

组织：中国互动出版网 (<http://www.china-pub.com/>)

RFC 文档中文翻译计划 (<http://www.china-pub.com/compters/emook/aboutemook.htm>)

E-mail: ouyang@china-pub.com

译者：沈进 (simon_shen shen_jin@263.net)

译文发布时间：2001-8-14

版权：本中文翻译文档版权归中国互动出版网所有。可以用于非商业用途自由转载，但必须保留本文档的翻译及版权信息。

Network Working Group
Request for Comments: 903

Finlayson, Mann, Mogul, Theimer
Stanford University
June 1984

反向地址转换协议

(RFC903——A Reverse Address Resolution Protocol)

Ross Finlayson, Timothy Mann, Jeffrey Mogul, Marvin Theimer
Computer Science Department
Stanford University
June 1984

本备忘录状态：

本 RFC 文档讲述了当工作站只知道硬件地址（例如：它们的物理网络地址）时，动态发现协议地址（例如：Internet 地址）的一种方法。

本 RFC 文档讲述了 ARPA 网络社区的一种建议协议，需要讨论和建议去加强。

目录

1.介绍.....	1
2.设计考虑.....	2
3.推荐协议.....	2
4.参考.....	3
附录 A. 4.2BSD Unix 下的两个实现例子	3

1.介绍

有些网络主机，例如无盘工作站，在启动的时候通常不知道协议地址，它们只知道硬件接口地址。为了使用象 IP 这样的高层协议进行通讯，它们必须从外部来发现它们的协议地址。我们的问题是没有标准机制来做这些。

Plummer 的“地址转换协议”(ARP) [1]被设计用来解决一个相关的问题：给定主机的协议地址得到它的硬件地址。这篇 RFC 提出了“反向地址转换协议”。和 ARP 一样，我们假设一种广播介质，例如以太网。

2. 设计考虑

以下的考虑指导我们对 RARP 协议的设计。

A. ARP 和 RARP 是不同的操作。ARP 假设每一个主机都知道它的硬件地址和协议地址间的映射。一个小缓存存放了收集到的关于其他主机的信息。所有的主机在状态上是平等的，没有客户机和服务器的区别。

另一方面，RARP 需要一个或者更多的服务器主机来维护一个存有从硬件地址到协议地址的映射的数据库，并对客户机的请求作出应答。

B. 前面提到 RARP 需要维护大数据库的服务器主机，这并不是所希望的，在某些情况下不可能在操作系统的内核中维护这样一个数据库。因此，大多数实现需要和内核外的程序做某种形式的互操作。

C. 实现的简单和对已存在主机软件影响最小是重要的，设计一个需要改变每一个主机的软件（而不管其是否参与）的协议是错误的。

D. 为了最小化开发成本和费用，希望考虑和已有软件共享代码的可能性。

3. 推荐协议

我们建议 RARP 作为数据链路层单独的协议。例如，如果介质使用以太网，RARP 包和 ARP 包将有不同的以太网类型。这意味着 ARP 和 RARP 是两种不同的操作，所有的主机并不平等地支持它们。对已存在系统的影响也最小，已有的 ARP 服务器不会被 ARP 包弄糊涂。它把 RARP 看作一个能把硬件地址映射到任何高层协议地址的常用工具。

这个方法使 RARP 客户端主机的实现最简单，但同时也使得 RARP 服务器端主机的实现很困难。然而这种困难是没办法的，从附录 A 中描述的 4.2BSD Unix 下两种可能的实现就可以看出。

RARP 使用和 ARP 相同的包格式。

ar\$hrd (硬件地址空间) 16 比特

ar\$pro (协议地址空间) 16 比特

ar\$hln (硬件地址长度) 8 比特

ar\$pln (协议地址长度) 8 比特

ar\$op (操作码) 16 比特

ar\$sha (源硬件地址) n 比特, n 来自 ar\$hln 字段

ar\$spa (源协议地址) m 比特, m 来自 ar\$pln 字段

ar\$tha (目的硬件地址) n 比特

ar\$tpa (目的协议地址) m 比特

ar\$hrd、ar\$pro、ar\$hln 和 ar\$pln 与 ARP 相同 (见[1])。

例如，假设硬件地址是 48 比特以太网地址，协议地址是 32 比特 Internet 地址，我们希望决定对应已知的以太网地址的 Internet 地址，那么在每个 RARP 包中，ar\$hrd = 1 (以太网)，ar\$pro = 2048 (IP 包的以太网类型)，ar\$hln = 6，ar\$pln = 4。

有两个操作码：3（请求）和 4（应答）。1 或 2 在[1]中有相同的意义，带有这样操作码的包被当作 ARP 包通过。带有其它操作码的包并没有定义。和 ARP 一样，RARP 没有“找不到”和“错误”的包，因为许多 RARP 服务器可自由地应答请求。如果经过一段时间后，没有收到应答，RARP 请求的发送者将超时。

RARP 包中 ar\$sha、ar\$spa、ar\$tha 和 ar\$tpa 字段的解释如下：

当操作码是 3（请求）：

ar\$sha 是发送者的硬件地址

ar\$spa 未定义

ar\$tha 是目的硬件地址

当发送者希望知道自己的协议地址的情况下，和 ar\$sha 一样将是发送者的硬件地址。

ar\$tpa 未定义

当操作码是 4（应答）：

ar\$sha 是响应者的硬件地址（应答包的发送者）

ar\$spa 是响应者的协议地址（见下面的注意）

ar\$tha 是目的硬件地址，必须和请求包中得到的相同

ar\$tpa 是目的协议地址，即需要得到的地址。

注意：在操作码为 4 的包中 ar\$spa 字段填响应者的协议地址，只是为了方便。例如，系统同时使用 ARP 和 RARP，有效的协议-硬件对（ar\$spa, ar\$sha）会减少以后发 ARP 请求的需要。

4. 参考

- [1] Plummer, D., “An Ethernet Address Resolution Protocol”, RFC 826, MIT-LCS, November 1982.

附录 A. 4.2BSD Unix 下的两个实现例子

下面的实现描述概括了 4.2BSD Unix 下实现 RARP 服务器的两种不同方法。

- A. 提供在内核外访问数据链路层包。RARP 服务器完全在内核外实现，只在收发 RARP 包时和内核进行互操作。内核被修改成能提供接口来访问这些包，目前 4.2 内核只允许访问 IP 包。CMU 的“包过滤”伪驱动是提供这种功能的现有机制。它在 CMU 和斯坦福实现“用户级”网络服务器排序上，使用的很成功。
- B. 在内核里维护一个数据库表项的缓存。整个 RARP 服务器数据库通过一个用户进程在内核外维护。RARP 服务器本身直接在内核中实现，并为应答维护一个小的数据表项缓存。这个缓存和传输 ARP 的相同。这个缓存通过两个新的输入输出控制从实际的 RARP 数据库得到（这像 SIOCIFADDR，因为他们不直接和具体的套接字相关）。一种是：“睡眠直到需要进行地址转换，然后把请求直接输出到用户进程”；另一种是：“把地址转换输入内核表”。因此，当内核不能在缓存中发现一个表项的时候，把这个请求放到队列中，然后调用 wakeup()。第一个输入输出控制的实现是 sleep()，从队列中取出第一项返回给用户进程，因为内核不能在中断级等待直到用户进程回应，它或者放弃（假设请求主机一定时间后会重传请求包），或者如果第二个输入输出控制把请求拷贝到内核，在那个时候应答。