# 摘要

无线传感器网络是一门新兴技术，是传感器技术、计算机网络技术、微机电技术系统发展的产物。无线传感器网络各方面研究进展迅速，但是能量问题仍然是无线传感器网络研究的瓶颈。由于无线传感器网络节点能量十分有限，如何高效使用能量来最大化网络生命周期是传感器网络面临的首要挑战。设计能够延长整个网络生存期的路由协议成为了传感器网络研究的重要目标。

本文简要分析了无线传感器网络的协议栈模型，无线传感器网络路由协议的设计目标和现有的几种典型的无线传感器网络的路由协议。然后，重点从软件节能角度对无线传感器网络路由协议进行了研究。为了解决低功率自适应成簇算法（LEACH）的缺陷和不足，提出了一种基于簇融合的无线传感器网络路由协议。这种改进型协议充分利用汇聚节点的高性能，使得整个网络中的分簇数目处于最优状态，并且通过簇的融合来保持这种最优状态。簇头之间可以进行多跳通信，避免了簇头节点直接向汇聚节点发送数据能量消耗高的问题。最后，基于 MATLAB 对 LEACH 协议和改进型协议进行了仿真实验；在 TinyOS 操作系统上用 nesC 语言实现了基于簇的改进型协议并进行了功能测试。

对比分析仿真结果发现，改进型协议在能量消耗，节点死亡率和网络生存期方面有较好表现。测试结果表明基于簇融合的改进型协议的 nesC 实现满足设计要求。

**关键词：** 无线传感器网络　LEACH　簇融合　TinyOS

# Abstract

Wireless sensor network is a new technology. It appeared after sensor technology, computer technology and MEMS technology developed. Almost all aspects of Wireless sensor network make rapid progress, but the energy remains a bottleneck in WSN research. As the energy of the wireless sensor network node is very limited, how to efficiently use the energy to maximize the lifetime of the network is the primary challenge of WSN. Designing protocol to extend the lifetime of the network has become an important goal of WSN.

In this dissertation, the stack model of WSN are introduced briefly,and we describe the wireless sensor network routing protocol design goals, technical challenges and summarize the characteristic of routing protocols. Then, this thesis mainly make a research on software to save energy. In order to overcome the shortcomings and deficiencies of LEACH, we creat a cluster-merge-based routing protocol for wireless sensor network. This new protocol takes full advantage of the high-performance of sink and ensure the number of cluster head is optimum. Through the integration of cluster, the protocol can maintain this optimal state. It is between the cluster heads using multi-hop communications that overcome the weakness that sending data to sink consumes more energy. Finally, we simulate LEACH and the cluster-merge-based routing protocol in MATLAB and implement the cluster-merge-based routing protocol with nesC in TinyOS.

Comparative analysis of simulation results showed that the improved procotol has better performance in energy consumption, mortality and network lifetime. The test results show that the nesC implementation of the cluster-merge-based routing protocol meets the design requirements totally.

**Keyword: WSN LEACH cluster-merge TinyOS**

# 西安电子科技大学
# 学位论文独创性（或创新性）声明

秉承学校严谨的学风和优良的科学道德，本人声明所呈交的论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果；也不包含为获得西安电子科技大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中做了明确的说明并表示了谢意。

申请学位论文与资料若有不实之处，本人承担一切的法律责任。

本人签名：李冰　　日期 2010.1.15

# 西安电子科技大学
# 关于论文使用授权的说明

本人完全了解西安电子科技大学有关保留和使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属西安电子科技大学。学校有权保留送交论文的复印件，允许查阅和借阅论文；学校可以公布论文的全部或部分内容，可以允许采用影印、缩印或其它复制手段保存论文。同时本人保证，毕业后结合学位论文研究课题再撰写的文章一律署名单位为西安电子科技大学。

（保密的论文在解密后遵守此规定）

本学位论文属于保密，在___年解密后适用本授权书。

本人签名：李冰　　日期 2010.1.15

导师签名：　　日期 2010.1.15

# 第一章 绪论

## 1.1 研究的背景与意义

无线传感器网络综合了传感器技术、分布式信息处理技术，无线通信技术和MEMS 技术，从而形成了国际上备受关注的多学科高度交叉的新兴前沿热点研究领域。传感器网络的出现使得我们能够通过在特定的区域内部署大量的廉价的传感器节点实时监测环境和各种对象，并利用传感器网络对采集和感知的信息进行初步处理，然后通过无线通信的方式将信息传给汇聚节点。传感器网络的出现极大地扩展了现有网络的功能和人类认识世界的能力。

传感器网络具有在任何时间、地点和环境条件下都可以获得大量可靠信息的能力，这使得它在许多重要领域如军事领域、工业控制和监测、医疗护理、环境监测及保护、抢险救灾等具有相当广阔的应用前景。这项对未来会产生巨大影响的技术在学术界和工业界都引起了高度重视。

国外的学术界、工业界以及政府部门早已启动了一系列与传感器网络相关的研究计划。几乎美国的所有著名大学都有从事传感器网络研究的科研小组。英国、日本、加拿大等国家的研究机构也逐渐开展了对无线传感器网络的研究工作。各国际大型 IT 公司如 Intel、Microsoft、欧姆龙、惠普等均为研发传感器网络的相关技术、产品和标准投入了大量的人力、财力。至今为止，已开发出了 Berkeley Motes、Berkeley Piconodes、Sensoria WINS、SmartMesh Dust Mote 以及 Intel Xscale Nodes 等无线传感器网络节点。美国的自然基金会委员会更是早在 2003 年就制定了传感器研究计划，投资了三千四百万美元用于 WSN 基础理论研究。以美国国防部为首的各军事部门也设立了一系列项目，如 WINS 计划、SmartDust 计划、SensIt 计划、SeaWeb 计划等来支撑军事传感器网络的研究。

在国内，我国政府部门也对传感器网络给予了高度重视。在国家“十一五”科技发展规划中，传感器网络已被列为重点发展的产业并作为一种基础网络来重点研究和建设。在“中国未来 20 年技术预见研究”中的 157 个技术课题中总共有 7 项技术科题是直接论述传感器网络的。国家发展改革委员会的下一代互联网示范工程中，也部署了无线传感器网络相关课题。国家自然科学基金委员会也已经审批了和无线传感器网络相关的重点课题和面上课题，例如“863”项目“基于无线传感器网络的海洋立体监测研究”，“973”项目“无线传感器网络的基础理论及关键技术研究”，还有国家自然科学基金项目“无线传感器网络在冶金工业监测中的应用研究”等等。正如《国家中长期科学和技术发展纲要规划》中明确指出的，我国要把传感器网络及智能信息处理作为发展的一个优先主题。

在早期的研究中，人们普遍认为 Ad boc 网络协议稍加修改就能够用于无线传

感器网络，因为无线传感器网络在概念上来说也是一种分布式的自组织网络，邻居发现和路由转发都是自动完成的，这和 Ad hoc 网络相似。随着对无线传感器网络认识的不断深入，人们认识到无线传感器网络具有自己的特点。这些特点决定了无线传感器网络协议需要重新设计。例如传感器网络节点的能量资源，计算资源非常有限，这就决定了无线传感器网络的各层协议设计都要充分考虑能源的有效性，对物理层和链路层要采用系统级的低功耗设计，在网络层上要通过建立能源有效性路径，形成可靠的数据转发机制，通过协议最大化网络的生命周期。无线传感器网络中的节点非常密集，网络节点出现故障的可能性很高，并且节点的安置环境特殊，这些特点要求无线传感器网络协议需要具有较强的容错性，使网络节点在发生故障时，可以利用节点容易获得的网络信息，重新恢复路由。

路由协议是网络传输的保证，因此网络协议是无线传感器网络研究中的关键问题。随着对传感器网络认识的不断加深，人们提出了许多新的适合无线传感器网络使用的路由协议。但是这些协议也存在着大量问题，并不能完全适应无线传感器网络的应用场景，因此，我们仍然需要对无线传感器网络协议进行更加深入的研究。

## 1.2 无线传感器网络协议的研究现状

至今，为了适应无线传感器网络能量受限这一特征研究人员共提出了三类路由机制：数据传输区域限定路由协议、代价估计路由协议和基于网络组织结构的路由协议。虽然对这些协议进行很多研究，但它们中仍然存在着未能解决的无线传感器网络的瓶颈问题，这说明为了提供更好的数据传送服务，在无线传感器网络路由领域仍有许多工作需要完成。下面通过简要介绍一些研究成果来说明该领域的研究现状。

在限定数据传送区域的路由协议方面的主要研究成果如下：Wen-Hwa Liao 等人在 GRID 中提出了利用目标节点和事件节点位置确定传送区域的策略，可分为扇形区域法、双扇形区域法、矩形区域法及条形区域法等。GRID 协议将限定区域划分为网络形状并通过每个网络内的网关节点负责采集数据。采用此方法的优点是可以提高路由稳定性、消除广播风暴；缺点是有障碍物存在时将无法正常传送数据，节点定位的准确性影响路由性能。S.Nikoletseas 等人提出了 PFR，它利用节点位置与目标节点和事件节点间的连线形成角度计算数据传送概率，然后使用给定阈值将数据传送区域限定在目标节点和事件节点之间。此方法优点是使用调节通讯半径的方法避开障碍物，通过限定数据传送范围减少网络能量消耗；缺点是数据在汇聚节点和事件节点间的狭长区域内传输导致消息冲突增多，延迟增大。另外对于不同网络规模的情况中，改进型的 SW-PFR 及 H-TEEN 算法在性能上有

所提高，但是能量问题的瓶颈并没有突破。

基于代价估计的路由协议的主要研究成果如下：USC 的 Estrin 等人提出了 GEAR 协议。GEAR 是基于代价估计的路由机制，并通过捎带机制获取实际路由代价以达到优化传输路径的目的。该协议的缺点是出现路由空洞时会降低路由效率，不适用于节点移动性强的环境。MIT 的研究人员提出了以节点间距离作为路由代价估计的方法，并利用节点无线广播优势来计算最短路径，进而使得数据传输能量消耗最小，该方法的缺点是不适用于只了解局部拓扑信息的网络，计算的复杂度高。路易斯安那州的学者提出了 EBRP 协议，在该协议中通过定义估计路由代价作为节点剩余能量的级别，利用剩余能量将网络划分为若干区域，通过在高能量级别的区域内传送数据达到能量均衡的目的。该协议的缺点是在形成能量等级时节点之间通信频繁而消耗能量。

基于网络组织结构的路由协议主要研究成果如下：MIT 的 Heinzelman 等人首次提出层次型路由协议 LEACH，以后的很多层次型路由协议如 TEEN、PEGASIS 等都是从 LEACH 协议基础上改进而来。LEACH 协议通过采用簇结构来达到减少传输能量消耗的目的。UCLA 学者 Fan Ye 等提出了两层分发模型 TTDD，根据事件发生点构造网络,模型中的低层负责寻找分发查询点而高层负责数据传送。该模型的优点是为连续查询提供优质服务并避免了查询请求的多次查询请求发送；缺点是事件节点频繁移动时网络将消耗大部分能量以构造网络，事件节点在某地较长时间将导致节点间的能量消耗不均衡。卡内基梅隆大学的 Newsome 等人提出 GEM 路由协议，协议中使用虚拟极坐标的方法使网络节点在逻辑上形成一个带环树，树的根节点是汇聚节点。该协议的优点是为无线传感器网络提供了一种不依赖于节点精确位置信息的路由机制；该协议的缺点是网络拓扑结构发生变化时树的调整过程复杂，能量消耗大。加州大学伯克利分校的 Qing Fang 等人提出了 RoamHBA 路由协议，协议中将移动目标逻辑组成 hub，通过 backbone 来获取目标位置以进行贪心路由算法。该算法的优点是为移动节点信息的查询提供有效路由机制；缺点是在节点频繁移动需要建立到 backbone 的新的路线，若路由长时间在 backbone 中数据传输将导致 backbone 重建而消耗能量。

## 1.3 本文的主要内容及论文安排

能量是无线传感器网络中非常珍贵的资源，路由协议的设计要尽可能减少节点能量的消耗，基于簇的路由协议是一类能源有效的路由协议。本文在对大量不同种类的传感器网络路由协议尤其是基于簇的层次路由协议研究的基础上提出了基于簇融合的路由协议，并且通过仿真工具验证了基于簇融合的改进型协议的有效性，在 TinyOS 上完成了基于簇融合协议 nesC 代码实现，并进行了相关测试。

论文的各章节的安排如下：

第一章，绪论。首先介绍了无线传感器网络的研究的背景和意义，然后通过一系列研究成果概述了无线传感器网络的研究现状，最后明确了文章的主要贡献和章节的安排。

第二章，无线传感器网络概述。首先分析了无线传感器网络的协议栈模型，然后介绍了无线传感器网络路由协议的设计目标，接着分别简要讨论了几种有代表性的路由协议，最后对 LEACH 协议进行了深入研究，详细描述了 LEACH 协议的典型思想、能量模型、工作流程和优缺点。

第三章，基于簇融合的改进型路由协议。针对 LEACH 协议的不足之处提出了基于簇融合的改进型路由协议，并详细讨论了基于簇融合的改进型协议的运行过程，该过程主要包含节点位置信息获取，计算最优簇头数目，生成节点链，簇头选择，节点退出及簇融合，节点丢失查找和簇间通信等子过程。详细描述了采用 MATLAB 对 LEACH 协议和基于簇融合的改进型路由协议进行仿真的流程，主要最后对试验结果进行对比分析，验证改进型协议的有效性。

第四章，协议代码实现。首先对操作系统 TinyOS 和编程语言 nesC 进行了介绍，然后提出了基于簇融合的改进型协议在 TinyOS 操作系统上的数据结构设计、模块和接口设计，最后对改进型协议的实现代码的测试。测试结果表明传感器节点可以正常运行，满足改进型协议的设计要求。

第五章，结论和展望。对本文进行总结，回顾了主要工作，提出了工作中的不足以及下一步的研究内容和方向。

# 第二章 无线传感器网络协议概述

## 2.1 无线传感器网络的协议栈模型

随着无线传感器网络的研究逐渐深入，传感器节点上的协议栈模型也在细化。图 2.1 是一个早期的协议栈模型，这个协议栈包括物理层，提供简单但健壮的信号调制和无线接收技术；数据链路层，负责数据组帧，帧检测，媒质访问和差错控制；网络层，负责路由建立和路由选择；传输层，负责数据流的传输控制，保证通信服务的质量；应用层，包括一系列的基于监测任务的应用软件。另外，协议栈中还包括能量管理平台，管理传感器节点的能源使用，在各个协议层中都需考虑能量节省；移动管理平台，主要负责检测、注册传感器节点的移动，维护到汇聚节点的路由，使得传感器节点能够动态跟踪邻居节点的位置；任务管理平台，在一个给定的区域内平衡和调度监测任务。在这些管理平台的协同工作下，节点能够以能源高效利用为前提正常工作。

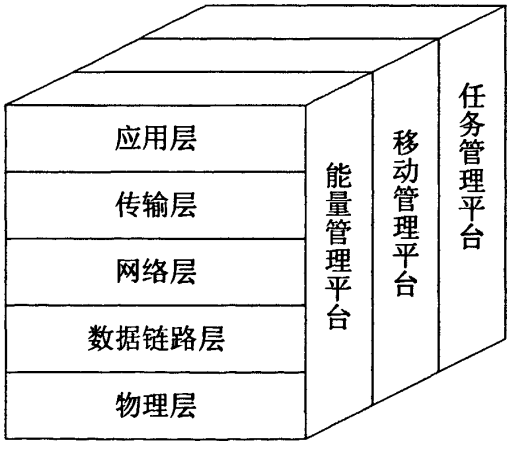


图 2.1 无线传感器网络早期协议栈模型

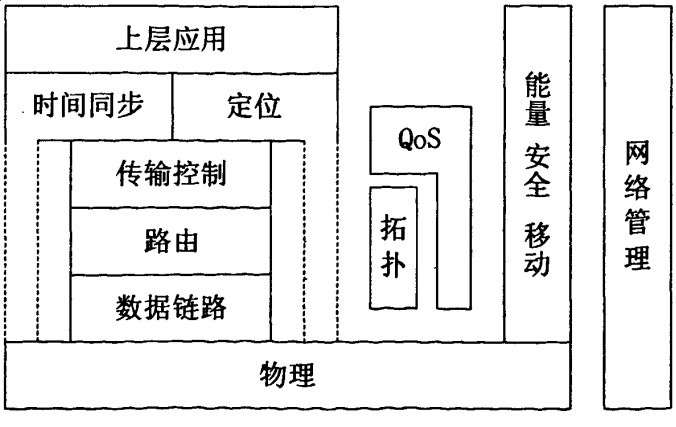


图 2.2 无线传感器网络改进型协议栈模型

更加细化的改进型协议栈模型如图 2.2 所示，它更适合表示无线传感器网络的协议栈。定位和时间同步子层既要依赖于数据传输通道进行协作定位和时间同步协商，又要为各网络协议层提供信息支持，如时分复用的 MAC 协议，基于地理位置的路由协议等很多传感器网络协议都需要定位和同步信息等，所以它们在协议栈中的位置比较特殊，它们渗透到了传输控制、路由和数据链路层中。像图 2.2 中右侧的部分机制在图 2.1 中没有明确表示，而是融入了各层协议当中，用以优化和管理协议的流程。但是，另一部分是独立在协议外层的，需要通过各种收集和配置接口对相应的机制进行配置和监控。例如，QoS 管理需在各协议层设计队列管理、优先级机制或者带宽预留等机制，并对特定应用的数据给予特别处理；能量管理需在每个协议层中都要增加能量控制代码，并提供给操作系统进行能量管理的分配策略；网络管理则要求协议各层嵌入各种信息接口，并定时收集协议运行状态和流量信息，协调控制网络中各个协议组件的运行；拓扑控制利用物理层、链路层或路由层完成拓扑生成，反过来又为它们提供基础信息支持，优化 MAC 协议和路由协议的协议过程，提高协议效率，减少网络能量消耗。

## 2.2 无线传感器网络协议的设计目标和技术挑战

网络层协议的主要设计目标是在提供高性能的服务质量和高效利用网络带宽的基础上将数据分组从源节点通过网络转发到目的节点。一方面，网络路由协议需要寻找源节点和目的节点之间的优化路径并将数据分组沿着优化路径正确转发；另一方面，它应该能够均衡网络流量，提高整个网络的利用率，避免产生通信拥塞。研究人员发现好的路由协议应该适应资源匮乏的限制，通常具有以下特点：

针对节点能量高度受限，路由算法必须高效利用能量以便延长网络生存时间；针对节点数据的相关性，包头开销大，节点能量有限等特点，路由算法需要采用数据融合和数据过滤等技术，尽量降低通信量；针对节点移动性不大的特点，路由算法不需维护节点的移动性；针对节点因所处位置及承担的责任不同而导致负载不平衡的特点，路由算法需采用通信量负载均衡技术；针对网络相对封闭、不提供计算等特点，路由算法只在汇聚节点考虑与其它网络互联；针对网络节点不常编址的特点，路由算法需采用基于数据或基于位置的通信机制；针对节点易失效的特点，路由算法采用多路径机制。

此外，应用于无线传感器网络的路由协议还必须考虑到无线传感器网络具有分布范围广，自组织，与应用相关的特点。通过这些分析可以得知无线传感器网络协议还应具有以下设计目标：

能量高效：传感器节点的资源有限，传感器网络的路由机制要能够简单高效

的选择出能量消耗小的传输路径，并且要从整个网络的角度出发，均衡网络的能量消耗，尽量延迟网络中的节点死亡，延长网络寿命。

可扩展性：无线传感器网络中，节点分布密度差异很大，节点会由于某种原因失效，也可能为了提高传感器网络的监测精度而部署新节点，另外某些传感器网络节点是可以移动的，这些情况都会导致网络拓扑结构发生变化。无线传感器网络的路由协议应该具有可扩展性以适应传感器网络拓扑的动态变化。

鲁棒性：无线传感器网络路由协议应该具有一定的容错能力以应对由于能量用尽或环境因素造成的传感器节点失效、环境因素导致的无线链路通信质量降低以及无线链路本身的缺陷导致的传输不可靠性。

快速收敛性：无线传感器网络路由协议应该能快速收敛，能够在节点有限的能量和通信带宽的基础上来降低通信协议开销，提高信息的传输效率，以适应网络拓扑的动态变化。

无线传感器网络设计的一个主要目标就是在进行数据采集和传输的同时，采用实时高效的能量管理策略，尽量延长网络的寿命，防止网络性能下降。无线传感器网络路由协议对实现这一目标起着关键作用。从传感器网络的固有特征出发，充分考虑限制其投入使用的节点能量有限，计算能力差，通信带宽窄等因素，人们发现如果想保证网络的有效通信，尽量延长网络寿命，就必须解决以下技术问题。

数据流传输模型。数据传输模型可以分为连续型、事件驱动型、查寻驱动型和混合型四种。在设计协议的过程中，需要根据具体应用需求和对数据的实时性要求来进行选择。周期性数据检查的应用系统中通常采用连续流数据传输模型。在对时间要求紧迫的应用系统中通常采用事件驱动或查询驱动数据传输模型。在特定事件出现或者收到汇聚节点的请求时，传感器节点的特定属性值就发生了变化，节点会立刻做出反应，将采集到的数据发给汇聚节点。在有些应用中需要前三种数据传输模型以组合的方式存在，这就构成了混合型数据传输模型。数据流传输模型对路由协议设计方法及性能诸如能耗和路由稳定性等方面有重要影响。

节点部署方式。根据具体应用场合的不同，无线传感器节点部署方式可以是确定的或是随机的。对于不同的节点部署方式应该采用不同的协议。对于确定型的网络，数据的路由路径已经确定，并且用户可以参与节点资源管理，掌握各节点的最新状况，路由设计相对简单，对于随机部署的网络，节点之间应采用自组织方式形成路由，建立网络。这需要采用如优化聚类等新的方法来应对节点分布的不均衡，以确保网络的互联性和网络操作的高效。可见，传感器节点部署方式对路由算法的设计是有很大影响的。

容错性。由于能量耗尽、环境干扰、自然或人为损坏的原因，传感器的一些节点可能会失效或者通信中断。无线传感器网络协议应该能够保证在这些节点永

久或暂时失效的时候整个传感器网络所承担的任务都不应该受到影响，其完成任务的质量不会明显下降。如果一些节点失效后，MAC 协议和路由协议必须能够适应这一变化，及时迅速的建立新的传输路径，便于数据快速路由到汇聚节点。这需要在路由算法上引入新的设计元素，保存必要的信息冗余来保证路由的稳定性，以建立具有强容错性的网络。

拓扑动态性。无线传感器网络协议需要适应网络的动态拓扑变化。无线传感器网络中导致拓扑结构动态变化的原因主要有两个：一是由于各种原因导致的传感器节点的失效或者通信中断；二是在一些具体的应用中汇聚节点或某些传感器节点具有移动性。为了构造一条稳定数据传输的通路，路由设计将考虑更多问题。首先，路由稳定性成为了路由协议设计的一个重要问题；其次，必须考虑拓扑变化所引起的能量消耗和带宽利用率降低；最后，观测对象本身也可以是移动的，这会带来新的问题。

能耗问题。无线传感器网络协议需要考虑每个节点上能量的有限性。在无线通信环境下，每个节点都在进行着数据采集和路由数据包的工作，在这个过程中节点上的能源必将耗尽，而能源耗尽会导致节点失效，这会引起明显的拓扑变化，从而使得网络需要重组，某些数据包需要重发，这就又增加了额外的消耗。所以，在设计无线传感器路由算法时必须重点考虑能耗问题，在各节点上采取必要的有关通信和计算的能量保护技术。

节点异构性。无线传感器网络协议需要考虑网络中的节点是否存在异构性。现在大多数研究中都假定网络中的所有节点都是同构的。但是，在某些实际应用中，传感器节点可能是异构的。如果网络节点存在异构，我们应充分利用这些节点的能量、计算能力和传输功率的方面的差异。例如，在层次型的路由算法中，可以使得能量和带宽较大、传输能力和存储能力强的节点充当簇头的机会比一般节点高，让这些优质节点完成将数据传送到汇聚节点的任务，这样可以延长传感器网络的生存寿命。因此，在设计路由协议时应充分考虑实际应用中网络节点是否存在异构性，并且根据具体情况调整路由策略。

传输媒介。无线传感器各个节点之间所需的传输带宽较低，各节点之间通过无线媒介进行通信。与此相关的是 MAC 层协议的设计问题，MAC 层协议的设计对整个节点的能耗高低有很大影响。例如相对于 CSMA 协议而言，TDMA 协议更加适用于无线传感器网络，因为该协议能节约更多的能量。在进行路由协议设计的同时，也应该同时考虑传输媒质和 MAC 层协议对整体性能的影响。

网络连通性。无线传感器网络协议设计需要考虑如何保证传感器节点之间具有较高的互联性。因为孤立的传感器网络节点即使没有死亡或故障也是无法发挥任何作用的。无线传感器网络节点密度高这一特征可以防止它们被孤立，但是由于网络运行过程中一些节点会由于能量耗尽而失效导致节点数量缩减，从而使得

网络中的节点密度降低甚至出现监测盲区。另外，节点的随机性分布也会影响网络的连通性。因此，路由协议设计时应充分考虑网络连通性的需求，让每个节点都尽可能的具有通往汇聚节点的通路。

伸缩性。无线传感器网络协议的设计需要考虑网络的伸缩性。在具体应用中，无线传感器网络的规模各不相同，差别很大。无线传感器网络协议应该具有足够的可伸缩性以适应这些具体的网络需求。另外，在不需要提供精确服务的时候可以只利用一部分节点负责采集数据，其余节点处于休眠的状态以节约能量。

覆盖率。覆盖率与网络连通性和节点部署方式密切相关，指的是在单位面积内一个或多个传感器节点所能监测到的区域情况。每一个特定的传感器节点所能检测的精度和范围都是有限的，它只能覆盖其周围环境的一个有限的物理区域。在路由设计时，区域覆盖率是一个需要考虑的重要参数。

另外还要考虑数据聚合、流量模式、组播路由、跨层协议设计、服务质量和安全性等问题。传感器节点间所监测到的数据是有冗余的，路由协议必须对多个相邻节点的数据进行融合，以减少数据传输量，使数据传输达到最优化，从而节省能量；无线传感器网络中的数据具有集中收集、多跳传输和多对一传输的特点，这些特点会导致严重的包碰撞、包丢失和网络拥塞，并且会导致出现能量消耗的热点，从而导致某些节点甚至整个网络生命周期缩短，在设计路由算法的过程中需要对这一问题加以解决；组播路由本身就是网络的一个难题，在无线传感器网络的环境内进行组播路由更具有挑战性；为了达到路由算法的最优化，可以将链路层协议和网络层甚至应用层协议结合起来进行跨层路由算法设计。在设计路由协议的过程中，需要在能耗和服务质量之间进行有效调控，传感器网络的最主要考虑因素是能耗问题，因此为了能够节约节点能耗和延长网络生存时间，用户可能允许适当降低输出结果质量；路由算法的设计需要考虑安全机制，这点在军事应用中特别重要，路由算法没有安全机制的保护极易受到外界的干扰和威胁，这会影响数据的正确性和完整性。

通过对目前各种无线传感器路由算法的分析和研究，可以得出如下策略来应对无线传感器网络路由协议设计的各种挑战：在协议中采取措施尽量减少数据通信量，因为数据通信在传感器网络中最为耗能，可能的措施例如通过过滤机制来抑制不必要的数据上传，采用数据融合机制去掉冗余信息；在协议中采取更加灵活的路由策略，使各个节点分担数据的传输，这样可以平衡各个节点之间的能量消耗，增加整个网络的生存期；路由协议在控制协议开销的前提下，尽量支持节点移动和拓扑感知，以提高对网络拓扑变化的适应性；路由协议可以根据易于获得的信息来选择路由，这可以保障在无线传感器网络节点在发生故障时路由可以尽快得到恢复，还可以采用多路径传输数据来提高可靠性；在进行协议设计时，可以考虑通过对链路层、网络层甚至应用层的协议进行跨层优化来达到使整个网

络达到最优化的目的；可扩展性是衡量网络协议性能的重要指标，不论是平面型的还是层次型的无线传感器网络协议都应通过提高可扩展性来适应网络规模的扩大或缩小；传感器网络协议应该可以通过纠错码、设置竞争门槛、使用短帧策略等方法应对链路层的碰撞攻击、耗尽攻击和非公平竞争攻击，通过使用冗余路径，探测机制、认证机制、冗余机制等来应对网络层的汇聚贪婪破坏、汇聚节点攻击，方向误导攻击等；另外，由于资源限制和动态拓扑等因素的影响，QoS 路由建立有一定挑战，但是 QoS 路由对于一些视频及图像传感等实时应用非常必要，因此在与图像视频等应用相关的无线路由协议中添加 QoS 路由是非常必要的。

## 2.3 几种典型的无线传感器网络路由协议

针对不同的传感器网络应用，研究人员提出了不同的路由协议。本文根据网络管理的逻辑结构将路由协议分为平面路由协议和层次路由协议平面结构是指网络中各节点没有特殊功能的节点，没有引入分层管理机制。平面结构路由的优点是由于网络中的节点在路由功能上地位相同，整个网络流量均匀地分散在网络中并且路由算法易于实现。与平面路由相对应的是层次结构的路由协议。层次路由协议采用簇的概念对传感器节点进行层次划分。层次路由协议包括成簇协议、簇维护协议、簇内路由协议和簇间路由协议四个部分。每个簇由若干个相邻的节点构成，由一个节点作为簇头节点。簇和簇之间可以通过簇头节点或者其它簇成员节点进行通信。层次路由协议具有很多优点，例如簇成员节点将数据在簇头处进行数据融合处理后再进行转发，减少了转发数据量，节省了网络能量；普通簇成员的节点功能简单，并且只需要很少的路由信息来获得路由表，这样网络中的控制信息的数量就大大减少，从而减少了通信量；分簇完成后，向上一层网络进行长距离数据转发的任务由簇头负责，普通的簇成员节点就可以在大部分时间内关闭通信模块，从而节省了能量；分簇的拓扑结构更适合大规模的网络应用，因为它具有较好的可扩展性，能够对系统变化作出快速反应，更有利于分布式算法的应用。

无线传感器网络的路由协议是目前国内外研究的热点。目前已有的协议分别适用于不同的环境，在性能评价指标上各有侧重。下面是对现有的一些具有代表性的路由协议的进行简要介绍。

泛洪算法和闲聊算法是网络中的相对简单的算法，与其它路由算法相比，它们不需要维护路由信息，因此对网络拓扑的依赖性更小，但是扩展性很差。泛洪算法中源节点向其所有邻居节点广播数据分组，其它节点在接到数据分组后再向自己的所有邻居节点广播这个数据分组，这个过程重复进行，直到数据包到达目的节点或到达最大跳数。泛洪算法所具有的主要优点是：实现简单；健壮性强；

不需要消耗计算资源获得网络拓扑信息和运行复杂的路由发现算法。泛洪算法主要缺点是：内爆，节点几乎同时从邻居节点收到多份相同的数据；转发数据交迭，即节点先后收到监控同一区域的多个节点发送的几乎相同的数据；盲目利用现有资源，节点在任何情况下都转发数据而不考虑自身资源限制。这些缺点会大大地降低网络生命周期。闲聊算法对泛洪算法进行了改进，接收到数据包的节点在自己的邻居节点中随机的选择一个没有发送数据的，把数据包传送给它。持续这个过程，直到数据包到达目的节点。闲聊算法随机选取节点并对数据进行有目的的传播，这实际上是以延长信息传播的代价来避免信息爆炸的问题。

定向扩散协议是一个基于数据的、查询驱动的按需路由协议，是以数据为中心的路由算法的一个重要里程碑。该协议用属性值对为数据命名，主要包括以下几个步骤：兴趣发布、梯度建立、数据传输和路径加强和修复。兴趣发布是为了建立路由汇聚节点对外广播属性列表、时间间隔等信息的查询请求；梯度建立是指在中间节点收到请求后，按需对各个请求进行缓存与合并并根据请求计算出梯度，梯度包含数据传输率、下一跳等信息；数据传输是指当梯度建立完成后，目标区域内节点按要求启动监测任务，并将数据通过梯度最大的那条路径周期性的上报，中间节点可对数据进行缓存与聚合。

定向扩散算法的优点是，该算法健壮性好；使用查询驱动机制按需建立路由从而避免了保存全网信息；基站节点根据实际情况可以采用增强或减弱的方式提高能量利用率；使用数据聚合能减少数据通信量。定向扩散算法的缺点是梯度建立的开销很大，不适合多基站节点网络；数据聚合过程中采用的时间同步技术在传感器网络中难以实现。

SPIN 协议是一种以数据为中心的平面式自适应通信路由协议。其目标是通过使用节点间的协商制度和资源自适应机制，解决泛洪法存在的不足之处。SPIN 有 3 种数据包类型：ADV，用于新数据广播，当一个节点有数据需要传输时，它可用 ADV 数据包对外广播；REQ，用于请求发送数据，当一个节点希望接收 DATA 数据包时，发送 REQ 数据包；DATA，包含有元数据头和传感器采集数据的数据包。在发送 DATA 数据包之前，传感器节点首先对外广播 ADV 消息，当一个邻近节点在收到 ADV 后希望接收 DATA 数据包，那它就向该节点发送一个 REQ；接着该节点向它发送 DATA 数据包。这个过程一直重复下去直到 DATA 数据包被传送到汇聚节点上。由于在传送数据之前传感器节点之间彼此进行了协商，并且协商采用的是比采集的数据小很多的元数据，它只包含了传感器节点采集数据的属性，这样就可以确保传输有用数据，避免内爆问题和部分重叠问题，降低能量消耗。并且各个节点可以在传输和接收数据之前检查各自的剩余能量情况，使得处于低能量水平的节点终止诸如充当路由器的角色这样的操作，以避免出现不顾自身能量情况而盲目使用资源的现象。

SPIN 协议的另外的优点是实现简单，节点仅需知道它的邻近节点而不需其它的拓扑信息。SPIN 协议的主要缺点是扩展受限，如果基站节点对网络中多个事件感兴趣，基站周围节点能量会很快耗尽；直接向邻居节点广播 ADV 数据包，而没有考虑其所有邻居节点由于自身能量的原因，不愿承担起转发新数据的功能，则新数据无法传输，将会出现“数据盲点”；数据在整个网络中传输。

PEGASIS 协议并不是严格意义上的分簇路由算法，但它是由 LEACH 协议改进而来，是基于链的能量有效协议。该协议的基本思想是：每个节点都知道网络中其它节点的位置，利用贪心算法形成一条数据传输链，所有数据都沿着该链传输到链头，链头在链上节点间动态选举，最后链头将数据进行融合后再将数据传给汇聚节点。PEGASIS 算法这样设计的目的有两个，一是通过各个网络节点间的协作来节省能量，二是使数据只是在本地节点间相互传输而减少对网络带宽的需求。

PEGASIS 算法的优点是避免了建立簇结构的过程，仅仅利用链中的一个节点和汇聚节点通信，免去了更换簇头节点和成簇开销；所有节点都在一条连接的汇聚节点的节点链上，节点采用小功率与最近距离节点通信，链中的每个节点都可以以多跳通信方式，将经过数据融合后的数据传输到汇聚节点，减少了数据传输和接收的数量；通过减少和基站直接通信的节点数，并让所有节点轮流担任簇头结点，这进一步延长了网络的生命周期；PEGASIS 算法的主要缺点是单簇头使得链头成为关键点，其失效会导致路由失败；且要求节点都具有与汇聚节点通信的能力；如果链太长，数据传输时延将会增大、不适合实时应用；成链算法要求节点维护全局位置信息、开销非常大。

TEEN 采用类似 LEACH 的分簇算法，只是在数据传送阶段使用不同的策略。TEEN 的具体做法是在协议中设置了硬、软两个阈值，在每轮簇头转换的时候将两个阈值广播出去，以减少普通节点发送数据的次数。当节点发现检测到的数据第一次超过设置的硬阈值时，就把这个值设置为新的硬阈值，并再接下来的时隙内发送它。之后，只有监测数据超过硬阈值并且监测数据的变化幅度大于软阈值时，节点才会传送最新的监测数据，并将它设为新的硬阈值。

TEEN 算法的主要优点是通过调节两个阈值的大小，可以在系统耗能和精度要求之间取得合理平衡，用这样的方法来监视一些突发事件和热点地区可以减少网络的通信量。TEEN 算法的主要缺点是：如果阈值不能达到，则用户将无法得到任何数据，也无法知道这个节点是否失效；数据一旦符合阈值要求，节点立即进行传送，容易造成信号干扰。

GAF 路由算法是以节点地理位置为依据的分簇算法，其最初是应用在 Ad hoc 网络中，但对于很多传感器网络同样适用。算法的基本思想是将监测区域划分成很多虚拟单元格，节点按照位置信息划入相应的单元格内。每个单元格内的节点

相互协作，一部分节点保持正常工作状态，完成数据收集和转发等任务，这部分处于正常工作状态的节点就相当于簇头节点；另一部分节点可以处于睡眠状态以节省能量，延长网络整体寿命。

GAF 算法的优点是通过使部分节点进入睡眠状态而节省能量，使网络寿命得以延长。GAF 算法的缺点是每个节点都需要获取自己的地理信息，这大大增加了节点的成本和复杂度，不适合很多场合；GAF 算法基于平面模型，没有考虑到实际网络中节点之间距离邻近并不能表示节点之间可以直接进行通信。

GEAR 是充分考虑了能源有效性的基于位置的路由协议。它比其它基于位置的路由协议能更好的应用于无线传感器网络中。GEAR 算法采用了查询驱动数据传送模式，充分利用传感器网络中的数据经常包含位置属性信息的特点和每个节点中包含的邻居节点剩余能量和到达目的地的距离信息，把整个网络中扩散的信息传送给适当的位置区域。这个协议的主要思想是利用节点地理位置信息，向特定的位置发送数据，而不是在整个网络内传输数据，限制网络传输数据的数量。它传送数据分组到目标区域中所有的节点的过程包括两个阶段：在目标域传送阶段，当节点收到数据分组，它将邻居节点同目标域的距离和它自己与目标域的距离相比较，若存在更小距离，则选择最小距离的邻居节点作为下一跳节点，若不存则认为存在空洞，节点将根据邻居节点的最小花销来选择下一跳节点；目标传送域阶段，通过域内直接洪泛或递归的目标数据传送直到目标域剩下唯一的节点方式让数据在域内扩散。GEAR 算法的优点是通过节点到事件区域的距离和节点剩余能量定义估计路由代价，利用捎带机制获取路由的实际代价并进行数据传输的路径优化，形成能量高效的数据传输路径；由于 GEAR 算法采用的是贪心算法只需知道局部拓扑信息，因此适合应用于无线传感器网络。GEAR 算法的缺点是由于缺乏足够的拓扑信息，路由过程中可能遇到路由空洞反而降低了路由效率；算法假定节点的地理位置固定或变化不频繁只适合节点移动性不强的应用环境。

SPAN 算法是一种基于位置的路由算法。算法假定各个节点了解自己的地理位置，从中选取出一些协调点。这些协调点的选取原则为：如果两个邻居节点之间不能直接通信，并且通过现有的一个或两个协调点依然无法连接通信，那么这个节点将成为新的协调点。协调点将组成一个骨干网，传感器节点收集的信息，沿协调点组成的骨干网传送到基站节点。SPAN 算法的缺点是协调点并不需要直接相连，协调点需要维护两到三个邻居节点的位置信息，这种算法使得路由协议的有效节能性较差。

## 2.4 LEACH 协议的研究

### 2.4.1 LEACH 协议的具体过程

LEACH 协议是麻省理工学院电子工程与计算机科学学院为无线传感器网络设计的低功耗自适应分簇路由算法，它的执行过程是以轮为单位周期性执行，每轮分为簇的建立阶段和稳定的数据通信阶段，它是第一个以分簇为基础的层次路由协议。在分层路由协议中，网络节点的功能是不完全相同的。网络中的节点分成被分成很多簇，每个簇有一个簇头节点和若干普通节点。普通节点只是负责信息的采集和发送，簇头节点负责数据融合，组织簇结构和向汇聚节点发送数据。

LEACH 协议对网络模型和节点做了以下假设：

（1）汇聚节点是固定的，并且放置在远离无线传感器网络的地方；节点是固定的，节点和节点之间，节点和汇聚节点之间可以相互通信；节点和汇聚节点直接通信消耗能量较高。

（2）传感器节点在监测区域内密集分布，传感器网络是同构网络。

（3）所有节点的无线电信号在各个方向上能耗相同，所有节点具有相同的初始能量且能感知自己的剩余能量。

（4）节点具有能量控制功能，可以对发送功率进行调整，调节无线收发器的工作能耗。

（5）节点间信道是对称的，节点具有足够的计算能力。以支持多种 MAC 协议（如 TDMA，CDMA 等），能实现信号处理功能。

（6）每一轮中，节点总有数据向汇聚节点发送，并且邻近节点的传感数据之间具有相关性，可以进行数据融合。

以上这些假设是合理的。因为随着无线通信和 MEMS 技术的进步节点足以满足这些条件，并且整个网络的模型可以看做是对分布式信息采集系统的抽象。在 LEACH 协议中，簇结构将大多数的通信都是在限制在簇内部，只有簇头节点才会和远处的汇聚节点进行通信。这是因为对于无线传感器节来说，无线通信的能量消耗和传输距离成正比，通信距离越远能量消耗越高，而近距离通信则比较节省能量，这种只是利用少数节点同汇聚节点进行通信的方式可以降低网络的能量消耗。在 LEACH 协议中，每个簇中有一个节点充当簇头，节点利用自组织算法形成簇型结构。簇成员节点先将数据发送到簇头节点，簇头节点对接收数据并对其进行处理后，将其发送到汇聚节点。簇头需要和汇聚节点直接通信，如果由簇内的某一节点固定的充当簇头，它的能量将很快被耗尽。如果簇头节点能量耗尽而失效，簇内其它节点由于无法正常通信而必须重新组建簇结构。所以 LEACH 协议采取簇头节点轮换的策略，这避免了簇头节点由于能量耗尽而过早失效，使得能量

消耗在网络内均衡。传感器网络中，计算耗能要远远小于通信耗能，因此 LEACH 协议在每个簇内部都对数据进行了处理，去掉了冗余信息使得网络的通信负担减轻，从而节省了大量能量。

LEACH 算法的典型思想总结如下：

1.LEACH 算法以簇结构为基础。网络中的所有节点都分到了各个簇型结构中。每个簇中的节点可分为簇头节点和普通节点。簇头节点负责组织簇结构和同汇聚节点进行长距离无线通信；普通节点负责数据采集，同簇头节点通信。这样更多的通信就局限在簇内部，减少了长距离无线通信。

2.整个算法完全是分布式的，没有任何集中式控制，只是完全依据本地信息，通过节点间的协作和控制完成簇的建立和操作。

3.簇头节点随机轮换和簇结构的动态改变。簇头节点要比普通节点消耗更多的能量。LEACH 协议通过采用簇头随机轮换的策略使得网络中所有节点轮流充当簇头，把簇头能量消耗分布到整个网络，这避免了由固定节点充当簇头导致的节点能量快速耗尽的现象。

4.簇内数据融合，降低网络通信量。LEACH 算法使得簇内普通节点向簇头发送数据，这使得簇头可进行数据融合减少冗余信息，减少了需要传输到汇聚节点的数据量。

LEACH 协议引入了轮的概念，并且采用了自适应的成簇技术和簇头节点轮换技术。LEACH 协议是一轮一轮来运行的，每一轮分为两个阶段。第一阶段是簇的建立阶段，这个阶段里通过自适应技术建立簇型结构。第二阶段是稳定运行阶段，在这个阶段中普通节点将数据发送给簇头节点，簇头节点将数据转发至汇聚节点。第二个的阶段运行时间要长于第一阶段，这样可以节约簇的建立过程所带来的能量消耗。

簇的建立过程包括簇头选举和簇的形成。在簇的建立过程中，随机决定某些节点作为簇头节点，其它节点决定自己需要加入到哪个簇结构中。簇头选择的随机性保证了同汇聚节点之间的进行数据传输的高能耗均匀的在所有传感器节点之间分摊。一个节点是否充当簇头主要取决于整个网络中簇头节点在所有节点中的比例和这个节点在过去充当过簇头的次数。LEACH 算法簇头的选举过程如下：节点 n 产生一个 0 和 1 之间的随机数，如果这个随机数小于阈值$T(n)$，这个节点充当簇头节点。阈值$T(n)$定义如式（2-1）：

$$T(n)=\begin{cases}\dfrac{p}{1-p(r \bmod \frac{1}{p})} & n \in G \\ 0 & otherwise\end{cases} \qquad \text{式（2-1）}$$

其中，$p$是簇头在所有节点中所占的百分比，$r$是已完成的轮数，$G$是本轮中未充当过簇头节点的集合。符号mod是求模运算符号。

在本次循环中，如果节点已经当选过簇头，则把$T(n)$设置为0，这样该节点不会再次当选为簇头；对于未当选过簇头的节点，则将以$T(n)$的概率当选。随着当选簇头节点数目的增多，$T(n)$的值越来越大，所以节点当选簇头的概率增大，当经过了$1/p-1$轮之后，此时就只剩一个节点未当选了，$T(n)=1$，这个节点就一定当选了。

当某个节点确定在本轮充当簇头后，它会向网络中其它节点广播数据包，告知其它节点自己是新簇头。在这个广播过程中MAC层采用的是CSMA协议，所有节点发送广告包的功率都是相同的。在这个过程中，簇内的其它节点为了接收这个广告信息而一直处于工作状态。每个节点都可能会收到来自几个不同簇头节点的广告包，根据自己与簇头之间的距离来选择加入哪个簇，并告知簇头。向簇头报告的消息主要包括自身ID和簇头节点ID。在这个过程中簇头节点的接收机也一直处于开启状态，来接收成员节点的加入消息。在接收到所有加入信息后，簇头节点会根据成员节点的数目来产生一个TDMA时隙表，这个表通过广播发送给簇内节点。簇成员节点和簇头节点之间采用TDMA机制进行通信，这保证了节点在数据传输时不会发生碰撞和冲突。普通节点在其它节点发送数据时可以关闭自身的无线装置节省能量。由于形成了簇结构，普通节点只是和本簇簇头节点通信，并且会自动屏蔽来自其它簇的节点的消息，因此簇头节点的时隙表不会被其它节点错误的接收。

当簇结构形成并且TDMA时隙表也在簇内分配完毕后，簇结构建立阶段就完成了，数据发送阶段开始。假设传感器节点总是有数据需要发送的，LEACH协议采取的是连续数据发送模式，成员节点在属于自己的时隙内将数据发送给自己的簇头节点。在自己的时隙没有到来的时候，成员节点通过关闭自己的收发机来节约能量。簇头节点的收发机一直处于开启状态用于接收来自不同成员节点的数据。并且每个成员节点通过功率控制调整成员节点的发射功率以节省能量。在一轮数据采集完成后，簇头节点会对接收到的数据进行融合，将收到的数据压缩成新的信号。通过这种数据压缩的方式可以降低发送到汇聚节点的数据量，从而降低远距离通信的数量，从而降低能量消耗。

以上是一个簇内部进行的通信过程，在网络正常运行的时候，同时有多个簇结构在同时工作，因此不同簇之间造成的相互干扰是难以避免的。因此，簇内部的通信通过CDMA机制来减少簇间干扰。如果一个节点成为了簇头节点，它可以决定本簇内所有节点所采用的CDMA码字，这个用于当前阶段的CDMA编码连同TDMA时隙表一起发送给成员节点。这样，在簇内通信的时候，非本簇的无线信号就会被滤除。

LEACH 协议在不同的轮中，利用不同的节点充当簇头节点，将远距离通信的任务轮流分配给了网络中的节点，这也就相当于在把负载均匀地分摊各个网络中所有节点。并且在簇头节点处使用了数据融合和压缩技术，使得向汇聚节点的数据传输量大大减小。

### 2.4.2 LEACH 协议存在的问题

LEACH 算法中，簇头的选取采用了阈值进行判定的方法。由于这是一个随机过程，我们只能从概率的角度研究簇头的产生，因此无法保证簇头数目的准确性。例如，假设我们在 100 个节点的区域内进行簇头选取，通过数学推导和仿真，选举簇头比例设定为 5%，即 p =0.05 网络的能耗最小。那么最优簇头个数为 5。总体而言，20 轮总共可能产生 100 个簇头，平均每轮 5 个，但每轮簇头个数并不完全等于 5，许多轮的簇头个数偏离 5 而上下波动。在 LEACH 算法中，虽然统计上平均簇头数可以满足 5%的要求，但每轮的簇头数目却存在较大的随机性，甚至出现严重偏离最优簇头数的情况，造成通信能耗迅速增大。由于网络载荷不是完全平衡，所以网络的寿命就肯定不是最长的。显然，该算法无法实现理论上的最优化设计。当簇头个数太少时，簇的半径过大，簇内节点与簇头距离增大，簇内节点必须将数据传送给相距很远的簇头节点，节点的通信能耗较大，众多簇内节点与簇头进行通信，势必加重簇头节点的负担，不利于延长网络的生命周期。当簇头个数太多时，局部数据融合的耗能将增加，融合的效率将降低，从而无法优化网络的性能。

LEACH 算法是让网络中的节点自组织地形成簇，簇头是随机产生的，这种随机产生的方式无法保证簇头节点的合理分布。当簇头节点位置比较集中时，簇的覆盖区域将出现部分重叠的现象，网络拓扑结构不够优化；当簇头分布在边缘区域时，簇内节点到簇头的距离将变大，使得簇的载荷也相应的变大。

LEACH 算法忽略了被选簇头在网络内的分布状态和节点间不同的通信距离而导致的节点能量损耗的不平衡。在能量不均衡的网络下继续运行随机簇头选择策略，使得担任簇头是严格等概率的，能量为 1 和能量为 10 的节点做簇头的概率完全相同，而能量较低的节点一旦选作簇头，很容易耗尽能量而失效，这一方面不利于网络生命周期的延长，另一方面簇头一旦失效，整个簇无法通信不利于网络的健壮性。

由于 LEACH 要求节点之间以及节点与汇聚节点之间均可以直接通信，所以网络的扩展性不强，并且不适用于大型网络。无线传感器网络的能量消耗主要包括电路能量消耗和功放能量消耗。对于大型网络而言，对离簇头较远的簇内节点和离汇聚节点较远的簇头而言，能量消耗主要是功放能量消耗，并且依据空间信道

模型，通信能耗所消耗的能量取决于传输距离。所以随着传输距离的增加，传输所消耗的能量将大大增加。这意味着离汇聚节点远的簇头能耗远远大于接近于汇聚节点的簇头，与此类似簇内也有相似的情况。这样簇头节点能耗分布不均匀，导致某些节点快速死亡，从而降低了网络的性能。

总之，虽然 LEACH 协议采用分簇结构减少了与基站直接通信的节点数，并且采用了数据融合技术和动态分簇技术，从而有效地减少了能量消耗、延长了网络的生命周期，但是 LEACH 协议仍存在着一些问题：没有根据网络的拓扑结构获得网络当前的最优簇头数目；不能保证簇头数目的准确性；不能保证簇头数目的合理分布，不能保证簇内结构的质量，进而降低网络的整体性能；簇头轮换的过程中节点之间严格等概率，没有考虑节点的剩余能量，不能有效地均衡网络中的能量分布；使得每个簇的簇头直接和汇聚节点通信，能量消耗较大；网络的扩展性不强。

# 第三章 基于簇融合的改进型路由协议

## 3.1 基于簇融合的改进型协议概述

在基于簇融合的改进型协议中，仍然采用了分簇结构，但簇的组织方式与LEACH协议不同。在LEACH中，P是期望簇头节点在所有传感器节点中的百分比，这个值是固定不变的。而在实际应用中，因为有些传感器节点由于能量的耗尽而死亡，所以P的最优值不是固定不变的。确切的说，随着网络的运行，整个网络中的总的传感器节点数目应该是减少的。而基于簇融合的改进型协议簇头结点在所有传感器节点中的百分比不是固定不变的，而是随着网络中节点数量的变化而变化的，并且保证簇头节点数处于最优状态，保证传感器节点的能耗最小。LEACH协议的每一轮都会重新分簇，并进行簇头的重新选举。基于簇融合的改进协议并不是每轮都进行重新分簇，但是每轮都进行簇头的重新选举，这是因为簇头的能量消耗远大于普通节点的能量消耗，所以每轮必须转换簇头，以免使得簇头节点消耗能量过快而死亡，影响传感器网络的整体寿命。但是每一轮都重新分簇不是必须的，只有在传感器网络节点数量变化影响到目前的最优节点分簇数目时才对簇结构进行调整。LEACH协议的簇头是随机产生的，这种随机产生的方式无法保证簇头节点的合理分布，基于簇融合的改进型协议是利用节点的位置信息完成分簇工作，使得簇头合理地分布在整个传感器网络当中。

传感器网络的汇聚节点与网络中的其它节点相比有很大的不同，汇聚节点的电源能量，通信能力，计算能力和存储能力要比其它节点大得多，特别是汇聚节点的电源能量与普通节点相比可以认为是无限大，并且能够接收到传感器网络中所有节点感知到的消息，它能够掌握传感器网络的整体情况。基于簇融合的改进型协议充分利用汇聚节点的电源能量充足，计算能力和存储能力巨大的特点，将一些复杂的计算和大量的数据维护工作，例如获得全局节点的拓扑信息、计算当前网络的最优分簇数目等，交由汇聚节点完成。

LEACH协议的簇头选举完全是依概率的，这样就会导致能量较多和能量较少的节点做簇头的概率完全相同，而能量较低的节点一旦选作簇头，很容易耗尽能量而失效。基于簇融合的改进型协议对簇头的选举策略同时考虑节点的剩余能量和节点采集数据的情况，剩余能量较多的节点优先被选为簇头，这可以避免该节点能量迅速消耗而失效；传感器网络的数据具有突发性，它会在短时间内产生大量的数据，由于传感器节点的计算能力非常有限，所以如果选择采集数据可能性较大的节点作为簇头，数据包发送和接收、数据融合等都会影响到节点的处理速度，会影响数据采集的完整性，基于簇融合的改进型协议优先选择那些采集信息数据相对较少的节点作为汇聚节点。

基于簇融合的改进型协议中，簇的组织形式不同于 LEACH 协议，簇头选举过程相对于 LEACH 协议来说较为简单。基于簇融合的改进型协议中，簇是链状结构。开始时，簇头节点为节点链的头节点，随着节点能量的不断消耗，簇头节点在节点链中转换，簇中节点并不直接和簇头节点通信，而是只向相邻节点转发数据，最终数据汇聚到簇头节点。当需要发生簇头轮换时，并不破坏整个簇的链状结构，只是在确定新的簇头节点之后，将上一轮的簇头节点和新簇头节点之间的那些节点的数据转发的目的地址改变即可。这样簇头轮换所需的时间就相应降低，并且发送数据包的长度也相应缩短。

在基于簇融合的改进型协议的使用过程中，可能由于某些环境因素造成节点的突然失效，而没及时通知相关节点。那些通过这些失效节点与簇头或汇聚节点通信的那些节点也会丢失，这带来了不必要的损失。改进型协会尽量及时发现某些节点的突然失踪，并且尽力发送数据包来查找这些丢失节点，挽回损失，并且将确实失效的节点汇报给汇聚节点，使得汇聚节点对网络中的整体信息有正确的了解，使其能够更加合理的确定当前的最优簇头数。

由于 LEACH 要求节点之间以及节点与基站之间均可以直接通信，随着传输距离的增加，传输所消耗的能量将大大增加。考虑到离汇聚节点远的簇头能耗远远大于接近汇聚节点的簇头，基于簇融合的改进型协议在簇头节点之间建立路由树，这样簇头中的数据就可以通过多跳到达汇聚节点，同样簇内的节点采集到的信息可以通过多跳到达簇头。这样就避免了簇头或者簇内节点数据传输在过远的距离上实现。

## 3.2 基于簇融合的改进型协议的具体描述

### 3.2.1 节点位置信息获取

在基于簇融合的改进型协议中，汇聚节点需要首先了解整个网络的拓扑结构，以便于网络中簇的划分在汇聚节点的控制下完成。为了能够使得汇聚节点了解整个网络的拓扑情况，在网络启动的时候，网络中所有节点会在汇聚节点的要求下建立路由树结构。

首先汇聚节点以足够大的约定功率向外广播一个启动数据包，网络中的所有节点通过底层硬件对这个接收到的数据包进行功率检测，从而节点能够确定到汇聚节点的距离，这个距离将用于簇头间选择路由时使用。然后，由汇聚节点通过泛洪方式向网络中广播路由树组建消息，树组建消息由组建标志位、发送节点 ID、发送节点级值组成。收到广播消息的节点将数据包中的发送节点 ID 记录为父节点，然后将级值数加 1，并以自身 ID 和级值更新消息后继续向邻节点传播，直到网络中所有节点都拥有自己的父节点和级值。由于广播传输中可能会导致已经发出消

息的节点再次收到邻节点将值加 1 后重新发送来的消息，从而导致消息循环。可以通过以下规则解决该问题：当一个节点从邻节点处收到广播消息时，首先检查消息中的级别是否低于自身级值，如果低于则以该级值更新，并向邻节点发送新的消息；否则将不予处理。在消息传播过程中节点选择最早发送消息且消息中所含级别值最低的节点作为父节点。

在生成树的过程完成后，网络中的每个节点将自己的位置坐标、ID 值组成数据包沿着生成的树的路径向汇聚节点发送。由于汇聚节点的存储和计算能力较强，汇聚节点可以在收到网络中所有节点的位置数据包后获得对网络拓扑结构的了解。由于此时汇聚节点对于网络中传感器节点的数量和位置有了充分了解，汇聚节点可以对网络中簇的划分作出最优化的结论。

### 3.2.2 计算最优簇头数量并生成节点链

汇聚节点可以根据网络中的节点数量和位置得出最优的分簇数。根据文献[17]，在知道网络中的节点总数和分布密度的前提下可以获得网络中的最优分簇数。在汇聚节点获得最优分簇数后，就可以根据每个节点的位置信息将网络中的节点分成簇，每个簇中有数量相同的网络节点。

计算传感器网络中最优分簇数的方法如下：假设 M 是正方形固定区域 S 的边长，固定区域内节点数目服从均值为 λ*A（S）的泊松分布，λ 为一个正常数，表示泊松分布强度，那么簇头在其中所占比例的最优值 $P_{opt}$ 的计算公式如式（3-1），

$$P_{opt}=\frac{0.0833C_1^2}{C_2^2}+\frac{0.1050(C_1^4+24C_1^2C_2^2)}{C_2^2(2C_1^6+72C_1^4C_2^2+432C_1^2C_2^4+83.1384C_1^2\sqrt{C_1^2C_2^6+27C_2^8})^{\frac{1}{3}}}$$
$$+\frac{0.0661}{C_2^2}+(2C_1^6+72C_1^4C_2^2+432C_1^2C_2^4+81.1384C_1^2\sqrt{C_1^2C_2^6+27C_2^8})^{\frac{1}{3}}$$

$C_1=2M\bullet E[X(S)^{1/2}]=3\sqrt{\pi}$，$C_2=0.3825M\bullet E[X(S)]=0.3825M^3\bullet\lambda$　　式（3-1）

具体分簇方法如下：首先将传感器网络的覆盖范围按其坐标划分成若干大的网格如图 3.1 所示，网格个数取决于区域大小。先从第一格开始扫描，第一格中的节点不足以构成一簇时，选择 2，3，4 格中节点数最多的作为下一个扫描格，假设为 4 格，仍然不足时，在第 5，6，3 格中选择节点最多的作为下一个扫描格。

在扫描各个大网格的时候需要将每个大的网格分成若干小网格，如图 3.2，按照图中表明的序号扫描顺序逐次将网络节点添加到分簇中，当一个分簇中的节点数到达上限时，这个簇就划分完成。在接下来的扫描过程中完成新的簇的划分。将网络覆盖范围内的每个小格都扫描过后，分簇过程也就完成了。

在分簇过程完成以后，汇聚节点掌握了各个分簇内的节点的位置信息，根据这些位置信息可以将簇内的节点连接成链。链的形成过程采用贪心算法，具体步

骤如下：在簇内任取一个节点，在簇内的剩余节点中选择距离该节点最近的点作为节点链的下一个节点；分别以链的两端为起点，在簇内查找不属于节点链上的距离链的两端点距离最小的节点，并将该节点添加到节点链上；重复执行以上步骤直到簇内的所有节点都添加到了节点链上。当成链过程完成以后，汇聚节点随机选择链的一端作为该簇的簇头，汇聚节点将分簇信息和节点链的信息通过成簇数据包传送到簇头节点。

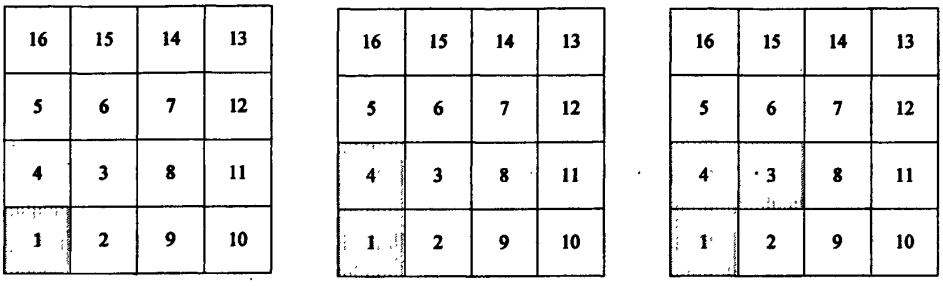


图 3.1 大格扫描顺序

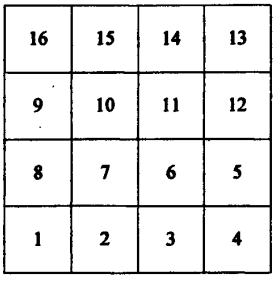


图 3.2 小格分布

成簇数据包中必须包含包类型，目的节点，包序号，路径信息，节点链信息，标志位等。由于传感器网络中的节点中不包含从汇聚节点到普通传感器节点的路径，所以在汇聚节点发往普通传感器节点的成簇数据包中需要包含数据包的路径信息。如果成簇数据包的长度超出了最大可传输的长度，则通过多个成簇数据包传输成簇信息，通过序号和标志位信息在目的节点对成簇信息进行重组。成簇数据包虽然长度比较大，但是由于其发送的次数很少，因此成簇数据包带来的性能影响可以忽略不计。

当选定的簇头收集到完整的成簇信息后，就以分簇信息为依据，真正的在传感器网络的节点中建立节点链。当节点收到节点链的建立数据包后，需要保存节点链上的左右邻居结点 ID 和簇头的方向，如果是起始点则只保存一个邻居结点，所有节点都记录自己在当前节点链的位置，根据当前节点链中的节点个数进行时隙划分，各个节点只在特定的时隙内接收或者发生数据包，当节点链中的点都建立了这种结构后，簇的建立过程就完成了。簇内的节点采集到信息的时候，数据包就沿着节点链传输并进行数据融合，这个过程类似于 PEGASIS 算法。

### 3.2.3 簇头选择

如果簇头没有收到簇合并的数据包，那么簇的下一轮和本轮具有相同的簇成员和相同的节点链。下一轮的簇头选择就在簇内的非簇头节点内选取。簇头选举不再采用随机的方式，而是以节点的剩余能量为依据，同时需要考虑节点采集信息和传递信息的情况。簇头是簇内最重要的节点，它不仅管理本簇成员，收集簇内成员的数据传输，还要融合簇成员收集的数据，再将处理过后的数据发送给汇聚节点。簇头间加入多跳路由后，簇头的负担进一步加重，所以应该尽量选择剩余能量较高的节点作为簇头。同样，由于簇头结点需要完成数据转发和数据融合的任务，传感器网络节点的计算能力和存储能力又都非常有限，所以数据的转发和融合会影响到数据的采集，所以我们要尽量选择那些在以前采集到信息较少的那些节点作为簇头节点。

在传感器节点向簇头节点发送数据的同时会附带发送下一轮簇头节点竞争信息，该项信息包括节点 ID、节点的剩余能量 E、转发数据与采集数据的比值 t 和该节点与簇头节点在数据链上的相对位置。这些信息会沿着节点链传输，并与途经的节点上的信息进行比较，最后把最适合作为簇头的节点的信息传递到当前簇头处，簇头将收到的新的信息与簇头中保存的作为下一轮的簇头的节点的信息进行比较，保留更适合作为下一轮簇头节点那个节点的信息。

为了比较簇中的那个节点更适合作为下一轮的簇头，引入如式（3-2）定义：

$$F_j = a_1 * E_j / E_{all} + a_2 * t_j \qquad \text{式(3-2)}$$

其中 $a_1$，$a_2$ 是大于 0 小于 1 的数，且 $a_1 + a_2 = 1$，用户可以根据需要设定 $a_1$，$a_2$ 的值，$E_j$ 是节点 j 的剩余能量，$E_{all}$ 是节点的初始能量，$t_j$ 是节点 j 采集数据和发送数据的比值。

同一簇内的节点 i 通过相邻节点 j 向簇头节点发送消息，节点 j 收到节点 i 发送的消息后，通过上述公式计算节点 i 的权值 $F_i$ 和自己本身的权值 $F_j$，选择权值较大的那个节点作为节点 j 向下一相邻节点转发的下一轮簇头竞争节点。簇头节点处也进行相似的比较。

在一轮数据采集完成以后，簇头节点中已经确定了哪个节点作为下一轮的簇头节点。簇头节点的轮换过程如下：本轮的簇头节点知道下一轮的候选簇头节点在节点链上与本轮簇头节点的相对位置。当本轮信息采集完成以后，在簇信息时隙内，本轮的簇头节点以合适的功率对外广播簇头节点转换数据包，由于数据包中含有下一轮簇头节点的 ID。当下一轮簇头节点和本轮簇头节点之间的节点收到轮换数据包后，同自己保留的节点数据包比较，如果 ID 相同的，则只需转变数据的传递方向即可。当下一轮的簇头候选节点收到簇头节点轮换信息并设置完簇头数据结构后，簇头轮换的工作就完成了。

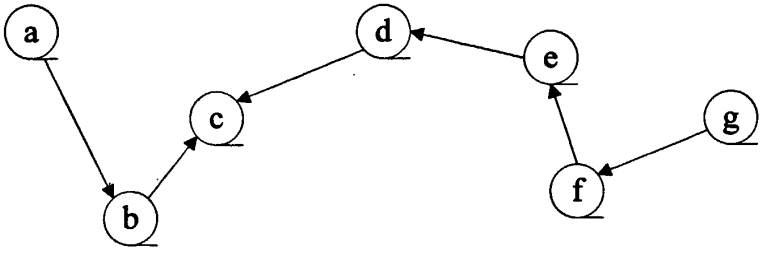


图 3.3 簇头轮换前工作模式

例如，假设参数$a_1$为 0.9，参数$a_2$为 0.1，在第 5 轮中簇头为 c 及簇中消息的传递方式如图 3.3 所示。

若节点 c 的剩余能量与总能量的比值为 0.9，转发信息与采集到的信息比值为 0.5，那么$F_c = 0.9*0.9+0.1*0.5 = 0.86$。若第六轮的簇头候选节点为 f，其剩余能量与总能量的比值为 0.95，转发信息与采集到的信息的比值为 0.48，那么$F_f = 0.9*0.95+0.1*0.48 = 0.903$，$F_f > F_c$,则 f 节点作为下一轮的簇头节点。

则在簇头轮换工作完成后，簇中的消息传递方式如图 3.4 所示：

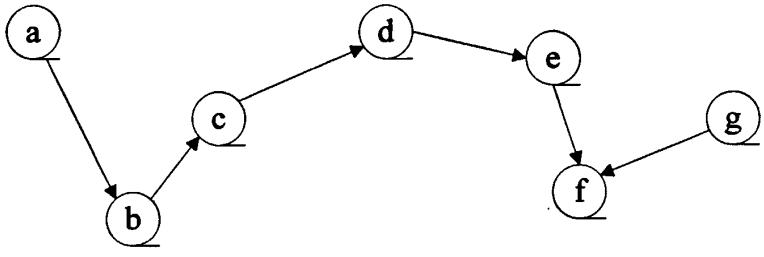


图 3.4 簇头轮换后工作模式

### 3.2.4 节点退出及簇融合

在传感器网络的工作过程中，若某一节点由于能量不足而不能继续维持工作，这会影响簇内节点链上数据的传输，也会影响传感器网络汇聚节点对整个网络的拓扑认知。所以当传感器网络节点中的能量不足以继续工作，需要向链中的相邻节点发送退出链的数据包，使得与这个节点相邻的两个节点连接形成新的链。这个节点退出数据包要传向簇头节点，最终发送给汇聚节点，当汇聚节点收到这个消息后，它会在相应改变在内存中存储的传感器网络节点所组成的拓扑结构。

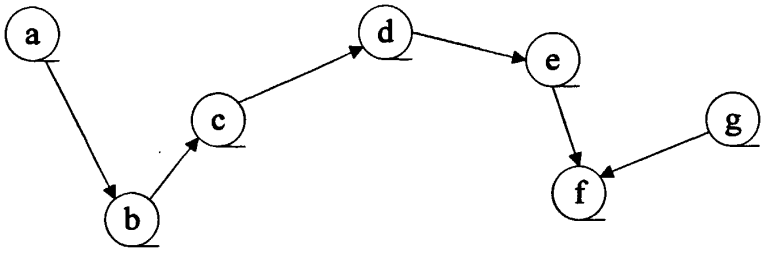


图 3.5 节点退出前拓扑图

初始时节点链的工作方式如图 3.5 所示。e 点能量不足而退出本簇的节点链后

形成新的节点链如图 3.6 所示

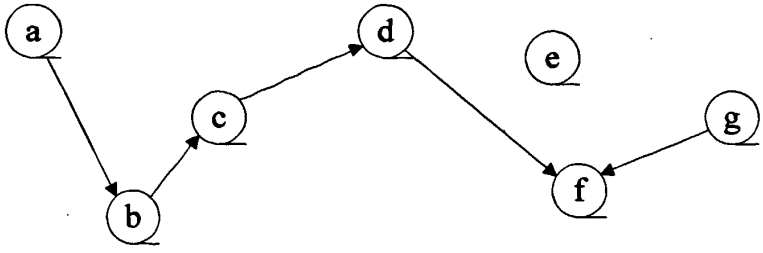


图 3.6 节点推出后拓扑图

随着传感器网络持续工作，会有越来越多的节点由于能量不足而停止工作，这使得整个传感器网络中节点数量越来越少，从而导致当前的分簇数不是最优的，需要通过簇的合并来减少分簇数，从而使得当前网络中的分簇数变为最优。每当有传感器节点通知汇聚节点自身要停止工作时，汇聚节点都会根据式（3-1）重新计算 $P_{opt}$，然后结合目前还在持续工作的节点数目，算出最优的分簇数，当最优分簇数与当前分簇数的差别达到 1 时，汇聚节点就在自己的内存中查找含有工作节点最少的那个分簇 a，并且根据位置信息在这个分簇的周围找出可以和这个分簇直接相连的节点数最少的分簇 b，然后汇聚节点向两个簇发送簇融合消息，簇融合数据包中包括包类型，目的节点，序号，路径信息，融合节点标号，方向，合并后簇内节点总数等信息，其中融合节点标号域表示的是需要与其它簇的节点链进行连接的标号，方向域表示的是融合节点在节点链上处于目的节点的相对位置。

当两个簇收到消息后，就将两个簇连接在一起组成一个新的簇，新簇的簇头由合并的两个簇的簇头 F 值更大的那个作为新的簇头节点。

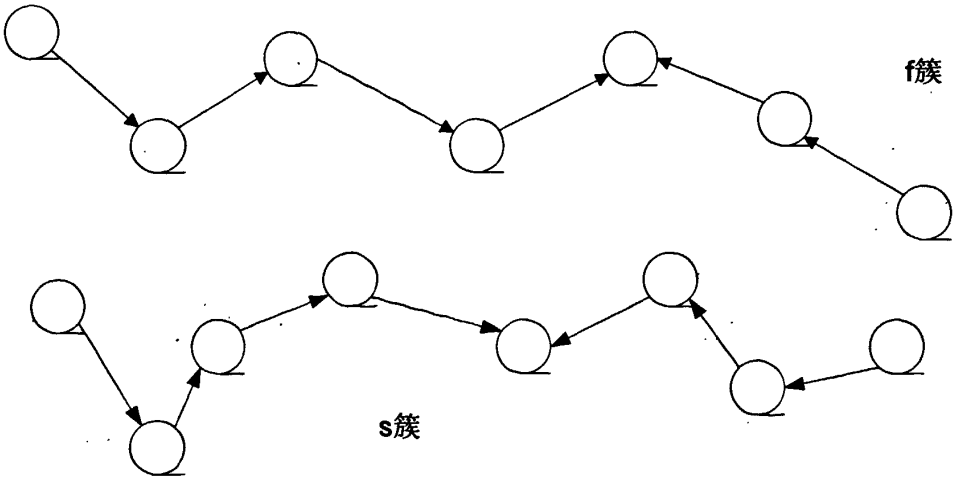


图 3.7 簇融合前两簇的拓扑结构

例如，初始时传感器网络中共有活动节点 500 个，根据式（3-1）计算地得到的 Popt 为 0.023，所以此时传感器网络的最优分簇数为 12 个。当传感器网络运行到第 r 轮时，此时传感器网络中共有活动节点 400 个，根据公式计算得到的 Popt

为 0.028，所以此时传感器网络的最优分簇数为 11 个。这样，传感器网络的最优分簇数的变化达到 1，汇聚节点就在存储结构中查找现有分簇中活动节点最少的那个簇，假定为 f，根据分簇的位置信息，在和 f 簇直接相邻的那些簇中查找含有活动节点最少的簇作为合并对象，假定为分簇 s。起始时 f 簇和 s 簇的拓扑关系如图 3.7 示。

当分簇 f 和分簇 s 收到汇聚节点发送的簇融合消息后，两个簇的簇头节点都会向着本簇的节点链上的融合节点处转发数据包，在经过各自的簇头节点时，在数据包中添加簇头竞争消息。当融合节点收到融合数据包后，就向周围的节点广播融合融合信息，这个数据包的结构较为简单，只需要包含包类型域和目的节点域。假定 f 簇的融合节点先收到簇融合消息，这个融合节点就向 s 簇的融合节点展开三次握手以完成建立连接的任务，如果此时 s 簇的融合节点也收到了簇融合数据包，则响应握手过程，否则并不响应握手过程。换句话说，只有两个节点链的融合节点都收到了簇融合数据包以后，簇融合过程才开始。然后，各个簇的融合节点向自己原来簇的簇头节点发送连接完成的消息，当链接完成以后，s 簇的融合节点和 f 簇的融合节点将对各自的簇头节点的竞争消息进行比较以确定新簇的簇头节点由谁担任。假定 s 簇的簇头更适合充当新簇头，则融合节点会将竞争结果回传给簇头节点。在新的节点链中处于 f 簇的簇头和 s 簇的簇头之间的那些节点将根据簇头竞争的比较结果来决定自己的消息传递方向，当 s 簇的簇头收到了簇头确定消息后，簇的融合过程就结束了。完成簇融合后新簇的拓扑结构图 3.8 所示。

当两个节点链发生簇融合时，新产生的节点链中的节点需要重新划分时隙，由于数据包中包含了新产生节点链的节点个数。在簇融合数据包在节点链的传递过程中，各个节点就根据新链上的节点个数重新划分了时隙。汇聚节点指定其中一个簇从时间段的开始处划分，另一个从时间段的结束处划分。

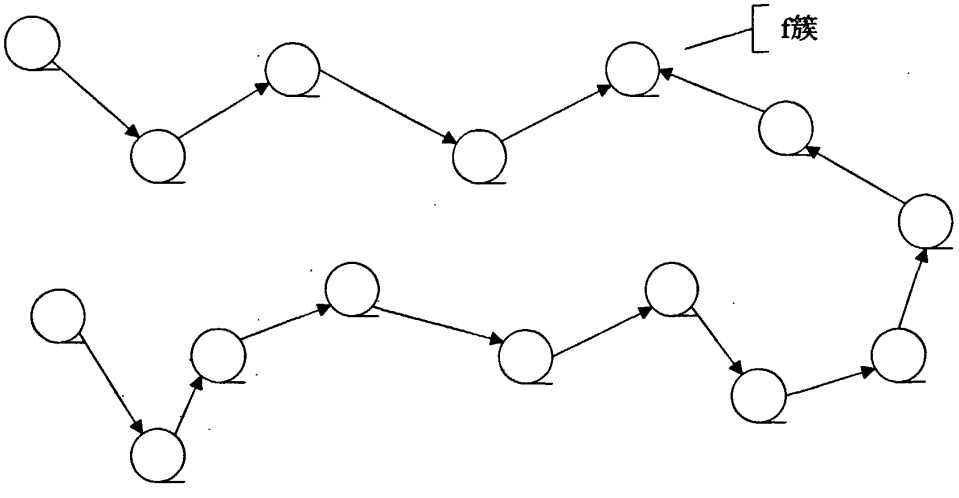


图 3.8 簇融合后新节点链的拓扑结构

### 3.2.5 丢失节点查找

由于改进型协议的分簇是基本固定的，在首轮分簇完成以后，只是进行了以簇为单位的融合操作而没有像 LEACH 协议那样以节点为单位进行簇生成操作。若传感器网络的节点由于某种原因而突然消失，而不能完成将它的相邻两节点连接后再退出的工作，这会导致传感器网络中的某些活动节点仍然可以继续工作，但是失去了和本簇簇头节点的联系而不能继续工作；或者某个分簇的簇头节点由于某种原因而突然损坏，这会导致整个簇不能正常工作，并且使得簇内其它节点可用的情况下与汇聚节点失去联系，这样会对整个传感器网络造成不必要的节点损失。

我们通过下面的机制来防止上述情况的发生：

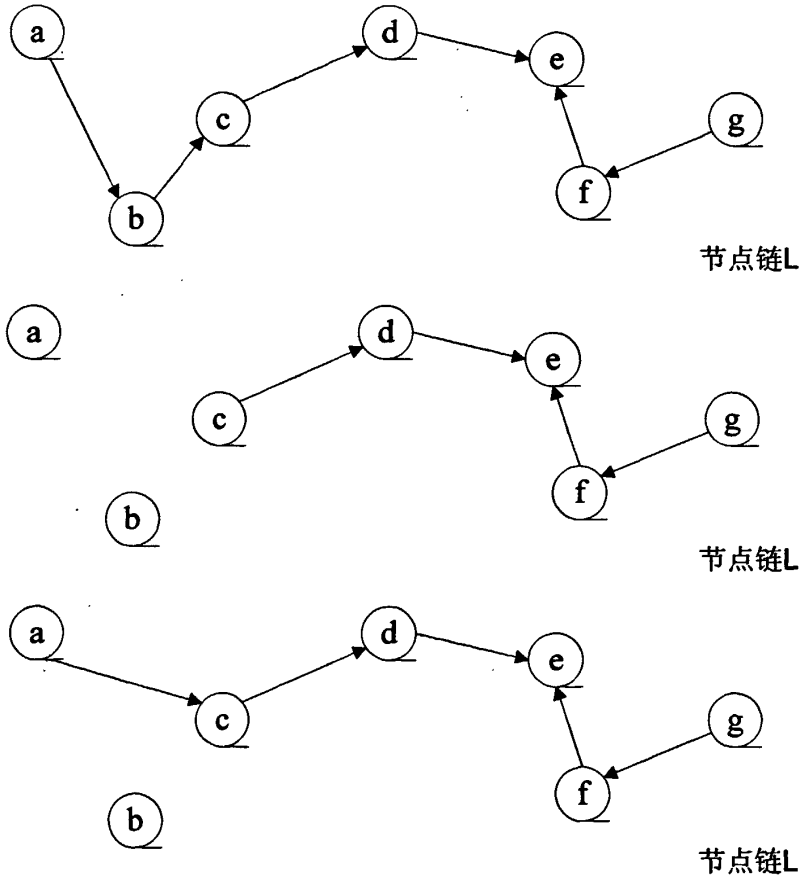


图 3.9 节点丢失后查找过程

当出现由于节点链中的某一节点突然消失而导致的节点丢失的情况时，处在接收位置的节点会主动向外广播丢失节点查找数据包。节点如果在一段时间内，未收到发送数据后的应答数据包，就确认自己成为了丢失节点，当丢失的节点接收到该数据包时，在等待一段时间后没有接收到距离它更近的节点发送的丢失节

点查找数据包，丢失节点就应答丢失节点查找数据包，然后和查找节点重新建立节点链。丢失节点是通过数据传输方向和查找节点在链中的位置来确定优先响应哪个查找数据包的。

如图 3.9 所示，对于节点链 L，如果其中的节点 b 没有向邻居节的发送退出数据包就突然消失，则 a 节点失去了与节点链的联系。经过一段时间后，c 节点仍然没能收到 b 节点发送的信息，则 a 节点向周围发送重新建立连接的数据包，这个数据包中包含包类型，c 节点的 ID，自身在链中的位置标号，所属分簇 ID，丢失节点 ID。当 a 节点经过一段时间后，未收到数据应答信息，确定自己是丢失节点，在接收到了这个数据包之后，会检查自己所属分簇的 ID，确定自己与 c 节点的距离。一段时间后未收到新的本簇内节点查找数据包，则 a 节点将自己的下一条节点的值改为 c，然后向 c 节点发送应答数据包。当收到应答数据包之后，c 节点就将自己的上一跳节点的值改为 a。这样 a 节点就重新进入了自己所在簇的节点链之中。这样节点 c 就检测到了节点丢失信息，这条信息会传递到汇聚节点，纠正汇聚节点对网络中拓扑的认识。

当簇头节点由于某种原因而突然不能工作所导致的整个簇不能与汇聚节点进行通信时，只是利用上述机制就不起作用。当汇聚节点在一段时间内没有收到某簇内簇头节点的周期性报告信息时，汇聚节点会根据自己内存中沿着整个网络中所有节点所生成的树的边向着该簇的节点链的两端发送簇头丢失信息，当链的端节点收到后，会沿着节点链向链的中间发送消息。数据包在节点链的传输过程中，会同时记录下最适合当做新簇头的节点 ID，当数据包传输到与簇头节点相邻的节点时会采用广播的方式向周围节点发送查找数据包，查找节点包括：原来的簇头节点；属于本簇的、节点链失去连接的节点。如果找到了原来的簇头节点则仍然以原簇头节点作为簇头继续工作。如果找到了属于本簇但与节点链失去连接的节点，就将最适合作为新簇头的节点信息发给那个节点，当那个节点收到从节点链的另一头传过来的簇头丢失信息后，比较两个信息中的新簇头节点的竞争信息，选出最适合当做新簇头的那个节点，并沿着节点链向着那个节点发送新簇头的确定消息，沿途经过的节点收到这个消息后，会确定自己采集到数据的传输方向，当新簇头节点收到这个消息后，簇头丢失的补救工作就完成了。

### 3.2.6 簇间多跳通信

让各个簇头直接同汇聚节点通信，能量消耗太大，所以簇间采用多跳路由协议。在进行簇间通信时，由于收到的数据包中已没有了冗余信息且来自不同簇的数据无法进一步融合，中继簇头只是将来自其它簇头的数据进行简单转发。

在簇头轮换完成后，每个簇的新簇头会向邻近的簇头节点广播消息，消息中

包含其节点 ID、剩余能量和到汇聚节点的距离。由于节点具有根据信号强度估算距离的能力，因此在收到消息后其它簇头就可以得到到此簇头的近似距离 d，并把这个簇头暂时存入自己的邻居簇头表中。这个过程完成后，每个簇头要从自己的邻居簇头表中选出适合作为下一跳的簇头，这是簇间路由的关键。选择依据是：选择链路能量消耗最小的路径。假设通信采用的是自由空间模型，链路的能量消耗同距离的平方 $d^2$ 呈线性关系。假设簇头 i 有数据需要传输，通过中继簇头 j 中转再到达汇聚节点的能量消耗可以用 $d_{toSink}(j)^2+d_{itoj}{}^2$ 来表示，簇头 i 直接将数据传送给汇聚节点的能量消耗可以用 $d_{toSink}(i)^2$ 来表示。当 $d_{toSink}(j)^2+d_{itoj}{}^2<d_{toSink}(i)^2$ 条件成立时，节点 i 就选择通过节点 j 来传输数据能量消耗更小，否则就直接向汇聚节点发送数据能量消耗更小。另外，节点的剩余能量太低也不适合数据包的转发，因此簇头节点在选择下一跳簇头节点时需要综合考虑中继节点的剩余能量和中继转发能量消耗两个方面的因素。为此定义参数 W，节点 j 的 W 权值为 $W_j=S(j).E/E_{ave}+(d_{toSink}(j)^2+d_{itoj}{}^2)/d_{toSink}(i)^2$，其中 $S(j).E$ 表示的是节点 j 的剩余能量，$E_{ave}$ 表示的是 i 节点路由表中所有邻居节点剩余能量的均值。

所有邻近簇头节点都加到簇头 i 的邻居簇头表中后，i 节点计算这些邻居簇头节点的权值 W。依照权值的大小顺序来排列路由表中簇头节点的顺序，簇头 i 从权值 W 大于 2 的节点中优先选择权值大的簇头节点作为下一跳路由。如果出现权值相等的情况，在其中随机选择下一跳路由。这样就建立了下一跳路由表，由于节点的存储量有限，下一跳路由表中只是存放权值最大的三个路由表项，它们在表中按权值大小依次排列。

充当中继的簇头节点可能在实际工作过程中为多个簇头服务，这就导致了中继簇头节点的能量快速消耗，迅速死亡。可以通过在中继簇头中引入能量变化阈值的 $\Delta E$ 方式来防止这种情况的出现。当中继节点为转发数据的能量消耗超过 $\Delta E$，就停止中继转发数据，并向邻居簇头节点广播不再为各邻居中继转发数据的消息。邻居节点收到这一消息后，就将这个路由项从下一跳路由表中删除，继续以路由表中的下一个路由项为中继转发节点。若路由表为空，则簇头将自己的数据直接传输到汇聚节点。

例如，簇头节点 6 在某一轮的路由表如同表 3.1 所示，该节点会选择 W 权值最大的节点 5 作为下一跳的中继节点。若在本轮工作过程中，中继簇头节点 5 用于转发的能量消耗高于阈值 $\Delta E$，此时，就会广播不再充当中继节点的消息。簇头节点 6 收到这一消息后，会将节点 5 从下一跳路由表中删除，然后选择簇头节点为 8 的那个路由项作为其下一跳路由。若下一跳路由表中的每个节点都充当过中继且能量消耗都达到了 $\Delta E$，那么路由表会变为空，簇头就不再利用中继而是直接将自己的数据送至汇聚节点。

当簇间多跳路由建立完成后，实际上各个簇头组成了一个以汇聚节点为根结

点的路由树，数据在路由树上向着汇聚节点转发。

表 3.1

| 顺序号 | 下一跳簇头节点的 ID 号 | 权值 W |
|---|---|---|
| 1 | 5 | 4.5 |
| 2 | 8 | 3.2 |
| 3 | 2 | 2.3 |

## 3.3 仿真实验

### 3.3.1 仿真模型和参数设定

LEACH 协议和基于簇融合的改进型协议仿真的基础是第一顺序无线电模型，所谓第一顺序无线电模型就是假设网络中所有节点完全相同，能量有限；汇聚节点远离整个无线传感器网络，并且位置固定；无线信道是对称的，从节点 A 发到节点 B 的能量消耗与从节点 B 发到节点 A 的能耗相同，并且无线电信号在各个方向上的能耗相同。

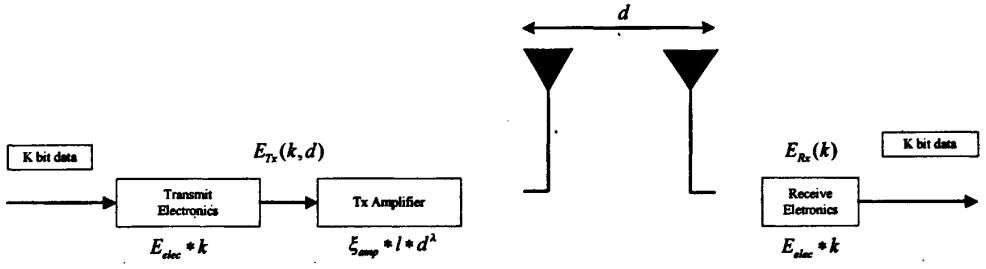


图 3.10 第一顺序无线电模型

根据图 3.10 所示的第一顺序无线电模式，传感器节点发送 k bit 数据消耗的能量分为两部分，一部分是信号发射电路所消耗的能量，一部分是信号放大电路所消耗的能量，写成式（3-2）

$$E_{Tx}(k,d)=E_{Tx-elec}(k)+E_{Tx-amp}(k,d)=k*E_{elec}+k*\xi_{amp}*d^{\lambda} \quad \text{式（3-2）}$$

信号接收电路接收 k bit 数据所消耗的能量如式（3-3）所示：

$$E_{Rx}(k)=E_{Rx-elec}(k)=k*E_{elec} \quad \text{式(3-3)}$$

其中，$\xi_{amp}$ 是信号放大器的放大倍数，而 $E_{elec}$ 是发送电路和接收电路消耗的能量，在这个模型里两者相等。而 d 是信号传输的距离，$\lambda$ 是无线电信道决定的常量，当两个相距很远的节点通讯时，采用自由空间能量衰减模型，一般取 $\lambda=4$，$\xi_{amp}=\xi_{fs}=0.0013\,pJ/bit/m^2$，当两个距离比较近的节点进行通信时，采用多径能量衰减模型，取 $\lambda=2$，$\xi_{amp}=\xi_{fs}=10\,pJ/bit/m^2$。因此，根据发送节点和接收节点之间的距离，发送节点可以使用不同的能耗模型计算发送数据所需的能量。

LEACH 协议和基于簇融合的改进型协议的仿真参数相同，设定如下：

网络中的节点数量 G 为 100，每个节点的初始能量都为 0.25 焦，每个传感器节点的位置是固定的，不具有移动性；

基站处于坐标（25，100）处，所有的传感器节点都分布在50*50的区域，横坐标范围 0 至 50，纵坐标范围也是 0 至 50；

在发射和接收机电路中，每处理 1 比特数据，就需要消耗能量 50 纳焦，即 $E_{elec}=50nJ/bit$ 发射机信号放大电路需要向单位面积发射 1 比特数据需要消耗 10 皮焦的能量，即 $\xi_{amp}=10pJ/bit/m^2$ 。

在簇头节点进行本地信号处理和数据融合时，平均每比特数据所消耗的能量为 5 纳焦，即 $\varepsilon=5nJ/bit$，数据包的平均长度为 2000 比特。

上述参数并不是恒定不变的，对于不同的仿真内容可根据需要改变参数值。

LEACH 算法的仿真模型：

固定基站离所有节点都很远，所有节点能够将采集到的数据直接发送给基站。网络中所有传感器节点都是同类型的节点并且具有相同的初始能量；节点具有足够的功率与其他节点和基站进行通信，且具有控制发送功率的能力，每次都是以所需的最小功率与其他节点或基站进行通信，有足够的计算能力来支持信号处理和计算路由。

传感器网络以数据为中心，即在每个采样周期，所有节点都要采集数据传送给簇头节点。采样周期由数据采集、将数据传送给簇头节点和簇头节点将数据传送给基站三个阶段组成。假设每个节点每个采样周期只有一个数据包需要发送且数据包的长度恒定不变。

簇头节点已经给其簇内的成员节点分配好 TDMA 时隙，成员节点发送数据给其簇头节点时，只在其特定的 TDMA 时隙内发送，不会发生多个成员节点同时向簇头节点发送数据而产生数据冲突的情况。

基于簇融合的改进型算法的仿真模型：

网络中所有传感器节点都是同类型的节点并且具有相同的初始能量；节点具有足够的功率与其它节点和汇聚节点进行通信，且具有控制发送功率的能力和通过功率检测来确定距离的能力；每次都是以所需的最小功率与其他节点或基站进行通信，有足够的计算能力来支持信号处理和计算路由。

在每个数据采集过程中，每个节点都采集数据并传给簇头节点。数据采集的过程包括数据采集，经过节点链将数据传递给簇头节点和簇头节点通过多跳将数据传给汇聚节点三个阶段。在每个采样周期中，每个节点只在发送一个不超过最大长度的数据包。

节点链生成数据包或者簇融合数据包在节点中传递时各个节点分配好 TDMA 时隙，时隙表中留有用于组织簇结构，节点查找，簇头间通信的时隙，节点检测到的数据只是在规定的时隙内向下一节点发送，因此不会产生多个节点向一个节

点同时发送数据而产生包碰撞的情况。

### 3.3.2 仿真流程

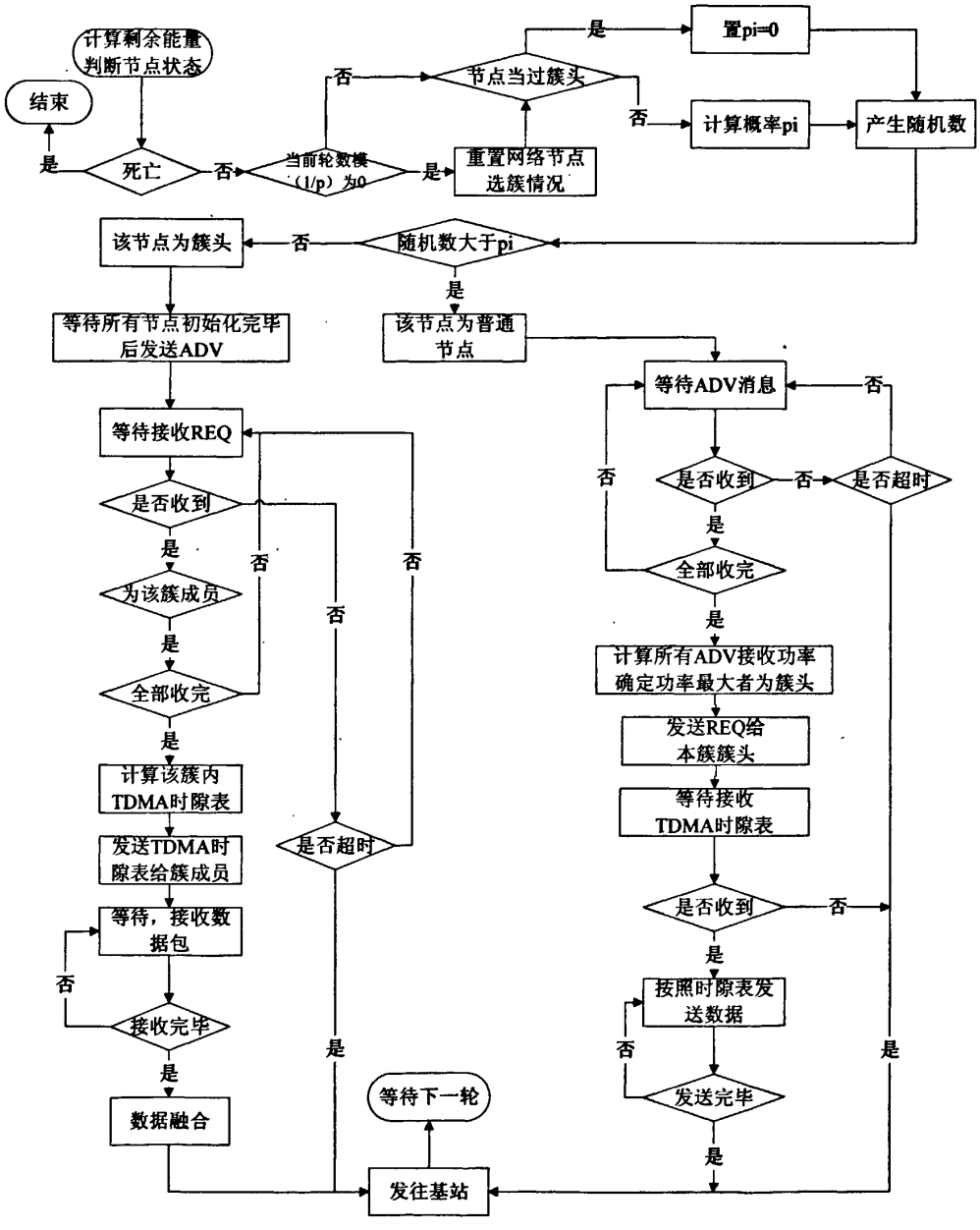


图 3.11 LEACH 协议仿真流程图

LEACH 协议的仿真流程如图 3.11 所示，这个流程图指明了网络中的节点从开始状态到仿真结束的流程，网络中所有节点均按照这个流程来进行，也就是说各个节点都是执行同样的程序，多个并行的程序之间还可以进行交互进而所有节点的相互作用体现出整个网络的行为。

基于簇融合的改进型算法的仿真过程可以分为以下几个部分：

1. 节点初始化，网络各个节点的初始化都相同，主要是对各个节点上的参数进行设置，主要包括节点位置信息，节点能量和当前状态等。

2. 节点位置获取，如图 3.12 所示，网络中节点在初始化完成以后，等待汇聚节点启动创建全局节点树的过程，在全局节点树构建完成后，节点向汇聚节点发送自身位置信息。

3. 汇聚节点计算最优簇头数并建立节点链，根据公式和网络中的节点分布，汇聚节点计算出整个网络中的最优分簇数，并根据网络中各个节点的位置信息对全局节点进行分簇和成链运算，汇聚节点向各个簇头节点发送节点链建立数据包。各个簇头节收到该数据包后，将该数据包在整个节点链的组成节点中依次转发，每个节点在接收到成链信息后都建立自己的数据结构，数据结构中包括是否充当簇头，簇编号，左节点，右节点，数据传输方向，节点链中编号，以及和时隙相关的数据信息。

4. 节点链建立完成后，分簇工作也就完成了，然后各个簇头节点都开始建立簇间多跳的通信结构，这些工作完成后，各个节点就开始进行正常的数据包的收发工作，在这一轮结束时，簇内所有节点竞争簇头位置并完成簇头节点的转移工作。整个流程如图 3.13 所示。

5. 在每一轮各个节点都进行数据收发操作，在这个收发数据包的过程中包含了节点丢失查找数据包。簇头节点接收数据的流程如图 3.14 所示，普通节点收发数据的流程如图 3.15 所示。

6. 在节点链接收到簇融合数据包后，它们将会融合成一个新的节点链，融合的流程如图 3.16 所示。

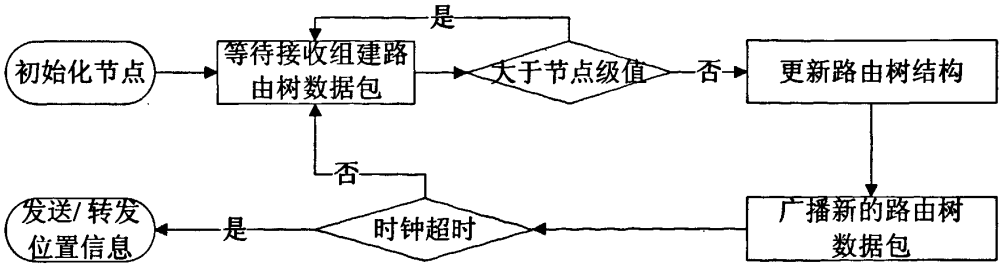


图 3.12 获取位置信息流程图

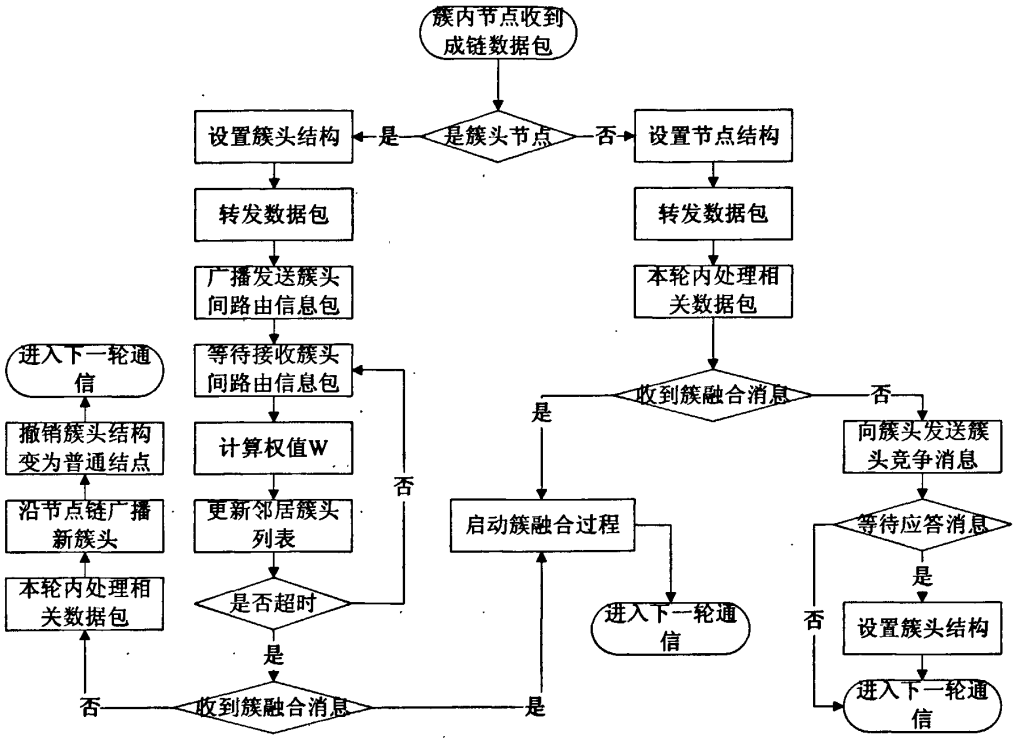


图 3.13 节点正常工作流程图

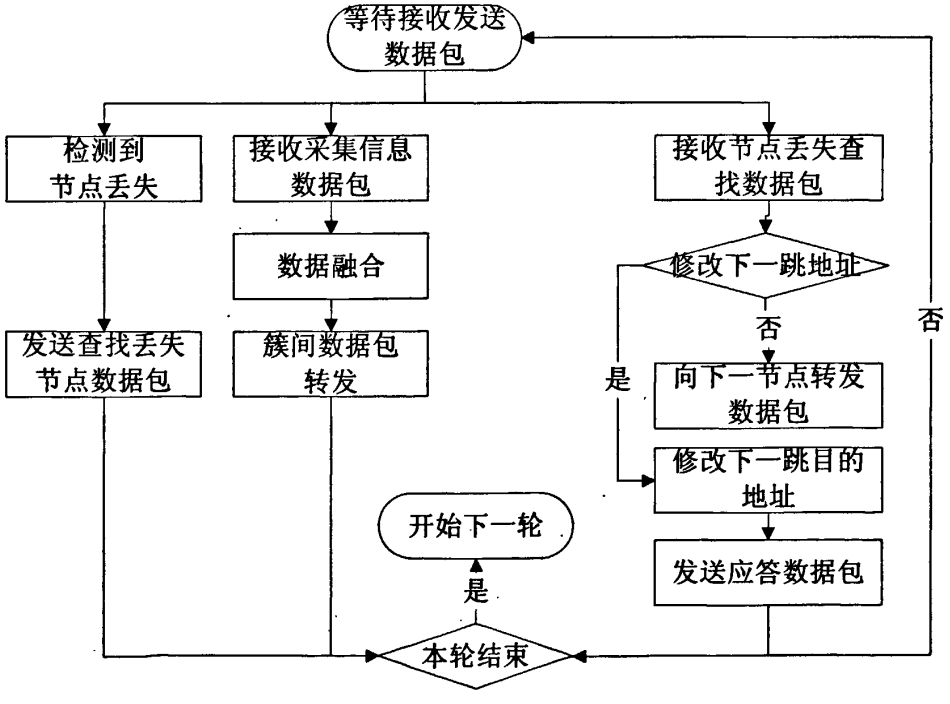


图 3.14 簇头节点收发数据包流程图

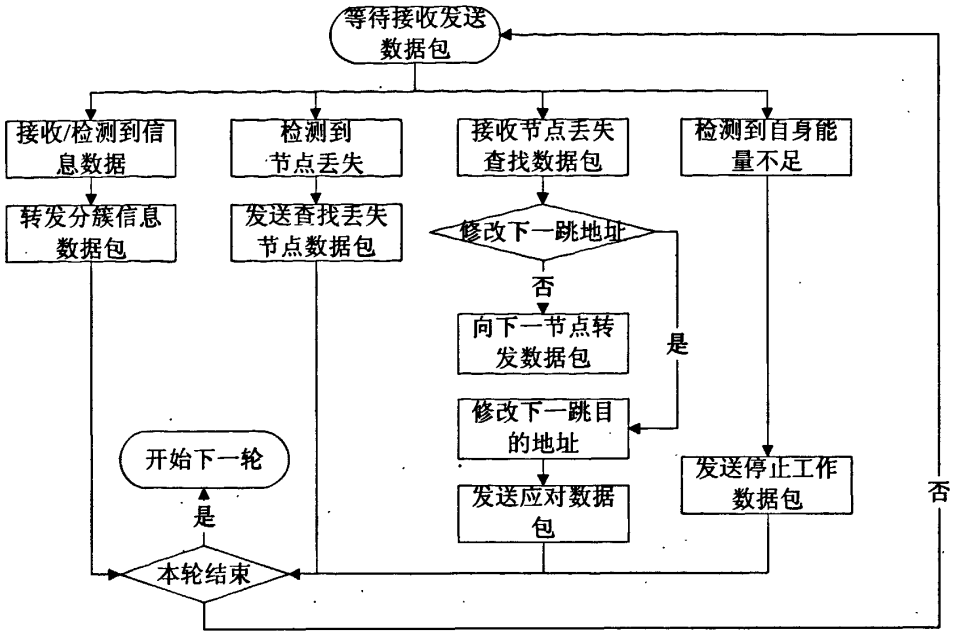


图 3.15 普通节点收发数据包流程图

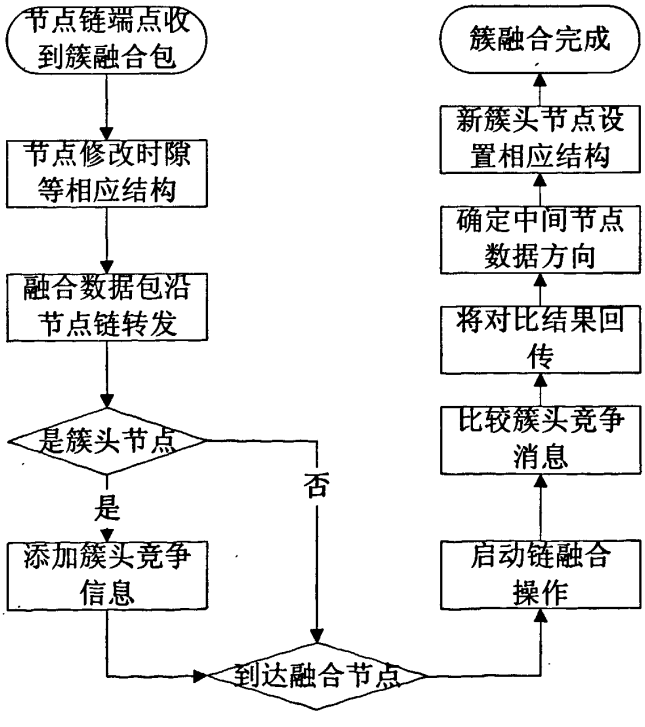


图 3.16 簇融合流程图

### 3.3.3 仿真结果分析

无线传感器网络中，节点的能量资源、计算能力和带宽等资源都非常有限，尤其节点有限的能量直接影响传感器网络的生命周期以及网络的信息质量。为此本文主要从能量效率的角度来分析评价算法性能。传感器网络的生命周期是指从网络启动到不能为观察者提供需要的信息为止所持续的时间，可以用网络生存节点数与网络运行时间的关系来表述。

对使用两种网络协议时网络生命周期的仿真结果如图 3.15 和图 3.16 所示，横坐标表示的是整个网络工作的轮数，纵坐标表示的是在每一轮运行结束后，网络中仍然处于存活状态的节点个数。LEACH 算法的仿真结果如图 3.15 所示，其中死亡节点出现在第 351 轮，节点半数死亡在 790 轮，节点全部死亡在第 949 轮。基于簇融合的改进型算法的仿真结果如图 3.16 所示，其中死亡节点出现在第 760 轮，节点半数死亡在 940 轮，节点全部死亡在第 1034 轮。将死亡节点出现推迟了 309 轮，将节点全部死亡推迟了 85 轮。由此可知，基于簇融合的改进型路由协议节省了能量从而提高了网络的生存周期。经计算可知，LEACH 协议中的节点从第 351 轮开始至 949 轮节点全部死亡，平均每 5.98 轮死亡一个节点，基于簇融合的改进型协议中的节点从第 760 轮开始至 1034 轮节点全部死，平均每 2.74 轮死亡一个节点，显然基于簇融合的改进型协议死亡节点出现的时间晚于 LEACH 协议，但是在出现死亡节点后的节点死亡速率要高于 LEACH 协议，这说明基于簇融合的改进型协议将网络中的能量消耗均衡分配到所有节点上，防止了节点的过早死亡。根据两图可得节点的死亡百分比表如表 3.2 所示。可见，基于簇融合的改进型算法由于考虑了采用了基本固定的分簇方式和全局最优的分簇数，在选择簇头时考虑了节点剩余能量，并且簇头和汇聚节点之间采用多跳通信，从而降低了网络节点的能量消耗，使得能耗更为合理的在网络内分配，避免低能量的节点过早的死亡，有效的延长了整个网络的生命周期。

表 3.2 节点死亡率表

| 节点死亡率 | 5% | 20% | 50% | 100% |
|---|---|---|---|---|
| LEACH 协议仿真轮数 | 415 轮 | 608 轮 | 790 轮 | 949 轮 |
| 改进型协议仿真轮数 | 747 轮 | 840 轮 | 940 轮 | 1034 轮 |

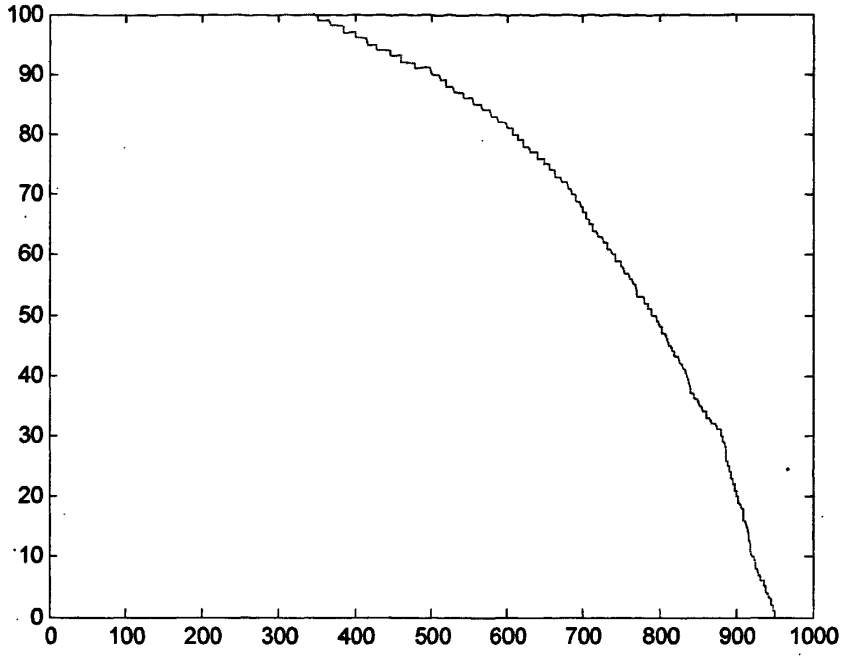


图 3.15 LEACH 算仿真结果

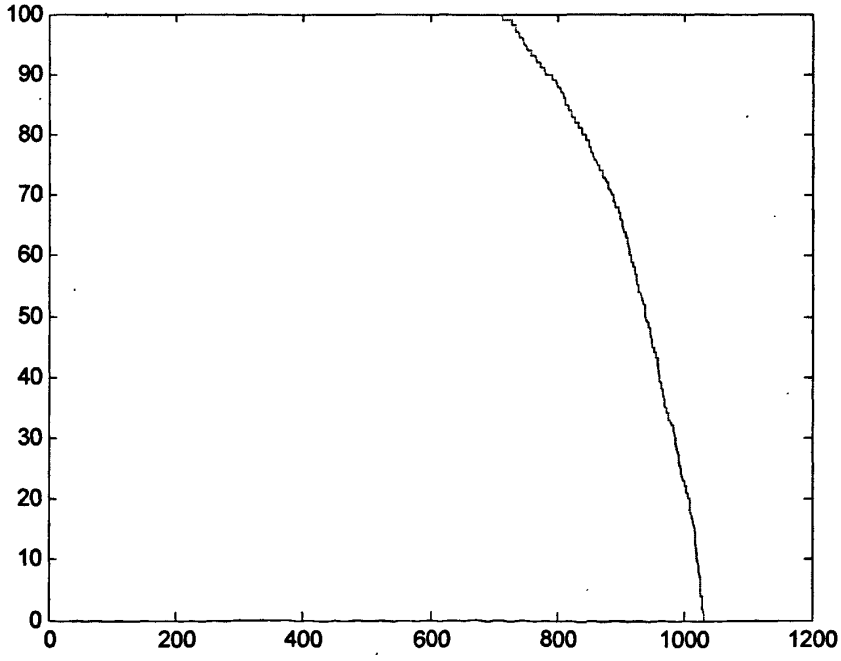


图 3.16 基于簇融合的改进型协议仿真结果

# 第四章 基于簇融合的改进型协议实现

## 4.1 TinyOS 操作系统

适合传感器网络节点的操作系统必须是事件驱动的操作系统而不能是多任务操作系统。因为多任务系统的内部任务切换频率的增加将产生巨大的开销，而事件驱动型系统支持数据流的高效并发，能适应系统的低功耗要求。

适合传感器网络节点的操作系统的核心代码应该比较小，并且运行在很小的空间中，以减少系统对 ROM 的需求量，满足节点内存资源有限的限制。

适合传感器网络节点的操作系统应该在没有任务时使得处理器处于睡眠状态，直到有事件唤醒才去处理事件和与事件相关的任务，然后在进入睡眠，这样可以保证节点大多数时间处于极低功耗的睡眠状态，节约系统能量，满足节点能量有限的要求。

适合传感器网络节点的操作系统应该能有效处理发生频繁、并发程度高、执行过程比较短的逻辑控制流的能力。因为在无线传感器网络中并发性很密集，即可能存在多个需要同时执行的逻辑控制。同时，无线传感器节点模块化程度很高，要求操作系统能够让应用程序方便地对硬件进行控制，且保证在不影响整体开销的情况下，应用程序中的各个部分能够比较方便地进行重新组合。

针对上述无线传感器网络节点操作系统的需求，加州大学伯克利分校科研人员通过比较、分析与实践，设计了 TinyOS 操作系统，基本上能够满足上述要求。在这个操作系统中引入了轻量线程、主动消息、事件驱动模型、组件化编程、硬件抽象层和并行处理的概念，使之更好地满足无线传感器网络运行特点。

TinyOS 通过任务的方式引入了轻量线程的概念，只是对线程进行处理而不处理线程所处的上下文环境，因此任务并没有线程控制模块。TinyOS 系统的任务调度策略采用 FIFO 队列，执行完本任务后才能执行队列中的下一任务，任务是从队列中取出的没有参数的相关函数地址，并且任务都是非抢占的，这就避免了任务切换时将 cpu 的寄存器状态存储到内存中，也不用创建线程控制块之类的结构，为处理能力有限的无线传感器网络节点节省了大量的时间和能量，当所有任务都执行完毕后，系统自动进入了节能状态。TinyOS 中的调度策略是可以修改的，用户只需按照要求编写 TinySchedulerC.nc 和 SchedulerBasicP.nc 模块即可。

主动消息模式是一个面向消息通信的高性能通信模式，早期一般应用于并行和分布式计算系统中，但其基本思想适合传感器网络的需求。在主动消息通信方式中，每一个消息都维护一个应用层的处理器(handler)。当目标节点收到这个消息后，就会把消息中的数据作为参数，并传递给应用层的处理器进行处理。应用层的处理器一般完成消息数据的解包操作、计算处理或发送相应消息等工作，从而

消除了一般通信协议中经常碰到的缓冲区处理方面的困难。主动消息不但可以让应用程序开发者避免使用忙等待的方式等待消息数据的到来，而且使得无线传感器节点的计算和通信重叠，让软件层的通信原语能够与无线传感器网络节点的硬件能力匹配，充分节省无线传感器节点的有限存储空间。为了让主动消息更适应传感器网络的需求，要求主动消息至少提供三个最基本的通信机制：带确认信息的消息传播，有明确的消息地址，消息分发。为了避免网络拥塞，还需要消息处理器能够实现异步机制。

TinyOS 采用事件驱动模式。所谓事件驱动是指系统在一个任务完成后就会触发另一事件并且进入相应的处理函数，并在处理函数运行完以后，进行事件通告。事件驱动的程序设计代码执行路径不是确定的而是响应不同事件执行不同的代码，代码执行的顺序同事件触发的顺序相关，每次运行程序所经过的代码路径千差万别。事件的来源有多种，可以由操作系统触发，也可能由用户触发，应用程序也可以触发事件。这些事件又可分为硬件事件和软件事件，硬件事件是通过硬件中断来触发中断处理程序，软件事件是通过 signal 来触发 event 事件。这完全不同于面向过程程序设计中程序必须按照预定执行路径来执行并且程序自身控制决定执行那一部分代码的方式，从而降低了编程难度。

TinyOS 充分借鉴了软件重用方法中的组件方法，在 TinyOS 中，户可以通过 configuration 和 module 关键字来声明自己定义的组件。这样每个组件只需对外提供标准接口，而用户在忽略接口内部实现情况下就可以设置和调用这些接口。因此快速地将不同来源的多个组件有机的结合成一个符合实际需要的应用程序就成为了可能。TinyOS 只支持 nesC 语言开发的组件，对其它语言开发的组件由于无线传感器节点的系统资源有限不具备组件容器实现的基础而未予支持。

硬件抽象层负责对目标系统的硬件平台进行操作和控制，用于屏蔽不同的硬件特性使得操作系统运行在虚拟的统一的硬件平台接口上，从而简化了程序开发，增强程序的可移植性。它向下与硬件交互，向上与系统服务层的各管理模块提供标准的接口。硬件抽象层可以分为硬件属性抽象模块和硬件行为抽象模块。硬件属性层模块对无线传感器网络中所有底层的硬件资源进行分类、划分，以及同类硬件属性的高度整合，形成通信类属性模块、数据处理融合类属性模块、控制类属性模块、传感类属性模块。硬件行为模块对硬件的行为进行整合划分，形成通信类行为模块、数据处理融合类行为模块、控制类行为模块、传感类行为模块。采用把硬件属性和行为划分开来描述的设计方法,有利于提高操作系统的运行效率,减小系统尺寸和增强跨平台特性。在对 TinyOS 进行移植时只需修改 Makefile 文件中的路径来指明所需参数的硬件实现的文件路径，并在编译时控制 make 命令的参数对不同的平台生产目标代码。

另外，轻量线程为了适应无线传感器节点上数据的突发性而对任务进行了优

化，使得每个任务运行的时间尽量的短以应付在不确定的时间点到达的大量数据。但由于任务调度是通过先进先出的队列完成的，并且任务具有原子性，因此 TinyOS 需要使用硬件中断来应对外部事件到来的不确定性，以及时处理外部事务。同其它操作系统类似，TinyOS 的硬件中断就是对每个中断向量进行映射来完成的。可以响应的外部事件种类同中断向量表的大小成正比。有了任务和中断处理的概念，TinyOS 就实现了并行处理。紧急事件采用中断方式处理，根据不同的优先级来确定事件的处理顺序，在这个过程中更高级的中断可以打断正在处理的低优先级的中断处理程序，从而能对紧急事件作出快速反应。一般任务只需放入 FIFO 队列中依次取出执行。

## 4.2 nesC 语言

TinyOS 是一种面向传感器网络的新型操作系统，最初是用汇编和 C 语言编写的，后来研究人员发现 C 语言不能有效、方便地支持无线传感器网络节点上应用程序开发和操作系统开发，于是将 C 语言进行一定的扩展，提出支持组件化编程的 nesC 语言。nesC 语言把组件化/模块化思想和基于事件驱动模型结合起来。相比 C 语言实现，用 nesC 实现的系统要小很多，nesC 更加适合于传感器网络。nesC 语言引入了接口和组件的概念。

接口是一系列声明的有名函数集合，同时也是连接不同组件之间的纽带，实际上是提供者和使用者组件间形成的一种多功能双向通道，nesC 语言编写程序就是通过接口将不同组件联系起来的。接口的提供者实现了接口的一组功能函数，通过关键 command 声明，称为命令，表示通过这个接口可以完成哪些功能调用。接口中还包括需要使用者实现的一组功能函数，通过关键字 event 声明，称为事件，表示通过本接口可以获得哪些事件通知。

接口是组件间进行通信的规范，它是由 interface 关键字声明的，其声明语法如下：

```
nesC-file:
    includes-listopt interface
    ...
interface interfaceName
{
    command result_t commandName(pram p);
    event result_t eventName(pram p);
}
```

根据接口声明是否带有参数，接口分为简单接口和可参数化接口两种，如

interface Receive as Snoop[uint8_t id]。这就是参数化的接口，可通过 id 对同一接口不同的句柄事件进行分别处理。

组件就是实现一定逻辑功能的模块，是 nesC 程序基本单位。整个 TinyOS 节点中程序是由许多组件构成，nesC 中的组件包括模块(module)和配件(configuration)。nesC 中通过关键字 configuration 和 module 来声明不同类型的组件。配件通过 Configuration 关键字声明，配件的主要功能是指明组件中使用其它组件的接口连接情况。在配置文件中必须定义其功能实现的核心模块组件。模块通过 Module 关键字声明，模块中包含具体实现接口中的命令和事件，它实际上是组件的逻辑功能实体。任务也是在模块中声明和实现的，它是一个返回型为 void 且无参数的 task 存储类型的函数，使用关键字 task 来预先声明一个任务，使用带 post 前缀的任务调用来提交(post)任务。

组件可以使用接口也可以对外提供接口，组件通过 provides 关键字来对外声明自己提供的接口，通过 uses 关键字来声明完成自身功能而使用的其它组件所提供的接口。在组件对外提供的接口中包含的 command 声明函数，应该在组件的核心模块中给出函数的实现，也应该对 event 声明的函数进行 signal 通告。在组件使用的接口中，应该在核心组件中直接调用接口中 command 声明的函数，也应该对 event 声明的函数进行响应。组件根据自己的功能和需要来决定自己需要提供和使用哪些接口，而没有要求必须提供和使用哪些接口。一个组件可以使用或提供多个接口或者同一接口的多个实例。

从层次的角度来看，nesC 中实现的组件可以分为三种不同类型：硬件抽象组件、合成组件和高层组件。硬件抽象组件是对硬件功能的抽象，如定时器等；合成组件，完成数据处理、通信处理等相对抽象的操作，在概念上形成一个更加强大的功能模块；高层组件指得是网络节点中的应用程序，它们形成独立的功能模块。

连接用来把定义的元素（接口、命令、事件等）联系在一起，以完成互相之间的调用。连接的语法定义如下：

Connection-list:

Connection

Connection-list connection

Connection:

Endpoint=Endpoint

Endpoint→Endpoint

Endpoint←Endpoint

nesC 中有三种连接语句

Endpoint1=Endpoint2，这是一种包含一个外部规范元素的链接，这种连接语句

有效地使两个规范元素等价。Endpoint1，Endpoint2 可以一个是内部的，一个是外部的，同时被提供或使用的；也可以都是外部的，而且一个是被提供的，另一个是被使用的。Endpoint1→Endpoint2，这是一种包含两个内部规范元素的链接，把 Endpoint1 定义的被使用的规范元素链接到 Endpoint2 定义的被提供的规范元素上。连接 Endpoint2←Endpoint1 等价于连接 Endpoint1→Endpoint2。

这三种连接中，被定义的两个规范元素必须是兼容的，即它们必须或都是命令，或都是事件，或都是接口实例。

通过以上对接口、组件和连接的分析，程序模块之间好像是绝对的分开：只能通过接口来实现信息流动。组件中除拥有接口中声明的函数系列外，可以定义特定函数，通过属性声明，改变其作用域。

nesC 语言使用_attribute_来声明一些变量和函数的属性，声明放在函数的后面，声明格式如下：

Datatype function()_attribute_(attribute_list)

nesC 支持三种类型的属性：

（1）C 通常在模板顶层配置文件中，使其成为全局域，而不是模块所在的域。函数将声明为全局函数。

（2）Spontaneous 表示此函数不仅可以在本文件中访问到，也可以在其它文件中访问到。

（3）combine 指定通过 typedef 中定义的数据类型的组合函数，组合函数指明如何组合命令和事件。

## 4.3 改进型协议的数据结构设计

LinkNode 是簇成员节点上构成节点链的数据结构，包括节点链上的左邻居节点 ID，右邻居节点 ID，数据传输方向，节点处于节点链的相对位置和节点所在的簇编号。如果节点处于节点链的端点处则将左邻居节点或者右邻居节点编号设为 0。

```
typedef struct LinkNode{
    uint16 leftnode;
    uint16 rightnode;
    uint16 direct;
    uint16 LinkLocation;
    uint16 clusterNO;
    bool isHead;
} LinkNode;
```

Cluster 是簇头节点数据结构，簇头节点用其来选择簇间通信的下一跳目的节点。其中包括三个节点和每个节点相关的 W 权值，如果相邻节点的编号值设为 0，则说明该节点从邻居簇头路由表中删除了。

```
typedef struct Cluster{
    uint16 neighbour1;
    float W1;
    uint16 neighbour2;
    float W2;
    uint16 neighbour3;
    float W3;
} Cluster;
```

TOS_Msg 是基本的通信数据结构，它包括目的地址，消息类型，长度，载荷等。最大载荷 TOSH_DATA_LENGTH 定义为 29 个字节。

```
typedef struct TOS_Msg {
    uint16_t addr;
    uint8_t type;
    uint8_t group;
    uint8_t length;
    int8_t data[TOSH_DATA_LENGTH];
    uint16_t crc;
    uint16_t strength;
    uint8_t ack;
    uint16_t time;
    uint8_t sendSecurityMode;
    uint8_t receiveSecurityMode;
}TOS_Msg;
```

TREE_Msg 是用来组建节点树的信息数据包，包括数据包类型，源地址，数据包级值。TREE_Msg 封装在 TOS_Msg 的 data 域中。

```
typedef struct TREE_Msg {
    uint8_t type;
    uint8_t sourceID;
    uint8_t level;
} TREE_Msg;
```

LocationMsg 中包含各个节点向簇头节点发送的节点位置信息，主要包含节点 ID 和位置信息。LocationMsg 信息封装在 TOS_Msg 的 data 域中。

```
typedef struct LocationMsg{
    uint8_t type;
    uint8_t ID;
    float x;
    float y;
} LocationMsg;
```

ClusterMsg 中包含汇聚节点向簇头节点发送分簇信息的路径，簇编号，簇头节点 ID，簇成员编号，簇中节点组成节点链的顺序，消息序列号，数据包类型。ClusterMsg 封装在 TOS_Msg 的 data 域中。

```
typedef struct ClusterMsg {
    uint8_t type;
    uint8_t clusterNO;
    uint8_t seqno;
    uint8_t path[PATH_LENGTH];//PATH_LENGT 定义为 10
    uint8_t node[TOSH_DATA_LENGTH-PATH_LENGTH-2];
} ClusterMsg;
```

其中，通过 path 数组来指明数据包的传输路径，按照其中节点顺序依次将数据包传送到目的节点。node 数组中第一个字节存放的是簇头节点编号，按照 node 数组中节点的排列顺序，依次组成节点链。

ChainMsg 是分簇的簇头节点接收完分簇信息后，分簇内建立节点链的数据包。数据包中包括消息类型，簇编号以及建立节点链的信息。ChianMsg 封装在 TOS_Msg 的 data 域中。ChainNode 数组中储存着节点链的标号，其中标号的顺序就是建立节点链的顺序。

```
typedef struct ChainMsg {
    uint8_t type;
    uint8_t clusterNO;
    uint8_t ChainNode[TOSH_DATA_LENGTH - 2];
} ChainMsg;
```

MTEMsg 中包含簇头间建立多跳路由的基本信息。各个簇的簇头通过广播和接收 MTEMsg 来建立簇头到汇聚节点的多跳路径。MTEMsg 数据包中包括数据包类型，发送节点的剩余能量，发送节点到达汇聚节点的距离，簇编号，簇头节点的 ID。MTEMsg 封装在 TOS_Msg 的 data 域中。

```
typedef struct MTEMsg {
    uint8_t type;
    uint8_t energy;
```

```
    uint16_t distance;
    uint8_t clusterNO;
    uint8_t headID;
} MTEMsg;
```

StopForward_Msg 消息是当处于转发状态的簇头节点用于转发的数据包所消耗的能量值大于$\Delta E$时，向周围节点广播的停止转发数据包。StopForward_Msg 数据包中包括数据类型，发送节点的 ID，簇编号。StopForward_Msg 消息封装在 TOS_Msg 的 data 域中。

```
typedef struct StopForward_Msg {
    uint8_t type;
    uint8_t clusterNO;
    uint8_t headID;
} StopForward_Msg;
```

当通过 StopForward_Msg 数据包中包含的 headID 节点转发数据包的节点收到本数据包后，该节点就停止向 headID 节点发送数据包，并将 headID 节点从自己的簇头转发数据结构中删除。

Data_Msg 消息中包含簇内节点检测到的信息，Data_Msg 消息只是在簇内进行传播。Data_Msg 数据包中包含数据类型，发送节点 ID，目的节点 ID，簇编号，数据信息。Data_Msg 消息封装在 TOS_Msg 的 data 域中。

```
typedef struct Data_Msg {
    uint8_t type;
    uint8_t sourceID;
    uint8_t targetID;
    uint8_t clusterNO;
    uint8_t data[TOSH_DATA_LENGTH - 4];
    uint8_t energy;
    float t;//转发信息与检测到信息的比值。
} Data_Msg;
```

ClusterData_Msg 消息是簇内节点检测到的消息，经过簇头节点的数据融合后产生的数据包，ClusterData_Msg 在簇间经过多跳传播，发送给汇聚节点。ClusterData_Msg 中包含数据类型，发送节点 ID，目的节点 ID，簇编号，数据信息。ClusterData_Msg 消息封装在 TOS_Msg 的 data 域中。

```
typedef struct ClusterData_Msg {
    uint8_t type;
    uint8_t sourceID;
```

```
    uint8_t targetID;
    uint8_t clusterNO;
    uint8_t data[TOSH_DATA_LENGTH-4];
} ClusterData_Msg;
```

ClusterMerge_Msg 消息是汇聚节点检测到当前分簇数大于最优分簇数时，发送的簇融合信息。收到 ClusterMerge_Msg 的两个簇会融合在一起生产一个新的簇。ClusterMerge_Msg 中包含数据类型，目的节点，路径信息，融合节点标号，方向信息。ClusterMerge_Msg 消息封装在 TOS_Msg 的 data 域中。

```
typedef struct ClusterMerge_Msg {
    uint8_t type;
    uint8_t targetID;
    uint8_t path[TOSH_DATA_LENGTH-4];
    uint8_t mergeID;
    uint8_t direction;
} ClusterData_Msg;
```

Head_Msg 消息是簇头广播数据包，上一轮簇头节点向簇内节点广播本轮簇头节点的数据包。Head_Msg 中包含数据包类型，目的节点，簇标号。Head_Msg 消息封装在 TOS_Msg 的 data 域中。

```
typedef struct Head_Msg {
    uint8_t type;
    uint8_t targetID;
    uint8_t clusterNO;
} Head_Msg;
```

ClusterHead_Msg 消息是在发生簇融合时，发起簇融合的簇头结点向被融合的簇的簇头节点发送的簇头竞争数据包。ClusterHead_Msg 消息中包含数据包类型，竞争节点 ID，剩余能量，检测到数据与转发数据比。ClusterHead_Msg 消息封装在 TOS_Msg 的 data 域中。

```
typedef struct ClusterHead_Msg {
    uint8_t type;
    uint8_t competeID;
    uint8_t energy;
    float t;
} ClusterHead_Msg;
```

Quit_Msg 消息是当节点能量耗尽，不足以支持其继续工作时，向周围节点广播的数据包。节点链中相邻节点接收到本数据包后，会将该节点从相应的结构中

删除。Quit_Msg 消息中包含数据包类型，退出节点的 ID，簇编号。Quit_Msg 消息封装在 TOS_Msg 的 data 域中。

```
typedef struct Quit_Msg {
    uint8_t type;
    uint8_t QuitID;
    uint8_t ClusterID;
} Quit_Msg;
```

SearchNode_Msg 消息是当节点链上的节点发现某节点非正常消失后发送的丢失节点查找消息。SearchNode_Msg 消息中包括消息类型，自身节点 ID，丢失节点 ID，自身处于节点链上的相对位，分簇 ID。SearchNode_Msg 消息封装在 TOS_Msg 的 data 域中。

```
typedef struct SearchNode_Msg {
    uint8_t type;
    uint8_t ID;
    uint8_t lostID;
    uint8_t location;
    uint8_t clusterID;
} SearchNode_Msg;
```

SearchHead_Msg 消息是由于某种原因导致了汇聚节点在一段时间内未收到某个簇的簇头节点的消息而发送的查找簇头节点数据包。SearchHead_Msg 消息中包括数据包类型，目的节点 ID，路径消息，簇编号。SearchHead_Msg 消息封装在 TOS_Msg 的 data 域中。

```
typedef struct SearchHead_Msg {
    uint8_t type;
    uint8_t targetID;
    uint8_t path[];
    uint8_t clusterNO;
} SearchHead_Msg;
```

## 4.4 改进型协议的模块和接口设计

改进型协议的功能模块划分如图 4.1 所示。改进型路由协议由全局树构建模块，簇间通信模块，簇维护模块，簇内信息处理模块和簇引擎模块组成。

全局树模块负责在传感器网络中构建全局节点树，分为节点树建立子模块和节点树维护子模块。其中节点树建立子模块负责建立全局节点树，并且将节点的

位置信息传递给汇聚节点；节点树维护子模块负责当有节点由于某种原因退出节点树时通知汇聚节点。

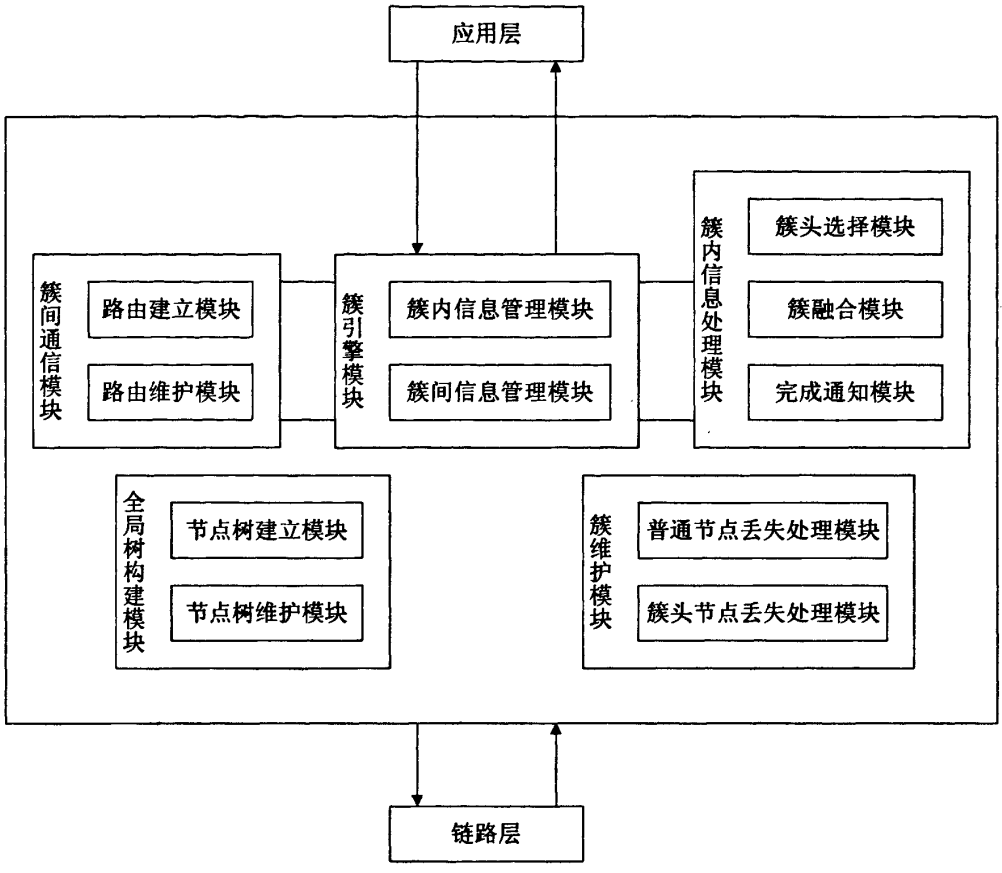


图 4.1 改进型协议功能模块图

簇维护模块完成使得由于某节点突然不能工作导致丢失的正常节点能够重新加入网络的拓扑结构的工作。该模块分为普通结点丢失处理模块和簇头节点丢失处理模块。普通节点丢失处理模块负责找回一般节点消失导致丢失的正常节点；簇头节点丢失处理模块负责找回由于簇头节点消失导致的整个簇的节点。

簇内信息处理模块使得分簇内所有节点按照协议正常地工作，分为簇头选择子模块，簇融合子模块和簇组织完成通知子模块。其中，簇头选择子模块是在节点部署结束后，完成簇头的选举工作；簇融合子模块完成在簇接收到簇融合数据包将两个簇合成一个新簇的工作；簇组织完成通知子模块是在簇头选举完成后或者簇融合完成后，将簇内相关信息反馈给簇引擎模块。

簇间通信模块完成整个网络各个簇的数据包的转发工作，分为路由建立和路由表维护两个子模块。其中，路由建立子模块是在簇组织结束后，完成簇头间路由建立工作；路由表维护子模块记录并更新节点的路由表，并负责将路由相关信息反馈给簇引擎模块。

簇引擎模块是整个路由协议与上下层的联系纽带，分为簇内信息管理子模块和簇间路由信息管理子模块，当部署的节点与 Sink 节点可通信时，簇引擎模块通过自身对于簇和簇间路由相关信息的记录，实现对上层的服务。

因为 TinyOS 是基于事件驱动机制的操作系统，所以使用 nesC 语言实现的路由协议是由各个相对独立的组件构成的。当簇组织完成或簇间路由建立完成后使用事件触发的方式通知簇引擎模块。而簇维护模块也是独立工作的组件，是由计时器溢出或者满足某种特定条件等事件触发工作。

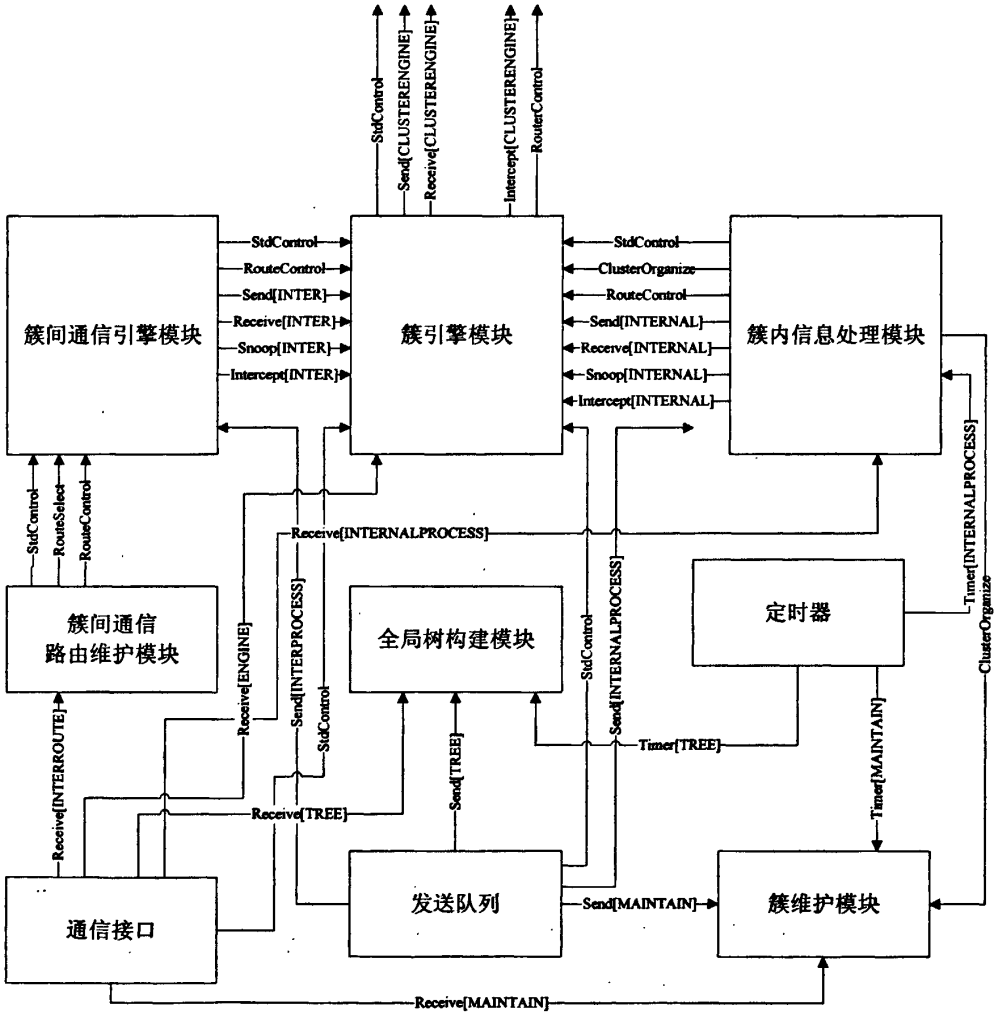


图 4.2 改进型协议的模块和接口图

Configuration 是 TinyOS 和 nesC 语言中特有的概念。在组建模型中每个组件通过定义好的组件接口与其它组件交互，并组成一个更大的组件。组成更大组件的各个模块之间的松耦合便于程序的改动，具有与面向对象语言类似的优点。改进型的路由协议总体上是一个配置，处于应用层程序的下方，链路层的上方。

TinyOS 的改进型路由协议总体结构框如图 4.2 所示。

其中簇引擎模块负责同应用层的通信，对外提供 StdControl 接口，Receive 接口，Send 接口，routerControl 接口和 intercept 接口。其中 StdControl 是 TinyOS 中的标准控制接口，包括 command error_t start()用于启动本组件及其子组件和 command error_t stop()用于关闭本组件及其相关子组件。

Send[]是 TinyOS 中的标准高层发送消息接口，包括了一系列与发送消息有关的操作，是一个可参数化的接口，在簇引擎模块中，用 CLUSTERENGINE 来参数化。包括以下操作：

command error_t send(message_t * msg,uint8_t len)用于发送一个长度为 len 的数据包。

command error_t cancel(message_t *msg)用于取消某数据包的发送请求。

event void sendDone(message_t*msg,error_t error)用于响应已接受发送请求触发的信号。

command uint8_t maxPayloadLength()用于获得本层数据包的最大负载长度。

command void *getPayload(message_t *msg)用于获得数据包负载区的指针。

Receive[]是 TinyOS 中的标准高层接收消息接口，包括一系列与接收消息有关的操作，是一个可以参数化的接口，在簇引擎模块中，用 CLUSTERENGINE 来参数化。包括以下操作：

event message_t* receive(message_t* msg, void* payload, uint8_t len)用于收到数据包触发的信号。

command void* getPayload(message_t* msg, uint8_t* len)用于返回数据包中负载区域的指针。

command uint8_t payloadLength(message_t* msg)用于返回消息负载的长度。

Intercept[]是 TinyOS 中的标准本地节点截获消息接口，包括一系列与截获消息有关的操作，是一个可参数化的接口在簇引擎模块中，用 CLUSTERENGINE 来参数化。包括以下操作：

event bool forward(message_t* msg, void* payload, uint16_t len)用于收到一个数据包而这个数据包应被转发到另一个目的地而触发的信号。

RouteControl 接口，是 TinyOS 中的标准接口，包括与路由信息有关的操作：

command uint16_t getParent()用于获得父节点地址。

command uint8_t getDepth()用于获得节点在网络中的深度。

command uint8_t getOccupancy()用于获得路由转发队列的长度。

command uint8_t getQuality()用于获得到父节点的链路质量。

command error_t setUpdateInterval(uint16_t Interval)用于设置路由组件内部更新的事件间隔。

command error_t manualUpdate()用于更新路由状态。

簇内信息处理模块主要负责簇内部的数据包发送、接收和路由选择等工作。对外提供 StdControl 接口，clusterOrganize 接口，RouterControl 接口、Send 接口、Receive 接口、Snoop 接口和 Intercept 接口。ClusterOrganize 接口包括了一系列与簇组织有关的操作。定义如下：

```
interface ClusterOrganize {
    command result_t start();
    command result_t stop();
    event result_t ClusterOrganizeDone()
    event void newHead();//新簇头收到簇头转换数据包触发的事件
    event void newDirect();//中间节点收到簇头转换数据包触发的事件
    command result_t HeadChange();//簇头转换命令
    event void HeadChangeDone();//簇头转换完成触发的事件
    command result_t NextHead();//确定哪个节点作为下一个簇头
    command bool isHead();//判断当前节点是否是簇头
    event void changeLeftNode();//收到节点退出数据包触发的改变节点链左邻居的事件
    event void changeRightNode();//收到节点退出数据包触发的改变节点链右邻居的事件
}
```

Snoop[]是 TinyOS 中的标准本地节点截获异地发送的消息接口，包括了一系列与截获消息有关的操作，是一个可参数化的接口。它和 Receive 接口，Send 接口，Intercept 接口一样，用 INTERNALPROCESS 来参数化。

定时器模块为各个模块提供定时服务，对外提供 Timer 接口 Timer 是 TinyOS 中的标准接口，包括了一系列与定时器有关的操作。

command void startPeriodic(uint32_t dt);用于启动周期性定时器

command void startOneShot(uint32_t dt);用于启动一次定时器

command void stop();用于停止定时器

event void fired();用于定时器到期时触发的信号

command bool isRunning();用于判断定时器是否运行

command bool isOneShot();用于判断定时器是否是一次定时器

command void startPeriodicAt(uint32_t t0, uint32_t dt);用于在时间 t0 启动定时间隔为 dt 的周期性定时器

command void startOneShotAt(uint32_t t0, uint32_t dt);用于在时间 t0 启动时间间隔为 dt 的一次定时器

command uint32_t getNow();用于获得当前时间

command uint32_t gett0();用于获得定时器起始时间

command uint32_t getdt();用于获得定时器定时间隔

簇间通信引擎模块用于簇间数据包传输，对外提供 StdControl 接口，RouteControl 接口，Send 接口，Receive 接口，Snoop 接口和 Intercept 接口。所有可参数化的接口用 INTERCLUSTERCOM 进行参数化。

RouteSelect 是 TinyOS 中的标准接口，包括了一系列与根据消息寻找路由有关的操作：

command bool isActive(); 用于判断是否是合法的路由

command error_t selectRoute(message_t* msg, uint8_t resend);用于为数据包选择一个路由并为其填充路由信息

command error_t initializeFields(message_t* msg);用于初始化数据包的路由相关域。

command uint8_t* getBuffer(message_t* msg, uint16_t* len);用于提供数据包内的数据域指针

簇间通信路由维护模块用于维护簇头节点和汇聚节点之间的多跳通信，完成邻居簇头节点的权值计算工作，在收到节点停止转发的数据包时，将该节点从路由表中删除。它对外提供 StdControl 接口，RouteSelect 接口和 RouteControl 接口。全局路由树模块负责建立全局路由树，并且负责将节点的位置信息转发到汇聚节点。簇维护模块负责簇头轮换、节点丢失查找等工作，对外提供 ClusterOrganize 接口。数据接收模块和发送队列模块分别和底层组件进行通信，为其它模块提供发送和接收数据包的接口，各个可参数化的接口分别用不同的参数进行参数化。

## 4.5 改进型协议实现的测试

### 4.5.1 硬件平台与编译环境

基于簇融合的改进型协议代码实现的目标硬件平台是使用 MCS-51 单片机作为处理器，CC2420 为射频芯片的自制节点。数据链路层可以采用 CSMA/CA 协议和有保证时隙数据传输。由于节点上不具有功率检测以确定节点间距离的模块，所以对协议的实现进行测试时将节点到汇聚节点的距离和节点之间的距离在代码中通过查表得出。

基于簇融合的改进型协议是在 XUBUNTU 中开发的，其中的 TinyOS 系统是安装好的，但是由于 GCC 不能支持对 MCS-51 系列的处理器进行交叉编译，并且由于 GCC 语法风格和 ANSI C 不完全一致，这使得 nesC 预编译器输出的 C 程序文件无法被大多数的编译器直接使用。因此需要将 TinyOS 移植到 GCC 不支持的

平台上需要进行 C51 编译器的移植。

由于在 LINUX 下，GCC 及其衍生编译器(如 MSPGCC)由开源组织维护，往往在更新速度上无法跟上处理器的发展。时至今日，虽然 GCC 可以很好的支持 AVR 系列处理器，但其并不支持对 MCS-51 系列核心处理器的交叉编译。此外，由于 GCC 的语法风格和 ANSI C 不完全一致，nesC 预编译器输出的 C 程序文件无法直接被大多数的编译器直接使用，从而限制了 TinyOS 在不同平台尤其是基于新处理器型号的平台上的应用。因此，将 TinyOS 移植到 GCC 不支持的平台上需要事先进行 C51 编译器的移植。TinyOS 官方推荐使用 SDCC 或者 Keil 作为编译工具，但是考虑到 SDCC 编译器以及相关工具在 TinyOS 平台上的测试并不成熟，所以选择 Keil 作为编译工具。Keil 是 C51 系列处理器最常用的编译器，支持所有兼容 C51 系列的处理器。为了保留 TinyOS 需要对工具链进行修改和扩展，这集中在 nesC 预编译器之后和 Keil 编译-连接器之前。这里的解决方案是通过加入一级预编译程序将 GCC 风格的 C 程序转换为 Keil 风格的 C 程序。这个过程中涉及大量的文本处理，所以采用 perl 脚本进行处理。TinyOS 中使用 GNU make 进行程序管理，由于 Keil 工具链能够提供完善的命令行调用接口，所以通过在 Linux 平台上安装 Windows 模拟器 wine，这样就可以很容易的将 Keil 工具链集成到 TinyOS 的 GNU make 体系中，从而完成整个 C51 的编译体系。

整个编译过程概述如下：先使用 ncc 将整个工程进行编译，生成 app.c 文件，之后，通过 perl 脚本语言将 GCC 风格的 app.c 文件改为 Keil 风格的 C 文件，然后再通过 Keil 将生成的 app.c 文件进行进行编译，生成 16 进制文件。最后，将该 16 进制文件下载至目标板。

将编译完成的 hex 文件烧写到节点时，将计算机和节点通过串口相连，采用 STCISP 作为烧写工具。运行 STC-ISP.exe 后，将其中 MCU Type 设为 STC89C52RC，串口的最高波特率设为 115200，点击 Download 烧写需要的目标文件。

### 4.5.2 测试环境与测试结果

测试时，通过汇聚节点收到的数据判断网络的工作情况。用一个节点通过串口和一台主机相连，主机通过这个节点收发信息，并且通过 USB 口为该节点供电。这台主机和节点就构成了汇聚节点。在主机上用 QT 和 C++编写了测试程序，用来发送数据包和监测网络的运行情况。另外还通过节点 ID 为 1-10 的节点进行组网操作，它们分布在实验平台上。由于硬件平台不支持获取位置信息和通过接收数据计算距离，节点的位置信息和与主机的距离是记录在节点内的。在节点部署完成后，就可运行测试程序，界面如图 4.3 所示：

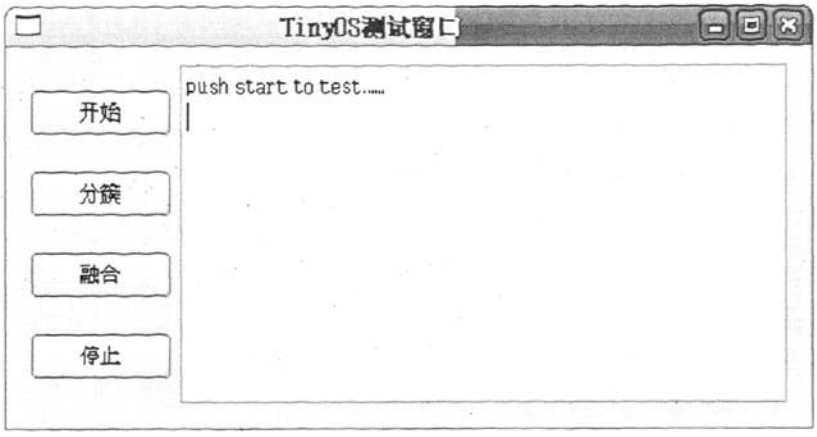


图 4.3

点击开始按钮，测试程序启动。汇聚节点向所有网络节点广播启动数据包和生成树数据包，界面如图 4.4 所示：

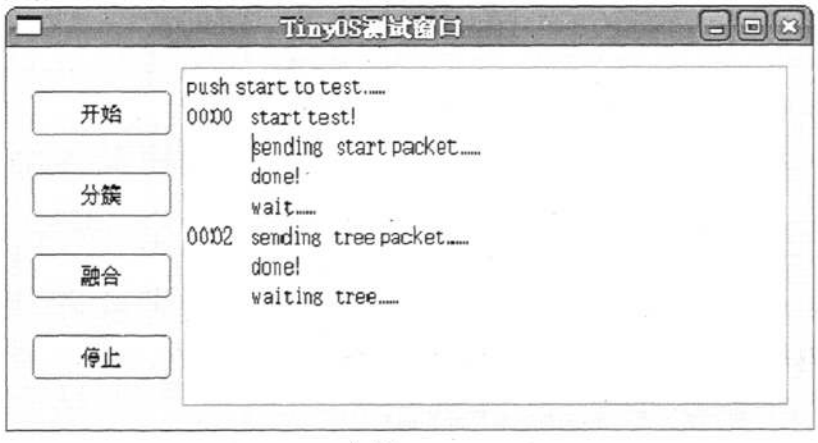


图 4.4

网络中的节点收到启动数据包和生成树的数据包之后，通过泛洪的方式生成全局节点树，所有节点沿着节点树向汇聚节点报告位置信息，结果如图 4.5 所示：

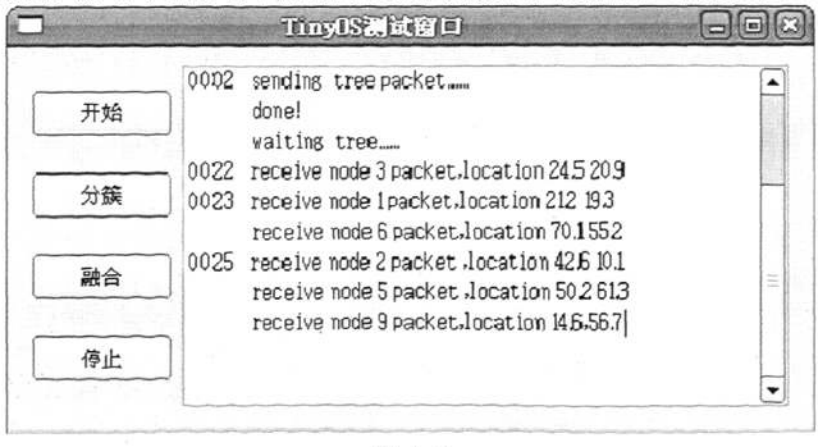


图 4.5

所有节点的位置信息收集完全后，点击分簇按钮对所有节点进行分簇。由于节点数量有限，为了对网络节点上的功能进行测试，测试程序将所有节点分为两个簇。结果如图 4.6 所示：

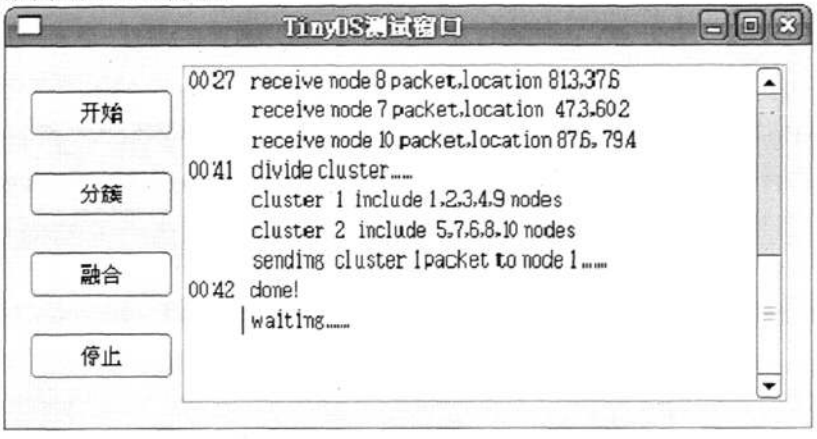


图 4.6

分簇完成后，节点 1、2、3、4、9 属于 1 簇，节点 5、7、6、8、10 属于 2 簇。传感器网络中节点正常工作和簇头轮换的结果如图 4.7 和图 4.8 所示：

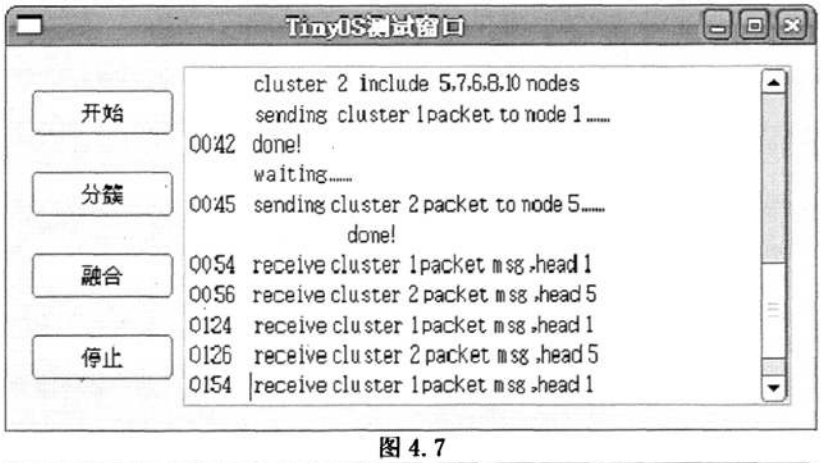


图 4.7

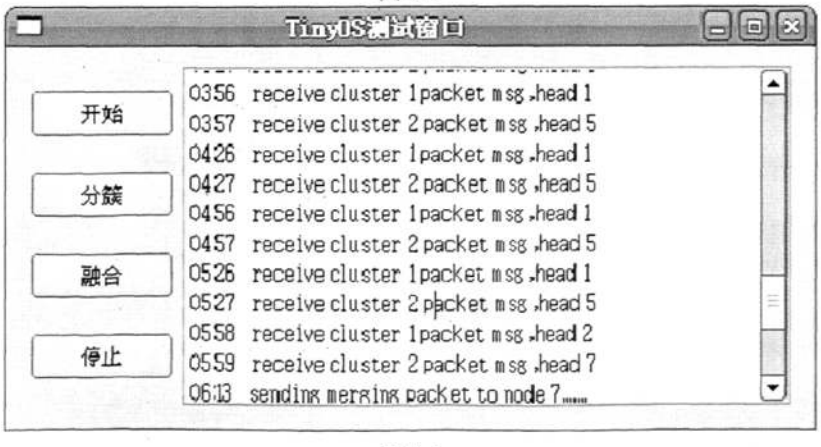


图 4.8

在图 4.8 中可以看出簇 1 在 05：58 时簇头节点从节点 1 换为节点 2，簇 2 在 05：59 时簇头节点从节点 5 换为节点 7。在点击融合按钮后，汇聚节点发送簇融合数据包如图 4.9 所示：

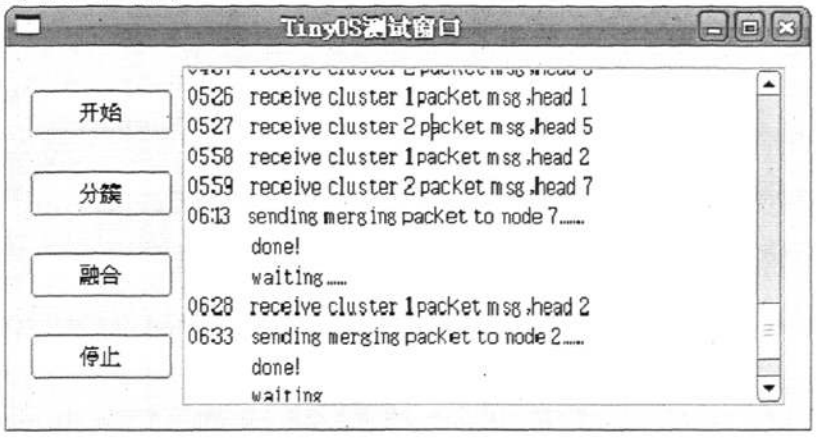


图 4.9

网络中的节点收到簇合并数据包后，就开始了簇合并过程，该过程完成后，簇 1 和簇 2 合并为新的簇 1，簇头节点是 7 号节点。其正常工作向汇聚节点发送数据包如图 4.10 所示：

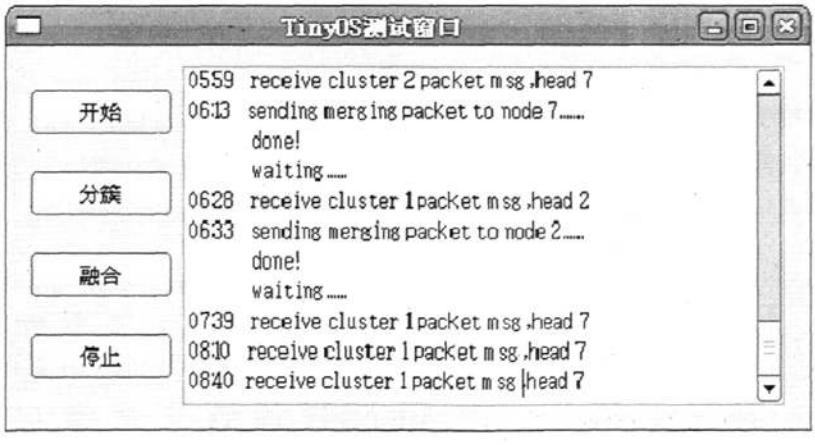


图 4.10

综上所述，在 TinyOS 实现的改进型协议可以完成如下功能：向汇聚节点报告位置信息的功能，根据汇聚节点的分簇信息形成节点链，向汇聚节点发送数据，进行簇头轮换，进行簇融合。

# 第五章 总结与展望

本文首先介绍了无线传感器网络的研究背景与意义,对无线传感器网络协议的研究现状进行了概述。简要分析了无线传感器网络的协议栈模型，无线传感器网络协议的设计目标和几种典型的无线传感器网络路由协议，然后着重研究了典型的层次型路由协议——LEACH 协议，指出了 LEACH 协议存在的不足，针对其不足提出了一种基于簇融合的改进型路由协议。最后对 LEACH 协议和基于簇融合的改进型协议进行了仿真对比研究，并给出了改进型协议在 TinyOS 操作系统上的 nesC 语言的设计实现和相应的测试。

本文从事了如下研究工作：

1.针对 LEACH 算法中的分簇数量具有概率性，而且不能保证簇头数量的最优化；簇头选择是严格等概率的，而没有考虑节点的剩余能量和采集信息的频率；簇头和汇聚节点直接通信，不合适大型网络中应用等缺点，本文提出了汇聚节点根据全局节点的位置信息计算当前的最优分簇数，采用固定分簇的方法来确定簇头的数量，簇内结构采用链式结构；并通过簇融合的方式改变分簇数，使其达到最优；在选择簇头时同时考虑节点的剩余能量和采集信息的数量；簇间信息的传递采用多跳路由协议，降低距离汇聚节点较远的簇头节点的能量消耗的基于簇融合的改进型算法。

2.用 MATLAB 对 LEACH 协议和基于簇融合的改进型协议进行了仿真研究，并对仿真结果进行了对比分析，发现基于簇融合的改进型协议在能量消耗、节点死亡率和网络生存周期方面要优于 LEACH 协议。在 TinyOS 上用 nesC 实现了改进型协议的数据结构设计，模块和接口设计，完成了基于簇融合的改进型协议的代码实现，并进行了相应的功能测试。

由于研究的时间和我本人的能力有限，本文的研究工作还不是很完善，需要进一步探索以待提高。

在设计改进型协议的过程当中，我们重点考虑的是节点能量的消耗问题，以确保网络有更长的生命周期，而对于其它方面的因素我们考虑的较少。改进型协议存在启动速度慢、簇间存在干扰，拓扑扩展不够灵活等问题还没有进行深入的研究。考虑到应用环境的不同，检测对象的不同，可能传感器节点具有不同的发射功率，可能传感器节点并不是固定不动的，所以，在将来的研究中，我们还应考虑到在传感器节点移动的情况下也能构建良好的通信协议。仿真中只是对节点死亡率和网络生存周期进行了仿真，而没有对网络的其它参数如时延，网络的健壮性等方面进行仿真。由于时间的关系和硬件平台支持的不足，只是提出了基于簇融合的改进型协议的程序进行了简单的功能测试，而没有进行大规模的测试，

也更未对实际应用中协议所表现出来的性能指标进行测试。

# 致谢

两年多的研究生生活即将结束，在毕业论文完成之际，回想起这两年多里老师和同学们给与我的无私帮助和细心指导，我感慨万千，我要向他们致以最诚挚的感谢。

我要特别感谢我的导师刘彦明教授，在他的精心指导下，我的动手能力有了很大提高，特别是在编写代码方面把我从一个门外汉变成一个有一定编程能力的程序员。特别是在论文撰写阶段，刘老师倾注了大量的心血和汗水，无论是在选题、构思和材料收集方面，还是在论文的研究方法方面，都得到了老师的悉心的指导，他为我指点迷津，帮我开拓研究的思路，精心点拨，他循循善诱的教导给予我无尽的启迪。他渊博的知识、深厚的学术素养、严谨的治学精神、一丝不苟的的工作作风，深深地感染和激励着我，是我今后的工作和生活中学习的榜样。谨此向这两年多在学业上给予我精心的指导、在生活和思想上给以无微不至关怀的刘彦明老师表示崇高的敬意和衷心的感谢！

感谢李小平老师，李老师在生活和学习上给了我很多关心和教诲，使我少走了很多弯路。感谢孙小平老师和王海老师，两位老师在学习和科研上的指导帮助使我更快地融入到项目的团队之中。感谢和我一起在团队中奋斗过的所有同学。让我感受到团队的力量，培养了我团队合作的精神。

最后感谢我的父母和亲人，他们多年来一直无怨无悔地支持我的学业。我能够在大学校园里无忧无虑的学习和生活并取得今天的成果完全是他们一如既往的支持的结果。在这里，我衷心地感谢他们！

# 参考文献

[1]. 孙利民，李建中，陈渝，朱红松.无线传感器网络. 清华大学出版社，2005年5月。

[2]. Hass ZJ,Halpern JY,Li L.Gossip-Bases ad hoc Routing[C].In:Proc of the IEEE INFOCOM.New York:IEEE Communication Society,2002.

[3]. Intanagonwiwat C,Govindan R,Estrin D,Heidemann J.Directed diffusion for wireless sensor networking[C].IEEE/ACM Trans . On Networking,2003,11(1): 2-16.

[4]. D.Braginsky,D.Estrin.Rumor Routing Algorithm for Sensor Networks.Proceedings of 1st Workshop on Sensor Networks and Application.2002.22-31.

[5]. J.Kulik,W.R Heinzelman, H Balakrishnan. Negotiation-based Protocols for Disseminating Information in Wireless Sensor Network[J]. Wireless Networks,2002,8(223):169-185.

[6]. R C Shah,J Rabaey.Energy Aware Routing for Low Energy Ad hoc Sensor Networks[C].Orlando:IEEE Wireless Communications and Networking Conference ,2002.

[7]. C.Schurgers,M,B Srivastava.Energy Efficient Routing in Wireless Sensor Networks[C].Mclean:Proceedings of Communications for Networks2Centric Operations:Creating the Information Force,2001.357-361.

[8]. Ganesan D, Govindan R,Shenker S,et al.Highly2resilient,Energy efficient Multi-path Routing in Wireless Sensor Networks[C].Mobile computing and Communications Review,2002.251-254.

[9]. LiL,Halpern Y j. Minimum energy mobile wireless networks revisited[C].Proceedings of IEEE International Conference on Communication. Piscataway,NJ,USA:IEEE,2001.

[10]. Newsome J.SongD.GEM:Graph Embedding for Routing and Data centric Storage in Sensor Networks without Geographic Information [C].Proc.of the 1st ACM Conf.onEmbedded Networked Sensor Systems,2003.76-88.

[11]. W.Heinzelman,A.Chandrakasan, H.Balakrishnan. Energy Efficient Communication Procotol for Wireless Micro-sensor Networks[C].Proceedings of the 33rd Hawaii International Conference on System .Sciences,2000.3005-3014.

[12]. Lindsey S,Raghavendra CS.PEGASIS : Power-efficient gathering in sensor

information systems[C].In : Proc.of the IEEE Aerospace Conf.Montana:IEEE Aerospace and Electronic Systems Society,2002.1125-1130.

[13]. A Manjeshwar, D P Agarwal.TEEN:A Routing Protocol for Enhanced Efficiency in Wireless Sensor Networks[C].The 1st International Workshop on Parallel and Distributed Computing Issues in Wireless Networks an Mobile Computing ,2001.2009-2015.

[14]. Y Xu,J Heidermann, D Estrin.Geograpy Informed Energy Conservation for Ad hoc Routing[C].Proceedings of the 7th Annual ACM/IEEE Inernational Conference on Mobile Computing and Networking,2001.70-84..

[15]. 李小亚.无线传感器网络节能路由策略研究.华南理工大学。

[16]. 余勇昌.无线传感器网络中能量有效的路由算法研究.华南理工大学。

[17]. 杨云升.基于 LEACH 的无线传感器网络路由协议的研究.吉林大学。

[18]. 杜茜.基于均与分簇的无线传感器网络路由协议研究.华中师范大学。

[19]. 陈本理.基于簇的无线传感器网络层次路由协议研究.西南交通大学。

[20]. 温阳.基于能量的无线传感器网络路由协议的研究.河北工业大学。

[21]. 邓鳌.无线传感器网络路由协议研究与仿真.武汉理工大学。

[22]. 林喜源.基于 TinyOS 无线传感器网络协议研究.武汉理工大学。

[23]. 张倩.基于 TinyOS 的无线传感器网络路由协议设计与实现.西北工业大学。

[24]. 雷远，熊建设，赵晓慧，贾吉庆.基于 TinyOS 的嵌入式无线传感器网络设计.MODERN ELECTRONICS TECHNIQUE,2009 年第 14 期。

[25]. 任丰源，黄海宁，林闯。无线传感器网络。软件学报.2003，14（7）。

# 在读期间研究成果

在硕士研究生期间取得的研究成果如下：

参加科研情况

1. 项目名称：基于二代身份证读卡器的门禁系统。负责的主要工作为，在充分理解软件系统的功能和设计后，编写并测试上位机软件，对整个系统又进行了重构和改进。
2. 项目名称：对 TinyOS 的协议栈进行分析。负责的主要工作为，对 TinyOS 操作系统中的调度策略的实现模块进行分析，分析 TinyOS 中原本可用的 CTP 协议自身需要使用的各个组件的调用关系，并对每个组件的实现代码进行了阅读和研究。