基于PCAP的网络分析程序的实现

PCAP

- - PCAP 是一组抓取网络流量的接口组合
- - 在类unix的系统上PCAP接口的实现是libpcap,在windows系统上是WinPcap
- - PCAP API都是用C写的.所以其他语言如java,net.或者脚本语言,如果要使用的话,都是对这些类库的调用

LIBPCAP & WINP

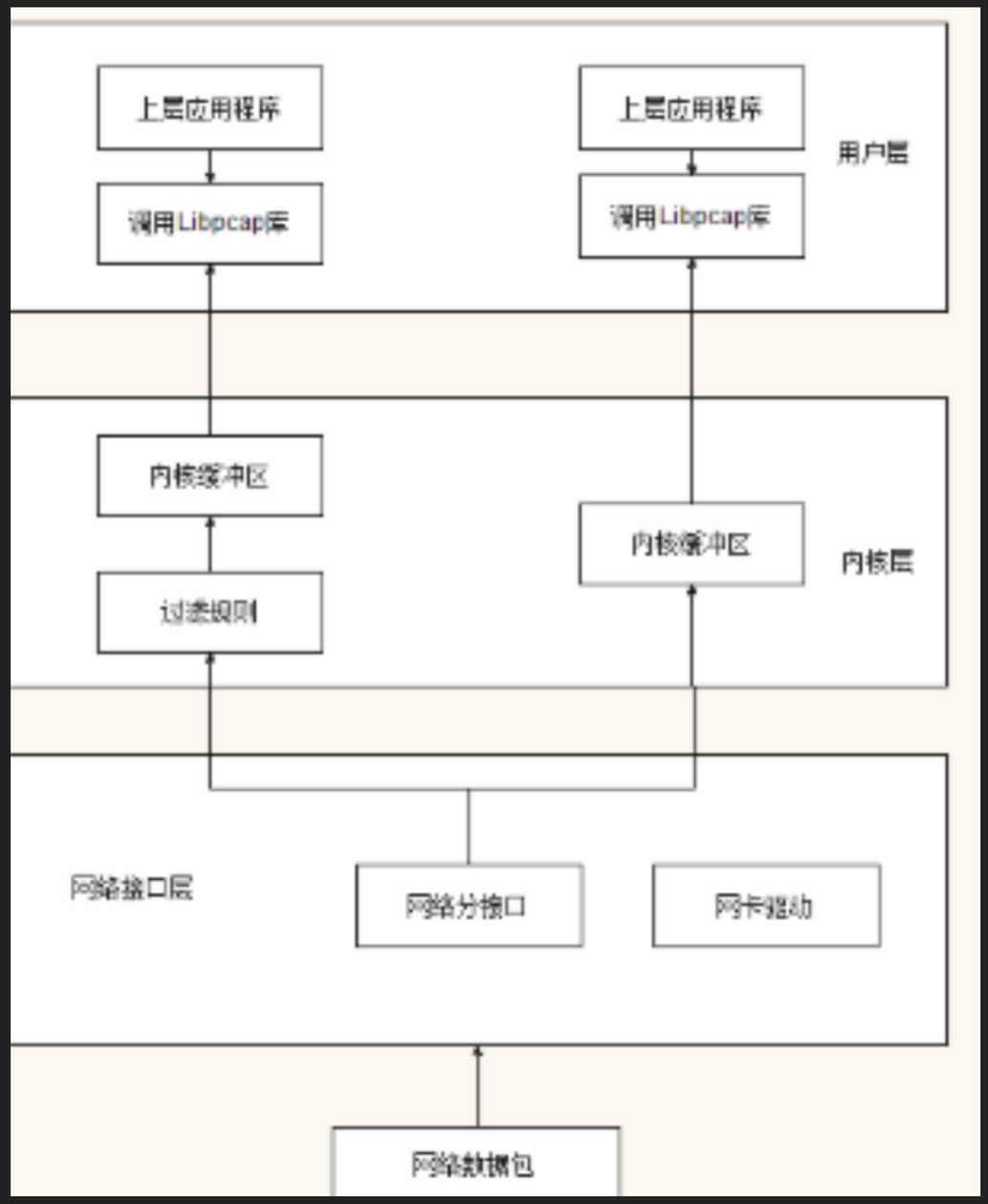
- libpcap和WinPcap提供许多开源和商业网络工具的数据包捕获和过滤引擎，包括协议分析器（数据包嗅探器），网络监视器，网络入侵检测系统，流量生成器和网络测试器。
- libpcap和WinPcap还支持将捕获的数据包保存到文件中，并读取包含已保存数据包的文件; 可以使用libpcap或WinPcap编写应用程序，以便能够捕获网络流量并对其进行分析，或者使用相同的分析代码读取已保存的捕获并对其进行分析。 以libpcap和WinPcap使用的格式保存的捕获文件可以由了解该格式的应用程序读取，例如tcpdump，Wireshark，CA NetMaster或Microsoft Network Monitor 3.x.
- libpcap和WinPcap创建和读取的文件格式的MIME类型是application / vnd.tcpdump.pcap。 典型的文件扩展名是.pcap，但.cap和.dmp也是常用的。

LIBPCAP

- libpcap最初由美国劳伦斯伯克力国家实验室网络研究小组的tcpdump开发人员开发。它是tcpdump的低级数据包捕获,捕获文件读取和捕获文件分析的类库,tcpdump基于libpcap实现
  -
- 官网
  - http://www.tcpdump.org/

LIBPCAP原理

- `


上层应用程序
调用Libpcap库
上层应用程序
调用Libpcap库
用户层
内核缓冲区
过滤规则
内核缓冲区
内核层
网络接口层
网络分接口
网卡驱动
网络数据包

LIBPCAP

- 捕获各种数据包，例如：网络流量统计
- 过滤网络数据包，例如：过滤掉本地上的一些数据，类似防火墙
- 分析网络数据包，例如：分析网络协议，数据的采集
- 存储网络数据包，例如：保存捕获的数据以为将来进行分析

LIBPCAP的转包框架

- pcap_lookupdev():函数用来查找网络设备，返回可被pcap_open_live()函数调用的网络设备名指针。
- pcap_lookupnet():函数获得指定网络设备的网络号和掩码。
- pcap_open_live():函数用于打开设备，并且返回用于捕获网络数据包的数据包捕获描述字。对于此网络设备的操作都要基于此网络设备描述字。
- pcap_compile():函数用于将用户制定的过滤策略编译到过滤程序中
- pcap_setfilter():函数用于设置过滤器
- pcap_loop():与pcap_next()和pcap_next_ex()两个函数一样用来捕获数据包
- pcap_close():函数用于关闭网络设备，释放资源

LIBPCAP函数库开发应用程序的步骤

- 打开网络设备
- 设置过滤规则
- 捕获数据
- 关闭网络设备

LIBPCAP函数库开发应用程序的步骤

- C程序实现

PCAP文件格式

Global Header	Packet Header	Packet Data	Packet Header	Packet Data	Packet Header	Packet Data	...

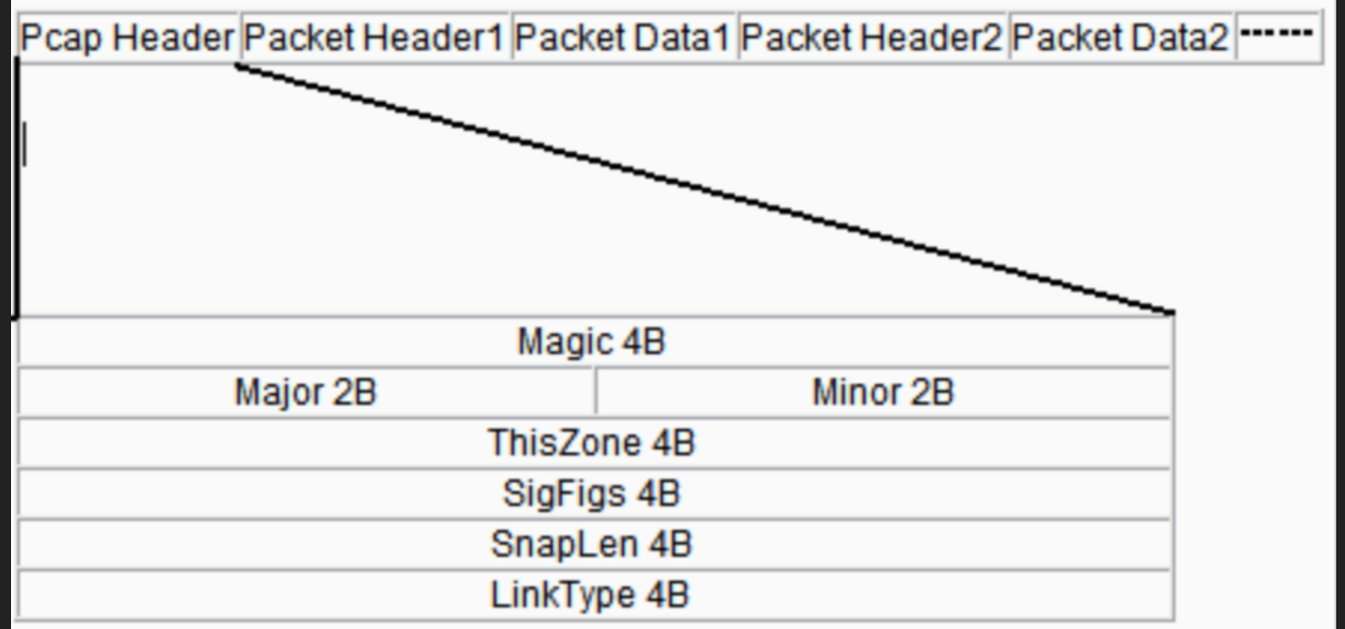
```
typedef struct pcap_hdr_s {
        guint32 magic_number;   /* magic number */
        guint16 version_major;  /* major version number */
        guint16 version_minor;  /* minor version number */
        gint32  thiszone;       /* GMT to local correction */
        guint32 sigfigs;        /* accuracy of timestamps */
        guint32 snaplen;        /* max length of captured packets, in octets */
        guint32 network;        /* data link type */
} pcap_hdr_t;
```

```
typedef struct pcaprec_hdr_s {
        guint32 ts_sec;         /* timestamp seconds */
        guint32 ts_usec;        /* timestamp microseconds */
        guint32 incl_len;       /* number of octets of packet saved in file */
        guint32 orig_len;       /* actual length of packet */
} pcaprec_hdr_t;
```

LIBPCAP文件格式

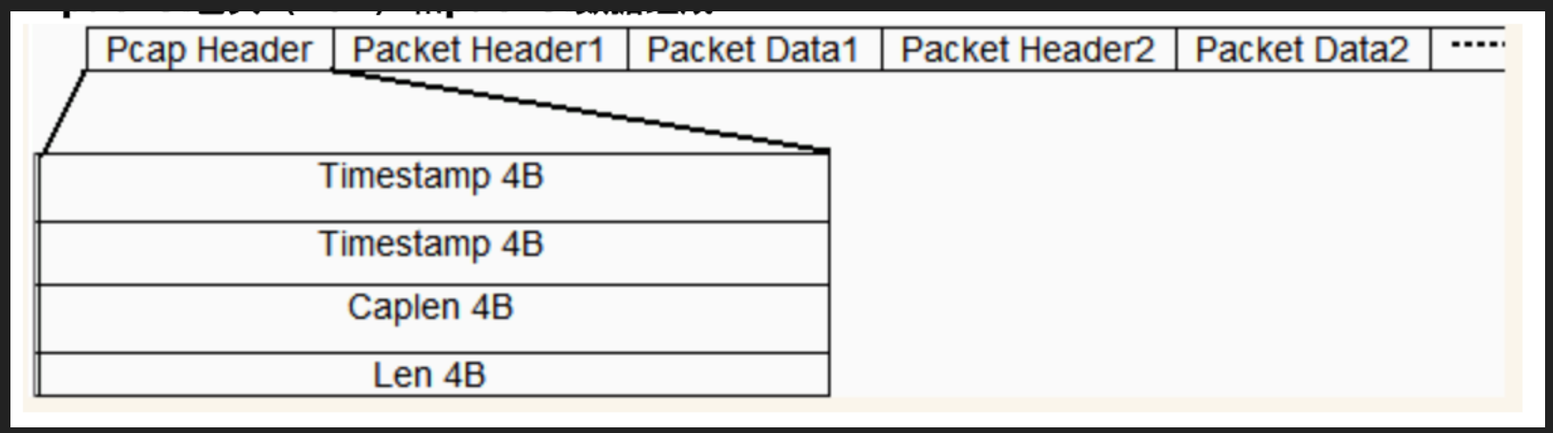
- pcap文件的格式为：
  文件头　24字节
  数据报头 + 数据报 数据包头为16字节，后面紧跟数据报
  数据报头 + 数据报 ……

1) Pcap文件整体结构和Pcap文件头结构



LIBPCAP文件格式

- Timestamp：时间戳高位，精确到seconds（值是自从January 1, 1970 00:00:00 GMT以来的秒数来记）
- Timestamp：时间戳低位，精确到microseconds （数据包被捕获时候的微秒（microseconds）数，是自ts-sec的偏移量）
- Caplen：当前数据区的长度，即抓取到的数据帧长度，由此可以得到下一个数据帧的位置。
- Len：离线数据长度：网络中实际数据帧的长度，一般不大于caplen，多数情况下和Caplen数值相等。



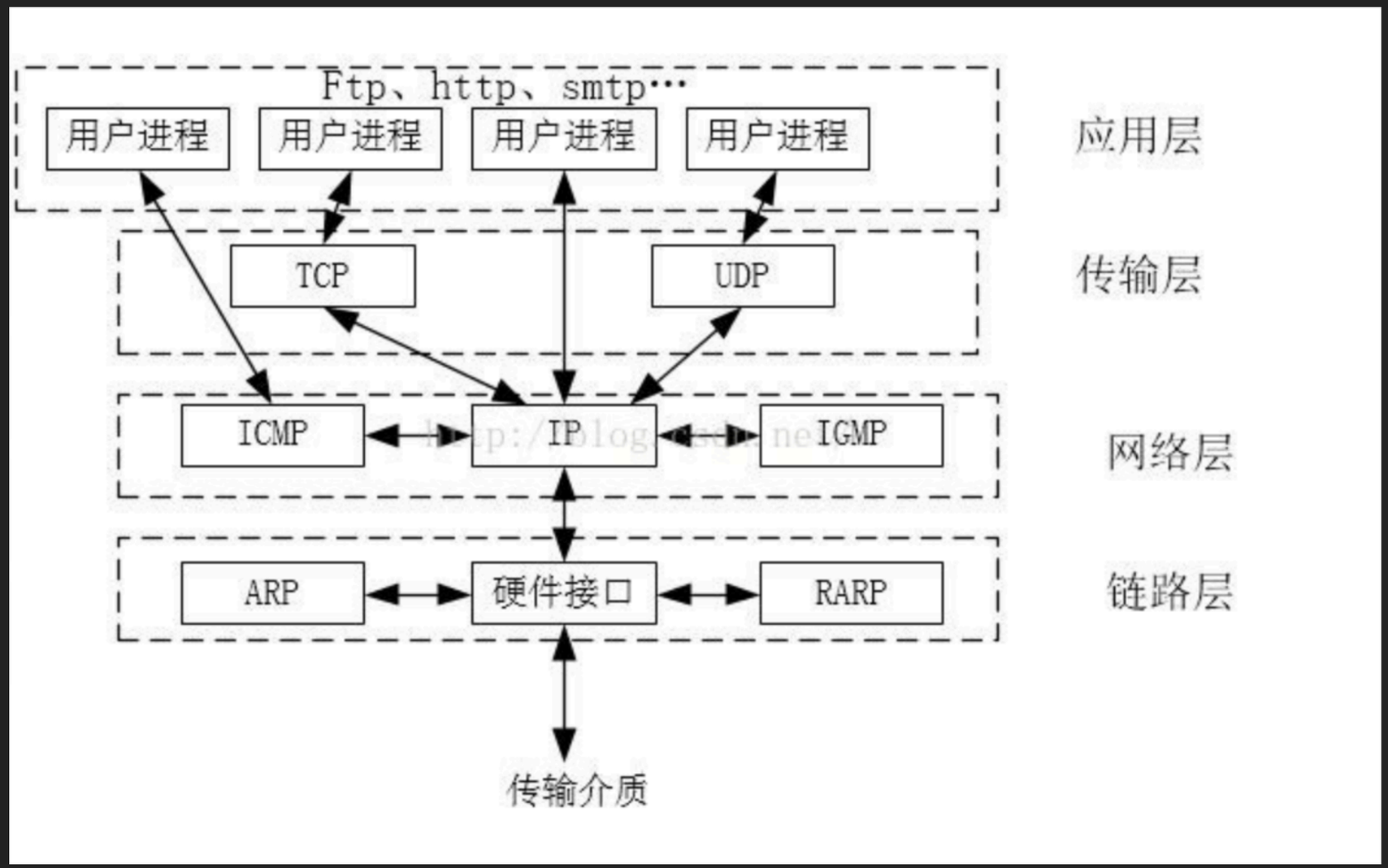
LIBPCAP文件格式

- 。

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
前 24 个字节包为数据包包头，包含了一些文件信息															
但对数据报的分析没什么用								数据报报头（16 字节）							
								以太网帧头（14 字节）							
以太网帧头								IP 头（一般 20 字节）							
IP 头								TCP 头							
TCP 头（一般 20 字节）														数据域	
数据域															
								数据报报头（16 字节）							
								下一数据报的以太网帧头							
以太网帧头								下一数据报的 IP 头（一般 20 字节）							
IP 头								下一数据报的 TCP 头							
TCP 头														数据域	
数据域								数据报报头							

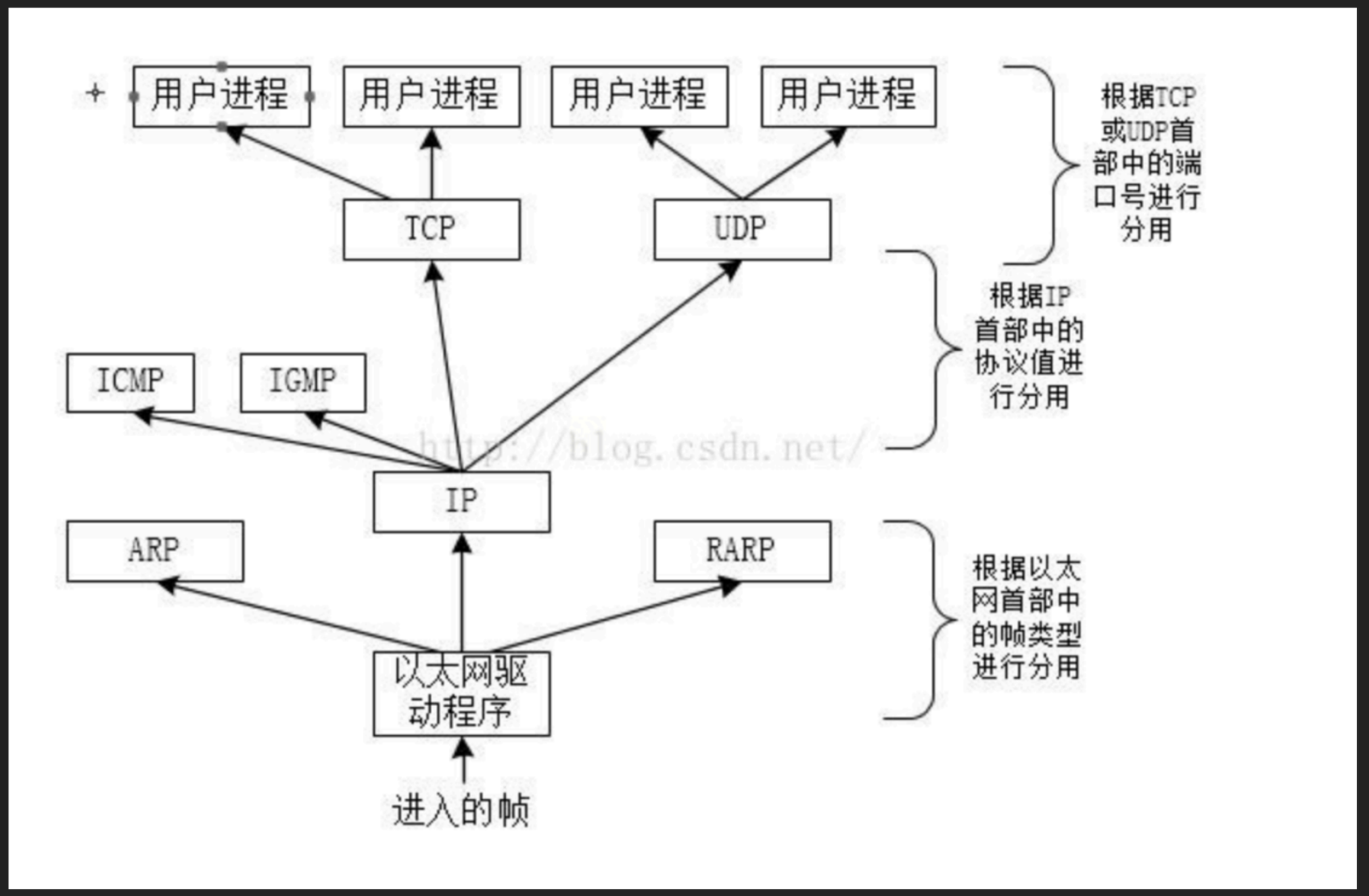
.TCP/IP协议各层关系

- 。



.TCP/IP协议各层关系

- 。



.TCP/IP协议各层关系

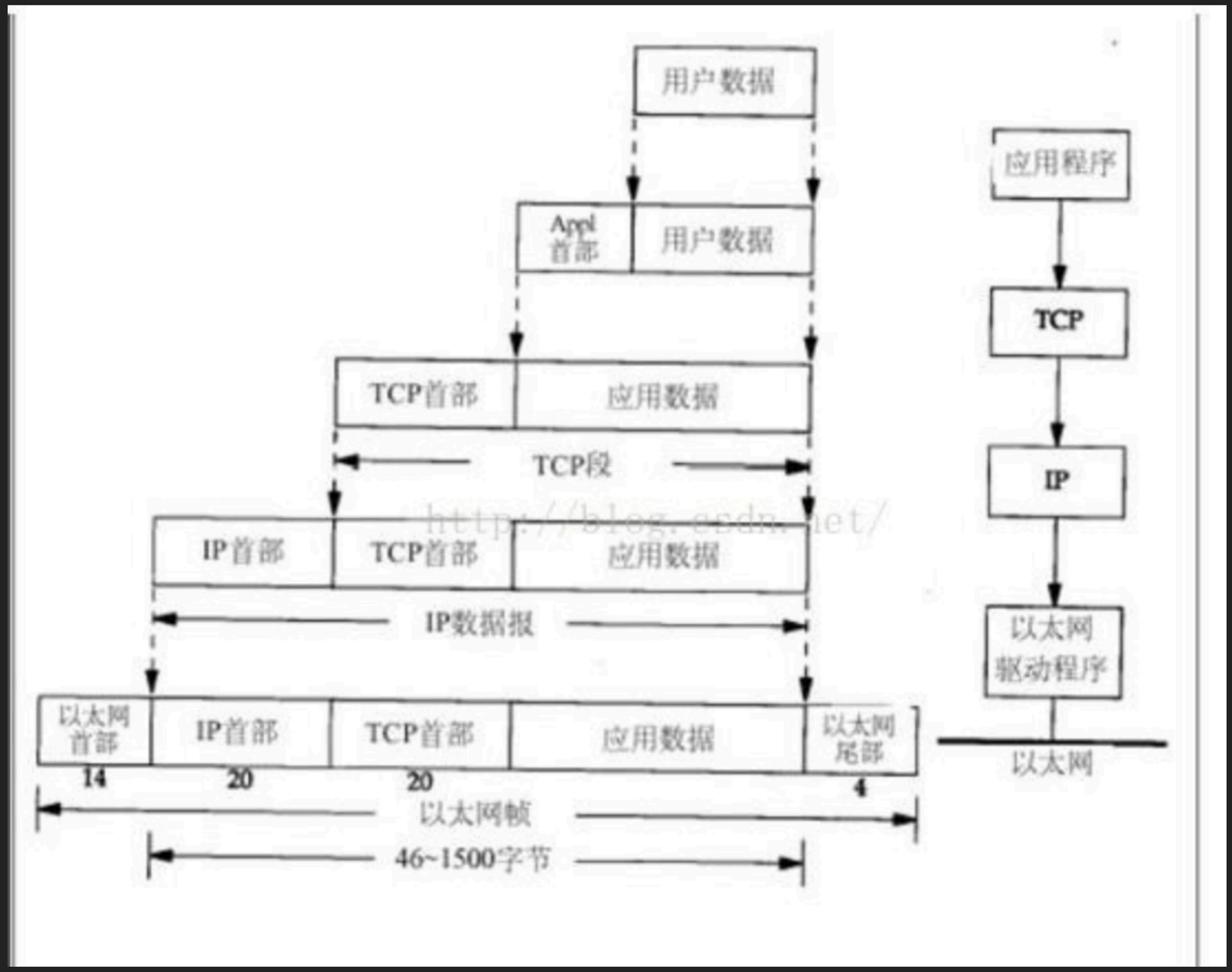
目的地址（6字节）	源地址（6字节）	类型（2字节）	数据（46-1500字节）	CRC（4字节）

类型0800（6字节）	数据（46-1500字节）	CRC（4字节）

目的地址（6字节）	ARP请求应答（28字节）	PAD（18字节）
目的地址（6字节）	ARP请求应答（28字节）	PAD（18字节）

.TCP/IP协议各层关系

- 。



.BPF(Berkeley Packet Filter (BPF)

- 当一个数据包到达网络接口时，数据链路层的驱动会把它向系统的协议栈传送。但如果 BPF 监听接口，驱动首先调用 BPF。BPF 首先进行过滤操作，然后把数据包存放在过滤器相关的缓冲区中，最后设备驱动再次获得控制。注意到BPF是先对数据包过滤再缓冲.
- 语法
- tcp src port *port*
- vlan && vlan 300 && ip
- ip host ace and not helios