基于 XML 的协议一致性测试系统的设计与实现*)

李 华 叶新铭 曾 敏 丁雪莲
(内蒙古大学计算机学院 呼和浩特 010021)

摘 要 本文详细介绍了一个协议一致性测试系统的实现,包括测试系统结构介绍、测试套编辑子系统和测试执行子系统的实现,同时实现了一个日志窗口,用于显示测试执行的过程。在测试套编辑子系统的实现部分,分析了编辑流程,设计的编辑子系统可以分为协议无关和协议有关部分,与协议无关部分的实现可以重用。文中还定义了存放测试套的 XML 文件的结构以及相应的元素与标记。以邻居发现协议的路由请求包为例,展示了测试套的结构及日志窗口。

关键词 协议,一致性测试,XML

Design and Implementation of a System for Conformance Testing of Protocol Based on XML

LI Hua YE Xing-Ming ZENG Min DING Xue-Lian
(College of Computer Science and Technology, Inner Mongolia University, City Huhhot 010021)

Abstract A protocol conformance testing system, including an instruction of the structure of the test system, an edit subsystem and an executing subsystem of test suite in many details, is designed and implemented. A log window which displays the procedure of execution of test is implemented simultaneously. Moreover the workflow of edit of test suite is analyzed. The designed subsystem is divided into two parts, protocol dependent and protocol independent, so that the protocol independent part is easy to reuse. The structure of XML(extensible Markup Language) file, related elements and tags are defined. A router solicitation packet of neighbor discovery protocol is used as an example to show the structure of test suite and log window.

Keywords Protocol, Conformance testing, XML

1 引言

协议一致性测试是保证协议实现产品达到设计需求的最关键的测试。协议一致性测试的方法是通过在测试器上执行设计好的测试套与被测进行交互,并将被测输出的数据与预定的结果进行比较,从而得到测试结果判定。生成测试套的方法大多没有改变,于是关于测试套的编辑表示、存储以及重用就成了协议测试研究的一个急需解决的问题,这也是目前协议测试的一个研究热点。

2 与相关工作的比较

目前已有的测试系统有清华大学的 IRIT(IP Routing Information Tester)[1]、Telelogic 的 Tau4. 0[2]和中科院在协议测试方面的相关研究[3]。其中清华大学的与 IRIP 相关的 RIPTS(Routing Information Processing Test Script)测试符合体系,给出了一套自己的测试套、测试例、测试步以及测试事件的定义方法和表示格式,并且基于 Linux 用 C 语言实现了该系统,它的特点是造价低,测试系统可以在任何 PC 机上架构。而我们的系统采用 XML[4]表示测试套,符合国际标准,易于测试套的长久保存以及重用。而且,我们的系统对于测试器的要求也较低,只要机器可以运行 Windows 即可。Tau4. 0 是 Telelogic 公司的产品,其中包含国际标准 TTCN-2 (The Tree and Tabular Combined Notation)的测试套的编辑、编译与执行。与我们的系统相比,Tau 4. 0 庞大复杂,价格昂贵,其中的 TTCN 关于测试套的设计与填写过程较难掌握,对于它的熟练过程需要较长时间,所以培训一个合格的测试人员需要很长的时间。而我们的系统可以简化测试例的构造过程,不必编译,培训测试人员的时间较短,并且对于测试人员的背景知识要求较低,只要会组协议包即可,整个测试过程造价较低。与中科院的研究工作相比,虽然双方都用树型结构表示测试例,但我们采用 XML 实现了测试套的保存,结构简单,适用面更宽,而且对于测试器的要求也较低。

3 测试系统的系统结构

一个测试系统的主要功能是编辑和执行测试套,即根据测试套内的测试例和测试步执行测试。一般的协议一致性测试系统结构如图 1 所示。

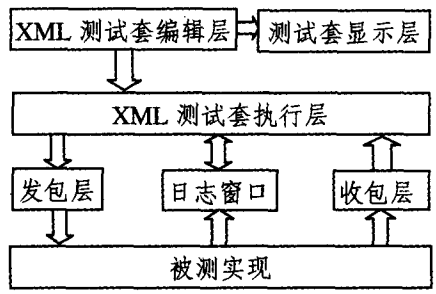

图 1 测试系统结构

协议一致性测试系统中主要部分由测试套编辑子系统、测试例执行子系统、日志窗口发包层、收包层以及测试套显示层组成。各个部分功能分述如下:

· 测试套编辑子系统主要用来新建测试套、编辑测试套,对测试套的 XML 文件进行解析以及把程序中测试套信息写入 XML 文件。

*)国家自然基金项目(60263002)、内蒙古科技攻关项目(2002061002)、内蒙古自然科学基金项目(200308020213)。李 华 副教授,硕士生导师,研究方向为分布式系统与网络计算;叶新铭 教授,博士生导师,主要从事计算机网络和分布式系统方面的研究;曾 敏 硕士,主要研究方向为协议测试;丁雪莲 硕士研究生,主要研究方向为协议测试。

·测试例执行子系统用于执行测试例，对被测协议实现进行一致性测试。

·日志窗口主要用来显示测试的执行过程以及测试结果。

·发包层用来将组织好的测试包发送出去。

·收包层负责接收被测发来的数据包，根据事先的约定处理收到的包。

·测试套显示层用于将收到的数据包进行显示，可用于测试人员的人工分析。

下面详细介绍测试套编辑、测试套执行以及日志窗口这3部分的实现。

4 测试套编辑子系统的实现

测试套的编辑子系统是直接面对测试人员的，它的实现直接关系到测试系统的通用性、工作效率以及测试人员的培训周期以及费用。我们分析了测试套编辑子系统的功能、工作流程、测试套的结构以及重用要求之后，选定用XML文件存储测试套，定义了相应的标记，并实现了该子系统。

4.1 测试套编辑子系统的工作流程

测试套编辑子系统的工作流程为：首先新建测试套或者是打开现有的保存测试套的XML文件。如果是打开现有的XML文件，编辑系统将分析XML文件中的测试套信息，将测试套的信息显示在主界面上，然后在界面上编辑测试例(例如添加或删除测试例)，最后是编辑测试步，在测试步中添加发送和预接收的数据包等信息。测试套编辑子系统的流程图如图2所示。

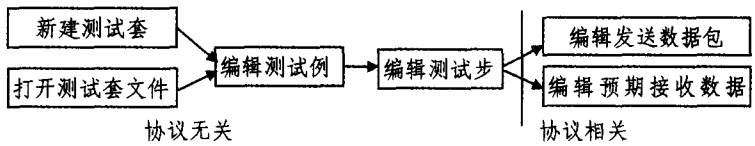

图2 测试套编辑子系统的工程流程

测试套编辑子系统前三步是与协议无关的，第四步编辑数据包针对不同的协议，相应的数据包是不同的，具体内容可以根据协议的RFC文档规定来确定。

4.2 测试套的保存

为了使测试套便于组织和管理，XML文件中测试套的组织采用了层次结构。根据XML可以自定义标记的规定，我们定义了在XML文件中需要用到的元素、标记及其表示的含义。

4.2.1 测试套在XML文件中的结构

在分析了协议一致性测试套的构成之后，用XML描述的测试套的结构也定为层次结构，共分为4个层次：测试套、测试组、测试例、测试步，如图3所示。

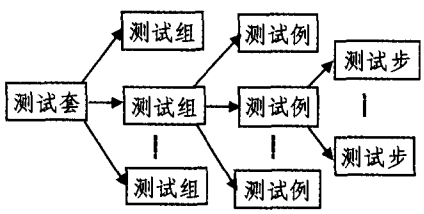

图3 XML文件中测试套的组织结构

·测试套：对应于一个协议或一簇协议，如IPv6基本描述协议和邻居发现协议[5]等。本文使用的是邻居发现协议的测试套。

·测试组：对应于此协议的一个测试目标。一个测试套中可以包含多个测试组。

·测试例：对应于一个标准协议的某一项功能描述，如邻居发现协议中关于目的节点是否可达、路由器是否可以对正确的路由器请求消息做出正确的反应等等。一个测试组由一个以上的测试例组成。

·测试步：一个测试过程的完成需要进行初始化、发包、收包等等，每一个动作就是一个测试步。测试步是测试集中最小的单位。一个测试例包含一个以上的测试步。

4.2.2 用于描述测试套的XML标记

XML文档的基本组成部分是元素，XML的内容包含在唯一的根元素中，这个根元素中通常还会按照层次结构关系定义一些其它元素，这些元素也可以再包含各自的子元素，从而形成整个XML文档的树状层次结构。XML的元素是由标记和标记中包含的内容组成的。在XML语言中，使用者可以定义标记和元素。共有3种合法标记：开始标记、结束标记、空元素标记。

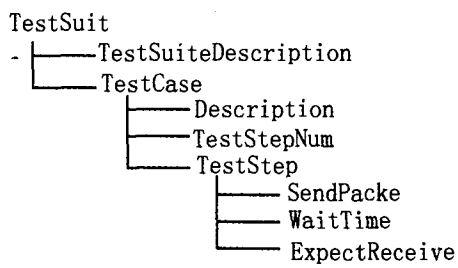

图4 XML文件中测试套元素的层次关系树

表1 9个元素以及标记

元素名字	含义
〈TestSuite〉 〈/TestSuite〉	测试套元素TestSuite的开始与结束标记
〈TestSuiteDescription〉 〈/TestSuiteDescription〉	测试套描述元素TestSuiteDescription的开始与结束标记
〈TestCase〉 〈/TestCase〉	测试例元素TestCase的开始与结束标记
〈Description〉 〈/Description〉	测试例描述元素Description的开始与结束标记
〈TestStep Num〉 〈/TestStep Num〉	测试例中包含测试步个数的元素TestStepNum的开始与结束标记
〈TestStep〉 〈/TestStep〉	测试步元素TestStep的开始与结束标记
〈Send Packet〉 〈/Send Packet〉	测试步中要发送的数据包元素Send-Packet的开始与结束标记
〈WaitTime〉 〈/WaitTime〉	发送数据包后接收响应需要等待的时间元素WaitTime的开始与结束标记
〈Expected Receive〉 〈/ Expected Receive〉	该测试步成功时应接收到的数据包元素ExpectReceive的开始与结束标记

本系统采用XML语言描述测试例，由使用者自己定义标记以及元素来描述测试套所包含的各项具体内容的含义。

为简单起见，我们没有给出测试组的表示，它的表示类似于测试例。

下面列出了 XML 文件中定义的 9 个元素及其开始标记和结束标记。其中，TestSuite 为根元素，TestSuiteDescription、TestCase 这两个子元素包含在根元素的里面，3 个子元素 Description、TestStepNum、TestStep 包含在 TestCase 这个子元素中，其它 3 个子元素 SendPacket、WaitTime、ExpectReceive 包含在 TestStep 子元素中。它们构成了图 4 所示的层次结构。表 1 是 XML 文件中将要用到的 9 个元素及其标记。

以上就是保存测试套的 XML 文件中包含的全部元素和标记。下面通过一个具体的例子，来说明如何用 XML 语言来描述和保存测试套。

4.2.3 举例

图 5 是一个用 XML 语言描述和保存的测试套文件，它描述了邻居发现协议关于路由器部分的测试套。由于浏览器可以解析标准的 XML 文件，所以这里使用浏览器显示 XML 文件。从图 5 中可以看出，浏览器标出了每一个标记，并利用行首的不同长度的缩进标志出不同标记之间的包含关系。每一个标记的折叠和展开可以通过点击它前面的按钮实现。测试套 IPv6_ND_Route_TestSuite 可以用来测试邻居发现协议的路由器部分。测试套包含 4 个测试例，展开的测试例 RSValid 是用来测试路由器收到合法的路由器请求包后，在规定的时间 1 秒内，是否能做出正确的响应，即发出路由器宣告数据包。这个测试套中的其它 3 个测试例的内容被折叠起来，点击它们标记前面的按钮可以将相应内容展开。

4.3 系统中测试套的组织

用 XML 描述的测试套被保存在文件中，是文本信息。为了能执行测试例，需要在执行测试例前，从 XML 文件中读入测试例信息，把这些信息存在程序的数据结构中，然后显示在界面上。在 XML 文件中，我们把测试套组织成层次结构。在程序中，我们使用链表结构来保存测试套的信息，其链表结构如图 6 所示。

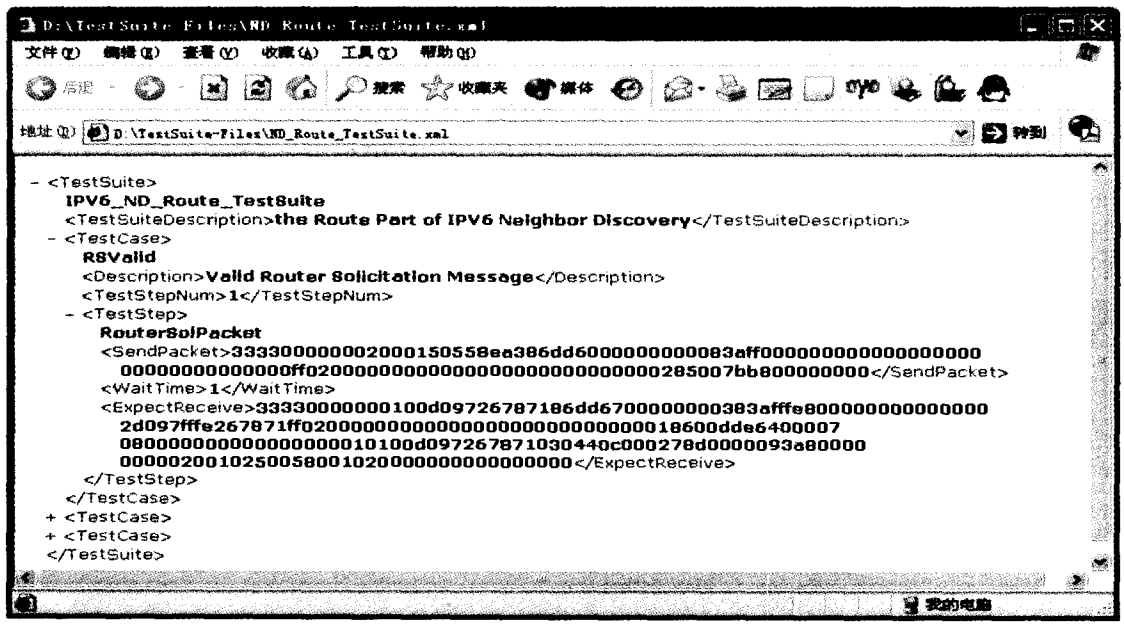

图 5 保存测试套的 XML 文件

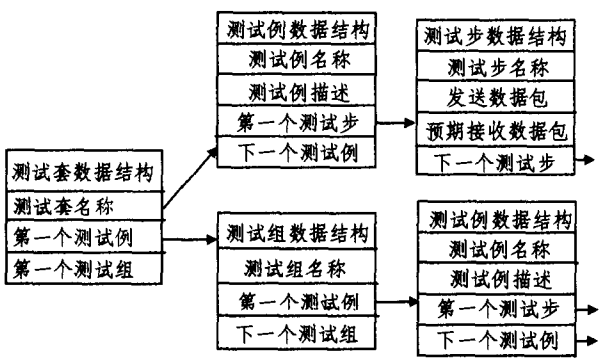

图 6 程序中保存测试套的链表结构

4.4 测试套编辑子系统的功能实现

编辑系统的功能包括：新建测试套、打开 XML 文件、添加测试例、编辑测试步、编辑数据包、保存测试套。下面详细介绍这些功能的实现。

· 打开 XML 文件

打开 XML 文件的工作流程由 3 步构成：读入 XML 文件、对 XML 文件解析，最后显示结果。在测试例编辑完成后，可以把它保存到 XML 文件中。下次使用时，打开 XML 文件，把测试套信息读入程序中，然后编辑或者执行测试例。

关于 XML 文件的解析算法如下：

```
TestSuite_tag=〈TestSuite〉;
While TestSuite_tag 〈〉 〈/TestSuite〉 do
{
Get TestSuite information;
While Pointer of TestCase 〈〉 Null do
{
Get the information of test case from test suite;
Create an object for the test case;
Fill the information into the object of test case;
While Pointer of TestStep 〈〉 Null do
{
Fetch the test step information of test case;
      Create an object for the test step;
      Fill the information into test step;
      Link this object into test step list;
Pointer of TestStep=Next Pointer of TestStep;
}
Pointer of TestCase=Next pointer of TestCase ;
}
}
```

· 保存测试套

编辑完测试套后，需要把程序中的测试套信息保存到 XML 文件中，下次可以直接使用。

算法思想如下：

1)建立一个 XML 文件;

2)写入测试套信息;

3)测试套中是否包含测试例。如果有测试例,写入测试例信息,继续第四步;如果没有,写入测试套结束标记,关闭文件;

4)测试例中是否包含有测试步。如果有测试步,写入测试步信息以及测试步中包含的发送数据包、等待时间、预接收数据包;

5)重复第四步,直到当前测试例中的所有测试步信息都被写入文件;

6)返回第三步。

5 测试例执行子系统的实现

在编辑完测试例后,为了测试协议的实现,还需要有测试工具,也就是测试例的执行系统。由于 VC++ 中封装的 Socket 类在发送数据包时与协议相关,所以我们采用了直接网络编程技术。通过使用这种技术在数据链路层发送数据包,被测协议实现接收到数据包后,做出相应的响应。由测试例执行子系统根据响应判断被测协议实现是否与协议说明相一致。在程序中可以使用线程技术来实现数据包的收发。

5.1 底层软件平台

由于 VC++ 6.0 中封装的网络接口类与协议相关,所以执行系统采用从数据链路层发送数据包的方法。为了使发包程序与底层设备无关,本测试系统使用了基于 Winpcap 的网络数据包捕获技术。它由 3 部分组成:一个数据包捕获驱动器、一个底层动态链接库(Packet. dll)、一个高层静态链接库(wpacp. lib)。这种网络数据包捕获技术可以从数据链路层发送和捕获各种数据包,而且不影响其它网络应用的 IP 数据包的收发。

5.2 测试例执行子系统的执行过程

在主界面上选中要执行的测试例后,点击执行测试例按钮,测试例执行子系统开始工作。

测试例执行子系统的执行算法如下:

```
For all test steps in the test case
{
Preparing the sending packet;
  Sending packet;
  Blocking sending process;
  Receiving the respondent packet;
  Analyze the received packet;
  Verdict the test step as pass, fail, incon. ;
  If fail break;
}
```

6 日志窗口

日志窗口记录了系统所进行的各种操作,例如记录打开的测试套文件路径和文件名、记录正在执行的测试例以及通过测试例执行子系统传送的数据、显示发送了哪些数据包和收到了哪些数据包。每一个测试步执行完毕后,都会显示这一步执行是否通过。在测试例执行完后,显示测试例是否通过。图 7 显示了发出的路由请求包以及接收到的数据包,最后给出了判定结果。

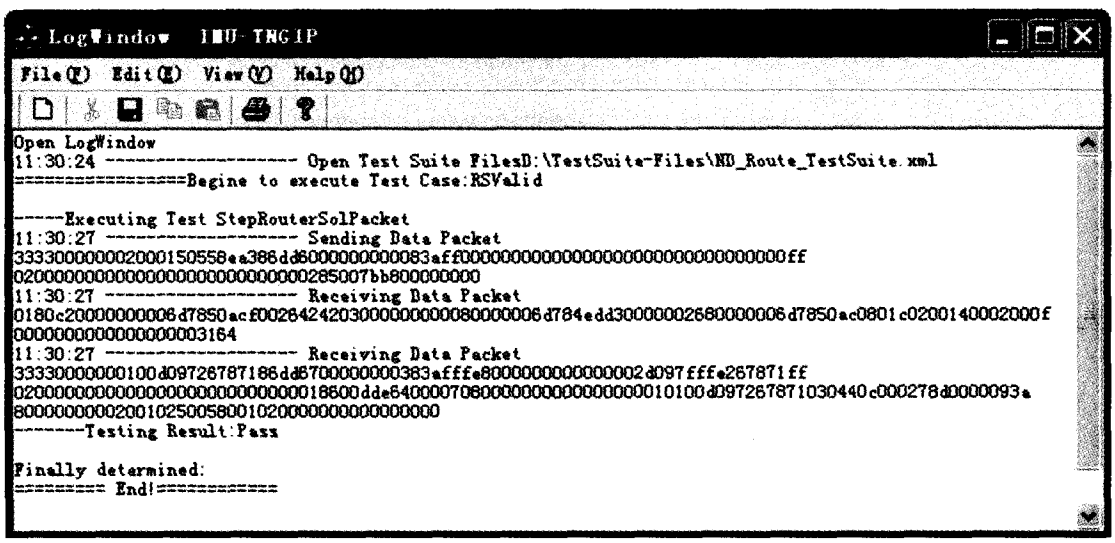

图 7 日志窗口

日志窗口每一行开始的数字表示开始执行的时间,下划线后表示执行的操作。

结论 本文介绍了一个基于 XML 的协议一致性测试系统实现,它使用 XML 来描述和保存测试套,测试系统可以使用较少的代码处理测试套文件。本系统相比于其它系统(如 Tau 4.0)要简单,易于理解与学习。另外,由于采用直接网络编程技术,测试例执行子系统在执行测试例时与协议无关,为将来其它协议的测试最大限度地实现了测试系统构件的重用,适用面更宽。由于测试系统按照列表中的测试例的位置查找它的数据结构,所以在不同目录下的测试例可以拥有相同的名字而不必担心产生混淆。

最后我们对邻居发现协议进行了测试[6]。相比支持 TTCN 的测试系统(如 Tau 4.0)而言,本系统简单易学,对于一个熟悉协议的测试人员,用于熟悉本系统的时间至少可以降低一半。我们将来的工作将集中在针对不同的协议,完善相应的测试套表示以及根据需求定制测试等方面。

参 考 文 献

1 Wu Jianping, Li Zhongjie, Yin Xia. Towards Modeling and Testing of IP Routing Protocols. Proceeding 15th IFIP International Conference, Test Com2003, Sophia Antipolis, France, May 26-28, 2003, Testing of Communicating Systems. 2003. 49~62

2 Tau. http://www.telelogic.com

3 田军,张玉军,于东. 邻居发现协议的形式化测试。计算机研究与发展,2001,38(12):1409~1417

4 Bray T, Paoli J, Sperberg-McQueen C M. Extensible markup language (XML)1.0. W3C recommendation, February 1998. http://www.w3.org/TR/REC-xml/. W. #28. Extensible markup language (XML)1.0 (second edition)- W3C recommendation, October 2000

5 RFC 2461. Neighbor Discovery for IP Version 6(IPv6). December,1998

6 叶新铭,孙美飞. IPv6 邻居发现协议的一致性测试. 计算机科学,2005,32(6):43~46