中国科学 E 辑：信息科学 2008 年 第 38 卷 第 10 期：1594 ~ 1613
www.scichina.com info.scichina.com
《中国科学》杂志社
SCIENCE IN CHINA PRESS

一种基于 TTCN-3 的协议测试系统及其扩展研究

尹霞[①③*]，王之梁[②③]，景传明[①③]，施新刚[②③]

① 清华大学计算机科学与技术系，北京 100084;
② 清华大学信息网络工程研究中心，北京 100084;
③ 清华信息科学与技术国家实验室(筹)，北京 100084
* E-mail: yxia@mail.tsinghua.edu.cn

收稿日期: 2008-06-03; 接受日期: 2008-06-30
国家重点基础研究发展计划(批准号: 2003CB314801)和国家自然科学基金(批准号: 60572082)资助项目

摘要 用户要求下一代互联网是一个更大、更安全、更快、更及时、更方便、更可管理的网络. 为下一代互联网服务的协议测试技术需要适应这些测试需求. 文中重点研究了协议测试中的测试集描述法和测试实现技术. 选取了 TTCN-3 作为测试集描述法, 并针对协议鲁棒性测试需求进行了语法和语义扩展. 开发了基于 TTCN-3 的协议集成测试系统 PITSv3, 并对其进行了鲁棒性测试扩展实现. 最后通过两个实际的测试应用, 体现出 PITSv3 是一个具有通用性、标准性、可扩展性特点的分布式测试平台.

关键词
协议测试
TTCN-3
PITSv3
鲁棒性测试

互联网已成为现代社会最重要的信息基础设施, 不断涌现的应用需求、快速扩张的网络规模都推动着世界走向下一代互联网. 面对着新协议、新技术、新产品的不断涌现, 为了保障下一代互联网的可靠健壮, 协议测试是一个很重要的研究领域. 对于下一代互联网, 用户希望是一个更大、更安全、更快、更及时、更方便、更可管理的网络. 新的协议测试技术需要适应下一代互联网的需求.

协议测试的研究涉及了协议描述、测试生成、测试集描述法、测试实现等多个环节, 本文将重点研究其中的测试集描述法和测试实现技术. 测试和测试控制表示法TTCN-3[1](testing and test control notation)是目前最适于描述下一代互联网协议测试的形式化测试集描述法, 但还是有很多不能满足的测试需求, 比如协议鲁棒性测试需求. 下一代互联网要求更安全, 协议实现就需要加强对异常事件的处理能力, 鲁棒性的测试就提供了这种测试能力. 为此需要对 TTCN-3 进行必要的语法和语义扩展, 以增强其测试描述能力. 本文以面向协议鲁棒性测试的

TTCN-3扩展为例研究了TTCN-3语言的扩展方法. 在此基础上, 本文研究了测试实现技术, 设计开发了一个基于TTCN-3 的分布式协议集成测试系统平台, 同时在该系统中实现了相应的TTCN-3 扩展. 同国外的TTCN-3 商用工具比较, PITSv3 不仅做到了功能全面、界面友好, 尤其是在可扩展性、支持新协议测试需求、支持多机分布式测试方面具有优势.

全文结构如下: 第 1 节介绍相关研究工作. 第 2 节扩展了 TTCN-3 的语法和语义, 使得TTCN-3 首次能够提供鲁棒性测试能力. 第 3 节设计实现了基于 TTCN-3 的协议集成测试系统PITSv3(protocol integrated test system). PITSv3 是一个完整的 TTCN-3 测试系统, 是具有通用性、标准性、可扩展性特点的分布式测试平台. 第 4 节介绍了 PITSv3 在 BGPv4+一致性测试和 OSPFv2 鲁棒性测试方面的应用情况. 最后是全文的总结.

1 相关工作

Piatkowski[2]在 1981 年提出“协议工程学”的概念后, 协议测试是其中最活跃的研究分支. 在协议测试的发展历程中, 形式化技术和方法一直贯穿其中. 本文是对基于形式化技术的协议测试中测试集描述法和测试实现技术的研究.

1.1 小节采用集合论和标号变迁系统简要定义了协议测试, 说明了协议测试的过程; 1.2 小节介绍了测试集描述法的最新研究成果; 1.3 小节介绍了测试实现技术的最新研究成果.

1.1 协议测试概述

协议测试实际上是对协议的“完成关系(implementation relation)”的测试, 即被测试者是否实现了预定的目标. 本文将首先用集合论和标号变迁系统来介绍协议测试[3].

完成关系存在于被测试方和协议之间, 任何一次测试活动都是有目的, 就是测试需求. 用测试需求来表述协议测试更易于理解, 也更易于实现.

令 SPECs 表示某个协议的各种形式化描述的集合, 其中的元素 s 表示某种测试需求的某个形式化定义, 则 $s \in$ SPECs. 所有测试需求的形式化说明的集合标识为 REQs. 那么对于某个协议的某个具体的形式化描述 $s \in$SPECs, 它的测试需求的形式化说明的集合 $R \subseteq$REQs. $r \in R$ 表示某个单一的测试需求. 则所有协议的形式化描述的集合 SPECs 是所有测试需求的形式化说明的幂积, 即 SPECs=P(REQs).

被测试方是一些具体的物理实现, 假设都可以通过形式化模型集合 MODs 中的某个形式化对象 $m_{\text{U-T}}$ 来建模, $m_{\text{U-T}} \in$ MODs .

这样, 讨论被测试方与协议之间的关系, 就可以转化为讨论被测试方对应的形式化模型 $m_{\text{U-T}}$ 和形式化的协议说明 s 之间的关系了.

定义 1 完成关系 imp. imp $\subseteq$ MODs×REQs $\subseteq$ MODs×SPECs.

如果被测试方的形式化模型 $m_{\text{U-T}} \in$MODs 与测试需求 $R \subseteq$REQs 之间满足完成关系, 即 $m_{\text{U-T}}$ imp R, 即表示被测试者 U-T $\in$ UTs 实现了测试需求 $R \subseteq$REQs. 另, 用 M_R 表示所有满足完成关系的被测试方的形式化模型的集合, 即 $M_R =_{\text{def}} \{m \in \text{MODs} \mid m \text{ imp } R\}$; 而 $\overline{M_R} =_{\text{def}}$ MODs/M_R 表示 M_R 的差集(补集).

测试所采用的依据就是测试经验, 其形式化定义可标识为 TESTs. 测试中的实际经验被

包含在测试例 tcase 中, 若干测试例的集合构成测试集 T, 而测试集是 TESTs 中的一个元素. 换句话说 TESTs 是测试例的集合的集合, 即有 tcase$\in T\in$TESTs.

利用测试例, 对一个具体的被测试方进行测试的过程称为测试执行. 通过测试执行将得到观测数据集合 OBs 中一个观测数据. 每个观测数据, 都将通过判决函数 Verdict 得到一个判决.

定义 2 判决函数 Verdict. $\text{Verdict}_{\text{tcase}}$: OBs $\to$ {PASS, FAIL,INCONCLUSIVE}.

其中 PASS 表示测试判决为“成功”, FAIL 表示测试判决为“失败”, INCONCLUSIVE 表示测试判决为“未定判决”.

下面, 利用标号变迁系统对上述定义进行描述.

定义 3 标号变迁系统 LTS. 标号变迁系统 LTS 是一个 4 元式$\langle S, L, T, s_0\rangle$, 其中: S 是一个状态的集合; L 是一个外部可观察动作的集合; $T\subseteq S\times\{L\cup\{\tau\}\cup\{\delta\}\}\times S$ 是变迁的集合, 其中τ表示不可观察的内部动作, δ表示成功的结束. T 中的元素(s, u, s') 可以记为: $s\xrightarrow{u}s'$, 其中 s, $s'\in S$, $u\in L\cup\{\tau\}\cup\{\delta\}$; s_0 为初态.

令$\langle S, L, T, s_0\rangle$为一标号变迁系统, 运算符‘·’表示接续运算, 若$\sigma=u_1\cdot u_2\cdot\cdots\cdot u_n$是一串 L 中标号的接续, 则σ就是 L^*上的一个迹(Trace), L^*是所有迹的集合. $\Rightarrow$表示了在两个状态 s, s'之间的状态变迁.

定义 4 迹 Trace. $s\overset{\sigma}{\Rightarrow}s'$ 表示通过执行可观察动作序列迹σ, 状态 s 可以变迁到状态 s'. 则迹的形式化定义为: $\text{traces}(s)=_{\text{def}}\{\sigma\in L^*\mid\exists\, s':\ s\overset{\sigma}{\Rightarrow}s'\}$.

从状态 s 的可达状态集合的定义为: $\text{der}(s)=_{\text{def}}\{s'\mid\exists\, s':\ s\overset{\sigma}{\Rightarrow}s'\}$.

对于 $S'\subseteq S$ 而言, S'中的初始状态集合定义为:

$$\text{init}(S')=_{\text{def}}\{x\in L\mid\exists\, s'\in S':\ s\overset{\sigma}{\Rightarrow}s'\}.$$

执行了迹σ后的可达状态集合定义为: s after $\sigma=_{\text{def}}\{s'\mid s\overset{\sigma}{\Rightarrow}s'\}$.

测试需求的定义. 可以用 L 的所有标号变迁系统 REQs 来表示协议的形式化定义集合 SPECs, 即: REQs=LTSs(L).

被测试方的定义. 由于大多数系统通信时都是将输入/输出动作分开研究, 被测试方的形式化模型 MODs 将采用输入输出变迁系统 IOTS 来定义. 可观察动作集合 L 被分成输入动作集合 L_{IN}和输出动作集合 L_{OUT}, 则: MODs= IOTS(L_{IN}, L_{OUT}), 而且 IOTS(L_{IN}, L_{OUT}) $\subseteq$ LTS($L_{\text{IN}}\cup L_{\text{OUT}}$).

测试的定义. 对于 L_{IN}和 L_{OUT}的测试是一个 6 元组$\langle S, L_{\text{IN}}, L_{\text{OUT}}, T, v_t, s_0\rangle$, 其中$\langle S, (L_{\text{IN}}\cup L_{\text{OUT}}), T, s_0\rangle\in$ LTS($L_{\text{IN}}\cup L_{\text{OUT}}$)是一个在 L_{IN}和 L_{OUT}上确定的有限行为的标号变迁系统; 而 v_t: $\text{der}(s_0)\to$ {PASS, FAIL, INCONCLUSIVE} 是可达状态 t 的的判决函数. 当对被测试方进行测试的时候, 或者是从被测试方得到输出, 或者是向它输入, 或者是死锁. 即任何测试都满足: $\forall s'\in\text{der}(s_0)$: $\text{init}(s')=\{a\}\subseteq L_{\text{IN}}$ 或者 $\text{init}(s')=L_{\text{OUT}}$或者 $\text{init}(s')=\varnothing$.

完成关系的定义. 完成关系是在测试需求 REQs 和被测试方的形式化模型 MODs 之间的

一种关系, 对于 $L \subseteq (L_{IN} \cup L_{OUT})$, 完成关系标识为 imp_L.

令 $m_{U\text{-}T} \in IOTSs(L_{IN}, L_{OUT})$, $R \in LTSs(L_{IN} \cup L_{OUT})$, $L \subseteq (L_{IN} \cup L_{OUT})$, 则有: $m_{U\text{-}T}\ imp_L\ R =_{def} \forall \sigma \in L$: $out(m_{U\text{-}T}\ after\ \sigma) \subseteq out(R\ after\ \sigma)$. 其中, out 为输出函数.

如果对于每个$\sigma \in L$ 而言, 被测试方 $m_{U\text{-}T}$ 在完成协议需求 R 后可以得到正确的输出, 那么 $m_{U\text{-}T}$ 在 R 和完成关系 imp_L 下就是正确的.

在明确了协议测试的定义后, 可以得到协议测试的过程, 如图 1 所示. 从协议标准中利用测试生成技术得到抽象测试集, 抽象测试集需要采用适宜的描述法表达, 通过协议实现得到可执行测试集, 进行测试执行, 最后得到测试判决.

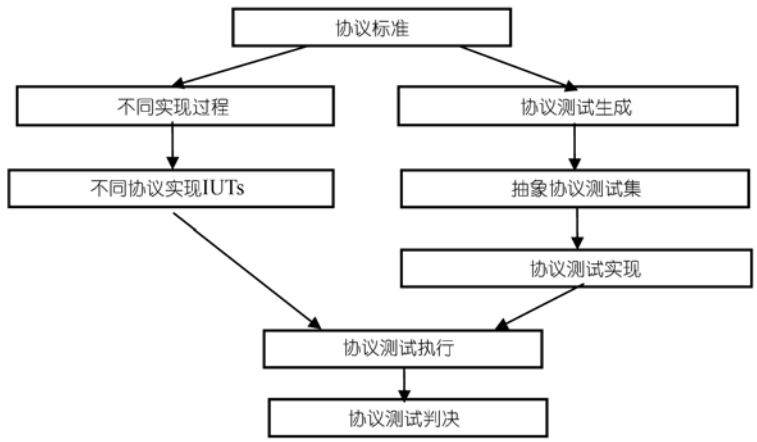

图 1 协议测试过程图

本文将采用形式化技术重点研究其中的协议测试实现技术和与其紧密相关的抽象测试集描述法.

1.2 测试集描述法研究进展

测试生成是从用自然语言描述的协议标准生成测试集的过程, 是测试实现的基础. 测试集描述法在测试方法学中起着承上启下的重要作用, 是连接测试生成和测试执行的纽带. 一个好的测试集描述法不仅有利于提高测试执行的自动化程度, 也有利于测试开发人员手工开发测试用例. 许多语言被用于测试集描述, 如形式化测试描述语言TTCN[1,4], 脚本语言tcl/tk, Perl, 编程语言C/C++/Java等. 其中, TTCN因为其突出的对测试的支持能力, 易于被测试人员理解和使用而成为协议测试中广泛使用的测试集描述法.

到目前为止, TTCN已经经历了 3 个版本. 国际标准化组织ISO在其ISO/IEC-9646[4]中定义了TTCN-1, 后来被ITU-T接收为标准X.292. TTCN-1 结合了树型和表型表示法的优点, 有静态说明部分和动态行为部分. 静态说明部分专用于协议的静态信息描述, 如协议数据格式、参数值、控制观测点、定时器、变量等的描述. 在动态行为部分, TTCN-1 定义了与消息的接收和发送相关联的操作与操作符, 用于描述测试事件及其之间的时序关系. TTCN-1 在当时被大量地应用, 出现了很多TTCN工具, 如法国VERILOG公司的Object, 瑞典Telelogic公司的ITEX等

工具软件.

由于 TTCN-1 只支持单一测试部件, 而网络协议的日益复杂使得其分布式特性越来越广泛, TTCN-2 在不改变 TTCN-1 整体结构和原则的基础上, 引入了主、从测试部件的概念, 通过增加 create 和 done 等语句, 使得 TTCN-2 可以支持分布式测试和多方测试.

但是随着网络的飞速发展, 各种新型协议和应用对测试描述能力的要求也越来越高, 原有的测试描述语言TTCN-2 逐渐显露出不足之处: 缺乏动态配置能力、没有同步通信能力、应用范围过窄等. 欧洲电信标准化协会ETSI经过深入研究, 于 1999 年至 2002 年间开发完成了TTCN的新版本TTCN-3, 其最新的版本是 2007 年发布的TTCN-3 规范的第 3 版[1]. TTCN-3 现已作为ITU-T Rec. Z.140 系列发布. 同TTCN前两个版本相比, TTCN-3 是一种更为灵活和强大的语言, 可以用于描述各种反应式系统(reactive system)的测试. TTCN-3 打破了TTCN一致性测试的局限, 能适应更广泛的测试需求, 如互操作性测试、性能测试、系统测试和集成测试 等[5].

在通讯系统测试领域学术界, TTCN-3 受到了广泛的重视, 例如, 文献 [5~7]集中在对TTCN-3 语言本身的设计研究, 文献 [8,9]集中在 TTCN-3 测试系统的实现研究等.

1.3 测试实现技术研究进展

随着 TTCN 语言的不断完善, 相应的测试系统也不断地更新. TTCN-3 作为一种最新的测试描述和实现语言, 在以德、法为代表的欧洲工业界受到了高度重视, 不断推出基于 TTCN-3 的测试实现工具. 这些商用工具大多可以集成编辑、编译和运行功能, 支持 TTCN-3 v2.0 以上版本的基本语言要素, 支持以 UML 序列图为基础的图形化测试说明.

其中较有代表性的产品有:

Telelogic公司的TAU/Tester[10]. Telelogic公司是一家一直致力于TTCN支持工具研发的公司, 拥有TTCN-1 和TTCN-2 的相关工具. TAU/Tester是一款基于TTCN-3 的软件测试工具, 主要用于软件开发期间的系统测试和集成测试.

OpenTTCN 公司对TTCN工具的研究也很有历史. 其最新的工具OpenTTCN3[11]支持TTCN-3 和TTCN-2 测试集, 由OpenTTCN Tester和OpenTTCN SDK两部分构成: OpenTTCN Tester用于执行TTCN-3 编写的测试集; OpenTTCN SDK是针对编解码、测试适配器、平台适配器等开发的函数库.

Danet公司的Danet TTCN-3 tool[12]. 该工具将TTCN-3 转换为C++语言, 生成可执行测试集, 多用于电信、媒体和汽车工业等领域的软件测试.

Testing Technologies 公司推出的TT系列产品[13], 包括Ttthree, Ttanalyze, Ttspec, TTtwo2three. 其中, TTthree是TTCN-3 编译器, 将TTCN-3 代码编译成Java, 再生成可执行测试集; TTanalyze是TTCN-3 代码分析器, 验证TTCN-3 测试集的词法、语法; TTspec是一个图形化的测试集说明工具; TTtwo2three是TTCN-2 到TTCN-3 的转换工具. 2007 年, TT系列被集成到一个新的工具平台TTworkbench上, 自动化程度更高, 使用也更方便.

近几年, TTCN-3 的应用范围也不断拓广: 文献 [14,15]分别将TTCN-3 应用于RIPng协议和移动IPv6 协议的一致性测试中; 文献 [16]研究了移动管理协议BCMP的性能测试, 文献 [17]则

将TTCN-3 应用于分布式负载测试(load test)中. 此外, TTCN-3 也广泛应用于其他类型的测试中, 如服务测试 [18]、软件测试 [19,20]、入侵检测和被动测试 [21]等.

目前我国对于TTCN-3测试实现技术的研究尚在起步阶段, 主要集中于测试执行系统中单元技术的实现研究 [22~24]. 文献 [22]实现了对TTCN-3 部分核心语言的编译工作; 文献 [23]研究和实现了TTCN-3 测试系统管理, 包括: 测试前配置、初始化、测试控制、日志记录、测试判定和回收; 文献 [24]设计实现了基于C++语言转换的TTCN-3 执行器. 文献 [25]实现了一个TTCN-3测试平台TTPlatForm, 但并没有考虑多机分布式测试环境. 此外, 在基于TTCN-3在协议一致性测试 [26]、性能测试方面也有一些初步的研究.

作为本文的工作基础, 协议集成测试系统PITS[27](protocol integrated test system)是清华大学计算机系网络与协议测试实验室(CNPT Lab)于 90 年代独立开发的基于TTCN的网络协议测试系统, 主要用于测试协议实现与协议规范的一致性. 由于在分布式协议(如路由协议)的测试中显示很大的局限性, 因此又开发了基于TTCN-2 的PITSv2[28], 主要用于对IPv6 协议族的测试. 本文将介绍的PITSv3 是本实验室开发的基于TTCN-3 的协议测试系统.

从上述研究进展可见, 尽管国际上已经开发出了不少基于 TTCN-3 的测试系统, 但这些商用工具不仅价格高昂, 而且一般多用于软件测试和软件开发. 国内的相关工作主要集中在测试系统中某些单元技术的实现, 开发的基于 TTCN-3 的测试系统对分布式环境支持较弱.

2 TTCN-3 及其扩展

TTCN-3 作为一种新型测试描述语言, 其应用不仅局限在协议一致性测试上, 还可以广泛用于各种反应式系统的黑盒测试中. 从语法的角度看, TTCN-3 与在 ISO/IEC 9646 中定义的 TTCN-1 和 TTCN-2 有很大区别. 然而, 它保留了大量前两个版本中经证实的基本功能, 并在某些方面做了改进. 随着下一代互联网的发展, 对于新型协议的需求会越来越多, 而对于这些新协议的测试需求也将越来越多. TTCN-3 虽然具备强大的测试描述能力, 但在面对各种新的测试需求的情况下, 仍会出现测试描述能力不足的状况. 在这种情况下, 对 TTCN-3 进行扩展研究, 提高其测试描述能力也是十分必要的.

2.1 小节简单介绍 TTCN-3 核心语言; 2.2 小节介绍 TTCN-3 扩展已有的相关研究; 2.3 小节重点介绍我们针对协议鲁棒性测试所提出的一种 TTCN-3 扩展方案.

2.1 TTCN-3 核心语言简介

TTCN-3 具有更丰富的表示格式: 核心语言(core language)格式、表格表示格式、图形表示格式和用户自定义格式. TTCN-3 核心语言格式不仅方便测试人员编写测试集, 也满足了机器处理的需要.

在 TTCN-3 核心语言中, 最顶层单元是模块(module), 模块与模块之间相互独立, 通过 import 语句共享数据定义. 模块由两部分构成: 定义部分和控制部分(可选). 定义部分定义了测试部件(test component, 简称 TC)、通信端口(port)、数据类型(date types)、常量、测试数据模板(template)、函数(function)、测试例(test case)等. 模块控制部分包含局部定义、测试例执行顺序等. 图 2 是一个 TTCN-3 模块的例子, 定义了一个测试集(test suite), 其中包含若干测试例.

```
module    MyTestSuite {
        …
    const    integer      MyConstant := 1;    //const definition
    type    record        MyMessageType{…}    //data type
    template      MyMessageType    MyMessage := {…} //template
        …
    function      MyFunction1() {…}    //function
        …
    testcase      MyTestcase1() {…}    //test case
    testcase      MyTestcase2() {…}    //test case
        …
    control     //control part
    {
        …
      execute(MyTestCase1());    //test case execution
      execute(MyTestCase2());
        …
        }
  }
```

图 2　TTCN-3 模块示例

测试例定义了测试所要执行的测试行为, 用于判断被测实现 IUT(implementation under testing)是否符合要求. 测试例在模块定义部分定义, 在模块控制部分被调用, 每个测试例返回一个测试判决(verdict). 函数实现了某一常用的测试行为或计算等功能, 可被测试例调用.

TCN-3 支持动态分布式并发测试配置, 图 3 是一个 TTCN-3 动态分布式测试配置示例. 其中测试部件 TC 用于执行测试例中的测试行为, 而测试系统接口(test system interface)则定义了测试系统同被测系统 SUT 之间的接口, 即测试系统的边界.

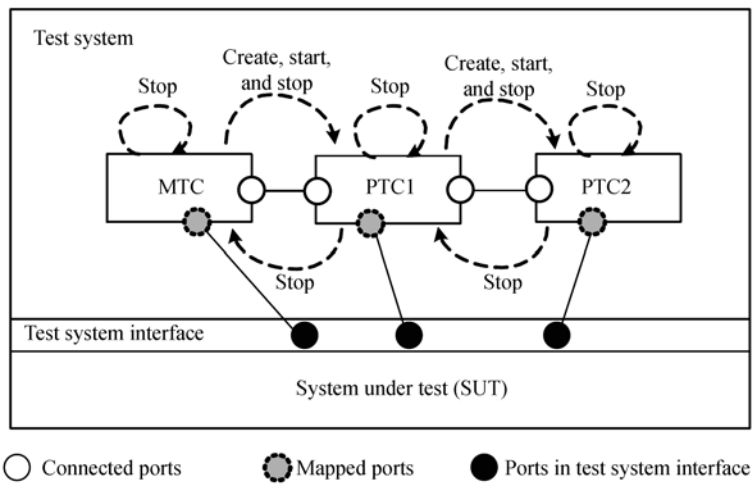

图 3　TTCN-3 动态分布式测试配置示例 [5]

在测试系统中可以包含多个测试部件, 有且只有一个主测试部件 MTC(main test component), 其余的都称为并行测试部件 PTC(parallel test component). MTC 在每个测试例开始执行时自动创建, MTC 终止时测试例也会终止; PTC 可以在测试执行过程中根据需要被动态地创建、执行和终止. 每个测试部件都包含一组端口, 有连接(connected)端口和映射(mapped)端口

两种类型: 连接端口用于测试部件间的通讯; 而映射端口则用于测试部件与 SUT 间的通讯.

2.2 TTCN-3 扩展技术

针对某些特定的测试需求, 通过扩展标准 TTCN-3 的语法和语义, 能够进一步增强 TTCN-3 的功能, 以满足更深入的测试描述需求. 现有的一些 TTCN-3 的扩展包括:

文献 [29,30]提出了一种TTCN-3 的实时扩展*TIMED*TTCN-3, 提供了测试实时需求的能力. *TIMED*TTCN-3 引入的主要新功能如下: 引入了新的测试判决conf, 表示功能通过, 非功能(时间需求)失败; 引入了绝对时间的概念, 提供了得到当前局部时间和测试部件定时执行的机制; 支持在线和离线评价(Online/offline evaluation). 文献 [31]将*TIMED*TTCN-3 应用于实时系统的互操作性测试中. 文献 [32]提出了一种TTCN-3 的面向对象扩展OO-TTCN-3, 并将其应用于电子商务测试中. 文献 [33]为了描述系统间通过持续信号的交互, 提出了一种称为Continuous的TTCN-3 扩展, 并将其应用于嵌入式控制系统的测试.

2.3 TTCN-3 基于报文变异的鲁棒性测试扩展

协议鲁棒性测试主要考察在无效输入以及大规模输入下协议实现能否正常工作. 在实际测试中, 一般采用大量异常注入的方法来测试协议实现的鲁棒性. TTCN-3 虽然已被应用于多种类型的测试, 但却不能很好地支持鲁棒性测试.

首先, TTCN-3 不能很好地支持数据变异操作. 异常报文是鲁棒性测试中必须的测试数据, 一般采用全局模板(global template)定义, 并且根据变异规则频繁地改动, 这在现有 TTCN-3 中很难实现. 另外, 尽管相关的鲁棒性测试算法并不复杂, 但采用 TTCN-3 对其进行描述却非常困难, 甚至不可能.

为了加强 TTCN-3 的鲁棒性测试描述能力, 本节对 TTCN-3 语法和语义进行了鲁棒性测试扩展, 分为单域变异扩展和多域变异扩展两个方面.

首先介绍单域变异鲁棒性测试的 TTCN-3 语法扩展:

- 增加关键词“loopreplace”. 该语句相当于针对单域的协议报文循环变异算法.
- 增加关键词“count”. count 的参数表示该域的变异次数.

在 loopreplace 语句里, 以全局模板格式定义的报文可以在测试例中被自动更改为无效报文, 这样测试例就不需要定义大量的异常报文. loopreplace 语句可以在函数和测试例中使用, 其 TTCN-3 语法的 BNF 范式定义如下:

```
FunctionStatement ::= ConfigurationStatements | TimerStatements | CommunicationStatements |
                      BasicStatements | BehaviourStatements |ValidateStatements |
                      SUTStatements | LoopReplaceStatement
LoopReplaceStatement::= LoopReplaceKeyword LoopReplacePar LoopReplaceBody
LoopReplaceKeyword::= “loopreplace”
LoopRepalcePar::=TemplateIdentifier ExtendedFieldReference OctetStringKeyword |
                 BitStringKeyword IntegerValue CountKeyword IntegerValue
                 [ConstIdentifier]
CountKeyword::= “count”
LoopReplaceBody::=BeginChar FunctionInstance SemiColon EndChar
```

下面是一个使用 loopreplace 语句的 TTCN-3 测试例描述.

```
testcase  Onebyone_HL1_Opt()  runs on MyTestComponentAsync
  system  SystemComponent  {
    map(mtc:MyPortAsync, system:SystemPort1);
    P1();
    loopreplace HL1.Opt octetstring 1 count 5 {
      Onebyone_HL1();
    }
    Normal_Verification();
    stop;
}
```

函数 P1( )执行后, 将执行 5 次函数 Onebyone_ HL1( ), 自动产生 5 个 octetstring 类型、长度为 1 的异常数据来代替报文 HL1 的 Opt 域, 每次循环注入一个异常报文. 5 次循环后, 执行函数 Normal_Verification()进行测试判决.

下面介绍多域变异鲁棒性测试的TTCN-3 语法扩展: 增加关键词“pairwise”. pairwise语句相当于针对多域的协议报文循环变异算法. 语句执行时, 分别针对报文的指定各域自动生成异常值, 然后自动根据pairwse算法 [34,35]生成异常值组合, 并循环注入到被测试系统中. pairwise语句同样可以在函数和测试例中使用, 其TTCN-3 语法的BNF范式定义如下:

```
FunctionStatement ::= ConfigurationStatements | TimerStatements | CommunicationStatements |
                      BasicStatements | BehaviourStatements |ValidateStatements |
                      SUTStatements | PairwiseStatement
PairwiseStatement::= PairwiseKeyword PairwisePar1  PairwisePar2 [PairwisePar3]
                     PairwiseBody
PairwiseKeyword ::= “pairwise”
PairwisePar1 ::=TemplateIdentifier
PairwisePar2 ::=LParen  pairwisefieldpar {SemiColon  pairwisefieldpar}  RParen
pairwisefieldpar ::= TemplateIdentifier OctetStringKeyword | BitStringKeyword  IntegerValue
                     CountKeyword  IntegerValue  [ConstIdentifier]
PairwisePar3 ::=TemplateIdentifier  TimerKeyword  pairwisefieldpar
PairwiseBody::= BeginChar FunctionInstance SemiColon EndChar
```

从上述扩展语法定义中可见, 该扩展除了支持协议报文域的变异之外, 还支持时钟取值的变异, 从而可以控制报文的发送时刻, 在鲁棒性测试中增加了时间约束.

下面给出一个采用 pairwise 语句并带有时间约束的测试例描述.

```
testcase Onebyone_HL1_Opt( )  runs on MyTestComponentAsync
  system SystemComponent  {
    map(mtc:MyPortAsync, system:SystemPort1);
    P1( );
    pairwise HL1(Mask octetstring 4 count 5;
                 Hint octetstring 2 count 6;
                 Opt octetstring 1 count 8 8bit_value )
          timer myTimer octetstring 1 count 8 {
      Onebyone_HL1();
    }
    Normal_Verification( );
    stop;
}
```

在函数 Onebyone_HL1()中, 异常报文 HL1 的发送时刻由 myTimer 控制. 该函数每执行一次, 报文及其到达时刻(即发送时刻)都会改变, Mask, Hint, Opt, myTimer 的可变值组合采用 pairwise 算法生成每次的多域变异数据.

扩展后的 TTCN-3 不仅可以方便地描述协议鲁棒性测试例, 而且可以在测试执行的过程中自动生成异常数据, 满足了协议鲁棒性测试的需求, 提高了 TTCN-3 的测试描述能力.

3 PITSv3 及其扩展

作为清华大学计算机网络协议测试实验室(CNPT Lab)的第 3 代协议测试系统, PITSv3 具备通用性、标准性、可扩展性 3 个重要特点, 是一个具有分布式多机并行测试能力的平台.

PITSv3 是独立于任何协议和任何协议实现的通用的测试平台, 它采用 TTCN-3 核心语言作为测试集描述法, 遵循国际标准化组织 ETSI 在 2003 年 2 月发布的国际标准 TTCN-3. PITSv3 不仅具有良好通用性, 而且能够支持复杂数据的描述和分布并发式的测试结构. PITSv3 支持描述结构嵌套和参数化的数据设计, 提高了测试数据的可复用性和编写效率. 同时, 也能够方便地实现多端口多主机的分布式并发测试结构.

另外, TTCN-3 本身具有一些描述性能测试的定义, 这些已经在 PITSv3 得到实现. 最重要的是, ETSI 宣布要继续加强 TTCN-3 对实时系统测试和性能测试方面的描述能力的研究. 本实验室也一直致力于 TTCN-3 描述能力的扩展研究, 并在 PITSv3 系统中加以实现. 这使得 PITSv3 具有良好的可扩展性, 能够不断地创新, 为下一代互联网中不断涌现的新协议新技术提供测试支持.

PITSv3 不仅仅注重功能设计和实现, 还提供一个界面友好, 使用方便, 将开发、调试和测试集于一体的集成开发环境 IDE(integrated development enviorement). 利用 PITSv3 的 IDE 可以方便地设计编译 TTCN-3 测试集、设计调试测试适配器, 选择测试例, 启动和中止测试活动, 显示和记录测试日志, 分析测试结果.

3.1 小节介绍 PITSv3 体系结构及各个主要模块的概要设计; 3.2 小节重点介绍 PITSv3 的分布式特性及其实现; 3.3 小节通过协议鲁棒性测试扩展实现介绍了 PITSv3 的可扩展性.

3.1 PITSv3 体系结构及概要设计

参考ETSI 给出的TTCN-3 结构建议, 结合前面两代测试系统的研究经验, 本节设计了协议集成测试系统PITSv3 的体系结构. PITSv3 由用户界面UI(user interface)、编译器(compiler)、测试管理TM(test management)、运行时系统T3RTS(TTCN-3 runtime system)、分布式运行环境DRE(distributed running environment)、编码解码器CD(coding and decoding)、系统时钟ST(system timer)、测试适配器SA (SUT adapter)8 个模块构成, 体系结构如 图4 所示.

PITSv3 是一个基于 TTCN-3 的协议测试系统, 能够支持分布式并行测试结构, 由一个主测试节点和若干个从测试节点组成. 每个测试节点上包含若干测试部件, 主测试部件位于主测试节点上, 其他测试部件均为从测试部件. 在测试过程中可以根据具体的测试需求和测试场景, 灵活选取实际的测试配置. 从测试节点的数目可以为 0, 这时只有主测试节点参与测试, 在这种情况下 PITSv3 退化为集中式测试系统. 其中主测试部件只能位于主测试节点中, 而从测试部件既可以位于主测试节点中也可以位于从测试节点中. 下面分别介绍各个模块的具体功能.

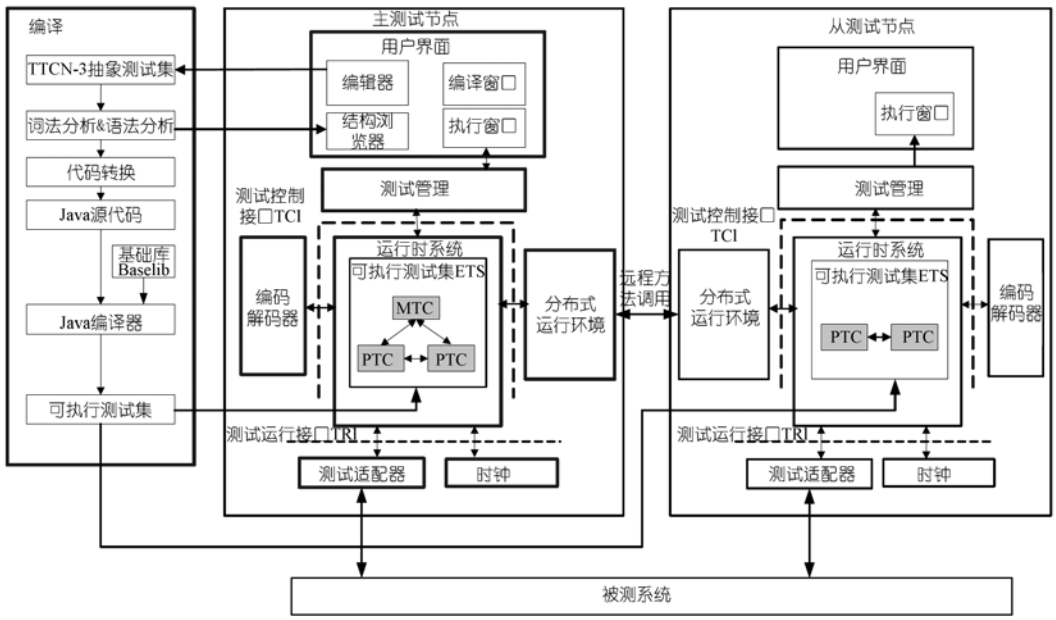

图 4　PITSv3 测试系统总体结构

用户界面 UI(user interface)是 PITSv3 与用户交互的窗口, 用户输入的信息、PITSv3 显示的信息都将在用户界面中体现, 用户界面贯穿了整个测试过程. 在测试准备阶段, 测试开发人员可通过用户界面的编辑器进行测试集及测试适配器的编写. 该编辑器提供 TTCN-3 核心语言及 Java 语言的编辑界面, 支持对编辑文件的浏览、修改, 并且能调用编译模块对其进行编译. 在测试执行开始的时候, 用户通过用户界面选定被测协议、对测试参数和分布式测试环境进行配置、选定测试执行模式、选择需要执行的测试例、并启动测试过程. 在测试执行的过程中, 用户界面实时显示测试日志. 在测试过程结束后, 用户界面生成测试结果文件, 而且还可以重现测试例的执行过程, 便于结果分析.

编译器(compiler)模块将用 TTCN-3 core language 描述的 TTCN-3 抽象测试集 ATS(abstract test suite)编译成可执行测试集 ETS(executable test suite). 编译模块首先对用户提交的 TTCN-3 ATS 进行词法分析和语法分析, 发现错误后, 及时向用户报错, 并进行错误定位. 通过词法分析和语法分析后, 进行 TTCN-3 代码转换, 生成相应的 Java 代码. 之后, 在基础库的支持下, 将生成的 Java 代码编译成可执行测试集 ETS.

测试管理模块(test manager)的主要功能是对测试系统进行全面的管理. 在 PITSv3 初始化完成后, 测试管理将维护本次测试执行过程中用户所选的测试例列表, 将用户设置的外部模块参数传递给 ETS, 启动所选的测试例. 另外, 在测试执行过程中, 测试管理将完成用户界面和可执行测试集 ETS 之间的交互. 最终, 测试执行结束后, 测试管理负责生成测试日志, 并在用户界面中显示.

作为TTCN-3 执行模块的一部分, 运行时系统(runtime system)的功能主要是为TTCN-3 可执行测试集ETS的正确执行提供支持. 运行时系统的主要功能包括: 定义并实现了ETS所需的

TTCN-3 测试数据类型; 在ETS运行过程中, 运行时系统提供了测试控制接口TCI (TTCN-3 control interface), 实现ETS与其他模块(例如: 测试管理TM、编码解码器CD等)之间的交互 [36]; 提供了测试运行接口TRI (TTCN-3 runtime interface), 实现ETS与时钟、测试适配器之间的交互 [37]. 总之, 可执行测试集ETS必须在运行时系统的支持下才能够正确执行测试例.

分布式运行环境(distributed running environment)负责分布在不同节点上的测试实体间的通信, 为实体间通信提供统一透明的接口, 屏蔽分布性. 分布式运行环境负责调度管理分布在不同测试节点上的测试部件的创建、执行、终止, 实现各个测试部件的通信、收集它们的测试判决、传递日志信息、并完成最后所有运行环境的清理复原工作.

PITSv3 为了实现测试数据在 TTCN-3 格式与实际传送的比特串之间的无缝转换, 专门设计了编码解码器 CD(coding and decoding)模块. 利用选定的编码器可以将 TTCN-3 的测试数据转换成适合于传送给被测系统 SUT (system under test)的比特串; 利用选定的解码器将测试适配器收到的数据转换成相应 TTCN-3 格式数据.

系统时钟 ST(system timer)模块的主要功能是向分布式运行环境中的各个测试节点提供统一时钟, 其功能是实现了所有的时钟及其相关的操作, 包括时钟的启动和停止, 读取时钟, 查询时钟状态等等.

测试适配器 SA (SUT adapter)主要实现 PITSv3 与被测系统 SUT 之间的通信. 测试适配器是 PITSv3 中唯一与具体协议相关的模块. 它在测试过程中主要完成相关协议测试数据的收发, 因此需要为不同的被测协议编写不同的测试适配器. 不同的协议采用不同的网络接口或者端口, 实现不同的校验和, 或者有些协议需要一些特殊的附加功能等, 所有类似的需求都将集中在测试适配器中得到解决. 因此, 当需要利用 PITSv3 开发一个下一代互联网的新协议或者新技术的测试能力时, 除了设计用 TTCN-3 描述的测试集外, 就只需要设计实现一个适当的测试适配器了. 由此可见, PITSv3 的通用性、标准性和可扩展性极大地缩短了开发周期, 提高了开发效率, 降低了开发难度.

3.2 分布式特性及其关键技术

PITSv3 是一个复杂的系统, 其设计实现需要考虑很多技术细节, 涉及很多技术创新. 本小节将只重点介绍实现分布式并发性的关键技术.

PITSv3 真正实现了多端口多测试节点的分布式测试, 能够提供并发能力, 从而大大加强了测试能力. 但是, 分布式特性也带来了一个必须解决的问题就是测试数据的分发和收集. 例如: 各个测试节点分布在各处之后, 在其上运行的所有指令都需要从主测试节点分发出去, 其产生的局部判决和局部日志都需要汇总回主测试节点. 另外, 每个测试节点都会使用一些全局环境变量, 对这些变量的统一管理也要在设计中充分考虑.

为了实现分布并发性, PITSv3 中分布式运行环境 DRE 模块负责提供一种分布式通信机制, 用于传递管理操作请求和测试数据. 根据通信双方的相对位置, 可分为本地通信(即通信双方位于同一个测试节点上)和异地通信(即通信双方位于不同测试节点上). 后者只能通过网络进行通信, 为保证其可靠性, 各测试节点将通过一个独立于被测实现的可靠 PITSv3 内部网络进行通信.

虽然本地通信与异地通信在实现机制上有很大差异, 但分布式运行环境 DRE 模块应屏蔽掉分布式细节, 为用户实体提供统一、透明的接口. 另外, 各个测试节点之间传输的数据规模有大有小, 通信机制应具有良好的数据规模适应性, 满足不同类型、不同大小的数据传输. 最后, 通信机制还应保证通信效率, 尽量避免在测试例执行过程中大量传输数据.

基于这些设计需求, PITSv3 中的分布式运行环境 DRE 的总体结构设计如图 5 所示.

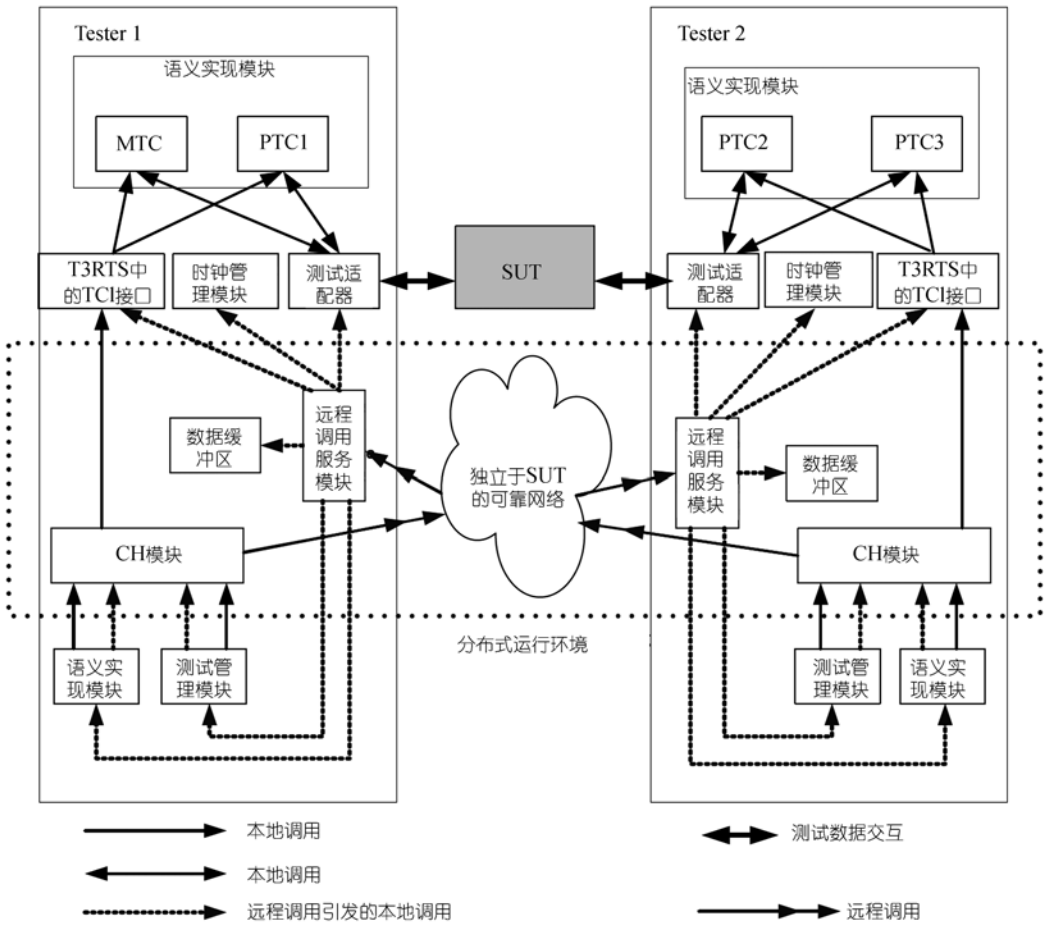

图 5　分布式运行环境的总体结构

分布式运行环境由部件控制 CH(component handing)模块、远程调用服务模块和数据缓冲区构成. CH 模块为系统提供了统一透明的接口, 屏蔽了主、从测试节点的差别, 本地通信和异地通信的差别. 在测试执行过程中, 测试执行模块调用 CH 模块进行通信, CH 模块根据命令要求分类处理: 如果是本地通信则调用本地的测试控制接口 TCI; 如果是异地通信则通过 Java RMI 来执行远程方法调用.

远程调用服务模块接收异地远程调用, 根据远程调用的参数从数据缓冲区取出相应的数据, 再通过执行本地调用来对远程调用进行响应并将响应的结果反馈给远程调用实体.

数据缓冲区存放测试部件间进行通信所需的数据. 数据缓冲区由索引表构成, 利用唯一的标识符进行区分, 即可以存放任意大小的数据, 也可以方便快捷地进行索引和存取数据, 从而解决了适应不同规模数据的问题.

当异地的用户实体间进行通信时, CH 模块执行远程调用, 而远程调用传递的参数是数据的标识符而不是真正的测试数据; 而要传递的测试数据则由接收方的远程调用服务模块根据标识符从本地数据缓冲区获取, 通过这种方式可以避免在测试执行期间大量数据的传输, 从而保证测试执行的效率.

3.3 可扩展性及其关键技术

根据 2.2 小节中的 TTCN-3 基于报文变异的鲁棒性测试扩展定义, 本节将说明如何在 PITSv3 中实现可扩展性. 任何对于 TTCN-3 的扩展, 都将从语法和语义两个方面在 PITSv3 中进行相应的扩展. 其扩展思路如图 6 所示.

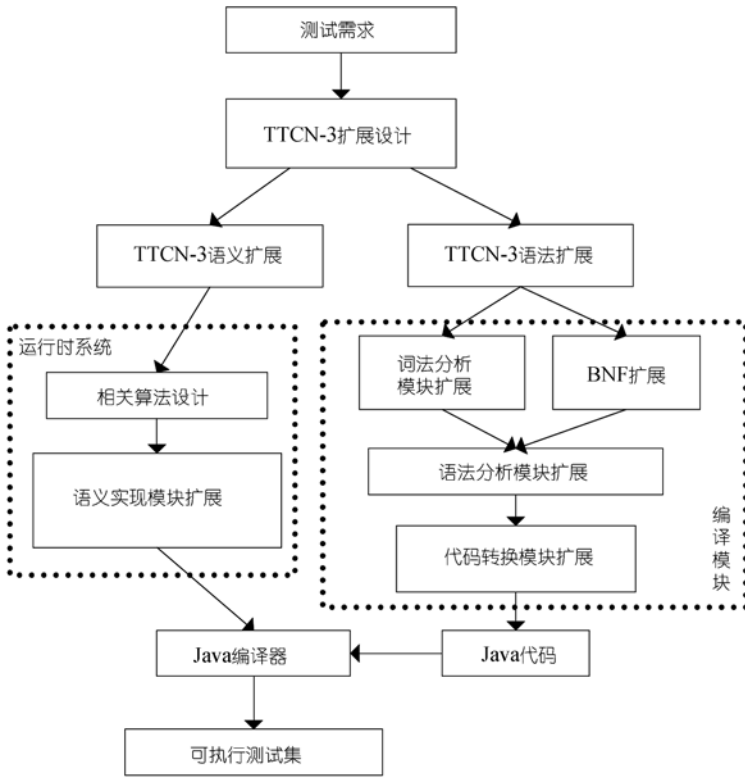

图 6 在 PITSv3 增加新功能的扩展思路

在语法方面, 对于 PITSv3 的扩展主要集中在编译模块: 首先根据增加的关键字, 对词法分析模块进行扩展; 根据扩展后的 BNF, 对语法分析模块进行扩展; 通过添加新的 Java 模板, 对代码转换模块进行扩展, 最终将扩展 TTCN-3 测试集转换为相应的 Java 代码.

在语义方面, 对于 PITSv3 的扩展主要集中于 TTCN-3 运行时系统的语义实现模块: 设计

相关算法(例如: 报文变异规则、pairwise 组合等), 进而对语义实现模块进行相应的扩展.

最后, 编译模块生成的 Java 代码和相应的语义实现代码通过 Java 编译器进行编译, 得到可执行测试集.

下面以 2.3 小节介绍的多域变异鲁棒性测试的 TTCN-3 扩展为例说明 PITSv3 系统可扩展性的实现方法: 首先, 在编译模块中增加 pairwise 语句, 分别根据 pairwise 等关键字、pairwise 语句的 BNF 范式等对词法分析模块、语法分析模块和代码转换模块进行扩展; 然后, 在 TTCN-3 运行时系统的语义实现模块实现 pairwise 的相关算法. 从而最终完成 PITSv3 系统对于多域变异鲁棒性测试的扩展. 限于篇幅关系, 具体实现细节不做赘述.

4 PITSv3 在协议测试中的应用

PITSv3 系统是一个通用的协议测试系统, 只需要做少量工作就可以提供对某种新协议或者新技术的测试支持. 首先在对协议内容和测试需求进行分析的基础上, 确定测试方法和测试配置, 进而设计实现该协议基于 TTCN-3 的测试集、基于 PITSv3 的测试适配器和编码解码器. 在上述工作完成之后, 就可以搭建由 PITSv3 系统和被测协议实现组成的测试环境, 在 PITSv3 系统上执行 TTCN-3 测试例, 得到测试结果并分析形成测试报告. 整个工作过程如图 7 所示.

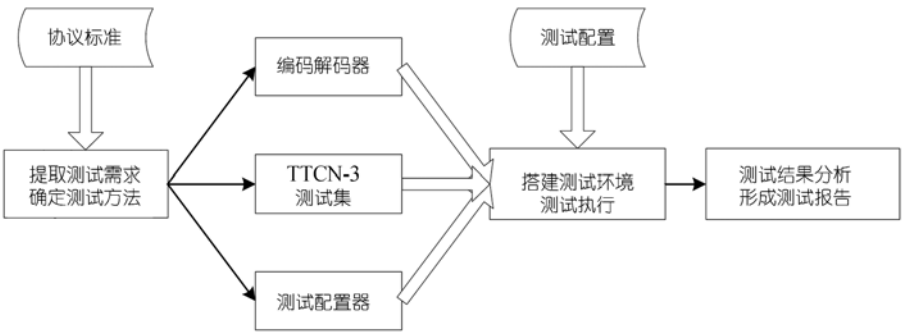

图 7 PITSv3 系统的协议测试流程

PITSv3 研发完成后, 已经在中国移动研究院、中国教育与科研计算机网、中国下一代互联网 CNGI 等单位和项目中承担过设备选型测试、招标测试、验收测试等工作.

4.1 小节的 BGPv4+一致性测试应用重点说明 PITSv3 通用的分布式特性, 4.2 小节的 OSPFv2 基于报文变异的鲁棒性测试应用重点说明 PITSv3 的可扩展性.

4.1 BGPv4+协议一致性测试

边界网关协议BGPv4(border gateway protocol version 4, RFC4271)[38]是一个用于自治系统之间的复杂的分布式动态路由协议, 也是目前唯一使用的域间路由协议. 但是BGPv4 只能广播IPv4 的路由信息, 为了能够使BGPv4 携带多种网络层协议(如IPv6, IPX等)的路由信息, 适应下一代互联网的需求, 互联网工程任务组IETF在RFC2858 定义了BGPv4+[39], 该协议是一种 BGPv4 的扩展机制, 用于支持多种网络层协议的路由. 目前, BGPv4+是下一代互联网的核心

边界网关协议, 本小节将介绍如何利用PITSv3 对BGPv4+协议一致性测试进行开发及应用.

根据BGPv4+的协议内容及其特点, 在一致性测试中将采用 3 种不同的测试配置方法进行测试. 对协议状态机和帧格式的测试采用单控制观察点测试配置(参见 图8(a)), 对路由信息转发功能的测试采用双控制观察点测试配置(参见 图8(b)), 对路由信息处理功能的测试采用多控制观察点测试配置(参见 图8(c)).

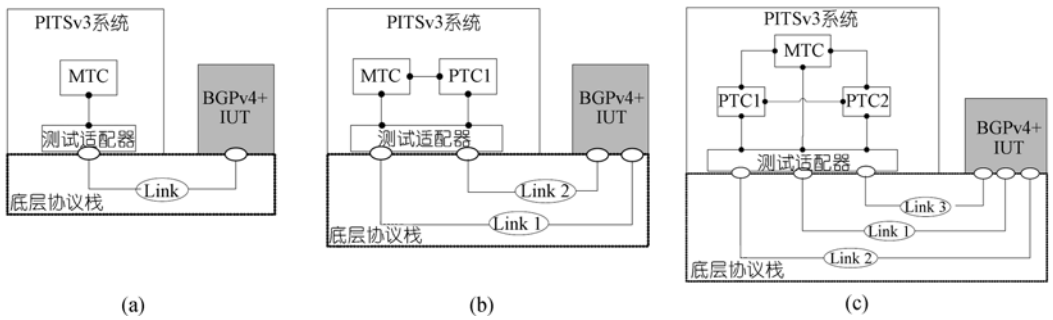

图 8 BGPv4+的协议一致性测试方法

(a) 单控制观察点测试配置: PITSv3 中只包含一个 MTC, 通过一个测试端口与被测系统相连; (b) 双控制观察点测试配置: PITSv3 中包含一个 MTC 和一个 PTC. 每个测试部件均有两个端口: 一个与被测系统相连, 一个与另一个测试部件相连; (c) 多控制观察点测试配置: PITSv3 中包含 3 个测试部件, 一个 MTC, 两个 PTC. 每个测试部件均有 3 个端口: 一个与被测系统相连; 另两个端口与其他测试部件相连

采用 TTCN-3 语言设计的 BGPv4+一致性测试集结构如表 1 所示. 其中测试组 BGP_STATE 和 BGP_FARME_ERROR 采用单控制观察点测试配置; 测试组 BGP_ROUTING_FORWARDING 采用双控制观察点测试配置; 测试组 BGP_ROUTING_HANDLING 采用多控制观察点测试配置.

表 1 BGPv4+的 TTCN-3 一致性测试集

测试组名称	测试组测试目的	测试例个数	测试方法
BGP_STATE	状态机的测试	15	单控制观察点测试配置
BGP_FRAME_ERROR	帧格式的测试	77	单控制观察点测试配置
BGP_ROUTING_FORWARDING	路由信息转发的测试	20	双控制观察点测试配置
BGP_ROUTING_HANDLING	路由信息处理的测试	14	多控制观察点测试配置

测试适配器是 PITSv3 系统中唯一跟协议相关的模块, 应针对不同协议设计不同的测试适配器. PITSv3 中测试适配器的实现方法是提供一个基类 TestAdapter, 实际协议如 BGPv4+协议的测试适配器通过继承自 TestAdapter 类的 BGP_TestAdapter 类完成具体实现.

BGPv4+协议是建立在 TCP 之上的路由协议, 所以测试适配器就直接调用 TCP 协议之上的 Socket 来完成相应测试数据的发送和接收. 在接收数据的时候, TCP socket 建立连接后启动接收线程进入端口监听状态, 当接收到测试数据后, 根据需要处理接收到的数据(如将带有错误具体信息的 notification 帧进行重组等), 并将处理后的数据压入接收队列并通知 PITSv3 进行

处理. 在发送数据的时候, TCP socket 建立连接后向相应的被测系统端口发送测试数据.

完成了 BGPv4+协议的一致性测试开发后, 将某一双栈核心路由器设备作为被测系统, 分别搭建了 3 种不同的测试配置, 进行了实际测试活动. 在测试活动中, 对于路由信息处理功能的测试, 将多控制观察点测试配置(参见图 8(c))分别在两个测试节点和 3 个测试节点的分布式多机环境中进行了实施, 其结构图参见图 9. 其中图 9(a)是 2 个测试节点的多机测试环境, MTC 位于主测试节点上, 而 2 个 PTC 都位于从测试节点; 图 9(b)是 3 个测试节点的多机测试环境, MTC 和两个 PTC 分别分布在 3 个不同的测试节点上.

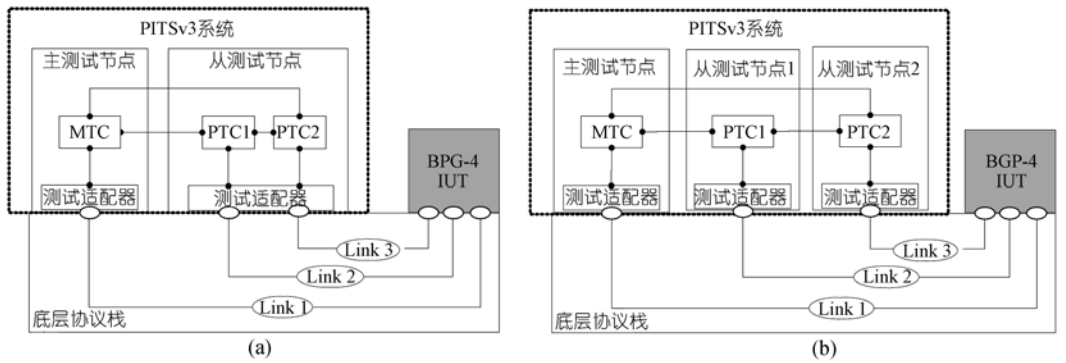

图 9　多控制观察点测试配置的多机测试环境结构图

(a) 2 个测试节点的多机测试环境; (b) 3 个测试节点的多机测试环境

由此可见, PITSv3 系统通过分布式运行环境 DRE 提供的底层支持, 提供了对各种分布式测试架构的通用支持, 可根据实际的测试需要和测试系统资源情况选择适当的分布式测试架构进行测试活动.

4.2　OSPFv2 基于报文变异的鲁棒性测试

本小节将介绍如何利用PITSv3 系统对OSPFv2[40]协议进行基于报文变异的鲁棒性测试.

根据 OSPFv2 协议内容及其特点, 鲁棒性测试将采用图 8(a)中的单控制观察点测试配置. OSPFv2 规范定义了 5 种报文: Hello, Database Description (DDP), Link State Request (LSR), Link State Update (LSU)以及 Link State Acknowledgment (Ack). 还定义了 5 种链路状态宣告 LSAs (link state advertisements). OSPFv2 鲁棒性测试集设计如表 2 所示, 其中包含 130 个单域变异鲁棒性测试例, 67 个多域变异的鲁棒性测试例. 12 个测试组涉及到了 OSPFv2 规定的 5 种报文和 5 种 LSA.

和 4.1 小节中介绍的 BGPv4+的测试适配器一样, PITSv3 中 OSPFv2 鲁棒性测试的测试适配器也是由继承自 TestAdapter 基类的 OSPFRobust_TestAdapter 类实现了发送和接收测试数据功能.

因为 OSPF 数据的发送与接收需要用 rawsocket 实现, 而 Java 不提供 rawsocket 函数, 所以在设计测试适配器时就使用了 Jpcap 类库. Jpcap 类库通过底层调用 winpcap 获取数据, 但是它并不能发送 rawsocket 数据, 还需要对其进行扩展. 发送 OSPFv2 数据时, 首先在 jpcap 中定义

PITSv3 需要调用的类或对象, 在得到 PITSv3 要发送的数据后, 生成 Jpcap 模块中发送对象, 将数据发出. 在接收数据的时候, 先判断接收端口, 若匹配, 则在相应的端口启动 Jpcap 接收数据.

表 2 基于报文变异的 OSPFv2 鲁棒性测试集 a)

测试组名	测试内容 (state/invalid PDU received)	测试例个数 (single-field/multi-field)
OSPF Head	Init / Hello	14 / 8
Hello	2-way / Hello	16 / 6
DDP0	Exstart / DDP0(without LSA Header)	8 / 4
DDP1	Exchange / DDP1(include one LSA Header)	22 / 12
LSR	Exchange / LSR	6 / 2
Ack	Exchange / Ack	16 / 6
LSA_HEAD	Exchange / LSU(include Router_LSA)	14 / 10
LSU_RLSA	Exchange / LSU(include Router_LSA)	16 / 10
LSU_NLSA	Full / LSU (include Network _LSA)	4 / 5
LSU_S3LSA	Full / LSU (include Type3 Summary _LSA)	4 /1
LSU_S4LSA	Full / LSU (include Type4 Summary _LSA)	4 /1
LSU_AsLSA	Full / LSU (include As External_LSA)	6 /2
	Total:	130 / 67

a) 测试内容一栏表示各测试数据是在状态机中那个相应状态注入的

完成了OSPFv2 协议的鲁棒性测试开发后, 将Zebra-0.94[41]作为被测系统, 进行了实际测试活动.

5 小结

本文首先介绍了协议测试的形式化定义、测试集描述法和测试实现技术的发展历程与最新进展; 之后, 研究了 TTCN-3 及其扩展技术, 设计了 TTCN-3 关于鲁棒性测试的语法语义扩展; 然后, 在设计实现了基于 TTCN-3 的协议集成测试系统 PITSv3, 并对其鲁棒性测试能力进行了扩展; 最后介绍了 PITSv3 在 BGPv4+协议一致性测试和 OSPFv2 协议鲁棒性测试方面的应用情况.

在后续工作中, 希望能够继续研究 TTCN-3 对于实时性和安全性等方面的扩展, 并对其在 PITSv3 中进行实现.

致谢 作者感谢田北航、陈宁、靳立法、王霆、黄次辉、陈元等在 PITSv3 系统实现方面所做的大量工作.

参考文献

1 European Telecommunications Standards Institute (ETSI). The Testing and Test Control Notation version 3; Part 1: TTCN-3 Core Language. ETSI standard ES 201 873-1 V3.2.1(2007-03). 2007

2 Piatkowski T F. An engineering discipline for distributed protocol systems. In: Proceedings of IFIP Workshop on

Protocol Testing, 1981
3 尹霞. 一种协议集成测试理论及其技术研究. 博士学位论文. 北京: 清华大学, 2000
4 ISO/IEC. Information technology, open systems interconnection, conformance testing methodology and framework. ISO/IEC 9646. 1991
5 Grabowski J, Hogrefe D, Rethy G, et al. An introduction to the testing and test control notation (TTCN-3). Comput Netw, 2003, 42(3): 375—403 [DOI]
6 Grabowski J, Wiles A, Willcock C, et al. On the design of the new testing language TTCN-3. In: Ural H, Probert R L, Bochmann G V, eds. Proceedings of the 13th IFIP International Conference on Testing of Communicating Systems (TestCom 2000). Ottawa: Kluwer Academic Publishers, 2000. 161—176
7 Baker P, Rudolph E, Schieferdecker I. Graphical test specification-the graphical format of TTCN-3. In: Reed R, Reed J, eds. Proceedings of the 10th International SDL Forum (SDL 2001). LNCS, Vol 2078. Copenhagen: Springer, 2001. 148—167
8 Schulz S, Vassiliou-Gioles T. Implementation of TTCN-3 test systems using the TRI. In: Schieferdecker I, König H, Wolisz A, eds. Proceedings of the 14th IFIP International Conference on Testing of Communicating Systems (TestCom 2002). Berlin: Kluwer Academic Publishers, 2002. 425—442
9 Schieferdecker I, Vassiliou-Gioles T. Realizing distributed TTCN-3 test systems with TCI. In: Hogrefe D, Wiles A, eds. Proceedings of the 15th IFIP International Conference on Testing of Communicating Systems (TestCom 2003). LNCS, Vol 2644. Sophia Antipolis: Springer, 2003. 95—109
10 Telelogic TAU/Tester. http://www.telelogic.com
11 OpenTTCN. http://www.openttcn.com
12 Danet's TTCN-3 Toolbox. http://www.danet.com
13 Testing Technology. http://www.testingtech.de
14 Floch A, Roudaut F, Sabiguero A, et al. Some lessons from an experiment using TTCN-3 for the RIPng testing. In: Khendek F, Dssouli R, eds. Proceedings of the 17th IFIP International Conference of Testing of Communicating Systems (TestCom 2005). LNCS, Vol 3502. Montreal: Springer, 2005. 318—332
15 Noudem F N, Viho C. Modeling, verifying and testing mobility protocol from SDL language. In: Prinz A, Reed R, Reed J, eds. Proceedings of the 12th International SDL Forum (SDL 2005). LNCS, Vol 3530. Grimstad: Springer, 2005. 198—209
16 Dibuz S, Szabo T, Torpis Z. BCMP performance test with TTCN-3 mobile node emulator. In: Groz R, Hierons R M, eds. Proceedings of the 16th IFIP International Conference of Testing of Communicating Systems (TestCom 2004). LNCS, Vol 2978. Oxford: Springer, 2004. 50—59
17 Din G, Tolea S, Schieferdecker I. Distributed load tests with TTCN-3. In: Uyar M Ü, Duale A Y, Fecko M A, eds. Proceedings of the 18th IFIP International Conference on Testing of Communicating Systems (TestCom 2006). LNCS, Vol 3964. New York: Springer, 2006. 177—196
18 Deussen P H, Din G, Schieferdecker I. A TTCN-3 based online test and validation platform for Internet services. In: Proceedings of the 6th International Symposium on Autonomous Decentralized Systems (ISADS 2003). Pisa: IEEE Computer Society, 2003. 177—184
19 Nyberg A J. Use of TTCN-3 for software module testing. In: Uyar M Ü, Duale A Y, Fecko M A, eds. Proceedings of the 18th IFIP International Conference on Testing of Communicating Systems (TestCom 2006). LNCS, Vol 3964. New York: Springer, 2006. 161—176
20 Blom S, Deiß T, Ioustinova N, et al. TTCN-3 for distributed testing embedded software. In: Virbitskaite I, Voronkov A, eds. Proceedings of the 6th International Andrei Ershov Memorial Conference - Perspectives of System Informatics (PSI 2006). LNCS, Vol 4378. Novosibirsk: Springer, 2007. 98—111
21 Brzezinski K M. Intrusion detection as passive testing- linguistic support with TTCN-3 (extended abstract). In: Hämmerli B M, Sommer R, eds. Proceedings of the 4th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA 2007). LNCS, Vol 4579. Lucerne: Springer, 2007. 79—88
22 钱向红. TTCN-3 编译技术研究与实现. 硕士学位论文. 成都: 西南交通大学, 2004

23 萨智海. TTCN-3 测试管理的设计与实现. 硕士学位论文. 呼和浩特: 内蒙古大学, 2005
24 蒋凡, 张辉. 基于 C++语言转换的 TTCN-3 执行器的设计与实现. 中国科学技术大学学报, 2007, 37(9): 1155—1158
25 张辉, 蒋凡. 基于 C++语言转换的 TTCN-3 测试系统的设计与实现. 计算机系统应用, 2007, 10: 64—67
26 黄传动, 蒋凡. 基于 TTCN-3 的 SIP 一致性测试. 小型微型计算机系统, 2007, 28(9): 1614—1618
27 吴建平, 陈修环, 郝瑞兵, 等. 基于形式化技术的协议集成测试系统——PITS. 清华大学学报(自然科学版), 1998, 38(S1): 26—29
28 王之梁. IPv6 协议测试系统核心模块的设计与实现. 硕士学位论文. 北京: 清华大学, 2003
29 Dai Z R, Grabowski J, Neukirchen H. *TIMED*TTCN-3 - a real-time extension for TTCN-3. In: Schieferdecker I, König H, Wolisz A, eds. Proceedings of the 14th IFIP International Conference on Testing of Communicating Systems (TestCom 2002). Berlin: Kluwer Academic Publishers, 2002. 407—424
30 Dai Z R, Grabowski J, Neukirchen H. *TIMED*TTCN-3 based graphical real-time test specification. In: Hogrefe D, Wiles A, eds. Proceedings of the 15th IFIP International Conference on Testing of Communicating Systems (TestCom 2003). LNCS, Vol 2644. Sophia Antipolis: Springer, 2003. 110—127
31 Wang Z L, Wu J P, Yin X, et al. Using *TIMED*TTCN-3 in interoperability testing for real-time communication systems. In: Uyar M Ü, Duale A Y, Fecko M A, eds. Proceedings of the 18th IFIP International Conference on Testing of Communicating Systems (TestCom 2006). LNCS, Vol 3964. New York: Springer, 2006. 324—340
32 Probert R, Xiong P, Stepien B. Life-cycle e-commerce testing with OO-TTCN-3. In: Núñez M, Maamar Z, Pelayo F L, et al, eds. Proceedings of FORTE 2004 Workshops on the FormEMC, EPEW, ITM. LNCS, Vol 3236. Toledo: Springer, 2004. 16—29
33 Schieferdecker I, Großmann J. Testing embedded control systems with TTCN-3 - an overview on TTCN-3 continuous. In: Obermaisser R, Nah Y, Puschner P P, et al, eds. Proceedings of the 5th IFIP Workshop on Software Technologies for Future Embedded and Ubiquitous Systems (SEUS 2007). LNCS, Vol 4761. Santorini Island: Springer, 2007. 125—136
34 Lei Y, Tai K C. In-parameter-order: a test generation strategy for pairwise testing. In: Proceedings of the 3rd IEEE International Symposium on High-Assurance Systems Engineering (HASE 1998). Washington: IEEE Computer Society, 1998. 254—261
35 Tai K C, Lei Y. A test generation strategy for pairwise testing. IEEE T Softw Eng, 2002, 28(1): 109—111 [DOI]
36 ETSI. Methods for Testing and Specification (MTS); The Testing and Test Control Notation Version 3; Part 6: TTCN-3 Control Interface. ETSI Standard ES 201 873-6 (V3.2.1). 2007
37 ETSI. Methods for Testing and Specification (MTS); The Testing and Test Control Notation Version 3; Part 5: TTCN-3 Runtime Interface. ETSI Standard ES 201 873-5 (V3.2.1). 2007
38 Rekhter Y, Li T, Hares S. A Border Gateway Protocol 4 (BGP-4). RFC 4271. 2006
39 Bates T, Rekhter Y, Chandra R, et al. Multiprotocol extensions for BGPv4. RFC 2858. 2000
40 Moy J. OSPF Version 2. RFC 2328. 1998
41 Zebra-0.94. http://www.zebra.org/