网络协议分析与实现

鲍青
杭州电子科技大学网络空间安全学院
E-mail： qbao@hdu.edu.cn

目录

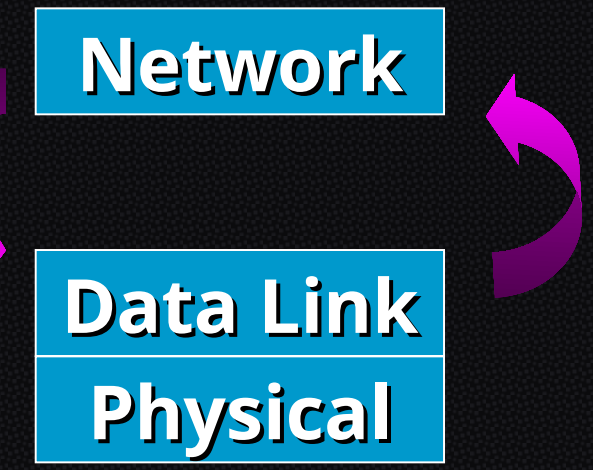
- ARP 协议
- ARP 报文格式
- ARP 整体结构
- ARP 输入处理
- ARP 请求
- ARP 缓冲
- ARP 攻击

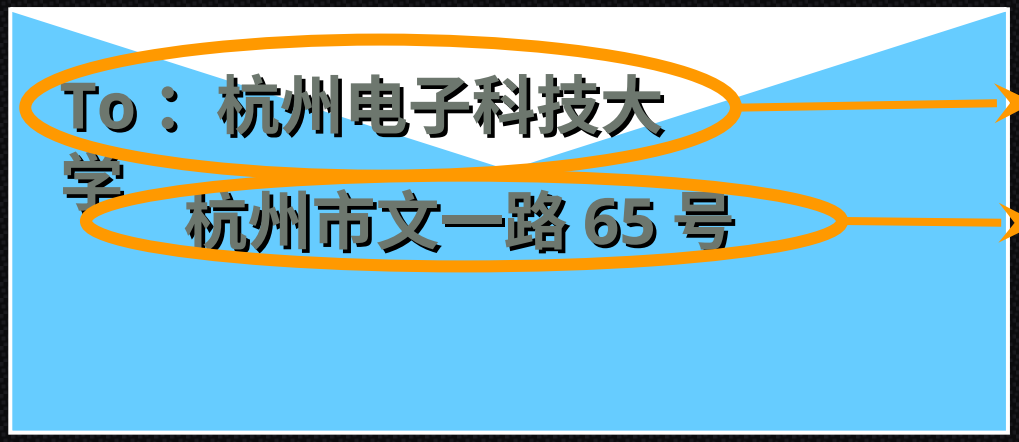
两级地址

- IP 分组交付到主机或路由器需要**两级地址**
  - 互连网级：**逻辑地址**标识主机 / 路由器
    - 全网统一编址，具有全局唯一性
    - 所有与互联网打交道的**软件**都要使用**逻辑地址**
    - 在 Internet 中，逻辑地址就是 **IP 地址**（32bit）
  - 物理网级：**物理地址**标识主机 / 路由器
    - 本地范围内具有唯一性，但在整个互联网内不一定具有全局唯一性
    - 分组需要通过物理网络才能到达路由器或主机
    - 以太网中，物理地址就是 **MAC 地址**（48bit）

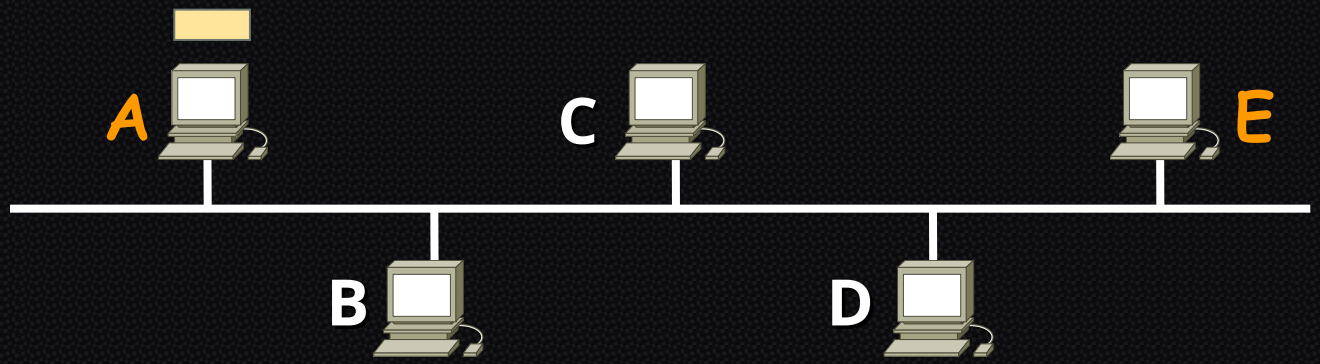
Mapping

- Logical address (IP)
- Physical address





Issues



- A 送往 E 的分组需封装在 Ethernet 帧中传送
  - IP 地址：全局性
  - 物理地址：本地性

A 仅知道：IP_A，MAC_A，IP_E

Ethernet Frame

目的 MAC	源 MAC	类型	IP Packet	FCS
?	MAC_A			

Address Mapping （地址映射）

- Logical address → Physical address

Static Table	
Logical address	Physical address
……	……
……	……

Static mapping
映射表固定设置

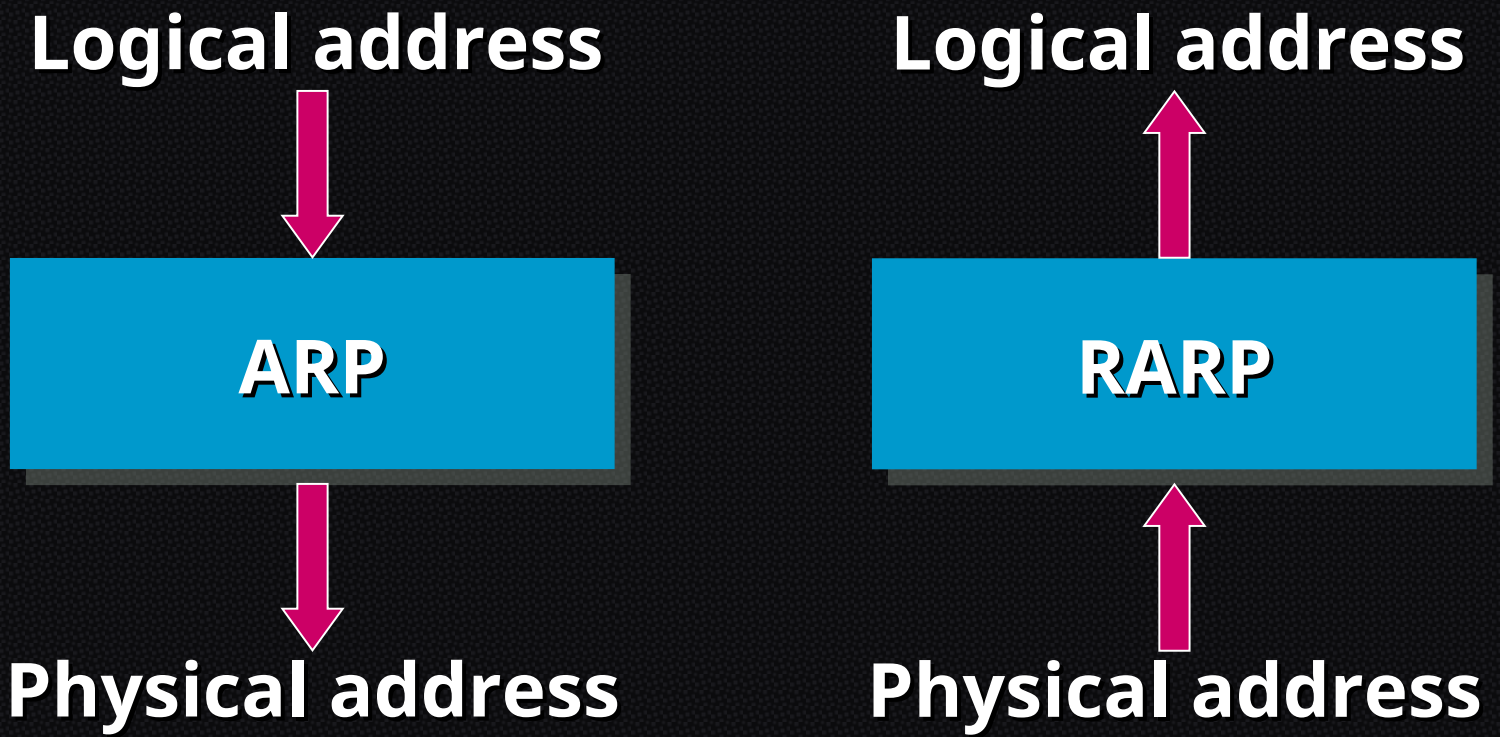
Consider:
- NIC changed
- Mobile computer

Dynamic mapping

Look for the target on demanding, using dynamic Address Resolution Protocol

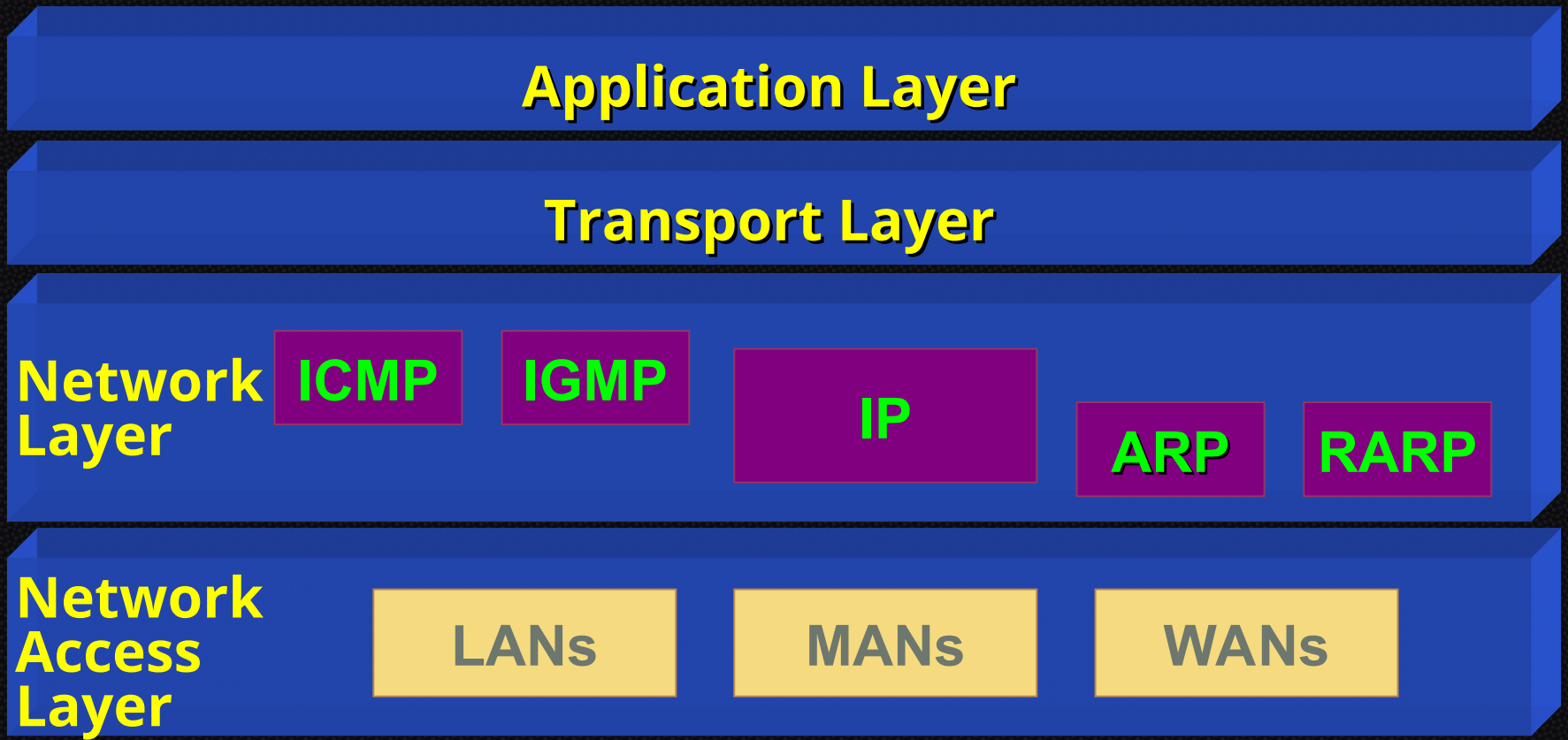
Cache	
Logical address	Physical address
……	……
……	……

ARP and RARP


Logical address
ARP
Physical address
Logical address
RARP
Physical address

ARP

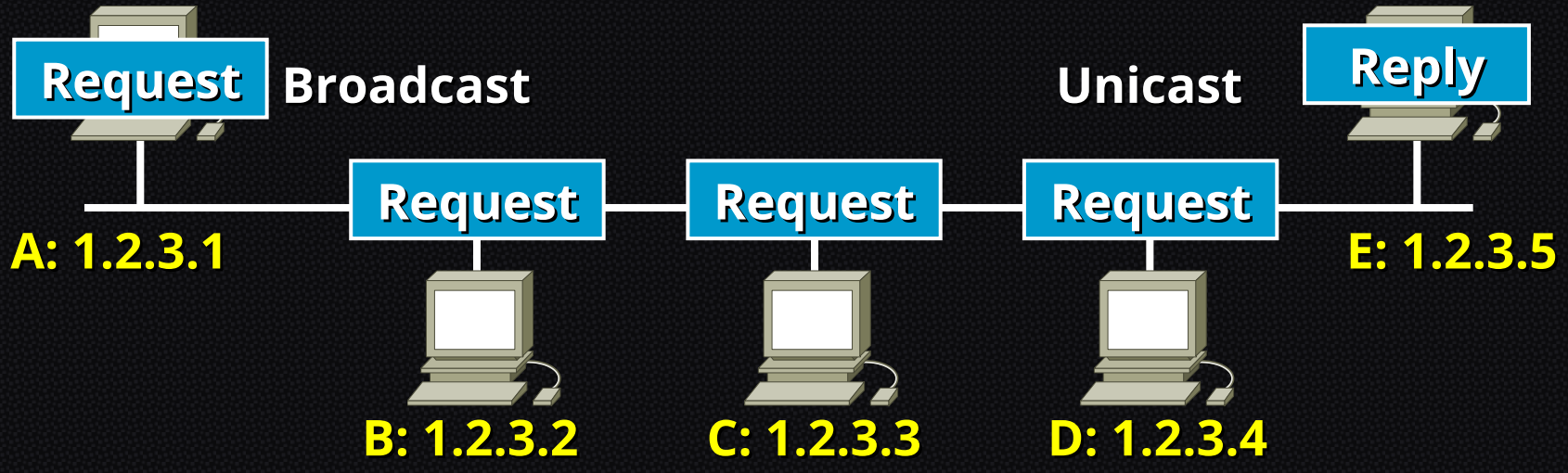
- Address Resolution Protocol , RFC 826
  - 地址解析协议： IP address → MAC address



ARP Operation

I'm looking for the physical address of a node whose IP address is: 1.2.3.5

I am the node you are looking for, and my physical address is: 0005.5D06.1418



A's ARP Cache :

IP address	MAC address
1.2.3.5	0005.5D06.1418

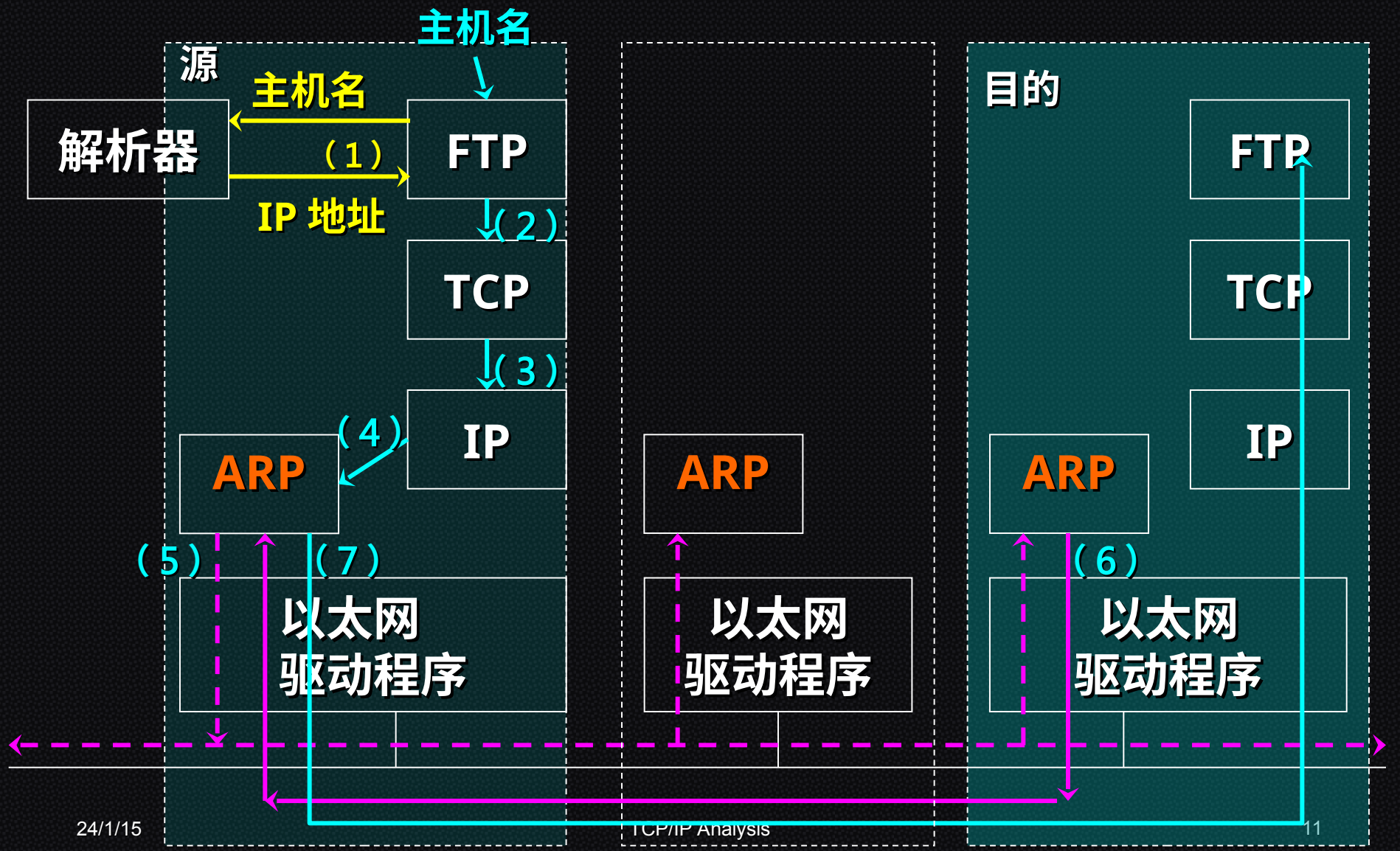
bind

ARP Cache

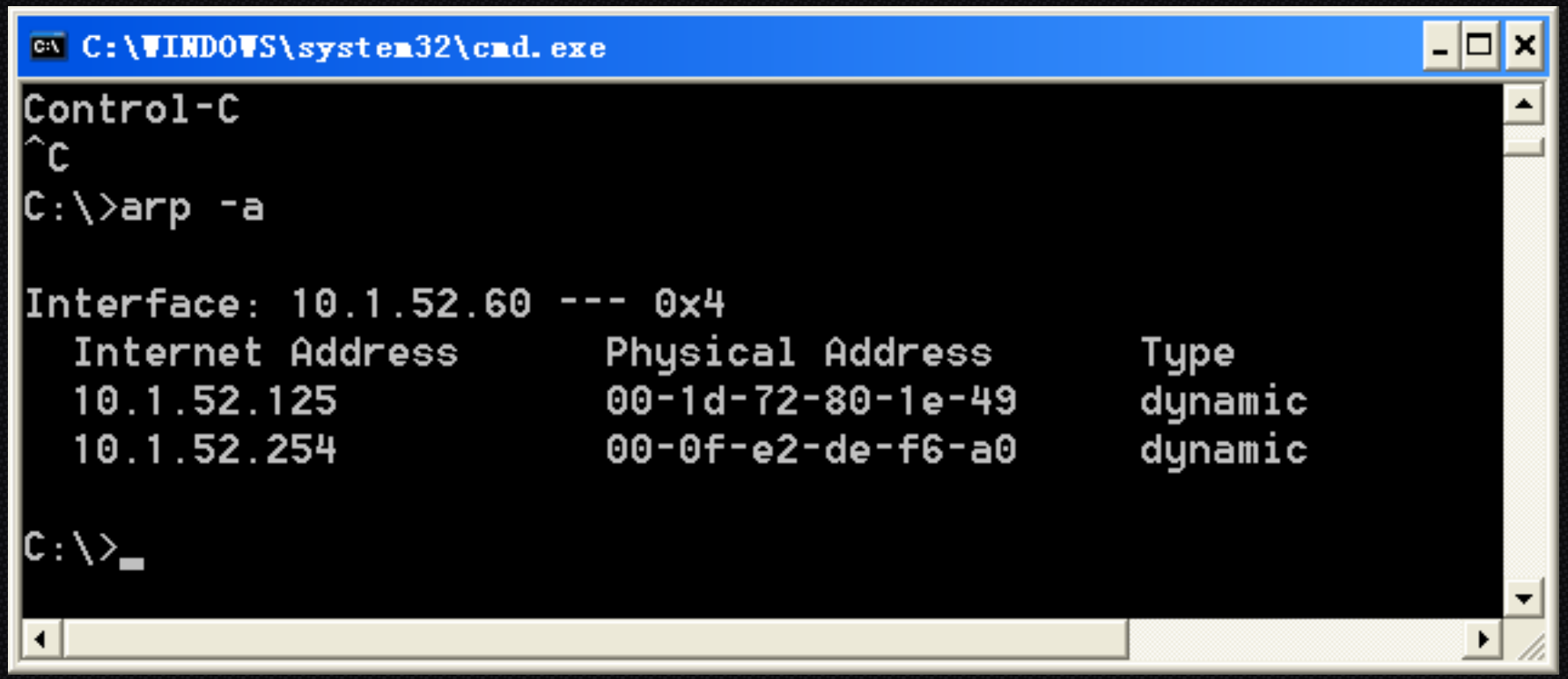
- ARP cache
  - To maintain the recent mappings from logical addresses (IP) to hardware addresses (MAC)
    - 典型存活时间： 2 minutes
  - Essential to the efficient operation of ARP
  - 举例：主机 ARP Cache

```
C:>arp -a
Interface: 172.18.64.38 --- 0x50002
Internet Address Physical Address     Type
172.18.64.62     00-03-31-b5-50-00    Dynamic
```

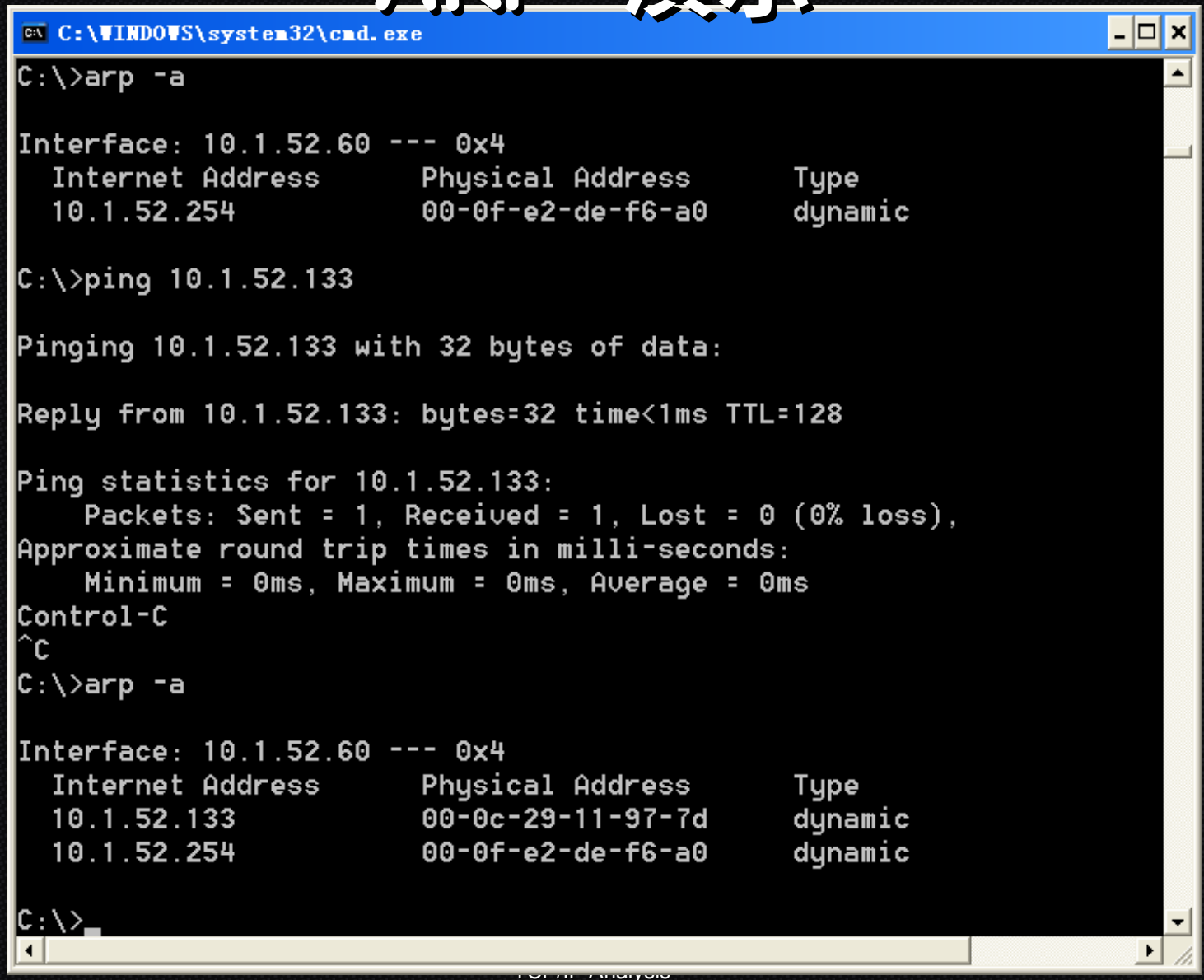
用户输入命令“ftp 主机名”时的操作



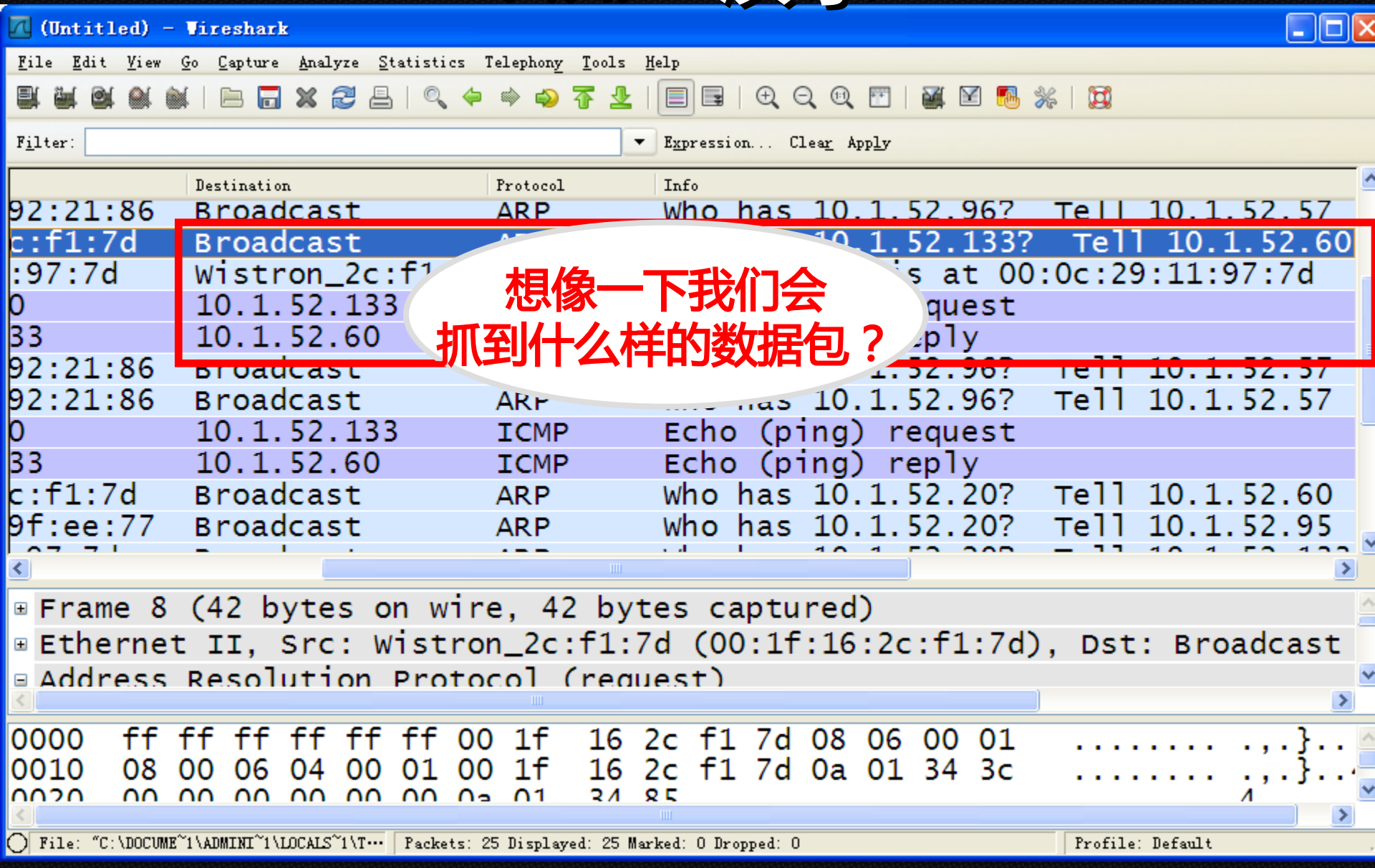
ARP 演示


C:\WINDOWS\system32\cmd.exe
Control-C
^C
C:\>arp -a
Interface: 10.1.52.60 --- 0x4
Internet Address Physical Address Type
10.1.52.125 00-1d-72-80-1e-49 dynamic
10.1.52.254 00-0f-e2-de-f6-a0 dynamic
C:\>

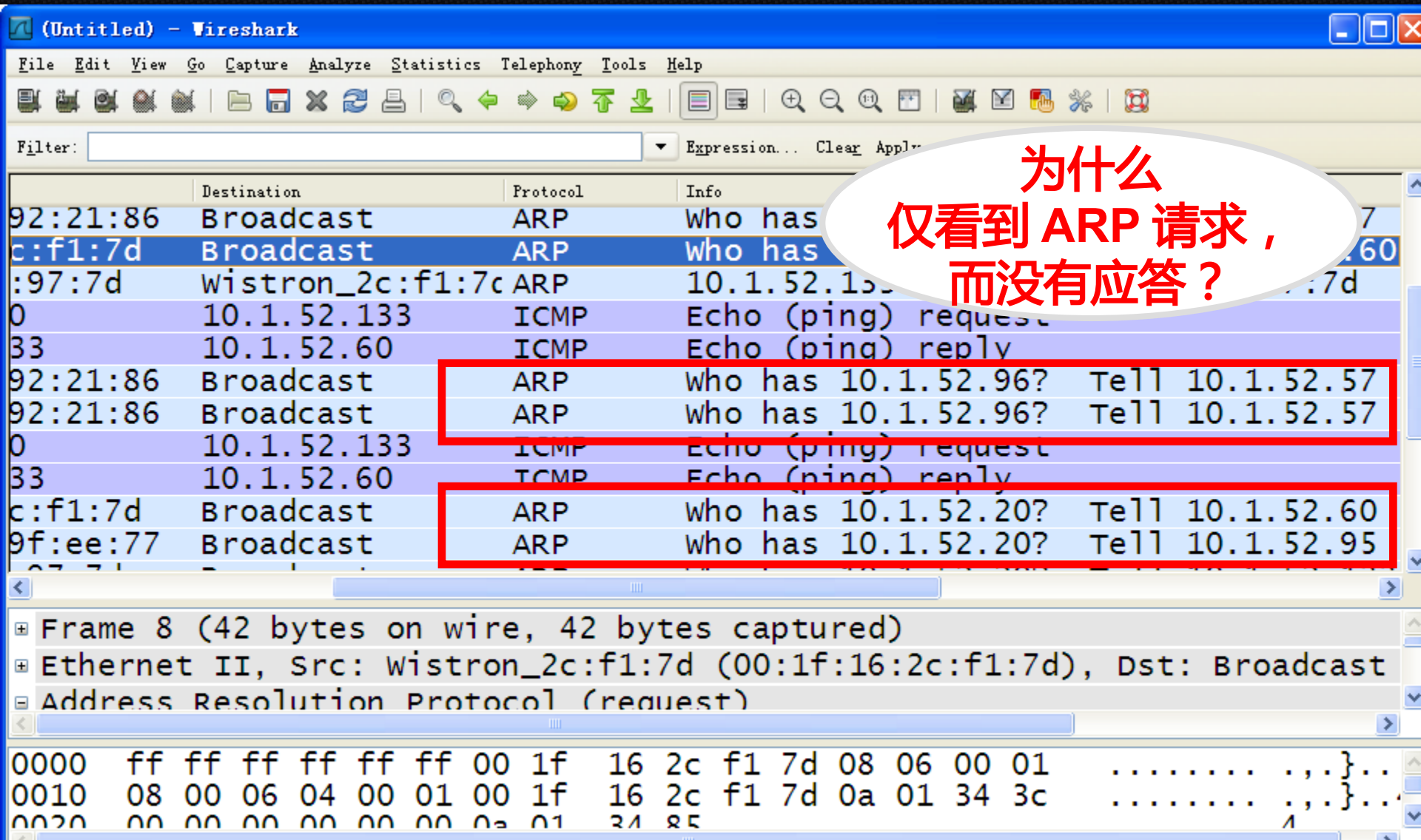
ARP 演示


C:\WINDOWS\system32\cmd.exe
C:\>arp -a
Interface: 10.1.52.60 --- 0x4
Internet Address Physical Address Type
10.1.52.254 00-0f-e2-de-f6-a0 dynamic
C:\>ping 10.1.52.133
Pinging 10.1.52.133 with 32 bytes of data:
Reply from 10.1.52.133: bytes=32 time<1ms TTL=128
Ping statistics for 10.1.52.133:
Packets: Sent = 1, Received = 1, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 0ms, Average = 0ms
Control-C
^C
C:\>arp -a
Interface: 10.1.52.60 --- 0x4
Internet Address Physical Address Type
10.1.52.133 00-0c-29-11-97-7d dynamic
10.1.52.254 00-0f-e2-de-f6-a0 dynamic
C:\>

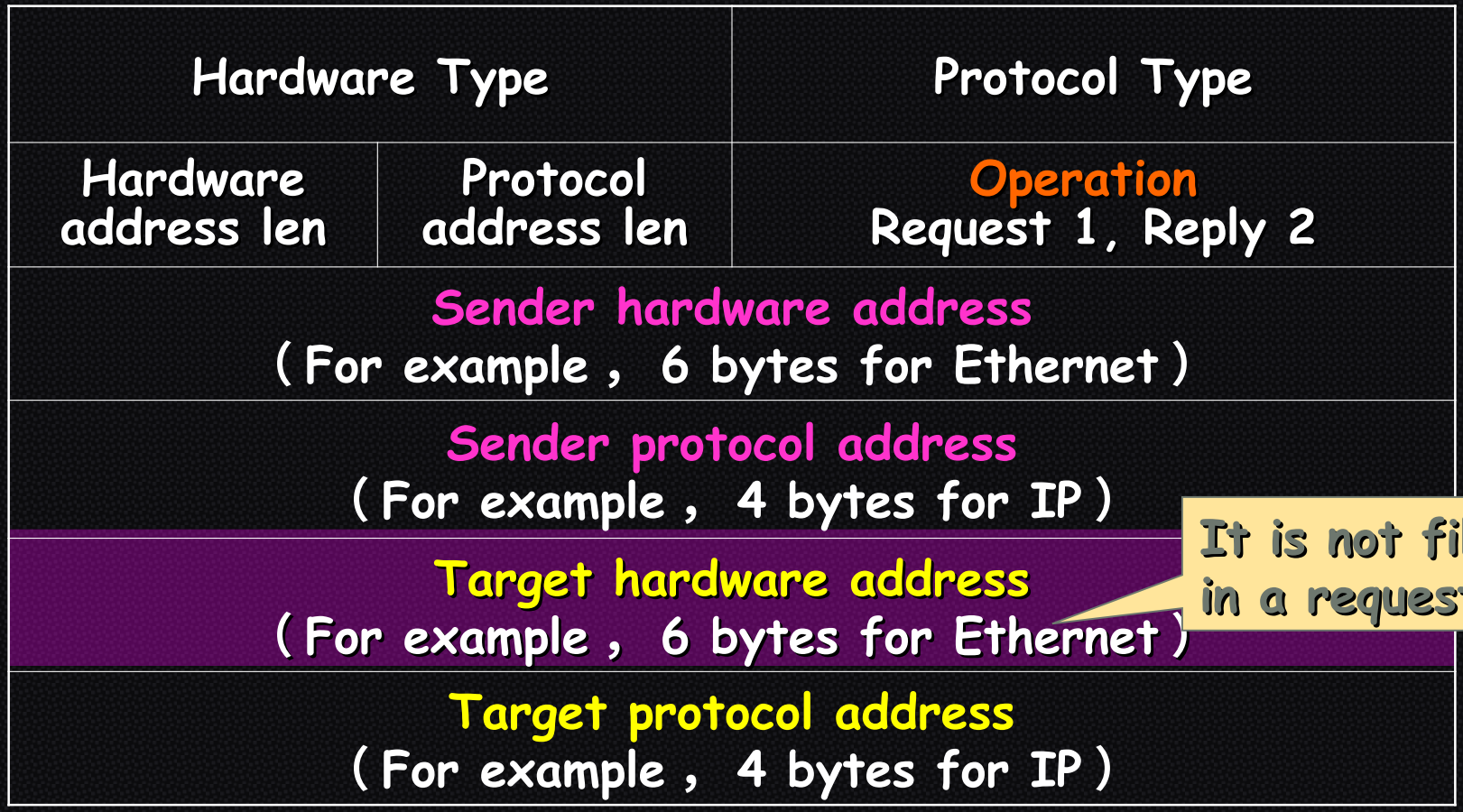
ARP 演示


(Untitled) - Wireshark
File Edit View Go Capture Analyze Statistics Telephony Tools Help
Filter: Expression... Clear Apply
Destination Protocol Info
92:21:86 Broadcast ARP Who has 10.1.52.96? Tell 10.1.52.57
c:f1:7d Broadcast 10.1.52.133? Tell 10.1.52.60
:97:7d Wistron_2c:f1 s at 00:0c:29:11:97:7d
0 10.1.52.133 quest
33 10.1.52.60 ply
92:21:86 Broadcast ARP Who has 10.1.52.96? Tell 10.1.52.57
0 10.1.52.133 ICMP Echo (ping) request
33 10.1.52.60 ICMP Echo (ping) reply
c:f1:7d Broadcast ARP Who has 10.1.52.20? Tell 10.1.52.60
9f:ee:77 Broadcast ARP Who has 10.1.52.20? Tell 10.1.52.95
想像一下我们会
抓到什么样的数据包？
Frame 8 (42 bytes on wire, 42 bytes captured)
Ethernet II, Src: Wistron_2c:f1:7d (00:1f:16:2c:f1:7d), Dst: Broadcast
Address Resolution Protocol (request)
0000 ff ff ff ff ff ff 00 1f 16 2c f1 7d 08 06 00 01
0010 08 00 06 04 00 01 00 1f 16 2c f1 7d 0a 01 34 3c
File: "C:\DOCUME~1\ADMINI~1\LOCALS~1\T... Packets: 25 Displayed: 25 Marked: 0 Dropped: 0 Profile: Default

ARP 演示

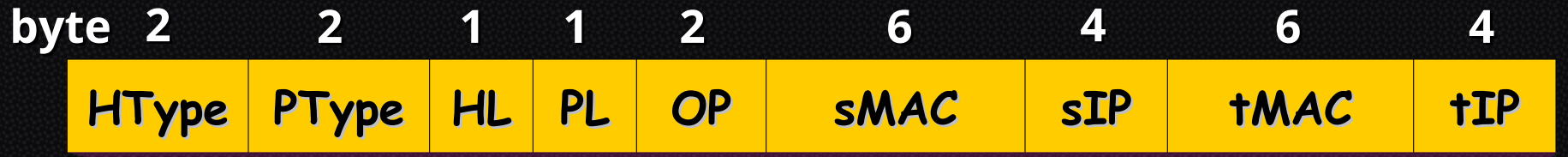

(Untitled) - Wireshark
File Edit View Go Capture Analyze Statistics Telephony Tools Help
Filter:
Expression... Clear Apply
Destination Protocol Info
92:21:86 Broadcast ARP Who has
c:f1:7d Broadcast ARP Who has
:97:7d Wistron_2c:f1:7c ARP 10.1.52.13
0 10.1.52.133 ICMP Echo (ping) request
33 10.1.52.60 ICMP Echo (ping) reply
92:21:86 Broadcast ARP Who has 10.1.52.96? Tell 10.1.52.57
92:21:86 Broadcast ARP Who has 10.1.52.96? Tell 10.1.52.57
0 10.1.52.133 ICMP Echo (ping) request
33 10.1.52.60 ICMP Echo (ping) reply
c:f1:7d Broadcast ARP Who has 10.1.52.20? Tell 10.1.52.60
9f:ee:77 Broadcast ARP Who has 10.1.52.20? Tell 10.1.52.95
为什么
仅看到 ARP 请求，
而没有应答？
Frame 8 (42 bytes on wire, 42 bytes captured)
Ethernet II, Src: Wistron_2c:f1:7d (00:1f:16:2c:f1:7d), Dst: Broadcast
Address Resolution Protocol (request)
0000 ff ff ff ff ff ff 00 1f 16 2c f1 7d 08 06 00 01
0010 08 00 06 04 00 01 00 1f 16 2c f1 7d 0a 01 34 3c

ARP Packet


Hardware Type
Protocol Type
Hardware address len
Protocol address len
Operation
Request 1, Reply 2
Sender hardware address
(For example , 6 bytes for Ethernet)
Sender protocol address
(For example , 4 bytes for IP)
Target hardware address
(For example , 6 bytes for Ethernet)
Target protocol address
(For example , 4 bytes for IP)
It is not filled in a request

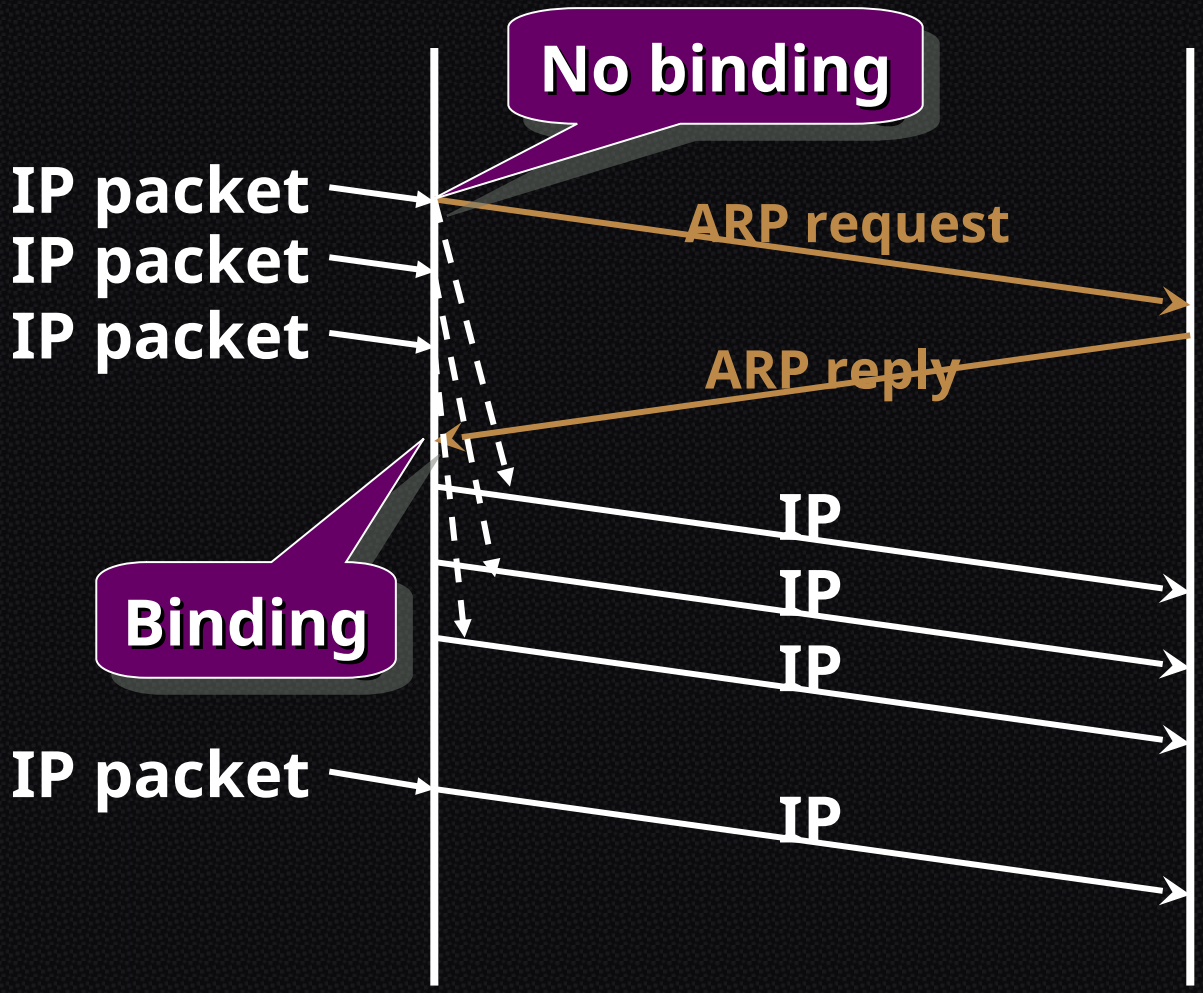
Encapsulation of ARP packet

- The byte order of ARP packet

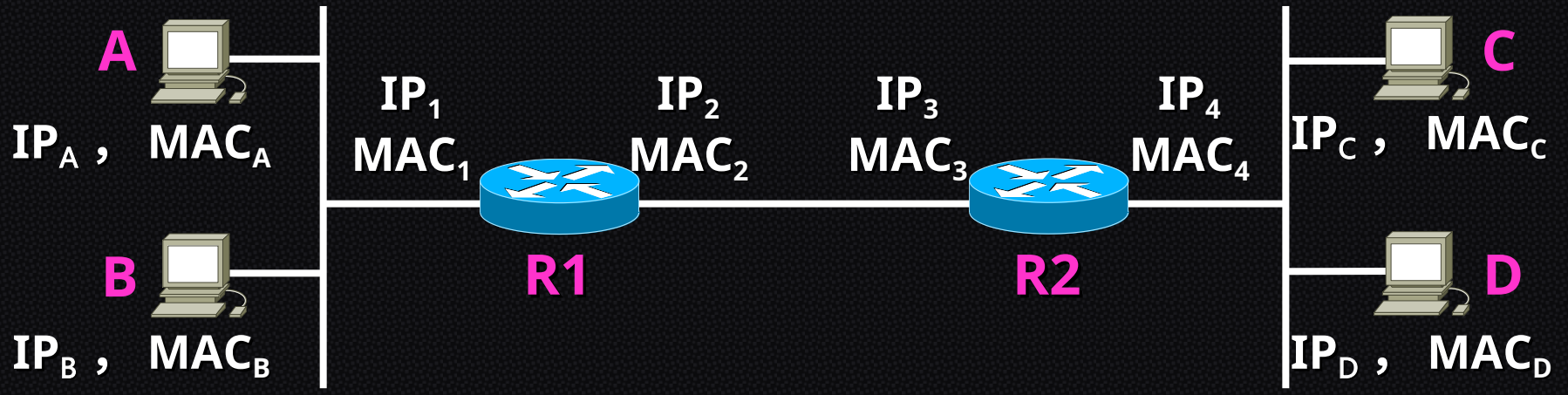


- Example： Ethernet frame

IP Packet and ARP Packet

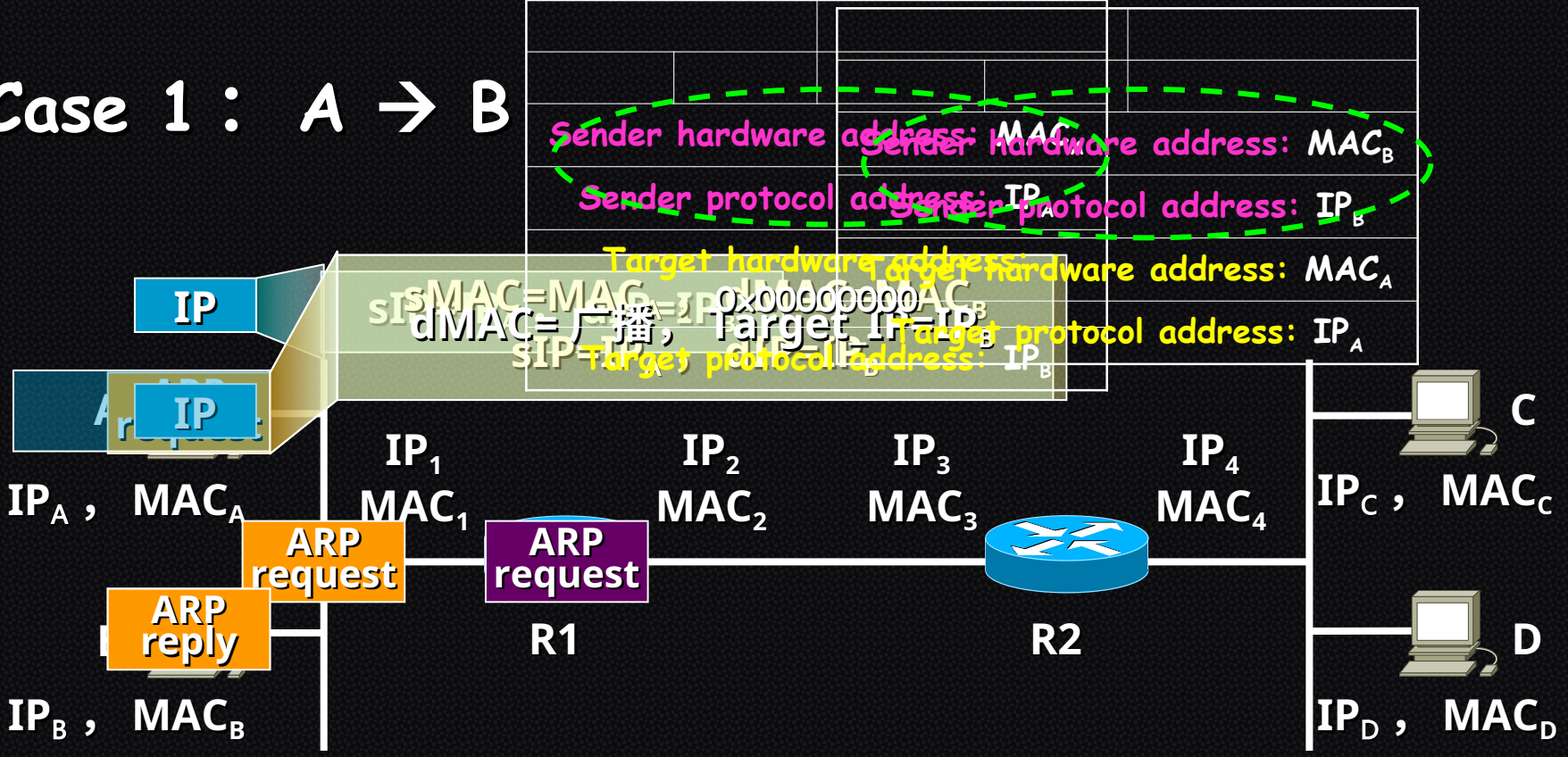


ARP Process



- Case 1 : A → B
  - In same IP network
- Case 2 : A → D
  - In different IP network

Case 1 : A → B



A's ARP Cache :

IP Address	MAC Address
IP_B	MAC_B

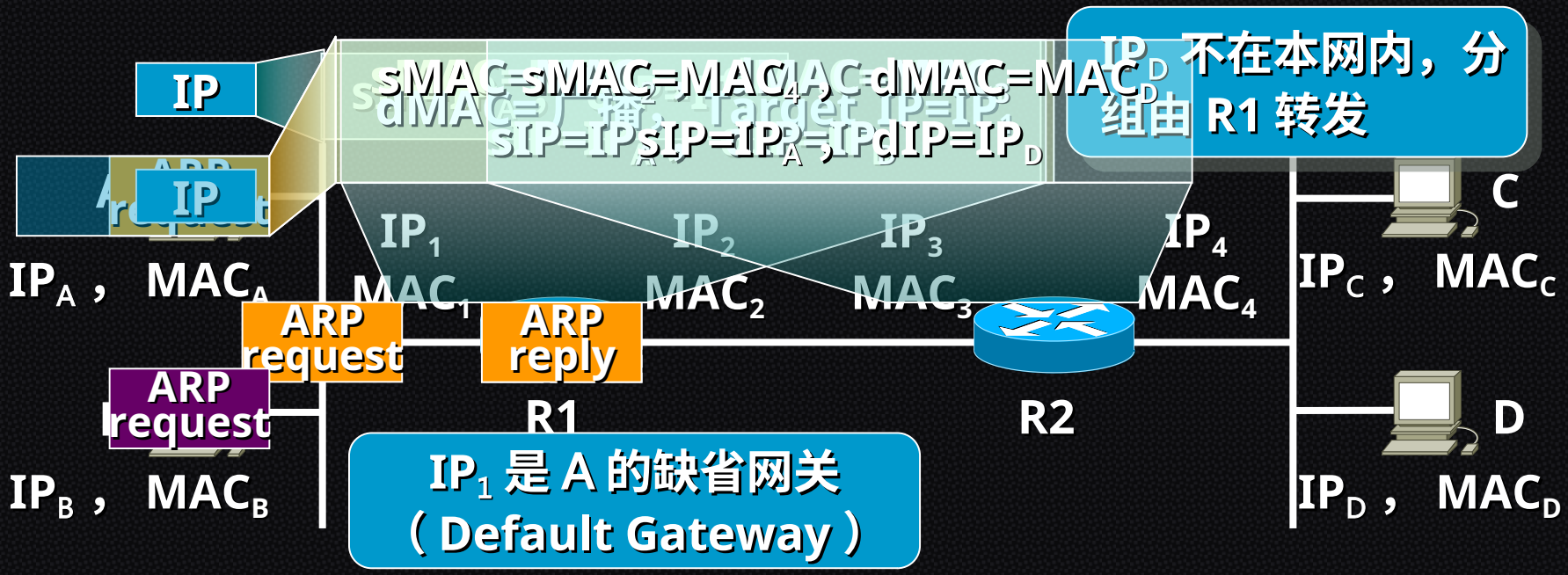
B's ARP Cache :

IP_A	MAC_A

R1's ARP Cache :

IP_A	MAC_A

Case 2： A → D



A's ARP Cache：

IP Address	MAC Address
IP_1	MAC_1

B's ARP Cache：

IP_A	MAC_A

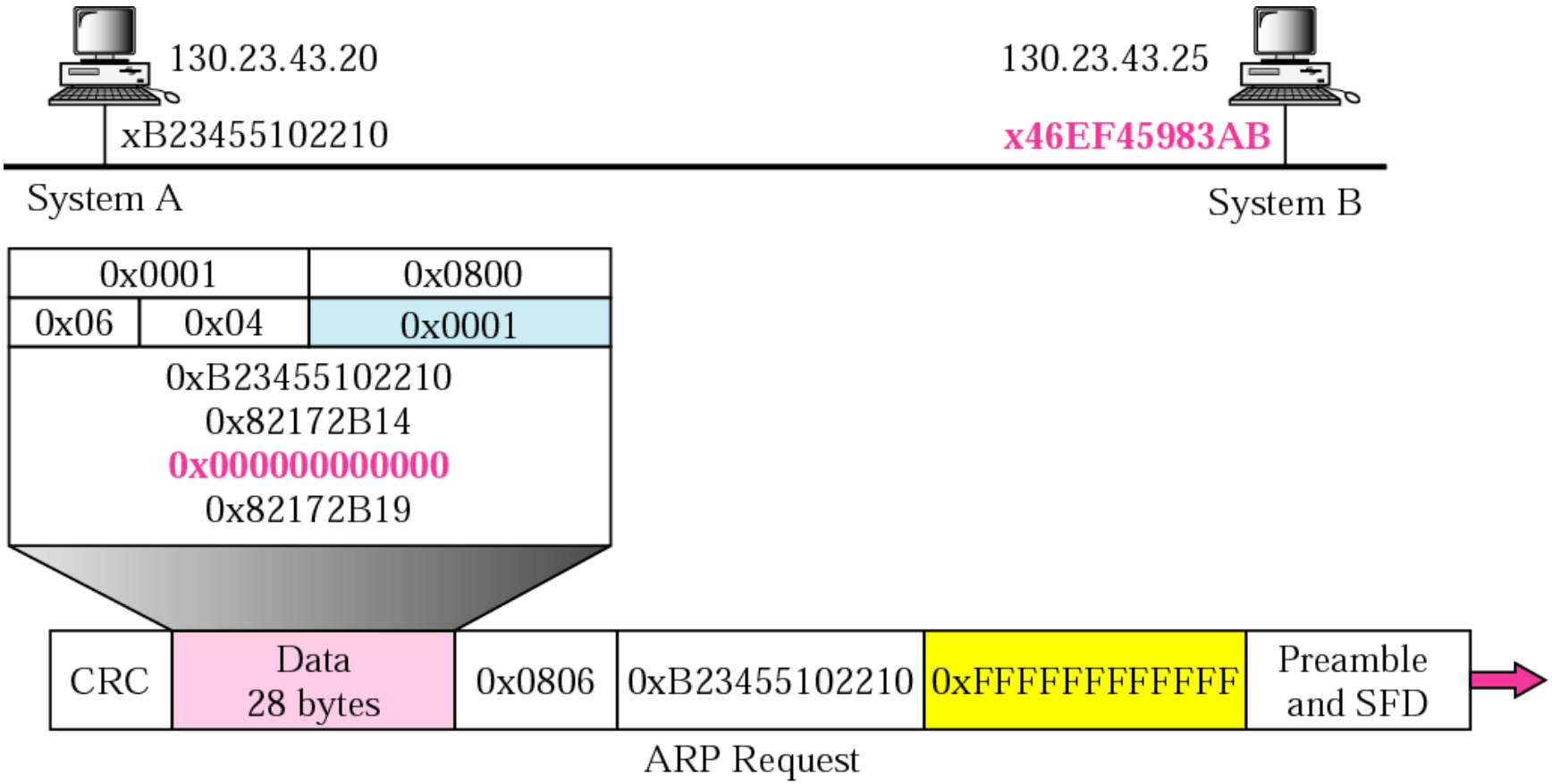
R1's ARP Cache：

IP_A	MAC_A

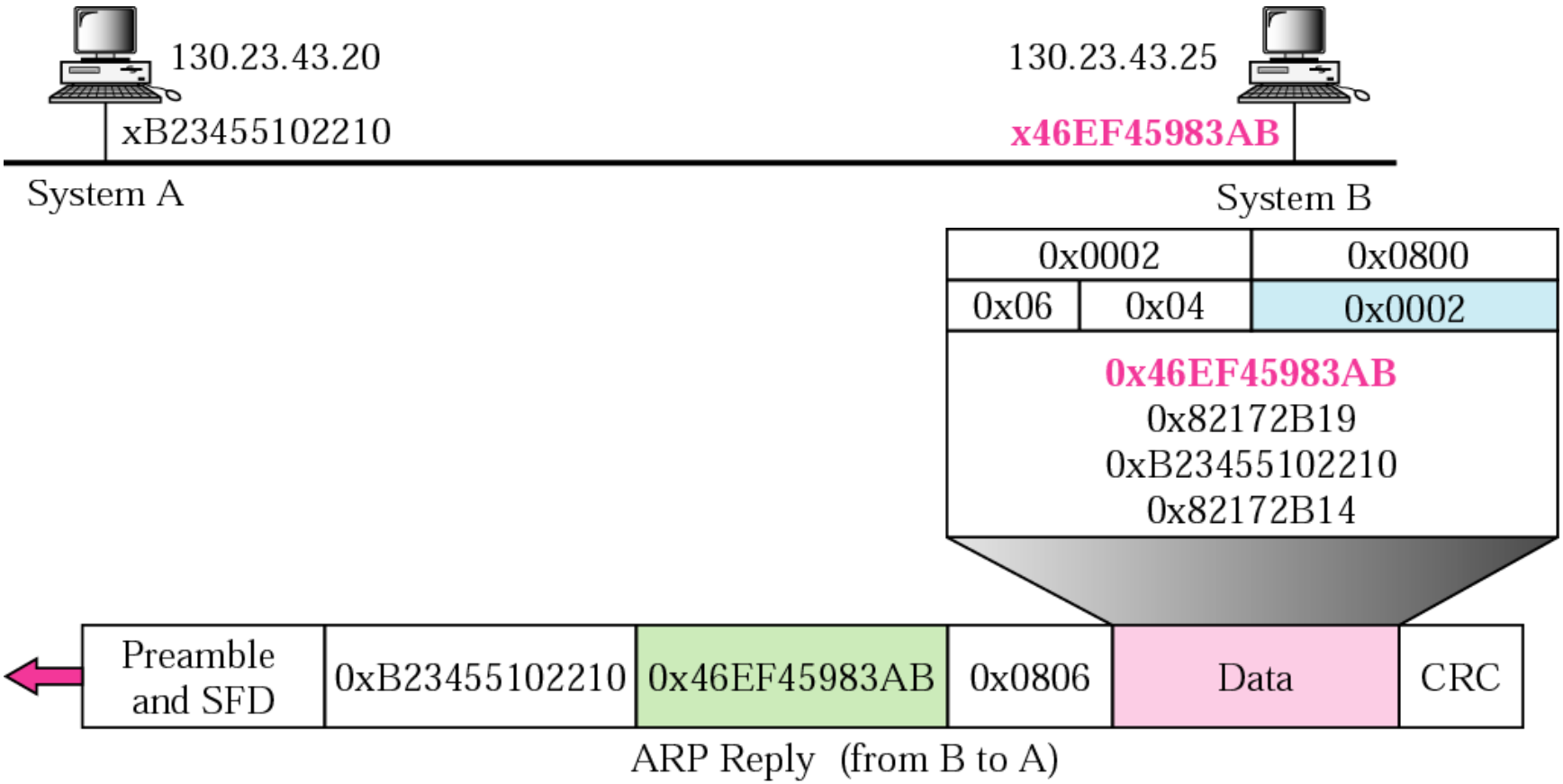
Example

- A host with IP address 130.23.43.20 and physical address 0xB23455102210 has a packet to send to another host with IP address 130.23.43.25 and physical address 0xA46EF45983AB. The two hosts are on the same Ethernet network. Show the ARP request and reply packets encapsulated in Ethernet frames.

Example: ARP Request

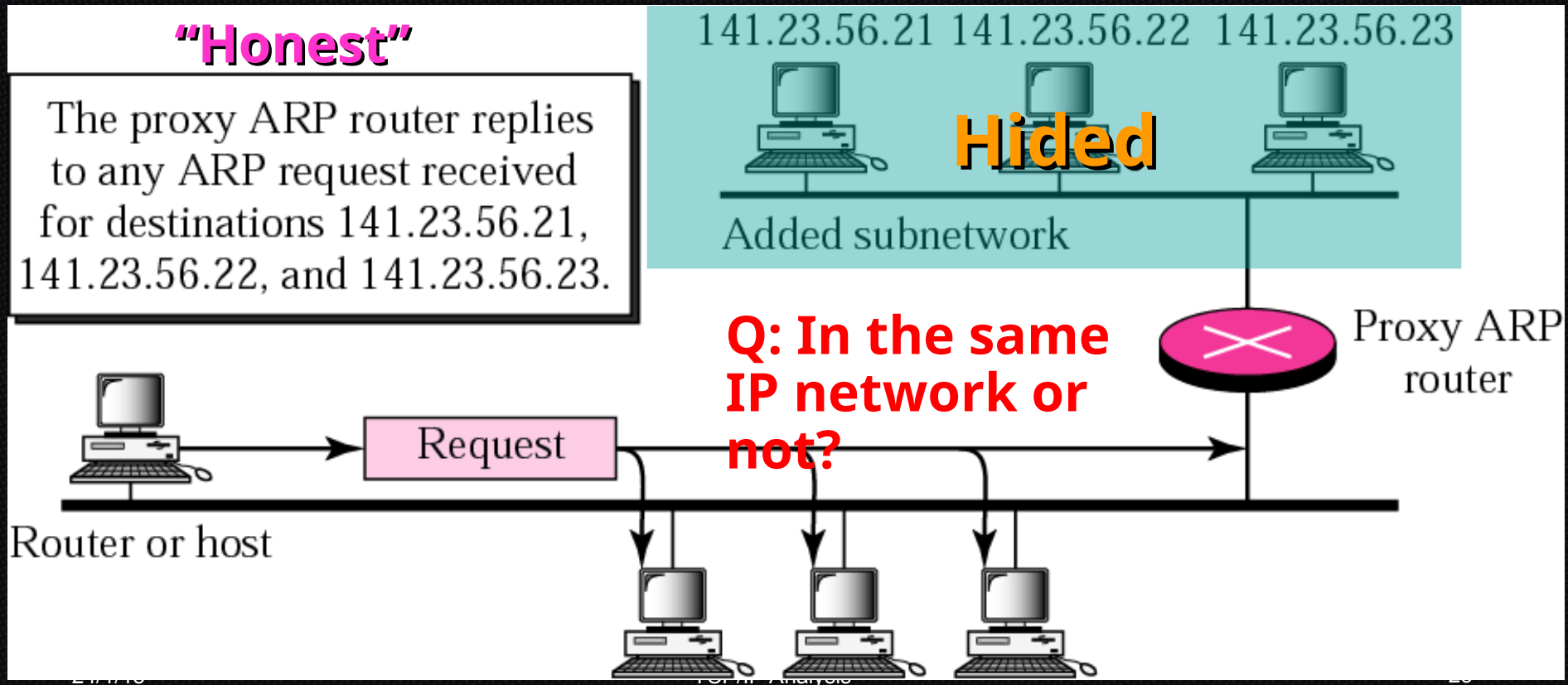


Example: ARP Request



Proxy ARP （代理 ARP ）

- Proxy ARP:
  - 代表另一个物理网络中一组主机回答 ARP Request，在 ARP Reply 中通告自己的 MAC 地址（即将解析的 IP 与代理 ARP 的 MAC 绑定）
  - To fool the sender of the ARP request into thinking that the router is the destination host, when in fact the destination host is "on the other side" of the router



Answer

- RFC 925: Multi-LAN Address Resolution
  - Explicit subnets
  - Transparent subnets （Extended ARP）
- RFC 1027: Using ARP to Implement Transparent Subnet Gateways
  - Routers: Explicit subnets
  - Hosts: Transparent subnets

“ From the host point of view, there are no subnets, and their physical networks are simply one big IP network. ”

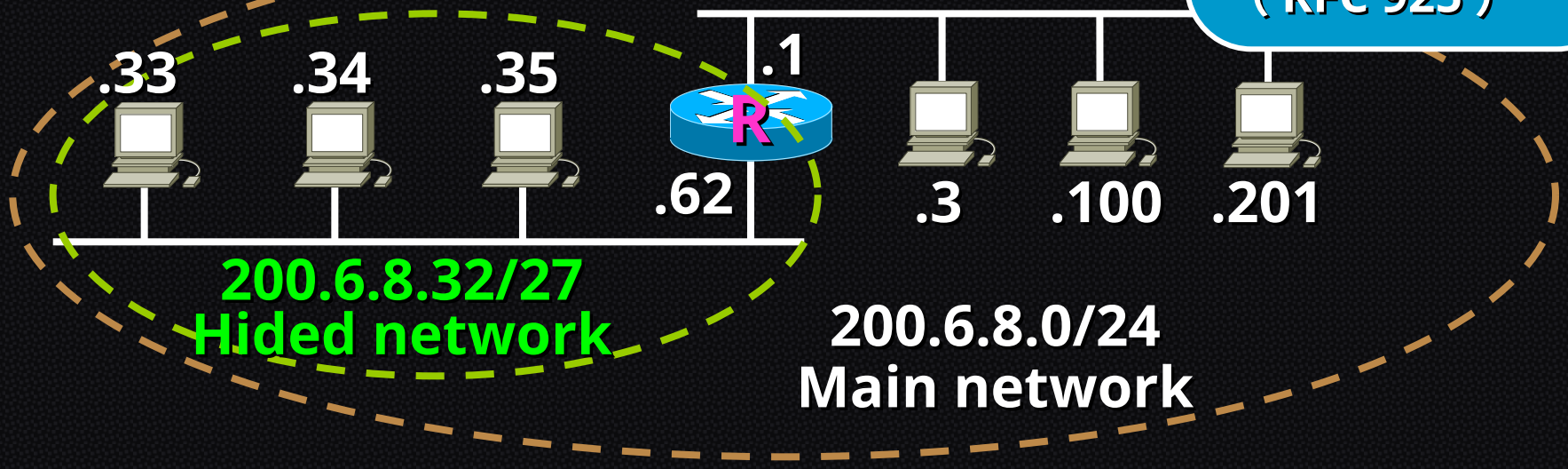
Proxy ARP

关键：创建子网，但不需要重新划分子网地址

- 功能
  - To be used to create a subnetting effect
    - 两个物理网络，具有相同网络地址
    - 使用路由器分别连接这两个网络，并执行 ARP 代理，实现两个逻辑子网
- 方案：路由器上运行 Proxy ARP 软件
  → transparent subnet gateway
  - 当路由收到对特定主机的 ARP 请求时，用自己的物理地址（接收端口）进行 ARP 应答
  - 代理 ARP 应答的条件（同时满足）：
    1. 与源站点不在同一逻辑子网的主机
    2. 路由器有到达该节点的路由（非默认路由）
    3. 且路由表项记录的发送接口≠接收该 ARP 请求的接口

Discussion

设想：
R 代替 .35 向 .3 返回 ARP 应答
—— Proxy ARP
（RFC 925）

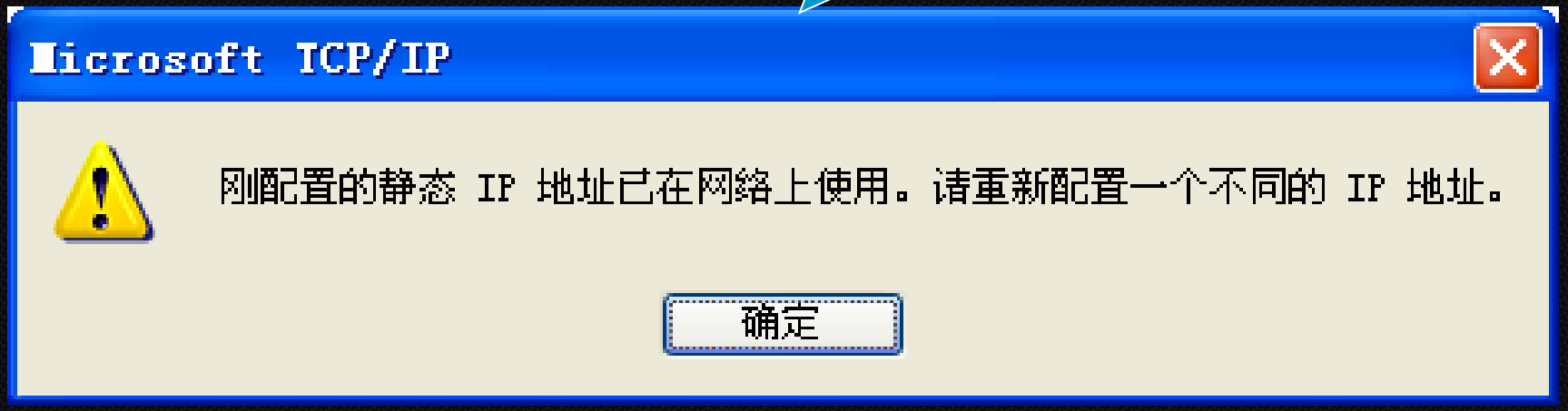


- .3 向 .35 发送 IP 分组 → IP 发送失败
  - .3 广播请求 .35 的 ARP 分组，R 不转发广播 → ARP 失败
- .35 向 .3 发送 IP 分组 → IP 发送成功
  - .35 广播请求 .62 的 ARP 分组 → ARP 成功

Gratuitous ARP

- 故意 ARP

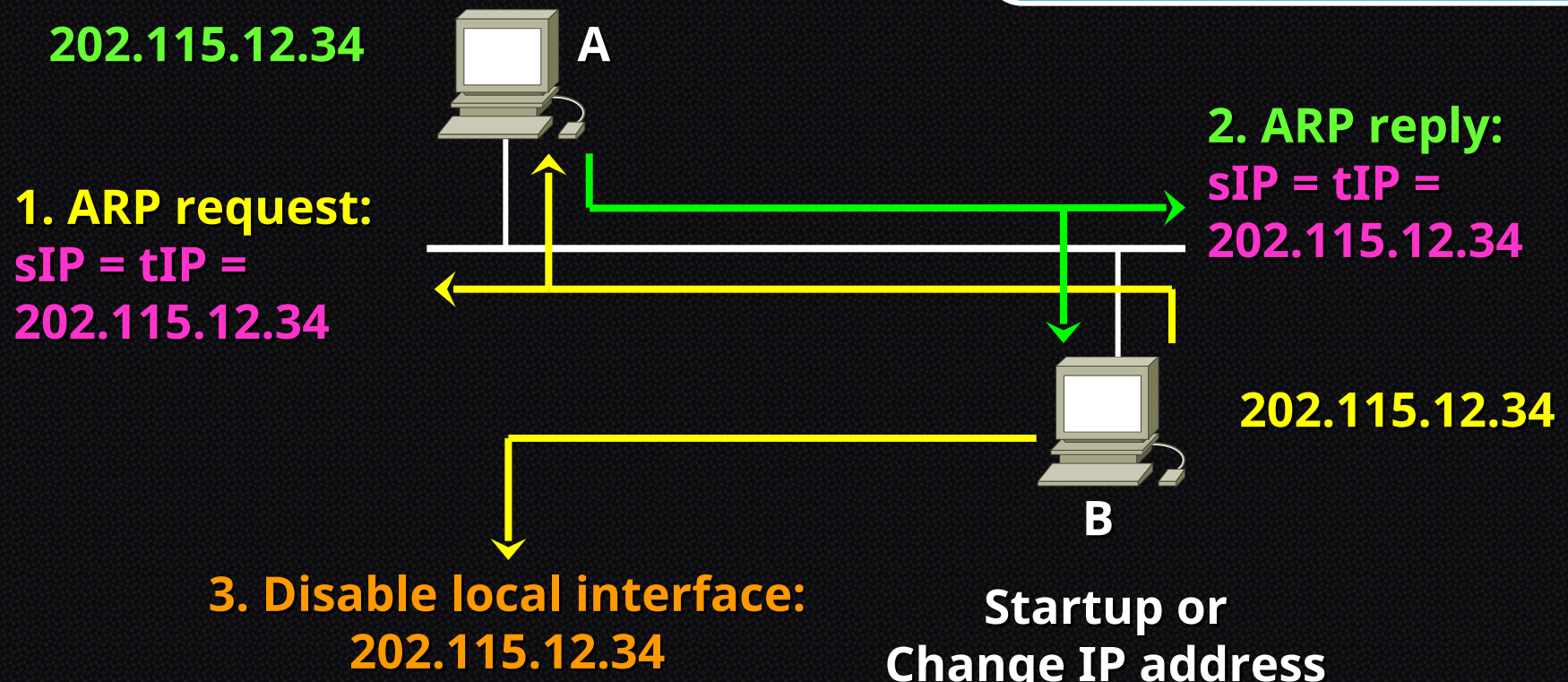
大家应该看到过这个对话框，这个是如何实现呢？



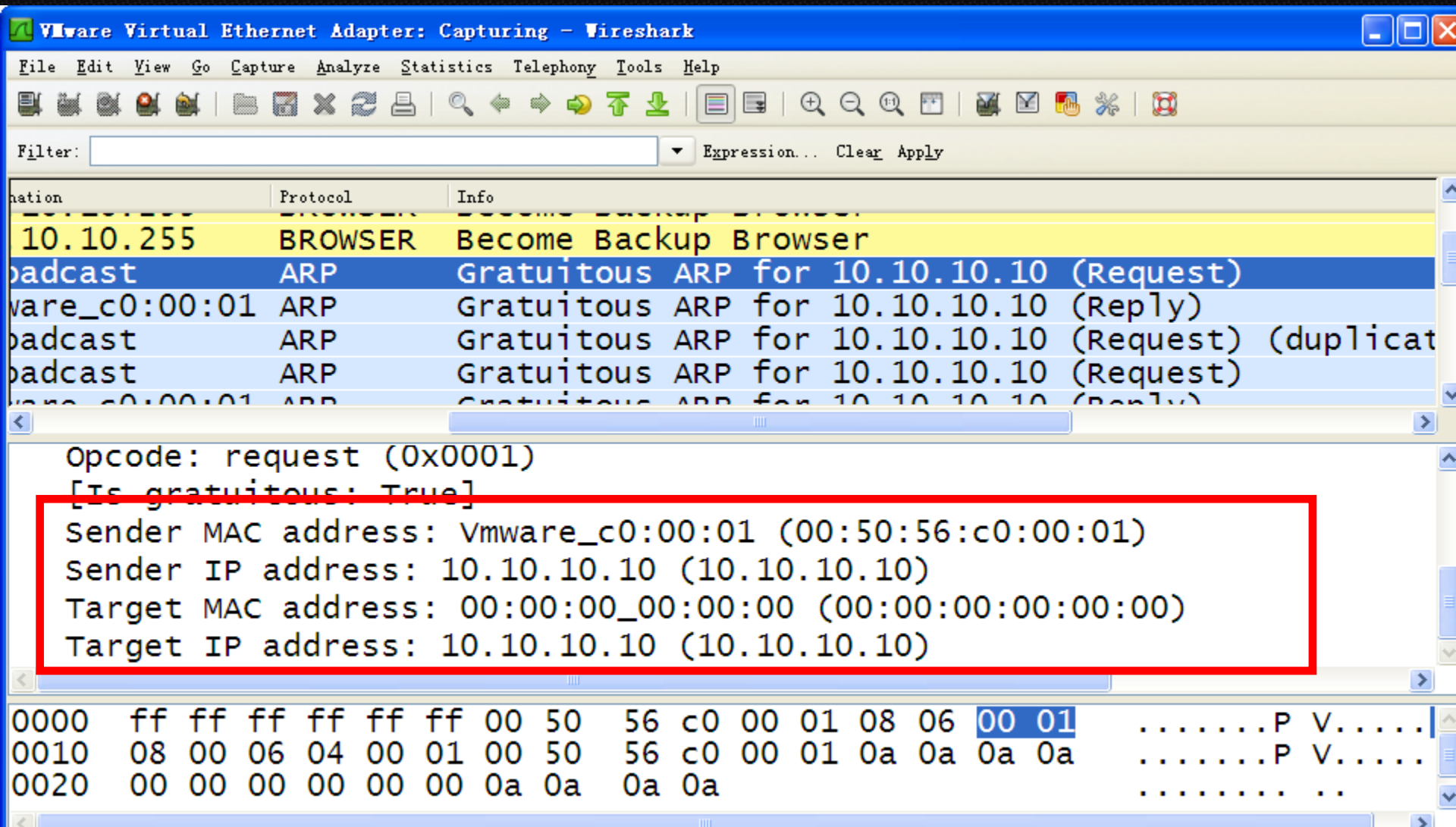
Gratuitous ARP

- 功能:
  - Duplicate address test

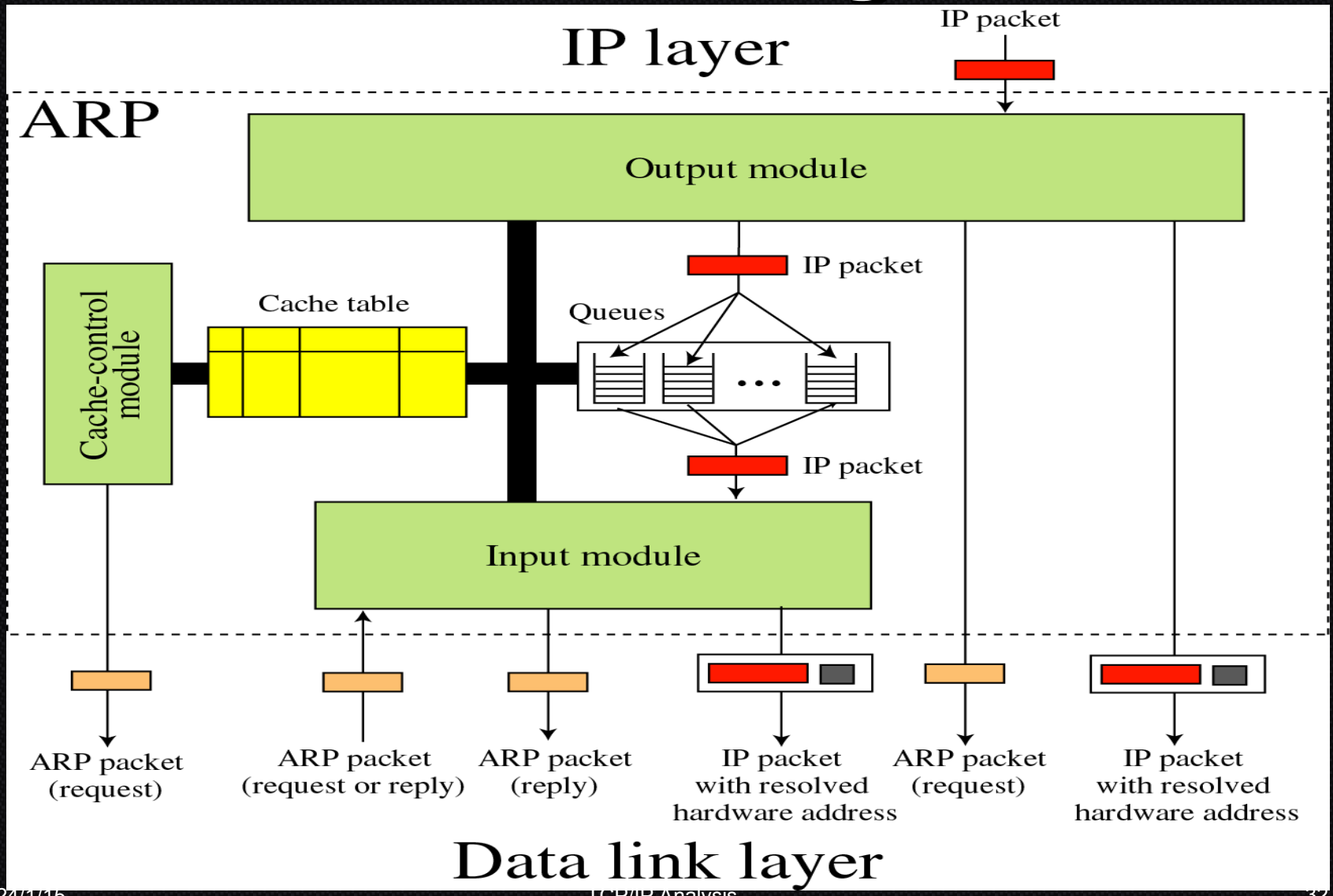
W. Stevens, TCP/IP Illustrated Volume 1: The Protocol



Gratuitous ARP



ARP Package


IP layer
IP packet
ARP
Output module
IP packet
Cache table
Queues
Cache-control module
IP packet
Input module
ARP packet (request)
ARP packet (request or reply)
ARP packet (reply)
IP packet with resolved hardware address
ARP packet (request)
IP packet with resolved hardware address
Data link layer

ARP Cache

- Host （Windows XP）

```
C:\> arp –a

Interface: 202.115.12.34 --- 0x2
  Internet Address      Physical Address      Type
  202.115.12.33         00-90-27-a7-98-41     dynamic
  202.115.12.47         00-90-27-1d-d9-94     dynamic
  202.115.12.62         00-90-27-1a-67-e7     dynamic
```

- Router （Cisco）

```
Router# show arp
Protocol  Address          Age (min)  Hardware Addr   Type   Interface
Internet  202.115.12.33         --   0090.27a7.9841  ARPA   Ethernet0
Internet  202.115.12.34          5   0005.5d06.1418  ARPA   Ethernet0
Internet  202.115.13.1          --   00e0.7bc0.b205  ARPA   Ethernet1
```

思考

- 更新 ARP 绑定时，发现已有的绑定与新的绑定不一样，是保持已有的还是替换它？
- 封装 IP 报文的以太帧中的源 MAC 和 IP 报文中的源 IP 可否用于刷新 ARP 表项？目的 MAC 和目的 IP 呢？
- 教材第 150 页中 ARP 输入模块描述与 RFC 826 中的 Packet Reception 一节有矛盾

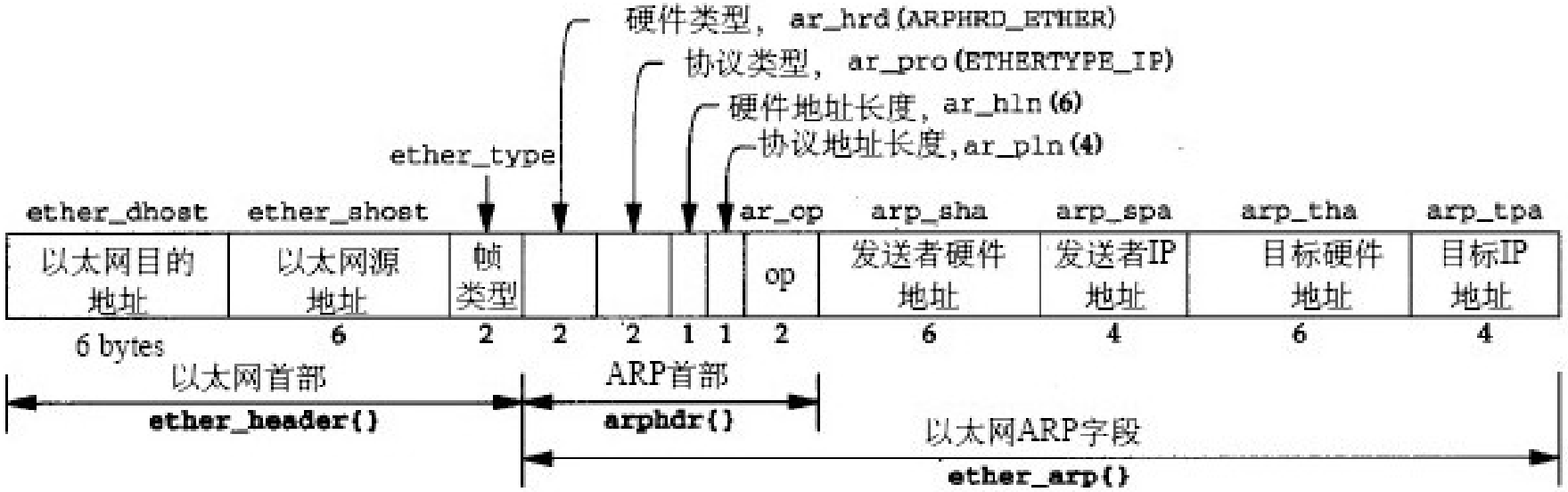
RFC References

- RFC 1122: Requirements for Internet Hosts -- Communication Layers
  - Section 2.3.2: ARP cache, ARP packet queue
- RFC 1812: Requirements for IPv4 Routers
  - Section 3.3.2
- RFC 1433: Directed ARP
  - ARP helper address
- RFC 1868: ARP Extension – UNARP
  - Announce leaving

Summary

- ARP
  - 作用、分组格式
  - 操作
    - 何时发送、送给谁
    - 发送方式（单播、广播）
    - 发送内容（ ARP 分组各字段的具体取值，以及封装该分组的以太帧中各字段的具体取值）
  - Proxy ARP 、 Gratuitous ARP

ARP 报文格式



硬件类型：表示硬件地址的类型，值为1表示以太网地址
协议类型：表示要映射的协议地址类型。它的值为0x0800表示IP地址类型
硬件地址长度和协议地址长度以字节为单位，对于以太网上的IP地址的ARP请求或应答来说，他们的值分别为6和4；
操作类型（op）:1表示ARP请求，2表示ARP应答
发送端MAC地址：发送方设备的硬件地址；
发送端IP地址：发送方设备的IP地址；
目标MAC地址：接收方设备的硬件地址。
目标IP地址：接收方设备的IP地址。

ARP 报文结构的实现

- XINU79\PCXNET\SRC\H\ARP.H

```
struct  arp {
    short   ar_hwtype;  /* hardware type            */
    short   ar_prtype;  /* protocol type            */
    char    ar_hwlen;   /* hardware address length      */
    char    ar_prlen;   /* protocol address length      */
    short   ar_op;      /* ARP operation (see list above)   */
    char    ar_addrs[1];    /* sender and target hw & proto addrs   */
/*  char    ar_sha[???];    /* sender's physical hardware address   */
/*  char    ar_spa[???];    /* sender's protocol address (IP addr.) */
/*  char    ar_tha[???];    /* target's physical hardware address   */
/*  char    ar_tpa[???];    /* target's protocol address (IP)   */
};
```

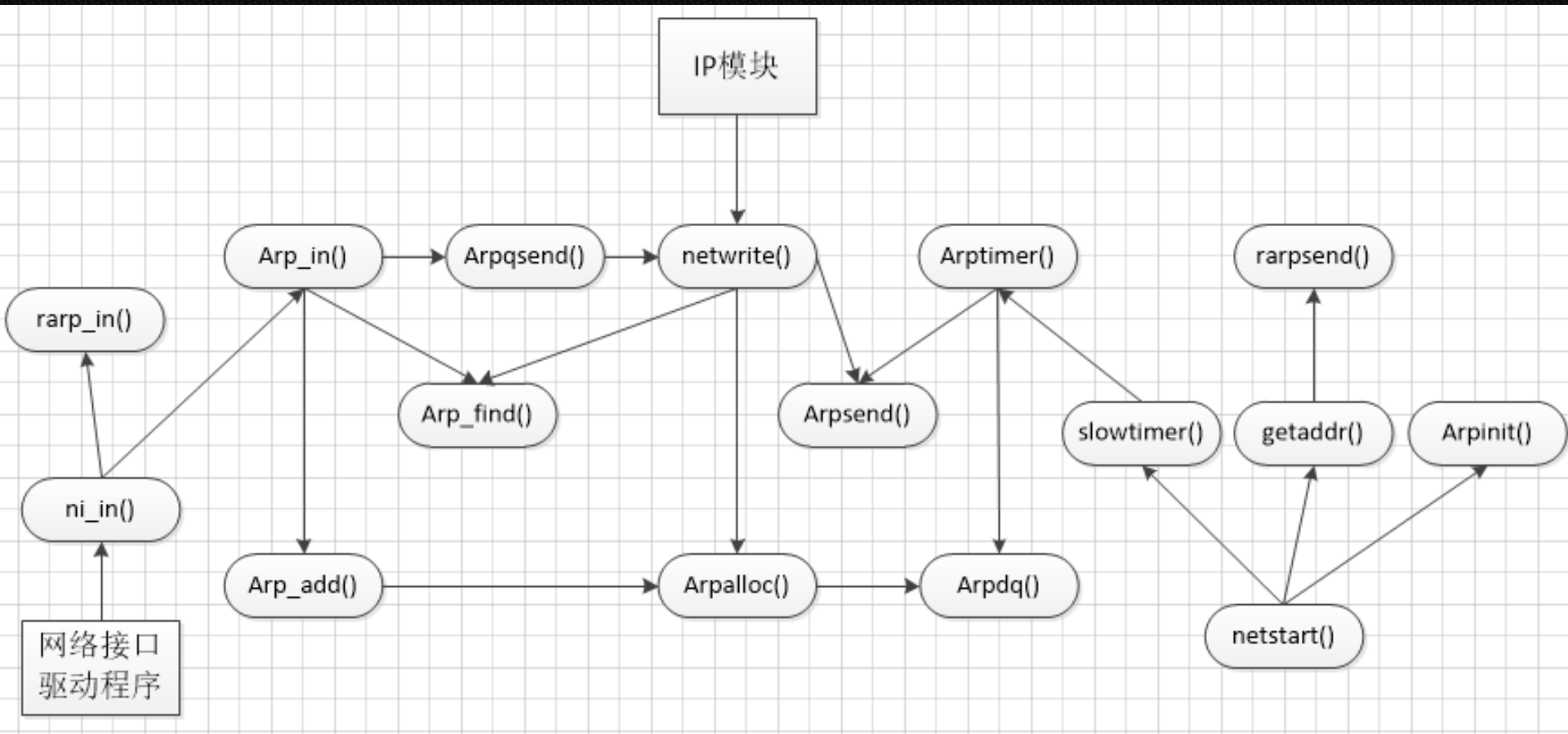
ARP 报文结构的实现

```
/* Definitions of codes used in operation field of ARP packet */

#define AR_REQUEST  1   /* ARP request to resolve address   */
#define AR_REPLY    2   /* reply to a resolve request       */

#define RA_REQUEST  3   /* reverse ARP request (RARP packets)   */
#define RA_REPLY    4   /* reply to a reverse request (RARP ")  */

struct  arp {
/*  char    ar_sha[???];    /* sender's physical hardware address   */
/*  char    ar_spa[???];    /* sender's protocol address (IP addr.) */
/*  char    ar_tha[???];    /* target's physical hardware address   */
/*  char    ar_tpa[???];    /* target's protocol address (IP)   */
};

#define SHA(p)  (&p->ar_addrs[0])
#define SPA(p)  (&p->ar_addrs[p->ar_hwlen])
#define THA(p)  (&p->ar_addrs[p->ar_hwlen + p->ar_prlen])
#define TPA(p)  (&p->ar_addrs[(p->ar_hwlen*2) + p->ar_prlen])


#define MAXHWALEN   EP_ALEN /* Ethernet             */
#define MAXPRALEN   IP_ALEN /* IP                   */

#define ARP_TSIZE   50  /* ARP cache size           */
#define ARP_QSIZE   10  /* ARP port queue size          */
```

ARP 软件整体结构

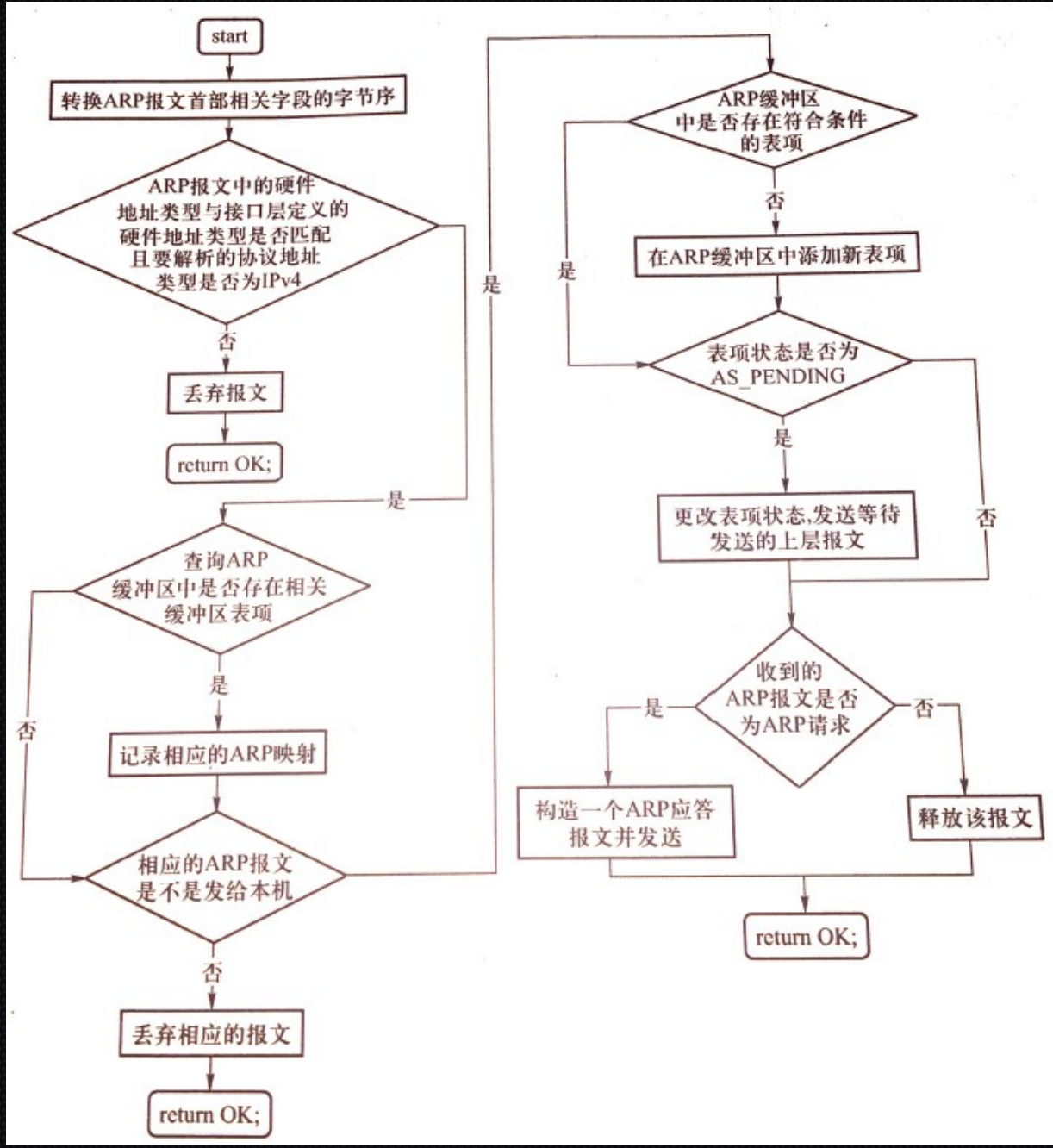
- Arp 软件初始化
- Arp 报文处理
- Arp 缓存维护
- Arp 请求发送

ARP 软件整体结构


IP模块
Arp_in()
Arpqsend()
netwrite()
Arptimer()
rarpsend()
rarp_in()
Arp_find()
Arpsend()
slowtimer()
getaddr()
Arpinit()
ni_in()
Arp_add()
Arpalloc()
Arpdq()
netstart()
网络接口
驱动程序

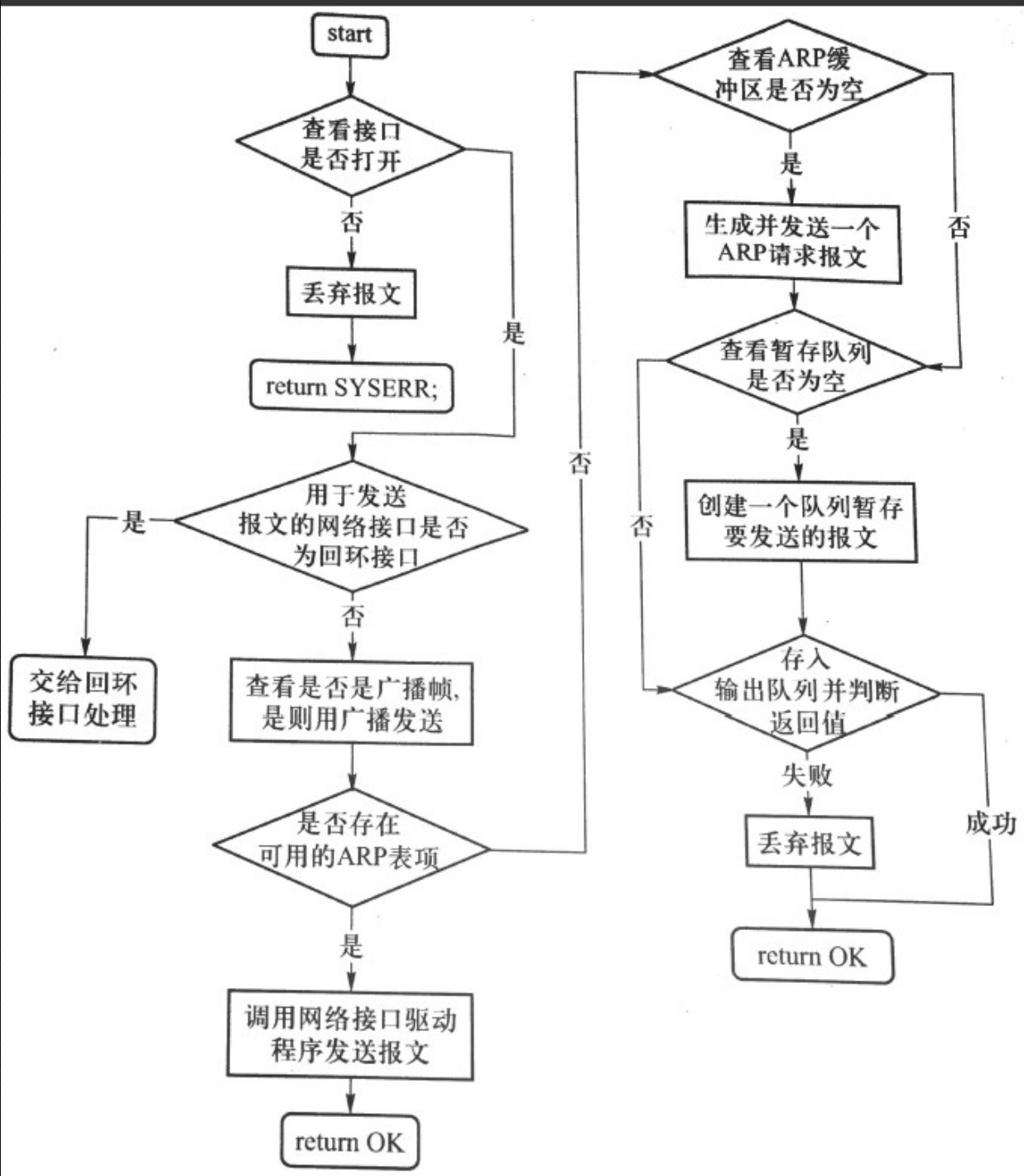
ARP 输入处理流程

Arp_in.c 文件



ARP 发送请求

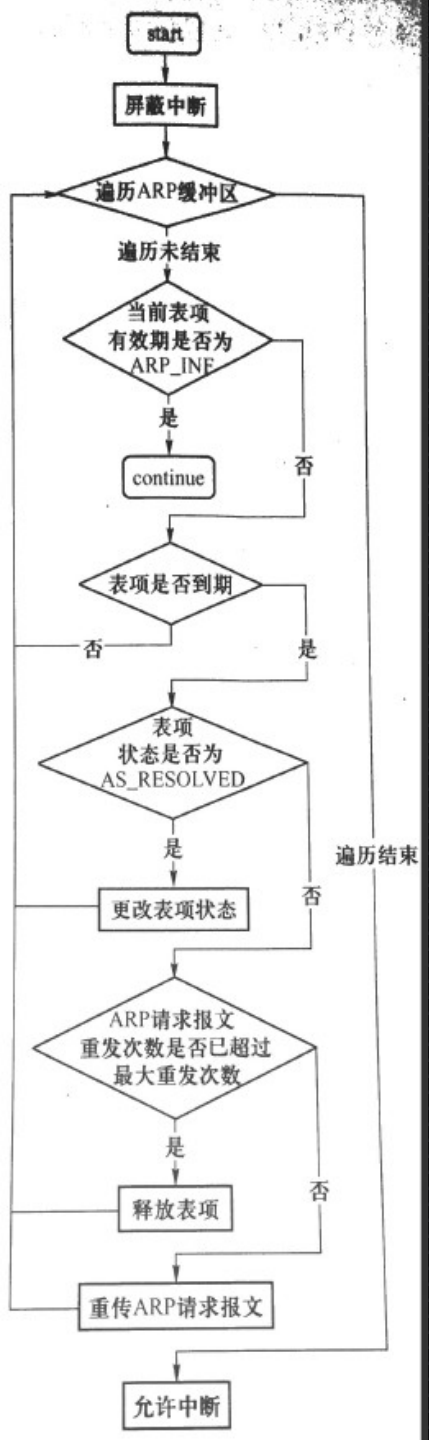
..net/netwrite.c



ARP 缓存管理

- Arptimer.c

```
struct  arpentry {      /* format of entry in ARP cache     */
    short   ae_state;   /* state of this entry (see below)  */
    short   ae_hwtype;  /* hardware type                    */
    short   ae_prtype;  /* protocol type                    */
    char    ae_hwlen;   /* hardware address length          */
    char    ae_prlen;   /* protocol address length          */
    struct netif *ae_pni;   /* pointer to interface structure   */
    int ae_queue;   /* queue of packets for this address    */
    int ae_attempts;    /* number of retries so far         */
    int ae_ttl;     /* time to live                         */
    char    ae_hwa[MAXHWALEN];  /* Hardware address         */
    char    ae_pra[MAXPRALEN];  /* Protocol address         */
};
```



Arp 攻击

- Arp 欺骗攻击
- ARP 溢出攻击

```
23      parp->ar_hwtype = net2hs(parp->ar_hwtype);
24      parp->ar_prtype = net2hs(parp->ar_prtype);
25      parp->ar_op = net2hs(parp->ar_op);
26      if (parp->ar_hwtype != pni->ni_hwtype ||
27          parp->ar_prtype != EPT_IP) {
28          freebuf(pep);
29          return OK;
30      }
31
32      if (pae = arpfind(SPA(parp), parp->ar_prtype, pni)) {
33          blkcopy(pae->ae_hwa, SHA(parp), pae->ae_hwlen);
34          pae->ae_ttl = ARP_TIMEOUT;
35      }
```

Arp 欺骗攻击实验

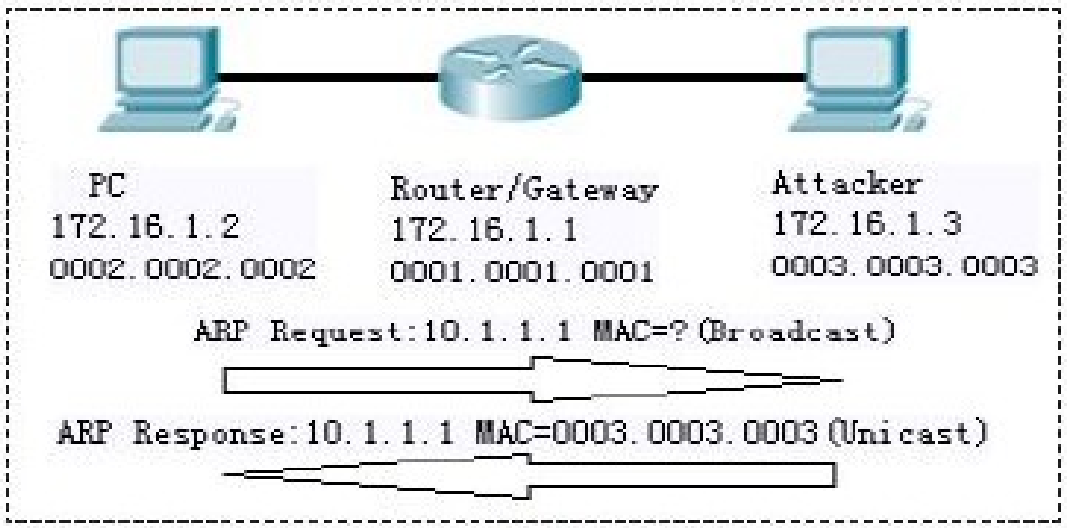


图2