

合肥工业大学

硕士学位论文

无线传感网络节点操作系统调度策略研究与改进

姓名：翟月

申请学位级别：硕士

专业：计算机技术

指导教师：周健;谢文全

20091001

无线传感网络节点操作系统调度策略研究与改进

摘 要

无线传感网络因其巨大的应用前景得到了学术界和工业界越来越多的关注。而无线传感网络操作系统必须能够满足无线传感网络的特殊性所带来的技术要求。

论文在分析了大量相关文献的基础上,对现有的部分无线传感网络操作系统的调度策略进行了分析。分析表明,事件驱动的 TinyOS 操作系统能够很好的满足能量有效性的原则,但随着无线传感网络应用环境的不断扩展, TinyOS 的实时响应方面尚显欠缺。在此基础之上,本文提出了基于优先级的可抢占式分级调度策略,来改进 TinyOS 对于实时性的支持。

基于优先级的可抢占式分级调度策略结合了事件驱动、分级调度和多线程的思想,将操作系统可处理的任务分为不同的优先级别,高优先级任务先得到响应。考虑到实时性要求严格的任务,抢占机制被引入到该调度系统中,即任务级别依次分为高优先级可抢占、高优先级不可抢占和一般优先级。高优先级可抢占任务能够抢占当前正在执行的任务而先执行;高优先级不可抢占任务具有先得到响应的权利,但不对当前正在执行的任务进行抢占;一般优先级任务即对应于一般 TinyOS 任务。系统调度系统包含主调度器和若干子调度器。每个子调度器负责按照一定的规则对同一个优先级别的任务进行调度,而主调度器则对每个子调度器的当前任务按照其优先次序进行调度。文章最后对该调度策略的性能进行了分析和论证。

关键词: 调度; 优先级; 抢占; 分级; TinyOS

Research And Improvement On The Scheduling Strategy Of Operating System For The Nodes Of Wireless Sensor Network

ABSTRACT

There has been more and more attention paid to wireless sensor network in both industry and academia because of its great application prospect. However, wireless sensor network operating system must be able to meet technical requirements brought about by specificities of the wireless sensor network.

Based on the analyses of a great number of related literatures, this paper was carefully analyzed the existing part of wireless sensor network operating system scheduling policies. The analysis results showed that: the event-driven Tiny OS operating system could meet the energy-efficiency principle. Thus, with wireless sensor network application environments constant enlargement, real-time response of Tiny OS was a little weak. On this basis, the classified preemptive scheduling strategy based on priority was proposed in this paper, which could improve Tiny OS, to support real-time response.

The scheduling strategy was a combination of the idea of event-driven, classification and multi-thread. Tasks that could be handled by the operating system were divided into different priority levels, and tasks with high-priority obtain to be response first. Considering tasks with strict real-time requirement, preemption was introduced into the scheduling system. Then, tasks were divided into three levels: high-priority tasks with preemption, high-priority tasks without preemption and common tasks. High-priority task with preemption can preempt the processing of other tasks to process first; high-priority tasks without preemption can be processed earlier than common tasks but do nothing to the task which was processing currently; Common tasks were original Tiny OS tasks. The scheduler of the system was consisted of main scheduler and some sub schedulers. Every sub scheduler deal with tasks in the same level following certain rules, while main scheduler deal with the current task of all the sub schedulers according the priority difference. At the end of the paper, analysis and demonstration of the performance of the scheduling policy were given in some aspects.

Keywords: Scheduling; Priority; Preemption; Classification; TinyOS

图表清单

图 2-1 无线传感网络体系结构.....4

表 2-1 TinyOS 支持的部分硬件平台特性..... 8

图 2-2 WMN OS 操作系统模型.....11

图 3-1 MANTIS OS 系统框架.....16

图 3-2 Mantis OS 调度体系.....17

图 3-3 SOS 调度体系....., 18

图 3-4 μ C/OS-II 中任务状态....., 19

图 3-5 TinyOS 系统框架.....20

图 3-6 TinyOS 的功能模块.....21

图 3-7 TinyOS 调度系统.....22

图 3-8 任务队列为空.....22

图 3-9 任务队列中任务数为三.....22

图 3-10 任务最大响应时间的构成....., 27

图 4-1 开放系统模型.....32

图 4-2 改进的操作系统任务处理.....35

图 4-3 分级调度系统结构.....37

图 4-4 任务入队流程.....39

图 4-5 系统调度结构.....40

图 4-6 实时任务断响应比较.....41

图 4-7 可调度性比较.....41

图 4-8 节点能量消耗比较.....42

独 创 性 声 明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。据我所知，除了文中特别加以标志和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得合肥工业大学或其他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作了明确的说明并表示谢意。

学位论文作者签字：



签字日期：



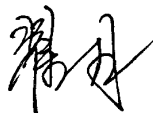
年 12 月 10 日

学位论文版权使用授权书

本学位论文作者完全了解合肥工业大学有关保留、使用学位论文的规定，有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许论文被查阅或借阅。本人授权合肥工业大学可以将学位论文的全部或部分论文内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

（保密的学位论文在解密后适用本授权书）

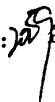
学位论文者签名：



导师签名：



签字日期：



年 12 月 10 日

签字日期：2009 年 12 月 10 日

学位论文作者毕业后去向：

工作单位：安徽电子信息职业技术学院

电话：13966042585

通讯地址：

安徽电子信息职业技术学院

邮编：233000

致 谢

本论文是在周健老师的悉心指导下完成的，没有周老师的指导、鼓励，论文是不可能得以顺利完成的。老师优秀的做人品质，严谨的治学态度，开拓创新的精神，高屋建瓴把握全局的能力，忘我的工作精神给学生树立起潜移默化的典范作用，这也是导师传授给学生最宝贵的财富。特别是周老师在我求学期间给予的各方面的充分理解和支持。在此，谨向我的导师致以深深的敬意！

感谢我的家人一直以来对我的无私的支持和信任。他们在生活上给予了我无微不至的关怀，是我最强大的后盾，使我能够尽我所能的投入学习研究而不用考虑其他。他们在学习上给予了我无限的支持和鼓励，使我克服了一路上遇到的困难，最终走到了现在。在此，我向他们致以最诚挚的谢意！

感谢我的同事和朋友们曾经给过的指导和帮助，他们在我的学习和生活中给予了我关心和帮助，共同扶持、共同进步。

最后，向所有曾经给过我帮助的老师、同事、朋友们致以我深深的谢意！谢谢大家！

作 者：翟 月

第一章 绪论

我们处在一个信息的时代，科技发展的脚步越来越快。传感器技术作为信息获取技术中很重要的一项，在近些年得到了极大的发展。

1.1 研究背景

随着无线通信技术、嵌入式计算技术和传感器技术的飞速发展和日益成熟，人们研制出了各种具有感知能力、计算能力和通信能力的微型传感器，许多微型传感器构成的无线传感网络^[1,2]（Wireless Sensor Network, WSN）引起了人们的极大关注。无线传感网络综合了传感器技术、嵌入式计算技术、分布式信息处理技术和通信技术，能够协作实时监测、感知、采集网络分布区域内的各种环境或监测对象的信息，并对这些信息进行处理，获得详尽准确的信息，传送到需要这些信息的用户。与传统的有线传感器网络相比，无线传感网络具有自组成网、配置灵活、扩展简便等特点，在危险的、人工无法进入的环境中，可以利用飞机投掷等手段将传感器节点布置到监测区域。这些节点通过自行组织成网络、进行数据采集和处理、传回相关的信息来完成监测任务。上述优势使得自组织无线传感网络可被广泛应用于森林防火、生物栖息地监控、仓储管理、环境监测、医疗监护等领域，其在军事领域的应用更具有极其重要的意义。无线传感网络是信息感知和采集的一场革命，在新一代网络中具有关键作用。美国《商业周刊》认为无线传感网络是全球未来四大高技术产业之一，是 21 世纪世界最具有影响力的 21 项技术之一。MIT 新技术评论认为，无线传感网络是改变世界的十大新技术之一。

无线传感网络作为一种新的计算模式正在推动科技发展和社会进步，关系到国家经济和社会安全，已成为国际竞争的制高点，引起了世界各国军事部门、工业界和学术界的极大关注。美国自然科学基金委员会 2006 年制定了无线传感网络研究计划，投资 6400 万美元支持相关基础理论的研究。美国国防部和各军事部门都对无线传感网络给予高度重视，把无线传感网络作为一个重要研究领域，设立了一系列的军事无线传感网络研究项目。英特尔公司、微软公司等信息工业界巨头也纷纷设立或启动相应的行动计划。世界很多国家都纷纷展开了该领域的研究工作。

我国最近几年也开始重视无线传感网络技术的研究。国家自然科学基金委员会资助了很多无线传感网络研究项目，包括重点项目和面上项目。在“中国未来 20 年技术预见研究”报告中，有 7 项技术课题直接论述了传感网络。2006 年初发布的《国家中长期科学与技术发展规划纲要》为信息技术确定了 3 个前沿方向，其中，有两个与无线传感网络研究直接相关。

1.2 研究挑战

由于无线传感网络电源能量有限、通信能力有限、节点计算能力有限、传感器节点数量大且分布范围广、网络动态性强、感知数据流巨大、以数据为中心等特点，无线传感网络在基础理论和工程技术两个层面都向研究者提出了大量挑战性问题。例如：

(1) 无线传感网络结构问题。无线传感网络一般节点众多，随机分布，采用自组织的方式形成网络连接，合理的网络组织形式和拓扑结构，对于网络的生存周期和工作效率都会产生很大的影响。

(2) 网络通信问题。无线传感网络中节点的发射信号强度受到其电源的限制，并且无线传感网络的一般的应用环境较为复杂，因此节点的信号可能被一些障碍物干扰，怎么保证信号的有效、准确地传递是有待进一步改善的问题。

(3) 系统能量供应问题。由于无线传感网络节点一般采用电池供电，而且不易更换电池，所以如何使得节点能够在现有电源的供应下工作尽可能长的时间是当前无线传感网络研究中的一个非常重要的部分。

(4) 安全问题。随着无线传感网络技术应用到各个领域，在某些特殊领域下的应用中，如何保证数据传输过程中的可靠性和安全性也是必须要考虑的问题。

(5) 成本问题。成本问题是任何一种技术应用到实际中必须要考虑到的问题。无线传感网络应用中，传感器节点数量大且分布广，每个节点的成本的改变都会对整个项目的成本产生相当可观的影响。

随着无线传感网络的不断发展，它在理论研究和工程实现上都已经取得了很多重要的成果。但是，目前迅速增长的信息获取和日益复杂的功能需求仍然要求对一些技术难点进行解决或者进一步的发展。

1.3 研究内容

本文针对无线传感网络技术中操作系统的任务调度机制进行了研究，分析了现有的多种操作系统的调度机制及其特点。在此基础之上，结合各个操作系统调度机制的特点，结合实际应用要求，设计了一种基于优先级的可抢占式分级调度机制，并分析了其特性。

具体内容如下：

(1) 研究了现有的几种无线传感网络操作系统的调度机制特点及具体实现，重点研究了当前应用较为广泛的 TinyOS 操作系统的调度机制。

(2) 结合实际应用要求，提出了一种基于优先级的可抢占式分级调度机制，给出了其具体实现算法，利用 TinyOS 的组件思想，将其组合到 TinyOS 操作系统当中。

(3) 对新的调度机制特性进行分析，并进行了实验研究。

1.4 论文结构

本文共分为六个章节，每个章节内容介绍如下：

第一章，介绍了本文的基本研究背景和基本内容。

第二章，介绍了无线传感网络及其操作系统的基本特点和研究现状，结合无线传感网络的特点，分析其操作系统的设计目标。

第三章，介绍了操作系统中的相关基本概念，分析了现有的几种无线传感网络操作系统调度机制特点和实现。其中，重点对 TinyOS 操作系统的调度机制进行了分析和研究。

第四章，在前面分析研究的基础上，针对一些实际应用需求，提出了一种基于优先级的可抢占式分级调度机制，并给出了其具体实现机制。

第五章，总结了工作内容，并提出了下一步工作方向。

第二章 无线传感网络操作系统

2.1 无线传感网络简介

2.1.1 无线传感网络结构及其基本特征

传感器网络结构如图 2-1 所示, 传感器网络系统通常包括传感器节点 (sensor node)、汇聚节点(sink node)和管理节点。大量传感器节点随机部署在监测区域(Sensor field)内部或附近, 能够通过自组织方式构成网络。传感器节点监测的数据沿着其他传感器节点逐跳的进行传输, 在传输过程中监测数据可能被多个节点处理, 经过多跳后路由到汇聚节点, 最后通过互联网或卫星到达管理节点。用户通过管理节点对传感器网络进行配置和管理, 发布监测任务以及收集监测数据。[2,3]

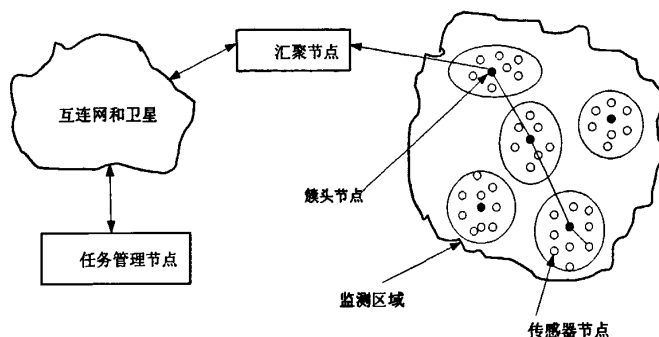


图 2-1 无线传感网络体系结构

传感器节点通常是一个微型的嵌入式系统, 它的处理能力、存储能力和通信能力相对较弱, 通过携带能量有限的电池供电。从网络功能上看, 每个传感器节点兼顾传统网络节点的终端和路由器双重功能, 除了进行本地信息收集和数据处理外, 还要对其他节点转发来的数据进行存储、管理和融合等处理, 同时与其他节点协作完成一些特定任务。目前传感器节点的软硬件技术是传感器网络研究的重点。

汇聚节点的处理能力、存储能力和通信能力相对较强, 它连接传感器网络与 Internet 等外部网络, 实现两种协议栈之间的通信协议转换, 同时发布管理节点的监测任务, 并把收集的数据转发到外部网络上。汇聚节点既可以是一个具有增强功能的传感器节点, 有足够的能量供给和更多的内存与计算资源, 也可以是没有监测功能仅带有无线通信接口的特殊网关设备。

无线传感网络具有动态性、适应性和分布处理能力, 它由以通信为中心的、大量的、小型和微型数据采集设备构成。它是一种特殊的 Ad Hoc 网络, 相对于传统无线网络, 其特点为网络节点密度高、数据传输具有突发性和间断性、

网络节点间可自组织通信、拓扑变化频繁、具备容错能力等；也是一种典型的嵌入式系统，和一般的实时系统相比，有其显著的特点，如传感器节点的存储容量小、运算能力弱、节点尺寸小、功耗低、支持并发密集型操作、有限的物理并行性和控制层次、多样化的设计和使用、操作的健壮性、节点的自主自治能力等。与这些网络相比，无线传感网络具有以下特点：^[1,2,3,5]

（1）硬件资源有限

节点由于受价格、体积和功耗的限制，其计算能力、程序空间和内存空间比普通的计算机功能要弱很多。这一点决定了在节点操作系统设计中，协议层次不能太复杂。

（2）电源容量有限

网络节点由电池供电，电池的容量一般不是很大。其特殊的应用领域决定了在使用过程中，不能给电池充电或更换电池，一旦电池能量用完，这个节点也就失去了作用。因此在传感器网络设计过程中，任何技术和协议的使用都要以节能为前提。

（3）无中心

无线传感网络中没有严格的控制中心，所有结点地位平等，是一个对等式网络。结点可以随时加入或离开网络，任何结点的故障不会影响整个网络的运行，具有很强的抗毁性。

（4）自组织。网络的布设和展开无需依赖于任何预设的网络设施，节点通过分层协议和分布式算法协调各自的行为，节点开机后就可以快速、自动地组成一个独立的网络。

（5）多跳路由

网络中节点通信距离有限，一般在几百米范围内，节点只能与它的邻居直接通信。如果希望与其射频覆盖范围之外的节点进行通信，则需要通过中间节点进行路由固定网络的多跳路由使用网关和路由器来实现，而无线传感网络中的多跳路由是由普通网络节点完成的，没有专门的路由设备。这样每个节点既可以是信息的发起者，也是信息的转发者。

（6）动态拓扑

无线传感网络是一个动态的网络，节点可以随处移动；一个节点可能会因为电池能量耗尽或其他故障，退出网络运行；一个节点也可能由于工作的需要而被添加到网络中。这些都会使网络的拓扑结构随时发生变化，因此网络应该具有动态拓扑组织功能。

（7）节点数量众多，分布密集

为了对一个区域执行监测任务，往往有成千上万传感器节点空投到该区域。传感器节点分布非常密集，利用节点之间高度连接性来保证系统的容错性和抗毁性。

2.1.2 研究现状

无线传感网络的研究最初来源于美国军方。美国国防预先研究计划局（Defense Advanced Research Projects Agency, DARPA）于 2001 年在“网络嵌入式软件技术”（Network Embedded Software Technology, NEST）项目的支持下，资助加州伯克利大学开发了名为“Smart Dust”（智能灰尘）或“Mote”的无线传感开发系统，从那时期起到现在，DARPA 每年都投入几千万美元进行无线传感网络技术的研究。2002 年 8 月初，美国国家科学基金委（NSF），DARPA、CNASA 等 12 个重要研究机构在加州伯克利大学联合召开了“未来传感系统”国家研讨会，旨在探讨未来传感器系统及其在工程应用中的前沿发展方向。会议讨论认为无线传感网络的实现及其传感数据的传输、分析及决策技术的研究同基于微机械、纳米技术的新型传感技术一样，代表着未来传感器研究的前沿方向，并且对于战场信息感知、国土安全与反恐战争、结构健康监测等工程应用领域具有极其重要的研究意义。在随后的几年里，加州大学伯克利分校有多个实验室开始了关于无线传感网络及其相关的工作，如：NEST、WEBS（Wireless Embedded System）、BARWAN（Bay Area Research Wireless Access Network），BWRC（Berkeley Wireless Research Center）等实验室，从不同的角度对无线传感网络进行了大量具有开创性的研究。^[1,4-13]

美国很多大学和研究机构在无线传感网络方面开展了大量工作。如加州大学洛杉矶分校（University of California, Los Angeles, UCLA）的 CENS（Center for Embedded Networked Sensing）实验室、WINS（Wireless Integrated Network Sensors）实验室、NESL（Networked and Embedded Systems Laboratory）实验室、LECS（Laboratory for Embedded Collaborative Systems）实验室、IRL（Internet Research Lab）等。另外，麻省理工学院（Massachusetts Institute of Technology, MIT）获得了 ARPA 的支持，从事着极低功耗的无线传感网络方面的研究，被业界广泛关注的 SPIN（Sensor Protocols for Information via Negotiation）协议也是出自 MIT；奥本大学（Auburn University）也获得 DARPA 支持，从事了大量关于自组织传感器网络方面的研究，并完成了一些实验系统的研制；宾汉顿大学（Binghamton University）计算机系统研究实验室在移动自组织网络协议、传感器网络系统的应用层设计等方面做了很多研究工作；州立克利夫兰大学（俄亥俄州，Cleveland State University, Ohio-CSU Ohio）的移动计算实验室在基于 IP 的移动网络和自组织网络方面结合无线传感网络技术进行了研究。另外，北亚利桑那大学（Northern Arizona University）的无线网络研究实验室（Wireless Network Research Lab, WNRL）、莱斯大学（Rice University）多媒体通信实验室的无线个人局域网工作组；斯坦福（Stanford）大学的无线传感网络实验室、新泽西（New Jersey）州立大学的无线传感网络实验室、伊利诺伊大学厄本那-香槟分校（UIUC）的 TIMELY 实验室、南加州大学的 RESL（The Robotic

Embedded Systems Laboratory) 实验室、佛蒙特大学 (University of Vermont) 的无线自组织网络实验室、西密西根大学 (Western Michigan University) 的无线传感网络实验室。此外新加坡国立大学 (National University of Singapore, NUS) 的无线传感网络实验室等也有关于无线传感网络方面的研究。

国内在无线传感网络领域的研究也已经在很多研究所和高校展开。中科院上海微系统所凭借其在微系统和微型机电系统 (Micro Electromechanical System, MEMS) 技术方面良好的基础, 自从 1998 年就对无线传感网络进行了跟踪和研究, 已经通过系统集成的方式完成了一些终端节点和基站的研发。他们的很多工作都是与 CDMA (Code-Division Multiple Acces) 和 GPS (Global Positioning System) 技术相关, 从某种程度上说已经超越了无线传感网络技术; 中科院电子所和沈阳自动化所也分别从传感器技术和控制技术角度入手开展工作, 他们专注于传感或控制执行部分, 对上层的通信技术和核心微处理器部分涉及较少; 浙江大学现代控制工程研究所成立了“无线传感网络控制实验室”, 联合相关单位专门从事面向传感器网络的分布自治系统关键技术及协调控制理论方面的研究; 山东省科学院于 2004 年 10 月正式启动了关于无线传感网络节点操作系统的研究; 另外国防科技大学、清华大学、中国科学技术大学、哈尔滨工业大学、东南大学等单位在无线传感网络方面也都有一定的工作; 其中针对面向结构健康监测的无线传感网络的研究, 南京航空航天大学、哈尔滨工业大学都开展了一些初步的探索。可以预计, 分布式无线传感网络的发展和广泛应用, 将对人们的社会生活和产业变革带来极大的影响并产生巨大的推动。

有关无线传感网络的具有代表性的研究项目有如下一些^[4-13]。

- uAMPS (u-Adaptive Multi-domain Power Aware Sensors): MIT 开展, 该项目的目标为实现自适应的能量感知的分布式微传感器开发一种框架, 研究重点在信号和能量调节, 通信以及协作等。

- ISIS (Information System for Industrial Control and Supervision Sensor Fusion): 关于传感器网络在传感器数据融合方面的一个研究项目, 研究重点在目标跟踪、导航和定位等应用。

- LEACH (Low Energy Adaptive Clustering Hierarchy): 由 MIT 人工智能实验室发起, 关于无线传感网络的路由协议的研究, 使得终端用户可以远程地监控环境;

- SensorNet: 由 BWN (Borland & Wireless Network lab) 实验室开展, 针对无线传感网络协议的研究;

- Smart Dust: 加州大学伯克利分校开展, 目标是展示一个完整的传感/通信系统可以集成到一个立方体状的毫米级的包中;

- Sensor webs: 由 NASA 发起, 研究一种新型的监控和探测环境的手段;

• WINS（Wireless Integrated Network Sensors）：由 UCLA 电子工程系展开，研究分布式的无线传感网络，以及和 Internet 连接，控制和处理器嵌入在装置、设备 and 环境中。

2.2 无线传感网络操作系统研究现状

作为无线传感网络的支撑技术之一，操作系统必须能够满足无线传感网络的特殊性所带来的技术要求。为了达到高效率 and 灵活性，它必须比传统操作系统微小但高效，具有高安全性并支持系统特有的动态配置 and 自适应特性，并且能够并行处理多任务，使硬件的性能得到充分发挥。

随着无线传感网络的发展，目前已经出现了好几种应用无线传感网络的操作系统，选择几种主要的介绍如下。^[14-19]

• TinyOS

TinyOS 是加州大学伯克利分校开发的开源微型操作系统，专为无线传感网络设计，目前在无线传感网络操作系统领域占据了主导地位。TinyOS 基于组件的架构使其能够快速实现各种应用。TinyOS 的组件库包括网络协议、分布式服务、传感器驱动以及数据获取工具等，一个完整的应用系统是由这些库组合起来的，不用的组件不会引入进来，从而达到减少内存需求的目的。TinyOS 采用了事件驱动模型，这样可以在很小的空间中处理高并发事件，并且能够达到节能的目的，因为 CPU 不需要主动去寻找感兴趣的事件。

目前 TinyOS 已经可以运行在很多硬件平台上，在 TinyOS 网站上公开原理图的硬件平台有 Telos(Rev A)，Telos(Rev B)，Mica2Dot，Mica2，Mica，此外，还有一些商业和非商业组织也有一些硬件平台可运行 TinyOS，主要有欧洲的 Eyes，MotelV 提供的 TmoteSky，Crossbow 公司的 MicaZ 以及 Intel 公司的 Mote。

表 2-1 TinyOS 支持的部分硬件平台特性

硬件平台	MCU	RAM	FLASH	RF 芯片
Telos(Rev A)	MSP430 F149	2K	60K	CC2420
Telos(Rev B)	MSP430 F1611	10K	48K	CC2420
Mica2Dot	ATMEGA 128L	4K	128K	CC1000
Mica2	ATMEGA 128L	4K	128K	CC1000
Mica	ATMEGA 103	4K	128K	TR1000
MicaZ	ATMEGA 128L	4K	128K	CC2420
Eyes	MSP430 F149	2K	60K	TR1001
Tmote	MSP430 1611	10K	48K	CC2420

TinyOS 目前也在实际项目中得到了广泛的应用，在其官方网站上列出了数十个采用了 TinyOS 的项目，并且正在不断更新中。

• MANTIS OS（MOS）

MANTIS OS 是由美国科罗拉多大学 MANTIS 项目组为无线传感网络而开发的源代码公开的多线程操作系统。它的内核和 API 采用标准 C 语言，提供 Linux 和 Windows 开发环境，易于用户使用。MANTIS OS 提供抢占式任务调度器，采用节点循环休眠策略来提高能量利用率，目前支持的硬件平台有 Mica2, MicaZ 以及 Telos 等，其对 RAM 的最小需求可到 500B，对 flash 的需求可小于 14KB。

MANTIS 在 2006 年 5 月 30 日和 31 日召开的嵌入式网络传感器（EmNets 2006）会议上，MANTIS 项目组提出了一种增强 TinyOS 性能的系统构架 TinyMOS。在这个架构上，TinyOS 作为一个线程运行在 MANTIS OS 上。通过 MANTIS OS，TinyMOS 在 TinyOS 中增加了优先级和多线程功能，并且从 TinyOS 的主线程中引入了从线程概念，应用于主线程的大量计算任务。MANTIS 项目组在 2006 年夏天发布 TinyMOS 架构。

MANTIS OS 也有一些成功的实际应用，2006 年 6 月 19 日到 22 日召开的国际移动系统（MobiSys 2006）会议上，一个应用了 MANTIS OS 的称之为火灾探测网络（Fire WxNet）的项目获得了大会的最佳论文奖。这个网络是在 2005 年 8 月和 9 月，MANTIS 项目组与蒙大拿大学合作，在爱达荷州的比特鲁特国家森林公园部署的。它由 3 个采用 MANTIS OS 无线传感网络组成，并且由一个 802.11 主干网支撑起来，被部署在森林火灾高发地带用以检测天气状况。在一系列严酷的测试条件下，MANTIS OS 的各个部分，包括内核、网络、任务循环以及应用支持能力等各方面都运行得很好。

• SOS

SOS 是由加州大学洛杉矶分校网络和嵌入式实验室（NESL）为无线传感网络节点开发的操作系统。SOS 使用了一个通用内核，可以实现消息传递、动态内存管理、模块装载和卸载以及其他的一些服务功能。SOS 的动态装载软件模块功能使得它可以创建一个支持动态添加、修改和删除网络服务功能的系统。

SOS 的开发者主要有二大目标。一是要实现动态可重配置，在无线传感领域，重配置功能可以使得网络在部署和初始化后，还能在各个节点上对软件进行修改，这就使得网络在被部署以后，还可以对网络进行更新，在节点上添加新的软件模块以及去除不再需要的软件模块。随着网络越来越庞大，越来越难以更新，可动态配置显得非常重要。二是创建一个能为开发人员提供各种通用服务的快速开发系统。许多无线传感应用往往需要一些非常通用的服务，比如从内存数据包管理到应用服务协议（如邻居发现协议等）。最后一个目标是吸收传统操作系统设计思想并且将他们应用到资源非常有限的传感器节点中来，便于对系统进行维护。

SOS 采用标准 C 写成，可以使用标准 C 调试工具如 GDB 等。其内核支持各种通用服务，如动态内存分配，简单内存碎片回收以及优先级调度等等。SOS

采用真正的模块化系统开发，应用模块在网络被部署后仍然保持模块化特性。日前 SOS 支持的硬件平台主要有 Crossbos 公司的 Mica 系列平台和耶鲁大学 XYZ 节点。

• Contiki

Contiki 是瑞典计算机科学研究所 Adam Dunkels 等人专为内存资源非常有限的嵌入式系统如网络传感器节点等开发的一个多任务操作系统。Contiki 完全用 C 语言写成，源代码开放，支持网络互联，具有高度的移植性，代码量非常小，支持从 8 位微控制器构成的嵌入式系统到老式的 8 位家用电脑。自从 2003 年 5 月推出以来，Contiki 已经被移植到了 20 种不同类型的硬件平台。

Contiki 提供一个简单的事件驱动内核，支持原型进程以及可选的抢占式多任务，通过传递消息来实现任务间通信，具有动态进程结构，支持加载和卸载程序。使用以 IP 协议栈实现本地 TCP/IP 协议，可以在直接相连的终端和通过网络相连的终端（如虚拟网络计算机和 Telnet）上实现图形化界面系统。

当前 Contiki 的一个从本系统（支持多任务，网络和图形界面）编译后代码大小为 32 K，而一个较为完整地支持 Web 服务器，Wb 浏览器等功能的系统编译后代码大小约为 64 K。目前能够运行 Contiki 的最小系统只有 2 K RAM，它能够运行从本系统、Web 服务器、虚拟网络计算机服务器和一个小的虚拟桌面。

• Magnet OS

Magnet OS 是由康奈尔大学为自组织和无线传感网络开发的分布式操作系统，其目标是为自组织网络应用提供一个节能，适应性强并且效率高的操作系统。Magnet OS 为一个由各种各样节点构成的自组织网络，提供一个单一的 Java 虚拟机系统映像。系统能够自动将应用程序分割成各种组件，并且以利于节能、延长网络寿命的方式将这此组件自动放置和迁移到最合适的节点。

MagneOS 最大的特点是采用了虚拟机的思想，目前可运行在 x86 笔记本和 StrongARM 的 PDA 上，如 iPAQ，Axins 和 Jornadas 等。

• TRON 与 T-Engine

与前面几种专门针对无线传感应用而重新开发的操作系统相比，TRON（The Real-time Operating System Nucleus）最明显的特点是它是一个通用的嵌入式操作系统，却在无线传感领域得到了广泛的应用。

TRON 是由日本东京大学坂村建教授于 1984 年提出的计算机操作系统规范，是目前在全世界应用最广泛的嵌入式操作系统。TRON 广泛使用在移动电话、数码相机、传真机、汽车引擎控制等领域，成为实现普适计算环境的重要的嵌入操作系统，并且，在使用了计算机的电器、家具、住宅、大楼、城市、博物馆等设计上，也被广泛地使用。TRON 已经安装到全球 30-40 亿台家用电

子产品中，占据全球微处理机嵌入式操作系统市场约 60%（Windows 大约只安装了 1.5 亿套），成为低价高性能嵌入式开源实时系统的典范。

“TRON 项目计划”为了向世界推广，一直采用自由开源、“弱标准化”的方针，也曾经出现过多种版本的开发环境及操作系统。为了实现更为理想的实时操作系统的嵌入式计算结构，TRON 项目计划开始了一次新的革命—启动了 T-Engine 项目计划。

“T-Engine”（T 引擎）是为在短时间内高效开发实时嵌入式系统而设计的，由标准化硬件结构（T-Engine）与标准开源实时操作系统核心（T-Kernel）组成的嵌入式系统的开放式标准平台。目前 T-Engine 硬件结构有四种规范标准：T-Engine（标准 T 引擎），uT-Engine（微型 T 引擎），nT-Engine（微毫 T 引擎），pT-Engine（微微 T 引擎）。其中 pT-Engine 面向普适计算环境中最小的硬件单元如开关、照明器件、传感器、锁以及阀门等器件。T-Engine 的软件环境主要包括 T-monitor、T-Kernel 的各种扩展、标准设备驱动以及中间件等外围软件几个部分。在 T-Engine 的基础结构规范中，不对开发环境进行标准化。但为了确保软件的兼容性，有必要规定源代码及二进制代码的标准规范形式。因此，关于源代码及对象代码的形式，规定以 gcc 中的内容为准。

为将 T-Engine 这种体系结构向世界推广，2002 年成立了 T-Engine 论坛，已经在全世界拥有 479 家会员公司（截至 2006 年 8 月 10 日）。

• 上海市计算技术研究所的 WMNOS

上海市计算技术研究所独立开发了无线微网节点专用操作系统 WMN OS，可以稳定运行在自行研制的 Z205、Z305 等硬件模块上，目前已经在多个项目上得到了应用。

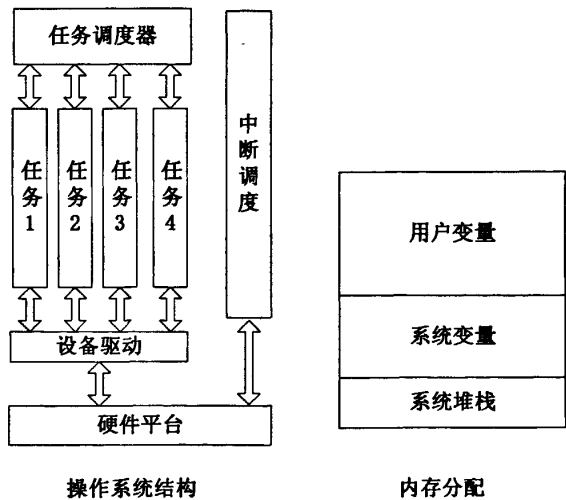


图 2-2 WMN OS 操作系统模型

图 2-2 是 WMN OS 操作系统模型。由图可知，WMN OS 主要由任务调度模块和中断调用模块组成，任务调度模块主要负责一般任务的调度，如传感器管理、电源管理以及无线通讯管理，中断调用主要负责突发事件的处理，节点内存则划分为系统堆栈、系统变量和用户变量三个区域，任务调度器根据系统变量来对各个任务进行调度，通过这种比较简单的方法实现了多任务无线传感操作系统。WMN OS 所需的节点资源也非常之小，最小可在 1K RAM 上运行。目前上海市计算技术研究所在此操作系统上实现的网络协议有：一级星型网络（SL SN），二级树型网络（DLTN），一级并行无线微网数据链（SP-WM data-link），多级并行无线微网数据链（MPVM data-link），多汇聚节点的自组织基站网（MSOBN）。

2.3 无线传感操作系统特征

无线传感网络是一种分布式的自组织网络计算机系统，它由大量廉价的、能量有限的多功能微型传感器节点组成。由于无线传感网络的特殊性，导致其对操作系统的需求相对于传统的操作系统有较大差异。因此，需要针对无线传感网络的特点，研究和设计面向无线传感网络的操作系统（WSN OS）和相关软件。

传统的实时嵌入式操作系统，如 uClinux，VxWorks，WinCE，Palm OS 和 QNX 等，一部分来源于桌面操作系统，因而一般提供了类似于传统桌面系统的运行环境，主要使用于 PDA、智能手机等；另一部分则为专用的应用系统设计，主要适用于机顶盒等嵌入式 PC 系统。它们都不能满足无线传感系统在存储空间、能耗和实时性等方面的需求。而一些微型实时内核，如 Creem，pOSEK 和 Ariel 等，主要用于发动机控制器或微波炉等深度嵌入式系统，有严格定义的运行和存储模式，并且以实现硬件资源的访问控制为主，即以控制为中心，不能满足无线传感系统以数据流为中心的应用特点。

同其它操作系统一样，无线传感网络操作系统是为了提高软件的重用性、降低应用开发的难度，它应该提供物理设备的抽象和高协调性的通用函数实现。其独特性在于资源极端受限（处理器速度、存储器大小、内存大小、通讯带宽、资源数量以及电源受限）、设备的特殊性和缺乏一致的抽象层次。因此，无线传感网络操作系统的设计目标是高效地使用传感器节点的有限资源，且能够对各种特定应用提供最大的支持。其设计策略应该是一个资源库，从中抽取一部分组成应用。它致力于提供有限资源的并发，而不是提供接口或形式。在面向无线传感网络的操作系统支持下，利用有限资源对整体系统进行高效率的事件处理、能源管理、命令处理和工作描述。

无线传感网络对操作系统的特殊要求主要表现在：

- （1）节点的计算资源有限，需要尽可能的减小系统开销；

(2) 节点由电池供电,且要求较长的工作周期,因此需要系统的能耗管理策略与方案,包括操作系统的支持;

(3) 节点的各模块之间需要一定的调度协调机制,同时支持并发控制;

(4) 观测任务需要操作系统支持实时性。

此外,在传感器网络中,单个传感器节点有两个突出特点:一是它的并发性很密集,即可能存在多个需要同时执行的逻辑控制,需要操作系统能够有效地满足这种发生频率、并发程度很高、执行过程比较短的逻辑控制流程;另外,传感器节点的模块化程度很高,要求操作系统能够让应用程序方便地对硬件进行控制,且在保证不影响整体开销的前提下,应用程序的各个部分能够比较方便地进行重新组合。

归纳起来,无线传感网络的特点决定了其操作系统的特点如下: [2, 14, 17]

(1) 由于节点存储资源极其有限,无线传感网络操作系统应该可以根据不同的应用系统进行裁剪和扩充,只实现必要的功能,从而以最小的代码量满足系统需求。

(2) 在无线传感网络中,操作系统和应用程序的区分并不明显。它们在宿主机上一起被编译,然后烧写到目标节点上,目标代码中同时包含了两者。另外,操作系统和应用程序在同一地址空间上运行,它们对硬件的访问权限相同,也就是说应用程序可以直接操纵硬件资源。所以在本质上,所有代码都是应用程序。

(3) 为了有效利用电能,降低功耗,大多数无线传感网络操作系统使用事件驱动模型。当没有任务需要处理时,CPU 睡眠,然后由外部事件来唤醒。

(4) 无线传感网络操作系统通常采用微内核结构,其核心只提供操作系统的基本功能,如进程/线程调度与同步、中断管理、时钟管理、原子操作等。其它功能模块如文件系统、存储管理、通信协议栈都可看成可选的应用。

(5) 无线传感网络操作系统具有可重构能力。当节点工作失常时,要求能自恢复自组织。

(6) 无线传感网络操作系统是一个分布式操作系统,没有中心服务器。

受制于无线网络传感器资源高度受限的特性,目前在操作系统领域的研究重点集中在资源约束上,主要包括:各种简单有效的运行模式,如事件模型、线程模型、状态机模型等;匹配传感器事件驱动的应用特征,如减小操作系统尺寸、降低能耗需求、支持操作的并发性等。

2.4 本章小结

本章介绍了无线传感网络的一些基本概念及发展状况。重点介绍了无线传感网络技术中操作系统的发展状况及其基本特征。根据对无线传感网络操作系

统基本特征的分析，总结出无线传感网络操作系统设计的基本目标和原则。本章是以下章节研究的理论基础和背景。

第三章 无线传感网络操作系统调度策略分析

3.1 基本概念

在分析无线传感网络操作系统的调度机制之前，我们有必要对无线传感网络任务属性中的一些基本的概念进行了解。

- **优先级(Priority)**: 用以表示任务获得处理器资源的优先权级别。传感器操作系统可以根据任务的优先级来安排 CPU 的时间，优先级高的任务优先运行，优先级低的任务则继续等待。优先级通常根据任务的重要性和紧迫性进行设定，可以是静态设定(在系统设计时确定，系统运行中不能改变)，也可以是动态设定(系统运行中可以动态调整)。

- **周期(Period)**: 指的是任务执行的时间间隔。根据触发的时间特征，任务可以被分为周期任务、间发任务和非周期任务三类。传感器操作系统中，周期任务按照固定的时间间隔重复执行；间发任务不具有固定的执行周期，但连续两次被触发之间具有确定的最小时间间隔，在分析系统最坏情况下的响应时间时，这个最小时间间隔通常作为任务的执行周期；非周期任务则没有周期。

- **时限(Deadline)**: 指的是对任务截止时间的要求。根据时限对系统性能的影响程度，任务可以分为软实时任务和硬实时任务。在目前实际运用的传感器系统中，通常允许软硬两种实时性同时存在，其中一些任务没有时限要求，另外一些任务的时限要求是软实时的，而对系统产生关键影响的那些任务的时限要求则是硬实时的。

- **执行时间(Execution Time)**: 指的是任务在独占处理器资源的情况下执行完毕所需要的时间。在传感器操作系统中，通常关心的是任务的最大执行时间(Worst-Case Execution Time, WCET)和最小执行时间(Best-Case Execution Time, BCET)，它们分别是任务可能执行的最坏情况和最优情况。

- **到达时刻(Activation Time)**: 在传感器操作系统中，任务一般由事件触发。触发任务的事件发生的那一时刻称为该任务的到达时刻。对周期任务而言，到达时刻是一个任务周期的开始。

- **释放时刻(Release Time)**: 指一个任务被触发后进入就绪状态，开始等待操作系统调度的时刻。任务的到达时刻与释放时刻之间的时间间隔称为该任务的释放延迟。在传感器操作系统中，造成释放延迟的原因有硬件的中断延迟、操作系统的中断处理延迟和调度开销、任务依赖消息或其他任务的触发等。

- **截止时间(Deadline)**: 传感器系统中的某些任务具有实时要求，它们必须在规定的时间内完成，使系统对外部事件及时做出反应。任务的截止时间表示任务必须完成运行的最迟时间。如果一个任务被要求在某个时刻之前完成，那么这个时刻称为该任务的截止时间。

- 释放抖动(Release Jittery)。对于周期任务来说，同一个任务在不同周期内的释放延迟可能会有所不同，释放延迟的最大偏差称为释放抖动。
- 响应时间(Response Time)。从任务的到达时刻到任务完成时所经历的时间间隔被称为任务的响应时间。响应时间不仅包含任务本身的执行时间，还包括任务等待时间以及操作系统开销等。在分析系统时间行为时，我们通常关心任务的最大响应时间(Worst-Case Response Time， WCRT) 和最小响应时间(Best-Case Response Time ， BCRT)这两种边界情况。

3.2 常见嵌入式操作系统调度策略分析

根据实现机制可以把现有的嵌入式操作系统分为两类，即通用的多任务操作系统(General purpose Multi-tasking OS)和事件驱动的操作系统(Event driven OS)，前者多用于便携式智能设备(如手机、PDA 等)和工业控制中。对于支撑几个独立的应用运行在一个虚拟机上的并行操作是高效的，在处理过程中任务的运行和挂起很好地支撑多任务或者多线程。但是，随着内部任务切换频率的增加将产生非常大的开销，典型代表如 μ C/OS-II 、嵌入式 Linux、WinCE、Mantis。而后者支持数据流的高效并发，并且考虑了系统的低功耗要求，在功耗、运行开销等方面具有优势，因此备受关注,典型的代表如 TinyOS、Contiki。

由于无线传感网络应用的多样性，节点的操作系统必须能够根据内存、处理器以及能量等满足应用的严格需求，也必须能够灵活地允许多种应用同时使用系统资源，如通信、计算和存储。以下介绍了几种典型的无线传感网络操作系统。[14, 18-24]

• MANTIS OS

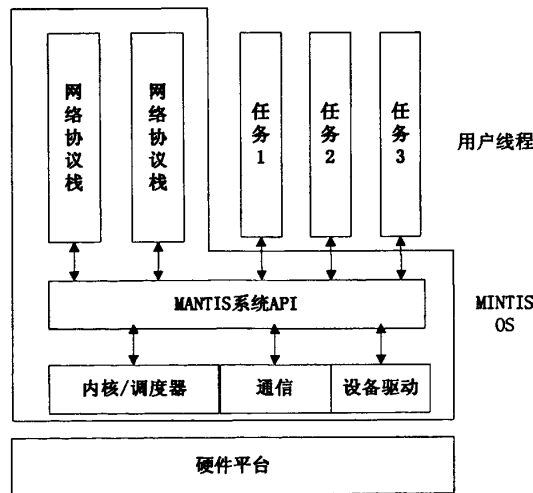


图 3-1 MANTIS OS 系统框架

美国科罗拉多大学开发的 MANTIS OS 是一个建立在加州大学伯克利分校的“Mote”硬件基础上，以易用性和灵活性为主要研究目标的无线网络传感器操作系统。

MANTIS OS 的系统框架如图 3-1 所示，MANTIS OS 以很小的内存需求实现了典型的分层操作系统结构，支持多线程、抢占式调度和同步信号量、设备驱动等操作系统功能。MANTIS OS 的网络协议采用用户线程的形式，考虑了性能和灵活性的折中。目前，一个典型的包含多线程可抢占调度器、标准 I/O 同步支持以及网络协议的 MANTIS OS 内核，只需不到 500Byte(不包括线程栈)的 RAM 和约 14KB 的 ROM 。

和 TinyOS 不同，MANTIS OS 使用目前广泛流行的 C 语言作为操作系统实现的编程语言，从而获得良好的跨平台开发和代码重用支持。

虽然 MANTIS OS 在操作系统的多线程化方面取得了很大的成果，但是，MANTIS OS 并没有很好的体现无线网络传感器系统所特有的基于事件驱动的特性；同时，由于 MANTIS OS 的设计建立在 C 语言的基础上，因而没有很好的实现无线网络传感器系统软件的模块化。

MANTIS OS 采用一个经典的类似 Unix 调度器，实现了一个轻量级的线程库，提供部分 POSIX 线程服务。MANTIS OS 线程由两部分构成：静态线程头以及动态线程堆栈。线程头存储于线程表中，最多允许 15 个线程。MANTIS OS 通过线程头的静态构造，缩短线程的构造时间。MANTIS OS 在调度策略上采用基于优先级的时间片轮转方法，共支持 5 个线程优先级：kernel，sleep，high，normal 和 idle。MANTIS OS 通过 sleep()函数实现节能，sleep()函数接收睡眠时间参数。当所有线程均睡眠时，MANTIS OS 使系统进入节能状态。

Mantis OS 调度系统调度流程如图所示：

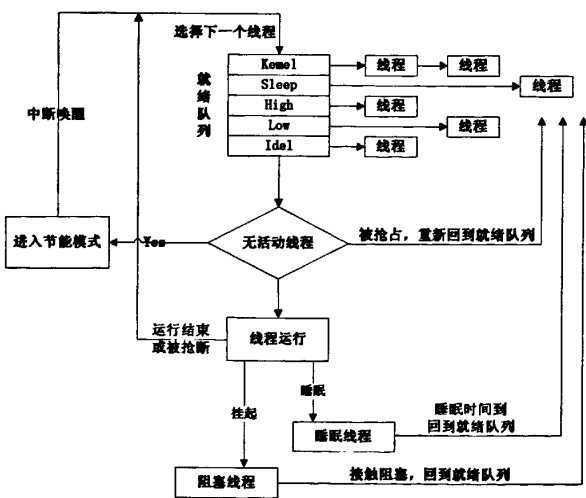


图 3-2 Mantis OS 调度体系

• SOS

SOS 是由加利福尼亚大学(University of California)开发的无线传感网络节点操作系统。它以 C 语言编写，用标准 C 编译器交叉编译。SOS 的主要改进在于对模块动态加载的支持，SOS 目前支持的硬件平台包括 mica2, micaz, XYZ mote, cricket, Tmote sky 以及 intelMote2 0。

SOS 沿用了 TinyOS 基于事件驱动模型，使用前后台协作的调度方式。前台是中断调度，后台是消息调度，前台调度可以抢占后台调度。与 TinyOS 不同的是，SOS 没有事件与任务的划分。SOS 中每一个应用程序模块都包含一个消息处理函数。硬件中断向调度器投递消息，在消息中包含目标模块信息，调度器根据此信息选择正确的消息处理函数，消息处理函数本身也可以再次投递消息。SOS 维护不同优先级的先入先出消息队列（SOS1.3 为两个，SOS 1.7 以后为 3 个），调度器优先选取高优先级队列中的消息，消息彼此之间不可抢占。从本质上看，SOS 实际将 TinyOS 定义为事件的操作写入消息处理，而通过给与对应消息更高的优先级，使这一操作得以尽快进行。SOS 以牺牲这部分操作一定程度的实时性为代价，实现瘦中断。SOS 调度系统如图 3-2 所示：

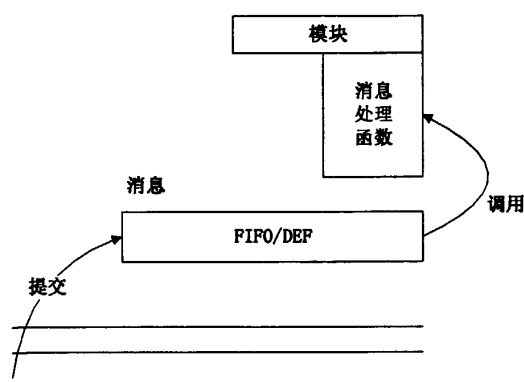


图 3-3 SOS 调度体系

• Contiki

Contiki 是由瑞典计算机科学研究所(Swedish Institute of Computer Science)开发的开源、高度可移植、可多线程的节点操作系统。它的基本特点包括动态加载特性，良好的灵活性及高效性。

Contiki 采用基于进程事件驱动单线程调度体系。进程之间是不可以抢占的，但可以被中断抢占。Contiki 的内核维护了一个进程链表，进程之间有优先级，内核根据优先级来决定其在进程链表中的位置。内核把事件队列里的事件派发给相应的进程（这里的进程不是传统意义上的进程，可理解为任务），同时内核提供了一种轮询机制，主要提供给那些需要不断检查硬件状态的进程使用。为了提供底层的实时性，Contiki 不禁止中断，但是不允许中断处理例程与进程之间出现竞争条件，进程只能通过事后轮询机制进行与中断相关的操作。

Contili 内核不支持多线程，但允许将线程封装到进程中，以用户级函数库的形式实现。

• $\mu\text{C}/\text{OS-II}$

$\mu\text{C}/\text{OS-II}$ 操作系统是一种性能优良、源码公开且被广泛应用的免费嵌入式操作系统。2002 年 7 月， $\mu\text{C}/\text{OS-II}$ 在一个航空项目中得到了美国联邦航空管理局（Federal Aviation Administration）对于商用飞机的、符合 RTCA DO-178B 标准的认证。它是一种结构小巧、具有可剥夺实时内核的实时操作系统，内核提供任务调度与管理、时间管理、任务间同步与通信内存管理和中断服务等功能，具有可移植性、可裁减、可剥夺性、可确定性等特点。

基于多任务的 $\mu\text{C}/\text{OS-II}$ 采用基于优先级的调度算法，CPU 总是让处于就绪态的、优先级最高的任务运行，而且具有可剥夺型内核，使得任务级的响应时间得以优化，保证了良好的实时性。其任务的切换状态如图 3-4 所示。在 $\mu\text{C}/\text{OS-II}$ 中，CPU 要不停地查询就绪表中是否有就绪的高优先级任务，如果有则做任务切换，运行当前就绪的优先级最高的任务；否则运行优先级最低的空闲任务（Idle Task）。

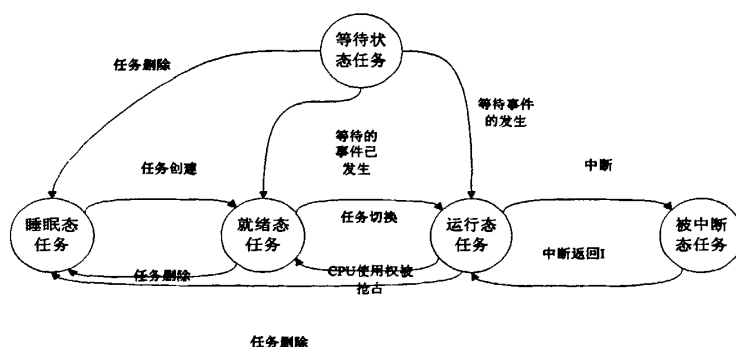


图 3-4 $\mu\text{C}/\text{OS-II}$ 中任务状态

由于多任务的系统需要进行任务切换或者中断服务与任务间的切换。而每次切换就是保存正在运行任务的当前状态，即 CPU 寄存器中的全部内容，这些内容保存在运行任务的堆栈内。入栈工作完成后，就把下一个将要运行的任务的当前状况从任务的堆栈中重新装入 CPU 的寄存器中，并开始下一个任务的运行。因此， $\mu\text{C}/\text{OS-II}$ 需要预先为每个任务分配堆栈空间。在 $\mu\text{C}/\text{OS-II}$ 中的任务堆栈空间可以根据任务的实际需求分配合适的大小。

3.3 TinyOS 操作系统分析

TinyOS 是专为无线传感网络设计的轻量级、低功耗的嵌入式操作系统。它采用基于组件的架构方式，以快速实现各种应用。TinyOS 的编程语言为 nesC，其程序采用模块化设计，这使得它能适应硬件的多样性，允许应用程序重用，

目前，它已经被成功的应用到多种硬件平台上，具有很高的应用价值和研究意义。

3.3.1 TinyOS 框架及特点

美国加州大学伯克利分校的 TinyOS 是一个为网络嵌入式系统定制的操作
系统。为了符合事件驱动应用和小内存的要求，TinyOS 裁制了一个新的编程模
型，一个基于构件的体系结构，一个简单的基于事件的并发模型。

TinyOS 聚焦了无线传感网络体系结构的 3 个高层目标：

- (1)考虑传感器网络和传感器网络节点当前的以及将来可能的设计；
- (2)允许操作系统服务和应用在不同的硬件(在不同系列的节点上)和软件
上；
- (3)满足传感器网络特殊的挑战：有限的资源、严格的并发操作、健壮性、
特定的应用需求。

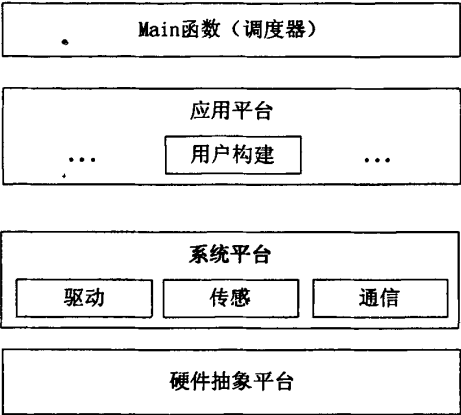


图 3-5 TinyOS 系统框架

TinyOS 的系统框架如图 3-5 所示。为了实现对多样化的无线网络传感器应用的支持，TinyOS 基于构件方式组成，主要由主控构件(包括调度器)、应用构件、系统构件和硬件抽象构件组成。硬件抽象构件实现对无线网络传感器硬件平台的抽象，包括底层的传感、通信、I/O 和电源控制等设备模块，为上层应用平台屏蔽底层硬件细节，增加系统平台的硬件独立性。系统构件包括设备驱动、通信、传感等构件，其中设备驱动构件包含对基本硬件模块的通用化接口；通信构件包括各种数据传输协议，如 MAC 层协议、传输层协议、路由协议和应用层协议等；传感构件支持传感设备的控制和数据收集。应用构件是用户根据具体应用的需求而定义的功能模块，实现具体应用相关的功能和策略。主控构件用于实现整个操作系统的控制流程，主要是进行无线传感网络的初始化、系统运行状态的维护，并进行任务调度。

TinyOS 的组件包括两类：模块(module)和配置(configuration)，组件间通过配置文件连接在一起，形成一个可执行程序。组件提供或使用接口，这些

接口是双向的并且是访问组件的唯一途径。每个接口都定义了一系列函数，包括命令(command)和事件(event)两类。对于命令，接口的提供者必须实现它；而对于事件，接口的使用者必须提供实现。组件的功能模块如图 3-6 所示，它包括一组命令处理函数、一组事件处理函数、一组任务集合和一个描述状态信息和固定数据结构的框架。除了无线传感网络操作系统提供的处理器初始化、系统调度和运行时库 3 个组件是必需的以外，每个应用程序可以非常灵活地选择和使用无线传感网络操作系统组件。

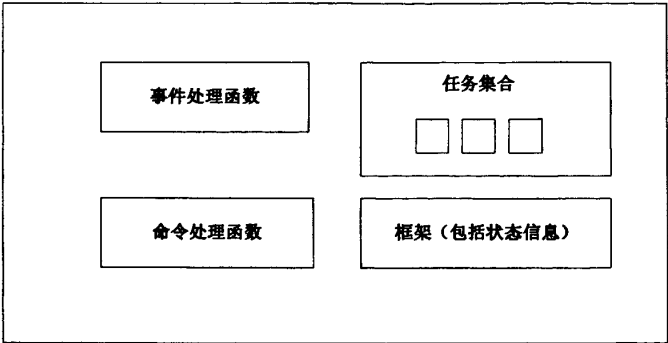


图 3-6 TinyOS 的功能模块

与一般的嵌入式操作系统相比，TinyOS 有其自身的特点：采用模块化设计，所以核心尺寸小（一般来说核心代码和数据大概在 400Bytes 左右），可突破无线传感网络存储资源少的限制；基于可重用组件的体系结构，使用事件驱动模型，通过事件触发来唤醒 CPU 工作；单一任务栈，内核非常简单，甚至在严格意义上说，称不上内核；没有进程管理和虚拟存储。这些特点使得 TinyOS 非常适合无线传感网络的需求，所以已得到了广泛应用。

3.3.2 TinyOS 调度机制

事件驱动的 TinyOS 采用两级调度：任务和硬件事件处理。任务是一些可以被抢占的函数，一旦被调度，任务运行完成彼此之间不能相互抢占。硬件事件处理被执行去响应硬件中断，可以抢占任务的运行或者其他硬件事件处理。TinyOS 的任务调度队列只是采用简单的先入先出算法。任务事件的调度过程如图 3-7 所示。TinyOS 的任务队列如果为空，则进入极低功耗的 Sleep 模式。当被事件触发后，在 TinyOS 中发出信号的事件关联的所有任务被迅速处理。当这个事件和所有任务被处理完成，未被使用的 CPU 循环被置于睡眠状态而不是积极寻找下一个活跃的事件。

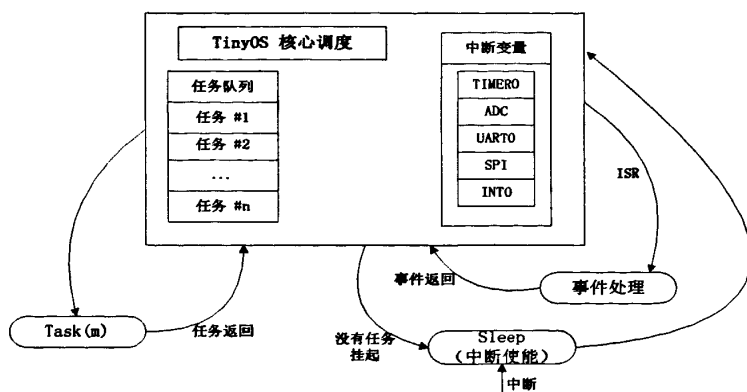


图 3-7 TinyOS 调度系统

为了减少中断服务程序的运行时间，降低中断响应延迟，中断服务程序的设计应尽可能精简，以此来缩短中断响应时间。TinyOS 把一些不需要在中断服务程序中立即执行的代码以函数的形式封装成任务，在中断服务程序中将任务函数地址放入任务队列，退出中断服务程序后由内核调度执行。内核使用一个循环队列来维护任务列表。默认情况下，任务列表大小为 8。图 3-8 是任务队列为空时的情形，图 3-9 表示有三个任务在队列中等待处理。内核根据任务进入队列的先后顺序依次调度执行。TOSH_run_next_task()函数负责从队列中取出指针 TOSH_sched_full 所指的的任务并执行。内核在一个无限循环中调用 TOSH_run_next_task()，当队列不为空时依次执行所有任务函数。

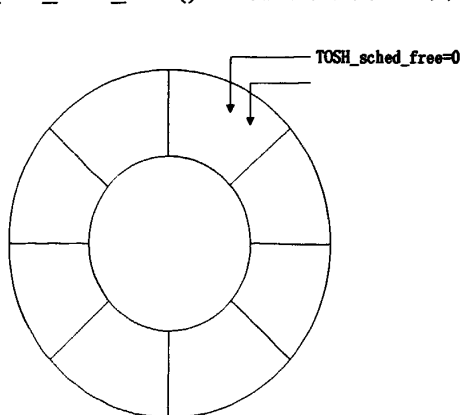


图 3-8 任务队列为空

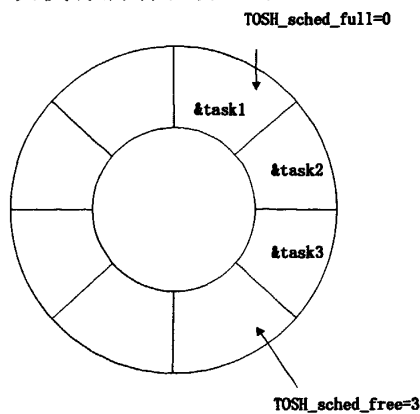


图 3-9 任务队列中任务数为三

由前所述，TinyOS 调度模型有以下特点：

(1) 任务单线程运行到结束，只分配单个任务栈，这对内存受限的系统很有利。

(2) 没有进程管理的概念，对任务按简单的先入先出队列进行调度。对资源采取预先分配，且目前这个队列里最多只能有 7 个待运行的任务。

(3) 先入先出的任务调度策略是电源敏感的。当任务队列为空时，处理器休眠，随后由外部事件唤醒 CPU 进行任务调度。

(4) 两级的调度结构可以实现优先执行少量同事事件相关的处理，同时打断长时间运行的任务。

(5) 基于事件的调度策略，只需少量空间就可获得并发性，并允许独立的组件共享单个执行上下文。同事事件相关的任务集合可以很快被处理，不允许阻塞，具有高度并发性。

(6) 任务之间互相平等，没有优先级的概念。

3.3.3 TinyOS 的局限性

尽管 TinyOS 被广泛使用，并且得到了相当的认可，但这并不意味着 TinyOS 能够适用于无线传感网络所有的应用场景。事实上，在某些场合下，TinyOS 并不能工作得很好，而可能出现过载，导致任务丢失、通信吞吐量下降等。^[20, 21]

以下几种场合，TinyOS 的调度策略也可能导致出现问题：

(1) 某些任务执行时间很长，这时如果某些实时任务在该任务之后才进入任务队列，就会影响实时性；对于数据包的收发，就会影响波特率。

(2) 本地任务发生频率过高，任务队列很快被本地任务填满，其它任务就可能丢失；此外，如果本地任务过多(例如多个通道同时进行采集，则本地任务数量多)，也会影响通信的正常进行、

(3) 任务队列中某个任务如果意外出现阻塞或异常时，会影响后续任务的运行，甚至导致系统瘫痪。

传感器网络中，节点典型的一个任务为：接收待转发的路由数据包、将接收到的数据包转发出去、处理本地的传感数据并将其发送出去。节点上任务的多少取决于节点处理数据的方式。如果节点只是直接把原始数据发往基站，则任务大多数是通信路由任务；如果节点在本地采集数据并处理后才往基站发送，则本地处理任务比较多。当节点上待处理的任务超过其处理能力时，就会发生过载。对于前一种情况，如果节点上发送数据的频率过高或者网络节点密度过大导致通信任务过多时，就可能发生过载；对于后者，如果本地待处理的数据量过大或者本地任务发生频率过高，也会导致过载的发生。

另外，当节点上中断发生频率很高，导致 CPU 除了进行中断处理外不执行其它任何任务时也会出现过载(也叫接收活锁)。当系统处理任务的速率低于任务发生的频率时，任务队列(当前只能存放 7 个任务)很快就满了，则会导致任务的丢失。对于本地的传感采集速率，可以人为调节控制，例如降低节点采样频率；但对于通信路由任务的发生，则不太好人为干涉。这时如果发生过载，则直接导致通信数据包吞吐量的下降。

发生这种现象的主要原因是数据包的发送和接收受制于本地任务。当本地任务发生频率过高时，任务队列很快就满了，这时发送或接收任务可能丢失，从而导致数据包丢失；另外，如果本地任务运行时间过长，则发送或接收数据包的任务要等待较长时间才能得到处理，从而降低通信速率。

3.4 两类典型调度策略比较

由以上介绍可知，目前无线传感网络节点操作系统的调度系统，可分为截然不同的两类：事件驱动单线程系统（Event-driven System）和多线程系统（Mufti-threading System）。^[19, 22-32]事件驱动系统以 TinyOS 为代表，其基本特点是任务由中断（事件）产生。任务持续运行直至结束（run-to-completion），能被中断抢占却不能被其他任务抢占。事件驱动系统所有任务一个堆栈空间，没有堆栈切换开销。多线程系统以 MANTIS OS 为代表，是一种类 Unix 的调度系统。事件驱动系统与多线程系统孰优孰劣的辩论由来已久。本节将结合无线传感网络节点资源限制，分析事件驱动系统与多线程系统各自优劣。

3.4.1 实时性

事件驱动系统诞生于简单的应用环境，节点操作系统中任务有限（TinyOS 1.x 任务队列最长为 8），且运行时间普遍很短，大大小于采样周期和任务的截止期。在这样的情况下，即使平均响应时间最长的 FCFS 调度方案也能保证每个任务的时间要求。

随着无线传感网络应用领域的不断拓展，尤其是在远程测控领域和视频采集领域的应用，使得任务数量和执行时间大大增加，网络任务的实时性要求也大大增强（紧急情况的响应）。在这样的情况下，一个长执行期的任务可能长期阻塞网络任务的执行，系统的实时性无法保证。多线程系统通过引入抢占机制和相应的实时调度算法（RM，DM，EDF），使得实时任务能够及时得到响应。

其中一个典型代表为生产者-消费者问题。生产者消费者问题的产生源于有限的缓冲区空间，它的一般描述为：生产者和消费者共享同一个缓冲区。生产者将产生的数据放入缓冲区，消费者在每次获取执行权时消费（即清空）缓冲区中的所有数据。当消费者长期得不到执行权，而生产者却不断往缓冲区添加数据时，缓冲区就会最终溢出。

具体到节点操作系统上，生产者-消费者问题发生在网络传输中。数据由感知节点（sensor node）发送到基站节点（base node）。每当一个网络数据包到达，都会触发一个中断。在中断处理程序中，数据包被放入缓冲区，等待另一个任务进行处理。这个任务通常是协议栈，它读取网络数据包，并将其发送出去。在这里，感知程序扮演生产者角色，而发送程序则是消费者。

随着节点操作系统上应用的复杂化, 在一个非抢占系统中, 协议栈可能由于一个长时间运行的任务长期得不到执行权, 从而导致缓冲区的溢出。MANTIS OS 在 4k RAM 的 AVR 芯片上, 统计出运算编解码时间分别为 321ms 和 504ms。假设节点操作系统使用 200 字节的 RAM 空间作为网络协议栈的缓冲区, 而网络数据包以 50 packets/second 的速度到来, 每个网络数据包的大小为 30 字节。那么, 网络数据的到达速度是 600 bytes/second。在此情况下, 运行时间达到 300ms 以上的任务, 就确定会造成缓冲区溢出。

3.4.2 存储资源消耗

事件驱动系统所有任务共享一个堆栈空间, 并且由于任务 run-to-completion 的特性, 一个 I/O 阻塞操作将分别由两个任务完成, 一个启动 I/O, 一个在 I/O 操作完成负责相应处理。使得任务中函数嵌套深度降低, 单个任务所需堆栈空间减小。

多线程系统为每个任务分配一个堆栈空间, 多线程任务不如事件驱动任务小巧, 并且由于往往过量估计任务所需堆栈, 多线程系统的运行时堆栈空间是事件驱动系统的 n 倍以上, n 为多线程系统任务个数。MANTIS OS 最多允许 15 个线程同时存在, 每个线程的缺省堆栈空间为 128 字节, 运行时堆栈空间接近 2K, 当前节点 RAM 空间在 2k-256k 之间, 并且大部分为 2k 和 4k。因此, 为每个任务分配一个堆栈的做法对于珍贵的节点 RAM 空间而言过于奢侈。

3.4.3 能量消耗

无线传感网络节点中最有效的节能方案是在节点无事可做时进入睡眠状态。节点操作系统普遍采用这一方案。在事件驱动系统中, 节能可以通过在任务队列没有任务时进入睡眠模式而自然进行。多线程系统也同样可以在所有线程都无事可做时进入睡眠状态。

虽然同样在节点无事可做时进入睡眠状态, TinyOS 睡眠时间大于 MANTIS OS。其原因在于多线程调度有着更大的调度开销, 与线程切换次数成正比。使得同样的任务集在多线程环境下的运行时间长于事件驱动单线程环境。

3.5 可调度性分析

可调度性分析^[21, 23, 24]用于在已知任务属性和运行方式的前提下验证系统运行是否满足截止时间要求。根据所使用的方法, 现有的可调度性分析技术也可以分为两类: 一类是基于 CPU 利用率的分析方法, 这类方法通过将整个系统的 CPU 利用率与某个边界值进行比较来确定系统的可调度性。这类方法的运行开销小, 适合应用于系统运行时的在线分析。但此类方法的缺点是, 满足边界比较只是判断系统可调度性的充分条件, 很多 CPU 利用率高的可调度任务集无

法通过条件验证，会被认为是不可调度的，因此这类方法的精确性不高。另一类调度分析方法基于对任务最大响应时间的计算。其原理是计算每个任务在各种可能情况下的最大响应时间，将这一时间与任务的截止时间做比较，当所有任务的最大响应时间都小于或等于各自的截止时间时，该任务集是可调度的，否则任务集是不可调度的。与基于 CPU 利用率的方法相比，这类方法的精确性更高（取决于任务最大响应时间的计算），但是算法的复杂度通常会比较高，尤其是在采用分布式结构或任务数量较多的情况下。因此这类方法多用于系统设计时的离线分析。由于可调度性分析问题是实时系统相关的调度理论中最重要的理论，下面将分别介绍两类分析技术的发展现状。

3.5.1 基于 CPU 利用率的可调度性分析

基于 CPU 利用率进行可调度性分析的关键是确定特定某个模型下的 CPU 利用率的边界值。不同的调度策略下计算边界值的方法和获得的边界值有所不同。

此调度分析方法需满足如下假设条件：

- 所有的任务都必须具有严格的周期性；
- 任务的截止时间与其周期相等；
- 所有任务都是相互独立的，不存在依赖关系；
- 任何任务都不会在运行过程中主动地挂起自己；
- 所有任务运行在单个处理器上。

考虑 n 个独立的可抢占任务在单个处理器上运行的情况。任务集 S 可以表示为： $t_1(C_1, T_1), \dots, t_n(C_n, T_n)$ ， T_i 和 C_i 分别是任务 i 的运行周期和最大执行时间。假设任务的截止时间与其运行周期相同，则整个任务集的 CPU 资源利用率 U 可以表示为：

$$U = \sum_{i=1}^n C_i / T_i \quad (3-1)$$

如果下面的不等式成立，则 RM 算法可以调度任务集 S 中所有的任务。

$$U \leq n(2^{1/n} - 1) \quad (3-2)$$

根据公式 3-2，当任务数量越多时，满足可调度条件的 CPU 利用率越低。当 n 趋向于无穷大时，满足可调度条件的最小 CPU 利用率将逼近 0.69。对于 EDF 算法，当满足 $U \leq 1$ 时，任务集是可调度的。与 RM 算法相比，EDF 算法可以达到更高的 CPU 利用率，但 EDF 算法的实现机制较为复杂，在实际应用中，RM 算法使用更为广泛。

基于 CPU 利用率的可调度性分析方法，由于算法计算复杂度不高，运行开销小，非常适合应用于在线分析。但根据 CPU 利用率给出的可调度条件往往只是充分条件，无法对所有的任务集做出精确的判断。另外，对于分布式实时系

统，由于各处理器丘任务模型的差异性和处理器之间的通信关系使得整个系统运行行为变得更加复杂，很难给出统一的利用率边界值。目前趋向于通过计算响应时间来分析系统的可调度性。

3.5.2 基于系统响应时间的可调度性分析

实时系统通过任务来响应外部事件并在指定的时间内做出反应，如果能够确定每个任务在各种可能情况下的最大响应时间，就可以通过比较这个响应时间和任务的截止时间来判断任务的可调度性。当系统中所有任务的最大响应时间都小于或等于其截止时间时，整个系统是可调度的。

在单处理器系统中，对于一个由 n 个任务组成的任务集 $S=\{t_1, \dots, t_n\}$ ，任务 i 的属性可用三元组 (C_i, T_i, D_i) 来表示， C_i 、 D_i 分别是任务 i 的最大执行时间和截止时间， T_i 是任务 i 连续两次到达的最小时间间隔，对于周期性任务 T_i 是任务的周期。一个任务实例的响应时间由任务实例本身的执行时间、任务实例被低优先级任务阻塞的时间以及被高优先级任务阻碍的时间三部分组成。低优先级任务的阻塞时间通常是由于等待不可抢占的低优先级任务或等待低优先级任务释放资源导致。而高优先级任务的阻碍时间是指任务实例运行过程中由高优先级任务优先执行或抢占执行所导致的等待时间。当三个部分的时间全部取得各自的最大值时，可以得到任务的最大响应时间。因此，任务 i 的最大响应时间可以表示为：

$$R_i = C_i + B_i + I_i \tag{3-3}$$

其中 R_i 为任务 i 的最大响应时间， C_i 是任务的最大执行时间， B_i 是被低优先级任务阻塞的最大时间（Maximum Blocking Time）， I_i 是被高优先级任务阻碍的最大时间（Maximum Interference Time）。任务的最大响应时间构成如图 3-10 所示，其中白色箭头表示任务实例的释放时刻，释放时刻是指一个任务被触发后进入就绪状态开始等待操作系统调度的时刻。

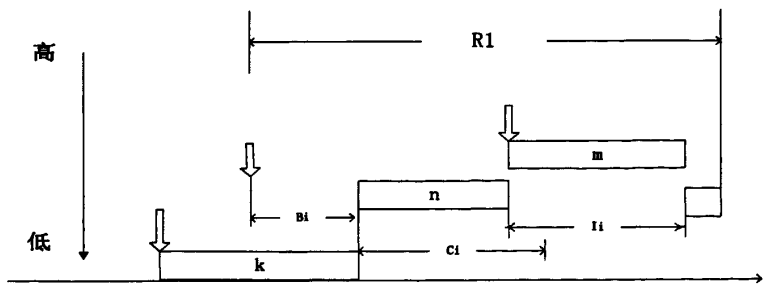


图 3-10 任务最大响应时间的构成

任务的最大执行时间 C_i 可以通过特定的分析和测量方法获得。 B_i 由系统所采用的任务调度和资源访问控制机制决定。当任务 i 就绪时无法抢占正在运行的低优先级任务时，任务 i 必须阻塞直到这个低优先级任务执行完毕，我们用

B_{wi} 表示这个阻塞时间的最大值。 B_{wi} 由系统采用的调度策略决定，例如当采用抢占调度时， $B_{wi}=0$ ，而采用非抢占调度时， B_{wi} 是所有低优先级任务中的某一个最长执行时间。任务运行时可能会出现资源竞争。在特定的资源访问控制协议的作用下，任务 i 被释放后可能需要等待一个低优先级任务释放一个共享资源，我们用 B_{ri} 表示这部分阻塞时间的最大值。 B_{wi} 由系统采用的资源访问控制协议决定。根据资源访问控制机制，当任务 i 就绪以后最多被个低优先级任务阻塞一次，因此 B_{ri} 和 B_{wi} ，不会同时存在， B_i 是 B_{ri} 和 B_{wi} 中的最大值，可表示为：

$$B_i = \max(B_{ri}, B_{wi}) \quad (3-4)$$

I_i 是由高优先级任务的运行造成的。在任务 i 运行过程中，高优先级任务运行次数越多，对任务 i 的阻碍时间越长，任务 i 的响应时间也就越长。任务 i 和高优先级任务之间的相位关系，决定了 I_i 的值。为了计算任务的最大响应时间，必须找到导致任务响应时间最长的运行相位。

3.6 本章小结

本章对无线传感网络中使用的操作系统的调度策略进行了研究分析。介绍了 MANTIS OS, SOS, Contiki 和 $\mu C/OS-II$ 的调度机制。接下来重点分析了 TinyOS 的调度机制，并在此基础上分析了其局限性。然后，本章针对两种典型的无线传感网络操作系统的调度策略：基于事件驱动的调度和多线程实时调度，从实时性，存储资源消耗，能量消耗等几个方面进行了比较分析。我们的工作正是结合了事件驱动和实时性这两种调度策略的思想所提出来的，在 TinyOS 操作系统的组件结构基础上加入了实时性的调度思想。最后，本章给出了可调度性分析的概念和方法。

第四章 基于优先级的可抢占式分级调度策略

目前,任务的调度算法很多,有基于优先级调度、基于时限的调度、时间片轮转调度、静态调度等。针对多的应用场景,以什么原则、选择何种调度策略是一个值得思考的问题。在第三章分析的基础上,我们提出了一种基于优先级的可抢占式分级调度策略,结合了事件驱动、实时性和分级调度的思想。

4.1 设计目标

对于无线传感网络而言,由于供能的限制,节点操作系统不以处理器利用率为目标。无线传感网络应用环境并非多用户交互式环境,对单位时间内完成的任务数量并不关心。同时,当前多数无线传感网络节点操作系统是一个弱硬实时操作系统,对实时性的支持不够。因此,我们将满足响应实时任务的时间性要求,即实时性,作为节点调度系统的首要设计目标。同时,由于节点存储资源的限制,节点操作系统要求尽量减少自身代码量和对堆栈空间的需求,这一要求仅次于系统实时性要求。另外由于传感器节点能量限制,调度系统需要特别考虑低功耗需要,这一点包含了如何缩短任务执行时间、尽可能减少内容交换的考虑。对于一个调度系统而言,不能每项指标同时达到最优,需要进行取舍。并且,由于无线传感网络应用领域的广泛性,调度系统需要一定的可定制性以适应不同领域的要求。

综上所述,我们给出调度系统设计目标:

(1) 实时性

由于节点部分任务的实时性要求严格,要求使用适当的实时调度策略,满足实时任务的时间要求。

(2) 灵活性

由于无线传感网络应用领域极其广泛,调度系统需要具备一定的可定制性,以满足不同应用领域的特定要求。

(3) 资源消耗

由于节点 RAM 空间和 Flash 空间的限制,要求调度系统的对数据存储空间(RAM)和代码存储空间(Flash)的占用尽量小。另外,无线传感网络节点需要利用有限的电池电量运行相当长时间(几个月乃至几年),调度系统在设计上也需要考虑到节点低功耗的要求。

4.2 常用实时调度策略分析

4.2.1 常用实时调度策略

在普通嵌入式系统领域，众多研究人员提出了各种实时调度算法，以满足该领域操作系统对任务执行有效性和容错性的要求。各种实时操作系统的实时调度算法可以分为如下三种基本类别：^[21, 23]

- 基于时间的进程调度算法

该调度算法适用于那些稳定且输入已知的简单系统。这种调度算法要求在系统的初始设计阶段，就能够清晰的确定系统中所有可能出现的处理情况，包括各个任务的开始、切换、以及结束时间等，并事先完成详细的调度分析，做出调度安排。

基于时间的进程调度算法本质上是一种确定的静态调度方法，因而其优点很明显：能够为任务运行情况提供良好的预测。但该机制的缺点也同样显著，就是过于僵化，缺乏灵活性，甚至可能出现有任务需要执行而 CPU 却保持空闲的情况。

基于时间的进程调度算法通常适用于那些很小且极其简单的嵌入式系统和自控系统。对于应用丰富多彩，任务类型变化多样的无线网络传感器系统来说，该算法由于无法精确预测各种突发任务，仍然有些不足。

- 基于优先级的调度算法

PD 调度算法为每个调度对象分配一个优先级，以此作为调度依据，调度器总是将 CPU 执行全分配给优先级最高的调度对象。根据优先级决定时间的不同，PD 调度算法可进一步分为如下两类，而每一类又有可抢占与不可抢占之分。

静态 PD 调度算法的调度器中，调度对象的优先级是静态决定的，即编译期决定。决定优先级的因素多种多样，如任务周期，抢占阈值等。基于速率(Rate-Monotonic- RM)算法和基于截止期(Deadline-Monotoni-DM)算法便是两种典型的静态 PD 调度算法，分别以调度对象的周期和截至期作为优先级决定因素，周期或截止期小的调度对象具有更高的优先级。先入先出算法从本质上看也属于静态 PD 调度算法，只是仅仅使用一个优先级。由于优先级编译期可决定，这类调度算法的可调度性分析通常离线执行。

动态 PD 调度算法，使用动态 PD 算法的调度器中，调度对象的优先级动态决定，即运行期决定，通常为调度对象释放的时刻。决定优先级的因素通常包括任务的绝对截止期、作业的执行时间等。最先截止优先(Earliest-Deadline-First - EDF)算法是最著名的动态 PD 调度算法之一，该算法总是选取任务队列中，拥有最小绝对截止期的任务进行调度。与静态 PD 算法不同，动态 PD 算法的可调度性测试通常在线执行。在每一个新任务到达时，对当前任务队列中的所有任务进行可调度性测试。

- 基于 CPU 使用比例的共享式调度算法

SD 调度算法没有定义优先级的概念,所有调度对象以一定比例共享处理器资源。当系统中任务数量增加,所有任务获取的处理器时间将按比例减少,所有任务的执行时间按比例增加。时间片轮转算法是 SD 调度算法的典型代表,处理器按照一定权重为任务分配时间片,时间片长度与权重成正比。为了保证系统中实时任务获取足够的处理器资源,通常采取预留处理器带宽和动态调节任务权重的方法。

以 SD 算法调度的任务,其时间片耗尽后,即使尚未执行完成,也必须放弃 CPU。因此仅仅适用于抢占式调度,即子调度器之间的调度。基于优先级的调度算法简单而有效,比较符合传感器应用中任务调度设计的需求。尽管基于该类型的调度算法多种多样,但大多由基于速率算法(RM)和最先截止优先算法(EDF)变化而来。在不同的系统状态下两种算法各有优劣:静态优先级调度算法主要用于静态周期任务的调度,在编译期就进行任务调度计算,节省了运行期消耗,但不能很好的适应多变的无线网络传感器应用。相反,动态优先级调度算法者主要用于动态调度,在运行时根据系统要求动态确定调度策略。然而,对于计算资源高度受限的传感器上微处理器,除了执行任务调度以外,还必须处理同步、并发、I/O 等其他复杂的操作,完全的动态调度可能加重本已高负荷的微处理器的负担。因而,简单的应用现有的这些技术很难保证能够在任务调度中实现传感器应用中常有的截止时限要求。这就要求我们在实际应用中的实现策略,往往是综合考虑各种因素的折中。

4.2.2 调度算法运行时代价分析

调度器的运行时代价可以定义为运行调度器代码所占用的时间。对于同一任务集合而言,调度器运行时代价越高,任务集整体执行完成时间将越长,对节点而言,处理器活动时间将越长。

我们首先作出如下定义:

T_{in} : 任务插入时间。包括选择合适的位置和具体的插入时间。

T_{out} : 从任务队列中选择下一个可执行任务的时间。

T_{switch} : 调度器切换时间

在先入先出的调度系统中,由于每一个任务 run-to-completion 的特点,仅有一个入队和出队操作,设任务在执行过程中被抢占次数为 0,则调度一个任务的运行时代价为:

$$\text{运行代价} = T_{in} + T_{out} + \sigma * T_{switch} \quad (4-1)$$

T_{switch} 时间由 MUC 芯片决定,以 avr 芯片为例,切换时间约为 400 个时钟周期。减少调度器堆栈切换开销的唯一途径是减少堆栈切换次数。为此,应将在不抢占方式下调度的不同任务放入同一个子调度器线程中。如此将调度

器的个数降低到到最小值，同时实现了另一个调度目标一尽量减少堆栈空间的消耗。 $T_{in} + T_{out}$ ，时间由子调度所选取的调度算法决定。

设 PD 算法共有 d 个优先级。则 $T_{in}+T_{out}$ ，的时间复杂度为 $O(d)$ 。当优先级个数为 1 时，PD 算法降级为 FCFS 算法， $T_{in}+T_{out}$ 的时间复杂度为 $O(1)$ ，当优先级个数与任务集中任务个数 n 相同时，PD 算法升级为 RM/DM/EDF 算法， $T_{in} + T_{out}$ 的时间复杂度为 $O(n)$ 。虽然时间复杂度相同，动态 PD 算法由于需要动态计算优先级， $T_{in}+T_{out}$ 的值将大于静态 PD 算法。

4.3 分级调度理论

比较分析得知，单一的事件驱动单线程调度系统或多线程调度系统均不能满足无线传感网络任务调度的要求，为了实现一个即满足无线传感网络节点资源限制，又适应多样的无线传感网络应用环境的操作系统，其调度系统必须同时克服事件驱动系统与多线程系统各自的弱点。

分级调度(Hierarchical Scheduling)^[23]是由 Liu 与 Z.Deng 于 1997 年首先提出。Z.Deng 设计的初衷是为了改进传统 OS 调度模型中所有任务混合调度的方案，将实时任务和非实时任务分开调度，使实时任务不受非实时任务的影响。Z.Deng 设计了如图 4-1 所示两级分级调度模型，取名为 Open System Model:

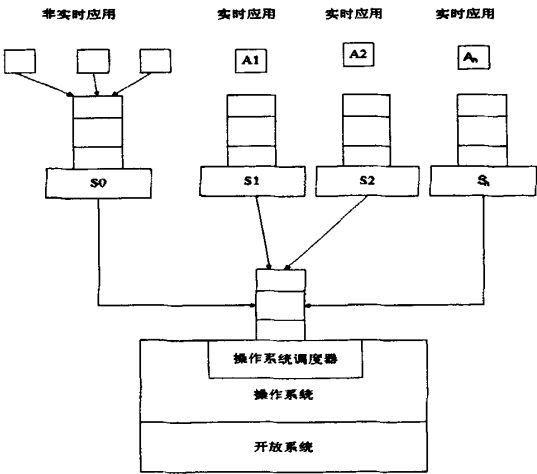


图 4-1 开放系统模型

在此模型中，系统任务集由一系列实时应用程序(Application，以 A_k 表示， $k \geq 1$)和非实时应用程序组成。系统为每一个实时应用程序分配一个常量利用率服务器(Constant Utilization Server，以 S_k 表示， $k \geq 1$)。所有非实时应用程序公用一个常量利用率服务器 S_0 。每一个应用程序由一系列工作组成，因此每一个常量利用率服务器维护一个工作队列，使用优先级驱动(priority-driven)算法调度工作。这些常量利用率服务器组成了分级调度模型的第二级，称为服务器调度器，调度对象为工作。

在模型第一级，系统为每一个服务器分配一定的处理器时间，实际上是处理器时间的占用比例，这也是常量利用率服务器名字的由来。同时，系统设置每一个服务器的截止期，以最早截止时间(Earliest-Deadline-First, EDF)算法调度各个服务器。这一级调度器被称为操作系统调度器，调度对象为常量利用率服务器。

具体说来，服务器调度器和操作系统调度器各自执行的操作如下：

对服务器调度器而言，当其执行预算大于 0 时，服务器是可执行的。服务器在执行过程中不断消耗执行预算，直到执行预算为 0。这时服务器变为不可执行，直到下一次获取新执行预算。这一点类似与 Linux 操作系统中时间片调度方案。与 Linux 操作系统不同的是，服务器调度器在获取执行预算的同时，设置自身的截止期。

在 Liu 与 Deng 基础之上，人们就两级之间各自采取何种调度方式以及可调度性分析进行了进一步研究。在整体上沿用了 Open System Model 为每一个应用程序分配一个服务器，服务维护一个预留时间片，且研究集中于两级调度器均可抢占的模型。

4.4 基于优先级的可抢占式分级调度策略设计

4.4.1 基本思想

为了满足长期无人看管的无线传感网络操作要求，传感器节点的设计必须是能量有效的。然而，能量有效性并不是传感器网络设计的唯一目标。在特定的应用环境下，实时的过程和感知信息的报告也是常常需要的。这样，感知的数据从某个节点，通过多跳网络，最终传送到基站的过程中，我们可能需要保证一个尽可能长的传输时间。为了能够提供这样的保证，具有确定的行为的网络组件是必需的。运行在传感器节点的操作系统就是这样一个组件。

事件驱动的操作系统被认为是建立能量有效的传感器网络的最佳选择，因为它们只需要很少的内存和处理资源。这样，事件驱动的操作系统 TinyOS 成为了当前传感器网络操作系统的首选。事件驱动的操作系统在任务实时性要求较高的环境下就不是很有效了。由于所有的任务都是按照次序顺序执行的，具有优先级的重要的任务要保证在时限之内被处理是不可能的。如果这样的要求是必须满足的，那么多线程操作系统就显得更加合适。线程可抢占和内容交换使得这样的系统能够按照优先级执行任务，实时任务能够及时得到响应。

MANTIS OS 就是一个专门针对无线传感网络设计的多线程操作系统。MANTIS OS 有一个相关的高处理开销的线程管理程序。这个处理过程的开销直接导致能量有效性的减少，因为这个使得 CPU 的活动性相对的增加了。

因此，这种状况导致了一种两难的情况，我们的两个设计目标：能量有效性和实时性，在一个状态下只能单独优化其中一个。人们必须选择在当前

的应用场景中哪个目标是更加重要的。因此，如果能够通过使得 MANTIS OS 更加的能量有效或者是使得 TinyOS 能更快的响应来解决这样两难的状况那是再好不过了。也就是说，我们要借鉴这两种系统的优势，并尽可能避免它们的局限性来设计调度策略。

传感器网络研究团体其实将 TinyOS 定为了事实上标准，大部分现有的应用、库和设备驱动都是对 TinyOS 的支持。因此，为了避免对已有软件的重复编码，并重用已有的 TinyOS 的设备，本文提出了一种改进方案，就是在 TinyOS 中引入基于优先级的抢占机制，在它原有的结构和特点的基础上形成可响应的系统。

我们对 TinyOS 改进的第一步就是开发一个新的基于组件的优先级调度器来代替 TinyOS 系统提供的标准的先入先出调度器。根据用户应用程序的性能要求，这个新的优先级层次调度器（PL 调度器）可以被嵌入应用程序中对任务首先执行的情况来达到更好的控制。PL 调度器提供了以下几种不同的优先级别：

（ P_1 ）高优先级可抢占

（ P_2 ）高优先级非可抢占

（ P_3 ）基本优先级（标准 TinyOS 任务使用）

在每个层次中，任务按照先进先出的方式进行调度。基本优先级层次的调度是必须实现的，因为所有的标准 TinyOS 任务都将默认在这个队列中。相邻的优先级层次提供了一个不可抢占的高优先级的队列和一个可抢占的高优先级队列。这样，这三个层次中的任意任务都能按照他们的优先级进行调度，但不会抢占正在执行的任务，正如图 4-2 所示。高优先级可抢占任务队列是用来调度抢占一般任务的。一个高优先级可抢占的任务将会从优先级低于它的任务层次中抢占任何正在运行的任务，并且这些层次中的任何任务可以优先于任何一个优先级低于自身的任务而先得到响应。这个抢占机制的具体的实现我们将在下文讨论。

在实际应用中，并不是所有层次的优先级都需要，因为将一个任务分配到多个不同的优先级层次中将会导致一个膨胀的调度器。TinyOS 的组件体系结构使得我们可以在编译期间统计任务的个数。PL 调度器就能够准确确定需要多少队列，而且 nesC 编译器的代码省略特性能够将多余的任务优先级结构去掉。如果需要超过三个的优先级层次，那么 PL 调度器也能够扩展来提供支持。然而，我们相信在一般的传感器网络应用环境中，三个层次已经足够支持了。

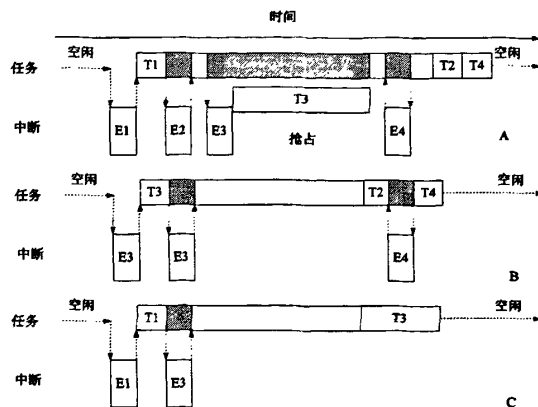


图 4-2 改进的操作系统任务处理

实时系统中，如果任务调度采用基于优先级的方式，则传统的资源共享访问机制在系统运行时很容易造成优先级翻转问题，即当一个高优先级任务访问共享资源时，该资源已被一低优先级任务占有，而这个低优先级任务在访问共享资源时可能又被其它一些中等优先级的任务抢先，因此造成高优先级任务被许多具有较低优先级的任务阻塞，高优先级任务在低优先级任务之后运行，实时性得不到保证。为此需要采用特定的资源访问控制协议来避免这类情况的发生，同时将任务访问资源所造成的阻塞时间控制在一个合理的范围之内，保证任务响应时间的可预测性。

在实时系统中，对传统的资源访问机制进行扩展时，通常引入了如下两种机制：优先级继承协议(Priority Inheritance Protocol)、优先级限高协议(Priority Ceiling Protocol)来解决了解决了优先级翻转^[24]的问题。

优先级继承协议适用于固定优先级抢占调度策略。根据该协议，当一个高优先级任务 A 访问的资源被低优先级任务 B 占用时，任务 B 的优先级将提升为任务 A 的优先级，这样优先级介于 A、B 之间的任务将不会抢占任务 B，也就不会导致任务 A 被这些低优先级任务不断阻塞。当任务 B 释放资源时，任务 B 将恢复原来的优先级。采用这种机制可以将每个任务的阻塞时间限制在一个比较小的确定时间间隔内，避免了不受控的优先级反转情况的发生，但是该协议本身无法避免死锁，必须采用其他辅助手段避免死锁的发生。

为了同时解决死锁和优先级反转问题，Sha 等人还给出了另外一种资源访问控制协议—优先级置顶协议(Priority Ceiling Protocol)。其思想是为系统中的每个资源预先分配一个置顶优先级，这个优先级必须大于或等于所有访问这个资源的任务的优先级同时还必须小于那些不访问该资源而优先级又高于所有访问该资源的任务的那些任务的优先级。当一个任务请求一个没有被占用的资源时，若任务的当前优先级低于资源的置顶优先级，该任务的优先级将上升为资源的置顶优先级。当任务释放了该资源时，任务的优先级又会重新恢复到访问

资源以前的优先级。此外，该协议禁止任务以嵌套的方式访问临界资源，即禁止在释放一个资源之前又去占用另一个资源，以避免死锁的发生。优先级置顶协议会给优先级低于置顶优先级的任务造成一定的等待延迟，但是这个延迟是受限的，最坏情况下是所有低优先级任务对这个资源最长的一次占用时间。优先级置顶协议在实时系统中得到了广泛应用。

在通用的任务调度不能满足应用程序的时间要求时，PL 调度器使得任务抢占变得更容易实现。

任务抢占会带来内容交换的代价，这需要由操作系统来支持。内容交换必须谨慎的实现，以避免造成系统开销和能量开销的巨大增加。我们之前关于多线程操作系统 MANTIS OS 的最佳抢占调度的研究表明了减少内容交换的次数是极其重要的。在这个设计要求下，PL 调度器用两种不同的可能的原理来避免抢占。

在第一种原理中，只有当需要匹配处理时限的时候才会进行内容交换。图 4-2 说明了这种思想的一个例子。任务 T_1 以基本优先级 P_3 在运行，任务 T_3 具有 P_1 优先级（高优先级可抢占），它在中断时间 E_3 结束时被调度。为了处理高优先级的任务 T_3 ，内容交换就是需要的。这样，内容并不被交换，原来的任务 T_1 执行到完成（见图 4-2A）。如果同样的高优先级任务 T_3 在系统空闲的这样一个环境下被调度（见图 4-2B），不需要有内容交换，那么任务就立即执行。在这种情况下，高优先级的任务 T_3 将会以标准内容执行。换句话说就是，内容交换并不和优先级层次相关，而是和抢占的需要相关。这种思想和多线程操作系统中现有的抢占机制相比，减少了内容交换的次数。多线程操作系统中高优先级线程的执行通常都一定要进行内容交换。

因为第二种原理减少了内容交换，我们采用了一个特殊的时段 t 来实现抢占。它假设，很多高优先级任务都需要在一个特殊的时间帧内被执行，但并不需要立即执行。为了满足时限要求，有一个时间戳来表示任务最晚被执行的时间点。如果当前运行的低优先级任务在特殊时间 t 之前完成，那么所有任务都能够被执行而不需要抢占。任务 T_1 以基本优先级 P_3 运行，任务 T_3 以优先级 P_1 （高优先级可抢占）在中断周期 E_3 结束之后被调度。任务 T_3 拥有特殊时段 t ，这样就并不需要及时地抢占来实现高优先级任务的调度。

PL 调度器需要三个独立的内存空间栈来存储三个抢占任务优先级层次的执行状态（ P_1 , P_2 , P_3 ）。因为只有三个可抢占优先级，所以只需要三个栈。需要的栈的数目依赖于可抢占优先级层次的数目而不是使用的任务的数目。由于所用的栈的数目是确定的，所需栈的大小计算就简化了。实际上，所有任务中的绝大部分都将和通常的 TinyOS 任务一样以基本优先级层次 P_3 运行在同一个栈中。

4.4.2 调度器结构设计

我们结合以上多种设计思想,设计出基于优先级的可抢占式分级调度系统,实现面向调度器的线程以及分层调度系统的智能构造算法。它实现了一个基于事件驱动系统的多线程调度系统,其结构如图 4-3 所示:

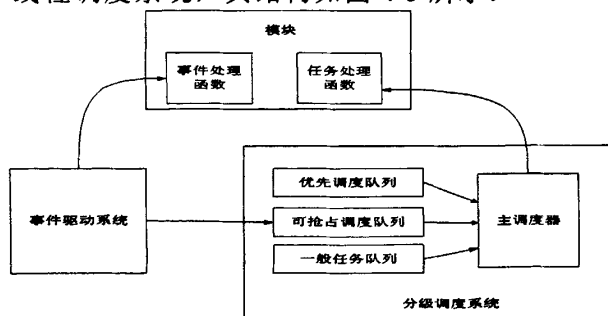


图 4-3 分级调度系统结构

系统程序模块提供三类调度对象:事件、任务与子调度器。事件由事件驱动系统调度,任务与子调度器则由多线程分层调度系统调度。因此,我们将系统调度分为三层,第一层为事件驱动系统与多线程分层调度系统;第二层为多线程分层系统中主调度器;第三层为多线程分层调度系统中子调度器。

事件驱动系统是硬件中断与多线程分层调度系统的通信枢纽。一个完整的事件驱动系统包括事件触发和事件处理两部分。硬件中断触发事件。而事件处理又分为两个阶段:第一阶段执行中断处理例程代码。第二阶段执行相应程序模块中的事件处理函数。与 TinyOS 相似,为了减少中断处理时间,提高系统并发性。主要处理工作被放到延迟执行的任务中。事件处理函数负责任务的投递,即将任务传递给上层多线程系统。

多线程分层调度系统是整个调度系统的核心。它接收事件驱动系统所提交的任务,调度任务并实现一定程度上的任务抢占。线程调度系统以分层调度的形式实现,共有二级调度器。

其中,子调度器负责任务的具体调度。每一个子调度器运行一个典型的 TinyOS 任务处理循环。实现一定调度算法。子调度器与事件驱动系统相结合,相当于 TinyOS 基于事件驱动的调度系统。每一个子调度器实现为一个独立的线程,作为主调度器的调度对象。主调度器提供线程创建、销毁和切换机制。这样,通过高优先级的子调度器对低优先级子调度器的抢占,实现了高优先级的任务对低优先级任务的抢占。

调度器通过进一步的分层,实现了事件驱动与多线程的结合,共有两级调度器。其中,第一级主调度器是一个微型轻量级线程管理模块,使用静态优先级可抢占的调度策略。第二级子调度器作为主调度器中的一个线程任务。子调度器可以灵活地选择自己的调度策略。但受到底层事件驱动模型的限制,调度

的对象为彼此不可抢占的任务。因此子调度器实现的调度策略都必须是非抢占式的调度策略，如非抢占 FCFS、非抢占 DM 算法。

在此分级调度系统中，最底层调度器不是应用程序。每一个底层调度器的调度对象是多个应用程序的任务集合。这样，多个应用程序共享一个堆栈空间。同时设计出分层调度系统非运行时构造机制，静态确定任务集合的可调度性和任务所对应的调度器，降低运行时调度开销。

4.5 实现细节

某个组件通过配置一个优先级任务接口到 PL 调度器组件中来指定一个优先级任务，并执行事件 runTask()函数接口。这个程序在任务和调度器方面和 TinyOS 增加建议 (TEP) 106^[33]是一致的。

算法 1 优先级任务结构

```
1:Module SomeComponentC{
2:    use interface PriorityTask<HighPreempt>;
3: }
4:Implementation{
5:    event void someEvent(){
6:        call PriorityTask.postTask()
7:    }
8:
9: event PriorityTask.runTask(){
10: //task code
11: }
12: }
13:Configuration SomeComponent{
14:}
15:implementation{
16: components new PriorityTask() as PreemptingTask;
17:    components SomeComponentC, ....
18:    SomeComponentC.PriorityTask->PreemptingTask;....
19:}
```

算法 2 优先级调度器结构

```
1:Module PLScheduler{
2:    provides interface TaskPriority<HighPreempt>[id];
3:    provides interface TaskPriority<HighNonPreempt>[id];
4:    provides interface TaskBasic[id];
```

```
5: provides interface TaskPriority<LowNonPreempt>[id];
6: provides interface TaskPriority<LowPreempt>[id];
7:}
```

算法 1 给出了一个实现了的优先级任务的例子。某个组件要使用优先级任务，它就必须实现优先级任务接口，并且指明任务的优先级作为接口的参数（算法 1，第 2 行）。这个接口提供了和基本任务语法 `post[task name]` 相同的 `postTask` 函数（算法 1，第 6 行），和存储任务功能的 `runtask` 事件处理器（算法 1，第 9 行）。当任务将要被调度执行的时候，事件处理器由调度器激活。

每个任务都必须被配置到由 PL 调度器提供的五个带参数的任务优先级接口中去（算法 2，第 2-6 行）。配置过程在某种程度上往往被普通优先级任务组件简化了（算法 1，第 16 行），它利用接口参数信息来确定任务的优先级并且唯一的配置每个任务到合适的调度器接口中去。

PL 调度器是对 TinyOS 提供的先进先出的调度器的扩展。根据操作系统任务优先级的数目，最多可以初始化五个不同的任务队列。有一小块存储将被初始化，用来跟踪任务的优先级，看它是被强占了还是正在执行。当接受到一个传来的任务时，调度器首先确认这个任务还没有被配置。第二，调度器检查这个任务会不会被一个低优先级的任务推迟运行。如果需要抢占，那么调度器就实行内容交换，或者设置一个特殊时段来延迟内容交换。特殊时段 `t` 对当前实现中的所有任务来说都是一个确定的全局值。

调度实现流程分别如图 4-4、4-5 所示，其中 4-4 表示了任务入对列流程，图 4-5 表示了系统的调度结构。

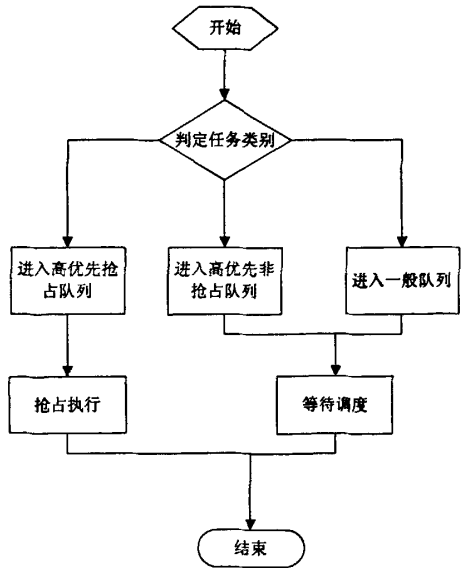


图 4-4 任务入队流程

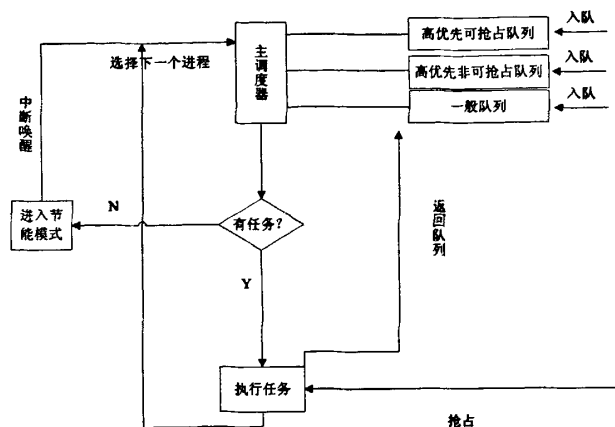


图 4-5 系统调度结构

内容交换要求将当前寄存器中内容保存到当前栈中，而且这个栈的指针寄存器要指向下一个栈的内容交换，这里面要包含被抢占的队列调度器。被抢占的队列调度器连续执行所有的任务，从最高优先级的任务一直到被抢占的任务。因此，调度器完成执行多个等待抢占的高优先级任务的要求，而只需要进行两次内容交换。当抢占队列中没有任务的时候，内容会交换回被抢占的任务内容，被抢占的任务就能够完成执行。

4.6 系统评测

4.6.1 实时性

基于优先级的可抢占式分级调度策略设计的首要目标是为了满足实时性要求，因此实时性是我们首先需要进行测试和验证的。我们通过优先级以及抢占机制的引入保证了系统的实时性。接下来我们从实时任务响应速度和任务集调度能力两方面进行分析。

TinyOS 系统中只有一个任务队列，一个任务一旦执行就一直执行到完成，任务不进行抢占。只有硬件事件才能对任务进行抢占，实时性要求高的任务若需要及时响应则需要设计成硬件中断。影响中断响应速度的根本原因在于中断关闭。在事件驱动系统中，由于中断负责产生任务，因此与任务队列相关的原子操作成为影响操作系统中断响应速度的主要原因。这样，实时任务的响应速度受到中断关闭的影响而会有延迟。通过软件的优先级和抢占，将中断处理例程中无需互斥而时间性强的部分与其余部分区分开，使前者始终能够立即执行，保证了中断响应速度。

TinyOS 使用先进先出的调度器，分级调度系统的子调度器也使用先进先出的调度，而主调度器按照优先级别进行调度进行实时任务响应的比较。我们假设三种不同优先级别的任务， T_0 ， T_1 ， T_2 ，分别对应高优先级可抢占任务，高

优先级非可抢占任务和一般任务。三种任务依次按照 T_0, T_1, T_2 的顺序每隔时间 t 循环发送，而每个任务的执行时间为 $2t$ 。

实验使用 Power-TOSSIM 进行仿真测试。使用的是 TinyOSv1.11 自带的 Power-TOSSIM 软件。该实验中每个节点上运行的用户任务为下 TinyOS 自带的例子程序 Surge，在使用我们提出的分级调度策略的 Tinyos 中对 Surge 进行了修改，确保其可以正常运行。仿真过程中，节点周期发送采集到的传感数据，无线通路为理想情况，不需要考虑无线信号、误码率等问题导致的丢包现象。但当无线传感节点分布密度较高时，对于连接度很高的多跳路由转发节点由于需要同时执行传感数据采集任务以及多跳路由转发任务，任务负载很重，可能会发生过载现象而导致丢包。其实时任务响应情况如图 4-6 所示：

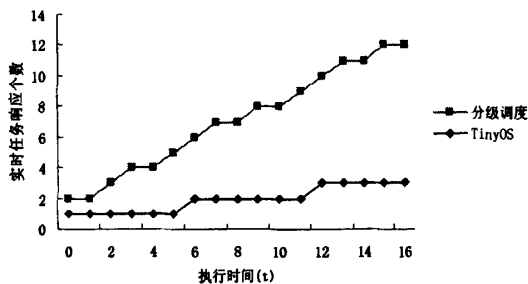


图 4-6 实时任务断响应比较

由此可以看出，分级调度策略能够对实时任务及时响应，而 TinyOS 任务由于严格按照次序调度，实时任务不能得到优先响应。

同时，随着任务数量的增多，任务包会发生丢失。分级调度中为每种任务分配单独的队列，且实时任务能够及时得到响应，所以实时任务包的丢失率很低。而 TinyOS 中并不能对实时任务进行区别处理，从而不能保证实时任务的安全性。实时任务丢包率和执行时间之间的关系如图 4-7 所示：

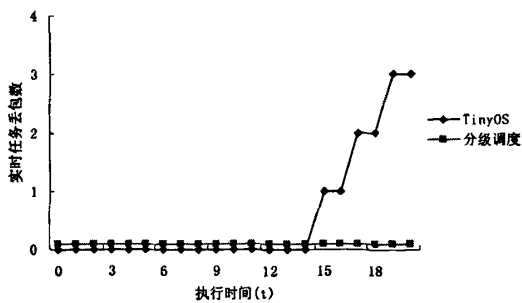


图 4-7 可调度性比较

由此可见，分级调度系统始终能满足任务集可调度性。TinyOS 由于其不可抢占性，任务集可调度性随利用率上升而下降。

4.6.2 灵活性

TinyOS, Mantis OS, SOS, Contiki 均使用固定的调度系统, 没有提供调度系统的定制机制。分级调度器的构造随传感器网络应用环境的不同可以很方便的进行变化。在实时要求低的环境下, 可以退化为 TinyOS 的完全非抢占调度队列, 所需子调度器数量为 1。在实时要求高的情况, 则构造出满足实时要求的可抢占层次调度体系, 各子调度器能够采用不同的调度策略, 根据实际应用兼顾实时性与内存空间限制的要求, 是一个灵活有效的调度体系。

4.6.3 资源消耗

TinyOS 中仅需在内存中分配一个任务队列, 而分级调度策略根据不同的应用需要配置 2-4 个任务队列, 需要更多的内存资源。在能量消耗方面, 虽然都采用了事件驱动机制, 但是 TinyOS 中任务是依次执行, 而分级调度中存在抢占, 因此会出现内容交换, 会增加能耗。因此, 相对于 TinyOS 来说, 分级调度策略会消耗更多的能量。与 MANTIS OS 等多线程调度系统来说, 分级调度的能量消耗又会比较少。

本实验基于 TOSSIM 仿真环境来比较原有 Tiny OS 调度算法和改进后分级调度算法的节点能量消耗,通过模拟无线传感网络节点运行并结合 Power-TOSSIM 来监控节点的能耗。我们每隔 5 s 对数据进行统计,图 4-8 表示的是网络中一个节点每 2 s 的能量消耗。

从图 4-8 中可以看出,采用分级调度算法所用的能量要略高于原有 Tiny OS 算法,这是因为改进的算法要进行时限的计算,系统开销相对大一些,但平均所耗能量也只比原有增加 4%左右,仍在可接受范围内。

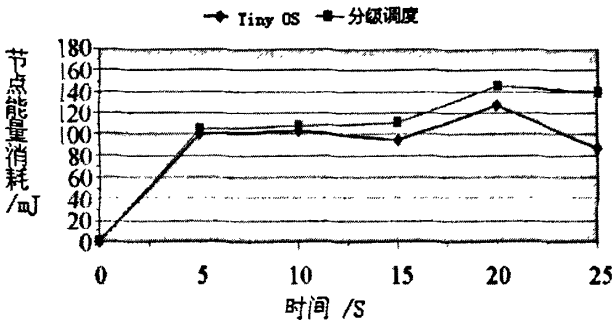


图 4-8 节点能量消耗比较

总之, 分级调度策略不能在每个方面都更加优秀, 它是在各个性能方面根据实际需求做出一个平衡的方案。

4.7 本章小结

本章在第三章分析的基础上,提出了我们的操作系统的设计目标。主要思想在于在保留 TinyOS 的基于事件驱动和组件结构的基础上,加入对于实时性的支持。结合分级模型和 MANTIS OS 的多线程实时性特点,我们提出了基于优先级的可抢占式分级调度策略,以此调度策略来平衡事件驱动和实时性,以适应不同的无线传感网络应用环境。然后,对所提出的基于优先级的可抢占式分级调度策略从实时性、灵活性和资源消耗等方面进行了性能分析和评测。分级调度策略在实时响应方面的性能优于 TinyOS,保证了重要任务的响应。同时,它具有更好的灵活性,能根据应用环境方便的进行调整。采用基于事件驱动的思想就是为了实现能量有效性,然而实时性和灵活性的提高使得分级调度系统对于系统内存的需求有所增加,在任务切换过程中会消耗更多的能量,其能量消耗介于 TinyOS 和 MANTIS OS 之间。根据实际应用需求,选择适用的系统才是最佳的方案。

第五章 结论

本文主要介绍和分析了无线传感网络操作系统的调度策略概念、主要类型和常用策略。在分析了现有的无线传感网络操作系统的调度策略的优缺点的基础上,提出了基于优先级的可抢占式分级调度的思想。

5.1 本文主要工作

本文的主要研究工作包括以下几点:

(1) 学习了无线传感网络的基本概念、体系结构以及基本特征等,了解了当前国内外有关于无线传感网络及其操作系统的研究现状。

(2) 分析当前无线传感网络中常用的操作系统的调度策略,重点分析了广泛应用的 TinyOS 操作系统的调度策略。然后针对事件驱动的和实时性操作系统的性能进行了分析比较。由此引出我们的设计思想。

(3) 针对第三章对于当前无线传感网络常见操作系统调度策略的分析,提出了基于优先级的可抢占式分级调度策略。该策略结合了事件驱动和实时性调度的思想,并引入了分级调度理论,旨在设计一种针对对于实时响应有要求的无线传感网络操作系统调度策略。它借鉴了 TinyOS 的事件驱动机制和组件模式,在此基础上引入优先级和抢占机制来满足实时任务的处理要求,并给出了部分实现算法。

(4) 针对前面提出的基于优先级的可抢占式调度策略,从实时性、灵活性和资源消耗的方面和 TinyOS 操作系统进行了比较分析。分级调度系统并不能在每个性能上都比其它系统优越,但是针对于具体的应用环境,如实时性要求较高的环境,更能满足环境的要求。它是在实际应用需求的基础上,对各个性能方面做出平衡的一种方案

5.2 进一步的工作

本文的研究过程中尚存在以下一些问题需要在以后的工作中进行进一步的研究和改进:

(1) 能量消耗问题。在引入实时性操作的同时,内容交换会随之增多,如何很好的控制能量消耗问题会直接影响到该系统是否能够应用于大范围的无线传感网络的应用环境中。如何在满足实时性要求的前提下,尽可能的减少能量消耗是当前迫切需要解决的问题。我们可以从内容交换的方式,任务的优先级别的区分等方面去考虑如何减少消耗。

(2) 我们提出的调度策略的鲁棒性等特性还需要进行验证,并进一步的改进。

(3) 构建真实环境中的无线传感网络环境来验证该系统，并发现新问题，进一步对其进行改进。

参考文献

- [1]李建中,高宏.无线传感网络的研究进展.计算机研究与发展,2008,45(1):1-15.
- [2]孙利民,李建中,陈渝,朱红松.无线传感网络[M].北京:清华大学出版社,2006.
- [3]马祖长,孙怡宁,梅涛.无线传感网络综述.通信学报[J].2004,25(4):114-124.
- [4]崔莉,鞠海玲,苗勇,李天璞,刘巍,赵泽.无线传感网络研究进展[J].计算机研究与发展,2005,42(1):163-174.
- [5]任丰原,黄海宁,林闯.无线传感网络[J].软件学报,2003,14(7):1282-1291.
- [6]盛超华,陈章龙.无线传感网络及应用[J].微型电脑应用.2005,21(6):11-14.
- [7]孙亭,杨永田,李立宏.无线传感网络技术发展现状.电子技术应用[J].2006,6:1-6.
- [8]纪阳,张平.无线传感网络的体系结构[J].中兴通讯技术.2005.11(4):32-35.
- [9]何宁,王漫,方昀,刘赐平,裴俊.面向无线传感网络应用的传感器技术综述[J].计算机应用与软件. 2007,24(9):91-94.
- [10]李凤保,李凌.无线传感网络技术综述[J].仪器仪表学报.2005,26(8):559-561.
- [11]刘敏钰,吴泳,伍卫国.无线传感网络(WSN)研究[J].微电子学与计算机.2005,22(7):58-61.
- [12]陈涛,刘景泰,郝志刚.无线传感网络研究与运用综述.总线与网络.2005,7:41
- [13]石军锋,钟先信,陈帅,邵小良.无线传感网络结构及特点分析[J].重庆大学学报(自然科学版). 2005,28(2):16-19.
- [14]赵建华.无线传感节点嵌入式操作系统的研究与应用[D].电子科技大学 2007
- [15]李凤保,李凌.无线传感网络技术综述[J].仪器仪表学报.2005,26(8):559-561.
- [16]肖健,吕爱琴,陈吉忠,朱明华.无线传感网络技术中的关键性问题[J].传感器世界.2004,7:14-18.
- [17]吴键.智能无线传感网络节点的设计和研究[D].南京航空航天大学,2006.
- [18]王漫,何宁,裴俊,冯改玲,刘海涛.面向无线传感网络应用的嵌入式操作系统综述[J].计算机应用与软件. 2007,24(6):44-48.
- [19]Shan Bhatti, James Carlson, Hui Dai, et. MANTIS OS: An Embedded Multithreaded Operating System for Wireless Micro Sensor Platforms[J]. Mobile Networks and Applications. 2005,10:563-579.
- [20]陈喜贞,王书茂,徐勇军.无线传感网络操作系统调度策略研究[EB/OL].中国科技论文在线. <http://www.paper.edu.cn>.
- [21]罗晓华.支持无线网络传感器的 γ OS 操作系统若干关键软件技术的研究和实现[D].浙江大学,2006.
- [22]李晶,王福豹,段渭军,王建刚.无线传感网络节点操作系统研究[J].计算机应用研究. 2006,8:28-30.

- [23]孙帆.无线传感网络节点操作系统分层调度模型设计与实现[D].浙江大学,2007.
- [24]王鸚鵡.基于智能传感器的实时系统任务调度分析[D].浙江大学,2006.
- [25]国成林,焦毅,吴百锋.支持时钟同步技术的无线传感网操作系统研究及内核实现[J].信息与控制. 2006,35(2):179-183.
- [26]Eric Trumpler, Richard Han. A Systematic Framework for Evolving TinyOS[J].
- [27]Jason Lester Hill. System Architecture for Wireless Sensor Networks[M]. University of California, 2003.
- [28]My Tra Thai. Energy Efficiency in Wireless Sensor Networks[D]. University of Minnesota, 2005.
- [29]Venkita Subramonian, Huang-Ming Huang, Seema Datar, Chenyang Lu. Priority Scheduling in TinyOS-A Case Study[J].2002.
- [30]彭向荣.TinyOS 无线传感网络研究[J].中国水运.2006,4(6):122-123.
- [31]孙益辉,陈凯,白英彩.嵌入式操作系统内存管理机制分析及改进[J].计算机应用与软件.2006,23(3):98-100.
- [32]周贤娟,韩树人,鄢化彪,刘生华.无线传感网络节点操作系统-TinyOS 综述[J].矿山机械.2007,35(9):112-115.
- [33] Cormac Duffy, Utz Roedig, John Herbert, Cormac J.Sreenan. Adding Preemption to TinyOS[C]. EmNets'97. 2007:88-92.
- [34]林华杰,史浩山.一种移动代理变种在 TinyOS 中的实现机制[J].计算机应用. 2007, 27(9):2212-2214.
- [35]林恺,赵海,尹震宇,毕远国.一种基于 TinyOS 的自适应双环调度策略[J].东北大学学报(自然科学版). 2007,28(7):985-988.
- [36]王万里,郑扣根,姚翔,吴朝晖.无线网络传感器及其微型操作系统的研究[J].计算机应用研究.2005,9:39-42.
- [37]李晶,王福豹,段渭军.无线传感网络中 TinyOS 的研究[J].计算机测量与控制.2006,14(6):838-840.
- [38]韩翠红,李立宏,曹晓波,赵尔沅.无线传感网络应用支撑技术研究进展介绍[J].广东通信技术.2005,5:51-55.
- [39]王晓东,戴华平,孙优贤.无线传感网络混合任务的系统级低功耗实时调度算法研究[J].传感技术学报. 2007,20(11):2487-2483.
- [40]尹震宇,赵海,徐久强,王小英.无线传感网络操作系统中抢占式任务调度策略[J].东北大学学报(自然科学版).2007,28(5):652-655.
- [41]尹震宇,赵海,林恺,刘楠,徐久强.无线传感网络操作系统调度策略[J].计算机工程.2007,33(17):77-80.

- [42]刘信新,邵明凯.无线传感网络操作系统 TinyOS 研究[J].计算机与数字工程. 2007,35(7):66-69.
- [43]王斌.无线传感网络操作系统 TinyOS 的研究.计算机与现代化.2008,1:6770.
- [44]张伟,何斌,赵霞,陈启军.开放的无线传感网络平台 OpenWSN[J].计算机研究与发展. 2008,45(1):97-103.
- [45]林华杰,史浩山.基于无线传感网络移动代理变种在 TinyOS 中的实现[J].传感技术学报. 2007, 20(10):2324-2327.
- [46]林喜源.基于 TinyOS 无线传感网络协议研究[D].武汉理工大学.2007.
- [47]丁锐.基于 TinyOS 的无线音频传感器节点系统的研究与实现[D].北京邮电大学.2007.
- [48]刘营,于宏毅.基于 TinyOS 的无线传感网络应用程序开发技术[J].传感器与微系统.2007,26(3):93-96.
- [49]林喜源.基于 TinyOS 的无线传感网络体系结构[J].单片机与嵌入式系统应用.2006,9:44-47.
- [50]章国安,徐晨,袁红林.基于 TinyOS 的无线传感网络节点机的柔性设计[J].传感器与仪器仪表. 2006,22(11):185-189.
- [51]肖建荣,钱建生,王营冠,陆晔锁.基于 TinyOS 的无线传感网络构建[J].单片机与嵌入式系统应用.2007,7: 32-35.
- [52]于继明,杨余旺,孙亚民,赵炜.基于 TinyOS 的微型无线感知网技术研究[J].计算机科学.2007,34(3):23-26.
- [53]张侃侃,刘晔,艾斯喀尔.喀迪尔,陈敬后,束秀梅,陈江波.基于 TinyOS 操作系统的无线传感网络自组网研究[J].传感技术学报.2007,20(6):1349-1352.
- [54]孙毅,王雷,杜晓通.基于 TinyOS2.x 的无线传感网络数据包传输机制的研究[J].电子器件. 2007, 30(5):1954-1958.
- [55]刘华,杨志家.基于 ARM 核处理器的 TinyOS 实现.信息与控制, 2006.
- [56]朱敬华,李建中,刘勇,高宏.传感器网络中数据驱动的睡眠调度机制[J].计算机研究与发展. 2008, 45(1):172-179.
- [57]何朝笋,石高涛,廖明宏.传感器网络随机睡眠节点调度算法研究及实现[J].计算机工程.2007,33(8):115-117.
- [58]苏威积,赵海,徐野,张昕.传感器网络结点 OS 及其传输协议的设计与实现[J].东北大学学报(自然科学版). 2007,27(6):639-641.
- [59]李丽娜,廖明宏.传感器网络及其操作系统[J].电脑学习,2005,5:7-9.
- [60]李丽娜,石高涛,廖明宏.传感器网络操作系统 TinyOS 关键技术分析[J].哈尔滨商业大学学报(自然科学版). 2005,21(6):724-728.
- [61]彭向荣.TinyOS 无线传感网络研究[J].中国水运(理论版).2006,4(6):122-123.

- [62] 庞同庆,田丰.TinyOS 体系结构及通讯机制分析[J].沈阳航空工业学院学报. 2007,24(5):35-38.
- [63] 刘奎安,郭文生,桑楠.TinyOS 任务调度机制与实时调度构件设计[J].计算机应用. 2007,27(11):2740-2742.
- [64] 陈喜贞,王书茂,徐勇军.TinyOS 内核调度机制及改进策略[J].计算机工程.2006,32(19):85-87.
- [65] 尹长青.无线自组网络若干技术的研究[D].复旦大学,2004.
- [66] 郑南.支持无线网络传感器的超微型操作系统 ZUOS 研究与应用[D].浙江大学,2004.
- [67] 张琼.无线传感网络的研究[D].西安建筑科技大学,2005.
- [68] 李晶.无线传感网络微型嵌入式操作系统的研究与应用[D].西北工业大学,2006.
- [69] 王万里.无线网络传感器及其微型操作系统的研究[D].浙江大学,2005.
- [70] 林喜源.基于 TinyOS 无线传感网络体系结构[EB/OL].中国科技论文在线.
<http://www.paper.edu.cn>.
- [71] 罗杰,林亚平.TinyOS 调度机制研究与改进[J].科学技术与工程.2007
- [72] 黄丹丹.基于 TinyOS 的无线传感节点 BSP 开发设计[D].信息工程大学,2006.
- [73] 丁锐.基于 TinyOS 的无线音频传感器节点系统的研究与实现[D].北京邮电大学,2007.
- [74] 林喜源.基于 TinyOS 无线传感网络协议研究[D].武汉理工大学,2007.
- [75] 赵建华.无线传感网络节点嵌入式操作系统的研究与应用[D].电子科技大学,2007.
- [76] 吴键.智能无线传感网络节点的设计和研究[D].南京航空航天大学,2006.
- [77] 黄光燕,李晓维.无线传感网络操作系统[J].
- [78] Chih-Chieh Han, Ram Kumar, Roy Shea, Eddie Kohler, Mani Srivastava. A Dynamic Operating System for Sensor Nodes.[J]
- [79] Operating System for Wireless Micro Sensor Platforms[J]. Mobile Networks and Applications.2005,10:563-579.
- [80] Panagiotis Galiotos. Sleep/Active Schedules as a tunable characteristic of a Wireless Sensor Network[J].2006.
- [81] Shahin Farshchi, Istvan Mody, Jack W.Judy. A TinyOS-Based Wireless Neural Interfade[C]. Proceedings of the 26th Annual International Conference of the IEEE.2004,4334-4337.
- [82] Hailun Tan. Maximizing Network Lifetime in Energy-constrained Wireless Sensor Network[C]. IWCMC'06. 2006,1091-1095.

- [83]Shahin Farshchi, Paul H.Nuyujukia, Aleksey Pesterev, Istvan Mody, Jack W.Judy. A TinyOS-Based Wireless Neural Sensing, Archiving, and Hosting System[C]. Proceedings of the 2nd International IEEE EMBS Conference on Neural Engineering.2005,5-8.
- [84]David Gay, Phil Levis, David Culler. Software Design Patterns for TinyOS[C]. LCTES'05. 2005,40-49.

作者：[翟月](#)
学位授予单位：[合肥工业大学](#)

相似文献(10条)

1. 学位论文 [谢鸣](#) [无线传感网络节点操作系统网络协议栈研究及实现](#) 2008

无线传感器网络(Wireless Sensor Network, WSN)是由集成传感、运算和无线通信能力的嵌入式节点组成的自组织动态网络。早在1999年,美国的《商业周刊》就将“网络化的微型传感器技术”评为21世纪最重要的21项技术之一。无线传感网络在森林防火,环境监测,军事部署等许多领域都有广泛的应用,是近几年来研究热点之一。随着无线传感网络节点研究的不断深入,节点操作系统的研究也取得了突破性的进展,特别是最近的五年来,相继涌现出了像TinyOS、SOS、Contiki、Mantis以及RETOS等无线传感网络节点专用的操作系统。

无线通信技术是无线传感网络操作系统的关键技术之一。一个无线传感网络节点操作系统网络协议栈的研究和设计的水平,直接关系到整个无线传感网络的通信性能,关系到无线传感网络节点能量的优化,关系到用户对无线传感网络应用程序的开发和应用。如何实现一个高效、低功耗、灵活易用的无线传感网络节点操作系统网络协议栈是无线传感网络操作系统研究中的一个极具挑战性的工作。

本文在综合考虑和分析现有主流无线传感网络操作系统网络协议栈的基础上,提出了自己的无线传感网络节点操作系统网络协议栈的分层设计原则。Senspire网络协议栈包括三个层次,分别为无线通信层、资源管理层和网络支持层。无线通信层直接与底层传感器硬件平台进行交互,提供硬件驱动同时为资源管理层提供统一的驱动接口。资源管理层位于无线通信层与网络支持层之间,使用无线通信层提供的接口,提供资源管理和并发服务。网络支持层为用户提供实现网络协议的高质量服务、邻居列表和消息缓冲池的完整接口支持。

本文最后展示了通过实验验证的Senspire网络协议栈的测试结果,包括发包率、低功耗、灵活性和易用性方面的表现,获得了良好的实验结果。

2. 期刊论文 [冯培昌](#),[周晓云](#),[陈孝伟](#),[Feng Peichang](#),[Zhou Xiaoyun](#),[CHEN Xiaowei](#) [无线传感网络探讨 - 电气自动化](#) 2005, 27 (5)

无线传感网络是由大量传感器结点通过无线通信技术自组织构成的网络。传感网络可实现数据的采集量化,处理融合和传输应用,它是信息技术的一个新的领域。本文对其关键技术,即节能优化技术、路由选择、拓扑控制、覆盖控制、操作系统进行初步探讨。

3. 学位论文 [姜琴](#) [TinySPOS: SenspireOS上的TinyOS运行环境](#) 2008

TinyOS是一个经典的无线传感网络(WSN)操作系统,在WSN发展历程中发挥了非常重要的作用。当前,许多有关无线传感网络的研究成果都是在TinyOS上研究并实现的。SenspireOS是浙江大学无线传感网络项目小组开发的一个新的WSN操作系统。在完成SenspireOS V1.0的研究和开发之后,我们面临如何在SenspireOS上继承无线传感网络的已有研究成果的问题,特别是如何继承在TinyOS上取得的有关无线传感网络的研究成果的问题。

本文的工作是设计SenspireOS上的TinyOS运行环境—TinySPOS。TinySPOS使用代理技术、中断钩子技术和虚拟时钟技术来构建TinyOS虚内核,并用TinyOS虚内核替代TinyOS内核。TinyOS虚内核使TinyOS应用程序与SenspireOS的内核相连接,使得TinyOS应用程序可以无缝移植到SenspireOS上,从而实现在SenspireOS上继承TinyOS上取得的WSN研究成果的目的。

TinySPOS具有几个特点。首先, TinySPOS采用SenspireOS的内核,而为应用程序提供完备的TinyOS的组件和接口。其次, TinySPOS可以无缝移植所有的TinyOS应用程序。第三, TinySPOS可以作为SenspireOS上的nesC开发环境使用。第四, TinySPOS可以实现在SenspireOS上扩展nesC语言功能的目的。

4. 学位论文 [赵春颖](#) [一个嵌入式实时内核SmartOS的设计及其在无线系统中的应用](#) 2005

随着半导体技术、计算机技术、通讯技术等信息技术的发展,作为21世纪的计算模式,普适计算将对人们的生活产生深远的影响。利用微型嵌入式计算设备采集处理各种相关信息,然后利用各种通信技术手段同其它异构的设备实现各种互联事务处理和数据交互,向人们提供需要的信息,这就形成了无线传感网络。其中对于无线传感节点中嵌入式操作系统的开发成为其关键的技术。本文力图在这个领域进行探索。

本文在研究几种常用的嵌入式操作系统的基础上,综合各个内核的基本特点并结合无线传感网络的应用需要,介绍了一种面向无线传感网络节点平台的嵌入式操作系统SmartOS的设计过程。

首先,分析了无线传感网络的特点和嵌入式实时系统的概念。在这个操作系统的设计中,围绕无线传感网络的特点和如何提高实时系统的性能,主要讨论了系统的进程管理、任务管理、中断管理和内存管理等组成内核不可缺少的部分的设计方法。本文结合了基于优先级的抢占式调度方法和时间片轮转法,辅助采用先进先出的队列,设计了进程的调度策略,在保证系统的实时性的同时赋予了进程比较公平的运行机会。考虑到一般嵌入式系统内存有限,作者尝试将任务堆栈核和中断嵌套堆栈进行分离,从而有效的节省了系统的内存开销。

然后,在此操作系统的基础上,构建了一个无线应用系统,作为无线传感网络节点平台,并进行了相关的系统测试实验,验证了系统的正确性。该系统为无线传感网络提供一个真实的节点,并为相关的网络协议的研究、以及探索新的嵌入式系统应用提供了一个理想的平台。

最后,本文总结了前面的工作,展望了微型嵌入式操作系统在无线传感节点上应用的未来发展情况,指出了今后的研究方向。

5. 期刊论文 [吴江川](#),[孙雷](#),[王鸿鹏](#),[周璐](#),[刘景泰](#),[WU Jiang-chuan](#),[SUN Lei](#),[WANG Hong-peng](#),[ZHOU Lu](#),[LIU Jing-tai](#) [基于Tmote-Sky的温湿度及光强检测系统 - 自动化与仪表](#)2008, 23 (7)

提出基于Tmote-Sky无线传感网络节点构建布网快捷、维护简单的温湿度和光强检测系统的设计方案。介绍了系统整体框架和软件体系的设计实现。设计实验验证了系统搭建的便捷性、工作的稳定性以及结果的正确性,并提出系统扩展应用的方法。

6. 学位论文 [吴福青](#) [基于无线传感器网络操作系统底层平台及内存管理的研究](#) 2009

随着集成智能传感器技术、微机电系统技术和网络通讯技术的快速发展,一种全新的信息获取处理技术,无线传感器网络应运而生,它的诞生就在国际上备受关注,并且它被认为是21世纪最重要的技术之一。无线传感网络操作系统技术是无线传感网络的支撑技术之一,是无线传感网络的基本环境和平台,是开发应用程序的基础。无线传感网络操作系统底层平台又是操作系统的技术关键,它是操作系统及其应用软件同操作系统的硬件的重要连接,针对无线传感网络操作系统底层平台的研究又有着重要的意义;其次,无线传感网络操作系统的内存管理是保护系统资源合理分配,保证系统的可靠性有着重要的意义。

目前,对WSN的研究主要集中在通信协议、能耗管理、定位算法以及体系结构设计上,它们占无线传感网络研究的绝大部分,而针对无线传感网络操作系统的研究却相对较少,尤其是对其底层平台的研究就更少了,所以针对无线传感网络操作系统底层平台的研究就有了更广阔的空间;其次,在内存管理方面,嵌入式操作系统不像在传统的Windows和Linux桌面操作系统,有自己完善的内存管理机制,它的有限资源和低成本,对其内存管理的研究又有了更多难点和挑战。

本论文研究内容源于本实验室承担的国家自然科学基金重点项目“月球探测系统的建模、传感、导航和控制基础理论及关键技术研究(60535010)”。本论文主要是针对无线传感网络操作系统的底层平台和内存管理进行研究。首先介绍了无线传感网络及无线传感网络操作系统的发展现状和基本理论,文中着重研究无线传感网络底层平台,对其无线通讯模块、无线通讯的过程以及DMA通道的配置进行研究,并实现了数据传输的两大硬件通讯模块的驱动程序。其次,操作系统内存管理模块结合μC/OS II内存管理的实现方法,提出新的数据结构,并根据新的数据结构,实现更好的内存分配和回收机制。在文章的最后,实现了人机交互的控制平台,并针对操作系统实现了系统测试,检验操作系统实现的功能。

7. 学位论文 [陈杰](#) [无线传感网络节点资源管理的研究与实现](#) 2007

无线传感器网络(Wireless SensorNetwork, WSN)是由大量传感器节点通过无线自组织的方式构成的网络。它结合了计算,通信,传感器三项技术,在森林防火,环境检测,以及军工等各个领域都有广泛的应用,是当前的研究热点之一。

