下一代网络协议一致性测试执行系统的实现*)

叶新铭　王红霞　石立新
(内蒙古大学计算机学院　呼和浩特010021)

The Implement of Test Executing System Based Linux Environment

YE Xin-Ming　WANG Hong-Xia　SHI Li-Xin
(Computer College, Inner Mongolia University, Huhehaote 010021)

Abstract　In order to go along the work of next generation network protocol conformance testing effectively and thorough, we bring forward a sign idea about test executing system and implement it base on Linux environment, which is obtained based ISO' criterion and method. The main function of test executing system is to execute automatically the test suit in TTCN form in course of protocol conformance testing and present the testing report in the end. In this paper, we first introduce the realization idea of test executing system base on Linux environment and some interrelated arithmetic. We emphasize on the sign flow and implement procedure of test executing system and discuss some arithmetic for central module. Finally, we give test procedure and test result for IPv6 protocol using this test executing system.

Keywords　Conformance testing, Test executing system, TTCN

0. 前言

近几年，随着计算机网络和通讯技术的不断发展特别是开放型异构网络的迅猛发展，协议的设计和实现变得越来越复杂，协议测试理论和技术的研究越来越重要，已经成为国际上计算机网络研究的热点之一。

协议一致性测试是指针对一个网络协议的实现，测试者依据协议标准文本通过一定的测试例对其进行某些方面的测试，以检验其实现是否与协议标准文本描述的功能相一致。测试例的运行需要一定的环境，一个能够运行测试集来进行协议一致性测试的实验环境称为测试平台。

近几年，随着计算机网络和通讯技术的不断发展，出现了MPLS协议[1]和IPv6协议[2]等一系列新的下一代网络协议。但由于各协议厂家对协议的不同理解，通信设备的协议非一致性问题日趋严重，因而协议的一致性测试也变得越来越重要。为了检查各个不同生产厂家的实现是否与标准文本相一致，进而确保不同厂家的网络协议实现之间能够互操作，必须进行实际的协议一致性测试。虽然协议一致性测试的技术已经发展了许多年，对于一致性测试执行的实现也做了许多研究[3~6]，但是关于这方面的工具却很少，尤其是一致性测试的执行工具。为了能够更有效、更深入地进行下一代网络协议一致性测试工作，我们结合ISO提出的关于测试例的描述语言、测试例的产生、测试例的选择、测试例的执行等一系列的标准和方法，提出了Linux环境下的测试执行系统的设计思想并进行了实现。我们实现的测试执行系统的主要功能是自动执行TTCN描述的测试集，给出测试结果和完整的测试报告。

我们开发的协议测试执行系统，具备易移植、易扩充、界面友好和执行效率高等一系列的优点，能够对各种网络协议进行一致性测试，从中找出协议实现过程中的问题，对下一代网络协议的开发和实现有着重要意义。

1. 协议一致性测试流程

测试的工作流程是根据协议的一致性测试的过程来确定的。首先，根据协议标准文本的自然语言描述，用形式说明方法(如:Lotos、Petri网、SDL、有限状态自动机FSM等)来产生协议的形式化描述。然后从协议的形式化描述出发，根据测试所要求达到的标准，从中抽取出测试目的，这些测试目的要满足所要求的差错覆盖标准。然后采用自动或人工的测试例生成方法产生由TTCN描述的测试例描述集。由于产生的测试集并不都是可以执行的，其中，某些测试例的执行可能是无意义的，因此，必须对测试例集中的测试例进行选择。选择后的测试例在测试平台中以解释方式执行，最终产生关于本测试例的测试结果，并送往测试报告输出。整个一致性测试流程如图1。

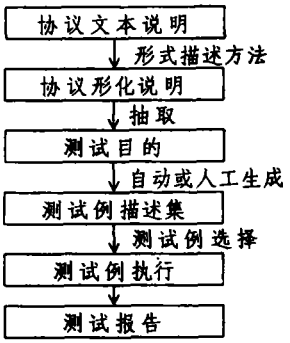

图1　协议一致性测试流程图

2. 测试执行系统的设计思想

被选择的测试例必须能够在测试环境中执行，产生相应的测试事件，在测试执行系统与被测实现IUT之间的多个PCO上交换测试数据，观察IUT对测试事件的响应，以判断其实现是否与协议说明相一致(图2)。

*)国家自然科学基金项目(69863001)和内蒙古重点领域项目(ZL9902)。叶新铭　教授，博士生导师。石立新　副教授。王红霞　硕士研究生。主要研究方向为协议一致性测试。

对整个测试流程来讲，测试例的成功执行是其全部功能的核心和关键。我们使用解释执行的方法来实现测试执行系统，并且采用了一些关键性技术，克服了解释执行方法效率低下的缺点。在具体实现过程中，我们参考清华大学提出的一种形式化测试执行方法[7]，将整个测试执行过程分为两个阶段：测试例的解释阶段和测试例的执行阶段。解释阶段的工作是由 TTCN 描述的测试例到 TTCN 求值树的转换，另外还要完成对 TTCN 描述的测试例的语法检查、静态语义检查和到 TTCN 求值树的转换，这部分工作由解释器来完成。对测试例的语法检查和静态语义检查基于 ISO9646中的 TTCN 的巴克斯语法和静态语义的定义来完成。我们着重介绍从正确的 TTCN 测试例到 TTCN 求值树的转换。执行阶段完成对求值树的执行，根据被测实现的响应，得到测试判决，这部分工作由执行器来完成。执行器从求值树的根出发，按照层次关系依次去执行求值树的各结点，即 TTCN 事件，然后调用相应的事件处理函数，同时根据被测实现的响应，得到判定结果。

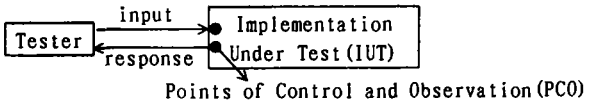

图2　测试过程

3. 测试执行系统的系统结构和实现

下面着重介绍这部分的设计和实现。我们采用 ISO 9646 中定义的分布式测试方法，在该方法中，下测试器位于测试系统中，而上测试器位于被测系统中。图3给出了这个测试结构图。其中测试执行系统由用户界面、解释器、执行器和驱动程序这几个核心部件组成，这些部件全部是用软件实现的。

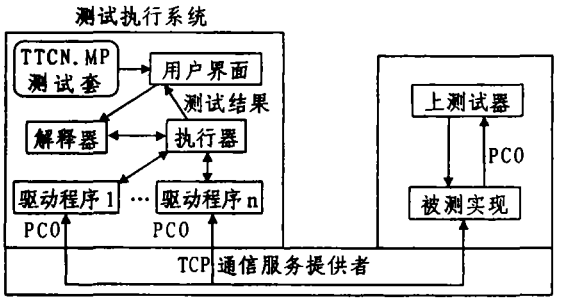

图3　测试结构图

在用户界面中，用户可以打开 MP 格式的测试套，然后用户界面对 MP 文件进行语法检查和静态语义检查，之后把正确的 TTCN.MP 文件送给解释器。解释器抽取出 MP 文件中定义部分、约束部分以及测试例的相关内容，填充在自定义的数据结构中，然后根据得到的数据结构来生成求值树。执行器再根据求值树来执行测试例，根据被测实现的响应，得到测试判决，最后把测试结果和测试报告交给用户界面。

在实际的环境中进行测试时，执行器和驱动程序一起使用，驱动程序是执行器和被测实现进行通信的一个接口，其中定义了被测协议中所定义的数据单元格式，以及和执行器通信的一些消息，包括：SM_SHAKE_HAND（握手消息）、SM_SET_PARA_VALUE（设置 PCO 参数消息）、SM_GET_PARA_LIST（读取参数列表消息）、SM_INIT_FOR_RUN（运行初始化消息）、SM_BEFORE_RUN_CASE（测试例运行前消息）、SM_AFTER_RUN_CASE（测试例运行后消息）、SM_PCO 消息等。执行器在执行求值树中 Send 事件时，把解释器构造好的 PDU/ASP 送给驱动程序，由驱动程序加上被测协议的数据包所需的头部和尾部，再送给被测实现去处理；当被测实现给出回应的数据包后，再由驱动程序负责剥掉包头和包尾，把数据包的实际内容交给执行器，执行器把收到的包的内容和预期包的内容进行比较，得到最终的测试结果。

3.1　解释器和执行器

测试执行系统的总体结构是根据 ISO 9646中提出的测试方法构造出来的，在测试结构图中，解释器的实现是重点部分，下面给出这部分的总体结构。

解释器完成从 TTCN 描述的测试例到 TTCN 求值树的转换，它首先由用户界面程序得到下一个将要执行的测试例，并对其进行解释。解释过程的核心内容共有三大模块：抽取 MP 文件中的数据内容、构造自己的数据结构和构造执行求值树。其总体模块图见图4。解释器的实现思想是，首先多次扫描正确的 MP 文件，把测试例中的测试步进行扩展，把自定义的简单和结构类型定义都转化成 TTCN 中预定义类型，这样处理之后，MP 议论中定义部分和约束部分的数据类型都是 TTCN 中预定义的数据类型。根据简化过的定义部分和约束部分，构造出自己的数据结构。然后，分析 MP 文件中的测试例，把层号、动态行为描述及约束参考等信息记录在相关的数据结构中，把测试例中的事件按照层次造成一棵二叉树，这就形成了一棵求值树。当 TTCN 事件是 Send 和 Receive 时，解释器根据测试例的约束参考来构造出 ASP/PDU 的内容，保存在执行树中。

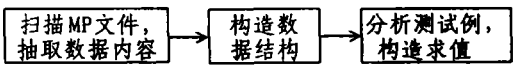

图4　解释器的总体结构模块图

解释器把生成的求值树放在共享内存中供执行器使用，执行器依据 TTCN 求值树，基于 TTCN 的操作语义来完成执行过程，从求值树的根出发，按照层次关系去执行求值树的各个结点，如果该结点执行成功，则去执行其顺序测试事件，否则执行该结点的选择测试事件。执行器通过驱动程序完成发送和接收事件，同时改变共享内存测试例中的某些变量值，最后通过被测实现的响应，得到测试判决，提交给用户界面。

3.2　算法思想

在以上三个模块中，构造数据结构是为执行 TTCN 事件做准备，执行器在执行 TTCN 事件 Send 时，把解释器构造好的 PDU/ASP 送给驱动程序；在执行 Receive 事件时，把收到的包的数据部分和构造出预期的包的数据部分进行比较，得到测试结果。求值树是测试例执行的依据，每个树的结点都是一个 TTCN 事件，执行测试例时，从执行求值树的根出发，按照层次关系，依法向下执行。我们构造的求值树是使用二叉树来实现的，每个结点的右结点表示与该结点是顺序关系的 TTCN 事件，而左结点表示与该结点是选择关系的 TTCN 事件，这样构造的好处是层次分明，易于处理。此外，在 ISO9646-3中，ISO 针对已经构造好的 TTCN 求值树提供了解释执行的伪代码，在测试执行系统的实现过程中，我们将这些伪代码翻译为具有单一功能的库函数，当解释执行某条测试事件时，执行器自动调用相应的库函数进行处理，以完成对具体测试事件的执行。

下面给出构造数据结构和构造求值树的算法。

算法1:构造数据结构

```
BEGIN
    Scan the TTCN.MP,expand the teststep in the test case;
      /*扩展测试步*/
     Translate the simple type definitions to predefined TTCN
       types; /*处理简单类型*/
    Replace all symbol constant with the value of constant; /*处
      理符号常量*/
    /*消除定义部分中的嵌套*/
    Eliminate the nested definition in structured type definition and
      PDU type definition;
    /*消除测试例中约束参考栏中带参数的约束*/
    Translate constraints reference with parameters to constraints
      reference without parameters in test case;
    /*消除约束部分的嵌套定义*/
     Eliminate the nested definition in structured type constraint
      declarations and PDU constraint declarations in the
      constraints part;
    /*构造自己的数据结构*/
     Construct the data structure based the declarations part and
      constraints part in TTCN.MP.
END;
```

算法2:构造求值树

```
BEGIN
  /*分析测试例,把层号、动态行为描述及约束参考等信息记录在
    相关的数据结构中*/
  Analyse the test case and fill the information such as label,
    behavior description and constraints ref etc in data structure;
  /*把测试例中的事件按照层次构造成一棵二叉树,形成求值树
    */
  Construct binary tree based level with TTCN events and form
    executing tree.
END;
```

3.3 驱动程序

驱动程序是执行器和被测实现之间的接口。解释器根据测试例中的动态行为描述和约束参考来构造 ASP/PDU,执行器执行 Send 事件时,把解释器构造好的 ASP/PDU 直接送给驱动程序,驱动程序按照协议说明中定义的消息或协议数据单元的格式,给收到的 ASP/PDU 加上规定的头部或尾部,再从相应的 PCO(控制观察点)上发送给被测系统。当被测系统有响应时,它把响应发给执行器,驱动程序首先从相应的 PCO 上接收响应的协议数据单元(或消息),之后驱动程序再按照所定义的协议数据单元(或消息)的格式,去掉收到的协议数据单元(或消息)的头部和尾部,只把协议数据单元的数据部分发送给执行器,执行器再和预期收到的内容进行匹配和比较,得到 PASS、INCONC、FAIL 的判定,这就是驱动程序的主要功能。

执行器和驱动程序进行通信时,要交换两类消息,一类是同步消息,用来通知对方做一些执行测试例的准备和善后工作,另一类是 PCO 消息,即执行器和被测实现要通信的协议数据单元及其数据部分。驱动程序收到执行器发来的同步消息之后,进行处理,处理完之后,发同步消息给执行器,所以同步消息是执行器和驱动程序用来协商的,与被测系统无关。而 PCO 消息是执行器和被测实现进行通信的主要数据内容。

图5 驱动程序的数据通信

4. 测试执行系统的特点以及一些关键技术

基于上述设计思想实现的测试执行系统有以下的优点:

1)易移植,易扩展。从实现的角度考虑,测试执行系统的运行环境为 Linux 系统,核心程序采用模块化方式进行编程。为了便于将系统移植到其他 UNIX 系统下,核心程序全部采用标准的 UNIX 系统调用。因此测试执行系统本身的运行环境不依赖特定的硬件和软件系统,具有一定的可移植性和可扩展性。

2)通用性好。从功能上看,该测试执行系统能对较广泛的网络协议进行一致性的测试,对不同的协议进行测试时,无需修改测试执行系统,只需在驱动程序中添加被测协议的协议数据单元的格式定义。因此在测试过程中,该测试执行系统对被测系统没有过多的限制。

3)界面友好,易使用。从应用的角度看,只要提供实际测试环境和使用 TTCN 书写的 MP 格式的测试套,就可以完成测试例的自动执行,给出 PASS、INCONCLUSIVE 或 FAIL 的判定结果,用户界面给出最终的测试报告,易于分析。该测试执行系统的用户界面见图6。

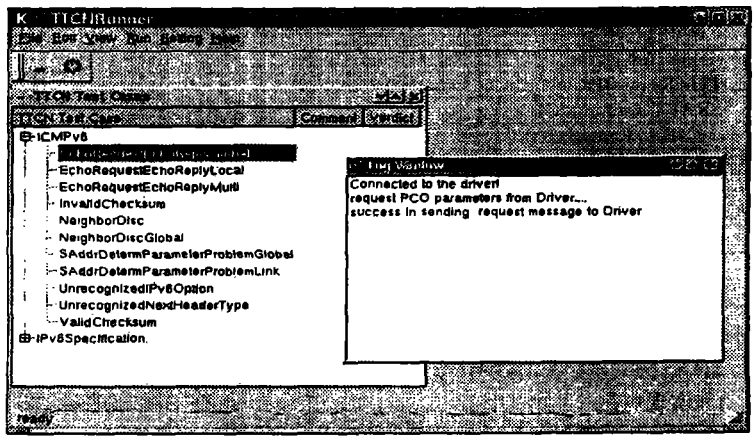

图6 测试执行系统的用户界面

4)用分布式测试方法,支持并行测试。该测试执行系统可以实现并行测试,驱动程序和其他的部件可放在同一台主机上,也可放在不同的主机上。当在不同的主机上有多个驱动程序时,就可以进行并行测试,解释器和执行器所在的主机是主测试器,其它有驱动程序的主机则是并行测试器。在进行并行测试时,并行测试器把被测试实现的响应发给主测试器,主测试器中的执行器得到最终的测试判决。

5)易与其它工具接口。使用 TTCN 描述测试集,测试集会变得比较庞大和繁琐,因此在国际上出现了许多其它描述测试集的语言。但我们还是使用基于 TTCN 的测试集,这是由于 TTCN 是一种国际通用的标准描述语言,在国际上有许多基于 TTCN 的相关软件对测试用例进行自动生成和分析,

因而我们的测试执行系统易于与这些软件接口，易于测试平台的整体开发。

6)硬件条件易于实现。基于 Linux 环境的测试执行系统在一台 PC 机上就可以使用，因此，从硬件上讲，只要有被测系统(或被测实现)和若干台单机就可以搭建测试环境，从而进行一致性测试。

另外，在具体的实现过程中，我们使用了一些关键性的技术，如解释器和执行器的并行执行和执行器和驱动程序之间的消息机制。

我们的测试执行系统采用的是解释执行的方法，因此为了加快解释执行的效率，我们采用多线程技术来实现解释器和执行器的并行执行，解释器和执行器分别作为两个单独的线程。解释器不断地从用户界面得到下一个要执行的测试例，将其转换成对应的求值树，存放在共享内存中。与此同时，执行器则从共享内存中取出一个 TTCN 求值树，进行执行，得到测试判决。这种并行工作方式可以明显地提高解释执行的效率，弥补了解释执行方式的缺点。

测试执行系统之间，我们使用了消息机制。因为在实际的测试工作中，测试环境中一些地址，如测试器的 IP 地址等都不是确定的，因而我们采用消息机制。这样每次硬件地址发生改变时，不用去改变驱动程序，只需在用户界面中改变一下配置，用户界面会把相应的配置以消息的形式发给驱动程序，驱动程序和被测实现进行通讯。这样，在测试不同协议时，我们不用改变测试系统，增加了系统的通用性和扩充性。

5. IPv6协议的测试过程

使用该测试执行系统，我们对 IPv6协议进行了部分的一致性测试，所书写的测试例都得到预期的测试结果。

测试 IPv6协议的测试环境如图7，其中 Tester 和 IUT 两台主机都支持 IPv6，两台 IPv6主机使用 IPv6数据包进行通讯。在 Tester 这台主机上装有测试执行系统和驱动程序的软件，从 Tester 向 IUT 发送 IPv6的数据包，观察从 IUT 收到的数据包是否同协议说明中预期的一样。

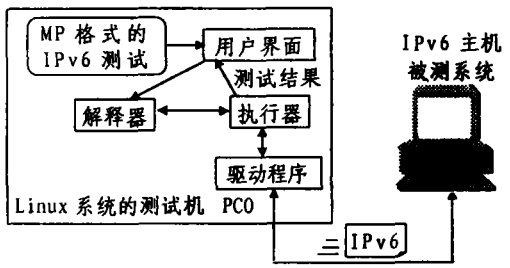

图7 IPv6协议的测试环境

下面以我们常用的 ping 命令为例，来测试测试主机向被测主机发送 ping 命令后，双方所进行的通讯过程。按照 RFC2463中的协议说明[8]，通讯过程如下：

1. 测试主机先向组播地址发出一个多播数据包，表示邻居请求。

2. 在同一个组播地址下的被测主机收到后，会向测试主机发出邻居宣告的数据包。这时测试主机就会知道被测主机的 MAC 地址了。

3. 测试主机再向被测主机发送回显请求。

4. 被测主机向测试主机发送回显响应。

用 TTCN. GR 格式描述这个测试例，来测试测试主机向被测系统发送 ping 命令的过程，其动态行为部分如下：

表1 GR 格式的测试例动态行为描述

Dynamic behavior					
Test Case Name		EchoRequestEchoReply			
Group		ICMPv6/			
Purpose		Test a process for ping command			
Nr	Label	Behavior Description	Constraints Ref	Verdict	Comments
1		L!NeighborSolPacket	NeighborSolPacket01		
2		START wait_timer 1			
3		L?NeighborAdvPacket	NeighborAdvPacket01		
4		L!EchoRequestPacket	EchoRequestPacket01		
5		SRART wait_timer2			
6		L?EchoReplyPacket	EchoReplyPacket01	PASS	
7		L?OTHERWISE		FAIL	
8		?TIMEOUT		FAIL	
9		L?OTHERWISE		FAIL	
10		?TIMEOUT		FAIL	

将以上 GR 格式的测试例使用 TTCN 编辑器或使用 Telelogic Tau 工具转换成 MP 格式后，在以上提出的 IPv6协议的测试环境中执行，会得到 PASS 的测试结果，表明这个测试例所测试的内容在协议说明和实现中是一致的。

讨论 本文以上几部分就协议一致性测试执行系统的设计和实现进行了详细的讨论，同国内外同类产品相比，我们的测试执行系统具有通用性好、易移植、易扩充等良好特性，同时采用分布式测试方法，支持并行测试。在实践中，我们进行了一系列的协议一致性测试工作，并且都取得预期的测试结果。但是从发展的角度看，该系统还有一些方面需要进一步完善。

参 考 文 献

1 石晶林，丁炜等编，中国通讯学会主编. MPLS 宽带网络互连技术. 人民邮电出版社，2001

2 Deering S, Hinden R. Internet Protocol, Version 6 (IPv6) Specification. Dec. 1998

3 Kim T H, et al. Automatic test case generation of real protocols: Framework and Methodology, FORTE XI/PSTV XV Ⅱ'98, p127～140

4 Avresky D R. Formal verification and testing of protocol. Computer Communications, 1999, 22: 681～690

5 Bi Jun, Wu Jianping. Application of a TTCN based conformance test environment on the Internet email protocol, Testing of Communication System, published by Chapman & Hall, 1997IFIP, p325～329

6 Tian Jun, Wu Qi, Li Zhongcheng. The next generation Internet protocol IPv6 and its testing. The sixth computer sciences and technology conference, China. July, 2000. 288～295

7 郝瑞兵，吴建平，史美林. 一种形式化的基于 TTCN 的测试执行方法. 软件学报，1997, 8(5): 367～374

8 Conta A, Deering S. Internet Control Message Protocol for the Internet Protocol Version 6 (IPv6) Specification, Dec. 1998. 1～15