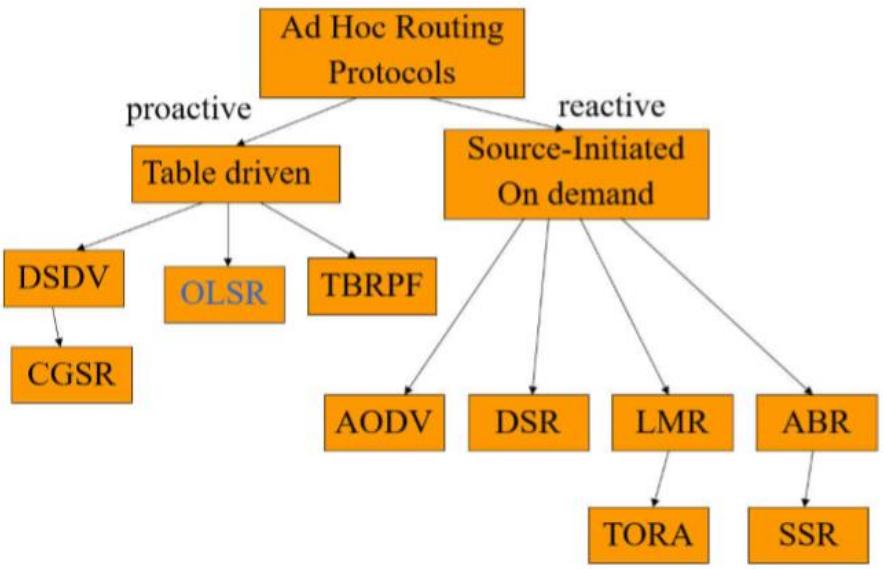
OLSR 协议

目录

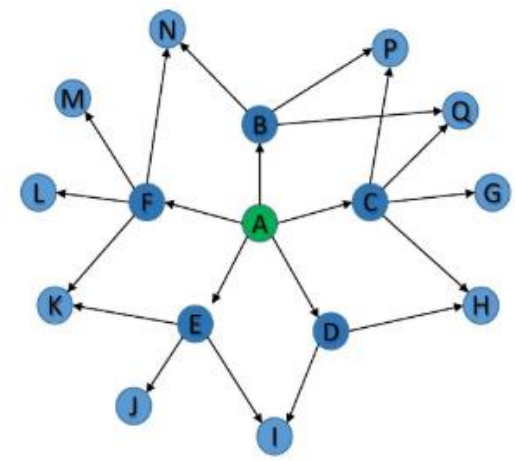
1. 介绍

OLSR 协议是针对移动无线网络需求而形成的经典链路状态算法的优化。是由 IETF MANET(移动自组织网络)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。该协议中使用的关键概念是多点中继（MPR)。MPR 是在洪泛过程中转发广播消息的选定节点。与传统的泛洪机制相比，该技术大大减少了消息开销，在传统泛洪机制中，每个节点在接收到消息的第一个副本时都会重传每个消息。在 OLSR 中，链接状态信息仅由当选为 MPR 的节点生成，负责转发流量。MPR 通过减少所需的传输次数，为泛洪控制流量提供了一种有效的机制。OLSR 协议在 Ad hoc 网络中的位置如下图所示：

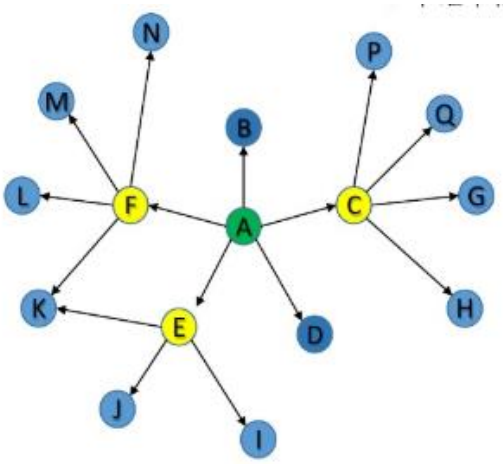


在 OSLR 协议中节点之间需要周期性地交换各种控制信息，通过分布式计算来更新和建立自己的网络拓扑图，被邻节点选为多点中继

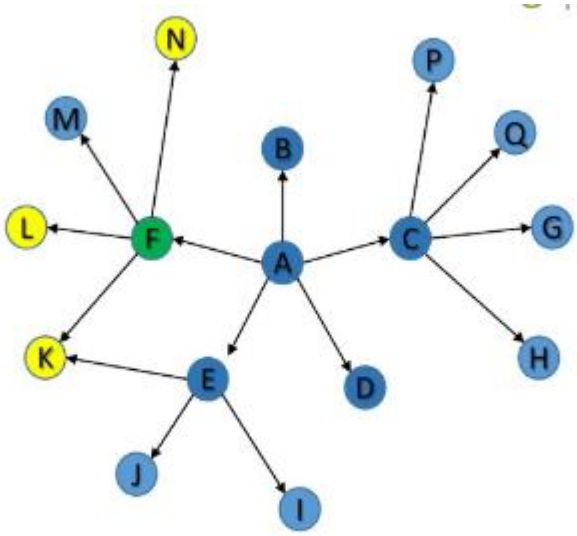
站 MPR 的节点需要周期性地向网络广播控制信息。



纯扩散机制



MPR机制



MPR机制

上图中黄色节点就是我们选择的中继节点，即使为多跳范围内的消息照样遵循 MPR 机制，在路由计算中，MPR 用于形成从给定节点到网络中任何目的地的路由。

1.1 OSLR 协议中的关键概念

OLSR 协议的标准中存在较多专业术语，其中关键概念有以下这些：

节点：一个 MANET 网络中的路由器，并运行 OLSR 协议。

OLSR 接口：一个运行 OLSR 协议的网络设备接口。一个节点可能会有多个 OLSR 接口，每个接口有不同的 IP 地址。

非 OLSR 接口：不运行 OLSR 协议的网络设备接口。

主要地址：一个节点的主要地址，被用于 OLSR 协议控制信息的初始地址，是节点 OLSR 接口的一个 IP 地址。

邻居节点：如果节点可以监听到节点 X，则节点 X 是节点的邻居节点。

2 跳邻居：通过邻居节点监听到的节点是 2 跳邻居节点，注意这里可以包含节点自身以及某些 1 跳邻居节点。

严格 2 跳邻居：即不是节点自身或其邻居节点，而是严格通过邻居节点监听到的节点，即邻居的邻居，但又不是本节点的邻居的所有节点。

MPR（多点中继）：某个节点被它的一跳邻居节点选定为 MPR，则需要转发选定它的邻居节点的多有广播信息。

MS（MPR 选择者）：选定某个节点为 MPR ，则该节点自身为 MPR 节点的 MS 节点。

链路：两个不同 OLSR 节点接口之间相互监听形成链路。

对称链路：两个 OLSR 接口之间已经认证的双向链路。

非对称链路：即两个 OLSR 接口之间只有一个方向上进行了验证单向链路。

对称一跳邻居：邻居间至少有一条对称链路。

对称 2 跳邻居：有对称链路的 2 跳邻居。

对称严格 2 跳邻居：有对称链路的严格 2 跳邻居。

1.2 协议中的一些常量

C=1/16S，C 是“有效时间”计算的比例因子,设计“有效时间”，使网络中的节点可以具有不同且可单独调整的发射间隔,同时仍可以完全互操作。

发送间隔：

HELLO_INTERVAL = 2 秒

REFRESH_INTERVAL = 2 秒

TC_INTERVAL = 5 秒

MID_INTERVAL = TC_INTERVAL

HNA_INTERVAL = TC_INTERVAL

保留时间：

NEIGHB_HOLD_TIME = 3 x REFRESH_INTERVAL

TOP_HOLD_TIME = 3 x TC_INTERVAL

DUP_HOLD_TIME = 30 秒

MID_HOLD_TIME = 3 x MID_INTERVAL

HNA_HOLD_TIME = 3 x HNA_INTERVAL

讯息类型

HELLO_MESSAGE = 1

TC_MESSAGE = 2

MID_MESSAGE = 3

HNA_MESSAGE = 4

连接类型

UNSPEC_LINK = 0

ASYM_LINK = 1

SYM_LINK = 2

LOST_LINK = 3

邻居类型

NOT_NEIGH = 0

SYM_NEIGH = 1

MPR_NEIGH = 2

连接滞后

HYST_THRESHOLD_HIGH = 0.8

HYST_THRESHOLD_LOW = 0.3

HYST_SCALING = 0.5

意愿

WILL_NEVER = 0

WILL_LOW = 1

WILL_DEFAULT = 3

WILL_HIGH = 6

WILL_ALWAYS = 7

其他常数

TC_REDUNDANCY = 0

MPR COVERAGE = 1

MAXJITTER = HELLO_INTERVAL / 4

1.3 OSLR 的适用性

OLSR 是用于移动自组织网络的主动路由协议。它非常适合大型密集型移动网络，因为使用 MPR 实现的优化在这种情况下效果很好。与传统的链路状态算法相比，网络越大，网络越密集，可以实现的优化程度越高。

OLSR 非常适合大型网络， OLSR 作为一种主动协议，也适用于通信对随时间变化的场景：在这种情况下，由于始终为所有已知目的地维护路由，因此不会生成其他控制流量。

1.4 OSLR 的特性

OLSR 可以通过减少周期性控制消息传输的时间间隔来优化对拓扑变化的反应性。由于 OLSR 持续维护到网络中所有目的地的路由，因此该协议对于流量模式很有用，在流量模式中，大量的节点子集与另一个较大的节点子集进行通信，并且[源，目标]对随时间变化。

OLSR 设计为以完全分布式的方式工作，并且不依赖于任何中央实体。该协议不需要可靠的控制消息传输：每个节点都会定期发送控制消息，因此可以合理地丢失某些此类消息。由于冲突或其他传输问题，这种损失在无线网络中经常发生。

OLSR 不需要按顺序传递消息。每个控制消息包含一个序列号，该序列号对于每个消息递增。所以即使消息在传输过程中已被重新排序，控制消息的接收者可以轻松地识别出哪些信息是最新的。

1.5 多点中继（MPR)

多点中继的想法是通过减少同一区域中的冗余重传来最大程度地减少网络中泛洪消息的开销。网络中的每个节点都会在其对称的 1 跳邻居中选择一组节点，这些节点可能会重传其消息。这组选定的相邻节点称为该节点的“多点中继”(MPR)集。节点 N 的邻居不是 MPR，它接收并处理广播消息，但不重传从节点 N 接收到的广播消息。

每个节点从其 1 跳对称邻居中选择其 MPR 集。选择该集合以使其覆盖所有对称的严格 2 跳节点。N 的 MPR 集（表示为 MPR（N））是 N 的对称 1 跳邻域的任意子集，它满足以下条件：N 的对称严格 2 跳邻域中的每个节点都必须具有对称链接对 MPR（N)。MPR 集越小，路由协议产生的控制流量开销越小。每个节点都维护有关已选择它作为 MPR 的邻居集的信息。

2.协议功能

OLSR 被模块化为功能的“核心”和一组辅助功能，该功能是协议运行所必需的。核心自行指定了能够在独立的 MANET 中提供路由的协议。每个辅助功能提供其他功能，这些功能可能适用于特定情况。

2.1 核心功能

OLSR 的核心功能指定了一个节点的行为，该节点配备了参与 MANET 并运行 OLSR 作为路由协议的 OLSR 接口。核心功能包括：数据包格式与转发、信息库、主地址、hello 消息、链路检测、邻居监测、拓扑发现、路由表计算、节点配置。

数据包格式的通用规范和优化的泛洪机制被用作所有 OLSR 控制流量的传输机制。

链路检测是通过定期在接口上发出 HELLO 消息来完成的，以检查连接性。链接检测的结果是一个本地链接集，描述了“本地接口”和“远程接口”之间的链接，即相邻节点上的接口。

邻居检测：给定一个只有单个接口节点的网络，一个节点可以直接从链接感知交换的信息中扣除邻居集，单个接口节点的“主地址”是该接口上唯一接口的地址节点。

MPR 选择的目的是使节点选择其邻居的子集，以便由这些选定的邻居重传的广播消息将被两跳远的所有节点接收。计算节点的 MPR 集，以使其对于每个接口都满足此条件。通过定期交换 HELLO 消息来获取执行此计算所需的信息。

辅助功能包括：非 OLSR 接口、链路层通知、高级链接监测、冗余拓扑、冗余 MPR 泛洪

2.1.1 数据包格式和转发

OLSR 使用统一的数据包格式对与该协议相关的所有数据进行通

信。这样做可以促进协议的可扩展性。还提供了一种将不同种类信息附加在单个数据包中传输，使给定的实现方式可以优化以利用网络提供的最大帧大小。这些数据包嵌入 UDP 数据报中，以通过网络传输。每个数据包封装一个或多个消息。消息共享一种通用的报头格式，这使节点能够正确接受和重传未知类型的消息。可以将消息泛洪到整个网络上，也可以将泛洪限制为距离消息始发者直径（以跳数为单位）以内的节点。节点可以检查消息的头，以获得有关到消息始发者的距离（以跳数计）的信息。

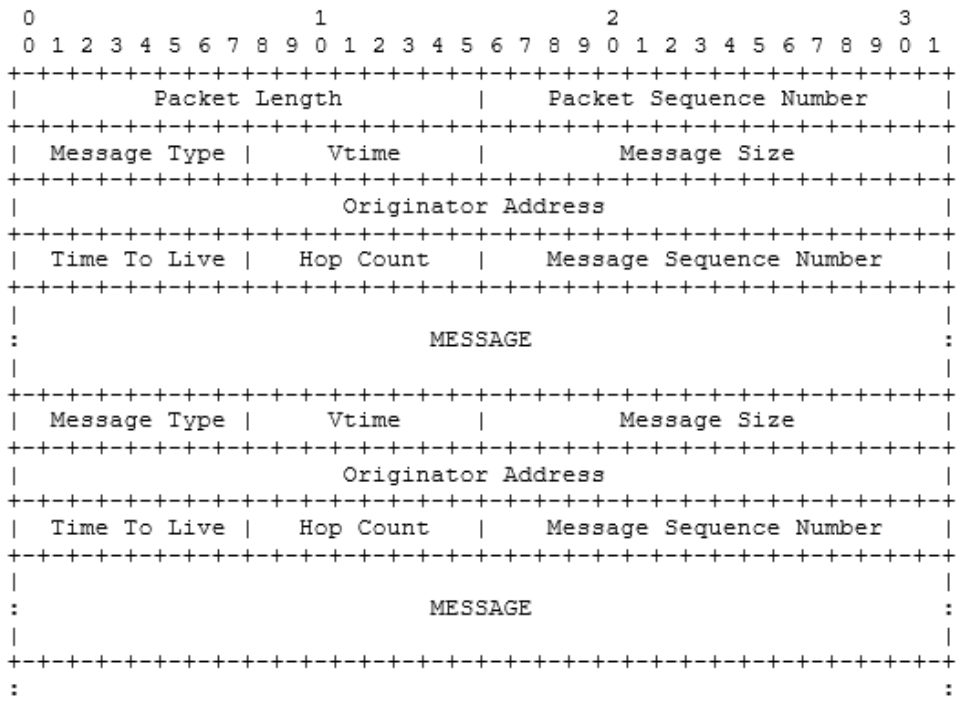
协议与端口号

OLSR 中的数据包使用 UDP 进行通信。698 端口分配给 OLSR 协议专用。

数据封装结构

MAC header	IP header	UDP header	OLSR header	Data

OLSR 中数据包格式如下（省略了 IP 和 UDP 头部）：



封包长度：以字节为单位来计算的数据包的长度

包序号：每次发送新的数据包的时候，数据包的序列号（PSN）增加 1，为每个接口维护一个单独的数据包序列号。我们可以通过 IP 标头获得传输数据包的接口的 IP 地址。如果数据包不包含消息，即数据包长度小于或等于数据包头的大小，则该数据包会被静默丢弃。

信息类型:表示在消息部分将可以接收到什么消息,例如Hello、TC、MID、HNA。

Vtime：表示节点在接收到信息之后多长时间将数据包中包含的信息视为有效。有效时间为 C*(1+a/16)* 2^b，其中 a 是由 Vtime 字段的四个最高位表示的整数，而 b 是由 Vtime 字段的四个最低位表示的整数。

信息大小：以字节为单位，从“消息类型”字段的开始到下一个“消息类型”字段的开始，没有后续消息就一直到这个数据包包的结

束。

发起人地址：该字段包含最初生成此消息的节点的主要地址。该字段不一定与 IP 标头中的源地址一样，源标头每次都更改为正在重新发送此消息的中间接口的地址。发起者地址字段是固定的，不要在重传中更改。

生存时间：该字段包含一条消息将被发送的最大跳数。每一次发送,生存时间必须减 1。当节点接收到生存时间等于 0 或 1 的消息时，该消息不得进行发送。可以通过设置这个字段，可以限制洪泛半径。

跳数：表示已将送达的节点数，在重新发送消息之前，跳数必须加 1.最初为 0。

消息序列号：在生成消息时，“发起者”节点将为每个消息分配一个唯一的标识号。该编号将插入到消息的“序列号”字段中。对于源自该节点的每个消息，序列号将增加 1。确保给定消息不会被任何节点多次转发。

2.2.2 数据包处理和消息泛洪

在接收到基本分组时，节点检查每个“消息头”。根据“消息类型”字段的值，节点可以确定消息是否可用。一个节点可能多次收到相同的消息。因此，为了避免重新处理已经接收和处理的某些消息，每个节点都维护一个重复集。在此集合中，节点记录有关最近接收到的消息的信息，其中要避免重复处理消息。对于此类消息，节点记录一个“重复元组”（D_addr，D_seq_num，D_retransmitted，

D_iface_list，D_time），其中 D_addr 是消息的始发者地址，D_seq_num 是消息的消息序列号，D_retransmitted 是一个布尔值，指示是否已经重发了消息，D_iface_list 都是接收到该消息的接口的地址列表，D_time 指定元组过期和必须删除的时间。

在接收到基本分组之后，如果数据包不含任何消息或者生存时间小于等于 0，或者消息的始发者地址是接收节点的主地址，就删除该消息。

消息加工条件：如果重复集中有一个元组，发起者地址和消息序列号地址一致，则消息就已经被处理过了。

转发条件：发起者地址和消息序列号一致，而且接口地址在 D_iface_list 中，该消息已被考虑转发，不应再次重发，未实现消息的消息类型则进行默认转发。

2.2.3 默认转发算法

默认转发算法：

1、 如果没检测到消息的发送方接口地址在节点的对称 1 跳附近，转发算法就此停止，消息不得转发。

2、 如果重复集中存在一个元组，发起者地址和消息序列号相同，D_retransmitted 为 false，并且 D_iface_list 中的地址中不包括接收消息的接口（的地址），则考虑是否进行转发。

3、 如果不存在始发者地址，则会进一步考虑转发。

4、 如果继续转发，就按照步骤继续下去，如果发送者接口地址是此节点的 MPR 选择器的接口地址，并且消息的生存时间大于“ 1”，则必须重新发送消息

5、 如果存在重复集中的条目，而且有相同的发起者地址和相同的消息序列号，将条目更新如下：

D_time =当前时间+ DUP_HOLD_TIME。

接收接口（地址）已添加到 D_iface_list。

当且仅当消息将根据步骤 4 重新发送时，D_retransmitted 才设置为 true。

否则，重复集中的条目将记录为：

D_addr =发起者地址

D_seq_num =消息序列号

D_time =当前时间+ DUP_HOLD_TIME。

D_iface_list 包含接收接口地址。

如果且仅当根据步骤 4 必须重新发送该消息时，则：

6、 消息的 TTL 减小 1。

7、 消息的跳数增加一。

8、 消息在所有接口上广播（注意：消息头的其余字段应保持不变）

注意：处理和转发消息是两个不同的动作，受不同规则的限制。处理涉及使用消息的内容，而转发涉及为网络的其他节点重新传输同一消息。

OLSR 的核心功能所需的消息类型为：

HELLO 消息:执行链路感知，邻居检测和 MPR 信令的任务，

TC 消息:执行拓扑声明的任务。

MID 消息:执行声明节点上存在多个接口的任务。

2.2.4 消息的发送

出于各种原因（主要是计时器与数据包处理的交互），从相邻节点发出的控制流量可能会变得同步，从而使多个相邻节点尝试同时传输控制流量。所以我们需要寻找一个办法去解决这个冲突，所以我们在产生消息的间隔中增加一定量的跳动，这个跳动是一个随机值。对于使用跳动的节点，实际的消息间隔是 MESSAGE_INTERVAL-跳动。其中跳动是一个值，是从间隔[0，MAXJITTER]中随机选择的，而 MESSAGE_INTERVAL 是为要发送的消息指定的消息间隔的值（例如，HELLO 消息为 HELLO_INTERVAL，TC 消息为 TC_INTERVAL 等）。

3. 信息库

3.1 多接口信息库

对于网络中的每个目的地，都会记录“接口关联元组”（I_iface_addr，I_main_addr，I_time）。I_iface_addr 是目的节点

的接口地址，I_main_addr 是目的节点的主地址。I_time 指定该元组过期并且必须被删除的时间。

3.1.1 链接集

节点记录一组“链接元组”（L_local_iface_addr，L_neighbor_iface_addr，L_SYM_time，L_ASYM_time，L_time）。L_local_iface_addr 是本地节点的接口地址(即,链接的一个端点)，L_neighbor_iface_addr 是邻居节点的接口地址（即，链接的另一端点)，L_SYM_time 是直到链接被视为对称的时间点，L_ASYM_time 是直到邻居接口被视为可以听到的时间，而 L_time 指定此记录到期并且必须被删除的时间。如果 L_SYM_time 没有过期，则必须将链路声明为对称的。如果 L_SYM_time 过期,则必须将链路声明为非对称的。当 L_SYM_time 和 L_ASYM_time 过期时，该链接被视为丢失。

3.1.2 邻居集

节点记录一组描述邻居的“邻居元组”（N_neighbor_main_addr，N_status,N_willingness)。N_neighbor_main_addr 是邻居的主地址，N_status 指定该节点是 NOT_SYM 还是 SYM。N_willingness 是介于 0 到 7 之间的整数，用于指定节点代表其他节点承载流量的意愿。

2 跳邻居集

节点记录一组“ 2 跳元组”（N_neighbor_main_addr，N_2hop_addr,N_time),描述其邻居与对称 2 跳邻域之间的对称链接。

N_neighbor_main_addr 是邻居的主要地址，N_2hop_addr 是具有到 N_neighbor_main_addr 的对称链接的 2 跳邻居的主要地址，并且 N_time 指定元组过期和必须删除的时间。

3.1.3 MPR 集

节点维护一组被选为 MPR 节点的邻居。它们的主要地址在 MPR 集中列出。

3.1.4 MPR 选择集

一个节点记录一组 MPR 选择者元组（MS_main_addr，MS_time），描述选择该节点作为 MPR 的邻居。MS_main_addr 是节点的主地址，该节点已选择该节点作为 MPR。MS_time 指定元组到期并且必须删除的时间。

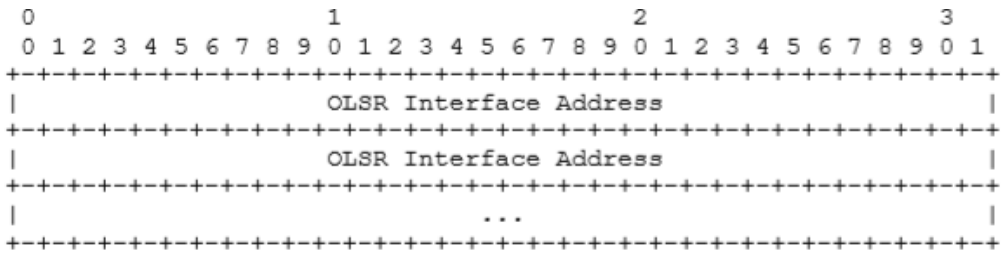
3.1.5 拓扑集

网络中的每个节点都维护有关网络的拓扑信息。该信息是从 TC 消息中获取的，并用于路由表计算。对于网络中的每个目的地，至少记录一个“拓扑元组”（T_dest_addr,T_last_addr,T_seq,T_time）。T_dest_addr 是目的地节点的主地址，可以从具有主地址 T_last_addr 的节点一跳到达。通常，T_last_addr 是 T_dest_addr 的 MPR。T_seq 是一个序列号，而 T_time 指定该元组到期且必须被删除的时间。

3.2 主要地址和多个接口

对于单个 OLSR 接口节点，OLSR 接口地址和对应的主地址之间的关系很简单：主地址是 OLSR 接口地址。对于多个 OLSR 接口节点，通过交换多接口声明（MID）消息来定义 OLSR 接口地址和主地址之间的关系。每个有多个接口的节点定期向网络其他节点使用 MPR 泛洪机制将接口配置信息泛洪到网络中的的所有节点。

3.2.1 MID 消息格式



将其作为通用数据包格式的数据部分发送，并将“消息类型”设置为 MID_MESSAGE。应将生存时间设置为 255（最大值），以将消息传播到整个网络，将 Vtime 设置为 MID_HOLD_TIME 的值。

除了始发者节点的主地址之外的所有接口地址都放在 MID 消息中。如果在仍然有接口地址尚未插入 MIDmessage 的同时达到了最大允许的消息大小（由网络施加），则将生成更多的 MID 消息，直到发送了设置的整个接口地址为止。

只有一个接口地址参与 MANET 的节点（即运行 OLSR）不得生成任何 MID 消息。具有多个接口的节点，其中只有一个接口参与 MANET 并运行 OLSR，则不得生成任何 MID 消息。具有多个接口的节点，其中

有多个接口参与MANET并运行OLSR，一定生成MID消息。

3.2.2 MID消息处理

收到MID消息后，必须从消息头的Vtime字段中计算出“有效时间”，并更新多接口信息库。更新规则：

1.如果该消息的发送者接口（注：不是发起者）不在该节点的对称1跳附近，则必须丢弃该消息。

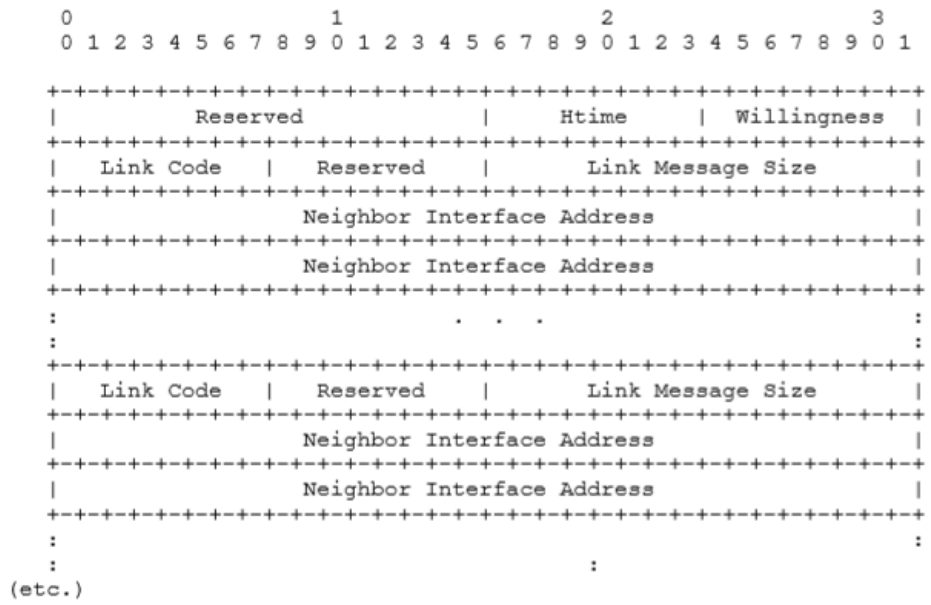
2.如果存在一个元组，I_iface_addr ==接口地址，I_main_addr ==发起者地址，然后将该元组的保持时间设置为：I_time =当前时间+有效时间。

3.如果不存在这个元组，将接口加在信息库中，I_iface_addr =接口地址，I_main_addr =发起者地址，I_time =当前时间+有效时间。

4.HELLO消息

4.1 HELLO消息格式

OLSR协议采用了周期性交换HELLO消息的机制去填充本地连接信息库和邻居信息库。消息格式如下：



将其作为通用数据包格式的数据部分发送，消息类型设置为 HELLO_MESSAGE，TTL 字段设置为 1，并且 Vtime 相应地设置为 NEIGHB_HOLD_TIME 的值

保留：此字段设置为“ 0000000000000”

Htime：该字段指定该特定接口上的节点所使用的 HELLO 发射间隔. HELLO 发射间隔＝ C *（1 + a / 16）* 2 ^ b [以秒为单位]，其中 a 是由 Htime 字段的四个最高位表示的整数，b 是由 Htime 字段的四个最低位表示的整数。

Willingness: 该字段指定节点为其他节点承载和转发流量的意愿。具有意愿 WILL_NEVER 的节点绝不能被任何节点选择为 MPR。愿意将 WILL_ALWAYS 的节点始终选择为 MPR。

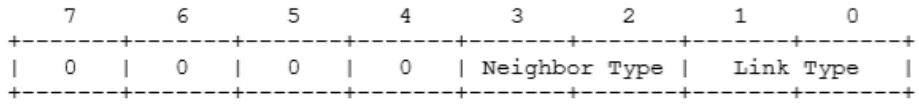
连接码：该字段指定有关发送方接口与邻居接口列表之间的链接的信息。它还指定有关邻居状态的信息。节点不知道的链接码会被丢弃。

连接消息大小：以字节为单位，从“链接代码”字段的开始到下

一个“链接代码”字段（或-如果没有更多的链接类型，则从消息结尾）开始计算。

邻居接口地址：邻居节点接口地址

连接代码字段:连接代码值小于 16 的情况下,解释为两个字段，每个字段两位，一个为邻居类型，另一个为连接类型



OLSR 有四个特定的链接类型：

UNSPEC_LINK 表示未提供有关链接的特定信息。

ASYM_LINK 表示链接是非对称的。

SYM_LINK 表示链接对称。

LOST_LINK 表示链接已丢失。

有三种邻居类型：

SYM_NEIGH 表示邻居与该节点至少有一条对称链路。

MPR_NEIGH 指示邻居具有至少一条对称链路,并且已被发送方选择为 MPR。

NOT_NEIGH 指示节点不再或尚未成为对称邻居。

4.2 Hello 消息的生成

Hello 消息执行三个独立的任务：链路感应、邻居监测、MPR 选择信令。

节点必须在每个接口上执行链路检测,以便检测接口和邻居接口

之间的链路。节点必须在每个接口上通告其整个对称的 1 跳邻居，以执行邻居检测。设置 Vtime 字段，使其与节点的 NEIGHB_HOLD_TIME 参数的值相对应。设置 Htime 字段，使其对应于节点的 HELLO_INTERVAL 参数的值。

对于链接集中的每个元组，其中 L_local_iface_addr 是要传输 HELLO 的接口，并且 L_time> =当前时间（即，未过期）。

链接类型按照以下规则进行设置：

1. 如果 L_SYM_time> =当前时间（未过期），链接类型= SYM_LINK
2. 如果 L_ASYM_time> =当前时间（未过期）且 L_SYM_time <当前时间（已过期），链接类型= ASYM_LINK
3. 如果 L_ASYM_time <当前时间（已过期）并且 L_SYM_time <当前时间（已过期），链接类型= LOST_LINK

邻居类型按照以下规则进行设置：

1.如果对应于 L_neighbor_iface_addr 的主地址包含在 MPR 集中，邻居类型= MPR_NEIGH

2.如果对应于 L_neighbor_iface_addr 的主地址包含在邻居集中：如果 N_status == SYM，邻居类型= SYM_NEIGH。如果 N_status == NOT_SYM，邻居类型= NOT_NEIGH。

5. 链路感应

链接检测会填充本地链接信息库。链路感测仅与 OLSR 接口地址以及在此类 OLSR 接口之间交换数据包的能力有关。链接感知的机制

是周期性交换HELLO消息。每个节点应检测其自身与邻居节点之间的链接。无线电传播的不确定性可能会使某些链路单向。因此，必须在两个方向上都检查所有链接，以便被视为有效。

为了链路感测的目的，每个邻居节点（更具体地，到每个邻居的链路）具有“对称”或“非对称”的关联状态。“对称”表示到该邻居节点的链路已被验证为双向，即可以在两个方向上传输数据。“非对称”表示已经听到了来自该节点的HELLO消息，但是无法确认该节点也能够接收消息。

5.1 链路感应中的HELLO消息处理

HELLO消息的“发起者地址”是发出该消息的节点的主地址。收到HELLO消息后，节点应更新其链接集。既不转发也不记录在重复集中。更新链接集的规则如下：

1.收到HELLO消息后，如果不存在带有L_neighbor_iface_addr ==源地址，则创建一个新的元组：L_neighbor_iface_addr =源地址，L_local_iface_addr =接收到HELLO消息的接口的地址，L_SYM_time =当前时间-1，L_time =当前时间+有效时间。

2.如果有这个源地址，修改L_ASYM_time =当前时间+有效时间；如果节点在链接消息中列出的地址中找到接收到 HELLO 消息的接口的地址，则对元组进行如下修改：

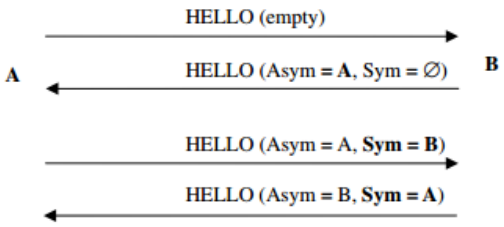
2.1 如果链接类型等于LOST_LINK，L_SYM_time =当前时间-1（即已过期）

2.2 如果链接类型等于 SYM_LINK 或 ASYM_LINK，L_SYM_time = 当前时间+有效时间，L_time = L_SYM_time + NEIGHB_HOLD_TIME，L_time =最大值（L_time，L_ASYM_time）

6. 邻居检测

邻居检测会填充邻居信息库，并将自身与节点和节点主地址相关联。链接集保留有关链接的信息，而邻居集保留有关邻居的信息。这两个集合之间存在明显的关联，因为当且仅当两个节点之间至少有一个链接时，一个节点才是另一个节点的邻居。

6.1 检测步骤



以 A 和 B 两个节点举例，想要检测它们之间链路的状态，需要进行以下几步：

1. A 向 B 发送空的 HELLO 消息。

2. B 接收到消息后，由于没有在消息中找到 B 自己的地址，所以将 A 设为非对称邻节点。2 秒后，B 向 A 发送携带 A 地址的 HELLO 消息。

3. A 接收到消息后，在消息中找到了自己的地址，所以将 B 定义为对称链路，接着再向 B 发送带有 B 地址的 HELLO 消息。

4. B 收到最新消息后，将 A 视为对称邻节点。

基于这个机制，OLSR 中的节点可以进行邻节点的探测，如果与某个邻节点的连接状态变为双向对称连接，则记录在 neighbor set 中，并开放接口连接邻节点；如果连接状态建立失败，则删除记录。

6.2 检测中 HELLO 消息处理

每次出现链接时，即每次创建链接元组时，都必须创建关联的邻居元组（如果尚不存在），N_neighbor_main_addr = L_neighbor_iface_addr 的主地址。每次链接改变时，即每次修改链接元组的信息时，节点必须确保相关邻居元组的 N_status 改变，如果邻居具有表示对称链接的任何关联链接元组（即 L_SYM_time>=当前时间），N_status 设置为 SYM，否则 N_status 设置为 NOT_SYM。

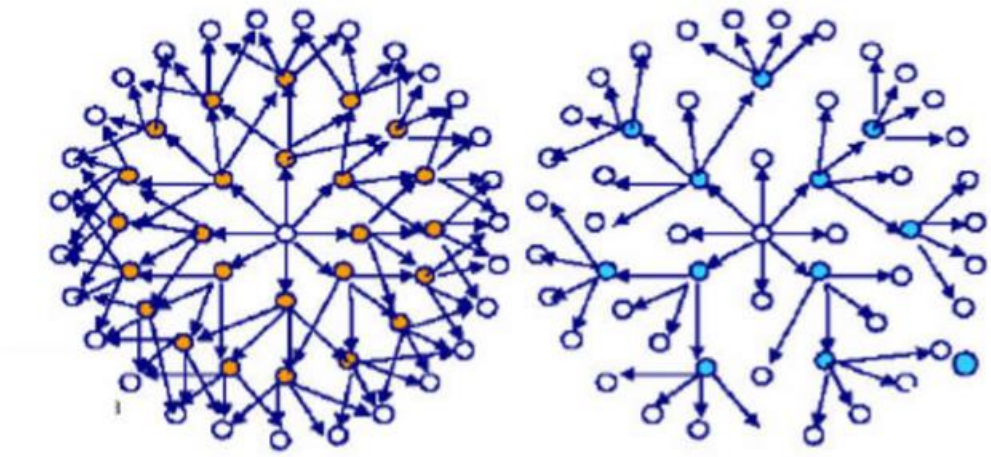
邻居监测的消息处理：首先应该进行更新，当如果发起者地址是邻居集中包含的邻居元组的 N_neighbor_main_addr，则 N_willingness =来自 HELLO 消息的意愿。

在收到对称邻居的消息后，节点也应该更新 2 跳邻居集，N_neighbor_main_addr =发起者地址；N_2hop_addr = 2 跳邻居的主地址；N_time =当前时间+有效时间。

7. MPR 计算

7.1 MPR 选择步骤

基于上述过程建立的邻居信息表，节点可以选择出邻居 MPR 节点集合。一个节点选定的 MPR 是负责转发此节点的广播消息的节点，通过控制 MPR 集合的大小可以减少洪泛的开销。如下图所示情况为例，左侧为经典的洪泛技术，每个中心节点需要传播信息到其所有邻节点；右侧为引入 MPR 技术的 OLSR 协议，每个节点只需将信息传送到其 MPR 节点集合，并由它们广播消息。

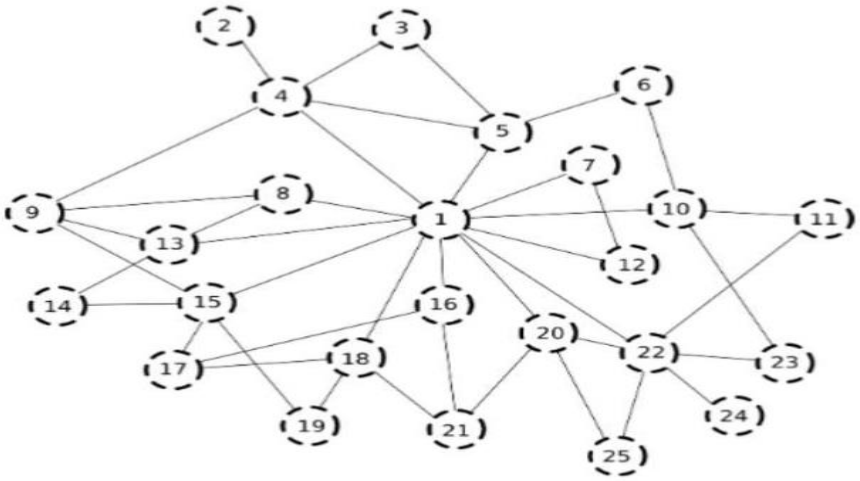


MPR 的选择主要分为两步：

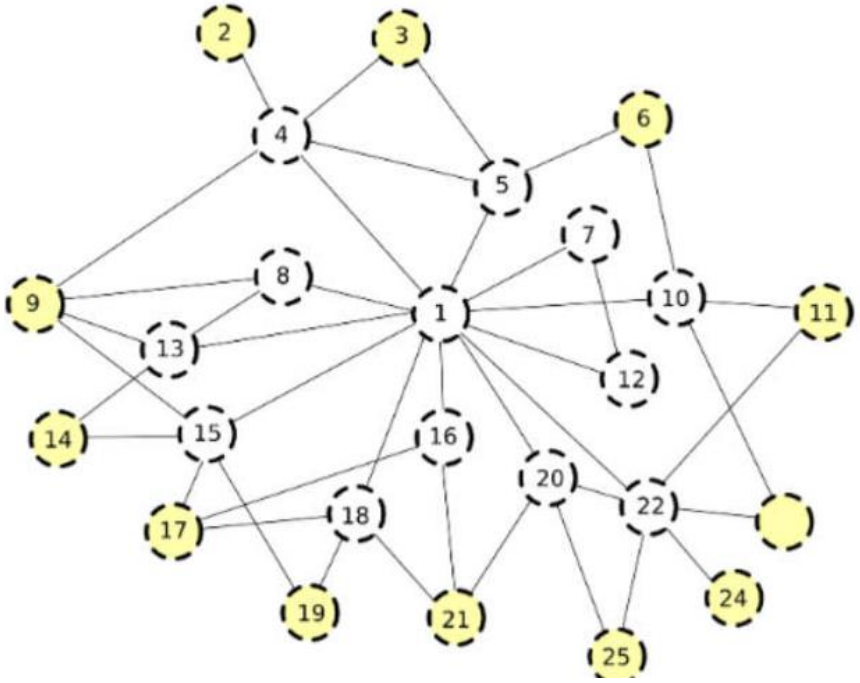
1. 首先选择能够覆盖孤立两跳邻节点的一跳邻节点。这里的孤立两跳邻节点是指仅通过一个邻节点同目标节点相连的两跳邻节点。

2. 在余下的一跳邻节点中，按照覆盖二跳邻节点的数量从高到低依次选择，直到覆盖所有的两跳邻节点。

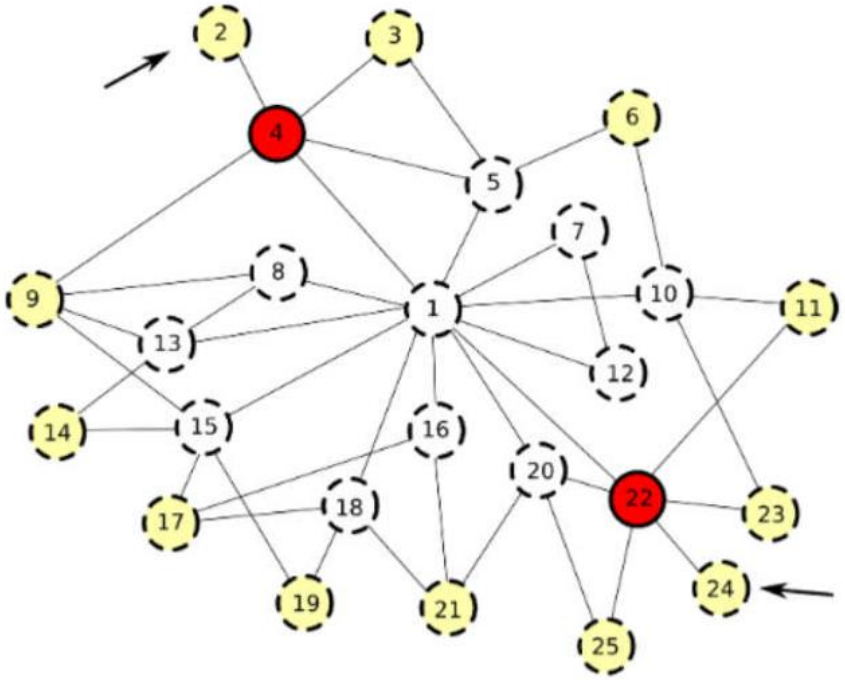
为了方便理解，以下图中的情况为例，找出“1”节点的最小 MPR 集合。



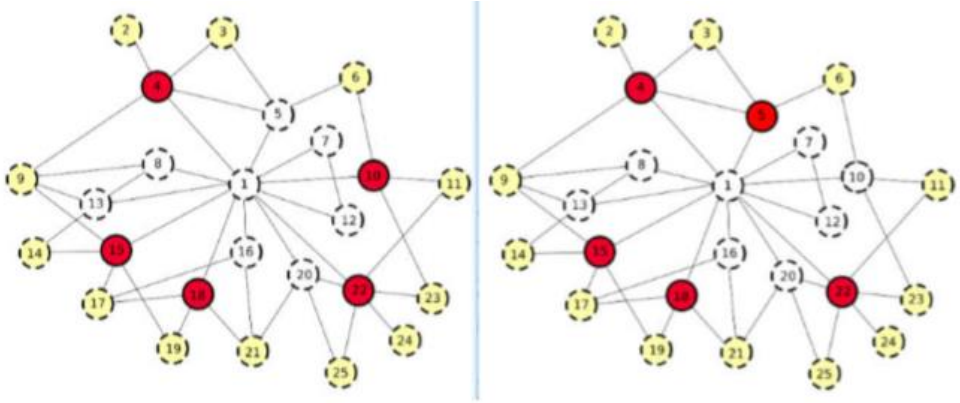
首先定义其所有的两跳邻节点（不包括一跳邻节点）。



然后找到其孤立两跳邻节点 2 和 24，然后将连接它们的一跳邻节点 4 和 22 加入 MPR 集合。



最后按照覆盖二跳邻节点的数量从高到低依次选择，其中 15 覆盖 4 个，18、5、10 覆盖三个，因为选择 5 和 10 都可以满足覆盖所有两跳邻节点，所以选择它们两个中任意一个即可。



在本地节点被选作 MPR 后，它会向其邻节点发送初始化为 MPR_NEIGH 的 HELLO 消息，邻节点收到消息后更新其 MPR selector set，此表表明节点应该转发来自哪些节点广播消息。

7.2 MPR 选择中的 Hello 消息处理

收到 HELLO 消息后，如果节点在列表中找到邻居类型等于 MPR_NEIGH 的自身接口地址之一，则必须将 HELLO 消息中的信息记录在 MPR 选择器集中。如果不存在发起者地址的 MPR 选择器，就新创建一个二元组：MS_main_addr =发起者地址，MS_time =当前时间+有效时间。

7.3 邻居与二跳邻居变化

链接元组的 L_SYM_time 字段到期。如果由过期元组描述的链接是与邻居节点的最后一条链接，则这被视为邻居丢失。

将新的链接元组插入 L_SYM_time 尚未到期的链接集中，或者将 L_SYM_time 到期的元组修改为未到期的 L_SYM_time。如果以前没有链接元组描述与相应邻居节点的链接，则这被视为邻居外观。

当检测到邻居或 2 跳邻居变化时，将发生以下处理：

如果邻居丢失，则必须删除所有 N_neighbor_main_addr ==邻居的主地址的 2 跳元组。

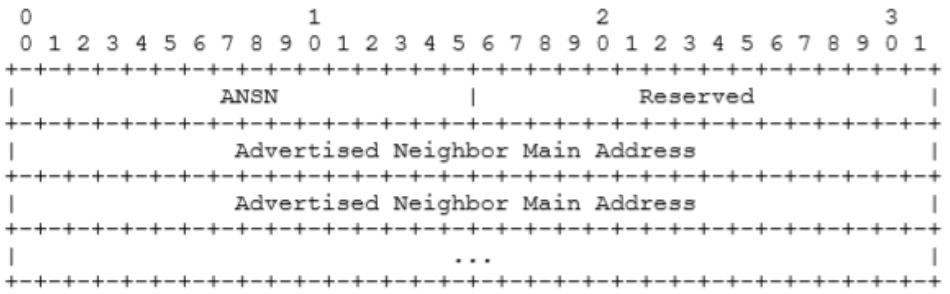
如果邻居丢失，则必须删除所有 MS_main_addr ==邻居主地址的 MPR 选择器元组

当检测到邻居出现或丢失或检测到两跳邻居变化时，必须重新计算 MPR 集。

当 MPR 集改变时，可能会发送一条附加的 HELLO 消息。

8. TC 消息

8.1 消息格式



和 hello 消息一样，都是作为通用消息格式的数据部分发送，并且“消息类型”设置为 TC_MESSAGE。应将生存时间设置为 255（最大值），以将消息传播到整个网络，将 Vtime 设置为 TOP_HOLD_TIME 的值。

ANSN(通告邻居序号)：序列号与通告的邻居集相关联。每当节点在其通告的邻居集中检测到更改时，它都会增加此序列号，该编号在 TC 消息的 ANSN 字段中发送，以跟踪最新信息。当节点收到 TC 消息，可以根据此序列号来确定消息是否最新。

预留：值为“ 0000000000000000”

通告邻居主地址：包含邻居节点的主地址，发起方将需要通告的邻居的所有主地址都放入 TC 消息中。

网络中的节点为了声明一组连接，向网络中的节点发送 TC 消息，称为通告连接集，包括该节点指向其 MPR 选择器集的所有节点的连接。当改变连接的时候，必须增加 ANSN 号。

基于 TC 消息的交换，各个节点可以维护一个 Topology Table

（拓扑表），拓扑表的结构如下：

Destination address	Destination's MPR	MPR SelectorSequence Number	Holding Time

Destination address 目标地址

Destination's MPR 目标地址的 MPR 节点

MPR SelectorSequence Number 序列号

Holding Time 该条目的保持时间

上面提到 TC 消息中包含的是发送节点的 MPR Selector 列表。那么当另一节点收到 TC 消息时，将 TC 条目中的 MPR Selector 作为目标地址，则发送节点即为其 MPR 节点，然后填入 TC 消息中的 Sequence Number，已经预定义的 Holding Time。

下面举一个例子说明拓扑管理的过程，这个例子中 A、B、C 三个节点均将 M 作为自己的 MPR 节点，那么 M 会建立如下的 MPR Selector 列表：

TC Originator	MPR Selector	MPR Selector Sequence
M	A	1
M	B	1
M	C	1

作为 MPR，M 会将其 MPR Selector 列表通过 TC 消息广播出去。当 Y 收到 M 发出的 TC 消息时，将 TC 消息中包含的 MPR Selector 信息转化成拓扑表 (Holding Time 省略)：

Destination address	Destination's MPR	MPR SelectorSequence Number
A	M	1
B	M	1
C	M	1
…	…	…

网络中周期性的通过 TC 消息保持拓扑表更新，通过拓扑表可以计算出全局路由表。

8.2 TC 消息处理

收到 TC 消息之后，从 Vtime 计算出有效时间，然后按照以下方式更新拓扑集：

1.如果发送者接口不是该节点的对称 1 跳节点，就丢弃。

2.存在 T_last_addr ==发起者地址，T_seq> ANSN，则不进行任何的操作。

3.若 T_last_addr ==发起者地址，T_seq< ANSN，则删除拓扑集。

4.若 T_dest_addr ==通告的邻居主地址，T_last_addr ==发起者地址，则修改保持时间为 T_time =当前时间+有效时间。

5.若找不到这些元组，就新创建一个元组，令 T_dest_addr =通告的邻居主地址，T_last_addr =发起者地址，T_seq = ANSN，T_time =当前时间+有效时间。

9. 路由表的计算

每个节点都维护一个路由表，该路由表允许其路由发往网络中其他节点的数据。路由表基于本地链接信息库和拓扑集中包含的信息。如果更改了这些集合中的任何一个，则将重新计算路由表以更新有关网络中每个目标的路由信息。路由表以以下格式记录下来：

```
1.  R_dest_addr    R_next_addr    R_dist   R_iface_addr
2.  R_dest_addr    R_next_addr    R_dist   R_iface_addr
3.      ,,             ,,           ,,         ,,
```

R_dest_addr：为目的节点地址

R_next_addr:为到达目的地址的所要经过的下一跳对称邻居节点地址

R_dist:目的节点距离本地节点的跳数

R_iface_addr：到达目的地址的本地接口地址

监测到连接集、邻居集、2 跳邻居集、拓扑集、多接口关联信息库改变的时候，将更新路由表。

按照以下例子去计算路由表：

1. 删除所有的条目
2. 从对称邻居开始添加新的条目，对于邻居集中的每个邻居元组，N_status =SYM，并且对于邻居节点的每个关联链接元组（使 L_time> =当前时间),新的路由条目记录在路由表中:R_dest_addr =L_neighbor_iface_add;R_next_addr = L_neighbor_iface_addr;R_dist=1;R_iface_addr=关联的链接元组的 L_local_iface_addr。
3. 如果 2 中没有，邻居的主地址，则新加一个条目：R_dest_addr =

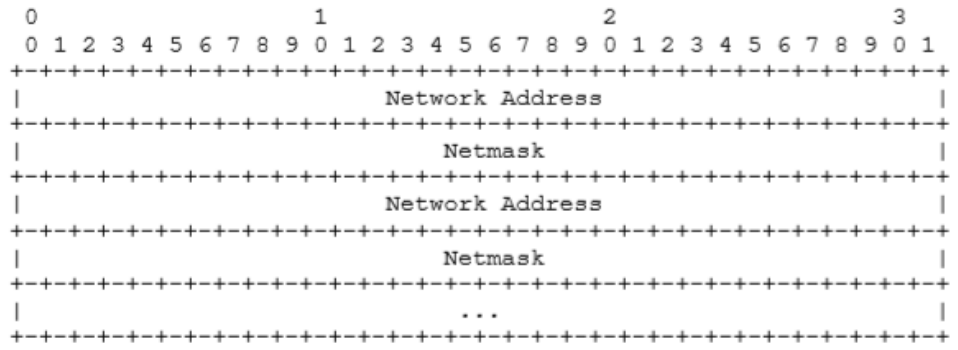
邻居的主地址；R_next_addr = L_neighbor_iface_addr，其中L_time>当前时间R_dist=1;R_iface_addr = L_local_iface_addr。

4. 对于 2 跳邻居集的节点：R_dest_addr = 2 跳邻居的主地址，R_next_addr =路由表中条目的 R_next_addr，R_dist = 2；R_iface_addr =路由表中条目的R_iface_addr，
5. 对于拓扑表中的条目，如果其 T_dest_addr 与路由表中任何路由条目的R_dest_addr不对应，并且其T_last_addr与R_dist等于 h 的路由条目的 R_dest_addr 相对应，则：R_dest_addr = T_dest_addr; R_next_addr =记录的路由条目的R_next_addr，R_dest_addr == T_last_addr，R_dist = h + 1;R_iface_addr = 记录的路由条目的R_iface_addr。
6. 多接口信息库中的R_dest_addr == I_main_addr没有路由条目，则添加：R_dest_addr = I_iface_addr，R_next_addr = R_next_addr（记录的路由条目），R_dist = R_dist（记录的路由条目），R_iface_addr = R_iface_addr（记录的路由条目）

10. HNA 消息

10.1 HNA 消息格式

非OLSR接口会定期发布主机和网络关联（HNA）消息，将外部路由信息注入MANET的功能，已便构造合适的路由表，格式如下



这是作为一般数据包格式的数据部分发送的，其中“消息类型”设置为 HNA_MESSAGE，TTL 字段设置为 255，Vtime 相应地设置为 HNA_HOLD_TIME 的值，网络地址和网络掩码都是对应的关联网络的地址和掩码。

每个节点通过记录“关联元组”（A_gateway_addr，A_network_addr，A_netmask，A_time）来维护有关哪些节点可以充当关联主机和网络的“网关”的信息，A_gateway_addr 是 OLSR 接口的地址网关，A_network_addr 和 A_netmask 指定可通过此网关访问的网络的网络地址和网络掩码，并且 A_time 指定该元组到期的时间，节点中所有关联元组的集合称为“关联集”。

10.3 HNA 消息处理

接收到 HNA 消息之后，按照以下规则更新关联集：

1. 如果该消息的发送者接口，不在该节点的对称 1-跳附近，则必须丢弃该消息。
2. 存在关联集中的条目，则只修改保持时间，令 A_time =当前时间+有效时间

3. 如果不存在，则记录一个新的元组，A_gateway_addr =发起者地址，A_network_addr =网络地址，A_netmask =网络掩码，A_time =当前时间+有效时间

同理，如果没有路由表，则创建一个新的路由表：

R_dest_addr == A_network_addr / A_netmask，R_dist =到A_gateway_addr 的距离，将 R_next_addr 和 R_iface_addr 设置为与使用 R_dest_addr == A_gateway_addr 的路由集中的元组相同的值。

11. 链路层丢失

本地链接集中的每个链接元组都应包含 L_LOST_LINK_time 字段，为一个计时器。在接收到节点和邻居接口之间的链路断开的链路层通知后，将对链路感知在 hello 消息中采取以下操作：

1. 如果 L_LOST_LINK_time 没有过期，则为 1，该链接以 LOST_LINK 的链接类型发布。另外，在相关邻居元组的更新中，它不被视为对称链接
2. 如果到相邻对称或非对称接口的链接断开，则修改相应的链接元组：L_LOST_LINK_time 和 L_time 设置为当前时间 + NEIGHB_HOLD_TIME。
3. 视为链路丢失，采用邻居丢失进行处理

12. 链路滞后

除了之前说的一些字段，本地链接集中的每个链接元组还应包括

L_link_pending 字段,L_link_quality 字段和 L_LOST_LINK_time 字段。L_link_pending 是一个布尔值，用于指定是否将链接视为待处理（即，不认为该链接已建立）。L_link_quality 是介于 0 和 1 之间的数字，描述了链接的质量。L_LOST_LINK_time 是一个计时器，用于在建立的链接待处理时将链接声明为丢失。

12.1 链路滞后的 Hello 消息的生成

1.如果 L_LOST_LINK_time 没有过期,则为 1,该链接以 LOST_LINK 的链接类型发布。

2.如果 L_LOST_LINK_time 过期并且 L_link_pending 设置为“ true”，则该链接完全不应被发布;

3.如果 L_LOST_LINK_time 过期并且 L_link_pending 设置为“ false”，则按照前面的描述发布链接。

4. 节点认为每个链接元组都有一个对称链接，L_LOST_LINK_time 已过期，L_link_pending 为“ false”， L_SYM_time 没有过期。