程序自动支持IPv4和IPv6,如果在支持IPv6的平台上在JDK 1.4上执行: 你可以连接到IPv4和IPv4服务器,可以是一个IPv4和IPv6两者的服务器,可以从IPv4和IPv6两者的客户端接收连接.

2.4.1 兼容性

IPv6通过'IPv4-兼容地址'支持IPv4.这些是128位的高96位为0的IPv6地址.比如,IPv4地址是192.168.1.24可以作为IPv4-兼容地址::192.168.1.24用在IPv6中.

Java的IPv6支持可通过系统属性控制.这些允许你禁用IPv6支持,这样,只有IPv4客户端和服务器被支持.你不能通过这些属性禁用IPv4支持,尽管你可以通过指定IPv6网络接口达到同样的效果.当创建或绑定套接字或服务器套接字,在后面将有一个套接字选项去获得在每个套接字基础上的仅IPv6支持的行为.

这些系统属性在表格 2.4中描述.

表格 2.4 IPv6系统属性

名称	值	描述
java.net.preferIPv4Stack	false(default), true	默认情况下,IPv6本地的套接字被使用,如果可以的话,允许应用程序与IPv4和IPv6主机两者通信. 如果这个属性设置为true,IPv4本地套接字总是被使用.应用程序将不能与IPv6主机通信.
java.net.preferIPv6Address	false(default), true	默认情况下,如果IPv6可用,IPv4映射地址优先于IPv6地址,'为了向后兼容-比如,应用程序只基于IPv4服务,或者...在IPv4地址的["点分"表示]'. 如果这个属性设置为true,IPv6地址优先于IPv4-式样的地址,'允许应用程序去测试和发布在应用程序期望连接到IPv6服务的环境中'.

2.4.2 在Java中的编程差异

在任何你需要动态的决定你是有一个IPv4还是IPv6套接字的情况下,可以使用下面的技术:

```
if (socket.getLocalAddress() instanceof Inet6Address)
; // you have an IPv6 socket
else
; // you have an IPv4 socket
```

除了实际的IP地址的格式,java.net.Inet6SocketAddress类在2.2节描述,Socket.setTrafficClass方法在3.19描述,这是你需要去关注IPv4和IPv6,在整个java.net包中只要关注的点.