1.概述

OSI7层协议

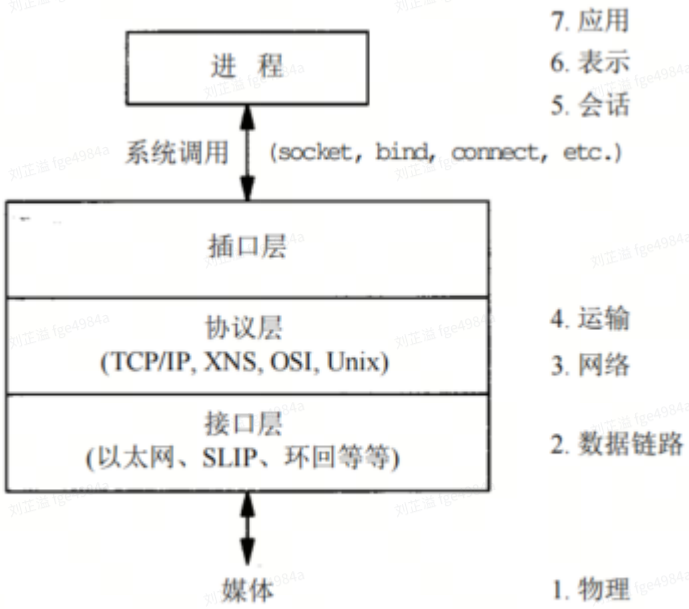


图1-3 Net/3联网代码的大概组织

描述符

filedesc{}

- fd_ofiles指向的数据结构是*file{}[]，指向file{}结构的指针数组；该数组的下标就是描述符本身

file{}

- f_type指示描述符的类型是DTYPE_SOCKET和DTYPE_VNODE。v-node允许内核支持不同类型的文件系统——磁盘文件系统、网络文件系统、CD-ROM文件系统、基于存储器的文件系统
- f_data指向一个socket{}或一个vnode{}结构
- f_ops指向一个有5个函数指针的向量；执行raead\write\select等系统调用时直接调用fileops{}的对应项即可

socket{}

- so_type指示socket的类型(SOCK_DGRAM)
- so_pcb指向为该socket分配的Internet协议控制块(PCB)，inpcb{}结构；inpcb{}结构的成员inp_socket指向结构socket

udb：全局结构，所有UDP PCB组成的链表表头

socket操作

- sendto：内核使用fd_ofiles索引到file结构指针向量，直到找到描述符对应的file结构，结构file指向socket结构，结构socket带有指向结构inpcb的指针
- recvfrom：当一个UDP数据报到达网络接口，内核搜索所有UDP的inpcb，寻找到合适的PCB，内核可以通过inp_socket指针寻找到对应的socket结构

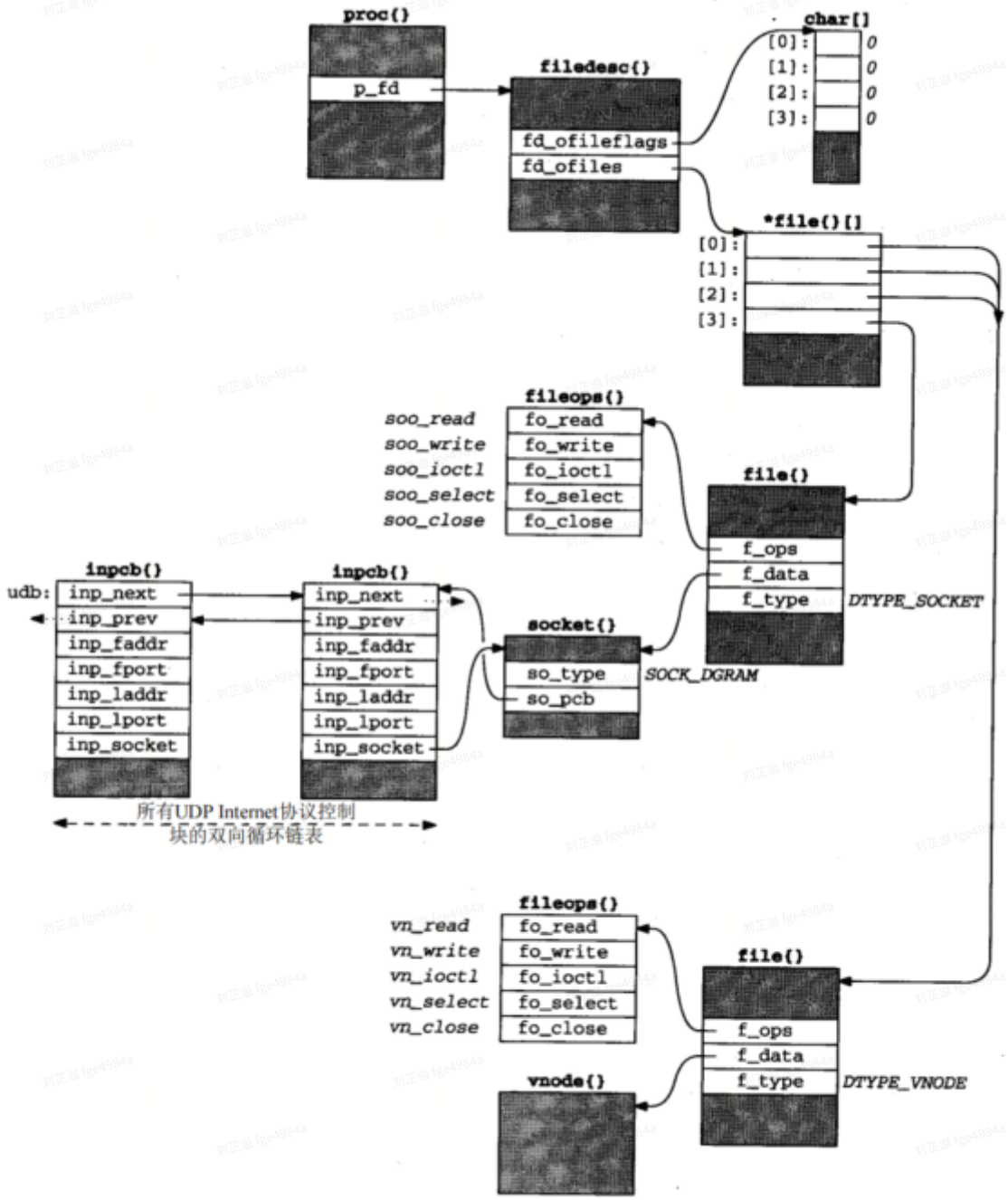


图1-5　在程序示例中调用socket后的内核数据结构

Mbuf

输出处理

- sendto调用过程

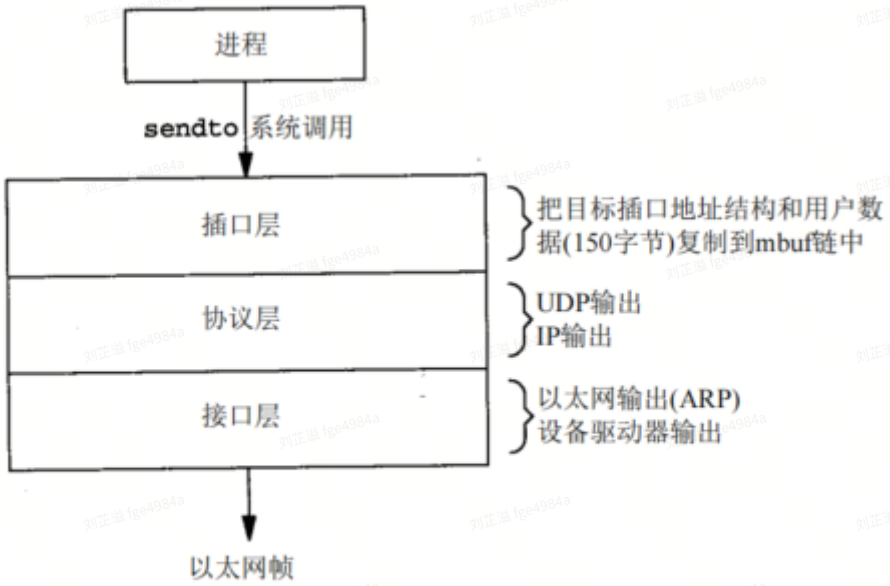


图1-9　三层处理一个简单UDP输出的执行过程

包含插口层(socket)的mbuf

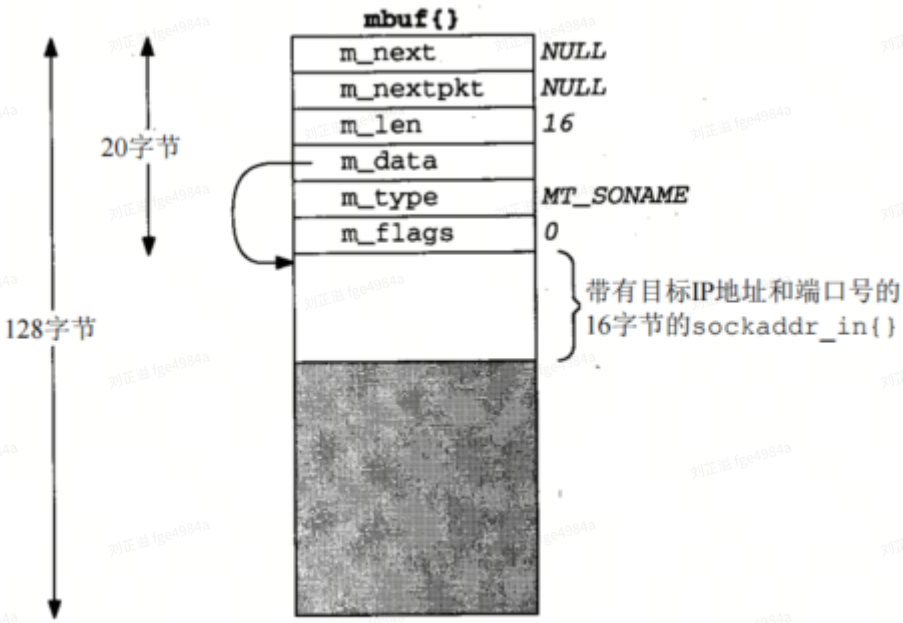
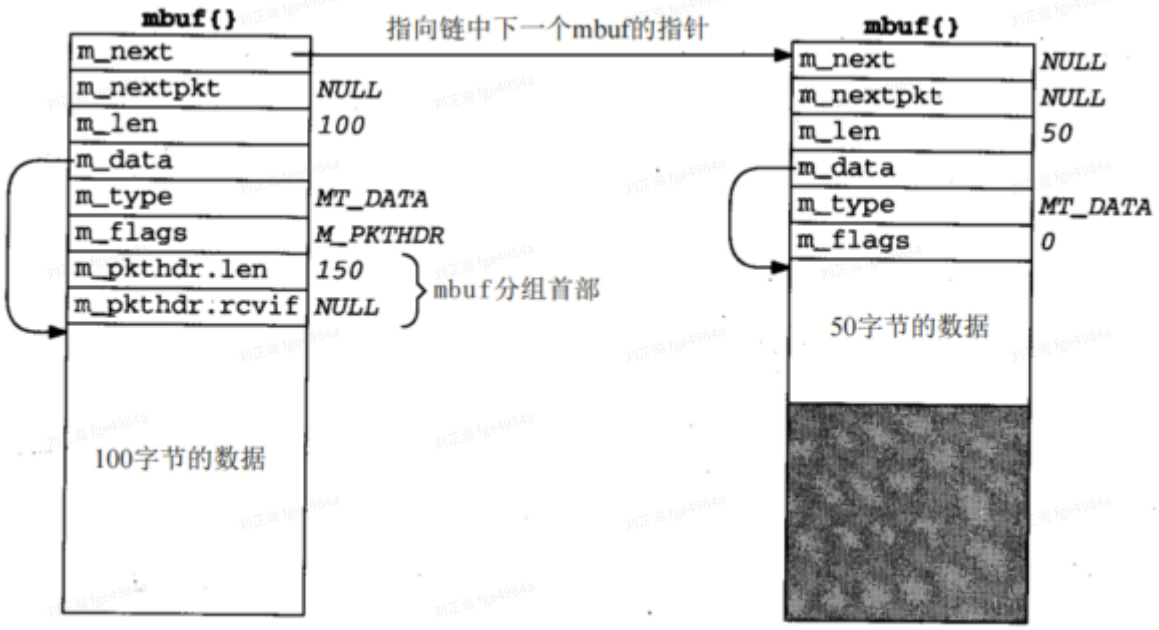


图1-6　mbuf中针对sendto的目的地址

包含数据的mbuf

- 在链表的第一个mbuf中维护一个带有总长度的分组首部，可以避免查看所有mbuf的m_len来求和



添加UDP和IP首部

- 另外分配一个mbuf并放置在链表头部；并将分组首部复制过来
- UDP首部和IP首部放置到最后面
- 这里，UDP首部计算校验和会第二次遍历150字节数据；第一次是从用户缓存中复制到内核的mbuf

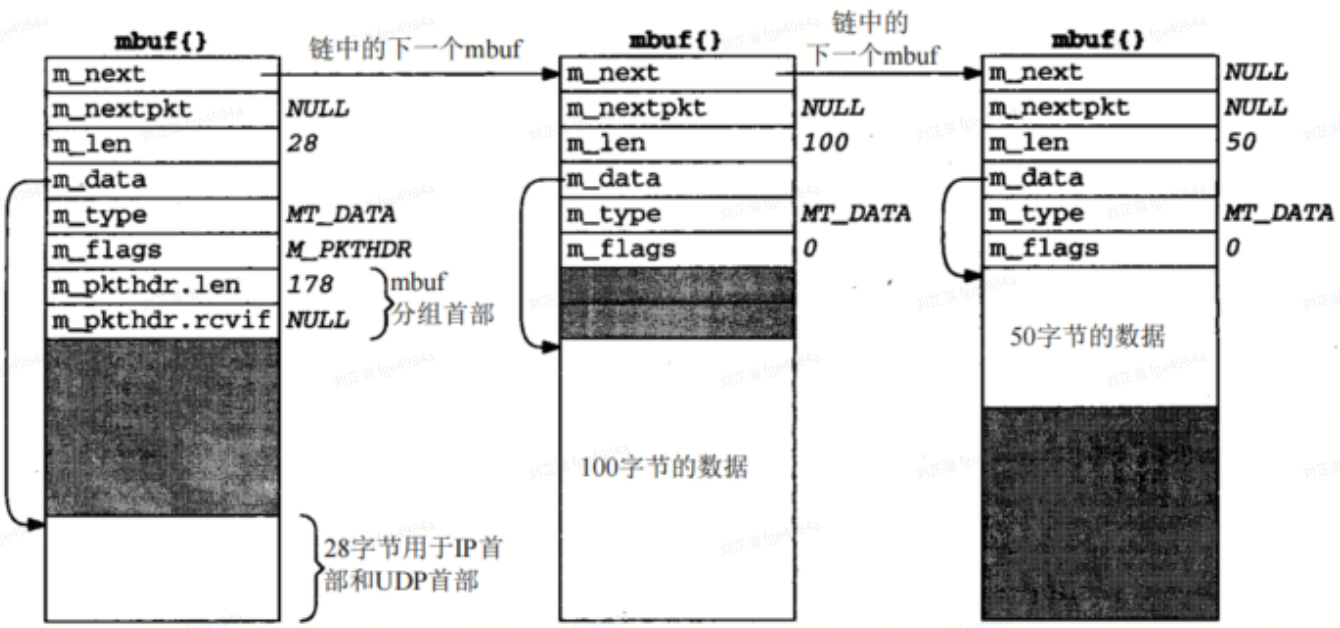


图1-8　在图1-7中的mbuf链表中添加另一个带有IP和UDP首部的mbuf

IP和以太网输出

- UDP输出例程调用IP输出例程，将mbuf链表指针传递给IP输出例程
- IP输出例程填写IP首部的字段，将该链表指针作为参数调用以太网输出函数
- 以太网输出例程将14字节的以太网首部添加到链表的第一个mbuf中，紧接在IP首部前面；mbuf链表被加到此接口的输出队列队尾。如果接口不忙，接口的开始输出例程立即被调用；若接口忙，在它处理完输出队列其余缓存后，会处理新mbuf

输入处理

以太网输入

- 以太网设备驱动程序处理该中断；数据从设备读到一个mbuf链表
- 这个mbuf是一个分组首部(m_flags被设置为M_PKTHDR)，成员rcvif用于接收分组而不是输出分组

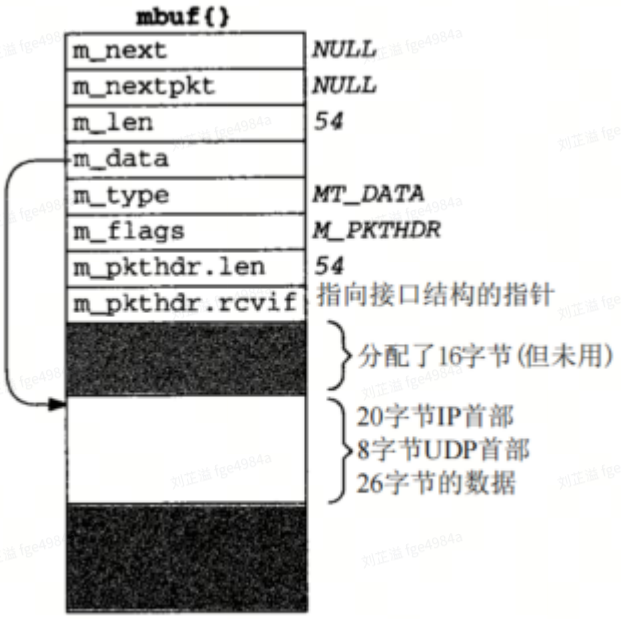


图1-10　用一个mbuf存储输入的以太网数据

IP输入

- IP输入是异步的，通过一个软中断执行；接口在系统上收到一个IP数据包，就设置该软中断；IP输入例程执行它时，循环处理在它的输入队列中的每一个IP数据报，并在整个队列被处理完后返回

UDP输入

- UDP输入例程从udb开始寻找一个本地端口号与UDP数据报目标端口号匹配的inpcb；inp_socket指向对应的socket，允许接收的数据在socket处排队
- 含有sockaddr_in的mbuf是插口层创建的

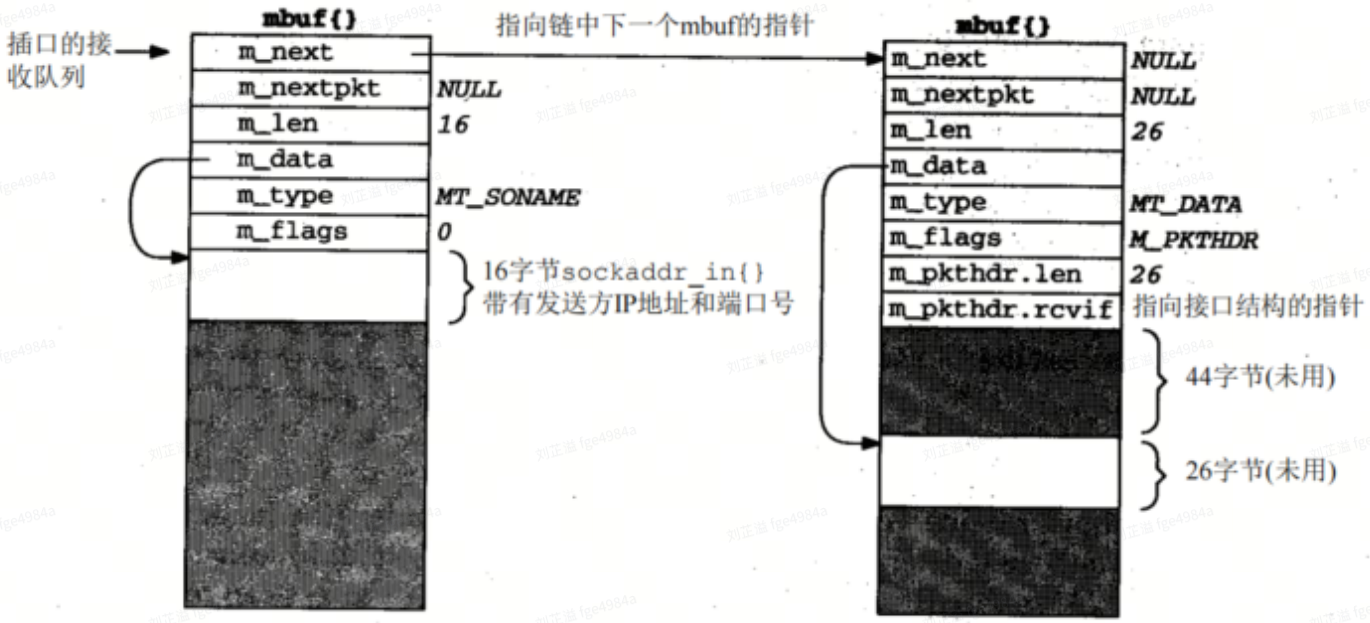


图1-11　发送方地址和数据

进程输入

- UDP层追加到插口接收队列的26字节数据被内核从mbuf复制到我们程序的缓存中

注意，我们的程序把recvform的第5，第6个参数设置为空指针，告诉系统在接收过程中不关心发送方的IP地址和UDP端口号。这使得系统调用recvfrom时，略过链表中的第一个mbuf(图1-11)，仅返回第二个mbuf中的26字节的数据。然后内核的recvfrom代码释放图1-11中的两个mbuf，并把它们放回到自由mbuf池中。

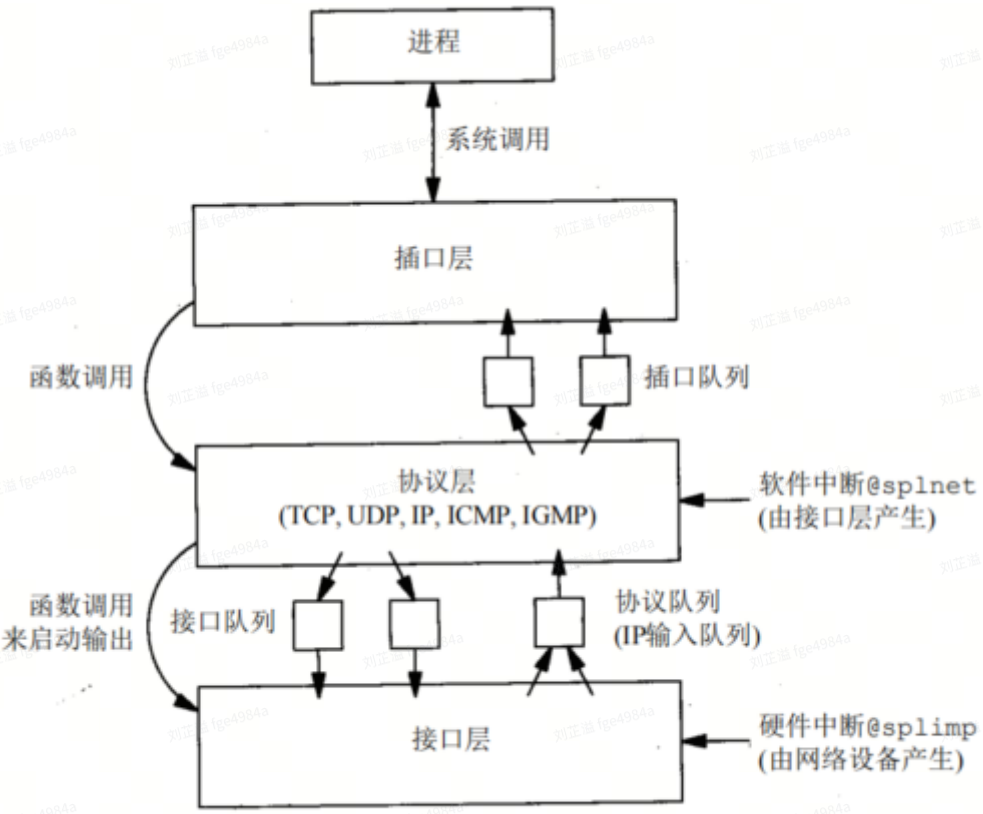


图1-12　网络输入输出的层间通信

中断级别与并发

函　数	说　明
spl0	正常操作方式，不阻塞中断　(最低优先级)
Splsoftclock	低优先级时钟处理
splnet	网络协议处理
spltty	终端输入输出
splbio	磁盘与磁带输入输出
splimp	网络设备输入输出
splclock	高优先级时钟处理
splhigh	阻塞所有中断　(最高优先级)
splx(s)	(见正文)

图1-13　阻塞所选中断的内核函数

```
1   struct mbuf *m;
2   int s;
3   s = splimp();
4   IF_DEQUEUE(&ipintrq, m);
5   splx(s);
6   if(m == 0)
7       return;
```

- 调用splimp将CPU优先级升高到网络设备驱动程序级，防止任何网络设备驱动程序中断发生

```
1   struct mbuf *m;
2   int s;
3   s = splimp();
4   // ...
5   IF_ENQUEUE(&ifp->if_snd, m);
6   // ...
7   splx(s);
```

- 设备中断被禁止，防止在协议层向队列添加分组时，设备驱动程序从发送队列取走下一个分组