目　录

基于链路质量的 OLSR 自组网路由改进方案

摘　要

无线自组织网（Ad Hoc）是一种不局限于特定的基础设备，可以随时随地快速构建的多跳临时性无中心网络，相较于传统的无线网络，无线自组织网的每一个节点均具备路由功能，而不依赖于传统无线网中的专用路由设备。对于无线自组织网络，路由协议决定了无线自组织网络邻居发现、拓扑建立的速度以及控制数据包广播和业务数据包转发的效率。

目前常见的路由协议主要分为按需路由（AODV，DSR），先验式路由（OLSR，DSDV），混合式路由（TORA，ZRP）。由于目前主流的无线自组织网络的路由协议比较成熟，所以本文主要是在简易实现 OLSR 的基础上，针对 OLSR 的 MPR 计算和 Willingness 更新策略进行改进实现。原始 OLSR 的 MPR 计算存在冗余，并且未考虑 Willingness 的初始值以及更新策略，本文将通过改变贪心策略尝试减少 MPR 计算产生的冗余，在此基础上，通过计算节点的传输延时、邻居距离、传输信噪比等表示链路质量的特性动态更新节点的 Willingness，使得无线自组织网络可以考虑物理层特性来构建网络拓扑和路由。

OPNET 是通信研究领域使用较为广泛的网络仿真软件，本文将使用 OPNET 来模拟现实使用环境，测试改进的效果。

关键词：无线自组网路由协议；OLSR 协议；链路质量

ABSTRACT

Ad Hoc is a kind of multi-hop temporary non-centralized network that is not limited to specific basic equipment and can be quickly constructed anytime and anywhere. Compared with the traditional wireless network, each node of the wireless ad hoc network has routing function, and it does not rely on dedicated routing equipment in traditional wireless networks. For wireless ad hoc networks, the routing protocol determines the speed of wireless ad hoc network neighbor discovery, topology establishment, and the efficiency of controlling data packet broadcasting and service data packet forwarding.

At present, common routing protocols are mainly divided into reactive routings (AODV, DSR), proactive routings (OLSR, DSDV), and hybrid routing (TORA, ZRP). As the current mainstream wireless ad hoc network routing protocol is relatively sophisticated, this article is mainly based on the simple implementation of OLSR, and improves OLSR's MPR calculation and Willingness update strategy. The MPR calculation of the original OLSR is redundant, and does not consider the initial value of Willingness and its update strategy. This article will try to change the greedy strategy to reduce the redundancy generated by the MPR calculation. On this basis of that, the Willingness of the node will be dynamically updated by calculating the transmission delay of the node, the neighbor distance, transmission signal-to-noise ratio and other characteristics which indicate link quality, so that the wireless ad hoc network can consider the physical layer characteristics to construct the network topology and routing.

OPNET is a widely used network simulation software in the field of communication research. This article will use OPNET to simulate the actual use environment and test the effect of improvement.

Keywords: Ad Hoc routing protocols; OLSR protocol; Quality of Link;

第1章 绪　论

1.1 选题背景和研究意义

伴随着通信技术和计算机科学技术的进步，无线通信技术快速增长，并在各行各业中得到了广泛的使用。传统的移动通信网络例如手机蜂窝移动网络，家用无线局域网络等一般具有中心节点，例如 AP，移动信号基站等设施。中心节点具有路由功能，其他节点通过中心节点转发业务数据，除中心节点之外的节点一般不能直接痛惜。但是有中心网络在战场，城市消防，灾后救援等场景下不能很好的发挥作用，特别是当中心节点瘫痪之后，整个网络将无法工作。

无线自组织网络（以下简称无线自组网）不局限于特定的基础设备，不依赖中心节点，可以随时随地快速构建，可以很好的应对传统网络在上述场景下构建困难的问题。

无线自组网起源自分组无线网(Packet Radio Network)，其因为军事通信的特殊需要而产生，传统的中心无线网络在战场环境下实施可能性极低，这种特殊场景带来了分组无线网并推动了其发展。1991 年，IEEE802.11 标准委员会采用“Ad Hoc 网络”描述这种特殊的对等式无线移动网络。

无线自组网目前在军事通信领域、传感器网络、灾后紧急应用网络、个人通信方面有广泛的应用。其快速构建、依赖性低的特性使得无线自组网拥有较好的发展前景和应用前景。

在无线自组网技术中，路由技术是其中的一项关键技术[1]，高效的路由算法可以提高邻居发现、拓扑建立的速度，提高控制数据包转发和业务数据包传送的效率。优化改进路由算法对提升无线自组网的性能有着重大意义。

本文参考了目前较为成熟且应用较为广泛的 OLSR 路由协议，在其基础上做了和物理层结合的尝试，并根据设计原理在 OPNET 平台上实现并仿真。

1.2 研究现状

无线自组网自诞生到现在，已经发展出了多种有效的路由协议[2]。

目前常见的对无线自组网的路由主要有三种分类[2]：按需路由（AODV[3]，

DSR[4]），先验式路由（OLSR[5]，DSDV[6]），混合式路由（TORA[7]，ZRP[8]）。

按需路由在网络构建和网络维持阶段并不计算并保存全部的瞬时全局路由信息，相反，只有当节点产生了发送业务数据包的需求时，节点才会发送例如 RREQ 等控制数据包，探测网络的拓扑和路由情况。这种路由协议的实现简单，应用广泛，但是由于节点本身并不保存路由信息，所以从节点计划开始发送数据包到实际发送数据包存在较大的时延，在军事、灾后救援等需要快速相应的场景不能快速传递消息，并且网络中不可控数量的控制数据包可能会带来较大的网络拥塞。

先验式路由将网络路由的建议作为周期性的任务，以 OLSR 协议为例，节点周期性的发送 HELLO、TC 等控制数据包，节点发送控制数据包与否与节点的业务数据包需求没有关系。当节点接收到来自其他节点的控制数据包，周期性的建立网络的整体拓扑关系并计算出全网路由，所有节点在网络生存周期内维护正确的路由信息。当节点需要发送业务数据包时，可以根据节点记录的路由表快速响应，极大的减少了发送延迟。但是先验式路由每个节点需要根据控制数据，周期性计算路由，节点的计算开销增加，例如 OLSR 协议的 MPR 计算、拓扑计算、路由计算均比较复杂且耗时，在应用于低功耗设备时，需要修改计算策略以减少平均功耗，延长节点生命周期。

混合式路由在网络组建初期采用先验式路由的思路，在网络维持阶段采用先验式路由的路由表和按需产生的路由发送数据包，减少了两者缺陷造成的影响。但不同的混合式路由仍有不同的妥协，例如 ZRP 的域生成可能会产生域的相互覆盖，产生冗余的节点归属等问题。

OLSR 目前的研究方向主要在 MPR 集最小化、OLSR 协议安全、OLSR 能量意识问题。MPR 的选择算法在原始协议中是一个冗余的“局部最优”解，而 MPR 的全局最优解是一个 NP 问题[9]，所以本文尝试了消除冗余，并在此基础上加入其他层例如物理层的特性。

1.3 论文组织结构

本文将简单介绍无线自组网和路由协议的背景知识，详细介绍 OLSR 的协议内容，例如数据结构和具体算法，在 OLSR 协议的基础上，本文将说明 OLSR 的

部分缺陷，以及设计的改进方案。最后，本文将对基于 OPNET 网络仿真软件得到的仿真结果进行分析，评估设计方案的效果。

本文第 1 章介绍了论文的研究背景和意义、常见的路由协议及其研究现状，提出了本文研究的方向和大致方法。第 2 章介绍了根据 OLSR 原始协议[5]简化之后的协议概念、数据结构、算法等，以及本文为了验证使用的网络仿真软件 OPENT。第 3 章分析了 OLSR 的 MPR 机制的不足，并针对其中的几个方面提出了具体的改进方案。第 4 章使用 OPNET 网络仿真软件建立 OLSR 协议仿真模型，进行网络仿真并分析仿真结果，评估改进方案的实际可行性。第 5 章总结本文的工作，并分析不足之处，展望后续工作。

第2章 OLSR 协议

2.1 OLSR 协议基本介绍

OLSR 协议[5]是表驱动的先验式路由协议，其核心思想是多点中继机制（Multi-Point Relay，以下简称 MPR）。

MPR 机制体现在对邻居的划分上，一个节点的有效一跳邻居可以划分为两类，MPR 节点和非 MPR 节点。当节点广播消息时，只有本节点选择的 MPR 节点才能转发该消息，非 MPR 节点只接收和处理消息。节点在广播用于建立拓扑的链路信息时，也只广播和 MPR 节点之间的链路关系，而不是广播所有可能的链路消息。

使用 MPR 机制可以带来以下优化：

1. MPR 机制可以有效抑制广播的范围。

   相较于泛洪机制，MPR 机制仅通过节点选取的 MPR 节点进行数据包的转发，就能覆盖所有的二跳邻居节点，极大的减少了数据包重复转发的数量。

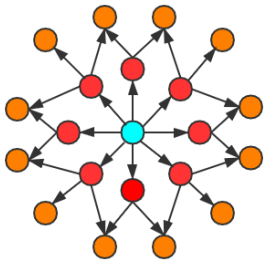

图 2-1 泛洪机制

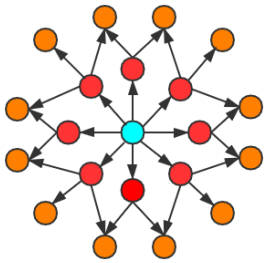

图 2-2 MPR 机制

   图 2-1 和图 2-2 展示了泛洪机制和 MPR 机制的区别，传统的泛洪会导致外围节点可能收到多个来自原始节点的广播数据，而 MPR 机制通过选取最少的能够覆盖网络的节点，然后仅使用 MPR 节点转发，降低了外围节点收到重复数据包的可能性，减少了网络开销，避免了广播风暴[10]可能导致的网络拥堵和数据包失效。

2. 减少控制数据的大小。

   当节点选取了 MPR 节点后，只需要根据 MPR 节点建立拓扑，在网络

中广播的拓扑消息将减少，承载拓扑的控制数据包也会随之减少，可以降低网络用于控制数据包的时间和资源开销，更好的服务业务数据包。

2.2 OLSR 的数据结构

本文设计的数据结构源自于论文[5]，根据具体实现对部分数据结构进行了拆分和简化。

2.2.1 链路码（LinkCode）

链路码是 HELLO 消息中表示本节点和记录的邻居节点之间链路状态的标识码，分为以下三种：

AYSM_LINK：表示对应的链路是单向可达链路（在当前时间，仅有一方可以确认该链路有效）。

SYM_LINK：表示对应的链路是双向可达链路或者堆成链路（在当前时间，双方均能确认这条链路有效）

LOST_LINK：表示对应的链路已经失效（在当前时间，链路无效，主要是反馈无效的链路以更新链路表）。

2.2.2 邻居码（NeighCode）

邻居码是 HELLO 消息中表示记录的邻居状态的标识码，分为以下三种：

NOT_NEIGH：表示对应的节点已经不属于发送节点的有效邻居。

SYM_NEIGH：表示对应的节点是发送节点的对称邻居，对应链路码中的 SYM_LINK。

MPR_NEIGH：表示对应的节点属于发送节点的 MPR 生成集。

2.2.3 邻居状态码（NeighStatus）

邻居状态码表示记录在一跳邻居表中的邻居的状态。

NOT_SYM：表示对应的邻居表中表项的邻居不是对称邻居。

SYM：表示对应的邻居表中表项的邻居是兑成邻居（包含 SYM 和 MPR 类型）。

2.2.4 消息类型码（messageType）

消息类型主要分为业务消息和控制消息，业务消息并没有特殊的格式，只要符合消息包的格式即可，控制消息主要分为两类，负责链路侦听和邻居发现的HELLO 消息、负责广播全局拓扑的 TC 消息。消息类型码表示不同的控制消息。

HELLO：HELLO 消息。

TC：TC 消息。

2.2.5 链路类型消息（link_status）

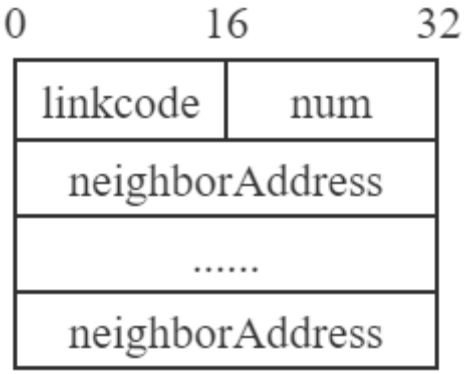

图 2-3 链路类型消息格式

链路类型消息包含了本地记录的发送节点周围的链路状态。图 2-3 是链路类型消息的格式。

linkcode 表示下面的 neighborAddress 和发送节点之间的链接均是 linkcode 对应的链路码。num 是 neighborAddress 的数量，为了方便发送前组包和接收后解包。

2.2.6 邻居类型消息（neigh_status）

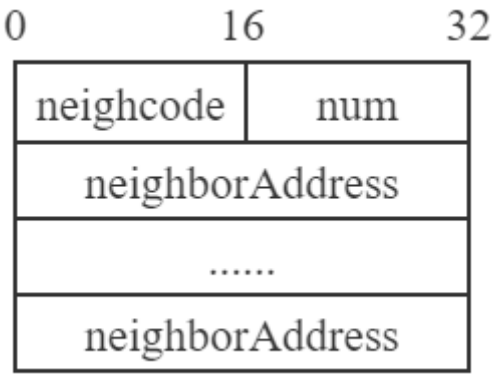

图 2-4 邻居类型消息格式

邻居类型消息包含了发送节点周围的邻居关系。图 2-4 是邻居类型消息的格式。

neighcode 表示下面的 neighborAddress 是均是发送节点的邻居，对应的邻居码是 neighcode。num 是 neighborAddress 的数量，作用同 2.2.5 中的 num。

2.2.7 HELLO 消息（message_hello）

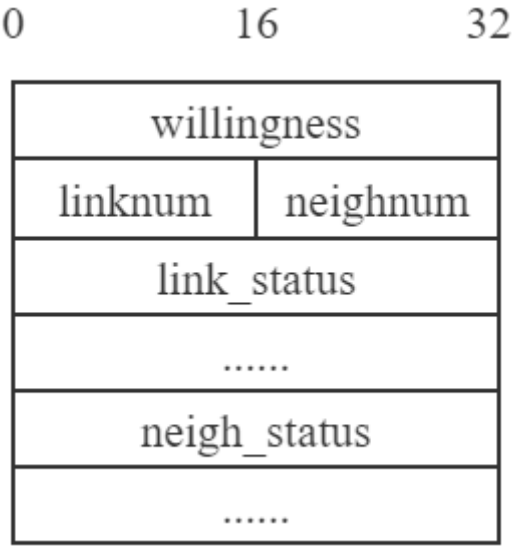

图 2-5 HELLO 消息格式

HELLO 消息携带了发送节点的链路信息和邻居关系，用于和接收者交换信息，实现链路侦听和邻居发现。图 2-5 是 HELLO 消息的格式。

willingness 是节点转发接受自其他网络节点的数据包的意愿程度，本文将其划分为七个等级：WILL_NEVER，WILL_LOWER，WILL_LOW，WILL_DEFAULT，WILL_HIGH， WILL_HIGHER， WILL_ALWAYS 。每个节点的初始值为 WILL_DEFAULT，节点的 willingness 将根据网络的变化而动态变化，其变化算法将在第 3 章中描述。

linknum 和 neighnum 分别表示 HELLO 消息携带的链路类型和邻居类型消息的数量，方便发送者组包和接收者解包。

link_status 和 neigh_status 是携带的链路类型消息和邻居类型消息。

2.2.8 TC 消息（message_tc）

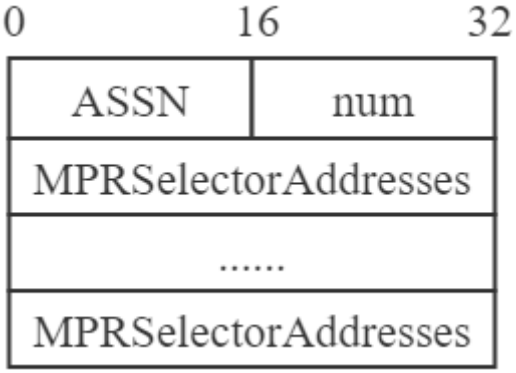

图 2-6 TC 消息格式

TC 消息携带了发送节点的 MPR 选择集包含的节点地址，本质上是向全网广播发送节点周围的拓扑信息（该拓扑信息是根据 MPR 机制筛选出的拓扑信息）。图 2-6 是 TC 消息的格式。

ASSN 是发送节点维护的标识 TC 消息发送序号的计数值，用来区分新旧 TC 消息，因为 TC 消息需要无线自组网中的所有节点接收，所以一个节点可能会乱序接收来自其他节点的多个 TC 消息，使用 ASSN 值可以丢弃掉过时的 TC 消息。

num 表示 TC 携带的 MPR 选择集的节点地址数量，作用同 2.2.7 中的 linknum 和 neighnum 一致。

MPRSelectorAddresses 是 TC 消息携带的 MPR 选择集的节点地址，当节点在网络初期尚未形成 MPR 选择集时，节点不允许广播 TC 消息。

2.2.9 消息包（message_packet）

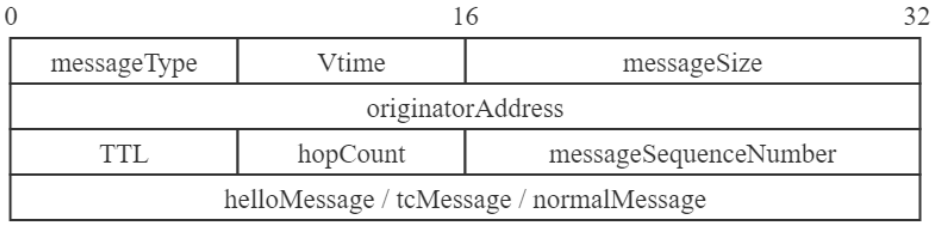

图 2-7 消息包格式

消息包将控制消息（helloMessage、tcMessage）和业务消息（normalMessge）的外层格式统一，每个消息包仅能包含一种类型的一条消息。图 2-7 是消息包的格式。

messageType 表示本消息包的消息种类，包含消息类型码（messageType）和

其他自定义的业务消息。

Vtime 表示本消息包的有效时间，对于 HELLO 消息，这个值用于计算邻居维持的时间，对于 TC 消息，这个值用于计算拓扑表维持的时间。

messageSize 表示本消息包的大小，在本文中以 32 位字为单位。

originatorAddress 表示产生本消息包的原始节点地址，不同于 IP 包中的源地址。在网络广播中，中间节点根据转发规则转发数据包时，originatorAddress 不发生变化，而源地址遵循 IP 协议，会更改为中间执行转发行为的节点的地址，两者在 OLSR 协议中的作用是不一致的。

TTL 表示本消息包的生存周期，在 OLSR 中是消息包被发送的最大期望跳数。HELLO 消息的 TTL 初始值为 1，因为 HELLO 消息不进行转发也不存储，仅用于链路侦听和邻居发现，传送距离限制在 1 跳以内。TC 消息的 TTL 初始值为尽可能大的值，一般为 255（8bit 表示的最大无符号整数），其目的是为了使网络中的所有节点都能接收到 TC 消息。

hopCount 表示本消息包已经被传送的跳数，初始值应为 0，每被转发一次，hopCount 加一，对应的，TTL 减一。

messageSequenceNumber 表示本消息包的序号，每一个节点会维持一个消息包的计数值，用于唯一标识每个节点的发送消息，接收节点可根据原始节点地址和消息包序号确认一个唯一的消息包。

helloMessage / tcMessage / normalMessage 表示本消息包携带消息，根据消息类型选择对应的消息放入该字段，但是仅能放入其中一种消息且只能放一个。

2.2.10 OLSR 数据包（OLSR_packet）

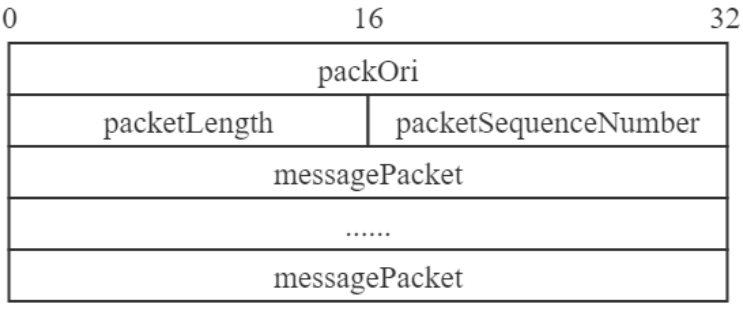

图 2-8 OLSR 数据包格式

OLSR 数据包是发送节点最终组成的数据包，嵌入采用的网络层协议数据包的数据部分，本文仅对 OLSR 协议进行验证，所以实际并不根据网络协议栈实现完整的网络收发系统。图 2-8 是 OLSR 数据包的格式。

packOri 表示 OLSR 数据包的发送节点地址，其概念不同于 2.2.9 中的 originatorAddress，消息包需要重传，其原始节点地址不能发生变化，当发送节点决定重传来自其他节点的消息包时，会将消息包放入 OLSR 数据包，packOri 始终为发送节点的地址，在重传中会更改为重传节点的地址，类似于 IP 协议中的发送地址。因为控制数据包均要进行广播，所以随着发送节点地址变化的 packOri 可以帮助节点筛除自身发送的数据包。

packetLength 表示本数据包的长度，为了方便实现，本文中以 32 位字为单位。

packetSequenceNumber 表示本数据包的序号，每一个节点会维持一个数据包的发送序号，本文中定义的目的是为了仿真结束后收集统计信息。

messagePacket 表示本数据包中携带的消息包，不论消息包的消息类型如何，在数据包的视角，消息包都是一致的。

2.2.11 链路表（local_link）

L_neighbor_addr	L_SYM_time	L_ASYM_time	L_time

图 2-9 链路表项

链路表记录了节点侦到的所有链路信息。图 2-9 包含了一个链路表项的所有内容。

L_neighbor_addr 表示该链路表项记录下的邻居地址。

L_SYM_time 表示该链路被认为是对称链路的有效截止时刻，当 L_SYM_time 小于当前时刻，则认为该链路不是对称的。

L_ASYM_time 表示该链路被认为是单向链路的有效截止时刻，当 L_ASYM_time 小于当前时刻，则认为该链路已经丢失，否则是单向或者对称的。

L_time 表示该链路表项的期望保持时间，当 L_time 小于当前时刻，认为该链路表项已经过期，需要被删除，同时一跳邻居表中对应的项也需要被删除。

2.2.12 一跳邻居表（one_hop_neighbor）

N_neighbor_addr	N_status	N_willingness
N_2hop		
.....		
N_2hop		

图 2-10 一跳邻居表项

一跳邻居表记录了节点周围传输距离为一跳的邻居，并记录其状态和对应的两条邻居。图 2-10 包含了一个一跳邻居表项的所有内容。

N_neighbor_addr 表示该一跳邻居表项的邻居地址。

N_status 表示邻居的状态，状态为邻居状态码（NeighStatus）。

N_willingess 表示邻居转发其他节点消息的意愿程度，对应 HELLO 消息（message_hello）的 willingness。

N_2hop 表示一跳邻居对应的二跳邻居，和原始协议不同的是，这里将二跳邻居表作为一跳邻居表的子项，和原始协议中分表记录相比，子项更方便建立一跳二跳邻居的映射关系。

2.2.13 二跳邻居表（two_hop_neighbor）

N_2hop_addr	N_time

图 2-11 二跳邻居表项

二跳邻居是根据一跳邻居可以到达的对称节点，即一跳邻居有通向二跳邻居的链路，反向链路也必须存在，或者说二跳邻居是一跳邻居的 MPR_NEIGH 或者 SYM_NEIGH。图 2-11 包含了构成一个二跳邻居表项的所有数据。

N_2hop_addr 表示该二跳邻居表项的二跳邻居地址。

N_time 表示该表项的维持时间，当 N_time 小于当前时刻，该二跳邻居表项被认为失效，需要被删除。

2.2.14 MPR 选择集（MPR）

MS_addr	MS_time

图 2-12 MPR 选择集项

MPR 选择集记录的是当前节点被哪些节点选为 MPR 节点，不同于 MPR 生成集的概念，MPR 生成集是当前节点根据 MPR 生成算法计算出 MPR 节点，而 MPR 选择集是根据接收到的 HELLO 消息中邻居码为 MPR_NEIGH 的节点记录地址产生的。图 2-12 包含了构成一个 MPR 选择集项的所有数据。

MS_addr 表示有哪些节点的 MPR 生成集包含本节点。

MS_time 表示该 MPR 选择集项的维持时间，当 MS_time 小于当前时刻时，该项被认为失效，需要被删除。

2.2.15 重复记录表（duplicate_set）

D_addr	D_seq_num	D_time	D_retransmitted

图 2-13 重复记录表项

重复记录表记录了除 HELLO 消息以外的重复广播消息，其目的是为了防止一个消息被多次处理和转发（HELLO 的传播仅在一跳范围内，也不进行转发，所以不会出现某个节点重复接收到同一个 HELLO 消息）。图 2-13 包含了构成一个重复记录表项的所有数据。

D_addr 表示该项消息的原始地址。

D_seq_num 表示该项消息的消息包序号。

D_time 表示该项的维持时间，当 D_time 小于当前时刻时，该项被认为失效，需要被删除。

D_retransmitted 表示该项是否被重传，仅当该项为 false 时，该项对应的消息才需要重传。

2.2.16 拓扑表（topology_item）

T_dest_addr	T_last_addr	T_seq	T_time

图 2-14 拓扑表项

拓扑表根据接收到的 TC 消息经过拓扑计算算法产生，是根据 MPR 机制简化的全网拓扑。图 2-14 包含了构成一个拓扑表项的所有数据。

T_dest_addr 表示拓扑的目的节点。

T_last_addr 表示被 T_dest_addr 选中的 MPR 节点，当消息需要达到 T_dest_addr，需要通过 T_last_addr 的转发。

T_seq 表示记录的 TC 消息的序号，当记录的 T_seq 小于接收的序号，则认为该拓扑项无效。

T_time 表示该项的维持时间，当 T_time 小于当前时刻时，该项被认为是失效，需要被删除。

2.2.17 路由表（route_item）

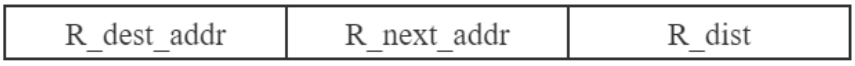

图 2-15 路由表项

路由表是根据拓扑表计算出的路由信息，由于本文中的改进不涉及路由部分，所以在 OLSR 算法中并不介绍路由算法。图 2-15 包含了构成一个路由表项的所有数据。

R_dest_addr 表示路由的目的节点。

R_next_addr 表示消息需要被发送的下一跳的地址。

R_dist 表示当前节点路由距离目的节点的跳数。

2.3 OLSR 算法

2.3.1 HELLO 消息生成

HELLO 消息需要携带链路信息和邻居信息。

1. 链路信息

   对于链路表中的每一项，若 L_SYM_time 大于等于当前的时刻，则链路码为 SYM_LINK，若 L_SYM_time 小于当前的时刻且 L_ASYM_time 大于等于当前的时刻，则链路类型为 ASYM_LINK，若前两者的预期存在时间均小于当前时刻，则链路类型为 LOST_LINK。

2. 邻居信息

   对于链路表中的每一项，若 L_neighbor_address 在 MPR 生成集中，则邻居码为 MPR_NEIGH，若在一跳邻居表中且 N_status 为 SYM，则邻

居码为 SYM_NEIGH，否则邻居码为 NOT_NEIGH。

willingness 设置为 WILL_DEFAULT。

将 HELLO 消息封装在消息包中，messageType 为 HELLO，vTime 为邻居保持时间（NEIGHB_HOLD_TIME），TTL 为 1，hopCount 为 0，originatorAddress 为当前节点地址，messageSequenceNumber 根据节点记录的计数值设置。

2.3.2 HELLO 消息处理

HELLO 消息的功能是链路侦听和邻居发现。节点按照预设的时间间隔周期性的广播 HELLO 消息，邻居节点接收到消息后，更新自己的链路表，一跳邻居表和二跳邻居表，并将链路表和一跳邻居表的信息再次广播出去，节点根据接收到的消息确定邻居。

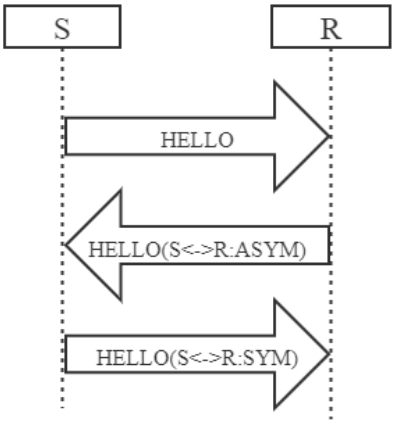

图 2-16 链路侦听 邻居发现

图 2-16 是一个 HELLO 消息的例子，节点 S 首先广播 HELLO 信息，此时 S、R 均不知道双方的链接情况，节点 S 广播的 HELLO 信息没有附带任何信息。当节点 R 接收到 S 广播的 HELLO 之后，R 认为可以接收到来自 S 的消息，但是并没有之前的记录，所以 R 认为 SR 之间的链路是单向的，当 R 广播 HELLO 时，会携带 R 记录的 SR 之间的链路状态（SR 存在单向链路）。当 S 再次收到来自 R 的 HELLO 消息，从 R 携带的 SR 链路状态可以知道，SR 存在单向链路，此时 S 又可以接收来自 R 的消息，则 SR 之间的链路状态是对称的，所以 S 确认了 SR 之间的对称链路。当 S 再次广播 HELLO 是，会携带 S 记录的 SR 之间的链路状态（SR 之间存在对称链路）。当 R 再次收到 S 的广播

HELLO 消息，就能都确认 SR 之间的对称链路，并记录在 R 的记录表中。

当一个节点接收到 HELLO 消息，需要按照算法 2-1，算法 2-2，算法 2-3，算法 2-4，更新节点的链路表，一跳邻居表，两跳邻居表，MPR 选择集。

算法 2-1 链路表更新

链路表更新

```
INPUT：HELLO 消息包；本地链路表
OUTPUT：none
1:  if HELLO 消息包的 messageType != HELLO then
2:      return
3:  end if
4:  if HELLO 消息包的 originatorAddress not in 本地链路表 then
5:      new 本地链路表项，新表项的值如下;
6:      L_neighbor_addr = HELLO 消息包的 originatorAddress;
7:      L_SYM_time = 当前时刻 – 1;
8:      L_ASYM_time = 当前时刻 + HELLO 消息包的 vTime;
9:      L_time = 当前时刻 + HELLO 消息包的 vTime;
10: else
11:     修改存在的表项如下;
12:     L_ASYM_time = 当前时刻 + HELLO 消息包的 vTime;
13:     if 节点地址 in HELLO 消息包的链路类型消息 then
14:         if 链路类型 == LOST_LINK then
15:             L_SYM_time = 当前时刻 – 1;
16:         else if 链路类型 == SYM_LINK or 链路类型 == ASYM_LINK then
17:             L_SYM_time = 当前时刻 + vTime;
18:             L_time = L_SYM_time + NEIGHB_HOLD_TIME;
19:         end if
20:     end if
21:     L_time = max(L_time, L_ASYM_time);
22: end if
23: return
```

算法 2-2 一跳邻居表更新

一跳邻居表更新

```
INPUT：HELLO 消息包；一跳邻居表；本地链路表
OUTPUT：none
1:  if HELLO 消息包的 originatorAddress not in 一跳邻居表 then
2:      new 一跳邻居表项，新表项的值如下;
3:      N_neighbor_addr = HELLO 消息包的 originatorAddress;
4:      N_willingness = HELLO 消息包的 willingness;
5:      if 本地链路表中对应地址的 L_SYM_time >= 当前时间 then
6:          N_status = SYM;
```

```
7:      else
8:          N_status = NOT_SYM;
9:      end if
10:   else
11:     N_willingness = HELLO 消息包的 willingness;
12:     if 本地链路表中对应地址的 L_SYM_time >= 当前时间 then
13:         N_status = SYM;
14:     else
15:         N_status = NOT_SYM;
16:     end if
17:   end if
18:   return
```

算法 2-3 二跳邻居表更新

二跳邻居表更新

```
INPUT：HELLO 消息包；一跳邻居表；二跳邻居表；本地链路表
OUTPUT：none
1:    for item in 一跳邻居表
2:      if item.N_status == SYM and item.N_neighbor == originatorAddress then
3:        for neighs in HELLO 消息包的邻居类型消息
4:          if 邻居类型 == MPR_NEIGH or 邻居类型 == SYM_NEIGH then
5:            if neighs != 节点自身地址 and neighs not in 二跳邻居表 then
6:              new 二跳邻居表项，新表项的值如下
7:              N_2hop_addr = neighs;
8:              N_time = 当前时刻 + vTime;
9:            else if neighs != 节点自身地址 and neighs in 二跳邻居表 then
10:             N_time = 当前时刻 + vTime;
11:           end if
12:         else if 邻居类型 == NOT_NEIGH then
13:           删除二跳邻居表中对应表项;
14:         end if
15:     end if
16:   return
```

算法 2-4 MPR 选择集更新

MPR 选择集更新

```
INPUT：HELLO 消息包；MPR 选择集
OUTPUT：none
1:    for item in HELLO 消息包的邻居类型为 MPR_NEIGH 的地址
2:      if item not in MPR 选择集 then
3:        new MPR 选择集项，新项的值如下
```

```
4:          MS_addr = originatorAddress;
5:          MS_time =  当前时刻  + vTime;
6:       else
7:          MS_time =  当前时刻  + vTime;
8:       end if
9:    return
```

2.3.3 TC 消息生成

TC 消息转发 MPR 选择集，协助节点构建网络的整体拓扑。由于广播的是每一个节点的 MPR 选择集，所以网络的整体拓扑是经过 MPR 机制简化的。

TC 消息仅携带每个节点的 MPR 选择集，按照 2.2.8 的格式加入 TC 消息。

ASSN 根据节点维护的 ASSN 值设置。

将 TC 消息封装在消息包中，messageType 为 TC，vTime 为拓扑保持时间（TOP_HOLD_TIME），TTL 为 255（尽可能大的数，为了是 TC 可以扩散到全网），hopCount 为 0，originatorAddress 为当前节点地址，messageSequenceNumber 根据节点记录的计数值设置。

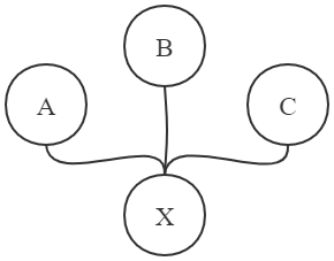

图 2-17 拓扑示意图

图 2-17 是根据 MPR 选择集得到拓扑的简易示意图。节点 ABC 均选择 X 作为 MPR 节点，在 HELLO 广播之后，X 可以建立自己的 MPR 选择集，包含了 ABC 三个节点。当 X 的 MPR 选择集不为空时，X 广播 TC 消息，即自己的 MPR 选择集信息，当网络中其他节点接收到之后，便可以确认 A<->X，B<->X，C<->X 的拓扑。

2.3.4 TC 消息处理

TC 消息的作用是生成网络拓扑。当节点具有 MPR 选择集的信息后，节点按照发送时间间隔，周期性发送 TC 消息，告知节点自身周围的拓扑，其他节点接收到 TC 消息后，将其记录下来，组合成全网的拓扑信息。

算法 2-5 是节点接收到 TC 消息的处理过程。

算法 2-5 更新拓扑表

更新拓扑表

```
INPUT：TC 消息包；拓扑表
OUTPUT：none
1:    for item in TC 消息包的地址
2:       if item not in  拓扑表  then
3:          new  拓扑表项，新项的值如下
4:          T_dest_addr = item;
5:          T_last_addr = originatorAddress
6:          T_seq = ASSN;
7:          T_time =  当前时刻  + vTime;
8:       else
9:          if T_seq < ASSN then
10:            删除该表项
11:         else
12:            T_seq = ASSN;
13:            T_time =  当前时刻  + vTime;
14:         end if
15:      end if
16:   return
```

2.3.5 MPR 生成

MPR 机制是 OLSR 的核心思想，节点通过节点存储的一跳邻居表、二跳邻居表计算出 MPR 生成集。由于 MPR 生成集依赖于一跳邻居表、二跳邻居表，所以当前两者改变后，MPR 需要再次计算。

MPR 的选择分为：

1. 选择能够唯一覆盖某些两跳邻居的一跳邻居作为 MPR。
2. 在剩余的一跳邻居中，按照覆盖二跳邻居的数量由高到低排序，优先选择数量搞得，直到覆盖所有的二跳邻居。

MPR 生成需要用到的概念。

N：节点的一跳邻居的子集，需要满足 N_status 为 SYM，即选择对称的一跳邻居。

N2：节点的二跳邻居的子集，需要排除节点自身，排除属于 N 中的节点，对于通过不愿转发的 N 节点才能达到的二跳邻居的点也需要排除（即节点的 willingness 为 WILL_NEVER，因为该节点不会转发接收到的广播消息）。

Dy：一跳邻居覆盖深度，即一跳邻居覆盖 N2 中的点的数量。

ReachNum：N2 删除被 MPR 节点覆盖的节点后，一跳邻居覆盖的删除后的 N2 的数量。

算法 2-6 是原始 OLSR 协议计算 MPR 的流程。

算法 2-6 原始 OLSR 协议 MPR 生成算法

Origin OLSR 协议计算 MPR

```
INPUT：一跳邻居表；二跳邻居表
OUTPUT：MPR 生成集
1:   计算 N 和 N2;
2:   根据 N 和 N2 计算 Dy;
3:   while N2 is not empty
4:     for item in N2
5:       if item 仅能通过 N 中的某一个节点 i 可达 then
6:         MPR.insert(i);
7:         将 i 从 N 中删除;
8:         将 N2 中被 i 覆盖的点删除;
9:       end if
10:    if N2 is empty then
11:      break
12:    end if
13:    计算 N 中每个点的 ReachNum;
14:    for item in N
15:      选择 ReachNum 最大的节点;
16:    if 选择的节点数量大于 1 then
17:      选择 Dy 最大节点 i
18:      MPR.insert(i);
19:      将 i 从 N 中删除;
20:      将 N2 中被 i 覆盖的点删除;
21:    end if
22:  return MPR
```

2.3.6 重复消息处理

当节点接收到广播消息后，需要判断广播的消息类型以及是否在之前已经接收过相同的消息，对于类型不属于 HELLO 的消息要进行转发。算法 2-7 是重复消息处理的过程。

算法 2-7 重复消息处理

重复消息处理

```
INPUT：重复记录表；消息包
OUTPUT：none
```

```
1:  if TTL <= 1 || originatorAddress == 节点自身 then
2:     return;
3:  end if
4:  if 消息包 not in 重复记录表 then
5:     new 重复记录表项，新项的值如下
6:     D_addr = orginatorAddress;
7:     D_seq_num = messageSequenceNumber;
8:     if 消息包的数据包的发送节点地址在节点的 MPR 选择集 then
9:        消息包的 TTL 减一;
10:       消息包的 hopCount 加一;
11:       转发消息包;
12:       D_retransmitted = true;
13:    else
14:       D_retransmitted = false;
15:    end if
16:    D_received = true;
17:    D_time = 当前时刻 + DUP_HOLD_TIME;
18: else
19:    if D_retransmitted = false and 消息包的数据包的发送节点地址在节点的
       MPR 选择集 then
20:       消息包的 TTL 减一;
21:       消息包的 hopCount 加一;
22:       转发消息包;
23:       D_retransmitted = true;
24:    end if
25: end if
26: end if
27: return;
```

2.4 OPNET 介绍

本文使用 OPNET modeler 进行 OLSR 协议的仿真。

OPNET 的网络建模分为三个层级：网络模型，节点模型，进程模型。

网络模型定义了仿真网络的拓扑结构，由标准模型库中的节点模型或者用户自定义的节点模型构成。有线网络在定义时会确定物理连接等拓扑信息，无线网络不能确定物理连接，但是可以设定运动轨迹，以模拟真实的节点运动行为。图 2-18 是网络模型的一个例子。网络模型中可以对节点暴露到上层的属性进行设置，例如节点名称/节点号，坐标等信息。其中坐标信息中的高度决定了

节点的理论极限传输距离。假定网络模型的场景在地球上，将地球近似为球体，公式(2-1)根据可以计算出节点的理论极限传输距离(h 是节点的高度，R 是地球半径 6311km)，这决定了无线自组网如何安排一跳的范围。

$$Dis = \sqrt{h^2 + 2Rh} \tag{2-1}$$

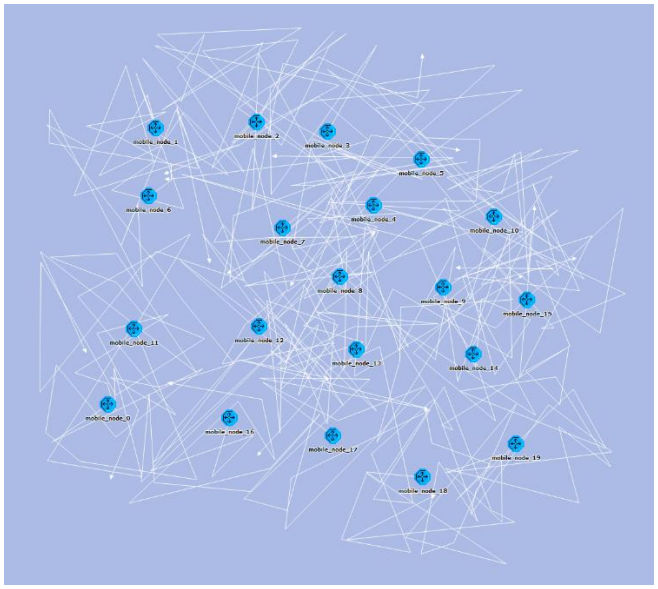

图 2-18 网络模型

节点模型定义了节点的模块。一个无线通信节点一般由天线模块 a_0，无线发射机 tra_0，无线接收机 rcv_0，自定义处理机 Route_manager 或者标准处理机模块构成。处理机定义了节点的行为。图 2-19 是节点模型的一个例子。

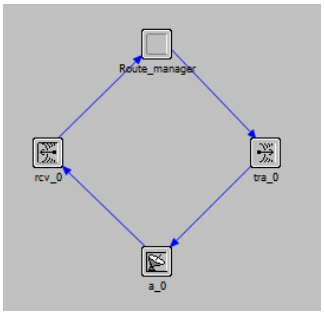

图 2-19 节点模型

进程模型是模块的具体实现，自定义模块的具体实现需要用户实现。进程模型其实是有限状态机。但是由于 OPNET 自身的编辑器以及编码方式对用户不友好，所以本文采用了一个自循环的状态机，将节点行为定义为类，节点进程作为一个实例，通过捕获 OPNET 中断并判断中断类型，实现进程不同状态的转化，这样可以大幅减少开发时间和开发过程中可能产生的 bug。图 2-20 是进程模型的一个例子。

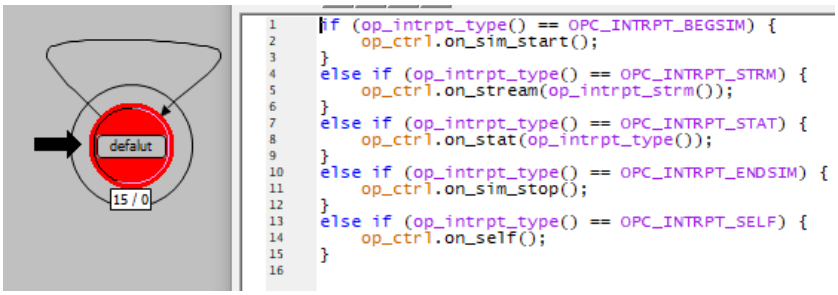

图 2-20 面向对象的进程模型

由于采用了上述方式开发，所以运行采用命令行形式，例如代码 2-1

代码 2-1 OPNET 终端运行命令

```
1:    op_runsim -net_name test_net_name -duration 20 -c
```

其中 op_runsim 是运行的函数，-net_name 是运行的网络名称，-duration 是仿真运行时长，-c 是强制重新编译。

第3章 MPR 机制改进

3.1 MPR 冗余分析与改进

传统 OLSR 的 MPR 选择算法存在冗余，其冗余性体现在图 3-1 所示的例子中。

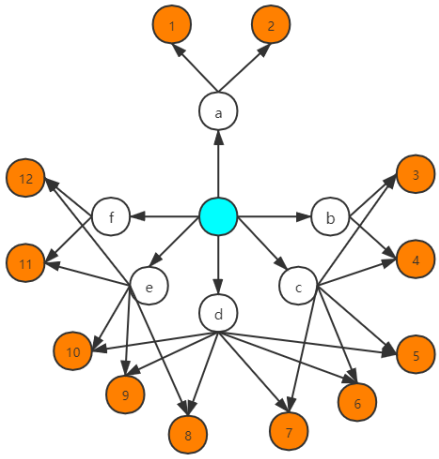

图 3-1 MPR 冗余示例拓扑图

中间的蓝色节点广播消息，根据传统的 OLSR 算法，选择 a、b、c、d、e、f 节点作为 N 集，1、2、3、4、5、6、7、8、9、10、11、12 作为 N2 集。首先 1、2 是 N2 中仅能通过 N 中的 a 节点到达的节点，所以将 a 加入 MPR 集。将 1、2 从 N2 中删除后，N2 不存在仅能通过 N 中的某个节点才能到达的节点，所以计算其余节点的 ReachNum。节点 d 的 ReachNum 最高，将节点 d 加入 MPR，将 5、6、7、8、9、10 从 N2 中删除。再次计算 ReachNum，3、4、11、12 可达性相同，所以根据节点的 Dy 值选择出较大的节点，后续会选择出 c、e 节点。MPR 生成结果如图 3-2 所示。

虽然 MPR 节点正确覆盖了所有的 N2 节点，但并不是最优的，节点 5、6、7 接收到来自 c、d 的重复的广播消息，节点 8、9、10 接收到来自 d、e 的重复的广播消息。原因是原始算法贪心策略并不合理，尽可能选择覆盖范围大的节点，可能会选择出两个覆盖较大范围的节点，但是覆盖范围有重合的部分，从而导致选择的 MPR 节点产生了冗余。

优化 MPR 机制可以改变贪心策略[11]，将网络中除了唯一可达某个 N2 节点以外的覆盖范围最小的节点删除，按照这种方法选择出的 MPR 节点可以达到

局部最优。在去除冗余的基础上，将全局因素引入 MPR 生成算法[12]，优先考虑被其他节点选为 MPR 的节点，得到全局最优的结果。本文采用首先尝试了局部优化，在论文[11]的基础上，删除不必要的 Dy 计算，简化 MPR 计算流程，得到效果相同的局部优化效果。

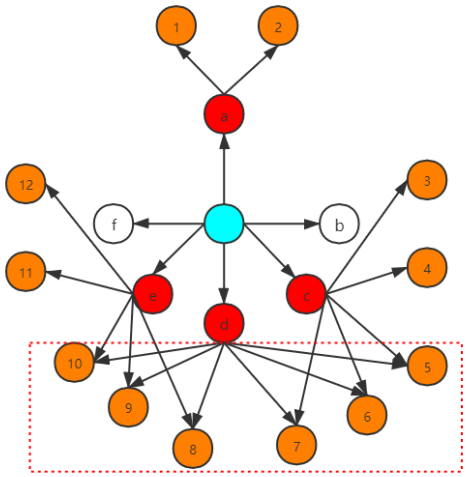

图 3-2 传统 OLSR 协议 MPR 生成结果

算法 3-1 改进的 MPR 生成算法

改进的 MPR 生成算法

INPUT：一跳邻居表；二跳邻居表
OUTPUT：MPR 生成集
1: 计算 N 和 N2;
2: 将 N 的二跳邻居中不属于 N2 的二跳邻居删除，得到 N 对应的 N2 集。
3: 根据 N 和 N2 计算 Dy;
4: **while** N2 is not empty
5: **for** item **in** N2
6: **if** item 仅能通过 N 中的某一个节点 i 可达 **then**
7: MPR.insert(i);
8: 将 i 从 N 中删除;
9: 将 N2 中被 i 覆盖的点删除;
10: **end if**
11: **if** N2 is empty **then**
12: **break**
13: **end if**
14: **for** item **in** N
15: 删除其 N2 集中不存在于 N2 中的节点，相当于重新计算 N 的 ReachNum;
16: 选择 N 中 N2 集最少的点 i，从 N2 中删除;
17: // 这里存在优化的可能性，可能存在 N2 集数量相同的 N 的节点
18: 删除节点 i 覆盖的 N2 中的点;
19: **return** MPR

根据改进的算法 3-1，重新计算图 3-1 的 MPR 生成集。

首先选择节点 a 加入 MPR，因为 1、2 节点仅能通过 a 达到，删除 N 中的 a 和 N2 中的 1、2，计算剩余节点的 ReachNum，节点 b、f 的 ReachNum 为 2，均是最小，任意删除其中一个节点。不妨删除节点 b，此时 3、4 仅能通过节点 c 得到，所以将 c 加入 MPR 集，并删除 N2 中被 c 覆盖的节点 3、4、5、6、7。再次计算 ReachNum，节点 f 的 ReachNum 最小为 2，删除节点 f，此时 11、12 仅能通过节点 e 到达，所以将 e 加入 MPR 集，并删除 N2 中被 e 覆盖的节点 8、9、10、11、12。此时 N2 为空，算法结束。

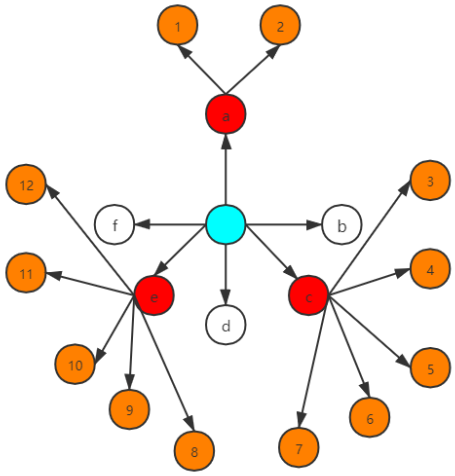

图 3-3 改进的 MPR 生成结果

图 3-3 是改进的 MPR 生成结果，对比原始的 MPR 生成算法，原本的 MPR 节点 d 被删除，消除了 5、6、7、8、9、10 节点的冗余数据包。

通过算法 3-1 选择 MPR 节点时，可能会出现两个 ReachNum 相同的节点，此时的处理方式是任意选择其中一个节点删除。全局优化[12]可以优先考虑被其他节点选为 MPR 节点的节点，所以可以将不属于其他节点 MPR 生成集的节点删除，但是此种方法并不能保证区分两个节点，因为在网络构建初期，可能所有的节点均未被选为 MPR 节点，那么这两个节点依然没有区别。

Romit Roy Choudhury, Nitin H. Vaidya 研究了 MAC 层任播在无线自组网中的应用[13]。论文[13]将 MAC 的信道状态（例如信道阻塞程度）作为网络层选择路由的依据。本文将无线自组网物理层的特性作为评估节点之间链路质量的依据，当出现需要在两个网络层条件相同的节点需要选择时，根据链路质量进行

选择。

原始 OLSR 协议在决定节点的 willingness 值时，初始设定的是 WILL_DEFAULT，后续没有更新，所以本文将链路质量作为更新 willingness 的依据，在生成 MPR 时，通过判断 willingness 值，就可以考虑物理层特性来生成 MPR 集。

算法 3-2 考虑物理层特性的 MPR 生成算法

考虑物理层特性的 MPR 生成算法

```
INPUT：一跳邻居表；二跳邻居表
OUTPUT：MPR 生成集
计算 N 和 N2;
将 N 的二跳邻居中不属于 N2 的二跳邻居删除，得到 N 对应的 N2 集。
根据 N 和 N2 计算 Dy;
while N2 is not empty
    for item in N2
        if item 仅能通过 N 中的某一个节点 i 可达 then
            MPR.insert(i);
            将 i 从 N 中删除;
            将 N2 中被 i 覆盖的点删除;
        end if
    if N2 is empty then
        break
    end if
    for item in N
        删除其 N2 集中不存在于 N2 中的节点，相当于重新计算 N 的 ReachNum;
    选择 N 中 N2 集最少的点 i，从 N2 中删除;
    if 存在多个相同的 N2 集最少的点 then
        将 i 更新为 willingness 最小的点;
    end if
    删除节点 i 覆盖的 N2 中的点;
return MPR
```

算法 3-2 在算法 3-1 的基础上加入了 willingness 的因素，使得 OLSR 协议可以考虑物理层特性。

3.2 链路质量评估

算法 3-2 的前提是链路质量评估。链路质量评估的常见方式有传输距离[14,15]，传输延时[16]，信噪比[17]等。在这些评估方式中，信噪比通常认为是更加有效的[18]。

在 OPNET 中，可以方便的获取数据包的传输距离，传输延时，信噪比等信息，根据这些信息，本文设计了一种评估链路质量的算法。

算法 3-3 链路质量评估

链路质量评估

```
INPUT：一段时间内的数据包数组 pack；链路质量计算间隔 interval；
OUTPUT：Averaged_quality
// 计算时间权重;
weigh[interval];
if interval is 偶数 then
    midb = interval / 2 – 1;
    mida = interval / 2;
    cnt = 1 / interval / interval
    for i in range(interval/2)
        weigh[midb-1] = 1 / interval – cnt * (i + 1);
        weigh[mida+1] = 1 / interval + cnt * (i + 1);
else if interval is 奇数 then
    mid = interval / 2;
    cnt = 1 / interval / interval
    for i in range(interval/2+1)
        weigh[mid-1] = 1 / interval – cnt * i;
        weigh[mida1] = 1 / interval + cnt * i;
end if
Averaged_quality = 0;
for item in pack
    Averaged_quality = item * 对应时间区间 i 的 weigh[i] + Averaged_quality;
return Averaged_quality;
```

算法 3-3 设置了评估链路质量的时间区间，统计区间内接收到的数据包物理层属性，根据按秒为单位计算出的时间权重，得到属性值的加权平均值。由于传输距离和传输延时受到发射功率，带宽，地形等影响不易于量化，本文仅讨论其可行性，而信噪比在确定了调制模式的情况下，和误码率的关系是一致的，且在误码率可接受的范围内，变化接近线性，如图 3-4 所示是 QPSK 的信噪比与误码率的对应关系，所以本文根据计算出的加权平均信噪比将其划分为 7 个区间，对应 willingness 的其中取值，具体的取值将在第 4 章描述。

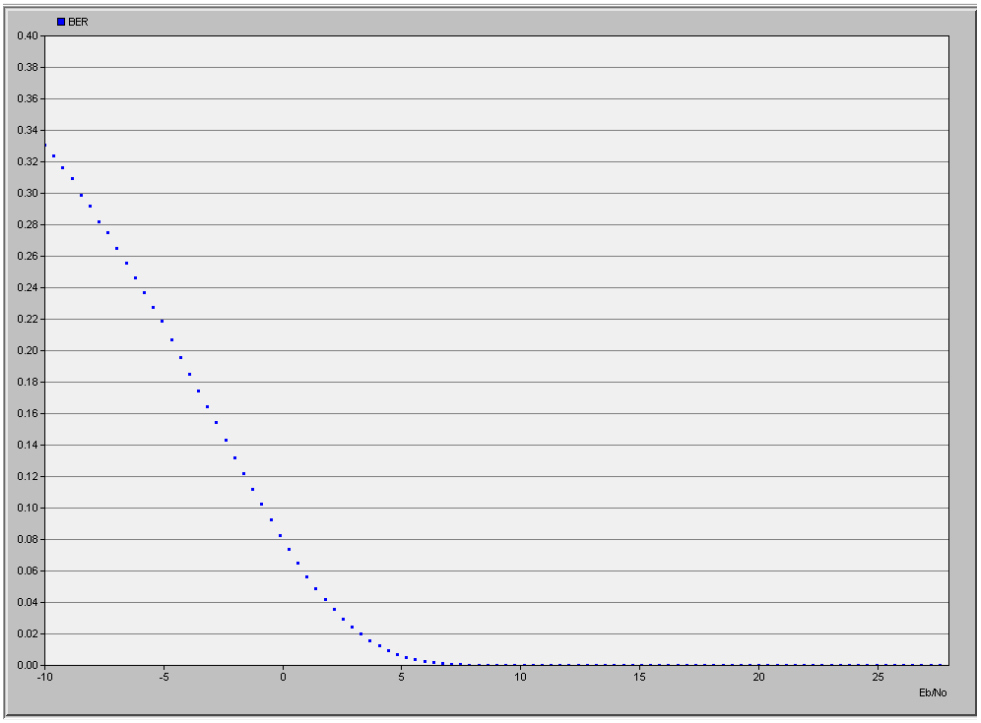

图 3-4 QPSK 调制解调方式的 Eb/no 与 BER 的关系

第4章 仿真结果分析

4.1 仿真实现

4.1.1 网络模型

网络包含了 20 个自定义的节点，运动轨迹由随机算法生成随机的不同运动范围的轨迹，单向变化范围分别为 10m，1000m，5000m，10000m。网络拓扑随机生成，并保证所有节点均可入网。

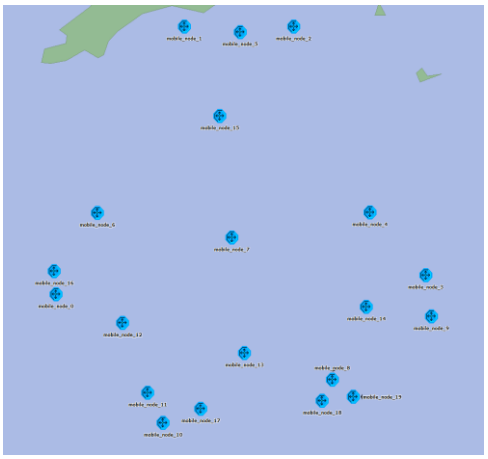

图 4-1 10m

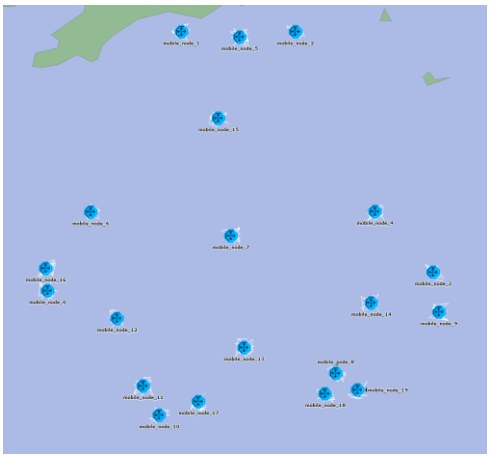

图 4-2 1000m

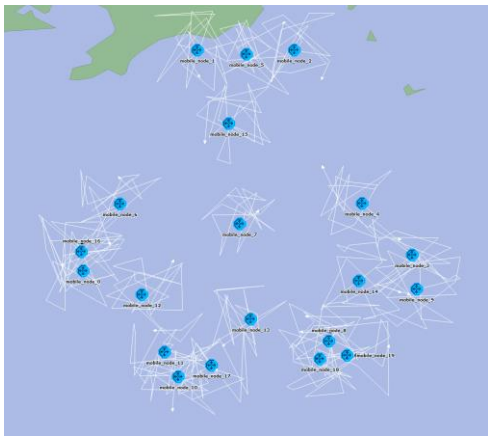

图 4-3 5000m

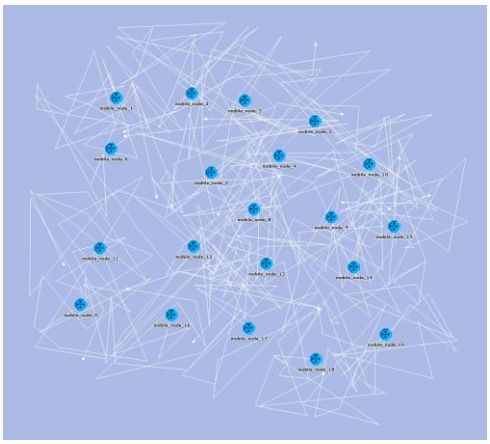

图 4-4 10000m

图 4-1,图 4-2,图 4-3,图 4-4 是生成的几种不同的网络模型图。图 4-1,图 4-2,图 4-3 是同一种网络的不同运动范围，图 4-4 是完全随机生成的 10000m 随机运动的网络模型。

4.1.2 节点模型

节点模型如图 4-5 所示，包含了一个自定义处理机模型 Route_manager、一个无线发射机模型 tra_0、一个无线接收机模型 rcv_0、一个天线模型 a_0。

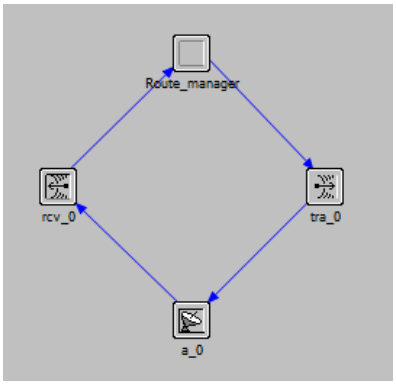

图 4-5 节点模型

Route_manger: 处理来自 rcv_0 的数据包，将生成的数据包发送到 tra_0，自身响应 OPENT 的中断，根据定义的算法进行收包发包的逻辑处理。

tra_0: 接收来自 Route_manger 的数据包，并通过天线 a_0 发送。

rcv_0: 接受来自天线 a_0 的数据包，发送到 Route_manger。

a_0: 发送或接受能感知到的数据包。

4.1.3 进程模型

进程模型由于采用了面向对象的设计方法，所以将进程模型简化为一个自循环的状态机，进程中断的类型进入不同的状态，进程自身产生自中断，并根据自中断的中断码实现不同的功能，如图 4-6 所示。

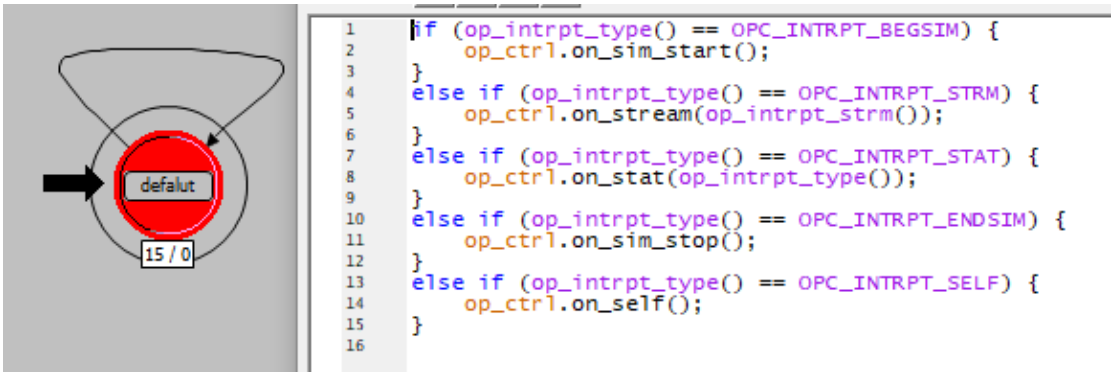

图 4-6 进程模型

节点的自中断主要包括 OPC_HELLO_SEND、OPC_TC_SEND、OPC_QUA_CAL，分别处理 HELLO 消息发送，TC 消息发送和链路质量评估的请求。节点通过函数 op_intrpt_schedule_self(time, code)产生自中断，time 是产

生自中断的时间，code 是区分不同自中断的中断码。OPC_HELLO_SEND 用于安排节点发送 HELLO 消息的中断，OPC_TC_SEND 用于安排节点发送 TC 消息的中断，OPC_QUA_CAL 用于安排节点计算链路质量的中断。其余中断由 OPNET 产生，OPC_INTRPT_BEGSIM 表示仿真开始，OPC_INTRPT_ENDSIM 表示仿真结束，OPC_INTRPT_SELF 表示检测到自中断，节点根据 op_intrpt_code()获取自定义的自中断码，根据自中断码判断需要处理的事务，OPC_INTRPT_STRM 表示节点接收到数据包，进行收包的后续处理，OPC_INTRPT_STAT 表示节点接收到数据统计中断。

4.1.4 仿真参数设置

HELLO_INTERVA：HELLO 消息发送间隔，设置为 2s。

TC_INTERVAL：TC 消息发送间隔，设置为 5s。

NEIGHB_HOLD_TIME：邻居保持时间，设置为 6s。

TOP_HOLD_TIME：拓扑保持时间，设置为 15s。

DUP_HOLD_TIME：重复记录表保持时间，设置为 30s。

WILL_UP_TIME：链路质量评估间隔，设置为 HELLO 消息发送间隔和 TC 消息发送间隔的最大值。

4.2 结果分析

4.2.1 MPR 消除冗余

使用图 4-1，图 4-2,图 4-3 所示的网络模型验证 MPR 消除冗余的效果。使用命令 op_runsim 运行仿真，仿真时间设置为 50s，记录节点生成 MPR 的数量。对比的算法是算法 2-6 和算法 3-1。

图 4-7、图 4-8、图 4-9、图 4-10、图 4-11、图 4-12 展示了节点 7 和节点 15 分别在单维运动范围为 10m、1000m、5000m 的情况下，根据原始协议 MPR 生成算法和消除冗余的 MPR 生成算法生成的 MPR 数量对比。其他节点的情况和节点 15 类似，仅有节点 7 消除了冗余，原因消除冗余的改进仅针对特定网络拓扑，即可能产生冗余的拓扑，对于一般的拓扑，产生的 MPR 集应该是一致的。

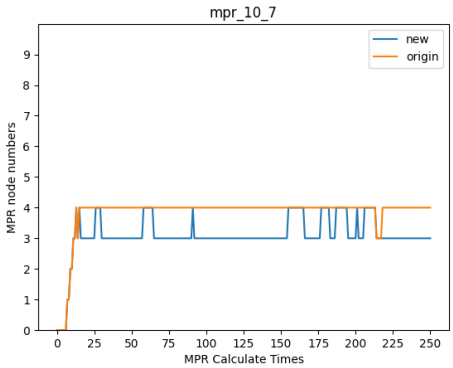

图 4-7 节点 7MPR 生成数量（10m）

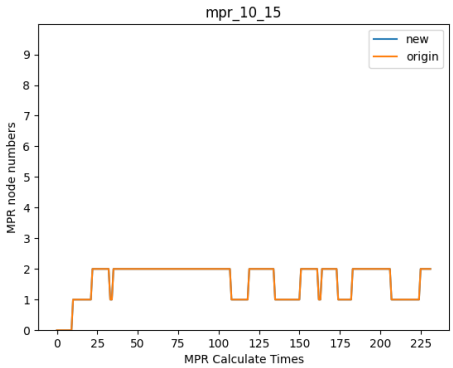

图 4-8 节点 15MPR 生成数量（10m）

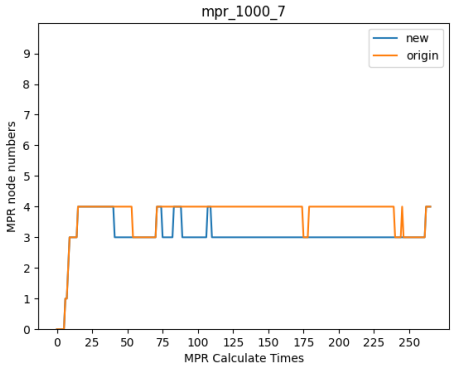

图 4-9 节点 7MPR 生成数量（1000m）

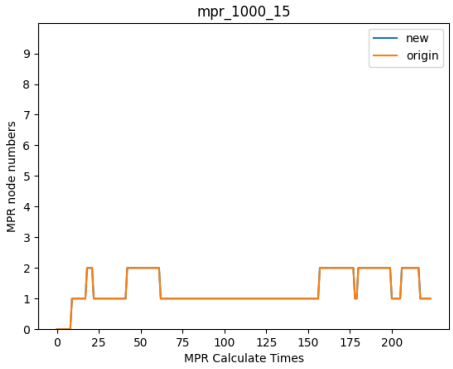

图 4-10 节点 15MPR 生成数量（1000m）

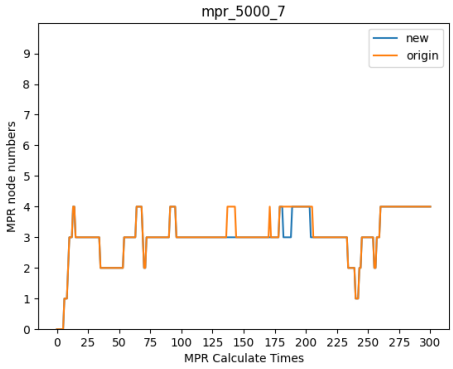

图 4-11 节点 7MPR 生成数量（5000m）

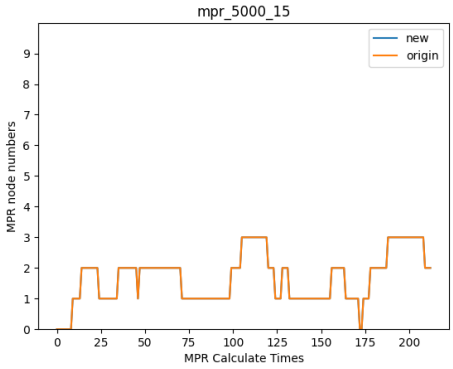

图 4-12 节点 15MPR 生成数量（5000m）

观察图 4-7、图 4-8、图 4-9、图 4-10、图 4-11、图 4-12，消除的 MPR 生成算法在节点运动范围较小的情况下，大部分时间产生的 MPR 的数量都要比原始 MPR 生成要少，而随着节点运动范围的增大，两者之间的数量差距在逐渐缩小。这是因为节点运动范围的增大会导致网络拓扑发生变化，原本存在可能

产生冗余的拓扑转变成不产生冗余的拓扑，所以导致两种算法的差距减小。上述的结果说明，这种 MPR 消除冗余的算法是有效的。

4.2.2 MPR 加入信噪比因素

使用图 4-4 所示的网络模型验证 MPR 加入信噪比因素后的提升。验证在动态网络下的效果。使用 op_runsim 运行仿真，仿真时间为 1200s，记录 TC 转发的数量和平均延迟。仅观察动态网络是因为静态网络下节点的信噪比在 OPENT 仿真中是不变化的，所以即使考虑了信噪比因素，节点的 willingness 也不会发生变化，对于 MPR 的选择没有影响。

在统计之前，进行长时间的数据包发送测试，统计出节点接收数据包的信噪比变化范围大致在 5.5 以上，根据图 3-4，5.5 以上基本没有误码，所以将加权平均信噪比为 5.5 以下的节点的 willingness 设置为 WILL_NEVER，5.5~6 为 WILL_LOWER，6~7 为 WILL_LOW，7~8 为 WILL_DEFAULT，8~9 为 WILL_HIGH，9~10 为 WILL_HIGHER，10~11 为 WILL_ALWAYS。

图 4-13 展示了动态网络 TC 转发数量的对比。

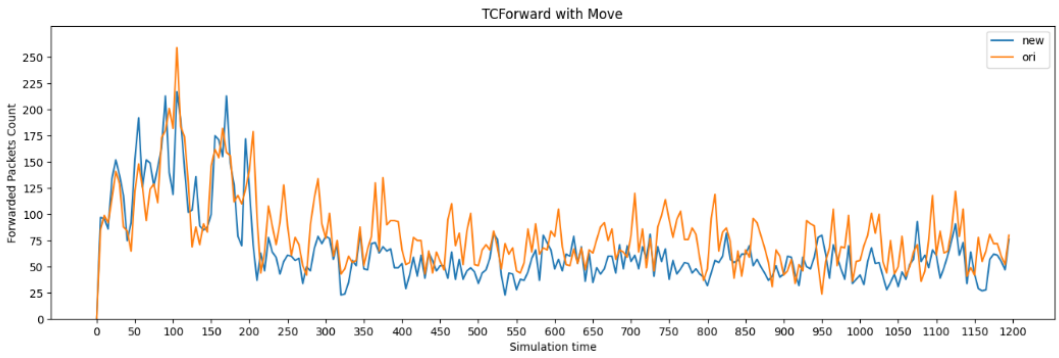

图 4-13 动态网络 TC 转发数量对比

观察图 4-13，网络初期转发 TC 的数量较多，后期逐渐平稳，而考虑了信噪比因素的网络在后期转发 TC 的数量明显少于原始的 OLSR 协议。这说明在选择 MPR 时，节点的加权平均信噪比是有影响的，考虑这一物理层特性将更有利于减少 OLSR 协议控制数据包的转发，减少网络的开销。

第5章 结束语

5.1 本文总结

OLSR 协议作为应用较为广泛的先验式无线自组网路由协议，很好的适应了无线自组网的不依赖于特定设备、随时随地快速构建的特点。其中 MPR 机制有效的减少了控制数据包的冗余转发数量。

但是由于 MPR 机制追求的最小 MPR 选择问题是一个 NP 问题，且原始 OLSR 协议存在冗余，本文着重与 MPR 选择的优化，完成以下工作：

1. 首先仅从网络层进行 MPR 选择优化，修改了 MPR 生成算法的策略，经过验证，改进之后的算法可以有效的减少 MPR 生成的数量，实现了局部最优，但是由于动态网络的拓扑变化较快，所以改进算法并不能始终保持较优的结果，因为仅仅是消除了冗余，但是对于情况相同的节点，并不能从中选择出更优的节点。
2. 为了解决网络层不能区分更优节点的问题，本文在计算 MPR 时考虑节点的物理层特性。具体做法是周期性统计节点的物理层属性，例如接收数据包的信噪比，根据时间加权得到平均值之后，将其划分不同的 willingness 等级，以此更新节点的 willingness 值，用于 MPR 的计算。这种做法有效的减少了 TC 控制数据包转发的数量，减少了无线自组网的控制开销。

5.2 工作展望

本文对于算法的验证仅在 OPNET 上实现，尚未在实际节点上实现，OPNET 仿真软件虽然可以较好的模拟真实网络状态，但是并不能反馈真实的自由空间。

本文采用的物理层特性是信噪比，但是实际中计算信噪比并不像 OPENT 一样方便。在信噪比的计算公式中，需要得到节点的发射功率，发射增益，接收功率，接受增益，环境噪声功率，背景噪声功率，自由空间路径损耗。其中自由空间路径损耗需要得知节点数据包的传输距离，在不使用特殊方法的情况

下，传输距离是上帝视角获取的，所以这套方案在实际中的实现难度较大。另一种比较有效的方式是计算误码率，可以采用和 HELLO、TC 等控制消息类似的方式，周期性发送一段特殊的数据，例如 PN23 序列，接收到数据包后，计算出其误码率，根据算法 3-3 计算出加权平均值，并划分为不同的 willingness 值。实现可行度要高于信噪比，而在 OPNET 中因为无线接收机会将经过纠错之后仍然无法读取的包丢弃，所以在 OPNET 中并没有尝试误码率的因素。

在后续的工作中，需要改变 OPNET 的无线接收机的模型，保留所有的数据包，使得 OPNET 可以验证误码率带来的提升，同时尝试将目前基于信噪比的算法移植到实际的设备上，验证其可行性。

致　谢

首先感谢我的导师申兆岩副教授，以及清华大学宽带传输研究室的张彧副教授，李钊，镡晶晶老师。感谢他们对我的毕业设计的悉心指导，在毕业论文的选题，关键问题的攻克等方面提供的无私教诲和热切的帮助。

申兆岩副教授对待工作的热情，对待学术的严谨，对待新知识的渴望，对待学生的认真负责给我留下了深刻的印象，影响了我本科以及未来对待学习和工作的态度，在此我再次向申兆岩副教授表示崇高的敬意和衷心的感谢。清华大学宽带传输研究室的各位老师们在工作繁忙之际，帮助我快速补全毕业设计所需的跨学科知识，为顺利完成这一交叉学科的项目提供了帮助，再次感谢各位老师毫无保留的指导。

其次，我要感谢本科四年所有的任课老师，他们讲授的专业知识是我完成毕业设计不可或缺的部分，也是我未来学习工作的基石。老师们高质量的课堂，将我从一名新生培养成合格的计算机专业本科毕业生，感谢他们四年来的辛勤付出。

感谢一直以来为我日夜操劳无条件支持我的父母和关心挂念我的家人们。在我本科求学最困难的时候，是他们施以援手，才能让我克服困难，顺利完成学业。我向他们表示我最真挚的感谢。

最后感谢所有参考文献的作者们，是他们组成了巨人的肩膀，借鉴他们的研究成果，我才能更进一步，完成毕业设计。感谢评阅本论文的各位专家老师在百忙中抽出时间对我的毕业设计批评指正。

参考文献

[1] 张莎,李腾飞.无线自组网技术研究综述[J].数字通信世界,2020(07):1-4.
[2] Ashutosh Sharma, Rajiv Kumar. Performance comparison and detailed study of AODV, DSDV, DSR, TORA and OLSR routing protocols in ad hoc networks[A]. 2016 Fourth International Conference on Parallel, Distributed and Grid Computing (PDGC)[C]. IEEE,2016:732-736
[3] C. Perkins, E. Belding-Royer, S. Das. RFC3561: Ad hoc On-Demand Distance Vector (AODV) Routing[M]. USA:RFC Editor,2003
[4] Yih-Chun Hu, Dave Maltz, David Johnson. The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4[Z]. datatracker.ietf.org. 2020-01-21
[5] Thomas Clausen, Philippe Jacquet Optimized Link State Routing Protocol (OLSR)[Z]. datatracker.ietf.org. 2017-08-22
[6] Hemanth Narra, Yufei Cheng, Egemen K. Çetinkaya, Justin P. Rohrer, James P. G. Sterbenz. Destination-sequenced distance vector (DSDV) routing protocol implementation in ns-3[A]. Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques[C]. Brussels, BEL:ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2011:439–446.
[7] Debajit Sensarma, Koushik Majumder. An efficient ant based qos aware intelligent temporally ordered routing algorithm for manets[J]. International journal of Computer Networks & Communications. 2013,5(4):189-203
[8] T. Ravi Nayak, Sake. Pothalaiah, Dr.K. Ashok Babu. Implementation of Adaptive Zone Routing Protocol for Wireless Networks[J]. Wireless Communication. 2010,2(11):401-410
[9] A. Qayyum, L. Viennot, A. Laouiti. Multipoint relaying for flooding broadcast messages in mobile wireless networks[A]. Proceedings of the 35th Annual Hawaii International Conference on System Sciences[C]. IEEE,2002:3866-3875
[10] Yu-Chee Tseng, Sze-Yao Ni, Yuh-Shyan Chen, Jang-Ping Sheu. The broadcast storm problem in a mobile ad hoc network[J].Wireless Networks. 2002,8(2/3):153-167
[11] 郑伟明.OLSR 路由协议研究及仿真[D].电子科技大学,2011.
[12] 刘杰,王玲,王杉,冯微,李文.基于 OLSR 协议的最小 MPR 集选择算法[J].计算机应用.2015,35(2):305-308
[13] Romit Roy Choudhury,Nitin H. Vaidya. MAC-layer anycasting in ad hoc networks[J].ACM SIGCOMM Computer Communication Review.2004,3(1):75-80
[14] Guoan Hu. A Link Quality Evaluation Model Based on the Three-dimensional Space in Wireless Sensor Network[J].Information Technology Journal. 2014,13(4):720-724
[15] Xinyan Zhou, Xiaoyu Ji, Bin Wang, Yushi Cheng, Zhuoran Ma, Francis Choi, Brian Helmuth, Wenyuan Xu. Pido: Predictive Delay Optimization for Intertidal Wireless Sensor Networks[J].Sensors,MDPI AG,2018,18(5):1464
[16] Xiaohan Lai, Xiaoyu Ji, Xinyan Zhou, Longdao Chen. Energy Efficient Link-Delay Aware Routingin Wireless Sensor Networks[J]. IEEE Sensors Journal. 2018.1,18(2): 837-848
[17] Wee Lum Tan, Peizhao Hu, Marius Portmann.SNR-Based Link Quality Estimation[A]. 2012 IEEE 75th Vehicular Technology Conference (VTC Spring)[C]. IEEE,2012:1-5

[18] Angelos Vlavianos, Lap Kong Law, Ioannis Broustis, Srikanth V. Krishnamurthy, Michalis Faloutsos. Assessing link quality in IEEE 802.11 Wireless Networks: Which is the right metric?[A]. 2008 IEEE 19th International Symposium on Personal, Indoor and Mobile Radio Communications[C]. IEEE,2008:1-6

附录 1 代码及相关附件

MPR 集改进算法实现：

```
UNINT table_manager::createMprSet() {
    this->mprSet.clear();
    set<unsigned int> N;
    map<unsigned int, set<unsigned int>> N2;
    map<unsigned int, set<unsigned int>> N_neghbor;
    map<unsigned int, unsigned int> N_will;

    for (auto &i: this->oneHopNeighborTable) {
        if (i.N_status == SYM) {
            N.insert(i.N_neighbor_addr);
            N_will.insert(make_pair(i.N_neighbor_addr, i.N_willin
gness));
        }
    }
    for (auto &i : this->oneHopNeighborTable) {
        if (i.N_status == SYM) {
            set<unsigned int> tmp;
            for (auto &j : i.N_2hop)
                if (i.N_willingness != WILL_NEVER && N.find(j.N_2
hop_addr) == N.end() && j.N_2hop_addr != this->nodeId) {
                    tmp.insert(j.N_2hop_addr);
                    if (N2.find(j.N_2hop_addr) == N2.end()) {
                        set<UNINT> oneNeighbor;
                        oneNeighbor.insert(i.N_neighbor_addr);
                        N2.insert(make_pair(j.N_2hop_addr, oneNei
ghbor));
                    }
                    else
                        N2[j.N_2hop_addr].insert(i.N_neighbor_add
r);
                }
            N_neghbor.insert(make_pair(i.N_neighbor_addr, tmp));
        }
    }
    while (!N2.empty()) {
```

```
        for(auto &i : N2) {
            if (i.second.size() == 1) {
                unsigned int mprN = *i.second.begin();
                this->mprSet.insert(mprN);
                for (auto &k : N_neghbor[mprN]) {
                    N2.erase(k);
                }
                N_neghbor.erase(mprN);
            }
        }
        for (auto &i : N_neghbor) {
            for (auto &j : i.second) {
                bool inN2 = false;
                for (auto &k : N2)
                    if (j == k.first) {
                        inN2 = true;
                        break;
                    }
                if (!inN2)
                    i.second.erase(j);
            }
        }
        if (N2.empty())
            break;
        unsigned int minCount = INT_MAX;
        unsigned int order;
        set<unsigned int> equalSet;
        for (auto &k: N_neghbor) {
            if (k.second.size() < minCount) {
                minCount = k.second.size();
                order = k.first;
                equalSet.clear();
                equalSet.insert(k.first);
            }
            else if (k.second.size() == minCount) {
                equalSet.insert(k.first);
            }
        }
        if (equalSet.size() > 1) {
            UNINT willing = 6;
```

```
            for (auto &i : equalSet) {
                if (N_will[i] < willing) {
                    willing = N_will[i];
                    order = i;
                }
            }
        }
        for (auto &k : N2) {
            if (k.second.find(order) != k.second.end())
                k.second.erase(order);
        }
        N_neghbor.erase(order);
    }
    return this->mprSet.size();
}
```

附录 2 文献英文原文

MAC-Layer Anycasting in Ad Hoc Networks

Romit Roy Choudhury and Nitin H. Vaidya

Abstract- A wireless ad hoc network is formed by a group of wireless hosts, without the use of any infrastructure. To enable communication, hosts cooperate among themselves to forward packets on behalf of each other. A key challenge in ad hoc networks lies in design efficient routing strategies. While several routing protocols have been proposed, most of them aim to select one optimal route between the source and destination. The MAC layer at each intermediate node is then required to forward packets to the next downstream node on that route. We argue that choosing a single optimal route at the network layer may not be sufficient. Knowledge of short-term channel conditions at the MAC layer can play an important role in improving end-to-end performance. Instantaneous interference, channel contention, power constraints and other considerations may be taken into account along with the network layer's long-term view. This paper proposes MAC-layer anycasting – a forwarding strategy that combines the guidelines from the network layer, with MAC layer knowledge of the local channel. We describe some applications of MAC-layer anycasting, and discuss the performance related tradeoffs.

I. INTRODUCTION

Wireless Ad hoc networks are infrastructureless multi-hop networks in which nodes behave as mobile routers. Routing protocols attempt to choose "optimal" routes based on some optimality criteria (e.g., number of hops). However, in the process of selecting an optimal route, the routing protocol is often faced with the decision to choose between two equally good routes. Ties are often broken randomly. MAC-layer anycasting is a proposal that aims to utilize the knowledge of instantaneous channel condition in selecting the suitable downstream neighbor on shorter time scales. The observation that routes chosen by the network layer are "optimal" on a longer time scale, and ignores the possibility of transient variations in link conditions, motivates

our work on MAC-layer anycasting.

The key idea behind MAC-layer anycasting is to achieve the goals of the network layer, while invoking short-term optimizations at the MAC layer, based on local channel conditions. With the proposed approach, the network layer is given the option of specifying multiple downstream destinations to the MAC protocol. The MAC protocol assumes that forwarding the packet to any one of these destinations is acceptable to the routing layer. Depending on the current channel state, the MAC layer then forwards the packet to one of the specified neighbors. Out-of-order packet delivery is a potential problem with proposed anycasting. We discuss this, and other tradeoffs associated with anycasting, later in the paper.

II. PRELIMINARIES

Routing protocols can be broadly classified into “source routed” or “table-driven” protocols. In source routing, the sender of a packet completely specifies the route that the packet must traverse to reach its final destination. Johnson et al. proposed dynamic source routing (DSR) in which the sender node floods a route request (RREQ) probe in search of a route to the destination. Intermediate nodes that forward this request probe, append their identifiers to the probe. The probe that arrives first at the destination is assumed to have arrived on the optimal path. DSR uses this path for subsequent communication.

Table-driven routing protocols store routing information locally. Nodes exchange routing messages, either reactively or periodically, to update each other about the status of links in the network. When a node intends to send data packets to another node, it consults its routing tables for a route to the destination. It forwards the data packet to the appropriate neighbor in the route, who in turn consults its own tables to forward the packet further. An intermediate node is often faced with the decision to choose between two of its neighbors, both of which may be equally good for forwarding the packet to the final destination. Ties are broken randomly, without respecting the possibility that one of the nodes may not be suitable for immediate transmission. We believe that anycasting can be useful here – the MAC layer can make educated decisions in such scenarios, leading to potential benefits in

performance. In this paper, we would refer to table driven protocols while discussing the details of MAC-layer anycasting. Issues arising from the use of source routing will be discussed separately in Section V.

Roy et al. propose the notion of maximally zone disjoint routes. Based on previous traffic conditions, a sender selects routes that can maximally bypass congested regions. Our idea of anycasting differs from in the sense that we base our forwarding decisions on factors that vary on a shorter time scale. The routing layer only provides a set of acceptable options (not all of which may be optimal). The MAC layer then chooses the next hop depending on the instantaneous network condition. Pursley et al. proposed the idea of using “decoder side information” to aid forwarding decisions. By observing the number of correct symbols received (from a sequence of known transmitted symbols), the receiver may be able to estimate, statistically, the reliability of the link. The authors propose a metric, resistance, which is indicative of link quality. Using this metric, a node examines two outgoing links, and transmits the packet over the one with lower resistance. While this scheme handles variation in channel fluctuations, it does not consider issues related to the MAC layer. MAC-layer anycasting adapts to several MAC protocol constraints, as detailed in the rest of the paper.

Larsson presents the idea of “selection diversity forwarding”, in which a transmitter includes a multicast address (or a list of addresses) in the data packet. Neighbors of the node that are included in the multicast group (or the address list), reply to the packet serially with an ACK packet. The transmitter chooses one of its neighbors, based on the guidelines of the routing layer and the current link conditions learned from the data-ACK exchange. A “forwarding order” is now transmitted to the chosen neighbor, requiring it to forward the packet further. The chosen neighbor replies to the “forwarding order” with a “forwarding order ACK”. Clearly, waiting for all the replies before initiating the “forwarding order” may be wasteful. Jain et al. propose an improvement on the protocol in [9]. The authors propose to specify the list of addresses (similar to [9]) in order of priority. The protocol requires all nodes, included in the address list, to reply in sequence of priority, with the highest priority

first. Upon receiving the first reply (not always from the highest priority node), the transmitter immediately begins data packet transmission to that node. This reduces the overhead associated with waiting for multiple replies before transmitting a packet. Unlike [9], the order of priority must be specified a priori without knowledge of the instantaneous link conditions. In addition, specifying preferences and multiple addresses increases packet-size, leading to higher control overhead.

Although similar in spirit, MAC-layer anycasting can be distinguished from the body of existing work. The key distinction lies in the basis of decision-making. Observe that most of the previous schemes rely on probing the channel in some form, and choose the suitable neighbor based on explicit or implicit feedbacks. We argue that in several cases, waiting for feedbacks may not be necessary – the MAC layer may already possess necessary information. For example (more examples elaborated later), we observe that the MAC layer may be aware of permissible transmit power-levels at a given point of time. Previous schemes may probe the channel with an impermissible power level, obtain a negative feedback, and converge to the permissible power level. Clearly, using the knowledge available at the MAC layer can be useful in such scenarios. We discuss some wireless medium access control (MAC) protocols next.

We assume that the reader is familiar with the IEEE 802.11 protocol. Briefly, when using 802.11, an exchange of request to send(RTS)/clear to send(CTS) precedes DATA communication. Nodes that overhear the RTS/CTS defer their own transmissions, for the proposed duration of the DATA communication. Once the DATA packet has been transmitted, the receiver replies with an ACK to acknowledge successful reception

Several proposals in the recent past have tuned 802.11. However, the key idea of the protocol remains unchanged. Recently, with advances in antenna technology, several protocols have been proposed that use directional antennas at the MAC layer. The key ideas when using directional antennas may be summarized as follows. Due to the ability to transmit signals in a desired direction, most of the protocols propose to use a combination of directional and omnidirectional RTS/CTS/DATA and/or ACK.

Spatial reuse of the channel increases due to reduced interference. The notion of directional NAV enables a node to initiate transmissions that will not interfere with ongoing communication. Range extension, possible due to the higher gain of antenna beams, is an additional benefit – fewer-hop routes can be formed between the source and the destination. Although promising, directional antennas also pose some difficulties. Neighbor discovery, new types of hidden terminals, deafness are some of the problems that arise from directional communication. We believe that anycasting can help, when using directional antennas.

Research on multi-user diversity in medium access control protocols has also been a topic of interest. Qin et al. proposes a channel-aware ALOHA protocol, that schedules transmissions based on instantaneous channel conditions. Using a distributed approach, the protocol requires a node to transmit when its local channel conditions are favorable. Tsatsanis et al. proposed “network assisted diversity protocols”, where the possibility of exploiting corrupted packets has been explored. Put differently, the authors propose the idea of allowing multiple transmitters to collide multiple times (synchronously). From the vector of corrupted packets, the receiver then separates the individual packets, using known signal processing algorithms. DeCouto et al. have recently proposed an ETX metric to favor paths that are characterized by fewer losses and retransmissions. Put differently, while making the routing decisions, the network layer considers the information available at the MAC layer. However, once a route has been chosen, it is used irrespective of the possible changes in instantaneous channel conditions. Yarvis et al. have also proposed similar ideas in the context of sensor networks. The key idea in this paper is somewhat opposite to that of [20], [21]. The MAC layer requires the network layer to supply a set of routes, that it deems suitable. Unlike the above approaches, the MAC layer performs the final decision of choosing the neighbor that appears to be most appropriate at that instant of time.

III. MAC-LAYER ANYCASTING

MAC-layer anycasting can be envisioned as an enhancement to existing MAC and routing protocols. In the rest of this paper, we would call a routing protocol

“basic” if it has not been “enhanced” with the anycasting features. One possible architecture to implement MAC-layer anycasting is shown in Figure 1. This section discusses the framework of MAC layer anycasting, in the context of a generic MAC and routing protocol. We also propose a simple variation, named Ordered Anycasting2 . Later, we visit the applications of anycasting and discuss the tradeoffs in the context of wireless ad hoc networks.

The anycast framework requires the “basic” routing protocol to discover/maintain multiple routes for each flow, whenever possible. Clearly, all the discovered routes may not be equally good. When a packet arrives at the network layer, the routing protocol consults the routing state to determine the routes that may be available for the packet’s final destination. From these available routes, the routing protocol selects a subset containing K routes that may be deemed as the best. The network layer now forms what we call the anycast group. The anycast group contains the set of distinct next-hop neighbors, on the selected K routes. As an example, for a packet destined to D, the anycast group specified by the network layer at node S in Figure 2 could be the set (A,X). The packet and the anycast group are then handed down to the MAC layer. Upon receiving the packet, and the anycast group, the MAC layer must select any one suitable neighbor and attempt transmission to it. Instantaneous network conditions may play an important role in determining the selection. The next section presents some of the potential applications of anycasting, and illustrates how the neighbor selection policies may be designed. However, first we propose a simple variation to anycasting, named ordered anycasting.

Ordered anycasting

The routing layer at a node may discover multiple routes to a particular destination. All the routes may not be optimal. For example, if routes R1 and R2 are equally good (e.g., in terms of hop-count), and if both are better than route R3, then the network layer may desire to use R3, only if communication over routes R1 or R2 is currently not possible. Ordered anycasting is a simple variation to anycasting that aims to achieve exactly this. The routing layer ranks the members of the anycast group in order of its preference. The MAC layer attempts communication to a node,

only if all other nodes higher in the preference order, have proved to be “unavailable”.

附录 3 文献中文译文

无线自组网 MAC 层任播

Romit Roy Choudhury 与 Nitin H. Vaidya

摘要：无线自组网由一组无线主机构成，无需使用任何基础设施。为了实现通信，主机之间相互协作来转发彼此的数据包。无线自组网的关键在于如何设计有效的路由策略。尽管目前已经提出了几种路由协议，但是大部分目的是从源节点到目的节点之间选择一条最优路由。然后每一个中间节点的 MAC 层需要将数据包转发到这个路由的下一个下游节点。我们认为，在网络层选择一条最优路由可能并不高效。在 MAC 层中，短期信道状况信息可以有效提高端到端的性能。瞬时干扰，信道争用，功率限制和其他因素可能与网络层的长期视角一同考虑在内。本文提出了 MAC 层任播，一种结合了网络层规则和 MAC 层本地信道信息的转发策略。我们描述了 MAC 层任播的一些应用并讨论了和性能相关的权衡。

一、简介

无线自组网是无基础设施的多跳网络，其中的节点同时作为移动路由。路由协议根据某些最佳标准（例如跳数）选择“最优”路由。但是在选择最优路由的过程中，路由协议往往要在两条同样好的路由中做出选择。链路中的连接经常会随机性中断。MAC 层任播目的是在较短时间内利用瞬时信道状况信息来选择合适的下游邻居。我们观察到网络层选择最佳路由的时间更长，并且忽略了链路状况瞬时变化的可能性，这使得我们在 MAC 层任播上投入了研究。

MAC 层任播的关键思路是实现网络层的目标，同时基于本地信道状况在 MAC 层做短期优化。根据我们提出的方法，网络层可以为 MAC 层协议提供多个下游目标，MAC 层假设路由层可以接受将数据包转发给这些目标的任意一个。根据当前信道的状态，MAC 层接下来会将数据包转发给特定的邻居。乱序分组传送是任播的潜在问题，我们将在本文的后面讨论这个问题，并探讨可能的折衷方案。

二、背景

路由协议可以大致分为源路由或者表驱动协议。在源路由协议中，数据包的发送方完全指定了数据包必须经过的路由，数据包才能到达目的地。Johnson

等人提出了动态源路由（DSR），发送方泛洪一个路由请求探针（RREQ）来寻找目的地的路由。中间节点转发这个探针，并将自己的标识符加入到探针中。首先达到目的地的探针假定达到了最优路径。DSR 将会在后续通信中使用这个路径。

表驱动的路由协议在本地存储路由信息。节点通过反应性或者周期性交换路由消息，来更新网络中链路状态的信息。当一个节点尝试向另外一个节点发送数据包时，它会在路由表中查询目的节点的路由信息。它会将数据包转发给路由中合适的邻居，后者再查询自己的路由表，然后转发数据包。中间节点经常要在两个同样适合向目的节点转发数据包的邻居中做出选择。不考虑其中一个节点不适合立即传输的可能性，链路中的连接往往是随机断开的。我们认为任播对此很有帮助，MAC 层可以在此场景下做出明智的选择，在性能上体现出优势。我们将在本文中依据表驱动协议讨论 MAC 层任播的细节。由源路由引起的问题将在第五章中讨论。

Roy 等人提出了最大区域不相交路由的概念。发送方基于之前的流量状况选择最大程度地避开拥挤的区域。我们任播的思路不同于我们基于较短时间内变化因素的转发决策。路由从仅仅提供一组可以接受的选项（并非所有的选项都是最优的）。然后 MAC 层将根据瞬时网络状况来决定下一跳。Pursley 等人提出了使用“解码器辅助信息”来协助转发决策。通过观察从一系列已知的发送信号中正确接收的信号数量，接收器可能能否估计出链路的可靠性。作者提出了一种度量方式，阻力——表示链路的质量。通过这种度量方式，节点可以在两条出战链路中选择较低阻力的链路传输数据包。尽管这种方案处理了信道波动的变化，但是并没有考虑 MAC 层相关的因素。MAC 层任播适应多种 MAC 层协议约束，这点将在本文下文论述。

Larsson 提出了“选择分集转发”，发送器将多播地址（或者地址列表）包含在了数据包中。在多播地址（或者地址列表）中的邻居用 ACK 包串行答复该数据包。发送器根据路由层的规则和从 data-ACK 交换中了解到的当前链路状况来选择其中的一个邻居。“forwarding order”被发送到指定的邻居，然后邻居需要将数据包继续转发。选中的邻居需要使用“forwarding order ACK”响应“forwarding order”。显然，等待在初始化“forwarding order”之前的响应是无用

的。Jain 等人对上述协议提出了一种改进。作者提出要根据优先级指定地址列表（和上述类似）。该协议要求地址列表中的所有节点都按照优先级响应，最高优先级者最先相应。在接收到第一个相应之后（并不一定是最高优先级的节点），发送器立刻向这个节点发送数据包。这样减少了在传输数据之前等待多个响应的开销。与 Larsson 不同的是，优先级必须提前声明，并且不包含瞬时链路状况信息。另外，指定首选项和多地址会增加数据包的大小，从而产生更高的控制开销。

尽管本质上相似，MAC 层任播仍然与现有研究有所区分。关键点在于决策产生的基础。可以看到，大部分以前的方案都用某种形式探测信道，然后根据显式或者隐式的反馈选择合适的邻居。我们认为，等待反馈并不是必须的，因为 MAC 层可能已经有必要的信息。例如（后面将会展示更多的例子），我们观察到 MAC 层可能在给定的时间点知道允许的发射功率水平。以前的方案可能会探测到不允许的发射功率水平的信道，然后获得负反馈，再收敛到允许的发射功率水平。显然，使用 MAC 层上可用的信息在这种情况下会有帮助。我们接下来讨论一些无限媒介访问控制协议（MAC）。

我们假定读者熟悉 IEEE 802.11 协议。简单来说，当使用 802.11 时，在数据通信之前，先进行发送请求（RTS）和清楚发送请求（CTS）。侦听到 RTS/CTS 的节点在已经建立的数据通信持续时间内推迟自己的数据传输。一旦数据包发送了，接收方会回复一个 ACK 包已确认成功接收。

最近的一些方案对 802.11 做了一些调整。尽管如此，该协议的基本思想不变。近来，随着天线技术的进步，已经提出了几种在 MAC 层使用定向天线的协议。使用定向天线的思路可以总结如下。由于具备向期望防线发送信号的能力，大多数协议建议使用定向和全向 RTS/CTS/DATA 和/或者 ACK 的组合。信道空间的重用程度将会随着干扰的减少而增加。定向 NAV 的概念使得节点可以启动新的传输，而不会干扰正在进行的通信。由于天线波束更高的增益，扩大范围是一项额外的益处，源节点和目的节点可以形成跳数更少的路由。定向天线虽然前景很好，但同时也带来一些问题。邻居发现，新型隐藏终端，耳聋是定向通信产生的一些问题。我们认为任播在使用定向天线时可以起到一定的帮助。

MAC 协议中的多用户分集研究也成为人们的研究热点。Qin 等人提出了可感知信道的 ALOHA 协议，该协议瞬时信道状况来调度传输。该协议使用分布式方法，要求节点在其本地信道状况良好的情况下进行传输。Tsatsanis 等人提出“网络辅助分集协议”，探讨了利用损坏的数据包的可能性。换句话说，作者提出了允许多个发送器多次（同步）碰撞的思路。然后接收器使用已知的信号处理算法将数据包从损坏的数据包中分离出来。DeCouto 等人最近提出了一种 ETX 度量标准来支持具有较少损耗和重传特征的路径。换句话说，在制定路由决策时，网络层会考虑在 MAC 层的可用信息。但是，一旦决定了一条路由，便会使用该路由，而不管瞬时信道状况可能的变化。Yarvis 等人在传感器网络的背景下也提出了相似的想法。本文的关键思想和 DeCouto、Yarvis 等人的思想有些相反。MAC 层需要网络层提供它认为合适的一组路由。和上述方法不同的是，MAC 层最终决定在这个时刻最合适的邻居。

三、MAC 层任播

可以将 MAC 层任播视为对现有 MAC 和路由协议的增强。在本文的剩余部分，我们将未通过任播功能“增强”的路由协议称为“基准”。图 1 展示了实现 MAC 层任播的一种可能的体系结构。本节在通用 MAC 和路由协议的背景下讨论了 MAC 层任播的框架。我们还提出了一个简单的变体，称为 Ordered Anycasting2。之后，我们将会看到一些任播的应用，并讨论无线自组网中的权衡问题。

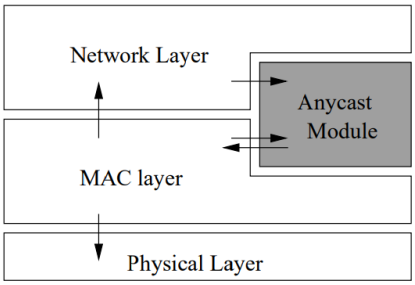

Fig. 1. An Anycasting Framework.

任播框架需要“基准”路由协议在可能的情况下发现并维护每个流的多个路由。显然，所有被发现的路由可能并不都是好的。当一个数据包到达网络层，路由协议将查询其路由状态，来决定可以有效到达数据包目的地的路由。从这些可用路由中，路由协议选择包含了 K 个可能被认为是最佳路由的子集。

现在，网络层形成了我们称为任播组的内容。任播组包含了在 K 个路由上的下一跳的邻居的集合。例如，对于目的地是节点 D 的数据包，由图 2 中的节点 S 所处的网络层指定的任播组可以是集合(A,X)。然后，网络层将数据包和和任播组传递到 MAC 层。在接收到数据包和任播组之后，MAC 层必须选择一个合适的邻居并尝试向其传输。瞬时网络状况可能对决策产生重大影响。下一章介绍了任播的一些可能的应用，并说明了如何设计邻居选择策略。但是首先，我们先对任播进行简单的修改，称之为有序任播。

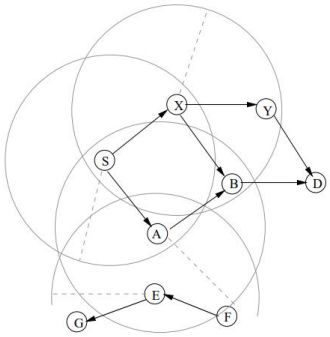

Fig. 2. An example scenario illustrating the possibility of anycasting

有序任播

节点的路由层可能会发现到特定目标的多个路由。所有的路由可能不都是最佳的。例如，如果路由 R1 和 R2 同样好（例如就跳数而言），并且如果两者均好于 R3，那么只有当通过 R1 或者 R2 通信不可行的时候，网络层才会选择使用 R3。有序任播是为了实现这个目标的任播的一种变体。路由层按照优先级对任播组的成员进行排名。当且仅当其他优先级更高的节点被认为“不可用时”，MAC 才会尝试与该节点通信。