第5章 传感器网络的应用开发基础

俗话说："万丈高楼平地起，一力承担靠地基"。传感器网络的应用开发基础技术就是它的成功应用和优秀方案设计的"地基"。传感器网络的应用开发基础技术是传感器网络完成应用功能的关键，这里主要介绍它的仿真平台、工程测试床、网络结点的硬件开发、操作系统和软件开发等内容。

5.1 仿真平台和工程测试床

5.1.1 传感器网络的仿真技术概述

1. 网络研究与设计方法

通常计算机网络的研究与设计方法包括分析方法、实验方法和模拟方法。

分析方法是对所研究对象和所依存的网络系统进行初步分析，根据一定的限定条件和合理假设，对研究对象和系统进行描述，抽象出研究对象的数学分析模型。这种方法是通过数学推理证明，或与现实实例对照，或与模拟的结果进行比较等手段，验证模型的有效性和精确性，对模型进行校验修正，最后利用数学分析模型对问题进行解答。其优点在于灵活性好，不受硬件或软件性能等物质资源的限制，但模型的有效性和精确性受假设条件的限制很大。当一个网络系统很复杂时，无法采用一些限制性假设来对系统进行详细描述。这种方法可以适用于网络结点协议理论研究和简单的网络行为分析。

实验方法的主要内容是建立测试床和实验室。它对所研究的对象和所依存的网络系统进行初步分析，设计出研究所需要的合理硬件和软件配置环境，建立有特定特性的实际网络，在现实的网络上实现对网络协议、网络行为和网络性能的研究。这种方法具有针对性，可以获得更真实的数据，不会丢失重要的详细资料。缺点是成本很高，重新配置或共享资源难，运用起来不灵活，只适用于小规模的网络性能评估，不能实现网络中的多种通信流量和拓扑的融合。

模拟方法主要是应用网络模拟软件来仿真网络系统的运行效果。它对所研究的对象和所依存的网络系统进行初步分析，自己开发或选用一个网络模拟工具，设计一个实际的或理论的网络系统模拟模型，在计算机上运行这个模型，并分析运行的输出结果。模拟方法比较灵活，可以根据需要设计所需的网络模型，以相对少的时间和费用来了解网络在不同条件下的各种特性，获取网络研究的丰富有效的数据。缺点是受软件和硬件资

源的限制，无法同时展现现实网络的全部特性。模拟方法适用于网络协议研究、网络性能研究和各种网络设计。

2. 网络仿真的应用意义

近年来随着网络技术的迅速发展，人们一方面要为未来网络的发展考虑新的协议和算法等基础技术，另一方面要研究如何对现有网络进行整合、规划设计，使其达到最高性能。最初网络规划和设计采用经验、数学建模分析和物理测试试验的方法。在网络发展初期，网络规模较小，网络拓扑结构比较简单，网络流量不大，通过数学建模分析和物理测试，结合网络设计者的个人经验，基本上能够满足网络的设计要求。

随着网络规模的逐渐扩大，网络结构的日益复杂，网络流量也迅速增长，网络规划和设计者面临着严重的挑战，对大型复杂网络的数学建模分析往往显得非常困难，也几乎不可能开展与完成网络规模相近的物理试验测试，网络仿真技术应运而生。

网络仿真技术是一种通过建立网络设备、链路和协议模型，并模拟网络流量的传输，从而获取网络设计或优化所需要的网络性能数据的仿真技术。从应用的角度来看，网络仿真技术具有以下特点：

（1）全新的模拟实验机理，使得这项技术具有在高度复杂的网络环境下得到高可信度结果的特点。网络仿真的预测功能是其他任何方法都无法比拟的。

（2）使用范围广，既可以用于现有网络的优化和扩容，也可以用于新网络的设计，而且特别适用于大中型规模网络的设计和优化。

（3）初期应用成本不高，而且建好的网络模型可以延续使用，后期投资还会不断下降。

计算机网络的仿真工作主要包括研究开发网络建模和模拟工具、使用这些工具研究网络的动态行为。无论对网络模拟进行哪一方面的研究，都要经过网络仿真的一般过程：模型建立和配置、仿真实现、结果分析。

网络仿真通过对网络设备、通信链路、网络流量等进行建模，模拟网络数据在网络中的传输、交换等过程，并通过统计分析获得网络各项性能指标的估计，使设计者较好地评价网络性能，并做必要的修改完善，优化网络运行的性能。

网络仿真技术一方面能够通过快速建立网络模型，方便地修改网络模型参数，通过仿真可以为网络设计规划提供可靠的依据；另一方面还能通过对多个设计方案（网络结构、路由设计、网络配置等）分别建模仿真，获得相应的网络性能估计，为设计方案的可行性验证和多个方案的比较选择提供可靠依据。

通常网络仿真的模拟软件体系结构如图 5.1 所示，下面分别给予简略介绍。

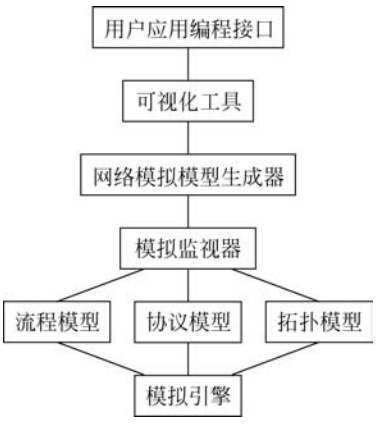

图 5.1　网络仿真的软件体系结构

用户应用编程接口负责提供各层用户编程接口，以便用户增加新的模型、新的工具等。

可视化工具包括：

（1）网络模拟动态显示工具。动态显示网络模拟的全过程，按需求对不同的采集数据进行选择性的显示，网络性能参数的统计和分析图形显示。

(2) 分析统计工具。对网络模拟数据进行统计分析,以图表、图形显示出结果。

网络模拟模型生成器负责通过图形化的方法生成网络模拟模型,利用加载对象的方法对网络进行不同的配置,它包括网络拓扑模拟模型的建立(网络结点、链接和属性)、选择流量模型、网络动态运行行为(包括结点和链接故障)。

模拟监视器的任务包括如下:

(1) 跟踪和监测网络模拟的全过程,采集模拟网络的状态参数和结果数据,包括网络的各种性能参数(带宽、延迟、延迟抖动、丢包率、拥塞状况等),形成统计数据。

(2) 提供配置各种监测粒度和范围的参数接口,提供跟踪网络行为和获取网络模拟过程中的各种网络性能参数的功能函数。

(3) 设置基于阈值的报警,设置探测例程。

网络仿真工具中的模型包括如下三种。

(1) 流量模型。它负责提供产生各种类型流量的模型,在模拟过程中根据用户的配置(速率、包的大小和分布等)加载到网络模拟模型中,产生模拟网络中的动态数据流。

(2) 协议模型。这种模型负责模拟网络协议的操作过程,对模拟网络中的动态数据流进行相应的操作,例如链路协议、路由协议等。

(3) 拓扑模型。这种模型提供网络拓扑的基本组件,包括各类结点、各种链路、各种拓扑原型等。

模拟引擎是模拟器的核心部分,包括线程调度、处理器分配、事件队列、同步设置的功能等,根据设定的模拟模型进行调度、控制整个模拟过程的执行。

3. 传感器网络仿真的特点

通常的无线传感器网络属于大规模网络,物理试验测试技术难以实行。网络仿真成为目前无线传感器网络系统研究、开发和设计的重要手段之一。

根据无线传感器网络不同于传统无线网络的特点,在它实现仿真时一般需要处理好如下问题。

① 分布性。无线传感器网络属于一种分布式的网络系统,每个结点一般只处理自己周围的局部信息。

② 动态性。正如前面提到的,在实际应用中无线传感器网络的整个系统应该是处于较为频繁的动态变化过程,网络模型是否能够较好地反映出这种动态变化,会影响仿真结果的可靠性。

③ 综合性。与传统无线网络相比,无线传感器网络集成了传感、通信和处理功能,这就要求网络仿真具有相应的综合性。

传感器网络的仿真包括仿真体系结构的设计和网络系统模型的设计。体系结构是对实际目标和物理环境中反映网络各因素及其相互联系进行抽象所得的结果。网络系统模型设计是实现网络仿真的基础。

例如,传感器网络仿真模型的设计可以涉及结点能耗模型设计、网络流量模型设计和无线信道模型设计,具体介绍如下:

① 结点能耗模型。目前人们对结点能耗模型的研究主要集中在无线电能耗模型、CPU能耗模型和电池模型,其中最主要的是无线电能耗模型。事实上传感器结点的能耗模型还

与结点分布密度、网络流量分布等诸多因素有关。

② 网络流量模型。传感器网络是面向应用的监控系统，在不同的应用背景和物理环境下，网络流量是不一样的。如果在被监控事件出现的地点附近部署传感器结点比较密集，网络将会产生瞬时的爆发流量；而在某些野外环境监控任务中，传感器结点采集数据比较固定，相应地网络流量就要稳定得多。另外，采用不同的网络协议和信息处理方法也会影响网络的整体流量。

通常我们可以把网络流量分为固定比特和可变比特两部分，分别对应稳定流量和爆发流量。在网络流量的模型分析中，人们大都将网络中数据包的到达假设为泊松过程。虽然理论上泊松过程对网络传输的性能评价具有较好的效果，但实际中未必都真正符合这一概率统计规律。

③ 无线信道模型。传感器结点之间需要通过无线信道进行通信，传统的无线网络对无线信道的传播特性和模型的建立已有较为成熟的成果。不过由于无线信道本身的不稳定性，影响因素复杂多变，而且传感器网络结点分布较为密集，使得无线信道模型的建立过程复杂。因此无线信道模型的建立是无线传感器网络研究的主要内容之一。

5.1.2 常用网络仿真软件平台

1. TOSSIM

1）简介

TinyOS 是为传感器网络结点而设计的一个事件驱动的操作系统，由加州大学伯克利分校开发，采用 nesC 编程语言。它主要应用于无线传感器网络领域，采用基于一种组件的架构方式，能够快速实现各种应用。

TOSSIM 是 TinyOS 自带的仿真工具，可以同时模拟传感器网络的多个结点运行同一个程序，提供运行时的调试和配置功能。它可以实时监测网络状况，并向网络注入调试信息，还可以与网络进行交互式的操作[52]。

由于 TOSSIM 仿真程序直接编译来自实际运行于硬件环境的代码，因而可以用来调试最后实际真正运行的程序代码。

TOSSIM 仿真软件工具的体系结构内容如下：

① 编译器支持。TOSSIM 改进了 nesC 编译器，通过选择不同的选项，用户可以把硬件结点上的代码编译成仿真程序。

② 执行模式。TOSSIM 的核心是仿真事件队列。与 TinyOS 不同的是，硬件中断被模拟成仿真事件插入队列，仿真事件调用中断处理程序，中断处理程序又调用 TinyOS 的命令或触发 TinyOS 的事件。这些 TinyOS 的事件和命令处理程序可以生成新的任务，并将新的仿真事件插入队列，重复此过程直到仿真过程结束。

③ 硬件模拟。TinyOS 把结点的硬件资源抽象为一系列的组件，通过将硬件中断替换成离散事件，以替换硬件资源。TOSSIM 通过模拟硬件资源被抽象后的组件的行为，为上层提供与硬件相同的标准接口。硬件模拟为模拟真实物理环境提供了接入点。通过修

改硬件抽象组件，可以为用户提供各种性能的硬件环境，满足不同用户和不同仿真配置的需求。

④ 无线模型。TOSSIM 允许开发者选择具有不同精确度和复杂度的无线模型。这种无线模型独立于仿真器，可以保证仿真器的简单性和高效性。用户可以通过一个有向图指定不同结点对之间的通信误码率，表示在该链路上发送数据时可能出现错误的概率。

⑤ 仿真监控。用户可以自行开发应用软件来监控 TOSSIM 的仿真执行过程，二者通过 TCP/IP 进行通信。TOSSIM 为监控软件提供实时的仿真数据，包括在 TinyOS 源代码中加入的 Debug 信息、各种数据包和传感器的采样值等。监控软件可以根据这些数据显示仿真执行情况；同时允许监控软件以命令调用的方式更改仿真程序的内部状态，达到控制仿真进程的目的。TinyOS 提供了一个自带的仿真监控软件界面软件 TinyViz(TinyOS Visualizer)。

总之，TOSSIM 是一个支持基于 TinyOS 的在 PC 上运行的模拟器，它能模拟 nesC 程序在 mote 等硬件上运行的过程。在建立一个 TinyOS 程序后，可以先在 TOSSIM 模拟器上运行和调试。TinyOS 模拟器提供运行时的调试输出信息，允许用户从不同的角度分析和观察程序的执行过程。用户通过 TinyViz 程序可以输入信息，也可以输出调试信息。

2) TOSSIM 模拟器运行 TinyOS 程序

下面介绍如何采用 TOSSIM 模拟器运行 TinyOS 程序。

在 PC 上安装好 TinyOS 之后，可以按照如下关键步骤打开 TinyViz 界面来执行某个应用程序的仿真任务：

① DoS 到应用的目录

② $ make pc

③ 若 TinyViz 还没有建立，则

- cd tools/java/net/tinyos/sim/tinyviz
- make

④ 将 tinyviz 复制到应用目录

⑤ $ DBG=sim

⑥ $ ./tinyviz -run build/pc/main.exe 10

如果输入 make mica2 命令，表示建立 mica2 目录，可以编译生成 mote 上的 exe、srec 和 ihex 文件。

例如我们希望针对 TinyOS 自带的 Blink 应用程序，模拟编译出可以在 TOSSIM 模拟器上运行的程序，主要是在应用程序目录下运行“make pc”命令，就可以把源代码编译在 TOSSIM 模拟器上运行 Blink 应用程序。Blink 应用程序可以在 mote 硬件结点上以频率 1Hz 让 LED 红灯显示。如果执行命令：$ ./tinyviz -run build/pc/main.exe 30，会出现图 5.2 所示的界面。

下面详细介绍如何在 TOSSIM 模拟器运行 Blink 应用程序，可按以下步骤操作：

```
cd app/Blink
make pc
```

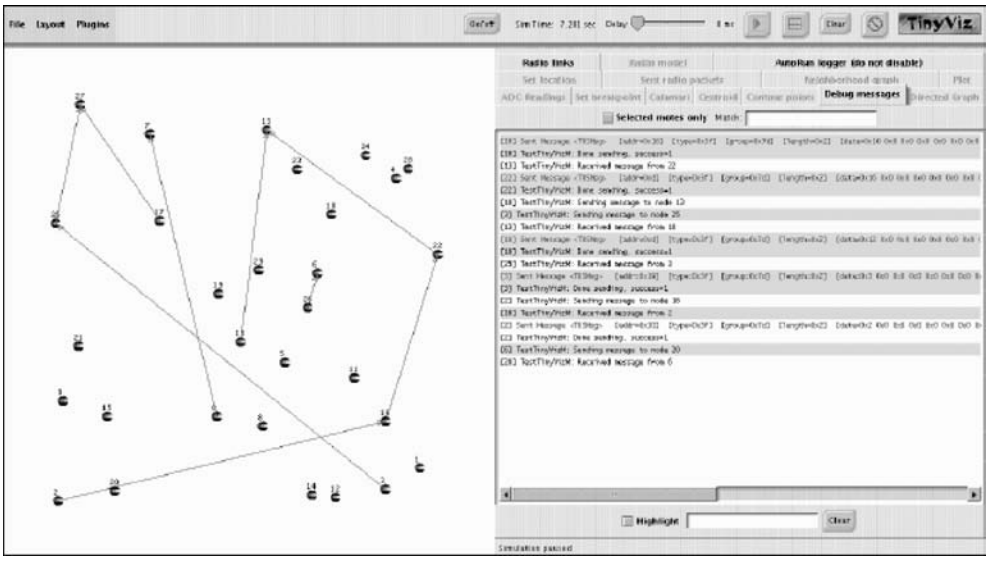

图 5.2　TOSSIM 运行结果的界面显示

这时会出现如下编译信息：

```
mkdir - p build/pc
compiling Blink to a pc binary
ncc  - o build/pc/main.exe  - g  - O0  - board = micasb  - pthread  - target = pc
    - Wall  - Wshadow  - DDEF_TOS_AM_GROUP  =  0x7d
    - WnesC - all  - fnesC - nido - tosnodes =  1000
    - fnesC - cfile =  build/pc/app.c   Blink.nc  - 1m
compiled Blink to build/pc/main.exe
   24784 bytes in ROM
   617960 bytes in RAM
```

最终编译器在 blink/build/pc 下生成可执行文件 main. exe。按照下面的命令运行 main. exe，在默认条件下，TOSSIM 打印出所有的调试信息如下：

```
[root@localhost Blink]# ./build/pc/main.exe 1
SIM: Initializing sockets
SIM: Created server socket listening on port 10584.
SIM: Created server socket listening on port 10585.
SIM: clientAcceptThread running.
SIM: commandReadThread running.
SIM: EEPROM system initialized.
SIM: spatial model initialized.
SIM: RFM model initialized at 40 kbit/sec.
```

按 Ctrl+C 可以终止该程序的运行。通过设置环境变量 export DBG=crc 可以关闭调试信息。通过输入“build/pc/mian. exe-help”能够打印出所有选项。

3）使用 gdb 调试程序

gdb 是一个强大的命令行调试工具，一般来说主要调试 C/C++程序。TOSSIM 的一个显著优点就是它可以运行在 PC 上，这样可以运用传统的调试工具来调试 nesC 程序。不

过，gdb 不是为 nesC 设计的。nesC 中的组件描述意味着单个命令可能有多个提供者。所以单个命令必须指定所处的模块、配件或者接口，才能唯一地确定究竟是哪个命令，例如：

```
module BlinkM {
              provides {
                        interface StdControl;
                       }
              uses {
                     interface Timer;
                     interface Leds;
                    }
              }
  Configuration SingleTimer {
                              provides interface Timer;
                              provides interface StdControl;
                             }
```

编译生成 C 代码时就将它们的名字分别改写，比如 StdControl 接口中的 init()命令在编译为 C 代码的时候，被改写为 BlinkM $ StdControl $ init()和 SingleTimer $ StdControl $ init()。

使用 gdb 进行调试与采用传统的调试方法大致相同，只是使用命令（如在命令处设断点）时必须按照上面的规则。例如，调试 Blink 程序的过程如下：

```
[root@localhost Blink] # gdb build/pc/main.exe
GNU gdb Red Hat Linux
Copyright 2003 Free Software Foundation,Inc.
GDB is free software,covered by the GNU General Public License,and you are welcome to change it
and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB. Type "show warranty" for details.
This GDB was configured as "i386 - redhat - linux - gnu"
(gdb) break main
Breakpoint 1 at 0xS049e10: file Nido.nc,line 113.
(gdb) run
Starting program: /opt/TinyOS/tinyos - 2.x/apps/Blink/build/pc/main.exe
[New Thread 1074102912(LWP 5803)]
[Switching to Thread 1074102912(LWP 5803)]

Breakpoint 1,main(argc = 1,argv = 0xbffff714) at Nido.nc: 113
113       int num nodes start = -1;
(gdb) s
115       unsigned long long maxrun_time = 0;
(gdb) break * BlinkM $ StdControl $ init
Note: breakpoint 1 also set at pc 0xS04d894.
Breakpoint 2 at 0xS04d894: file BlinkM.nc,line 52.
(gdb)
```

注意在这里 BlinkM $ Std Control $ init 是必要的，因为只有这样 gdb 才能够正确解析该命令。

2. OMNeT++

OMNeT++是 Objective Modular Network Testbed 的简写，也被称作离散事件模拟系统(Discrete Event Simulation System，DESS)。它是一种面向对象的、离散事件建模仿真器，属于免费的网络仿真软件[53]。

与大部分的网络模拟器一样，为了更有效地模拟实际网络的运行功能，并使整个系统具有较好的可扩充性，OMNeT 采用了将网络结构定义与网络功能定义分开的方式。对于网络的拓扑结构，由于在模拟的过程中可能需要根据不同的条件进行反复的重定义，并且可能会经常进行相关参数的改动，因此根据这个特点，即不具有很高的复杂度，但要求具有较好的可重用性和较短的开发周期，这种仿真软件工具采用了特别定义的 NED 语言来完成。

对于主要的模块实现和算法实现部分，由于常常会涉及很复杂的数据结构定义，并且在引入面向对象的编程方法后，通过封装也可以实现良好的可扩展性，因此采用了 C++语言来作为功能实现语言。

与其他网络模拟器不同的是，OMNeT++采用的是以 C++为核心的工作模式。用 NED 语言生成的网络拓扑结构的脚本，在生成模拟器的目标文件时，是通过特殊的编译器改写成 C 语言代码，再嵌入到整个工程当中的。在 C++部分所进行的模拟功能部分的实现时，则充分引入和利用了面向对象的编程方法，即由 OMNeT++的开发者通过类封装的方式，向用户提供包含了各种不同的基本的网络器件和相关操作的库，而用户则根据自己的需要，通过对这些包含了基本功能和底层实现接口的库(类)进行重载，从而适应不同应用场合的需要。

这种工作方式可以将较为复杂的底层实现操作与上层用户隔离开来，同时由于采用了封装的方式，使各个功能模块之间具有较好的功能独立性，从而大大降低了实现的复杂程度，同时也方便调试，缩短开发周期。

在 OMNeT++中，网络功能的模拟是针对每一个具体网络来进行的。在每一个具体实现中，所模拟的网络被设计成若干个模块(Module)的集合，对应不同类型的模块在 NED 文件中通过专用对象互相连接成网络。

在 OMNeT++中，所有的网络元素都是以模块形式实现的，每一种网络元素在 C++代码中对应一个特定的类，用户可以对基本类进行重载来加入自己的算法。关键是 OMNeT++允许用户进行模块的嵌套定义，即由若干个基本模块组成一个复合模块，从而可以实现复杂的网络功能定义。

有些网络模拟器在模拟任务的运行方面所采取的方式，是将用户所编写的与本次任务有关的代码，与其他所有代码一起编译到一个共同的可执行文件中。当执行模拟任务时，用户通过另外编写一个运行脚本，指示可执行文件的动作。这一方式与 UNIX 中的内核编程方式比较相似，所生成的可执行文件实际上是一个命令解释器，通过它对脚本中的语句进行解释执行，从而生成网络拓扑，并执行相对应的功能。

OMNeT++在这方面所采用的方式则不同，它为每个模拟任务生成独立的可执行文件。当用户需要改变网络参数，或者需要改变网络动作时，则必须要重新修改源代码，再重新进行编译。OMNeT++的优点在于所生成的可执行文件只包含本次任务所需要的功能，所以在生成时间和稳定性上都有优势。但是，其不足在于可扩展性。一些网络仿真软件工具在

改变网络拓扑、参数和网络动作时，只需要修改相应的脚本，而 OMNeT++要对源代码进行修改，再重新编译。

OMNeT++所采用的编译方式与大多数的网络编译器类似，也是通过使用一个预先生成的编译脚本(通常为 makefile. vc)，利用 C 语言的编译器 nmake 完成编译工作。OMNeT++提供相应的工具，根据不同的工程自动生成相应的编译脚本。

NED 语言是 OMNeT++的专用语言，用于生成静态的网络拓扑，设置网络的相关参数，并对这些参数进行初始化。在编译阶段 NED 代码通过专用的编译器转换为 C++代码，进行二次编译，其中网络拓扑结构和相关参数的设置也可以在 C++代码中进行动态的修改。

为了满足用户的不同需要，OMNeT++为模拟程序的运行提供了可选择的两种不同的方式，即通过命令行运行的 Cmdenv 方式和具有图形界面、便于直观分析的 Tkenv 方式。OMNeT++的模拟程序采用一种名为. vec 的文件格式，作为它的模拟输出，并提供相应的工具进行分析。

OMNeT++在模拟程序的运行和结果的分析方面，为用户提供了许多功能选项，这些选项中的绝大部分是在模拟工程的 omnetpp. ini 配置文件中设定的。omnetpp. ini 文件是每一个 OMNeT++模拟程序的默认配置文件，在模拟过程开始时，系统模块自动在当前目录寻找并读取这个文件，根据它的指令设置运行方式和相关参数。

3. MATLAB

MATLAB 是矩阵实验室(Matrix Laboratory)的意思。它除了具备卓越的数值计算能力外，还提供专业水平的符号计算、文字处理、可视化建模仿真和实时控制等功能，也可以进行网络仿真，用于模拟传感器网络的运行情况和某些应用算法的性能。

MATLAB 作为美国 MathWorks 公司开发的用于概念设计、算法开发、建模仿真的理想集成环境，是目前非常好的一种科学计算类软件。它作为和 Mathematica、Maple 并列的三大数学软件，其强项就是强大的矩阵计算和仿真能力。

MATLAB 的基本数据单位是矩阵，它的指令表达式与数学、工程中常用的形式十分相似。用户可以将自己编写的实用程序导入到 MATLAB 函数库，方便自己以后调用，此外许多 MATLAB 爱好者编写了一些经典的程序，用户可以直接进行下载使用，非常方便。

MATLAB 的基础是矩阵计算，开放性使 MATLAB 广受用户欢迎，除内部函数外，所有 MATLAB 主包文件和各种工具包都是可读、可修改的文件，用户通过对源程序的修改或加入自己编写的程序，可以构造出新的专用工具包。MATLAB 的官方网站地址为 http://www. mathworks. com。

在 MATLAB 软件工具中，典型的无线传感器网络应用程序是 WiSNAP(Wireless Image Sensor Network Application Platform)[54]。这是一个针对无线图像传感器网络而设计的基于 MATLAB 的应用开发平台，由斯坦福大学研制。它使得研究者能使用实际的目标硬件来研究、设计和评估算法应用程序，还提供了标准易用的应用程序接口(Application Program Interface，API)来控制图像传感器和无线传感器结点，而不需要详细了解硬件平台，另外开放的系统结构还支持虚拟的传感器和无线传感器结点。

4. OPNET

OPNET 是 MIL3 公司开发的网络仿真软件产品。这是一种优秀的图形化、支持面向对象建模的大型网络仿真软件。它具有强大的仿真功能，几乎可以模拟任何网络设备、支持各种网络技术，能够模拟固定通信模型、无线分组网模型和卫星通信网模型。OPNET 还提供交互式的运行调试工具和功能强大、便捷直观的图形化结果分析器，以及提供能够实时观测模型动态行为的动态观测器[55]。

OPNET 产品主要面向专业人士，帮助客户进行网络结构、设备和应用的设计、建设、分析和管理。OPNET 的产品主要针对三类客户，分成四个系列。三类客户是指网络服务提供商、网络设备制造商和一般企业。它的四个系列产品核心包括：

(1) OPNET Modeler。为技术人员提供一个网络技术和产品开发平台，可以帮助他们设计和分析网络和通信协议。

(2) ITGuru。帮助网络专业人士预测和分析网络和网络应用的性能、诊断问题、查找影响系统性能的瓶颈、提出并验证解决方案。

(3) ServiceProviderGuru。面向网络服务提供商的智能化网络管理软件。

(4) WDM Guru。用于波分复用光纤网络的分析、评测。

OPNET 的主要特点包括以下几方面。

(1) 采用面向对象的技术。对象的属性可以任意配置，每一对象属于相应行为和功能的类，可以通过定义新的类来满足不同的系统要求。

(2) 提供了各种通信网络和信息系统的处理构件和模块。

(3) 采用图形化界面来建模，为使用者提供三层（网络层、结点层、进程层）建模机制来描述现实的系统。

(4) 在过程层次中使用有限状态机来对其他协议和过程进行建模，用户模型和 OPNET 的内置模型将会自动生成 C 语言，实现可执行的离散事件的模拟流程。

(5) 内建了很多性能分析器，自动采集模拟过程的结果数据。

(6) 几乎预定义了所有常用的业务模型，如均匀分布、泊松分布、爱尔朗分布等。

Modeler 作为 OPNET 软件的核心工具，其应用领域包括端到端结构、系统级的仿真、新的协议开发和优化、网络和业务层配合如何达到最佳性能等方面。举例来说，在端到端结构的应用中，从 IPv4 网络升级为 IPv6，分析采用哪种技术方式对转移效果比较好。在新协议的开发方面，如目前流行的 3G 无线协议，可以分析协议设计的运行效果。在系统级的仿真中，分析一种新的路由或调度算法如何使路由器或者交换机达到用户满意的服务质量。在网络和业务之间如何优化方面，可以分析新引进的业务对整个网络性能的影响、网络对业务的要求。由于实际中网络和业务是一对矛盾，通过 Modeler 模拟来查找网络和业务之间所能达到的最优指标。

Modeler 采用阶层性的模拟方式，从协议间的关系来看，结点模块建模完全符合 OSI 标准，即业务层→TCP 层→IP 层→IP 封装层→ARP 层→MAC 层→物理层。从网络层次关系来看，提供了三层建模机制，最底层为进程（Process）模型，采用状态机来描述协议；其次为结点（Node）模型，由相应的协议模型构成，反映设备特性；最上层为网络模型。三层模型和实际的协议、设备、网络完全对应，全面反映了网络的相关特性。

Modeler 采用面向对象建模(Object-oriented Modeling,OOM)方式,每一类结点开始都采用相同的结点模型,再针对不同的对象,设置特定的参数。采用离散事件驱动的模拟机理,与时间驱动相比,计算效率得到了很大提高。例如在仿真路由协议时,如果要了解封包是否到达,不必要每隔很短时间周期性地查看一次,而是收到封包和事件到达才去查看。在 Modeler 中所有代码包括各种协议的代码都完全公开,每一个代码的注释也非常清楚,使得用户容易理解协议的内部运作。采用混合建模机制,把基于包的分析方法和基于统计的数学建模方法结合起来,既可得到非常细节的模拟结果,也大大提高了仿真效率。

在仿真引擎的效率方面,Modeler 的仿真速度很高,这是它的一个显著优点,同时引入并行仿真,使得无论无线还是有线的仿真速度更加快速。Modeler 还提供了多种业务模拟方式,具有丰富的收集分析统计量、查看动画和调试等功能。它可以直接收集常用的各个网络层次的性能统计参数,能够方便地编制和输出仿真结果的报告,如图 5.3 所示[56]。

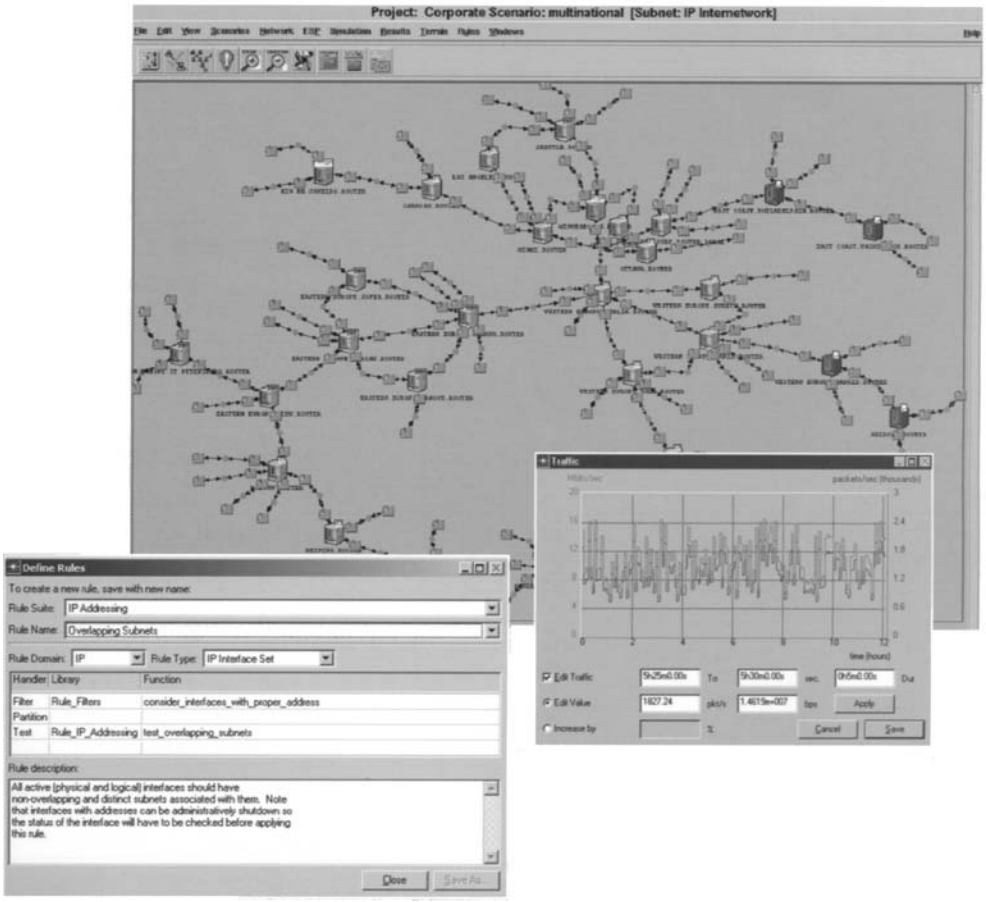

图 5.3 OPNET 的 Modeler 运行界面和仿真结果分析与显示

总之,OPNET 网络仿真软件是目前世界上最为先进的网络仿真开发和应用平台之一。它曾被一些机构评选为“世界级网络仿真软件”第一名。由于其出众的技术而被许多大型通信设备商、电信运营商、军方和政府研发机构、高等教育院校、大中型企业所采用,也可以进

行传感器网络的各种应用业务仿真和网络协议运行性能模拟。使用它的最大问题在于它作为一种商业化高端网络仿真产品，价格十分昂贵。

5. NS

NS(Network Simulator)是一种针对网络技术的源代码公开的、免费的软件模拟平台，研究人员使用它可以很容易地进行网络技术的分析。目前它所包含的模块内容已经非常丰富，几乎涉及网络技术的所有方面，成为目前学术界广泛使用的一种网络模拟软件。在每年国内外发表的有关网络技术的学术论文中，利用 NS 给出模拟结果的文章最多，通过这种方法得出的研究结果也是被学术界所普遍认可的[57]。

NS 也可作为一种辅助教学的工具，广泛应用在网络技术的教学方面。这种网络仿真软件有多个版本，这里以 NS2(Network Simulator，version 2)[58]为例进行介绍。作为一种面向对象的网络仿真器，NS2 本质上是一个离散事件模拟器，由加州大学伯克利分校开发。它本身有一个虚拟时钟，所有的仿真都由离散事件驱动。NS2 使用 C++和 OTcl 作为开发语言。NS 可以说是 OTcl 的脚本解释器，它包含仿真事件调度器、网络组件对象库以及网络构建模型库等。

事件调度器用来计算仿真时间，并且激活事件队列中的当前事件，执行一些相关的事件，网络组件通过传递分组来相互通信，但这并不耗费仿真时间。所有需要花费仿真时间来处理分组的网络组件都必须要使用事件调度器。它先为这个分组发出一个事件，然后等待这个事件被调度回来之后，才能做下一步的处理工作。事件调度器的另一个用处就是计时。由于效率的原因，NS 将数据通道和控制通道的实现相分离。为了减少分组和事件的处理时间，事件调度器和数据通道上的基本网络组件对象都使用 C++写出并编译的，这些对象通过映射对 OTcl 解释器可见。

当仿真完成以后，NS 将会产生一个或多个基于文本的跟踪文件。只要在脚本中加入一些简单的语句，这些文件中就会包含详细的跟踪信息。这些数据可以用于下一步的分析处理，也可以将整个仿真过程展示出来。

在进行网络仿真之前，首先分析仿真涉及哪个层次，NS 仿真分两个层次：一个是基于 OTcl 编程的层次。利用 NS 已有的网络元素实现仿真，无须修改 NS 本身，只需编写 OTcl 脚本。另一个是基于 C++和 OTcl 编程的层次。如果 NS 中没有所需的网络元素，则需要对 NS 进行扩展，添加所需网络元素，即添加新的 C++和 OTcl 类，编写新的 OTcl 脚本。

假设用户已经完成了对 NS 的扩展，或者 NS 所包含的构件已经满足了要求，那么进行一次仿真的步骤大致如下：

(1) 编写 OTcl 脚本。首先配置模拟网络拓扑结构，此时可以确定链路的基本特性，如延迟、带宽和丢失策略等。

(2) 建立协议代理，包括端设备的协议绑定和通信业务量模型的建立。

(3) 配置业务量模型的参数，从而确定网络上的业务量分布。

(4) 设置 Trace 对象。NS 通过 Trace 文件来保存整个模拟过程。在仿真过程结束之后，用户可以对 Trace 文件进行分析研究。

(5) 编写其他的辅助过程，设定模拟结束时间，至此 OTcl 脚本编写完成。

(6) 用 NS 解释执行刚才编写的 OTcl 脚本。

(7) 对 Trace 文件进行分析,得出有用的数据。

(8) 调整配置拓扑结构和业务量模型,重新进行上述模拟过程。

NS2 采用两级体系结构,为了提高代码的执行效率,NS2 将数据操作与控制部分的实现相分离。事件调度器和大部分基本的网络组件对象后台使用 C++实现和编译,称为编译层,主要功能是实现对数据包的处理。NS2 的前端是一个 OTcl 解释器,称为解释层,主要功能是完成模拟环境的配置和建立。

从用户角度来看,NS2 是一个具有仿真事件驱动、网络构件对象库和网络配置模块库的 OTcl 脚本解释器。NS2 中编译类对象通过 OTcl 连接建立了与之对应的解释类对象,这样用户间能够方便地对 C++对象的函数进行修改与配置,充分体现了仿真器的一致性和灵活性。

NS2 仿真器封装了许多功能模块,基本模块包括事件调度器、结点、链路、代理、数据包格式等。各个模块的大致情况简介如下:

(1) 事件调度器。目前 NS2 提供了四种具有不同数据结构的调度器,分别是链表、堆、日历表和实时调度器。

(2) 结点(node)。由 TclObject 对象组成的复合组件,在 NS2 中可以表示端结点和路由器。

(3) 链路(link)。由多个组件复合而成,用来连接网络结点。所有的链路都是以队列的形式来管理分组的到达、离开和丢弃。

(4) 代理(agent)。负责网络层分组的产生和接收,也可以用在各个层次的协议实现中。每个代理连接到一个网络结点上,由该结点给它分配一个端口号。

(5) 包(packet)。由头部和数据两部分组成。一般情况下包只有头部、没有数据部分。

NS2 软件由 Tcl/Tk、OTcl、NS、Tclcl 构成。这里 Tcl 是一个开放脚本语言,用来对 NS2 进行编程。Tk 是 Tcl 的图形界面开发工具,可帮助用户在图形环境下开发图形界面。OTcl 是基于 Tcl/Tk 的面向对象扩展,有自己的类层次结构。NS 模块作为这种软件包的核心,是面向对象的仿真器,用 C++编写,以 OTcl 解释器作为前端。Tclcl 模块提供 NS 和 OTcl 的接口,使对象和变量出现在两种语言中。为了直观的观察和分析仿真结果,NS2 提供了可选的 Xgraphy、可选件 Nam。

5.1.3 仿真平台的选择和设计

1. 仿真平台的选择

无线传感器网络的仿真要能够在一个可控制的环境里,分析和研究它的网络性能和应用业务的实现情况,包括操作系统和网络协议栈,能够仿真数量众多的结点,并可以观察由不可预测的干扰和噪声引起的结点间的相互作用,从而获取结点间组网和数据传输的详细细节,提高结点投放后的网络运行成功率,减少投放后的网络维护工作。

在传感器网络中,通常单个传感器结点具有两个突出特点:一个特点是它的并发性很密集,另一个特点是传感器结点的模块化程度很高。上述这些特点使得无线传感器网络仿

真需要解决可扩展性与仿真效率、分布与异步特性、动态性、综合仿真平台等问题。

通过对任务需求和现有仿真平台的分析，要了解选择的仿真平台是否满足需求。如果现有平台能够满足要求，则选择现有平台，否则需要设计新的仿真平台。

根据前面的内容介绍，我们知道现有的仿真平台种类较多、功能各异，每个仿真软件平台的侧重点也不同。仿真平台所采用的设计方法也不一样，例如面向对象设计和面向组件设计等，也会影响仿真平台的执行效率、速度、扩展性、重用性和易用性等。每个仿真器都是在某些性能方面比较突出，而在其他方面又不重视。在选择仿真平台时，需要综合考虑各个因素，在其中寻找一个平衡点以获得最佳的仿真效果。

2. 仿真平台的自主设计

如果开发者决定构建一个自己的传感器网络仿真工具，首先需要决定是在现有仿真平台上开发还是单独构建。如果开发时间有限并且只有一些需要用到的特定特性在现有工具中不存在，那么最好是在现有仿真平台上做开发。如果有足够的开发时间，并且开发者的设计思路比现有工具在仿真规模、执行速度、特点等方面优越，那么从头开始创建一个仿真工具是最有效的。

从底层构建一个仿真工具需要做许多工作，主要是做好方案的选择。开发者必须考虑不同的编程语言的优缺点、仿真驱动的方式（基于事件还是时间）、基于组件结构还是基于面向对象结构、仿真器的复杂程度、包括哪些特点以及不包括哪些、是否使用并行执行、与实际结点交互的能力。

为了提高效率，许多仿真器使用离散事件模型来驱动引擎。基于组件的结构比面向对象结构优越，但是通过模块化的方式比较难以实现。定义每个传感器为对象也可以确保结点之间的独立性。在面向对象设计中，不同协议之间新算法的移植也较容易。然而，通过仔细编程和实验，人们发现基于组件结构会表现更加有效。

通常自主构建仿真器的复杂程度与设计者的目的和时间有关。在大多数情况下，人们使用简单的 MAC 协议就足够满足传感器网络组网的无线通信仿真需求。其他的设计内容还取决于具体应用业务的特定场景、编程能力、设计时间等。

5.1.4 传感器网络工程测试床

在无线传感器网络中，仿真是一个重要的研究手段。但是仿真通常仅局限于特定问题的研究，并不能获取结点、网络和无线通信等运行的详细信息，只有实际的测试床（Testbed）才能够捕获到这些信息。

虽然在验证大型传感器网络方面有一些有效的仿真工具，但只有通过对实际的传感器网络测试床的使用，才能真正理解资源的限制、通信损失及能源限制等问题。另外，无线传感器网络的测试是比较困难的，在实验过程中需要对许多结点反复地进行重编程、调试，而且还需要获得其中的一些数据信息。通过测试床可以对无线传感器网络的许多问题进行研究，简化系统部署、调试等步骤，使得无线传感器网络的研究和应用变得相对容易。

最近几年来随着传感器、无线通信设备等的价格降低和尺寸减小，人们设计出了一些测试床来研究和验证这些系统的属性。这里主要介绍 3 种应用于无线传感器网络的测试床：

Motelab、SensoNet 和 IBM 的无线传感器网络测试床。

Motelab 是哈佛大学开发的一个开放的无线传感器网络实验环境，是基于 Web 的无线传感器网络测试床。它包括一组长期部署的传感器网络结点，以及一个中心服务器，体系结构如图 5.4 所示[59]。中心服务器负责处理重编程、数据访问，并提供创建和调度测试床的 Web 接口。Motelab 通过流线型结构访问大型和固定的网络设备，加速了应用程序的部署。各地用户通过互联网可以实现自动的数据访问，允许离线对传感器网络软件的性能进行验证，方便系统调试和开发。因此它的一个重要特点是允许本地和远程用户通过 Web 接口接入测试床，例如我们可以通过互联网上传自己的数据和文件，进行传感器网络测试。Motelab 已经用于多个项目的研究和开发，也可以用于教学用途。

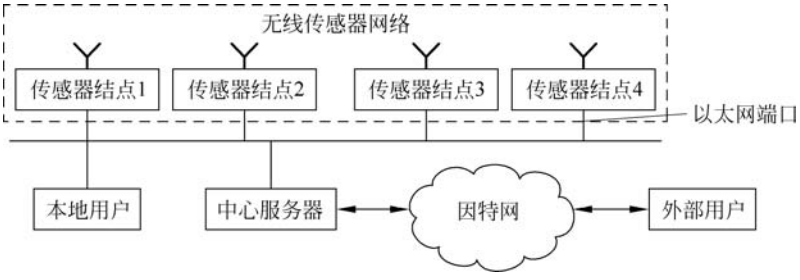

图 5.4 Motelab 工程测试床的结构组成

SensoNet 是美国亚特兰大市佐治亚州技术学院电子与计算机工程学校宽带与无线网络实验室研制的传感器网络试验床，其体系结构如图 5.5 所示[60]。

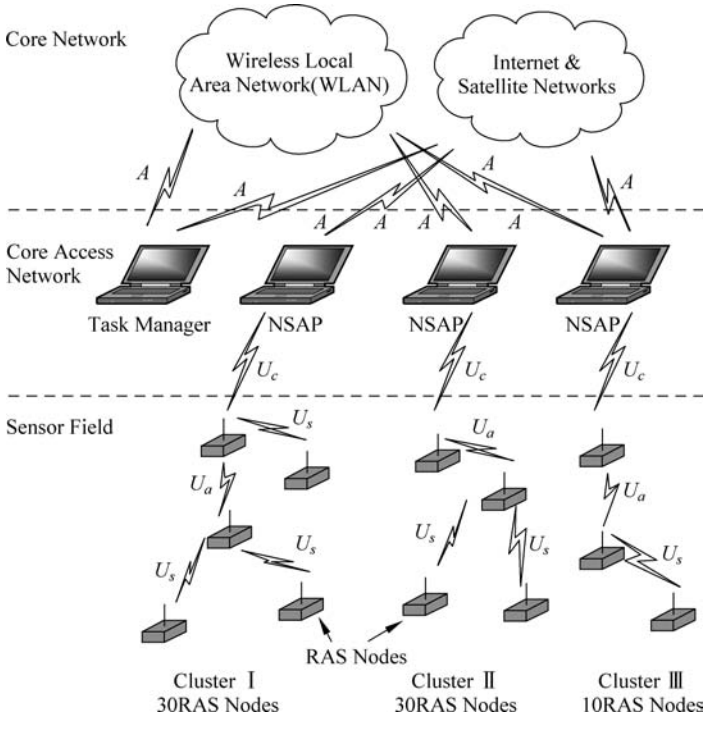

图 5.5 SensoNet 工程测试床的结构组成

SensoNet 试验床是由三部分组成：核心网、核心接入网和传感器现场。核心网是整个传感器网络的主干，包括无线局域网（WLAN）、因特网和卫星网。核心接入网是由 NSAP 组成，即传感器网络的信宿，它是用笔记本电脑来模拟。NSAP 将核心网中的协议与传感器网络中的协议结合在一起。传感器现场是一个区域，其中部署 RAS（Route，Access，Sense）结点。每个 RAS 结点可以是移动的或者静止的，它由以下部分组成：

（1）MPR300CA MICA 程序/无线主板：TinyOS 分布式软件操作系统，Atmel Atmega 103L 处理器（运行在 4MHz，128KB 的内部 Flash），4KB 的外部 SRAM，4KB 的外部 EEPROM，RFM TR1000 916MHz 无线收发器（最大速率为 50Kb/s），MICA 传感器为 51 针扩充连接器。

（2）MTS310CA MICA 型号的光、温度、声音、加速计传感器。

（3）笔记本电脑，带有 WLAN 和 916MHz 的无线收发器。

（4）视频和音频获取装置。

（5）差分全球定位系统（GPS）。

（6）遥控车或者布朗李机动车。

试验床的部分场景和实物图片如图 5.6 所示。

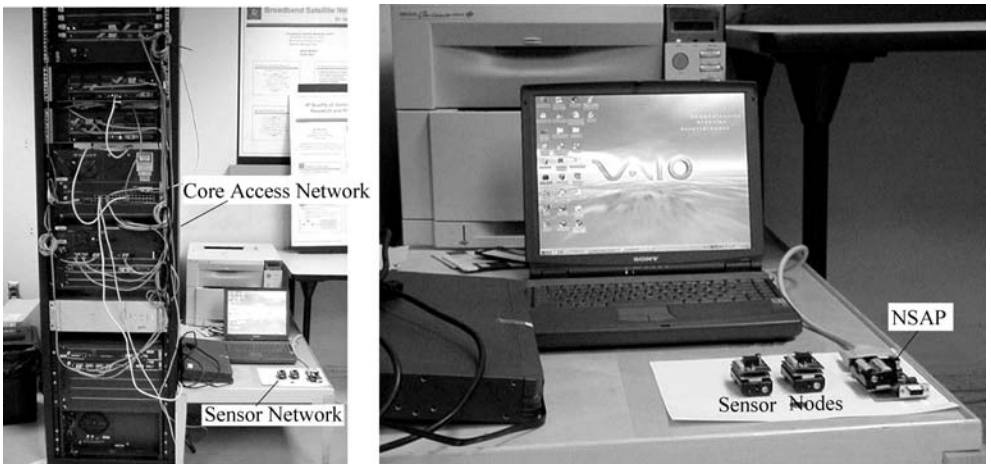

图 5.6　SensoNet 工程测试床的场景和部分实物

IBM 苏黎世研究实验室开发的测试床提供了完整的端到端解决方案，包括从传感器与执行机构到运行企业服务器上的应用软件。它由多种无线传感器网络、一个传感器网关、连接传感器网络和企业网络的中间件和传感器应用软件组成。这个测试床用于评估与传感器网络相关的无线通信技术（如 IEEE 802.15.4/ZigBee 网络、蓝牙无线局域网和 IEEE 802.11b 无线局域网）的性能，测试在传感器和应用服务器之间实现异步通信的轻量级消息协议，并可以开发具体的应用，例如实现远程测量和位置感知。具体地说，该测试床包括以下几部分[61]：

- 配有多种类型传感器的传感单元（如加速度计、温度计、陀螺仪等）；
- 一个无线传感器网络；
- 一个连接无线传感器网络和企业网络的网关；

- 分发传感器数据到各个传感器应用的中间件组件；
- 传感器应用软件。

5.2 网络结点的硬件开发

5.2.1 硬件开发概述

1. 硬件系统的设计特点和要求

无线传感器网络具有很强的应用相关性，在不同应用场合下需要配套不同的网络模型、软件系统和硬件平台。传感器网络结点作为一种微型化的嵌入式系统，构成了传感器网络的基础层支撑平台。设计传感器网络的硬件结点需从以下方面考虑。

1）微型化

传感器结点应该在体积上足够小，保证对目标系统本身的特性不会造成影响，或者所造成的影响可忽略不计。

2）扩展性和灵活性

传感器网络结点需要定义统一、完整的外部接口，在需要添加新的硬件部件时可以在现有的结点上直接添加，而不需要开发新的结点。同时结点可以按照功能拆分成多个组件，组件之间通过标准接口自由组合。在不同应用环境下，选择不同的组件自由配置系统，这样不必为每一个应用都开发一套全新的硬件系统。

3）稳定性和安全性

稳定性要求结点的各部件都能够在给定的外部环境变化范围内正常工作。在给定的温度、湿度和压力等外部条件下，传感器结点的处理器、无线通信模块和电源模块要保证正常功能，同时传感器部件保证工作在各自量程范围内。

安全性设计主要包括代码安全和通信安全两方面。在代码安全方面，某些应用场合可能希望保证结点的运行代码不被第三方了解。例如在某些军事应用中，在结点被敌方捕获的情况下，结点的代码应该能够自我保护并锁死，避免被敌方获取。很多微处理器和存储器芯片都具有代码保护的能力。在通信安全方面，有些芯片能够提供一定的硬件支持，如CC2420具有支持基于AES-128的数据加密和数据鉴权能力。

4）低成本

这是传感器结点的基本要求，只有低成本才能大量布置在目标区域，表现出传感器网络的各种优点。低成本对传感器各个部件提出苛刻的要求。首先，供电模块不能使用复杂而且昂贵的方案；其次，能量的限制要求所有的器件都必须是低功耗；另外传感器不能使用精度太高、线性很好的部件，这样会造成传感器模块成本过高。

5）低功耗

传感器网络对低功耗的需求一般都远远高于目前已成熟的蓝牙、WLAN等无线网络。传感器结点的硬件设计直接决定了结点的能耗水平，还决定了各种软件通过优化（如网络各层通信协议的优化设计、功率管理策略的设计）可能达到的最低能耗水平。通过合理地设计

硬件系统，可以有效降低结点能耗。

由于具体应用背景不同，目前国内外研制了多种无线传感器网络结点的硬件平台。典型的结点包括 Mica 系列、Sensoria WINS、Toles、μAMPS 系列、XYZnode、Zabranet 等。实际上各平台最主要的区别在于采用了不同的处理器、无线通信协议和与应用相关的不同传感器。常用的无线通信协议有 IEEE 802.11b、IEEE 802.15.4(ZigBee)、Bluetooth、UWB 和自定义协议。采用的处理器从 4 位的微控制器到 32 位 ARM 内核的高端处理器，还有一类结点如 WiseNet，采用集成了无线模块的单片机。

2. 硬件系统的设计内容

大多数传感器网络结点具有终端探测和路由的双重功能：一方面实现数据的采集和处理；另一方面实现数据的融合和路由，对本身采集的数据和收到的其他结点发送的数据进行综合，转发路由到网关结点。网关结点往往个数有限，而且常常能够得到能量补充。网关通常使用多种方式(如因特网、卫星或移动通信网络等)与外界进行通信。

通常普通的传感器网络结点数目大，采用不能补充的电池提供能量。传感器结点的能量一旦耗尽，那么该结点就不能进行数据采集和路由，直接影响整个传感器网络的健壮性和生命周期。因此，传感器网络设计的主要内容在于传感器网络结点。

由于具体应用不同，传感器网络结点的设计也不尽相同，但是基本结构通常大致是一样的。传感器结点的基本硬件模块组成如图 5.7 所示，主要由数据处理模块、换能器模块、无线通信模块、电源模块和其他外围模块组成。数据处理模块是结点的核心模块，用于完成数据处理、数据存储、执行通信协议和结点调度管理等工作；换能器模块包括各种传感器和执行器，用于感知数据和执行各种控制动作；无线通信模块用于完成无线通信任务；电源模块是所有电子系统的基础，电源模块的设计直接关系到结点的寿命；其他外围模块包括看门狗电路、电池电量检测模块等，也是传感器结点不可缺少的组成部分。

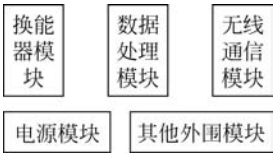

图 5.7 无线传感器网络结点的结构组成

5.2.2 传感器结点的模块化设计

1. 数据处理模块

分布式信息采集和数据处理是无线传感器网络的重要特征之一。每个传感器结点都具有一定的智能性，能够对数据进行预处理，并能够根据感知的情况做出不同处理。这种智能性主要依赖于数据处理模块来实现的，数据处理模块是传感器结点的核心模块之一。

从处理器的角度来看，传感器网络结点基本可以分为两类：一类采用以 ARM 处理器为代表的高端处理器。该类结点的能量消耗比采用微控制器大很多，多数支持 DVS(动态电压调节)或 DFS(动态频率调节)等节能策略，但是其处理能力也强很多，适合图像等高数据量业务的应用。另外，采用高端处理器来作为网关结点也是不错的选择。另一类是以采用低端微控制器为代表的结点。该类结点的处理能力较弱，但是能量消耗也很小。在选择处理器时应该首先考虑系统对处理能力的需要，然后再考虑功耗问题。

对于数据处理模块的设计，主要考虑如下五方面的问题。

1）节能设计

从能耗的角度来看，对于传感器结点，除通信模块以外，微处理器、存储器等用于计算和存储数据的模块也是主要的耗能部件。它们都直接关系到结点的寿命，因此应该尽量使用低功耗的微处理器和存储器芯片。

在选择微处理器时切忌一味追求性能，选择的原则应该是“够用就好”。现在微处理器运行速度越来越快，但性能的提升往往带来功耗的增加。一个复杂的微处理器集成度高、功能强，但片内晶体管多，总漏电流大，即使进入休眠或空闲状态，漏电流也变得不可忽视；而低速的微处理器不仅功耗低，成本也低。另外，应优先选用具有休眠模式的微处理器，因为休眠模式下处理器功耗可以降低 3～5 个数量级。

选择合适的时钟方案也很重要。时钟的选择对于系统功耗相当敏感，系统总线频率应当尽量降低。处理器芯片内部的总电流消耗可分为两部分，即运行电流和漏电流。理想的CMOS 开关电路，在保持输出状态不变时是不消耗功率的。但在微处理器运行时，开关电路不断由“1”变“0”、由“0”变“1”，消耗的功率由微处理器运行所引起，称之为“运行电流”。运行电流几乎与微处理器的时钟频率成正比，尽量降低系统时钟的运行频率能够有效地降低系统功耗。

现代微处理器普遍采用锁相环技术，使其时钟频率可由程序控制。锁相环允许用户在片外使用频率较低的晶振，可以减小板级噪声；而且，由于时钟频率由程序控制，系统时钟可在一个很宽的范围内调整，总线频率往往能升得很高。但是，使用锁相环也会带来额外的功耗。如果仅针对时钟设计方案来说，使用外部晶振且不使用锁相环是功耗最低的一种方案选择。

2）处理速度的选择

过快的处理速度可能会增加系统的功耗，但如果处理器承担的处理任务较重，那么若能尽快完成任务，可以尽快转入休眠状态，从而降低能耗。另外，由于需要支持网络协议栈的实时运行，处理模块的速度也不能太低。

3）低成本

低成本是传感器网络实用化的前提条件。在某些情况下，例如在温度传感器结点中，数据处理模块的成本可能占到总成本的 90％以上。片上系统（SoC）需要的器件数量最少，系统设计最简单，成本最低，一般来说对于独立系统应用（如电灯开关等）而言，它是最合适的选择。但是基于 SoC 的设计通常仅对某些特殊的市场需求而言是最优的，由于 MCU 内核速度和内部存储器容量等不能随应用需求进行调整，必须有足够大的市场需求量才能使产品设计的巨大投资得到回报。

4）小体积

由于结点的微型化，应尽量减小数据处理模块的体积。

5）安全性

很多微处理器和存储器芯片提供内部代码安全保密机制，这在某些强调安全性的应用场合尤其必要。

微处理器单元是传感器网络结点的核心，负责整个结点系统的运行管理。表 5.1 所示为各种常见的微控制器性能比较。

表 5.1 各种常见的微控制器性能列表

厂　商	芯片型号	RAM 容量/KB	Flash 容量/KB	正常工作电流/mA	休眠模式下的电流/μA
Atmel	Mega103	4	128	5.5	1
	Mega128	4	128	8	20
	Mega165/325/645	4	64	2.5	2
Microchip	PIC16F87x	0.36	8	2	1
Intel	8051 8 位 Classic	0.5	32	30	5
	8051 16 位	1	16	45	10
Philips	80C51 16 位	2	60	15	3
Motorola	HC05	0.5	32	6.6	90
	HC08	2	32	8	100
	HCS08	4	60	6.5	1
TI	MSP430F14x16 位	2	60	1.5	1
	MSP430F16x16 位	10	48	2	1
Atmel	AT91 ARM Thumb	256	1024	38	160
Intel	XScale PXA27x	256	N/A	39	574
Samsung	S3C44B0	8	N/A	60	5

在选择处理器时应该首先考虑系统对处理能力的需要，然后再考虑功耗问题。不过对于功耗的衡量标准不能仅仅从处理器有几种休眠模式、每 MHz 时钟频率所耗费的能量等角度去考虑处理器自身的功耗，还要从处理器每执行一次指令所耗费的能量这个指标综合考虑。表 5.2 是目前一些常用处理器在不同的运行频率下每指令所耗费能量的数据列表。

表 5.2 常用处理器的每指令所耗费能量

芯片型号	运行电压/V	运行频率	单位指令消耗能量/nJ
ATMega128L	3.3	4MHz	4
ARM Thumb	1.8	40MHz	0.21
C8051F121	3.3	32kHz	0.2
IBM 405LP	1	152MHz	0.35
C8051F121	3.3	25MHz	0.5
TMS320VC5510	1.5	200MHz	0.8
Xscale PXA250	1.3	400MHz	1.1
IBM 405LP	1.8	380MHz	1.3
Xscale PXA250	0.85	130MHz	1.9

目前处理器模块中使用较多的是 Atmel 公司的单片机。它采用 RISC 结构，吸取了 PIC 和 8051 单片机的优点，具有丰富的内部资源和外部接口。在集成度方面，其内部集成了几乎所有关键部件；在指令执行方面，微控制单元采用 Harvard 结构，因此指令大多为单周期；在能源管理方面，AVR 单片机提供多种电源管理方式，尽量节省结点能量；在可扩展性方面，提供多个 I/O 口，并且和通用单片机兼容；此外，AVR 系列单片机提供的 USART（通用同步异步收发器）控制器、SPI（串行外围接口）控制器等与无线收发模块相结合，能够实现大吞吐量，高速率的数据收发。

TI公司的MSP430超低功耗系列处理器，不仅功能完善、集成度高，而且根据存储容量的多少提供多种引脚兼容的系列处理器，使开发者可以根据应用对象灵活选择。

另外，作为32位嵌入式处理器的ARM单片机，也已经在无线传感器网络方面得到了应用。如果用户可以接受它的较高成本，那么可以利用这种单片机来运行复杂的算法，完成更多的应用业务功能。

2. 换能器模块

所谓换能器(transducer)是指将一种物理能量变为另一种物理能量的器件，包括传感器和执行器两种类型。它涉及各种类型的传感器，如声响传感器、光传感器、温度传感器、湿度传感器和加速度传感器等。另外，传感器结点中还可能包含各种执行器，如电子开关、声光报警设备、微型电动机等。

大部分传感器的输出是模拟信号，但通常无线传感器网络传输的是数字化的数据，因此必须进行模/数转换。类似的，许多执行器的输出也是模拟的，因此也必须进行数/模转换。在网络结点中配置模/数和数/模转换器(ADC和DAC)，能够降低系统的整体成本，尤其是在结点有多个传感器且可共享一个转换器的时候。作为一种降低产品成本的方法，传感器结点生产厂商可以选择不在结点中包含ADC或DAC，而是使用数字换能器接口。

为了解决换能器模块与数据处理模块之间的数据接口问题，目前已制定了IEEE 1451.5智能无线传感器接口标准。IEEE 1451系列标准是由IEEE仪器和测量协会的传感器技术委员会发起并制定的，具体细节在本书第6章给予介绍。

3. 无线通信模块

无线通信模块由无线射频电路和天线组成，目前采用的传输介质主要包括无线电、红外、激光和超声波等，它是传感器结点中最主要的耗能模块，是传感器结点的设计重点。

现今传感器网络应用的无线通信技术通常包括IEEE 802.11b、IEEE 802.15.4(ZigBee)、Bluetooth、UWB、RFID和IrDA等，还有很多芯片的通信协议由用户自己定义，这些芯片一般工作在ISM免费频段，表5.3为目前传感器网络应用的常见无线通信技术列表。

表5.3 传感器网络的常用无线通信技术

无线技术	频率	距离/m	功耗	传输速率/Kb·s^{-1}
Bluetooth	2.4GHz	10	低	10 000
802.11b	2.4GHz	100	高	11 000
RFID	50kHz～5.8GHz	＜5	～	200
ZigBee	2.4GHz	10～75	低	250
IrDA	Infrared	1	低	16 000
UWB	3.1～10.6GHz	10	低	100 000
RF	300～1000MHz	10X～100X	低	10X
X表示数字1～9				

在无线传感器网络中应用最多的是ZigBee和普通射频芯片。ZigBee是一种近距离、低复杂度、低功耗、低数据速率、低成本的双向无线通信技术，完整的协议栈只有32KB，可以

嵌入到各种微型设备，同时提供地理定位功能。

对于无线通信芯片的选择问题，从性能、成本和功耗方面考虑，RFM 公司的 TR1000 和 Chipcon 公司的 CC1000 是理想的选择。这两种芯片各有所长，TR1000 功耗低，CC1000 灵敏度高、传输距离更远。WeC、Renee 和 Mica 结点均采用 TR1000 芯片；Mica2 采用 CC1000 芯片；Mica3 采用 Chipcon 公司的 CC1020 芯片，传输速率可达 153.6Kb/s，支持 OOK、FSK 和 GFSK 调制方式；Micaz 结点则采用 CC2420 ZigBee 芯片。

另外有一类无线芯片本身集成了处理器，例如 CC2430 是在 CC2420 的基础上集成了 51 内核的单片机；CC1010 是在 CC1000 的基础上集成了 51 内核的单片机，使得芯片的集成度进一步提高。WiseNet 结点就采用了 CC1010 芯片。常见的无线芯片还有 Nordic 公司的 nRF905、nRF2401 等系列芯片。传感器网络结点常用的无线通信芯片的主要参数如表 5.4 所示。

表 5.4　常用短距离无线芯片的主要参数

芯片/参数	频段/MHz	速率 /Kb·s^{-1}	电流/mA	灵敏度/dBm	功率/dBm	调制方式
TR1000	916	115	3	−106	1.5	OOK/FSK
CC1000	300～1000	76.8	5.3	−110	20～10	FSK
CC1020	402～904	153.6	19.9	−118	20～10	GFSK
CC2420	2400	250	19.7	−94	−3	O～QPSK
nRF905	433～915	100	12.5	−100	10	GFSK
nRF2401	2400	1000	15	−85	20～0	GFSK
9Xstream	902～928	20	140	−110	16～20	FHSS

目前市场上支持 ZigBee 协议的芯片制造商有 Chipcon 公司和 Freescale 半导体公司。Chipcon 公司的 CC2420 芯片应用较多，该公司还提供 ZigBee 协议的完整开发套件。Freescale 半导体公司提供 ZigBee 的 2.4GHz 无线传输芯片包括 MC13191、MC13192、MC13193，该公司也提供配套的开发套件。

在无线射频电路设计中，主要考虑以下三个问题。

1）天线设计

在传感器结点设计中，根据不同的应用需求选择合理的天线类型。天线设计是无线收发的一个重要因素，直接关系到无线通信的质量，尤其关系到无线通信的距离和接收信号的质量。天线的设计指标有很多种，无线传感器网络结点使用的是 ISM/SRD 免证使用频段，主要从天线增益、天线效率和电压驻波比三个指标来衡量天线的性能。

天线增益是指天线在能量发射最大方向上的增益，当以各向同性为增益基准时，单位为 dBi；如果以偶极子天线的发射为基准时，单位为 dBd。天线的增益越高，通信距离就越远。当发射机采用高增益的定向天线时，能显著提高通信方向上的功率密度，而接收机采用高增益定向天线时能显著改善信号/噪声比，并提高接收场强，从而大幅度地提高通信距离。

天线效率是指天线以电磁波的形式发射到空中的能量与自身消耗能量的比值，其中自身消耗的能量是以热的形式散发。对于无线结点来说，天线辐射电阻较小，任何电路的损耗都会较大程度地降低天线的效率。

天线电压驻波比主要用来衡量传输线与天线之间阻抗失配的程度。当天线电压驻波比值越高,表示阻抗失配程度越高,则信号能量损耗越大。

在通常情况下,内置天线由于便于携带,且具有免受机械和外界环境损害等优点,常常是设计时的首选方案。这种采用电路板上的金属印刷线作为天线,其优点是不必购买、加工或安装任何形式的元件到电路板,因而成本低。另外这种天线非常薄,可以降低网络结点的体积,缺点是性能较差。如果选用低电阻率的铜材料,由于其厚度太薄,使得串联电阻相对较高,而且低品质的电路板材料也会增加介质损耗。印刷天线的调谐误差通常很大,这是由电路板加工过程的蚀刻误差引起的。

第二种天线是将简单的导线天线或金属条带天线作为元件,安装在电路板上。这种天线因损耗很低,并置于电路板上方,比印刷天线的通信性能有明显提高。导线天线可以是偶极子天线,也可以是环天线。导线天线需要介质材料(如塑料)支撑,从而使其机械外形和相应的谐振频率达到必要的容差要求。它们很难在自动装配线上进行安装,只能人工安装和焊接。总之导线天线是介于低成本、低效率的印刷天线与相对高成本、高效率的外置天线之间的一种很好的折中天线方案。

第三种选择是特殊的陶瓷天线元件。此类天线可以进行自动安装,尺寸比导线天线小,并且不需要调整。但价格比导线天线昂贵,且只在最常使用的频段(如 915MHz 和 2450MHz ISM 波段)才有成品。

外置天线通常没有内置天线那样的尺寸限制,通常离结点中的噪声源的距离较远,因而具有很高的无线通信传输性能。对那些需要尽可能大的距离、必须选用定向天线的应用来说,外置天线几乎是必选的。

2) 阻抗匹配

射频放大输出部分与天线之间的阻抗匹配情况,直接关系到功率的利用效率。如果匹配不好,很多能量会被天线反射回射频放大电路,不仅降低了发射效率,严重时还会导致结点的电路发热,缩短结点寿命。由于传感器结点通常使用较高的工作频率,因而必须考虑导线和 PCB 基板的材质、PCB 走线、器件的分布参数等诸多可能造成失配的因素。

3) 电磁兼容

由于传感器结点体积小,包括微处理器、存储器、传感器和天线在内的各种器件,它们聚集在相对狭小的空间,因而任何不合理的设计都可能带来严重的电磁兼容问题。例如,由于天线辐射造成传感器的探测功能异常,或微处理器总线上的数据异常等。

由于高频强信号是造成电磁兼容的主要原因,所以包括微处理器的外部总线、高速 I/O 端口、射频放大器和天线匹配电路等是电磁兼容设计中考虑的主要因素。

电磁兼容问题容易导致微处理器和无线接收器出现不正常的工作状况。因为微处理器有很多外部引脚,各引脚上的引线通常连接到结点内部的各个部位,受到干扰影响的可能性很大。无线接收器本身就是用于接收电磁信号的,因此如果信号或强信号的高次谐波分量落在接收电路的通带范围内,就可能造成误码和阻塞等问题。

4. 电源模块设计

电源模块是任何电子系统的必备基础模块。对传感器结点来说,电源模块直接关系到传感器结点的寿命、成本、体积和设计的复杂度。如果能够采用大容量电源,那么网络各层

通信协议的设计、网络功率管理等方面的指标都可以降低，从而降低设计难度。容量的扩大通常意味着体积和成本的增加，因此电源模块设计必须首先合理选择电源种类。

众所周知，市电是最便宜的电源，不需要更换电池，而且不必担心电源耗尽。但在具体应用中，市电的应用一方面因受到供电线路的限制，削弱了无线结点的移动性和使用范围；另一方面，用于电源电压转换电路需要额外增加成本，不利于降低结点造价。对于一些市电使用方便的场合，比如电灯控制系统等，仍可以考虑使用市电供电。

电池供电是目前最常见的传感器结点供电方式。按照电池能否充电，电池可分为可充电电池和不可充电电池；根据电极材料，电池可以分为镍铬电池、镍锌电池、银锌电池、锂电池和锂聚合物电池等。一般不可充电电池比可充电电池能量密度高，如果没有能量补充来源，则应选择不可充电电池。在可充电电池中，锂电池和锂聚合物电池的能量密度最高，但是成本比较高；锂聚合物电池是唯一没有毒性的可充电电池。常见电池的性能参数如表 5.5 所示。

表 5.5　常见电池的性能参数

电池类型	铅　酸	镍　镉	镍　氢	锂　离　子	锂聚合物	锂　锰	银　铅
重量能量比 /(W·h·kg^{-1})	35	41	50～80	120～160	140～180	330	
体积能量比 /(W·h·L^{-1})	80	120	100～200	200～280	＞320	550	1150
循环寿命/次	300	500	800	1000	1000	1	1
工作温度/℃	－20～60	20～60	20～60	0～60	0～60	－20～60	20～60
记忆效应	无	有	小	很小	无	无	无
内阻/mΩ	30～80	7～19	18～35	80～100	80～100		
毒性	有	有	轻毒	轻毒	无	无	有
价格	低	低	中	高	最高	高	中
可充电	是	是	是	是	是	否	否
漏电流(%/月)	30	30	15	8	8	20	25

原电池是把化学能转变为电能的装置，它以其成本低廉、能量密度高、标准化程度好、易于购买等特点而备受青睐。例如，我们日常使用的 AA 电池（即通常所说的 5 号电池，尺寸为直径 14mm/高度 49mm）、AAA 电池（即通常所说的 7 号电池，尺寸为直径 11mm/高度 44mm）就是典型的原电池。

虽然使用可充电的蓄电池似乎比使用原电池好，但蓄电池也有缺点，例如它的能量密度有限。蓄电池的重量能量密度和体积能量密度远低于原电池，这就意味着要想达到同样的容量要求，蓄电池的尺寸和重量都要大一些。

另外与原电池相比，蓄电池自放电更严重，这就限制了它的存放时间和在低负载条件下的服务寿命。如果考虑到传感器网络规模庞大，蓄电池的维护成本也不可忽略。尽管有这些缺点，蓄电池仍然有很多可取之处，例如蓄电池的内阻通常比原电池要低，这在要求峰值电流较高的应用中有用途。

传感器结点在某些情况下可以直接从外界的环境获取足够的能量，包括通过光电效应、机械振动等不同方式获取能量。如果设计合理，采用能量收集技术的结点尺寸可以做得很

小，因为它们不需要随身携带电池。最常见的能量收集技术包括太阳能、风能、热能、电磁能和机械能等。

结点所需的电压通常不止一种。这是因为模拟电路与数字电路所要求的最优供电电压不同，非易失性存储器和压电换能器需要使用较高的电源电压。任何电压转换电路都会有固定开销，即消耗在转换电路本身而不是在负载上。对于占空比非常低的传感器结点，这种开销占总功耗的比例可能比较大。

5. 外围模块设计

传感器网络结点的外围模块主要包括看门狗电路、I/O 电路和低电量检测电路等。

看门狗(Watch Dog)是一种增强系统稳健性的重要措施，它能够有效地防止系统进入死循环或者程序跑飞。传感器结点工作环境复杂多变，可能由于干扰造成系统软件的运行混乱。

例如，在因干扰造成程序计数器计数值出错时，系统会访问非法区域而跑飞。看门狗解决这一问题的过程如下：在系统运行以后启动看门狗的计数器，看门狗开始自动计数。如果到达了指定的置位，那么看门狗计数器就会溢出，从而引起看门狗中断，造成系统复位，恢复正常程序流程。为了保证看门狗的正常动作，需要程序在每个指定的时间段内都必须至少置位看门狗计数器一次(俗称“喂狗”)。对于传感器结点而言，可用软件设定看门狗的反应时间。

通常休眠模式下微处理器的系统时钟将停止，由外部事件中断来重新启动系统时钟，从而唤醒 CPU 继续工作。在休眠模式下，微处理器本身实际上已经不消耗电流，要想进一步减小系统功耗，就要尽量将传感器结点的各个 I/O 模块关闭。随着 I/O 模块的逐个关闭，结点的功耗越来越低，最后进入深度休眠模式。需要注意的是，通常在让结点进入深度休眠状态前，需要将重要系统参数保存在非易失性存储器。

另外，由于电池寿命有限，为了避免结点工作中发生突然断电的情况，当电池电量将要耗尽时必须要有某种指示，以便及时更换电池或提醒邻居结点。噪声干扰和负载波动也会造成电源端电压的波动，在设计低电量检测电路时应予考虑。

5.2.3 传感器结点的开发实例

1. Mica 系列结点

Mica 系列结点是由美国加州大学伯克利分校研制，Crossbow 公司生产的无线传感器结点[62]。Crossbow 公司是第一家将智能微尘无线传感器引入大规模商业用途的公司，现在给一些财富百强企业提供服务和智能微尘产品。基于 Crossbow 的无线传感器平台，可实现功能强大、无线和自动化的数据采集和监控系统。它的产品的大部分部件属于即插即用，而且所有的组成部分是靠 TinyOS 操作系统运行的。Mica Processor/Radio boards (MPR)即所谓的 Mica 智能卡板组成硬件平台，它们由电池供能，传感器和数据采集模块与 MPR 集成在一起。

图 5.8 是 Mica 系列结点的组网示意图。Mica 系列结点包括 WeC、Renee、Mica、

Mica2、Mica2Dot～MicaZ。WeC、Renee 和 Mica 结点均采用 TR1000 芯片，Mica2 和 Mica2Dot 采用 CC1000 芯片，Micaz 结点采用 CC2420 ZigBee 芯片。

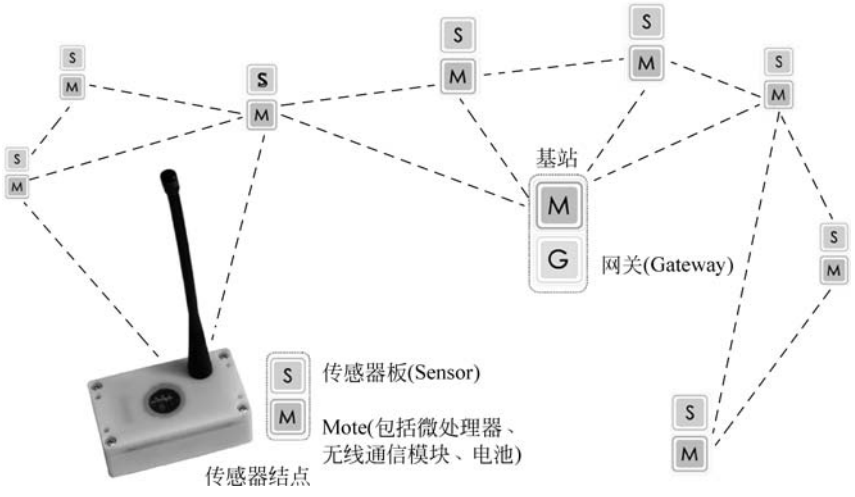

图 5.8　Mica 系列结点的组网示意图

表 5.6 列出了 Mica 系列网关和网络接口板，表 5.7 列出了 Mica 系列数据处理和通信板的型号和功能。

表 5.6　Mica 系列网关和网络接口板

型　号	功　能	Mote 接口	编　程　口	数　据　口
MIB500	并口编程器	MICA、MICA2(51 针连接器) MICA 系列传感板(51 针连接器) MICA2DOT(19 针圆形连接器)	并口	串口 (RS-232)
MIB510	串口编程器	MICA、MICA2(51 针连接器) MICA 系列传感板(51 针连接器) MICA2DOT(19 针圆形连接器)	串口 (RS-232)	串口 (RS-232)
MIB600	Ethernet 编程 接口板	MICA、MICA2(51 针连接器) MICA2DOT(19 针圆形连接器)	Ethernet	Ethernet

表 5.7　Mica 系列数据处理和通信板

Mote 硬件平台		MICAz	MICA2	MICA2DOT	MICA
模块系列		MPR2400	MPR400/410/420	MPR500/510/520	MPR300/310
MCU	芯片	ATMega128L			ATMega103L
	参数	7.37MHz,8 位		4MHz,8 位	4MHz,8 位
	程序存储器容量/KB	128			
	SRAM 容量/KB	4			
传感器板接口	类型	51 脚		18 脚	51 脚
	10 位 A/D	7,0～3V 输入		6,0～3V 输入	7,0～3V 输入
	UART	2		1	2
	其他接口	DIO,I^2C		DIO	DIO,I^2C

续表

Mote硬件平台		MICAz	MICA2	MICA2DOT	MICA
RF收发器	芯片型号	CC2420	CC1000		TR1000
	RF频率/MHz	2400	300～1000		916
	数据速率/(Kb/s)	250	78.6		115
	无线连接器	MMCX		PCB焊孔	
Flash	芯片型号	AT45DB014B			
	通信接口	SPI			
	容量/KB	512			
电源	类型	AA,2×		纽扣电池(CR2354)	AA,2×
	容量/(mA·h)	2000		560	2000
	3.3V升降压转换器	无			有

Crossbow有三种MPR模块,即MICAz(MPR2400)、MICA2(MPR400)和MICA2DOT(MPR500)。MICAz用于2.4GHz ISM频段,支持IEEE 802.15.4和ZigBee协议。MICA2和MICA2DOT可以采用315、433、868/900MHz频段,支持的频率范围多。

在图5.9中,左上所示为MICA2系列MPR4x0的实物,右下为MICA2DOT系列MPR5x0的实物。

图5.10所示为MICAz系列MPR2400的实物。

图5.9 MICA2系列MPR4x0(左上)和MICA2DOT系列MPR5x0(右下)的实物

图5.10 MICAz系列MPR2400的实物

图5.11(a)所示为MICA2多传感器模块MTS300/310,图5.11(b)所示为MICA2DOT多传感器模块MTS510。MTS310多传感器板包含光、温度、传声器、声音探测器、两轴加速计、两轴磁力计等探测设备,与MICA、MICA2相兼容,可以集成在一起使用。MTS510传感器板包含光传感器、传声器、两轴加速计,只与MICA2DOT兼容,它可以应用在有关声音目标的跟踪、机器人技术、地震监测、事件检测和普适计算等应用领域。

图5.12所示为串行网关MIB510,这种编码和串行接口板用作MICA2和MICA2DOT的编程连接器。例如,将MICA2插入到MIB510,具有RS-232串行接口,可以采用交流电供电,通过115Kb串行端口,能将程序快速载入到网络结点。

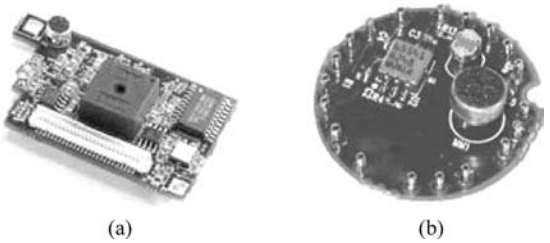

图 5.11　多传感器模块 MTS300/310(a)和 MTS510(b)的实物

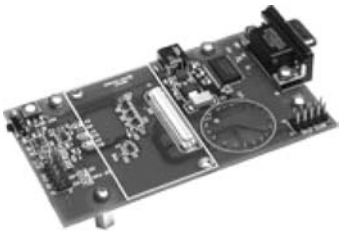

图 5.12　串行网关 MIB510 的实物

图 5.13 所示为 Stargate 网关 SPB400，由母板和子板组成，下层为母板，上层为子板。Stargate 是 Crossbow 公司的一款较为高端的产品，提供了非常齐全的接口，主要作为无线传感器网络的网关结点，负责网络数据的汇聚和与计算机等设备的接口，也可以作为普通传感器结点使用，具有较强的计算与处理能力。

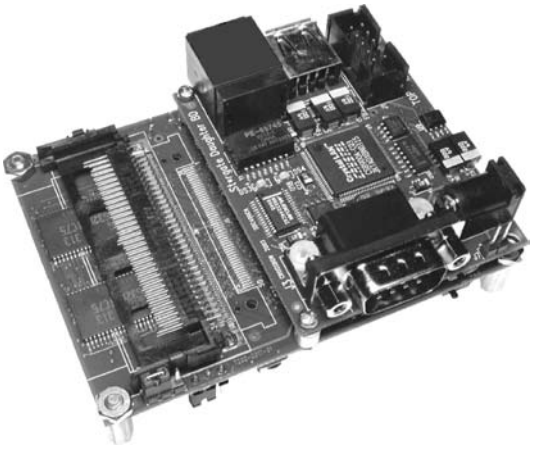

图 5.13　Stargate 网关 SPB400 的实物

Stargate 作为一款高性能的单板机，具有较强的通信和传感器信号处理能力，采用 Intel 公司新一代的 X-Scale 处理器(PXA255)。Stargate 是 Intel 普适计算研发小组的研究成果，授权 Crossbow 生产。Stargate 板有 32MB 的 Intel StrataFlash 用于存储操作系统和应用程序，有 64MB SDRAM 内存足够支持较大的操作系统和应用程序，并可扩展包括串口、USB、以太网接口、WiFi、JTAG 等在内的多种接口形式。

2. Mica 系列处理器/射频板

1) 微处理器电路

Mica 系列产品的处理器均采用 Atmel 公司的 ATmega128L。ATmega128 为基于 AVR RISC 结构的 8 位低功耗 CMOS 微处理器。由于其先进的指令集以及单周期指令执行时间，ATmega128 的数据吞吐率高达 1MIPS/MHz，从而可以缓解系统在功耗和处理速度之间的矛盾。它的主要特点包括如下：

① 先进的 RISC 架构，内部具有 133 条功能强大的指令系统，而且大部分指令是单周期

的；32 个 8 位通用工作寄存器＋外围接口控制寄存器。

② 内部有 128KB 的在线可重复编程 Flash、4KB 的 EEPROM 和 4KB 的 SRAM。

③ 有 53 个 I/O 引脚，每个 I/O 口分别对应输入、输出、功能选择、中断等多个寄存器，使功能口和 I/O 口可以复用，大大增强了端口功能和灵活性，提高了对外围模块的控制能力。

④ 内部有两个 8 位定时器/计数器和两个具有比较/捕捉寄存器的 16 位定时器/计数器；1 个具有独立振荡器的实时计数器；1 个可编程看门狗定时器；2 通道 8 位 PWM 通道；8 路 10 位 A/D 转换器；双向 I^2C 串行总线接口；主/从 SPI 串行接口；可编程串行通信接口；片内精确的模拟比较器等。

⑤ 功耗低。具有 6 种休眠模式：空闲模式、ADC 噪声抑制模式、省电模式、掉电模式、Standby 模式以及扩展的 Standby 模式。

在空闲模式时，CPU 停止工作，SR_AM、T/C、SPI 端口以及中断系统继续工作。在 ADC 噪声抑制模式时，CPU 和所有的 I/O 模块停止运行，而异步定时器和 ADC 继续工作，以降低 ADC 转换时的开关噪声。在省电模式时，异步定时器继续运行，以允许用户维持时间基准，器件的其他部分则处于休眠状态。在掉电模式时，晶体振荡器停止振荡，所有功能除了中断和硬件复位之外都停止工作，寄存器的内容则一直保持。在 Standby 模式时，振荡器工作而其他部分休眠，使得器件只消耗极少的电流，同时具有快速启动能力。在扩展 Standby 模式时，允许振荡器和异步定时器继续工作。

ATmega128L 的软件结构也是针对低功耗而设计的，具有内外多种中断模式。丰富的中断能力减少了系统设计中查询的需要，可以方便地设计出中断程序结构的控制程序、上电复位和可编程的低电压检测。

⑥ 带 JTAG 接口，便于调试。JTAG 仿真器通过该 JTAG 口，可以很方便地实现程序的在线调试和仿真。编译调试正确的代码通过 JTAG 口直接写入 ATmega128 的 Flash 代码区中。

另外，它还支持 Bootloader 功能，即 MCU 上电后，首先通过驻留在 Flash 中的 Boot-Loader 程序，将存储在外部媒介中的应用程序搬移到 ATmega128L 的 Flash 代码区。搬移成功后自动去执行代码，完成自启动。这对于产品化后程序的升级和维护提供了极大的方便。

⑦ 电源电压为 2.7～5.5V，动态范围较大，能够适应恶劣的工作环境。

ATmega128L 的上述特点非常适合传感器结点，尤其是其低功耗特性有利于延长结点寿命。

2）射频板

Mica 结点的无线通信射频芯片均采用 Chipcon 公司的 CCXXXX 系列射频产品。该系列产品是专门为低功耗、低速率的无线传感器网络开发的。例如，MICAz 结点采用了 CC2420 通信芯片。

CC2420 是 Chipcon 公司推出的首款符合 2.4GHz IEEE 802.15.4 标准的射频收发器。该器件是第一款适用于 ZigBee 产品的 RF 器件。CC2420 的选择性和敏感性指数超过了 IEEE 802.15.4 标准的要求，可确保短距离通信的有效性和可靠性。

CC2420 的主要性能参数如下：工作频带范围是 2.400～2.4835GHz，共有 16 个可用信

道，单位信道带宽 2MHz，信道间隔 5MHz；采用 IEEE 802.15.4 规范要求的直接序列扩频方式；数据速率达 250Kb/s，码片速率达 2MChip/s，可以实现多点对多点的快速组网；采用 O-QPSK 调制方式；超低电流消耗（RX：19.7mA，TX：17.4mA）；高接收灵敏度（－94dBm）；抗邻频道干扰能力强（39dB）；内部集成有 VCO、LNA、PA 以及电源整流器；采用低电压供电（2.1～3.6V）；输出功率编程可控；IEEE 802.15.4 MAC 层硬件可支持自动帧格式生成、同步插入与检测、16bit CRC 校验、电源检测和完全自动 MAC 层安全保护（CTR，CBC-MAC，CCM）。

CC2420 只需要极少的外围元器件，外围电路包括晶振时钟电路、射频输入输出匹配电路和微控制器接口电路三部分。芯片本振信号既可由外部有源晶体提供，也可由内部电路提供。由内部电路提供时，须外加晶体振荡器和两个负载电容，电容的大小取决于晶体的频率和输入容抗等参数。射频输入输出匹配电路主要用来匹配芯片的输入输出阻抗，使其输入输出阻抗为 50Ω。

3. Mica 系列传感器板

Mica 系列传感器板是较早实现商用的无线传感器结点部件，它的电路原理图设计是公开的。这里简要介绍部分主要的电路设计内容。

1）传感器电源供电电路

一些传感电路的工作电流较强，应采用突发式工作的方式，即在需要采集数据时才打开传感电路进行工作，从而降低能耗。由于一般的传感器都不具备休眠模式，因而最方便的办法是控制传感器的电源开关，实现对传感器的状态控制。

对于仅需要小电流驱动的传感器，可以考虑直接采用 MCU 的 I/O 端口作为供电电源。这种控制方式简单而灵活，对于需要大电流驱动的传感器，宜采用漏电流较小的开关场效应管控制传感器的供电。在需要控制多路电压时，还可以考虑采用 MAX4678 等集成模拟开关实现电源控制。

2）温湿度和照度检测电路

MTS300CA 使用的温湿度和照度传感器，分别是松下公司的 ERT-J1VR103J 和 Clairex 公司的 CL9P4L。由于温湿度传感器和照度传感器的特性曲线一般不是线性的，因而信号经过 A/D 转换进入到 MCU 后，还需根据器件特性曲线进行校正。

3）磁性传感器电路

磁性传感器可用于车辆探测等场合。在嵌入式设备中，采用的最简单的磁性传感器是霍尔效应传感器。霍尔效应传感器是在硅片上制成，产生的电压只有几十毫伏/特[斯拉]，需要采用高增益的放大器，把从霍尔元器件输出的信号放大到可用的范围。霍尔效应传感器已经把放大器与传感器单元集成在相同的封装中。

当要求传感器的输出与磁场成正比时，或者当磁场超过某一水平时开关要改变状态，此时可以采用霍尔效应传感器。霍尔效应传感器适用于需要知道磁铁距离传感器究竟有多远的场合，最适宜探测磁铁是否逼近传感器。

图 5.14 是采用美国 Honeywell 公司生产的双通道磁性传感器 HMC1002 的参考设计电路。传感器输出经放大后送给两个 A/D 转换器，放大器增益的控制通过 I^2C 总线控制数字电位计（D/A 转换器）的输出电压来实现。HMC1002 的磁芯非常敏感，容易发生饱和现

象，而 MTS300/310 的电路中没有设计自动饱和恢复电路，因而不能直接应用在罗盘等需要直流输出电压信号的应用中。MTS310 的 PCB 上预留了 4 个用于连接外部自动饱和恢复控制电路的引脚。

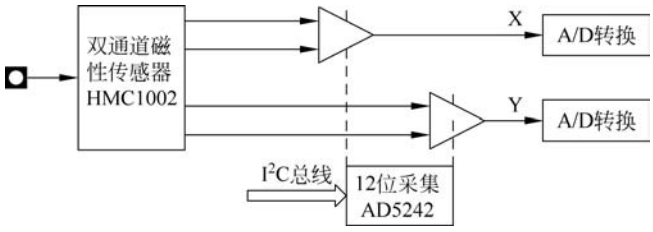

图 5.14 MTS300CA 传感器板的磁阻传感器电路设计框图

4. 编程调试接口板

Mica 系列结点在很大程度上是作为教学和研究试验用途的，人们通过在由多个 Mica 结点组成的实验床验证自己的算法和体验多跳自组网的特性。为了方便开发，Crossbow 公司开发了一系列的编程调试工具，比较常见的是 MIB510 和 MIB600 接口板[63]。

1) MIB300/MIB500/MIB510/MIB520 接口板

MIB300/MIB500 系列接口板是最早开发的编程调试工具，现在这两种开发工具已经由 MIB510 取而代之。使用 MIB510 串行接口板可以编程调试基于 ATMega128 处理器的 MICAz/Mica/Mica2/Mica2DOT 结点。

MIB510 接口板上主要是一些用于连接不同种类 Mica 结点的接口，另外还有一个在系统处理器(In-System Processor，ISP)Atmega16L，它用于编程 Motes 结点。主机下载的代码首先通过 RS-232 串口下载到 ISP，然后由 ISP 编程 Motes 结点。ISP 和 Motes 共享同一个串口，ISP 采用固定的 115.2Kb/s 的通信速率与主机通信。它不断监视串口数据包，一旦发现了符合固定格式的数据包(来自主机的命令)，则立刻关闭 Mote 结点的 RX 和 TX 串行总线，并接管串口，传输或转发调试命令。

MIB510 支持基于 JTAG 口的在线调试。在编程 Motes 结点的过程中，要求主机中安装有 TinyOS 操作系统。有关 TinyOS 的内容请参阅本章的后续内容。

例如，MIB510CA 型号的接口板可以将传感器网络数据汇总，并传输至 PC 或其他计算机平台。任何 IRIS/MICAz/MICA2 结点与 MIB510 连接，均可作为基站使用。除了用于数据传输，MIB510CA 还提供 RS-232 串行编程接口。

由于 MIB510 带有一个板载处理器，可运行 Mote 处理器/射频板。处理器还能监测 MIB510CA 的电源电压，如果电压超出限制，则通过编程将其禁止。

总之，MIB510 的作用在于：①编程接口；②RS-232 串行网关；③可连接 IRIS、MICAz 和 MICA2。它的连线和结点装配如图 5.15 所示。

在使用 MIB510 烧入程序时，需要指明编程板型号和串口号，如：

```
bash % MIB510 = /dev/ttyS1 make install mica2
```

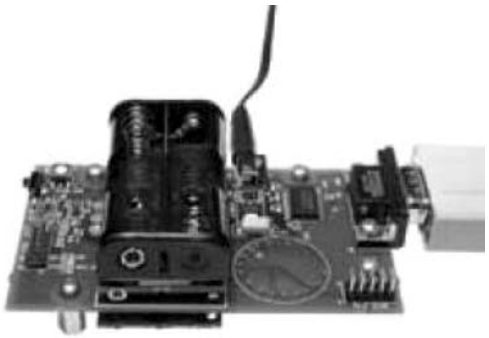

图 5.15　MIB510 的连线和结点的装配

在 Linux 中串口对应的设备文件为/dev/ttyS*，如果是 COM1 口，则对应/dev/ttyS0。如果不知道连接的是哪一个 COM 口，可以用以下命令测试：

```
bash% AT > /dev/ttyS1
```

若该端口不存在，会提示找不到相应的文件。

在 Windows 命令行中也可以使用“mode”命令查看串口信息。

MIB520 采用了 USB 总线与主机连接，使用更加方便。

2）MIB600

MIB600 较之前面的 MIBXXX 接口板的主要区别在于，它提供了与以太网直接互联的能力，即 MIB600 可作为以太网和 Motes 网络之间的网关。MIB600 的另一个特点是实现了局域网接口供电协议，在连接支持供电功能的交换机时可以直接从交换机取电。

由于具有上述两个特点，MIB600 不仅可以作为一般的编程接口板使用，而且可以通过以太网对远程结点配置、编程、收集数据或调试，大大方便了开发过程。另外，MIB600 可以配合 Mica2 结点，直接作为无线传感器网络的汇聚结点（Sink）使用。

5.3　操作系统和软件开发

5.3.1　网络结点操作系统

1. 网络结点操作系统的设计要求

这里先对常见的操作系统、嵌入式系统、嵌入式操作系统的概念进行简单介绍。

通常操作系统（Operating System，OS）是指电子计算机系统中负责支撑应用程序运行环境和用户操作环境的系统软件。它是计算机系统的核心与基石，职责包括对硬件的直接监管、对各种计算资源（如内存、处理器时间等）的管理，以及提供诸如作业管理之类的面向应用程序的服务等。在操作系统的帮助下，用户使用计算机时，避免了对计算机系统硬件的直接操作。对计算机系统而言，操作系统是对所有系统资源进行管理的程序的集合；对用户而言，操作系统提供了对系统资源进行有效利用的简单抽象的方法。人们将安装了操作

系统的计算机称为虚拟机(Virtual Machine,VM),认为是对裸机的扩展。

嵌入式系统是指用于执行独立功能的专用计算机系统。它由微处理器、定时器、微控制器、存储器、传感器等一系列微电子芯片与器件,以及嵌入在存储器中的微型操作系统、控制应用软件组成,共同实现诸如实时控制、监视、管理、移动计算、数据处理等各种自动化处理任务。

嵌入式操作系统是一种支持嵌入式系统应用的操作系统软件,它是嵌入式系统的重要组成部分。嵌入式操作系统具有通用操作系统的基本特点,能够有效管理复杂的系统资源,并且把硬件虚拟化。

传感器网络结点作为一种典型的嵌入式系统,同样需要操作系统来支撑它的运行。传感器网络结点的操作系统是运行在每个传感器结点上的基础核心软件,它能够有效地管理硬件资源和任务的执行,并且使应用程序的开发更为方便。传感器网络操作系统的目的是有效地管理硬件资源和任务的执行,并且使用户不用直接在硬件上编程开发程序,从而使应用程序的开发更为方便。这不仅提高了开发效率,而且能够增强软件的重用性。

但是,传统的嵌入式操作系统不能适用于传感器网络,这些操作系统对硬件资源有较高的要求,传感器结点的有限资源很难满足这些要求。

由于传感器网络操作系统的设计是面向具体应用的,这是与传统操作系统设计的主要区别。在设计传统操作系统时,一般要求操作系统为应用开发提供一些通用的编程接口,这些接口是独立于具体应用的,只需调用这些编程接口就能开发出各种各样的应用程序。这样设计出来的传统操作系统需要实现复杂的进程管理和内存管理等功能,因而对硬件资源有较高的要求。只有针对具体应用,才能量体裁衣地开发出对硬件资源要求最低的操作系统。

根据传感器网络的特征,通常设计操作系统时需要满足如下要求:

(1) 由于传感器结点只有有限的能量、计算和存储资源,它的操作系统代码量必须尽可能小,复杂度尽可能低,从而尽可能降低系统的能耗。

(2) 由于传感器网络的规模可能很大,网络拓扑动态变化,操作系统必须能够适应网络规模和拓扑高度动态变化的应用环境。

(3) 对监测环境发生的事件能快速响应,迅速执行相关的处理任务。

(4) 能有效地管理能量资源、计算资源、存储资源和通信资源,高效地管理多个并发任务的执行,使应用程序能快速切换并执行频繁发生的多个并发任务。

(5) 由于每个传感器结点资源有限,有时希望多个传感器结点协同工作,形成分布式的网络系统,才能完成复杂的监测任务。传感器网络操作系统必须能够使多个结点高效地协作完成监测任务。

(6) 提供方便的编程方法。基于传感器网络操作系统提供的编程方法,开发者能够方便、快速地开发应用程序,无须过多地关注对底层硬件的操作。

(7) 有时传感器网络部署在危险的不可到达区域,某些应用要求对大量的传感器结点进行动态编程配置。在这种情况下,操作系统能通过可靠传输技术对大量的结点发布代码,实现对结点在线动态重新编程。

2. TinyOS 操作系统介绍

TinyOS 是一个开源的嵌入式操作系统，它是由加州大学伯克利分校开发，主要应用于无线传感器网络方面。它是一种基于组件（Component-Based）的架构方式，能够快速实现各种应用[64]。TinyOS 程序采用的是模块化设计，程序核心往往都很小。一般来说，核心代码和数据大概在 400B 左右，能够突破传感器存储资源少的限制，使得 TinyOS 可以有效地运行在无线传感器网络结点上，并负责执行相应的管理工作。

TinyOS 本身提供了一系列的组件，可以很方便地编制程序，用来获取和处理传感器的数据，并通过无线方式来传输信息。我们可以把 TinyOS 看成一个与传感器进行交互的 API 接口，它们之间能实现各种通信。

在构建无线传感器网络时，TinyOS 通过一个基地控制台即网关汇聚结点，来控制各个传感器子结点，并聚集和处理它们所采集到的信息。TinyOS 只要在控制台发出管理信息，然后由各个结点通过无线网络互相传递，最后达到协同一致的目的。

1）TinyOS 的安装

TinyOS 软件包是开放源代码的，用户可以从网站 http://www.tinyos.net 下载。下面介绍 1.1.0 版本的 TinyOS 软件包的安装过程，其他版本的 TinyOS 软件包的安装与此类似。

如果在 Windows 2000/XP 上安装，可下载 tinyos-1.1.0-lis.exe，按照提示逐步执行，就能自动完成安装，然后在 Cygwin 环境下操作命令。

Cygwin 是一个在 Windows 平台上运行的 Linux 模拟环境，是 Cygnus Solutions 公司开发的自由软件。人们通过 Cygwin 可以很容易地远程登录到任何一台 PC，在 UNIX/Linux 外壳下解决问题，在任何一台 Windows 操作系统的计算机上运行外壳脚本命令。高级外壳脚本命令可以用标准 shell、sed 和 awk 等创建。标准 Windows 命令行工具甚至可以与 UNIX/Linux 外壳脚本环境共同管理 Windows 操作系统。

Linux 作为一种计算机操作系统，它是自由软件和开放源代码发展中最著名的代表。如果在 Linux RedHat 9.0 上安装，需要手动安装软件包，执行步骤如下。

① 下载和安装 IBMs 1.4JDK 和 javax.comm rpITIs 软件包，选择与 Intel 兼容的 IBM SDK for 32bit xSeries，注册并且下载 IBMJava2-SDK 和 IBMJava2-JA-VACOMM rpms，然后安装。

② 从 http://webs.cs.berkeley.edu/tos/dist-1.1.0/tools/linux 和 http://webs.cs.berkelev.edu/tos/dist-1.1.0/tinyos/linux 下载以下软件包：

```
avarice-2.0.20030825 CVS-1.i386.rpm
avr-binutils-2.13.2.1-1.i386.rpm
avr-gcc-3.3tinyos-1.i386.rpm
avr-insight-pre6.0cvs.tinyos-1.3.i386.rpm
avr-libe-20030512cvs-1.i386.rpm
graphviz-1.10-1.i386.rpm
nesc-1.1-1.i386.rpm
tinyos-tools-1.1.0-1.i386.rpm
```

然后用“rpm-ivh *.rpm”命令将这些工具包安装在保存它们的目录。

③ 通过命令“chmod 666/dev/ttyS * ”和“chmod 666/dev/parport0”,更改 TinyOS 将会使用的串口的权限。

④ 从网址 http://webs.cs.berkeley.edu/tos/dist-1.1.0/tinyos/linux/tinyos-1.1.0-1.noarch.rpm 下载 tinyos rpm 软件包,用以下命令将 TinyOS 安装到 TINYOSDIR/tinyos-1.x 目录:

```
rpm -ivh --prefix TINYOSDIR tinyos-1.1.0-1.noarch.rpm
cd TINYOSDIR; chown -R USER.GROUP tinyos-1.x
```

在 PC 的各种操作系统下安装完成 TinyOS 软件之后,还可以对软件进行升级。从 http://webs.cs.berkeley.edu/tos/dist-1.1.0/tinyos/linux 下载得到在 Linux 操作系统中升级 TinyOS 的 rpm 软件包,从 http://webs.cs.berkeley.edu/tos/dist-1.1.0/tinyos-/windows 可以下载到在 Windows 操作系统中升级 TinyOS 的 rpm 软件包。在下载这些软件包之后,可以用 rpm -Uvh <rpm file name>安装这些软件包,从而完成升级 TinyOS。

如果要检测 TinyOS 的环境是否搭建好,可以运行 tos-check-env 命令:

```
$ tos check-env
```

系统会检测各个程序是否正常,如果最后出现正确的提示,则表明 PC 上的 TinyOS 操作系统已经可以使用了。

2) 创建应用程序

在安装 TinyOS 后,可以在 apps 目录下创建应用程序目录,用来存放应用程序文件。例如,可以在 apps 目录下创建 Blink 目录用来存放 Blink 程序的文件。用户为应用程序设计的每个组件都要独立地包含在单个文件中,而且文件名取为“组件名.nc”。例如,Blink 程序包含 Blink 和 BlinkM 两个组件,Blink 组件包含在 Blink.nc 文件中,而 BlinkM 组件包含在 BlinkM.nc 文件中。这些文件可以用任何文本编辑软件来创建。

TinyOS 操作系统最初是用 C 语言实现的,产生的目标代码比较长。后来研究设计出基于组件化和并行模型的 nesC 语言,产生的目标代码相对较小。用 nesC 语言可开发 TinyOS 操作系统和其上运行的应用程序。

3) TinyOS 的特点

TinyOS 的主要特点如下。

① 采用基于组件的体系结构,这种体系结构已经被广泛应用在嵌入式操作系统中。组件就是对软、硬件进行功能抽象。整个系统由组件构成,通过组件提高软件重用度和兼容性,程序员只关心组件的功能和自己的业务逻辑,而不必关心组件的具体实现,从而提高编程效率。

在 TinyOS 这种体系结构中,操作系统用组件实现各种功能,只包含必要的组件,提高了操作系统的紧凑性,减少了代码量和占用的存储资源。通过采用基于组件的体系结构,系统提供一个适用于传感器网络开发应用的编程框架,在这个框架内将用户设计的一些组件和操作系统组件连接起来,构成整个应用程序。这使用户能够方便地构建应用程序。

② 采用事件驱动机制,能够适用于结点众多、并发操作频繁发生的无线传感器网络应用。当事件对应的硬件中断发生时,系统能够快速地调用相关的事件处理程序,迅速响应外

部事件，并且执行相应的操作处理任务。事件驱动机制可以使 CPU 在事件产生时迅速执行相关任务，并在处理完毕后进入休眠状态，有效提高了 CPU 的使用率，节省了能量。

③ 采用轻量级线程技术和基于先进先出(First In First Out，FIFO)的任务队列调度方法。轻线程主要是针对结点并发操作可能比较频繁，且线程比较短，传统的进程/线程调度无法满足的问题提出的，因为使用传统调度算法会在无效的进程互换过程中产生大量能耗。

由于传感器结点的硬件资源有限，而且短流程的并发任务可能频繁执行，所以传统的进程或线程调度无法应用于传感器网络的操作系统。轻量级线程技术和基于 FIFO 的任务队列调度方法，能够使短流程的并发任务共享堆栈存储空间，并且快速地进行切换，从而使 TinyOS 适用于并发任务频繁发生的传感器网络应用。当任务队列为空时，CPU 进入休眠状态，外围器件处于工作状态，任何外部中断都能唤醒 CPU，这样可以节省能量。

④ 采用基于事件驱动模式的主动消息通信方式，这种方式已经广泛用于分布式并行计算。主动消息是并行计算机中的概念。在发送消息的同时传送处理这个消息的相应处理函数和处理数据，接收方得到消息后可立即进行处理，从而减少通信量。由于传感器网络的规模可能非常大，导致通信的并行程度很高，传统的通信方式无法适应这样的环境。TinyOS 的系统组件可以快速地响应主动消息通信方式传来的驱动事件，有效提高 CPU 的使用率。

4) TinyOS 的应用程序示例

在介绍具体应用程序示例之前，我们先重点介绍这里的两个基本概念：接口和组件。

接口(interface)是一个双向通道，表明接口具有的功能和事件通知能力是双向的，向调用者提供命令和实现命令者进行事件通告。例如下面是一个接口的例子：

```
interface NAME {
    asy commandresult_t CNAME(pram p);
    asy eventresult_tENAME(pram p);
}
```

在接口中声明命令和事件实现不同的功能，命令是接口具有的功能，事件是接口具有通告事件发生的能力。asy 可以在命令或事件中断处理程序中调用。

接口体现事件驱动功能和模块化。通过事件通告让使用接口者对事件进行响应；任何满足接口功能的实现者都可被其他需要这个接口功能的组件调用。

组件是配线文件或模块文件，是逻辑功能的抽象。程序员完全可直接调用组件进行程序开发。配线文件只是完成组件之间的接口连接，模块文件则具体实现接口中的命令和事件。

在这两个文件中都可使用 provides、uses 语句。provides 表明这个组件可以提供哪些接口，实现这些接口的命令和事件通知。uses 表明这个组件使用哪些接口，组件能以接口中提供的命令和实现对接口中事件进行响应。基于组件的思想，一个组件可通过多个组件实现一定的逻辑功能，对外声明需要哪些接口和提供哪些接口。

这里以 TinyOS 自带的一个简单应用程序 Blink 为例，说明基于 TinyOS 的应用程序结构、运行机理、编译和连接的过程，以及它的调度机制和事件驱动机制的实现。

Blink 程序包含两个文件 BlinkM.nc 和 Blink.nc，它们是由 TinyOS 软件包 1.1.0 版本的 apps/blink 目录下 Blink 程序修改而成。BlinkM.nc 文件包含了 BlinkM 模块的代码，而 Blink.nc 文件包含了 Blink 配件的代码。

下面首先分析 Blink 程序的配件和模块的代码，然后介绍 ncc 编译 nesC 程序的过程，以及 Blink 程序的运行过程，最后介绍 TinyOS 的调度机制和事件驱动机制的实现。

(1) Blink 程序的配件。

Blink 程序的 Blink 配件的代码如下：

```
configuration Blink{
    //Blink 配件没有提供或使用任何接口
}
implementation{
//Blink 配件包含了 Main 配件、BlinkM 模块、TimerC 配件
components Main,BlinkM,TimerC;
  // Main 使用的 StdControl 接口实现
  // 由 TimerC 和 BlinkM 提供的 StdControl 接口实现
// BlinkM 使用的 Timer 接口由 TimerC 提供的 Timer[unique("Timer")]接口实现
Main. StdControl -> TimerC. StdControl;
Main. StdControl -> BlinkM. StdControl;
BlinkM. StdControl -> TimerC. Timer[unique("Timer")];
}
```

这里 nesC 语句 A. C→B. C 代表 A 使用的接口 C 是由 B 提供的。在将 nesC 语句预编译为 C 语句时，该连接符会被转化为命令处理函数的调用。Blink 配件描述了 Blink 程序的整体结构，如图 5.16 所示。组件上部包含由该组件提供的接口，组件下部包含由该组件使用的接口，向下箭头代表调用命令处理程序，向上箭头代表触发事件处理程序。

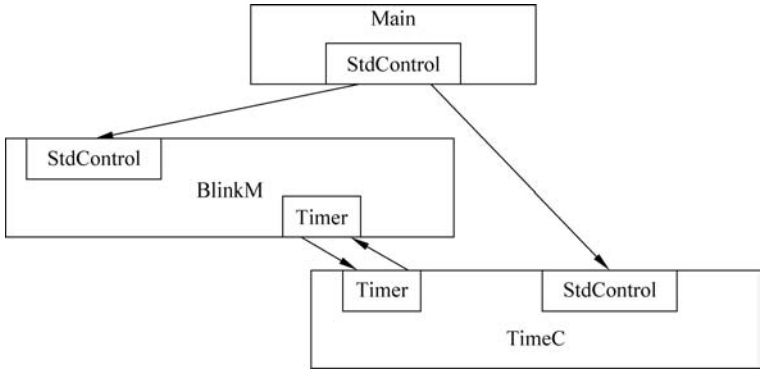

图 5.16 Blink 程序的结构

Main 配件和 TimerC 配件的实现细节如下。

① Main 配件在 Blink 程序的运行过程中发挥重要的作用。文件 tos/system/Main. nc 给出了 Main 配件的代码：

```
configuration Main{
                    uses interface StdControl;
                }
implementation
{
    // Main 配件包含 RealMain 模块、PotC 配件、HPLinit 模块；
components RealMain,PotC,HPLinit;
```

```
StdControl = RealMain. StdControl;
RealMain.hardwareInit -> HPLinit;      // 将 HPLinit.init 缩写为 HPLinit
RealMain. Pot -> Pot;                  // 将 PotC. Pot 缩写为 PotC
```

在语句 StdControl= RealMain. StdControl 中，连接符“=”表示如果给 Main 使用的 StdControl 接口指定了一个实现，那么 RealMain 使用的 StdControl 接口会采用相同的实现。这里使用 nesC 语句 A. C=B. C，代表接口 C 在模块 A 中的使用或实现等同于在模块 B 中的使用或实现，而且接口 C 被模块 A 和 B 同时使用或者同时提供，否则会产生编译错误。

Main 配件包含了 PotC 配件，文件 tos/system/PotC. nc 给出了 PotC 配件的代码：

```
configuration PotC{
                    provides interface Pot;
                  }
implementation{
    // PotC 配件包含 PotM 模块、HPLPotC 模块
components PotM,HPLPotC;
Pot = PotM;   // 这是 PotC. Pot = PotM. Pot 的缩写
PotM. HPLPot -> HPLPotC; // HPLPotC 为 HPLPotC. HPLPPot 的缩写
}
```

图 5.17 所示为 Main 配件的结构。

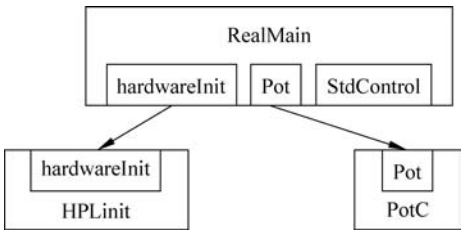

图 5.17 Main 配件的结构框图

② 文件 tos/system/TimerC. nc 给出了 TimerC 配件的代码：

```
configuration TimerC{
                    // 提供 StdControl 接口和参数化的 Timer 接口
                    provides interface Timer[unit8_id];
                    provides interface StdControl;
                  }
implementation{
    // TimerC 配件包含 TimerM 模块、ClockC 配件、NoLeds 模块和 HPLPowerMangement 模块
    components TimerM,ClockC,NoLeds,HPLPowerMangementM;
    TimerM. Leds -> NoLeds;
    TimerM. Clock -> ClockC;
    TimerM. PowerMangement -> HPLPowerMangementM;
    StdControl = TimerM; //这是 TimerC. StdControl = TimerM. StdControl 的缩写
    Timer = TimerM; //这是 TimerC. Timer = TimerM. Timer 的缩写
    TimerC 配件包含 ClockC 配件，文件 tos/system/ClockC.nc 给出了 ClockC 配件的代码：
configuration ClockC {
                      provides interface Clock;
```

```
                provides interface StdControl;
            }
implementation
{
    // ClockC 配件包含 HPClock 模块
    components HPClock;
    TimerM.Leds -> NoLeds;
    Clock = HPClock;          //这是 Clock. Clock = HPClock. Clock 的缩写
    StdControl = HPClock;     //这是 ClockC. StdControl = HPClock. StdControl 的缩写
}
```

图 5.18 所示为 TimerC 配件的结构。

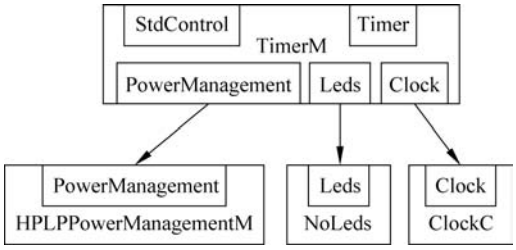

图 5.18 TimerC 配件的结构

(2) BlinkM 模块。

BlinkM 模块实现了 Blink 程序的主要功能,它的代码如下:

```
module BlinkM{
      provides{
    // BlinkM 提供 StdControl 接口,其他组件可以调用这个 StdControl 接口
    // StdControl 接口被声明为
    //   interface StdControl {
    //     command result_t_init();
    //     command result_t_start();
    //     command result_t_stop();
    //                       }
    //根据 nesC 语言规范,BlinkM 实现 StdControl 接口中的三个命令处理程序
    interface StdControl;
              }
      uses{
    //BlinkM 使用了 Timer 接口
    // Timer 接口被声明为
    //     interface Timer{
    //         command result_t statrt(char type,unit32_t interval);
    //              command result_t stop();
    //              event result_t fired();
    //                    }
    // 根据 nesC 语言规范,BlinkM 实现 Timer 接口中的 fired()事件处理程序
    // BlinkM 可以调用 Timer 接口中的两个命令处理程序
          interface Timer;
       }
    }
```

```
implementation {
// 以下代码实现了 StdControl 接口中的 init()命令处理程序
// 关键字 command 定义了一个命令处理程序
// result_t 是一个数据类型,在 tos/system/tos.h 中被定义为 unit8_t
// success 是一个枚举型数据,在 tos/system/tos.h 中被定义为
//  enum{FALL = 0,SUCCESS =1};
command result_t StdControl.init(){
                                  return SUCCESS;
                                }
// 以下代码实现了 StdControl 接口中的 start()命令处理程序
// 直接调用 Timer 接口的 start()设置并且开启定时器,call 代表调用
// TIMER_REPEAT 代表重复定时,TIMER_ONE_SHOT 则表示一次定时
// 定时的事件间隔是 1000 个单位事件
    command result_t StdControl.start() {
      return call Timer.start(TIMER_REPEAT,1000);
                                          }
// 以下代码实现了 StdControl 接口中的 stop()命令处理程序
// 直接调用 Timer 接口中的 stop()停止定时器
    command result_t StdControl.stop() {
                                return call Timer.stop();
                                          }
// 以下代码实现了 Timer 接口的 fired()事件处理程序
// 关键字 event 定义了一个事件处理程序
    event result_t Timer.fired() {
                                return SUCCESS;
                                    }
                }
```

(3) ncc 编译 nesC 程序。

ncc 可以将 nesC 语言编写的程序编译成可执行文件。ncc 是在 gcc 的基础上修改和扩充而来的,ncc 首先将 nesC 程序预编译为 C 程序,然后用交叉编译器将 C 程序编译成为可执行文件。

在编译 nesC 程序时,ncc 的输入通常是描述程序整体结构的顶层配件文件,例如包含 Blink 配件的 Blink.nc 文件。ncc 首先装入 tos.h 文件,该文件包含了基本的数据类型定义和一些基本函数；然后 ncc 装入需要的 C 文件、接口或者组件。具体步骤如下：

① 在装入 C 文件时,先定位 C 文件,然后进行预处理,例如展开宏定义和包含的头文件。

② 在装入接口时,先定位接口,然后装入该接口包含的头文件。

③ 在装入组件时,先定位组件,然后递归装入该组件使用的其他组件和相关文件。

④ ncc 装入顶层配件文件。

在将 nesC 语言程序预编译为 C 语言程序时,ncc 按照下面的规则,将 nesC 语言程序中的标识符转化为 C 语言程序中的标识符：

① 如果 C 文件包含的标识符与 nesC 的关键字相同,则在该标识符前加上前缀_nesC_keyworQ_；否则,标识符保持不变。例如,C 文件定义了变量 module,预编译后该变量被转化为_nesC_keyword_module。

② 组件 C 中的变量 V 被转化为 C$V。

③ 组件 C 中的函数 F 被转化为 C$F。

④ 组件 C 中的命令或事件 A 被转化为 C$A。

⑤ 组件 C 中的接口 I 中的命令或事件 A 被转化为 cIA。

(4) 应用程序导入结点。

通过仿真调试确认应用程序确实能够执行指定任务后,可以将应用程序编译成为在实际结点硬件上运行的可执行代码。TinyOS 支持多种硬件平台,每个硬件平台对应的文件存放在目录 tos/platform 内对应该硬件平台的子目录。

在应用程序所在的目录输入"make 平台名称",就能编译出运行在该平台的可执行代码。例如,在 apps/blink 目录中输入 make mica2,就能编译出在 mica2 平台上的可执行代码 main.exe。

make 命令调用 ncc 执行编译任务。ncc 提供了一些选项,常用的选项包括:

```
-target =X            //指定硬件平台
-tosdir =dir          //指定 TinyOS 目录
-fnesC -file =file    //指定存放预编译生成的 C 代码的文件
```

至此,我们就把 Blink.nc 编译为在 mica 平台上运行的可执行文件 main.exe。但是此时还不能把 main.exe 载到结点中,必须先把 main.exe 转化为可下载的机器码。

硬件平台可能采用特定格式的机器码,例如 mica 平台采用 Motorola 公司定义的 srec 格式的机器码,而其他平台采用 Intel 公司定义的 hex 格式的机器码。srec 是摩托罗拉定义的 s-record 格式的二进制文件,另外一种常见的格式是 Intel 的 hex 格式文件。如果使用其他一些只支持 hex 的编程器(如 avr-studio)对目标系统编写程序,则需要转换成 hex 格式文件。

以下命令可以将 main.exe 转化为 srec 格式的机器码 main.srec:

```
avr -objecopy -output -target = srec main.exe main.srec
```

这时可以通过下载工具把 main.srec 下载到传感器网络结点中,这里是通过 uisp 下载的,可以分为三步进行:

① 擦除结点的 Flash 存放的原始代码;

② 把 main.srec 下载到结点的 Flash 中;

③ 验证写入程序和原始文件是否一致。

Crossbow 公司为 MICAz 结点提供了 MIB500、MIB510 和 MIB600 三种编程板,下面我们以 MIB510(串口)为例说明程序烧录过程。

首先生成 Blink 程序的原始二进制代码(为 Intel 的 hex 格式或 Motorola 的 s-record 格式):

```
make micaz Blink
```

然后使用 uisp 工具将 Blink 程序烧录到结点的 Flash:

```
uisp -dprog =mib510 -dserial =COM -dpart =Atmega128 --erase --upload if =Blink.srec
```

下面是通过 uisp 下载的分步操作过程和程序烧录的各阶段结果演示:

```
uisp  -dprog = dapa -erase
```

```
pulse
Ateml AVR ATmega128 is found.
Erasing device …
pulse
Reinitializing device
Atmel AVR ATmega128 is found.
sleep 1
uisp - dprog = dapa -- upload if = build/mica.srec
pulse
Atmel AVR ATmega128 is Found.
Uploading: Flash
sleep 1
uisp - dprog = dapa -- upload if = build/main.srec
pulse
Atmel AVR ATmega128 is Found.
Verifying: Flash
```

5.3.2 软件开发

1. 传感器网络软件开发的特点和要求

传感器网络在环境监测、医疗监护、军事、家庭娱乐、工业、教育等领域的应用日趋广泛。在这些多样化的应用中，各类应用系统或中间件系统都是针对某类特定应用和特定环境的，开发传感器网络应用程序需要一定的周期。

传感器网络结点的软件系统用于控制底层硬件的工作行为，为各种算法、协议的设计提供一个可控的操作环境，同时便于用户有效管理网络，实现网络的自组织、协作、安全和能量优化等功能，从而降低传感器网络的使用复杂度。通常传感器网络的软件运行采用分层结构，如图 5.19 所示。

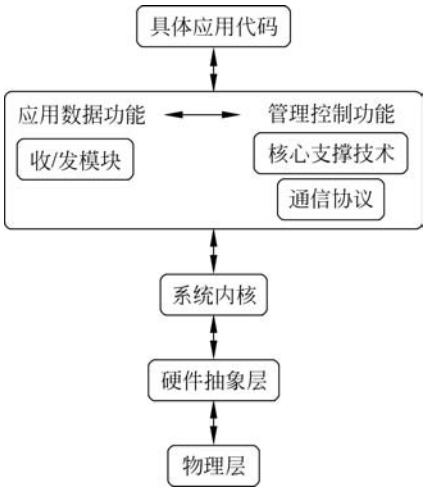

图 5.19　传感器网络结点软件系统的分层结构

这里硬件抽象层在物理层之上，用来隔离具体硬件，为系统提供统一的硬件接口，诸如初始化指令、中断控制、数据收发等。系统内核负责进程调度，为应用数据功能和管理控制功能提供接口。应用数据功能协调数据收发、校验数据，并确定数据是否需要转发。管理控制功能实现网络的核心支撑技术和通信协议。在编写具体应用代码时，我们要根据应用数据功能和管理控制功能提供的接口和一些全局变量来设计。

传感器网络因资源受限、动态性强和以数据中心，网络结点的软件系统开发设计具有如下特点。

(1) 具有自适应功能。由于网络变化不可预知，软件系统应能够及时调整结点的工作状态，设计层次不能过于复杂，且具有良好的事件驱动与响应机制。

(2) 保证结点的能量优化。由于传感器结点的电池能量有限，设计软件系统时尽可能考虑节能，用比较精简的代码或指令来实现网络的协议和算法，并采用轻量级的交互机制。

(3) 采用模块化设计。为了便于软件重用，保证用户根据不同的应用需求快速进行开发，将软件系统的设计模块化，让每个模块完成一个抽象功能，并制定模块之间的接口标准。

(4) 面向具体应用。软件系统面向具体的应用需求进行设计开发，运行性能满足用户的要求。

(5) 具有维护和升级功能。为了维护和管理网络，软件系统宜采用分布式的管理办法，通过软件更新和重配置机制来提高系统运行的效率。

2. 网络系统开发的基本内容

传感器网络软件开发的本质是从软件工程的思想出发，在软件体系结构设计的基础上开发应用软件。我们通常需要使用基于框架的组件，来支持传感器网络的软件开发。

这种框架运用自适应的中间件系统，通过动态地交换和运行组件，支撑起高层的应用服务架构，从而加速和简化应用系统的设计与开发。传感器网络软件设计的主要内容就是开发这些基于框架的组件，主要包括以下三方面的环节。

(1) 传感器应用。这种应用负责提供必要的传感器结点本地基本功能，包括数据采集、本地存储、硬件访问和直接存取操作系统等。

(2) 结点应用。这种应用包含针对专门应用的任务和用于建立与维护网络的中间件功能，它涉及操作系统、传感驱动和中间件管理三部分。结点应用层次的框架组件如图 5.20 所示。这里的各组件功能简介如下。

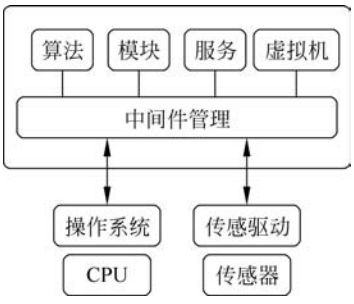

图 5.20 结点应用框架的组件

操作系统组件：由裁剪过的只针对特定应用的软件组成，专门处理与结点硬件设备相关的任务，包括启动载入程序、硬件初始化、时序安排、内存管理和过程管理等。

传感驱动组件：负责初始化传感器结点，驱动结点上的传感单元执行数据采集和测量任务，由于它封装了传感器探测功能，可以为中间件提供良好的 API 接口。

中间件管理组件：作为一个上层软件，用来组织分布式结点间的协同工作。

模块组件：负责封装网络应用所需的通信协议和核心支撑技术。

算法组件：用来描述模块的具体实现算法。

服务组件：负责与其他结点协作完成任务，提供本地协同功能。

虚拟机组件：负责执行与平台无关的一些程序。

(3) 网络应用。这种应用的设计内容描述了整个网络应用的任务和所需要的服务，为用户提供操作界面，管理整个网络并评估运行效果。网络应用层次的框架组件结构如图 5.21 所示。

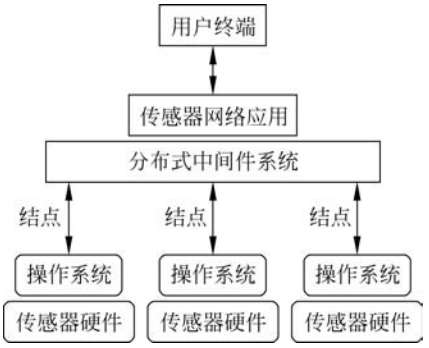

图 5.21　网络应用框架的组件

网络中的结点通过中间件的服务连接起来，协作地执行任务。中间件逻辑上是在网络层，但物理上仍存在于结点内，它在网络内协调服务间的互操作，灵活便捷地支撑起无线传感器网络的应用开发。

通常人们需要依据上述三个环节的应用，通过程序设计来开发实现各类组件，这也是传感器网络软件设计的主要内容。

3. 传感器网络的软件编程模式

传感器网络的软件开发需要采取一定的编程模式，运用适当的编程框架来指导具体的程序设计。通用软件的编程模式并不完全适合于传感器网络的软件开发，为此需要考虑设计适合于传感器网络开发特征的编程模式，这里主要简介 3 种常见的编程模式。

1) 抽象域编程

抽象域编程方式将网络底层的通信机制、数据采集、数据共享和路由机制等细节屏蔽起来，为用户提供基于本地域的高层程序接口，以便简化应用操作。采用这种编程模式的作用在于，使用户能够方便地控制网络的资源消耗、通信机制和探测精度，适应网络的变化状况。

抽象域为结点之间定义了“邻居”关系，这里具有邻居关系的结点是指包括 N 跳内的结点集合或者是距离小于某个确定值的结点集合等。每一个结点可以属于多个抽象域，通过初始化操作来发现邻居。在某个域内随着结点的加入或离开，这些变化情况要通知给该域内的每个结点。

抽象域还支持枚举操作(即返回参与该域的所有结点)和数据共享操作(即变量或数值可以在域内结点间共享)，以及约简操作(即通过一定的计算如求和、最值等来减少域内存储的数据)。

以 TinyOS 操作系统为例，它的并发模型要求实现并发的“执行上下文”，将应用程序分割成多个任务来执行。TinyOS 的同步操作并不要求全线程机制，而是用事件驱动代码来支持有限范围的模块操作，以便处理异步事件。抽象域编程使用 TinyOS 轻量级的、线程化

的同步模型，以缺省的、事件驱动的上下文来支持单个模块的执行上下文。它采用模块化接口来简化应用设计，并非将应用拆分为一系列不同的事件句柄，而是采用简单的循环来编写代码。

2）以对象为中心的编程

针对传感器网络自身状况和外在环境动态多变的特点，以对象为中心的编程模式给编程者提供高级别的网络抽象，用“对象”来表示网络所要监测的物理现象，采用面向服务的方法来分析对象的活动行为。

这种编程框架主要包括四个部件。

① 终端用户编程接口。该接口负责将服务请求注入网络，终端用户使用记录在服务储存库里的服务写入一个应用。服务储存库类似于 Web Service 所使用的“统一描述、发现和集成协议”(UDDI)服务器，它负责保存所有在网络中发现的服务登记，输出服务请求。

② 服务规划。该组件将服务作为输入，产生一个由服务储存库里的服务所组成的服务图。通常一个服务请求被扩展成多个子服务，直到没有更多的服务需要被扩展。

例如车辆跟踪的应用场景，车辆跟踪服务需要将车辆类型和位置数据作为输入，车辆分类服务提供分类和位置数据，但需要在传感设备感知到车辆后才建立车辆的逻辑对象。

③ 网络服务调度。该组件在满足一些限制的条件下，如结点数目和位置等受限，对提供目标感知功能的传感器结点服务进行调度。它的输出包含服务配置消息，该消息被传送到结点，从而组成服务图。

④ 结点管理。该组件负责使用配置信息对结点服务进行配置，初始化服务发现，并管理由网络服务调度所需要的本地运行服务和资源信息。结点管理通过初始化服务发现，来将服务设置在运行当前服务的对象上。运行服务发现的结点发出一个公共消息，所有收到这个消息的结点做出响应，响应的信息包含可得到的服务和使用这些服务的代价(如能量)。接收结点处理这些信息，并建立一个本地服务数据库。

3）以状态为中心的编程模式

针对目标跟踪的应用问题，人们设计了以状态为中心的编程模式，主要解决了在时间和空间上建立有关物理现象的状态的演化模型。

传感器网络监测得到的被跟踪目标的物理状态，如位置、形状和运动方向，通常在时间和空间上是连续的，可以通过一系列的状态更新来处理传感和控制问题：

$$x_{k+1} = f(x_k, u_k) \tag{5.1}$$

$$y_k = g(x_k, u_k) \tag{5.2}$$

其中 x 是系统状态，k 是时间或空间上的更新系数，u 是输入，y 是输出，f 是状态更新方程，g 是输出或观测方程。

在这种以状态中心的设计框架中，物理现象的整体全局状态被分成层次化的独立可更新的模块。每个模块都有一个叫作“责任者”的计算实体。为了更新状态，某个负责者可以要求其他负责者提供输入，执行传感的负责者完成最底层的探测任务。通过在这些负责者之间定义协作群，为每个负责者制定相应的角色，就可以确定通信模式。负责者封装了一系列的状态，每个状态表征某一特定的物理特征。

仿真结果表明，在多目标跟踪的场景下，以状态为中心的编程模式能够对多目标进行精确分类，并且能够实施有效的目标身份管理。

5.3.3 后台管理软件

1. 结构与组成

可视化的后台管理软件是传感器网络系统的一个重要组成部分，是获取和分析传感器网络数据的重要工具。人们在选定传感器网络的硬件平台和操作系统之后，通过设计相应的网络通信协议，将这些硬件设备组建为网络，在这个过程中需要对网络进行分析，了解传感器网络的拓扑结构变化、协议运行、功耗和数据处理等方面的情况。这都需要获取关于传感器网络运行状态和网络性能的宏观和微观信息，通过对这些信息进行处理，才能对网络进行定性或定量分析。

由于传感器网络本质上是一种资源受限的分布式系统，网络中大量的无线自主结点相互协作分工，完成数据采集、处理和传输任务。从微观角度来看，传感器网络结点状态的获取难度远大于普通网络的结点。从宏观角度上来分析，传感器网络的运行效率和性能也比一般网络难以度量和分析。因此，传感器网络的分析与管理是应用的重点和难点，传感器网络的分析和管理需要一个后台系统来支持。

通常传感器网络在采集探测数据后，通过传输网络将数据传输给后台管理软件。后台管理软件对这些数据进行分析、处理和存储，得到传感器网络的相关管理信息和目标探测信息。后台管理软件可以提供多种形式的用户接口，包括拓扑树、结点分布、实时曲线、数据查询和结点列表等。

另外，后台管理软件也可以发起数据查询任务，通过传输网络告知探测结点执行查询任务，例如后台管理软件询问“温度超过 80℃的地区有哪些”，网络在接收到这种查询消息后，将温度超过 80℃地区的数据信息返回给后台管理软件。

后台管理软件通常由数据库、数据处理引擎、图形用户界面和后台组件四部分组成，如图 5.22 所示。

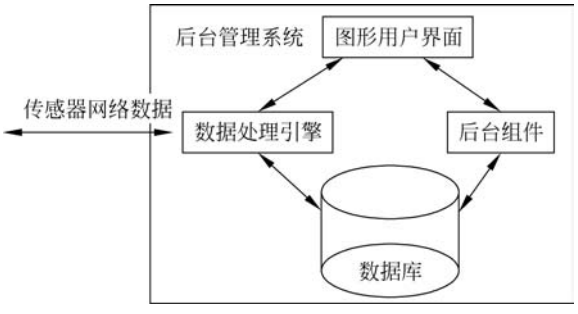

图 5.22　后台管理软件的组成

数据库用于存储所有数据，主要涉及网络管理信息和传感器探测数据信息两种，包括传感器网络的配置信息、结点属性、探测数据和网络运行的一些信息等。

数据处理引擎负责传输网络和后台管理软件之间的数据交换、分析和处理，将数据存储到数据库。另外它还负责从数据库中读取数据，将数据按照某种方式传递给图形用户界面，

以及接收图形用户界面产生的数据等。

后台组件利用数据库中的数据实现一些逻辑功能或者图形显示功能，它主要涉及网络拓扑显示组件、网络结点显示组件、图形绘制组件等。PC 的操作系统、选用的数据库系统和一些图形软件工具都可以提供这类组件，协助开发人员设计和丰富后台管理系统的功能。

图形用户界面是用户对传感器网络进行检测的可视化窗口，用户通过它可以了解网络的运行状态，也可以给网络分配任务。该界面既要保证操作人员对整个网络系统的管理，又要方便使用和操作。

在传感器网络领域已经有一些后台管理软件工具，如克尔斯博公司的 MoteView、加州大学伯克利分校的 TinyViz、加州大学洛杉矶分校的 EmStar、中科院开发的 SNAMP 等。这些软件都在传感器网络的数据收集和网络管理中得到了应用。

2. MoteView 软件介绍

MoteView 是 Windows 平台下支持传感器网络系统的可视化监控软件。无线网络中所有结点的数据通过基站储存在 PostgreSQL 数据库中。MoteView 能够将这些数据从数据库中读取并显示出来，也能够实时地显示基站接收到的数据。网络管理者通过 MoteView 随时掌握目标监测的情况。管理者可以通过数据、图表或结点拓扑结构的直接形式，快速整理、搜寻或查阅每个结点的数据信息。MoteView 还可以根据管理者的设置，采用手机短信和电子邮件的方式提供报警信息。

MoteView 作为无线传感器网络客户端管理和监控软件，功能是提供 Windows 图形用户界面，主要作用包括：①管理和监控系统；②发送命令指示；③报警功能；④Mote 编程功能；⑤网络诊断。

例如，MoteView 可以提供实时数据和历史数据显示功能、可视化网络拓扑图功能、数据输出功能、图表打印功能、结点编程程序 MoteConfig、支持对传感器网络的命令发送、E-mail 报警服务。

MoteView 支持 CrossBow 公司的所有传感器和数据采集板、MICA 系列平台，包括 MICA2、MICA2DOT、MICAz，另外还可以配置和监测基于 MSP 系列结点的安全/入侵监测系统和基于 MEP 系列结点的环境监测系统。

MoteView 支持的应用程序包括 Surge-Reliable、Surge-Reliable-Dot 以及运行 XMesh 和 XSensor 的应用程序。

MoteView 支持的操作系统包括 Windows XP、Windows 2000，使用 MIB510 时通过串口、使用 MIB600 和 Stargate 时通过以太网端口来与计算机相连接，并配置 PostgreSQL 8.0 数据库服务、PostgreSQL ODBC 驱动和微软.NET 构架。

PostgreSQL 是一个开放源码的免费数据库系统，它最初由加州大学伯克利分校计算机科学系开发，倡导了很多关系对象的观念，这些观念现在已经用在一些商业数据库系统中。它提供了 SQL92/SQL99 语言支持、事务处理、引用集成、存储过程以及类型扩展。PostgreSQL 可以说是最富特色的自由数据库管理系统，它基本包括了目前世界上最丰富的数据类型，有些数据类型连商业数据库都不具备，如 IP 类型和几何类型等。

MoteView 在使用时需要链接到数据库，并与无线传感器网络相连接。在链接数据库时需要选择被链接的数据库名称和表的名称。在连接传感器网络时需要选择被连接的设备

和设备的地址。

在以上的设备和连接配置正确实施之后，MoteView 即可运行起来，对整个传感器网络进行监测。它支持结点自动发现功能，提供许多菜单和工具条方便用户使用。MoteView 具有结点列表，显示数据和服务器消息，进行系统管理和数据库管理。结点列表能够显示部署的所有结点及其状态，并且可以对结点进行操作，如添加结点、修改结点属性、结点排序等。数据可以通过多种视图方式进行显示，包括数据视图、命令视图、图形视图和拓扑视图。服务器消息显示部分包括服务器端消息、数据库错误和一般状态消息的显示。

如图 5.23 所示为 MoteView 显示的传感器数据列表示例。如图 5.24 所示为 MoteView 输出的传感器信号波形示例。

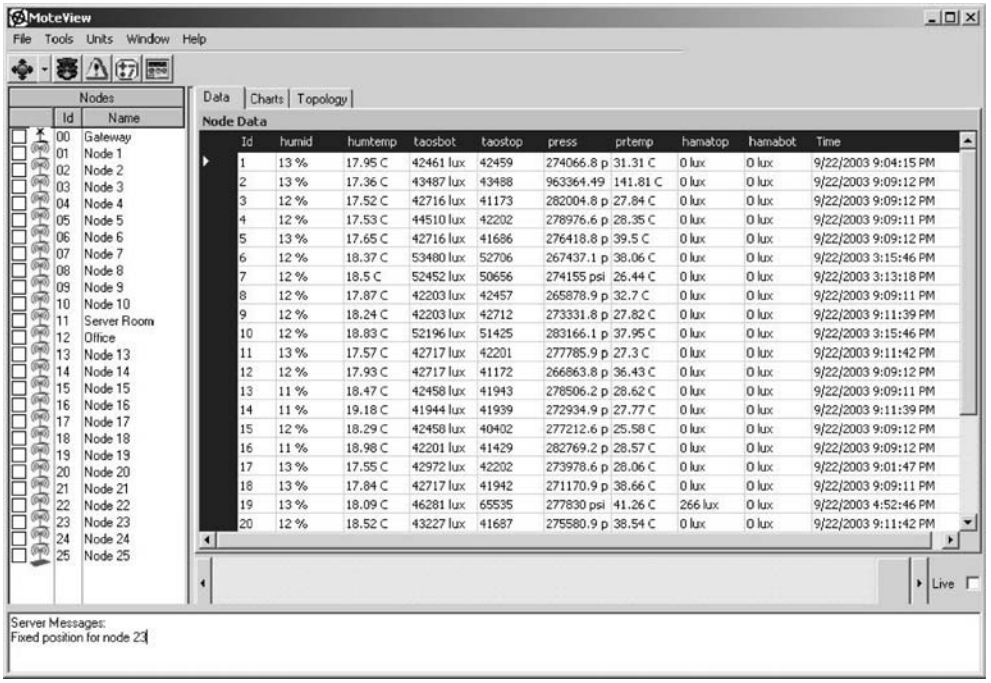

图 5.23　MoteView 显示的传感器数据列表

3. SNAMP 软件介绍

中科院开发的 SNAMP(Sensor Network Analysis and Management Platform)包括串口、数据处理模块、实时显示模块等主要模块。模块化的设计使得整个系统层次扩展性良好[65]。SNAMP 还提供了多种形式的用户接口，包括拓扑树、实时点列表等，满足用户在分析和管理无线传感器网络时的种种需求。

SNAMP 后台管理软件运行在与 Sink 结点相连的主机，通过串口可以读取 Sink 结点收集到的数据，并且对这些数据进行分析，还可以显示网络运行时的动态效果。计算机的后台界面程序负责从串口读取数据包，并进行解析、曲线绘制等。后台显示的结点拓扑示例如图 5.25 所示，这里 Base Station 表示 Sink 结点，5、6、7 表示传感数据的采集结点，实线表示两个结点之间正在有数据包传输。

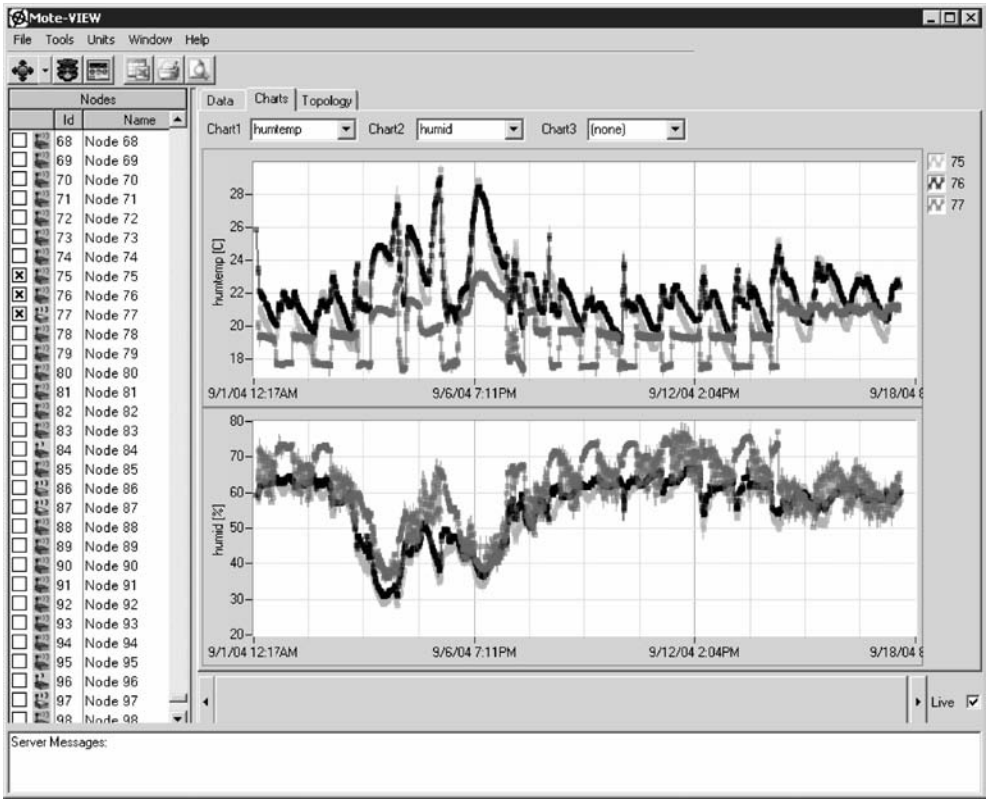

图 5.24 MoteView 输出的传感器信号波形

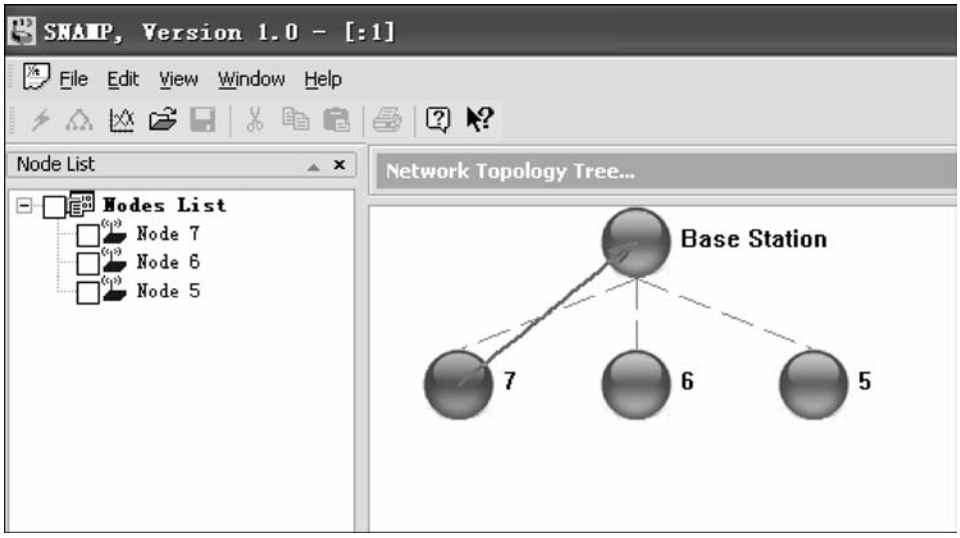

图 5.25 SNAMP 实时显示传感器网络拓扑结构的示例

SNAMP 的实时曲线部分采用了 Gigasoft 公司(www. gigasoft. com)的“ProEssentials”控件(试用版),它是应用于 Windows 服务器端和客户端开发的一种图表组件,是对绘制图表和图表分析功能所需要的数据和方法的简单封装。它的图表类型较多,包括一般图表、科学图表、三维图表、极坐标图表、饼状图表,几乎覆盖了所有常见的图表类型。

如果将上述传感器结点的感知数据以曲线形式实时地在表格中绘制出来,则如图 5.26 所示,这是采用 ProEssentials 控件实现传感器探测数据的实时显示结果。

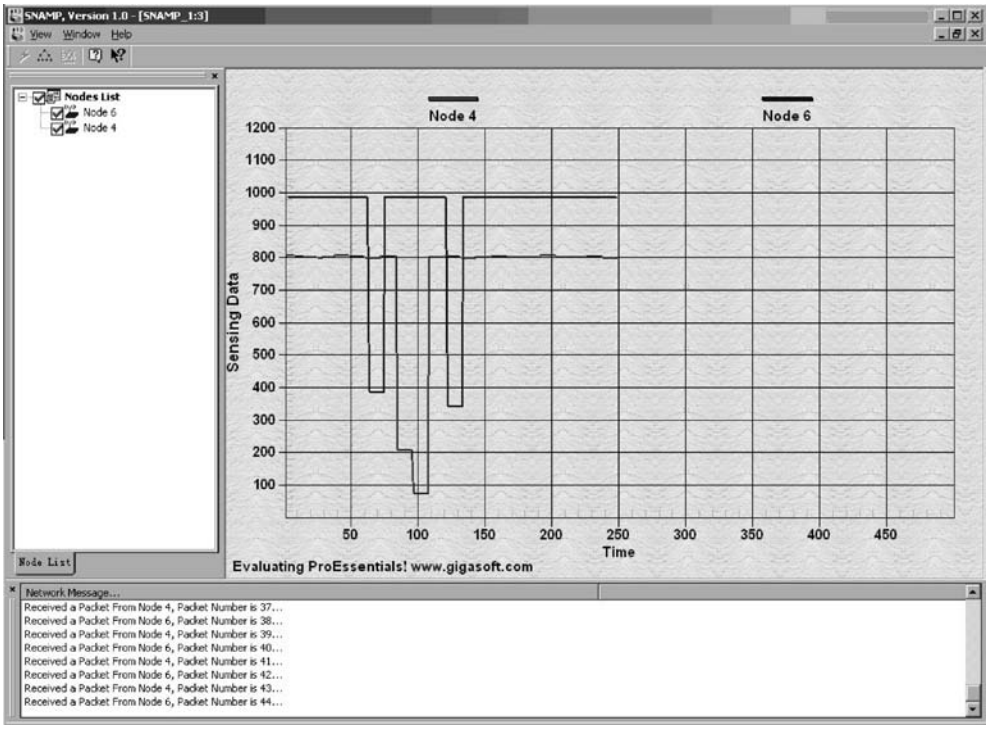

图 5.26　传感器数据曲线的实时显示

思考题

(1) 网络仿真技术具有哪些特点?

(2) 简述网络仿真软件的体系构成。

(3) 列举传感器网络仿真的常用软件平台,并说明各种平台的技术特点。

(4) 简述 TOSSIM 的体系结构和功能。

(5) 选择传感器网络的仿真平台时应该注意哪些问题?

(6) 传感器网络工程测试床的作用是什么?

(7) 简述 SensoNet 试验床的组成和模拟方法。

(8) 列举作为传感器结点数据处理模块的几种常见微控制器芯片。

(9) 简述传感器网络常用的几种无线通信技术及其特点。

(10) 天线的性能有哪些评价指标?

(11) TinyOS 操作系统有哪些特点?

(12) 试写(画)出传感器网络结点软件系统的分层结构。

(13) 传感器网络系统设计的结点应用框架包括哪些组件?

(14) 画出无线传感器网络后台管理软件的一般组成框图并解释说明。

(15) 在教师的指导下,由 2～3 名学生合作完成某传感器网络课题项目的方案设计任务,要求以技术报告的形式写出项目需求、设备选择、解决方案和现实应用价值等内容。