|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | | | | | | | |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  | 计算机网络研究性实验 | | | | | |  |  |
|  |  |  |  |
|  |  | BGP协议功能的设计与实现 | | | | | |  |  |
|  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | 专业名称 | 计算机科学与技术（创新实验班） |  |  |
|  |  | 指导老师 | 张力军 |  |  |
|  |  | 学生姓名 | 胡京徽 |  |  |
|  |  | 学生学号 | 11061191 |  |  |

2014 年 7 月 10 日

**摘要**

本文介绍了笔者在GNU/Linux环境下，根据rfc1771文档，通过网络编程接口实现的BGP协议模拟器。

文中首先着重分析了BGP报文的格式和相关作用，同时详细解读了BGP协议的有限状态机的结构。然后，给出了笔者对将要实现的系统设计思路。接着给出了笔者实现BGP协议的相关细节。最后给出了笔者在GNU/Linux实际环境中的测试报告。

论文主体分为五章，章节安排如下

第一章, 说明课题的来源及研究目标

第二章，解读BGP协议相关内容

第三章，对系统的设计进行详细说明

第四章，对系统的实现进行详细说明

第五章，对实现的测试进行说明

最后是笔者的个人总结。

**关键字：BGP，rfc1771，GNU/Linux，网络编程**

目录

[第一章 绪论 3](#_Toc393023212)

[1.1 课题来源 3](#_Toc393023213)

[1.2 课题要求 3](#_Toc393023214)

[1.3 项目管理 3](#_Toc393023215)

[第二章 BGP协议研究 4](#_Toc393023216)

[2.1 BGP协议的简介 4](#_Toc393023217)

[2.2 BGP报文格式 5](#_Toc393023218)

[2.2.1 报文首部格式 5](#_Toc393023219)

[2.2.2 OPEN报文格式 6](#_Toc393023220)

[2.2.3 UPDATE报文格式 8](#_Toc393023221)

[2.2.4 KEEPALIVE报文格式 12](#_Toc393023222)

[2.2.5 NOTIFACATION报文格式 12](#_Toc393023223)

[2.3 路径属性使用 13](#_Toc393023224)

[2.3.1 ORIGIN 13](#_Toc393023225)

[2.3.2 AS-PATH 13](#_Toc393023226)

[2.3.3 NEXT-HOP 14](#_Toc393023227)

[2.4 BGP有限状态机 14](#_Toc393023228)

[2.4.1 Idle状态 15](#_Toc393023229)

[2.4.2 连接状态 15](#_Toc393023230)

[2.4.3 Active状态 16](#_Toc393023231)

[2.4.4 OpenSent状态 16](#_Toc393023232)

[2.4.5 OpenConfire状态 17](#_Toc393023233)

[2.4.6 建立状态 17](#_Toc393023234)

[2.4.7 状态图 18](#_Toc393023235)

[2.4.8 BGP的事件 19](#_Toc393023236)

[第三章 系统设计 20](#_Toc393023237)

[3.1 系统总述 20](#_Toc393023238)

[3.2 IP报文转发子系统 21](#_Toc393023239)

[3.2.1 IP报文监听模块 21](#_Toc393023240)

[3.2.2 IP报文转发模块 22](#_Toc393023241)

[3.2.3 ARP报文管理模块 23](#_Toc393023242)

[3.2.4 网卡管理模块 23](#_Toc393023243)

[3.2.5 路由表查找模块、 24](#_Toc393023244)

[3.2.6 路由器功能模块 24](#_Toc393023245)

[3.3 BGP协议管理子系统 25](#_Toc393023246)

[3.3.1 BGP有限状态机管理模块 25](#_Toc393023247)

[3.3.2 BGP报文监听模块 25](#_Toc393023248)

[3.3.3 BGP报文发送模块 26](#_Toc393023249)

[3.3.4 BGP报文解析模块 26](#_Toc393023250)

[3.3.5 BGP配置读取模块 27](#_Toc393023251)

[3.3.6 BGP报文更新模块 27](#_Toc393023252)

[3.4 调度子系统 28](#_Toc393023253)

[第四章 系统实现 29](#_Toc393023254)

[4.1 实现总述 29](#_Toc393023255)

[4.2 IP报文转发系统 29](#_Toc393023256)

[4.2.1 Watcher类 30](#_Toc393023257)

[4.2.2 Router类 31](#_Toc393023258)

[4.2.3 Interface类 31](#_Toc393023259)

[4.3 BGP管理系统 32](#_Toc393023260)

[4.3.1 Simulator类 32](#_Toc393023261)

[4.3.2 Peer类 33](#_Toc393023262)

[4.3.3 Dispatcher类 33](#_Toc393023263)

[4.3.4 Timer类 33](#_Toc393023264)

[4.4数据结构 34](#_Toc393023265)

[4.4.1 网卡表项 34](#_Toc393023266)

[4.4.2 路由表项 34](#_Toc393023267)

[4.4.3 ARP表项 34](#_Toc393023268)

[4.4.4. BGP报文首部 35](#_Toc393023269)

[4.4.5 OPEN报文 35](#_Toc393023270)

[4.4.6 其它报文 35](#_Toc393023271)

[4.4.7 UPDATE信息结构体 35](#_Toc393023272)

[第五章 测试效果 37](#_Toc393023273)

[5.1 实验组网 37](#_Toc393023274)

[5.2 配置内容 38](#_Toc393023275)

[5.3 报文分析 39](#_Toc393023276)

[5.4 测试结论 40](#_Toc393023277)

[个人总结 41](#_Toc393023278)

[参考文献及网址 42](#_Toc393023279)

# 第一章 绪论

## 课题来源

该课题是2014年北航大三计算机科学与技术网络提高层次众多选题之中的一个，主要是要求学生独立地对边界网关协议（BGP）的研究与实现。

边界网关协议（BGP）是运行于 TCP 上的一种自治系统的路由协议。 BGP 是唯一一个用来处理像因特网大小的网络的协议，也是唯一能够妥善处理好不相关路由域间的多路连接的协议。 BGP 构建在 EGP 的经验之上。 BGP 系统的主要功能是和其他的 BGP 系统交换网络可达信息。网络可达信息包括列出的自治系统（AS）的信息。这些信息有效地构造了 AS 互联的拓朴图并由此清除了路由环路，同时在 AS 级别上可实施策略决策。

当时笔者出于兴趣，参与到这个课题中。从头实现一个简单的BGP协议，同时可以学会了在GNU/Linux下网络编程技巧，受益良多。

## 课题要求

1. 学习BGP协议，了解BGP协议的原理；
2. 基于Linux实现基本的BGP协议；
3. 实现BGP协议规定格式数据的封装发送；
4. 接受符合BGP协议规定格式的数据包，并进行逻辑处理；
5. 根据BGP原理生成单播路由表，并使得单播数据包能根据路由表进行转发；
6. 需识别携带基本路由属性的报文种类。

## 项目管理

笔者将项目的代码托管在CSDN上，工程首页快速链接<https://code.csdn.net/jeanhwea/bgpsim>

开发环境：

|  |  |  |  |
| --- | --- | --- | --- |
| ubuntu10.04 | Kdevelop4 | g++ 4.6 | gdb 7.4 |

# 第二章 BGP协议研究

## 2.1 BGP协议的简介

边界网关协议（BGP）是自治系统间路由协议。它的建立来源于RFC904[1]中定义的EGP以及RFC1092[2]和RFC1093[3]中描述的EGP在NSFNET骨干网中的使用。

BGP交互系统的主要功能是和其他的BGP系统交换网络可达信息。网络可达信息包括可达信息经过的自治系统（AS）清单上的信息。这些信息有效地构造了AS互联的图像并由此清除了路由环路同时在AS级别上实施了策略决策。

BGP-4提供了一套新的机制支持无类域间路由。这些机制包括支持网络前缀的广播取消BGP网络中“类”的概念。BGP-4也引入机制支持路由聚合，包括AS路径的聚合。这些改变为[8, 9]建议的超网方案提供了支持。

为了刻画BGP执行的路由决策，集中讲述BGP发言者通告他自己使用的路由到相邻AS中对端（与之通信的别的BGP发言者）的规则。这些规则反映了当今互联网广泛使用的“一跳一跳”路由范例。注意一些策略不被“一跳一跳路由范例支持所以需要比如源路由之类的技术来增强。例如，BGP不支持AS发送流量到相邻的AS但是路由和源自相邻AS流量有不同的路由。另一方面，BGP支持任何与“一跳一跳”一致的策略。由于当前互联网只使用“一跳一跳”路由范例同时BGP支持与范例一致的策略，BGP作为AS间路由协议非常适用于当今互联网。

## 2.2 BGP报文格式

### 2.2.1 报文首部格式

0 1 2 3

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| |

+ +

| |

+ +

| Marker |

+ +

| |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Length | Type |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

Marker（标记）:

本16字节的域包含报文接收者可以预测的值。如果报文类型是OPEN，或者OPEN报文承载了认证信息（作为可选参数），标记必须是全1。否者，标记的值要使用认证机制来计算（认证机制是通过认证信息的一部分来指定的）。标记可以用来探测BGP对端的同步丢失，认证进入的BGP报文。

长度(Length):

两字节无符号整数指定了报文的字节全长，包括头部的字节。这也就是说，允许在传输层数据流定位下一个报文（的标记域）。长度的值必须最少19字节最大4096字节，同时由于不同的报文有更多的约束。不允许“填充”多余的数据在报文后，所以长度域是需要的最小值。

类型（Type）:

一字节的无符号整数制定了报文类型编码。如下定义：

1 - OPEN

2 - UPDATE

3 - NOTIFICATION

4 – KEEPALIVE

### 2.2.2 OPEN报文格式

在传输协议连接建立之后，两边发送的第一个报文是OPEN报文。如果OPEN报文可以接受，需要发回一个KEEPALIVE报文来确认OPEN报文。一旦确认了OPEN报文，UPDATE，KEEKPALIVE和NOTIFICATION报文可以交换。

在定长的BGP报文头后面，OPEN报文包含下列域：

0 1 2 3

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

+-+-+-+-+-+-+-+-+

| Version |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| My Autonomous System |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Hold Time |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| BGP Identifier |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Opt Parm Len |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| |

| Optional Parameters |

| |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

Version（版本）：

1字节无符号整数指示报文的协议版本号。当前的BGP版本号是4。

My Autonomous System（我的自治系统）：

2字节无符号整数指示发送者自治系统号。

Hold Time（保持时间）：

2字节的无符号整数指示了发送者期望的Hold计时器的秒数。在接收OPEN报文后，BGP发言者必须使用配置的Hold计时器和收到的Hold计时器来计算Hold计时器的值。Hold计时器必须要末是0要末最少3秒。应用可以根据Hold计时器来拒绝连接。计算好的值指示了在连续的KEEPALIVE和/或UPDATE报文之间可以流逝的最大秒数。

BGP Identifier (BGP标示符):

4字节无符号整数指示了BGP发言者的标示符。给定的BGP发言者设置BGP标示符为IP地址。在启动的时候决定BGP表示符，对每一个本地端口和每一个对端是一样的。

Optional Parameters Length（可选参数长度）:

1字节无符号整数指示可选参数域的字节总长度。如果这个域是0，说明没有可选参数。

笔者的实现中这个域为0；

Optional Parameters（可选参数）:

这个域包含了可选参数清单，每一个参数编码为<参数类型，参数长度，参数值>三元组。

0 1

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-...

| Parm. Type | Parm. Length | Parameter Value (variable)

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-...

可选参数说明详细参考rfc1771文档。

**OPEN报文的最小长度是29字节（包括报文头）。**

### 2.2.3 UPDATE报文格式

UPDATE报文用来发送路由信息到BGP对端。UPDATE报文报内的信息可以被用来构造AS之间的关系描述。通过应用以下讨论的规则，路由环路和别的异常可以测出并清除出AS间路由。

UPDATE报文用来广播一条可用路由到对端，或者撤销多条不可用路由。UPDATE报文可以同时广播可用路由并撤销多个不可用路由。UPDATE报文总是包括定长报文头，同时可选的包括下面的域：

+-----------------------------------------------------+

| Unfeasible Routes Length (2 octets) |

+-----------------------------------------------------+

| Withdrawn Routes (variable) |

+-----------------------------------------------------+

| Total Path Attribute Length (2 octets) |

+-----------------------------------------------------+

| Path Attributes (variable) |

+-----------------------------------------------------+

| Network Layer Reachability Information (variable) |

+-----------------------------------------------------+

Unfeasible Routes Length（不可用路由长度）：

2字节无符号整数指示了撤销路由的字节总长度。这个值必须保证网络层可达信息域的长度被确定。0说明没有撤销路由， UPDATE报文内部没有撤销路由。

Withdrawn Routes（撤销路由）：

可变长路由域包括一系列的IP前缀说明撤销服务的路由。每一个IP前缀编码为〈长度，前缀〉二元组，如下描述：

+---------------------------+

| Length (1 octet) |

+---------------------------+

| Prefix (variable) |

+---------------------------+

使用和含义如下：

a) Length(长度)：

长度指示了IP前缀的比特数。0长度指示了匹配所有IP地址的前缀（前缀本身为0字节）

b) Prefix（前缀）:

前缀包含了IP地址前缀后面是填充比特保证域结尾符合字节边界。注意填充比特的值无意义。

Total Path Attribute Length（总的路径属性长度）:

2字节无符号整数值时路径属性域字节总长度。值必须使下文中网络层可达域的长度能够被探测到。值0指示在UPDATE报文中没有网络层可达信息域。

Path Attributes（路径属性）:

在每一个UPDATE报文中有变长的路径属性序列。每一个路径属性是<属性类型，属性长度，属性值>变长三元组

属性类型是2字节域包括了属性标志字节和属性类型码字节。

0 1

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Attr. Flags |Attr. Type Code|

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

属性标志字节第一高位比特（比特0）是可选比特。定义了属性是否是可选的（设为1）或者是公认的（设为0）。

属性标志字节第二高位比特（比特1）是转发比特。定义一个可选的属性是否是转发的（如果设置为1）或者不是转发的（设为0）。公认属性的转发位必须设为1。（参看部分5讨论转发属性）。

属性标志字节的第三比特（比特2）是部分比特。定义是否包括在可选转发属性内的信息是部分的（设置为1）或者是完整的（设置为0）。公认属性和可选非转发的部分位必须是0。

属性标志字节的第四比特（比特3）是扩展长度比特。定义了是否属性长度是1字节（如果设置为0）或者是2字节（如果设置为1）。仅仅当属性值超过255字节的时候，扩展长度可以使用。

属性标志字节低字节顺序4比特没有被使用。必须填0（接收不处理）。

如果属性标志字节的扩展长度比特被设置为0，路径属性的第三个字节包含了属性数据的字节长度。

如果属性标志字节的扩展长度比特设置为1，那末路径属性的第三和第四个字节包含了属性数据的字节长度。

路径属性剩下的字节代表属性值应该通过属性标识和属性类型码翻译。支持的属性类型码，它们的属性值和使用如下定义：

a) ORIGIN (类型码 1):

ORIGIN是公认强制属性定义了路径信息的来源。本数据字节假定如下值：

值 含义

0 IGP – 网络层可达信息和来源AS同内部

1 EGP – 网络层可达信息通过EGP学习

2 INCOMPLETE – 通过别的方式学习网络层可达信息

b) AS\_PATH (类型编码2):

AS-PATH是公认强制属性由一系列AS路径段组成。每一个AS路径段表示为三元组<路径段类型，路径段长度，路径段值>。

路径段类型是1字节长度域有下列定义值。

值 段类型

1 AS\_SET: 在UPDATE报文中的路由经过的AS的无序集

2 AS\_SEQUENCE: 在UPDATE报文中的路由经过的AS的有序集

路径段长度是1字节长度的域包含了在路径段值域的AS的数量。

路径段值域包含了一个或者多个AS号，每一个编码为2字节长度的域。

c) NEXT\_HOP (类型码3):

公认强制属性定义了作为到达UPDATE报文网络层可达域地址所用的下一跳的边界路由器的IP地址

笔者简化了update报文的众多属性，而是仅仅实现了上面的三个公认强制属性。

Network Layer Reachability Information（网络层可达信息）:

这个变长域包含了IP地址前缀的清单。网络层可达信息的字节长度不是明确编码的，但是可以计算如下：

UPDATE报文长度-23-总的路径属性长度-不可用路由长度。

其中UPDATE报文长度是定长BGP报文头的编码值，总的路径属性长度和不可用路由长度是作为部分UPDATE报文的编码值。23是定长的BGP报文头，总的路径属性长度域和不可用路由长度域的组合长度。

可达信息编码时作为一个或者多个二元组格式为〈长度，前缀〉，它们的域描述如下：

+---------------------------+

| Length (长度，1字节 ) |

+---------------------------+

| Prefix (变量 ) |

+---------------------------+

域使用和含义如下：

a) 长度:

长度域指示了IP地址前缀的比特长度。0地址指示了匹配所有IP地址的前缀（前缀本身0字节）

b) 前缀:

前缀域包含了IP地址前缀跟随足够的填充比特使=域的结尾能够落在字节边界。注意填充比特的值不关紧要。

UPDTAE报文的最小长度是23字节—19字节定长报文头+2字节不可用路由长度+2字节总的路径属性长度（不可用路由长度是0同时总的路径属性长度是0）。

UPDATE报文能够广播至少一条路由，路由可用几个路径属性描述。所有的路径树形包括在一个给定的UPDATE报文适用于在UPDATE报文的网络层可达信息域内包含的目的地。

一个UPDATE报文能够列出多个路由撤销服务。每一个路由通过目的地制定（表示为IP前缀），明白的根据上下文指定了BGP发言者-BGP发言者连接先前广播过的路由

一个UPDATE报文可以仅仅撤销路由，这样就不需要包括路径属性或者网络层可达信息。相反，也可以仅仅广播可达路由，这样WITHDRAWN ROUTES不需要了

### 2.2.4 KEEPALIVE报文格式

BGP不使用任何基于传输协议的心跳机制来确定对端是否可达。相反，KEEPALIVE报文在对端之间交换频率要满足HOLD计时器不溢出的标准。合理的最大KEEPALIVE报文时间是HOLD计时器间隔的1/3。KEEPALIVE报文必须不能比每秒一个更频繁。应用可以调整发送KEEPALIVE报文的速率使用HOLD计时间隔的函数。

如果商议哦HOLD计时间隔是0，周期性的KEKPALIVE报文必须不发送。

KEEPALIVE报文只包含报文头长度是19字节。

### 2.2.5 NOTIFACATION报文格式

NOTIFACATION报文在探测到错误情况时发送。BGP连接发送他之后要立即关闭。除了顶长BGP报文头，NOTIFICATION报文包括下面的域。

0 1 2 3

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

| Error code | Error subcode | Data |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ +

| |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

错误码:

1字节的无符号整数指示了NOTIFICATION的类型。下列的错误类型编码定义：

错误码 符号名称

1 报文头错误

2 OPEN报文错误

3 UPDATE报文错误

4 Hold 计时器溢出

5 FSM错误

6 终止

错误子码：在这里不赘述，详细可参见rfc1771文档。

## 2.3 路径属性使用

BGP的UPDATE报文的路由属性承载了路由器之间交流的众多信息，为建立了BGP协议的路由器之间进行修改路由起到了不可估量的作用。下面针对于3个重要的公认强制属性来说明。

### 2.3.1 ORIGIN

ORIGIN式是一个公认强制属性。ORIGIN 属性是产生路由的自治系统产生的。所有选择通告路由到别的BGP发言者的BGP发言者可以把这个包括在UPDATE报文内。

### 2.3.2 AS-PATH

AS-PATH是公认强制属性。本属性定义了UPDATE报文那的路由信息经过的AS。列表的元素可以是AS\_SET或者AS\_SEQUENCE。

当BGP发言者通告从别的BGP发言者 UPDATE的报文学习到的路由，应该根据路由发送到的BGP发言者的位置，修改路由 AS-PATH的属性。

a)当给定的BGP发言者通告路由到本AS的别的BGP发言者，通告发言者应该修改路由的AS-PATH属性。

b)当给定的BGP发言者通告路由到邻居AS的BGP发言者，通告发言者因该修改路由的AS-PATH属性。

1）如果AS-PATH的第一个路径属性是AS-SEQUENCE类型，本地系统应该把字节的AS号码作为序列的最后一个AS号码加在后面（放在最左面）。

2）如果AS-PATH的第一个属性类型是AS-SET类型，本地系统应该添加一个新的路径段AS-SEQUENCE类型，在段的内部放AS号码。

当BGP发言者产生路由：

a)起源发言者应该包括自己的AS号码在发送到邻居AS自治系统的BGP发言者的所有UPDATE报文的AS-PATH属性中。（在这种情况下，起源发言者的自治系统号因该是AS-PATH属性的入口）。

b)起源发言者因该包括一个空的AS-PATH属性在发送到本地自治系统的BGP发言者的所有UPDATE报文的AS-PATH属性中。（空的AS-PATH属性是长度域是0的属性）

### 2.3.3 NEXT-HOP

NEXT-HOP路径属性定义了边界路由器的IP地址，作为到达UPDATE报文的目的地列表的下一跳。如果边界路由器和对端属于同一个AS，对端是内部边界路由器。否则，是外部边界路由器。BGP发言者可以通告任何内部边界路由器作为下一跳，如果本边界路由器和IP地址对应的接口（说明在NEXT-HOP路径属性中）和本地以及远端边界路由器共享公共的子网。BGP发言者可以通告任何外部边界路由器作为下一跳，如果本边界路由器的IP地址是通过对端BGP发言者学习到的，同时边界路由器的IP地址相应的接口（在NEXT-HOP路径属性中说明）和本地以及远端的BGP发言者共享了公共的子网。BGP发言者需要能够支持外部边界路由器的通告能力不足。

BGP发言者必须不通告对端的一个地址作为NEXT-HOP到这个对端，作为这个发言者产生的路由。BGP发言者必须不能安装路由把自己作为下一跳。

当BGP发言者通告路由到本地AS的BGP发言者，通告发言者不应该修改路由的NEXT-HOP属性。当BGP发言者通过内部链路受到路由，可以转发包到NEXT-HOP地址，如果属性中包含的地址是和本地以及远端BGP发言者在公共的子网上。

## 2.4 BGP有限状态机

BGP主要包含下表中的六个状态:

|  |  |
| --- | --- |
| 1 - Idle | 4 - OpenSent |
| 2 - Connect | 5 - OpenConfirm |
| 3 - Active | 6 - Established |

开始BGP在Idle状态。

### 2.4.1 Idle状态

在这个状态，BGP拒绝任何进入的BGP连接。不为对端分配任何资源。响应Start事件（系统或者操作者初始化），本地系统初始化所有的BGP资源，开始ConnectRetry计时器，初始化传输连接到别的BGP对端，当检听到远端BGP对端初始化BGP连接，改变状态到连接。ConnectRetry计时器的确切值是本地设置，但是要有效大于允许TCP初始化。

如果BGP发言者探测到错误，关闭连接转换状态到Idle。脱离Idle状态需要Start事件的产生。如果这个事件自动产生，连续的BGP错误会导致发言者的抖动。为了避免这个情况，建议先前由于错误而转换到Idle状态的对端的Start事件不应该立即产生。在连续产生的Start事件之间的时间，如果事件时自动产生的，应该指数增长。初始计时器的值应该是60秒。计时应该每连续产生一次就加倍。

**在Idle状态下任何别的事件被忽略。**

### 2.4.2 连接状态

在这个状态BGP等待传输协议连接的完成。

如果传输协议连接成功，本地系统清除ConnectRetry计时器，完成初始化，发送OPEN报文到对端，改变状态到OpenSent.

如果传输协议连接失败（比如，重穿超时），本的系统重启ConnnectRetry计时器，继续侦听远端BGP对端初始化的连接，改变它的状态到Active状态。

响应ConnectRetry计时器溢出事件，本地系统重启ConnectRetry计时器，初始化传输连接到BGP对端，继续侦听远端BGP对端初始化的连接，停留在Connect状态。

**Start事件在Active状态被忽略。**

响应其他的事件（被别的系统或者操作者初始化），本地系统释放连接占有的所有的BGP资源，转换状态到Idle。

### 2.4.3 Active状态

在这个状态，BGP尝试通过初始化传输协议连接来得到对端。

如果传输协议连接成功，本地系统清除ConnectRetry计时器，完成初始化，发送OPEN报文到对端，设置Hold计时器为一个很大值，改变状态到OpenSent。计时器值建议是4分钟。

响应ConnectRetry计时器溢出事件，本的系统重启ConnectRetry计时器，初始化传输连接到别的BGP对端，继续侦听远端BGP对端初始化的连接，改变状态到Connect.

如果本的系统探测到远端尝试建立BGP连接到自己，远端的IP地址不是期望的，本的系统重启ConnectRetry计时器，拒绝尝试连接，继续侦听远端BGP对端初始化的连接，停留在Active状态。

**Start事件在Active状态被忽略。**

响应任何别的事件（别的系统或者操作者初始化），本的系统释放连接占有的所有的资源，改变状态到Idle.

### 2.4.4 OpenSent状态

在这个状态BGP等待来自对端的OPEN报文。当OPEN报文受到，所有的域要检查正确性，如果BGP报文头检查或者OPEN报文检查探测到错误（见部分6。2），或者由连接冲突（见部分6。8），本的系统发送NOTIFACATION报文，改变状态到Idle。

如果在OPEN报文内没有错误，BGP发送KEEPALIVE报文设置KeepAlive计时器。Hold计时器，先前被设置为一个大值（见上面），被商议的Hold Time值替代（见部分4。2）。如果商议的Hold Time值是0，Hold Time计时器和KeepAlive计时器要重启。如果Autonomous System 域的值是和本地AS号码一样的，连接是“内部”连接，否则是“外部”连接。（这会影响下面所描述的UPDATE报文的处理。）最后，转态转换到OpenConfirm。

如果从承载传输协议收到断开通告，本的系统关闭BGP连接，重启ConnectRetry计时器，同时继续侦听远端BGP初始化的连接，进入Active状态。

如果Hold计时器溢出，本的系统发送NOTIFICATION报文，错误码是Hold Timer Expired ，同时改变状态到Idle。

响应Stop事件（系统或者操作者初始化），本地系统发送NOTIFICATION报文，错误码是Cease同时改变状态到Idle。

**Start事件在OpenSent状态被忽略。**

对别的事件的响应，本的系统发送NOTIFICATION报文，错误码是Finite State Machine Error同时改变状态到Idle。

无论何时BGP改变状态从OpenSet道Idle,关闭BGP（以及传输层）连接释放连接占用的所有的资源。

### 2.4.5 OpenConfire状态

在这个状态，BGP等待KEEPALIVE或者NOTIFICATION报文。

如果本的系统受到KEEPALIVE报文，改变状态到Established。

如果在收到KEEPALIVE报文之前，Hold 计时器溢出，本的系统发送NOTIFICATION报文，错误码是Hold Timer Expired，改变状态到Idle.

如果本的系统受到NOTIFICATION报文，改变状态到Idle。

如果KeepAlive计时器溢出，本的系统发送KEEPALIVE报文，重启他的KeepAlive计时器。

如果从底层的传输协议受到断开通告，本的系统状态转换到Idle。

响应Stop事件（系统或者操作者初始化）本地系统发送NOTIFICATION报文，错误码是Cease ，改变状态到Idle。

**Start事件在OpenConfirm状态被忽略。**

响应别的事件，本的系统发送NOTIFICATION报文，错误码是Finite State Machine Error，改变状态到Idle。

无论何时BGP改变状态从OpenConfirm到Idle，关闭BGP（传输层）连接同时释放所有连接占用的资源。

### 2.4.6 建立状态

在建立状态，BGP交换UPDATE, NOTIFICATION,和KEEPALIVE报文到对端。

如果本的系统受到UPDATE或者KEEPALIVE报文，开启Hold计时器，如果商议的Hold Time值不是零。

如果本的系统受到NOTIFICATION报文，状态转换到Idle.

如果本的系统受到UPDATE报文，UPDATE报文的错误处理过程（见部分6。3）探测到错误，本的系统发送NOTIFICATION报文，改变状态到Idle。

如果断开通告通过承载传输协议受到，本的系统改变状态到Idle。

如果Hold计时器溢出，本的系统发送NOTIFICATION报文，错误码是Hold Timer Expired ，改变状态到Idle。

如果KeepAlive计时器溢出，本的系统发送KEEPALIVE报文，重启KeepAlive计时器。

每次本的系统发送KEEPALIVE或者UPDATE报文，重启KeepAlive计时器，除非商议的计时器值是零。

响应Stop事件（通过系统或者操作者初始化），本地系统发送NOTIFICATIOIN报文，错误码是Cease，改变状态到Idle。

**Start事件在Established状态被忽略。**

响应别的事件，本的系统发送NOTIFACATION报文，错误码是Finite State Machine Error，改变状态到Idle。

无论何时P改变状态从Established到Idle，关闭BGP（以及传输层）连接，释放连接占用的所有资源，删除所有的连接产生的路由。

### 2.4.7 状态图

对于BGP状态机的转换可简要的使用下面的状态图描述

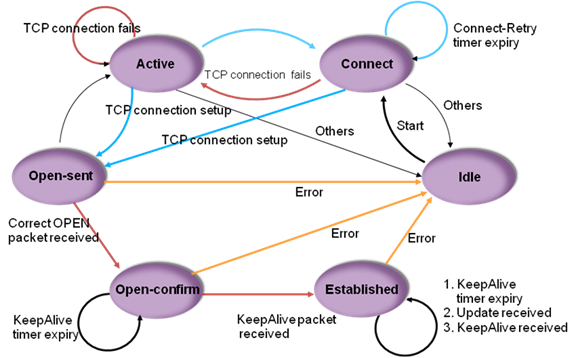


图 1：BGP有限状态机

### 2.4.8 BGP的事件

BGP的十三个事件:

|  |  |
| --- | --- |
| 1 - BGP Start | 8 - Hold Timer expired |
| 2 - BGP Stop | 9 - KeepAlive timer expired |
| 3 - BGP Transport connection open | 10 - Receive OPEN message |
| 4 - BGP Transport connection closed | 11 - Receive KEEPALIVE message |
| 5 - BGP Transport connection open failed | 12 - Receive UPDATE messages |
| 6 - BGP Transport fatal error | 13 - Receive NOTIFICATION message |
| 7 - ConnectRetry timer expired |  |

对于事件和状态相互间的转化以及相互间触发的动作，在rfc1771文档的附录中有详细的总结，可以参考理解。

# 第三章 系统设计

## 3.1 系统总述

系统的整体设计主体上包括十三个模块和一个核心数据结构（参见下面的系统图）。十三个模块分别为IP报文转发模块，IP报文监听模块，ARP报文管理模块，网卡管理模块，路由表功能模块，路由表查找模块，BGP报文监听模块，BGP报文发送模块，BGP报文解析模块，BGP报文读取模块，BGP报文更新模块，BGP有限状态机管理模块和计时调度模块；一个核心的数据结构就是系统的路由表。具体的模块依赖关系参见下面的系统图。

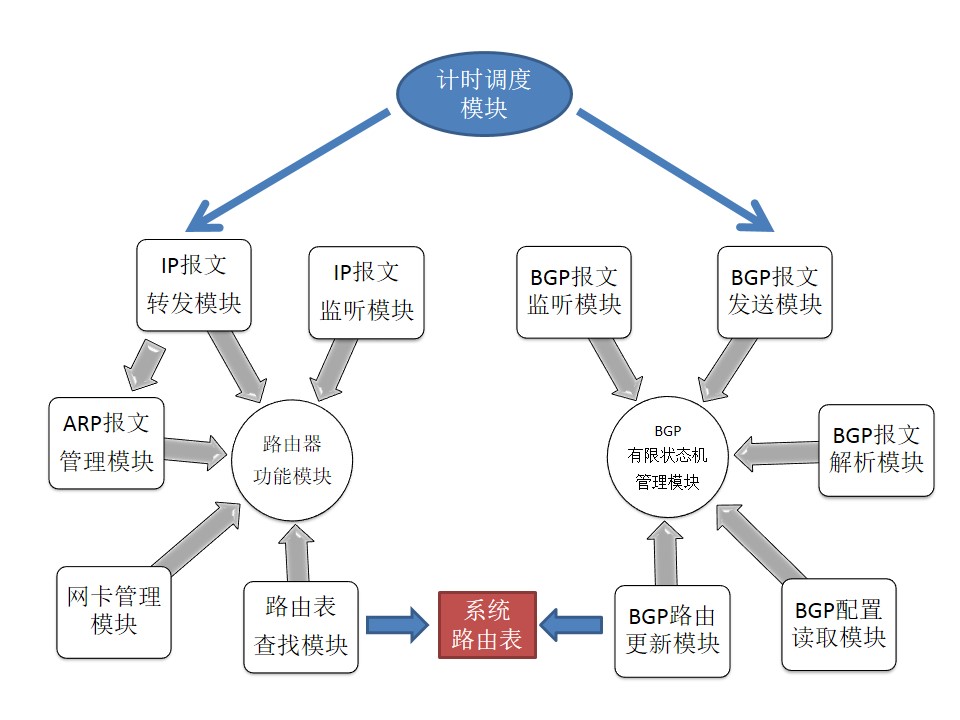


图 2：系统图

这十三个模块又可以分为三类。第一类是IP报文转发子系统，这一子系统是以路由器功能模块为核心，以IP报文转发模块，IP报文监听模块，ARP报文管理模块，网卡管理模块，路由表查找模块为辅助。主要完成对系统中接受到的IP报文进行正确的转发。第二类是BGP协议管理子系统，这一子系统是以BGP有限状态机管理模块为核心，以BGP报文监听模块，BGP报文发送模块，BGP报文解析模块，BGP配置读取模块，BGP报文更新模块为辅助。主要是对BGP协议的功能进行具体实现。第三个子系统是调度系统，主要是包括计时调度模块。这一子系统是对系统报文缓冲队列进行周期性调度的实现。

系统路由表这一数据结构是维系IP报文转发子系统和BGP协议管理子系统之间路由信息和数据交互的一个重要媒介。它是实现BGP协议和IP报文转发的重要信息交互枢纽。

## 3.2 IP报文转发子系统

笔者在该课题中没有调用Linux内核的IP报文转发模块，而是自己实现了一个较为全面的IP报文转发系统。该IP报文转发子系统实现了对Linux的各个网卡报文进行监听，同时根据路由表进行正确地转发IP报文。

### 3.2.1 IP报文监听模块

笔者在这个模块实现的是监听Linux网卡较为底层的报文，报文包括IP报文，ARP报文。这个模块主要监听这两类报文，并将这两类报文交付给IP报文转发模块，和ARP报文管理模块来处理。具体可参加下面的流程图片段。

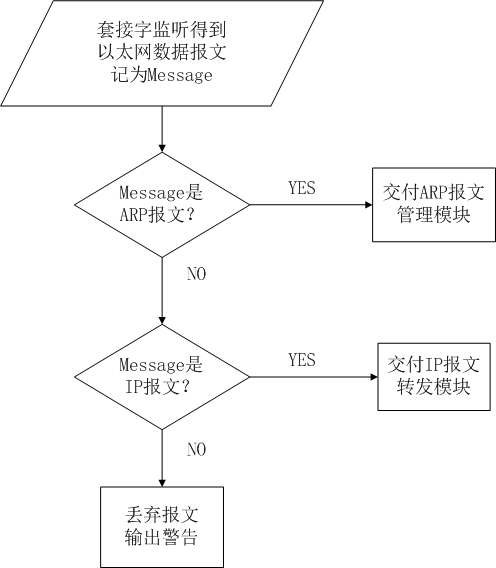


图 3：IP监听流程片段

### 3.2.2 IP报文转发模块

该模块是对IP报文转发，下面是转发一个报文的主要流程图。IP报文的转发主要流程是通过路由表项判断报文是否可达，同时通过ARP缓存来修改MAC地址，在相应网卡转发，同时修改TTL值等相应参数。



图 4：IP报文转发流程图

### 3.2.3 ARP报文管理模块

该模块是对系统的ARP缓存进行管理。它包括ARP报文的请求，ARP报文应答的处理。同时提供查找ARP表项，添加ARP表项。为IP报文的转发提供了有利的工具。

图 5：ARP功能图

### 3.2.4 网卡管理模块

该模块在系统启动时就读入系统的所有网卡信息，并将其缓存在相应的数据结构之中。主要是提供其他模块对网卡的查找提供一系列的工具。也为网络是否可达提供了非常有利的工具。

图 6：网卡管理图

### 3.2.5 路由表查找模块、

该模块是提供给一个查找路由表的接口，它为其他模块提供了访问和查询路由表的一个有利的接口。

图 7：路由表查找功能图

### 3.2.6 路由器功能模块

该模块集成了所有其他模块的功能。它包括对路由表的管理，查找，更新；以及提供了IP报文的相关小工具，为路由功能提供了许多接口。

图 8：路由器功能模块图

## 3.3 BGP协议管理子系统

这个子系统实现了BGP协议的全部功能。为和华为路由进行报文交互，对等体的状态迁移提供了良好的管理。

### 3.3.1 BGP有限状态机管理模块

该模块实现了一个完整的BGP有限状态机（FSM）。为BGP对等体（Peer）的状态随着事件的迁移提供了非常明确的手段。对于BGP状态机的设计参照rfc1771文档。上文也给出了具体的分析。

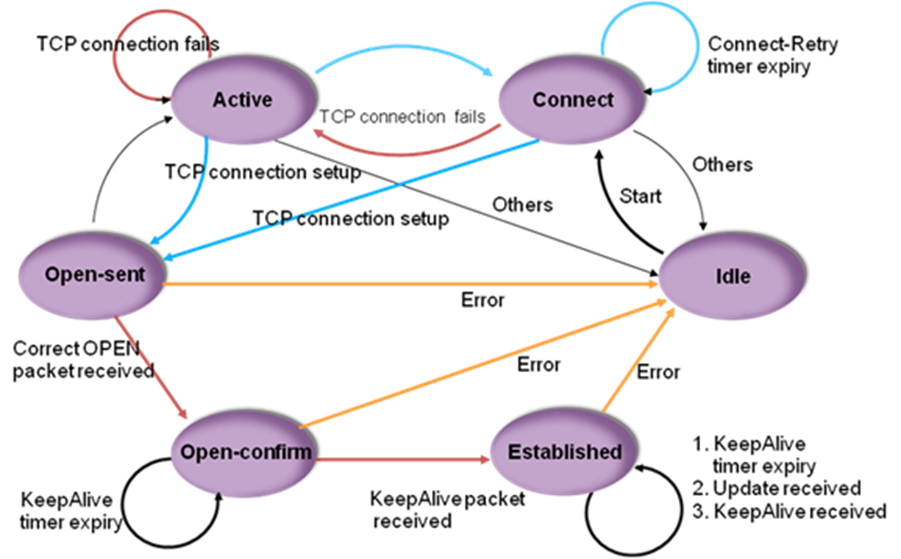


图 9：BGP有限状态机

### 3.3.2 BGP报文监听模块

该模块是监听BGP报文的重要模块。它主要实现了对BGP报文的监听，然后将监听的报文缓存的对等体的接收消息的缓存队列。

与此同时，还对消息的类型进行解析，同时触发对等体接收到相关消息的事件。对等体会对这样的事件重启一个线程做出相关的处理。

### 3.3.3 BGP报文发送模块

该模块是对BGP报文的转发。对于报文的转发过程见下图描述。对于要转发的报文，不是直接发送，而是将报文缓存的队列中，等待计时器的调度。

图 10：BGP报文转发模块

### 3.3.4 BGP报文解析模块

该模块是对BGP各种报文的解析。它包括对BGP报文首部，OPEN报文，UPDATE报文，KEEPALIVE报文，NOTIFICATION报文的解析。

图 11：BGP报文解析模块

### 3.3.5 BGP配置读取模块

该模块是对用户的BGP初始配置读取并建立相应的功能。大体上的配置就是BGP的AS\_NUMBER, BGPID以及BGP对等体所在AS的AS\_NUMBER和相应的IP地址。

关于配置的实现是用户手动编写文本配置文件。然后系统调用相应的函数来读取用户的配置操作。

图 12：配置读取模块

关于配置文件的格式后面会有说明，这里先略去。

### 3.3.6 BGP报文更新模块

当系统接收到UPDATE报文时，会对系统的路由表进行更新和通告其他对等体相关路由信息。该模块是完成对BGP报文的更新，同时对通过其他对等体UPDATE信息也有相应的操作。

UPDATE的通过操作要求修改UPDATE信息的下一条属性，这样做事维持路由可达性，使得对等体的路由表更新可以顺利加入通告的路由表项。

具体的路由更新可以抽象为下面的流程图。



图 13：路由更新流程图

## 3.4 调度子系统

调度子系统是对这个系统的调度。该系统主要组成部分就是计时调度模块。

主要是对那些未能发送成功的缓存队列里面的报文进行重发。此外调度系统还要定时检验计时器是否超时。若超时，则需激发超时事件。

图 14：调度模块功能图

# 第四章 系统实现

## 4.1 实现总述

笔者在上面系统设计的基础上，利用c++编程语言，在GNU/Linux平台上，实现了整个系统。下图是对这个工程所有实现的类的概览。

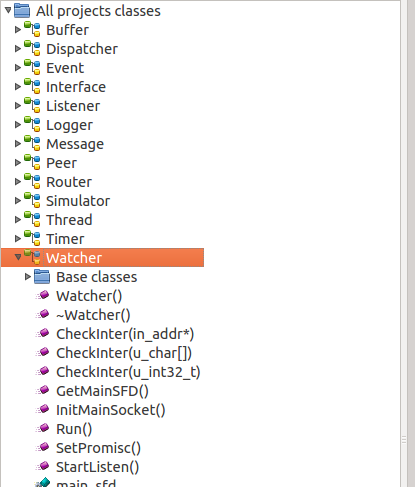


图 15：所有工程的类

对于每个类的功能以及相关的数据结构，在下面的实现说明部分将会给出具体的说明。

## 4.2 IP报文转发系统

报文转发系统实现了对网卡之中的报文进行监听，对于要转发IP报文需要修改的MAC地址这一需求。这个子系统还实现了ARP请求报文的发送。

转发IP报文是这个模块的最基础的功能，对于一个特定的IP包的转发，可以使用Router类的PacketForward函数，并且若未转发成功，则需要将报文放到等待队列之中。

同时这个子系统需要大量地和计算机的接口进行交互，所以Interface类也是实现了对机器之中的管理。

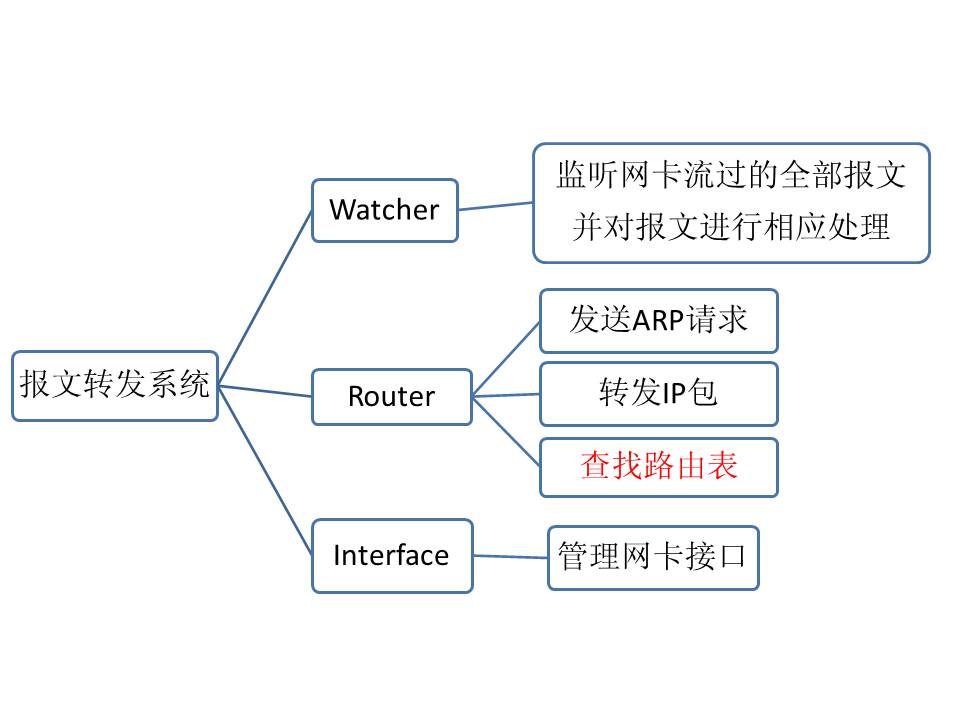


图 16：IP报文转发功能图

### 4.2.1 Watcher类

实现了对网卡的所有报文的监控以及转发的相应的处理，下面是具体实现对于监听的报文进行处理的代码片段。主要的函数是**Watcher::StartListen()。**这个函数完成了对ARP报文和IP报文的监听。下面是具体监听的代码片段。

… …

switch (ntohs(pEthhdr->h\_proto)) {

case ETH\_P\_ARP: // 处理arp报文

pArphdr = (struct eth\_arphdr \*) ((u\_char \*)pEthhdr + sizeof(struct ethhdr));

if (CheckInter(pArphdr->ar\_tip)) {// 检查目的ip是不是当前的网卡接口

**g\_rtr->ARPRosHandle(pArphdr);**  //添加ARP缓存

} else {

if (isDebug) {

g\_log->Tips("arp already known");

}

}

break;

case ETH\_P\_IP: // 处理IP报文

pIphdr = (struct iphdr \*) (pMsg->ReadPos() + sizeof(struct ethhdr)); // IP报头

pTcphdr = (struct tcphdr \*) (pMsg->ReadPos() + // TCP报头

sizeof(struct ethhdr) + sizeof(struct iphdr) );

if (!CheckInter(pIphdr->daddr)) {

// try to forward packet, if ip des is not in my Interface

if (pTcphdr->dest != htons(BGP\_PORT)

&& pTcphdr->source != htons(BGP\_PORT)) {

**g\_rtr->PacketForward(pMsg);** // 对于非BGP的IP报文进行转发

}

}

break;

default:

continue;

}

… …

### 4.2.2 Router类

实现了ARP请求和ARP缓存处理；IP报文的转发；同时实现了查找路由表。主要，这里查找路由表是和BGP管理系统的交互点。



图 17：Router类概览

### 4.2.3 Interface类

读取网卡信息和管理网卡。在系统启动初就对网卡信息进行收集，然后提供给报文转发系统对网卡的一系列操作。

## 4.3 BGP管理系统

BGP管理系统比较复杂，下面给出一个层次图。这个图说明了BGP管理系统的主体实现位于Simulator这个模拟器类里面，Timer是用来计时并且调度的。其它底层的类下面有说明。

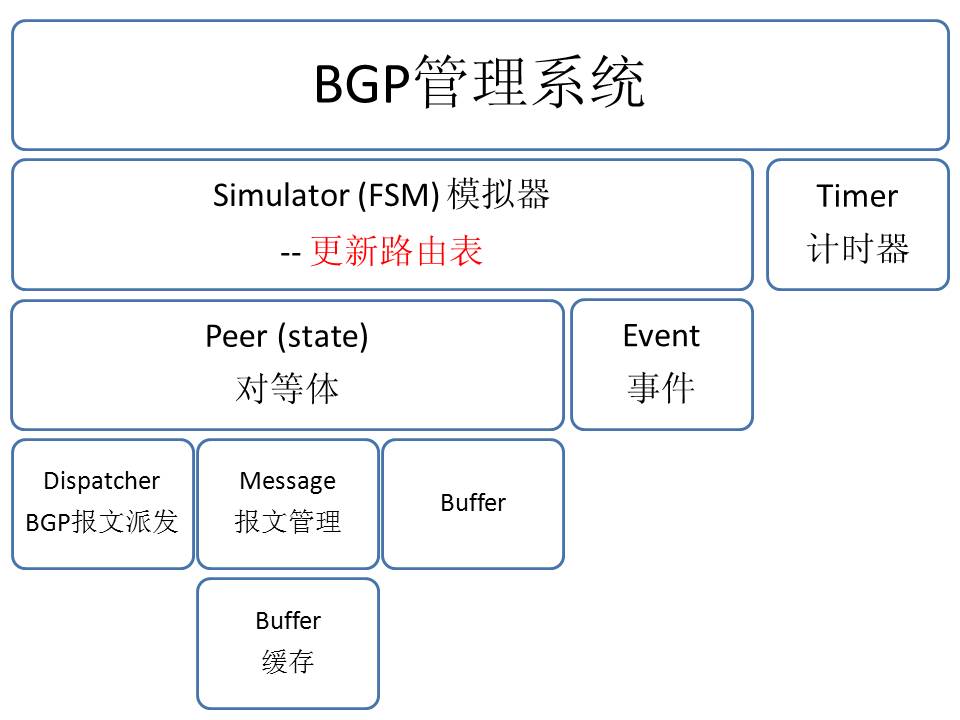


图 18：BGP管理系统层次图

### 4.3.1 Simulator类

其中的主体类是Simulator类，这个类实现了BGP的有限状态机

**Simulator::FSM(Peer \* pPeer, event\_t eve)**

同时实现了一系列对BGP报文解析的函数

**bool ParseHeader(Peer \*, u\_char \*, u\_int16\_t &, u\_int8\_t &);**

**bool ParseOpen(Peer \*);**

**bool ParseNotification(Peer \*);**

**bool ParseUpdate(Peer \*);**

**bool ParseKeepalive(Peer \*);**

以及实现了BGP各类报文的发送函数

**void SimOpen(Peer \*);**

**void SimKeepalive(Peer \*);**

**void AdvertUpdate(Peer \*);**

**void TransUpdate(struct \_bgp\_update\_info \*);**

**void SimUpdate(Peer \*, struct \_bgp\_update\_info \*);**

**void SimUpdate(Peer \*, void \*, size\_t); // 更新路由表 --- 和报文转发系统的交互**

**void SimNotification(Peer \*, u\_int8\_t, u\_int8\_t, void \*, ssize\_t);**

### 4.3.2 Peer类

这个类是对BGP对等体的信息进行收集和管理的主要的类。它存储了一个对等体生命期的所有参数，维护了发送消息队列和接受消息队列，维护了每个对等体的各个计时器。

**queue<Message \*> qMsg; // messages to be sent to others**

**queue<Buffer \*> qBuf; // messages recieved from others**

### 4.3.3 Dispatcher类

激发接受到BGP报文的事件，通知对等体接受到报文，同时必须进行周期性的调度，以检查是否接受到BGP报文。

### 4.3.4 Timer类

计时器类，进行周期性的调度，检查计时器是否超时。主要实现的函数如下：

**void Timer::DoSchedule()**

这个调度函数每秒重新唤醒一次，对系统之中的计时器进行检查，如果任意一个计时器超时，则会触发计时器超时事件。同时对缓存队列里面的报文进行重发。

## 4.4数据结构

### 4.4.1 网卡表项

每张网卡信息是存在一个结构体ifcon里面，具体定义如下：

**struct ifcon {**

**u\_char mac[ETH\_ALEN];**

**struct in\_addr netmask;**

**struct in\_addr ipaddr;**

**struct in\_addr broadcast;**

**int ifid;**

**char name[IFNAMSIZ];**

**};**

这个结构体包含了网卡的mac地址，ip地址，子网掩码，广播，以及网卡id和网卡名称。网卡信息可以通过Linux内核中的netlink中socket读入。

最终所有的网卡信息被同一存储在一个vector里面：

**vector<struct ifcon \*> vIntConf;**

### 4.4.2 路由表项

每条路由表存储在结构体rtcon里面，具体定义如下：

**struct rtcon {**

**struct in\_addr dest;**

**struct in\_addr nhop;**

**struct in\_addr mask;**

**int ifid;**

**};**

这个结构体包含路由表项的目的地址，下一跳，子网掩码以及所经过的网卡ID。

### 4.4.3 ARP表项

每条ARP缓存存储在结构体arpcon里面，具体定义如下：

**struct arpcon {**

**u\_char mac[ETH\_ALEN];**

**struct in\_addr ipadd;**

**};**

这是一个ip地址和mac地址的二元组。

### 4.4.4. BGP报文首部

**struct bgphdr {**

**u\_char marker[16]; // Marker**

**u\_int16\_t length; // Length**

**u\_int8\_t type; // Type**

**};**

### 4.4.5 OPEN报文

**struct openmsg {**

**struct bgphdr msghdr; // Message header**

**u\_int8\_t version; // Version**

**u\_int16\_t myas; // My Autonomous System**

**u\_int16\_t holdtime; // Hold Time**

**u\_int32\_t bgpid; // BGP Identifier**

**u\_int8\_t optparamlen; // Optional Parameters Length**

**};**

### 4.4.6 其它报文

由于其他报文没有固定结构体表示，通常是发送时直接构造。

### 4.4.7 UPDATE信息结构体

这个结构体\_bgp\_update\_info是根据UPDATE报文的结构设计出来的，用来存储UPDATE报文的全部信息。大多数通过这个结构体可以发送路由通告，同时也能分析UPDATE报文。

**struct \_prefix { // 网络前缀二元组**

**u\_int8\_t maskln;**

**struct in\_addr ipaddr; // local host order**

**};**

**struct \_path\_attr\_type { // 路径属性类型**

**u\_char flag;**

**u\_char typecode;**

**};**

**struct \_as\_path\_segment { // AS Path 属性存储信息**

**u\_int8\_t type;**

**u\_int8\_t length; // number of AS**

**Buffer \* value; // list of 2-octet value, in network order**

**};**

**struct \_bgp\_path\_attr { // BGP路径属性结构体**

**u\_int8\_t origin;**

**vector<struct \_as\_path\_segment \*> aspath;**

**struct in\_addr nhop;**

**};**

**struct \_bgp\_update\_info { // UPDATE信息结构体**

**vector<struct \_prefix \*> withdraw; // Withdrawn Routes**

**struct \_bgp\_path\_attr \* pathattr; // Path Attributes**

**vector<struct \_prefix \*> nlri; // Network Layer Reachability Information**

**};**

# 第五章 测试效果

## 5.1 实验组网

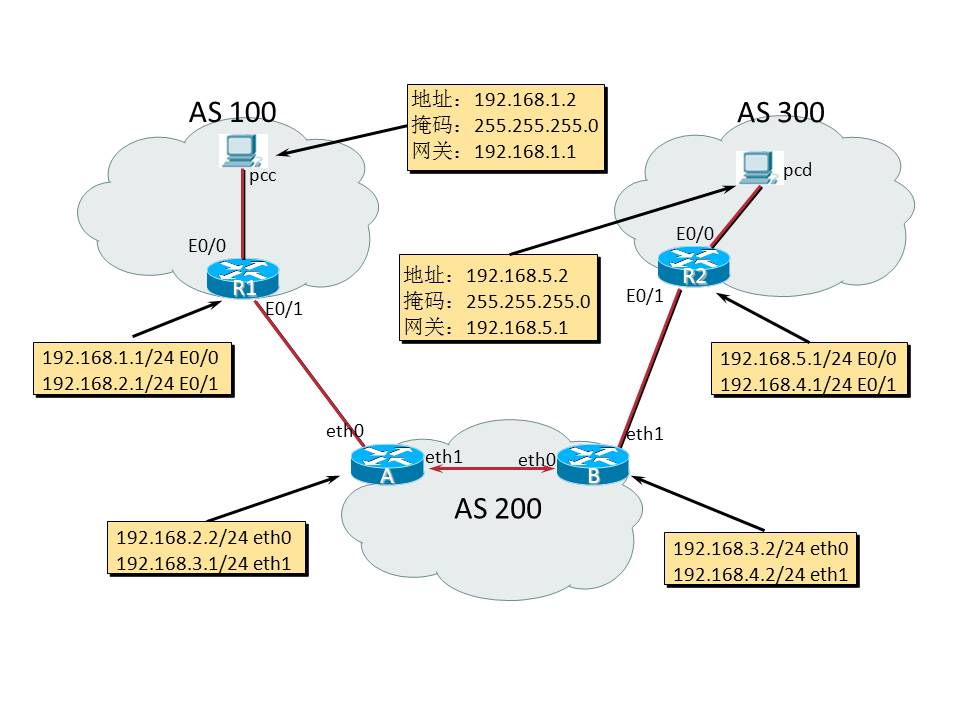


图 19：实验组网图

按照上图进行组网，设置好对于的接口信息。

图中的pcc，pcd是任意两台计算机。用来充当一个AS域里的主机。

图中的R1和R2是两台Quidview2600系列路由器。需要提前开启它们的BGP功能，用来和笔者所实现的BGP协议进行交互。

图中的A，B是一个具有双网卡的Linux主机，在本次实验之中充当路由器。笔者实现的BGP程序将在这两台计算机上运行。

最终的目的是通过BGP协议交互路由信息，最终pcc将能ping通pcd。

## 5.2 配置内容

主机A网卡配置命令如下：

ifconfig eth0 192.168.2.2 netmask 255.255.255.0

ifconfig eth1 192.168.3.1 netmask 255.255.255.0

ifconfig eth0 promisc

ifconfig eth1 promisc

路由器R1初始化命令如下：

system

sysname R1

interface e0/1

ip address 192.168.2.1 24

quit

interface e0/0

ip address 192.168.1.1 24

quit

bgp 100

group nice\_r1 external

peer nice\_r1 as-number 200

peer 192.168.2.2 group nice\_r1

network 192.168.1.2 255.255.255.0 # 设置对等体信息

quit

A的src/config/peer.conf的内容如下：

200 192.168.2.1

200 192.168.3.2

100 192.168.2.2

第一行表示A的AS\_NUMBER=200， BGPID=192.168.2.1

第二行表示A的peer的AS\_NUMBER=200， IP=192.168.3.2

第二行表示A的peer的AS\_NUMBER=100， IP=192.168.2.2

其它设备的配置类似，在此不再赘述。

## 5.3 报文分析

从R1和A链接的端口截获的EBGP报文和在A和B之间截获的IBGP报文如下：

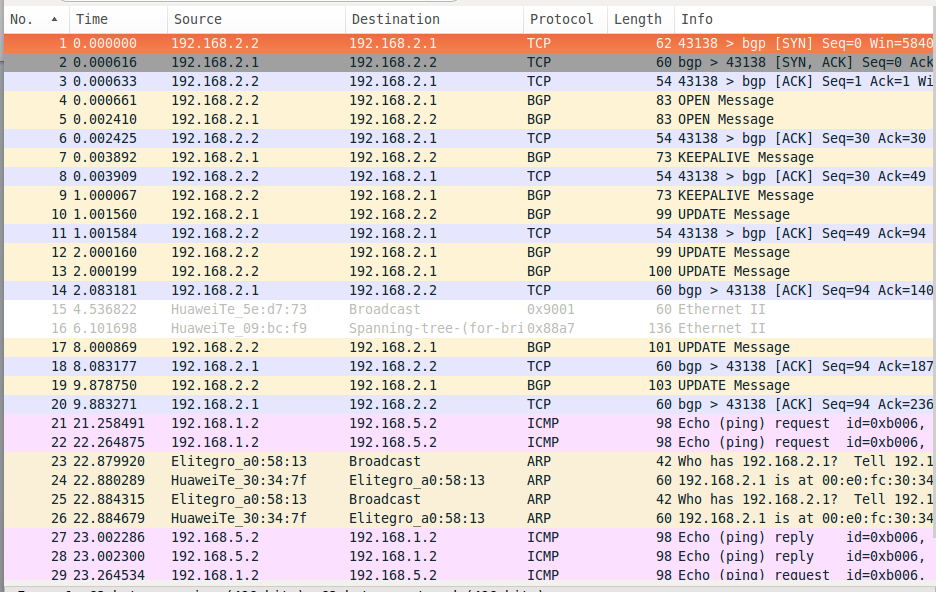


图 20：EBGP报文

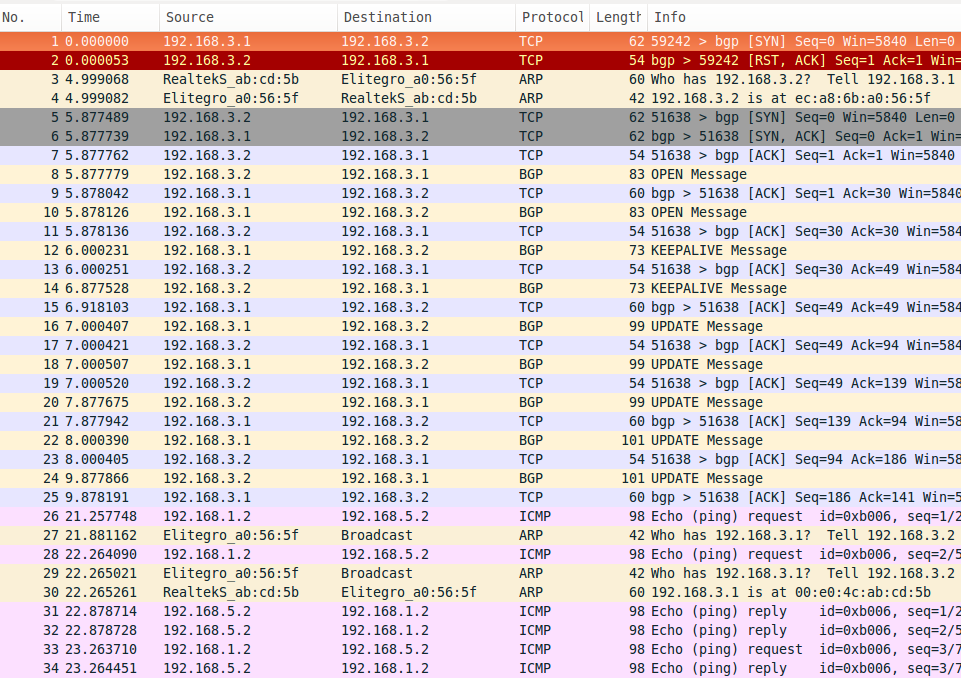


图 21：IBGP报文

最终测试pcc和pcd能互相ping通。

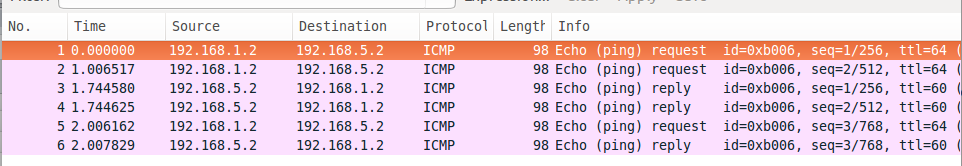


图 22：pcc上截获的icmp报文

同时R1和R2的路由表可以看出通过BGP学习得到的路由表项。下图是R1的表项，可以看出到192.168.5.0/24的路由表项是通过BGP学得到的。

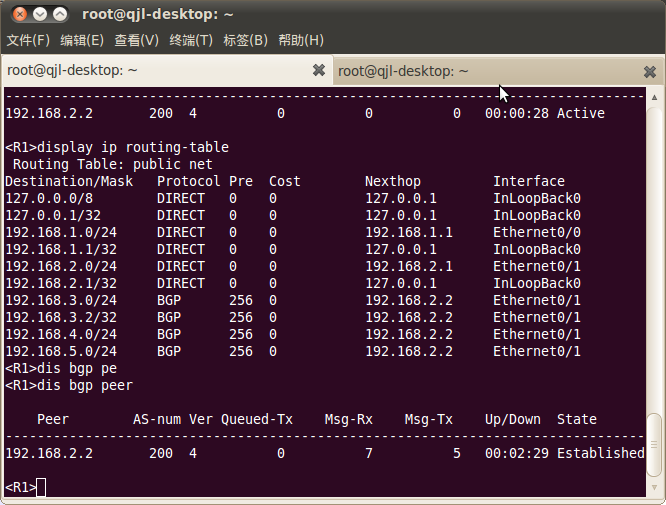


图 23：R1的最终路由表

## 5.4 测试结论

由此可以得出实现的BGP协议正确运行。

# 个人总结

笔者通过大约三个月的学习BGP协议和实现了这个课题。

四月份期间，笔者通读了BGP的rfc1771文档。刚开始对rfc1771的那些BGP报文知识和路由属性是一知半解，感觉一个浩大的课题无法入手。然后，笔者学习学长的一个实现，突然茅塞顿开，开始架构起了属于自己工程课题。同时也改善了一些学长的设计缺陷。

四月末，笔者开始编码，从零开始编码，到七月初最终编写的4600多行c++代码实现了BGP协议。期间当然是受益良多，比如学习了Linux下的套接字编程，体会的Linux下的netlink套接字的强大，学会了多线程程序的一些调试技巧，更加深入了解了如ARP，IP协议的精髓。同时也感慨那些设计这些协议的作者的智慧与汗水。

笔者非常感谢最后期间在实验室调试时张老师给的指导和帮助。每次总是在笔者想要放弃时候，老师的一句话让我找到了一条可以走下去的新的道路。其实学习本来就是这样，在跌倒中不断前进，最终才能达到所期待的目标。

笔者将课题的开发全过程的代码放在了CSDN的课题托管上（没放在github上是因为csdn速度快些），现在回头看着自己开发过程的git提交路线，突然感觉自己真的走了好远，同时也学到了许多。自己的努力不是白费。一分耕耘，一分收获。希望这份宝贵的经历能够激励自己在以后的学习或工作中不断前进。

# 参考文献及网址

[1] 计算机网络实验教程 钱德沛 高等教育出版社

[2] TCP/IP详解 (美)W.R.史蒂文斯(W.RichardStevens)著机械工业出版社

[3] RFC1771 http://www.rfc-editor.org/rfc/rfc1771.txt

[4] UNIX环境高级编程 (美)W.R.史蒂文斯(W.RichardStevens)著，人民邮电出版社

[5] UNIX网络编程，卷一：套接字联网API（第三版）(美)W.R.史蒂文斯(W.RichardStevens)著，人民邮电出版社

[6] Border Gateway Protocol (BGP) <http://www.inetdaemon.com/tutorials/internet/ip/routing/bgp/index.shtml>

[7] netlink socket编程实例 <http://blog.chinaunix.net/uid-14753126-id-2983915.html>

[8] Linux原始套接字实现分析 <http://blog.chinaunix.net/uid-27074062-id-3388166.html>

[9] Linux多线程编程（不限Linux） <http://www.cnblogs.com/skynet/archive/2010/10/30/1865267.html>