**网络协议栈分析与设计课程大作业**

|  |
| --- |
| **IS-IS路由协议代码分析**  （仅供参考，分析可以更详细点） |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | 学号 | 姓名 | 班级 | 负责模块 | 成绩 | | 201492\*\*\* | 裴\*\* | 软网1\*\*1 | 1.代码介绍  2.子网相关功能  3.PDU的处理  ，选择局域网DIS  4.排版 |  | | 201492\*\* | 王\*\* | 软网1\*\*3 | 1.引言  2.子网独立功能  3.链路状态机  ，动态主机名  4.文献搜集 |  | | 201492\*\* | 刘\*\* | 软网1\*\*3 | 1.数据结构  2.PDU结构与编码  3.SPT算法的实现  ，更新路由  4.总结 |  | |

目录

[摘要 2](#_Toc470863378)

[Abstract 3](#_Toc470863379)

[第一章 引言 4](#_Toc470863380)

[第二章 代码介绍 5](#_Toc470863381)

[2.1 符号与缩写 5](#_Toc470863382)

[2.2 文件介绍 6](#_Toc470863383)

[2.3 全局变量 7](#_Toc470863384)

[2.4 统计量 7](#_Toc470863385)

[第三章　数据结构 8](#_Toc470863386)

[3.1 struct isis 8](#_Toc470863387)

[3.2 struct isis\_area 9](#_Toc470863388)

[3.3 struct isis\_circuit 10](#_Toc470863389)

[3.4 struct isis\_lsp 11](#_Toc470863390)

[3.5 struct isis\_adjacency 12](#_Toc470863391)

[第四章 IS-IS 路由协议的实现 13](#_Toc470863392)

[4.1 PDU的处理 13](#_Toc470863393)

[4.1.1 概述 13](#_Toc470863394)

[4.1.2 isis\_handle\_pdu 函数 15](#_Toc470863395)

[4.1.3 tlv相关函数 18](#_Toc470863396)

[4.2 选择局域网DIS 20](#_Toc470863397)

[4.2.1 概述 20](#_Toc470863398)

[4.2.2 run\_dr\_elect 函数 21](#_Toc470863399)

[4.2.3 run\_dr\_resign 函数 24](#_Toc470863400)

[4.2.4 run\_dr\_commence 函数 25](#_Toc470863401)

[4.3 链路状态机 27](#_Toc470863402)

[4.1 概述 27](#_Toc470863403)

[4.2 isis\_csm\_stat\_change 函数 28](#_Toc470863404)

[4.4 动态主机名 29](#_Toc470863405)

[4.4.1 概述 29](#_Toc470863406)

[4.4.2 cache相关函数 30](#_Toc470863407)

[4.5 SPT算法的实现 32](#_Toc470863408)

[4.5.1 概述 32](#_Toc470863409)

[4.5.2 add\_to\_path 33](#_Toc470863410)

[4.5.3 isis\_run\_spf函数 34](#_Toc470863411)

[第五章 总结 37](#_Toc470863412)

# 引言

IS-IS协议是一种基于链路状态算法的路由协议，这意味着作为中间系统的路由器，必须完全知晓自己所在区域内部所有其它的路由器和它们的链路状态。IS-IS协议通过发送LSP（链路状态协议数据单元）进行路由信息通告，通过Hello PDU发现邻居和创建邻接，[5]通过SNP（序号协议数据单元）确保路由器持有最新的、有效的LSP。路由器在接收到的LSP的基础上，通过SPF（最短路径优先）算法最终确定路由表。

IS-IS协议把路由器分为三种：L1路由器、L2路由器和L1/L2路由器。其中，L1路由器是为端系统（比如个人电脑）直接提供服务的路由器，L2路由器是骨干路由器。L1/L2路由器向L1路由器隐藏骨干区域的细节，并负责将由L1路由器组成的非骨干区域的数据请求转发到由L2路由器组成的骨干区域。和OSPF协议直接使用IP地址标识路由器不同，IS-IS协议使用OSI的地址方案NASP标识路由器节点。因此，纯粹的IS-IS协议并不直接适合于IP网络；而集成IS-IS能够实现IP路由的关键，在于RFC 1195引入携带IP路由信息的CLV字段。

标准的IS-IS协议是由国际标准化组织制定的ISO/IEC 10589:2002 所规范的。但是标准的IS-IS协议是为无连接网络服务（CLNS）设计的，并不直接适合于IP网络，因此互联网工程任务组制定可以适用于IP网络的集成化的IS-IS协议，称为集成IS-IS，它由RFC 1195等RFC文档所规范。由于IP网络的普遍存在，一般所称的IS-IS协议，通常是指集成IS-IS协议。

# 第二章 代码介绍

## 2.1 符号与缩写

|  |  |  |  |
| --- | --- | --- | --- |
| 符号 | 说明 | 符号 | 说明 |
| PDU  SNSDU  NSPU  NPDU  SNPDU  ESH PDU  ISH PDU  RD PDU  IIH  LSP  SNP  CSNP  PSNP | 协议数据单元  子网服务数据单元  网络服务数据单元  网络协议数据单元  子网协议数据单元  终端系统呼叫协议数据单元  中间系统呼叫协议数据单元  重定向协议数据单元  IS-IS 呼叫协议数据单元  链路状态协议数据单元  数字序列协议数据单元  完全数字序列协议数据单元  部分数字序列协议数据单元 | DA  DED  DIS  DTE  ES  IS  L1  L2  LAN  MAC  NLPID  PCI  QoS  SN | 动态分配  动态建立数据连接  指定中间系统·  数据终端设备  终端系统  中间系统  Level 1  Level 2  局域网  媒体访问控制  网络层协议标示符  协议控制信息  服务质量  子网 |
| AFI  DSP  IDI  IDP  NET  NSAP  SNPA  TLV  TTY  VTY | 权威的格式化指示器  特殊部分域  初始域标示符  初始域部分  网络实体名称  网络服务访问节点  子网连接节点  标签长度值  电传  虚电传 | SNAcP  SNDCP  SNICP  SRM  SSN  SVC | 子网访问协议  子网相关性集中协议  子网独立性集中性协议  发送路径消息  发送数字序列消息  转变虚流程 |

表2.1 ISIS协议符号缩写说明

## 2.2 文件介绍

ISIS协议是由C语言实现的，头文件加上源文件一共48个，下表给出了主要的文件并进行了说明：

|  |  |
| --- | --- |
| 文件名 | 说明 |
| dict.[ch]  isis\_adjacency.[ch]  isis\_circuit.[ch]  isis\_common.h  isis\_constants.h  isis\_csm.[ch]  isis\_dr.[ch]  isis\_dynhn.[ch]  isis\_events.[ch]  isis\_flags.[ch]  isis\_lsp.[ch]  isis\_main.c  isis\_misc.[ch]  isis\_network.[ch]  isis\_pdu.[ch]  isis\_route.[ch]  isis\_spf.[ch]  isis\_tlv.[ch]  isis\_zebra.[ch]  isisd.[ch]  iso\_checksum.[ch] | Kazlib 红黑树  邻域数据库  链路相关路由  所有模块的定义  一些常量的定义  链路状态机  局域网DIS路由  动态主机名缓存  事件包装器（eg.主机类型更改事件）  SRM和SSN中flags 的实现  链路状态数据库和链路状态路由  路由协议主函数  多方辅助路由  OSI第二网络层I/O  IS-IS 协议包的处理  路由的处理  SPT（最短路径树）算法  TLV 的处理  Zebra接口  顶层数据结构和相关路由  Fletcher 校验和的实现 |

表2.2 ISIS 路由协议主要文件

## 2.3 全局变量

|  |  |  |
| --- | --- | --- |
| 变量 | 数据类型 | 说明 |
| \_caps\_p  isisd\_privs  longopts  config\_default[]  master  pid\_file  ALL\_L1\_ISS  ALL\_L2\_ISS  ALL\_ISS  ALL\_ESS  sock\_buff  isis | struct zebra\_capabilities\_t[]  struct zebra\_privs\_t  struct option  char[]  struct thread\_master\*  char\*  u\_char[]  u\_char[]  u\_char[]  u\_char[]  char[]  struct | Zebra 配置  ISIS的Zebra 配置  ISIS 路由选项  默认配置  主线程  初始化系统所需的进程ID  L1所有的IS  L2所有的IS  系统中所有的IS  系统中所有的ES  socket缓存  当前处理的IS |

表2.3 ISIS项目中的全局变量

## 2.4 统计量

结构esis\_stat中为每个系统（端系统和中间系统）收集的统计量如表2.4.

esis\_stat 描述了一个系统的基本状况，包括系统内存，各中协议收发的分组数等。

|  |  |  |
| --- | --- | --- |
| 变量 | 数据类型 | 说明 |
| es\_nomem  es\_badcsum  es\_badvers  es\_badtype  es\_toosmall  es\_eshsent  es\_eshrcvd  es\_ishsent  es\_ishrcvd  es\_rdsent  es\_rdrcvd | u\_short  u\_short  u\_short  u\_short  u\_short  u\_short  u\_short  u\_short  u\_short  u\_short  u\_short | 为发送hello包保留的内存  校验和错误的分组数  IP版本错误的分组数  PDU类型域不明的分组数  太小的分组数  ESH发送的分组数  ESH接收的分组数  ISH 发送的分组数  ISH 接收的分组数  RD 发送的分组数  RD 接收的分组数 |

表2.4 结构esis\_stat 维护的统计量

# 第三章　数据结构

主要的数据结构和它们之间的关系如下图所示：

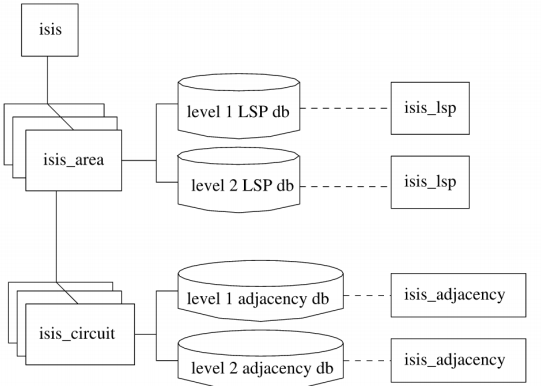
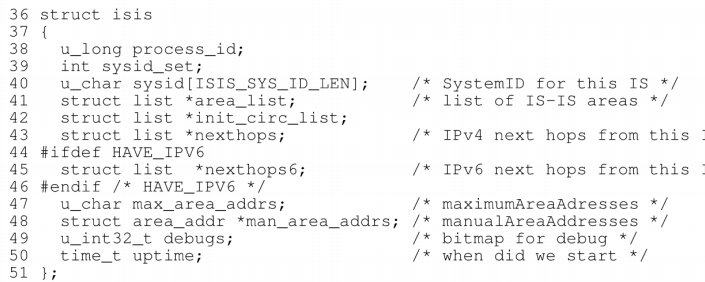


图3.1 isisd的主要数据结构

Isis的结构是上图所示所有的数据结构的起点。每个isisd结构只有其中的一个结构块。Isis struct 有一系列的is-is area。而每个area 都有着一系列的关于area的回路配置，同时每个area都有两级LSP数据库。接下来的部分将会是详尽的分析这些isis结构。

## struct isis

Struct isis如下截图所示：

isisd.h

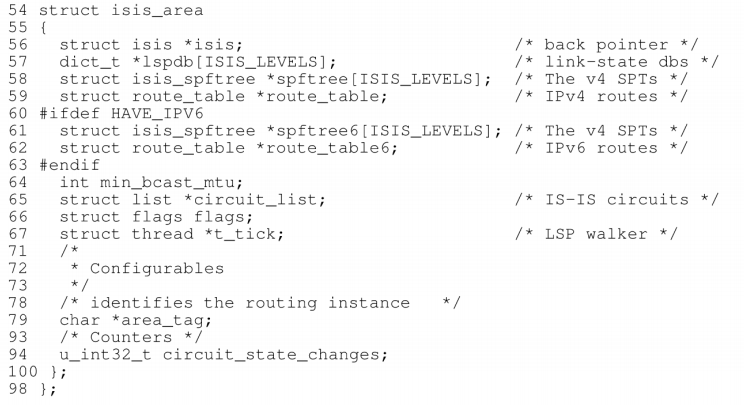
isisd.h

图3.1 struct isisd的代码定义

Struct拥有的是is中的systemid，它由具有固定的6字节长度来实现它（isis\_sys\_id\_len）。这是一个区域地址配置列表，作用是管理以及此处存在最大区域地址参数的值，area\_list拥有所有IS-IS区域。调试的水平被影响变量的命令所控制。下一跳的列表存储在ISIS的结构，而不是isis\_area或isis\_circuit（在接下来将讨论），为节省内存，由于IPv4和IPv6下一跳可以被区别到不同地区，init\_circ\_list拥有的circuit the Zebra的进程列表会通知我们，但它此时尚未配置参加IS-IS路由。

## struct isis\_area

isisd.h



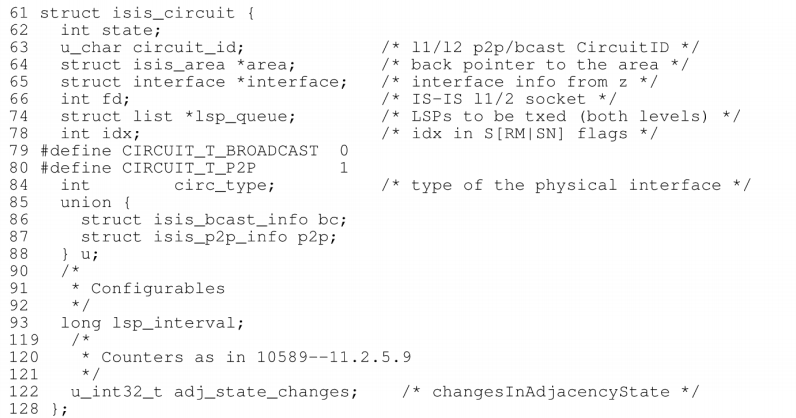
isisd.h

图3.2 struct isis\_area的代码定义

struct isis\_area重要组成部分，如图3.3所示。LSP数据库的每级都由红黑树实现，它是自由的可分配得kazlib软件包的一部分。结构isis\_area拥有SPF树，每个受支持的协议（IPv4和IPv6）,每一级以及IPv4和IPv6路由表来自SPF树。circuit\_list字段指向一个列表。Isis的地区结构有着许多的配置，它们在初始的建立时就被设立了一个默认值，然后根据用户的命令更改。区域实例是用用户指定的区域标签标识。

## struct isis\_circuit

isis\_circuit.h



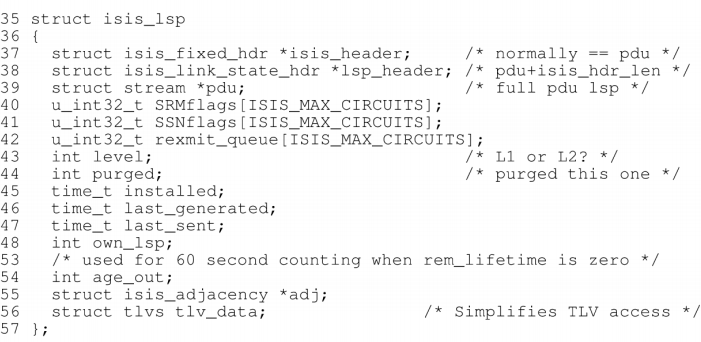
isis\_circuit.h

图3.3 struct isis\_circuit的代码定义

每个circuit参与IS-IS路由提出了ISIS电路结构，其中的重要部分如图3.4所示。该结构是一个接口的指针结构,从Zebra接收，直接映射到物理接口的。socket发送和接收IS-IS PDU以及存在线程控制发送，这是因为lsp\_queue需要发送LSP必须加以控制，使LSPs不能发送更多的参数。circuit只有邻接和线程发送点对点的IIH PDU需要struct ISIS BCAST信息更为复杂。它包含了线程运行并选择局域网DIS，刷新LSPs和发送LAN IIH PD Us。两水平邻接数据库内也设有该结构。这个组合跟着一个数字电路相关的配置和计数器。、

## struct isis\_lsp

isis\_lsp.h



isis\_lsp.h

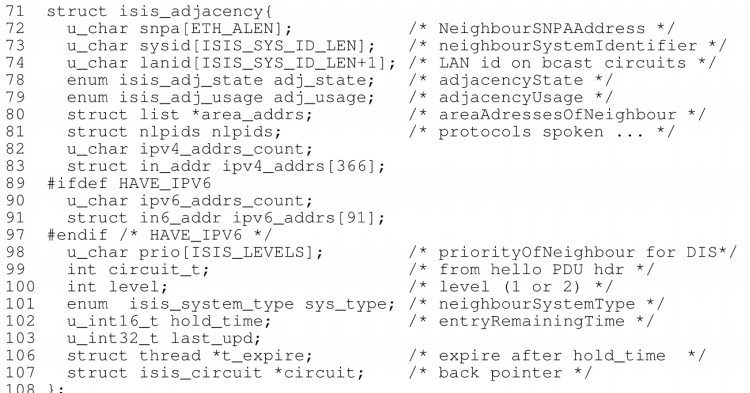
图3.4 struct isis\_lsp的代码定义

每个LSP结构在isis\_ LSP被描述，如图3.5所示。它所实现的想法是，收到的LSP保存来自网络的PDU，然后指向提供了LSP的各个领域(area)。LSP的可变部分可以引用tlv\_data领域，它是一个转化的列表结构。每一个TLU型的列表和列表项点都在PDU处实现转化。

结构isis\_lsp也包括SRM和SSNflags和一个用于接收的指向两者之间的cency LSP。last\_generated用于调试信息和控制再生和发送的LSP。当LSP到期时必须保留一份标题为zeroagelife的时间，在LSP可以完全从数据库中删除。当LSP是由该系统的标志own\_lsp设置生成，然后，adjacency将被设置为空。

## struct isis\_adjacency

isis\_adjacency.h



isis\_adjacency.h

图3.5 struct isis\_adjacency的代码定义

通过结构ISIS adjacency描述adjacency系统，代码部分如图3.6。该结构包含邻居SNPA地址，系统ID和地区地址从IIH PDU的复制。如果IPv4或IM地址目前保存在adjacency。对于adjacency，它将从广播电路网络ID和优先领域进行复制。当前实现在保持时间一用完，将会立即移除adjacency。一个建议是，adjacency将被保存在一个向下的状态一些时间，然后删除了一个特殊的垃圾收集功能。这样它会更容易让网络管理员注意到一个邻居已经下降的事实。Adjacency的状态值保存在在变量adj\_state中。

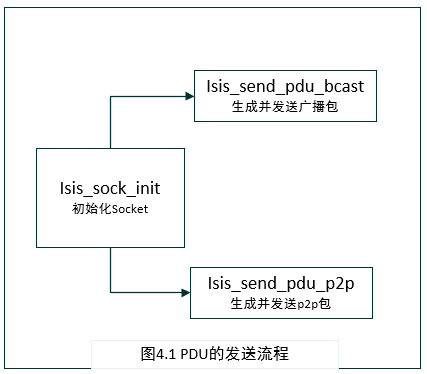
# 第四章 IS-IS 路由协议的实现

## PDU的处理

### 概述

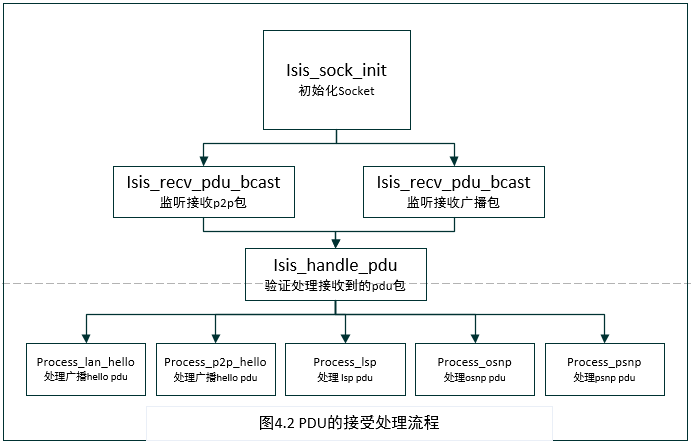
网络中各个节点收发IS-IS PDU的相关实现在isis\_network.[ch] 文件中。

IS-IS PDU 的收发发生在OSI网络协议栈中的第二层——链路层，因此收发双方系统都要预处理这些协议包。如果在GNU/Linux 环境下，首先需要使用系统调用socket() 来创建socket，socket的协议族设置为PF\_PACKET，socket 的类型设置为SOCK\_DGRAM。然后为了能够接受到同层的广播包，还需要使用函数setsockopt() 把socket绑定到物理层的多播组中。PDU的构建发送过程如下图所示



在发送到广播链路之前，IS-IS PDU 需要加上3字节LLC 头字段。这些字段值的定义在ISO 9577 中。一般情况下，当这些协议包的目的是路由时，这些字段的值总是0xFE-FE-03。端系统中的Socket 读出这些PDU 后会把他们传递给定义在isis\_pdu.h 和isis\_tlv.h 的函数处理。

首先，这些PDU会被一个普通的包装函数isis\_handle\_pdu() 处理。这个函数能够处理所有种类的PDU。这个函数的功能就是验证协议包的头部。如果这些PDU通过验证就会根据它自身的种类标示被传递到不同的函数处理。例如Level 1 和Level2 LAN IIH PDU 就会被传递到函数process\_lan\_hello() 中处理。PDU的接收处理函数之间的关系如下图所示：



IS-IS PDU 中值域的处理函数在文件isis\_tlv.[ch]中。IS-IS PDU 中的值域部分是两个32位的地址值（found和expected）。这两个地址值被用来作为每个TLV 类型的位域。所有的TLV 都会被传递到函数parse\_tlvs()处理。这个函数的两个参数是 IS-IS PDU 中的值域部分（两个32位的地址值）和一个指向结构体tlvs的指针。调用者把TLV 设置为它感兴趣的类型，而这个函数则把TLV为预设值为参数IS-IS 值域中found 的部分。结构体 tlvs 包括每种TLV的列表。因此PDU 值域的数据可以通过tlvs 轻松获取。但是只有函数参数中expected 出现的tlv 类型会被返回。

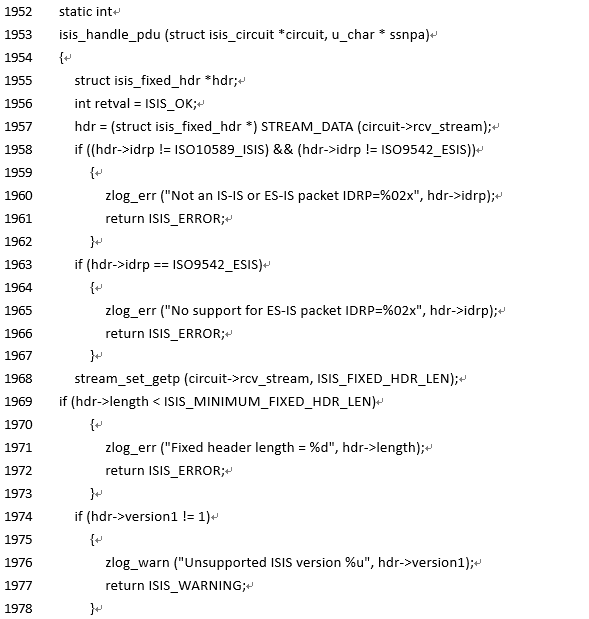
TLVs的写入函数也定义在文件isis\_tlv.[ch] 中。每种TLV 都有不同的写入函数。这些写入函数的参数是TLV列表和一个需要写入的PDU。

接下来我们会详细了解一下函数isis\_handle\_pdu() 和函数 parse\_tlvs()。

### isis\_handle\_pdu 函数

函数isis\_handle\_pdu 针对所有种类的PDU。它的功能就是检验校正每个收到的PDU 并且根据PDU的种类把它传递到特定的处理函数中。

Isis\_pdu.c



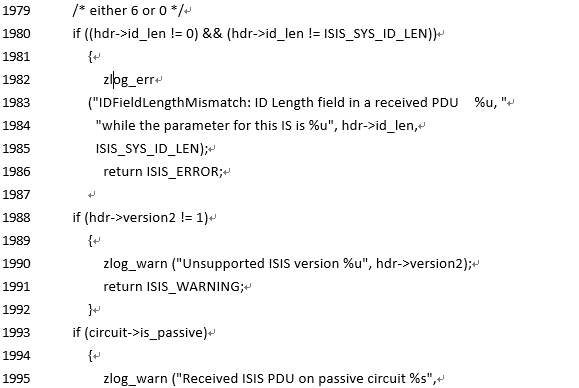
Isis\_pdu.c

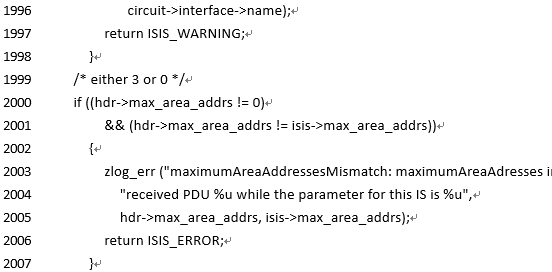
1958 – 1962 验证收到的PDU是否是ISIS 协议包

1963 – 1967 如果PDU是ISO9542， 则报错说明系统不支持该版本协议

1969 – 1978 如果PDU头部长度小于协议要求长度或者协议版本不对， 报错处理。

Isis\_pdu.c





Isis\_pdu.c

1980 – 1981 如果PDU 的id长度不符要求，报错处理

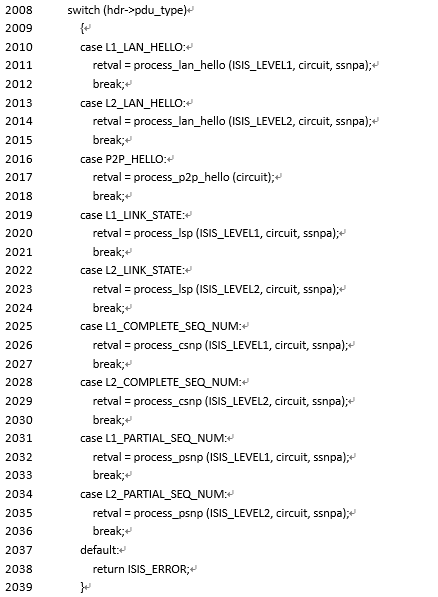
1988 – 1992 如果PDU 版本不对，报错处理

1993 – 1998 如果链路状态是passive， 则把收到的PDU包丢弃处理

2000 – 2007 如果包的地址长度超过要求，则把收到的PDU包丢弃处理

Isis\_handle\_pdu 函数的前半段主要是要证PDU是否符合ISIS协议要求，主要检查是否受损，是否符合版本和协议要求，是否发生包的部分丢失与多出等。

Isis\_pdu.c



Isis\_pdu.c

2008 – 2039 如果isis\_handle\_pdu函数通过检验发现PDU无误的话，则通过判断pdu\_type 的值选择相应处理函数处理该PDU。

2010 – 2015 PDU是局域网内的hello包的话，调用函数process\_lan\_hello处理，并根据发 送方局域网的级别把第一个参数设置为ISIS\_LEVEL1或者ISIS\_LEVEL2。

2016 – 2018 PDU是点对点的hello包的话，调用函数process\_p2p\_hello处理。

2019 – 2024 PDU 是链路状态包的话，调用函数process\_lsp 处理，并根据发送方局域网的 级别把第一个参数设置为ISIS\_LEVEL1或者ISIS\_LEVEL2。

2025 – 2030 PDU是完全有序包的话，调用函数process\_csnp处理，并根据发送方局域网的 级别把第一个参数设置为ISIS\_LEVEL1或者ISIS\_LEVEL2。

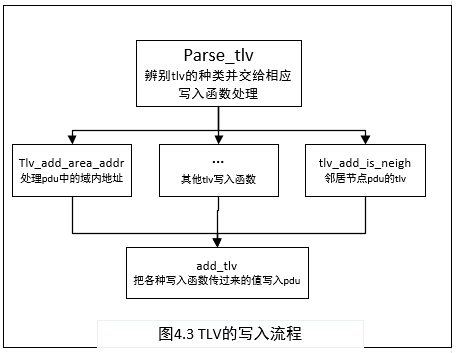
2031 – 2036 PDU是部分有序包的话，调用函数process\_psnp处理，并根据发送方局域网的 级别把第一个参数设置为ISIS\_LEVEL1或者ISIS\_LEVEL2。

2037 – 2038 如果PDU的pdu\_type 不等于以上所有类型，则返回错误。

### tlv相关函数

pdu中tlv 的处理文件isis\_tlv.[ch]中。

如下图所示：系统首先会调用parse\_tlvs 函数辨别通过参数传递过来的tlv种类，然后调用相应的写入函数进行处理。



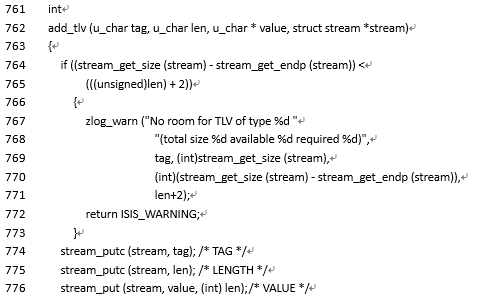
**parse\_tlvs函数**

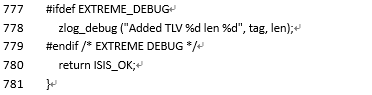
parse\_tlvs 函数的功能是处理系统socket接收到的IS-IS PDU 的值域。调用函数设置expected的相关位来通知parse\_tlvs函数希望保留的TLV 种类，parse\_tlvs 就会使用相应写入函数把相应种类的tlv 保存在结构体tlvs 中。

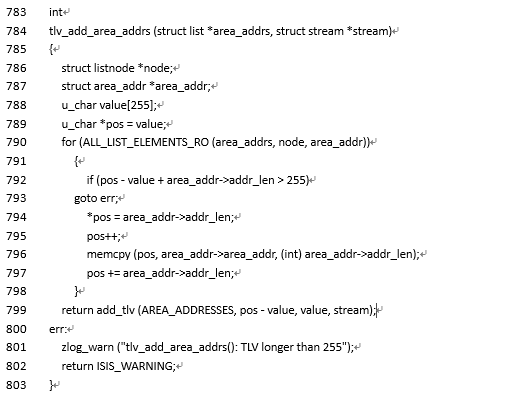
**add\_tlvs 函数**

tlvs的写入函数有很多种，parse\_tlvs()读出PDU类别后会根据其类别调用不同的tlvs写入函数。add\_tlvs 是最基本的写入函数，每种不同的写入函数都会调用add\_tlvs函数写入tlvs。以下是 Area Addresses TLV 的写入函数。

isis\_tlv.c







isis\_tlv.c

764 – 773 检查stream 中是否有足够的空间写入数据，如果发出警告并返回状态值 ISIS\_WARNING。

774 – 776 分别把便签名（tag），标签长度（length），以及标签值写入stream 结构体中。

777 – 779 定义debug 时的输出log

783 – 784 tlv\_area\_addrs 是各种tlv 写入函数的一种，它的功能是把一个域内地址的链表写 入stream中

790 – 798 循环把链表中的域内地址写入stream 中。

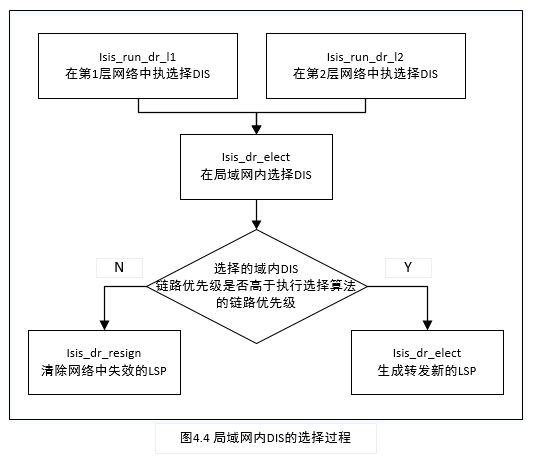
792 – 793 如果结构体中空间不足，跳转至错误处理相关代码。

794 – 797 把指针指向节点的数据复制进stream 中，并把指针指向下一结点。

## 选择局域网DIS

### 概述

局域网中指定中间节点（DIS）的选择的实现在文件isis\_dr.[ch]中。



当链路被创建或被设置成一个新的等级后，网络必须等待一段时间才能开始选择局域网中的DIS。这段时长介于0和ISIS hello 包往返时长之间。需要注意的是如果链路中没有邻域那么网络就不会选择DIS。DIS 的选择由结构体 isis\_circuit中的线程和标志run\_dr 控制。

这个选择过程由函数run\_dr\_elect() 实现。这个函数首先从邻域数据库中获取并构造一个节点邻域链表，之后根据这个链表获得在邻域中具有最高优先级的节点。如果节点之间的优先级相同，那么就比较它们的子网连接节点（SNPA）的地址值。找到具有最高优先级的节点邻域后，再比较这个邻域的和运行选择过程的链路的优先级。

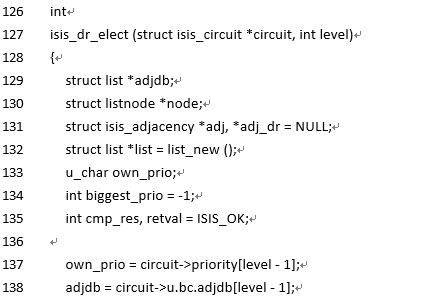
如果链路的优先级低并且这个链路之前是DIS 的话，系统就会调用函数isis\_resign 来清除网络中伪节点的链路状态协议包（LSP），然后重新分配链路中DIS的角色。如果链路的优先级大于等于这个邻域的优先级并且链路之前不是DIS 的话，函数isis\_dr\_commence 就会被调用来控制生成并装发链路的伪节点的链路状态包（LSP）。

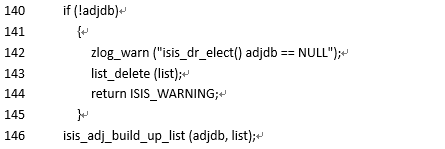
接下来详细讲解一下函数run\_dr\_elect(), run\_dr\_resign(), run\_dr\_commence()

### run\_dr\_elect 函数

函数run\_dr\_elect的功能就是在局域网中选择指定的DIS。这个函数首先从网络中选出一个具有最高优先级的候选节点，之后再把该节点和选择过程本身执行所在的节点比较。

isis\_dr.c





isis\_dr.c

129 – 135 函数局部变量：邻域中的候选节点链表；链表的节点结构体 ；结构体isis\_adjacency

保存邻域中的状态及组成信息； biggest\_prio 保存节点中最大的优先级的数值。

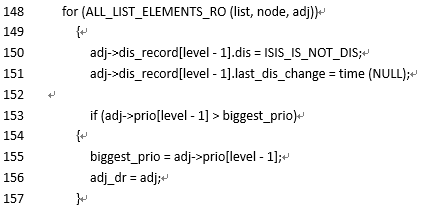
137 – 137 利用own\_prio 保存链路本身的优先级数值；

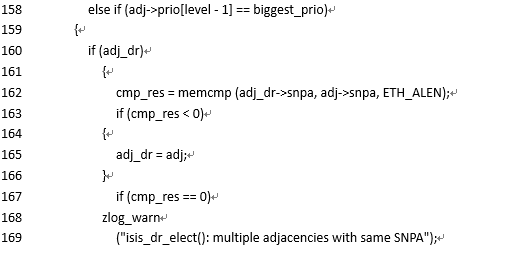
138 – 138 利用isis\_adjacency结构体类型的指针保存链路的邻域状态数据库；

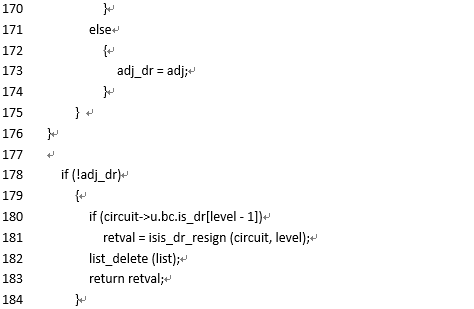
140 – 145 判断adjdb 指针是否是空值。如果是的话，在log 中打印警告信息， 删除创建的 list 链表，回收其空间； 如果不是，继续执行DIS 选举代码。

146 – 146 根据节点的邻域状态数据库构建邻域中的节点数据库。

isis\_dr.c







isis\_dr.c

本段代码的作用是循环遍历邻节点的节点，依次比较优先级，选出具有最大优先级的节点进行一下操作。

150 – 151 清除标记。把节点中标记是否是DIS的Flag设置为ISIS\_ISNOT\_DIS。并把节点中的 修改时间设置为当前时间。

153 – 157 判断该节点的优先级是否大于当前保存的最大优先级。如果是的话，把最大优先 级修改为当前节点的优先级，并用adj\_dr保存当前节点的指针。

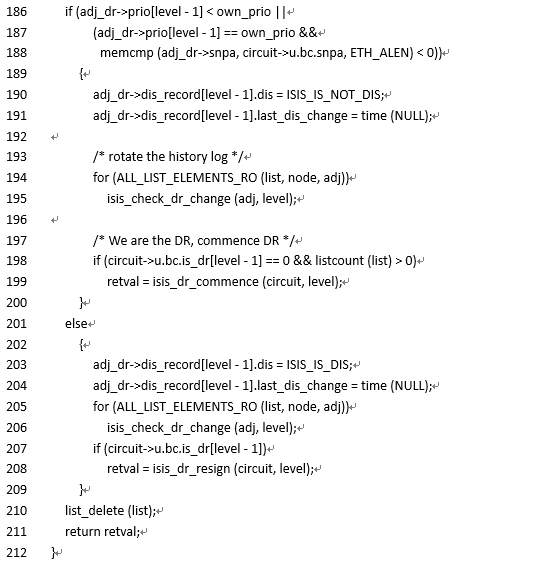
158 – 158 如果该节点优先级不大于最大优先级，判断当前节点优先级是否等于最大优先级。

160 – 169 如果当前节点优先级等于最大优先级，判断adj\_dr 是否为空。如果不为空，比较 两个节点子网地址的字母顺序。令adj\_dr指向子网地址字母顺序靠前的节点。

171 – 174 如果当前节点优先级等于最大优先级且adj\_dr指针为空，令adj\_dr指向当前节 点。

178 – 184 如果函数不能从邻节点中选出具有最大优先级的节点（即：指针adj\_dr 在遍历邻 节点链表空仍为空），判断函数调用者是否是中间系统。是的话把当前节点设置为 DIS，并调用函数isis\_dr\_resign()。

isis\_dr.c



isis\_dr.c

如果函数从邻域中找到了具有最大优先级的节点，执行上图代码。函数会首先比较自身优先级和选出的节点的优先级，之后根据比较结果调用不同的函数。

186 – 200 函数判断adj\_dr的优先级是否小于自身优先级或者两者优先级相等但是自身节点 的子网地址的字母顺序较大。如果是的话，执行函数的本身节点被设置为DIS。

190 – 191 函数把adj\_dr 的标志设置为ISIS\_IS\_NOT\_DIS，并记录当前时间。

194 – 195 回滚log，检查邻域节点的标志是否已经被设置为ISIS\_IS\_NOT\_DIS。

198 – 199 当前节点被设置为DIS，调用函数isis\_dr\_commence生成并转发新的LSP

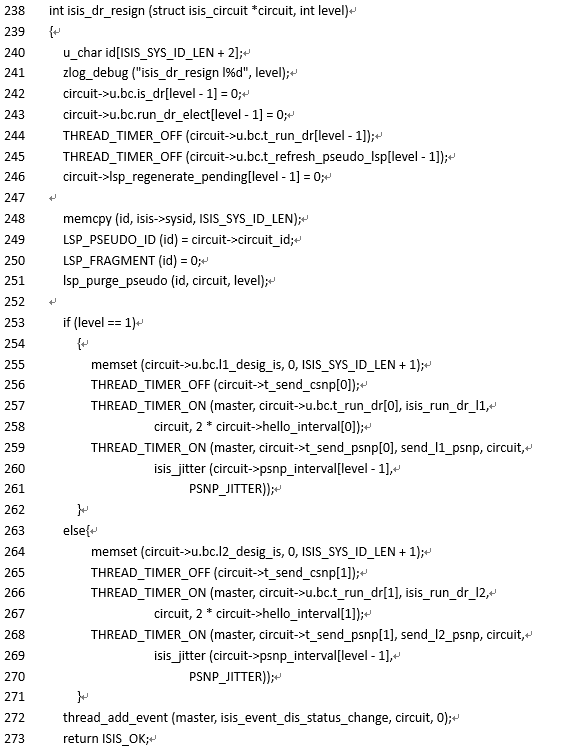
201 – 209 如果上一判断为假， 执行以下代码。首先把adj\_dr 指向的节点的dis 标志设置 为ISIS\_IS\_DIS，并记录下当前时间。然后回滚log，检查邻域节点的标志是否已经被 设置为ISIS\_IS\_NOT\_DIS。如果当前节点是DIS， 执行函数isis\_fr\_resign 使之前转 发的LSP包失效。

210 – 212 删除局部变量申请的空间，返回结果。

### run\_dr\_resign 函数

函数run\_dr\_resign的功能是清除网络中之前伪节点发送的链路状态包（LSP），之后再重新分配DIS 角色。

isis\_dr.c



isis\_dr.c

240 – 248 初始化函数：保存链路ID，设置链路相应网络层的DR值为0，设置链路才 run\_dr\_elect值为0。（说明该链路不是选择的DIS，也不是执行选择算法的IS）。

252 – 254 设置LSP的相应状态值：链路ID， 以及链路当前网络层次值。

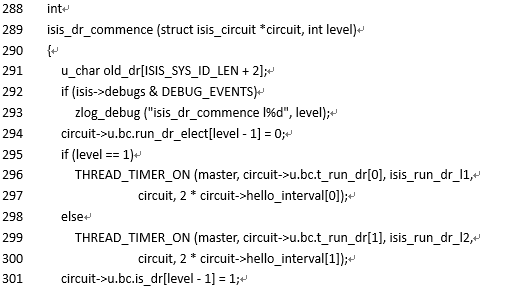
253 – 262 如果当前执行选举的网络层是level1， 在level1 层网络中转发LSP。

263 – 271如果当前执行选举的网络层是level2， 在level2 层网络中转发LSP。

### run\_dr\_commence 函数

函数run\_dr\_commence 的功能是当节点系统判定出DIS后控制生成新的伪节点的链路状态协议包并把这些包转发到与该节点相连接的节点。

isis\_dr.c



isis\_dr.c

291 – 291 声明变量old\_dr，用来保存过时的DIS的ID；

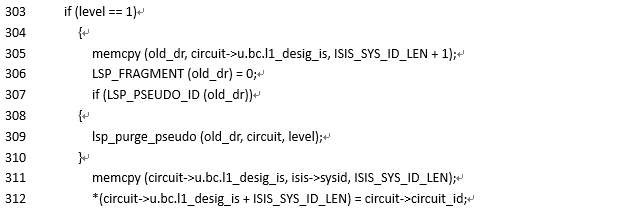
297 – 297 把链路的状态量run\_dr\_elect 设置为0，暂停执行函数run\_dr\_elect的进程。

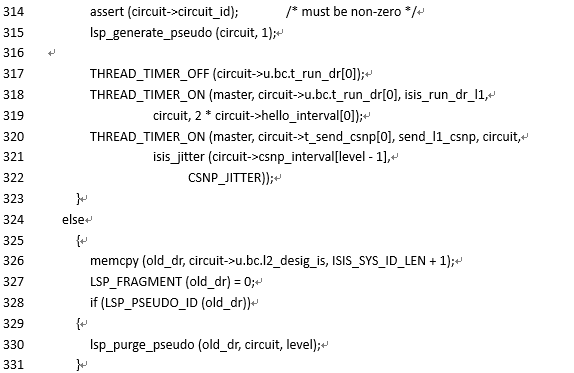
295 – 300 根据链路中设置的level 值，打开相应链路层的计时器。

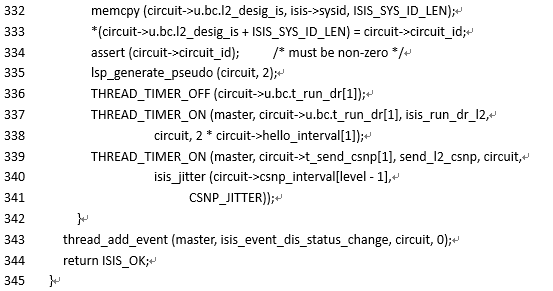
301 – 301 把当前链路设置为DIS。

以上代码的功能是为接下来LSP的转发做准备，相当于初始化链路状态。接下来的代码是在相应的网络层构造LSP并转发到链路邻域中。

isis\_dr.c







isis\_dr.c

303 – 323 如果节点是在level1 网络层中执行的DIS选举，则执行这段代码：函数会在LSP 报文中把选择的DIS相应的状态量设置为1，然后把old\_dr的状态量设置为0；接 着清除邻域状态数据库中old\_dr 的LSP； 最后生成最新DIS的LSP，并插入到邻域 状态数据库中，之后在level1网络中转发。

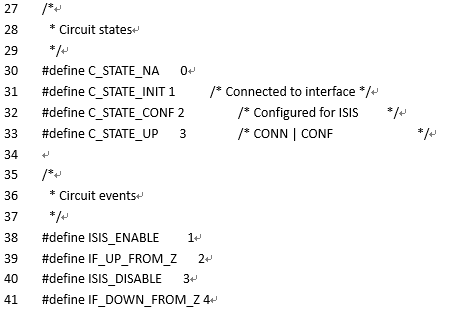
324 – 345如果节点是在level2 网络层中执行的DIS选举，则执行这段代码：函数会在LSP 报文中把选择的DIS相应的状态量设置为1，然后把old\_dr的状态量设置为0；接 着清除邻域状态数据库中old\_dr 的LSP； 最后生成最新DIS的LSP，并插入到邻域 状态数据库中，之后在level2网络中转发。

## 链路状态机

### 概述

ISO10589中给链路定义了两种状态：up 和down。ISIS路由协议中使用了Zebra，因此ISIS路由协议中链路状态拓展成了四种：N/A（链路处于未知的状态），init（链路具有Zebra规定的基本特征，但还没有配置），configured（链路为了ISIS路由进行了配置但是没有连接到网络接口），up（链路根据ISIS配置好了并且连接到了网络接口）。

isis\_csm.h



isis\_csm.h

图4.5 链路状态机状态和事件的常量定义

链路状态机对实现在文件isis\_csm.[ch] 中。链路状态机的运行仅由一个函数isis\_csm\_state\_change控制。这个函数的两个参数分别是图5.1中定义的链路和事件。

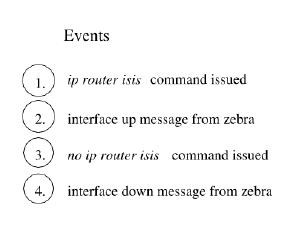
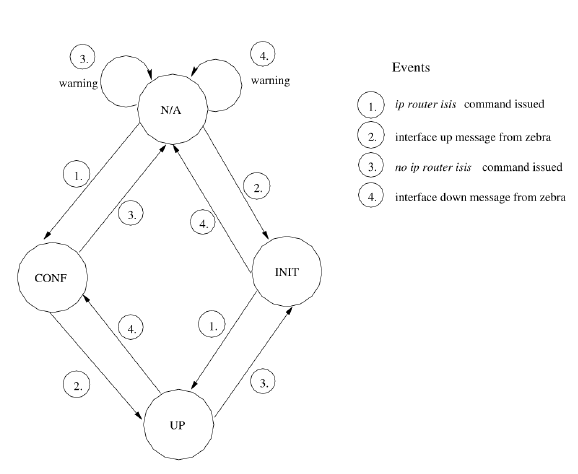


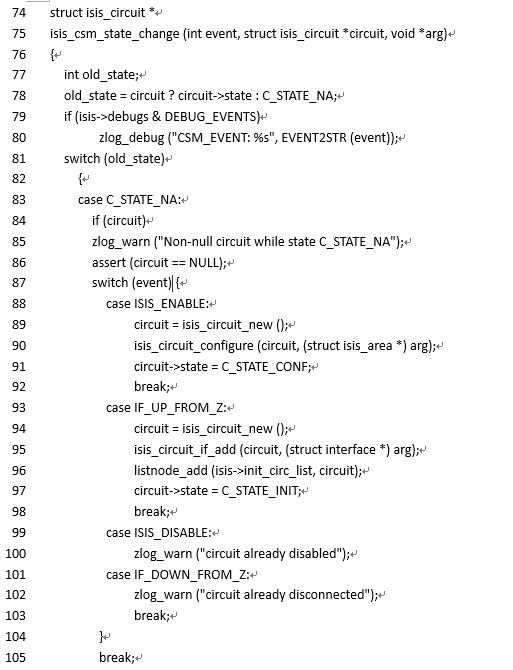
图4.6 链路状态机各个状态间的转换

图5.2表示了链路状态机的各个状态之间的转换。每个状态都可由两种其他的状态转换而来。

### isis\_csm\_stat\_change 函数

isis\_csm\_stat\_change函数的功能是控制链路状态机的运行，当不同的事件发生时，该函数会根据具体的事件类型进行处理。以下是在当链路状态机处于N/A状态时，各种事件触发的反应。

isis\_csm.c



isis\_csm.c

77 – 78 获取并保存状态机当前状态

79 – 80 当系统处于debug 状态时，打印log 日志。

83 – 86 若old\_state 的值为STATE\_NA，则加上断言circuit不为空。

88 – 92 若事件为ISIS\_ENABLE（即isis 路口器传递命令要求状态机准备使用），则创建个新 链路并把值赋给circuit，之后调用函数isis\_circuit\_figure配置链路，并把当前链路 状态改为C\_STATE\_CONF。

93 – 98 若事件为 IF\_UP\_FROM\_Z （即zebra 传递消息要求打开某端口），则创建个新链路 并把事件参数所指定的端口加入到链路中。最后把当前状态改为 C\_STATE\_INIT。

99 – 103 若事件值为ISIS\_DISABLE（即 路由器要求关闭链路状态机），则系统在log 中打印 出警告信息。

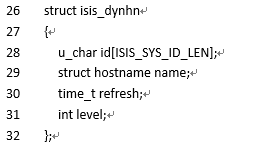
101 – 103 若事件为IF\_DOWN\_FROM\_Z（即 zebra 传递命令要求关闭某端口），则系统在log 中打印出警告信息。

## 动态主机名

### 概述

ISIS路由协议支持主机名动态更新机制。主机名缓存的实现在文件isis\_dynhn.[ch]中。节点从链路中获取的主机名保存在结构体isis\_dynhn中（如图5.3），这个结构体能够保存主机名，进程ID和节点的等级（level）。节点每遇到一个TLV，就会更新结构体中变量refresh的值，从而删除过时失效的主机名。需要注意的是，函数的print函数打印缓存中适配了进程ID的主机名而不是节点的进程ID。

isis\_dynhn.h

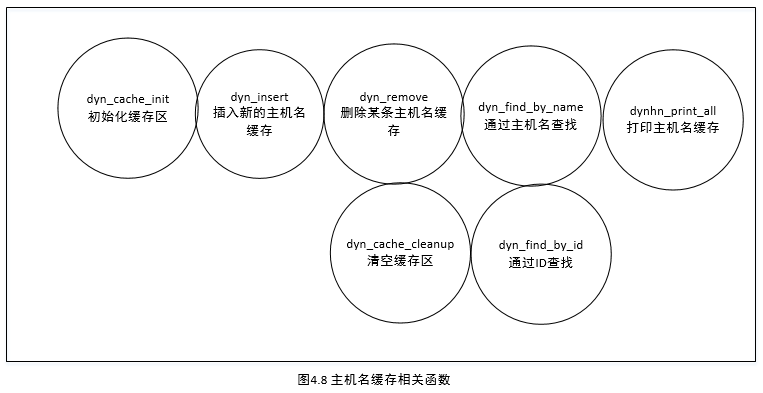


isis\_dynhn.h

图4.7 结构体isis\_dynhn的定义

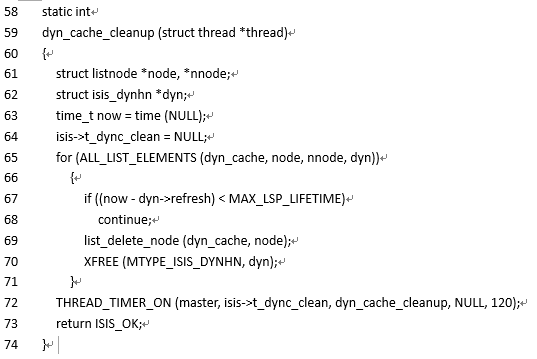
### cache相关函数

主机名动态更新机制的实现在文件isis\_dynhn.c 中。



Isis\_dynhn中一共有7个函数。其中，dyn\_cache\_cleanup, dyn\_cache\_init, dynh\_print\_all 函数是关于主机名缓存的处理函数； isis\_dynhn\_insert, isis\_dynhn\_remove 是处理结构体isis\_dynhn的函数，分别负责往缓存中插入和删除结构体；dynhn\_find\_by\_id, dynhn\_find\_by\_name 是在缓存中查找主机名缓存条的函数。

isis\_dynhn.c



isis\_dynhn.h

dyn\_cache\_cleanup 函数的作用是删除主机名缓存区内所有的缓存条目。它的参数是一个线程结构体。删除的操作将会在这个线程内异步进行。

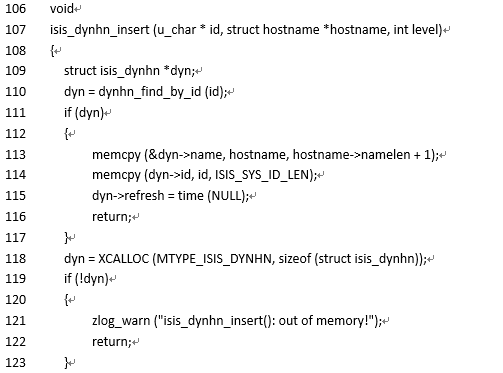
61 – 62 声明局部变量： 缓存条目结构体，和缓存区结构体

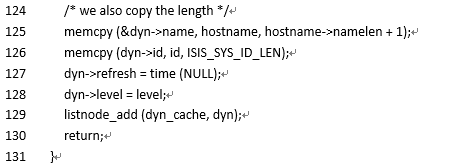
63 – 64 记录当前时间，并把isis 结构体的t\_dync\_clean值设置为空（意味着清理未完成）。

65 – 71 循环遍历缓存区链表上的缓存条目，依次删除后释放指针空间

函数dynhn\_insert 的作用是往主机名缓存区中插入的新主机名缓存条。它的三个参数分别是缓存条id，主机名名， 以及该主机名所在网络层级。函数会首先确认缓存区中是否已经存在该主机名条目，根据结果进行不同的插入处理。

isis\_dynhn.h





isis\_dynhn.h

109 – 110 声明局部结构体变量dyn存储dynhn\_find\_by\_id函数的返回结果。即函数在缓存 区中查找ID等于id的缓存条目，使用dyn保存查找结果。

111 – 117 如果缓存中已经包含ID 为id 的缓存条目，更新其保存的主机名，并记录当前时 间后函数返回。

118 – 123 如果缓存区中存在ID为id的缓存条目，则在缓存区中创建新的缓存条目，结果 保存在局部变量dyn 中。如果创建失败，则在log中打印出警告信息后返回。

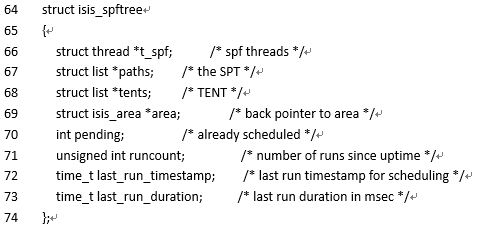
125 – 130 如果主机名缓存条目创建成功，则把参数传过来的信息拷贝到创建的缓存条目中， 并记录当前时间。把该缓存条目加入到缓存区中的条目链表后函数返回。

## SPT算法的实现

### 概述

中间节点必须定期执行最小生成树算法（SPF）来更新链路状态数据库或者邻域数据库。SPT算法的实现在文件isis\_spf.[ch] 中。每个域都要为所有的层（level）构建一个最小生成树（SPT）。SPT保存在结构体isis\_spftree中，如图4.9所示。这个结构体保存了定期运行SPF 算法的进程ID和上一次 SPF算法运行的时间戳。同时，也可以通过这个结构体连接到PATH和TENT数据库。注意，SPF 的运行频率应该小于系统配置变量minimumSPFInterval。参数pending和lastrun 用来规划SPF的持续运行，从而保证满足系统的需要。

isis\_spf.h

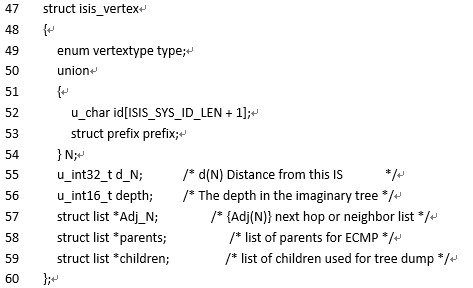


isis\_spf.h

图4.9 结构体isis\_spftree 的定义

PATH和TENT数据库中保存的内容是结构体isis\_vertex 中所描述的的网络节点。结构体中每个节点的包括一个指向节点本身来源（LSP）的指针和这个节点邻域（由一组与该节点具有同等花费连接的节点表示）。每个节点邻域的数量都被参数maximumPathSplits 限制。

isis\_spf.h



isis\_spf.h

图4.10 结构体isis\_vertex的定义

节点的类型由union体N 的值决定。对于中间节点和端节点来说，N保存的值是它们的系统ID。如果该节点是一个伪节点，N 还会追加一位非零值。如果节点描述的是一个网络可达地址，那么N 保存的就是该地址的网络地址和子网掩码。

在执行SPF算法时，节点到算法执行节点的距离(distance)保存在变量d\_N中，树的度（depth）保存在变量depth中。

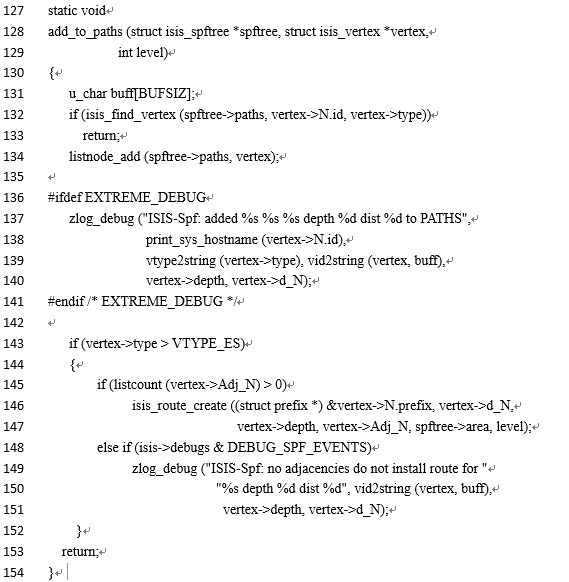
SPT算法的实现基于Annex C of ISO 10589[8]。在这个协议中，SPT 算法在基于Dijkstra 算法上增加了对多路同等花费路径的复用。这种对多路同等花费路径的复用的实现使用了负载均衡的技术，这意味着如果存在多条同等花费的路径到达目的地，系统会选择把数据流均衡的分布到这些路径中。

需要注意的是，如果节点同时支持IPV4和IPV6，那么该节点会分别运行SPT算法。

### add\_to\_path

函数add\_to\_path的功能是在节点运行SPF 算法时，把新的节点加入到最短路径树中（SPT）。函数首先检查SPT中是否已含有该节点，含有就直接返回；然后函数会检查新节点的类型是否是端节点，符合要求的话就会调用isis\_route\_create 函数为该节点创建路由并插入到SPT 中。

isis\_spf.c



isis\_spf.c

129 – 131 函数局部变量：isis\_spftree结构体spftree，它会定期运行SPF 算法的进程ID和上一次 SPF算法运行的时间戳 ；isis\_vertex 结构体vertex描述网络节点； level整形变量描述优先级。

132 – 134 isis\_find\_vertex函数如果找到了结构体spftree连接到的PATH数据库，vertex对应的网络节点编号及网络节点类型，则结束函数返回；若没有找到符合条件的，则利用listnode\_add将该节点添加进去。即找到SPT是否已经存在节点，存在则直接返回；不存在则添加该节点。

136 – 141 利用zlog\_debug进行终端调试，查看vertex描述的网络节点的id,类型，buff以及网络节点树的度（depth）的变量depth，节点的距离(distance)的变量d\_N是否正常；

143 – 147 条件如果vertex对应type> VTYPE\_ES,继续一个条件如果vertex->Adj\_N数量>0,则调用isis\_route\_create将结构体vertex的N.prefix, vertex的d\_N,vertex的depth, vertex的Adj\_N,及结构体spftree的area与 level添加进去。

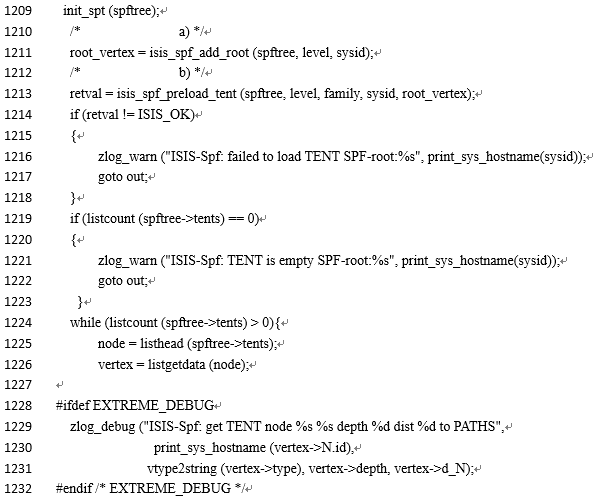
148 – 152 如果上述条件不符合，则又进行极限调试zlog\_debug。

153 – 153 执行完上述代码，函数返回。

### isis\_run\_spf函数

函数isis\_run\_spf 的功能就是在网络中运行Dijestra算法求得最短路径树。首先节系统会预加载节点数据库（TENT database），然后通过一个循环处理数据库中的节点并把产生的路径信息依次添加到路径数据库中（PATH database）。

isis\_spf.c





isis\_spf.c

1209 – 1213 init\_spt初始化构造spftree, root\_vertex为利用isis\_spf\_add\_root函数添加的节点，retval为利用 isis\_spf\_preload\_tent加载的节点数据库。

1214 – 1218 如果retval不等于ISIS\_OK，则利用zlog\_warn()函数输出异常信息，说明没有成功加载节点数据库。

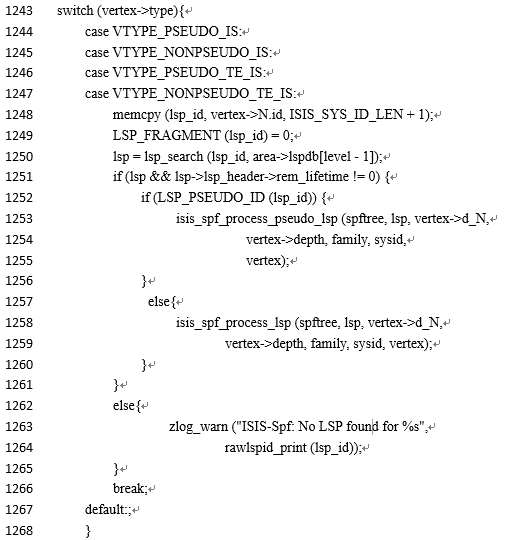
1219 – 1226 如果结构体spftree对应的tents数据库数量为0,利用zlog\_warn()函数输出异常信息，说明此时的tent数据库节点为空。

1228 – 1232 进行极限调试，输出各项相关有用信息。

1234 – 1236 执行完上述代码，删除节点，并把产生的路径信息依次添加到路径数据库

（PATH database）

isis\_spf.c



isis\_spf.c

1243 – 1247 switch结构，vertext对应的type有VTYPE\_PSEUDO\_IS, VTYPE\_NONPSEUDO\_IS, VTYPE\_PSEUDO\_TE\_IS, VTYPE\_NONPSEUDO\_TE\_IS四种类型

1248 – 1250 如果对应的是以上的四种类型之一，则继续执行，利用内存拷贝函数memcpy函数(的功能是从源src所指的内存地址的起始位置开始拷贝n个字节到目标dest所指的内存地址的起始位置中) vertex->N.id的起始位置开始拷贝 ISIS\_SYS\_ID\_LEN + 1个字节到lsp\_id所在的起始位置中。LSP\_FRAGMENT (lsp\_id) 赋值为0，lsp 定义为lsp\_search (lsp\_id, area->lspdb[level - 1])搜索到的lsp的编号以及area对应的数据库优先级。

1251 – 1261 条件判断，如果lsp不为空并且lsp对应的header对应的lefetime不为0，则继续执行。利用LSP\_PSEUDO\_ID函数分析lsp\_id，如果执行通过，将spftree, lsp, vertex->d\_N,vertex->depth, family, sysid,vertex信息添加到isis\_spf\_process\_pseudo\_lsp函数中；否则执行isis\_spf\_process\_lsp函数。

1260 – 1266 否则利用zlog\_warn输出异常信息

1267 – 1268 结束，如果vertex不是以上的四种类型，则走default结构,什么都不执行。

# 第五章 总结

ISIS作为一个分级的链路状态路由协议，基于DECnet PhaseV [路由算法](http://baike.baidu.com/view/2276401.htm)，它使用Hello协议寻找临近[节点](http://baike.baidu.com/view/47398.htm)，使用一个传播协议发送链接信息。ISIS可以在不同的[子网](http://baike.baidu.com/view/65511.htm)上操作，包括广播型的LAN、WAN和点到点链路。ISIS是一个开放式协议，适用于中型及特大型网络。在分析该ISIS协议的过程中，我们深刻体会到了这一点，同时也感受到ISIS在不断发展中体现出在中型乃至大型的重要地位，也深深佩服发明ISIS协议以及不断发展该协议的人的伟大之处。

ISIS是一个在不断发展的协议，虽然IS-IS协议最初是用于OSI路由器协议的,但是,ISO(国际标准组织)开发IS-IS是为了支持无连接网络服务/无连接网络协议(CLNS/CLNP)｡支持IP协议的集成的IS-IS是后来开发的｡其目的是提供一个能够为无连接网络服务(CLNS)提供路由的单一的路由器协议｡目前ISP(互联网服务提供商)正在使用IS-IS｡我们小组认为这一点非常好，事物发展才能彰显强大的生命力，发展才能使一个协议始终紧跟时代的发展，这也是ISIS的一个独特的优势。

在分析代码的过程中，我们可以清晰地看到，ISIS整体的代码架构十分清晰，风格极其严谨，当然作为一份给大家学习的代码，它的注释也非常棒。每每看到这些，不禁令人叹为观止，实在是经典之作，巅峰之作。同时，各个代码段间函数功能也特别清楚，命名及其规范，各个模块之间充分体现了“高内聚，低耦合”的程序设计原则。通过阅读以及分析ISIS的代码，我们真的学到了很多，为我们的以后的代码整体布局与编写都做到了一个极好的模范作用。

当然，任何事物都不是完美的。在我们分析的过程中，我们发现了ISIS存在的两个技术问题：第一是ISIS使用一个小的[度量值](http://baike.baidu.com/view/466367.htm)（6 比特），严重限制了能与它进行转换的信息；而且链接状态也只有8 比特长，[路由器](http://baike.baidu.com/view/1360.htm)能通告的记录只有256个。第二是ISIS受OSI约束，使得与OSPF相比它的发展比较缓慢。这个限制的原因是由于SPF的要求；但Wide-metric使这个范围变成24位的扩展解决了这个问题。

我们可以看到，身处当代，随着人类的不断发展，技术的不断创新，无线网络技术的运用越来越广泛。而我们作为网络专业的学生，这正是我们应该研究的问题，同时我们也希望ISIS能继续不断发展，在今后的网络中发挥越来越重要的作用。