审计缓冲区的形式化模型及其验证*)

丁志军[1,2] 刘海峰[3] 蒋昌俊[1]

(同济大学计算机科学与技术系 上海 20092)[1] (山东科技大学信息科学与工程学院 青岛 266510)[2]

(北京信息安全测评中心 北京 100037)[3]

摘 要 审计系统作为安全信息系统的一个重要组成部分,对于监督系统的正常运行、保障安全策略的正确实施、构造计算机入侵检测系统等都具有十分重要的意义。审计缓冲区的管理是审计系统的核心部分,本文利用时序 Petri 网对审计缓冲区管理的实现方案进行建模,进而对系统的安全性和活性进行了分析和验证。该方法利用时序逻辑扩充了 Petri 网缺乏描述系统事件之间时序关系的局限性,同时发挥了 Petri 网对系统并发和物理结构的有效描述及分析的优势,达到了系统验证的目的。

关键词 审计,时序 Petri 网,缓冲区,验证

Formalization Model and Verification of Audit Buffers

DING Zhi-Jun[1,2] LIU Hai-Feng[3] JIANG Chang-Jun[1]

(Department of Computer Science and Technology, Tongji University, Shanghai 20092)[1]

(Institute of Information Sci. and Eng., Shandong University of Science and Technology, Qingdao 266510)[2]

(Beijing Information Security Test and Evaluation Center, Beijing 100037)[3]

Abstract As a very important component of secure information system, audit system plays a key role in monitoring the system, insuring proper implementing of security policy and building Intrusion Detection System. This paper introduces a subclass of Petri net——temporal Petri nets which is used to build a model for implementing the management scheme of audit buffers. Based on it, we analyzed and verified the properties of system safety and liveness. By using temporal logic, this method can break the limitations imposed by Petri nets, which lack the ability to describe the temporal relations between system events. Moreover, temporal Petri nets exert the advantages of Petri nets which can effectively describe and analyze the system concurrency and physical structures for the purpose of system verification.

Keywords Audit, Temporal Petri nets, Buffer, Verification

1 引言

审计就是对系统中有关安全的活动进行记录、检查及审核,其主要目的就是检测和阻止非法用户对计算机系统的入侵,并显示合法用户的误操作[1,2]。审计作为系统安全的重要组成部分,在 DoD 的 TCSEC(Trusted Computer System Evaluation Criteria)中要求 C2 级以上的安全系统必须包含审计功能[3],在计算机信息系统安全保护等级划分准则(GB 17859-1999)中也有相应的要求[4]。对于高安全级别的计算机信息系统,在 TCSEC 中 B2 级以上及计算机信息系统安全保护等级划分准则第四级以上,都要求该系统建立在一个明确定义的形式化安全策略模型之上[5~7]。审计缓冲区的管理是审计系统的核心部分,由于它涉及多个进程对多个缓冲区的操作,是一个典型的并发系统。因此,对它的设计是较为困难的,而且,系统的并发特性导致了系统运行行为的不确定性,从而使得实现方案的正确性很难得到保证。这就更加需要建立形式化的模型,分析和验证系统的性质,证明实现方案的正确性。

本文利用时序 Petri 网[8]对审计缓冲区管理的实现方案进行建模。一方面,通过构造 Petri 网模型,描述了系统的物理框架;另一方面,利用网模型上的时序逻辑公式,给出了审计缓冲区管理方案中存在的读、写指针之间的同步规范,而这种时序概念上的同步要求是 Petri 网本身无法描述的。从而给出一个完整的审计缓冲区管理方案的形式化模型。在此基础上,结合 Petri 网的可达图分析技术和时序逻辑的演绎公式,对系统的安全性和活性性质进行严格的理论验证,以达到正确性证明的目的,从而论证了原型系统的正确性。

本方法利用时序逻辑扩充了 Petri 网缺乏描述系统事件之间时序关系的局限性,同时发挥了 Petri 网具有对系统并发和物理结构的有效描述及分析的优势,达到了系统验证的目的。

本文在第 2 部分简要介绍了审计缓冲区管理方案;第 3 部分建立了其时序 Petri 网模型;第 4 部分对该模型进行了形式化的验证;最后总结全文。

2 审计系统及其缓冲区的设计[3]

文[3]中对本审计系统进行了详述,它是我们在 Linux 核心 2.2.15 基础上自主开发的,是安全标记保护级安全操作系

*)国家自然科学基金资助项目(60473094)、国家杰出青年科学基金资助项目(60125205)。丁志军 博士生,讲师,主要研究方向为 Petri 网理论、并发处理;刘海峰 博士,主要研究方向为信息系统安全理论与技术;蒋昌俊 教授,博士生导师,主要研究方向为并发理论、模型验证、模糊推理。

统的重要组成部分。

2.1 审计系统的总体设计与实现

Linux 系统运行态分为用户态和核心态两种。用户程序只能通过系统调用陷入核心，才能存取系统资源(文件、目录、设备等)，运行完系统调用，又返回用户态。由此得到启发，如果在此处(称作审计点)增加审计控制，就可以成功地审计系统调用，也就可以成功地审计与安全有关的事件。另外，对于一些特权命令，如果只对其所调用的系统调用进行审计，则审计记录过于琐碎，不便于后期的审计分析，所以应该对其建立相应的审计事件，以便单独审计。

系统初启时，创建审计进程；在审计点收集所需的审计信息，然后调用审计进程，将审计信息记录、转储和归档。在用户界面上，则建立相应的操作命令，以灵活地开启/关闭审计机制，选择和设置审计事件，查询和检索审计日志，并创建守时进程，定期自动清除审计日志等。审计系统的结构图如图 1。

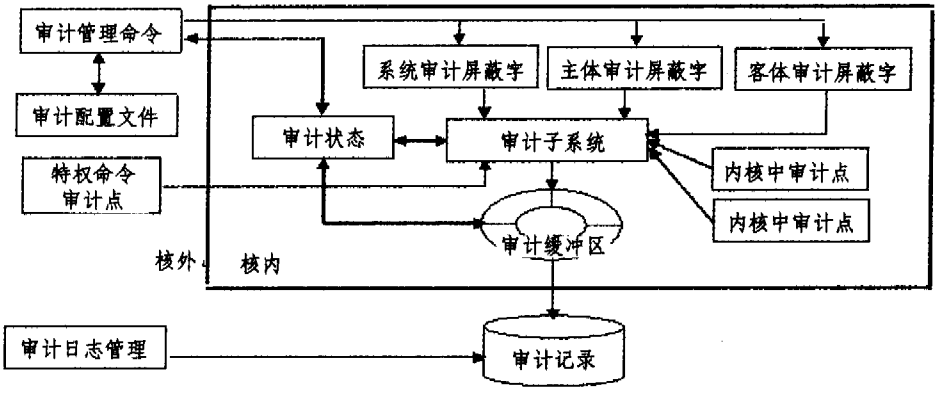

图 1 审计子系统结构图

待审计的用户活动按各自的性质不同，被视为不同的审计事件。审计事件是系统审计用户动作的最基本单位。常见的审计事件中有像 open，exec 这样的系统调用，它们跟系统的安全使用有关，所以要记录它们的执行情况。还有一些与系统安全关系密切的系统命令和特权命令(如 login，su，passwd 等)，也需要对其进行详细审计。

在对用户实施审计之前，先要确立审计标准，即需要审计哪些事件。这个问题可以从两方面来看：

(1) 从主体(用户和代表用户的进程)角度上看，系统要记录用户进行的所有活动，每个用户有自己的待审计事件集，称为用户事件标准，一旦其行为落入用户事件集，系统就会将事件信息记录下来；

(2) 从客体(文件、消息、信号量、共享区等)角度上看，系统要有能力记录关于某一客体的所有存取活动。为了有更好的针对性，在我们的实现中是先定义关于该对象的哪些操作事件要求被审计，即对象事件标准，再确定一个待审计的客体的强制存取(MAC)安全级范围(由一些连续的范围和离散的点构成)。只有属于该范围的客体，其对象事件标准才被审计。

在用户事件标准中，每个用户都需审计的公共部分称为基本事件集，它由审计管理员设定。基本事件集和用户的事件集做并集运算，结果才是真实的用户事件标准。

另外，有些事件与系统安全性关系非常大，任何时候都不应免于审计。为此，定义了一个固定的审计事件集，它是系统最基本的事件标准。在审计管理员设置或修改基本事件集时，固定事件集总是包含在系统基本事件集中。为了保障安全，固定审计事件通过硬编码固化在内核中，管理员不能改变它，除非修改内核程序。在我们的安全系统里，固定审计事件包括：有关审计系统自身运行的一组事件(审计开关、策略改变等)；失败的用户登录、添加删除用户帐号、分配 MAC 安全级等。

除了固定审计事件集，其他审计事件都是审计员可选的事件。通过 auditset 系统命令，审计员可以设置系统基本事件标准、用户事件标准及用户组事件标准。

2.2 缓冲区管理

审计数据来自于系统各处，在审计点搜集到的各种审计事件信息汇集到一起，等到信息量达到一定程度，就要将其写入物理介质(磁盘、磁带)，以便永久保存。由于外部存储介质的 I/O 操作十分耗时，为了提高整体效率，很自然地要引入缓冲。审计缓冲区机制的总体示意图如图 2。

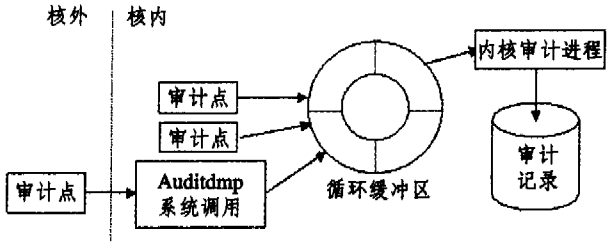

图 2 审计缓冲区机制的总体示意图

从图 2 看出，我们用多个相同大小的缓冲区构成缓冲池，以便提高并发度。这样，当某进程写某个缓冲区时，审计驻留进程可以读取其它已经准备好的缓冲区。至于缓冲区大小和数目，可由审计管理员根据实际系统的运行状况和审计范围灵活地调整。

缓冲区作为系统中审计数据的集散地，各审计点独立地将数据写入其中，再由一个系统驻留服务进程定期将缓冲区内容倾倒到文件中。缓冲区作为临界区，需要协调进程间的同步和互斥。不难观察，这是较常见的多个生产者和一个消费者的关系：各审计点是审计信息的生产者，但在生产前先要获得缓冲区资源；写磁盘的审计驻留进程是审计信息的消费者，消费完毕将释放缓冲区资源。具体实现中，我们利用 Linux 内核机制中的等待队列和信号量实现了进程间的同步和互斥。

多个生产者(adt_recwr，为审计点往循环缓冲区写的进程)要互斥使用当前写缓冲区，引入内核 semaphore 结构的变

量 bufwrite_lock 作为互斥锁。写前先申请锁，若已被别的进程占据，则睡眠等待；得到锁才操作缓冲区，操作完毕再释放锁。生产者和消费者(adt_bufdump，为循环缓冲区往磁盘上写的进程)之间的同步通过两个等待队列 wait_buf_read，wait_buf_write 进行。adt_recwr 写审计记录时，先看当前写缓冲区是否足够容纳待写数据，不够则将缓冲区标记为满，并唤醒等待在 wait_buf_read 上的写磁盘进程，写指针下移，若下一个缓冲区标记并不为空，表明写磁盘的进程操作已滞后，这时应该等待在 wait_buf_write 队列上。等到有可用的空缓冲区时，adt_recwr 被唤醒并开始写入记录数据。写磁盘的进程通常等待在 wait_buf_read 队列上。某个缓冲区满时被唤醒，它调用 adt_bufdump 将当前读指针有已满缓冲区逐一写入磁盘，并每次唤醒 wait_buf_write 队列，使 adt_recwr 可以继续写审计记录。在该写磁盘的缓冲区都已写入后，它再次睡眠在 wait_buf_read 队列。

3 审计缓冲区的形式化模型

3.1 时序 Petri 网

Petri 网模型是异步并发系统建模与分析的一种重要工具[9,10]。时序 Petri 网是 Suzuki[8]在 1985 年提出的一种新网子类。通过在 Petri 网中引入时序逻辑操作，如○(next)、□(henceforth)、◇(eventually)和 until 等，利用时序逻辑公式控制或限定 Petri 网的变迁引发序列，描述系统事件之间的时序关系及系统的需求规范，反映出系统的基本性质。文[11,12]分别利用时序 Petri 网分析和验证了莲花链式握手协议与位协议。

下面，首先给出本文所用到的有关 Petri 网的概念。

一个 P/T 网是一个五元组 $N=(P,T;F,K,W)$，其中 P 和 T 称为库所集和变迁集($|P|=m,|T|=n$)；$F\subseteq(P\times T)\cup(T\times P)$是有向弧集；$K:P\to IN$ 称为库所容量；$W:F\to IN$ 称为弧的权。网的关联矩阵 $A=[a_{ij}]_{n\times m}$是一个 n 行 m 列矩阵，其中 $a_{ij}=W(t_i,p_j)-W(p_j,t_i)$。对于 $P\cup T$ 中的任意元素 x，${}^{\cdot}x=\{y\in P\cup T|+(y,x)\in F\}$和 $x^{\cdot}=\{y\in P\cup T|(x,y)\in F\}$分别为 x 的前集和后集。

N 为非负整数集，映射 $M:P\to N_0$ 称为网的一个标识。库所/变迁系统(P/T 系统)$\Sigma=(N,M)$，其中，N 又称为基网，M_0 是初始标识。变迁 $t\in T$ 成为标识 M 下使能的，当且仅当 $\forall p\in{}^{\cdot}t:M(p)\geqslant W(t,p)$，记作 $M[t>$；在 M 下使能的变迁 t 可以引发发射，引发后得到后继标识，$M'=M+A$，记作 $M[t>M'$，称 M'为从 M 直接可达的。如果存在变迁序列 $\sigma=(t_1,t_2,\cdots,t_k)$和标识 M_k，使得 $M[\sigma>M'$，则称 M_k 为从 M 可达的。从 M 可达的一切标识集合记用 $R(M)$，约定 $M\in R(M)$。

根据上述定义，我们给出建立在库所/变迁系统基础上的时序 Petri 网的定义。

定义 1 设 $TPN=(PN,f)$是时序 Petri 网，其中 PN 是一个库所/变迁系统的基网，f 是具有如下语法的公式：

1)(库所 p 拥有托肯)、(变迁 t 使能)、(变迁 t 引发)都是原子命题，原子命题都是公式。

2)若 f 和 g 是公式，$f\wedge g$，$f\vee g$，$\neg f$，$f\supset g$，$\bigcirc f$，$\Box f$，$\Diamond f$，f until g 都是公式。其中，∧(AND)，∨(OR)，¬(NOT)，⊃(implication)是布尔连接符。$\bigcirc f$ 表示在下一状态为真。$\Box f$ 表示自当前状态以后的所有状态(包括当前状态)下 f 均为真。$\Diamond f$ 表示自当前状态以后的所有状态(包括当前状态)中，总存在至少一个状态，使得 f 为真。f until g 表示 f 为真，直至 g 为真。

设 M 是网 PN 的一个标识，$L(PN,M)$表示由 M 可以引发的有限序列集，$L^{\infty}(PN,M)$表示由 M 可以引发的无限序列集。$L^{\omega}(PN,M)=L(PN,M)\cup L^{\infty}(PN,M)$则表示可由 M 引发的所有可能序列集。设 $\alpha\in L^{\omega}(PN,M)$，$\forall i$，有 $0\leqslant i\leqslant|\alpha|$。$\alpha=\beta_i\gamma_i$，其中，$|\beta_i|=i$，设 $M_i:M[\beta_i>M_i$。$\langle M,\alpha\rangle\models f$ 表示在标识 M 和引发序列 α 下，f 为真。

$\langle M,\alpha\rangle\models p$ iff $M(p)>0$。

$\langle M,\alpha\rangle\models t$ iff $\alpha\neq\lambda$ 且 $t=\beta_1$。$t\in T$

$\langle M,\alpha\rangle\models\uparrow t$ iff 变迁 t 在标识 M 下使能。

$\langle M,\alpha\rangle\models f_1\cdot f_2$ iff $\langle M,\alpha\rangle\models f_1$ and $\langle M,\alpha\rangle\models f_2$。

$\langle M,\alpha\rangle\models f_1\vee f_2$ iff $\langle M,\alpha\rangle\models f_1$ or $\langle M,\alpha\rangle\models f_2$。

$\langle M,\alpha\rangle\models\neg f_1$ iff not $\langle M,\alpha\rangle\models f_1$。

$\langle M,\alpha\rangle\models f_1\supset f_2$ iff 若$\langle M,\alpha\rangle\models f_1$ 有$\langle M,\alpha\rangle\models f_2$。

$\langle M,\alpha\rangle\models\Box(f_1\supset\Diamond f_2)$ iff $\forall i,0\leqslant i\leqslant|\alpha|$，若$\langle M,\beta_i\rangle\models f_1$ 有$\langle M_i,\gamma_i\rangle\models f_2$。

$\langle M,\alpha\rangle\models\bigcirc f$ iff $\alpha\neq\lambda$ and $\langle M_1,\gamma_1\rangle\models f$。

$\langle M,\alpha\rangle\models\Box f$ iff $\forall i,0\leqslant i\leqslant|\alpha|$，有$\langle M_i,\gamma_i\rangle\models f$。

$\langle M,\alpha\rangle\models\Diamond f$ iff $\exists i,0\leqslant i\leqslant|\alpha|$，有$\langle M_i,\gamma_i\rangle\models f$。

$\langle M,\alpha\rangle\models f_1$ until f_2 iff ($\forall i,0\leqslant i\leqslant|\alpha|$，有$\langle M_i,\gamma_i\rangle\models f_1$) or($\exists i,0\leqslant i\leqslant|\alpha|$，有$\langle M_i,\gamma_i\rangle\models f_2$ and $\forall j,0\leqslant j<i$，有$\langle M_j,\gamma_j\rangle\models f_1$)。

公式 f 对网 PN 上的变迁引发序列进行了限制。因此，只有满足公式 f 的那些序列才能发生。形式描述为：$TPN=(PN,f)$是一时序 Petri 网，M 为网 PN 的一个标识，$L(TPN,M)$表示时序 Petri 网 TPN 的所有由 M 开始的可引发序列，即

$$L(TPN,M)=L^{\omega}(N,M)\cap\{\alpha\in T\infty|\langle M,\alpha\rangle\models f\}$$

TPN 中标识 M 的可达标识集 $R(TPN,M)$定义为：

$$R(TPN,M)=\{M'|M[\beta>M'$$

其中 β 为 α 的前缀，且 $\alpha\in L(TN,M)\}$。

3.2 审计缓冲区的时序 Petri 网模型

现在，我们就利用时序 Petri 网为审计缓冲区管理方案进行建模。

在这里，不失一般性，我们构建了拥有 3 个审计点(库所 $aduitpoint_i$，$i=1,2,3$，下同)、3 个缓冲区(库所 $buffer_i$)所组成的缓冲池以及相关的互斥锁(库所 $lock_i$)和缓冲区读、写指针(库所 $readpoint_i$、$writepoint_i$)的审计缓冲区模型 $auditPN=(P,T;F,K,W)$，见图 3。审计点所搜集到的数据通过申请互斥锁，从而得到操作缓冲区的机会(即变迁 $entry\text{-}buf_i$ 的发生)。在操作缓冲区时，利用写、读指针的配合，将数据首先写入缓冲区中(变迁 $acp\text{-}buffer_i$ 的发生)，并进而写到磁盘上(变迁 $acp\text{-}disk_i$ 的发生)，以便永久保存。

另外需要说明的是，为了有效模拟审计点获得数据的实时性和不确定性，在模型中，我们利用 3 个源变迁 $get\text{-}autpt_i$ 来加以表示。

为了更清晰地反映系统的基本框架和实现机制，以及系统性质验证的需要，在建模中，我们忽略了量的描述(如缓冲区大小)，而专注于刻画系统的本质。规定所有库所的容量 $K=1$，即只反映系统中资源的“有”和“无”，同时规定弧上的权值 $W=1$。

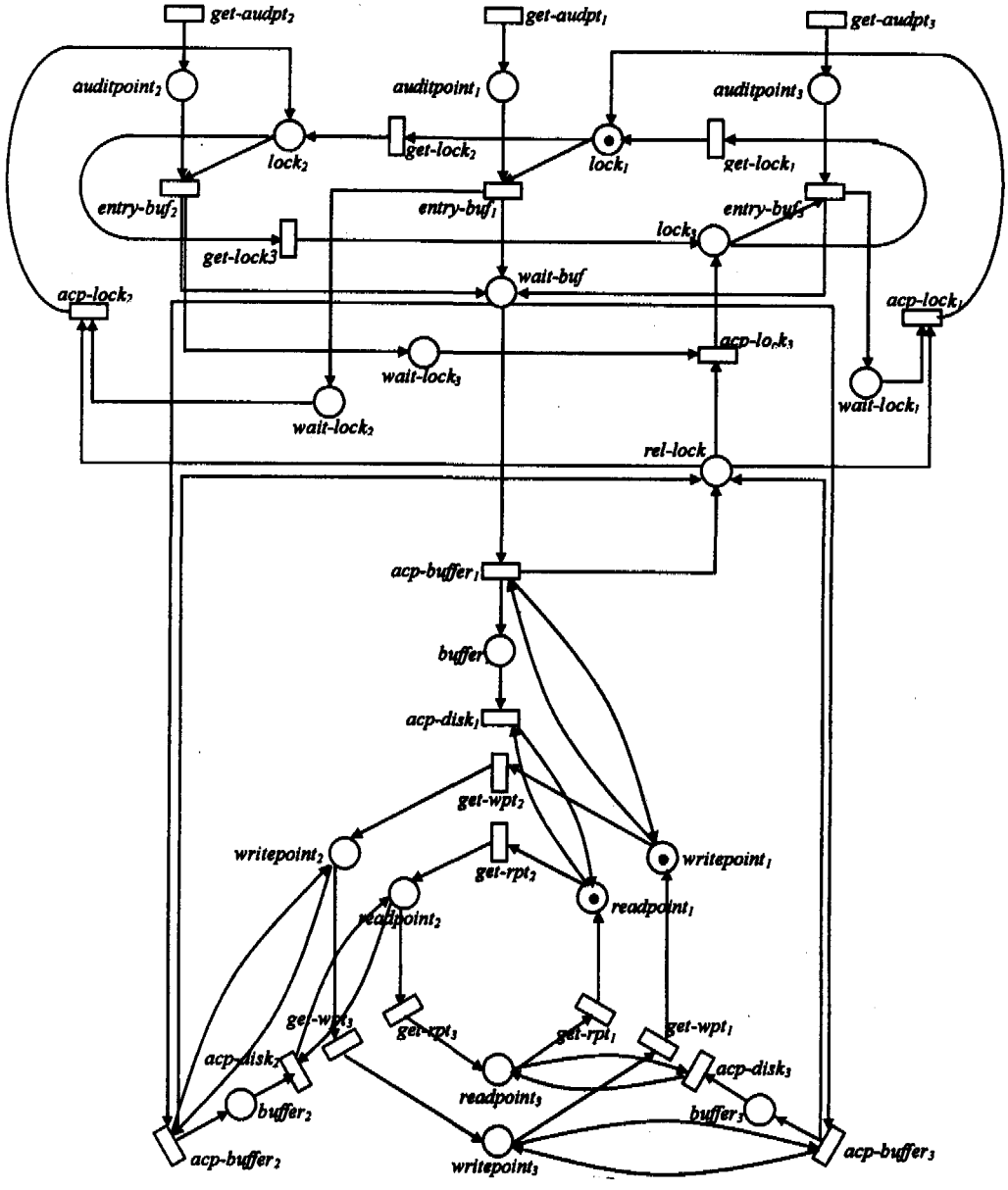

图 3　审计缓冲区模型

同样，库所 $lock_i$ 代表了互斥锁的状态。在实际实现过程中，我们是利用 Linux 内核机制来实现锁的分配的。为了模拟的需要，我们假定(它的)锁是顺序分配的。当然，若当前审计点没有数据需要操作，则锁可以被下一审计点得到。

模型中，我们分别利用库所 $readpoint_i$、$writepoint_i$ 表示缓冲区的读、写指针的实际状态。读、写指针之间的同步可归纳为如下规范：

1)若当前写指针所指的缓冲区为空，则等待写的审计点数据可以写入该缓冲区。

2)若当前写指针所指的缓冲区已满，则写指针移至下一缓冲区，同时唤醒读指针，进行写磁盘操作。

3)若下一缓冲区仍不为空，表明写磁盘的进程操作已滞后，这时写指针应该挂起等待，而读指针则依序将当前开始的所有已满的缓冲区写到磁盘上，直至操作完该缓冲区。然后唤醒写指针。

4)一般地，读指针在所指缓冲区满时被唤醒，并依次操作缓冲区。

5)为了保证缓冲区操作的正确性，我们要求读、写指针的顺序是不可颠倒的。

如下的 7 条公式就描述了读、写指针动作的时序关系，对应着读、写指针同步要求。公式 f_1 还刻画了有关审计点数据动作行为的时序要求。

$f_1=\Box(\uparrow t\supset\Diamond t)$，其中，

$t=\{acp\_buffer_i, entry\_buf_i, acp\_disk_i, acp\_audpt_i, get\_audpt_i\}, i=1,2,3$

$f_2=\Box wait\_buf\wedge writepoint_i\wedge buffer_i\supset\bigcirc get\_wpt_j$

其中，$\langle i,j\rangle=\{\langle 1,2\rangle,\langle 2,3\rangle,\langle 3,1\rangle\}$

$f_3=\Box(get\_wpt_i\supset(wait\_buf\wedge writepoint_i\wedge buffer_i$ until $wait\_buf\wedge writepoint_i\wedge\neg buffer_i))$

其中，$i=1,2,3$

$f_4=\Box(get\_wpt_i\supset\Diamond acp\_disk_k)\quad i,k=1,2,3$

$f_5=\Box(writepoint_i\wedge readpoint_j\supset\Box(writepoint_j\wedge readpoint_j))$ until $writepoint_j\cdot\neg readpoint_j))$

其中，$\langle i,j\rangle=\{\langle 1,2\rangle,\langle 2,3\rangle,\langle 3,1\rangle\}$

$f_6=\Box(readpoint_i\wedge writepoint_j\supset\Box(readpoint_j\wedge writepoint_j)$ until $readpoint_j\cdot\neg writepoint_j))$

其中，$\langle i,j\rangle=\{\langle 1,2\rangle,\langle 2,3\rangle,\langle 3,1\rangle\}$

$f_7=\Box(acp\_disk_k\supset\Diamond get\_rpt_j)$

其中，$i=1,2,3$

公式 f_8，f_9 则反映了系统对互斥锁动作的要求。

$f_8=\Box(\uparrow t\supset\Diamond\rightarrow\uparrow t)$，其中，

$t=\{get\_lock_i,acp\_lock_i\}$，$i=1,2,3$

$f_9=\Box(auditpoint_i\wedge lock_i\supset auditpoint_i\wedge lock_i$ until $entry\_buf_i$

其中，$i=1,2,3$

综上所述，审计缓冲区的时序 Petri 网模型定义如下：

定义 2 审计缓冲区的时序 Petri 网模型为 $auditTPN=(auditPN,f)$，其中 $auditPN$ 是一个基网，$K=1$，$W=1$。f 是网上的时序逻辑公式：$f=f_1\wedge\cdots\wedge f_9$。初始标识 M_0 如图 3 所示，表示系统初始时互斥锁处于等待状态，而读、写指针均指向缓冲区一。

本文中的审计缓冲区是一个典型的有关缓冲区操作的并发系统。一般而言，对于并发系统，安全性和活性是保证其正确的两个基本性质[13]。对于审计缓冲区，所需满足的安全性和活性性质如下：

安全性。缓冲区的操作是互斥的，即任意两个审计点的数据必须互斥地进入同一缓冲区，以保证缓冲区数据的有效性。

活性。任一审计点的数据都能够写入缓冲区，并最终存储到磁盘上。

如果能够形式化地验证我们所设计的审计缓冲区满足如上的安全性和活性性质，就能保证实现方案的可靠性和正确性，从而也就满足了系统设计的规范要求。

在第 4 部分，我们就通过对时序 Petri 网模型的分析，形式化地验证该模型能够满足安全性和活性性质，证明了实际系统的正确性。

4 验证

通过对模型的分析不难发现，互斥锁的设置有效保证了系统的安全性。在任一时刻，只可能有一个审计点能够获得互斥锁，从而得到操作缓冲区的机会，从而满足了缓冲区操作的互斥性。

系统的安全性只是一种保障，而一个系统总是具有一定功能，解决一定问题的，这就表现为系统的活性性质。下面，我们就证明该模型同样也满足活性性质。

定理 1 设时序 Petri 网 $auditTPN=(auditPN,f)$ 如定义 2 所述，对于 $\forall\alpha\in L(TPN,M)$，有

1) $\langle M_0,\alpha\rangle\models\Box(wait\_buf\supset\Diamond\rightarrow wait\_buf)$

2) $\langle M_0,\alpha\rangle\models\Box(lock_i\supset\Diamond lock_j)$

3) $\langle M_0,\alpha\rangle\models\Box(auditpoint_k\supset\Diamond auditpoint_k\wedge lock_k$

4) $\langle M_0,\alpha\rangle\models\Box(auditpoint_k\supset\Diamond entry\_buf_k)$

5) $\langle M_0,\alpha\rangle\models\Box(vbyffer_k\supset\Diamond acp\_disk_k)$

其中，$\langle i,j\rangle=\{\langle 1,2\rangle,\langle 2,3\rangle,\langle 3,1\rangle\}k=1,2,3$

证明：先证明 1)式。不妨设 $\exists M\in(auditTPN,M_0)$，有 $M_0[\beta>M$，其中 $\alpha=\beta\cdot\gamma$。有 $M(wait\_buf)=1$，设当前缓冲区写指针指向缓冲区一，即 $M(write\_point_1)=1$。分以下两种情况讨论：

case1：设当前标识下，库所 $buffer_1$ 中无托肯，表示缓冲区一为空，即 $M(buffer_1)=0$。利用时序逻辑形式化描述为：

$\langle M,\gamma\rangle=\models wait\_buffer\wedge writepoint_1\wedge\rightarrow buffer_1$

$\langle M,\gamma\rangle\models wait\_buffer\wedge writepoint_1\wedge\rightarrow buffer_1\supset\uparrow acp\_buffer_1$

显然，根据时序逻辑公式 f_1 有

$\langle M,\gamma\rangle\models\uparrow acp\_buffer_1\supset\Diamond acp\_buffer_1$。

因此有

$\langle M,\gamma\rangle\models wait\_buffer\wedge writepoint_1\wedge\rightarrow buffer_1\supset\Diamond acp\_buffer_1$

即 $\langle M,\gamma\rangle\models wait\_buffer\wedge writepoint_1\wedge\rightarrow buffer_1\supset\Diamond(\rightarrow wait\_buffer\wedge writepoint_1\wedge buffer_1)$

因此结论成立。

case2：设当前标识下，库所 $buffer_1$ 中有托肯，表示缓冲区一已满，即 $M(buffer_1)=1$。

利用时序逻辑形式化描述为：

$\langle M,\gamma\rangle\models wait\_buffer\wedge writepoint_1\wedge buffer_1$

显然，由于给出的网模型是一个 P/T 系统，因此根据 P/T 系统的引发规则，知此时变迁 acp_buffer_1 无法引发，则根据时序逻辑公式 f_2，有

$\langle M,\gamma\rangle\models wait\_buffer\wedge writepoint_1\wedge buffer_1\supset\bigcirc get\_wpt_2$

因此有 $\langle M,\gamma\rangle\models\Box(get\_wpt_2\supset\bigcirc wait\_buffer\wedge writepoint_2\wedge buffer_1)$

即 $\exists M'$：$M[\sigma>M'$，有

$M'(wait\_buffer)=M'(writepoint_2)=M'(buffer_1)=1$

这里，$\gamma=\sigma\cdot\delta$。现在，根据库所 $buffer_2$ 中的托肯情况分别加以讨论：

1) 假设 $M'(buffer_2)=0$，表明缓冲区二中未放入数据，则

$\langle M',\delta\rangle\models wait\_buffer\wedge writepoint_2\wedge\rightarrow buffer_2\supset\uparrow acp\_buffer_2$

根据时序逻辑公式 f_1，

$\langle M',\delta\rangle\models wait\_buffer\wedge writepoint_2\wedge\rightarrow buffer_2\supset\Diamond acp\_buffer_2$

即 $\langle M',\delta\rangle\models wait\_buffer\wedge writepoint_2\wedge\rightarrow buffer_2\supset\Diamond(\rightarrow wait\_buffer\wedge writepoint_1\wedge buffer_2)$

命题得证。

2) 假设 $M'(buffer_2)=1$，表明此时缓冲区二也已经满了，则根据时序逻辑公式 f_3 有

$\langle M',\delta\rangle\models\Box wait\_buf\wedge writepoint_2\wedge buffer_2$ until $wait\_buf\wedge writepoint_2\wedge\rightarrow buffer_2$

不妨假设当前读指针指向缓冲区 $k(k=1,2,3)$，显然读缓冲区的进程落后于写缓冲区的进程。因此，根据公式 f_4，$\langle M',\delta\rangle\models\Diamond get\_disk_k$，即将读指针所指的当前缓冲区中的数据写到磁盘上。再相继利用公式 f_7、f_1 进行网的运行，最终有

$\langle M',\delta\rangle\models wait\_buf\wedge writepoint_2\wedge buffer_2\supset\Diamond wait\_buf\wedge writepoint_2\wedge readpoint_2\wedge\rightarrow buffer_2$

即读指针依次读取缓冲区中的数据，直至写指针所指的当前缓冲区二。

$\langle M',\delta\rangle\models\Box wait\_buf\wedge writepoint_2\wedge readpoint_2\wedge\rightarrow buffer_2\supset\uparrow acp\_buffer_2)$

根据公式 f_1，得到

$\langle M',\delta\rangle\models\Box wait\_buf\wedge writepoint_2\wedge readpoint_2\wedge$

$$\neg buffer_2 \supset \neg wait\_buf \wedge writepoint_2 \wedge readpoint_2 \wedge buffer_2$$

命题得证。综上所述,1)式成立。

按照1)式的证明方法,可以证明其它各式,这里就不一一证明了。

1)式表明,若有数据等待写入缓冲区,则它总能最终进入缓冲区;2)式则表明互斥锁的发放是按序发放的,因而任一审计点都能得到互斥锁,即有3)成立;4)表明任一审计点的数据都能申请到互斥锁,并且可以执行缓冲区入口操作;5)式表明任一缓冲区中的数据都能写入到磁盘上。

根据定理1的结论,我们就可以证明该缓冲区的设计是满足活性要求的,即任一审计点的数据都能够写入缓冲区,并最终存储到磁盘上。定理2的结论就反映了这一点。

定理2 设时序Petri网$auditTPN=(auditPN,f)$如定义2所述,对于$\forall \alpha \in L(TPN,M)$,有

$$\langle M_0,\alpha\rangle \models \Box(auditpoint_i \supset \Diamond acp\_disk_k)$$

其中,$i,k=1,2,3$

证明:根据4)式知,

$$\langle M_0,\alpha\rangle \models \Box(auditpoint_i \supset \Diamond entry\_buf_i)$$

由网结构,得到

$$\langle M_0,\alpha\rangle \models \Box(entry\_buf_i \supset \bigcirc wait\_buf)$$

再利用1)式,有

$$\langle M_0,\alpha\rangle \models \Box(wait\_buf \supset \Diamond \neg wait\_buf)$$

根据网结构,可知

$$\langle M_0,\alpha\rangle \models \Box(wait\_buf \supset \Diamond acp\_buffer_k)$$

$$\langle M_0,\alpha\rangle \models \Box(acp\_buffer_k \supset \bigcirc buffer_k)$$

综合上式,得到

$$\langle M_0,\alpha\rangle \models \Box(auditpoint_i \supset \Diamond buffer_k)$$

最后,结合5)式,可得

$$\langle M_0,\alpha\rangle \models \Box(auditpoint_i \supset \Diamond acp\_disk_k)$$

命题得证。

结论 本文引用时序Petri网,对审计缓冲区进行建模、分析和验证,取得了较好效果。一方面,利用Petri网描述系统物理结构。另一方面,通过时序逻辑公式有效地反映了系统事件间的时序关系,描述了系统的规范。更为重要的是,借助于时序Petri网,可以通过时序逻辑的推导、演绎对系统性质进行严格的理论验证,以达到正确性证明的目的,而网的运行则可以简化这种推导过程并使之直观化。

参 考 文 献

1 National Computer Security Center. A Guide to Understanding Audit in Trusted Systems [S]. Jul. 1987

2 US DOD. Trusted Computer Systerm Evaluation Criteria [S]. Dec. 1985

3 刘海峰,卿斯汉,刘文清. 安全操作系统审计的设计与实现[J]. 计算机研究与发展,2001,38(10):1262~1268

4 GB 17859-1999. 计算机信息系统安全保护等级划分准则[S]

5 刘震,丁红兵,主编. 网络安全、审查与控制——WindowsNT Server安全性专辑[M]. 学苑出版社

6 Grottola M G. The Unix Audit: Using Unix to Audit Unix [M]. McGraw-Hill Inc, 1993

7 Axelsson S, Lindqvist Ulf, Gustafson Ulf. An Approach to UNIX Security Logging [A][C]. In: Proccedings of the 21st National Information Systems Security Conference, 1998. 62~75

8 Suzuki I. Fundamental properties and application of temporal Petri nets [A][C]. In: Proc. 9thAnnu Conf Information Sciences and Systems, Baltimore, MD: Johns Hopking Univ, 1985. 641~646

9 Peterson J L. Petri Net Theory and the Modeling of Systems [M]. Englewood Cliffs, N J. Prentice-Hall, 1981

10 Murata T. Petri nets: Propertier, analysis and applications [A] [C]. Proc of the IEEE, 1989, 77, (4): 541~580

11 Suzuki I, Lu H. Temporal Petri Nets and their application to modeling and analysis of a handshake daisy chain arbiter[J]. IEEE Trans Comput, 1989, C-38(5): 696~704

12 Suzuki I. Formal analysis of the alternating bit protocol by temporal Petri nets[J]. IEEE Transactions on Software Engineering, 1990, 16(11): 1273~1281

13 袁崇义. Petri网原理[M]. 北京:电子工业出版社,1998

(上接第48页)

结构等网络层关心的特征。

两种休眠方式有机结合,能够充分发挥采用关闭节点射频技术在节省能量中的作用,减少节点射频在不需要时工作带来的电源浪费,同时将关闭节点射频对网络连通性和QoS性能的影响减至最小,最大化网络生存时间。

结论 本文针对自组网设计中,通过关闭节点射频节省能量研究进行了探讨。在分析了关闭射频在节省能量中的重要作用和介绍了已有的具有代表意义的设计方案后,本文分析了关闭部分节点射频对网络可能产生的影响;进一步讨论了在关闭节点射频设计中需要考虑的关闭时间、控制信息交换、节点密度、移动性和代表节点等相关因素;在此基础上,提出了节点关闭射频的条件,最后介绍了我们的设计方案。

我们将在后续工作中把本文提出的关闭射频设计方案应用到实际的自组网系统中,指导其中的关闭节点射频算法实现,最终达到优化自组网应用系统功耗特征,拓展自组网应用范围的目的。

参 考 文 献

1 英春, 史美林. 自组网体系结构研究[J]. 通信学报,1999,20(9)

2 Powers RA. Batteries for low power electronics. Proceedings of the IEEE, 1995,83(4):687~693

3 Stemm M, Katz R H. Measuring and Reducing Energy Consumption of Network Interfaces in Hand-Held Devices [J]. IEICE Transactions on Communications, 1997, E80-B(8): 1125~1131

4 Chen B, Kyle J K, Balakrishnan H, et al. Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks. ACM SIGMOBILE 7/01, 2002

5 Kasten O. Energy consumption. ETH-Zurich, Swiss Federal Institute of Technology, 2001

6 Feeney L M, Nilsson M. Investigating the Energy Consumption of a Wireless Network Interface in an Ad Hoc Networking Environment, IEEE INFOCOM 2001

7 Xu Y, Bien S. Topology Control Protocols to Conserve Energy in Wireless Ad Hoc Networks [R]: [Technical Report #6]. Center for Embedded Networked Sensing (CENS), 2003

8 Schurgers C, Tsiatsis V, Srivastava M. STEM: Topology Management for Energy Efficient Sensor Networks. IEEEAC paper, 2002

9 Singh S, Raghavendra C S. PAMAS-Power Aware Multi-Access protocol with Signalling for Ad Hoc Networks [A]. ACM SIGCOMM Computer Communication Review, 1998,28(3):5~26