第 33 卷　第 22 期　　计　算　机　工　程　　2007 年 11 月
Vol.33　No.22　　Computer Engineering　　November 2007
·网络与通信·　文章编号：1000—3428(2007)22—0119—02　文献标识码：A　中图分类号：TP393.02

一种新型 IPv6 路由器控制平面的设计与实现

陈晓梅，王宝生，赵　峰，涂　睿
(国防科技大学计算机学院，长沙 410073)

摘　要：转发与控制分离体系结构将路由器严格划分为控制和转发两个层面，具有开发成本低、系统可扩展性强、可靠性高等优点。文章介绍了一种基于转发与控制分离设计思想的 IPv6 路由器，给出了控制平面软件模块设计，包括路由子系统、内核协议栈、OpenRouter Master、OpenRouter Agent 等模块。

关键词：IPv6 路由器；控制与转发分离；体系结构

Design and Implementation of Control Plan of New IPv6 Router

CHEN Xiao-mei, WANG Bao-sheng, ZHAO Feng, TU Rui
(School of Computer Science, National University of Defense Technology, Changsha 410073)

【Abstract】 Based on ForCES(forwarding and control element separating) theory, the router can be separated into control layer and forwarding layer. This kind of design may bring forward great benefits, such as lower development cost, more scalable and reliable system. This paper introduces a kind of ForCES-based IPv6 router, gives the design and implementation of the control plan software modules, including route module, kernel protocol module, OpenRouter Master module and OpenRouter Agent module.

【Key words】 IPv6 router; control and forward separating; architecture

传统路由器通常基于单个通用 CPU，采用实时操作系统，CPU 既要完成报文的转发还要运行路由协议和其他控制和管理协议。这种将控制和转发集成在一起的紧耦合结构导致对控制层面和转发层面的任何改动都会牵一发而动全身，致使路由器的扩展性和软件的移植性较差。

转发与控制分离体系结构(forwarding and control element separating, ForCES)[1]的关键思想是打破控制和转发紧耦合的集成方式，将路由器严格地划分为控制和转发两个层面。控制层面使用通用CPU负责路由的控制和计算，转发层面的每块转发板使用专用CPU(例如网络处理器)负责报文的转发，转发与控制各司其职，在提供高性能报文转发的同时保证了路由控制和计算的高可靠性。典型的转发与控制分离体系结构研究包括：NPF(the network processing forum)[2]；IETF ForCES[3]；Intel[4]；XORP(extensible open router platform)[5]等。与传统路由器体系结构相比，转发与控制分离的体系结构具有以下特点和优势：(1)较低的开发和应用成本；(2)软件开发可以独立于具体的硬件平台；(3)增强了系统的扩展性和可靠性。

本文介绍一种基于 ForCES 思想的 IPv6 路由器——OpenRouter 的设计与实现，着重介绍 ForCES 体系结构下 OpenRouter IPv6 路由器控制平面的设计与实现，有关转发平面的硬件与微码设计将由另文介绍。

1　OpenRouter 总体框架

1.1　系统体系结构

OpenRouter IPv6 路由器是国防科大计算机学院在自行研制的传统 IPv4 路由器结构基础上，开发研制的一款支持 IPv6、采用转发与控制分离设计理念的新型路由器原型系统。系统继承了传统 IPv4 路由器的硬件与微码系统，修改微码，使其能够支持 IPv6 报文转发。所有有关 IPv6 的路由协议和控制报文的计算与处理不再由路由器内部的 CPU 完成，而是使用外接通用计算机处理，称之为路由服务器。整个 OpenRouter IPv6 路由器由路由服务器和传统路由器设备组成。

作为转发平面的传统路由器设备功能包括：IPv6 报文硬件转发；重定向目的报文到路由服务器；接收和响应来自路由服务器的控制报文，并转化为对被控路由器的操作执行；路由分离控制协议；主动向路由服务器报告异步事件。

作为控制平面的路由服务器基于开源GUI zebra[6]路由软件，完成下列功能：IPv6 控制协议(如路由协议、IPv6 协议栈、邻居发现等)；通用路由器抽象模型及控制接口；路由器分离控制协议；路由器用户界面。

1.2　软件总体视图

OpenRouter 路由器系统软件结构如图 1 所示。所有和 IPv6 路由协议相关的处理，全部由外置的路由服务器 IPv6

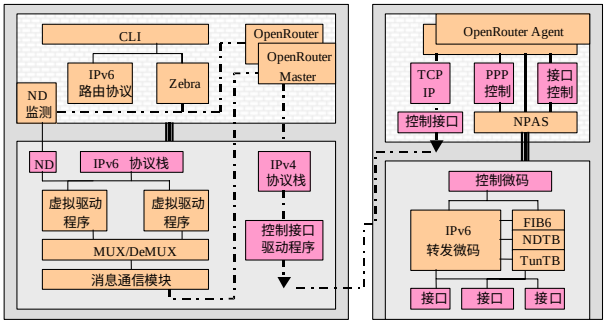

图 1　OpenRouter 路由器系统软件结构

作者简介：陈晓梅(1974－)，女，副研究员，主研方向：高性能网络与通信技术；王宝生，副教授；赵　峰、涂　睿，博士研究生
收稿日期：2006-11-23　**E-mail：**cxmnudt@sina.com

系统完成，IPv6 转发表同样由外部 IPv6 系统生成之后发送给设备的主控，主控通过 NPAS 下载到 NP 转发板。在外置路由服务器和路由器的主控之间遵循标准的协议和接口。NPAS 和 NP 微码之间的接口保持不变。

OpenRouter 路由器控制软件划分为以下子系统：路由子系统(实现 RIPng、OSPFv3、BGP4+等协议)；协议栈及路由设备抽象(包括虚拟 Driver、NP-API)；OpenRouter Master(用户空间的应用级进程)；OpenRouter Agent；NPAS 和控制微码扩展；IPv6 微码功能支持。

2 路由子系统设计与实现

如图 2，OpenRouter 路由器的路由子系统在外置路由服务器上实现，系统采用开源的 zebra 软件为原型，借助其现有的 RIPng、OSPFv3、BGP4+路由协议以及路由管理子系统实现。有关 IPv6 协议栈功能借助了 Linux 操作系统现有的协议栈。Zebra 是一个路由器的软件实现，如何实现软件控制、硬件转发，是 OpenRouter 路由器必须要解决的关键问题，也是集中体现转发与控制分离设计理念的地方。为实现路由器硬件转发，外置路由服务器系统必须为路由器设备提供：(1)接口地址添加、删除等更新信息；(2)转发表添加、删除等更新信息；(3)邻居表添加、删除等更新信息。

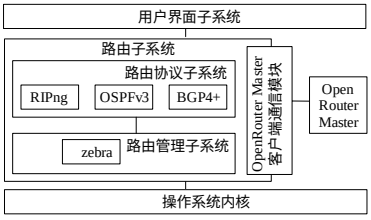

图 2 OpenRouter 路由子系统软件结构

整个外置路由控制系统与硬件路由器设备的通信都是通过 OpenRouter 通信协议完成的。OpenRouter 通信协议由外置主控方的 OpenRouter Master 和硬件路由器设备方的 OpenRouter Agent 共同组成：二者之间分别建立数据通路和控制通路。数据通路用于重定向数据信息；控制通路用于重定向转发表、邻居表操作等控制信息。

在 zebra 路由软件中，有关接口地址操作和转发表操作是在 zebra 进程中实现的。为把相关信息重定向到路由器硬件设备上，建立与 OpenRouter Master 的通信连接，并由 OpenRouter Master 把信息转发到硬件设备上。有关邻居表的维护是在 Linux 操作系统内核完成的。为获得邻居表的实时状态信息，笔者定时查询邻居表，监测其状态变化，并把邻居表的添加、删除、更新等信息通过 OpernRouter Master 重定向到路由器硬件设备上。

3 基于虚拟路由器模型的协议栈

内核协议栈需实现的功能包括：慢速路径的转发平面(TCP/IP 协议栈)；通用路由器的实现表示(如虚拟驱动程序)；内核消息通信(内核与 OpenRouter 的接口规范)。

3.1 虚拟驱动程序

外接的路由服务器负责 IPv6 路由协议和控制报文的处理，位于一台独立的 PC 上并通过以太网同路由器硬件平台连接。由于外接的控制服务器系统的协议栈之下没有具体的硬件设备，因此协议栈必须建立在一个虚拟的通用路由器设备之上。图 3 给出内核协议栈和虚拟通用路由器设备层次示意图。通过虚拟的通用路由器模型，可以使用户在外接路由服务器 PC 上的操作“感觉”就像真实作用在路由器硬件设备上一样。

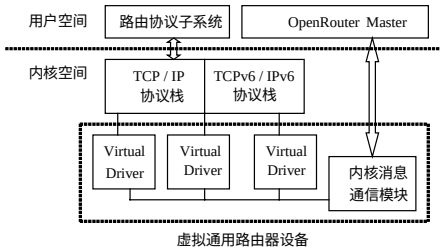

图3 内核协议栈和虚拟通用路由器设备

虚拟通用路由器设备的功能：(1)动态反映真实路由器硬件设备的状态变化；(2)把用户对虚拟设备的操作转化为对真实硬件设备的操作；(3)通过与协议层和用户空间 OpenRouter Master 的交互实现控制信息和 IPv6 报文的转发。

虚拟通用路由器设备通过虚拟驱动程序在内核中为系统虚拟出真实路由器的硬件接口，虚拟接口通过内核消息通信模块与用户空间的 OpenRouter Master 交互，获得真实硬件接口的状态以及转发报文和控制信息；通过与协议层的交互，接收控制信息的转发报文。

本原型系统设计了 5 种类型的虚拟驱动程序，分别对应百兆以太网、千兆以太网、155 兆 POS、622 兆 POS 和 2.5 吉 POS 接口。每种虚拟驱动程序各有 14 个驱动程序，编号为 1~14，分别对应于路由器硬件设备相应槽口号的相应类型的网络接口板。这样通过动态加载、卸载驱动程序模块就可以满足对设备热插拔的要求。

3.2 内核协议栈与用户进程的通信

路由服务器同路由器硬件平台之间的通信是通过在用户空间建立的 TCP 连接来实现的。为了实现协议栈报文到驱动程序的重定向就必须依靠内核协议栈和用户进程的通信。

在本原型系统中，虚拟驱动程序位于 Linux 内核空间，与之交互的 OpenRouter Master 位于用户空间。因此为了完成控制信息和 IPv6 报文的转发，就必须解决好内核空间与用户空间的通信问题。综合考虑了多种通信策略，结合本系统的实际情况，采用为内核增加一套新的系统调用的方式来满足内核与用户空间通信的需要。

4 OpenRouter Master

OpenRouter Master 负责实现基本的控制报文重定向、数据报文重定向、通用控制协议过程实现、协议报文的编码、IPv6 控制配置分发(路由表分发，邻接表分发)、路由器异步事件报告。在现有程序设计中，OpenRouter Master 是 zebra、neigh、OpenRouter Agent 以及协议栈内核的通信枢纽。它负责如下功能：(1)zebra 模块与 OpenRouter Agent control 通路的双向数据通信。(2)邻居发现监测模块与 OpenRouter Agent control 通路的双向数据通信。(3)从协议栈内核读取数据，写向 OpenRouter Agent Data 通路。采用定时轮询方式读取协议栈内核数据。(4)从 OpenRouter Agent Data 通路读取数据，写给协议栈内核。采用 select 调用方式读取 OpenRouter Agent Data 通路的数据由于 OpenRouter Master 在 Linux 系统中负责协调调度来自 3 方(zebra 进程、Linux 内核与 OpenRouter Agent)的消息，在 zebra 进程、Linux 内核同时与 OpenRouter Master 通信的过程中面临着访问阻塞的问题，并且考虑到方案实现的阶段性，系统采用双 OpenRouter Master 的结构。其中，OpenRouter Master A 负责控制报文的转发，OpenRouter Master B 负责数据报文的转发。 (下转第 126 页)