支持多应用任务的WSNs中间件的设计与实现

胡 智 闻英友 赵 宏
(东北大学信息科学与工程学院 沈阳 110819) (东北大学计算机软件国家工程研究中心 沈阳 110819)

摘 要 无线传感器网络作为物联网的基础设施,需要同时运行多种应用任务来满足不同用户的复杂需求。现有的无线传感器网络中间件主要是为某一类应用领域而设计和开发的,其上运行的是单一应用任务。设计了一种支持多应用任务的无线传感器网络中间件架构,并根据此架构设计与实现了整个系统与各个功能组件,整个中间件可以根据不同需求承载多种应用任务,并且通过模拟网络的实验验证了该中间件系统具有较好的可行性和实用性。

关键词 无线传感器网络,中间件,多应用任务
中图法分类号 TP393 **文献标识码** A

Design and Implementation of WSNs Middleware Supporting Multiple Application Task

HU Zhi WEN Ying-you ZHAO Hong
(School of Information Science and Engineering, Northeastern University, Shenyang 110819, China)
(Research Center of National Computer Software Engineering, Northeastern University, Shenyang 110819, China)

Abstract As the infrastructure in Internet of Things, wireless sensor networks must support, multiple application tasks to satisfy the requirement from diversified users. The current solution to wireless sensor networks middleware focuses mainly on a kind of applied domain. The architecture of middleware supporting multiple application tasks was proposed in this paper. Based on the architecture, the entire system and all relative components were designed and implemented. According to the diverse requirement, the middleware may support the different application tasks. Experiments based on simulative network prove that the middleware has good feasibility and practicability.

Keywords Wireless sensor networks, Middleware, Multiple application tasks

1 概述

最近几年无线传感器网络(WSNs)由于应用前景广泛,已成为社会各界研究的热点。无线传感器网络综合应用传感器技术、嵌入式技术、网络与无线通信技术,通过微型传感器感知和采集各种环境数据,利用无线方式形成自组织的网络,将感知到的数据传输到用户监控中心[1]。WSNs与传统网络相比有其自身的特点,如资源受限、通信带宽低、节点异构和网络拓扑易变等,使得WSNs的应用开发极为困难。中间件技术将底层操作系统接口封装,为应用程序开发提供平台组件。WSMs中间件是一种为WSNs的应用程序开发提供服务的中间件,这种中间件建立在WSNs节点的操作系统和相关固件系统之上[2]。

由于WSNs的特点,传统的中间件技术无法适用,研究人员根据无线传感器网络的特点和需求提出了许多设计方法。文献[3]提出了一种基于虚拟机的中间件Maté,它通过虚拟机屏蔽硬件资源和操作系统,使用字节代码包进行网络应用开发。TinyDB[4]是一种基于数据库的WSNs中间件,将整个网络看成一个分布式数据库,用户使用类似SQL的查询命令获取所需的数据,查询分发到每个节点,再收集数据传给用户。Agilla[5]是一种基于移动代理的WSNs中间件,由移动代理实现应用程序的行为,对WSNs进行编程,一般与元组空间相结合。MiLAN[6]是一种支持QoS的自适应WSNs中间件,获取基于状态变化的QoS需求,根据这些信息控制网络和节点,平衡应用资源,延长生存期,并提出了系统的框架。

尽管这些中间件实现了统一编程的目的,但都是针对单一应用或简单WSNs应用开发的,且资源和可扩展性都受限制。随着WSNs应用技术的不断发展,系统的复杂性也不断增加,呈现出一个WSNs上承载多种应用任务的趋势,简单的WSNs应用已不能满足综合应用部署的需求。因此,本文设计并实现了一种可支持多应用任务同时运行的中间件平台,动态承载各种应用任务。

2 中间件系统架构设计

2.1 系统模型

WSNs系统通常包括硬件、操作系统和应用程序。硬件

到稿日期:2011-05-30 返修日期:2011-09-29 本文受国家自然科学基金青年基金(60803131),沈阳市科技计划项目(1091176-1-00)资助。
胡 智(1978-),男,博士生,主要研究方向为传感器网络中间件、网络与信息安全,E-mail:huzhi20060302@gmail.com;闻英友(1974-),男,博士后,副教授,主要研究方向为下一代网络、移动通信技术和网络安全;赵 宏(1954-),男,教授,博士生导师,主要研究方向为下一代网络、网络与信息安全和网络管理。

包括感知模块、MCU 模块和通信模块。操作系统和应用程序是软件，而中间件是运行在操作系统和应用程序之间的一种系统软件。系统模型如图 1 所示。

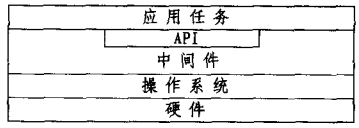

图 1　系统模型

WSNs 中间件是由许多组件构成的，它们为应用程序提供相应的服务功能。可以一个组件提供一个服务，也可以多个组件合起来提供一个服务，或者一个组件提供多个服务。最后通过统一的应用程序开发接口(API)提供给用户。

2.2　系统架构设计

在整个 WSNs 系统中，WSNs 节点主要是为基站或网关实现数据的感知和采集功能，并且在整个 WSNs 系统运行中，可以为用户消息提供相应的支持。因此，可以在 WSNs 节点上建立相应的功能组件，协调完成中间件平台的功能，为应用任务的执行提供服务。根据 WSNs 中间件平台的这些特点，将中间件分成两个子系统：控制管理子系统和应用运行子系统。整体系统、各子系统以及子系统内部各组件的架构关系如图 2 所示。

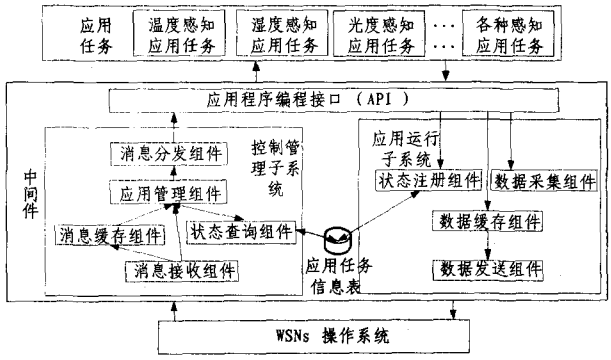

图 2　WSNs 中间件的架构设计

WSNs 中间件各组件的功能如下：

(1)消息接收组件。接收来自基站或网关的用户消息，这些消息包括两类：对节点进行控制的节点级消息，对应用任务进行控制的应用任务级消息。对这些消息进行优先级的划分，分别是高优先级和普通级。将高优先级的消息直接发送给应用管理组件，而将普通级的消息送入缓存排队，等待逐个处理。

(2)消息缓存组件。缓存普通级的消息，节点只有一个 MCU，同一时间仅能处理一个消息，将陆续到达的消息进行缓存，以保障消息的可靠处理，按照先来先服务的原则，等待应用管理组件分发消息给应用任务，或进行节点控制。

(3)应用管理组件。负责对接收到的消息进行处理与分发。收到高优先级的消息时，中断当前的处理工作，立即响应高优先级消息，其消息的分发方式是对应用任务产生中断处理请求。通过对状态查询组件进行操作，可获得整个节点的应用任务信息，从而对节点和应用任务进行管理操作。

(4)消息分发组件。通过标准的应用程序编程接口，将消息转交给应用任务。在应用任务的开发过程中，必须事先引入接收中间件消息的接口。参数格式在中间件开发手册中都已预先定义，使得消息可以准确接收。

(5)应用程序编程接口(API)。中间件平台必须为应用任务提供统一、易用和标准的开发接口，以使应用任务开发更为便利，提高开发效率。

(6)状态查询组件。为应用管理组件提供当前节点上应用任务的状态信息，通过查询应用任务信息表，获得应用任务的状态。每一个应用任务的启动、关闭和运行都会在应用任务信息表中保留状态信息。

(7)应用任务信息表。维护节点上应用任务的启动、关闭和运行的相关状态信息，记录应用任务启动时的注册信息、关闭时的取消信息和运行时的采样频率等信息。

(8)状态注册组件。为应用任务提供标准的开发 API。在应用任务启动、关闭和运行时，都可以通过相应的 API 在应用任务信息表中修改和维护自身的信息。

(9)数据采集组件。负责屏蔽底层传感器的感知数据的读操作。传感器的类型、产品和型号较多，在这里主要以感知数据的类型进行分类封装，如温度、湿度、光度等，将这些感知数据的读操作封装成统一接口，供应用任务开发使用。

(10)数据缓存组件。在数据发送过程中，缓存应用任务感知完成后等待发送的数据，节点只有一个通信模块，同一时间仅能处理一个感知数据的传送，将产生的感知数据进行缓存，以保障数据通过网络进行可靠的传输。

(11)数据发送组件。负责屏蔽底层的 WSNs 传输协议，将不同网络路由协议的数据发送接口进行封装。WSNs 存在许多路由通信协议，且使用方式和数据发送接口等均不同。这些在这里都进行统一封装和管理，以便于数据根据需求进行发送。

3　中间件系统实现

3.1　系统整体实现

WSNs 中间件的内部组件与应用任务协作完成工作。首先，接收并缓存基站或网关发来的消息，应用管理组件从缓存中取出消息，根据消息内容进行节点级和应用任务级的分类处理，如果是节点消息，根据消息的需求进行节点的动态配置、应用任务调整和控制管理等操作；如果是应用任务级，则分发给相应的应用任务自行操作。然后，各应用任务组件根据消息的要求，实现相应的采集功能，并通过发送组件将感知到的数据发送给基站或网关。WSNs 中间件的工作流程如图 3 所示。

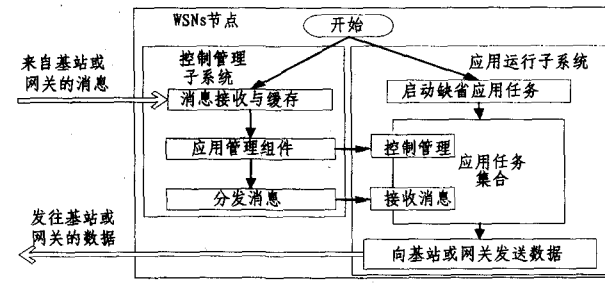

图 3　WSNs 中间件的工作流程

节点开机时，首先初始化和启动一些缺省的应用任务集，如温度感知应用任务等。当没有基站或网关发来消息时，根

据缺省的配置，采集相关的数据，然后发送到网关，或者将节点设置成休眠等待状态，不进行任务工作。当收到来自基站或网关消息时，按照消息进行相关功能调整，打开、关闭或重编程应用任务，根据要求进行不同的操作。

3.2 各子系统实现

3.2.1 控制管理子系统

控制管理子系统的主要功能是接收和响应来自基站或网关的消息。WSNs 系统需要支持不同用户的多种需要，节点根据不同需求调整功能。

其设计思想是：由消息接收组件接收来自基站或网关的消息，并根据处理时效，将其分为高优先级和普通级消息。普通级消息存入消息缓存，而高优先级需立即处理。缓存的消息由应用管理组件根据先到先服务原则分发给相应任务处理；而对高优先级消息，则立刻根据消息需求进行相应操作。如在接收的消息中，有一种消息是融合消息，具有高优先级，需要及时处理，因此不进入消息缓冲组件，直接分发给应用任务组件。

系统启动后，首先初始化消息接收组件和消息缓存组件，然后等待来自基站或网关的消息。在控制管理子系统中，根据消息处理对象，又将消息分为两类：节点级和应用任务级消息。节点级的消息主要是根据节点运行情况决定哪些应用任务打开和关闭，应用任务级的消息主要根据用户需求进行自定义。图 4 显示了整个控制管理子系统的工作流程。

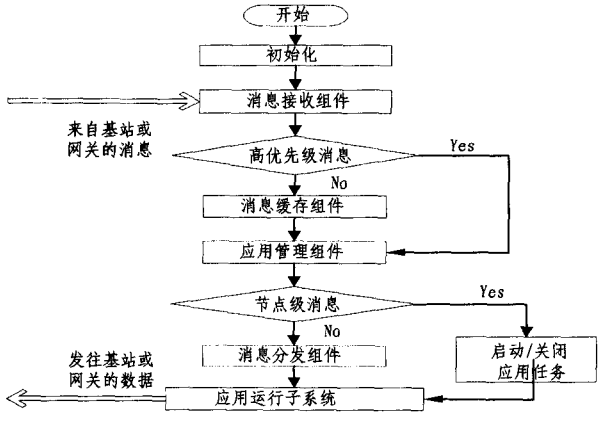

图 4 控制管理子系统的工作流程

首先判断消息的时效级别，将普通级消息存储在消息缓冲中，将高优先级消息直接发送到应用管理组件，再决定是节点控制处理，还是分发给相应的应用任务处理。应用管理组件从消息缓冲中取出普通级消息，如果是节点级的消息，如打开和关闭某一个或某几个应用任务，则调用相应功能实现打开与关闭的操作；如果是某个应用任务级的指令，则将其分发给相应的应用任务，待处理后将产生的感知数据发往基站或网关。

3.2.2 应用运行子系统

应用运行子系统的主要功能是承载执行数据感知功能的应用任务。由于 WSNs 应用的多样化，并且只有一个通信单元，无法同时发送到来的多份感知数据，因此必须进行缓存再发送。应用任务是并发运行的，对通信单元产生资源竞争，所以消息缓存以队列的方式工作，将产生的数据进行排队，然后数据发送组件按照先来先服务原则，通过通信单元发送出去。

应用任务有许多种，如温度、湿度、光度等感知应用任务，它们可以组合使用，也可以将几个感知应用任务融合成一个应用任务。发送组件屏蔽掉底层的硬件平台及使用的网络协议，应用任务的开发可以不管底层硬件使用的是哪种硬件节点，也不管使用的网络协议。图 5 显示了应用运行子系统的工作流程。

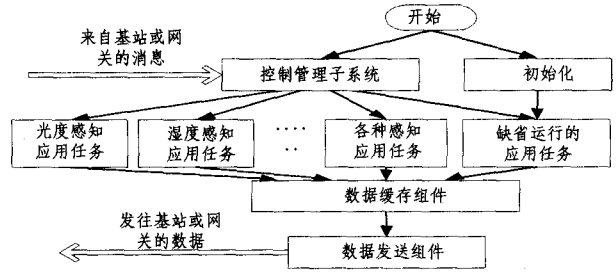

图 5 应用运行子系统的工作流程

系统启动后，首先初始化缺省的采集应用任务，如温度感知应用任务等，应用任务采集感知数据，然后通过路由协议发送到基站或者网关。控制管理子系统可以根据来自基站或网关的用户消息调整某些应用任务，打开、关闭和重编程某些应用任务。所有的应用任务将感知到的数据存入缓存，然后再发送到远程基站或网关。

4 系统验证与分析

基于主流的 WSNs 操作系统 TinyOS 实现中间件，硬件平台选用 MICAz [7]。在东软研究院，建立 WSNs 实验运行环境，用 1 个节点作基站，10 个节点自组织形成一个网络。图 6 是 WSNs 中间件的验证环境。每一个节点上的应用任务均不相同，有感知温度的，也有感知光度的等，它们形成一个复杂的 WSNs 应用系统。

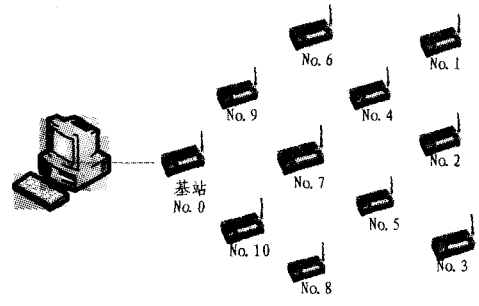

图 6 WSNs 中间件验证环境

在 10 个 WSNs 节点上部署不同的应用任务，进行综合的数据采集。在采集的数据类型中，不仅包含物理环境的感知数据，通过长时间的开发经验积累，项目组还开发了一些与网络状态相关的应用任务，如邻居发现、动态路由信息检测等，这些应用任务更加丰富了 WSNs 中间件的应用范围。

表 1 中的数据是 30min 内的平均值，并且进行了多次实验。采用的传感器统一集成在一个传感板 MTS300[8] 上面，主要有温度、光度和微麦克风传感器。其工作方式是当温度变化时，电阻值也随之变化，通过相应公式推导出温度，如节点 1 的热敏电阻值 0x01C6，根据传感器手册上的公式计算，得到温度是 20.27℃；而节点 2，4，6 感知到的温度分别是 20.27℃，20.19℃和 20.11℃。节点 3，4，8 和 10 的光度也可以相应计算出来。对于麦克风的感知值可以与射频数据包到达时间结合使用，测量相关节点的距离。在验证过程中，节点

4,5 和 7 上,也部署了邻居发现应用任务,可以和其它应用任务结合感知融合数据,然后一起发送到基站或网关。在节点 1,2,5 和 8 上,部署动态路由应用任务,记录感知数据包在网络中传递的路由路径,采用了消息的捎带技术。可以看出节点 1 和 2 的父节点都是节点 4,而节点 7 又是节点 4 和 5 的父节点,最后都通过节点 9 到达基站节点 0,而节点 8 是通过父节点 10 到达基站节点 0。因此,可看出整个网络由两棵树组成。

表 1 应用任务的感知数据

节点号	热敏电阻值	光敏电阻值	麦克风值	邻居节点集	动态路由路径
1	0x01C6	未部署	未部署	未部署	1→4→7→9→0
2	0x01C6	未部署	未部署	未部署	2→4→7→9→0
3	未部署	0x037D	未部署	未部署	未部署
4	0x01C5	未部署	未部署	(6,1,5,2,7)	未部署
5	未部署	0x0369	未部署	(4,3,2,8,7)	5→7→9→0
6	0x01C4	未部署	未部署	未部署	未部署
7	未部署	未部署	未部署	(4,5,9,10)	未部署
8	未部署	0x037A	未部署	未部署	8→10→0
9	未部署	未部署	0x01D8	未部署	未部署
10	未部署	0x0373	0x01CA	未部署	未部署

验证的测试结果表明,各种应用任务都可以并行运行,并且可以根据用户的消息随时调整。开发人员也可以根据 WSNs 中间件提供的接口开发出多种不同的应用任务。在整个的验证测试过程中,WSNs 中间件运行稳定。

结束语 本文介绍了 WSNs 的发展趋势,系统更加复杂,承载的应用也更加多样,简单的 WSNs 中间件很难适用于未来多种 WSNs 应用的需求。提出了一种支持多应用任务的 WSNs 中间件,并详细地描述了中间件的整体设计、组件划分和实现过程。通过实物节点的组网模拟实验,成功地验证了整个中间件系统和所承载应用任务能够稳定地工作。本中间件的设计与实现为进一步研究 WSNs 理论与技术奠定了平台基础。

参 考 文 献

[1] Akyildiz I F, Su W, Sankarasubramaniam Y, et al. Wireless sensor networks: a survey[J]. Computer Networks, 2002, 38(4): 393-422

[2] Wang M M, Cal J N, Li J, et al. Middleware for wireless sensor network: A survey[J]. Journal of Computer Science and Technology, 2008, 23(3): 305-326

[3] Levis P, Culler D. A tiny virtual machine for sensor networks [C]// Proc of the 10th Int'l Conf on Architectural Support for Programming Languages and Operating Systems (ASPLOSX). New York: ACM Press, 2002: 85-95

[4] Madden S R, Franklin M J, Hellerstein J M. TinyDB: An acquisitional query processing system for sensor networks[J]. ACM Trans on Database Systems, 2005, 30(1): 122-173

[5] Fok C, Roman G, Lu C. Mobile agent middleware for sensor networks: An application case study[C]// Proc. the 4th Int. Conf. Information Processing in Sensor Networks (IPSN). Los Angeles, California, USA, Apr. 2005: 382-387

[6] Heinzelman W B, Murphy A L, Carvalho H S, et al. Middleware to support sensor network application [J]. IEEE Networks, 2004, 18(1): 6-14

[7] Crossbow MICAz Mote Specification [EB/OL]. http://www.cro-ssbow.com

[8] Crossbow sensor platform[EB/OL]. http://www.crossbow.com

(上接第 31 页)

[14] Li Z, Lu S, Myagmar S, et al. CP-Miner: Finding Copy-Paste and Related Bugs in Large-scale Software Code[J]. IEEE Trans. on Software Engineering, 2006, 32(3): 176-192

[15] Falke R, Frenzel P, Koschke R. Empirical Evaluation of Clone Detection using Syntax Suffix Trees[J]. Empirical Software Engineering, 2008, 13(6): 601-643

[16] Evans W S, Fraser C W, Ma F. Clone Detection via Structure Abstraction[J]. Software Quality Journal, 2009, 17(4): 309-330

[17] Duala-Ekoko E, Robillard M. Clone Region Descriptors: Representing and Tracking Duplication in Source Code [J]. ACM Trans. on Software Engineering and Methodology, 2010, 20(1): 1-31

[18] Roy C K, Cordy J R. Near-Miss Function Clones in Open Source Software: An Empirical Study[J]. Journal of Software Maintenance and Evolution: Research and Practice, 2010, 22(3): 165-189

[19] Hummel B, Juergens E, Heinemann L, et al. Index-based Code Clone Detection: Incremental, Distributed, Scalable[C]// Proc. of the 26th IEEE Int'l Conf. on Software Maintenance. 2010: 1-9

[20] Liu H, Ma Z, Zhang L, et al. Detecting Duplications in Sequence Diagrams Based on Suffix Trees[C]// Proc. of the 13th Asia Pacific Software Engineering Conf.. 2006: 269-276

[21] Pham N H, Nguyen H A, Nguyen T T, et al. Complete and Accurate Clone Detection in Graph-based Models[C]// Proc. of the 31th Int'l Conf. on Software Engineering. 2009: 276-286

[22] Gold N, Krinke J, Harman M, et al. Issues in Clone Classification for Dataflow Languages[C]// Proc. of the 4th Int'l Workshop on Software Clones. 2010: 83-84

[23] Störrle H. Towards Clone Detection in UML Domain Models [C]// Proc. of the 4th Euro. Conf. on Software Architecture. 2010: 285-293

[24] Baxter I, Yahin A, Moura L, et al. Clone Detection Using Abstract Syntax Trees[C]// Proc. of the 14th IEEE Int'l Conf. on Software Maintenance. 1998: 368-377

[25] Nguyen H A, Nguyen T T, Pham N H, et al. Accurate and Efficient Structural Characteristic Feature Extraction for Clone Detection[C]// Proc. of the 12th Int'l Conf. on Fundamental Approaches to Software Engineering. 2009: 440-455

[26] Kuramochi M, Karypis G. Finding Frequent Patterns in a Large Sparse Graph[J]. Data Mining and Knowledge Discovery, 2005, 11(3): 243-271

[27] Bron C, Kerbosch J. Algorithm 457: Finding All Cliques of an Undirected Graph[J]. Communications of the ACM, 1973, 16 (9): 575-577

[28] Andoni A, Indyk P. Near-optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions[J]. Communications of the ACM, 2008, 51(1): 117-122

[29] Zou Z, Li J, Gao H, et al. Mining Frequent Subgraph Patterns from Uncertain Graph Data[J]. IEEE Trans. on Knowledge and Data Engineering, 2010, 22(9): 1203-1218