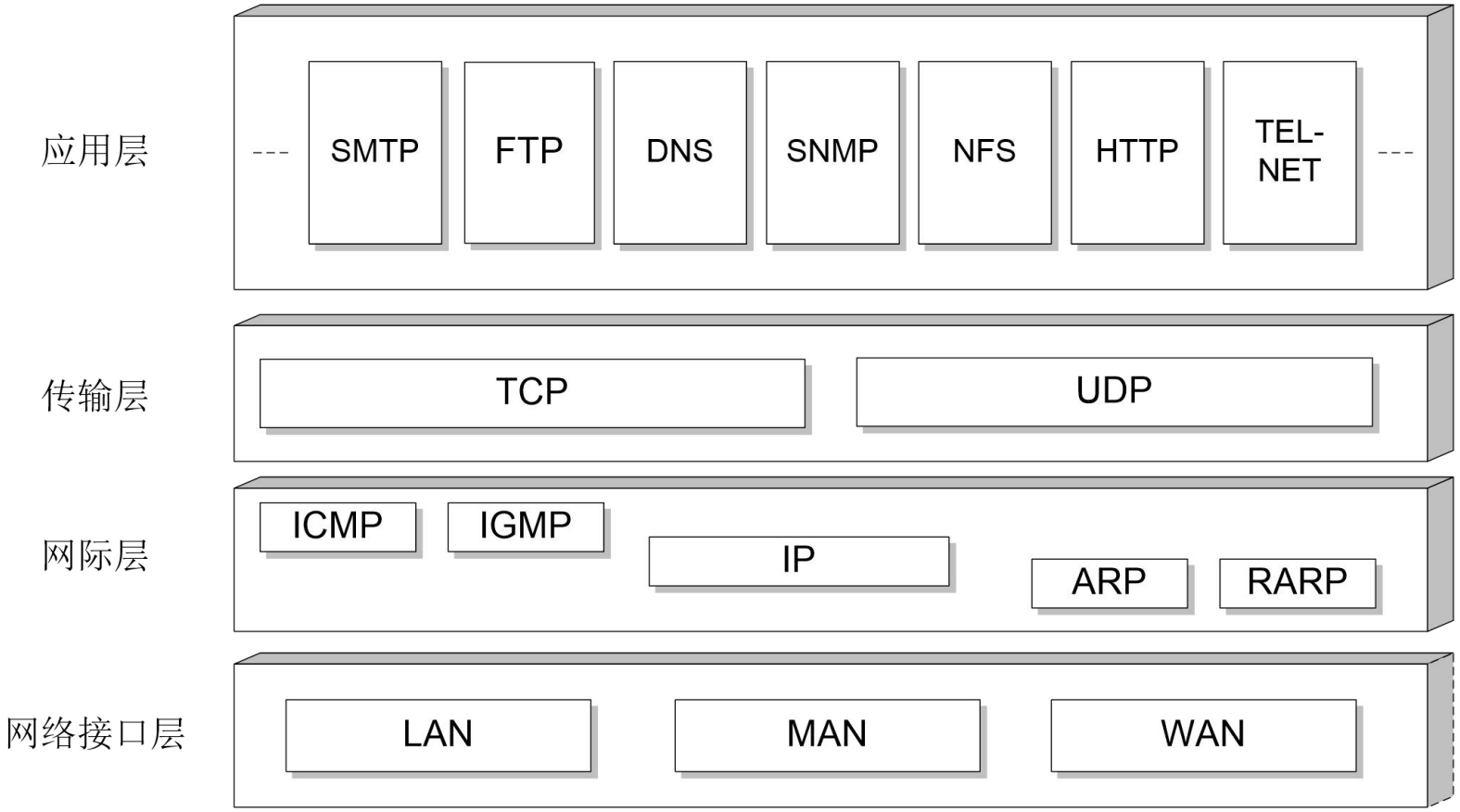
网络编程

SOFTCITS@2017

SoftCITS
SOFT CULTURE IT SALON
软件园IT文化沙龙

网络编程


应用层
SMTP
FTP
DNS
SNMP
NFS
HTTP
TEL-NET
传输层
TCP
UDP
网际层
ICMP
IGMP
IP
ARP
RARP
网络接口层
LAN
MAN
WAN

IP/TCP

IP协议是Internet上使用的一个关键协议，它的全称是Internet Protocol，即Internet协议，通常简称IP协议。通过使用IP协议，从而使Internet成为一个允许连接不同类型的计算机和不同操作系统的网络。

TCP协议被称作一种端对端协议。这是因为它为两台计算机之间的连接起了重要作用：当一台计算机需要与另一台远程计算机连接时，TCP协议会让它们建立一个连接：用于发送和接收数据的虚拟链路。
TCP协议负责收集这些信息包，并将其按适当的次序放好传送，在接收端收到后再将其正确地还原。TCP协议保证了数据包在传送中准确无误。TCP协议使用重发机制：当一个通信实体发送一个消息给另一个通信实体后，需要收到另一个通信实体确认信息，如果没有收到另一个通信实体的确认信息，则会再次重发刚才发送的信息。
通过这种重发机制，TCP协议向应用程序提供可靠的通信连接，使它能够自动适应网上的各种变化。即使在 Internet 暂时出现堵塞的情况下，TCP也能够保证通信的可靠。

端口

端口是一个16位的整数，用于表示数据交给哪个通信程序处理。因此，端口是应用程序与外界交流的出入口，它是一种抽象的软件结构，包括一些数据结构和I/O（基本输入/输出）缓冲区。

不同的应用程序处理不同端口上的数据，同一台机器上不能有两个程序使用同一个端口，端口号可以从0到65535，通常将它分为三类：

公认端口（Well Known Ports）：从0到1023，它们紧密绑定（Binding）一些服务。

注册端口（Registered Ports）：从1024到49151。它们松散地绑定一些服务。

动态和/或私有端口（Dynamic and/or Private Ports）：从49152到65535，这些端口是应用程序使用的动态端口，应用程序一般不会主动使用这些端口。

Socket/ServerSocket

客户端通常可使用Socket的构造器来连接到指定服务器，Socket通常可使用如下两个构造器：

Socket(InetAddress/String remoteAddress, int port)：创建连接到指定远程主机、远程端口的Socket，该构造器没有指定本地地址、本地端口，默认使用本地主机的默认IP地址，默认使用系统动态指定的IP地址。

ServerSocket对象用于监听来自客户端的Socket连接，如果没有连接，它将一直处于等待状态。ServerSocket包含一个监听来自客户端连接请求的方法：
Socket accept()：如果接收到一个客户端Socket的连接请求，该方法将返回一个与连客户端Socket对应的Socket.
为了创建ServerSocket对象，ServerSocket类提供了如下几个构造器：

ServerSocket(int port)：用指定的端口port来创建一个ServerSocket。该端口应该是有一个有效的端口整数值：0～65535。

网络通信

当客户端、服务器端产生了对应的Socket之后，程序无需再区分服务器、客户端，而是通过各自的Socket进行通信，Socket提供如下两个方法来获取输入流和输出流：

InputStream getInputStream()：返回该Socket对象对应的输入流，让程序通过该输入流从Socket中取出数据。

OutputStream getOutputStream()：返回该Socket对象对应的输出流，让程序通过该输出流向Socket中输出数据。

代码 1.1

思考

当有多个客户端同时访问服务器时，服务端又如何提供服务呢？

多线程网络通信

实际应用中的客户端则可能需要和服务器端保持长时间通信，即服务器需要不断地读取客户端数据，并向客户端写入数据；客户端也需要不断地读取服务器数据，并向服务器写入数据。

使用传统BufferedReader的readLine()方法读取数据时，当该方法成功返回之前，线程被阻塞，程序无法继续执行。考虑到这个原因，因此服务器应该每个Socket单独启动一条线程，每条线程负责与一个客户端进行通信。

客户端读取服务器数据的线程同样会被阻塞，所以系统应该单独启动一条线程，该线程专门负责读取服务器数据。

代码: 1.2

UDP协议

UDP(User Datagram Protocol)协议是一种不可靠的网络协议，它在通信实例的两端各建立一个Socket，但这两个Socket之间并没有虚拟链路，

UDP面向非连接的协议，通信前不必与对方建立连接，不管对方状态就直接发送。至于对方是否可以接受到这些内容，UDP协议无法控制，UDP适用于实时性很强的场景，如网络游戏，视频会议等。

这两个Socket只是发送、接收数据报的对象，Java提供了DatagramSocket对象作为基于UDP协议的Socket，使用DatagramPacket代表DatagramSocket发送、接收的数据报。

UDP协议

DatagramSocket本身只是码头，不维护状态，不能产生IO流，它的唯一作用就是接受和发送数据报，Java使用DatagramPacket来代表数据报，DatagramSocket接收和发送的数据都是通过DatagramPacket对象完成的。

DatagramSocket的构造器：

DatagramSocket()：创建一个DatagramSocket实例，并将该对象绑定到本机默认IP地址、本机所有可用端口中随机选择的某个端口。

DatagramSocket(int prot)：创建一个DatagramSocket实例，并将该对象绑定到本机默认IP地址、指定端口。

DatagramSocket(int port, InetAddress laddr)：创建一个DatagramSocket实例，并将该对象绑定到指定IP地址、指定端口。

DatagramSocket实例，通常在创建服务器时，我们创建指定端口的DatagramSocket实例——这样保证其他客户端可以将数据发送到该服务器。一旦得到了DatagramSocket实例之后，就可以通过如下两个方法来接收和发送数据：

receive(DatagramPacket p)：从该DatagramSocket中接收数据报。

send(DatagramPacket p)：以该DatagramSocket对象向外发送数据报。

代码:1.3

UDP与TCP

TCP协议：可靠，传输大小无限制，但是需要连接建立时间，差错控制开销大。
UDP协议：不可靠，差错控制开销较小，传输大小限制在64K以下，不需要建立连接。

谢谢观看

SeeYou！

SoftCITS 软件园IT文化沙龙

SOFT CULTURE IT SALON