可扩展传感器网络模拟器 SensorSSF 及性能预测

王换招[1] 库尔班·哈斯木[2] 王艳武[1] 陈培军[1]
(西安交通大学计算机科学与技术系 西安 710049)[1]
(新疆师范大学数理信息学院计算机系 乌鲁木齐 830054)[2]

摘 要 仿真模型越来越复杂,受单机计算能力和存储容量的限制,模拟需要花费的时间也越来越长。PDES(Parallel Discrete Event Simulation)策略能够加快仿真程序的执行,因此一度成为研究热点。但是,并行仿真最终并没有在工业界得到广泛应用,其原因在于:并行仿真建模理论缺乏,并行仿真性能具有不可预测性,以及并行程序行为的不可预测性。本文在讨论模拟器并行化的一般方法基础上,给出了一个基于 SSF 的传感器网络并行仿真环境 SensorSSF。SensorSSF 设计遵循:可扩展性和简洁性。可扩展性保证 CPU 执行时间随求解问题的规模和仿真模型的复杂度线性增长;简洁性使得仿真应用人员无需了解太多并行程序设计知识,就可以编写出高效的仿真程序。实验结果表明,SensorSSF 具有良好的可扩展性,同 NS2 相比具有较好的时间特性。

关键词 并行离散事件仿真,无线传感器网络,可扩展模拟器框架,性能预测

Study of Scalable Sensor Network Simulator and Performance Prediction

WANG Huan-Zhao[1] Kurban Kasim[2] WANG Yan-Wu[1] CHEN Pei-Jun[1]
(Department of Computer Science and Technology, Xi'an Jiaotong University, Xi'an 710049)[1]
(School of Maths-Physics and Information Science, Xinjiang Normal University, Urumqi 830054)[2]

Abstract As the system to be simulated becomes more and more complex, the execution time may be unacceptable due to computational and memory constrain of single processor. PDES(Parallel Discrete Event Simulation) can accelerate the execution of simulation program, thus gained great interest. At the same time, PDES did not become popular in industry. The reasons include at least three aspects: lack of parallel simulation modeling theory, the unpredictable simulation performance of PDES and unpredictable behavior. After discussion general approach of simulator parallelization, this paper gives a detailed description of SensorSSF: a parallel simulator for wireless sensor networks. From the very beginning, designing of SensorSSF obeys two rules: scalability and simplicity. The scalability provides liner CPU performance as the problem size grows; Simplicity relaxes the programmer from knowledge of parallel programming while highly efficiency. The experiment shows that SensorSSF has a better performance than NS at some simulation application and can gain moderate parallelism in some scenery.

Keywords PDES, Wireless sensor networks, SSF, Performance prediction

1 引言

无线传感器网络(Wireless Sensor Networks, WSNs)在军事、医疗和民用等方面的应用具有深远意义,被广泛应用于目标区域成像、入侵检测、天气预报、安全、战术监视,以及分布式计算和环境探测。对于拥有成千上万个节点的大规模无线传感器网络研究,出于成本考虑,只能通过模拟器仿真执行,从而收集相关数据,用于协议优化、网络管理等相关研究。同时,无线信道访问协议的复杂性、节点移动性和无线电特性等,使得对其仿真非常耗费 CPU 计算时间。一种解决方案就是采用多处理机并发执行,加快仿真速度。

并行离散事件仿真 PDES(Parallel Discrete Event Simulation)是指在并行平台上执行离散仿真应用程序。PDES 策略能够加快仿真程序的执行,因此一度成为研究热点。但是,并行仿真最终并没有在工业界得到广泛应用,根本原因归结于:并行仿真建模理论的缺乏,并行仿真性能的不可预测性,以及并行程序行为的不可预测性[1]。F. Alois 等人指出,并行离散模拟性能预测虽然困难,但不可缺少[2]。通过事前预测 PDES 应用的性能,可以决定是否值得采用 PDES 策略。

基于以上分析,提供一个可重用、可扩展的并行仿真环境十分重要。对于大型复杂模型,可扩展性保证内存占用和 CPU 时间线性增加。同时,性能分析尤其是性能预测,对于 PDES 策略广泛采用十分重要。本文给出了一个基于 SSF (Scalable Simulation Framework)的传感器网络并行仿真环境 SensorSSF 的设计,描述了其部分实现;同时,对基于 SensorSSF 的仿真应用程序的性能预测,给出了一个多粒度的性能预测框架。

2 研究背景

2.1 PDES 同步策略

在 PDES 中,一个模型一般包含 n 个逻辑进程:LP_1, LP_2,…,LP_n,它们通过发送包含时间戳的事件消息进行交互。每个 LP 拥有自己本地的虚拟/逻辑时钟(Local Virtual Time, LVT),代表本 LP 当前推进到的仿真时间。在多处理器环境下,不同的 LP 并发执行,每个 LP 的虚拟时间推进是不一样的,因此必须采取措施,保证仿真的因果关系不被破坏,即不允许出现早的事件被发生晚的事件作用。要保证 PDES 中正确的因果关系,在各 LP 间必须采取相应的同步措施,从而产生正确的仿真结果。

PDES 同步算法大致分为两类:保守同步和乐观同步机

王换招 在职博士,副教授,CCF 高级会员,研究方向:网络安全和传感器网络。

制。保守机制要求所有事件严格按照时间戳顺序执行；而乐观同步允许 LP 在保持自己按时间戳执行的条件下，向前推进 LVT 时钟。当某个 LP 收到一个时间戳比自己 LVT 时钟小的事件时，事件的因果序被破坏，需要将系统状态回滚到最近的一次正确状态。CMB[3] 和 Time Warp[4] 分别是具有代表性的保守同步和乐观同步算法，其它的同步算法基本上都是以这两者为基础，不断优化改进而来。

2.2 网络模拟器

大规模网络模拟需要占用计算机大量的存储空间，仿真执行时间一般很长，对于无线传感器网络模拟尤其如此。为了加快仿真执行速度，ParseC[5]，SSFNet[6]，PDNS[7] 等并行仿真器被用于在并行平台上执行网络仿真程序。一般可将并行仿真器依据其实现分为三类：

1)采用并行语言作为模拟器的开发语言，例如 PareseC。

2)在不同机器上执行求解问题的某个子模型，各个子模型的仿真同步进行，用户编写仿真程序时需要负责并行划分，典型例子为 SSF。

3)联合多个串行模拟器，实现顺序模拟器的并行执行[7~9]，其代表是 PDNS。

2.3 可扩展框架 SSF[5]

可扩展仿真框架 SSF 是 S3 项目的研究成果。它是一组规范，指出了模拟器引擎应该实现的接口，但对实现语言、平台没有要求。遵循 SSF 规范的应用程序具有很好的互操作性。因此，增强了模型的可重用性，降低了模拟器和平台的相关性。SSF 为离散事件模拟提供了一套简单、统一的编程接口。采用面向对象的技术构建的 SSF 模型，在 SSF 兼容(SSF-compliant)的模拟环境中都可以移植。目前，SSF 的实现主要包括两个版本：基于 Java 的 SSFNet[6] 和 Dartmouth 大学的 DaSSF[10]。SSFNet 除了实现 SSF 核心外，还包含丰富的可重用 Internet 协议库。

3 SensorSSF 设计与实现

SensorSSF 采用 Dartmouth 的 SSF 核心作为其模拟引擎，具有如下特点：1)支持共享内存的 SMP，以及分布式内存的 SMP 工作站；2)高效内存管理，较少的 CPU 执行时间；3)对于大规模网络模拟，SensorSSF 具有良好的可扩展性；4)良好的结构，采用 SensorSSF 能够快速构建传感器网络模拟应用，网络模拟采用域建模语言 DML 描述，具有良好的可重用性；5)网络模型可采用 SensorSSF 提供的工具实现并行划分，映射到不同处理机上执行，用户编写程序的步骤和编制一般的串行程序没有差异。

SensorSSF 的体系结构如图 1 所示。SensorSSF 分为四层，由下至上分别是 SSF 核心、抽象网络模型和性能分析层、无线传感器网络开发包和实用工具集，最上层为仿真应用程序。其中仿真引擎采用 Dartmouth 的 SSF 实现，它支持共享/分布式内存多处理器并发执行。

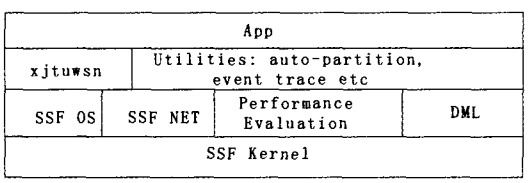

图 1 SensorSSF 体系结构

3.1 抽象网络模型

为了实现组件的可重用性，抽象网络模型从包交换网络抽象出最基本的功能组件，同时保持高度的灵活性，使得不同需求用户能够定制自己的仿真程序。SensorSSF 中网络抽象模型主要包括两个部分：基本网络组件 SSF. NET 和协议栈组件 SSF. OS。基本网络组件包括接口 interface、主机 host、路由器 router、子网 net、服务单元 server 和客户单元 client。一个主机可以有多个接口，通过链路连接到同一子网其它主机或路由器接口。路由器除了具有主机的功能单元外，还负责路由选择和数据包转发。服务单元是仿真模型中流量发生器，比如支持恒定速率流量、指数分布时间戳事件到达流量；客户单元是指向服务单元提出请求的网络元素。子网是由主机、路由器、链路和子网构成。一个复杂网络模型可以看作子网组建的分层系统。对一个节点运行的协议栈实现可插拔配置，SensorSSF 采用了与 x-kernel 类似的设计模式，即将节点运行的协议栈看作一个协议图。

3.2 性能预测

对于 PDES 应用来说，性能预测和事后性能分析是非常重要的。SensorSSF 提供了一个性能预测框架，这也是 SensorSSF 不同于其它 PDES 模拟器的一个特点。有关性能预测框架及其实现将在本文第 4 节中详述。

3.3 无线传感器网络开发包 xjtuwsn

xjtuwsn 是在抽象网络模型之上设计的无线传感器网络仿真开发包，它提供了丰富的接口，可用于传感器网络中节点定位算法、数据链路层协议、路由协议及数据融合算法等模拟。得益于抽象网络模型良好的结构以及完善的功能，xjtuwsn 开发包主要实现了无线传感器网络协议栈中不同层次协议的模拟。由于 NS-2 中关于链路层、路由层的协议仿真已经很成熟，xjtuwsn 实现参照了 NS 中无线仿真的实现。

模拟无线传感器网络行为一般需区别三种节点：监测事件的传感节点(sensor node)、处理事件的信息中心(sink node)、产生事件的目标节点(target node)[11]。如图 2 所示。

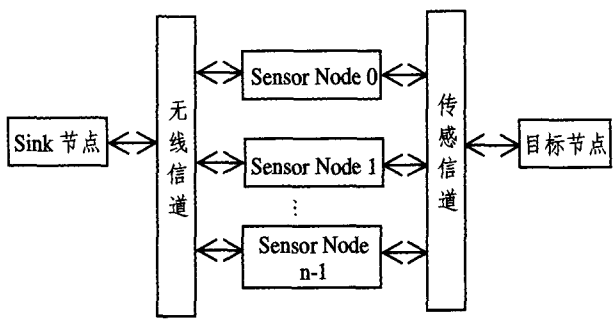

图 2 Sink、Sensor 和目标节点

Sensor 节点运行两个协议栈：监控目标节点的传感协议栈，以及和 sink /sensor 节点通讯的无线协议栈。Target 节点只运行传感协议栈，sink 节点只运行无线协议栈。为了便于研究节点耗能情况，Sensor 节点还包括能量提供(电池)和能量消耗组件(如 CPU、网络接口)。同时，xjtuwsn 还包括移动模型库，可以仿真节点移动的场景。

3.4 SensorSSF 实现

SensorSSF 在 SSF 的五个核心类 Entity，inChannel，outChannel，Process，Event 之上提供了一个可扩展的网络模拟器开发库。SSF. OS 和 SSF. NET 实现的几个核心类如下。

3.4.1 DML 配置接口 ConfigObj

对于 SensorSSF 的网络模型对象，如主机、路由器、服务器和接口等通过 DML 文件配置，具有极大的灵活性。DML 配置接口 ConfigObj 作为 SensorSSF 中所有可配置类的父类，提供了 createInstance 和 DMLConfigure 两个接口，用于类实例的创建和参数配置，其定义如下所示：

```
class DmlObject {
  public:
  DmlObject(DmlObject * parent);
  virtual ~DmlObject();
  virtual int Configure(DML_AttribList * root)=0;
  DmlObject * create_instance()
};
```

3.4.2 协议栈 ProtocolGraph

SensorSSF 抽象网络模型中每个节点都运行一个网络协议栈，这些协议栈通过 DML 来配置。ProtocolGraph 类描述了一个可插拔的协议栈。它通过 GetProtocolByName 创建对应协议处理实例，然后根据协议的上下层关系，使得数据能够正确地上传和下递。

```
class ProtoGraph: public ConfigObj
{
  public:
  ProtoGraph(Machine * machine);
  int Configure(DML_AttribList * protocols);
      //向此协议栈中增加一个包处理协议
  static int RegisterProtocol(char * name, CreationFunction fcn);
  int Initialize();
      // 通过协议名字创建协议处理对象
  SSF_Protocol * GetProtocolByName(char * name);
  private:
      //初始化协议栈
      int init(SSF_Protocol * root);
      // 节点
  Machine * myMachine;
      //本节点安装协议栈列表
  TxHashString(SSF_Protocol * ) ProtocolTable;
};
```

3.4.3 协议 ProtocolSession

ProtocolSession 对应着某个具体的网络协议对协议数据的处理，包括对下层 ProtocolSession 传入数据的解包和对上层传送给自己数据的封包。类 ProtocolSession 的接口如下：

```
class SSF_Protocol: public ConfigObj
{
  SSF_Protocol(uint32 ID, char * name, Machine * pMachine);
  virtual ~SSF_Protocol();
    // 向下层协议发送数据 data
  virtual int send(msglist * data, general_data * options=NULL,
      uint32 oplength=0)=0;
      // 接受下层向本协议提交的数据 data
  virtual int receive (byte * data, uint32 length, general_data * op-
      tions=NULL,
      uint32 oplength=0)=0
    virtual int Configure(DML_AttribList * config)=0;
    // 协议初始化
  virtual int Initialize() = 0;
}
```

4 基于 SensorSSF 性能预测

4.1 影响并行仿真的因素与衡量标准

影响 PDES 性能的因素可以分为三类：仿真模型、PDES 同步机制和执行平台。仿真模型体现了被仿真系统的固有性质，PDES 采用同步策略可能带来一定的开销，而执行平台的处理机数量、并行划分及负载也影响着执行的 CPU 时间。这三者相互制约，共同决定了某个具体应用的执行时间。

加速比和效率通常被用来衡量并行程序的性能。这里的性能，仅就执行时间而言。但并行仿真程序有其自身的特点，故而有研究者引入单位时间内执行事件的数量作为度量。本文引入并行度作为衡量标准：对有 n 个 LP 并发执行的仿真程序，W_i 代表 LP_i 上事件执行时间的总和，T_n 是 n 个 LP 并发执行仿真程序耗费的 CPU 时间，LP_i 的利用率 U_i 定义为

$$U_i = W_i / T_n$$

可以证明 PDES 应用程序的并行度为系统中所有 LP 的利用率之和。

4.2 基于 SensorSSF 的仿真程序性能预测框架

由于影响 PDES 性能的因素众多，并且这些因素之间也是相互关联的。不同的 PDES 实现，其性能衡量的参数也不同，因此性能预测无法考虑所有的因素。已有的性能预测算法通用性较差，只适用于某类模型，或是采用某种同步协议的应用。本文给出基于粒度的性能预测模型，试图在不同准确度上给出性能预测的算法。用户根据自己的需要，选择某个层次的算法来预测 PDES 模型的性能。这里的粒度是指对性能预测准确度的期望，也包括用户开展仿真活动前需要的知识、仿真活动开展的不同阶段。

图 3 给出了 SensorSSF 的性能预测框架。左边部分对应影响性能的三方面因素，右边从上至下分别是对应的 PDES 性能分析算法。

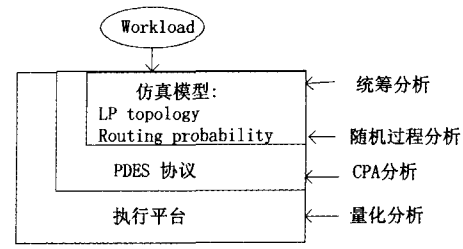

图 3 SensorSSF 性能预测框架

运筹分析(operation analysis)和随机分析(stochastic analysis)属于采用排队网络、petri 网等数学工具来对仿真模型进行分析。这些分析算法考虑 LP 到 processor 的映射、LP 拓扑结构(仿真模型)、各个 LP 的参数，包括吞吐率 X、执行能力 μ 和事件到达率 λ 等。最后仿真模型的并行度表示为 $U = f(X,\mu,\lambda)$ 的函数。并行分布式仿真由于采用了 PDES 同步策略，所以有些事件到达 LP 后并不能立刻被执行(受事件因果序限制)，事件好像被瞬时丢失了一样，称之为丢失事件(loss event)。在实现运筹分析算法时，考虑了丢失事件(loss event)对吞吐算法 MVA[12] 的影响，并针对 MVA 做了修改。

CPA[13] (Critical Path Algorithm)分析方法综合考虑 PDES 同步协议和仿真模型的影响，故而其性能预测较解析方法预测结果要准确。CPA 算法主要是分析事件在 LP 上交错执行的轨迹，记录 LP 并行执行 Tp 和串行执行的时间 Ts，那么预测的并行度 $U = Ts/Tp$。

量化分析(Quantitive analysis)和 PDES 应用运行的模拟器相关，就本文而言，就是 SensorSSF。它首先分析执行中的关键代码，通过实验得出各个部分关键代码的开销，从而估计整个仿真的 CPU 时间。这样，用户可以根据自己的需求，选择某一层的算法来估计并行执行的开销。

4.3 量化分析性能预测

利用量化分析方法预测 PDES 程序的 CPU 时间，其难点在于：1)将 PDES 程序分解为一系列可测量的基本操作，基本操作是指该操作对所有 PDES 程序都通用。2)准确测量基本操作的开销。

根据 SSF 的特点，本文将 SensorSSF 中基本操作划分为：上下文切换开销 C_{cs}、过程调用开销 C_{cp}、路障同步开销 C_{syn}、

事件队列查找开销 C_{evl} 和对象创建/销毁开销 C_{obj}。

4.3.1 上下文切换操作开销 C_{cs}

上下文是指逻辑进程 LP 的上下文。当某个 LP 由于等待或其它原因挂起时，调度器选择一个新的 LP 来执行，引发 LP 上下文切换。上下文切换的 CPU 开销主要和两个因素相关：1) LP 在何种场合被唤醒执行，即引发切换的原因；2)上下文切换时，时间轴(SSF Timeline)中进程的数量。

4.3.2 过程调用开销 C_{cp}

SensorSSF 采用堆方式实现面向进程的栈执行，需要对用户编写完成后的源代码实现部分修改。过程调用开销包括：在堆上创建 LP 的执行栈开销；当某个 LP 处于准备状态时搜索该 LP 来执行的开销；LP 执行完毕或是挂起时，退出线程堆栈的开销。

4.3.3 同步开销 C_{syn}

SensorSSF 中采用基于窗口的保守同步协议，该协议中共有两个同步点。

4.3.4 事件队列操作开销 C_{evl}

SSF 中一组结盟的实体拥有相同的事件列表。事件队列操作开销是指插入、移出事件所花费的 CPU 时间。很明显，事件列表开销和其组织结构相关，如采用有序链表、平衡树或日历队列，其空间和时间复杂度都是不同的。

4.3.5 对象创建、销毁开销 C_{obj}

对于复杂系统的仿真，需要创建和销毁大量的对象。对于基于 SSF 的仿真程序，主要是考虑实体对象 Entity、事件对象 Event 的创建和销毁的 CPU 时间。

4.4 量化分析实例

给定仿真模型和基本操作开销，需要将基于 SensorSSF 的仿真应用程序分解为这些基本的操作，累加这些时间可得出预测执行时间。对于图 4 所示的层次网络模型，可以将总的执行时间分为三部分：模拟数据包产生耗费的时间，模拟数据转发的开销，以及事件同步的开销。如果用一个 SSF 事件代表主机中发送的数据包，那么其开销可以看作 $C_{evl}+C_{obj}+C_{cs}$。因为一个事件带来的开销包括：创建事件对象、插入事件列表和写入通道导致的上下文切换。转发一个数据包开销为 $C_{cs}+C_{lookup}+C_{evl}$，即路由 LP 运行导致的上下文切换、路由表查询开销，以及事件插入列表的开销。事件同步的开销是指事件跨越不同时间轴所带来的开销 C_{syn}。那么模拟一个数据包从发送到达目的主机，其 CPU 时间为：$C_{evl}+C_{obj}+C_{cs}+N\times(C_{cs}+C_{lookup}+C_{evl})+C_{syn}$，$N$ 为数据包经过的中间节点个数。值得注意的是，不同仿真模型的工作流量影响着基本操作的时间，如 C_{evl}，C_{cs}。

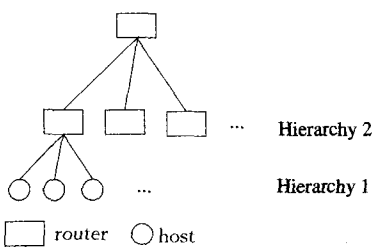

图 4 一个典型网络模型

5 实验与结果

5.1 实验环境

实验中采用 NS2 版本号为 ns-2-2.27，SensorSSF 的模拟器内核为 Datmouth SSF 3.2。并行执行环境为 100M 以太网互联的 linux 集群，包括一个主节点和 N 个计算节点。主节点安装网络文件系统 NFS，与计算节点共享仿真软件安装的目录，并且通过运行网络信息服务 NIS 共享帐号。这样从任何一个节点(无论主、从)可用同一帐号登陆，并且增加新的计算节点时，只需更新主机配置文件/etc/hosts 即可。SensorSSF 中 LP 之间的同步通过消息传递完成，实验中采用消息传递接口 MPICH1.2.6。主、从节点安装操作系统为 Red Hat linux 7.2，配备主频为 800MHz 的 P3 处理器、256M 内存、8GB IDE 磁盘和 100M 以太网卡。

5.2 单机执行性能测试与结果分析

实验中采用的哑铃模型是，$2n$ 台主机通过 2 台路由器互联(左边 n 台客户机，右边 n 台服务器)。路由器缓存大小为 $4*n$ kb，路由器之间链路带宽为 1.5Mbps，而主机到路由器链路为 10Mbps。数据包大小为 1kb。图 5 和图 6 所示是 NS 和 SensorSSF 分别在单机上执行所耗费的 CPU 时间。

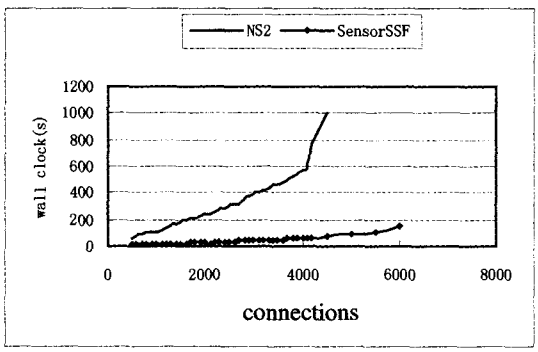

图 5 哑铃模型单机执行 500s

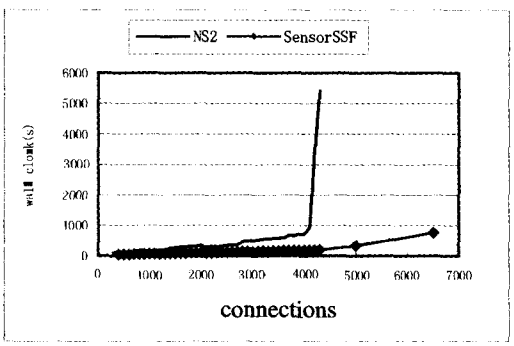

图 6 哑铃模型单机执行 2000s

哑铃模型的执行时间包括两个部分：模型建立时间和模型执行时间。模型建立指仿真环境的启动、网络模型对象的创建；模型执行指模拟 TCP 协议的行为，如发送端最大努力发送、超时重传等。在仿真时间分别为 500 和 2000 情况下，当模型中 TCP 会话数目少于 4×10^3 时，NS2 和 SensorSSF 的执行时间都近似成线性增加，但当 TCP 连接数超过 4×10^3 时，NS2 执行时间上升很快，而 SensorSSF 时间增长速度略有加快。

5.3 并行执行性能测试与结果分析

为了验证 SensorSSF 并行执行的性能，针对哑铃模型和哑铃模型变种(两边分别有 $n/2$ 台和 $3n/2$ 台主机)，分别测量了并行执行的 CPU 时间。实验中，两台 linux 机器作为计算

主机，通过100M以太网交换机互联哑铃模型下CPU执行时间如表1所示。哑铃模型变种的执行时间如表2所示。

表1 哑铃模型仿真执行2000s

TCP连接	串行时间	并行时间	加速比
400	37.6	49.3	0.762677
1000	59.8	69.9	0.855508
1600	82.3	93.3	0.882101
2000	101.8	114.7	0.887533
2600	122	135	0.903704
3000	138.3	145.7	0.949211
3600	166	164.5	1.009119
4200	190	185.2	1.025918
5000	340	287.8	1.181376
6500	760	617	1.231767

表2 哑铃模型变种仿真执行100s

TCP连接	串行时间	并行时间	加速比
40	53.8	70	0.768571
120	207.9	204.3	1.017621
200	401.3	371.8	1.079344
280	588.6	532.1	1.106183
360	786	700	1.122857
440	980	840.2	1.166389
520	1183	1008.8	1.172687
540	1235.2	1040	1.187692
900	2405.2	1893.3	1.270374

无论哪种模型，加速比都随着TCP连接数量的增加而增长。相同条件下，哑铃模型变种具有相对较好的加速比。由于哑铃模型是对称的，很自然地可以将哑铃的一端放在一台处理机上执行，另外一端放到另一台处理机上执行。哑铃模型中所有跨越路由器的数据流量，都对应着跨越CPU的事件调度和执行。哑铃模型中客户机和服务机分别位于哑铃的一端，哑铃变种模型中一部分服务机和客户机在哑铃的同一端，跨越CPU调度的事件数目减少。简而言之，变种模型的天生并行度较之哑铃模型要大，相同条件下较好的加速比在情理之中。值得指出的是，变种模型两端不对称，所以两台处理器上负载并不平衡，也直接影响了加速比。

结束语 仿真工具的缺乏、建模技术的不成熟和性能的不可预测性阻碍了PDES机制的广泛采用。作者在开发无线传感器网络仿真环境SensorSSF时，提供简洁的程序设计模式和相应的性能预测支持工具。实验结果表明，SensorSSF具有良好的可扩展性，比NS2具有较好的执行时间。

进一步的工作主要是对SensorSSF用户接口的改善，加入对高层体系结构HLA的支持，将性能预测和建模结合起来，这需要对UML建模语言的语义进行拓展。

参考文献

1 Liu J, Nicol D, Premore B, et al. Performance Prediction of a Parallel Simulator [C]. In: Thirteenth Workshop on Parallel and Distributed Simulation, 1999. 156～164
2 Ferscha A, Johnson J, Turner S J. Early Performance Prediction of Parallel Simulation Protocols [C]. In: Proceedings of 1st World Congress on Systems Simulation, 1997. 282～287
3 Chandy K M, Misra J. Distributed Simulation: A Case Study in Design and Verification of Distributed Programs [J]. IEEE Transactgions on Software Engineering, 1979, SE-5 (9): 440～452
4 Jefferson D A. Virtual Time [J]. ACM Transactions on Programming Languages and Systems, 1985, 7(3): 404～425
5 Bagrodia R, Meyer R, Takai, et al. Parsec: A Parallel Simulation Environment for Complex Systems [J]. Computer, 1998, 31 (10): 77～85
6 SSF Specification. http://www.ssfnet.org
7 PDNS: Parallel/Distributed NS [EB/OL]. http://www.cc.gatech.edu/computing/compass/pdns/index.html
8 Riley G F, Fujimoto R M. A Generic Framework for Parallelization of Network Simulations [C]. In: Proceedings of Seventh International Symposium on Modeling, 1999
9 李越，钱德沛. 基于NS的分布式并行网络模拟器[J]. 电子学报，2004, 32(2)
10 Liu J, Nicol D. DaSSF manual 3.1
11 王换招，赵青苹，陈骞，等. 一个并行无线传感器网络模拟框架[J]. 微电子学与计算机，2005 (10)
12 Jain R. The art of computer systems performance analysis: techniques for experimental design, mearurement, simulation, and modeling [M]. John Wildey & Sons Inc, 1994
13 Overeinder B J, Sloot P M A. Parallel Performance Evaluation through Critical Path Analysis [J]. High-Performance Computing and Networking, 1995, 919(5): 634～639

(上接第57页)

结束语 在大量的仿真的基础上，对DSR、AODV以及OLSR三种典型MANET路由协议进行了深入分析和研究，并得到了一些有用的结论。在高负荷、高移动性、网络拓扑剧烈变化的网络环境下，AODV协议的性能最好，但DSR性能却最差。这是因为快存中的路由信息快速过时，导致DSR的“快速路由恢复”机制变成“慢速路由恢复”，最终使其性能恶化。因此，DSR可以通过加强对快存中路由信息的管理来保证快存内路由信息的可靠性，增加有效区分路由快存中路由信息新旧程度的机制和及时删除过时路由信息，这样才能充分发挥其路由快存的优势，进而改善DSR在高负荷、高移动性环境下的性能。在高负荷、结点移动性不是很强的应用环境下，OLSR的性能是最好的。在网络业务负荷与结点移动性的变化范围比较宽的场合，也就是网络结点的移动性和业务量不固定的环境中，AODV是最优的选择。在各种网络环境下，用OLSR选择的路由最短。

参考文献

1 Stefano B, Marco C, Silvia G, Ivan S. Mobile Ad Hoc Networking. USA: IEEE Press, 2004. 12～17
2 IETF. RFC 2501: Mobile Ad hoc Networking (MANET)
3 Mehran A, Tadeusz W, Eryk D. A review of routing protocols for mobile ad hoc networks. Elsevier B. V. Ad Hoc Networks, 2004(2): 1～22
4 IETF. Internet-Drafts: The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR)
5 IETF. RFC3561: Ad hoc On-Demand Distance Vector (AODV) Routing
6 IETF. RFC3626: Optimized Link State Routing Protocol (OLSR)
7 Garcia-Luna-Aceves J, Roy S. On-demand loop-free routing with link vectors. IEEE Journal on Selected Areas in Communications, 2005, 23(3): 533～546
8 Wu Xiaoxin, Bhargava B. AO2P: ad hoc on-demand position-based private routing protocol. IEEE Transactions on Mobile Computing, 2005, 4(4): 335～348
9 Grilo A, Macedo M, Sebastiao P, Nunes M. Stealth optimized fisheye state routing in mobile ad-hoc networks using directional antennas. In: IEEE 61st Vehicular Technology Conference, 2005, 4: 2590～2596
10 Bein D, Datta A, Villain V. Self-stabilizing optimal local routing in ad hoc networks. In: 25th IEEE International Conference on Distributed Computing Systems Workshops, 2005. 564～570
11 http://www.ietf.org/html.charters/manet-charter.html
12 http://www.isi.edu/nsnam/ns/index.html