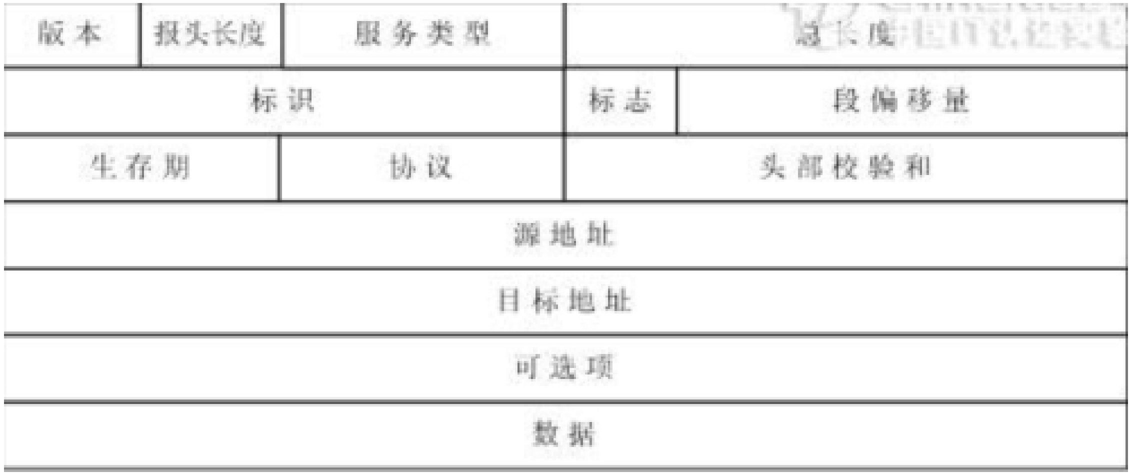
sniffer

sniffer

- 实验目的：
- 根据捕获的包的类型，来解析包的格式
  - ICMP
  - TCP
  - UDP

sniffer

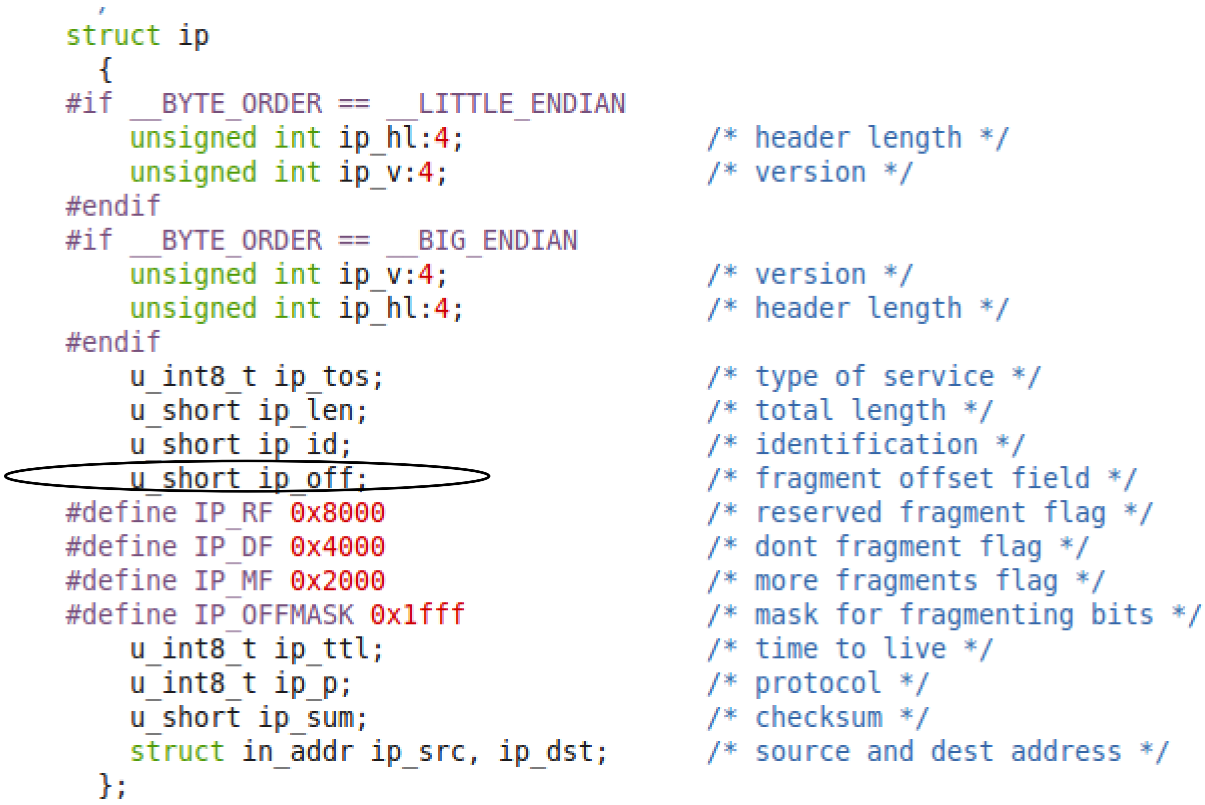
- 这三种包就封装在IP报文内
  - 解析IP报文



<table>
<tr><td>版本</td><td>报头长度</td><td>服务类型</td><td colspan="2">总长度</td></tr>
<tr><td colspan="3">标识</td><td>标志</td><td>段偏移量</td></tr>
<tr><td colspan="2">生存期</td><td>协议</td><td colspan="2">头部校验和</td></tr>
<tr><td colspan="5">源地址</td></tr>
<tr><td colspan="5">目标地址</td></tr>
<tr><td colspan="5">可选项</td></tr>
<tr><td colspan="5">数据</td></tr>
</table>

sniffer

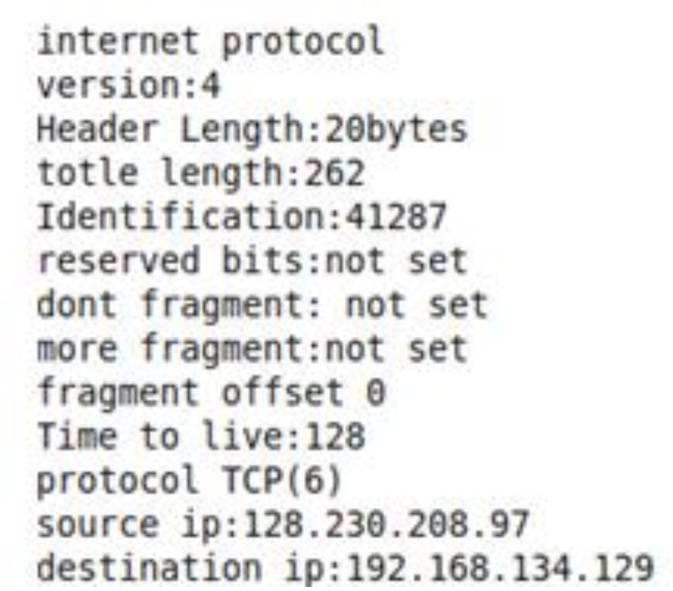
- IP报文的数据结构



```
struct ip
  {
#if __BYTE_ORDER == __LITTLE_ENDIAN
    unsigned int ip_hl:4;               /* header length */
    unsigned int ip_v:4;                /* version */
#endif
#if __BYTE_ORDER == __BIG_ENDIAN
    unsigned int ip_v:4;                /* version */
    unsigned int ip_hl:4;               /* header length */
#endif
    u_int8_t ip_tos;                    /* type of service */
    u_short ip_len;                     /* total length */
    u_short ip_id;                      /* identification */
    u_short ip_off;                     /* fragment offset field */
#define IP_RF 0x8000                    /* reserved fragment flag */
#define IP_DF 0x4000                    /* dont fragment flag */
#define IP_MF 0x2000                    /* more fragments flag */
#define IP_OFFMASK 0x1fff               /* mask for fragmenting bits */
    u_int8_t ip_ttl;                    /* time to live */
    u_int8_t ip_p;                      /* protocol */
    u_short ip_sum;                     /* checksum */
    struct in_addr ip_src, ip_dst;      /* source and dest address */
  };
```

sniffer

- 解析出来的IP报文



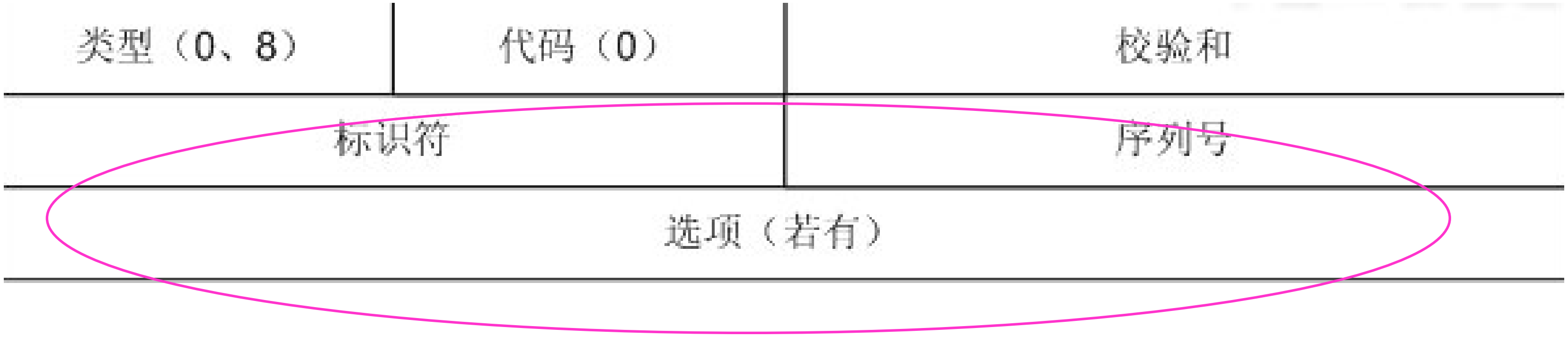
```
internet protocol
version:4
Header Length:20bytes
totle length:262
Identification:41287
reserved bits:not set
dont fragment: not set
more fragment:not set
fragment offset 0
Time to live:128
protocol TCP(6)
source ip:128.230.208.97
destination ip:192.168.134.129
```

sniffer

- 根据ip_p中的协议类型
- IPPROTO_ICMP:ICMP
- IPPROTO_UDP:UDP
- IPPROTO_TCP:TCP

sniffer

- ICMP报文格式



类型（0、8）	代码（0）	校验和
标识符		序列号
选项（若有）		

sniffer

```
struct icmphdr
{
  u_int8_t type;                /* message type */
  u_int8_t code;                /* type sub-code */
  u_int16_t checksum;
  union
  {
    struct
    {
      u_int16_t id;
      u_int16_t sequence;
    } echo;                     /* echo datagram */
    u_int32_t   gateway;        /* gateway address */
    struct
    {
      u_int16_t __unused;
      u_int16_t mtu;
    } frag;                     /* path mtu discovery */
  } un;
};
```

- /usr/include/netinet/ip_icmp.h

sniffer

- 普通的ICMP的报文(ping包)
- struct
-     {
-         u_int16_t id;
-         u_int16_t sequence;
-     } echo;   /* echo datagram */

- icmp->icmp_id
- icmp->icmp_seq

sniffer

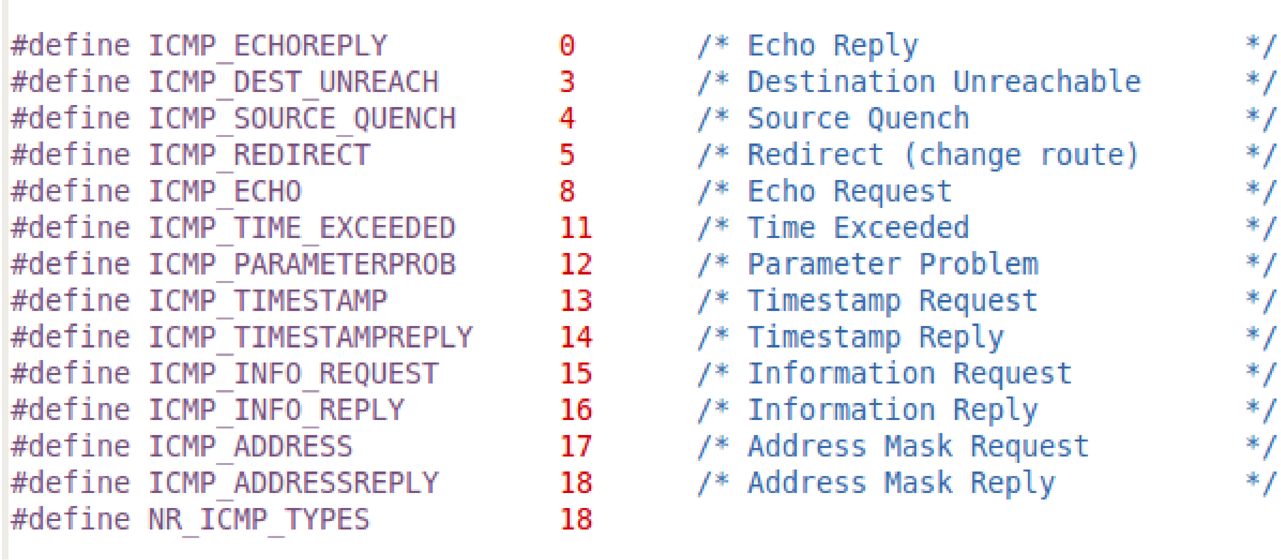
- 重定向包
- u_int32_t   gateway;
- icmp->icmp_gwaddr

sniffer

- 路由发现
- struct
- {
- u_int16_t __unused;
- u_int16_t mtu;
- } frag; /* path mtu discovery */
- type=3
- type=4

sniffer

- icmp_type

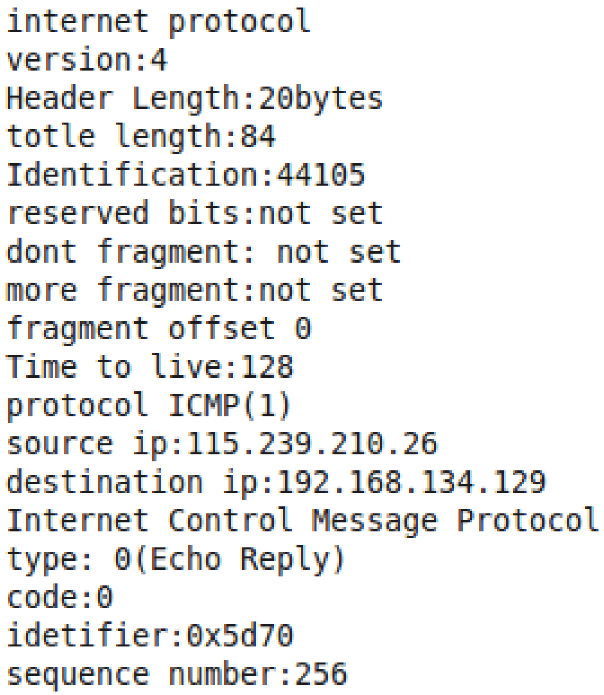


```
#define ICMP_ECHOREPLY          0       /* Echo Reply                   */
#define ICMP_DEST_UNREACH       3       /* Destination Unreachable      */
#define ICMP_SOURCE_QUENCH      4       /* Source Quench                */
#define ICMP_REDIRECT           5       /* Redirect (change route)      */
#define ICMP_ECHO               8       /* Echo Request                 */
#define ICMP_TIME_EXCEEDED      11      /* Time Exceeded                */
#define ICMP_PARAMETERPROB      12      /* Parameter Problem            */
#define ICMP_TIMESTAMP          13      /* Timestamp Request            */
#define ICMP_TIMESTAMPREPLY     14      /* Timestamp Reply              */
#define ICMP_INFO_REQUEST       15      /* Information Request          */
#define ICMP_INFO_REPLY         16      /* Information Reply            */
#define ICMP_ADDRESS            17      /* Address Mask Request         */
#define ICMP_ADDRESSREPLY       18      /* Address Mask Reply           */
#define NR_ICMP_TYPES           18
```

sniffer

- icmp报文分类：
- 1.响应请求
  - ping
  - 就是响应请求（Type=8）
  - 应答（Type=0）
- 2.目标不可到达、源抑制和超时报文
  - 目标不可到达报文（Type=3）
- 3.时间戳
  - 时间戳请求报文（Type=13）和时间戳应答报文（Type=14）用于测试两台主机之间数据报来回一次的传输时间。

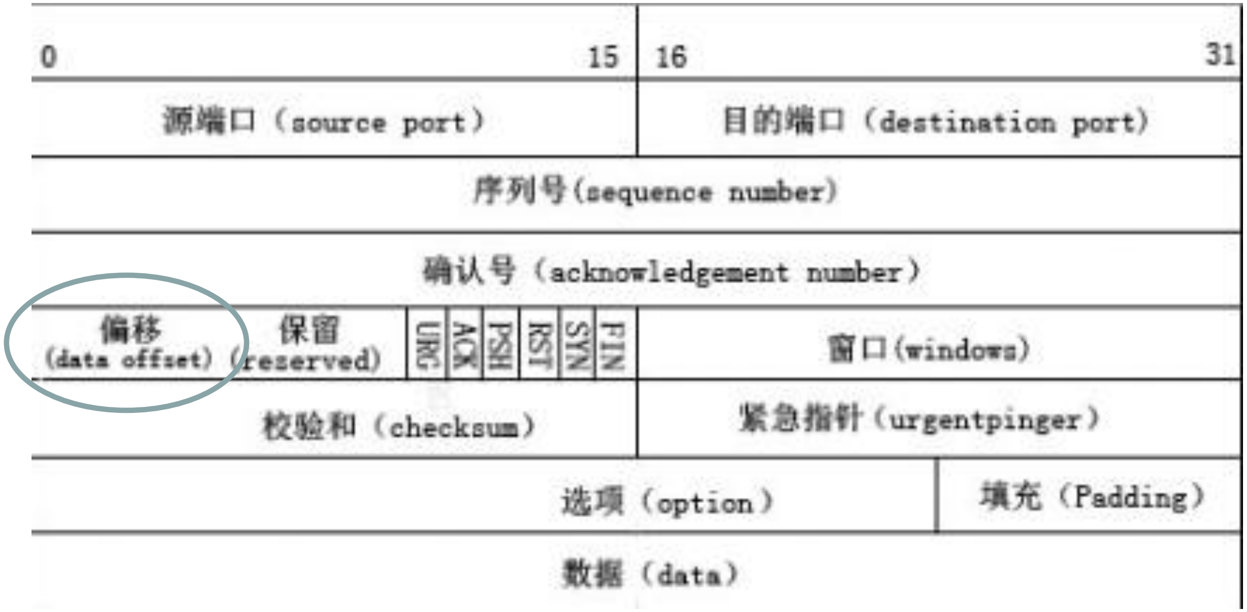
sniffer



```
internet protocol
version:4
Header Length:20bytes
totle length:84
Identification:44105
reserved bits:not set
dont fragment: not set
more fragment:not set
fragment offset 0
Time to live:128
protocol ICMP(1)
source ip:115.239.210.26
destination ip:192.168.134.129
Internet Control Message Protocol
type: 0(Echo Reply)
code:0
idetifier:0x5d70
sequence number:256
```

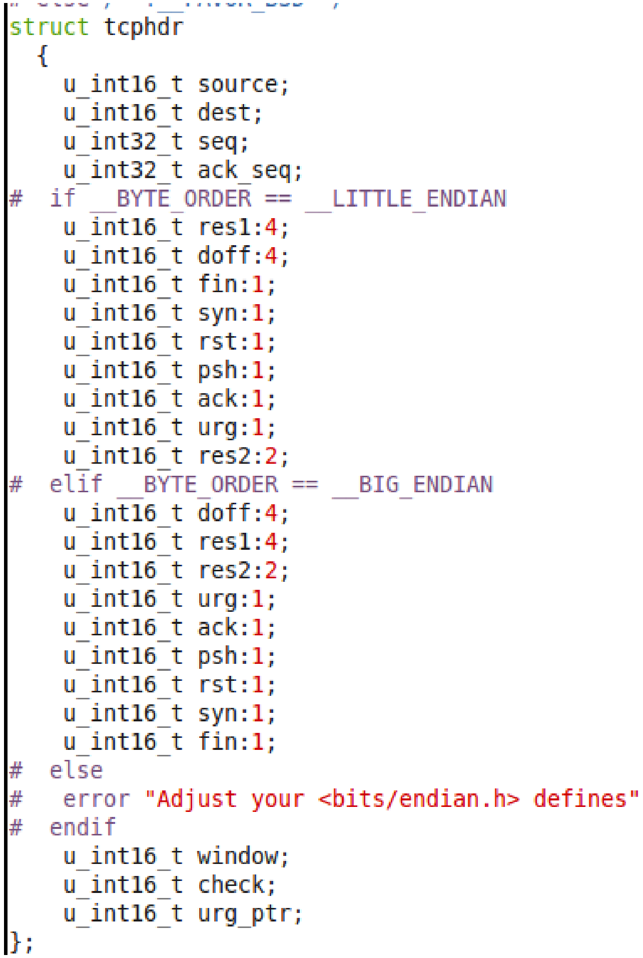
sniffer

- TCP



<table>
<tr><td colspan="8">0 15</td><td colspan="2">16 31</td></tr>
<tr><td colspan="8">源端口（source port）</td><td colspan="2">目的端口（destination port）</td></tr>
<tr><td colspan="10">序列号(sequence number)</td></tr>
<tr><td colspan="10">确认号（acknowledgement number）</td></tr>
<tr><td>偏移
(data offset)</td><td>保留
(reserved)</td><td>URG</td><td>ACK</td><td>PSH</td><td>RST</td><td>SYN</td><td>FIN</td><td colspan="2">窗口(windows)</td></tr>
<tr><td colspan="8">校验和（checksum）</td><td colspan="2">紧急指针（urgentpinger）</td></tr>
<tr><td colspan="9">选项（option）</td><td>填充（Padding）</td></tr>
<tr><td colspan="10">数据（data）</td></tr>
</table>

sniffer

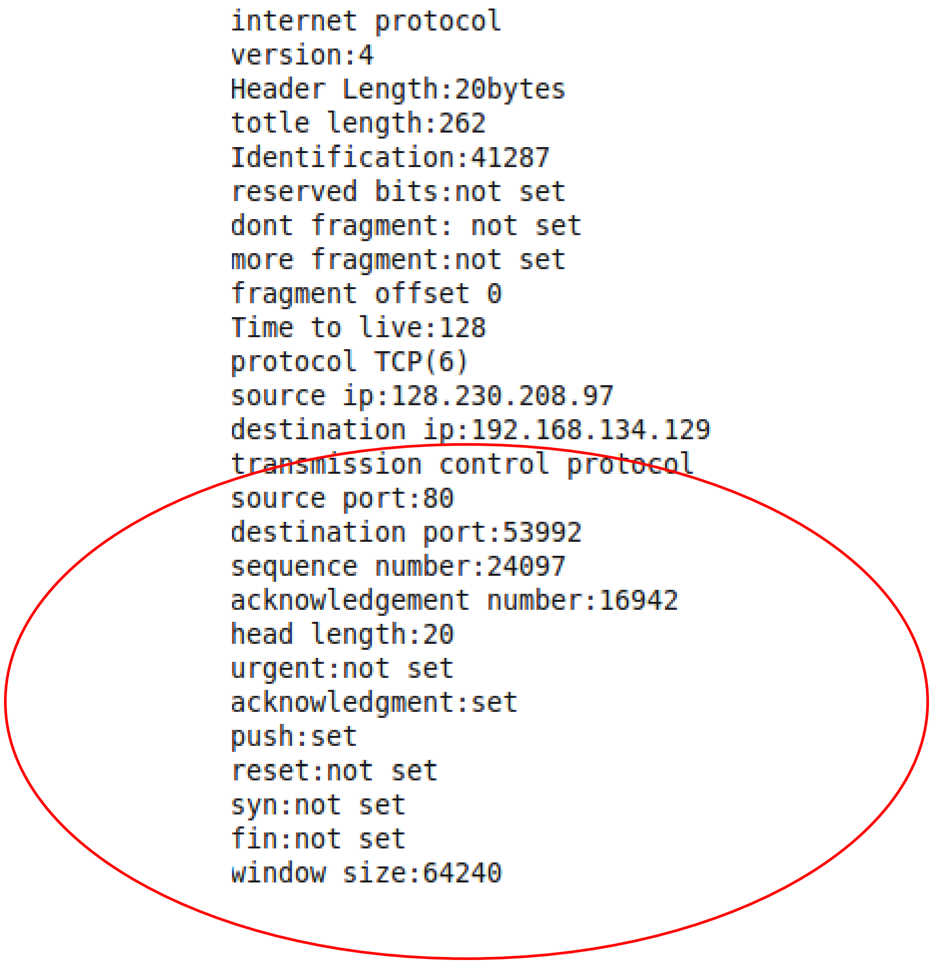


```
# else /* !__FAVOR_BSD */
struct tcphdr
  {
    u_int16_t source;
    u_int16_t dest;
    u_int32_t seq;
    u_int32_t ack_seq;
#  if __BYTE_ORDER == __LITTLE_ENDIAN
    u_int16_t res1:4;
    u_int16_t doff:4;
    u_int16_t fin:1;
    u_int16_t syn:1;
    u_int16_t rst:1;
    u_int16_t psh:1;
    u_int16_t ack:1;
    u_int16_t urg:1;
    u_int16_t res2:2;
#  elif __BYTE_ORDER == __BIG_ENDIAN
    u_int16_t doff:4;
    u_int16_t res1:4;
    u_int16_t res2:2;
    u_int16_t urg:1;
    u_int16_t ack:1;
    u_int16_t psh:1;
    u_int16_t rst:1;
    u_int16_t syn:1;
    u_int16_t fin:1;
#  else
#   error "Adjust your <bits/endian.h> defines"
#  endif
    u_int16_t window;
    u_int16_t check;
    u_int16_t urg_ptr;
};
```

sniffer

- struct tcphdr *tcp=(struct tcphdr*)...
- tcp->doff:TCP首部的真实长度
- data offset TCP头部大小

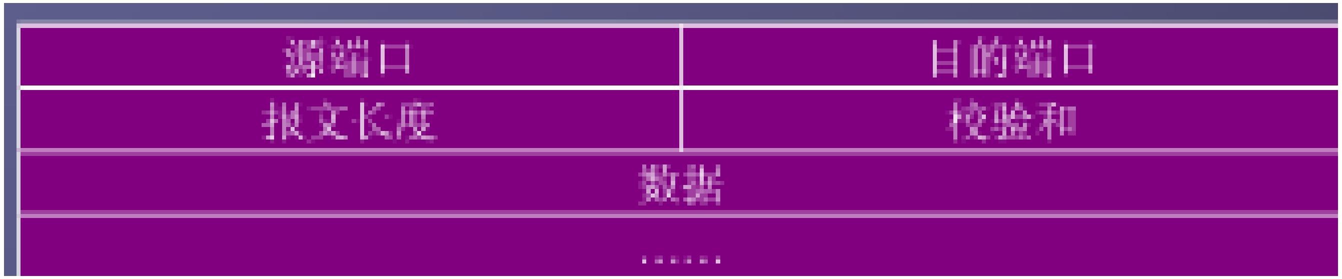
sniffer



```
internet protocol
version:4
Header Length:20bytes
totle length:262
Identification:41287
reserved bits:not set
dont fragment: not set
more fragment:not set
fragment offset 0
Time to live:128
protocol TCP(6)
source ip:128.230.208.97
destination ip:192.168.134.129
transmission control protocol
source port:80
destination port:53992
sequence number:24097
acknowledgement number:16942
head length:20
urgent:not set
acknowledgment:set
push:set
reset:not set
syn:not set
fin:not set
window size:64240
```

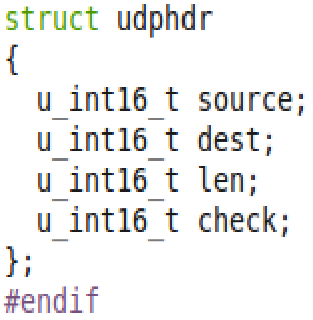
sniffer

- UDP报文格式



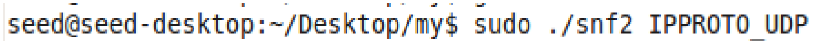
源端口	目的端口
报文长度	校验和
数据	
......	

sniffer

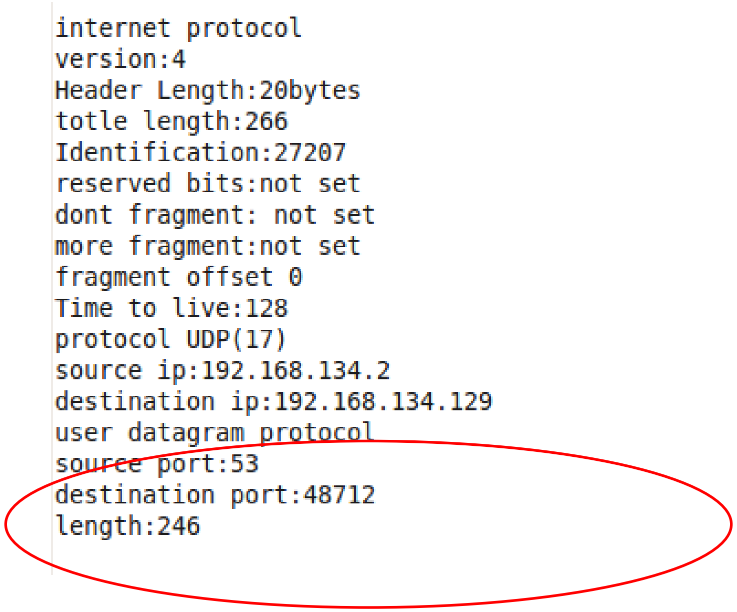


```
struct udphdr
{
  u_int16_t source;
  u_int16_t dest;
  u_int16_t len;
  u_int16_t check;
};
#endif
```

sniffer



```
seed@seed-desktop:~/Desktop/my$ sudo ./snf2 IPPROTO_UDP
```



```
internet protocol
version:4
Header Length:20bytes
totle length:266
Identification:27207
reserved bits:not set
dont fragment: not set
more fragment:not set
fragment offset 0
Time to live:128
protocol UDP(17)
source ip:192.168.134.2
destination ip:192.168.134.129
user datagram protocol
source port:53
destination port:48712
length:246
```