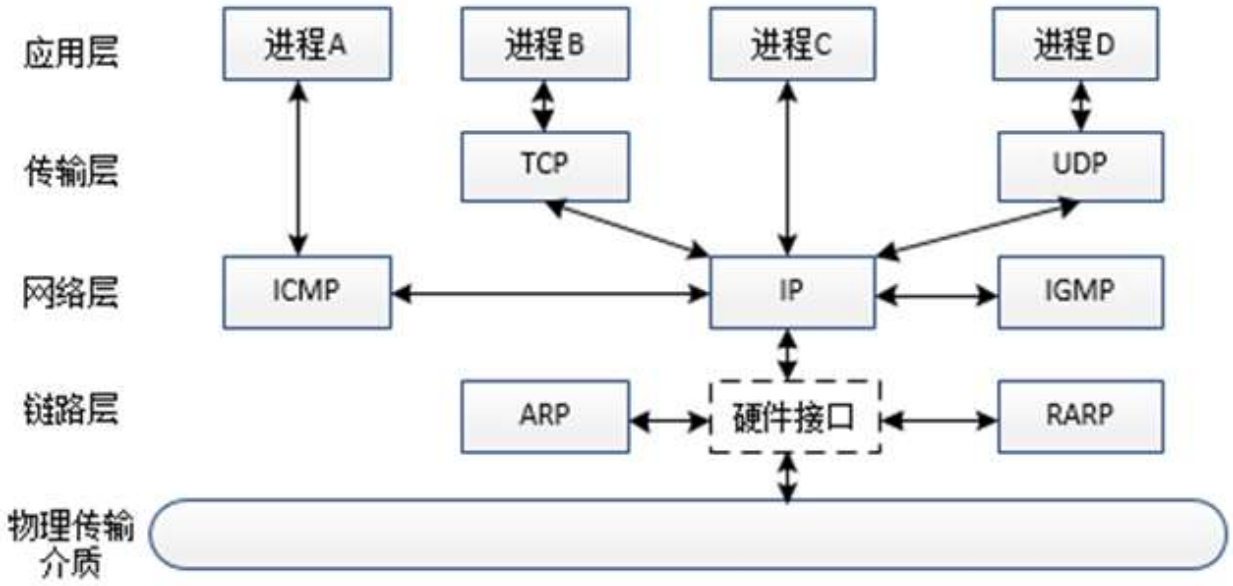
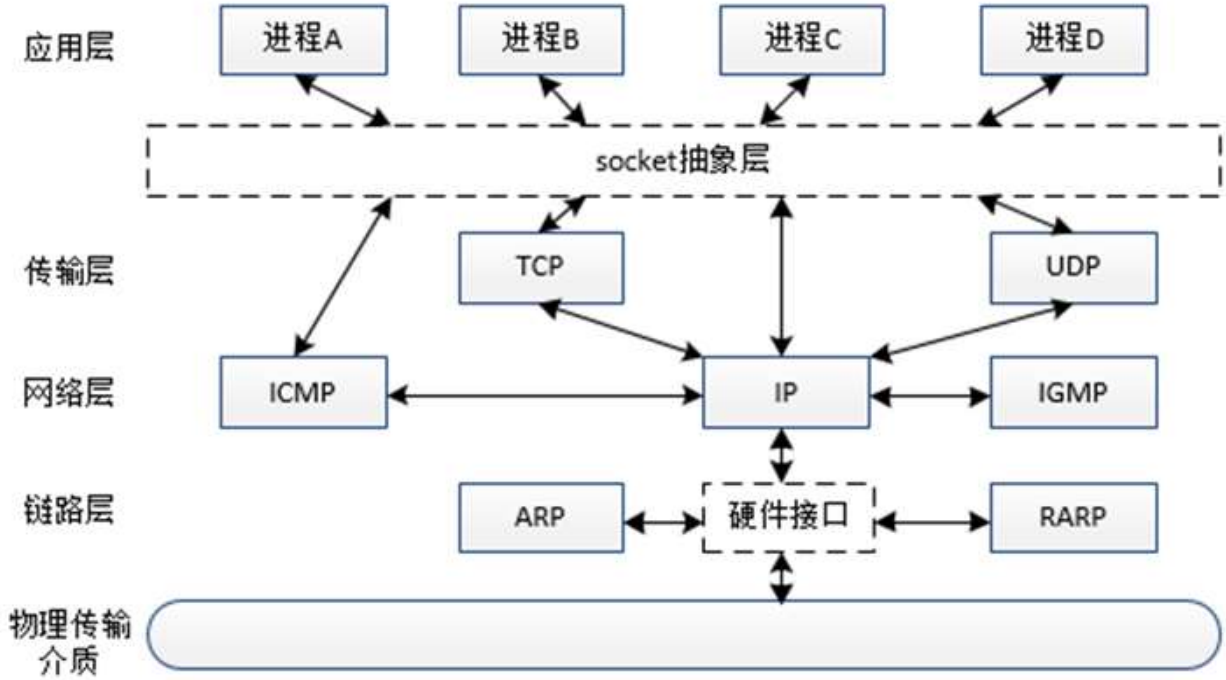
C10 Socket编程

计算机网络概述



socket编程基础

socket本意为“插座”，常被称为套接字，当使用socket进行通讯时，进程会先生成一个socket文件，之后再通过socket文件进行数据传递。



服务器专用接口

bind()

使服务器端的一个socket文件与网络中的一个进程进行绑定，因为文件描述符可标识socket文件，“主机名+端口号”可标识网络中的唯一进程，因此bind()函数实际上是将服务器端的socket文件与网络中的进程地址进程绑定。

listen()

listen()函数仍用于服务器端，从字面上看，其功能为使已绑定的socket监听对应客户端进程状态，但实际上，该函数用于设置服务器同时可建立的连接的数量。

accept()

accept()函数在listen()函数之后使用，其功能为阻塞等待客户端的连接请求。

当传输层使用TCP协议时，服务器与客户端在创建连接前，会先经过“三次握手”机制测试连接，“三次握手”完成后，服务器调用accept()函数处理连接请求，此时若还没有客户端的请求到达，便阻塞等待调用accept()函数的进程，直到接收到客户端发来的请求，且服务器中已创建的连接数未达到backlog，accept()函数才会返回，并传出客户端的地址。

recv()

该函数用于从已连接的套接字中接收信息

客户端专用接口

connect()

向服务器发起连接请求

send()

向处于连接状态的套接字中发送数据

服务器和客户端共用

socket()

socket()函数用于创建套接字，也可以说socket()函数用于打开网络通讯端口，该函数类似于文件操作中的open()函数，若调用成功，也返回一个文件描述符，之后应用程序可以采用socket通信中的读写函数在网络中收发数据；若调用失败会返回-1，并设置errno。

close()

用于释放系统分配给套接字的资源

socket网络编程实例

socket本地通信

socket原本是为网络通讯设计的，但后来在socket框架的基础上发展出了一种IPC（进程通信）机制，即UNIX Domain Socket，专门用来实现使用socket实现的本地进程通信。

本地通信的流程与使用的接口与基于TCP协议的网络通信模型相同，其大致流程如下：

（1）调用socket()函数通信双方进程创建各自的socket文件；

（2）定义并初始化服务器端进程的地址，并使用bind()函数将其与服务器端进程绑定；

（3）调用listen()函数监听客户端进程请求；

（4）客户端调用connect()函数，根据已明确的客户端进程地址，向服务器发送请求；

（5）服务器端调用accept()函数，处理客户端进程的请求，若客户端与服务器端进程成功建立连接，则双方进程可开始通信；

（6）通信双方以数据流的形式通过已创建的连接互相发送和接收数据，进行通信；

（7）待通信结束后，通信双方各自调用close()函数关闭连接。

与socket网络通信不同的是，在本地通信中用到的套接字的结构体类型为socket sockaddr_un。