1概述

最佳链路状态路由协议（Optimized Link State Routing Protocol，OLSR），是专门为无线移动Ad Hoc网络提出来的一种标准化的先验式的优化链路状态路由协议。

1.1设计原理

OLSR由传统LSR路由协议改进而来。传统链路状态协议每个节点通过周期性的交换链路状态信息维护整个网络的拓扑信息。在OLSR协议中，网络中的每个节点只选择对称邻居节点的一个子集作为多点中继集MPR（Multipoint Relay），只有被选为MPR的节点才产生并转发TC（Topology Control）分组，同时OLSR只利用MPR节点到MS（MPR Selector）节点之间的链路状态信息来建立最短路由，这样很大程度上减少了转发的信息，减少了网络中洪泛的控制信息。

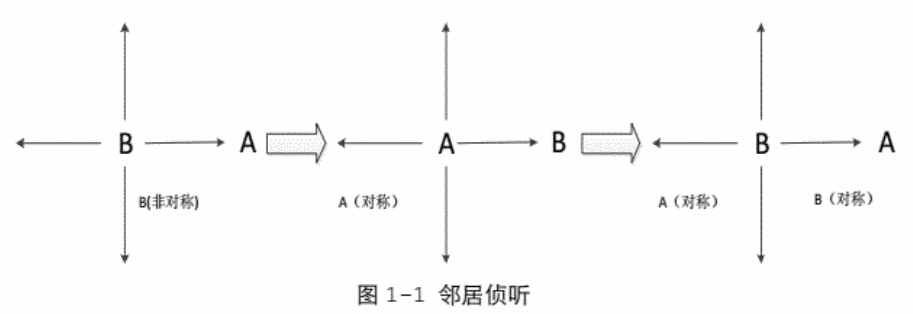
1.2协议的主要流程（未完待续）

1.2.1链路感知

因为无线电传播的不确定性可能会导致一些链路是单向的，所以每个节点必须检测自己和邻居节点之间的链接是否是双向的，也就是在两个方向上都能传输数据。本地链路信息表存储了到邻居的链接的信息，完善这一集合的过程称为“链路感知”。链路感知通过HELLO分组的周期性交互实现，当一个节点收到HELLO分组时，更新自身的本地链路信息表。

1.2.2邻居侦听

每个节点必须检测与哪些邻居节点具有双向链路，节点周期性地广播HELLO分组。HELLO分组用来侦听邻居节点的状态，分布在一跳范围内转播，不能被转发。如图1-1所示，在初始化阶段，节点B广播HELLO分组，当节点A收到这个分组之后，将B放入到自己的邻居节点集中，并标记A到B链路状态为非对称的；然后A广播HELLO分组，其中包含B是A的邻居节点且B到A的链路状态为非对称这一信息，当B收到这一分组时，会在自己的邻居集中将到A的链路状态更新为对称；当B再次广播HELLO分组，A收到这一分组就会将邻居集中的A到B的链路状态更新为对称的。



1.2.3MPR选择

OLSR采用MPR机制对路由信息进行选择性的洪泛。网络中的每个节点选择自己的一跳对称邻居节点的子集作为中继节点，即MPR，而该节点本身作为MS节点。

计算MPR集需要知道自身一跳和两跳邻居的信息。一跳邻居信息通过HELLO分组来获取。对于节点i的一跳邻居节点j来说，i的一跳邻居节点（除j）就是j的二跳邻居节点，所以节点i在发送HELLO分组时附上自己的一跳邻居节点列表，当节点j收到来自i的分组时，就获得了节点j的两跳邻居节点信息。MPR集应满足以下两点要求：节点与MPR之间必须是一跳双向链路；节点能通过MPR集到达所有的严格两跳邻居节点。

1.2.4TC分组处理

TC分组由MPR广播和转发。MPR节点每隔一段时间就向全网洪泛TC消息来维护网络拓扑信息。

1.2.5路由表的计算与维护

网络中的每一个节点都维护一个路由表，路由表的计算是基于节点维护的本地链路信息表和拓扑集。因此，如果本地链路信息表和拓扑表发生变动时，路由表会重新计算并更新表项。在OLSR标准协议中，协议根据最小跳数建立每个节点的路由表。任意一个节点的路由表添加过程如下：首先，添加对称链路的一跳邻居节点；然后，添加两跳邻居节点；最后，循环添加跳数等于h+1（h=2开始）的节点进入路由表。

**文件介绍**

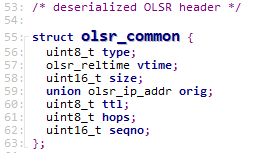
OLSR协议中共有123个源文件，以下对部分重要文件进行罗列。



2数据结构部分

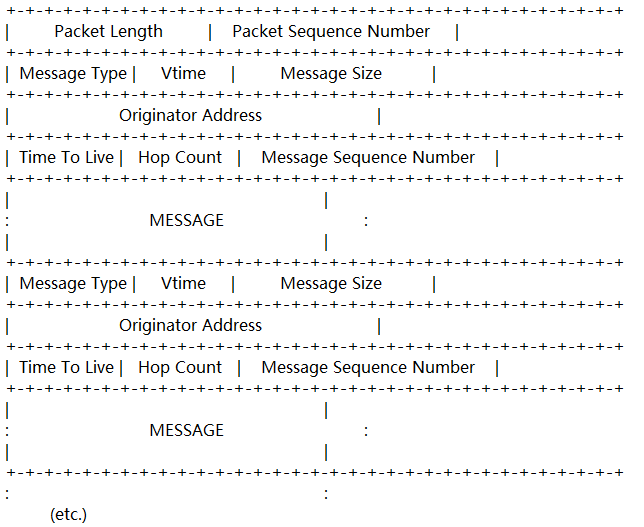
2.1部分消息包的报文格式

2.1.1OLSR首部（省略IP和UDP报头）



53-63：olsr\_common是OLSR协议的基本数据包首部。

对于与协议相关的所有数据，OLSR使用统一的数据包格式进行通信，这样做的目的是在不破坏向后兼容性的情况下促进协议的可扩展性。这也提供了一种简单的方法，将不同“类型”的信息汇集到一个单一的传输中。这些数据包嵌入在UDP数据报中，使用UDP通信，IANA将端口698分配给OLSR协议专用。每个分组封装一个或多个消息，这些消息共享一个通用的报头格式，使节点能够接受和重传未知类型的消息。OLSR协议分组的基本格式如图所示：



a.数据包首部

*Packet Length：*数据包的长度（单位：字节）

*Packet Sequence Number：*数据包序列号，每当传送一个新的OLSR分组时，序列号加一。为每个接口维护一个单独的分组序列号，以便对通过接口发送的分组进行顺序枚举。

b.消息首部

*Message* *Type* ：消息类型，在0-127范围内的消息类型保留给本协议中的所有消息和以后可能的扩展。

*Vtime* ：该字段表明接收节点后多长时间必须将消息中包含的信息视为有效，除非接收到对信息的最新更新。有效时间由它的尾数a（Vtime字段的四个最高位）和它的指数b（Vtime字段的四个最低位）表示。有效期=C\*（1a/16）\*2^b[秒]，C为一系数。

*Message Size*：消息的大小，以字节为单位计算。并从“消息类型”字段的开头到下一个“消息类型”字段的开头（如果没有下一个“消息类型”，则到该消息的结尾为止)。

*Originator Address*：此字段包含最初生成此消息的节点的主地址，重传过程中地址不发生改变。此字段不应与IP标头中的源地址混淆，后者每次都更改为重传该消息的中间接口的地址。

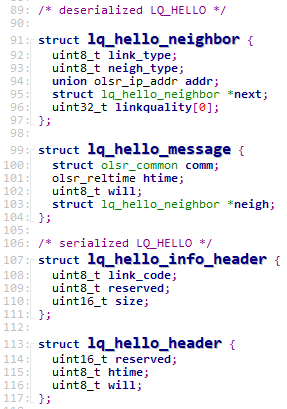
*Time To Live*：此字段包含将发送消息的最大跳数。消息每次重传前TTL减1，当TTL为0或1时，则不再重传。通过此方式可以限制一个消息的洪泛范围。

*Hop Count*：一个消息获得的跳数的数量。每次重传消息前，跳数加一。初始值为0。

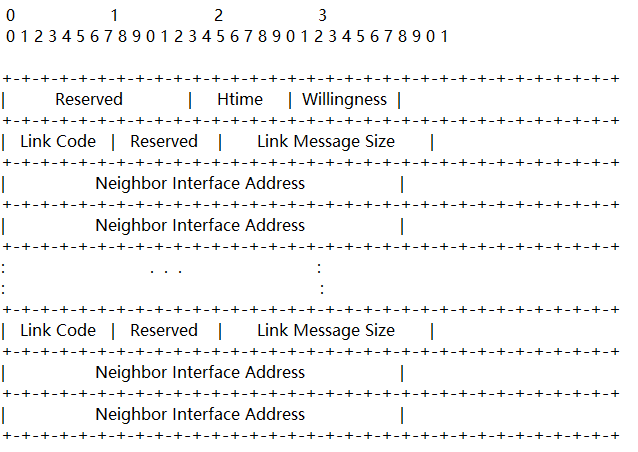
*Message Sequence Number*：由源节点产生的一个消息的唯一标识，用来保证一个消息不会被任何节点重传一次以上。

2.1.2HELLO消息

OLSR协议采用一种通用的机制来填充本地链路信息库和邻居信息库，即周期性地交换HELLO消息。



106-117：结构体lq\_hello\_info\_header和lq\_hello\_header共同组成HELLO消息数据包的首部。格式如下图：



这一部分是将OLSR基本数据包首部的"Message Type"设为HELLO\_MESSAGE，TTL设为1，Vtime设为NEIGHB\_HOLD\_TIME。

Reserved:保留字段,必须设置设为“0000000000000

Htime:指定节点在特定接口上的HELLO发射间隔，即下一个HELLO传输前的时间。Hello发射间隔用尾数(Htime的四个最高位)和指数(Htime的四个最低位)来表示。即HELLO发射间隔=C\*（1+a/16）\*2^b[秒]。其中a是Htime字段的四个最高位表示的整数，b是Htime字段的四个最低位表示的整数。

willingness:此字段指定节点为其他节点承载和转发流量的意愿。

Willingness有三种取值：

1. WILL\_NEVER：永远不会选择意愿为WILL\_NEVER的节点作为MPR。
2. WILL\_ALWAYS：具有WILL\_ALWAYS意愿的节点将始终被选择为MPR。
3. WILL\_DEFAULT：默认情况下，节点意愿为WILL\_DEFAULT。

Link Code:

此字段指定发送方的接口和邻居列表中邻居接口之间的链路类型。它还指定有关邻居状态的信息。链路类型不为节点所知的邻居信息被静默丢弃:

链路类型有以下三种：

(1) ASYM\_LINK:发送HELLO分组的节点与邻居列表中的节点间的链路是非对称的。表示可以收到邻居节点的消息,但不确定邻居节点能否收到本节点的消息。

(2) SYM\_LINK:发送HELLO分组的节点与列表中的邻节点间的链路是对称的。表示链路已经被验证为双向的。

(3)MPR\_LINK:表示列表中的节点已被发送该HELLO分组的节点选择为MPR。

Link Message Size：本链路消息的大小，以字节为单位，从Link Code字段开始到下一个Link Code字段之前（如果没有下一个Link Code，则到该消息尾部）。

Neighbor Interface Address：邻居节点的接口地址。每一种链路类型之后都紧跟一组邻居节点接口地址，表明该节点与这组邻居节点中的每一个节点链路类型都相同，都为前面给出的链路类型。

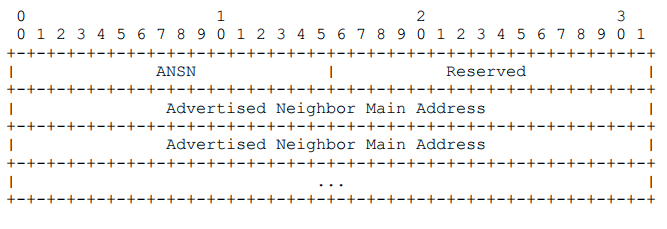


49-57：结构体hello\_neighbor是HELLO消息邻居节点集。Status，记录邻居的状态；link指明链路类型；main\_address是邻居的主地址；address是邻居的其他地址；cost，链路代价；linkquality，链路质量。

59-69：hello\_message是消息数据包。vtime，消息有效时间；htime，HELLO消息的发射间隔；source\_addr，发送消息的原地址；packet\_seq\_number，数据包的序列号；hop\_count，消息已经经历的跳数；ttl，数据包生命周期；willingness，节点进行转发的意愿；neighbors，下一个传递的邻居节点。

2.1.3.TC消息

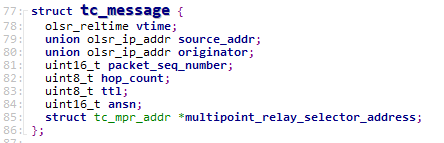
该部分是将OLSR首部中“MessageType”设置为TC\_Message。TTL设置为255(最大值)，以便将消息传播到网络中Vtime相应地设置为Top\_hold\_time的值。



ANSN：序列号与公布的邻居集相关。每当节点检测到其邻居集中的更改时，它都会递增这个序列号。当节点接收到TC消息时，它可以根据该消息来判定所接收到的信息是否比现有消息更新。

Advertised Neighbor Main Address：此字段包含邻居节点（此处的邻居节点仅为该节点的MPR Selector集中的节点）的主地址。

Reserved：此字段是保留的，必须设置为“0000000000000000”。

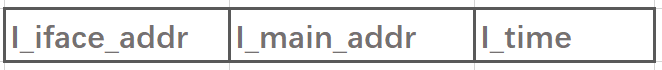


77-86：tc\_message是TC消息数据包格式。OLSR协议利用TC拓扑表记录接收到的TC消息内容。TC拓扑表的介绍见“节点存储的表项”的“TC拓扑表”。

2.2节点存储的表项

2.2.1. 接口关联集合

网络中的每一个目的地节点存储了接口关联多元组。



i\_iface\_addr是节点的接口地址；

i\_main\_addr是该节点的主地址；

i\_time指定此元组过期的时间，以及必须删除的时间。

2.2.2.本地链路信息表



L\_local\_iface\_addr是本地节点(即链路的一个端点)的接口地址；

L\_neighbor\_iface\_addr是相邻节点的接口地址；

L\_SYM\_time是链路被视为对称的时间；

L\_ASYM\_time是被认为听到邻居接口的时间；

L\_Time指定了此记录过期的时间，并且必须被删除。当L\_SYM\_TIME和L\_ASYM\_TIME过期时，该链接被视为丢失。

2.2.3.一跳邻居表

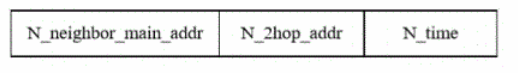


N\_neighbor\_main\_addr：邻居的主地址；

N\_status：指定节点是NOT\_SYM还是SYM；

N\_willingness：指定节点的携带意愿，取值为0到7之间的整数。

2.2.4.二跳邻居表



N\_neighbor\_main\_addr是邻居的主地址；

N\_2hop\_addr是具有与N\_neighbor\_main\_addr对称链接的2跳邻居的主地址，也就是说节点通过邻居节点N\_neighbor\_main\_addr到达其二跳邻居节点N\_2hop\_addr；

N\_time指定元组过期和必须删除的时间。

2.2.5.MPR表

一个节点维护着其被选为MPR的邻居节点的集合，这些被选为MPR的邻居节点的主地址存放在MPR集中

2.2.6.MPR Selector表

描述选择本节点为MPR的邻居节点的地址。



MS\_main\_addr：是节点的主地址，该节点选择本节点为MPR；

MS\_time指定元组过期的时间和必须删除的时间。

2.2.7.TC拓扑表

网络中的每个节点维护有关网络的拓扑信息。此信息从TC消息中获取，并用于路由表计算。



T\_dest\_addr：目的地节点的主地址，它可以从地址为T\_last\_addr的节点经过一跳到达；

T\_last\_addr：是T\_dest\_addr的MPR；

T\_seq是一个序列号；

T\_time指定此元组过期的时间，以及必须删除的时间。

2.2.8.路由表



R\_dest\_addr：目的地节点地址

R\_next\_addr：下一跳节点地址

R\_dist：本节点到目的节点的距离

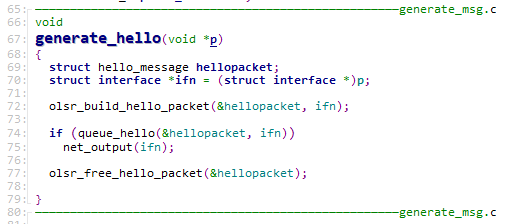
R\_iface\_addr：该节点转发路由信息的接口

**3.代码分析部分**

3.1链路感知与邻居检测

链路感知的机制是HELLO消息的周期性交换。节点必须在每个接口上执行链路感知，以便检测接口和邻居接口之间的链路。因此，对于给定的接口，HELLO消息将包含该接口上的链路列表，以及整个邻居的列表。

3.1.1首先介绍HELLO消息的生成。原则上，HELLO消息服务于三个独立的任务：链路感知，邻居检测和MPR选择。



66-79：generate\_hello()函数用来产生hello消息包。调用olsr\_build\_hello\_packet()函数为指定的接口生成要发送的HELLO数据包。如果创建成功，则通过net\_output()将该HELLO数据包通过指定的接口发送出去。最后释放该HELLO消息。

接下来介绍一下olsr\_build\_hello\_packet函数。（代码部分过长，重新截取了重要部分。）

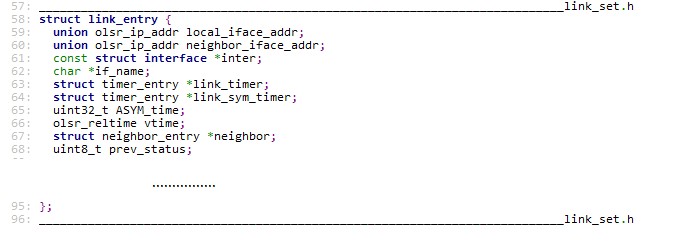


106和113：设置willingness和ttl。HELLO消息只能在一跳范围内传播，所以ttl值设为1.

120-180：遍历该接口的所有链路得到链路状态，用以计算邻居的状态。如果邻居的主地址位于该节点的MPR集合，则将邻居的状态更新为MPR\_NEIGH；如果邻居的主地址位于该节点的邻居列表（即不为MPR），则判断之间的链路状态是否对称并更新邻居的状态域。

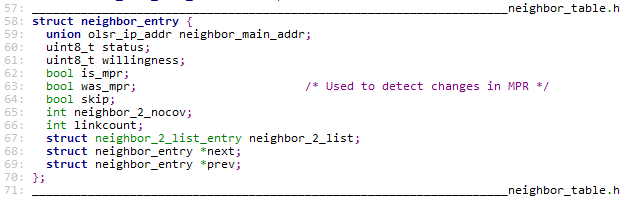
182-186：设置邻居节点的接口地址以及主地址。

3.1.2这部分介绍节点信息的存储方式。

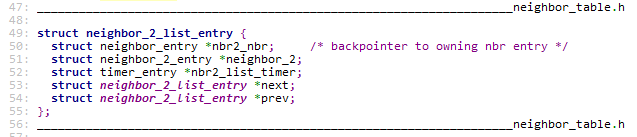


58-62：local\_iface\_addr存储了节点接口的IP地址，neighbor\_iface\_addr存储邻居节点的接口IP地址。

63-68：各种时间的存储结构。link\_timer定时器，link\_sys\_timer定时器。Vtime表示接受该消息之后多少时间之内视其为有效。Neighbor以链表的形式存储邻居节点信息，pre\_status记录上一个节点的状态。

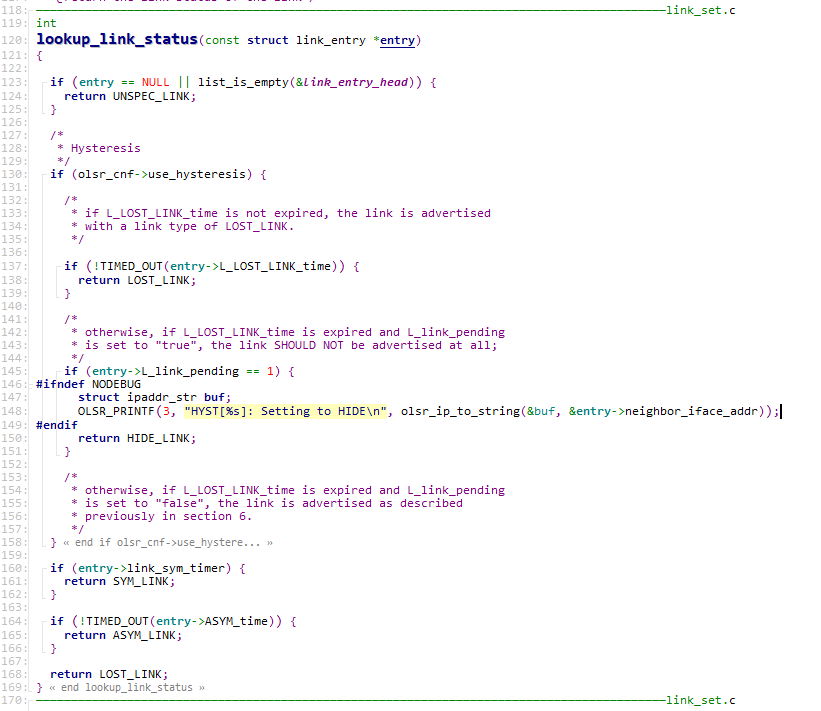


58-69：结构体neighbor\_entry用来存储邻居节点的信息。分别存储了邻居的主地址、状态、转发意愿、覆盖两跳邻居的数量、节点连接的链路数量。其中还存储两个布尔类型的值用来判断该节点是否是MPR以及该MPR是否发生过改变。最后包含了指向两跳邻居列表的指针。

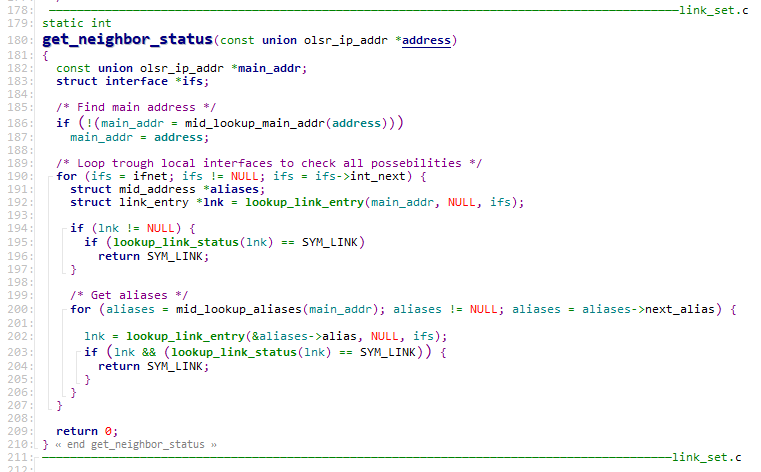


49-54：结构体neighbor\_2\_list\_entry用来存储两跳节点列表中的信息。均采用指针的方式记录节点信息及结点的两跳邻居节点的信息以及列表的有效时间等。

3.1.3该部分介绍节点的操作函数



160-168：该函数的功能是获取链路的状态，状态基于链路条目中的不同超时，根据不同的定时器值将返回不同的链路状态信息，如link\_sys\_timer对应链路状态为对称链路。



182-197：通过查找main\_addr找到节点，然后通过look\_link\_status找到节点链路状态，并判断是否对称。

200-206：查找主地址并找出节点其他端口的IP，判断该节点其他端口的链路状态。并判断该IP所在的链路状态是否对称，并返回对称链路的信息。



该函数的功能是删除节点连路上的所有信息。

369-373：通过邻居端口地址找到拓扑边缘链路SPF并删除。

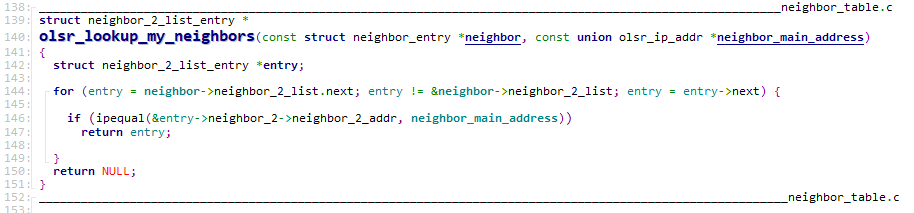
376-381：删除邻居链路信息，删除存储在哈希表中的链路表。

383-397：删除一些定时器的属性，释放链路资源空间，同时通知其他节点及时更新邻居表信息。

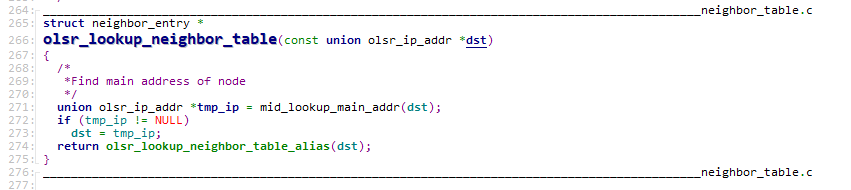
3.1.4关于邻居表的操作函数



59-63：初始化邻居表。将每一个邻居表neighbortable[i]初始化为指向自身的仅有一个节点的链表。



142-150：该函数的功能是检查两跳邻居是否可以通过给定的邻居访问。遍历邻居节点的两跳邻居节点列表。如果找到列表中存在IP地址与给定的地址相匹配的两跳邻居节点，则返回该两跳邻居节点列表结构体。



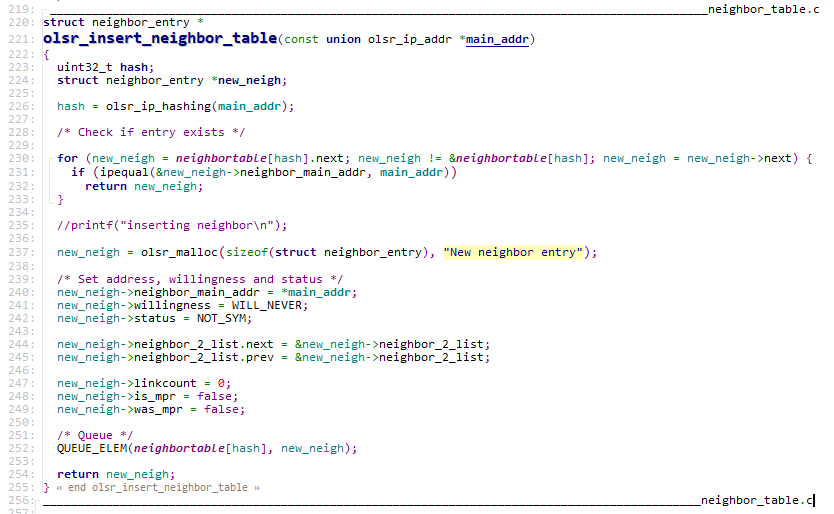
271-274：该函数的功能是基于给定的地址在邻居表中查找邻居项。



该函数的功能为更新邻居的状态。

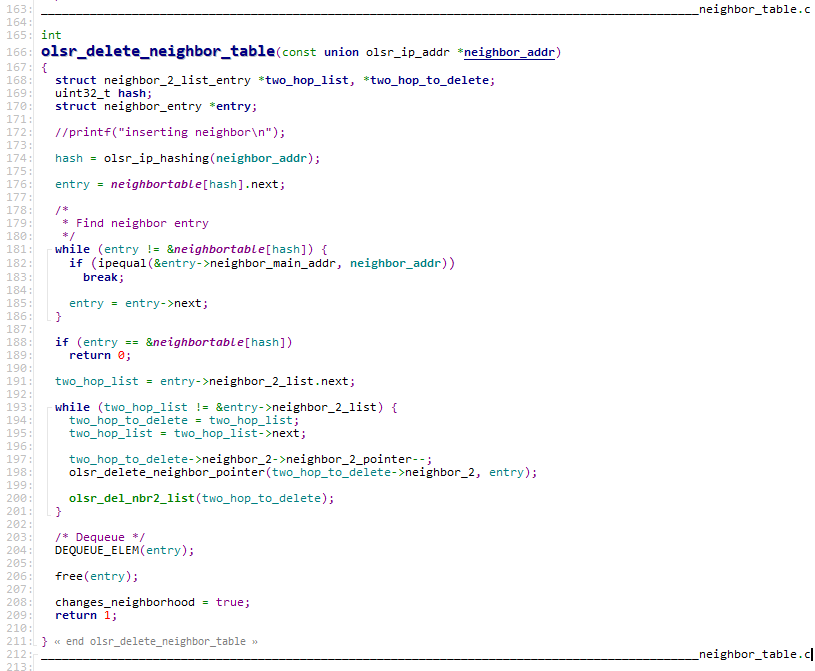
313-328：如果链路状态更新为SYM\_LINK时，原来非对称的链路通知网络重新选举MPR和更新路由表并删除通过该邻居节点到达的两跳邻居节点。

329-339：如果链路状态原先为SYS\_LINK，通知网络重新进行MPR的选取和路由表的更新。

该函数的功能是在邻居节点表中插入一条邻居节点信息。如果已存在该信息返回0，插入成功则返回1。

230-233：检查表项是否存在。

237-249：添加邻居节点信息，将地址、意愿、状态等值初始化。



165-211：该函数的功能是删除邻居表项，删除邻居表项的同时会删除掉该节点存储的两跳邻居列表。

**3.2MPR的生成**

OLSR采用MPR机制对路由信息进行选择性的洪泛。MPR作为OLSR协议的核心部分，算法描述已经在前面描述过了，此处不再赘述。

3.1.1MPR节点的操作



该函数的作用是添加willingness为WILLALWAYS的邻居节点到MPR集合中。

368-383：对于非对称以及转发意愿为WILLNEVER的邻居节点不进行处理，其余节点添加到MPR集合中，并返回添加的节点的数量。



该函数的功能是清除被选为MPR的节点。

246-249：如果节点目前是MPR节点，那就将is\_mpr项设为false，was\_mpr设为true，表明该节点过去是MPR节点，现在不是了。

252-255：当删除一个MPR节点时，该节点存储的两跳邻居列表也应该清除。所以将邻居节点覆盖的两跳邻居节点的数量置0。

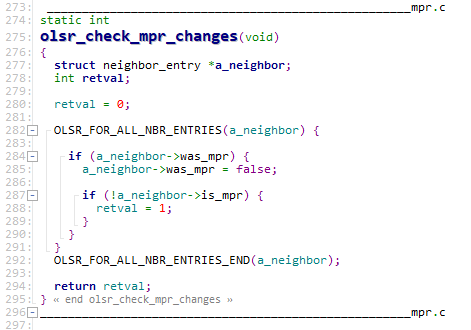


该函数的作用是用来处理已经选定的MPR节点，函数的返回值为一跳邻居节点中MPR的数量。

159-165：对second\_hop\_entries进行遍历

173：该邻居节点被MPR覆盖，将两跳邻居节点被MPR覆盖的数量+1。

181-188：如果两跳节点被MPR覆盖的数量大于全局变量mpr\_coverage，将count递减。



该函数的作用是遍历所有的MPR节点判断其状态是否发生变化，若发生变化返回值为1，否则为0。

282-292：对于结点如果其was\_mpr值为true，表明它过去是MPR，is\_mpr值为false，表示现在不是MPR，所以它的状态发生了变化，将retval置为1。

3.1.2MPR的选择原则是MPR为对称邻居节点，通过MPR可以到达所有的两跳邻居节点，并且MPR的数量要尽可能少。接下来介绍有关两跳邻居节点的函数以及如何周到能够覆盖最多两跳节点的MPR。



该函数的作用是查找一条连接两跳邻居节点的链表。

92-95：定义了两个两跳邻居节点链表（two\_hop\_temp，two\_hop\_list），前者是函数的返回值。

Dup\_neighor记录邻居节点集合中已经存在的节点，two\_hop\_neighbor记录一个两跳邻居节点的局部变量。

97-112：遍历两跳邻居列表two\_hop\_neighbortable。

114-117：寻找邻居表中已经存在的邻居地址neighbor\_2\_addr，忽略与本节点不对称的邻居节点。

122-126：如果该两跳邻居节点不在两跳邻居链表中，且只有一个邻居节点，则将该两跳邻居节点加入链表。



该函数的作用是找到能够覆盖最多两跳节点的MPR。

225-233：如果该一跳邻居节点不是MPR，而且该节点覆盖的两跳邻居节点的数量比maximum大，则更新maximum的值并将该一跳邻居节点作为候选MPR节点。