高技术通讯 2009 年 第 19 卷 第 6 期:572 ~ 579

doi:10.3772/j.issn.1002-0470.2009.06.004

分布式任务关键系统生存性自动分析与验证①

王 健②* 王慧强* 赵国生* **

(*哈尔滨工程大学计算机科学与技术学院 哈尔滨 150001)

(**哈尔滨师范大学网络中心 哈尔滨 150080)

摘 要 提出了一种应用概率模型检测技术进行分布式任务关键系统生存性的量化分析研究方法。该方法对攻击者和系统的交互行为进行精简抽象,在此基础上使用 PRISM 高级语言构造连续时间马尔可夫链系统概率模型。针对不同程度的攻击故障及系统服务水平,以连续随机逻辑建立系统生存性的形式化规约。借助概率模型检测工具 PRISM 对模型进行统计和验证,并图形化地表示出系统生存性的自动分析结果。理论分析和实验结果验证了上述方法的合理性和有效性,这些结果可在理论上指导可生存系统的设计和实现。

关键词 生存性,概率模型检测,形式化规约,任务关键系统,量化分析

0 引 言

生存性是指在遭受攻击、故障或意外事故时,系统能够及时地完成关键任务的能力[1]。在生存性研究的诸多领域中,生存性分析,特别是量化分析是一个重要的基础工作,也是当前的研究热点之一。这一问题的研究,可为系统在各种环境下的服务性能准确地提供预报,进而在理论上指导可生存系统的设计和实现。已有的研究主要基于以下三种思路:(1)针对系统物理结构拓扑模型,把生存性的计算转化到图论问题空间进行讨论[2-4];(2)基于系统服务组件建模,在数学上表现为一个类似树的结构,通过评估涉及到的各个系统组件的生存性状况,从而得到该系统的生存性[5-8];(3)根据系统状态或其运行环境事件状态的变化来分析系统生存性[9-12]。其中,用模型检测对系统的状态模型进行分析是可生存性研究的一种较新方法。它可以自动产生故障场景,实现模型的自动验证,这种机器辅助的生存性分析技术将是今后的一个研究热点。但目前基于通用模型检测的生存性分析方法,其检测结果是对状态转换系统的可达性进行分析,仅返回 true 或 false,表示模型是否满足性质,无法对系统生存性进行更直接更准确的评估。针对这一缺点,本文提出了一种基于概率模型检测的分布式任务关键系统生存性分析方法。该方法结合了概率分析和通用模型检测技术,使用 PRISM 高级语言构造具有随机行为的系统概率模型,以时态逻辑建立模型需要满足性质的形式化规约,并利用 PRISM 工具实现系统生存性的自动分析和验证。

1 概率模型检测

概率模型检测是一种验证存在随机行为系统的形式化分析技术,包括构造系统的概率模型和建立性质的形式化规约两部分。在概率模型检测中,不是简单的标识转换存在与否,而是对状态间转换的概率进行编码,对转换的可能性进行数值计算。

1.1 概率模型

概率模型已经广泛地应用于软件和硬件系统的设计与分析中。针对不同的系统特征,对应如下几种概率模型[13]:(1)离散时间马尔可夫链(discrete-time Markov chains, DTMC),特征是只有概率选择;(2)马尔可夫判定过程(Markov decision processes, MDP),特征是既具有概率选择又具有非确定性选择;(3)连续时间马尔可夫链(continuous-time Markov chains, CTMC),特征是对连续时间进行建模,而且没有非确定性选择。

1.2 概率性质规约

概率性质规约的形式化方法包括概率计算树逻

① 863 计划(2007AA01Z401)和国家自然科学基金(90718003)资助项目。

② 女,1979 年生,博士生;研究方向:可信网络与可信计算;E-mail:wangjianlydia@163.com

(收稿日期:2008-06-04)

辑(probabilistic computational tree logic, PCTL)和连续随机逻辑(continuous stochastic logic, CSL)。PCTL是计算树逻辑(computational tree logic, CTL)的概率扩展,应用于DTMC和MDP中;CSL是在CTL和PCTL基础上的扩充,应用于CTMC中。

1.3　概率模型检测工具

目前,较为成功的概率模型检测工具是由英国伯明翰大学Kwiatkowska教授所负责的项目小组开发的PRISM工具[14],版本现为PRISM3.2,支持Windows、Linux等操作系统。PRISM工具的结构如图1所示,它支持DTMCs、MDPs和CTMCs三种概率模型,DTMCs和MDPs的性质规约为PCTL,CTMCs的性质则规约为CSL。

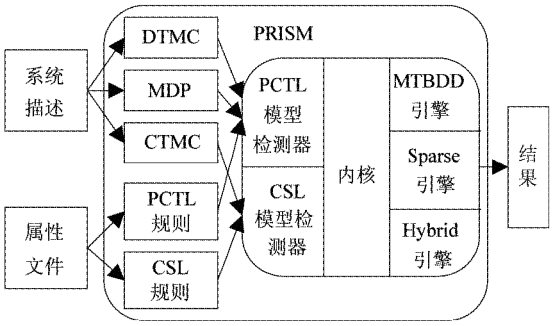


图1　PRISM结构图

2　系统概率模型

生存性技术的要求与可靠性、容错技术一样,都是在异常情况下维持关键服务的运行,但容错针对的是各类故障和外部灾害等往往是随机发生的事件,而生存性技术更重要的是应对有预谋的攻击。事实上,攻击者和被攻击系统之间的行为存在着交互性。一方面,攻击者试图探寻系统的安全漏洞,由此最大程度地破坏系统,阻止服务提供,而另一方面,被攻击系统则实施可生存机制(如自动响应、重配置等),尽力保证服务的持续提供,反映这种交互关系的系统拓扑结构如图2所示。系统具有以下特征:(1)系统中有多种通用的物理和逻辑资源,可以动态地给它们分配任务;(2)系统中分散的物理和逻辑资源通过计算机网络实现信息交换;(3)系统存在一个以全局方式管理系统资源的分布式操作系统;(4)系统中各联网计算机既合作又自治;(5)系统内部结构对用户是完全透明的。

针对上述分析,生存性分析要建立在攻击行为对系统造成影响的形式化描述的基础上。同时是对

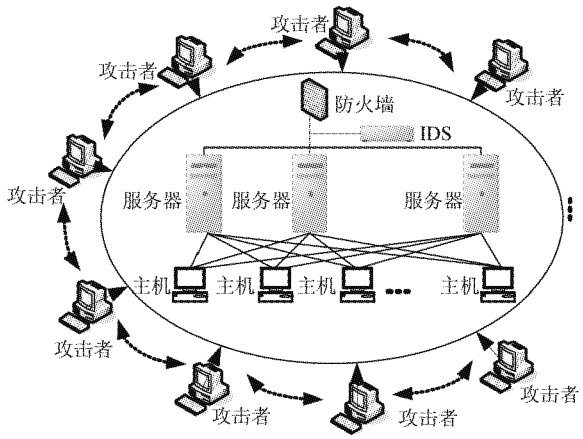


图2　系统拓扑结构图

实时系统建模,因此使用PRISM建模语言[15]构造CTMC系统概率模型,各个变量和模块的定义如下:

```
ctmc
const int M;
const int N;
const int master_server_max = M;
const int secondary_server_max = N;
const double master_server_fail = 0.001;
const double secondary_server_fail = 0.075;
module Master_Server
    master_server_number: [0..master_server_max] init master_server_max;
    masterserver: bool;
    [startMasterServer] ! masterserver & (master_server_number < M) -> 1: (masterserver' = true);
    [repairMasterServer] masterserver & (master_server_number < M) -> 1: (masterserver' = false) & (master_server_number' = master_server_number + 1);
    [] (master_server_number > 0) -> master_server_fail: (master_server_number' = master_server_number-1);
endmodule
module Secondary_Server
    secondary_server_number: [0..secondary_server_max] init secondary_server_max;
    secondaryserver: bool;
    [startSecondaryServer] ! secondaryserver & (secondary_server_number < N) -> 1: (secondaryserver' = true);
    [repairSecondaryServer] secondaryserver & (second
```

```
dary _ server _ number < N) -> 1: (secondaryserver' = false) & (secondary _ server _ number' = secondary _ server _ number + 1);
    [] (secondary _ server _ number > 0) -> secondary _ server _ fail: (secondary _ server _ number' = secondary _ server _ number-1);
endmodule
module Repair
    r : bool;
    [startMasterServer] ! r -> 10 : (r' = true);
    [startSecondaryServer] ! r -> 10 : (r' = true);
    [repairMasterServer] r -> 2: (r' = false);
    [repairSecondaryServer] r -> 2 : (r' = false);
endmodule
rewards "percent _ normal"
    true: 100 * (master _ server _ number + secondary _ sever _ number) / (M + N);
endrewards
rewards "below _ min"
    ! minimum: 1;
endrewards
```

模型中定义的常量 M 和 N 分别表示系统中主、备选服务器的数量，单个服务器失效比率通过系统日志分析等由 master _ server _ fail 和 secondary _ server _ fail 给出，表明其失效的平均时间。模型由三个模块组成，分别为 Master _ Server，Secondary _ Server 和 Repair，并定义了报酬结构。

3 系统生存性形式化规约

根据构造的系统 CTMC 概率模型，使用 CSL 来规格系统生存性。生存性定义中说明允许系统提供不同的预定义服务等级，同时并没有限制某一种类型的攻击或故障。

考虑到能够正常运行的服务器数量决定了系统在处理用户请求时的服务性能，依赖于可用的不同服务器数量定义了两个服务等级：premium 和 minimum。且在各服务等级下，保证至少有一台主服务器能够提供服务。CSL 描述如下：

```
service = minimum | premium
minimum = secondary _ server _ number > = (0.25 * N) & master _ server _ number > = M/2
premium = secondary _ server _ number > = (0.75 * N) & master _ server _ number = M
```

不同类型的攻击或故障会对系统造成不同的影响，比如，当系统遭受到较轻的攻击后在很短的时间内即可恢复到完全服务状态，而当系统遭受到较严重的攻击后需要快速地恢复到一个基本服务等级，再经过相对较长的时间间隔恢复到完全服务状态。不关心具体的攻击或故障类型，只考虑其给系统带来的影响，定义三个攻击故障级别：light _ disaster，medium _ disaster 和 severe _ disaster。CSL 描述如下：

```
disaster = light _ disaster | medium _ disaster | severe _ disaster
light _ disaster = (master _ server _ number = 1) & (second dary _ server _ number < = 3 * N/4) & (secondary _ server _ nu mber > N/2)
medium _ disaster = (master _ server _ number = 0) & (second dary _ server _ number < = N/2) & (second dary _ server _ nu mber > N/4)
severe _ disaster = (master _ server _ number = 0) & (second dary _ server _ number < N/4) & (secondary _ server _ numb er > 0)
```

生存性要求系统在任何攻击故障状态下都可以及时地提供预定义等级的服务，并在一定时间内恢复到完全服务状态，因此系统生存性的 CSL 可描述为：P > = p [true U < = T "service" {"disaster"} {min}]。结合服务及攻击故障等级的定义，文中定义了 3 种不同程度的攻击故障和 2 种服务等级，将产生 6 个生存性形式化规则。

4 系统生存性分析与验证的 PRISM 实现

4.1 模型统计与稳态概率计算

PRISM 工具使用二叉决策图(binary decision diagrams, BDDs)和多端二叉决策图(multi-terminal binary decision diagrams，MTBDDs)来实现，并且在进行数值计算时支持三种不同的模型检测引擎，包括基于符号模型检测的 MTBDD、基于稀疏矩阵的 Sparse 以及综合两者的混合方法 Hybrid，用于分析概率系统的相关概率统计性质。

表 1 给出了本文所建系统生存性模型的相关统计信息，包括表示该模型的 CTMC 中状态数和非零转换数，模型的构建时间及确定可达状态需要的迭代次数。

表 1　系统模型统计(M=2)

N	模型		构建	
	状态	NNZ	时间(s)	迭代次数
4	37	96	0.00	8
8	69	188	0.00	12
16	133	372	0.00	20
32	261	740	0.016	36
64	517	1476	0.031	68
128	1029	2948	0.047	132
256	2053	5892	0.203	260
512	4101	11780	0.485	516

NNZ:非零转换数

其中,当 M = 2,N = 4 时,系统模型共有 37 个状态,各状态及稳态概率如表 2 所示。

表 2　状态及稳态概率(M=2, N=4)

状态	稳态概率
0:(0,false,0,false,false)	2.6792584473305365E-15
1:(0,false,0,true,true)	5.84968977825241E-12
2:(0,false,1,false,false)	6.524754476500103E-13
·	·
·	·
·	·
35:(2,false,3,true,true)	0.03579005113598445
36:(2,false,4,false,false)	0.9544010310947552

使用 JOR 方法计算稳态概率,表 3 分别给出了 MTBDD,Sparse 和 Hybrid 三种引擎所需的时间及迭代次数。

表 3　稳态概率计算(M=2)

N	每次迭代时间(s)			迭代次数
	MTBDD	Sparse	Hybrid	
4	0.11	0.00	0.00	75
8	0.17	0.00	0.00	65
16	0.5	0.00	0.00	109
32	1.16	0.00	0.00	203
64	3.63	0.01	0.01	394
128	13.05	0.02	0.03	779
256	35.02	0.14	0.25	1482
512	101.98	0.70	1.75	2332

从表中可知,MTBDD 引擎适用于大规模及结构化的模型,检测速度慢,但实际占用内存少;Sparse 引擎适用于规模较小的模型,检测速度比 MTBDD 快,但需占用较多内存;Hybrid 则在前两者之间进行了折衷,是 PRISM 工具默认的模型检测引擎。

使用稳态概率计算系统生存性如下属性:(M = 2, N = 16)

(1)S = ? [“premium”]:表示系统将提供 premium 服务等级的长期运行概率,返回结果为“Result: 0.9993765531342764”。

(2) S = ? [! “minimum”]表示系统提供低于 minimum 服务等级的长期运行概率,返回结果为“Result: 3.366440534193837E-7”。

4.2　模型检测

利用概率模型检测器 PRISM,对系统生存性相关属性进行如下分析与验证:

(1)P > = 1 [true U “premium”]:表示系统在未来某一时刻总能够(大于等于 1 的概率)提供 premium 等级服务,模型检测时间由表 4 给出。

表 4　模型检测时间(M=2)

N	验证预计算	
	时间 (s)	迭代次数
4	0.00	24
8	0.00	36
16	0.015	60
32	0.032	108
64	0.062	204
128	0.156	396
256	0.22	780
512	0.453	1548

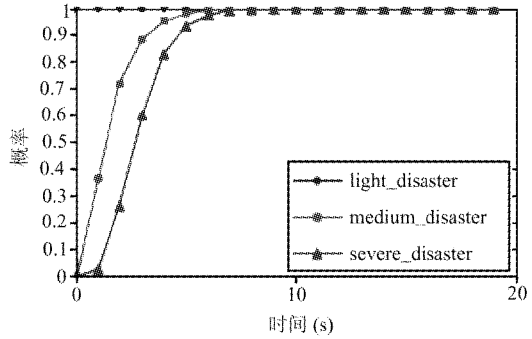
针对该属性,模型检测返回结果为“true (property satisfied in all states)”。

(2) P = ? [true U < = T “service” {“disaster”} {min}],其中,service = minimum|premium,disaster = light _ disaster|medium _ disaster|severe _ disaster:表示在 T 个时间单元内,系统从不同程度的攻击故障状态中恢复到不同服务等级的最小概率。如果这个最小概率在确定的生存性阈值之上,则说明当系统处于相应攻击故障状态中具有可生存性。使用 CSL 模型检测算法的属性实现得到的分析结果如图 3 所示。

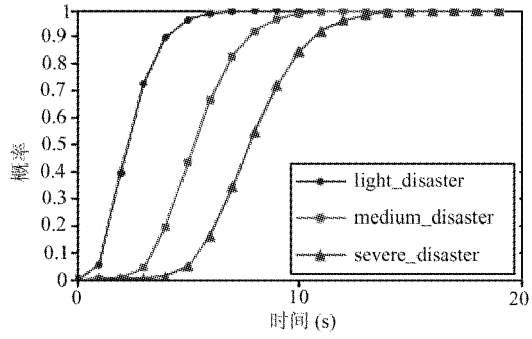
从图 3(a)中可以看出,当系统遭受“light _ disaster”级别的攻击故障时,在任何时刻都可以持续提供“minimum”等级的服务。当系统遭受“medium _ disaster”和“severe _ disaster”级别的攻击故障时,提供“minimum”等级服务的最小概率随着时间单元 T 增加而增大,且概率增长速度显著。

从图 3(b)中可以看出,三条最小概率曲线都沿 T 轴方向而呈上升趋势,即随着时间单元的增加,系统在各种攻击故障状态下提供“premium”服务的最小概率都随之增长。图中三条曲线的坡度不尽相同,其中,“light _ disaster”对应曲线的坡度较大,“medium _ disaster”对应曲线次之,“severe _ disaster”对应曲线坡度则较缓。但图中曲线的整体增长速度都较之图 3(a)中曲线缓慢。

这说明,系统所受攻击故障越严重,及时提供各等级服务的可能性就越小。但比较而言,系统在较短时间单元内,即可恢复提供“minimum”服务以保证服务的持续提供,而提供“premium”服务则需相对较长的时间单元。系统遭受攻击故障时,可通过降低所提供服务的种类、规模、性能等,以换取及时提供某些关键的、基本的服务,再通过修复/重配置等人工干预措施使系统逐渐重新恢复到较高及完全服务状态。



(a)恢复于 minimum 服务等级的最小概率

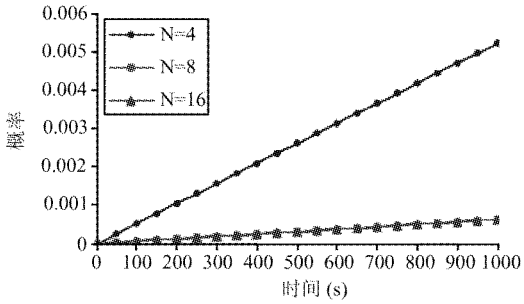


(b)恢复于 premium 服务等级的最小概率

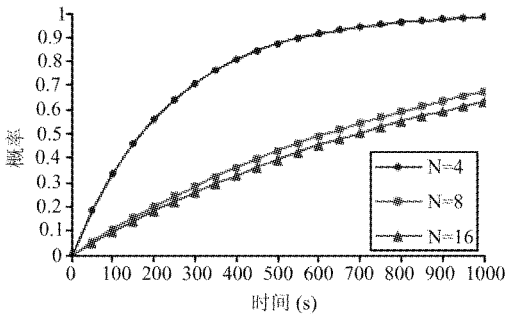
图 3 T 个时间单元内从不同攻击故障影响下恢复于不同服务等级的最小概率(M=2, N=16)

(3)P=?[true U<=T!“minimum”] & P=?[true U<=T!“premium”]:表示在 T 个时间单元内,系统提供服务的能力降到 minimum(premium)之下的概率,试验结果如图 4 所示。

从图中可以看出,系统提供服务的能力降到各服务等级之下的概率随着 N 的增大而减小。对(a),当 $N=4$ 时,在 1000 个时间单元内,系统提供服务的能力降到 minimum 之下的概率仅约为 0.0052。这说明一般情况下系统均能通过降低服务等级以持续提供服务。对(b),当 $N=4$ 时,系统提供服务的能力降到 premium 之下的概率相对较大,在 100 个时间单元内概率为 0.3366,在 1000 个时间单元内概率则达 0.9837。但此时系统仍能继续提供服务,系统生存性并未受到影响。



(a)系统服务 minimum 等级之下的概率



(b)系统服务在 premium 等级之下的概率

图 4 T 个时间单元内系统服务在各等级之下的概率(M=2)

这说明,随着 N 的增大,系统提供服务的能力降到各服务等级之下的概率逐渐减小,系统生存性逐渐增强。但当 $N=8$ 和 $N=16$ 时,图 4(a)、(b)中相应两条曲线都已非常接近。这说明此时,再增加 N 的值对整个系统的生存性不会提高很多,但 N 的增大给系统带来的代价却非常大。这为可生存系统设计与代价之间的权衡给予了指导,依据实际需求分析,参数 N 的选择应当适中,既能有效增强系统生存性,提高系统在攻击故障状态下的服务能力,又能减少系统开销。

(4)通过给模型中的状态或转换联系实值,定义了代价和报酬结构,使得实验中我们不仅能够得到某一确定方式下系统模型运行的概率,而且能推断出系统运行中与生存性相关的更广范围的量化测度。结合模型定义,进行如下检测:R{“percent _ nor-

mal"}=?[I=T{!"minimum"}{min}]:表示在时刻 T,当从最低服务等级 minimum 之下开始,系统中正常运行服务器的百分比,试验结果如图 5 所示。

R{"below_min"}=?[C<=T]:表示直到时刻 T,系统提供服务的水平低于 minimum 的期望时间,试验结果如图 6 所示。

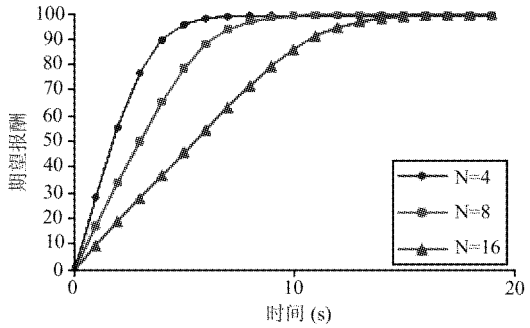


图 5　正常运行服务器百分比

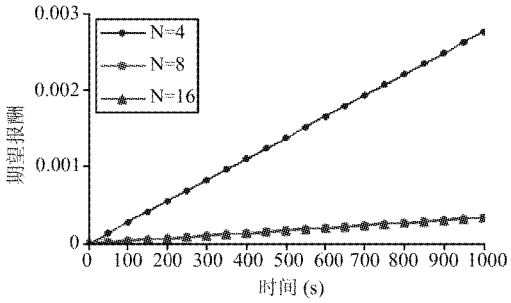


图 6　服务等级在 minimum 之下的期望时间

5　性能分析与相关工作比较

5.1　相关工作比较

文献[2-4]主要从系统结构角度考虑,将组成系统的硬件设施用图来表示,将系统生存性问题表述为图中节点之间的可达性问题。但由于系统物理结构直接依赖于路由技术、通信量管理等策略,其分析技术相对比较复杂。因其更多地考虑具体的物理结构,对小规模或者结构简单的系统尚可适用,一般应用较少。

文献[5-8]将系统看作实现特定服务的组件构成,通过评估涉及到的系统组件的生存性状况得到该系统的生存性。但因为是从系统服务出发,所以要详细地了解系统各个用户群的需求以便对系统服务进行等级划分。且一般只能通过概率统计等方法对系统生存性进行评估,在分析方面显得不足,不利于系统的改进。

文献[9-12]依据系统状态或运行环境事件状态的变化来分析系统的生存性。这种生存性分析方法能够表示出系统的状态转移关系,有利于对系统的动态行为进行分析,应用也更为广泛。但其直接涉及模型状态,很难从系统行为直接获得,需对系统运行情况进行抽象分析。同时,使用其分析较大规模系统会遇到状态空间爆炸的问题。

本文则针对系统状态模型难以从系统行为直接获得的问题,使用高级语言构造精确描述系统行为的生存性形式化模型,该模型对应着一个连续时间马尔可夫链。通过概率模型检测工具对系统的状态模型进行分析,以图形化地形式表示系统生存性定量分析结果并观测其变化趋势。不同于已有的研究工作,该方法使用形式化规范语言描述和推理基于计算机的系统,同时能够通过等价定理自动化减状态空间。在技术上通过精确的数学手段和强大的分析工具得到支持,可实现系统生存性的自动分析和验证,大大减少了在分析过程中人为因素的影响和分析费用。

表 5 总结了上述比较,给出了本文所做工作与已有工作的异同点。

表 5　相关工作比较

相关工作	建模角度	描述水平	数学表示	分析手段	分析能力	计算开销
文献[2-4]	物理结构	静态、具体,模型直观性较强	图	理论推导	弱,适用于结构简单的系统	大
文献[5-8]	服务组件	系统业务及内部结构的逻辑描述	树	理论推导	弱,仅能得到评估结果	大
文献[9-12]	状态变化	动态、抽象,但不易理解	有限自动机、马尔可夫链等	理论推导+分析工具	弱,返回评估值或定性分析结果	较大
本文工作	状态变化	动态、抽象,易理解	(潜在的)马尔可夫链	形式化分析工具	强,图形化地表示量化分析结果	小,分析过程自动化

5.2 性能评价

为了进一步测试验证提出的系统生存性分析方法在实际应用中的合理性和有效性，本文设计了一个仿真实验平台环境，其拓扑结构如图 7 所示。

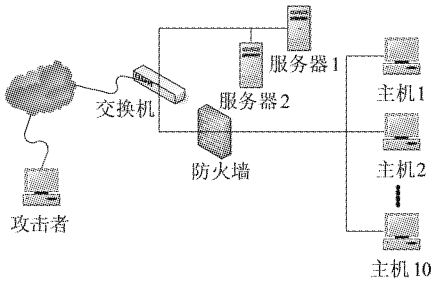


图 7 实验网络结构图

系统中有两台服务器，运行 Web 服务，其中，Server1 2.8GHz/512MB/WindowsXP，Server2 1.4GHz/256MB/ Windows2000。防火墙为运行 Linux 操作系统的工作站，实现网关的功能。防火墙内部有一个子网，子网中有 10 台主机。攻击者可以通过 Internet 从外部访问该实验系统，且只能和任一服务器建立直接连接。服务器之间互相独立，且服务器上存在可供攻击者利用的漏洞，漏洞互相独立。通过模拟攻击行为测试系统在不同外部环境下的可生存性能，每组实验测试时间为 60s，令 $\Delta t = 2$s，实验结果如图 8 所示。

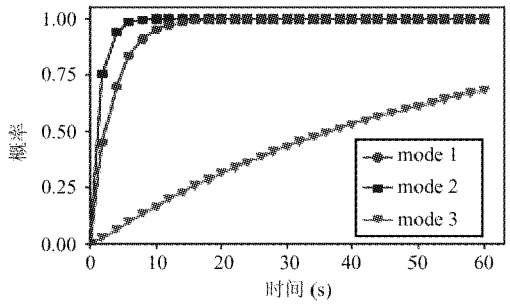


图 8 不同攻击模式下的系统生存性分析结果

攻击模式 1 和攻击模式 2 中，在 $T = 0$s 时分别利用 HGOD 对 Server1、Server2 实施单点 DOS 攻击，图中系统生存性曲线的变化基本反映了系统在该种模拟攻击模式下的生存状况。因为是单点攻击，系统中存在冗余服务器保证 Web 服务的持续提供，因此，不论是对 Server1 还是对 Server2 进行攻击，攻击的过程中系统生存性虽然有所降低，但并不显著。从图中可以看出，在攻击起始阶段，mode 1 和 model 2 所对应的系统生存性曲线均随着 T 的渐增而快速增长。其中，mode 1 所对应的系统生存性曲线在 $T = 2$s 时值为 0.451，$T = 10$s 即达到 0.95；mode 2 所对应的系统生存性曲线在 $T = 2$s 时值为 0.753，$T = 4$s 时即为 0.939。此后生存性曲线处于稳步上升态势，实验过程系统生存性曲线基本维持在 0.95 以上。因 HGOD 占用大量 CPU 资源，被攻击服务器将启动恢复功能对可疑进程进行定位并结束运行，使系统生存性恢复到原始正常水平。

攻击模式 3 中，在 $T = 0$s 时利用 SYN FLOOD 工具对 Server1、Server2 同时进行溢出攻击。观察发现，沿 T 轴方向系统生存性曲线虽呈上升趋势，但速度相对缓慢。当 $T = 20$s 时系统生存性值为 0.314，而当 $T = 60$s 时仅为 0.682。图中曲线的变化说明这种结构化攻击模式对系统生存性的影响较大，这恰好与实际情况相符。SYN FLOOD 不仅消耗大量的 CPU 资源，同时内存的占用率也大幅提升，当入侵者相互协同对服务器实施该种攻击时，攻击的成功率得到提升，系统提供 Web 服务的能力降低，表现为系统生存性的显著降低和较缓增长。随着恢复功能的启动，系统逐渐恢复到正常服务水平。

从图 8 的分析结果可以看出，仿真试验结果与理论分析基本相符，本文提出的系统生存性量化分析方法具有合理性和有效性。实验中还发现，生存性取值在 0.7 左右，都可保证 web 服务的持续提供。

6 结 论

本文提出了一种基于概率模型检测工具——PRISM 的任务关键系统生存性形式化建模与自动分析验证方法。该方法将图理论分析与数值求解、统计、仿真等技术相结合，以图形化地形式表示系统生存性定量分析结果并观测其变化趋势。实验结果与理论分析相符，表明系统生存性不仅与其所受攻击的严重性、攻击强度有关，还与系统的可恢复性等生存属性密切相关，是一个整体性的综合评估值。

有别于以往该领域所作的研究工作，本文提出的生存性分析方法具有以下优点：(1)以逻辑精确性为特色，建立系统生存性形式化模型；(2)不同于基于大规模仿真的近似求解，该方法可计算出精确的数值，且具有机器验证的无遗漏特性；(3)能够应用于系统设计和实现过程中的任何阶段，同时为其提供理论上的指导；(4)可实现系统的重复分析，较少依赖特定分析者的技术和能力；(5)基于计算机的工具所支持，实现分析和验证的自动化，减少在分析方

面的费用。

参考文献

[1] Ellison R J, Fisher D A, Linger R C, et al. Survivable network system: an emerging discipline: [Technical Report, CMU/SEI-97-TR-013]. Pittsburgh: Carnegie Mellon Software Engineering Institute, USA, 1997

[2] Louca S, Pitsillides A, Samaras G. On network survivability algorithms based on trellis graph transformations. In: Proceedings of the 4th IEEE Symposium on Computers and Communications, Red Sea, Egypt, 1999. 235-243

[3] Krings A W, Azadmanesh A. A graph based model for survivability applications. *European Journal of Operational Research*, 2005, 164(3): 680-689

[4] 包秀国, 胡铭曾, 张宏莉等. 两种网络安全管理系统的生存性定量分析方法. 通信学报, 2004, 25(9): 34-41

[5] Gao Z X, Ong C H, Tan W K. Survivability assessment: modeling dependencies in information systems. In: Proceedings of the 4th IEEE/CMU/SEI Information Survivability Workshop, Vancouver, Canada, 2001

[6] Hevner A, Linger R, Sobel A, et al. The flow-service-quality framework: unified engineering for large-scale, adaptive systems. In: Proceedings of the 35th Annual Hawaii International Conference on System Sciences, Hawaii, USA, 2002. 4006-4015

[7] 郭渊博, 马建峰. 分布式系统中服务可生存性的定量分析. 同济大学学报, 2002, 30(10): 1190-1193

[8] Casey F, Chen Y L, Wang X Y, et al. Survivability analysis of distributed systems using attack tree methodology. In: Proceeding of IEEE Military Communications Conference, Atlantic City, New Jersey, 2005. 583-589

[9] Jha S, Wing J, Linger R, et al. Survivability analysis of network specifications. In: Proceedings of the 2000 International Conference on Dependable Systems and Networks, New York, USA, 2000. 613-622

[10] Jha S, Wing J M. Survivability analysis of network System. In: Proceedings of the 23rd International Conference on Software Engineering, Toronto, Canada, 2001. 307-317

[11] McDermott J. Attack-potential-based survivability modeling for high-consequence systems. In: Proceedings of the 3d IEEE International Workshop on Information Assurance, College Park, Maryland, 2005. 119-130

[12] Dong S K, Khaja M S, Jong S P. A framework of survivability model for wireless sensor network. In: Proceedings of the 1st International Conference on Availability, Reliability and Security, Vienna, Austria, 2006. 515-522

[13] Kwiatkowska M. Model checking for probability and time: from theory to practice. In: Proceedings of the 18th Annual IEEE Symposium on Logic in Computer Science, Ottawa, Canada, 2003. 351-360

[14] Kwiatkowska M, Norman G, Parker D. PRISM 2.0: a tool for probabilistic model checking. In: Proceedings of the 1st International Conference on Quantitative Evaluation of Systems, Enschede, Netherlands, 2004. 322-323

[15] Kwiatkowska M, Norman G, Parker D. PRISM users' guide. http://www.cs.bham.ac.uk/ ~dxp/prism,2007

Automated analysis and validation for survivability of distributed mission-critical systems

Wang Jian*, Wang Huiqiang*, Zhao Guosheng* **

(* College of Computer Science and Technology, Harbin Engineer University, Harbin 150001)

(** Center of Computer Network and Information, Harbin Normal University, Harbin 150080)

Abstract

The paper proposes a method for guantitative analysis of the survivability of distributed mission-critical systems based on the probabilistic model checking technology. The method abstracts the interactive behaviors between intruders and the system, and constructs the continuous-time Markov chains probabilistic model of the system using the PRISM language, then identifies the formal specification for the system survivability by continuous stochastic logic aiming at different disaster degrees and service levels, and ultimately, analyzes the model statistically and validates the model with the probabilistic model checking tool PRISM, and graphically demonstrates the automated analysis results of the system's survivability. The results of the theoretical analysis and the experiment show the proposed method's rationality and effectiveness. These conclusions can help to direct the design and implementation of survivable systems.

Key words: survivability, probabilistic model checking, formal specification, mission-critical systems, quantitative analysis