**动态源路由协议协议代码分析**

**（29组）**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 学号 | 姓名 | 班级 | 负责模块 | 成绩 |
| 201792249 | 杨璐 | 软网1702班 | 原理分析  代码分析  文档撰写 |  |

大连理工大学

Dalian University of Technology

目录

[第一章 引言 3](#_Toc28956984)

[第二章 DSR基本原理 3](#_Toc28956985)

[2.1 路由发现 3](#_Toc28956986)

[2.2 路由维护 4](#_Toc28956987)

[第三章 DSR代码介绍 5](#_Toc28956988)

[3.1 文件和函数关系 5](#_Toc28956989)

[3.1.1 文件 5](#_Toc28956990)

[3.1.2 函数调用 6](#_Toc28956991)

[3.2 数据结构 8](#_Toc28956992)

[3.2.1 路由缓存Route Cache 8](#_Toc28956993)

[3.2.2 发送缓冲区Send Buffer 9](#_Toc28956994)

[3.2.3 路由请求表Route Request Table 10](#_Toc28956995)

[3.2.4 DSR选项头 12](#_Toc28956996)

[3.2.5 数据包 21](#_Toc28956997)

[第四章 DSR具体实现 23](#_Toc28956998)

[4.1 一般数据包处理 23](#_Toc28956999)

[4.1.1 生成一个数据包 23](#_Toc28957000)

[4.1.2 给数据包添加一个DSR选项头 25](#_Toc28957001)

[4.1.3 给数据包添加一个DSR源路由选项 26](#_Toc28957002)

[4.1.4 处理接收到的数据包 26](#_Toc28957003)

[4.1.5 处理接收到的DSR源路由选项 29](#_Toc28957004)

[4.2 路由发现 30](#_Toc28957005)

[3.2.1 生成一个路由请求 31](#_Toc28957006)

[4.2.2 处理一个接收到的路由请求选项 32](#_Toc28957007)

[4.2.3 发送一个路由回复 35](#_Toc28957008)

[4.2.4 防止路由回复风暴 36](#_Toc28957009)

[4.2.5 处理一个接收到的路由回复选项 37](#_Toc28957010)

[4.3 路由维护 37](#_Toc28957011)

[4.3.1 确认 37](#_Toc28957012)

[4.3.2 生成一个路由错误 38](#_Toc28957013)

[4.3.3 处理一个接收到的路由错误选项 39](#_Toc28957014)

[4.3.4 重发一个数据包 40](#_Toc28957015)

[第五章 小结 41](#_Toc28957016)

# 第一章 引言

动态源路由协议（DSR）是一个专门为多跳无线Ad Hoc网络设计的简单高效的路由协议。所有的路由都是由DSR路由协议动态地、自动地确定和维护，它提供快速反应式服务，以便帮助确保数据分组的成功交付。

DSR的基础版本使用了“显式的”、“源路由”的方法，也就是说每一个数据包在它的首部都包含了它经过的完整的、有序的节点列表。

这种方法的作用如下：

1.发送者可以选择并控制发送数据包的路由；

2.支持使用多个路由到达任意目的地（这种方法可以应用于达到负载平衡的目的）；

3.因为每个数据包的头部都包含了它的源路由，因此这种简单有效的机制避免了产生回路，此外，其他转发或者监听该数据包的节点也可以储存路由信息以供将来使用。

# 第二章 DSR基本原理

DSR主要包括路由发现和路由维护两个部分。

## 2.1 路由发现

只有在源节点需要发送数据才启用，帮助源节点获得到达目的节点的路由。

以一个例子说明。

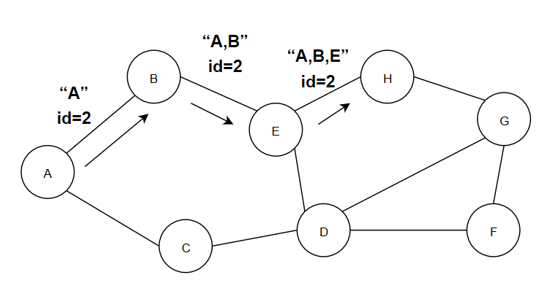


图2.1 路由发现

假设节点A想要发现一条到达节点E的路由。初始化路由发现，节点A发送一个广播数据包作为“路由请求” (RREQ)，A节点的无线传输范围内的每一个节点都接收到了这个请求数据包。路由请求中包含：

1. 每一个路由请求标识了发送节点和目的节点，还有一个唯一的请求标识（图中的例子：id=2）。

2. 每一个路由请求还包含了一个记录，记录中列出了每个中间节点的地址，这个记录被初始化为一个空的列表。（图中例子：记录 “A”。如果它是一个中间节点，接收到路由请求之后，它会将自己的地址附在路由请求的记录后面。（图中的例子：记录“A，B”）

接收到RREQ的节点有两类，下面分别讨论：

1. 如果这个节点是中间节点：

如果这个节点已经收到过同样的路由请求，或者这个节点的IP已经出现在路由请求的路由记录中，那么这个节点会丢弃这个数据包。(重复检测)

否则，这个节点会把自己的地址追加到路由请求的路由记录中，然后把这个路由请求广播出去。（这时候标识符id不做改变，仍然是原来的）。如图中例子所示，B是中间节点，那么就会广播路由请求出去，C就会收到这个路由请求。

1. 如果这个节点是目的节点：

向源节点回复一个路由回复，包含了路由请求消息中路由记录的拷贝。当源节点收到这个路由回复之后，它把这条路由记录放到它的路由缓存中。

## 2.2 路由维护

路由维护主要完成两个任务:

1. 当源节点给目的节点发送数据时监测当前路由的可用情况。
2. 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的过程。

它的基本机制是逐跳证实机制: 当使用源路由发起或转发数据包的时候，每个发送数据包的节点负责确认该数据包可以从该节点通过链路传递到下一跳。比如，在图中的例子中，数据包的源节点是A，目的节点是E，中间节点为B、C和D。

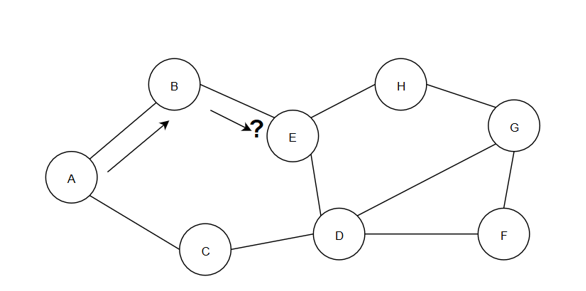


图2.2 路由维护

在图中的例子中,节点A负责从A到B的链接,节点B负责从B到C的链接,节点C负责从C到D的链接,节点D负责从D到E的链接。

# 第三章 DSR代码介绍

## 3.1 文件和函数关系

### 3.1.1 文件

|  |  |
| --- | --- |
| 文件名 | 文件内容 |
| dsr-ack.h, dsr-ack.c | 确认 |
| dsr-io.h, dsr-io.c | 输入输出操作 |
| dsr-module.c | dsr中的模块 |
| dsr-opt.h, dsr-opt-c | dsr选项 |
| dsr-pkt.h, dsr-pkt.c | 数据包 |
| dsr-rerr.h, dsr-rerr.c | 路由错误 |
| dsr-rrep.h, dsr-rrep.c | 路由回复 |
| dsr-rreq.h, dsr-rreq.c | 路由请求 |
| dsr-srt.h, dsr-srt.c | 源路由选项 |
| dsr.h | 动态源路由协议 |
| endian.c | 大小端 |
| link-cache.h, link-cache.c | 路由缓存 |
| link.h | 链表 |
| maint-buf.h, maint-buf.c | 路由维护 |
| neigh.h, neigh.c | 邻居结点 |
| send-buf.h, send-buf.c | 发送缓存 |

表3.1 主要文件

### 3.1.2 函数调用

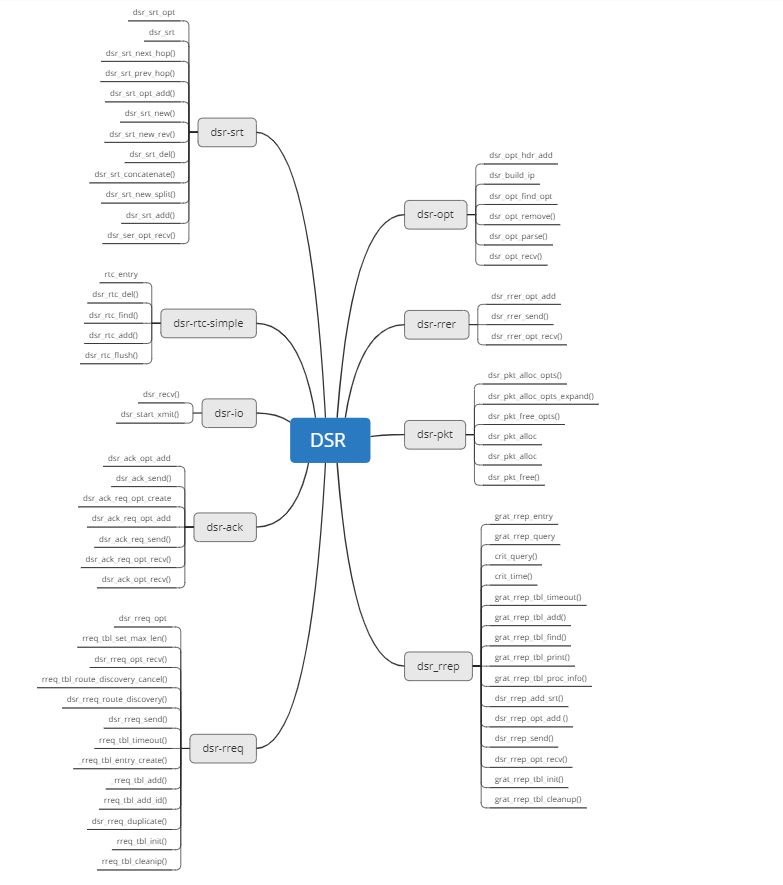


图3.1 主要函数

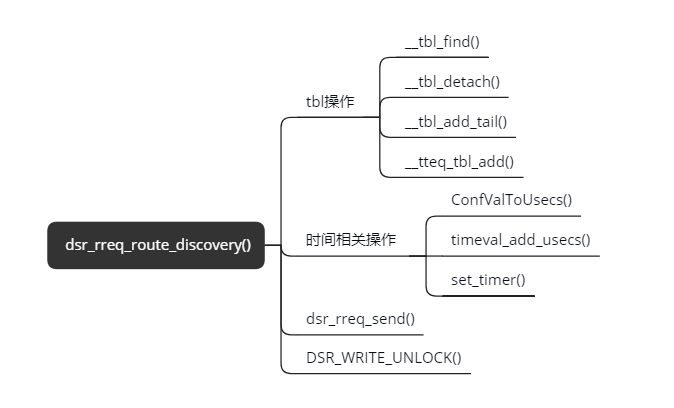


图3.2 路由发现相关函数调用（一）

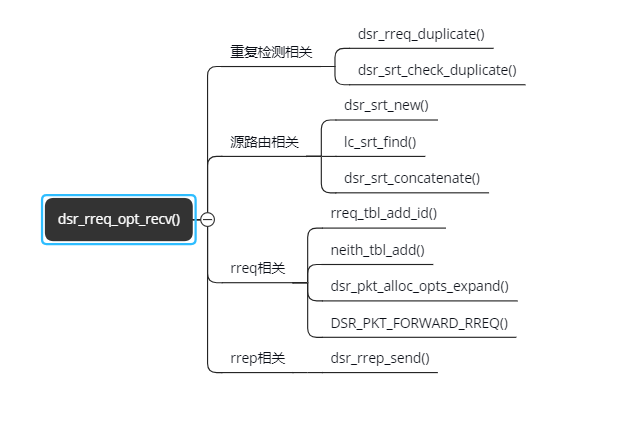


图3.3 路由发现相关函数调用（二）

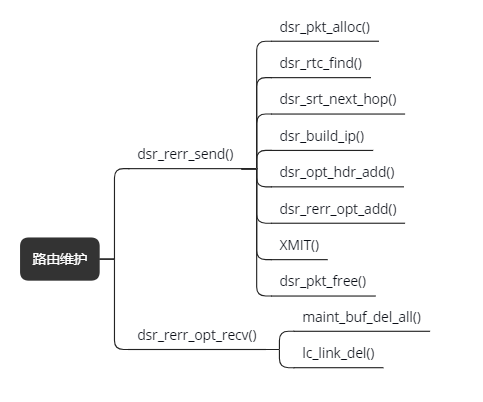


图3.4 路由维护相关函数调用

## 3.2 数据结构

在大作业的这个部分，将介绍DSR协议的数据结构，并简单介绍它们的使用。

### 3.2.1 路由缓存Route Cache

每一个节点包含一个路由缓存，它包含了节点需要的路径信息。

当一个节点知道了网络中节点之间的连接关系时，就会把相应的路由信息添加到路由缓存中来。（比如说，一个节点收到了路由请求或者路由回复，就会向路由缓存中添加相关信息）。

当一个节点知道了网络中节点的一些连接被破坏时，就会在路由缓存中删除相应的路由信息。（比如节点收到了路由错误)

路由缓存的结构如下代码所示：

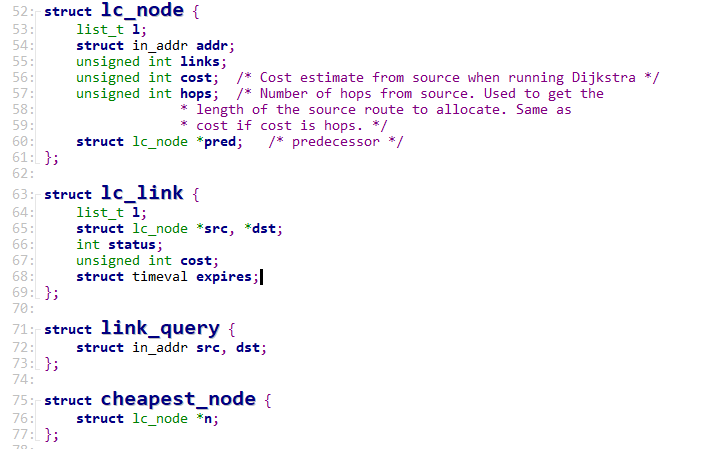


图3.5 路由缓存

这是链式的缓存结构。

52-6 一个节点的路由缓存信息。addr是节点的地址。cost指的是从源节点到当前节点运行Dijkstra算法的开销。hops指的是从源节点到当前节点的跳数。pred指向该节点的前驱结点。

63-69 一个节点链表，src指向源节点，dst指向目标节点。status指的是节点的状态，expires指的是时间期限。

75-77 cheapest\_node会记录花销最小的节点。

### 3.2.2 发送缓冲区Send Buffer

发送缓冲区指的是该节点不能发送的数据包队列，（数据包不能发送的原因是在这个节点处，数据包还没有找到能到达它目的节点的合适的路由）。节点会为这个数据包启动一次路由发现的过程，并且把数据包放到发送缓冲区的队列中。该结构的代码如下：

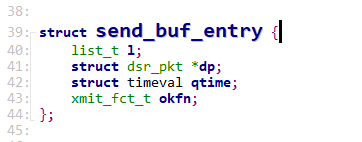


图3.6 发送缓存

40 l是链表。

41 dp是指向数据包的指针。

42 qtime是指数据包被放入Send Buffer中的时间。

发送缓存队列中的数据包会面临两种情况：

1. 如果路由发现过程找到了该数据包的路由，那么节点会把该数据包取出来。
2. 如果在规定的时间timeout之后，还没有找到路由，该数据包会被丢弃掉，以防止发生缓冲区溢出。

与发送缓存相关的函数有一些常规的操作，如创建发送缓存：***send\_buf\_entry\_create()***等等。

### 3.2.3 路由请求表Route Request Table

一个节点的路由请求表记录关于最近的由该节点发起或者是转发的路由请求的信息。表按照IP地址建立索引。

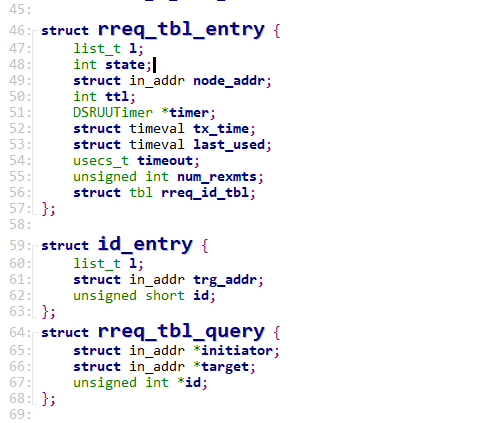


图3.7 路由请求表

Route Request Table记录了如下的信息：

46-57 rreq*tbl*entry记录由该节点发起的路由信息：

48 路由请求表的状态。

49 node\_addr是目的节点的IP地址。

50 TTL 生存时间。

53 last\_used此节点最后一次发起该目标节点的路由请求的时间，以此实现对请求时间的限制。

54 num\_rexmts是该节点自接收到有效的路由回复并给出到对应的目的节点的路由之后，为它的目标节点发起的连续路由发现的次数，也就是寻找路由的次数。以此实现对请求次数的限制。

59-63 id\_entry是记录经过该节点(也就是由该节点转发）的路由信息.

61 trg\_addr：是该节点转发的路由的目的节点的ip地址。

62 id是该路由的标识符。

DSR协议中的一些数据结构还包括：主动的路由应答表（Gratutious Route Reply Table）。这个结构是用于实现路由缩短这一个特定功能的。以及网络接口队列和维护缓冲区(Network Interface Queue and Maintenance Buffer）、黑名单(blacklist)等等。由于它们对于DSR协议的主要原理相关性不是那么密切，就不一一列举了。

### 3.2.4 DSR选项头

对于选项头的格式分析的流程主要是依照RFC当中的第六章的顺序。

DSR协议利用了一个特殊的头，它可以携带着DSR的控制信息，并且可以被包含在任何现有的IP数据包中。数据包中的DSR选项头包含一个固定大小部分（4个字节），后面跟着若干可选的DSR选项。对于IPv4来说，DSR选项头必须紧跟着IP选项头。（也就是说，在DSR选项头之后，才能插入其他协议，比方说TCP、UDP之类的协议的头部）。

在数据包插入DSR选项头之后，IP头的域会进行一些调整，比如IP数据包中的总长度会加上DSR选项头的总长度。针对路由选项的不同，所需要调整的IP域也会不同。在后面对于DSR选项的详细分析中，会给出说明。



图3.8 DSR选项头

29 nh是next header,也就是下一个报头。大小是8位。定义了紧跟着DSR选项之后的的哪个选项的头部。在插入DSR报头的时候直接赋值为之前IP数据包当中的next header的值。

29-30 小端序的相关定义。

31 res指的是Reserved，保留位。必须作为0发送，在接收的时候忽略。

32 标志位。必须设置为0。

33-35 大端序的相关定义。标志位和结构位的规则与小端序相同。

39 p\_len指的是payload length，有效载荷长度。这个字段的值定义了所有DSR选项的总长度（不包括固定部分）。

54 options[0]数组是DSR的可选项，是个变长的字段，它包含零到多个可选的选项。这个字段的长度真正的大小是DSR选项中有效载荷长度p\_len制定的。数组的大小是0，这里是为了实现结构体中变长的数组结构。因为可选选项的长度可以是0个或者多个，采用“动态大小”的数组就可以实现DSR可选选项的这种机制。使用TLV(type-length-value)的编码格式。

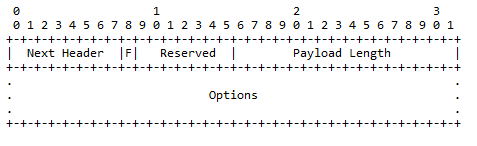


图3.9 DSR选项头

通过以上分析可以看出来，DSR选项头主要分为两个部分：一个是固定长度的DSR固定选项头，另一部分是不同的DSR选项。DSR选项头中的DSR选项的数量可以是零到多个，种类主要有：路由请求选项、路由回复选项、路由错误选项、确认请求选项等等。接下来我会逐个进行分析。

•固定的DSR选项头

大小是4个字节。包括了下一报头、标志位和保留位、有效载荷长度和DSR选项。

•路由请求选项 Route Request Option

路由请求选项最多在一个DSR选项头中出现一次。

加入这个选项会相应地对数据包的IP域进行更改：

源地址。设置成生成这个数据包的节点的IP地址。传输这个数据包的中间节点不能改变这个字段。

目的地址。由于路由请求是洪泛请求的，因此需要设置为广播地址（255.255.255.255）。

跳数限制。

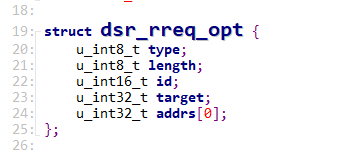


图3.10 路由请求选项

20 选项类型Option Type，8位的选项。

21 选项长度Opt Data Len，8位的选项。整个该DSR选项的长度。

22 路由请求的唯一标识，由源节点生成。这个唯一的标识可以支持重复检测的功能（如果收到了同样的路由请求，则会丢弃重复的路由请求）。

23 路由请求的目的节点的地址。

24 addrs[0]也是采用了变长数组的形式。addrs[1,...n]是路由请求选项中记录的第i个节点的的地址。1,...，n是路由地址的记录，包含目的节点。其中，节点1指的是在源节点之后的那个节点，n指的是目的节点。每一个转发路由请求的节点都会在这个记录中追加上自己的地址，并且把选项长度的值加4。

它的结构如下图所示：

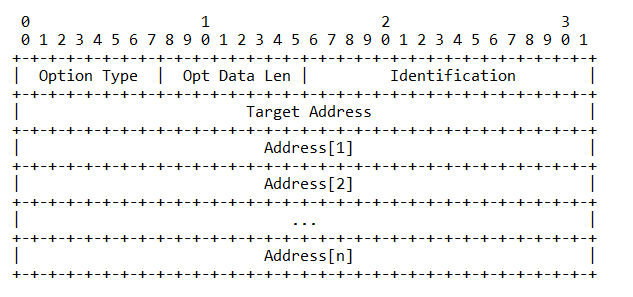


图3.11 路由请求选项

•路由回复选项 Route Reply Option

一个路由回复选项可以在DSR选项头中出现一到多次。

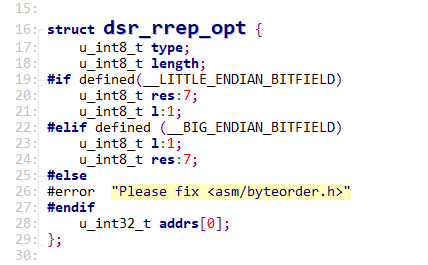


图3.12 路由回复选项

添加路由回复对IP域的更改主要有：

源地址：设置为发送这个路由回复的节点的地址。

目的地址：设置为这个路由回复的最终接收节点的地址，也就是这个路由回复要回复给的特定节点。

代码分析如下：

17 type是选项类型。

18 length是整个选项的长度。

19-21 小端的相关定义。

21 l是外部标志位。当l的值为1的时候，表示最后一跳在DSR网络的外部。在路由缓存中选择发送的数据包的时候，不会去选择l值为0的数据包。也就是说，会尽量选择在DSR网络内部的路由，而不去经过外部网络。

22-24 小端的相关定义。

25 addrs[0]变长的数组，用来记录节点的地址。

它的结构如下：

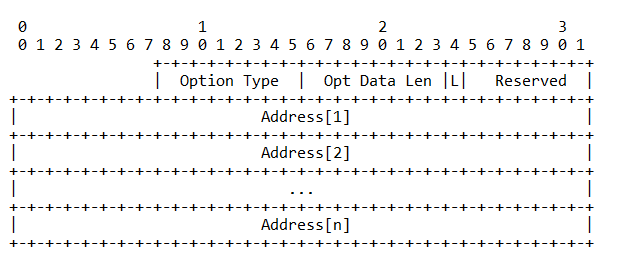


图3.13 路由回复选项

•路由错误选项

路由选项错误可以在一个DSR选项头中出现一次或者多次。

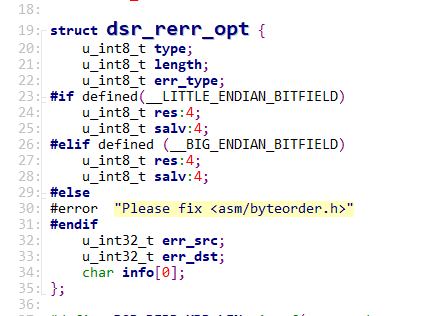


图3.14 路由错误选项

结构体的分析如下：

22 错误类型。常见的错误有类型有：当它的值为1表示NODE*UNREACHABLE，*节点不可达错误。值为2是FLOWSTATE*NOT*SUPPORTED不支持的流状态错误。值为3时是OPTIONNOTSUPPORTED不支持的选项错误。

25 salv是数据包被重发的次数的总和。用这个域来判断数据包是否达到了重发次数的上限，如果达到上限，就会丢弃数据包。

32 err\_src是错误源地址。

33 err\_dst是错误目的地址。

34 info[0]是特定的错误信息。针对不同的err\_type，就会存在相应的info。

路由错误选项的结构如下所示：

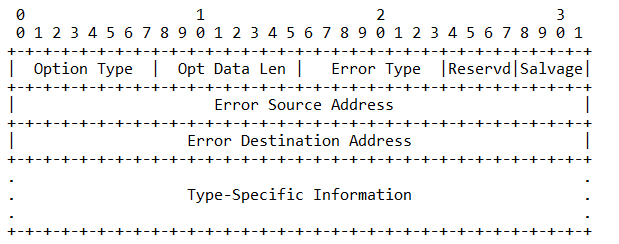


图3.15 路由错误选项

•ACK请求选项

一个ACK请求选项最多只能在一个DSR选项头中出现一次。

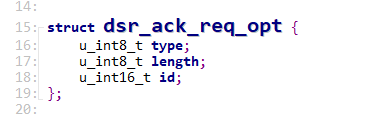


图3.16 ACK请求选项

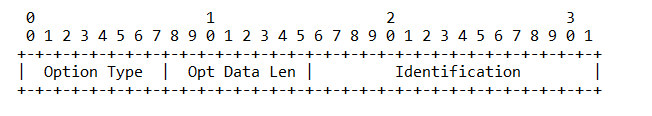


图3.17 ACK请求选项

•ACK选项

一个ACK选项可以在一个DSR选项头中出现一次或者多次。

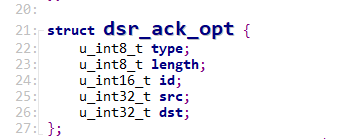


图3.18 ACK选项

它的结构如下：

24 id是从ACK请求选项中复制过来的。

25 生成ACK的节点地址。

26 ACK的目的节点的地址。

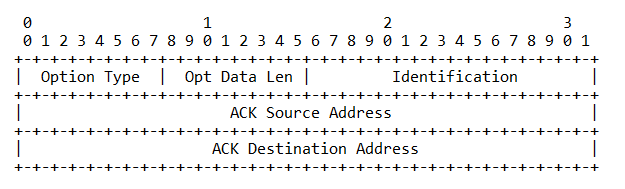


图3.19 ACK选项

•DSR源路由选项

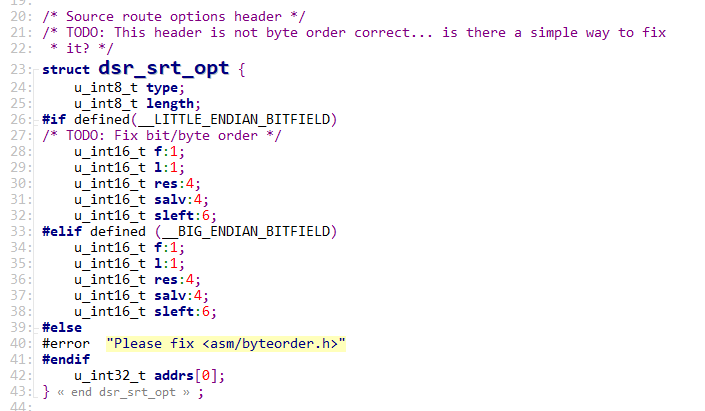


图3.20 DSR源路由选项

26-32 小端相关的定义。

28 f标志位。如果值为1，则表示路由的第一跳在DSR网络外部。

29 l标志位。如果值为1，路由的最后一跳到达了DSR网络外部。

31 salv是指数据包被重发的次数。

32 sleft指的是路由剩余的跳数。

33-38 大端相关的定义。

42 addrs[0]数组是路由信息。

它的结构如下：

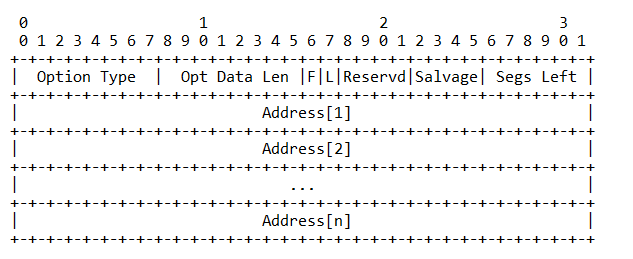


图3.21 DSR源路由选项

### 3.2.5 数据包



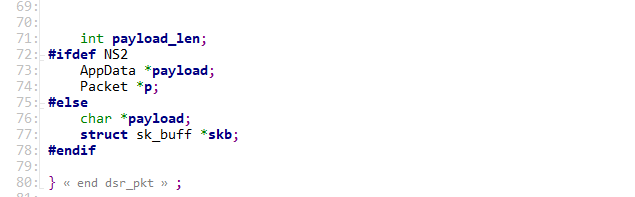


图3.22 DSR数据包

DSR数据包的结构体的定义如下：

26-31 IP域内相关数据的定义。

26 源IP地址。

27 目的IP地址。

28 下一跳的IP地址。

29 上一跳的IP地址。

31 重发次数。

37-41 nh联合体实现了对IP域数据的操作。

33-36 mc联合体实现了对MAC域的相关操作。

53-59 dh联合体实现了对DSR选项的操作。

62 源路由选项。

63 路由请求选项。

64 路由回复选项。

65 路有错误选项。

66 确认选项。

67 确认请求选项。

73 负载长度。

这样的数据包的结构使得不仅可以对DSR选项进行处理，还可以对IP域的值进行处理。

# 第四章 DSR具体实现

分析代码的顺序主要是按照RFC协议来进行的，分析的大致思路是按照RFC协议的第八章：Detailed Operation的思路进行分析，并结合自己的一些理解进行了修改。

## 4.1 一般数据包处理

### 4.1.1 生成一个数据包

当生成任何一个数据包时，使用DSR协议的节点必须执行以下步骤：

* 数据包的首部中包含了目的结点的地址，在这个节点的路由缓存中寻找一条到目的节点的路由。
* 如果在路由缓存中找不到，那么启动到目的节点的路由发现过程。（启动到目的节点地址的路由发现过程, 那么该节点会在这个数据包的DSR选项头中添加路由请求选项, 或者将数据包保存到它的发送缓存中并启动路由发现过程, 发送一个单独的包含这样一个路由请求选项的数据包。）如果节点选择第一种方式: 它将会用广播地址(255.255.255.255)取代IP目的地址,也就是以洪泛的形式发送。同时把原来的IP目的地址复制到目的地址字段。
* 如果数据包不包含一个路由请求选项，那么这个节点一定会有一条到目的地址的路由。如果这个节点有多条到达该目的地址的路由，那么这个节点会选择一条路由用于此数据包。如果这条路由长度大于1跳，或者节点决定从该路由的第一跳节点请求一个DSR网络层确认，那么就会添加一个DSR选项头到数据包当中,并且插入一个DSR源路由选项。 数据包中的源路由被初始化为了从所选路由到包的目的地址。
* 将数据包传输到选定源路由中给定的第一跳地址,使用路由维护来决定下一跳的可达性。

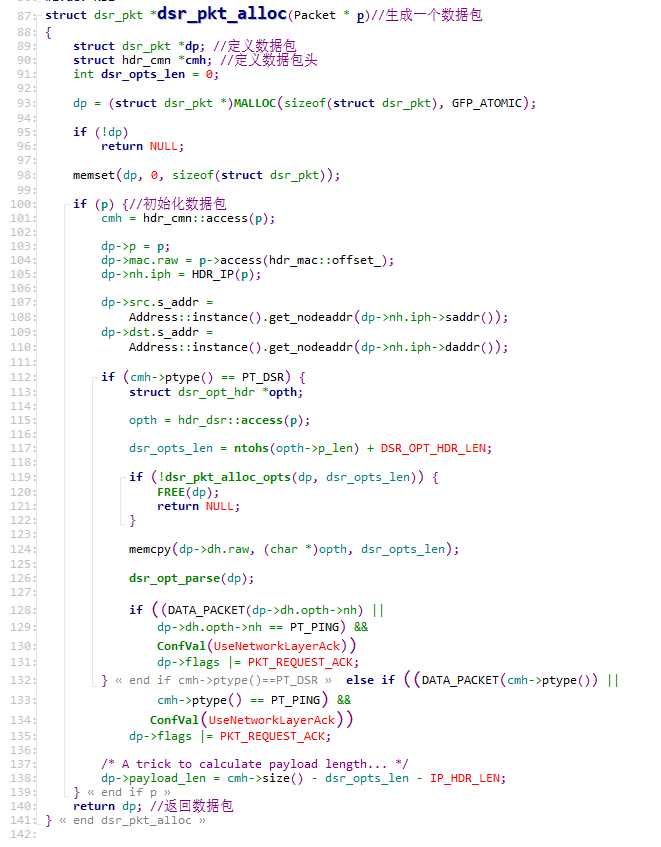


图4.1 生成数据包

89 定义一个数据包

90 定义数据包首部

100-139 对数据包进行初始化，包括初始化数据包dp的mac联合体部分、nh联合体（IP域）部分，进行了对标志位赋值、对各个长度字段进行赋值等操作。

### 4.1.2 给数据包添加一个DSR选项头

生成数据包的节点向数据包中添加DSR选项头, 用来携带路由协议所需的信息。一个数据包只能包含最多一个DSR选项头。DSR选项头添加过程如下:

* DSR选项头插入的位置是在IP头之后,其他所有可能出现的协议头之前。
* 将DSR选项头中的Next Header字段设置为数据包的IP头中的协议编号字段。
* 将包的IP头中的协议字段Protocol field设置为DSR协议编号。

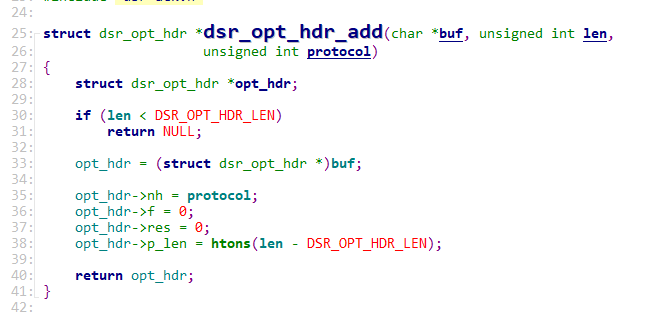


图4.2 给数据包添加一个DSR选项头

30-31 　检查长度是否符合标准，因为DSR选项头有一个固定长度的首部，因此长度一定大于4个字节。

33 　把DSR选项头中的Next Header字段者之谓数据包的IP头中的协议编号字段protocol。

35-38 　给DSR选项头的各个部分赋值。把f标志位置0，res保留位也设为0。使用了htons函数，把将主机的无符号短整形数转换成网络字节顺序。

### 4.1.3 给数据包添加一个DSR源路由选项

DSR源路由选项头是由生成数据包的节点添加的，为了包含从源节点到目的节点的源路由信息。步骤如下：

* 源节点生成一个DSR源路由选项，把它添加到数据包的DSR选项头中。
* Address[i]字段是数据包的源路由。DSR源路由选项中的left field字段初始化的时候被设为n。left field字段记录的是到达目的节点的过程中还需要访问的节点的数量。
* 数据包的源路由内的地址被复制到DSR源路由选项中的Address[i]字段中。

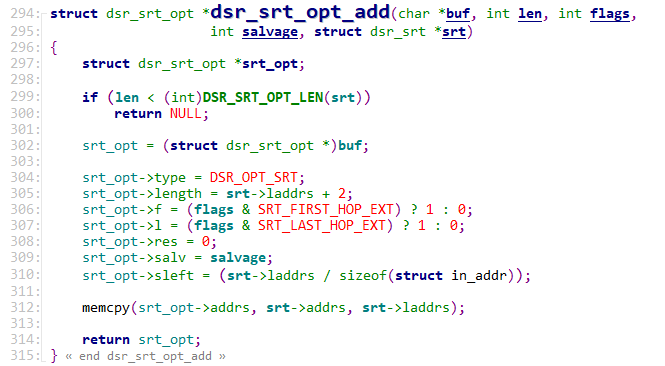


图4.３给数据包添加一个DSR源路由选项

310　 sleft字段是到达目的节点还需要访问的节点的数量。

312 　把DSR源路由中的addrs地址复制到DSR源路由选项的地址字段。

### 4.1.4 处理接收到的数据包

　　这个部分指的是一个节点接收到任何数据包后的处理操作，不管当前的节点是作为中间节点转发这个数据包、或者是监听到了这个数据包，或者是作为目的节点收到这个数据包，只要数据包包含一个DSR选项头，那么这个节点就需要进行如下操作：

　　如果DSR选项头包含一个路由请求选项。节点会把源路由从路由请求中提取出来,把路由信息放到路由缓存当中.n指的是路由请求选项中记录的节点地址的数量. 在更新完路由缓存之后, 节点要进行路由请求选项的处理。

　　如果DSR选项头包含一个路由回复选项。从路由回复中提取源路由信息,并把路由信息添加到路由缓存中。

　　同样的,如果DSR选项头包含一个路由错误选项\ACK请求选项\ACK选项和DSR源路由选项时,进行相应的处理。它们的处理过程都大致相似,但是有一个不同点:Address数组中的n个节点地址,有的是包含源节点和剩余的n-1个中间节点的,另一种情况是包含目的节点和剩余n-1个中间节点的。



图4.４ 处理接收到的数据包

35　接收到一个数据包后, 首先调用***dsr＿opt＿recv()***函数,对数据包的DSR选项进行判断和处理。

43-111　对包含不同选项的数据包进行处理。



图4.５ 处理接收到的数据包

66-78　 对TTL生存时间进行判断,如果TTL为0,就丢弃数据包,否则,就调用***XMIT()***函数,对数据包进行转发。

89-99 　把当前节点的地址追加到数据包的地址字段。

### 4.1.5 处理接收到的DSR源路由选项

　　当节点接收到包含DSR源路由选项的数据包时,首先去判断这个数据包有没有机会进行路径缩短. 即如果收到这个数据包的节点不是下一跳,但是被包含在后面的路由信息当中,那么就可以进行自动路由缩短过程。



图4.6 自动路由缩短

433-435　如果该数据包的下一跳不是这个节点并且数据包剩下的路由当中有当前的节点,就去查找它的主动路由应答表,进行路由缩短。

463　发送一个路由回复,去通知源节点这个路由的信息发生了变化,进行了缩减。

## 4.2 路由发现

　　如前面基本原理中分析,路由发现是一种"按需"的过程。

　　路由发现过程涉及到对两种信息的处理:路由请求和路由回复。

### 3.2.1 生成一个路由请求

　　一个节点生成并且初始化在IP数据包的DSR选项头中的路由请求选项. 路由请求选项必须紧跟在在DSR选项头之后。

　　IP头的源地址设置为源节点的IP地址, 目的地址设置为广播地址255.255.255.255。

这个节点必须维护自己的路由请求表. 当生成一个新的路由请求时, 这个节点需要在路由请求表中添加这个路由请求的记录.设置的内容包括生存时间TTL, 目标节点的路由请求表为此目标节点发起的连续的路由请求的数量等等。



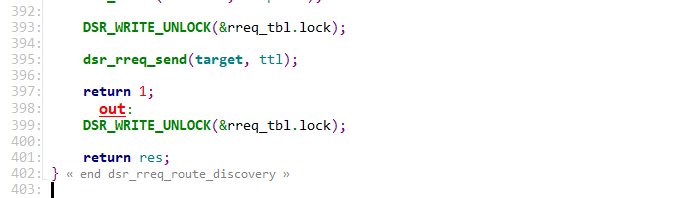


图4.7 生成一个路由请求

344　指向路由请求表。

346　 expires是失效时间变量,如果超出这个时间,就会在路由请求表中删除掉这个请求。

352 　调用***\_tbl\_find()***函数,如果在路由请求表中找目标节点。

355-356 　如果路由请求表中找不到相应的记录,则调用***\_rreq\_tbl\_add()***函数添加新的记录。

357-361 　调用***\_tbl\_detach()***删除原来的路由请求,调用***\_tbl\_add\_tail()***函数把记录添加到表的尾部。

368-372 　如果当前已经处在路由发现的状态中,则直接返回。

375-391　 对时间相关的操作,对时间进行更新,如果超时则会进行相应的处理

395 调用***\_dsr\_rreq\_send()***发送路由请求数据包。

393-399 　读写锁,保护路由请求表的读写操作的一致性。

### 4.2.2 处理一个接收到的路由请求选项

　　一个节点接收到了一个包含路由请求选项的数据包之后的处理过程如下:

* 如果这个节点的IP地址与路由请求选项中目的节点的IP地址一致,那么这个节点发送一个路由回复。
* 如果不同,那么节点需要进行重复检测.如果收到过相同的路由请求,那么丢弃这个数据包。
* 通过了重复检测的数据包,需要进行:把这个路由请求添加到缓存中, 把自己的地址追加到数据包的addrs中,在自己的路由缓存中搜索是否有到达目的节点的路由,如果没有,就以广播的实行发送这个数据包,如果有,则发送"缓存路由回复"给源节点。



图4.8 处理一个接收到的路由请求选项

515-518　 如果收到的数据包中有超过一个路由请求,那么报错。

526-529 　调用dsr*rreq*duplicate()函数进行路由请求的重复检测. 如果重复,则会丢弃这个数据包。

531　 把这个路由请求添加到路由请求表中。

533 　调用dsr*srt*new()函数,建立新的源路由。

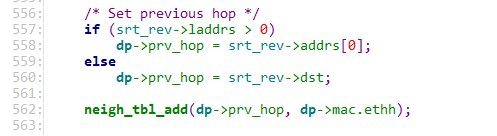


图4.9 处理一个接收到的路由请求选项

557-560 设置上一条的信息。

562 调用neigh*tbl*add()在上一跳,也就是邻居节点的表中添加信息。

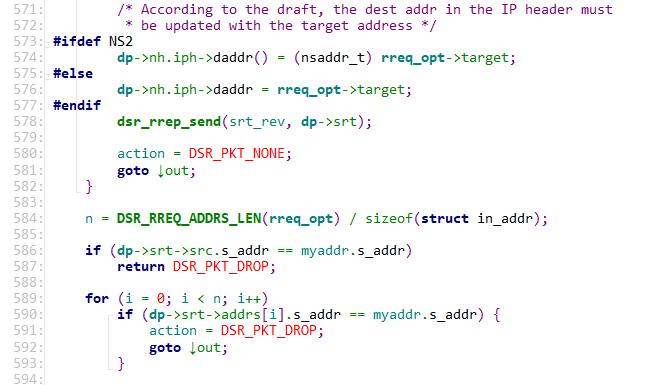


图4.10 处理一个接收到的路由请求选项

571-593 更新数据包的地址信息。

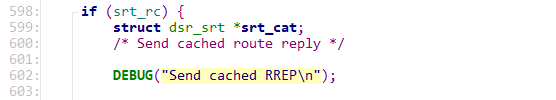


图4.11 处理一个接收到的路由请求选项

598-602 发送缓存路由回复。



图4.12 处理一个接收到的路由请求选项

626 回复路由请求数据包。

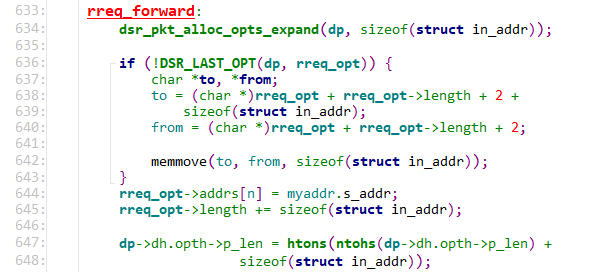


图4.13 回复路由请求选项

633-648 调用***dsr\_pkt\_alloc\_opts\_expends()***分配更多的内存空间。在路由请求数据包中追加当前节点的信息,并且把这个数据包转发出去。

### 4.2.3 发送一个路由回复

一个节点生成一个路由回复来应答收到的路由请求.这个数据包只需要在DSR选项头中包含路由回复选型头即可。

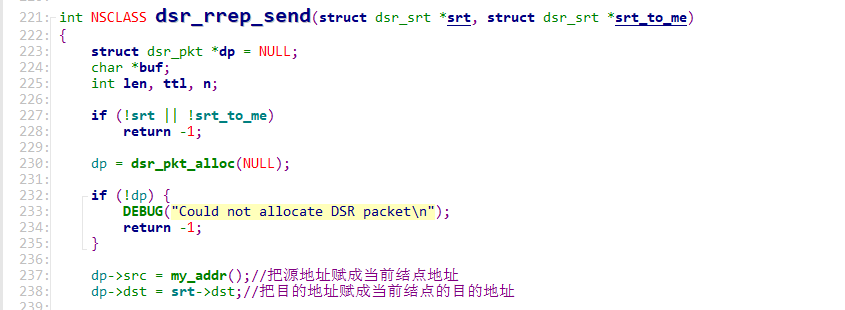


图4.14 发送一个路由回复

230 生成一个书包并且初始化。

237 把源地址设为当前节点地址。

238 把目的地址设成当前节点的目的地址。



图4.15 发送一个路由回复

258 把路由回复的TTL设置成为ｎ＋１，确保路由回复到达源节点。



图4.16 发送一个路由回复

318 调用***XMIT()***函数进行转发数据包。

### 4.2.4 防止路由回复风暴

节点对于路由请求的回复可能会造成路由回复风暴. 比方说一个节点广播发送了一个路由请求,那么它的所有邻居节点都会收到这个路由请求.如果这些邻居节点的路由缓存中都有到这个节点的路由信息,那么每一个邻居都会向源节点发送一个路由回复。

这种现象有两个缺点。一是会造成带宽的浪费,引起资源浪费;二是可能会引发网络的拥塞。

为了减少这些影响,如果一个节点可以把他的网络结构设置为混杂的接收模式,那么它可能会在段时间内延迟发送自己的路由应答.同时,这个节点会侦听初始节点是否首先使用更短的路由.也就是说,这个节点延迟发送自己的路由应答,这个延迟的时间是随机的。

它的计算方式如下:

d = H \* (h - 1 + r)

其中, h是节点要发送的路由回复中路由的跳数. r是一个随机的浮点数, 在0到1之间. H是一个小的常数, 代表恒定的延迟。

### 4.2.5 处理一个接收到的路由回复选项

一个节点接收到路由回复选项后的处理非常简单. 它只需要检查自己的发送缓存是否包含这条路径. 如果包含,那么就发送该数据包并且从发送缓存中移除。

## 4.3 路由维护

路由维护主要完成两个任务:

1. 当源节点给目的节点发送数据时监测当前路由的可用情况。
2. 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的过程。

路由维护的具体实现机制主要依赖于确认和路由错误.在数据包传输之后,如果没有收到确认,则节点就会确定路由的下一跳是断开的状态.这种确认机制可以使用链路层确认、被动确认和网络层确认来实现. 如果没有收到确认,那么节点应该向包的发送路由错误。

### 4.3.1 确认

路由维护中的确认机制有三种:链路层的确认、被动的确认和网络层的确认.

* 链路层的确认

如果MAC协议提供成功传输数据包的反馈,那么就无需使用DSR的ACK确认请求和ACK确认选项. 如果链路层的返回是有效的,那么就不应该再用其他的路由确认机制了，不需要用被动的确认,也不需要借助网络层的确认。

使用链路层确认进行路由维护时，重传时间和重传尝试预定的时间一般都由网络层总使用特定的链路层进行控制。

当一个节点收到其路由维护缓冲区中的数据包的链路层确认时，该节点应该将该数据包以及具有相同下一跳目的地的路由维护缓冲区中的所有包删除掉。

* 网络层的确认

主要依赖于数据结构部分定义的ACK请求选项和ACK选项来实现。

### 4.3.2 生成一个路由错误

当一个节点在达到最大的重传尝试次数之后,无法验证下一跳的可达性时，节点应该向数据包的IP源地址发送一个路由错误。当发送包含路有错误选项或者确认选项的数据包时，节点应该将这些选项添加到路由错误中。



图4.17 生成一个路有错误

fig:

图4.18 生成一个路由错误

65-86 创建一个IP数据包，并把IP头中的源地址设置为发送路由错误节点的地址。

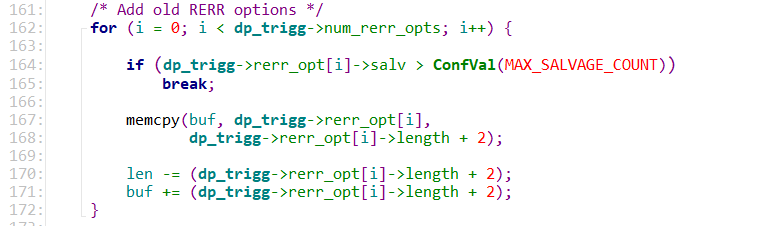


图4.19 生成一个路由错误

161-172 插入路由错误选项到这个新的数据包中，设置错误类型为***NODE\_UNREACHABLE***, 设置错误源地址为该节点的IP地址,设置错误目的地址域为新的数据包的IP目的地址，但是需要满足一些约束条件。

最后调用***XMIT()***函数把数据包转发出去.

### 4.3.3 处理一个接收到的路由错误选项

当一个节点收到一个包含路由错误选项的数据包时，它要进行以下的处理:



图4.20 处理一个接收到的路由错误选项

215 选择该路由错误选项的错误类型,进行相应的处理。

217-227 当错误类型是NODE*UNREACHABLE时,节点必须从其路由缓存中删除起于错误源地址节点终于不可达节点的路由记录.通过调用maint*buf*del*all()实现删除操作。

230 调用***lc\_link\_del()***函数把断的链路从缓存中移除出去。

此外当处理完上面的路由错误选项之后,这个节点可能还会启动一次新的目的节点为任意节点的路由发现过程。

### 4.3.4 重发一个数据包

如果一个中间节点在转发数据包的时候，发现该数据包到它的下一跳的链路被破坏，但是该节点自己的路有缓存中存在一条可以到达这个数据包的目的节点的路由，那么节点应该重发这个数据包。

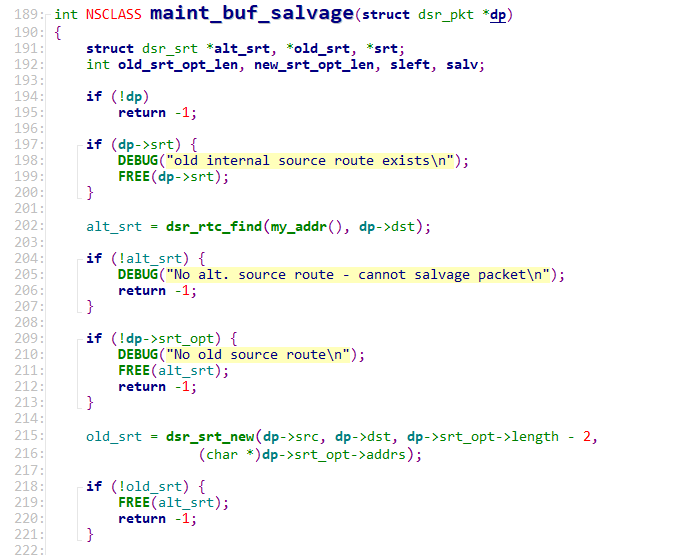


图4.21 重发一个数据包

194-213 检查数据包的下一跳是否被破坏。

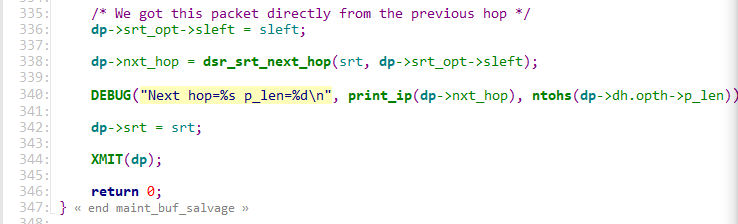


图4.22 重发一个数据包

223-347 在用路由缓存中通的路由信息替换掉数据包中原来的断路的路由信息,之后重新发送数据包。

# 第五章 小结

经过八周的协议栈的课程以及上机，我完成了DSR协议的分析。通过分析我明白了DSR这个简单高效的路由协议的运行原理和过程。

我主要是按照RFC文档提供的逻辑顺序去分析了这个协议。通过DSR协议的分析主要是由三部分组成：首先，通过阅读RFC文档和查阅相关资料，了解DSR协议的算法基本原理和基本组成。然后，对DSR协议使用的数据结构进行分析，重点分析了DSR选项头中的多种选项。最后，对DSR协议的函数进行分析，包括对函数的调用关系、函数的整体功能、函数的输入输出和函数的实现细节，对照DSR基本原理理解，更深入理解了DSR协议的实现。并在分析过程中绘制函数关系的调用图。

但是，大作业只是分析了DSR协议的主体内容，对于代码中的函数也只是重点关注了函数的功能，函数的输入输出，而没有具体去考虑DSR协议的一些细节和巧妙的功能。也并没有覆盖到所有的函数的分析，对于一些常规的、不重要的操作，进行了一些简单的调用关系的分析或者进行了省略。针对协议的分析主要思路为针对包含不同DSR选项的数据包的发送和接收处理, 以及DSR协议的一些特殊的功能,比如路由发现中的防止路由回复风暴等。