**网络协议栈分析与设计课程大作业**

|  |
| --- |
| **OLSR路由协议代码分析** |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | 学号 | 姓名 | 班级 | 负责模块 | 成绩 | | 201692037 | 张晓萌 | 软网1603 | 1. 引言、总结、整理文档 2. MID消息相关 3. HNA消息相关 4. 信息表 5. 拓扑建立、路由算法 |  | | 201492240 | 许楚薇 | 软网1601 | 1.HELLO消息相关  2.TC消息相关  3.OLSR首部  4.代码介绍  5.协议实现函数总结 |  | |

目录

[1 引言 2](#_Toc8726)

[1.1 协议介绍 2](#_Toc23989)

[1.2 OLSR的术语 2](#_Toc29256)

[1.3 OLSR基本原理 3](#_Toc26477)

[2. 代码介绍 5](#_Toc5074)

[2.1 文件介绍 5](#_Toc15828)

[2.2 全局变量 5](#_Toc31276)

[3 数据结构 6](#_Toc32319)

[3.1 Olsr首部 6](#_Toc20004)

[3.2 HELLO消息 8](#_Toc29641)

[3.3 TC消息 9](#_Toc1486)

[3.4 MID消息 11](#_Toc15892)

[3.5 HNA消息 12](#_Toc9397)

[3.6 信息表 14](#_Toc29478)

[3.6.1 邻居表 14](#_Toc11979)

[3.6.2 MPR Selector表 14](#_Toc15652)

[3.6.3 拓扑表 15](#_Toc27567)

[3.6.4 路由表 15](#_Toc15100)

[4 路由协议介绍 17](#_Toc17722)

[4.1 协议实现 17](#_Toc25402)

[4.2 邻居发现处理 17](#_Toc9086)

[4.2.1 HELLO消息生成 17](#_Toc25239)

[4.2.2 邻居表操作 18](#_Toc31120)

[4.2.3 邻居节点操作 18](#_Toc31755)

[4.3 MPR 19](#_Toc4568)

[4.3.1 MPR节点操作 19](#_Toc24168)

[4.3.2 MPR算法 20](#_Toc24001)

[4.4 MID消息 22](#_Toc17350)

[4.4.1 消息生成 22](#_Toc27642)

[4.4.2 消息处理 23](#_Toc4419)

[4.5 拓扑建立 23](#_Toc424)

[4.5.1 TC消息生成 24](#_Toc8684)

[4.5.2 TC 消息处理 24](#_Toc30261)

[4.5.3 拓扑表处理 26](#_Toc32203)

[4.6 路由算法 28](#_Toc11945)

[4.7 HNA消息 38](#_Toc1634)

[4.7.1 消息生成 39](#_Toc15530)

[4.7.2 消息处理 39](#_Toc11306)

[5 总结 40](#_Toc1685)

## 引言

### 1.1 协议介绍

OLSR路由协议是IETF MANET工作组为无线移动Ad Hoc（无线自组）网提出的作为RFC标准化的一种表驱动先应式路由协议。该协议在传统的链路状态算法基础上采用MPR(multipointreplay，多点中继)机制来减少协议开销。

OLSR协议中每个节点通过周期性地交换链路状态信息来维护整个网络的拓扑信息。节点周期性地向邻居节点广播HELLO消息以得到邻居及链路状态，节点在接收到HELLO分组后，启动HELLO消息处理过程，判断邻居节点的链路状态并维护自己的一跳邻居表和二跳邻居表，HELLO分组只在产生HELLO分组的节点的一跳范围内传播。

每个节点为了避免消息在网络中的洪泛，只选用一跳邻居中的部分节点作为MPR节点，这些节点形成多点中继集。只有MPR节点才会向网络中发送TC控制消息并且参与路由计算，而其他不被任何节点选做MPR节点的节点则不参与路由计算也不发送和转发TC消息。MPR机制削减了整个网络中洪泛的TC消息的数量和转发次数，大大降低了协议的开销。

TC消息的作用是声明节点MPR信息，网络中节点周期的发送TC分组，以声明自己的MPR Selector集，如果此节点没有被任何节点选做MPR，则它的MPR Selector集为空,此节点不发送TC分组。节点的TC分组需要扩散至整个网络中的所有节点。

当某个物理节点包含多个参与OLSR的接口时，他必须周期性地发送MID（Mutliple Interface Declaration）消息，节点接收到MID消息之后会将MID消息中包含的多个接口的地址和主地址填充进拓扑集。节点通过MID消息获得多端口信息。

为了提供注入外部路由的这种能力，信息进入OLSR MANET，具有这种非MANET接口的节点，定期发布HNA（Host and Network Association）消息，其中包含足够的信息，以便收件人能够构造一个合适的路由表。HNA将OLSR的移动自组网接口与没有接口的OLSR相连。

### 1.2 OLSR的术语

|  |  |
| --- | --- |
| node | 一种MANET路由器，实现OLSR路由协议 |
| olsr interface | 参与运行OLSR的MANT的网络设备。一个节点可以有几个OLSR接口，每个接口分配一个唯一的IP地址 |
| single OLSR interface node | 具有单个OLSR接口的节点，参与OLSR路由域 |
| multiple OLSR interface node | 具有多个OLSR接口的节点，参与OLSR路由域。利用MID消息声明 |
| main address | 一个节点的主地址，用于OLSR控制流量，作为该节点发出的所有消息的“始发地址”。它是节点的OLSR接口之一的地址 |
| neighbor node | 一跳邻居节点，只有一条链路在节点X的OLSR接口到另一节点Y的OLSR接口之间 |
| 2-hop neighbor | 两跳邻居，被邻居节点侦听到的节点（不包括本身） |
| multipoint relay (MPR) | 从一跳邻居节点中选出，作为中继节点，可以减少洪泛 |
| multipoint relay selector (MPR selector, MS) | 此节点被节点A选为MPR节点，那么节点A为此节点的MPR Selector |
| link | 两个不同节点上的OLSR接口可以相互听到 |
| symmetric link | 对称链路，在两个OLSR接口之间验证的双向链接 |
| asymmetric link | 两个OLSR接口之间的链接，仅在一个方向中验证。 |

### 1.3 OLSR基本原理

1. 邻居侦听和链路检测。节点周期性地向邻居节点广播HELLO分组以得到邻居节点和链路状态。

链路状态判断：节点B向邻居节点广播HELLO分组，A接收到HELLO分组，A将B放到自己的邻居节点中，并将B标记为非对称；A向邻居节点广播HELLO分组，节点B接收到HELLO分组，B接收到分组时标记到节点A的链路状态为对称的；B再次广播HELLO分组，A接收到分组，标记到B的链路状态为对称。

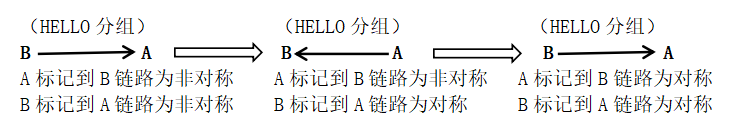


图 1-1

1. MPR处理。节点根据已知的邻居节点和链路状态计算自己的MPR集合，然后将结果送给HELLO处理函数，HELLO处理函数维护邻居表，被选为MPR节点的邻居表项信息is\_mpr置为true。
2. MPR信息。在发出的HELLO消息中声明MPR信息，接受节点收到HELLO消息查看自己是否被选为MPR节点并维护MS信息。

（4）TC消息发送。如果此节点没有被任何节点选做MPR，则它的MPR Selector集为空,此节点不发送TC分组。如果节点MS表不为空，那么节点需要向全网发送TC消息，TC分组中包括MPR Selector节点的地址。

（5）TC消息转发。只有当消息的产生者是自己的MPR Selector节点，并且TC消息的存活时间大于0时，接受节点才转发该TC消息。

（6）路由表计算。每个节点根据自己收到的TC分组和邻居信息计算出整个网络的拓扑信息。每个节点维护一张包含各个目的节点的路由表，节点通过路由表寻找到目的节点的路径信息，如果网络发生变化，路由表要随之更新。计算路径信息采用Dijkstra的最短路径优先算法。

（7）MID消息发送处理。当某个物理节点包含多个参与OLSR的接口时，他必须周期性地发送MID（Mutliple Interface Declaration）消息，节点接收到MID消息之后会将MID消息中包含的多个接口的地址和主地址填充进拓扑集。

（8）HNA消息。为了提供注入外部路由的这种能力，信息进入OLSR MANET，具有这种非MANET接口的节点，定期发布HNA（Host and Network Association）消息，其中包含足够的信息，以便收件人能够构造一个合适的路由表

## 2. 代码介绍

### 2.1 文件介绍

|  |  |
| --- | --- |
| 文件 | 说明 |
| lq\_packet.h | Olsrt首部结构，HELLO，TC消息首部结构 |
| mid\_set.c | MID消息初始化，处理 |
| hna\_set.c | HNA消息初始化，处理 |
| mpr.c | MPR消息处理 |
| Link\_set.c | 节点信息结构 |
| Neighbor\_table.c | 邻居节点结构，邻居表处理 |
| Routing\_table.c | 路由表结构与处理 |
| Generate\_msg.c | 消息处理 |
| Main.c | 主函数 |
| Olsr\_cfg.c | 定义大部分变量 |
| Olsr.c | 实现一些全局函数 |
| mpr\_selector\_set.c | 对MS集合的操作 |
| olsr\_spf.c | 实现路由算法 |
| packet.c | 定义消息数据包的结构 |
| process\_package.c | 对HELLO消息的处理 |
| scheduler.c | 定时器的各种操作 |
| tc\_set.c | TC消息的处理 |
| two\_hop\_neighbor\_table.c | 两跳邻居表的处理 |

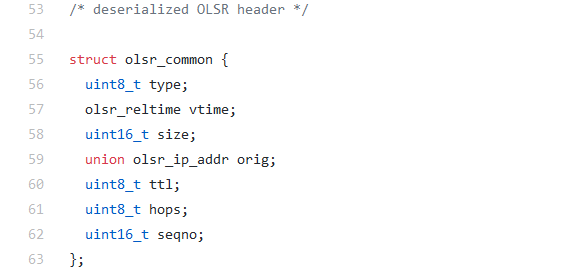
### 2.2 全局变量

|  |  |  |
| --- | --- | --- |
| 变量 | 数据类型 | 说明 |
| changes\_topology | Bool | 拓扑信息是否变化 |
| changes\_neighborhood | Bool | 邻居表是否变化 |
| olsrport | Uint16\_t | Olsr路由端口 |
| willings | nit8\_t | 意愿程度 |
| changes\_neighborhood | bool | 邻居表信息是否变化 |
| changes\_topology | bool | 拓扑信息是否变化 |
| changes\_force | bool | HNA是否发生变化 |

## 3 数据结构

### 3.1 Olsr首部

lq\_packet.h



lq\_packet.h

olsr\_common是olsr数据包消息首部的基本结构。

53-63：基本内容为：type：消息类型；vtime:本条消息中的内容在多长时间内可以保证真实有效；size：消息的大小；orig：消息的发送端地址；ttl：跳数，本条消息的寿命，每经过一个节点，消息的跳数减一，当跳数为0，则不再转发；hops：目前本消息已经经过的跳数；seqno：消息序列号，唯一标识本消息，保证消息不会被发送回来。

olsr协议使用同样的数据包头，保证消息的兼容性，一个分组中可以包含多个不同的消息。olsr协议分组的基本结构如下图：

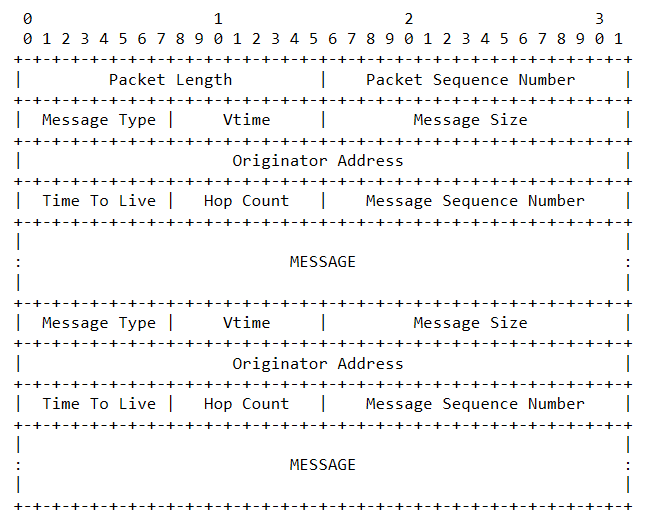


图3.1 olsr分组结构

分组中各模块内容为：

Packet Length：分组的总长度，以byte为单位。

Packet Sequence Number：分组序列号。

Message Type：消息类型。

Vtime：消息有效时长。

Message Size：消息的长度。

Originator Address：消息发送端地址。

Time To Live：消息被传送的最大跳数。

Hop Count：目前为止消息经过的跳数。

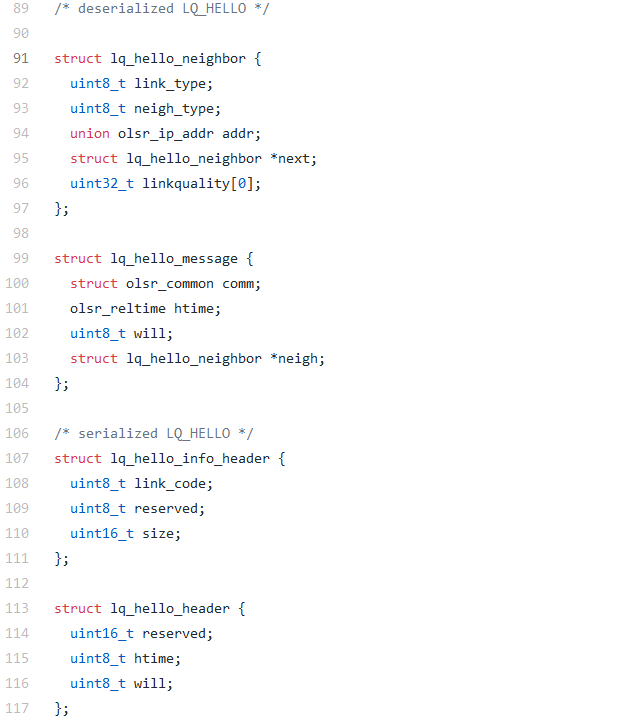
Message Sequence Number：消息序列号。

Olsr协议由HELLO控制分组和TC控制分组两个控制分组，HELLO分组的作用是链路检测和发现邻居，TC控制分组的作用是声明MPR信息。TC分组包含拓扑信息，节点通过该信息计算路由。

### 3.2 HELLO消息

HELLO消息用于建立一个节点的邻居表，报文中可以包括邻居节点的地址以及本节点到邻居节点的开销，olsr采用周期性的广播HELLO分组来侦听邻居节点的状态。HELLO分组只在一跳的范围内广播。

lq\_packet.h



lq\_packet.h

91-97: lq\_hello\_neighbor是邻居节点集的头部结构体，基本内容为：link\_type：连接类型；neigh\_type：邻居类型；addr：地址；next：消息传送的下一个节点；linkquality：链路质量，开销。

99-104：lq\_hello\_message是hello消息数据包的头部结构体，基本内容为：comm：olsr协通用消息头；htime：hello消息的发送时间间隔；will：节点意愿，一个节点是否愿意为其他节点携带流量。willing\_nerver:永远不会成为MPR；willing\_always 永远成为MPR，默认为willing\_default。

107-117：lq\_hello\_info\_header和lq\_hello\_header一起组成了hello消息的头部。如下图所示

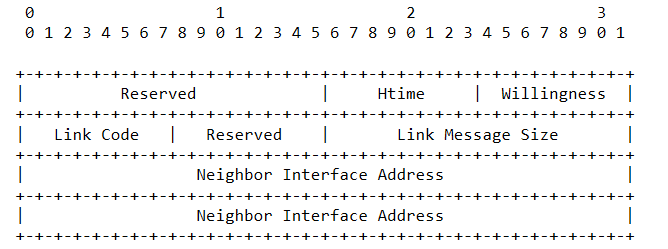


图3.2 Hello消息

Hello消息各模块为：

Reserved：保留字，初始化为0。

Htime：hello消息发送间隔。

Willingness：节点意愿，一个节点是否愿意为其他节点携带流量。willing\_nerver:永远不会成为MPR；willing\_always 永远成为MPR，默认为willing\_default。

Link Code：发送端和邻居列表中的邻居节点之间的链路类型.

ASYM\_LINK：非对称，确认可以接受，无法确定是否成功发送。SYM\_LINK：对称，链路双方双向确认可以接受。MPR\_LINK：列表节点被选为MPR。

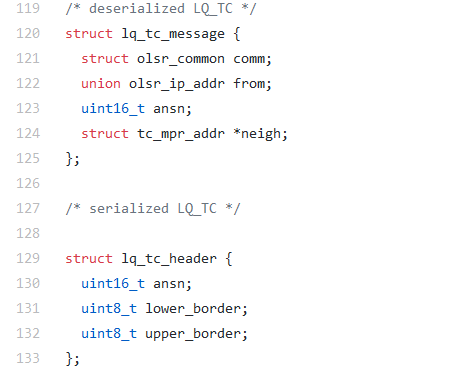
Link Message Size：从链路类型到下一个链路类型字段的消息大小。

### 3.3 TC消息

TC (Topology Control)消息:声明MPR信息，包含拓扑结构信息，通过该信息计算路由。网络中节点周期的发送TC分组，以声明自己的MPR Selector集，如果此节点没有被任何节点选做MPR，则它的MPR Selector集为空,此节点不发送TC分组。节点的TC分组需要扩散至整个网络中的所有节点。

和Hello分组不同，TC分组必须被广播到全网。

lq\_packet.h

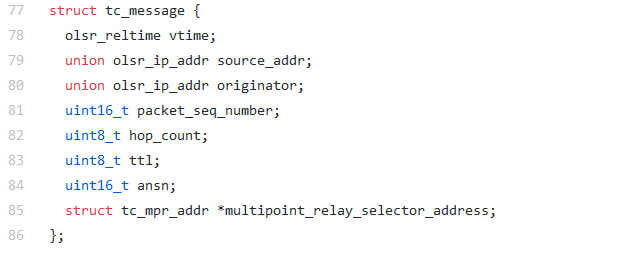


lq\_packet.h

120-125：lq\_tc\_message是封装后的TC数据包结构体。内容有：from：到达目的地的倒数第二跳地址；ansn：本节点最近接受的一个TC分组的ANSN 号； neigh：广播邻居集。

129-133：lq\_tc\_header是TC数据包头。内容有：ansn：本节点最近接受的一个TC分组的ANSN 号；lower\_border：下一级的边界；upper\_border：上一级的边界。

packet.h



packet.h

77-86：TC消息结构体。ansn：广播邻居节点序列号；multipoint\_relay\_selector\_address：MPR Selector地址集合。

TC分组的结构如下图所示：

0 16 32

|  |  |
| --- | --- |
| MSSN | Reserved |
| MPR Selector Address | |
| MPR Selector Address | |
| ... | |

图3.3 TC分组结构

TC分组各部分为：

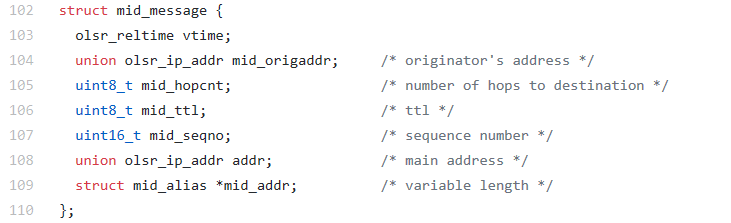
MSSN :MPR Selector 序列号，与MPR Selector集合相对应，集合变化时，MSSN更新。

### 3.4 MID消息

当某个物理节点包含多个参与OLSR的接口时，他必须周期性地发送MID（Mutliple Interface Declaration，多接口声明）消息。

节点接收到MID消息之后会将MID消息中包含的多个接口的地址和主地址填充进拓扑集。节点通过MID消息获得多接口信息。

packet.h



packet.h

102-110：MID消息结构体。vtime：时间戳，用于判断消息是否过期；

mid\_origaddr：始发者（最初发送该MID消息的节点，MID消息会经过多次转发）的id地址，包括ipv4地址和ipv6地址；

mid\_hopcnt：到接收方节点的跳数；

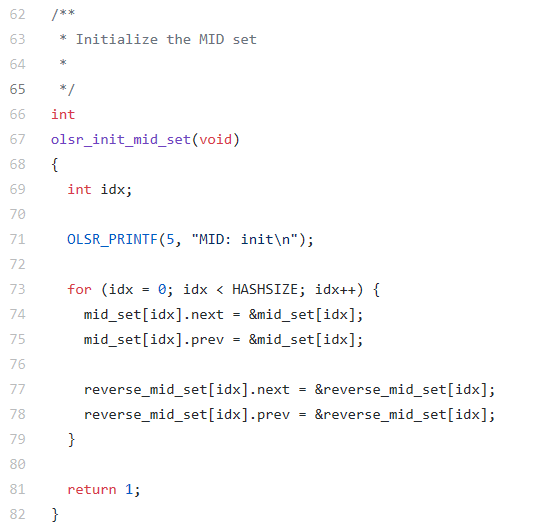
mid\_ttl：time to live，初始时应该被设为255（最大值），以扩散消息进入整个网络并且被设置为相应的值；

mid\_seqno：消息序列号；

addr：接口的主地址；

mid\_addr：接口可能出现的别名。

mid\_set.h



mid\_set.h

66-82：MID消息的初始化代码，mid\_set.h文件中的olsr\_init\_mid\_set函数完成MID消息的初始化。

77-78：初始化MID消息的通信对象。reverse\_mid\_set[idx].next标记下一跳的节点，即该节点将该MID消息发给谁；reverse\_mid\_set[idx].prev标记上一跳节点，即该节点从谁接收到MID消息，这两个数组标记了所有和该节点距离不超过一跳的节点。

### 3.5 HNA消息

为了提供注入外部路由的这种能力，信息进入OLSR MANET，具有这种非MANET接口的节点，定期发布HNA（Host and Network Association）消息，其中包含足够的信息，以便收件人能够构造一个合适的路由表。

HNA将OLSR的移动自组网接口与没有接口的OLSR相连。

HNA消息结构如下图所示

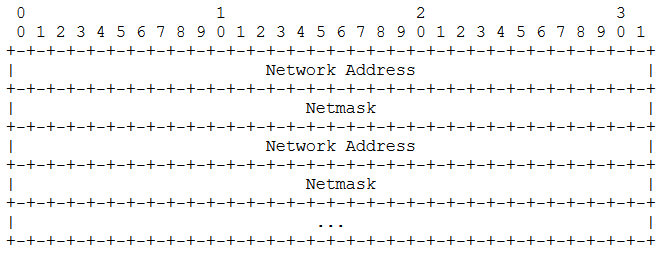


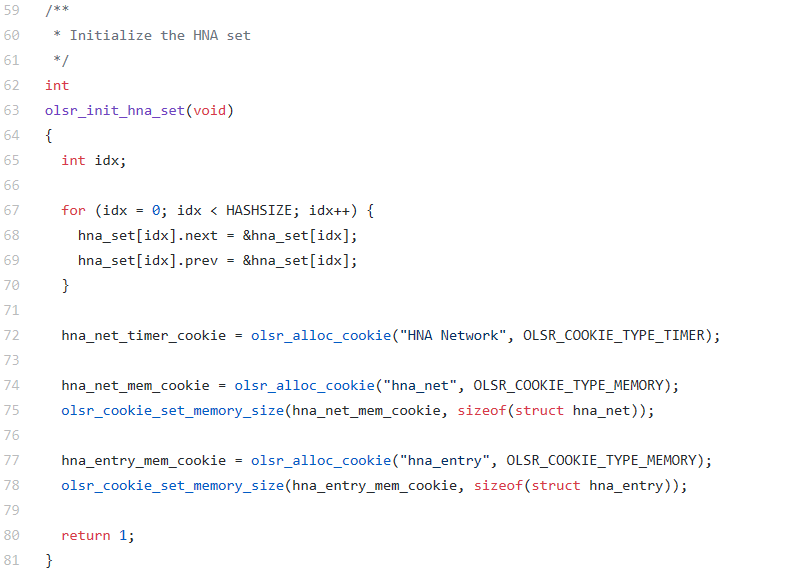
图3.4 HNA消息结构

Network Address：连接网络的网络地址

Netmask：对于上述网络地址的网络掩码

上述结构作为正常封包的数据部分，包的数据类型为“HNA\_MASSAGE”，ttl（time to live）数值设置为255。

hna\_set.h



hna\_set.h

62-80：HNA消息的初始化。mid\_set.h文件中的olsr\_init\_hna\_set函数完成MID消息的初始化。

72-73：利用olsr\_alloc\_cookie函数建立新的cookie，hna\_net\_timer\_cookie保存数据包头部的Vtime值

74-78：olsr\_cookie\_set\_memory\_size函数记录发送消息的表，更具规则接收消息。

### 3.6 信息表

#### 3.6.1 邻居表

一跳邻居表结构如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| neighbor\_main\_addr | status | willingness | is\_mpr | was\_mpr |
| skip | neighbor\_2\_nocov | linkcount |  |  |

neighbor\_main\_addr：本节点的一跳邻居节点的IPV6地址和IPV4地址

status：本节点与其一跳邻居节点的链路连接状态。链路状态连接分为NOT\_SYM（非对称的）和SYM（对称的）两种。

willingness：表示该邻居节点愿意为其他节点转发分组的意愿程度。按意愿程度从低到高分为五个级别，分别为WILL\_NEVER、WILL\_LOW、WILL\_DEFAULT、WILL\_HIGH、WILL\_ALWAYS。

is\_mpr：现在是否为多点中继节点。

was\_mpr：曾经是否为MPR节点，用此发现MPR集的变化。

skip：此邻居节点是否被选入MPR集。

neighbor\_2\_nocov：此邻居节点所覆盖的两跳邻居节点的数量。

linkcount：该邻居节点连接链路的数量。

二跳邻居表结构如下：

|  |  |  |  |
| --- | --- | --- | --- |
| neighbor\_2\_addr | mpr\_covered\_count | processed | neighbor\_2\_pointer |
| neighbor\_2\_nblist | prev | next |  |

neighbor\_2\_addr：两跳邻居的ip地址。

mpr\_covered\_count：经过为mpr节点的一跳邻居的个数。

processed：是否被处理过，作为mpr的一跳邻居节点已经到达过。

neighbor\_2\_pointer：两跳节点个数。

neighbor\_2\_nblist：两跳节点队列。

prev：两跳节点上一跳队列。

next：两跳节点下一跳队列。

#### 3.6.2 MPR Selector表

节点并不发布与所有邻居节点相连的链路信息，而只发布与其部分邻居的链路子集，即判断自己是否被此邻居选做MPR节点，这些被选择的邻居节点是此节点的多点中继选择节点（MPR Selector）。MPR Selector表结构如下：

|  |  |
| --- | --- |
| MS\_main\_addr | MS\_timer |

MS\_main\_addr：MS节点IPV6地址和IPV4地址；

MS\_timer：MPR Selector集的保持时间，如果MPR selector集失效，则删除此集合。

#### 3.6.3 拓扑表

拓扑表结构如下：

|  |  |  |  |
| --- | --- | --- | --- |
| vertex\_node | addr | cand\_tree\_node | path\_cost |
| path\_list\_node | edge\_tree | prefix\_tree | next\_hop |
| edge\_gc\_timer | validity\_timer | refcount | msg\_seq |
| msg\_hops | hops | ansn | ignored |
| err\_seq | err\_seq\_valid |  |  |

vertex\_node：作为二叉树的顶点。

addr：ip地址作为节点的key值。

cand\_tree\_node：候选堆节点，key值为路径代价。

path\_cost：路径的代价。有三种特殊情况：LINK\_COST\_BROKEN (1<<22)；ROUTE\_COST\_BROKEN (0xffffffff)；ZERO\_ROUTE\_COST（0）。

path\_list\_node：根据最短路径优先算法计算出的路径列表。

edge\_tree：边建子树。

prefix\_tree：前缀节点建子树。

next\_hop：下一跳，由最短路径算法计算出的到达目的节点的下一跳。

edge\_gc\_timer：用来维护路由信息。Garbage-Collect定时器超时，该路由仍没有得到更新，则该路由将被彻底删除。

validity\_timer：TC信息的有效时间。

refcount：引用计数器。

msg\_seq：TC（拓扑控制）信息的序列号。

msg\_hops：根据拓扑控制信息计算出的跳数。

hops：根据最短路径算法计算出的跳数。

ansn：通知给邻居的序列号。

ignored：在一个序列中忽略的TC消息总数。

err\_seq：不合格TC消息的序列号（更新）。

err\_seq\_valid：有没有收到错误消息（不合格序列号）。

#### 3.6.4 路由表

路由表结构如下：

|  |  |  |  |
| --- | --- | --- | --- |
| rt\_dst | rt\_best | rt\_nexthop | rt\_metric |

rt\_dst：目的节点的ip地址。

rt\_best：最优路径。

rt\_nexthop：网关路由器的ip地址和输出接口。

rt\_metric：用于路径选择，包括连接代价和跳数。

## 4 路由协议介绍

### 4.1 协议实现

消息发送函数：

olsr\_input\_hello，发送hello消息。

olsr\_input\_tc，发送TC消息。

olsr\_input\_mid，发送MID消息。

olsr\_input\_hna，发送HNA消息。

接收到HELLO包后，使用到的函数如下：

更新邻居表：

olsr\_init\_neighbor\_table，初始化邻居表。

olsr\_insert\_neighbor\_table，邻居表中插入数据。

olsr\_delete\_neighbor\_table，从邻居表中删除数据。

更新两跳邻居表：

olsr\_init\_two\_hop\_table，初始化两跳邻居表。

olsr\_insert\_two\_hop\_table，向两跳邻居表中插入数据。

olsr\_delete\_two\_hop\_table，从两跳邻居表删除数据。

更新MPR表

olsr\_check\_mpr\_changes，检查MPR表是否改变。

olsr\_chose\_mpr，选择MPR。

olsr\_optimize\_mpr\_set，优化MPR选择。

接收到TC包后将更新路由表，使用到的函数如下：

olsr\_insert\_routing\_table，向路由表中插入数据。

olsr\_lookup\_routing\_table，路由表中查找最佳路径。

olsr\_delete\_routing\_table，路由表中删除数据。

接收到MID消息后，将更新mid\_set，使用到的函数有：

olsr\_init\_mid\_set，初始化mid\_set。

olsr\_update\_mid\_table，更新mid\_table。

接收到HNA消息后，将更新hna\_set，使用到的函数有：

olsr\_init\_hna\_se，初始化hna\_set。

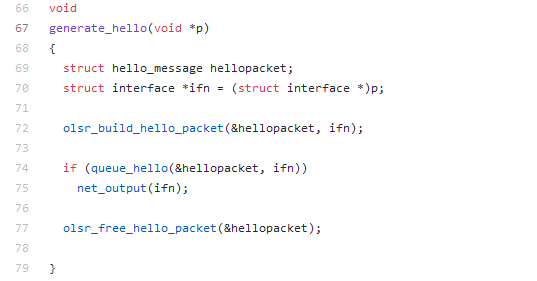
olsr\_update\_hna\_entry，初始化hna\_entry。

邻居表更新后，邻居信息变化，需要更新MPR表和路由表，拓扑表改变后，路由表也需要更新。

### 4.2 邻居发现处理

#### 4.2.1 HELLO消息生成

generate\_msg.c



generate\_msg.c

69-77：generate\_hello函数生成一个消息，经过queue\_hello (build\_msg.c)处理后生成带有参数hellopacket内容的HELLO包，创建成功，则可以net\_output函数将包发送出去。

#### 4.2.2 邻居表操作

节点接受到HELLO包后，将会对自己的邻居表进行操作，涉及到以下几个函数。

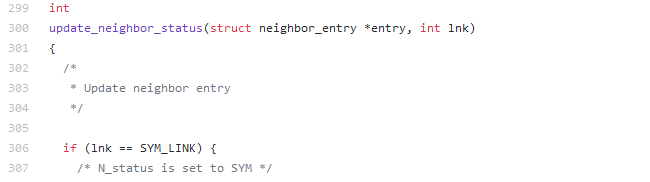
olsr\_init\_neighbor\_table(neighbor\_table.c)，将每个邻居表初始化为仅有一个指向自身的节点的链表。

olsr\_del\_nbr\_list(neighbor\_table.c)，删除释放一个两跳邻居节点，并将全局变量change\_neightborhood,change\_topology置为真，通知网络重新计算MPR集和路由表。

olsr\_delete\_neighbor\_2\_pointer(neighbor\_table.c)，根据给定的邻居节点地址删除对应的两跳邻居节点。

#### 4.2.3 邻居节点操作

neighbor\_table.c



neighbor\_table.c



neighbor\_table.c

函数功能：更新邻居表中的状态为link。

306-321：将连接状态置为SYM\_LINK，如果之前的状态为NOT\_SYM，通知重新进行MPR选举和路由表更新并删除通过这个节点连接到的两跳邻居节点。

323-331：如果之前状态就是NOT\_SYM，仅统治全网进行MPR选举和路由表更新，不删除两跳邻居节点。

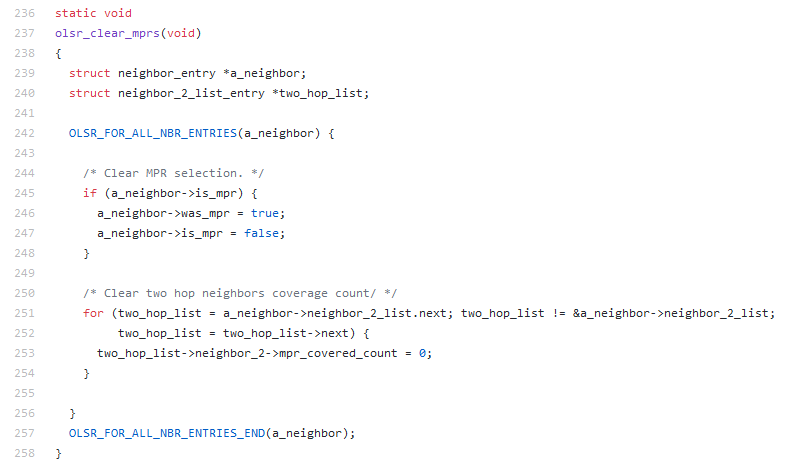
### 4.3 MPR

#### 4.3.1 MPR节点操作

MPR节点操作为节点添加到MPR集中，和从MPR集中删除记录，具体涉及到以下两个函数：

add\_will\_always\_nodes (Mpr.c)：添加willlingness为WILL\_ALWAYS的邻居节点到MPR集中。

Mpr.c



Mpr.c

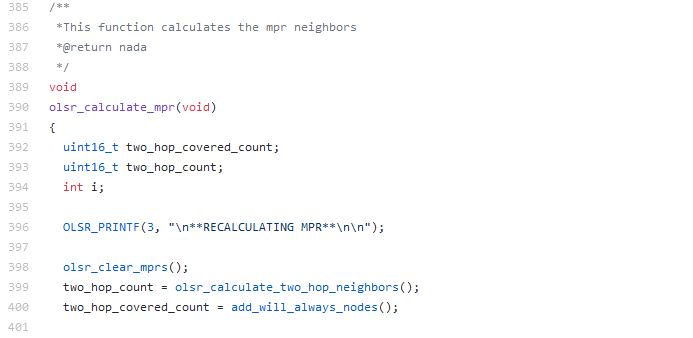
函数功能：将被选为MPR节点的记录清除。

245-247：如果is\_mpr为真，则将is\_mpr置为假，并将was\_mpr置为真，表明曾经是MPR。

251-253：遍历邻居节点覆盖到的两跳邻居节点，并将数量置为0。

#### 4.3.2 MPR算法

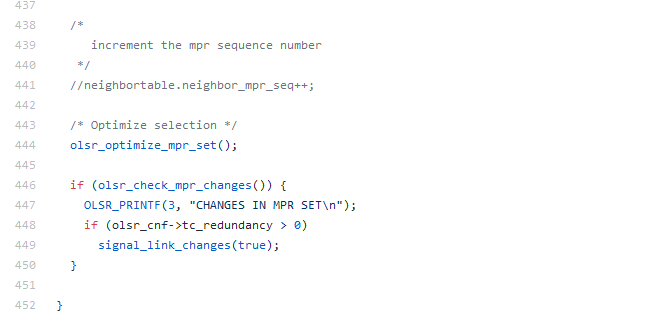
Mpr.c



Mpr.c

Mpr.c





Mpr.c

399-400：计算两跳邻居个数和所有willingness为always的节点个数。

406-418：查找存在一条连接的两跳邻居节点链表，判断节点是否每个邻居都可达，如果可达，对MPR计时器进行更新。

426-428：找到覆盖最多两跳节点的MPR。

444-449：删除选择的MPR节点所覆盖的两跳节点，进行优化。检查所有MPR节点，判断是否发生变化，如果发生变化，更新连接。算法

MPR算法是典型的贪婪选择，算法的具体过程是：  
 1）将所有willingness为WILL\_ALWAYS的成员加入MPR集中，

2）计算所有节点的深度，

3）将所有存在唯一连接的两跳邻居节点的节点加入MPR。

4）若两跳邻居节点集中存在不被MPR集中任何节点覆盖的节点，执行以下过程，若不存在，结束算法。

a）对一条邻居节点中所有节点计算可达性，

b) 在一条邻居节点中达到新非0，且具有最高willingness的节点中选择一个MPR，有多种选择时，选择可达性最高的。还有最高选择的时候，选择深度最高的。移除二跳邻居节点中被MPR覆盖最多的节点，返回第三步。

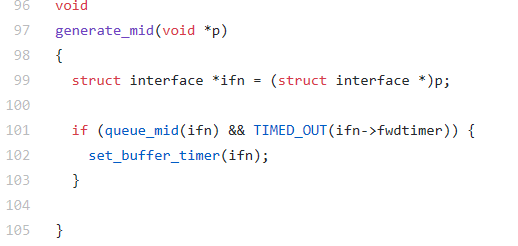
5）将节点没个接口的MPR集；；组合在一起，建立起节点的MPR集。

### 4.4 MID消息

#### 4.4.1 消息生成

当某个物理节点包含多个参与OLSR的接口时，他必须周期性地发送MID（Mutliple Interface Declaration）消息，节点接收到MID消息之后会将MID消息中包含的多个接口的地址和主地址填充进拓扑集。节点通过MID消息获得多端口信息。

generate\_msg.c



generate\_msg.c

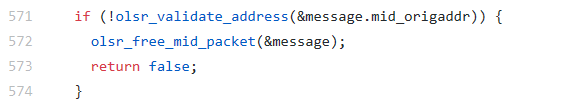
96-105：MID消息的生成。

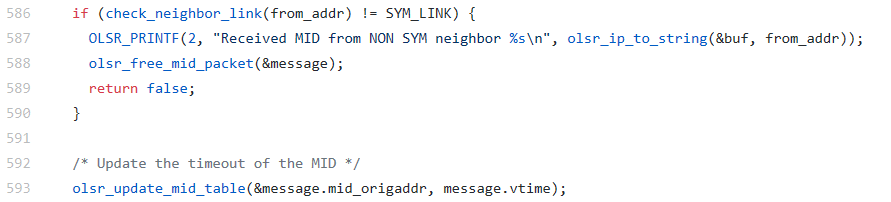
101-103：生成MID消息的节点先将该MID消息放入队列中，并且为消息打上时间戳，当读取到该消息时，先通过时间戳确定消息是否过期，如果消息未过期，那么读取消息内容，并将其放入缓存区内，消息等待发送。

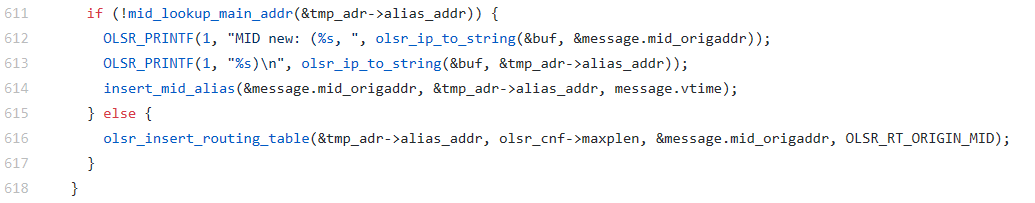
#### 4.4.2 消息处理

当一个节点接收到MID消息时，会根据MID消息更新自己保存的多接口拓扑集。

mid\_set.c







mid\_set.c

571-618：MID消息的处理（接收和转发）函数，由mid\_set.c文件中的olsr\_input\_mid函数实现。

571-574：判断MID消息的发送方是否为有效的地址，如果不是，则释放消息，并且不再转发；

586：检查消息的发送方。

587-593：如果这条消息的发送者对于接受消息的节点来说不是对称的一跳邻居节点，那么这条MID消息将会被丢弃，不被继续转发。根据MID消息的头来更新MID消息的最初发送方和时间。

611-617：对于别名，如果找到了对应的main address那么在节点中相应地添加别名。

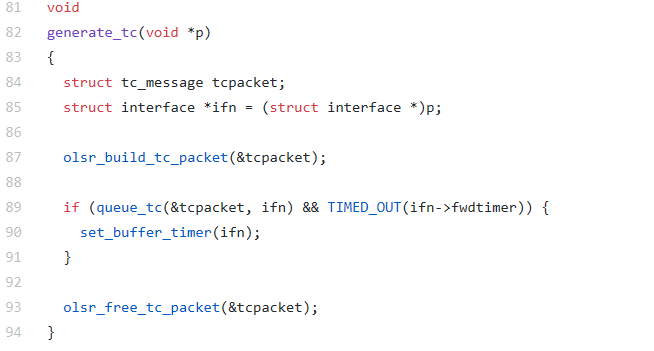
### 4.5 拓扑建立

为了构建全网络的拓扑信息图，网络中被选为MPR节点的节点每隔一段时间就会向网络中广播拓扑控制（TC）消息，来维护网络中的拓扑信息。为了减少网络中的洪泛，olsr采用了如下规则，对于相同的TC消息，接收节点只在从发送方为自己的MPR节点且第一次收到的情况下转发，在之后收到不是相同的TC消息时不予转发。这则规定有效地避免了广播风暴。

节点通过TC消息在网络中的扩散获得关于全网的拓扑图，再根据自己的一跳邻居表、两跳邻居表和拓扑表，计算出路由表。

#### 4.5.1 TC消息生成

generate\_msg.c



generate\_msg.c

81-94：TC消息生成算法。generate\_mag.c文件中generate\_tc函数实现。

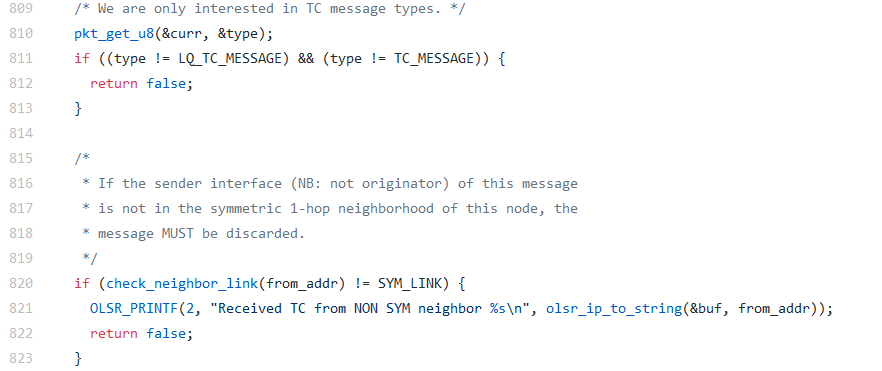
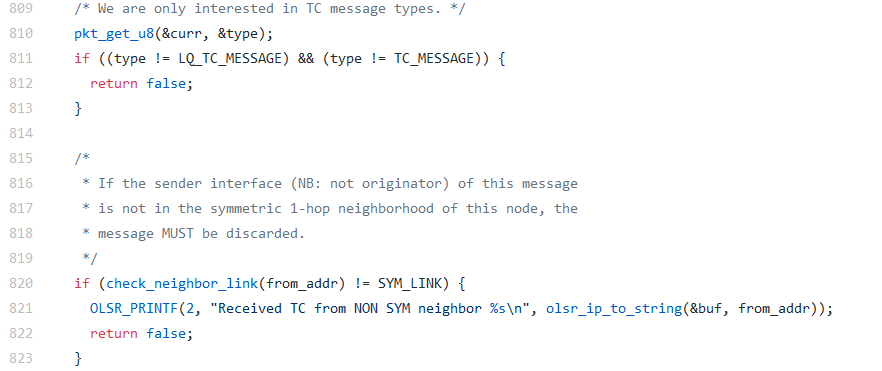
87：olsr\_build\_tc\_packet函数实现构建本节点的TC包。

89-91：生成TC消息的节点先将该TC消息放入队列中，并且为消息打上时间戳，当读取到队列中的该消息时，先通过时间戳确定消息是否过期，如果消息未过期，那么读取消息内容，并将其放入缓存区内，消息等待发送。

#### 4.5.2 TC 消息处理

只有当消息的产生者是自己的MPR Selector节点，并且TC消息的存活时间大于0时，接受节点才转发该TC消息。

tc\_set.c

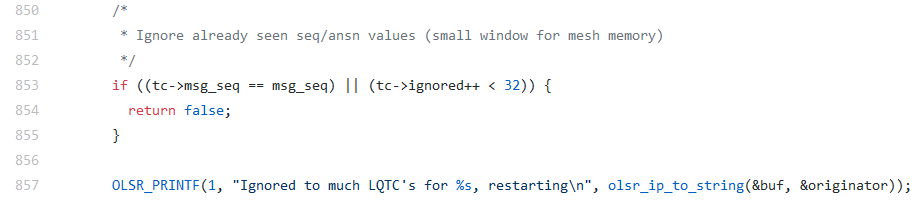


tc\_set.c

810-813：当节点接收到TC消息时，关注TC消息的类型，如果该TC消息的类型是不是LQ\_TC\_MESSAGE并且不是TC\_MESSAGE，那么返回FALSE。

820-823：之后再检查TC消息的发送者接口是不是这个节点的对称的一跳邻居，如果不是，则将此条TC消息丢弃。

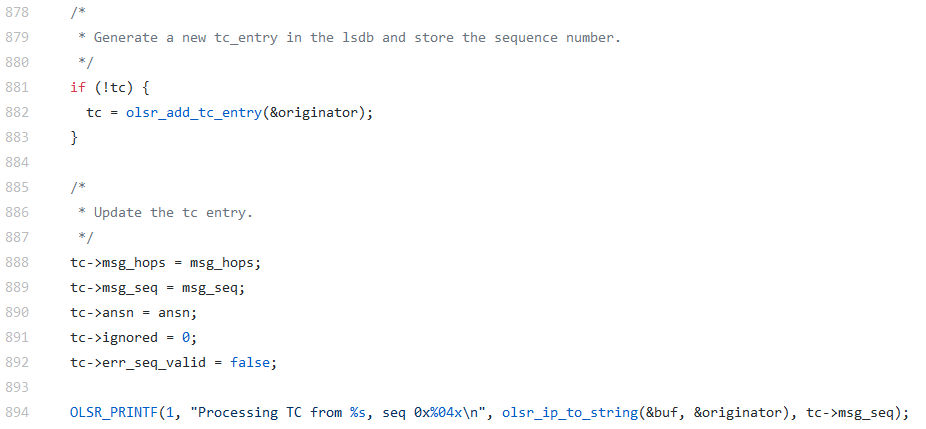
tc\_set.c



tc\_set.c

853-857：如果该TC消息的序列号已经存在或者ignored的个数不超过32，那么表示该消息已经被处理过，这条TC消息被忽视。

tc\_set.c

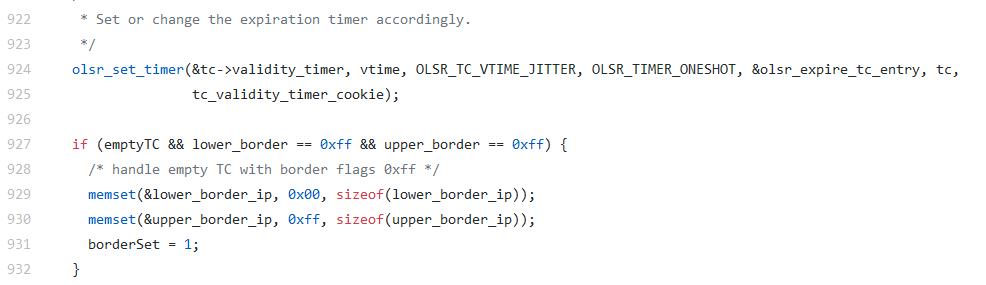


tc\_set.c

878-883：如果TC消息的初始产生地不一样，那么添加新的条目到此节点的拓扑信息中并且保存序列号。

888-892：根据接收到的TC消息（允许添加新的TC\_ENTRY）的首部更新tc entry的信息。

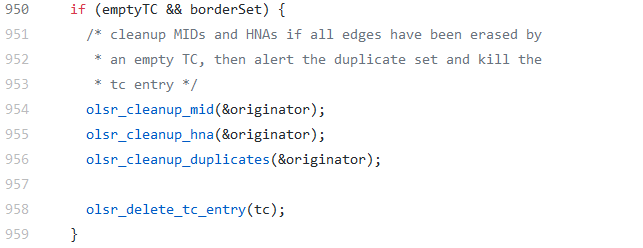
tc\_set.c



tc\_set.c

924-932：设置或者改变到期的计时器。

tc\_set.c



tc\_set.c

950-958：清空MID和HNA消息，如果所有的边都被一个空白的TC消息清除。

#### 4.5.3 拓扑表处理

tc\_set.c



tc\_set.c

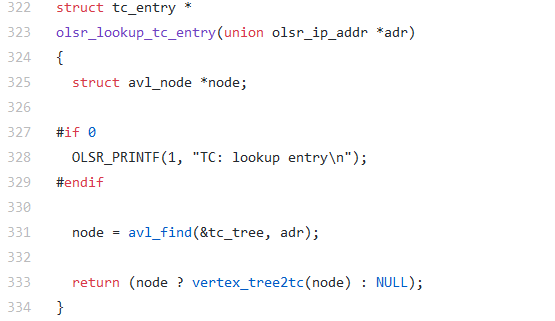
185-208：拓扑集的初始化。

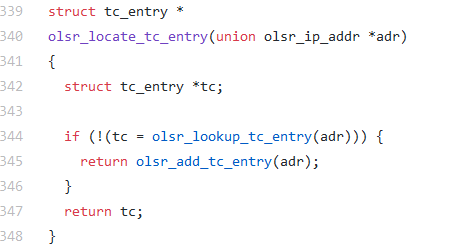
198：avl\_init函数初始化AVL树

195-202：olsr\_alloc\_cookie函数从cookie中获取相关数据对拓扑集进行初始化。

207：olsr\_add\_tc\_entry：在TC树种添加一个entry实例。

tc\_set.c





tc\_set.c

322-334：从拓扑集中根据地址找到一条符合的entry。

339-348：如果找不到符合的entry，创造一个符合的entry。

tc\_set.c



tc\_set.c

278-304：tc\_set.c文件中的olsr\_delete\_tc\_entry函数实现删除一条TC实例。

289-291：如果宏定义了LINUX\_NETWORK\_ROUTING，那么删除和该TC entry相关的网关信息。

295：olsr\_delete\_routing\_table：删除路由表中和被删除实例的路径信息。

298-300：边的信息和前缀节点信息之前必须为空。

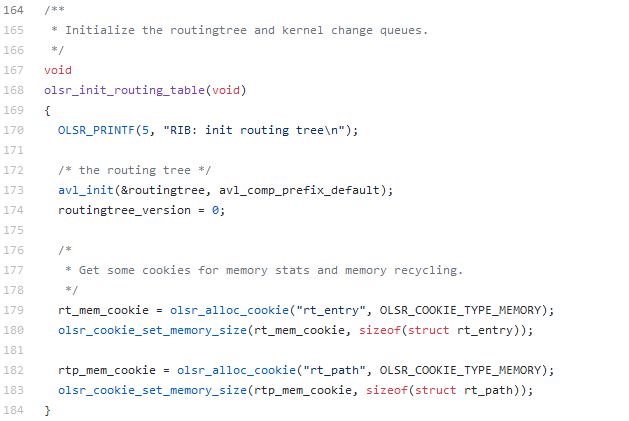
306-310：停止相关计时器。

312：从生成的avl树中删除相关节点。

### 4.6 路由算法

节点通过TC消息得到全网的拓扑信息，然后根据自己的邻居表和拓扑表，按照Dijkstra算法得到路由表。节点通过路由表得到到目的节点的路径信息。

Routing\_table.c



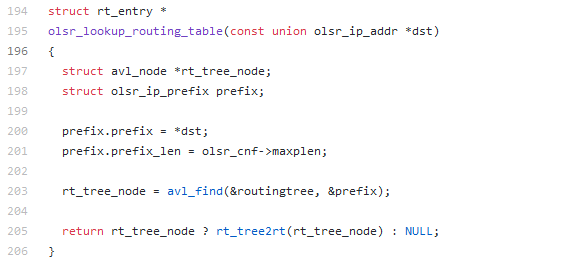
Routing\_table.c

164-184：初始化路由表（路由树和内核改变队列）。

173-174：初始化路由avl树，routingtree\_version版本号为0

179-183：建立相应的cookie保存路由实例和路径。

Routing\_table.c

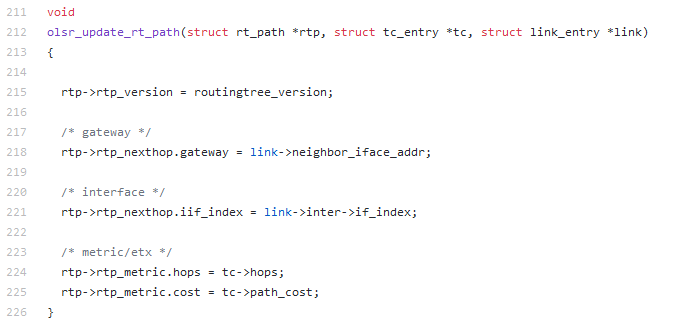


Routing\_table.c

194-206：在avl树中查找得到路由表条目。

205：如果节点不为空，则利用rt\_tree\_node函数返回re\_entry类型，如果节点为空返回NULL。

Routing\_table.c



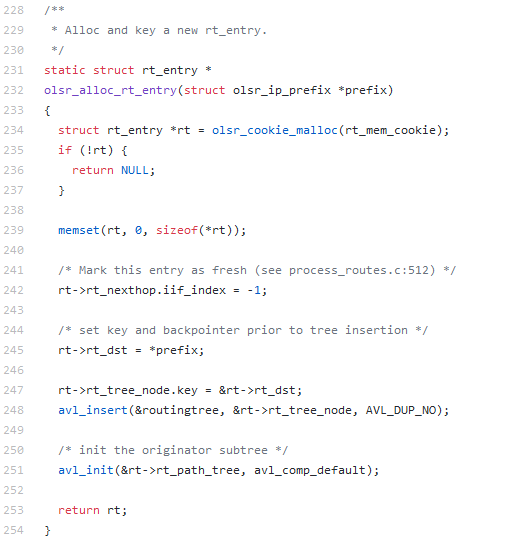
Routing\_table.c

211-226：更新路由表路径

215：修改路由表的版本

218-225：根据传入函数的数值更新路径的网关，接口，跳数和路径花销。

Routing\_table.c



Routing\_table.c

228-254：创建一个新的路由条目。

234-239：申请内存空间并将此空间清空。

243-248：把入口的树节点插入到路由表，并初始化树。

Routing\_table.c



Routing\_table.c

259-285：创建一个新的路由路径。

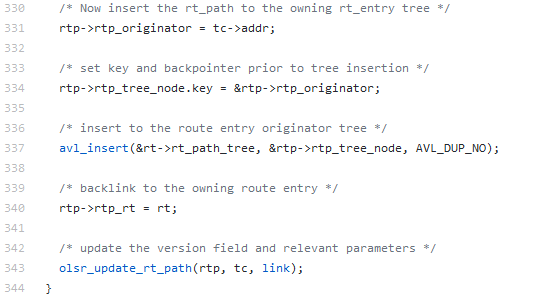
262-268：为此路由路径申请内存空间，并将内存空间清空。

275：将此路由路径信息中前缀节点插入到相应的前缀树中。

279-283：根据传入数据保存该路径信息对应的TC条目，并保存路径信息的起点。

Routing\_table.c





Routing\_table.c

292-344：对于给定的路由路径信息创建一条路由实例并且把它插入RIB树中。

298-303：判断tc\_entry是否可达，如果不可达，那么也没有必要创建路由实例。

312-328：用avl\_find函数判断对于前缀节点是否有一条路由路径可到达前缀节点，如果不存在，那么、olsr\_alloc\_rt\_entry函数创建一条新的路由实例。

330-343：根据函数传入数据插入路由实例并更新相关信息。

Routing\_table.c



Routing\_table.c

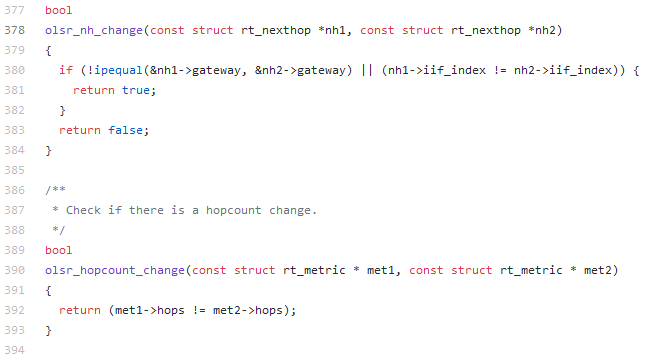
349-372：删除一条路由路径。

353-357：把该路由路径所指向的树节点从TC树中删除，并将相应树节点置空。

359-364：该路径信息的前缀节点从前缀树中删除，并解锁相关的tc\_entry，并将tc\_entry置空。

366-371：如果路径信息被移除。那么网关信息也要置空，并释放cookie占用内存。

Routing\_table.c

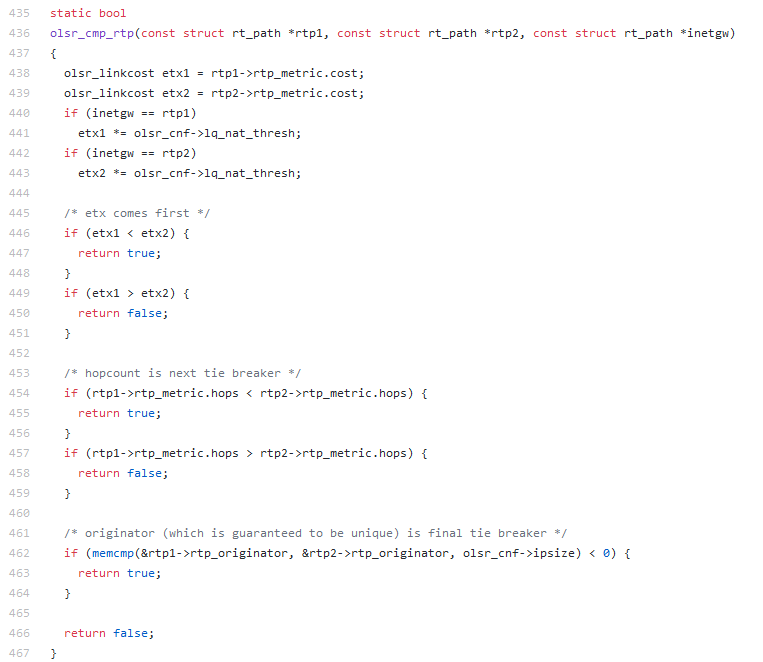


Routing\_table.c

377-384：接口和网关是否更改。

380-383：如果接口发生了更改或是网关信息发生了更改（ipequal），那么返回真。否则返回假。

Routing\_table.c



Routing\_table.c

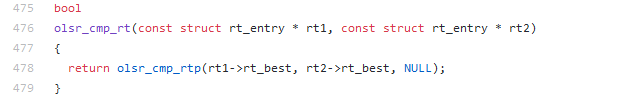
435-467：比较两个路由路径。如果传入的第一条路径较好返回真，第二条路径较好返回假。

438-451：首先比较两跳路径的连接花销，花销少的路径更好。

453-459：如果花销相同，那么比较跳数，跳数少的路径更好。

461-464：如果花销和跳数都相同，那么比较路径源地址的ip大小,地址小的路径更好。

Routing\_table.c



Routing\_table.c

475-479：比较对于两条不同的路由实例找到更好的路径。前者更优返回真，后者更优返回假。

478：调用olsr\_cmp\_rtp比较两条路由实例最优的路由路径找到哪条更优。

Routing\_table.c



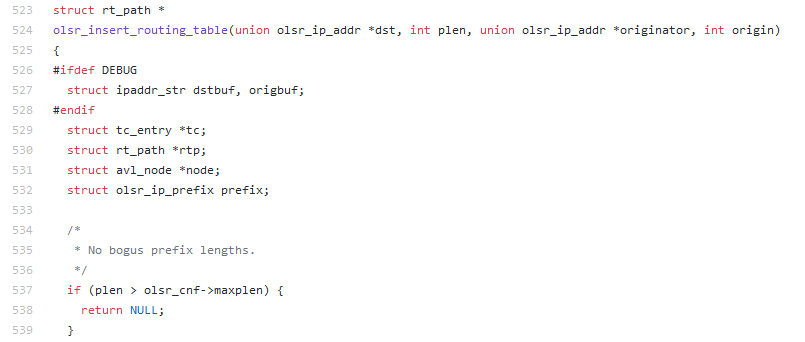
Routing\_table.c

485-507：找到最优的路径。

488-493：利用avl\_walk\_first函数在路径树中得到第一个条目，并把节点类型转变为rt\_entry类型。

495-506：利用avl\_walk\_next函数实现遍历，比较得出最优路径，并且根据求得的最优路径耿勋路径。

Routing\_table.c





Routing\_table.c

526-577：向路由表中插入路由路径信息。

523-539：前缀长度过长，则返回NULL。

548-554：利用avl\_find函数查看对于前缀节点是否已经存在一个路由路径。如果没有节点存在，那么利用olsr\_alloc\_rt\_path函数创建一个新的路由路径。

Routing\_table.c



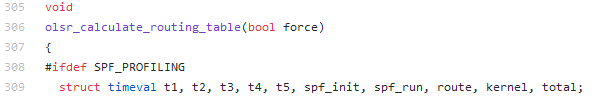
Routing\_table.c

582-621：删除路由表中的相关信息。

582-604：查看前缀节点的长度和在查找相关源点的tc entry是否存在，如果长度过长或者在路由表中不存在相关的tc entry，那么路由表中相关信息不存在，即不用删除，直接返回。

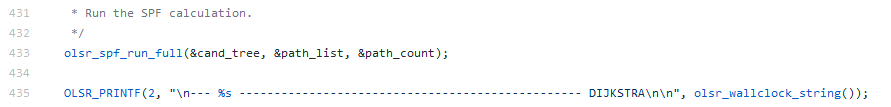
609-616：删除相关信息，利用olsr\_delete\_rt\_path函数从路由表中删除相关路由路径信息。

olsr\_spf.c









olsr\_spf.c

305-435：计算路由表

321-329：检测退避计时器是否正在运行，如果退避计时器没有在运行，那么用olsr\_start\_timer函数开启新的退避计时器。

338-340：准备候选树和结果列表。

356-364：利用olsr\_change\_myself\_tc函数检测主IP地址是否发生变化，如果发生变化，那么利用olsr\_update\_rib\_routes函数和olsr\_update\_kernel\_routes函数更新RIB表和核心路由表。

431-435：olsr\_spf\_run\_full函数根据最短路径优先算法计算出最优路径。

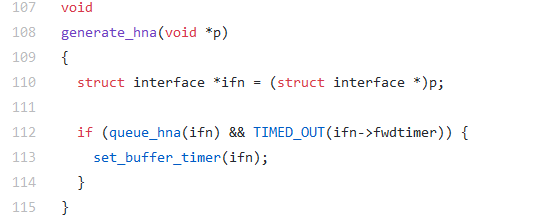
### 4.7 HNA消息

为了提供注入外部路由的这种能力，信息进入OLSR MANET，具有这种非MANET接口的节点，定期发布HNA（Host and Network Association）消息，其中包含足够的信息，以便收件人能够构造一个合适的路由表。

HNA将OLSR的移动自组网接口与没有接口的OLSR相连。

#### 4.7.1 消息生成

generate\_msg.c



generate\_msg.c

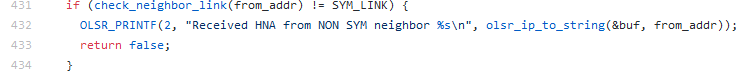
107-115：HNA消息生成算法。generate\_mag.c文件中generate\_hna函数实现。

110-114：生成HNA消息的节点先将该HNA消息放入队列中，并且为消息打上时间戳，当读取到队列中的该消息时，先通过时间戳确定消息是否过期，如果消息未过期，那么读取消息内容，并将其放入缓存区内，消息等待发送。

#### 4.7.2 消息处理

hna\_set.c





hna\_set.c

372-379:HNA消息发送的包为OLSR标准包，所以HNA的发送方式为洪泛发包。

372-379：HNA消息在发送时记录了消息的olsr消息类型，vtime,起始发送者，跳数和序列号

387-397：检查是否一切都准备好，消息包是否符合olsr消息包的规范，如果不符合，则不发送包。

431-434：判断数据包是否为对称邻居节点发送，如果不是则丢弃。

## 

## 5 总结

OLSR路由协议是IETF MANET工作组为无线移动Ad Hoc（无线自组）网提出的作为RFC标准化的一种表驱动先应式路由协议。该协议在传统的链路状态算法基础上采用MPR(multipointreplay，多点中继)机制来减少协议开销。

OLSR协议使用同样的数据包头，保证消息的兼容性，一个分组中可以包含多个不同的消息。当节点接收到一个数据包时，先检查数据包的头查看节点是否已经处理过该数据报。

OLSR协议中每个节点通过周期性地交换链路状态信息来维护整个网络的拓扑信息。节点周期性地向邻居节点广播HELLO消息以得到邻居及链路状态，节点在接收到HELLO分组后，启动HELLO消息处理过程，判断邻居节点的链路状态并维护自己的一跳邻居表和二跳邻居表，HELLO分组只在产生HELLO分组的节点的一跳范围内传播。

Olsr协议由HELLO控制分组和TC控制分组两个控制分组，HELLO分组的作用是链路检测和发现邻居，TC控制分组的作用是声明MPR信息。HELLO消息用于建立一个节点的邻居表。TC消息声明MPR信息，包含拓扑结构信息，通过该信息计算路由。

当某个物理节点包含多个参与OLSR的接口时，他必须周期性地发送MID（Mutliple Interface Declaration，多接口声明）消息。为了提供注入外部路由的这种能力，信息进入OLSR MANET，具有这种非MANET接口的节点，定期发布HNA（Host and Network Association）消息，其中包含足够的信息，以便收件人能够构造一个合适的路由表。

每个节点为了避免消息在网络中的洪泛，只选用一跳邻居中的部分节点作为MPR节点，这些节点形成多点中继集。只有MPR节点才会向网络中发送TC控制消息并且参与路由计算，而其他不被任何节点选做MPR节点的节点则不参与路由计算也不发送和转发TC消息。MPR机制削减了整个网络中洪泛的TC消息的数量和转发次数，大大降低了协议的开销。

OLSR路由协议的优点有以下几个方面。OLSR是一种先应式路由协议，查找路由延迟小；OLSR采用MPR机制，只有MPR节点才会向网络中发送TC控制消息并且参与路由计算，而其他不被任何节点选做MPR节点的节点则不参与路由计算也不发送和转发TC消息，减少了整个网络中的洪泛，减少了协议开销。OLSR采用Dijkstra最短路径优先算法选择路径。