**Vx交换机RIP与BFD联动功能软件需求分析说明书**

目录编号：

文件编号：

版本：0.1T

时间：2024-11-20

修订表

本表包含文档修订历史记录。下面的条目仅用于说明目的。

本文档应当在版本控制下存储，每个版本的简述应当输入版本控制系统。该简述在本段重复记录。修订不需要在文档中其它部分描述除非它们解释开发计划本身。

| 版本 | 主要作者 | 联系方式 | 版本描述 | 完成日期 |
| --- | --- | --- | --- | --- |
|  |  |  | 试用及征求意见 |  |

|  |  |
| --- | --- |
| 模块ID | RIP |
| 模块名字 | RIP与BFD联动 |
| 文档版本号 |  |
| 选用模板版本 |  |

**目录**

[1. 概述 4](#_Toc21978)

[1.1 模块概述 4](#_Toc22602)

[1.2 文档描述 4](#_Toc32691)

[1.3 术语说明 4](#_Toc19256)

[1.4 应用范围 4](#_Toc31130)

[2. 重要概念 5](#_Toc30995)

[2.1 BFD相关概念 5](#_Toc25214)

[2.1.1 BFD状态机 5](#_Toc6059)

[2.1.2 BFD检测模式 5](#_Toc26886)

[2.1.3 BFD Echo功能 6](#_Toc4800)

[2.2 单跳限制 6](#_Toc10974)

[2.3 RIP处理BFD会话Down行为 6](#_Toc5775)

[3. 功能需求 7](#_Toc7050)

[3.1 软件模块基本功能需求 7](#_Toc18399)

[3.1.1 Enable/Disable RIP与BFD联动 7](#_Toc5823)

[3.2 软件模块配置功能需求 8](#_Toc23923)

[3.2.1 ip rip bfd-enable 8](#_Toc8846)

[3.3 软件模块信息显示功能需求 9](#_Toc11783)

[3.3.1 show running-config 9](#_Toc18410)

[3.4 软件模块Debug显示功能需求 10](#_Toc8008)

[3.4.1 debug ip rip message 10](#_Toc23627)

[3.5 软件模块管理配置功能需求 11](#_Toc1375)

[3.6 对SNMP的支持 11](#_Toc11644)

[4. 接口需求 12](#_Toc16394)

[4.1 模块使用的服务接口 12](#_Toc18653)

[4.1.1 CMD模块 12](#_Toc2399)

[4.2 模块提供的服务接口 12](#_Toc8599)

[4.2.1 RIP模块 12](#_Toc19363)

[5. 数据结构和算法 14](#_Toc13721)

[5.1 数据结构 14](#_Toc16091)

[5.1.1 数据结构的的组织意图以及之间的关系 14](#_Toc3927)

[5.1.2 内部数据结构 14](#_Toc21516)

[5.1.3 外部可见数据结构 16](#_Toc12960)

[5.2 主要算法 16](#_Toc25405)

[5.2.1 bfd会话的建立 16](#_Toc13576)

[5.2.2 bfd会话的销毁 16](#_Toc31413)

[5.2.3 bfd通告rip进程链路状态down的处理 16](#_Toc1716)

[5.2.4 bfd通告rip进程链路状态up的处理 17](#_Toc24850)

[5.3 过程设计 18](#_Toc22166)

[5.3.1 register the bfd-enable cmd 18](#_Toc10535)

[5.3.2 rip inft bfd-enable submodule 19](#_Toc5451)

[6. 附录 21](#_Toc28104)

[6.1 参考文献 21](#_Toc29764)

图形列表

采用Caption段落风格的新图将自动加入下表。更新请按F9。

# 概述

## 模块概述

RIP是一种基于距离矢量的动态路由协议，因其简单易用而被广泛应用于中小型网络。然而，当链路出现故障时，RIP存在诸多局限性。比如，当链路发生中断或者RIP端口发生故障down时，RIP需通过触发更新定时器（3~5秒）来宣告不可达路由，致使故障处理滞后。由于RIP缺乏主动的故障检测机制，无法实时感知链路状态变化，这可能导致在故障期间仍使用无效路由，从而引发数据丢失。

BFD是一种能够快速检测链路可达性和状态变化的协议，能够在毫秒级内感知链路中断或恢复，为路由协议提供实时的链路状态反馈。将RIP与BFD联动，可以显著提升RIP的故障检测能力和路由收敛速度，从而增强网络的可靠性与稳定性。

RIP与BFD联动功能旨在将BFD绑定到RIP运行的端口，通过BFD快速检测邻居链路状态的变化，并及时通知RIP进程。这样可以确保RIP在链路故障发生时迅速调整路由。该功能不仅能缩短网络中断时间，还能有效提高网络的可用性和整体性能。

## 文档描述

1. 概述 对RIP与BFD进行概述性描述。
2. 重要概念 对BFD相关概念、单跳检测和RIP处理BFD会话Down行 为进行了介绍。

第三章 功能需求 阐明了RIP与BFD联动应该实现的基本功能。

第四章 接口需求 描述了外部模块提供的接口和RIP与BFD联动提供给外部的 接口。

第五章 数据结构和算法 描述了主要数据结构的定义和主要算法。

第六章 附录 列出了相关参考文献。

## 术语说明

RIP

Routing Information Protocol的缩写，一种基于距离矢量的动态路由协议。

BFD

Bidirectional Forwarding Detection的缩写，一种链路状态检测协议。

## 应用范围

该功能适用于在中小型网络中运行RIP协议的设备通过BFD实现快速链路状态检测。在链路中断时，BFD能及时感知故障并通知RIP进程，使其快速收敛，从而有效缩短网络中断时间，提升网络可靠性。

在对网络性能和稳定性要求较高的场景（如企业网络或数据中心），该功能能够显著增强链路故障的检测与处理能力。同时，在与OSPF、BGP等协议协同工作时，BFD与RIP的联动可以进一步优化路由性能，提高整个网络的运行效率。

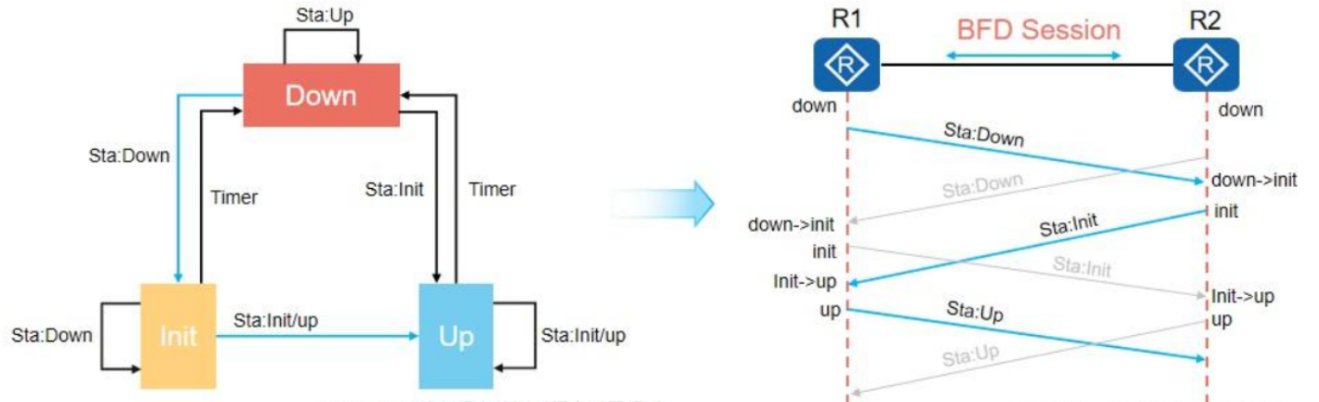
# 重要概念

## BFD相关概念

BFD是一种与协议无关、介质无关的故障检测机制，用于快速检测、监控网络中链路或者IP路由转发连通状况。

### BFD状态机

BFD状态机的建立和销毁都采用三次握手机制：



BFD会话状态的变化是通过BFD控制报文的State字段传递的，发送时通过报文的State字段来宣告本端的状态，接收时根据收到报文的State来获取远端的状态。

·若处于Down状态时：

1. 接收到Down包，会话状态：Down => Init
2. 接收到Init包，会话状态：Down => Up
3. 接收到Up包，会话状态：Down => Down

·若处于Init状态（会话正在建立）：

1. 接收到Init或Up包，会话状态：Init => Up
2. 接收到Down包，会话状态：Init => Init
3. 若BFD控制报文检测时间定时器超时，会话状态：Init => Down

·若处于Up状态（会话建立成功）：

1. 接收到Init包或Up包，会话状态：Up => Up
2. 接收到Down包或者检测超时，会话状态：Up->Down

### BFD检测模式

#### 异步模式

开启BFD的设备之间相互周期性发送BFD控制报文，如果某台设备在指定检测时间内未收到对端发来的控制报文，就宣布会话状态为Down。

#### 查询模式

该模式在成功建立完会话，且链路状况正常情况下使用，用于减少控制报文分发送频率，从而降低网络流量和设备的处理开销，即仅在必要时才会发送控制报文，通过设置Demand位为1，表示设备处于查询模式。

两种模式的本质区别在于检测位置不同，异步模式检测远端位置，检测对端是否周期性发送BFD控制报文；而查询模式检测本端，检测本端发送的BFD控制报文是否得到了响应。

### BFD Echo功能

BFD Echo功能也称为BFD回声功能，是指由本地发送BFD Echo报文，再由远端将此报文进行环回，以判断链路的连通性。该功能适用于在不支持BFD的设备（但支持网络层转发）或者不方便配置BFD的场景。回声模式可用于异步模式和查询模式。

## 单跳限制

BFD最初是为了解决传统路由协议（如 OSPF、BGP）中的链路故障检测问题，旨在通过实时故障检测和快速通知上层协议来提高网络的稳定性和可靠性。在存在多跳路径的情况下，BFD本身能够支持跨越多跳的链路健康检测。它通过在两台设备之间建立会话，并通过中间设备转发BFD心跳包来进行故障检测，从而能够实时检测整个路径的健康状况。

然而，由于RIP是基于距离矢量的，它只能与物理链路直接相连的邻居设备进行通信，并依赖邻居提供的路由信息。RIP协议的路由更新是基于其邻居的状态进行的，而无法直接感知多跳路径上的其他设备的链路状态。因此，RIP不能快速响应多跳链路的状态变化。如果多跳路径中的某个中间节点发生故障，RIP只能依赖其邻居的路由更新来感知变化，而不能通过BFD直接检测到跨多个跳的链路故障来改变路由信息。因此，RIP与BFD的联动模式只能在单跳网络中生效，无法跨越多跳来实时响应链路故障。

## RIP处理BFD会话Down行为

RIP并不像OSPF或BGP等协议那样动态维护邻居的邻接状态，而是依赖于路由条目的下一跳地址来判断路由的来源。当BFD检测到与某一邻居的会话状态为Down时，会及时通知RIP进程，由RIP进程将从该邻居学到的路由进行删除，来实时更新路由表信息，并通过触发更新定时器将这些不可达路由信息及时宣告给其他邻居设备。

# 功能需求

## 软件模块基本功能需求

### Enable/Disable RIP与BFD联动

* ***功能概述***

RIP与BFD联动功能旨在提高RIP协议的路由收敛速度和故障检测能力。在传统的RIP协议中，由于依赖周期性更新和超时机制，链路故障可能导致较长的网络中断时间。通过启用BFD双向转发检测功能，RIP可以实现更快速的链路状态检测，及时感知链路故障，并快速调整路由，从而显著提升网络的可靠性与稳定性。此功能通过在Vx交换机的VLAN配置模式下通过bfd enable命令来开启BFD功能，并通过 ip rip bfd-enable 命令开启RIP与BFD的联动。

* ***激励/响应序列***
* 进入VLAN配置模式，配置bfd enable，开启BFD检测功能。
* 进入VLAN配置模式，配置**ip rip bfd-enable**，开启RIP与BFD联动。
* 进入VLAN配置模式，配置**no** **ip rip bfd-enable**，关闭RIP与BFD联动。

#### **Enable RIP and BFD Interworking on interface**

* ***功能定义***

端口上开启RIP与BFD联动功能。

* + ***输入***

通过在VLAN模式下配置联动，如:

Switch\_config\_v10#ip rip bfd-enable

* + ***处理描述***
* 如果端口已经设置了bfd\_enable\_flag标志，则直接返回。
* 否则，设置该端口bfd\_enable\_flag为1，并将bfd通知RIP进程的回调函数注册入BFD模块中。
  + ***输出***
* 无
  + ***出错处理***
* 无
* ***相关属性***

无

#### **Disable RIP and BFD Interworking on interface**

* ***功能定义***

端口上关闭RIP与BFD联动功能。

* + ***输入***

通过在VLAN模式下取消配置联动，如:

Switch\_config\_v10#no ip rip bfd-enable

* + ***处理描述***
* 如果端口没有设置了bfd\_enable\_flag标志，则直接返回。
* 否则，设置该端口bfd\_enable\_flag为0，并将bfd通知RIP进程的回调函数从BFD模块中注销。如果端口状态为RIP\_INTF\_LINK\_UP，需要重新在端口上激活RIP。
  + ***输出***
* 无
  + ***出错处理***
* 无
* ***相关属性***

无

## 软件模块配置功能需求

### ip rip bfd-enable

***ip rip bfd-enable***

***no ip rip bfd-enable***

* **命令描述**

设置RIP协议下的端口与BFD进行绑定。

no掉***ip rip bfd-enable***，可取消该端口与BFD的绑定。

* **参数说明**

| 名称 | 中英文提示 | 说明 |
| --- | --- | --- |
| ip rip bfd-enable | -- Enable RIP and BFD Interworking  -- 开启RIP与BFD的联动功能 | 配置了这个命令后，当BFD检测到该端口与其直连邻居连接的链路发生中断或恢复时，BFD会立即将链路状态变化通知给RIP进程，从而触发RIP快速进行路由调整 |

* **命令模式**

VLAN配置模式

* **默认值**

无

* **执行结果**

配置运行RIP协议的端口与BFD绑定功能后，链路状态变化（如 link down 或 link up）的触发函数会被注册到BFD模块。当BFD检测到与其邻居的链路状态发生down或up时，会立即通知RIP进程。随后，RIP根据链路状态的变化进行相应的路由调整，以确保路由的快速收敛。

no掉以后，即使链路状态发生改变，也不会触发该函数。

* **注意事项**

无

* **示例**

Switch\_config\_v10#no ip rip bfd-enable

## 软件模块信息显示功能需求

### show running-config

***show running-config***

***show running-config interface vlaN <num>***

* **命令描述**

执行show running-config命令，可显示当前VLAN的相关配置。若开启了RIP与BFD联动，则会显ip rip bfd-enable标识；否则，默认不显示。

* **参数说明**

| 名称 | 中英文提示 | 说明 |
| --- | --- | --- |
| **show *running-config*** | -- Current configuration  -- 当前配置信息 | 查看此交换机全部的配置信息 |
| **show *running-config interface vlaN <num>*** | -- VLAN Interface current configuration  -- VLAN接口的当前配置 | 查看指定VLAN接口的相关配置信息 |

* **执行结果**

Switch\_config#show running-config

Building configuration...

Current configuration:

!

.........

!

interface GigaEthernet0/2

switchport pvid 10

!

!

interface VLAN10

ip address 192.168.10.2 255.255.255.0

no ip directed-broadcast

ip rip 1 enable

ip rip bfd-enable

bfd enable

!

!

vlan 1-2,10,20

!

!

router rip 1

version 2

no auto-summary

!

Switch\_config#show running-config interface vlaN 10

Building configuration...

Current configuration:

!

interface VLAN10

ip address 192.168.10.2 255.255.255.0

no ip directed-broadcast

ip rip 1 enable

ip rip bfd-enable

bfd enable

!

* **注意事项**

|  |  |
| --- | --- |
| Field | Description |
|  |  |

## 软件模块Debug显示功能需求

### debug ip rip message

* **命令描述**

执行debug ip rip message命令后，可以实时跟踪RIP进程的Message消息。当向BFD注册或注销回调函数时，系统应提供相应提示，便于管理员了解配置的状态。当BFD检测到邻居链路状态发生变化时，会触发RIP进程注册的回调函数。在该回调函数中，需明确记录具体收到BFD Up或Down消息的接口编号，方便管理员能够快速定位故障点或确认链路恢复状态。

* **参数说明**

| 名称 | 中英文提示 | 说明 |
| --- | --- | --- |
| ***debug ip rip message*** | -- Debug RIP packet information  -- 追踪RIP消息信息 | 用于追踪RIP的Message消息 |

* **执行结果**
* **注意事项**

|  |  |
| --- | --- |
| Field | Description |
|  |  |

## 软件模块管理配置功能需求

无

## 对SNMP的支持

无

# 接口需求

## 模块使用的服务接口

### CMD模块

#### 提供统一的处理命令的函数接口

| 用途 | 接口说明 |
| --- | --- |
| 当argc不为0时，如果argv[0]为”?”则提示结束命令行，否则提示命令参数过多 | int cmdend(int argc, char \*argv[], struct user \*u); |
| 判断命令前是否有no和default关键字 | TypeOfFunc(u) /\* 宏 \*/ |

## 模块提供的服务接口

### RIP模块

#### 提供开启/关闭RIP与BFD联动的函数接口

**需求：**

**当**在VLAN配置模式下，输入ip rip bfd-enable命令，开启RIP与BFD联动功能。

当在VLAN配置模式下，输入no/default ip rip bfd-enable命令，关闭RIP与BFD联动功能。

**函数接口：**

int rip\_cmd\_intf\_bfd\_enable(int argc, char \*\*argv, struct user \*u);

**输入：**

- argc: 命令行参数个数

- argvs：命令行参数

- u：命令用户

**返回**：

执行结果

#### 提供RIP端口注册/注销BFD的函数接口

**需求：**

在运行RIP协议的端口上，支持与BFD模块的绑定与解绑操作。通过绑定，可以将链路状态发生变化后触发的回调函数注册到BFD模块，确保在链路状态发生变化时，BFD能够及时通知RIP进程进行路由调整；而通过解绑，则会将该回调函数从BFD模块中注销，停止将链路状态变化的通知传递给RIP进程。

**函数接口：**

int rip\_intf\_bfd\_register(uint32 device\_index, uint32 peer\_addr, uint8 state);

**输入：**

- device\_index: 端口索引

- peer\_addr：

Not 0：某个peer的ip地址

0：代表所有peer

- state:

1：注册

0：注销

**返回**：

执行结果

#### 提供RIP供BFD触发的函数接口

**需求：**

当BFD检测与邻居直连链路状态发生变化时，调用RIP进程提供的回调函数，以快速做出路由调整。

当检测到bfd会话状态down了，可能是与邻居的直连链路断开了、邻居的端口地址发生了改变或者端口down了：若此端口的RIP协议状态是ACTIVE，则需删除端口从peer学习到的路由，并广播/组播发送请求报文，请求其他peer的路由表信息（可能次优路由被替换了）。

当检测bfd会话状态up了，则可能是与邻居的直连链路恢复、邻居端口地址变化后又重建了会话或者端口重新启动了：将端口状态置为RIP\_INTF\_PROCESS\_ACTIVE，并将此端口设置为能够收发组播报文、添加直连路由，再发送请求报文，请求邻居设备的整个路由表信息。

**函数接口：**

int rip\_intf\_bfd\_callback(uint32 src\_ip, uint32 dst\_ip, uint32 connected, uint32 down);

**输入：**

- src\_ip: 源ip地址

- dst\_ip: 目的ip地址

- connected: 0为直连

- down:

1：表示bfd会话状态为up

0：表示bfd会话状态为down

**返回**：

执行结果

# 数据结构和算法

## 数据结构

### 数据结构的的组织意图以及之间的关系

略

### 内部数据结构

#### rip\_intf\_

##### **基本结构**

/\*RIP端口信息\*/

typedef struct rip\_intf\_

{

uint32 vrf\_id;/\*端口对应的VRF\*/

uint32 process\_id;/\*被使能的进程号\*/

uint32 device\_index; /\* 接口ID\*/

struct rip\_process\_info\_ \*pprocess;

/\* int ref\_num;/\*该端口被RIP网络覆盖的次数\*/

uint32 encap\_type; /\* 接口封装类型\*/

uint8 state;/\*端口链路状态\*/

uint8 addr\_type;/\*端口地址类型\*/

uint32 address;/\* 接口IP地址\*/

uint32 mask; /\* 接口IP地址掩码\*/

uint32 aid; /\* address id ，当在接口配置多个地址时，表示其id\*/

char intf\_name[ INTERFACE\_DEVICE\_MAX\_NAME];/\*端口名字\*/

uint8 split\_flag;/\*标志位SPLIT\_HORIZON\_FLAG\*/

uint8 special\_flag;/\*passive,notreceive,v1demand,v2demand--dangzhw,20091111\*/

uint8 send\_version;/\*enum RIP\_SEND\_VERSION\*/

uint8 recv\_version;/\*enum RIP\_RECV\_VERSION\*/

uint8 auth\_type;/\*端口认证类型\*/

uint8 md5\_type;/\*md5 key的存储类型\*/

char simple\_key[16+1];

char md5\_key[16+1];

uint8 md5\_keyid;

struct rip\_route\_ \*connect\_route;/\*端口对应的直连路由\*/

struct rip\_route\_list\_ nbr\_route\_list;/\*该端口学习到的路由链表\*/

uint rip2IfStatRcvBadPackets;/\*端口接收到的错误报文数\*/

uint rip2IfStatRcvBadRoutes;/\*端口接收到的错误路由数\*/

uint rip2IfStatSentUpdates;/\*端口发送的更新报文数\*/

uint rip2IfStatRecvUpdates;/\*端口收到的更新报文数\*/

int nbr\_route\_num;

uint32 auth\_commit\_timer\_id;

uint32 auth\_commit\_time;

/\* Commented by dangzw in 2009.03.04 19:10:01 \*/

uint8 dynamicflag;

uint32 sequence;/\*序列号\*/

uint32 key\_start\_timer\_id;/\*key开始生效定时器\*/

uint32 key\_lift\_timer\_id;/\*key有效时间长度定时器\*/

struct dynamic\_key\_list\_ key\_list;/\*邻居的key链表\*/

struct dynamic\_key\_list\_ key\_timeout\_list;/\*邻居间超时的key链表\*/

uint8 bfd\_enable\_flag; /\* 端口enable bfd，0:disable 1:enable \*/

}rip\_intf\_t;

##### 用途

定义RIP端口的数据结构，用来存储RIP端口上绑定的配置信息。

##### 操作函数

##### 存储方式

全局结构变量

##### 主要变量

struct rip\_intf\_ \*rip\_intf\_array[INTERFACE\_DEVICE\_MAX\_NUMBER + 1];

##### 初值及取值范围

取值范围参见结构部分的注释

#### ip rip bfd-enable

/\* 注册端口下的与bfd联动开关 \*/

{

"bfd-enable",

MATCH\_AMB,

cmdPref(PF\_CMDNO, PF\_CMDDEF, 0),

0,

rip\_cmd\_intf\_bfd\_enable,

NULL,

NULL,

0,

0,

"bfd-enable -- Bind bfd on the interface",

"bfd-enable -- 在端口上绑定关注bfd状态",

NULLCHAR,

NULLCHAR

},

#### 主要宏和常量定义

无

### 外部可见数据结构

#### 包含内容

无

#### 分析及备注

## 主要算法

### bfd会话的建立

RIP与BFD联动功能开启后，会根据peer与邻居建立bfd会话，共存在两种情况：第一种情况，peer还未存在时：当该端口首次接收到peer发来的数据包时，会新建一个peer，如果端口上开启了RIP与BFD联动功能（bfd\_enable\_flag为1），则可与该peer建立新的bfd会话，并将bfd通告rip进程链路状态的触发函数注册到bfd模块中；第二种情况，peer已存在，由于这些peer信息是存储在RIP进程对象的peer\_list中，需遍历进程对象的peer\_list链表，找到与该端口通信的peer，并与该peer建立bfd会话，将bfd通告rip进程链路状态的触发函数注册到bfd模块中。

### bfd会话的销毁

由于bfd会话是基于peer建立的，当peer被删除时，如果开启了RIP与BFD联动功能，需要将与该peer的bfd会话关闭，并将bfd邻居从端口的rip\_bfd\_peer\_list\_中清除。peer被删除的情况有两种，一种是被peer超时定时器删除了，即当距离上次收到该peer的时间间隔大于等于3\*peer超时时间，或者peer没有被network覆盖或不是neighbor；另一种是当某个端口被删除了，则需要将端口对应的所有peer都删除。

/\* 伪代码 state: 1 注册，state:0 注销\*/

ret = bfd\_proto\_announce(pintf->address, peer\_addr, 0, RTPROTO\_RIP, pintf->process\_id, !state, rip\_intf\_bfd\_callback);

if(ret < 0){

rip\_debug(RIP\_DEBUG\_IP\_RIP\_RETURN,rip\_intf\_bfd\_register,device\_index=%d,state=%d fail! \n", device\_index, state);

return RIP\_FAIL;

}

### bfd通告rip进程链路状态down的处理

由于通过bfd\_proto\_announce函数向bfd模块注册RIP的回调函数涉及4个参数，分别是src\_ip、dst\_ip、connected和down，其中并不存在端口的device\_id，所以需要遍历全局RIP进程链表，根据src\_ip从进程对象的端口链表中查询到对应的端口对象。

如果bfd检测到会话状态down：删除端口从peer学习到的路由，再发送请求报文，请求其他peer的路由表信息。

### bfd通告rip进程链路状态up的处理

同理5.2.3，根据src\_ip从进程对象的端口链表中查询到对应的端口对象。

若bfd检查到与某个直连peer的链路up了并且端口状态是RIP\_INTF\_PROCESS\_ACTIVE，则单播请求peer的路由表信息。

/\* 伪代码\*/

int rip\_intf\_bfd\_callback(uint32 src\_ip, uint32 dst\_ip, uint32 connected, uint32 down)

{

/\* 1. 遍历全局RIP进程链表，根据src\_ip查询到进程的某个端口对象 \*/

......

/\* 2. 链路down或up处理逻辑 \*/

/\*端口down\*/

if(!down){

/\* 若当前端口非进程激活端口，不需要操作\*/

if(BIT\_TEST(pintf->state , RIP\_INTF\_PROCESS\_ACTIVE))

{

/\*删除端口从peer学习到的路由\*/

ret = rip\_clear\_peer\_route\_list(device\_index, dst\_ip);

if (RIP\_SUCCESS != ret)

{

rip\_debug(RIP\_DEBUG\_IP\_RIP\_RETURN, "RIP: %s %d.\n", \_\_FILE\_\_,\_\_LINE\_\_);

return RIP\_FAIL;

}

/\*发送request报文,请求其他peer的路由信息（可能次优路由被替换了）\*/

ret = rip\_send\_request(device\_index);

if (RIP\_SUCCESS != ret)

{

rip\_debug(RIP\_DEBUG\_IP\_RIP\_RETURN, "RIP: %s %d.\n", \_\_FILE\_\_,\_\_LINE\_\_);

return RIP\_FAIL;

}

}

}

else{/\*端口up\*/

if((!pintf) || (pintf->address == 0) || (!BIT\_TEST(pintf->state ,RIP\_INTF\_LINK\_UP)))

return RIP\_FAIL;

if(BIT\_TEST(pintf->state ,RIP\_INTF\_PROCESS\_ACTIVE)){

/\*端口发送request单播报文\*/

ret = rip\_send\_request\_to\_peer(device\_index, dst\_ip);

if( RIP\_SUCCESS != ret )

{

rip\_debug(RIP\_DEBUG\_IP\_RIP\_RETURN, "RIP: %s %d.\n", \_\_FILE\_\_,\_\_LINE\_\_);

return RIP\_FAIL;

}

}

}  
 return RIP\_SUCCESS;

}

## 过程设计

### register the bfd-enable cmd

#### rip\_cmd\_intf\_bfd\_enable

1. 概述

执行ip rip bfd-enable命令后的解析函数:：

* 1. 命令行参数边界检查；
  2. 端口对象非空检查；
  3. 根据命令类型做不同处理；
     1. 若执行的是ip rip bfd-enable命令，判断是否已经开启了rip与bfd联动，若没有开启，则需向bfd模块注册关注回调，并将bfd\_enable\_flag标识位置为1。
     2. 若执行的是no/default ip rip bfd-enable命令，判断是否未开启rip与bfd联动，若已经开启了，则需从bfd模块注销关注回调，并将bfd\_enable\_flag标识位置为0；如果rip端口实际状态是up，需要激活端口上的rip进程。

1. 接口描述 输入输出

int rip\_cmd\_intf\_bfd\_enable(int argc, char \*\*argv, struct user \*u)

输入参数：argc/argv:命令行参数

u:执行命令的用户

输出参数： 无

返回值 : 处理成功：RIP\_SUCCESS

处理失败：RIP\_FAIL

1. 处理描述

/\* 往bfd注册关注回调 \*/

case NORMAL\_FUNC:

if(pintf->bfd\_enable\_flag == 1)

break;

/\* 往bfd注册关注回调 \*/

ret = rip\_intf\_bfd\_register(device\_index, 0, 1);

if(ret != 0){

return RIP\_FAIL;

}

pintf->bfd\_enable\_flag = 1;

break;

case NOPREF\_FUNC:

case DEFPREF\_FUNC:

if(!pintf->bfd\_enable\_flag)

break;

/\* 注销注册回调\*/

ret = rip\_intf\_bfd\_register(device\_index, 0, 0);

if(ret != 0){

return RIP\_FAIL;

}

rip\_intf\_array[device\_index]->bfd\_enable\_flag = 0;

/\*端口实际状态up，rip端口需up\*/

if(BIT\_TEST(rt\_get\_if\_status(device\_index), RIP\_INTF\_LINK\_UP)){

rip\_debug(RIP\_DEBUG\_IP\_RIP\_MSG, "rip intf RIP\_INTF\_LINK\_UP no use bfd\n");

if(BIT\_TEST(pintf->state , RIP\_INTF\_PROCESS\_ACTIVE))

return RIP\_SUCCESS;

/\* 打上进程端口激活的标志\*/

BIT\_SET(pintf->state , RIP\_INTF\_PROCESS\_ACTIVE);

ret = rip\_process\_intf\_activate(device\_index , TRUE);

}

break;

1. 错误处理

无

1. 注释/限制/局限

无

1. 设计语言和文件描述

rip\_cmd.c

1. 涉及资源

无

### rip inft bfd-enable submodule

#### **rip\_intf\_bfd\_register**

1. 概述

向bfd模块注册回调或从bfd模块注销回调的函数:：

* 1. 端口索引的检查；
  2. 端口对象非空检查；
  3. 根据不同的peer\_addr值进行不同处理
     1. 若peer\_addr不为0，如果是注册操作，则与指定peer建立bfd会话；如果是注销操作，将其从bfd模块中注销。
     2. 若peer\_addr为0，如果是注册操作，与RIP进程全局对象的peer\_list中每个满足条件的peer对象建立bfd会话；如果是注销操作，将其从bfd模块中注销。

1. 接口描述 输入输出

int rip\_intf\_bfd\_register(uint32 device\_index, uint32 peer\_addr, uint8 state)

输入参数：device\_index:端口索引

peer\_addr:peer的ip地址，若该值为0，代表所有peer

state: 1代表注册，0代表注销

输出参数： 无

返回值 : 处理成功：RIP\_SUCCESS

处理失败：RIP\_FAIL

1. 处理描述

/\* 注册指定peer的bfd会话 \*/

if(peer\_addr != 0){

ret = bfd\_proto\_announce(pintf->address, peer\_addr, 0, RTPROTO\_RIP, pintf->process\_id, !state, rip\_intf\_bfd\_callback);

if(ret < 0){

rip\_debug(RIP\_DEBUG\_IP\_RIP\_RETURN,rip\_intf\_bfd\_register,device\_index=%d,state=%d fail! \n", device\_index, state);

return RIP\_FAIL;

}

return RIP\_SUCCESS;

}

peer = pprocess->peer\_list.forw;

while(peer != &(pprocess->peer\_list))

{

if(peer->peer\_intf == device\_index)

{

ret = bfd\_proto\_announce(pintf->address,peer->peer\_addr, 0, RTPROTO\_RIP, pintf->process\_id, !state, rip\_intf\_bfd\_callback);

if(ret < 0){

rip\_debug(RIP\_DEBUG\_IP\_RIP\_RETURN,rip\_intf\_bfd\_register,device\_index=%d,state=%d fail! \n", device\_index, state);

return RIP\_FAIL;

}

peer = peer->forw;

}

1. 错误处理

无

1. 注释/限制/局限

无

1. 设计语言和文件描述

rip\_inft.c

1. 涉及资源

无

# 附录

## 参考文献

1. RIP Version 2.（RFC2453）

[2]Bidirectional Forwarding Detection (BFD).（RFC5880）

[3]Generic Application of Bidirectional Forwarding Detection (BFD).（RFC5882）