##### Linux raw socket

我们平常所用到的网络编程都是在应用层收发数据，每个程序只能收到发给自己的数据，即每个程序只能收到来自该程序绑定的端口的数据。收到的数据往往只包括应用层数据。某些情况下我们需要执行更底层的操作，比如监听所有本机收发的数据、修改报头等。

通过原始套接字，我们可以抓取所有发送到本机的IP包（包括IP头和TCP/UDP/ICMP包头），也可以抓取所有本机收到的帧（包括数据链路层协议头）。普通的套接字无法处理ICMP、IGMP等网络报文，而SOCK\_RAW可以。利用原始套接字，我们可以自己构造IP头。

有两种原始套接字  
一种是处理IP层即其上的数据，通过指定socket第一个参数为AF\_INET来创建这种套接字。  
另一种是处理数据链路层即其上的数据，通过指定socket第一个参数为AF\_PACKET来创建这种套接字。

**AF\_INET**表示获取从网络层开始的数据  
socket(AF\_INET, SOCK\_RAW, …)  
当接收包时，表示用户获得完整的包含IP报头的数据包，即数据从IP报头开始算起。  
当发送包时，用户只能发送包含TCP报头或UDP报头或包含其他传输协议的报文，IP报头以及以太网帧头则由内核自动加封。除非是设置了IP\_HDRINCL的socket选项。  
如果第二个参数为SOCK\_STREAM, SOCK\_DGRAM，表示接收的数据直接为应用层数据。

**PF\_PACKET**，表示获取的数据是从数据链路层开始的数据  
socket（PF\_PACKET，SOCK\_RAW，htos（ETH\_P\_IP））：表示获得IPV4的数据链路层帧，即数据包含以太网帧头。14+20+(8:udp 或 20:tcp)  
ETH\_P\_IP: 在<linux/if\_ether.h>中定义，可以查看该文件了解支持的其它协议。  
SOCK\_RAW, SOCK\_DGRAM两个参数都可以使用，区别在于使用SOCK\_DGRAM收到的数据不包括数据链路层协议头。

总结起来就是：  
socket(AF\_INET, SOCK\_RAW, IPPROTO\_TCP|IPPROTO\_UDP|IPPROTO\_ICMP)发送接收ip数据包  
能：该套接字可以接收协议类型为(tcp udp icmp等)发往本机的ip数据包  
不能：收到非发往本地ip的数据包(ip软过滤会丢弃这些不是发往本机ip的数据包)  
不能：收到从本机发送出去的数据包  
发送的话需要自己组织tcp udp icmp等头部.可以setsockopt来自己包装ip头部  
这种套接字用来写个ping程序比较适合

socket(PF\_PACKET, SOCK\_RAW|SOCK\_DGRAM, htons(ETH\_P\_IP|ETH\_P\_ARP|ETH\_P\_ALL))发送接收以太网数据帧  
这种套接字比较强大，可以监听网卡上的所有数据帧

能: 接收发往本地mac的数据帧  
能: 接收从本机发送出去的数据帧(第3个参数需要设置为ETH\_P\_ALL)  
能: 接收非发往本地mac的数据帧(网卡需要设置为promisc混杂模式)  
协议类型一共有四个  
ETH\_P\_IP 0x800 只接收发往本机mac的ip类型的数据帧  
ETH\_P\_ARP 0x806 只接受发往本机mac的arp类型的数据帧  
ETH\_P\_RARP 0x8035 只接受发往本机mac的rarp类型的数据帧  
ETH\_P\_ALL 0x3 接收发往本机mac的所有类型ip arp rarp的数据帧, 接收从本机发出的所有类型的数据帧.(混杂模式打开的情况下,会接收到非发往本地mac的数据帧)