**OLSR路由协议是由IETF MANET(Mobile Ad hoc NETwork)工作组为无线移动Ad Hoc网提出的**

**一种标准化的表驱动式优化链路状态路由协议。**

参考资料：

<https://wenku.baidu.com/view/abcf45f264ce0508763231126edb6f1aff0071aa.html>

adhoc网络层路由协议总结

**节点之间需要周期性地交换各种控制信息，**

**通过分布式计算来更新和建立自己的网络拓扑图，**

**被邻节点选为多点中继站MPR(MultipointRelay)的节点需要周期性地向网络广播控制信息。**

MPR节点间通信的中继站

**控制信息中包含了把它选为MPR的那些节点的信息(称为MPR Selector)，**

**只有MPR节点被用作路由选择节点，非MPR节点不参与路由计算。**

**OLSR还利用MPR节点有效地广播控制信息，非MPR节点不需要转发控制信息。**

**OLSR主要采用两种控制消息分组，HELLO分组和TC(Topology Control)分组。**

HELLO消息部分

**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延迟或开销，**

**OLSR采用周期性地广播HELLO分组来侦听邻居节点的状态。**//并建立邻居表

**HELLO分组只在一跳的范围内广播，不能被转发。**//是因为hello分组只有一跳的范围吗？

//大概是A为了告诉邻居节点B：“B有邻居A”

（Hello分组消息格式）



Reserved ：保留字段，设为“000000000000”。

Htime：描述此接口的HELLO消息发送时间间隔。

Willingness：描述一个节点为其他节点携带网络流量的意愿。

一个具有 willing\_never的节点永远不会被任何节点选为MPR

一个具有 willing\_always 的节点永远会被选为MPR

默认情况下,一个节点此域应设为willing\_default 。

Link Code：链路类型：描述发送HELLO 分组的节点与该分组中随后的邻 居列表中的邻节点间的链路类型，至少有三种：

(1) ASYM-LINK ：发送HELLO分组的节点与列表中的邻节点间的链路是非对称的。表示可以收到邻节点的消息，但不确定邻节点是否能收到本节点的消息。

(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。

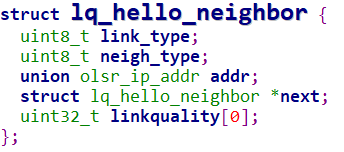
(3) MPR\_LINK ：表示列表中的节点已被发送该HELLO分组的节点选择为 MPR。

该字段还可以提供附加信息，如表示链路中断的 LOST\_LINK等。

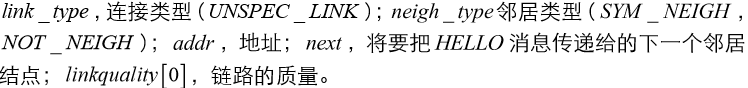
Reserved ：保留字段，必须为“00000000”。

Link Message Size：本链路消息的大小，从“链路类型”字段开始直到下一个“链路类型”字段之前（若无下一个“链路类型”字段，则到分组结尾）。

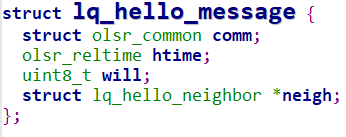
Neighbor Interface Address：邻居地址列表，每一种链路类型之后都紧随一个邻居列表地址，表示发送该HELLO分组的节点到这个邻居列表中的所有节点的链路类型是相同的，都为前面给出的链路类型

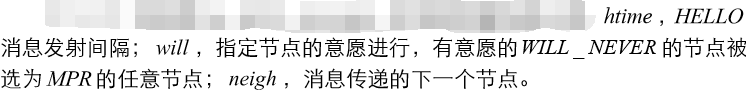
此段代码来自于lq\_packet.h 表示邻居节点集hello消息头部。

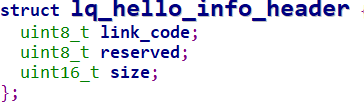
Lq可能是链路质量，link quality的意思



协议中节点利用周期广播的探测包评估与邻居节点之间的链路质量,并交换地理位置信息。

此段代码来自于lq\_packet.h 表示消息数据包的头部。

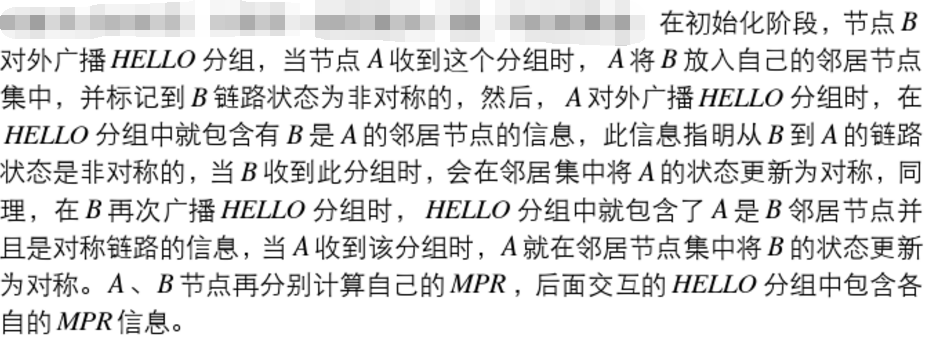






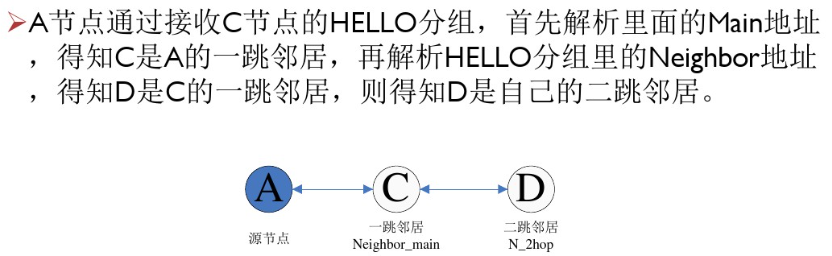
对称链路寻找过程大概就是实现两个节点之间的三次握手

下图是两个节点之间对称链路状态的建立：

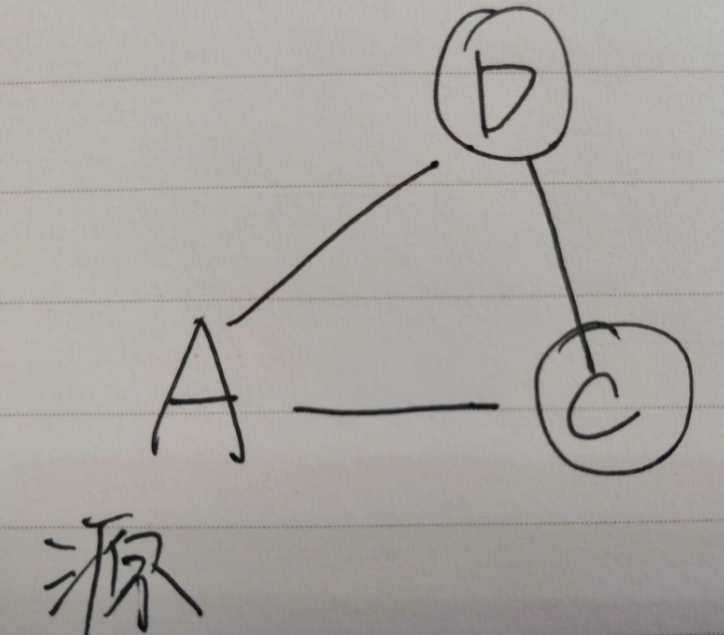


通过以上步骤，网络中节点均建立起自己的对称一跳邻居集合（是通过接收到邻居节点发出的hello信息）和两跳邻居集合（因为hello消息中带有邻居节点信息，所以被认为是二跳邻居）

（如下图所示）



在想如果出现下图所示的情况是怎么处理的？



TC消息部分

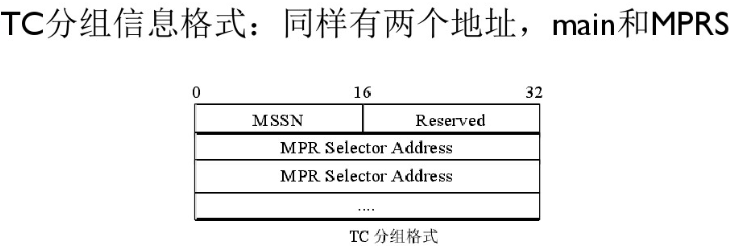
**与HELLO消息相反，TC分组必须被广播到全网。**

//大概hello和TC很像dijkstra和floyd算法的区别

//dijkstra是全局的广播，类似于TC

//而hello类似于floyd

TC消息格式



MSSN不知道是什么意思，大胆猜测是任务序列号的意思

拓扑表是根据TC 分组中的拓扑信息建立的。

根据TC分组中信息建立拓扑表的过程是：

1. 如果拓扑表中存在某个表项，其T \_last(大概是最后一条接收到的TC信息吧) 对应于TC 分组发送源节点地址且其T\_ seq（最后一条接收到的消息的序列号） 大于收到消息中的 MSSN（新接收到的TC消息的序列号） 的值，那么，就不再对TC 分组做进一步处理，丢弃该TC 分组。

（T\_seq和MSSN都属于控制消息的序列号，每发一个控制消息序列号就会增加1，通过接收到序列号的大小来判断消息的新旧程度（即通过比较T\_seq和MSSN的大小关系来判断消息的新旧程度）

2. 删除拓扑表中所有T \_last 对应于TC 分组发送源节点地址，且其T \_seq ）

小于收到分组中 MSSN 的值的表项。

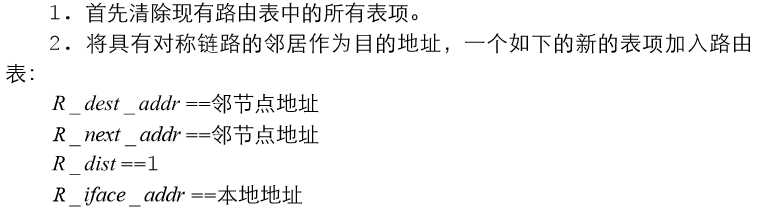
3. 对从TC 分组中接收到的每个 MPR Selector 的地址：如果拓扑表中存在某个条目，其T\_ dest(最后一条接收到的消息的目的地址) 对应于TC 分组中的 MPR Selector 地址，且其T \_last 对应于TC 分组中初始发送节点地址，则更新该条目的保持时间T \_time（不知道是什么）；

否则，就在拓扑表中记录新的拓扑条目

**路由表的计算：**

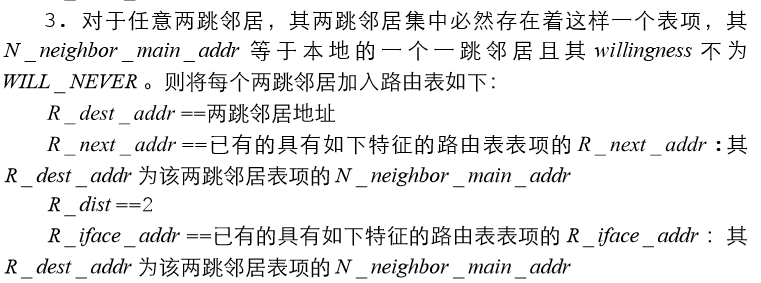
每个节点维护一张保存到每个目的节点的路由表。路由表的计算是基于节点存储的拓扑结构的。接收到TC分组的节点，分析并存储[last-hop, node]连接对，其中“node”是在发送TC分组列表中发现的节点地址。简单来说，为了找到从 给定源节点到较远节点R的路径，必须找到连接对（X，R），然后是找到连接对（Y，X），以此类推，直到发现节点Y 就是源节点的邻居节点时结束连接对的查找。 为了使得到的路径是最优的，转发节点仅选择最小路径上的连接对。路径选择算法在拓扑图的基础上采用了多重Dijkstra算法。当邻居表和拓扑表发生变化或路由失效时都需要更新路由表。

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 节点路由表的添加过程可分为三部分：首先，添加自己的邻节点进入路由表，即跳数 h=1 ；其次，添加自己的两跳邻节点进入路由表，即h=2；最后，循环添加跳数等于 h+1 ( h=2开始)的节点进入路由表。

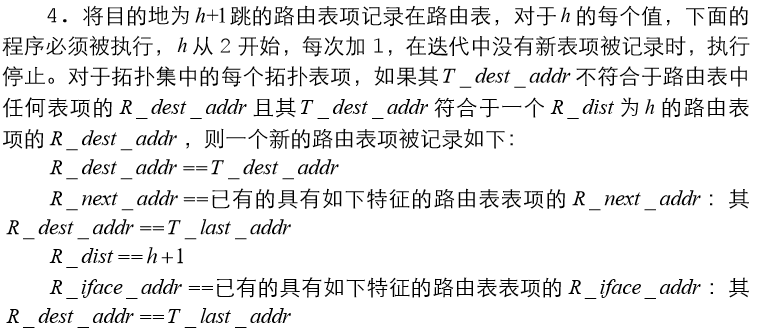


R\_dest\_addr 是目标节点的地址，R\_next\_addr指的是下一跳地址

R\_dist 是指可以经过n的跳数到达目标节点吗?



1. 不是很明白什么叫R\_iface\_addr的目标地址是两跳能够到达的邻居节点的邻居节点



T\_\*\*\*\_addr是指拓扑表项的

R\_\*\*\*\_addr是指路由表项的

**节点保存的信息表：**

**本地链路信息：本地链路信息表存储了该节点和邻居节点的链路信息**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **L\_local\_iface\_addr** | **L\_neighbor\_iface\_addr** | **L\_SYM\_time** | **L\_ASYM\_time** | **L\_time** |

L local iface addr：本地节点的接口地址。

L neighbor iface addr：邻节点的接口地址。

L SYM TIME ：直到此时刻前，链路被认为是对称的。

L ASYM TIME：直到此时刻前，链路被认为是单向的。

L time：链路维护时刻，链路在该时刻失效，必须被删除，当L SYM TIME 和 L ASYM TIME都过期，链路被声明为丢失。（大概就是节点AB之间没有互发hello消息进行验证，链路就会被认定为不存在吧，因为hello是发给所有邻居节点的）

**两跳邻居表：描述邻居节点与对称两跳邻节点间的对称链路。**

|  |  |  |
| --- | --- | --- |
| **N\_neighbor\_main\_addr** | **N\_2\_hop\_addr** | **N\_time** |

N neighbor main addr ：表示邻节点的地址。

N 2hop addr ：表示与 N neighbor main addr有对称链路的两跳邻节 点的地址。

N time：表示表项到期必须被移除的时间。

**MPR Selector 表:只有在MPR节点中才会存在ms表，因为ms节点是选择其成为mpr节点的节点吖**

节点为判断转发哪些控制消息，需要维护关于其MPR Selector的信息。

根据 接收到的HELLO分组，节点就可以构造自己的MPR Selector表。

1.MPR节点是如何确定自己成为MPR节点的

MS main addr ：MPR Selector节点的地址。

MS time：该MPR Selector集表项的保持时间，当MPR Selector 集过期时 要及时删除。

**TC 分组重复记录表**

MPR节点广播和转发TC 分组，在这个过程中，一个节点可能会多次收到同 一个TC 分组，为了避免重新处理已经收到并处理过的TC 分组，每个节点维护 一个TC 分组重复记录表，

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| D\_addr | D\_seq\_num | D\_retransmitted | D\_iface\_list | D\_time |

D addr：最初发送该分组的节点的地址.

D seq num：TC 分组的序列号，用于区分新旧TC 分组。

D retransmitted ：为一个布尔值，用来表示此消息是否被重传过。

D iface list：这个消息被接收的接口地址列表。

D time：该表项的保持时间，表项到期时必须被删除。

节点接收到TC 分组后，首先检测其分组重复记录表，判断此前是否已收到 过相同分组。如果发现了相关条目，就丢弃该分组。

**拓扑表**

网络中每一个节点都维护一张拓扑表，表中记录了从TC 分组获得的网络拓 扑信息。节点根据这一信息计算路由表。节点将网络中其他节点的多点中继的信息作为拓扑表项 记录在拓扑表中

|  |  |  |  |
| --- | --- | --- | --- |
| T\_dest\_addr | T\_last\_addr | T\_seq | T\_time |

T dest：MPR选择节点的地址，表示该节点已经选择节点T last作为其 MPR。

T last：被 \_T dest选为MPR的节点的地址。

T seq：表示 \_T last已经发布了它保存的序列号为 \_T seq的MPR Selector 集合的控制信息。

T time：表项的保持时间，过期后就失效，必须被删除。

**路由表**

网络中每个节点维护一个路由表，表中保存了节点到网路中所有可达目的节 点的路由，对于路由已知的网络中的每一个目的地，表项被存储在路由表中，所 有路由未到达或部分已知的表项不被记录在表中

|  |  |  |  |
| --- | --- | --- | --- |
| R\_dest\_addr | R\_next\_addr | R\_dist | R\_iface\_addr |

R dest addr ：路由目的节点地址。

R next addr：路由的下一跳节点地址。

R dist：本节点到目的节点的距离。

R iface addr ：表示下一跳节点通过本地接口 R iface addr 到达