LINUX下RARP SERVER的一种实现*

李建刚** 卢显良
(电子科技大学计算机学院 成都 610054)

【摘要】 讨论了LINUX下对链路层数据包进行收发的方案，介绍了在一些专用系统中用LINUX来实现RARP SERVER的一种具体做法：设计一个虚设备驱动程序来实现了RARP报文的收发，而自定义的RARP配置信息则由PostgreSQL数据库进行管理，通过应用层的查询程序可实现对信息的检索。

关 键 词 反向地址解析协议； PostgreSQL数据库； 虚设备驱动程序； 套接字缓冲区
中图分类号 TP316.8； TP393.02

反向地址解析协议(RARP)通常用来给没有配置IP地址等基本信息的设备提供配置信息，如无盘工作站使用RARP报文获得自己的IP地址。一些网络上的嵌入式系统，由于操作系统和应用程序都已经作了固化，所以也可通过RARP报文或修改过的RARP报文在系统启动期间获得自己所需的配置信息。目前，LINUX系统中也带有RARP模块，但是该模块只支持标准的RARP报文，而且是将地址信息放在Cache中，无法长期保存。因此，如果希望RARP报文携带自定义信息，则须采取新的方法。

1 RARP SERVER的实现分析

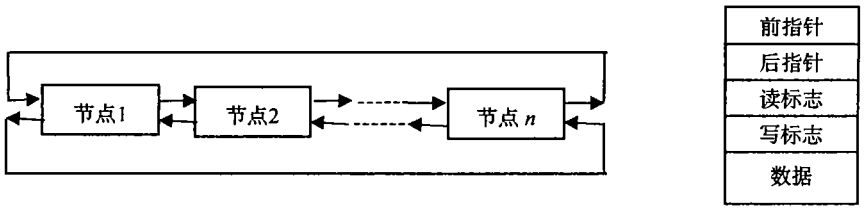
由于RARP请求是在设备启动期间通过广播发出的，而一个子网中的设备通常不会频繁启动，所以RARP SERVER不必将配置数据放在内存缓冲中。通常，客户设备除了IP地址外，还需要一些其他信息，如端口的配置、网关和DNS的地址。为了管理这些信息，可以采用专门的数据库管理系统。如选用LINUX下的PostgreSQL或MySQL，这两种数据库都提供了标准的SQL语言和对C语言的接口。

在一个系统中，RARP SERVER接收客户的RARP请求报文，从数据库中检索出相应信息，同样以RARP响应报文响应。由于LINUX在应用层并不直接提供对数据链路层的写操作，所以RARP响应报文的发送部分必须运行在操作系统的核心空间，而对于RARP请求报文的接收有两种做法。LINUX中提供了一种Packet Capture的共享库(类似于BSD的包过滤BPF)，所以一种做法就是使用pcap方法，将所有的帧包从链路层读上来，然后从帧包的协议字段中识别出RARP报文(协议类型0x8035)。由于这种方法要将所有的帧包送到应用层，故其效率受到很大影响。实验表明，当客户方连续发RARP请求时，将有50%的请求丢失。

2 一种更好的方法

本文提出一种更好的做法，就是让接收部分也运行在核心空间，并维护一定数量的缓冲区，将来不及处理的请求报文放在缓冲区中，该缓冲区的结构如图1所示。这是一个双向链表，表中每个节点除了有指向前后节点的指针外，还带有自己的标志，如读(ReadFlag)、写(WriteFlag)等。由RARP CLIENT和RARP SERVER构成的系统如图2所示。

2000年6月25日收稿
* 电子部生产发展基金资助项目
** 男 28岁 硕士 讲师



(a) 缓冲区结构 (b) 节点结构

图1 缓冲区结构图

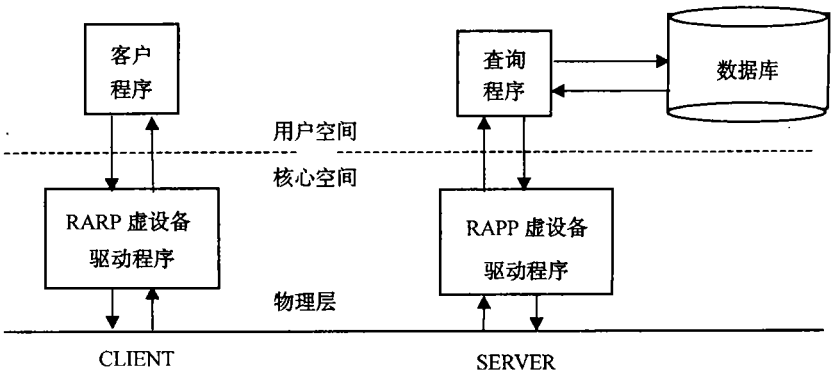


图2 系统结构图

在图2中，将 RARP 报文的收、发两部分做在一起，成为一个虚设备驱动程序，向应用层提供 open，close，read，write 四种操作，同时在 read 实现中提供睡眠机制，当没有请求时，让应用程序睡眠[2]。运行在应用层的查询程序可以使用标准的 read ，write 调用，实现对 RARP 报文的收发。

2.1 接收和发送数据结构[3,4]

在虚设备驱动程序中，对 read 和 write 两种系统调用的实现，都将涉及到内核中的一个数据结构 sk_buff，系统的结构如图3所示，由该结构构成的缓冲区通常称为套接字缓冲区。在 LINUX 中，网络接口设备和网络协议层之间数据的传递都必须使用该套接字缓冲区。

在图3中，对于 protocol 字段可以填上链路层对应的协议类别，如 IP、ARP、RARP 和 AX25 等。为便于处理，这些协议在 LINUX 中被分成16类，每类包含一两种协议。dev 字段指向一个 device 结构，该 device 描述的就是要发送本数据包的网卡设备。而数据包中的实际数据，则由 data 指向。另外，由于 RARP 和 ARP 具有相同的报文头部，两者共享一个相同的指针 nh.arph。对于 sk_buff 更详细的描述，可直接查阅<linux/skbuff.h>文件。

在协议层的实现中，所有的网卡设备都要访问一个称为 backlog 的队列，该队列是一个由 sk_buff 组成的双向链表。当帧包到来时，网卡通过中断处理程序进行接收，并将其加入到 backlog 队列。为了不影响后续帧包的接收，对于 backlog 中包的处理，则是在中断处理程序之后，由一个 bottom half 程序来进行处理。

2.2 Bottom half 程序

Bottom half 程序的功能是完成帧包接收之后的工作。在这个程序中，要对所收到包的类型进行识别，并调用相应的包处理程序。在 LINUX 协议层中，维护了一个包类型的 hash 列表，该列表是一个有16个元素的数组，每个元素都是一个指针，指向一个 packet_type 数据结构，如图4所示。该结构描述了需要 Bottom half 程序处理的包的类型，如 IP、RARP、IPX 等，相应的处理程序以及对应的设备等。因此，可以将自己的 RARP 接收程序加入到该 hash 列表中，每当 bottom half 程序从 backlog 队列中发现一个 RARP 包时，通过查找 hash 列表，便会调用 RARP 处理程序。在

处理程序中，就可以将这些请求写入到图1所示的接收缓冲区中。在这里，接收缓冲区不必太大，因为查询程序通过 read 调用在不断的读该缓冲区。

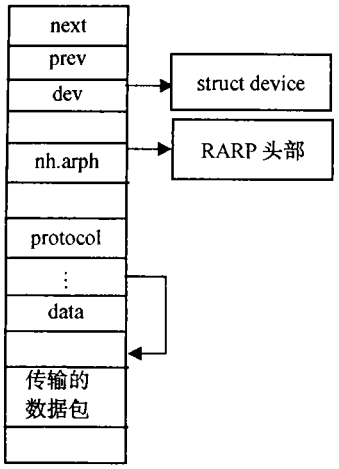


图3　sk_buff 数据结构

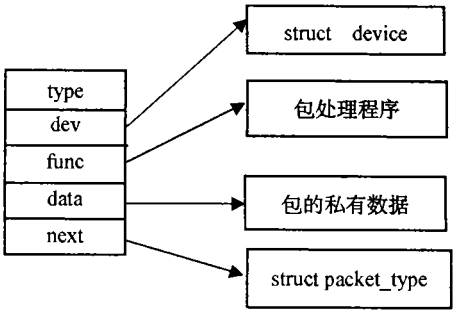


图4　packet_type 数据结构

2.3 RARP 包的发送

RARP 响应报文的发送要比接收简单得多，无须缓冲，也不需要分包。每个网卡设备驱动程序都向操作系统注册了自己的发送程序，当调用内核函数 dev_queue_xmit 时，便会转入相应网卡驱动程序的发送例程。所以，首先可用响应信息构造一个 sk_buff 数据结构，然后调用 dev_queue_xmit 就可直接将报文从网卡发送出去。

3 结 束 语

经过测试，采用本文所述的利用 RARP 虚设备驱动程序进行报文收发和利用数据库对配置信息进行管理的方案后，即使在 RARP 请求报文连续到达的情况下，也能作到100%的响应。

参 考 文 献

1　Stevens W.Richard. unix network programming. New York: Prentice Hall Inc, 1998

2　彭寿全, 宋　杰, 严海锦. UNIX 设备驱动程序的剖析与实例. 成都: 电子科技大学学报, 1998, 27(1): 78~82

3　Beck Michael. Linux Kernel Internals, 2ndEditioned. New York: Addison-Wesley Pub Co, 1998

4　Rubini Alessandro. Linux Device Drivers. CA: O'Reilly&Associates Inc, 1999

Implement of RARP Server on Linux

Li Jiangang　　Lu Xianliang

(College of Computer Science and Engineering, UEST of China　Chengdu　610054)

Abstract　In this paper, the method of designing a virtual device driver to receive RARP requests and send RARP responses is discussed. To manage the configure information, a database in PostgreSQL DBMS is established. Information can be obtained by query procedure and delivered to virtual device driver.

Key words　reverse address resolution protocol;　database;　virtual device driver;　backlog