网络协议一致性测试研究综述

朱雪峰[1,2] 许建军[3] 邹 彪[1] 张 哲[1] 孙 雷[1,2]
(中国石油大学(北京)计算机科学与技术系 北京 102249)[1]
(地球探测与信息技术北京市重点实验室 北京 102249)[2] (中国标准化研究院 北京 100088)[3]

摘 要 一致性测试是网络协议验证中最为基本的部分。虽然大量的研究与实践对此问题做过深入的探讨,但是到目前为止,仍然缺乏系统、有效而实用的协议一致性测试方法。从协议一致性描述方法入手,分别从一致性测试的体系结构、方法以及测试生成技术等方面对协议一致性测试技术进行了综合研究,最后对其中存在的问题给出了基本解决思路。

关键词 网络协议,一致性测试,形式化方法
中图法分类号 TP393 **文献标识码** A

Network Protocol Conformance Testing: An Overview

ZHU Xue-feng[1,2] XU Jian-jun[3] ZOU Biao[1] ZHANG Zhe[1] SUN Lei[1,2]
(Department of Computer Science and Technology, China University of Petroleum, Beijing 102249, China)[1]
(Key Laboratory of Earth Prospecting and Information Technology, Beijing 102249, China)[2]
(China National Institute of Standardization, Beijing 100088, China)[3]

Abstract Conformance testing is fundamental to the analysis of network protocol, while there are lots of work on research and practice aspect of this problem, we still lack a systematically, efficiently and usable method for conformance testing of network protocol. Begining with the formal description method for describe network protocol, we surveied major techniques on testing architecture, testing method and testing sequence generating in conformance testing. And finally, gave our method on remaining problem.

Keywords Network protocol, Conformance testing, Formal method

网络协议的一致性测试是一种功能性的黑盒测试,它根据协议的描述对协议的某个实现进行测试,以判别此实现与所对应的协议标准是否一致。一致性测试包括静态测试和动态测试两类。静态一致性测试是将协议实现者向测试方提供的"协议实现一致性声明"与协议中的静态一致性要求相比较;动态一致性测试就是运行测试集对 IUT 进行测试。

协议一致性测试包括 3 个阶段:第一阶段是测试生成,为特定协议产生独立于所有协议实现的抽象测试集;第二阶段是测试实现,把抽象测试集中的测试例转换成在实际系统上可执行的测试例;第三阶段为测试执行,在特定的 IUT 上执行测试用例,并且观察 IUT 的外部行为结果,最后对 IUT 与协议说明是否一致给出判定结果。

1 一致性测试的形式化描述技术

形式化描述是一致性测试的基础。形式化方法建立在严格的数学基础上,可以精确而完整地表达协议的功能、性能及行为等,它为协议的分析、验证、实现、测试等活动的系统化、自动化提供了良好的基础。形式化描述技术包括形式化描述方法和形式化描述语言。

1.1 形式化描述方法

目前,大量的形式化描述方法已不断提出并应用到实际的协议开发中。常用的形式化描述技术包括有限状态机、Petri 网、进程代数、时序逻辑和构造类别代数等。

有限状态机(FSM)是一种最常见的模型,它具有直观、易于实现的特点,能够很方便地与其它形式化方法进行组合。随着 FSM 的广泛应用和发展,出现了很多扩展模型,主要包括扩展有限状态机、确定有限自动机、通信层次化状态机、通信有限状态机等。有限状态机模型存在的问题是:无法描述并发行为,不利于协议验证的实现;难以描述复杂的系统,在进行大规模复杂协议的描述时,FSM 模型会面临状态爆炸的难题。

Petri 网是一种建立在并发概念基础上的、特殊的自动机模型,可以清楚地表达两个进程之间的通信,能够直观地表示非确定性,可用于描述通信系统中异步成分之间的关系。Petri 具有一套成熟的数学理论工具,在通信领域特别是通信协议验证方面得到了广泛的应用。Petri 网有多种扩展模型,包

到稿日期:2009-01-20 返修日期:2009-03-25 本文受国家自然科学基金重大项目(60496324)和中国标准化研究院中央基本科研业务费支持项目(56076S-1524)资助。

朱雪峰(1973—),男,博士,讲师,CCF 会员,主要研究方向为形式化验证,E-mail:xuefeng.zhu@cup.edu.cn;许建军(1974—),男,博士,副研究员;孙 雷(1970—),女,副教授。

括数字 Petri 网和时间 Petri 网等。Petri 网以其清晰直观的图形表示、坚实的数学基础和分析技术得到了广泛的应用，但是在描述大型复杂协议时，同样存在状态爆炸的问题。

进程代数将协议描述成进程的集合，通过进程事件的集合和进程的迹来描述进程的行为，通过并发、选择、递归等来描述进程之间的关系。进程代数能够严密地表述协议的逻辑结构以及协议的时序性，而且有助于协议验证。R. Milner 提出的通讯进程演算(CCS)和 C. A. Hoare 提出的通信顺序进程(CSP)均基于进程代数理论。

时序逻辑是模态逻辑的扩展，以状态为可能世界，以状态的演变次序关系为可能世界间的可达性关系。在时序逻辑描述中，使用辅助的时序算子定义时序逻辑公式，每个公式都是关于状态序列的一个断言；一个时序逻辑描述由一组时序公理组成，这些公理描述了从系统执行开始产生的所有状态序列都为真的性质；时序逻辑通过状态关联进行推理，它附加有与时间相关的操作属性。时序逻辑应用较为成熟，并且数学抽象能力很强，它侧重于通过定义系统外部可见的行为事件来描述系统，即直接描述系统的输入/输出行为，不关心协议实体的内部变化，比 FSM，Petri 网更易于刻画协议的活动性，因而有利于对协议的各种性质进行分析验证；其缺点是协议描述的可读性差，协议描述复杂。

构造类别代数是一种基于代数规范的形式化描述技术，通过定义构造函数或延拓函数来定义其可观察和可控制的行为，并通过定义公理集合来限制构造函数和延拓函数，即公理集合构成了协议行为的规范。构造类别函数的这种特性使得它非常适合于描述协议数据部分及其处理过程。这种类别代数的方法与基于有限状态机的方法类似，但是当状态机状态较多时，它比基于有限状态机的方法更容易描述协议并生成测试用例。

1.2 形式化描述语言

国际标准化组织根据形式化描述方法提出了 3 种通信协议的形式化描述语言，分别是 ISO 的 ESTELLE，LOTOS 和 CCITT 的 SDL。

ESTELLE 是由 ISO 组织专门为协议描述而设计开发的一种基于扩展有限状态机的形式化描述语言。它是 PASCAL 语言的扩充，其描述的协议很容易转换成 PASCAL 或 C 代码，是一种面向协议实现的 FDL：模块实例可通过初始化语句动态产生，如果协议实现后模块实例对应于一个进程或任务，那么网络进程或任务也是动态产生的；模块之间的通信为异步通信。ESTELLE 对并发、不确定性、超时、异步通信状态转移具有较强的表达能力，但是对于递规、共享通道、同步通信、协议性质的表示缺乏有力的手段。用 ESTELLE 描述的协议易于提取 FSM 模型和 Petri 网模型，但不容易转变成时态逻辑模型和 CCS 模型。

LOTOS 是由 ISO 组织开发的一种形式化描述语言。在基于 LOTOS 的描述中，一个通信系统可以描述成一系列有时间顺序的、可由外部观察的事件。LOTOS 使用 CCS 来描述进程的行为和交互，并且使用 ACT ONE 语言来描述数据结构和表达式。ACT ONE 是一个抽象数据类型语言，它的数学模型是代数规范。LOTOS 语言存在一些扩展，如 D-LOTOS，G-LOTOS 等。

SDL 是由 CCITT 组织开发的基于 EFSM 的电信领域的国际标准。SDL 存在两种表示方法：图形表示和语句表示。SDL 语言主要由以下几个部分组成：结构、行为和通信。SDL 是基于扩展有限状态机的形式化描述语言，可用于从需求分析到具体实现的整个开发过程，适用于具有实时反应的系统；用图形的形式表示，可视性好；具有面向对象的特征。基于 SDL 的上述优点，SDL 主要用于实时交互分布式系统的形式化描述。

2 一致性测试体系结构

抽象测试方法描述由下测试器、上测试器和测试协调过程组成的抽象测试结构以及它们与测试系统和 SUT 的关系组成。一致性测试的抽象测试方法分为两大类：端系统的抽象测试方法和中继系统的抽象测试方法。

2.1 端系统 IUT 的抽象测试方法

根据不同的控制观察点，现有的端系统抽象测试方法可以分为 4 类，即本地测试法、协调测试法、分布测试法和远程测试法，如图 1 所示。

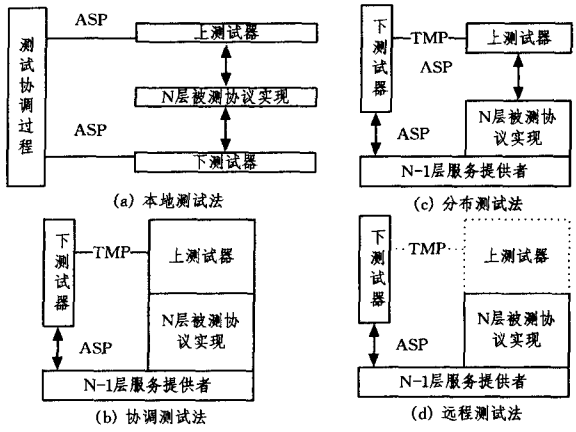

图 1 端系统抽象测试方法

本地测试方法是下测试器和上测试器都在测试系统中且在 IUT 的上层服务边界上有一个 PCO 的抽象测试方法。协调测试方法是上测试器在 SUT 中且为测试协调过程定义了标准化的 TMP，使得控制和观察(包括测试管理 PDU 的控制和观察)仅用下测试器活动的术语说明的抽象测试方法。分布测试方法是上测试器在 SUT 中且在 IUT 的上层服务边界上有一个 PCO 的抽象测试方法。远程测试方法是测试事件的控制和观察仅用下测试器活动的术语说明，且对测试协调过程的一些要求可能在 ATS 中暗示或非正式地表达，但没有做关于这些要求的可行性或实现的假定的抽象测试方法。

2.2 中继系统 IUT 的抽象测试方法

开放中继系统的抽象测试方法包括两类：回环测试方法和穿越测试方法，如图 2 所示。

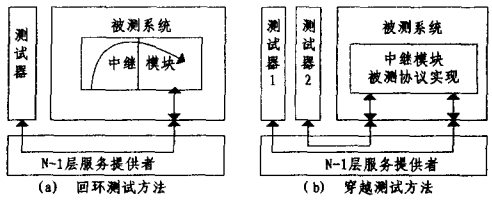

图 2 中继系统抽象测试方法

回环测试方法只需要一个测试器，但要求在被测实现或系统内部或外部链路上实现回环，而且其测试能力过于简单，

因而不够实用。另外，被测中继系统只有一端的行为被直接观察到，而另一端的行为不能被正确地评价。穿越测试方法能够测试中继系统的全部中继功能，但需要至少两个测试系统。各测试系统之间的协调是实现穿越测试方法的困难所在，穿越测试方法则使被测中继系统在平常的操作模式下得到测试，在两端的行为都能够观察到。

3 测试生成方法

协议测试方法中最主要的工作就是生成测试序列。目前，绝大多数一致性测试序列的生成算法都是基于 FSM 模型的。基于有限状态机的测试方法基本过程如下：测试系统向 IUT 施加输入事件序列，接收校验输出事件序列，检查状态转移；根据输出事件和状态的转移，判定 IUT 的行为是否符合协议规范的描述。测试生成方法包括可达性分析与测试序列两种。

3.1 可达性分析

可达性分析从一个初始状态出发，生成并检查系统能够到达的所有状态，以实现协议验证的自动化。可达性分析包括 3 种：全局搜索、控制下的部分搜索、随机仿真。

3.1.1 全局搜索

全局搜索算法的主要思想是：首先从初始状态集 S_0 中取出一个状态 s，从初始状态集 S_0 中删去 s 并将 s 加入到已分析状态集 S' 中；如果 s 是一个错误状态，则报告出错；否则对于 s 的每一个后继状态 s'，如果 s' 不在初始状态集 S_0 或已分析状态集 S' 中，则将 s' 不放入初始状态集 S_0 中，并递规地执行以上过程，最后从 S_0 中删除 s'。重复执行上述过程，直到初始状态集为空。

算法中工作集 S_0 控制系统状态空间树的搜索顺序。如果工作集 S_0 中的状态以先进先出的顺序取出，那么算法执行的是状态空间树的宽度优先搜索；如果是以后进先出的顺序取出，则是深度优先搜索。

全局搜索算法可以证明协议中没有错误，但是其应用范围有限，能够分析的最大状态数目依赖于协议、描述方法和可用的计算资源。当系统状态的数目非常大时，会发生状态空间爆炸。

3.1.2 局部搜索

局部搜索与全局搜索基本类似，只是在后继状态的处理上选择的是某些后继状态，而不是像全局搜索那样选择每一个后继状态，这就是部分搜索最主要的特征。

局部搜索只能用来证明错误的存在，无法证明不存在错误。与全局搜索算法相比，可以有效地解决状态爆炸的问题，这样就能利用有限的资源来验证协议中最重要的部分，从而最大限度地发现错误。缺点是必须能够预先判断出协议中错误的位置，然而这是无法预先做到的；另外，虽然这些方法能够减少状态空间的大小，但是它们都没有提供任何工具将状态空间的大小与可用内存相匹配。

3.1.3 分支搜索

分支搜索的基本思想是：从一个初始状态出发进行判定；如果此状态为一错误状态，则报告出错，再从初始状态集中选择一个初始状态；否则选取此状态的一个后继状态，重复执行以上过程。分支搜索中，只选取当前状态下状态转换中的一条分支，因而算法相当简单。

分支搜索算法与协议系统的大小和复杂性无关。对于复杂的验证问题，分支搜索是实际中唯一可用的方法。不足之处在于：首先，分支搜索没有明确的终止，无法判断是否已经访问过系统的所有可达状态；其次，由于没有算法的终止，也就无法判断是否已经发现了系统的所有错误。这种算法也只能发现错误，不能证明协议中没有错误。

3.2 测试序列方法

目前协议测试方法都是在转移级别来做的，即针对 FSM 中的单个转移生成相应的测试子序列，再将这些测试子序列连接起来作为完整的测试序列。

设 M 是有限状态机，$M=\langle I,O,S,S_0,T\rangle$。$M'$ 是 M 的一个实现，则针对转移 $t=\langle s_i,s_j,i/o\rangle$ 的测试通常包括以下 3 步：(1)将 IUT 从初始状态置成状态 s_i；(2)向 IUT 输入激励 i，接收并核对 IUT 的输出事件 o；(3)验证 IUT 所到达的新状态是否为 s_j。

通过以上 3 步得到转移 t 的测试子序列，它由 3 部分组成。第一部分将 IUT 从初始状态置成状态 s_i，称为路径序列 $PS(s_i)$，该序列可以通过宽度优先搜索得到；第二部分就是待测转移 t 的输入输出 i/o；第三部分验证 IUT 所到达的新状态是否为 s_j 所采用的特征序列 $CS(s_j)$，该特征序列可以唯一地确定有限状态机的当前状态，后两部分合起来称为测试段。针对如何生成测试序列，主要的方法有以下几种。

3.2.1 T 方法

在一个有限状态机中，最直观的一种测试方法就是从初始状态出发，将其中所有的转移至少遍历一次，最终回到初始状态。转移回路方法相对于其它基于有限状态机的测试序列生成方法，具有简单、生成序列长度短的优点，并且这种方法仅要求有限状态机是强连通的，不必完全定义；但是这种方法在测试过程中只检查了转移的输出情况，没有对转移到达的状态进行检查，因而错误检测能力较低。

3.2.2 D 方法

对于一个输入序列，如果它分别作用于 FSM 的每一个状态，从这些状态分别开始的输出均不相同，则称此输入序列为该 FSM 的区分序列。

D 方法的基本思想就是利用区分序列来生成测试子序列的第三步。区分序列在状态确认上是一种非常有效的方法，但是这种方法不具有普适性：一方面，大多数有限状态机中不存在 DS 序列；另一方面，即使存在 DS 序列，其长度也可能太大而无法使用。

3.2.3 W 方法

针对 D 方法的缺陷，W 方法采用多个输入事件来确定状态。W 方法基于 W 集和 P 集来构造测试序列，其中 W 集是 FSM 的输入序列的集合，其输出能够唯一标识状态；P 集是使 FSM 从初始状态到任意状态的输入序列的集合。

IUT 的 W 集合是一个包含 k 个输入事件序列的集合。对于 IUT 的各个状态来说，W 集合是相同的，但是各个输入事件序列所产生的最后输出事件所组成的输出模式不同。

如果 W 集合包含 k 个输入事件序列，那么需要对 IUT 施加 k 次测试，才能判定 IUT 是否处于状态 s_j，这样就大大加长了测试序列。但是相对于区分序列而言，W 集合一般是存在的。

(下转第 36 页)

[7] Frauenthal J C. Mathematical Modeling in Epidemiology [M]. New York: Springer-Verlag, 1980

[8] Zou C C, Gong W B, Towsley D. Code Red worm propagation modeling and analysis[C]// Proceedings of the 9th ACM Symposium on Computer and Communication Security. Washington, 2002: 138-147

[9] Zou C C, Towsley D, Gong W. On the performance of Internet worm scanning strategies[J]. Performance Evaluation, 2006, 63: 700-723

[10] Zou C C, et al. The monitoring and early detection of Internet worms[J]. IEEE/ACM Transactions on Networking, 2005, 13 (5): 961-974

(上接第 7 页)

3.2.4 U 方法

区分序列和 W 集合都是在有限状态机所处的状态完全未知的情况下对状态进行确定的一种方法，但是实际上，在测试子序列生成过程的第三步，对 IUT 所处的状态有一个期望值 s_j，在测试中只需要确定 IUT 的实际状态是否为 s_j。由此来看，区分序列和 W 集合的状态确认能力相对于测试要求来说太强了，这就是 U 方法的基本出发点。

IUT 状态 s_j 的 UIO 序列是 IUT 所有其它状态不能表现的 I/O 行为，它唯一地标识状态 s_j。为了找出各个状态的 UIO 序列，必须罗列出 IUT 各个状态的 I/O 序列(一棵 I/O 序列树)，从树的根部开始比较各个状态的 I/O 序列，直至为每个状态找到唯一的 I/O 序列为止。

相对于 D 方法和 W 方法，U 方法能够生成更短的测试序列，并且 UIO 在大部分的有限状态机中是存在的，因而其适用范围更广。

3.3 测试序列到抽象测试集的转换

测试集可分为通用测试集 GTS、抽象测试集 ATS 和可执行测试集 ETS。测试生成阶段得到的测试序列属于通用测试集的初级阶段，需要经过规范化转换到通用测试集。采用适当的测试描述法描述通用测试集就能够得到抽象测试集；输入到特定的测试系统并结合被测协议实现信息就能够得到针对该实现的可执行测试集。

实际测试时，把自动生成的测试序列按照相当确定的测试目标进行分解，得到针对每一个测试目标的测试子序列，在测试子序列的基础上构造每一个测试例。分析每一个测试子序列，找到其明确的测试点(比如测试一个状态、一个变迁等)，对子测试序列进行分割，把从子序列起始位置到测试点的初始位置作为前测试步序列，完成测试驱动和状态验证的剩下部分作为测试体序列。根据子序列执行后的最终状态，添加能够使被测协议转换到初始状态的后测试步序列。三部分序列组合起来，得到针对特定测试点的完整的测试序列。采用适当的测试描述法对上述得到的测试序列进行描述，就得到一个完整的测试例。如此构造出针对每一个特定测试点的测试例，就能够组合成抽象测试集。

4 测试实现和测试执行

4.1 测试实现

在测试实现阶段，根据协议实现的 PICS 和 PIXIT 从一致性测试集中选取适当的测试例，去除没有意义的测试，并使用 PIXIT 提供的信息来量化这些测试例。从抽象测试集生成可在一实际的测试系统上执行的参数化的可执行测试集，即可在特定测试设备上对某个 IUT 进行测试运行的测试集。

4.2 测试执行

在测试执行阶段，一个特定的 IUT 被实际测试，并得出 IUT 一致性判定结果。测试执行过程分为两步：第一步是静态一致性需求检查，根据协议标准的静态一致性需求对 IUT 的 PICS 进行检查；第二步为在测试器上执行测试例来检查 IUT 对动态一致性要求的满足程度，对每个测试例做出测试判断：通过、失败或不确定。最后，静态一致性检查的结果和所有的测试例的执行判定结果组合在一起，形成一个有关 IUT 的一致性判决。当且仅当所有的测试都未失败时，最终的判决才会是通过。

现有的测试执行方法可划分为两类：基于编译的测试执行(CTE)和基于解释的测试执行(ITE)。基于编译的测试执行，是指在测试执行之前，由抽象测试集 ATS 到可执行测试集 ETS 的转换已经由转换器或编译器完成，这一过程非常耗时，但是提高了测试执行的效率。在基于解释的测试执行中，从 ATS 到 ETS 的转换是在测试执行过程中完成的，这种方法使得用户可以对测试过程进行动态观察和控制，但测试执行的效率较低。

结束语 协议一致性测试所面临的挑战是双重的。一方面，随着协议的全方位发展，协议的功能越来越强，协议的复杂性也越来越高，使得协议的一致性测试变得越来越困难；另一方面，随着形式化验证技术的发展，对于协议一致性的验证必然会面临如何提高形式化验证的效率的问题。

目前一致性测试的研究和实践中需要解决的几个关键问题包括测试理论的形式化、高速计算机网络协议和路由协议的测试、通用测试平台的研制。所有这些都需要新的、高效可行的形式化验证技术的支持。

我们认为，面对协议一致性测试领域的困境，形式化的领域本体可能会对协议的形式化验证起到关键的作用，我们未来的工作将主要沿着这条路径展开。

参 考 文 献

[1] Zhang Y, Li Z. A New Formal Test Suite Specification Language for IPv6 Conformance Testing[C]// Proceedings of ICCT2003. 2003: 174-177

[2] Wu J, Samuel T, Gao Q. Formal Methods for Protocol Engineering and Distributed Systems[M]. Kluwer Academic Publishers, 2001

[3] Tian J, Li Z. The Next Generation Internet Protocol and its Test [C]// Proceedings of IEEE International Conference on Communication. 2001: 210-215

[4] Zhang F, Cheung T. Optimal Transfer Trees and Distinguishing Trees for Testing Observable Nondeterministic Finite State Machines[J]. IEEE Transactions on Software Engineering, 2003, 29 (1): 1-14

[5] 朱雪峰，金芝. 关于软件需求中的不一致性[J]. 软件学报，2005, 16(7): 1221-1232