

第4章 ARP：地址解析协议

4.1 引言

本章我们要讨论的问题是只对 TCP/IP 协议簇有意义的 IP 地址。数据链路如以太网或令牌环网都有自己的寻址机制（常常为 48 bit 地址），这是使用数据链路的任何网络层都必须遵从的。一个网络如以太网可以同时被不同的网络层使用。例如，一组使用 TCP/IP 协议的主机和另一组使用某种 PC 网络软件的主机可以共享相同的电缆。

当一台主机把以太网数据帧发送到位于同一局域网上的另一台主机时，是根据 48 bit 的以太网地址来确定目的接口的。设备驱动程序从不检查 IP 数据报中的目的 IP 地址。

地址解析为这两种不同的地址形式提供映射：32 bit 的 IP 地址和数据链路层使用的任何类型的地址。RFC 826 [Plummer 1982] 是 ARP 规范描述文档。

本章及下一章我们要讨论的两种协议如图 4-1 所示：ARP（地址解析协议）和 RARP（逆地址解析协议）。

ARP 为 IP 地址到对应的硬件地址之间提供动态映射。我们之所以用动态这个词是因为这个过程是自动完成的，一般应用程序用户或系统管理员不必关心。

RARP 是被那些没有磁盘驱动器的系统使用（一般是无盘工作站或 X 终端），它需要系统管理员进行手工设置。我们在第 5 章对它进行讨论。

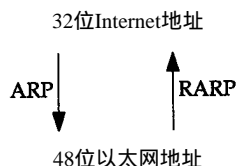


图4-1 地址解析协议：ARP 和 RARP

4.2 一个例子

任何时候我们敲入下面这个形式的命令：

```
% ftp bsdi
```

都会进行以下这些步骤。这些步骤的序号如图 4-2 所示。

- 1) 应用程序 FTP 客户端调用函数 `gethostbyname(3)` 把主机名 (bsdi) 转换成 32 bit 的 IP 地址。这个函数在 DNS（域名系统）中称作解析器，我们将在第 14 章对它进行介绍。这个转换过程或者使用 DNS，或者在较小网络中使用一个静态的主机文件（`/etc/hosts`）。
- 2) FTP 客户端请求 TCP 用得到的 IP 地址建立连接。
- 3) TCP 发送一个连接请求分段到远端的主机，即用上述 IP 地址发送一份 IP 数据报（在第 18 章我们将讨论完成这个过程的细节）。
- 4) 如果目的主机在本地网络上（如以太网、令牌环网或点对点链接的另一端），那么 IP 数据报可以直接送到目的主机上。如果目的主机在一个远程网络上，那么就通过 IP 选路函数来确定位于本地网络上的下一站路由器地址，并让它转发 IP 数据报。在这两种情况下，IP 数据报都是被送到位于本地网络上的一台主机或路由器。
- 5) 假定是一个以太网，那么发送端主机必须把 32 bit 的 IP 地址变换成 48 bit 的以太网地址。

从逻辑Internet地址到对应的物理硬件地址需要进行翻译。这就是 ARP的功能。

ARP本来是用于广播网络的，有许多主机或路由器连在同一个网络上。

- 6) ARP发送一份称作 ARP请求的以太网数据帧给以太网上的每个主机。这个过程称作广播，如图 4-2中的虚线所示。ARP请求数据帧中包含目的主机的 IP地址（主机名为 bsd1），其意思是“如果你是这个 IP地址的拥有者，请回答你的硬件地址。”

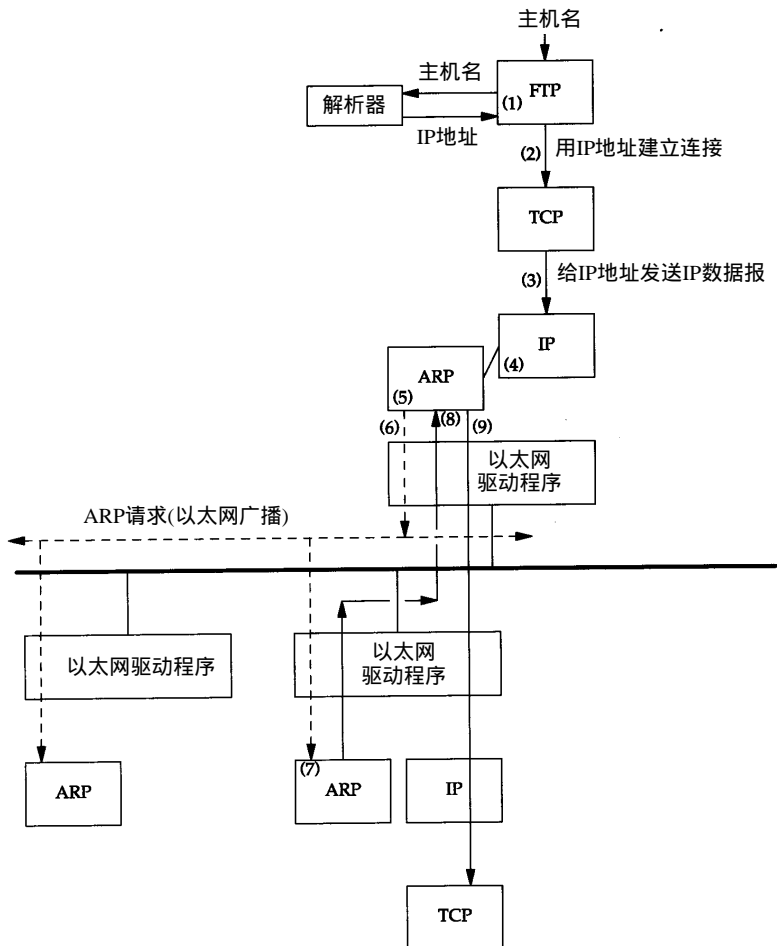


图4-2 当用户输入命令“ftp 主机名”时ARP的操作

- 7) 目的主机的 ARP层收到这份广播报文后，识别出这是发送端在寻问它的 IP地址，于是发送一个 ARP应答。这个 ARP应答包含 IP地址及对应的硬件地址。

- 8) 收到 ARP应答后，使 ARP进行请求—应答交换的 IP数据报现在就可以传送了。

- 9) 发送 IP数据报到目的主机。

在 ARP背后有一个基本概念，那就是网络接口有一个硬件地址（一个 48 bit 的值，标识不同的以太网或令牌环网络接口）。在硬件层次上进行的数据帧交换必须有正确的接口地址。但是，TCP/IP有自己的地址：32 bit 的 IP地址。知道主机的 IP地址并不能让内核发送一帧数据给主机。内核（如以太网驱动程序）必须知道目的端的硬件地址才能发送数据。ARP的功能是在 32 bit 的 IP地址和采用不同网络技术的硬件地址之间提供动态映射。

点对点链路不使用 ARP。当设置这些链路时（一般在引导过程进行），必须告知内核链路

每一端的IP地址。像以太网地址这样的硬件地址并不涉及。

4.3 ARP高速缓存

ARP高效运行的关键是由于每个主机上都有一个 ARP高速缓存。这个高速缓存存放了最近Internet地址到硬件地址之间的映射记录。高速缓存中每一项的生存时间一般为 20分钟, 起始时间从被创建时开始算起。

我们可以用arp(8)命令来检查ARP高速缓存。参数 -a 的意思是显示高速缓存中所有的内容。

```
bsdi %arp -a
sun (140.252.13.33) at 8:0:20:3:f6:42
svr4 (140.252.13.34) at 0:0:c0:c2:9b:26
```

48 bit的以太网地址用6个十六进制的数来表示, 中间以冒号隔开。在 4.8小节我们将讨论 arp命令的其他功能。

4.4 ARP的分组格式

在以太网上解析IP地址时, ARP请求和应答分组的格式如图 4-3所示 (ARP可以用于其他类型的网络, 可以解析 IP地址以外的地址。紧跟着帧类型字段的前四个字段指定了最后四个字段的类型和长度)。

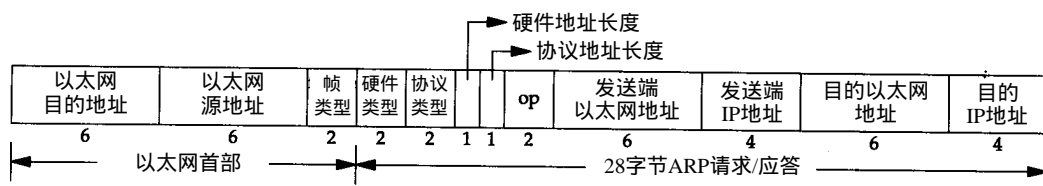


图4-3 用于以太网的ARP请求或应答分组格式

以太网报头中的前两个字段是以太网的源地址和目的地址。目的地址为全 1 的特殊地址是广播地址。电缆上的所有以太网接口都要接收广播的数据帧。

两个字节长的以太网帧类型表示后面数据的类型。对于 ARP请求或应答来说, 该字段的值为0x0806。

形容词hardware(硬件)和protocol(协议)用来描述 ARP分组中的各个字段。例如, 一个 ARP请求分组询问协议地址 (这里是 IP地址) 对应的硬件地址 (这里是以太网地址)。

硬件类型字段表示硬件地址的类型。它的值为 1 即表示以太网地址。协议类型字段表示要映射的协议地址类型。它的值为 0x0800 即表示 IP地址。它的值与包含 IP数据报的以太网数据帧中的类型字段的值相同, 这是有意设计的 (参见图 2-1)。

接下来的两个 1 字节的字段, 硬件地址长度和协议地址长度分别指出硬件地址和协议地址的长度, 以字节为单位。对于以太网上 IP地址的 ARP请求或应答来说, 它们的值分别为 6 和 4。

操作字段指出四种操作类型, 它们是 ARP请求 (值为 1)、ARP应答 (值为 2)、RARP请求 (值为 3) 和 RARP应答 (值为 4) (我们在第 5 章讨论 RARP)。这个字段必需的, 因为 ARP请求和 ARP应答的帧类型字段值是相同的。

接下来的四个字段是发送端的硬件地址 (在本例中是以太网地址)、发送端的协议地址 (IP地址)、目的端的硬件地址和目的端的协议地址。注意, 这里有一些重复信息: 在以太网

的数据帧报头中和ARP请求数据帧中都有发送端的硬件地址。

对于一个ARP请求来说，除目的端硬件地址外的所有其他的字段都有填充值。当系统收到一份目的端为本机的ARP请求报文后，它就把硬件地址填进去，然后用两个目的端地址分别替换两个发送端地址，并把操作字段置为2，最后把它发送回去。

4.5 ARP举例

在本小节中，我们用tcpdump命令来看一看运行像Telnet这样的普通TCP工具软件时ARP会做些什么。附录A包含tcpdump命令的其他细节。

4.5.1 一般的例子

为了看清楚ARP的运作过程，我们执行telnet命令与无效的服务器连接。

```
bsdi % arp -a          检验ARP高速缓存是空的
bsdi % telnet svr4 discard 连接无效的服务器
Trying 140.252.13.34...
Connected to svr4.
Escape character is '^]'.
^]                      键入Ctrl和右括号，使Telnet回到提示符并关闭
telnet> quit
Connection closed.
```

当我们在另一个系统（sun）上运行带有-e选项的tcpdump命令时，显示的是硬件地址（在我们的例子中是48 bit的以太网地址）。

图4-4中的tcpdump的原始输出如附录A中的图A-3所示。由于这是本书第一个tcpdump输出例子，你应该去查看附录中的原始输出，看看我们作了哪些修改。

```
1  0.0                0:0:c0:6f:2d:40 ff:ff:ff:ff:ff:ff arp 60:
   arp who-has svr4 tell bsdi
2  0.002174 (0.0022)  0:0:c0:c2:9b:26 0:0:c0:6f:2d:40 arp 60:
   arp reply svr4 is-at 0:0:c0:c2:9b:26
3  0.002831 (0.0007)  0:0:c0:6f:2d:40 0:0:c0:c2:9b:26 ip 60:
   bsdi.1030 > svr4.discard: S 596459521:596459521(0)
   win 4096 <mss 1024> [tos 0x10]
4  0.007834 (0.0050)  0:0:c0:c2:9b:26 0:0:c0:6f:2d:40 ip 60:
   svr4.discard > bsdi.1030: S 3562228225:3562228225(0)
   ack 596459522 win 4096 <mss 1024>
5  0.009615 (0.0018)  0:0:c0:6f:2d:40 0:0:c0:c2:9b:26 ip 60:
   bsdi.1030 > svr4.discard: . ack 1 win 4096 [tos 0x10]
```

图4-4 TCP连接请求产生的ARP请求和应答

我们删除了tcpdump命令输出的最后四行，因为它们是结束连接的信息（我们将在第18章进行讨论），与这里讨论的内容不相关。

在第1行中，源端主机（bsdi）的硬件地址是0:0:c0:6f:2d:40。目的端主机的硬件地址是ff:ff:ff:ff:ff:ff，这是一个以太网广播地址。电缆上的每个以太网接口都要接收这个数据帧并对它进行处理，如图4-2所示。

第1行中紧接着的一个输出字段是arp，表明帧类型字段的值是0x0806，说明此数据帧是一个ARP请求或回答。

在每行中，单词arp或ip后面的值60指的是以太网数据帧的长度。由于ARP请求或回答

的数据帧长都是42字节(28字节的ARP数据, 14字节的以太网帧头), 因此, 每一帧都必须加入填充字符以达到以太网的最小长度要求: 60字节。

请参见图1-7, 这个最小长度60字节包含14字节的以太网帧头, 但是不包括4个字节的以太网帧尾。有一些书把最小长度定为64字节, 它包括以太网的帧尾。我们在图1-7中把最小长度定为46字节, 是有意不包括14字节的帧首部, 因为对应的最大长度(1500字节)指的是MTU——最大传输单元(见图2-5)。我们使用MTU经常是因为它对IP数据报的长度进行限制, 但一般与最小长度无关。大多数的设备驱动程序或接口卡自动地用填充字符把以太网数据帧充满到最小长度。第3, 4和5行中的IP数据报(包含TCP段)的长度都比最小长度短, 因此都必须填充到60字节。

第1行中的下一个输出字段 `arp who-ha` 表示作为ARP请求的这个数据帧中, 目的IP地址是 `svr4` 的地址, 发送端的IP地址是 `bsd1` 的地址。 `tcpdump` 打印出主机名对应的默认IP地址(在4.7节中, 我们将用 `-n` 选项来查看ARP请求中真正的IP地址。)

从第2行中可以看到, 尽管ARP请求是广播的, 但是ARP应答的目的地址却是 `bsd1` (`0:0:c0:6f:2d:40`)。ARP应答是直接送到请求端主机的, 而是广播的。

`tcpdump` 打印出 `arp repl` 的字样, 同时打印出响应者的主机名和硬件地址。

第3行是第一个请求建立连接的TCP段。它的目的硬件地址是目的主机(`svr4`)。我们将在第18章讨论这个段的细节内容。

在每一行中, 行号后面的数字表示 `tcpdump` 收到分组的时间(以秒为单位)。除第1行外, 其他每行在括号中还包含了与上一行的时间差异(以秒为单位)。从这个图可以看出, 发送ARP请求与收到ARP回答之间的延时是2.2 ms。而在0.7 ms之后发出第一段TCP报文。在本例中, 用ARP进行动态地址解析的时间小于3 ms。

最后需要指出的一点, 在 `tcpdump` 命令输出中, 我们没有看到 `svr4` 在发出第一段TCP报文(第4行)之前发出的ARP请求。这是因为可能在 `svr4` 的ARP高速缓存中已经有 `bsd1` 的表项。一般情况下, 当系统收到ARP请求或发送ARP应答时, 都要把请求端的硬件地址和IP地址存入ARP高速缓存。在逻辑上可以假设, 如果请求端要发送IP数据报, 那么数据报的接收端将很可能会发送一个应答。

4.5.2 对不存在主机的ARP请求

如果查询的主机已关机或不存在会发生什么情况呢? 为此我们指定一个并不存在的Internet地址——根据网络号和子网号所对应的网络确实存在, 但是并不存在所指定的主机号。从图3-10可以看出, 主机号从36到62的主机并不存在(主机号为63是广播地址)。这里, 我们用主机号36来举例。

这次是Telnet的一个地址, 而不是主机名

```
bsd1 % date ; telnet 140.252.13.36 ; date
Sat Jan 30 06:46:33 MST 1993
Trying 140.252.13.36...
telnet: Unable to connect to remote host: Connection timed out
Sat Jan 30 06:47:49 MST 1993      在前一个日期输出后76秒

bsd1 % arp -a                    检查ARP高速缓存
? (140.252.13.36) at (incomplete)
```

`tcpdump` 命令的输出如图4-5所示。

```
1  0.0                arp who-has 140.252.13.36 tell bsdi
2  5.509069 ( 5.5091)  arp who-has 140.252.13.36 tell bsdi
3  29.509745 (24.0007)  arp who-has 140.252.13.36 tell bsdi
```

图4-5 对不存在主机的ARP请求

这一次，我们没有用 `-e` 选项，因为已经知道 ARP 请求是在网上广播的。

令人感兴趣的是看到多次进行 ARP 请求：第 1 次请求发生后 5.5 秒进行第 2 次请求，在 24 秒之后又进行第 3 次请求（在第 21 章我们将看到 TCP 的超时和重发算法的细节）。`tcpdump` 命令输出的超时限制为 29.5 秒。但是，在 `telnet` 命令使用前后分别用 `date` 命令检查时间，可以发现 Telnet 客户端的连接请求似乎在大约 75 秒后才放弃。事实上，我们在后面将看到，大多数的 BSD 实现把完成 TCP 连接请求的时间限制设置为 75 秒。

在第 18 章中，当我们看到建立连接的 TCP 报文段序列时，会发现 ARP 请求对应于 TCP 试图发送的初始 TCPSYN（同步）段。

注意，在线路上始终看不到 TCP 的报文段。我们能看到的是 ARP 请求。直到 ARP 回答返回时，TCP 报文段才可以被发送，因为硬件地址到这时才可能知道。如果我们用过滤模式运行 `tcpdump` 命令，只查看 TCP 数据，那么将没有任何输出。

4.5.3 ARP 高速缓存超时设置

在 ARP 高速缓存中的表项一般都要设置超时值（在 4.8 小节中，我们将看到管理员可以用 `arp` 命令把地址放入高速缓存中而不设置超时值）。从伯克利系统演变而来的系统一般对完整的表项设置超时值为 20 分钟，而对不完整的表项设置超时值为 3 分钟（在前面的例子中我们已见过一个不完整的表项，即在以太网上对一个不存在的主机发出 ARP 请求。）当这些表项再次使用时，这些实现一般都把超时值重新设为 20 分钟。

Host Requirements RFC 表明即使表项正在使用时，超时值也应该启动，但是大多数从伯克利系统演变而来的系统没有这样做——它们每次都是在访问表项时重设超时值。

4.6 ARP 代理

如果 ARP 请求是从一个网络的主机发往另一个网络上的主机，那么连接这两个网络的路由器就可以回答该请求，这个过程称作委托 ARP 或 ARP 代理 (Proxy ARP)。这样可以欺骗发起 ARP 请求的发送端，使它误以为路由器就是目的主机，而事实上目的主机是在路由器的“另一边”。路由器的功能相当于目的主机的代理，把分组从其他主机转发给它。

举例是说明 ARP 代理的最好方法。如图 3-10 所示，系统 `sun` 与两个以太网相连。但是，我们也指出过，事实上并不是这样，请把它与封内图 1 进行比较。在 `sun` 和子网 140.252.1 之间实际存在一个路由器，就是这个具有 ARP 代理功能的路由器使得 `sun` 就好像在子网 140.252.1 上一样。具体安置如图 4-6 所示，路由器 Telebit NetBlazer，取名为 `netb`，在子网和主机 `sun` 之间。

当子网 140.252.1（称作 `gemin`i）上的其他主机有一份 IP 数据报要传给地址为 140.252.1.29 的 `sun` 时，`gemin`i 比较网络号（140.252）和子网号（1），因为它们都是相同的，因而在图 4-6 上面的以太网中发送 IP 地址 140.252.1.29 的 ARP 请求。路由器 `netb` 识别出该 IP 地址属于它的一个拨号主机，于是把它的以太网接口地址 140.252.1 作为硬件地址来回答。主机 `gemin`i 通过以太网发送 IP 数据报到 `netb`，`netb` 通过拨号 SLIP 链路把数据报转发到 `sun`。这个过程对于所有

140.252.1子网上的主机来说都是透明的, 主机sun实际上是在路由器netb后面进行配置的。

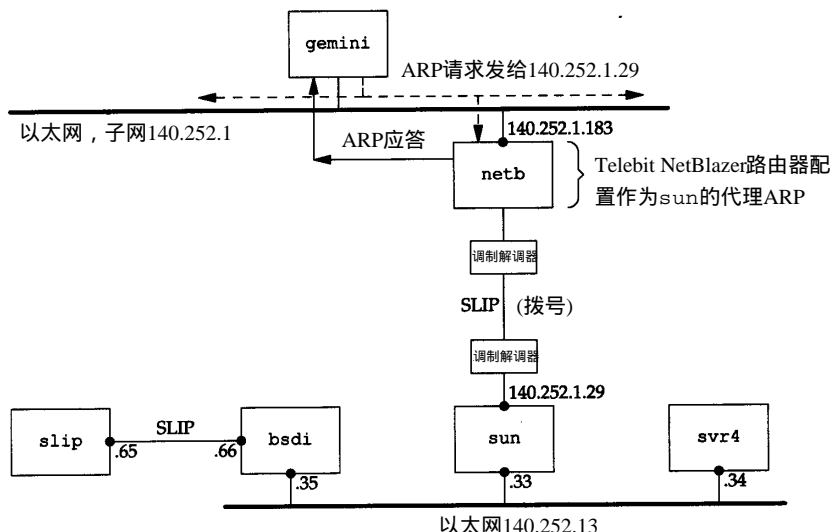


图4-6 ARP代理的例子

如果在主机gemini上执行arp命令, 经过与主机sun通信以后, 我们发现在同一个子网140.252.1上的netb和sun的IP地址映射的硬件地址是相同的。这通常是使用委托ARP的线索。

```
gemini %arp -a
```

这里是子网140.252.1上其他主机的输出行

```
netb (140.252.1.183) at 0:80:ad:3:6a:80
sun (140.252.1.29) at 0:80:ad:3:6a:80
```

图4-6中的另一个需要解释的细节是在路由器netb的下方(SLIP链路)显然缺少一个IP地址。为什么在拨号SLIP链路的两端只拥有一个IP地址, 而在bsd和slip之间的两端却分别有一个IP地址? 在3.8小节我们已经指出, 用ifconfig命令可以显示拨号SLIP链路的目的地址, 它是140.252.1.183。NetBlazer不需要知道拨号SLIP链路每一端的IP地址(这样做会用更多的IP地址)。相反, 它通过分组到达的串行线路接口来确定发送分组的拨号主机, 因此对于连接到路由器的每个拨号主机不需要用唯一的IP地址。所有的拨号主机使用同一个IP地址140.252.1.183作为SLIP链路的目的地址。

ARP代理可以把数据报传送到路由器sun上, 但是子网140.252.13上的其他主机是如何处理的呢? 选路必须使数据报能到达其他主机。这里需要特殊处理, 选路表中的表项必须在网络140.252的某个地方制定, 使所有数据报的目的端要么是子网140.252.13, 要么是子网上的某个主机, 这样都指向路由器netb。而路由器netb知道如何把数据报传到最终的目的端, 即通过路由器sun。

ARP代理也称作混合ARP(promiscuous ARP)或ARP出租(ARP hack)。这些名字来自于ARP代理的其他用途: 通过两个物理网络之间的路由器可以互相隐藏物理网络。在这种情况下, 两个物理网络可以使用相同的网络号, 只要把中间的路由器设置成一个ARP代理, 以响应一个网络到另一个网络主机的ARP请求。这种技术在过去用来隐藏一组在不同物理电缆上运行旧版TCP/IP的主机。分开这些旧主机有两个共同的理由, 其一是它们不能处理子网划分, 其二是它们使用旧的广播地址(所有比特值为0的主机号, 而不是目前使用的所有比特值为1

的主机号)。

4.7 免费ARP

我们可以看到的另一个ARP特性称作免费ARP (gratuitous ARP)。它是指主机发送ARP查找自己的IP地址。通常，它发生在系统引导期间进行接口配置的时候。

在互联网中，如果我们引导主机 `bsdi` 并在主机 `sun` 上运行 `tcpdump` 命令，可以看到如图4-7所示的分组。

```
1 0.0 0:0:c0:6f:2d:40 ff:ff:ff:ff:ff:ff arp 60:
arp who-has 140.252.13.35 tell 140.252.13.35
```

图4-7 免费ARP的例子

(我们用 `-n` 选项运行 `tcpdump` 命令，打印出点分十进制的地址，而不是主机名)。对于ARP请求中的各字段来说，发送端的协议地址和目的端的协议地址是一致的：即主机 `bsdi` 的地址 `140.252.13.35`。另外，以太网报头中的源地址 `0:0:c0:6f:2d:40`，正如 `tcpdump` 命令显示的那样，等于发送端的硬件地址（见图4-4）。

免费ARP可以有两个方面的作用：

1) 一个主机可以通过它来确定另一个主机是否设置了相同的IP地址。主机 `bsdi` 并不希望对此请求有一个回答。但是，如果收到一个回答，那么就会在终端日志上产生一个错误消息“以太网地址：`a:b:c:d:e:f` 发送来重复的IP地址”。这样就可以警告系统管理员，某个系统有不正确的设置。

2) 如果发送免费ARP的主机正好改变了硬件地址（很可能是主机关机了，并换了一块接口卡，然后重新启动），那么这个分组就可以使其他主机高速缓存中旧的硬件地址进行相应的更新。一个比较著名的ARP协议事实 [Plummer 1982] 是，如果主机收到某个IP地址的ARP请求，而且它已经在接收者的高速缓存中，那么就要用ARP请求中的发送端硬件地址（如以太网地址）对高速缓存中相应的内容进行更新。主机接收到任何ARP请求都要完成这个操作（ARP请求是在网上广播的，因此每次发送ARP请求时网络上的所有主机都要这样做）。

文献 [Bhide、Elnozahy 和 Morgan 1991] 中有一个应用例子，通过发送含有备份硬件地址和故障服务器的IP地址的免费ARP请求，使得备份文件服务器可以顺利地接替故障服务器进行工作。这使得所有目的地为故障服务器的报文都被送到备份服务器那里，客户程序不用关心原来的服务器是否出了故障。

不幸的是，作者却反对这个做法，因为这取决于所有不同类型的客户端都要有正确的ARP协议实现。他们显然碰到过客户端的ARP协议实现与规范不一致的情况。

通过检查作者所在子网上的所有系统可以发现，SunOS 4.1.3 和 4.4BSD 在引导时都发送免费ARP，但是SVR4却没有这样做。

4.8 arp命令

我们已经用过这个命令及参数 `-a` 来显示ARP高速缓存中的所有内容。这里介绍其他参数的功能。

超级用户可以用选项 `-d` 来删除ARP高速缓存中的某一项内容（这个命令格式可以在运行

一些例子之前使用, 以让我们看清楚 ARP 的交换过程)。

另外, 可以通过选项 `-s` 来增加高速缓存中的内容。这个参数需要主机名和以太网地址: 对应于主机名的 IP 地址和以太网地址被增加到高速缓存中。新增加的内容是永久性的 (比如, 它没有超时值), 除非在命令行的末尾附上关键字 `temp`。

位于命令行末尾的关键字 `pub` 和 `-s` 选项一起, 可以使系统起着主机 ARP 代理的作用。系统将回答与主机名对应的 IP 地址的 ARP 请求, 并以指定的以太网地址作为应答。如果广播的地址是系统本身, 那么系统就为指定的主机名起着委托 ARP 代理的作用。

4.9 小结

在大多数的 TCP/IP 实现中, ARP 是一个基础协议, 但是它的运行对于应用程序或系统管理员来说一般是透明的。ARP 高速缓存在它的运行过程中非常关键, 我们可以用 `arp` 命令对高速缓存进行检查和操作。高速缓存中的每一项内容都有一个定时器, 根据它来删除不完整和完整的表项。`arp` 命令可以显示和修改 ARP 高速缓存中的内容。

我们介绍了 ARP 的一般操作, 同时也介绍了一些特殊的功能: 委托 ARP (当路由器对来自于另一个路由器接口的 ARP 请求进行应答时) 和免费 ARP (发送自己 IP 地址的 ARP 请求, 一般发生在引导过程中)。

习题

- 4.1 当输入命令以生成类似图 4-4 那样的输出时, 发现本地 ARP 快速缓存为空以后, 输入命令
`bsd1 % rsh svr4 arp -a`
如果发现目的主机上的 ARP 快速缓存也是空的, 那将发生什么情况? (该命令将在 `svr4` 主机上运行 `arp -a` 命令)。
- 4.2 请描述如何判断一个给定主机是否能正确处理接收到的非必要的 ARP 请求的方法。
- 4.3 由于发送一个数据包后 ARP 将等待响应, 因此 4.2 节所描述的步骤 7 可能会持续一段时间。你认为 ARP 将如何处理在这期间收到相同目的 IP 地址发来的多个数据包?
- 4.4 在 4.5 节的最后, 我们指出 Host Requirements RFC 和伯克利派生系统在处理活动 ARP 表目的超时时存在差异。那么如果我们在一个由伯克利派生系统的客户端上, 试图与一个正在更换以太网卡而处于关机状态的服务器主机联系, 这时会发生什么情况? 如果服务器在引导过程中广播一份免费 ARP, 这种情况是否会发生变化?