基于 UML 模型的统计测试方法研究*)

朱虹虹　熊光泽　雷　航　桑　楠
(电子科技大学计算机科学与工程学院　成都 610054)

摘　要　为增强统计测试的可行性、可测试性，降低建立使用模型的难度，通过研究作为工业标准的统一建模语言 UML，提出了一种基于 UML 模型的统计测试方法。论文首先建立了基于 UML 模型的统计测试过程，然后结合扩展的 UML 模型和使用模型的形式化描述，给出了从 UML 模型导出使用模型的一种形式化算法，并以工控机器人软件为例说明了应用该方法的完整过程。

关键词　统计测试，使用模型，UML 模型，马尔可夫链，有限状态机

Research on UML-Based Statistical Testing Method

ZHU Hong-Hong　XIONG Guang-Ze　LEI Hang　SANG Nan
(Computer Science and Engineering School, University of Electronic Science and Technology, Chengdu 610054)

Abstract　In order to enhance the feasibility, testability of statistical testing and degrade the difficulty of constructing the usage model, after researched on the UML as the industry-standard, brought forward a UML-based statistical testing method. In this paper, the UML-based statistical testing procedure was firstly proposed, then presented a formalized algorithm of deriving usage model from UML model, and took the control software of the industry robot as an example. Finally, the conclusion as well as the further study direction were discussed.

Keywords　Statistical testing, Usage model, UML model, Markov chain, Finite state machine

1　引言

Harlan D. Mills(1987)和 James A. Whittaker(1992)提出的统计测试(statistical testing)是一种根据软件规范建立使用模型以用于软件可靠性测试的方法。它利用使用模型产生基于状态转移概率的测试用例，可有效地进行软件的可靠性评估和预测。统计测试被称为最成功的基于模型的软件测试，是净室(CleanRoom)软件工程的核心方法，在微软、Raytheon、美国联邦航空署(FAA)等都得到了成功的应用[1]。然而由于缺乏成熟的模型建立方法，使用模型通常由程序员依赖经验直接从软件规范绘制，这限制了统计测试技术的使用和发展。

近年来，在 OMG 组织和各大 IT 厂商的推动下，作为工业标准的统一建模语言 UML (Unified Modeling Language)得到了极大发展。基于 UML 的设计和开发过程受到广泛的关注，UML 已经成为面向对象建模的事实标准。UML 定义了丰富的图元，可以从不同视角和层次描述系统的静态结构和动态特性，方便对软件的使用特性的描述和进行系统建模。如果能从 UML 模型中导出使用模型，将极大地简化统计测试的测试过程，提高测试效率。本文探讨并给出了一种基于 UML 模型的统计测试方法，该方法可从 UML 模型直接导出使用模型，降低了使用模型建立的复杂度，简化和加快了统计测试的进程。

2　相关工作

国内外在基于 UML 的统计测试领域的研究处于起步阶段。主要的研究有：L. Briand 研究了 use case 之间的顺序关系，提出一种从用案和顺序图中导出系统的测试需求的方法[2]。Matthias 研究了利用用案和精化(refine)后的状态图产生系统级测试用例的方法[3]。Philippe 提出统计功能性测试(statistical functional testing)方法，以转移覆盖作为测试准则，从 UML 状态图产生测试用例，重点研究了输入、输出变量的自动产生[4]。然而因为 UML 的灵活性，在模型的主体中不可避免地存在个性化的差异，因而阻碍了测试过程的自动化[5]。上述的研究主要集中在理论方面的探讨，并没有给出从 UML 模型导出使用模型的形式化方法。文[6]提出了一种利用 UML 顺序图形式化生成使用模型的算法，然而在 UML 模型中，顺序图主要用于对类或对象的交互序列建模，其描述能力有限，应用领域并没有状态图广泛。本文从 UML 的用案和状态图入手，研究从 UML 模型导出使用模型的形式化算法，导出的使用模型可直接用于软件的统计测试。

本文第 3 节建立了基于 UML 模型的统计测试过程，第 4 节定义了扩展 UML 模型和使用模型的形式化描述方法，第 5 节给出了从 UML 模型导出使用模型的形式化算法，第 6 节研究了基于使用模型的统计测试的用例生成和执行，最后给出结论和下一步的研究方向。

3　基于 UML 模型的统计测试过程

(1)使用 UML 语言对真实世界建模

本文使用 UML 模型中的用案(use case)和状态图来对真实世界建模。

用案(use case)描述一组序列，每一个序列表示系统外部

*)基金项目：国家"十五"预研项目(No. 41315040106)。朱虹虹　硕士，主要研究领域为软件测试技术，软件可靠性工程；熊光泽　教授，博士生导师，主要研究领域为实时计算机系统及应用、软件可靠性测评；雷　航　副教授，主要研究领域为实时软件设计与可靠性评测；桑　楠　副教授，主要研究领域为实时软件工程。

的事物(系统的参与者)与系统本身(和它的关键抽象)的交互。

状态图(State Charts)用于对系统的动态方面建模。UML状态图描述了一个对象在它的生命周期中响应事件并对时间作出反应所经历的状态序列。

本文将状态图附加到一个用案,以可视化、详述、构造和文档化一个 use case 的动态特性。

(2)从 UML 模型导出使用模型

用 UML 模型描述的系统通常包含一系列的用况,其中一个或几个用况组合起来为客户提供服务,用案之间不仅有延伸《extend》和包含《include》等关系,而且还存在反映系统的事务处理过程的执行顺序关系。统计使用测试必须考虑执行顺序关系,因为不同的用案执行顺序可能导致不同的软件失效。本方法首先导出每个 use case 对应的使用模型,然后根据用案之间的顺序执行关系,将导出的多个使用模型合成整个系统的使用模型。

(3)统计测试用例的生成和执行

根据统计理论,利用合成后的系统使用模型,可随机产生测试用例,对系统进行测试。

图 1 总结了本文提出的基于 UML 模型的统计测试的过程。

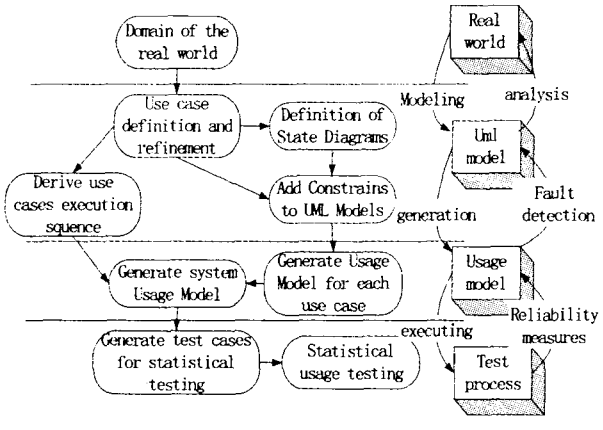

图 1 基于 UML 模型的统计测试过程

4 UML 模型和使用模型的形式定义

4.1 增加限制的 use case

如前所述,统计测试必须考虑用案间的执行顺序关系。本文通过在 UML 模型中引入前置条件(pre-condition)和后置条件(post-condition)来描述这种关系。前置条件表明了用案执行的先决条件和相应的初始化操作,后置条件表明了用案成功执行完后进入的状态。定义增加限制条件的用案 U 为:

$$U = \langle PreCond, SC, PostCond \rangle,$$

其中:(1) $PreCond$ 是 *Use case* U 在执行前必须满足的条件集合。(2) SC 是与用案 U 相对应的状态图,每个用案或案况可用一个状态图来描述,其定义见 4.2 节。(2) $PostCond$ 是用案 U 在成功执行后系统进入的状态。

如果用案的前置条件和后置条件为空,则表示用案能够并发执行。

图 2 是一个实时工控机器人控制系统在添加限制条件后的用案图。

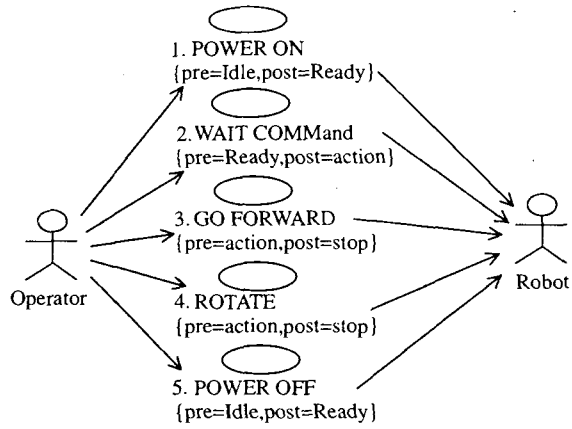

图 2 某实时工控机器人用案图

4.2 扩展的 UML 状态图

使用模型中,状态迁移具有概率特征。统计测试(可靠性测试)根据迁移的概率特征,将优先并重点测试软件中使用频率较高,对软件可靠性影响较大的部分,因而能有效地发现对软件可靠性影响较大的缺陷。

传统的 UML 模型不具备这一概率特征。为了支持基于 UML 模型的统计测试方法,本文在 UML 状态图的迁移中扩展了两个概率属性。1)事件 EV 到达的概率 $Prob\,[EV]$;2)布尔条件 $Cond$ 为真的概率 $Prob\,[Cond]$。定义扩展后的 UML 状态图 SC 为:

$$SC=\langle States, Trans, p, Labs, S_0, \mu, EventS, CondS, ActS\rangle$$

其中:(1) $States$ 是非空有穷状态集。状态 $s \in States$ 包括名字区域、进入和退出动作、内部转换(internaltransition)区域。

(2) $\rho: States \to 2^{States}$ 是状态精化函数,刻画状态间的层次(父/子)关系。

$p(s)$ 定义状态 s 的子状态集;$p^*(s)$ 代表包含自身状态的子状态集;$p^+(s)$ 代表不包含自身状态的子状态集,即 $p^+(s) = p^*(s) - \{s\}$;如果,状态 $root \in S$,对 $\forall s \in S$, $root \notin p^+(s)$,则称该状态为根状态。

(3) $Trans \subseteq 2^{States} \times Labs \times 2^{States}$ 是迁移关系,对 $\forall t=(X, L, Y) \in Trans$,存在 $X, Y \subseteq States$, $X \neq \phi$, $Y \neq \phi$, $L \in Labs$。其中 X 表示迁移 t 的源状态集,Y 表示迁移 t 的目标状态集,L 表示迁移 t 的标记。

(4)迁移标记 $Labs$ 是迁移 $Trans$ 到字符串的映射,是迁移 $Trans$ 的标签。规定迁移标记 L 具有结构 $Ev(Prob\,[EV])\,[Cond(Prob\,[Cond])]/Act$,所有迁移标记集合记为 $Labs$。

其中 Ev 是激发事件,$Cond$ 是布尔条件,Act 是动作。$Prob\,[EV]$ 表示事件 $Event$ 到来的概率,$Prob\,[Cond]$ 表示在此迁移上布尔条件 $Cond$ 为真的概率。

(5) S_0 是初始状态,μ 是终止状态的集合。

(6)UML 状态图中激发事件、布尔条件和动作的相应集合分别为 $EventS$, $CondS$ 和 $ActS$。

以上 SC 中的大部分属性可从 UML 状态图中直接获取。而对于事件概率 $Prob\,[EV]$ 和布尔条件概率 $Prob\,[Cond]$ 的确定,本文沿袭统计测试中的三种添加转移概率分布的方法[8],并稍做扩展得:(1)无信息方法,设定在一个状态下所有事件的到来均具有相同的概率;且迁移上布尔条件为真或假的概率均为 1/2。(2)有信息方法,通过类似软件或老版本软件的历史使用记录获得。(3)扩展方法,与方法(2)类似,但是它的历史使用记录是通过经验丰富的专家或用户、推断系统

运行时的情况得到的。

图 3 是图 1 所述实时工控机器人控制系统的用案 3(GoForward)的扩展 UML 状态图。

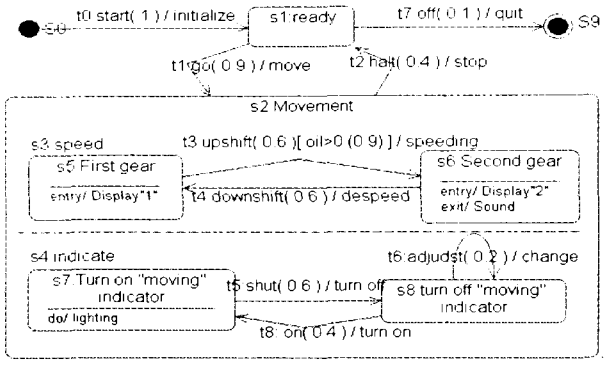

图 3　用案 GoForward 的扩展 UML 状态图

另外为了方便描述模型信息，令函数 *name*(*s*)、*entry*(*s*)、*do*(*s*) 、和 *exit*(*s*) 分别表示状态 *s* 的名字、进入动作、内部转换(或活动)以及退出动作；函数：*source*(*t*)，*target*(*t*)，*label*(*t*)，表示迁移 *t*(*t*∈ *Trans*)的源状态集、目标状态集和标记；函数 *event*(*t*)，*guard*(*t*)，*action*(*t*)，*ProbEv*(*t*)，*ProbCond*(*t*) 分别取得迁移 *t*(*t*∈ *Trans*)的标签 *label*(*t*)上的属性：激发事件、布尔条件、动作、事件发生的概率，以及布尔条件为真的概率。

4.3　使用模型(马尔可夫链)

使用模型(Usage Model)是统计测试产生测试用例的基础。使用模型有多种表示形式，其中马尔可夫链(Markov Chain)应用最广。马尔可夫链实质上是一种迁移具有概率特征的有限状态机。在马尔可夫链中，下一个状态由当前的状态和当前状态的所有离开转移的概率来决定，而不依赖于之前的状态。定义基于马尔可夫链的使用模型 *MC* 为：

$MC=\langle C,\delta,\Sigma,C_0,F\rangle$

其中，*C* 是用案 *U* 中可执行的状态的集合。δ 是转移关系：$C\times\Sigma\times[0,1]\rightarrow C$。$\Sigma$ 是一组转移的标签，对于转移 $mt\in\delta$，*mt* 的标签 $\Sigma(mt)$ 表示该转移应完成的操作集合。C_0 是系统在进入用案前必须满足的初始状态。*F* 是用案 *U* 的终止状态。

5　从 UML 模型导出使用模型

UML 状态图在传统有限状态机上增加层次、并发和广播通信机制，是一种有力的、灵活的状态迁移图。将 UML 状态图转换为使用模型的关键在于消除 UML 状态图的层次、并发和广播消息，以及给状态迁移添加概率特征。

在给出从 UML 模型导出使用模型的算法之前，先作如下定义：

定义 1　对于一个或状态 *s*，进入该状态后将进入的第一个子状态 *s'* 称为 *s* 的缺省子状态。如果 $s_1\in p^*(s)$，则称 s_1 是 *s* 的子孙，*s* 是 s_1 的祖先。如果 $s_1\in p^*(s)$ 并且 $s_1\neq s$，则称 s_1 是 *s* 的严格子孙，*s* 是 s_1 的严格祖先。

定义 2　格局(Configuration)是指 UML 状态图中能同时执行的最大状态集。任何一个格局 *C* 都应满足以下 3 个条件：

1)*C* 包含根状态；

2)对于每一个与状态 $s\in C$，或者 *s* 和 *s* 的所有子状态都包含在 *C* 中，或者都不包含在 *C*；

3)对于每一个或状态 *s*，或者 *s* 和 *s* 的某一子状态包含在 *C* 中，或者都不包含在 *C* 中。

定义 3　状态集合 $S\subseteq Sstates$，如果格局 *C* 包含 *S*，*C* 中的任一或状态 *s* 都不是 *S* 的严格祖先，并且 *s* 的缺省子状态都在 *C* 中，那么称 *C* 为 *S* 的完全构造，记为 *complete*(*S*)。

定义 4　对于迁移 $t\in Trans$，如状态 *s* 满足 $source(t)\subseteq p^*(s)$ 和 $target(t)\not\subset p^*(s)$，并且所有满足该条件的状态都是 *s* 的子孙，则称 *s* 是 *t* 的最大离开状态，记为 *gds*(*t*)，*DS*(*t*)表示 *t* 的离开状态集合，定义为 $p^*(gds(t))$。

定义 5　对于迁移 $t\in Trans$，如状态 *s* 满足 $source(t)\not\subset p^*(s)$ 和 $target(t)\subseteq p^*(s)$，并且所有满足该条件的状态都是 *s* 的子孙，则称 *s* 是 *t* 的最大到达状态，记为 *gas*(*t*)，*AS*(*t*)表示 *t* 的到达状态集合，定义为：$p^*(gas(t))\cap complete(target(t))$。

定义 6　满足 $source(t)\subseteq C, C\subseteq States, event(t)=e$ 的迁移 *t* 的集合 *T*，并且 *T* 中的任意两个变迁不相互冲突。则称 *T* 对 *C* 关于 *e* 有效。如果对 *C* 关于 *e* 有效而不包含在 *T* 中的变迁都与 *T* 中的某一变迁相互冲突，则称 *T* 是对 *C* 关于 *e* 有效的最大变迁集。

定义 7　$prob(\Sigma(mt))$ 表示使用模型中转移 *mt* 的操作 $\Sigma(mt)$ 在整个系统中的执行概率。

有了上述定义，下面先给出导出单个用案使用模型的算法；然后给出多个使用模型集成的算法。

5.1　导出单个用案的使用模型的算法

输入　用案 *U*，和与之对应的 UML 状态图 *SC*

1 创建马尔可夫链 *mc*，其初始状态 $C_0\rightarrow S_0$

2 求出 *SC* 中的所有格局的集合 $Conf^{All}$

1)令 $Conf^{All}:=\{\{S_0\}\}$；

2)取 $Conf^{All}$ 中的下一个格局 $Conf\in Conf^{All}$

3)取 *Conf* 中的下一个状态 $S\in Conf$

4)搜索 *SC* 中所有满足 $source(t)=S$ 迁移的迁移集合 $T\subseteq Trans$，

5)取 $t\in T$，到新格局 $Conf':=Conf+Tanget(t)-Source(t)$；将 $Conf'$ 放入集合 $Conf^{All}$

6)返回步骤 5)，直到遍历 *T* 中所有的迁移。

7)返回步骤 3)，直到遍历 *Conf* 中所有的状态。

8)返回步骤 2)，直到 $Conf^{All}$ 中不再增加新的格局。

3 状态的转换

1)取 *SC* 中的下一个根状态 $root\in States$

2)如果 $p+(root)=\phi$，在马尔可夫链 *MC* 中插入一个新的状态 *C*；转到步骤 4)

3)取得 *root* 对应的格局集合 *Confroot*，对于 *Confroot* 中的每一个 *conf*，在马尔可夫链中插入一个新的状态 *C*，*C* 对应 *root* 的子集 $p^*(root)$ 中的一组状态。

4)反回步骤 1)，直到所有 *root* 状态遍历完毕。

4 迁移的转换

1)取马尔可夫链 *mc* 中的下一个状态 *C*

2)取在事件集 *EventS* 中的下一个事件 *e*

3)遍历 UML 状态图，求出所有从 *C* 出发，并且激励事件为 *e* 的迁移集 $T=\{t\mid sourcet(t)\subseteq C, event(t)=e\}$

4)如果 *T* 是对 *C* 关于 *e* 有效的最大变迁集时，事件 *e* 的触发必然导致 *C* 状态的改变，将使 *C* 到达一个确定的状态 *C'*，求出 $C'=(C-Y_{t\in T}DS(t))\bigcup Y_{t\in T}AS(t)$。否则，转到 2。

5)在马尔可夫链中增加一个转移 $mt\in\delta$，$mt=(C,\Sigma$

$(mt), prob'(\Sigma(mt)))\rightarrow C'$，其中：$\Sigma(mt)=Y_{t\in T}A_1(t)\ Y_{t\in T}A_2(t)\ Y_{t\in T}A_3(t)$，

$A_1(t)=Y_{(s\in LS(t))\cap(s\in C)}exit(s)$，

$A_2(t)=action(t)$，

$A_3(t)=Y_{(S\in AS(t))\cap(s\in C')}entry(s)$，

$prob'(\Sigma(mt)=ProbEv(t)\times\sum_{t\in T}(ProbCond(t))$

6)转到2)，直到所有的事件 $e\in EventS$ 都遍历完毕，得到 C 对应的迁移集合 MT^C

7)添加转移 $mt=(C1,\Sigma(mt),proc'(\Sigma(mt)))\rightarrow C2$，其中：

$C1=C2=C;\Sigma(mt)=\bigcup s\in Cdo(S)$，$\Sigma(mt)$代表在 C 中的 do 动作集合，$prob'(\Sigma(mt))=1$。将迁移 mt 加入迁移集合 MT^C。

8)将迁移集合 MT^C 中的所有迁移 mt 上的概率 $prob'(\Sigma(mt))$进行归一化得到 $prob(\Sigma(mt))$

9)转到1)，直到所有的状态 C 都遍历完毕。

图4是由扩展的UML状态图3转换而来的使用模型。

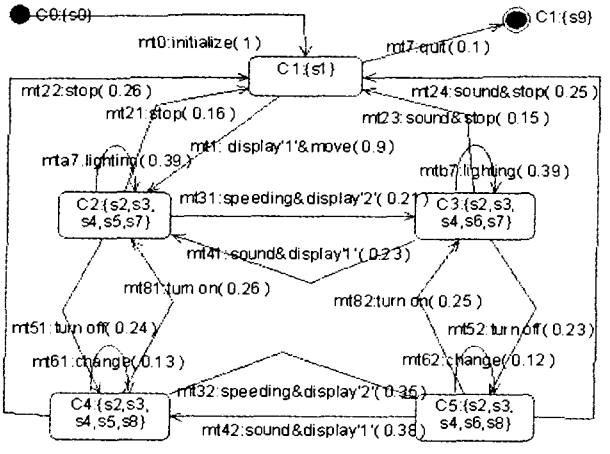

图4 GoForward用案导出的使用模型

5.2 集成系统使用模型的算法

输入：系统的用案集合，对应的使用模型集合

1)取下一个用案 U，取得用案 U 在成功执行后系统进入的状态 $U.PostCond$

2)任取一个用案 U'，$U'\neq U$，取得用案 U' 前置条件集合 $U'.PreCond$

3)如果 $U.PostCond=U'.PreCond$，则将 U 导出的使用模型 $UsageModel^U$ 的结束状态 F 和 U' 导出的使用模型 $UsageModel^{U'}$ 的初始状态 S_0 合并。

4)转到步骤1)，直到所有用案遍历完毕。

6 测试用例的生成与执行

测试用例的生成依赖于测试所使用的模型。利用使用模型，最小覆盖和随机测试用例可由CASE工具[9]自动生成。模型覆盖测试确保了在随机测试开始之前模型的最低功能，而随机测试为投入运行时的可靠性评估提供了依据。

测试用例的执行，应首先构建系统的仿真激励环境，每个仿真脚本(simulation script)对应与使用模型中不同的迁移；然后把测试用例集合翻译为测试脚本(test script)。测试用例的执行就是测试脚本的执行过程。

结束语 本文将UML模型和统计测试有机地结合起来，提出了一种基于UML模型的统计测试方法，通过从UML导出使用模型解决了统计测试中使用模型建立困难的问题。本文首次给出了从用案和状态图模型导出并集成使用模型的形式化算法。该方法增强了统计测试的可行性、可测试性，并降低了对软件系统，尤其是复杂系统进行统计测试的难度。本文正在进行的、以及今后的工作主要在于：

(1) 通过扩展RationalRose，将本文方法软件化，构建基于UML模型的统计测试CASE工具平台。

(2) 在UML模型中扩展时间限制，增加对实时软件测试的支持。

(3) 进一步扩展UML模型，增加对分布式软件计测试的支持。

参 考 文 献

1 颜炯，王戟，陈火旺. 基于模型的软件测试综述. 计算机科学，2004，31(2)

2 Briand L, Labiche Y. A UML-Based Approach to Sysyem Testing: [Carleton University TR SCR-01-01-Version 2]. 2002

3 Riebisch M, Philippow I, et al. UML-Based Statistical Test Case generation. LNCS 2591, Springer, 2003

4 Chevalley P, Thevenod-Fosse P. An Empirical Evaluation of Statistical Testing Designed from UML State Diagrams: the Flight Guidance System Case Study. IEEE. 1071-9458/01, 2001

5 Basanieri F , Bertolino A. A Practical Approach to UML-Based Derivation of Integration Tests. In: Proc. 4th Intl. Software Quality Week Europe, Brussels, Nov. 2000. 20～24

6 Yan Jiong, Wang Ji, Chen Huo Wang. Deriving Software Statistical Testing Model from UML Model. In: Proc. of the Third Intl. Conf. on Quality Software, QSIC03, 2003

7 Kim Y G, Hong H S, Bae D H, et al. Test cases generation from UML state diagrams. IEEE Proc-softw, Aug. 1999, 146(4)

8 Hubner M, Philippow I, et al. Statistical Usage Testing Based on UML. IEEE, 2003

9 Prowell S J. Jumbl. A Tool for Model-Based Statistical Testing. In: Proc. of the 36th Hawaii Intl. Conf. on System Sciences (HICSS03), 2003

(上接第45页)

省按RTP端口号加1发送)。但是，使用TURN方式时，内网对外网的所有数据包都必须通过TURN Server转发，这样TURN Server负担就比较重，增大了包的延迟，效率会降低，同时丢包率也升高了。

结束语 虽然NAT技术已经被广泛应用到各个领域，但在使用NAT技术时解决NAT穿越是个不可避免的问题，我们必须先要清楚知道NAT技术使用的环境、所用的NAT的类型，然后使用适合自己网络环境的NAT穿越方案。

参 考 文 献

1 Srisuresh P, et al. IP Network Address Translator(NAT) Terminology and Considerations [S]. RFC 2663, IETF issue, Aug. 1999

2 严军. NGN网络业务NAT穿透技术探讨[S]. 通信世界网，总第133期 http://www.cww.net.cn/cwwservice/Weekly/Article.asp? SpecialId=163&SpecialRowId=302&Id=8567

3 Takeda Y. Symmetric NAT Traversal using STUN. draft-takeda-symmetric-nat-traversal-00 (work in progress), June 2003

4 Rosenberg J , et al. Traversal Using Relay NAT(TURN). draft-rosenberg-midcom-turn-05 (work in progress), July 2004