**网络协议栈分析与设计课程大作业**

动态源路由协议协议代码分析

|  |  |  |  |
| --- | --- | --- | --- |
| 学号 | 姓名 | 班级 | 成绩 |
| 201792234 | 李明洋 | 软网1703 |  |

大连理工大学

Dalian University of Technology

目 录

[1 DSR协议概述 4](#_Toc28796689)

[2 DSR协议分组及路由过程 5](#_Toc28796690)

[2.1 分组格式 5](#_Toc28796691)

[2.2 路由发现 6](#_Toc28796692)

[2.3 路由维护 6](#_Toc28796693)

[3 DSR源代码分析 8](#_Toc28796694)

[3.1 文件介绍 8](#_Toc28796695)

[3.2 数据结构 9](#_Toc28796696)

[3.2.1 DSR数据分组 9](#_Toc28796697)

[3.2.2 DSR选项头 10](#_Toc28796698)

[3.2.3 DSR选项 11](#_Toc28796699)

[3.3 函数关系介绍 14](#_Toc28796700)

[3.3.1 接收DSR数据包 15](#_Toc28796701)

[3.3.2 发送DSR数据包 16](#_Toc28796702)

[3.4 函数分析——基本包处理 16](#_Toc28796703)

[3.4.1 初始化数据包 16](#_Toc28796704)

[3.4.2 将DSR选项头加入数据包 17](#_Toc28796705)

[3.4.3 向数据包中添加源路由选项 18](#_Toc28796706)

[3.4.4 处理接收到的数据包 19](#_Toc28796707)

[3.4.5 处理接收到的源路由请求 21](#_Toc28796708)

[3.5 函数分析——路由发现处理 23](#_Toc28796709)

[3.5.1 发起路由请求 23](#_Toc28796710)

[3.5.2 处理接收到的路由请求选项 26](#_Toc28796711)

[3.5.3 发起路由应答 29](#_Toc28796712)

[3.5.4 处理接收到的路由应答选项 30](#_Toc28796713)

[3.6 函数分析——路由维护处理 31](#_Toc28796714)

[3.6.1 确认请求选项 32](#_Toc28796715)

[3.6.2 确认选项 33](#_Toc28796716)

[3.6.3 发起路由错误 35](#_Toc28796717)

[3.6.4 处理接收到的路由错误选项 37](#_Toc28796718)

[3.7 路由缓存 38](#_Toc28796719)

[3.8 发送缓冲区 39](#_Toc28796720)

[4 总结 40](#_Toc28796721)

# 1 DSR协议概述

动态源路由协议(Dynamic Source Routing Protocol, DSR)是在移动[自组网](https://baike.baidu.com/item/%E8%87%AA%E7%BB%84%E7%BD%91)([MANET](https://baike.baidu.com/item/MANET/2063573))中使用的一种路由协议。它工作在TCP/IP协议族的网络层。

DSR是一种基于源路由方式的按需路由协议，它允许节点动态地发现到达目的节点的多跳路由。

源路由，是指在每个数据分组中包含了源节点到目的节点的完整路由信息，该路由信息由网络中的若干节点的地址组成，各个节点按照该路由信息来转发分组。

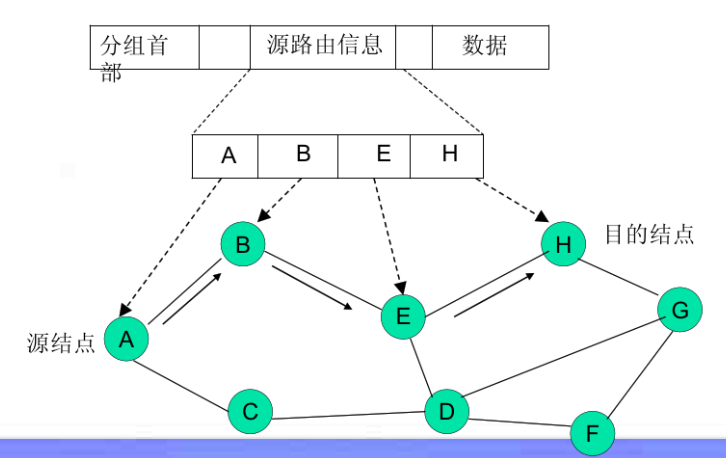


图1.1-1 DSR基于源路由的工作方式

按需路由，是指当需要时才进行路由发现。与主动式路由协议相比，按需路由协议中的节点并不实时地维护网络路由，只有在节点有数据要发送时才激活路由发现机制。由源节点在网络中发起路由查找过程，找到相应的路由后，才开始发送分组。

DSR协议主要由路由发现和路由维护组成。

# 2 DSR协议分组及路由过程

## 2.1 分组格式

DSR协议使用三种基本的控制分组，分别是路由请求（RREQ，Route Request）、路由应答（RREP, Route Reply）和路由错误（RERR, Route Error）分组。

1. 路由请求分组



图2.1-1 路由请求分组格式

Sid: 源节点ID号

Did: 目的节点ID号

Route record: 路由记录，按顺序记录从源节点到目的节点经过的路由（节点序列）

Request id: 路由请求ID，由源节点设置，同一个节点所发送的路由请求分组中的此ID是不同的

1. 路由应答分组



图2.1-2 路由应答分组格式

Did: 目的节点ID号（或者路由请求帧的源节点）

Date Route: 源节点到目的节点建立起来的路由（节点序列）

Reply Route: 路由应答帧所要经过的路由（节点序列）

1. 路由错误分组



图2.2-3 路由错误分组格式

Node1: 链路出错的一端节点ID号

Node2: 链路出错的另一端节点ID号

Did: 目的节点ID号或者遇到链路出错的数据分组的源节点ID号

Rerr Route: 路由出错分组所要经过的路由（节点序列）

## 2.2 路由发现

路由发现只有在源节点需要发送数据时才启动，帮助源节点获得到达目的节点的路由。

当源节点S需要传输新的分组到目的节点D时，S首先在自己的路由缓存中搜寻是否有到达D的现有路由信息。若存在到达D的有效路由，则直接使用此路由发送分组。否则，S就启动路由发现过程，动态寻找一条到达D的新路由。

以下为路由发现的过程：

1. 源节点向邻居节点广播路由请求（RREQ）。
2. 一个中间节点收到路由请求（RREQ）后，首先查遍自己的路由表，如果存在到达目的节点的路由，就直接发出一个路由应答（RREP）给源节点。如果中间节点没有在自己的路由表中找到到达目的节点的路由，就会把自己的地址加入路由记录，并将该路由请求（RREQ）作为广播分组发送给邻居节点。注意由本节点转发出去再次到达的请求会被丢弃。
3. 如果收到路由请求（RREQ）的节点是目的节点，这时路由记录中记录的节点地址序列就构成了从源节点到目的节点的路由信息，目的节点将此路由信息加入到路由应答（RREP）中返回给源节点。
4. 源节点收到路由应答（RREP）后，将路由信息存到路由缓存。

## 2.3 路由维护

路由维护可以在源节点给目的节点发送数据时检测当前路由的可用情况，当网络拓扑结构变化导致路由故障时，切换到另一条路由或者重新发起路由发现。路由维护主要通过路由错误分组（RERR）和确认分组来实现。

路由维护主要分为一下阶段：

1. 转发或发送数据分组的节点根据数据分组头中携带的源路由将其转发给下一跳节点，下一跳节点收到该数据分组后必须向上一跳节点返回一个确认分组，如果上一跳节点得到确认信息则说明现有路由正确，否则说明两个节点间的链路发生断裂。
2. 如果节点通过确认机制发现到下一跳节点的链路已经断裂，该节点就删除其路由缓存中包含该失效链路的所有路由，并创建一路由错误分组（RERR）发送给源节点。
3. 收到路由错误（RERR）的节点会检查自己的路由缓存，删除与该路由错误（RERR）中包含的断裂链路有关的所有路由表项，并转发该路由错误（RERR）。
4. 源节点接收到路由错误（RERR）后，将失效路由从缓存中删除，再尝试用缓存中的其他路由转发数据，如果不成功就重新启动路由发现过程。

# 3 DSR源代码分析

## 3.1 文件介绍

|  |  |
| --- | --- |
| 文件 | 说明 |
| dsr.h  list.h  tbl.h  timer.h | 整个代码中的宏定义  链表结构定义  路由表定义  计时器定义 |
| main-buf.c  neigh.c  debug.c  send-buf.c  dsr-srt.c  dsr-ack.c  dsr-rerr.c  dsr-rrep.c  dsr-rreq.c  dsr-opt.c  dsr-io.c  dsr-dev.c  dsr-pkt.c  dsr-module.c  dsr-rtc-simple.c  endian.c  link-cache.c | 协议的初始化和组织运行  邻居节点的相关操作  记录日志  发送缓存的定义及相关操作  源路由请求相关操作  Ack确认分组  RERR路由错误相关操作  RREP路由应答相关操作  RREQ路由请求相关操作  路由选项相关操作  用于发送与接收分组函数的实现  相关环境设置函数  DSR数据分组的初始化  IP包相关处理  源路由相关操作  主函数  链路缓存定义及相关操作 |
| Changelog  LICENCE  Makefile  README | 更改日志记录文件  授权文件  用于编译  简单介绍介绍 |

表3.1-1 主要文件内容

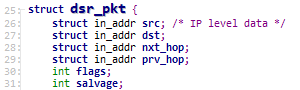
DSR的源代码根目录下主要包括17个.c文件，20个.h文件，1个记录更改日志的Changelog文件，1个LICENCE授权文件，1个用于编译的Makefile文件，1个简单介绍DSR的README文件，以及两个文件夹等共计47个文件。

## 3.2 数据结构

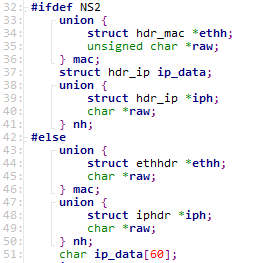
在进行详细的代码分析之前，首先对代码中出现的各种结构体、联合体进行简单的介绍。

### 3.2.1 DSR数据分组

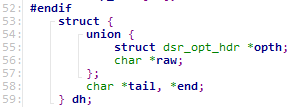
以下代码来自dsr-pkt.h和dsr-pkt.c两个文件。



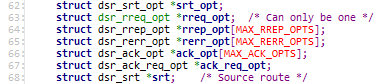
26—31 定义了DSR数据分组的基本结构，包括该数据包的源地址、目的地址、下一跳和上一跳等信息。



32—51 定义了nh联合体，指向网络层协议头（IP头），联合体使得我们能够使用结构体的方式访问IP头或者直接读取IP头的二进制数据。



52—59 定义了联合体dh指向DSR选项头。

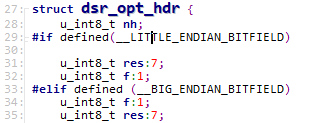


62—68 给出了DSR数据分组可以添加的选项，下面将对这些选项进行更为详细的介绍。DSR协议利用了一个特殊的头，它携带着可以包含在任何现有IP包中的控制信息，包中的这个DSR选项头包含一个固定大小的4-八位字节的小部分，然后是一个包含零个或多个DSR选项的序列，其中包含可选信息，即62—68行代码所展示的选项。

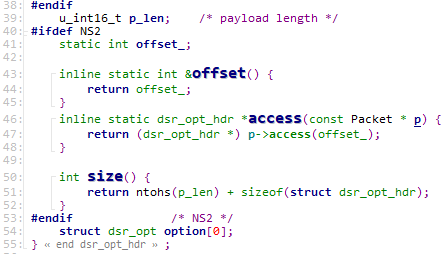
### 3.2.2 DSR选项头

以下代码来自dsr-opt.h文件。

DSR选项头包含一个固定大小的4字节的小部分，然后是一个包含零个或多个DSR选项的序列，其中包含可选信息。



27—35 nh是next header下一报文头，区分大小端分别定义了标志位和保留位及其所占位数。



39 DSR选项头的长度不包括4- 8字节固定部分，有效载荷长度字段的值定义了DSR选项头中包含的所有选项的总长度。

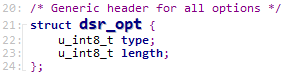
54 定义了option，包含一个或多个可选信息(DSR选项)，以类型长度值(TLV)格式编码。

### 3.2.3 DSR选项

1. 通用选项

以下代码来自dsr-opt.h文件。

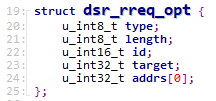
DSR选项头包含一个固定大小的4字节的小部分，然后是一个包含零个或多个DSR选项的序列，其中包含可选信息。



20—24 定义了DSR的通用选项的类型和长度。

1. RREQ选项

以下代码来自dsr-rreq.h文件。



在一个数据包中DSR选项报头必须立即跟随IP报头，因为在IP部分包含了源地址、目的地址以及跳数限制，所以在RREQ选项中不再重复包含。

20-21 成员type与length是通用选项头结构，分别定义了选项的类型和长度。

22 id是路由请求的发起者(原始发送者)生成的唯一值。发起路由请求的节点为每个路由请求生成一个新的标识值，例如基于节点发起的所有路由请求的序列号计数器。根据id值接收节点可以确定它最近是否看到过此路由请求的副本。如果接收节点在其路由请求表中找到对同一IP源地址和目标地址的相同id值，则此接收节点必须放弃此路由请求。当传播路由请求时，节点必须从正在传播的路由请求的接收副本中复制id值。

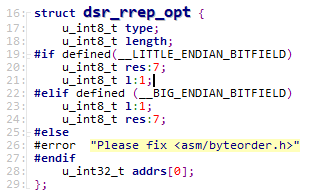
23 target是路由请求的目的节点的地址。

24 Address [i]是路由请求选项中记录的第i个节点的IPv4地址。IP头中源地址字段中给出的地址是路由发现的发起方的地址，不得在“ Address [i]”字段中列出该地址。因此，Address [1]中给出的地址就是路由请求发起方之后的路径上第一个节点的IPv4地址。在路由请求过程中，每个传播路由请求的节点都将自己的地址添加到此列表中。

注意，路由请求的源地址、目的地址以及TTL都包含在IP头中，而且此目的地址必须设置为IP受限广播地址（255.255.255.255）。

1. RREP选项

以下代码来自dsr-rrep.h文件。



17-18 定义了选项类型和选项长度。

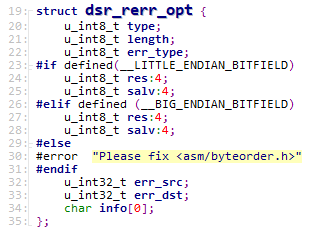
19-24 对于大小端的情况做了不同的处理，l是Last Hop External，置1表示路由应答给出的最后一跳实际上是DSR外部网络中的任意路径，置0表示最后一跳节点在DSR网络内部，节点在选择路由时，应该尽量选择这个标志位为0的路由。res是Reserved保留位，必须置为0。

28 路由应答选项必须返回源路由，addrs[0]就记录了源路由。

注意，与路由请求一样，路由应答的源地址、目的地址也存在于IP头中。

1. RERR选项

以下代码来自dsr-rerr.h文件。



20-22 定义了选项类型type、选项长度length以及错误类型err\_type。

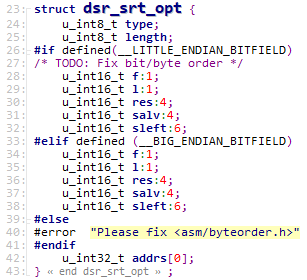
23-28 对于大小端的情况做了不同的处理，res是Reserved保留位，必须置为0。salv是Salvage，用来表示这个数据包被重发的次数，根据这个数值来决定该数据包是否应该被丢弃。

32-33 分别定义了路由错误选项的发送者和接收者的地址。

34 info[0]记录了错误类型的相关信息。

1. 源路由选项

以下代码来自dsr-srt.h文件。



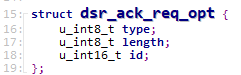
24-25 定义了选项类型type和选项长度length。

27-38 对于大小端的情况做了不同的处理，f和l分别表示First Hop External和Last Hop External，它们都是在置0时表示第一/最后一跳节点在DSR网络内部，在置1时表示第一/最后一跳节点在DSR网络外部节点，在选择路由时，应该尽量选择这两个标志位为0的路由。res是Reserved保留位。salv是Salvage，记录了这个数据包被拯救重发的次数。sleft是Segments Left，记录剩余的路由段数，即到达最终目的节点前仍要访问的中间节点数。

42 addr[0]记录了源路由的地址序列。

1. 确认请求选项

以下代码来自dsr-ack.h文件。



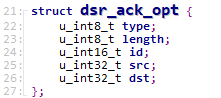
16-17 定义了选项类型type和选项长度length。

18 此id对应了路由请求选项中的id，表明是回复地哪个路由请求。

确认请求选项在DSR选项标头中不得出现多次。

1. 确认选项

以下代码来自dsr-ack.h文件。



22-23 定义了选项类型type和选项长度length。

24 此id从正在确认的数据包的确认请求选项的id字段中复制。

25-26 ACK Source Address源地址是发起确认的节点的地址，ACK Destination Address目的地址是该确认将被传递到的节点的地址。

注意，与确认请求选项不同，确认选项可以在DSR头中出现一次或多次。

## 3.3 函数关系介绍

下面从整个代码的输入输出函数入手，整体介绍以下DSR中的主要函数。

### 3.3.1 接收DSR数据包

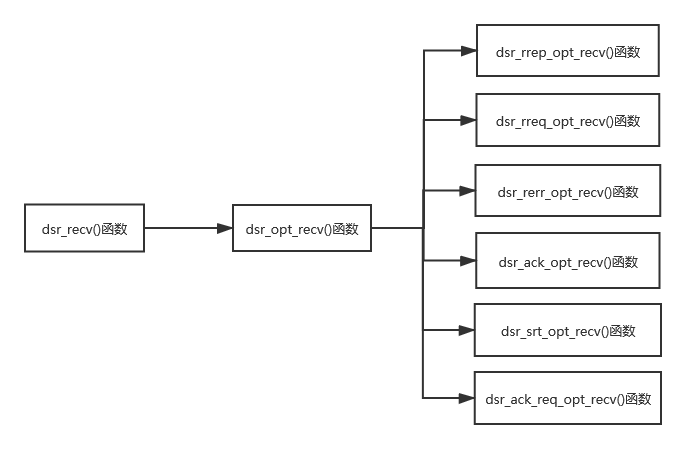


图3.3-1 dsr\_recv()函数调用关系介绍

### 3.3.2 发送DSR数据包

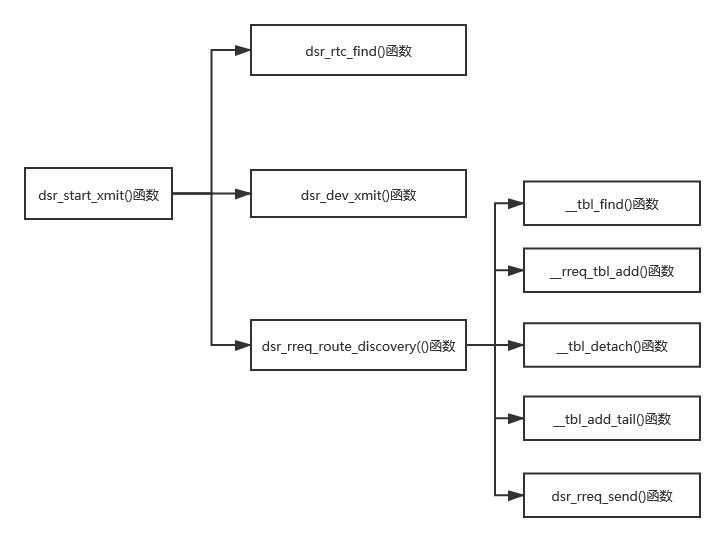


图3.3-2 dsr\_start\_xmit ()函数调用关系介绍

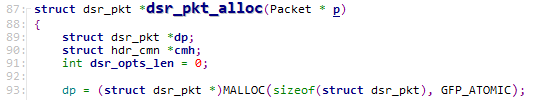
## 3.4 函数分析——基本包处理

### 3.4.1 初始化数据包

1. dsr\_pkt\_alloc()函数

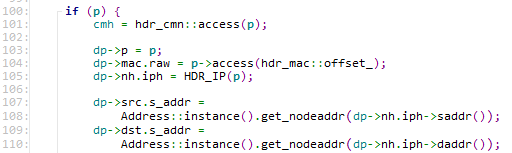
以下代码来自dsr-pkt.c文件。

该函数的作用是初始化一个数据包。



89-91 dp是一个数据包结构体，cmh是一个数据包的common头指针，dsr\_opts\_len是将要加入包中的DSR选项的长度。

93 为数据包申请一块内存空间。



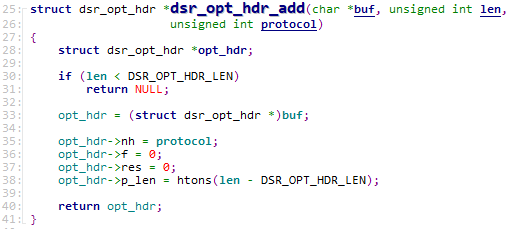
100—110 判断数据包非空后的一系列将p赋值给dp的初始化操作。

### 3.4.2 将DSR选项头加入数据包

1. dsr\_opt\_hdr\_add()函数

以下代码来自dsr-opt.c文件。

该函数的作用是将DSR选项头加入数据包中，注意，一个数据包中只能包含一个DSR选项头。



28 定义一个选项头。

30-31 先判断长度是否能放下该选项头。

33 如果长度足够，则将选项头指向缓冲区。

35-38 对选项头进行初始化操作。

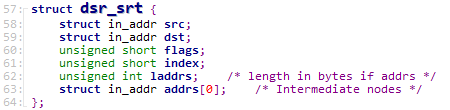
### 3.4.3 向数据包中添加源路由选项

发起数据包的节点在必要时向该数据包添加DSR源路由选项，以便将源路由从该始发节点传送到数据包的目的节点。

1. dsr\_srt结构体

以下代码来自dsr-srt.h文件。

DSR具体实现代码中利用如下结构体来记录一条源路由。



58-59 定义了源路径的源节点和目的节点。

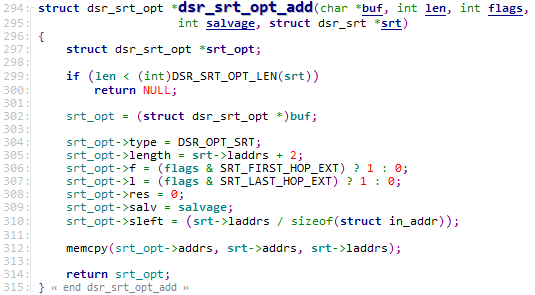
60-61 flags用来表示该条路径是否有效，index是一个标号用来唯一地确定一条源路由。

62-63 laddrs记录了一条源路由地址总长度，addrs[0]记录了一条完整的路径，由经过地节点来表示。

1. dsr\_srt\_opt\_add()函数

以下代码来自dsr-srt.c文件。

该函数的作用是将源路由选项加入到选项头中。



299-300 判断缓冲区的长度是否足够容纳的下该选项长度。

302-310 若缓冲区长度足够，则将选项的指针指向该缓冲区，然后进行各个值的初始化赋值操作。

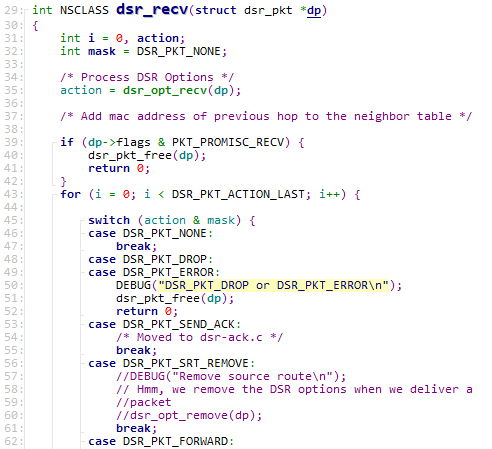
312-314 对列表地址进行复制并返回该指针。

### 3.4.4 处理接收到的数据包

以下代码来自dsr-io.c文件。

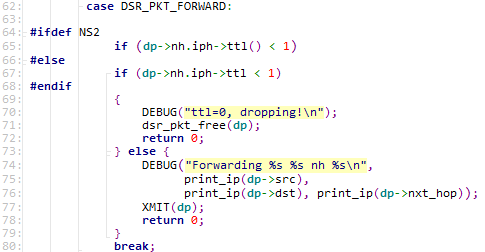
1. dsr\_recv()函数

该函数的作用是初步处理一个接收到的数据包，主要是对其中的选项进行类型判断。



35 调用的dsr\_opt\_recv()函数分析数据包中选项的类型，然后将相应的操作类型返回给action。

43-62 进入for循环，使用switch语句，根据数据包中选项的不同类型，进行相应处理。

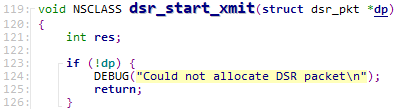


67-72 特别地，当接收到的数据包是需要被转发的数据包时，会检查其TTL，若TTL为0，则调用dsr\_pkt\_free()函数，释放数据包空间，丢弃该数据包。

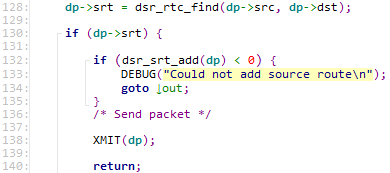
73-79 若TTL不为0，那么会打印源地址，目的地址和下一跳地址，然后执行XMIT()传输数据包，这里XMIT()是函数dsr\_dev\_xmit()的宏定义，关于此函数，将会在后面做详细介绍。

1. dsr\_start\_xmit()函数

该函数的作用主要是决定接收到的数据包的下一步操作，是直接转发出，还是暂存到缓存中等待路由发现。

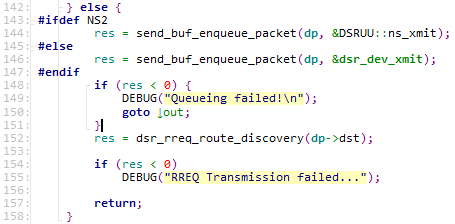


123-126 首先判断接收到的是否是一个数据包，若否则打印错误并退出。



128 调用dsr\_rtc\_find()查找是否存在从源节点到目的节点的已有路由。

130-140 若存在已有路由，则调用XMIT()直接发送数据包。



143-146 若不存在已有路由，则调用send\_buf\_enqueue\_packet()将数据包加入到发送缓存中。

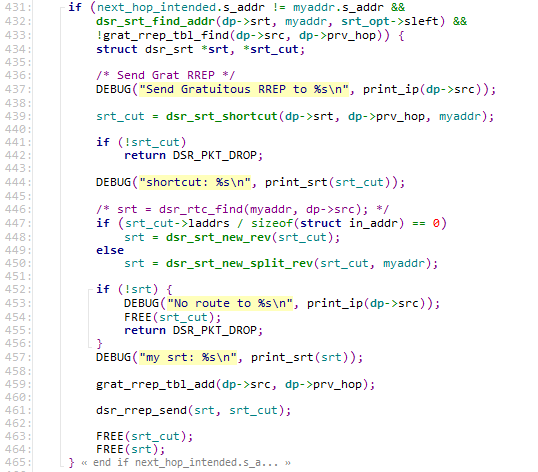
152 然后调用dsr\_rreq\_route\_discovery()函数，进行路由发现。

### 3.4.5 处理接收到的源路由请求

对接收到的源路由请求进行处理，主要是判断是否能够进行路由缩减。就是，当一个节点收到包含源路由选项的数据包，并且此节点不是该数据包的预期下一跳目标时，节点就会在这个数据包的剩余源路由中搜索是否有自身的地址，如果存在，意味着路由能够缩减。

以下代码来自dsr-srt.c文件。

1. dsr\_srt\_opt\_recv()函数



431-439 首先判断一个节点是否可以进行路由缩短：如果节点不是预期下一跳地址，并且在路由后续地址中查找到了自己的地址的同时自身应答列表中尚没有该路由回复，则可以调用dsr\_srt\_shortcut()函数进行路由缩减，返回一个缩减后的源路由结构src\_cut。

447-448 根据路由缩减后的不同情况再分别进行处理，如果缩减后剩余节点个数为0，则意味着该节点就为目的节点，那么直接调用dsr\_srt\_new\_rev()函数将srt\_cut中的路径进行反转，将反转后的路由记录在srt中。

449-450 如果节点本身不是目的节点，则调用dsr\_srt\_new\_split\_rev()函数，对路由路径进行先切割后反转。为达成这一目的，该函数会先调用dsr\_srt\_new\_split()函数切割掉多余的路径，然后再调用dsr\_src\_new\_rev()函数进行反转，也将反转后的路由记录在srt中。

459-461 向路由回复表中添加一条路由回复信息，并且发送一个路由回复用于通知新的路由变更。

如果不能进行路由缩减则直接转发数据包。

## 3.5 函数分析——路由发现处理

路由发现过程利用路由请求和路由应答，来主动搜索自组织网络以查找到所需目标目的地的路由。

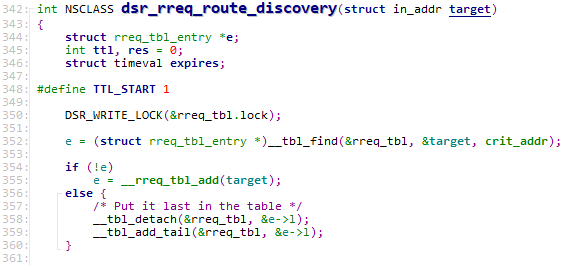
### 3.5.1 发起路由请求

启动路由发现的节点会在某些IP数据包的DSR选项头中创建并初始化路由请求选项。

1. dsr\_rreq\_route\_discovery()函数

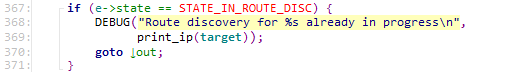
以下代码来自dsr-rreq.c文件。

该函数主要实现发起路由请求的功能。



352-355 调用\_tbl\_find()函数，在路由请求表中查找是否有相同目的地的路由请求，如果没有则调用\_\_rreq\_tbl\_add()函数直接添加进路由请求表。

356-360 如果在路由请求表中有相关记录，则先删除原有记录，再调用\_tbl\_add\_tail()函数将本次路由请求添加到路由请求表中的最后。



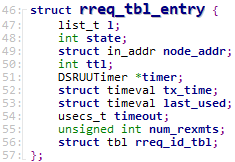
367-371 检查该路由请求表条目的state值，如果表示“当前状态已经在路由发现中”，那么打印信息，然后函数执行结束，否则将后面对条目中与时间相关的变量进行设置，最后返回res结束函数。

以下将对上述提到的路由请求表结构和其他函数做具体介绍。

1. rreq\_tbl\_entry路由请求表条目结构体

以下代码来自dsr-rreq.c文件。

节点必须在其路由请求表中维护有关其发起的路由请求的信息。当发起一个新的路由请求时，节点必须使用路由请求表中记录的信息来确定该路由请求的目的地，并且必须更新表中的信息以供下一个目的地相同的路由请求使用。

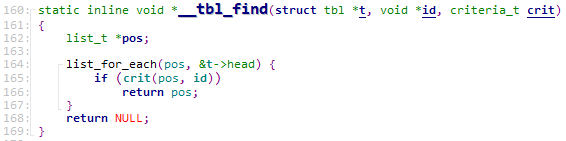


48-49 定义了路由请求的状态和目标地址。

53 last\_used表示上一次被使用到现在的时间间隔。

1. \_\_tbl\_find()函数

以下代码来自tbl.h文件。

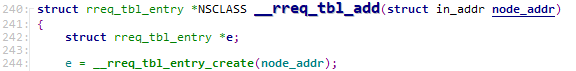


164-168 \_\_tbl\_find()函数遍历整个路由请求表，寻找是否有和参数id代表的目的地址相同的地址，若成功找到则返回指向匹配元素的指针，没有找到就返回NULL。

1. \_\_rreq\_tbl\_add()函数

以下代码来自dsr-rreq.c文件。

\_\_rreq\_tbl\_add()函数的功能是向路由请求表中添加表项。



244 调用\_rreq\_tbl\_entry\_create()函数，对表项进行内存分配，然后初始化项目值。

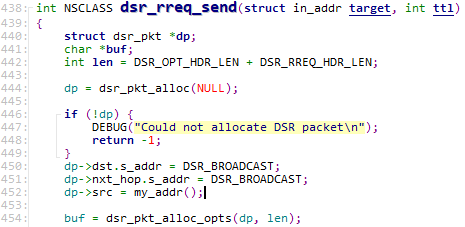


266 在此之前会先判断路由请求表是否已满，依据先入先出原则视不同情况，调用\_\_tbl\_add\_tail()函数将新表项插入到路由请求表的最后。

1. dsr\_rreq\_send()函数

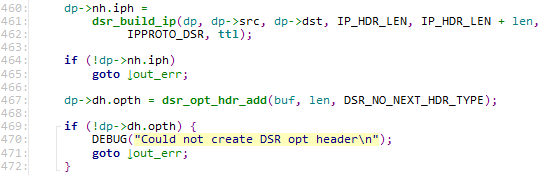
以下代码来自dsr-rreq.c文件。

该函数实现发送路由请求分组的功能。



440-444 构造一个DSR数据包，并申请为其分配内存。

450-454 对DSR数据包进行初始化赋值操作，将源地址设为本节点地址，将目的地址和下一跳地址都设为广播地址。

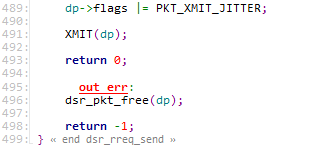


460-462 调用dsr\_build\_ip()函数构造一个IP头。

467 调用dsr\_opt\_hdr\_add()函数向IP头中添加DSR选项头。



477 调用dsr\_rreq\_opt\_add()函数向DSR选项头中添加路由请求选项。



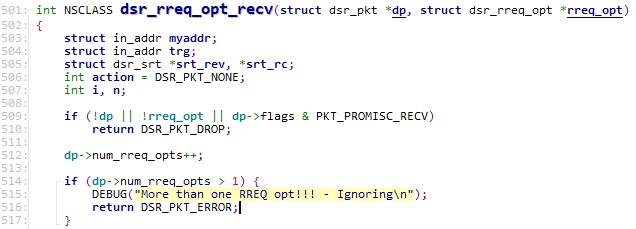
491-493 上述步骤都成功完成后，最后调用XMIT()函数发送数据分组，完成路由请求分组的发送。

### 3.5.2 处理接收到的路由请求选项

在2.2节中已经大体介绍了一个节点在接受到一个路由请求分组后所要做的主要工作，我们可以看出要根据接收节点是否为目的节点分情况处理，下面将对完成这一功能的函数做详细分析。

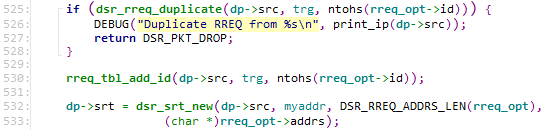
1. dsr\_rreq\_opt\_recv()函数

以下代码来自dsr-rreq.c文件。



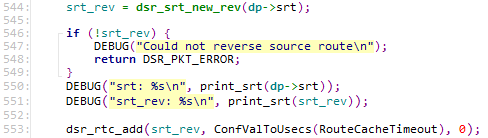
509-510对接收到的数据包和选项进行判断，若不满足条件，则丢弃该数据包。

514-517检查数据包中路由请求选项的数目，如果发现该路由请求选项超过了一个，则打印错误并返回“包错误”标记。



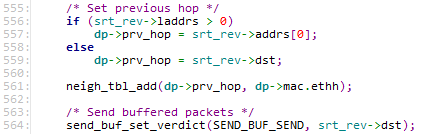
525-528 调用dsr\_rreq\_duplicate()函数判断接收到的路由请求是否已经在自己的路由请求表中，如果是则丢弃该路由请求包。

530-533 先调用rreq\_tbl\_add\_id()函数将该请求添加进路由请求表中，并关联该请求的标识id，再调用dsr\_srt\_new()函数，新建一个源路由选项，根据当前节点的路由，建立源路由。

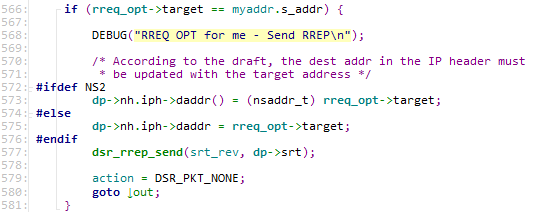


544 调用dsr\_srt\_new\_rev()函数将所收到的数据包中的路由信息反转，以便之后路由应答分组可能使用。

553 调用dsr\_rtc\_add()函数将该路由信息添加进节点自身的路由缓存中。



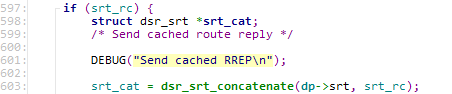
556-564 设置上一跳信息，然后将该数据分组复制到发送缓冲区中。



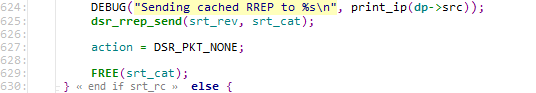
566-581 如果当前节点恰为路由请求的目标地址，则将IP Header字段设置为目标地址，然后调用dsr\_rrep\_send()函数发送路由应答，退出函数。



595 如果当前节点不是目标地址，则调用lc\_srt\_find()函数，查看自身节点中是否存在到目的节点的路由。



603 如果存在到目的节点的路由，就调用dsr\_srt\_concatenate()函数将这部分路由和路由请求中的源路由部分拼接起来。

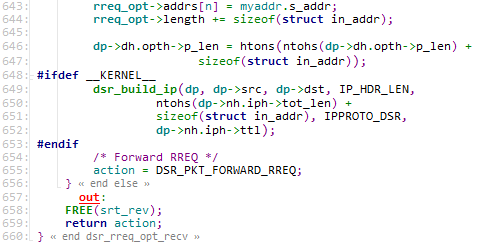


625 检查完拼接后的路由没有重复节点后，调用dsr\_rrep\_send()函数使用反转后拼接路由发送缓存路由应答。

在这里介绍以下缓存路由应答，如果一个节点接收到路由请求分组后，发现在其路由缓存中具有从其自身到路由请求目标节点的路由，则该节点将避免再次广播该路由请求，而是利用其自身的缓存路由生成到路由请求目标的路由应答，这称为“缓存路由应答”。该机制通过减少路由请求的泛滥，可以大大减少网络上路由发现的总体开销。



630-633 如果不存在到目的节点的路由，则调用dsr\_pkt\_alloc\_opts\_expand()函数，为DSR选项分配更多的内存空间。



643-647 将自身节点地址添加到路由请求选项的addrs数组中，更新路由请求选项长度和DSR数据包中的选项长度。

649-652 调用dsr\_build\_ip()函数，构造新的IP头，然后转发该路由请求分组。

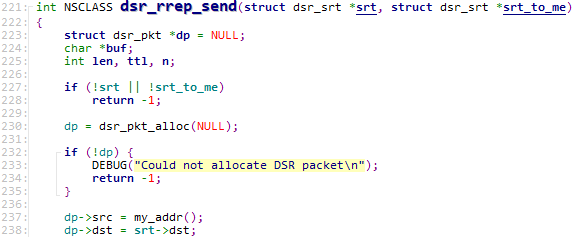
### 3.5.3 发起路由应答

DSR利用路由应答选项中返回路由应答，路由应答选项可以在单独的IP数据包（该数据包仅用于携带此路由应答选项）中返回给路由请求的发起者，或者也可以包含在发送到路由请求发起方的任何其他IP数据包中，路由应答选项包含在这些IP数据包的DSR选项头中。

以下代码来自dsr-rrep.c文件。

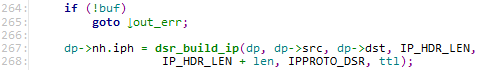
1. dsr\_rrep\_send()函数

该函数实现发送路由应答分组的功能。



223-230 构造一个DSR数据包，并申请为其分配内存。

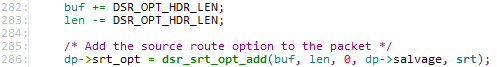
237-238 对DSR数据包进行初始化赋值操作，将源地址设为本节点地址，将目的地址设为源路由的目的地址。



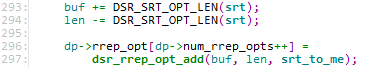
267-268 调用dsr\_build\_ip()函数构造一个IP头。



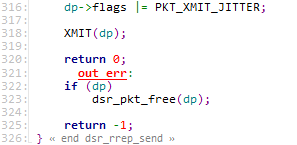
275 调用dsr\_opt\_hdr\_add()函数向IP头中添加DSR选项头。



286 调用dsr\_srt\_opt\_add()函数向DSR选项头中添加源路由选项。



296-297 调用dsr\_rrep\_opt\_add()函数向DSR选项头中添加路由应答选项。



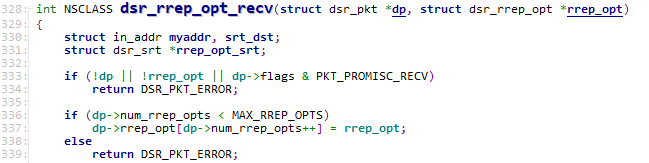
318 上述步骤都成功完成后，最后调用XMIT()函数发送数据分组，完成路由应答分组的发送。

### 3.5.4 处理接收到的路由应答选项

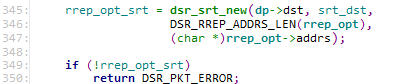
在3.4.4部分描述了接收到的DSR数据包后的一般处理，包括将路由信息从数据包DSR选项头中的选项部分，添加到接收节点的路由缓存中。包含路由应答选项的数据包与一般数据包的处理无很大不同，只要节点向其路由缓存添加新信息（包括从该路由应答选项添加的信息），那么该节点就会检查自己的发送缓冲区中的每个数据包，以确定其缓存中是否存在到达该路由应答目的地址的已有路由，如果存在，则应该使用该路由发送数据包，并将其从发送缓冲区中删除，这就是处理收到的路由应答选项所需的所有操作。

以下代码来自dsr-rrep.c文件。

1. dsr\_rrep\_opt\_recv()函数



333-339 首先对数据包、选项进行判断，若不满足条件，则丢弃该数据包。

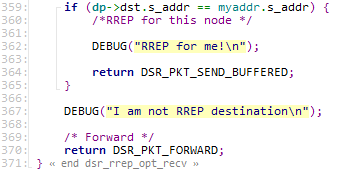


345-349 调用dsr\_srt\_new()函数，根据当前节点的路由，建立源路由，然后进行判断。



352 调用dsr\_rtc\_add()函数将路由应答中的路由信息添加进自身的路由缓存中。

355 调用rreq\_tbl\_route\_discovery\_cancel()函数删除路由请求表中对应此路由应答的路由请求条目。



359-365 如果接收节点恰为路由应答的目的节点，则输出相应信息，函数结束。

370 如果接收节点不是路由应答的目的节点，则继续转发该路由应答分组。

## 3.6 函数分析——路由维护处理

当网络的拓扑结构发生变化后，如果节点S通向节点D的路由记录不再有效时，DSR协议就会启动路由维护，尝试使用其他路由或者通过路由发现寻找一条新的路由。

路由维护功能的实现需要依靠DSR确认请求、确认以及路由错误选项的支持。

### 3.6.1 确认请求选项

在介绍DSR自身确认机制之前，其实DSR路由维护还可以使用其两种确认机制即链路层确认和被动确认。在这里简单介绍一下这两种ACK。

链路层ACK主要指MAC协议提供的成功交付数据包的反馈，如果链路层反馈可用，它将代替该路由维护中其他任何确认机制，节点不应将被动确认或网络层ACK用于路由维护。

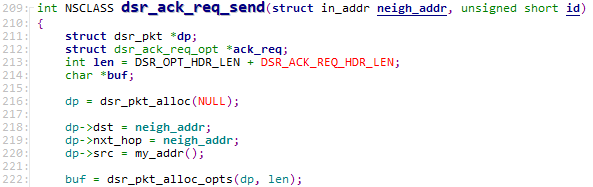
当链路层ACK不可用，而被动ACK可用时，在沿着最后一跳以外的任何一跳发起或转发包时，应使用被动ACK来进行路由维护。

当一个节点始发或转发一个数据包，并且没有其他确认机制（链路层ACK和被动ACK）可用于确定下一跳节点在源路由中的可达性以进行路由维护时，该节点应该向该下一跳节点请求网络层ACK，为此，节点在包的DSR选项头中插入确认请求选项。

以下代码来自dsr-ack.c文件。

1. dsr\_ack\_req\_send()函数

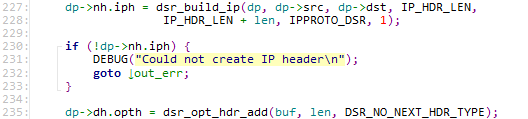
该函数用来发送确认请求选项，向目的节点请求ACK。



211-212 定义一个DSR数据包和ACK请求选项。

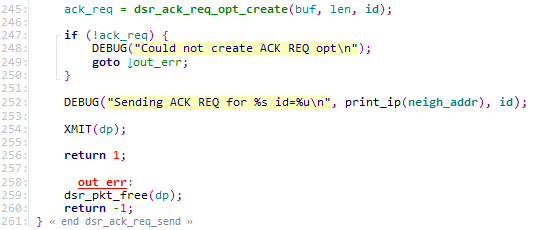
216-220 调用dsr\_pkt\_alloc()函数为数据包申请内存，然后进行初始化赋值操作，目的地址和下一条地址设为邻居节点地址，源地址调用my\_addr()函数设为自身的地址。

222 调用dsr\_pkt\_alloc\_opts()函数为DSR选项申请内存。



227-228 调用dsr\_build\_ip()函数，构造IP数据包的头部。

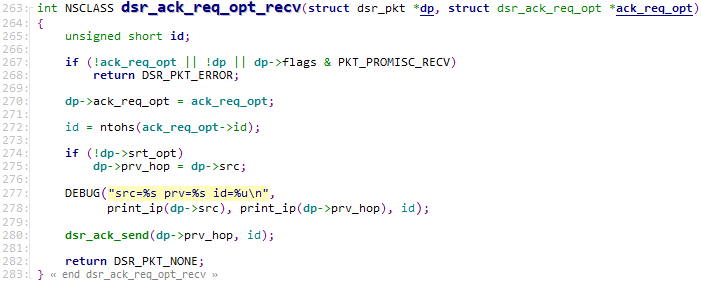
235 调用dsr\_opt\_hdr\_add()函数，将DSR选项头加入数据包。



245 调用dsr\_ack\_req\_opt\_create()函数，创建一个确认请求选项。

252-254 如果创建成功，调用XMIT()函数发送含有确认请求的数据包。

1. dsr\_ack\_req\_opt\_recv()函数



267-268 对收到的数据包行进行判断，如果不满足条件就返回包错误。

274-275 如果该包有DSR源路由选项，则将此包中上一跳节点的IP地址设置为此包中的IP源地址。

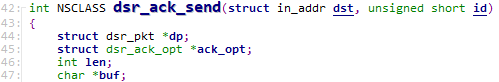
280 调用dsr\_ack\_send()函数，回复请求方一个ACK数据包。

### 3.6.2 确认选项

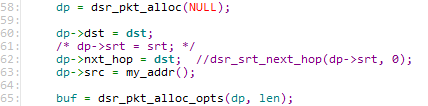
以下代码来自dsr-ack.c文件。

1. dsr\_ack\_send()函数

该函数用于向请求方返回ACK确认。



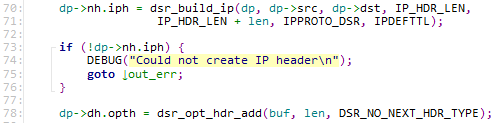
44-45 定义一个DSR数据包和ACK请求选项。



58 调用dsr\_pkt\_alloc()函数为数据包申请内存。

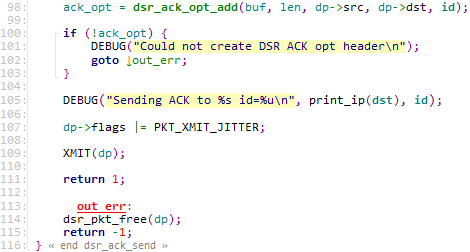
60-63 对数据包进行初始化赋值操作，将目的地址和下一条地址设为ACK回复地址，调用my\_addr()函数将源地址设为自身地址。

65 调用dsr\_pkt\_alloc\_opts()函数为DSR选项申请内存。



70-71 调用dsr\_build\_ip()函数，构造IP数据包的头部，包括源IP地址、目的IP地址、IP头长度、TTL等信息。

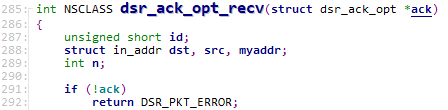
78 调用dsr\_opt\_hdr\_add()函数，将DSR选项头加入数据包。



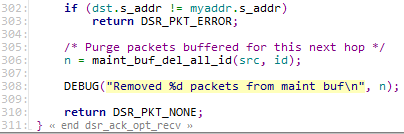
98 调用dsr\_ack\_opt\_add()函数，将ACK选项头添加至数据包。

109 上述各种创建操作都成功完成后，调用XMIT()函数，发送ACK确认分组。

1. dsr\_ack\_opt\_recv()函数



291-292 首先判断接收到的的是否为正确的ACK数据包，如果不满足条件，直接返回包错误。



302-303 判断当前节点是否为接收到的ACK消息的目的节点，如果不是则返回包错误。

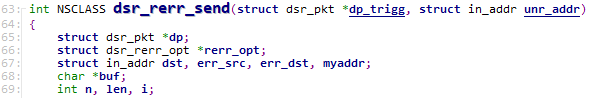
306 如果当前节点是目的节点，调用maint\_buf\_del\_all\_id()函数，删除维护缓存中发送给ACK源端的数据包，因为该数据包已经证明被成功接收。

### 3.6.3 发起路由错误

当节点在达到最大重传次数之后，无法验证下一跳节点的可达性时，应将路由错误发送到数据包的IP源地址。

以下代码来自dsr-rerr.c文件。

1. dsr\_rerr\_send()函数

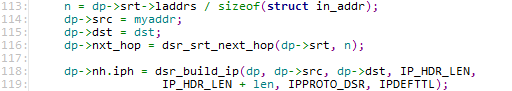


65-66 新建一个DSR数据包和路由错误选项。



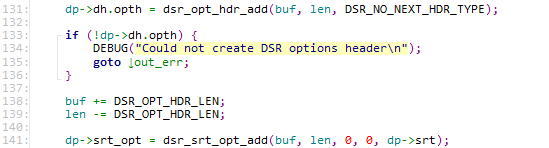
81-82 如果触发路由错误的数据包中，源路由选项中的Salvage字段为0，则将新数据包IP头的目的地址设为触发路由错误的数据包的源地址。

83-84 否则，将新数据包IP头的目的地址设为触发路由错误的数据包的源路由选项中的addrs[1]。



113-116 对DSR数据包进行初始化赋值操作。

118-119 调用dsr\_build\_ip()函数，创建一个IP数据包，并将IP协议字段设置为DSR的协议号，将源IP地址设置为该节点的地址。

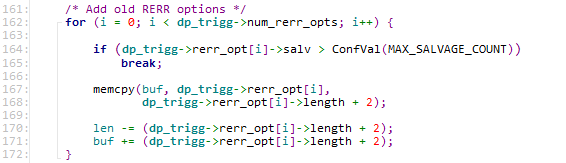


131 调用dsr\_opt\_hdr\_add()函数，在DSR数据包包中插入一个DSR选项头。

141 调用dsr\_srt\_opt\_add()函数，在DSR数据包包中插入一个源路由选项。



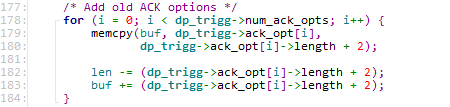
151-153 调用dsr\_rerr\_opt\_add()函数，将路由错误选项添加新数据包，并将错误类型设置为NODE\_UNREACHABLE，将源地址设置为本节点地址，将目的地址设置为新数据包的目的地址，将无法到达的节点地址设置为原始源路由中的下一跳节点的地址，将Salvage值设置为触发路由错误的数据包的DSR源路由选项中的Salvage值。



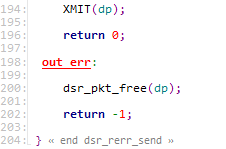
如果触发路由错误的数据包，包含任何路由错误或确认选项，那么节点可以将这些选项中的每一个都附加到其路由错误上，但有一定的约束条件。

此处为添加之前的路由错误选项。

164-165 节点不能包含触发新路由错误的数据包中的任何路由错误选项，因为其中包含的路由错误的总回收计数将大于新包中的MAX\_SALVAGE\_COUNT值。



此处为添加之前的确认选项。

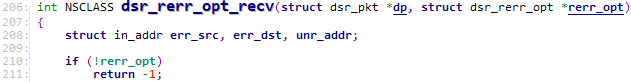


194-196 上述操作成功完成后，调用XMIT()函数，发送路由错误分组，函数结束。

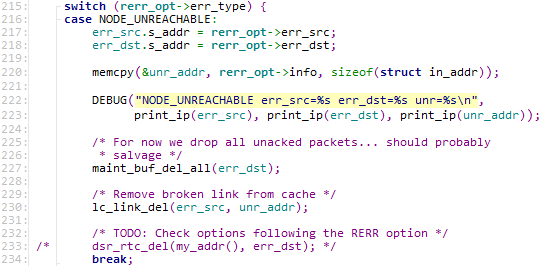
### 3.6.4 处理接收到的路由错误选项

以下代码来自dsr-rerr.c文件。

1. dsr\_rerr\_opt\_recv()函数

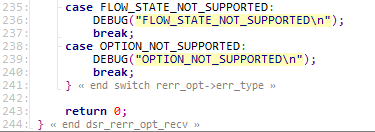


210-211 判断接收到的是否为路由错误选项，不是的话直接退出。



215 利用switch结构判断接收到的路由错误选项中的错误类型。

216-218 错误类型是不可达，则输出不可达地址信息，并调用maint\_buf\_del\_all()函数将缓冲区中所有等待该路由错误选项目的地址返回ACK确认的数据包全部删除，调用lc\_link\_del()函数删除这条不可达的路径，更新自己的路由缓存。



235-238 错误类型是流状态不支持或者是选项不支持，则打印错误信息。

## 3.7 路由缓存

DSR中的每个节点都要维护一个路由缓存，其中包含该节点所有的路由信息。存储路由信息的数据结构有多种，如路径缓存（path cache）和链路缓存（link cache）。

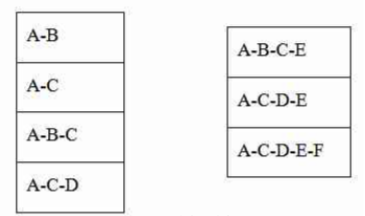
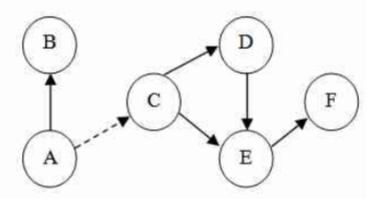
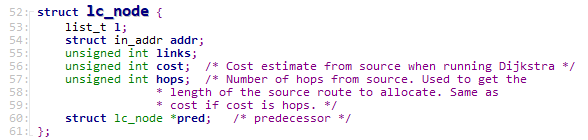
 

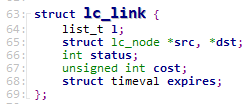
图3.7-1 路径缓存 图3.7-2 链路缓存

本代码中使用的是链路缓存，在链路缓存中，节点把获得的路由信息按照每条链路的方式存储到节点关于网络的拓扑图中，当有数据要发送时，节点执行图的搜索算法，比如Dijikstra算法，查找到目的节点的路由。

以下代码来自link-cache.c文件。



52—61 定义了链路中的一个节点，包含该节点地址，前一节点，花费的跳数等信息。



63—69 定义了一整条链路，timeval结构体类型的expires记录了该条路径的生存期。

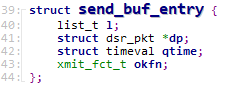
此外该文件中还有维护链路缓存的一系列相关函数，如在链路中添加、删除、查找节点等，但与DSR协议主要内容关系不大这里不再赘述。

## 3.8 发送缓冲区

每个DSR节点都有一个发送缓冲区，发送缓冲区是一个无法由该节点发送的数据包组成的队列，这些数据包不能发送是因为该节点还没有到他们目的地的源路由。

发送缓冲区中的数据包与它们被放置在缓冲区中的时间相关联，超时后会被从缓冲区中删除。

以下代码来自send-buf.c文件。



39-44 定义了缓冲区中的一个个体，l是其在队列中的标识，dp记录了该块缓冲区内的数据包，qtime是排队时间。

该文件中也同样包有维护缓冲区的相关函数，如给到达的数据包排队、超时处理、冲刷缓冲区等，这里不再一一介绍。

# 4 总结

通过阅读DSR协议的RFC文件以及分析该DSR协议代码，我对DSR协议的基本工作原理和具体实现已经基本掌握。

DSR协议作为一种适用于动态自组织网络的按需动态源路由协议，拥有很多的有点，如：

1. 没有节点要发送数据时，网络中没有通信开销；
2. 转发数据分组的中间节点不需要存储路由信息，节省了存储空间；
3. 支持到目的节点的多条路径；
4. 用路由缓存技术减少了路由建立的开销，同时使用路由缓存技术还可以减少路由建立的开销。

但DSR协议也仍存在一些需要改进之处，如：

1. 每个数据分组头部都需要携带路由信息，数据分组的额外开销较大，不适合网络直径大的自组网；
2. 路由请求消息采用洪泛方式，相邻节点路由请求消息可能发生传播冲突并可能会产生重复广播。