Ad Hoc 路由协议实现技术①

张爱民，马志强，易晓蓉

(总参谋部通信训练基地，宣化 075100)

摘 要：针对目前的 Ad-Hoc 网络的研究大多在模拟环境进行的现状，对 AODV 路由协议实现展开了研究，提出了一种在 Linux 系统中基于 Netfilter 功能框架的 AODV 协议设计与实现方法。搭建了以 PXA310 处理器和 IEEE802.11b 无线网卡为核心的硬件平台，完成了基于 Netfilter 功能框架的 AODV 路由协议的软件框架，实现了路由查询、更新和维护等过程。对 Ad-Hoc 网络协议的实现具有较高的参考和实用价值。

关键词：Ad-Hoc；AODV 路由协议；Linux 系统；Netfilter 功能框架；路由表

Implementation of Ad Hoc Routing Protocol

ZHANG Ai-Min, MA Zhi-Qiang, YI Xiao-Rong

(Communication Training Base of PLA General Staff Headquarters, Xuanhua 075100, China)

Abstract: The current Ad Hoc network researches are mainly based on simulated experiments. This paper is centered upon realization of AODV routing Protocol. A method on AODV routing Protocol of design and implementation based on Netfilter function framework is proposed. The hardware platform is founded based on PXA310 processor and IEEE802.11b wireless IEEE802.11b network card, a design of software based on the Netfilter function framework was accomplished, has achieved the major Processes such as the routing request, routing updating and maintenance, this design will be important worthiness for implementation of AODV routing protocol.

Key words: Ad Hoc; AODV routing Protocol; Linux operation system; Netfilter function framework; routing table

1 引言

Ad Hoc 网络技术首先主要用于军事通信、救灾抢险和野外作业等领域中具有重要应用价值，路由协议是 Ad Hoc 网络研究的热点和难点问题，近些年来，人们已经提出 20 种以上的 Ad Hoc 网络路由协议，如：DSR、TOAR、AODV、DSDV、CGSR 和 ABR 等，但具有原创性的路由协议不过几种。大多数的 Ad Hoc 网络路由协议都是在模拟仿真环境下进行研究的，并没有真正实现。本文借鉴具有代表性的 AODV-UU 的实现方法，阐述嵌入式 Linux 系统中基于 Netfilter 功能框架的 AODV 协议设计与实现方法。

2 AODV路由协议

AODV 一种典型的按需路由协议，它借鉴了 DSDV 的路由维护机制和 DSR 的路由发现机制，是两者结合的产物[1]。当网络的拓扑结构发生变化时，该协议能够快速收敛并且具有断路自我修复的功能，通过使用目的节点序列号，避免了无穷计数的问题，实现了无环路由，该协议为了避免单向链路引起的误操作，引入了黑名单机制，把自己是单向链路的邻居节点放入黑名单。大量研究结果标明，AODV 的性能在大多数应用场合中强于其它已有的路由协议，并且具有计算量小，存储资源消耗小，编程容易实现等优点。是 AODV-UU 是瑞典乌普萨拉大学与爱立信公司基于 GPL 联合发布的一种 Ad Hoc 网络路由协议，目前还在不断地维护和完善，其较新版本是 AODV-UU0.9.5，在 AODV 协议草案 RFC3561 基础上增加了一些其他功能。例如 Hello 消息的增加了一些功能，还提供了

① 收稿时间:2010-10-27;收到修改稿时间:2010-12-18

Internet 网关和多种网络接口支持等。在 AODV-UU 的设计中，利用了内核中 Netfilter 的挂接功能，主要的协议逻辑是建立在用户层的守护进程，所有关注的报文都可以通过用户层守护进程的处理。Netfilter 是 Linux 内核中的一个通用架构，实现了多种网络功能，主要有数据包过滤、状态保持、NAT 以及安全等。

3　AODV路由协议实现软硬件平台搭建

3.1 硬件平台

AODV 路由协议实现软硬件平台采用 PXA310 开发板，PXA310 处理器专门为手持设备、GPS 定位系统、无线手持和其他消费类电子设备而设计。该处理器运行频率 624MHz，内存为 128MB DDR 和 1GB NandFlash，提供了 2 个 RS232 串口，一个 10M/100M 自适应以太网接口，网络接口芯片为 DM9000，2 个 USB2.0 接口，一路 VGA 输出，方便连接电脑显示器一路红外 IrDA 接口，标准 20-Pin JTAG 接口，4 个存储卡接口：SD/MINI SD/T-FLASH CARD/SIM CARD 等，功能强大，完全满足 Ad Hoc 网络手持设备开发要求。

无线网卡在 PXA310 处理器的控制下，对数据包的进行收发，网卡采用的华硕 WL-167g 无线网卡，通过 USB 接口与主板进行通信，网卡和接口的体积小，符合手持设备便携性的要求。该网卡符合 IEEE802.11b 标准，支持 1Mb/s、1Mb/s、1Mb/s、11Mb/s 四种速率，满足该系统对数据传输的要求，在无线网卡的配置上选用 Ad Hoc 模式，该模式的选择对 Ad Hoc 网络的性能影响至关重要。

本平台需要实现数据、话音和图像通信，因此设计了声卡部分和 Camera 部分，提供话筒和扬声器接口。另外，SDRAM、NAND Flash 和电源等模块也是必不可少的部分。以 PXA310 处理器和无线网卡为核心的硬件平台框图如图 1 所示：

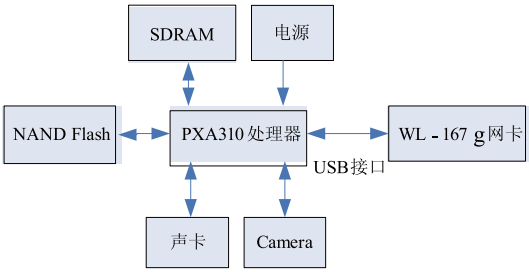

图 1　硬件平台框图

3.2 软件环境

Linux 系统具有源码开放、网络功能强大、支持多种硬件平台、可裁剪、易移植等特点，非常适合于嵌入式计算机系统。本设计采用嵌入式 Linux 操作系统中基于 Netfilter 功能框架来实现 AODV 协议，Ad Hoc 网中的按需路由在现有 Linux 系统内核中并不直接支持。由于缺少必要的系统支持和 API 函数，按需路由协议不得不进行底层的系统编程。

内核是整个操作系统的核心，本设计采用 Linux 2.6.21 内核，通过网络可以下载 Linux 内核源代码，由于 AODV 协议的实现需要用到 netfilter、ip_queue 和 Netlink device 等模块，因此在 Linux 内核的配置过程中需要把以下几项的网络功能编译进内核，具体步骤如下：在内核配置菜单 Networking option 的子菜单中 Netlink device emulation 和 Network packet filtering (replace ipchains)选中，Networking option 的子菜单中 IP: Netfilter Configuration 的配置菜单中 Use repace queueing via NETLINK 选中，这些项目均配置为编译到内核。配置完后保存设置，通过 make zlmage 生成系统映像文件。

AODV 协议的实现过程中无线网卡是重要部件，目前绝大多数的商业无线网卡提供有 Linux 驱动程序，本网卡为驱动 rt73，通过网络下载 RT73_Linux_STA_Drv1.0.4.0.tar.gz，解压后生成 Module 和 WPA Supplicant 两个目录，将目录 Module 中的所有文件都拷贝到内核源码包 drivers/usb/net/rt73 下，并将 rt73 编译到内核，同时网卡的配置上选用 Ad Hoc 模式，使得无线网卡正常工作，编译和配置的具体方法请参考文献[2,3]。

4　AODV路由协议具体实现

4.1 Linux 下 AODV 路由协议实现方案

Linux 操作系统的路由体系结构分为路由功能模块和转发功能模块，路由功能模块作为后台进程在用户层运行，主要完成与其他网络节点的信息交流，采用适当的路由算法建立路由、更新和维护 Linux 内核路由表。转发功能在操作系统内核中实现，该模块根据路由表中的信息，将需要发送的网络数据分组通过正确的网络接口发送到下一跳节点。Linux 系统的路由体系结构如图 2 所示。内核路由表是 Linux 自身的分组路由转发模块工作的基础，为了保持通用性和可移

植性，我们不必对其表项结构进行任何修改。

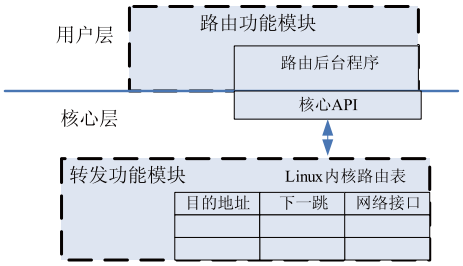

图 2 Linux 系统的路由体系结构

在本设计中，在用户层实现 AODV 协议路由功能模块，路由功能模块根据 AODV 路由协议算法计算出正确的路由后，通过核心层 API 来维护路由表。这样在转发功能模块保持不变的情况下，通过修改路由功能模块，就可以实现 AODV 路由协议。

4.2 Netfilter 对数据包的过滤和处理过程分析

在 AODV 协议的具体实现过程中，Netfilter 在处理数据包流程时起到关键性的作用[4]， Netfilter 由处于 Linux 协议栈中不同点上的五个钩子(hook)函数组成，用户可以在这些位置注册自己定义的操作函数，经过 hook 点的数据分组将执行函数操作，帮助用户完成数据分组的过滤和修改等功能。如图 3 所示。具体作用为：数据包网络入口以后，首先进行对 IP 进行校验后，经过钩子函数 NF_ IP_PIFORWARD 处理后，进入内核路由处理模块，接着判断数据包是发送给本机的还是需要转发的，如果是发给本机的，则该数据包经过钩子函数 NF_IP_LOCAL_IN 传递给上层协议，若该数据包应该被转发则把它被交给 NF_IP_PIFORWARD 处理，转发的数据包经过钩子函数 NF_IP_LOCAL_OUT 处理后，再发送到网络上。本地产生的数据包经过钩子函数 NF_IP_LOCAL_OUT 处理后，进行路由选择处理，然后经过 NF_IP_POST_ROUTNIG 处理，最终发送到网络上。

4.3 AODV 路由协议实现结构和功能

AOVD 路由协议在 Linux 操作系统下路由协议实现结构如图 4 所示。对数据包的处理在核心层和用户层进行，核心层模块为 kaodv.ko，用户层的模块为 aodvd。该协议的主要部分工作在用户层，用于建立和维护 linux 内核路由表。

kaodv.ko 是一个可加载内核模块[5]，是 AODV 协议核心层的主程序，通过它使得内核程序正常工作，内核的初始化和注销都在这里。在其中定义了一些回调函数，并将这些回调函数分别与 Netfilter 的三个 hook 点进行挂载。分别为：NF_IP_PRE_ROUTING（流入报文在决策路由前执行）、NF_IP_LOCAL_OUT（本地产生数据包流出路由前执行）、NF_IP_POST_ROUTING(本地或转发报文发送之前执行)。

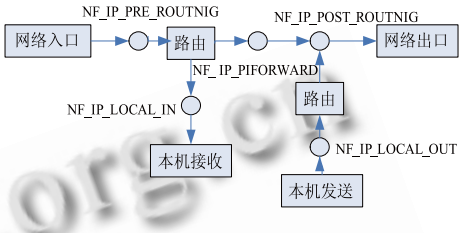

图 3 Netfilter 的 5 个 hook 点

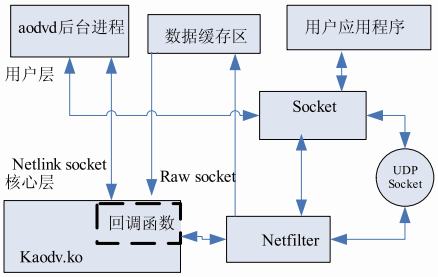

图 4 AODV 路由协议实现结构图

在用户层 aodvd 后台进程程序是一个无限循环读取事件队列的进程，首先判断是属于哪一种事件(RREQ、RREP、RERR、HELLO、ACK 等)，再开始启动该事件的处理程序，不断重复读取事件队列，直到返回值是 NULL 为止。路由算法的功能通过该进程实现，并作为后台守护进程在用户层执行，负责与其它节点进行信息的交互，建立和维护路由表。在用户层主要完成如下功能。

4.3.1 AODV 控制信息的处理

AODV 路由的发现和维护是通过发送和接收到控制信息分组经过 AODV 算法处理后来完成的，该控制信息由 UDP 套接字发送和接收，并由 aodv_socket 模块进行处理。AODV 控制信息分组的“type”字段指明该消息的类型，根据控制信息类型，即可正确地处理这些控制信息分组。各类控制信息分组的生成、发送、接收、转发、处理是 aodv_hello、aody_rerr、aody_rrep、aody_rreq 等模块协作完成的。

4.3.2 AODV 数据报文的处理

在本地直接发送或者转发报文时，首先判断核心

路由表中是否存在与该报文的目的地址相匹配的路由条目，如果存在，根据该路由条目信息提供的相应网络接口将其发送到对应的下一跳节点地址上，如果不存在与目的地址相匹配的路由条目，该报文就通过挂接在 NF_IP_LOCAL_OUT hook 点的回调函数进行处理，并通过用 Linux 的原始套接口将数据包送往用户层的数据缓冲区中进行排队。原始套接口能比一般的 UDP 或者 TCP 套接口提供更多的功能。同时，aodvd 后台进程启动路由发现过程，进行路由查找。如果查找到与被缓存报文目的地址匹配的路由，则将这条路由插入到内核路由表中，用原始套接口(Raw Socket)将缓存的数据分组重新发送出去。如果在规定的时间内未找到相应的路由，则这些缓存的数据报文会被丢弃，并通知源节点出错。

4.3.3 AODV 用户层与核心层之间的信息交互

Netlink socket 可以在核心层与用户层之间进行双向的数据交互[6]，当路由后台程序将查找到的路由信息通知给内核以修改核心路由表时，用户层使用标准的 socket 就可以实现 Netlink 所提供的强大功能，并且在 Netlink 的基础上，使用 rtnetlink 可以方便的操作 Linux 的核心路由表。当内核控制程序需要通知用户层的后台程序进行路由查找时，核心层需要使用专门的内核 API 来实现 Netlink 数据交互，Netlink socket 将核心路由表的使用状态传递给用户层，告知 aodvd 后台进程核心路由表的使用状况，aodvd 后台进程据此更新路由缓冲表的定时器，同时通过 Netlink socket 删除内核路由表中过时的路由条目或增加新的路由表项。

4.3.4 AODV 路由表的操作

在 AODV 路由协议实现的过程中，除了内核路由表外，在用户层还需建立了 AODV 路由表，用来记录各种路由信息，路由表条目的建立或更新是基于所收到 AODV 控制报文信息的，AODV 协议实现的路由表数据结构如表 1 所示，该结构是根据 AODV 协议构造的，记录了每条路由的目的序列号、到目的节点地址和跳数等信息，比内核路由表内容更丰富，路由查找时需要利用这些信息，程序在用户层的路由表定义了各种操作函数，用 Routing-table 的数据表项实现对 AODV 路由表项进行删除和添加。

表 1　AODV 协议实现的路由表数据结构

数据项名称	数据类型	数据项说明
dest_addr	struct in_addr	目的地 IP 地址
dest_seqno	u_int32_t	目的地址序列号
ifindex	unsigned int	网络接口索引
next_hop	struct in_addr	下一跳 IP 地址
hcnt	u_int8_t	到目的地的跳数
flags	u_int16_t	路由标志
state	u_int8_t	端口状态
rt_timer	struct timer	端口定时器
ack_timer	struct timer	目的地 rrep_ack 定时器
hello_timer	struct timer	发送 hello 消息定时器
Last_hello _time	struct timeval	最近一次 hello 消息
hello_cnt	u_int8_t	发送 hello 消息的次数
hash	hash_value	迅速定位本地端口
nprec	int	先前使用序列号数目
precursors	list_t	使用此路由节点列表

5　结语

本文以 PXA310 处理器和 IEEE802.11b 无线网卡为核心，搭建了 AODV 路由协议实现硬件平台，分析了嵌入式 Linux 系统中基于 Netfilter 功能框架的 AODV 协议设计与实现方法，具有较好的通用性和扩展性，为 Ad Hoc 路由协议的实现提供重要的参考价值。

参考文献

1 Perkins CE. Royernad5 EM.Das.Ad Hoc on Demand Distance Vector(AODV)Routing. IETF RFC3561,July, 2003.

2 沈晓松,宿景芳,武穆清.基于 ARM 的嵌入式 Ad Hoc 网络平台的实现.电子设计应用,2009.11.

3 钱晓华,郭继红.基于嵌入式 Linux 的无线网卡驱动程序.辽宁大学学报(自然科学版),2008,(l):36−45.

4 Chakeres I, Belding D, Royer EM. AODV Routing Protocol Implementation Design. 24th International Conference on 2004.698−703.

5 邵薇.AODV 路由协议研究及其在无线测试床平台上的实现与应用[硕士学位论文].长沙:国防科学技术大学,2007.

6 Dhandapan I, Sundaresan G. A Netlink Sockets Overview. University of Kansas, 1999.