基于 Linux 嵌入式系统按需路由协议架构设计与实现*

谢毅勇[1],谢维波[2]

(1.华侨大学 计算机科学与技术学院,福建 厦门 361021;
2.华侨大学 厦门软件园嵌入式技术开放实验室,福建 厦门 361008)

摘 要:针对在 Linux 操作系统原有的路由体系结构上实现按需路由的制约问题,分析了 Linux 操作系统路由体系结构特点以及实现按需路由的难点,提出了一种适合按需路由的通用路由体系结构,并基于 Linux 系统实现了按需路由之一的 Aodv 路由协议的嵌入式实现。结果表明了此路由体系结构很好地解决了 Linux 传统的路由体系结构瓶颈。

关键词:按需路由;嵌入式系统;路由体系结构;Ad hoc

中图分类号:TP399 **文献标识码**:A **文章编号**:1674-7720(2011)11-0056-04

Design and implement of the on-demand routing protocol architecture based on the Linux embedded system

Xie Yiyong[1], Xie Weibo[2]

(1.College of Computer Science & Technology, Huaqiao University, Xiamen 361021, China;
2.Open Laboratory of Embedded Technology, Xiamen Software Park, Huaqiao University, Xiamen 361008, China)

Abstract: For the problems to implementation on-demand routing in Linux operating system routing architecture, this paper analyzes the Linux operating system routing architecture features and its difficulty to achieve on-demand routing. It proposes a common route for on-demand routing architecture, and implements Aodv routing protocol which is one of the on-demand routing based on Linux system. The results show that this routing architecture is a good solution to the bottleneck of traditional routing architecture.

Key words: on-demand routing; embedded system; routing architecture; Ad hoc

Ad hoc 网络[1]是一种特殊的对等式网络,它不需要固定的通信设施支持即可快速建立无线网络,其最初的动机是满足战场生存的军事需求。由于其自组能力、低成本、工作寿命长等特点,Ad hoc 网络逐渐进入了民用和商用领域,遗憾的是发展较为缓慢。其主要原因之一就是 Ad hoc 路由协议通常采用新的路由模式,并不支持目前操作系统路由体系结构。若没有通用的操作系统路由体系结构解决方案,路由算法研究员就得被迫进行低级系统程序设计,而且往往会无计划地更改系统内部,以获得更多功能的 Ad hoc 路由要求,这容易导致系统不稳定。许多研究人员为了回避这种低级系统程序设计,大多从事基于仿真路由算法工作[2-3]而很少真正在产品或嵌入式上实现,所以实际应用价值不大。为了加速 Ad hoc 网络民用商用产品发展,设计一个 Ad hoc 通用嵌入式系统路由体系结构迫在眉睫,Ad hoc 网络和嵌入式系统相结合是必然的。在嵌入式系统下实现无线自组网才有实际应用意义。

针对上述问题,本文首先分析了 Linux 操作系统的原有工作方式以及在此体系结构上实现按需路由协议的问题,并给出了合理的解决方案。提出一种基于 Linux 操作系统实现通用的按需路由协议[4]体系结构,使 Ad hoc 研究员易于在此路由体系结构上进行路由算法编程,而不用进行底层内核系统编程。最后,本文根据此方案实现 Aodv 路由协议并应用在实际的网络中,从而验证方案的性能及可行性。

1 Linux 系统现有路由体系结构

目前的主流网络体系结构一般将路由功能分为两部分:分组转发功能模块和分组寻路功能模块[5]。分组

*基金项目:厦门市重点产学研项目(厦经技[2009]233-03);福建省自然科学基金项目(2010J01334)

转发是指根据内核路由表(转发表)的信息,将需要发送的数据分组,通过指定的网络接口发送到下一跳的节点。分组寻路是指建立转发表的过程,实现路由协议的逻辑计算,根据与其他主机交换的路由信息,计算出到其他节点的正确路由。转发表含有充分的信息来完成转发使命,而路由表中包含路由算法用于发现和维护路由的信息。

在通用的嵌入式开源操作系统之一的 Linux 操作系统的体系结构中,分组转发功能模块的转发表是在内核空间实现的,在查找内核转发表的时候,根据转发表项的最佳匹配原则进行转发。如果找不到匹配的转发表项,则按缺省路由发送,一般是将网关作为缺省路由的下一跳节点。如果缺省路由又不存在,操作系统则将把这个数据分组丢弃。分组寻路功能模块可以在内核空间实现,也可以在用户空间实现。本文采用在用户空间实现[6],如图 1 所示。

图 1 Linux 系统现有路由体系结构

把分组转发功能模块和分组寻路功能模块分开的主要原因是数据分组转发必须尽可能快且有效地遍历转发表,故分组转发功能模块需驻留在内核中。然而分组寻路功能模块涉及复杂的数据分组或内存密集型读写任务,所以应该置于内核外。这种分离的原则使 Linux 操作系统中的路由既有效率又灵活。

2 按需路由协议实现的关键问题

目前大多数通用嵌入式操作系统中的路由体系结构都是按照有线网络路由协议的方式来构造的,这些路由协议为主动路由协议。在这种路由体系结构上,只能实现 Ad hoc 主动路由协议(如 Dsdv 路由协议),而不能实现 Ad hoc 按需路由协议(如 Aodv、Dsr 路由协议)。本文将探讨不能在通用路由体系结构实现 Ad hoc 按需路由协议三个关键的难点。

(1)当发送数据分组的时候,如果不存在从节点到目的地址的路由,则需要发起路由查找过程,但在通常情况下,没有路由匹配,内核将立即删除该数据包。然而,这不是一个按需临时路由想要的情况。在按需路由中,并不是所有的路线被固定安排好,有些路线是必须在需要的时候“发现”。所以在这种情况下,正确的行为应该是请求发现路由,并在路由表更新路由或寻找结束以前,数据分组将不会被发送出去或者丢弃处理。

(2)需要建立一个缓冲区,在按需路由协议等待路由发现的过程中,数据分组需要进入一个缓冲区,该缓冲区设置在内核外,因为可减少内存和 CPU 的负荷。

(3)为了减少查找内核路由表的代价,就需要定期维护更新这路由表,即时把失效的路由清除。为了实现这功能,需要建立一个路由检查表。所有表都有一个相关定时器,当该表的路由被使用时,它的定时器就必须重置。当定时器的时间用完时,则该表项就要从内核路由表中删除。关键问题是内核中没有记录路由使用的机制,所以路由守护进程根本无法知道内核中使用过哪些路由或什么时候使用过路由。

以上分析可知目前通用嵌入式操作系统没有这种机制来支持上述行为,所以通用的嵌入式系统应该为 Ad hoc 网增加以下系统功能:

(1)判断是否是按需路由,如果是,则查看是否有路由请求;

(2)把请求告知 Ad hoc 守护进程;

(3)建立一个高速缓冲区,将需要路由请求的数据分组送到缓冲区;

(4)当路由发现完成或发现失败,将数据分组注入内核中或对其抛弃。

(5)建立路由检查表,且能定期对其维护,以便能更新内核路由表。

3 通用的按需路由体系设计实现方案

3.1 Ad hoc 按需路由协议体系结构

针对本文提出的问题,所设计的按需路由体系如图2 所示。

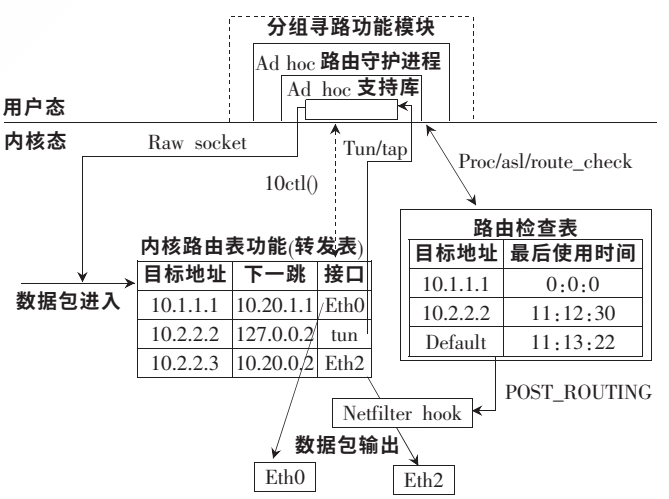

图 2 通用的 ad hoc 按需路由协议体系结构图

3.2 Ad hoc 按需路由协议体系结构组成

通用的 Ad hoc 按需路由协议架构主要组成有:操

作系统原有的内核转发表(本文不对其修改)、一个可加载的内核模块 route_check(即路由检查表)、netfiilter 钩子程序[7]、用户空间的 Ad hoc 支持库 ASL(Ad hoc Support Library)和 Ad hoc 路由守护进程。netfiilter 钩子程序是系统自带的,用来捕捉每一个发送数据分组,记录每条路由的最后使用时间,然后对路由检查表做相应的处理,路由检查表约 800 行代码。其中,ASL 是一个共享库,约 2 500 行代码,用来实现按需路由功能。当 ASL 接到一个延时发送的数据分组,就会告知 Ad hoc 守护进程,以便对数据分组的目的地址进行路由发现。之后它将该数据分组储存在一个临时缓冲区中,等待守护进程返回路由发现的结果。一旦这些处理完成,相应的内核路由表就加入新的元组,数据分组移出缓冲区,重新注入到包转发功能块的队列中。ASL 基本函数如表 1 所示。

表 1 ASL 基本函数

int route_add (addr_t dest, addr_t next_hop, char*dev)	在内核路由表中添加记录
int route_del(addr_t dest);	在内核路由表中删除记录
int open_route_request();	通知 Ad hoc 路由守护进程有路由请求
int route_discovery_done(addr_t dest, int result);	守护进程通过该函数通知内核路由发现已经完成,result 表示路由发现是否是成功的
int query_route_idle_time (addr_t dest);	该函数在被给予一个目的地以后,向内核路由表返回空闲时间记录。即最后一次使用这条路由记录到现在经过的时间
int close_route_request (int fd);	该函数在守护进程不再接收路由请求时调用
int set_route_auto_timeout (addr_t dest, int sec);	守护进程可以用该函数通知内核,删除在规定时间内未使用的路由记录

3.3 Ad hoc 按需路由协议体系结构数据分组传递流程

在实现方案中,为了确认是否有路由请求,利用一个系统不用的隧道设备,Universal TUN/TAP (tun)[8],此设备虚拟了一个网络接口,当数据分组发送的时候,如果目的地址不存在其路由,则这些数据分组将 tun 作为缺省路由的下一跳“接口”。这样就可以通过/dev/net/tun 设备将所有需要路由发现的数据分组传递到用户空间做进一步的处理。

处理工作第一步为调用 open_route_request()函数来打开。当一个新的数据包从/dev/net/tun 中读出时,Ad hoc 守护进程就会利用 read_route_request()函数读取路由请求,之后就可以初始化路由发现。同时,数据分组会在一个临时缓冲区排队。因为缓冲区在用户空间,所以可以有一个较大的缓冲区来存储数据分组,这样即使分组在路由发现延时很大的情况下也不会出现丢失的情况。在路由发现成功完成以后,就将数据分组通过 raw socket 重新注入内核。

为了保持路由表的有效性,内核可加载模块 route_check 在 Netfilter 的 NF_IP_POST_ROUTING 钩子中注册,这样每个外出的数据包都要经过这个模块。该模块只是查看数据包的首部,更新相应的时间戳值,并输出到/proc/asl/route_check 中。ASL 通过 API 中的 query_route_idle_time()告知 Ad hoc 路由守护进程内核路由表的使用情况,这样守护进程就可以检查路由的更新和删除内核路由表失去时效的路由。

route_add()和 route_del()函数使用 ioct()接口加入或者删除内核中的路由。

4 Aodv 按需路由协议架构

通过上述设计架构,在不用改变系统的内核情况下,就可以实现移动自组网按需路由协议在通用的操作系统上的应用。

图 3 是根据 Aodv 基本的路由功能(如 RREQ、RREP、RERR 等路由功能),置入通用分组寻路功能模块使其成为 Aodv 专用的分组寻路功能模块,如图 3 所示。

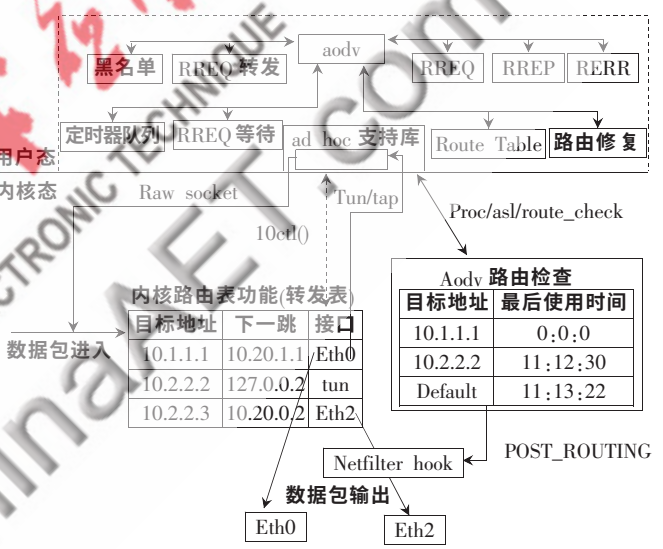

图 3 Aodv 按需路由协议架构

5 方案实施验证分析

实验现场为华侨大学厦门软件园嵌入式技术开放实验室(厦门软件园二期 54 号 402,华侨大学产学研基地)。网络节点由两个 PC 机节点和一个嵌入式系统节点组成。PC 机节点配置是 DELL 工作站 PY597 + ZD1211b 无线网卡,安装 CentOS 5 操作系统(内核版本号为 26.18-128.e15),加载 Aodv 路由协议模块,支持直接或多跳的通信。嵌入式系统节点配置则为致远 MagicARM2410 + Asus WL-167 无线网卡,加载 Aodv 路由协议模块,支持直接或多跳的通信。PC1 的 IP 为 192.168.0.5;PC2 的 IP 为 192.168.0.55;K1 的 IP 为 192.168.0.45。网络拓扑如图 4 所示。

由于测试区域较小,节点 PC1 不需要经过中继节点 K1 就可直接到达节点 PC2。为了测试 Aodv 路由协议的“跳转”功能,采取屏蔽策略,使 PC1 与 PC2 相互屏蔽。

(1)PC1 对 PC2 的屏蔽

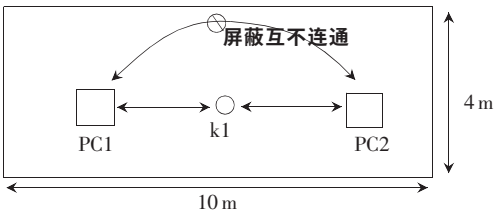

图 4 实验环境

在 PC1 执行：iptables -A INPUT -p ALL -m mac --mac-source 00:02:72:61:ED:4B -j DROP。其中，00:02:72:61:ED:4B 为 PC2 的 Mac 地址，使得 PC1 拒绝 PC2 发送的数据包。

(2)PC2 对 PC1 的屏蔽

在 PC2 执行：iptables -A INPUT -p ALL -m mac --mac-source 00:02:72:61:ED:53 -j DROP。其中，00:02:72:61:ED:53 为 PC1 的 Mac 地址，使得 PC2 拒绝 PC1 发送的数据包。

①K1 节点定期广播传送 HELLO 消息，发现了附近的两个节点 PC1 和 PC2，如图 5 所示。

图 5 K1 节点发现 PC1、PC2 过程

②PC1 节点定期广播传送 HELLO 消息，先发现了附近的 K1 节点，然后透过 K1 发现了 PC2，并对其建立路由。如图 6 所示。

图 6 PC1 与 PC2 路由建立过程

③PC1 节点 Ping PC2 节点，信息返回成功，如图 7 所示。

从 Ping 命令的输出结果中可以看到，数据包所经过的路径是 PC1→K1→PC2→K1→PC1。经过 K1 节点上 Aodv 模块的路由功能将 Ping 数据包转发，使得不能直接通信的节点 PC1 和 PC2 实现了数据传输，具有了路由发现和网络自组的功能。

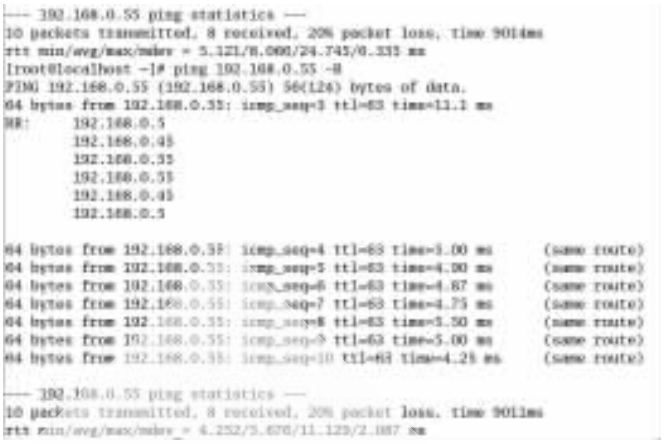

图 7 路由建立成功

本文提出按需路由协议的实现方案，充分考虑了按需路由的特点，针对按需路由在 linux 嵌入式环境下实现的难点，提出了解决方案和设计实现方案，其功能函数具有良好的通用性，该方案为以后研究员验证和比较各种协议在实际网络中的性能提供了良好的基础。

参考文献

[1] HARTENSTEIN H. Topics in ad hoc and sensor networks - A tutorial survey on vehicular ad hoc networks [J]. IEEE Communications Magazine, 2008, 46(6).

[2] GIOVANARDI A, MAZZINI G. Ad hoc routing protocols: emulation vs simulation [C]. 2nd International Symposium on Wireless Communication Systems, 2005: 140-144.

[3] BOUKHALKHAL A, YAGOUBI M B, DJOUDI M, et al. Simulation of mobile ad hoc routing strategies [C]. 4th International Conference on Innovations in Information Technologies, Dubai, United Arab Emirates, 2007: 128-132.

[4] 余旭涛，毕光国.Ad Hoc 网络按需路由协议的改进[J]. 计算机学报，2004，27(6).

[5] PETERSON L L, DAVIE B S. Computer networks: a system approach[M]. Morgan Kaufmann Publishers, 2nd edition, 2000.

[6] RANDHAWA T, RICHAROS J. Implementation of a kernel mode IPv6 AODV routing daemon to improve data throughput [C]. 2005 IEEE International Conference on Communications (ICC 2005), 2005(5): 16-20.

[7] Netfilter/Iptable homepage [EB/OL]. http://www.netfilter.org, 2005-09.

[8] Tun/Tap universal driver [EB/OL].http://vtun.sourceforge.net/tun/, 2005-10.

(收稿日期：2011-01-10)

作者简介：

谢毅勇，男，1986 年生，硕士研究生，主要研究方向：网络嵌入式技术。

谢维波，男，1964 年生，教授，博士，硕士生导师，主要研究方向：嵌入式技术，数字信号处理。