文章编号:1001 - 2486(2006)03 - 0044 - 05

基于 ForCES 体系结构的 IPv6 路由器的研究与实现*

王宝生,夏 毅,陈晓梅,赵 锋
(国防科技大学 计算机学院,湖南 长沙 410073)

摘 要:分析了开放网络体系结构研究现状,并在 ForCES 体系框架下提出一个 OpenRouter 路由器结构模型,该模型由四个逻辑层次组成,三个标准的 API 对路由器进行功能模块化分离,使标准模块化路由器实现成为可能。在此基础上,重点介绍了遵从 OpenRouter 结构模型的 IPv6 路由器系统的实现及关键技术。

关键词:控制与转发分离;IPv6;可扩展路由器
中图分类号:TP393.4 文献标识码:A

Research and Implementation of ForCES-based IPv6 Router

WANG Bao-sheng, XIA Yi, CHEN Xiao-mei, ZHAO Feng
(College of Computer, National Univ. of Defense Technology, Changsha 410073, China)

Abstract: After analyzing the background of open network architecture, the paper proposes an OpenRouter model. The model consists of 4 logic layers, between which there are 3 standard APIs to separate the router into variant modular functions. Based on the OpenRouter model, an IPv6 router system is implemented and the key points of implementation are described within the paper.

Key words: ForCES; IPv6; extensible router architecture

1 问题提出

互联网规模持续增长,对网络设备的计算处理能力、报文转发性能和物理接口支持密度提出更高要求。传统路由器的控制平面无法适应下一代互联网在设备的可靠性、性能和服务的可扩展性等方面的需求。

传统路由器中的控制平面和转发平面以各种方式紧耦合在一起。控制处理器执行控制平面的功能并对线卡进行配置;线卡则执行转发平面的报文处理功能;它们共用一块路由器背板。控制处理器和转发线卡之间的通信采用非标准化的机制,使得来自不同设备提供商的控制处理器和转发单元无法进行交互,这也同时导致转发单元与线卡之间的静态绑定。网络操作者除了购买来自同一家设备提供商的价格昂贵的具有更高处理能力的控制单元之外,无法使用通用 PC 强大的计算处理能力和巨大的存储资源来辅助和改善路由器控制平面的性能。

2 相关工作

开放的可编程网络和主动网络的目标是:通过增加网络的可编程能力,使未来网络能够快速地开发和部署新的服务。它们紧密相关,但又有各自不同的途径和方法。

主动网络[1-2]将存储—转发的传统网络模型转变成存储—计算—转发的模型,报文不再仅是被动地被转发,其主动性体现在报文承载数据的同时,还携带了可执行代码。这些代码被分发到"主动结点"并被执行,以对报文数据进行操作,同时改变结点的当前状态,以便对后续报文进行处理。

OpenSig[3]研究在 ATM、互联网和移动多媒体网络平台中开发新的信令、中间件和服务时出现的开放网络控制的问题。开放可编程网络所倡导的方法是通过标准化的可编程接口来进行开放的网络控

* 收稿日期:2005 - 12 - 01
基金项目:国家自然科学基金资助项目(90412011,90104001);国家 973 资助项目(2003CB314802)
作者简介:王宝生(1970—),男,副教授,博士。

制。这些可编程接口允许应用程序和中间件操纵底层的网络资源,以构建和组织网络服务。

IEEE 的 P1520 参考模型[4]在已有的网络技术基础上提供一个通用的框架,该框架将编程接口映射到实际的对网络的操作中去。将各种网络体系结构及其相应功能映射到 P1520 模型将非常必要。

网络处理器论坛[5](NPForum)同样关注于定义用来构建功能块的标准接口。与下面的 ForCES 不同,NPForum 使用的模型是静态的:它不能表示所有可能实现的功能块的组合,而仅仅是 FE 能够支持的一种简单的组合。另一方面,NPForum 模型中控制与转发的分离是逻辑性的 API,而 ForCES 模型中的控制与转发的分离是协议性的互操作规范。

ForCES[6-7]专注于在协议的层次来分离网络控制单元(CE,control element)和转发单元(FE,forwarding element),在网络单元间快速、可靠和可扩展地分发协议报文,这些网络单元可能执行在分布的结点上。此外,它采用的模型不仅描述了构建功能块的接口,同时也描述了对特定转发单元中的功能块进行配置和编程时的约束和限制,因而进一步提升了互操作能力。

利用廉价的 PC 机构建功能灵活的软件路由器,可以充分发挥基于软件基础的灵活性,有效地满足网络边缘对控制复杂性的要求。这方面的典型研究包括:亚利桑那大学和普林斯顿大学的 Scout[8],MIT 的 Click[9]和华盛顿大学的 Plugins[10]。

3 OpenRouter 模型和 ORP 协议

OpenRouter 是一个控制与转发分离、支持控制和转发平面的开放可编程的路由器体系结构。我们定义了一个开放式路由器协议(ORP,open router protocol)① 以及规范转发路由交换机与控制服务器分层协议实现的 3 层 API。该协议的目标与 IETF 的 ForCES 协议相同,API 则类似于 IEEE 的 P1520 的 L-interface、CCM-interface 和 NPForum 的 SW-API。

为了描述 OpenRouter 模型,我们借用了 ForCES 中的控制平面单元和转发平面单元的表达方式。控制单元和转发单元彼此分离(一跳或者多跳),它们通过运行 ORP 协议来进行彼此间的协同。控制单元和转发单元的集合构成一个互联网络环境中的第 3 层网络单元(即通常所说的路由器)。OpenRouter 的模型如图 1 所示。

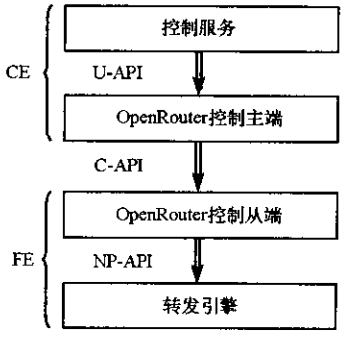

图 1 OpenRouter 模型
Fig.1 OpenRouter model

在 OpenRouter 体系结构中共有三层编程接口:U-API、C-API、NP-API。

U-API 是控制服务实施控制行为的作用接口,具体形态表现为一组供路由协议实现调用的公共函数,分为 4 组:接口管理、路由表管理、状态管理和邻接表管理。C-API 是一个 over-the-wire 的互操作接口,具体形态表现为包括 ForCES 协议的一组控制约定,主要约束控制服务器与路由交换机之间的控制行为的协调一致,控制信息表示的统一以及控制服务器与路由交换机之间的互操作。NP-API 是转发引擎的控制接口。对于基于网络处理器的转发引擎实现,NP-API 遵循 NPForum 的 CPIX(common programming interface)标准。对于常规的转发引擎实现,NP-API 将通过一个控制代理进行统一控制编程接口封装,模拟 CPIX 接口实现。NP-API 是控制服务作用到被控制转发实体的注入接口,具体形态表现为一组转发功能构件的访问接口。

OpenRouter 协议采用主/从的模式,因此,可以支持多个 CE 控制多个 FE 的情形。多个 CE 间可以构成活跃/备份的冗余关系,也可以构成负载均衡的并发关系。这两种关系都可以支持控制平面的容错。

OpenRouter CE 和 FE 的通信采用 TCP/IPv4 连接的方式。CE 和 FE 之间有两条连接:一条用来分发控制消息,另一条用来传输数据报文。控制单元通过对象信息来控制或配置转发单元中的被控对象:这种对象的消息还用来封装需要经过慢速路由处理路径的数据报文。

① ForCES 相关协议还在制定过程中,我们借鉴 ForCES 的思想和基础框架,设计了一套分离控制协议,命名为 ORP 协议。

4 基于 ForCES 结构的 IPv6 路由器的实现

我们按照 OpenRouter 体系结构实现了一套 IPv6 路由系统,图 2 给出了 OpenRouter IPv6 路由系统的典型物理形态图。OpenRouter 系统包括一台基于网络处理器进行 IP 转发的路由交换机以及一台基于 PC 的路由控制服务器。路由交换机提供高性能的 IPv6 报文转发,作为转发引擎主要负责数据平面的报文处理功能:第三层的 IPv6 转发,通过对报文传输进行调度来实施不同级别的 QoS 控制以及缓冲空间的管理等等。转发引擎由分离的控制服务器进行控制,控制服务器运行所有的路由协议(目前的实现中包括了 RIPng、OSPFv3、BGP4 +)。同时为了支持 QoS,它还负责资源管理功能。

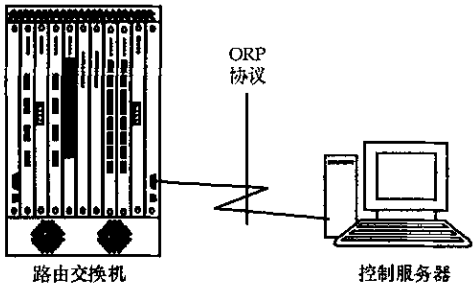

图 2 系统的物理形态图

Fig.1 Physical view of system

4.1 基于 NP 的 IPv6 转发平台

为避免引起歧义,我们将系统中的路由交换机称作转发平台,它对应于 OpenRouter 模型中的 FE,其硬件体系结构包括了基于网络处理器的线卡、内部交换 crossbar 以及控制板。

1. 基于 NP 的线卡

线卡是路由器的网络接口,在我们设计的线卡中使用了 IBM 的 PowerNP。PowerNP 由 16 个 RISC 微处理器构成,它们以 133MHz 主频运行,其指令集类似于 PowerPC 的指令集。每个微处理器可以支持两个线程,因此同时最多可有 32 个线程运行。为了加速执行微码(运行于 NP 核的程序)中的共用任务,每对 NP 核有 8 个专用协处理器。它们执行一些异步功能,比如最长前缀匹配查找、全匹配查找、报文分类、哈希运算以及报文的存储操作等[11]。我们实现一组 IPv6 处理程序,它是一组动态的微码,作为线程运行于微处理器上。

2. 控制板与 crossbar 交换网络

控制板是转发平台的局部控制点。它采用通用的嵌入式微处理器实现,控制同一机柜里的基于 NP 的线卡,同时,控制板有一个以太网控制接口用来支持开放控制。我们采用共享存储器的 crossbar 技术来实现交换网络,它不仅提供高性能,而且支持多种 QoS 保证。

4.2 基于通用计算机系统的 IPv6 控制服务器

IPv6 控制服务器采用了通用计算机服务器平台,可以根据需要提高计算机的配置(比如更大的 DRAM,处理能力更高的 CPU)。该服务器上可以运行如 Linux、Windows、UNIX 等通用操作系统,在我们的实现中,采用 Linux 操作系统作为软件环境。

控制服务器是 OpenRouter 系统的控灼者,它提供 IPv6 路由服务(路由协议),我们基于 Zebra 路由套件[12]来构建路由服务软件。

Zebra 是一个专门用于路由器的模块化软件包。Zebra 包括三个路由协议守护进程:RIPng、OSPFv3 和 BGP4 + 。它们将学习到的路由通告给 Zebra 守护进程,再由 Zebra 守护进程统一与操作系统通信,通过操作系统管理和配置路由表。因为所有的路由协议与相应的守护进程都是组件程序,所以对已有的协议的更新和新协议的添加都十分方便。

4.3 系统软件视图

整个系统软件的详细结构如图 3 所示,由四部分组成:路由服务软件和内核协议栈(控制服务),OpenRouter 控制协议的主端(ORP C-Master),OpenRouter 控制协议的从端(ORP C-Slave)以及 IPv6 转发微

码(转发引擎)。

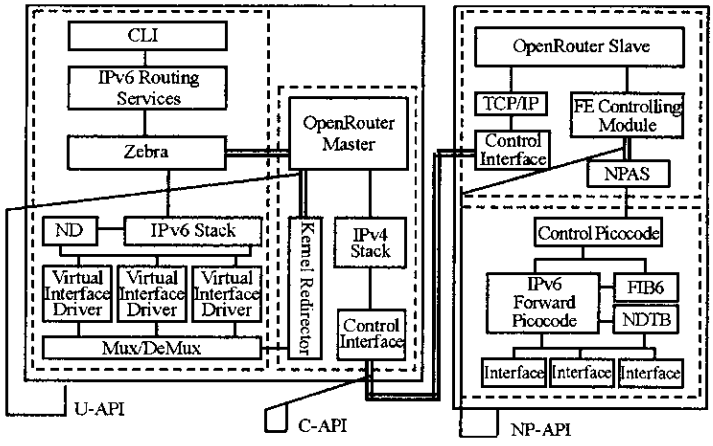

图 3 OpenRouter IPv6 系统总体软件结构图

Fig.3 Software struct of OpenRouter IPv6 system

Zebra 守护进程在各个协议与操作系统的路由表之间提供了一个抽象层,它负责维护一个路由信息表,收集和处理各路由协议学习到的路由,并由 Zebra 守护进程统一将新的路由信息更新到操作系统的路由表中。利用 Linux 内核 IPv6 网络协议栈(包括了 TCP6,ND,ICMP6,ET 等)和一组虚拟网络驱动程序来构造转发引擎抽象,并将其作为 Zebra 的直接控制对象。

ORP 控制协议主端用一个用户态进程来实现,它与 OpenRouter 控制协议从端的交互使用两个 TCP/IP连接:一个是控制连接,另一个为数据连接。C-master 实现 OpenRouter 协议主端的功能,并向内核协议栈和路由服务软件提供 U-API。C-master 与路由服务软件之间采用进程间通信(IPC)的方法来传递控制信息;与内核协议栈之间则通过内核通信模块来重定向报文。ORP 控制协议从端作为 VxWorks 中的一个任务被执行,它执行 OpenRouter 协议从端的功能,并向 OpenRouter 控制协议主端提供 C-API。与此同时,它需要完成与 FE 控制模型相对应的特定 FE 的控制行为。NPAS 再完成从特定 FE 控制行为到基于网络处理器的 FE 模型的映射。NPAS 实现了与 NPForum 类似的 NP-API。

IPv6 转发微码运行在 PowerNP 上,它根据转发表(由路由服务软件进行配置和管理)完成正常的 IPv6 报文转发。

4.4 数据流和控制流

1. 接收报文

报文经过转发平台的某一个接口进入系统。NP 微码首先对报文进行分析,以决定如何处理该报文。它可能选择从系统的另一个接口将报文转发出去,在这种情况下,CE 的网络协议栈无需对报文进行处理;微码也可能发现该报文是路由协议更新报文,或发现是目的地为控制实体的控制报文,在这种情形下,报文被转发给局部控制点——控制板(Slave)。OpenRouter 控制从端将报文转换成 ORP 协议的重定向数据报文,通过 ORP 的 TCP 数据连接发送给分离的控制服务器(Master)。

重定向数据报文首先到达分离控制服务器的 OpenRouter 控制协议主端,OpenRouter 主端将分析重定向数据报文构造相应的元数据,将元数据交给内核重定向模块,元数据中包括了报文的接收端口等信息。内核重定向模块构造 2 层帧,并将其发送到相应的虚拟接口,从而将重定向的数据报文注入到了内核网络协议栈中,Linux 的网络协议栈便可以正常地处理该报文。如果重定向的报文是路由协议报文,它将被进一步被交付给 Zebra,Zebra 及其路由协议将处理该协议报文,并作出路由决策。

2. 发送报文

路由服务软件发送路由协议报文给 Zebra,Zebra 则通过系统调用将该报文发送给内核协议栈。此外,协议栈也可能会触发控制报文。协议栈将会选择一个虚拟接口作为目的外出接口。此时报文将被封装成 2 层帧,并交付给虚拟接口。而虚拟接口把它收到的数据帧继续转交给内核重定向模块,由内核重定向模块解析报文,获取数据帧中的元数据,并将元数据转交给 OpenRouter 主端。OpenRouter 主端收

到这些元数据后,将元数据转化为 ORP 协议的重定向报文,并通过 ORP 的 TCP 数据连接发送给 OpenRouter 从端。

3. 控制流

控制消息分两个方向:下行(CE 至 FE)和上行(FE 至 CE)。这些消息包括:IPv6 转发表添加/修改/删除,IPv6 邻接表添加/修改/删除,端口状态查询,转发逻辑配置和异常通告等。下行方向消息的处理采用同步方式,CE 发送控制消息给 FE,等待 FE 返回的结果。而上行方向的消息处理则采用异步方式,FE 发送异常通告给 CE,CE 异步处理该消息。

5 总结

本文介绍了当前路由器设计的显著趋势:追求高性能的同时,还需要具备良好的可扩展性。目前的路由器,要么仅有非常高的性能(商用路由器),要么只具备灵活的扩展能力(开放的软件路由器)。

为了解决上述问题,我们提出一种开放路由器结构模型——OpenRouter,并基于 OpenRouter 体系结构实现了一套 IPv6 路由系统,将基于网络处理器的高性能转发引擎和开放的 Linux 路由控制服务结合在一起,具备如下优点:

- 与商用路由器的性能级别接近;
- 通过 OpenRouter 协议能够支持多个 FE,从而得到非常高的性能可扩展性;
- 控制平面具备与软件路由器一样灵活的服务可扩展性;
- 控制服务基于通用 Linux 系统,因而拥有广大的开发资源和智力支持;
- 路由器控制平面和转发平面协议级分离;
- 使用分离的专用控制服务器,可以更快和更方便地部署新的路由或控制协议。

参考文献:

[1] DARPA Active Network Program[EB/OL]. http://www.darpa.mil/ato/programs/activenetworks/ actnet.htm,1996.

[2] Tennenhouse D, et al. A Survey of Active Network Research[J]. IEEE Communications, 1997,35(1):80 – 86.

[3] OpenSig Project[EB/OL]. http://www.comet.columbia.edu/opensig/.

[4] Biswas J, et al. The IEEE P1520 Standards Initiative for Programmable Network Interfaces[J]. IEEE Communications, Special Issue on Programmable Networks, 1998,36(10).

[5] Network Processor Forum[EB/OL]. http://www.npforum.org.

[6] Khosravi H, et al. Requirements for Separation of IP Control and Forwarding[S]. RFC 3654, November 2003.

[7] Yang L, Dantu R, Anderson T, et al. Forwarding and Control Element Separation (ForCES) Framework[S].RFC 3746, April 2004.

[8] Scout Operating System Project[EB/OL]. http://www.cs.arizona.edu/scout/.

[9] The Click Modular Router Project[EB/OL]. http://www.pdos.lcs.mit.edu/click/.

[10] Decasper D, Dittia Z, Parulkar G, et al. Router Plugins: A Software Architecture for Next Generation Routers[A]. In Proceedings of ACM SIGCOMM'98[C], Vancouver, Canada, September 1998.

[11] Allen J R, Bass Jr B M, et al. IBM PowerNP Network Processor: Hardware, Software, and Applications[J]. IBM J. Res. & Dev. 2003,47(2/3).

[12] Zebra Project[EB/OL]. http://www.zebra.org/.