基于 OMNeT++的“实代码”仿真模式研究*

单卫龙[1]，马 奎[2]，周武能[1]
(1. 东华大学 信息科学与工程学院，上海 201620；
2. 中科院上海微系统与信息技术研究所，上海 200050)

摘 要：尝试搭建一个基于 OMNeT++仿真器的“实代码仿真模式”的研究，将整套 UCOS 系统上运行的协议代码封装成动态链接库，加载到仿真器上利用其离散仿真事件队列机制进行调试，运行通过后可以直接在实际硬件芯片上运行，无需二次修改，大大提高研究开发进度。成功地将 UCOS 上运行的 aloha 协议栈封装成 DLL 加载到 OMNeT++，调试运行通过并且已经投入到实际应用中。

关键词：无线传感网；OMNeT++；实代码仿真；动态链接库
中图分类号：TP311.52 **文献标识码**：A **文章编号**：1674-7720(2010)20-0011-05

Research of real code simulation model based on OMNeT++

SHAN Wei Long[1], MA Kui[2], ZHOU Wu Neng[1]
(1. College of Information Science and Technology, Donghua University, Shanghai 201620, China;
2. Shanghai Institute of Microsystem and Information Technology, Shanghai 200050, China)

Abstract: And this thesis try to build a "real code simulation model" based on OMNeT++, which hasn't been proposed before. In this project, protocol stack running on UCOS is encapsulated into DLL, and loaded on the emulator with its discrete event simulation queque, and after the actual debug it can be transferred on the actual microchip. This thesis has successfully encapsulated the protocol stack into DLL and load it on the OMNeT++ emulator, which has been debugged successfully and into practical application.

Key words: wireless sensor network; OMNeT++; real code simulation; dynamic link library

1 OMNeT++

无线传感器网络由称为“微尘(mote)”的微型计算机构成。这些微型计算机通常指带有无线链路的微型独立节能型计算机。无线链路使得各个微尘可以通过自我重组形成网络，彼此通信并交换有关现实世界的信息。目前，对于传感网的研究越来越受到关注，其中网络协议算法更是其中的热点之一。

为评价传感器网络协议算法的性能，仅通过实验是无法实现的，特别是包含大量节点的大规模无线传感器网络[1]，更是很难通过实验来实现(实际上，上百个节点的实验已经比较难以管理与实现)。为了实现无线传感器网络的仿真，研究人员设计开发了许多仿真平台(或在现有平台建立无线传感器网络模型)，包括 NS-2、OPNET、SensorSim、EmStar、OMNet++、GloMoSim、TOSSIM、PowerTOSSIM 等。

OMNeT++[2-3](Objective Modular Network TestBed in C++) 是开源的基于组件的模块化的开放网络仿真平台，是近年来在科学和工业领域里逐渐流行的一种优秀的网络仿真平台。OMNeT++作为离散事件仿真器，具备强大完善的图形界面接口和可嵌入式仿真内核，同 NS2[4]、OPNET[5]和 JavaSim 等仿真平台相比，OMNeT++可运行于多个操作系统平台，可以简便定义网络拓扑结构，具备编程、调试和跟踪支持等功能。OMNeT++主要用于通信网络和分布式系统的仿真。

OMNeT++具有模块化的结构，图 1 是 OMNeT++仿真的高层体系结构。

2 实代码仿真

2.1 总体思路

对于传统的有线网络，利用有限的具有代表性的节点拓扑就可以相当大程度地模拟整个网络的性能，但是对于无线传感器网络，由于其大冗余度和高密度节

* 基金项目：新一代宽带无线移动通信网国家科技重大专项(2009ZX03006-004)

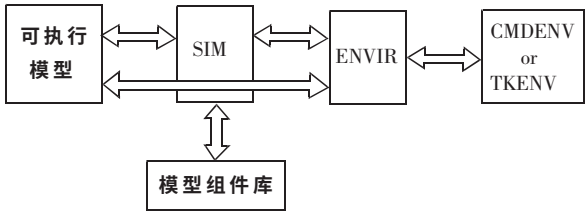

图 1 OMNeT++仿真程序的体系结构

点拓扑构造类型而无法用有限的节点数目来分析其整体性能。因此在仿真规模上必须考虑大量节点的并行运算,WSN 节点有可能成千上万甚至更多。

仿真可以在算法和实现之间起到桥梁作用,从仿真到实现不要进行二次编码,而是平滑过渡。经过仿真测试和验证了的代码能够直接在硬件上运行,但经常出现算法仿真通过而实际却不能实现的情况。

本论文研究实代码仿真的主要原理就是将硬件中断换成离散仿真事件,由仿真器事件抛出的中断来驱动上层应用,也即节点实代码。

总体思路框架图如图 2 所示,不同节点类型、不同数目的同一类型节点都可以在支持分布式仿真的仿真模拟器 OMNET++上运行,仿真的具体思路就是将每一个节点(对应一个对象)抽象成一个线程(线程始终运行),其次将需要仿真的实代码以动态链接库的形式调入线程中,由 OMNeT++底层的离散仿真事件驱动编译成 DLL 的代码。

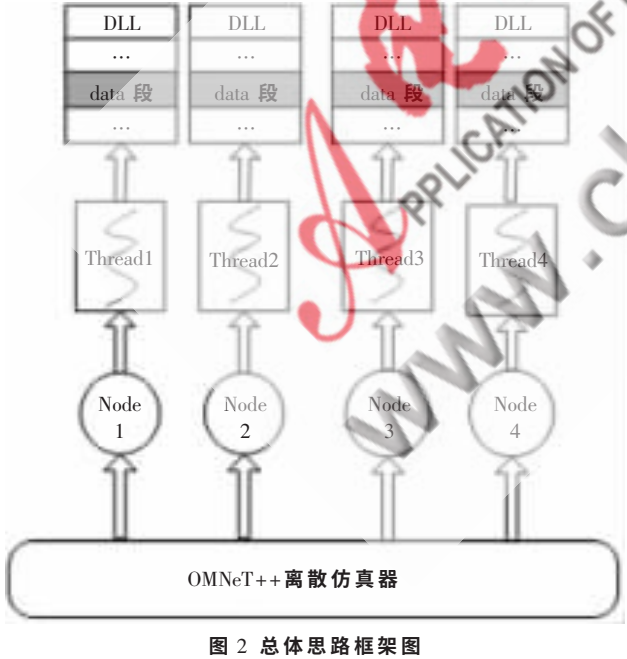

图 2 总体思路框架图

本次试验的 DLL 动态链接库为 ucos 系统上的系统软件协议栈 。其中附加运行的任务为测试程序,实际应用过程中存放的是在硬件上运行的 OS 的节点代码。

2.2 动态链接库

动态链接库简称 DLL,它是基于 Windows 程序设计的一个非常重要的组成部分,可以被其他应用程序所共享的程序模块,其中封装了一些可以被共享的里程和资源[6]。与使用普通的函数库相比,它并不是将库中的代码拷贝到可执行文件中,而是在建立应用程序的可执行文件时动态装载到动态链接库 DLL[7-8],装载时 DLL 被映射到进程的地址空间中,在程序中记录函数的入口点和接口,不管多少程序使用 DLL,内存中都只有一个 DLL 副本,当没有程序使用时,系统将其移出内存,减少了对内存和磁盘的要求。

本文简单地编译了一个 DLL,将运行在 UCOS 上的 Aloha 协议封装成 DLL,协议功能就在于实现简单的侦听功能。为了进行验证,在 stack 中附加输出 test()函数,函数功能就是将全局变量 m 自增 1,返回 m 值。

```
int m=0;
int test()
{
m++;
return m;
}
```

然后通过.def 文件将 OS 启动入口函数 test()导出如下:

```
LIBRARY dlltest
EXPORTS
test
```

这样在 vc6.0++环境目录中,可以得到 dlltest.dll 文件,将其拷贝到 omnetpp-4.0\samples\test 下即可以在 OMNeT++中的源文件中调用,具体调用方式为 GetProcAddress()显式调用。

2.3 节点切换:PE 文件解析

将代码以动态链接库的方式加载入内存后,每个节点将会加载该 DLL,这样将会出现 DLL 中的全局变量等信息无限次地被每个节点所更改。其结果将是第二个节点的全局变量信息也许会成为第一个节点的全局变量信息。

但是,所希望得到的结果是第一个节点的全局变量信息始终是第一个节点的,第二个节点的全局信息也始终是第二个节点的。这样,自然而然想到的就是如何保护全局变量的问题。本工程所采用的一个小技巧就是每个节点产生一个数组,来保存节点信息。在这里要重点区别于多进程共享全局变量问题,本文涉及到的内容主要为线程间共享数据。

DLL 为 PE 文件结构如图 3 所示。

DOS MZ-HEADER
DOS stub
PE header
Section table
Section 1
Section 2
Section …
Section *n*

图 3 PE 文件结构图

PE[9]文件格式是 Win32 平台上(包括 Windows9x/NT/2000/XP/2003/Vista/CE)主流的可执行文件

格式，是 Portable Executable File Format（可移植的执行体）简写。它衍生于早期建立在 VAX/VMS 上的 COFF (Common Object File Format) 文件格式。对 PE 格式和 COFF 文件的主要描述存放在 winnt.h 文件中，它是 PE 文件定义的最终决定者。

文件偏移地址是指当 PE 文件存贮在磁盘上时，某个数据的位置相对于头文件的偏移量，称为文件偏移地址(File Offset)或物理地址(RAW Offset)。文件偏移地址从 PE 文件的第一个字节开始计数，起始值为 0。

相对虚拟地址(RVA)只是内存中的一个简单的相对于 PE 文件装入地址的偏移位置，它是一个“相对”地址，或称“偏移量”。图 2 显示了 PE 文件在装入前后的相对位置变化。虚拟地址(VA)=基地址(ImageBase)+ 相对虚拟地址(RVA)。

本文所编动态链接库(DLL)函数中的全局变量是储存在.data 段中的。这样就牵扯到每个节点运行后，要将其.data 段的信息保存在数组中，在下次这个节点运行时再将其拷贝回.data 段中，保证加载在内存中的 DLL 不被其他的节点所更改。

其中重要部分是找到动态链接库在内存中的位置地址，然后根据 PE 文件结构来解析 data 段的位置，以及其 data 段的大小。

伪代码如下：

```
const char *szSecName=".data";
hInst=LoadLibrary();
……
IMAGE_DOS_HEADER *pDosHead;
IMAGE_FILE_HEADER *pPEHead;
IMAGE_SECTION_HEADER *pSection;
……
strncmp (szSecName, (const char*)pSection [i].Name, IMAGE_SIZEOF_SHORT_NAME);
……
```

在每个节点的 node::initialize()中，为每个节点建立一个线程，而此线程就是将文中提到的“实代码”以显式链接的方式加载到其中。需要注意的是，这个线程是一直运行的，线程间以信号量的形式进行通信。也即在主线程中设置信号量为：

```
……
SetEvent(eventM);
……
WaitForSingleObject(eventT, INFINITE);
……
```

当然，在各个线程中：

```
……
WaitForSingleObject(eventM, INFINITE);
……
SetEvent(eventT);
……
```

通过这种方式来进行线程间的交互，达到主线程(Omnet++)模拟的硬件中断驱动上层的目的。其中多个线程中的主线程用来仿真所有节点应用的运行，而分线程主要用来向外部动态链接库程序提供服务和接受外部程序发送过来的命令。

实际消息处理过程中数组中，将保存的信息段数据复制到 PE 文件中的全局数据段地址就是在 SetEvent (eventM) 消息处理之前进行的，而 WaitForSingleObject (eventT, INFINITE)之后再将运行完成后的数据段拷贝回数组中。

2.4 OMNeT++配置

(1) Ned 文件的编写

NED 语言用来刻画定义模型的拓扑结构，方便对一个网络的模型化描述，这意味着一个网络的描述可以包括一组元件的描述(通道，简单/复杂模型)，这些组件的描述可以在其他网络描述中得以重用。包含网络描述的文件带有.Ned 的后缀，.Ned 文件动态地载入到模拟程序，或者用 Ned 编译器或 C++代码链接到模拟器执行。NED 文件可以使用任何文本编辑器或 GNED 图形编辑器来编写。

本次试验中设定了两种不同类型的节点：普通节点和汇聚节点。每种类型的节点为一个简单模块。而普通节点的数目作为一个可变参数由输入者自己确定。

模块声明只定义了模块类型，要确实地获得一个仿真器能运行的模块，需要书写网络定义。网络定义将前面定义的模块类型声明为一个仿真模块实例，尽管可以将一个模块作为自包含的简单模块并实例为一个网络，但应用中更希望使用复合模块类型。在 NED 文件中可以有几个网络定义仿真程序，使用 NED 文件可运行其中任何一个，可以在配置文件时选择最想使用的那个，本次试验的网络定义语法如下：

```
network net
{
  parameters:
    @display("bgi=background/terrain");
    int numNodes;
  submodules:
    node[numNodes]: node {
      parameters:
        @display("i=,cyan");
    }
    s_node: s_node {
      parameters:
        @display("i=,gold");
}
```

其中用图形化编辑如图 4 所示。

(2) node.cc 文件编写

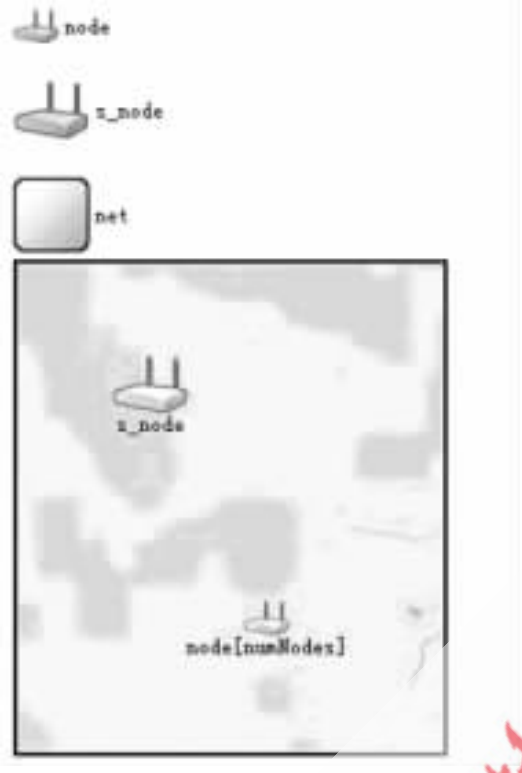

图 4 图形化编辑拓扑

正如前文所述，每个节点抽象为一个线程，node::initialize()中，为每个节点建立一个线程，该线程是一直运行的。同时，对于 PE 文件结构的解析也在此函数中完成。node::handlemessage()中，模拟中断的产生，通过自发信息延迟一定时间来实现模拟定时中断，然后置信号量来实现。

s_node.cc 文件的编写类似 node.cc 文件。

(3) 信道配置

离散事件系统是指一个系统的状态改变是离散的，在两个连续事件之间没有任何事件发生。简单地说，事件规定了系统状态的改变，状态的修改仅在事件发生时进行。离散事件系统可以使用离散事件模拟仿真。例如，计算机网络通常被看作是离散事件系统。部分事件包括包传输的开始、包传输的结束、重传等待时间到达。

3 结果讨论

按照上述基本步骤的编写结果布局如下，普通节点数目设置为 20。

分析以上结果可以看到：图 5 所示为初始布局，配置文件中设置的普通节点数目为 20，汇聚节点数目为 1。图 6 清晰地显示了动态链接库在进程中的内存加载位置，并且由其 PE 文件结构可以得知其各个段的分布情况：起始位置、长度等。data 数据段的地址以及长度就可以顺利得到。当不采用数组时每个节点是顺序增加的，如图 7 所示。这说明每个节点的全局变量信息都是上一个节点的全局变量信息，全局变量依次递增。相反，采用数组进行保存后，其中的各个节点全局信息将自己进行保存，如图 8 所示。即达到本文试验研究的初步预期结果：将实代码以 DLL 形式加载到进程后，多节点抽象出的多线程将会共享 DLL，表示节点状态的全局变量将会被不同节点修改，此工程设计的 PE 文件数据段的拷贝很好地解决了这个问题。

基于 OMNeT++的“实代码”仿真具有很强的可伸缩性。对于无线传感器网络，由于其大冗余度、高密度节点拓扑构造类型，所以无法用少量的节点数目来分析其整体性能。因此在仿真规模上必须考虑大量节点的并行运算，WSN 节点有可能成千上万甚至更多；同时仿真应能够在算法和实现之间起到桥梁作用，从仿真到实

图 5 仿真结果节点布局

图 6 工程初始阶段的 PE 文件解析结果

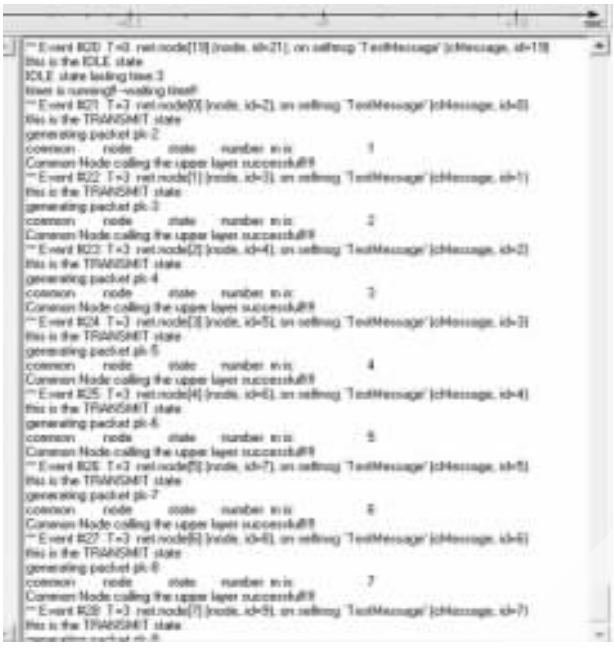

图 7 没有数组的输出结果

图 8 有数组的输出结果

现不要进行二次编码，而是平滑过渡。仿真时测试和验证了的代码能够直接在硬件上运行，因为经常出现算法仿真通过而实际却不能实现的情况。

本项目的主要贡献就在于提出了一种全新的验证协议代码的方案。它实现了在 OMNeT++这款离散仿真软件的基础上，对运行在 UCOS 上的协议栈代码直接仿真调试，将其封装成为动态链接库的形式在 OMNeT++上调试运行，协议栈等代码无需二次修改，运行后的代码可以直接在节点上运行。具体思路已经如上验证通过，并且已经将 Aloha 简单的协议运行在芯片上，更深入的研究正在进行中。

参考文献

[1] AKYILDIZ I F,SU W, SANKARASUBRAMANIAM Y,et al. Wireless sensor networks: a survey[J]. Computer Networks, 2002,38:393-422.

[2] OMNeT++ Home Page. http://www.omnetpp.org, 2007(9).

[3] VARGA A. 2001. The OMNeT++ discrete event simulation system. In the proceedings of the european simulation Multiconference(ESM 6-9, 2001(6). Prague, Czech Republic).

[4] BAJAJ S, BRESLAU L, ESTRIN D. Improving simulation for network research. IEEE Computer. to appear, a preliminary draft is currently available as USC technical report 2000:99-702.

[5] OPNET Technologies, Inc. OPNET Modeler. http://www.opnet.com, 2007(9).

[6] KRUGLINSKI D L. Visual C++技术内幕[M]. 潘爱民，王国印，译.北京：清华大学出版社，2000.

[7] 费佩燕. VC++中动态链接库的实现[J]. 现代电子技术，2003,28(8):7-18.

[8] 李海霞. VC 中的动态链接库应用 [M]. 软件应用,2004.

[9] MATT PIETREK.Windows95 系统程式设计大奥秘[M]. 台湾:旗标出版有限公司,1997.

(收稿日期：2010-06-22)

作者简介：

单卫龙，男，1986 年生，硕士研究生，主要研究方向：无线传感网(WSN)，嵌入式。

马奎，男，1979 年生，博士，主要研究方向为无线传感网。