并发 TTCN 测试执行机的设计与实现*

张卫星, 蒋 凡+

(中国科学技术大学 计算机科学技术系,安徽 合肥 230026)

Design and Implementation of Test Executor for Concurrent TTCN

ZHANG Wei-Xing, JIANG Fan+

(Department of Computer Science and Technology, University of Science and Technology of China, Hefei 230026, China)

+ Corresponding author: Phn: 86-551-3601340 ext 201, E-mail: fjiang@ustc.edu.cn

http://www.ustc.edu.cn

Received 2001-09-13; Accepted 2002-04-10

Zhang WX, Jiang F. Design and implementation of test executor for concurrent TTCN. *Journal of Software*, 2003,14(3):606~611.

Abstract: The method of designing a common-used concurrent TTCN test executor is proposed in this paper. When testing an implementation of concurrent protocol, the problem of executing concurrent test cases by FIFO scheduling algorithm is solved and the PTI (packet transmitting interface) part based on the abstract I/O queue theory is proposed. The PTI part offers the test executor the independency on implementations of a given protocol. What's more, the test executor provides a visual interface to trace the executing of test cases, which makes the locating of faults easier. Banding with corresponding PTI part, the test executor can start a testing. Now it is already in use.

Key words: protocol conformance testing; concurrent TTCN; packet transmitting interface; test component; test component configuration

摘 要: 提出了一种通用并发 TTCN 测试执行机的设计方法.在测试并发协议实现时,采用 FIFO 调度算法解决了并发测试例的执行问题,并在借鉴抽象 I/O 队列思想的基础上提出了 PTI(packet transmitting interface)部分,使得执行机与特定的协议实现无关,而且提供了可视化的测试执行跟踪界面,使错误定位变得更加容易.实现的执行机在附加上相应的 PTI 部分之后就可以进行测试,目前已投入使用.

关键词: 协议一致性测试;并发 TTCN;报文传输接口(PTI);测试组件;测试组件配置

中图法分类号: TP393 **文献标识码**: A

协议一致性测试旨在检测所实现的协议实体(或系统)与协议规范的符合程度[1],根据测试结果来判断一个协议实现是否与规范一致[2,3].协议一致性测试使用的国际标准是 ISO9646(对应于 ITU-T 的 Recommendation X.290-X.296),该标准由 7 个相关的文件组成,其中 ISO9646-3(X.292)则定义了测试套(test suite)描述语言 TTCN(tree and tabular combined notation).

在 ISO9646-3(1991)中,TTCN 语言并不支持并发[4],当时的 X.25 测试套就是用这个版本的 TTCN 描述的;

☛ **第一作者简介**: 张卫星(1977－),男,安徽黄山人,工程师,主要研究领域为通信协议测试.

在新版的 ISO9646-3(1997)中,TTCN 开始支持并发,这使得对并发协议测试套的描述成为可能.一致性测试可以分成两个大的阶段:一致性测试生成和一致性测试执行.测试套作为测试生成的输出和测试执行的输入,是连接整个协议测试过程的枢纽.用形式化的方法进行测试可以使得测试结果具有可比性,因此 TTCN 这种形式化的测试套描述语言得到了广泛的应用.我们实现的并发 TTCN 测试执行机具有测试多种协议实现和自动重复测试的能力,并且提供了可视化的跟踪界面,使错误定位变得更容易.

本文主要由以下几部分组成:并发 TTCN 简介、执行机的框架设计、编译器的设计、数据/环境的设计、执行器的设计、可视化跟踪界面的设计.

1 并发 TTCN 简介

TTCN 的并发特性是通过对测试组件(test components)以及测试组件配置(test component configurations)的描述来体现的[2,3].测试组件是指在并发测试例中能与其他测试组件一起并行执行的命名子部分(named subdivision),它有固定个数(fixed number)的 PCO(point of control and observation)和固定个数的 CP(coordination point).测试组件配置是用来描述并发测试例中测试组件、PCO 以及 CP 的连接关系.

一个并发测试例是指用并发 TTCN 描述的测试例,描述的结构是:在测试例头部(header)指明使用的测试组件配置,在动态行为部分用 CREATE 语句来启动在测试组件配置中申明的 PTC(parallel test component).

设有测试步(test step),具体情况见表 1.

Table 1 The test step TestStepIncrease

表 1 测试步 TestStepIncrease

Test step name		TestStepIncrease (CPPar: CP, BaseNumber: INTEGER)			
Description					
Nr	Label	Behaviour description	Constraints Ref	Verdict	Comments
1		CPPar!CMIntType	CMInt (BaseNumber+1)	(PASS)	

表 1 中的测试步是一个有参数的测试步,其中 CPPar 是一个 CP(coordination point)类型的参数,BaseNumber 是一个 INTEGER 类型的参数,在语句中出现的 CMIntType 是 CM(coordination message)类型名,CMInt 是对应 CM 类型的 Constraint.

测试例见表 2.

Table 2 The concurrent test case TestPTC

表 2 并发测试例 TestPTC

Test case name		TestPTC			
Configuration		TestCompConfig1			
Description					
Nr	Label	Behaviour description	Constraints Ref	Verdict	Comments
1		CREATE (PTC1: TestStepIncrease (CP1, 0),PTC2: TestStepIncrease (CP2,0))			
2		CP1?CMIntType	CMInt(1)		
3		CP2?CMIntType	CMInt(1)		
4		?DONE()		PASS	
5		CP2?OTHERWISE		FAIL	
6		CP1?OTHERWISE		FAIL	

在表 2 的测试例中,第 1 条语句创建了 PTC1 和 PTC2,随后它们就并发执行,第 4 条语句用来判断系统中是否所有的 PTC(即 PTC1 和 PTC2)都终止.在这个例子中,PTC1 和 PTC2 使用的是同一个测试步,只不过传的参数不同而已.实际上,PTC 的动态行为都对应到一棵本地子树(local tree)或者一个测试步上,MTC 对应的则是一个测试例.

在我们给出的测试步和测试例中省略了表头(header)部分的一些内容,比如所在的 Group、测试目标以及使用的 Default 等信息,旨在突出并发 TTCN 的一些特殊选项.

对应的测试组件配置信息见表 3.

Table 3 The test component configuration information

表 3 测试组件配置信息

Configuration name	TestCompConfig1		
Comments			
Components used	PCOs used	CPs used	Comments
MTC1		CP1,CP2	
PTC1		CP1	
PTC2		CP2	

按照测试组件配置信息,我们可以推断出 MTC1 与 PTC1 是通过 CP1 通信的,而 MTC1 与 PTC2 是通过 CP2 通信的,如图 1 所示.

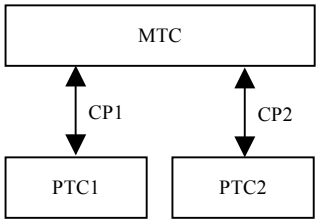

Fig.1 The configuration of test components

图 1 测试组件配置图

2 执行机的框架设计

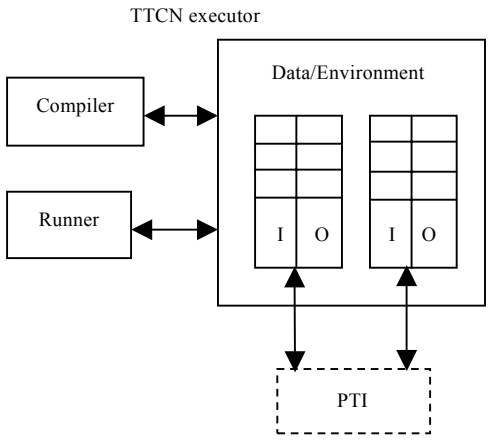

Fig.2 The structure of TTCN executor

图 2 TTCN 执行机的构成

TTCN 执行机可以分成 3 大部分:编译器、数据/环境、执行器.如图 2 所示.

图 2 中的 I/O 队列[5]归属于数据/环境部分,PTI(packet transmitting interface)则不是执行机的组成部分.

由于 TTCN 执行机的输入是一个 TTCN.MP (machine processable)格式的测试套,因此必须设计一个相应的编译器对测试套进行语法分析,并且生成代码.在这一点上我们采取了不同的处理方法,不是生成纯代码,而是生成包含代码和数据的 C++对象(object),由于我们是基于 VC 平台进行开发的,只要为每个类型设计好相应的类(class),在编译时就可以方便地将 TTCN 对象生成对应的 C++对象.

数据/环境部分用来保存编译信息和临时变量,并且负责管理动态内存分配,其中的 I/O 队列部分用来保存收发的报文、协调信息以及定时器超时的信息.PTI 虽然不是 TTCN 执行机的组成部分,但是它是一个执行时不可缺少的部分,该部分对应到一个动态链接库(DLL),正是由于该部分的分离才使得 TTCN 执行机具有通用性.

至于执行器部分,很显然,它是执行测试例并记录测试日志,在测试例执行完毕的时候作出判决的部分,除了这些必备的功能之外,我们还提供了可视化的跟踪界面和全自动的测试执行方式.

经过系统的分析之后,我们得出的流程如图 3 所示.

首先,按照用户的要求装入 TTCN.MP 格式的测试套,对其进行编译,生成 C++对象和中间结果(数据),之后提示用户进行测试套参数(SetTSPara)设置(包括 PICS 和 PIXIT 信息的输入)、静态测试选择(selection)和 PCO 设置(PCOSetting)工作,最后由用户进行自动测试的设置便开始测试执行,生成测试日志并根据测试结果给出判决.

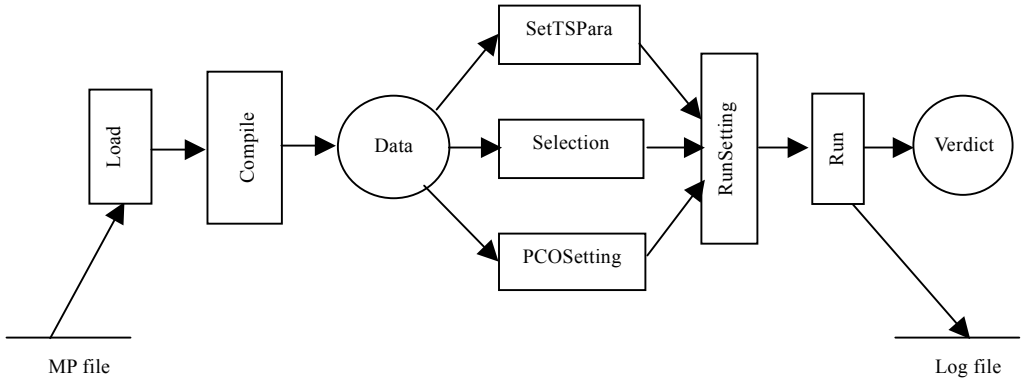

Fig.3 A simplified process of TTCN executor
图 3 TTCN 执行器的简化流程

3 编译器的设计

该部分的设计借用了 YACC 这个工具,但是仍然有许多额外的工作要进行,这是因为测试套可以引用(import)其他模块(module)中的 TTCN 对象,以及在 TypeDefByReference 里面引用在 ASN.1 Module 中定义的 ASN.1 类型.这些引用使得一次测试执行可能要涉及多个 MP 文件或者 ASN.1 Module 文件,因此在独立编译完所有的文件之后还要进行交叉匹配,以使得所有的引用都指向正确的对象.

在该部分的设计中遇到的最大困难是将 BNF 范式直接转换成产生式之后会产生不少语法冲突,而且我们的系统为了提供对 ASN.1 的支持,也将 ASN.1 的 BNF 范式转换成了产生式,这就使得语法冲突增多.然而,语法冲突只有两种类型:归约-归约冲突和移进-归约冲突.我们的解决方法是:采用公共子表达式提取的方法解决归约-归约冲突,并且在语义动作上进行语义的区别;对于移进-归约冲突,我们根据各种具体情况设置不同的优先级,这样就基本解决了语法冲突的问题.

4 数据/环境部分的设计

该部分的 I/O 队列是使得 TTCN 执行机可以通用(即与特定的协议实现无关)的基础,报文的收发不再是直接与实际的网络交互,而是通过一个抽象的 I/O 队列[1]与 PTI 部分交互,这样就使得 TTCN 执行机与低层服务无关,从而可以测试大部分的协议实现.实际上,真正完全通用的 TTCN 执行机是不存在的,因为不同的协议总是具有不同的通信方式,而我们这里是把这些不同的地方放到了 PTI 部分去实现了,根据被测的协议实现使用的低层服务,就可以利用该服务实现报文的收发工作,也就是实现 PTI 部分.

众所周知,发送的报文要经过编码后才发出去,同样,接收到的报文要经过解码后才提交给上层协议实体(或应用程序).不同的协议采用的编解码方式基本上都是不同的,因此在测试套中一般都要指明采用的编解码标准或者采用用户自定义函数来完成编解码操作,我们实现的 TTCN 执行机支持 ASN.1 的 BER,PER 编解码以及用户自定义函数的编解码方式.为了使我们的系统具有很好的可扩展性,我们支持用户自定义的编解码函数采用 DLL 的方式来实现,也支持用 DLL 实现的新的编解码标准.

在测试执行时,各种参数的传递、局部判决变量的管理、执行树状态信息的维护等工作需要同时进行.为此,我们设计了一个运行上下文类,用来完成这些工作.每个测试组件(test component)都对应一个运行上下文对象,在执行并发测试例的时候,如果有 N 个测试组件在运行,那么就会有 N 个运行上下文对象维护相关信息.

另外,该部分还要负责 PICS,PIXIT 信息的输入、测试套参数的输入、静态测试选择、PCO 配置等工作.

5 执行器的设计

执行器是整个 TTCN 执行机的核心,它负责测试例的执行,在并发测试例执行中需要重点解决的几个问题是:测试组件的设计、测试组件配置的合法性检查以及测试组件的调度.

5.1 测试组件的设计

由于测试组件是可以并发执行的,为了实现和理解上的方便,我们设计了一个测试组件类.每个测试组件都将有一个 Status(状态)属性,用来表示该测试组件是处于就绪、运行和阻塞 3 种状态中的哪一种.由于测试组件在执行的时候肯定跟某棵树对应,因此每个测试组件都有一个保存对应树的指针,除此之外,每个测试组件还有一些其他的属性,如:运行上下文、PCO 列表、CP 列表等.但是最重要的一点是,每个测试组件都提供了一个执行该测试组件的方法,考虑到在程序内部调度仿真并发的特殊性,该方法每次只执行一层(level),并根据执行结果改变测试组件的 Status,使得测试组件的状态从运行变为就绪或者阻塞.

在测试组件的设计中,另外一个问题是各个测试组件如何有效而快速地通信,对于 PTC 分布在远程系统的情况,实现上有一定的困难,因此我们设计的执行机只能支持 PTC 和 MTC 同在一个系统的测试方法,即只能支持本地测试(local test method)和远程测试(remote test method)两种测试方法[6].为此,我们采用了本地缓冲区的通信方式,它可以很好地解决测试组件之间的通信问题,这是因为测试组件的调度是在程序内部进行的,因而不需要信号量来解决同步问题.

5.2 测试组件配置的合法性检查

并发测试例是指那些在执行中动态启动 PTC 的测试例,一般用 CREATE TestCompName(Par1,Par2,…)的形式启动,其中 TestCompName 是测试组件的名字,Par1,Par2 是启动该 PTC 的实参,对于并发的测试例,必须在测试例 Header 部分指出该测试例使用的测试组件配置(test component configuration)信息,该信息包含了各 PTC 间(或者 PTC 与 MTC 间)使用哪个 CP 进行通信以及 PTC,MTC 使用哪个 PCO 与 IUT 进行通信.在实际的设计中,我们在全部文件编译完成之后对测试组件配置的合法性进行检查,比如 CP 的两端是否连接正确;PCO 是否被正确使用等.

5.3 测试组件的FIFO调度算法

系统中维护着就绪测试组件和阻塞测试组件两个队列,由于在程序内部没有使用多线程技术,所以每执行 K 个就绪测试组件就去处理报文和 Windows 消息以及定时器消息,这里,K 的选取比较重要,不能太大,也不能太小,我们实际应用时的数值是 10.该调度算法的伪代码描述如下:

```
#define MAX_TCOMP 10   //就是上面提到的 K
while (IsMTCDone==FALSE)
{
  ReadyTCompNumber=ReadyTCompQueue.GetSize(); //计算就绪队列中的测试组件个数
  I=0;
  while ((I<=MAX_TCOMP_ONE_TIME) && (ReadyTCompNumber>=1))
  {
    m_CurrentComponent=m_ReadyTCompQueue[0]; //从队头取出一个就绪测试组件
    m_ReadyTCompQueue.RemoveAt(0); //将该就绪测试组件从就绪队列中删除
    m_CurrentComponent->EvaluateTestComp(); //执行该就绪测试组件(执行之后就绪队列可能会发生变化)
    ReadyTCompNumber=m_ReadyTCompQueue.GetSize(); //重新计算就绪测试组件个数
    ProcessWindowsMessage(); //处理 Windows 消息
    I++;
  }
  ProcessPacketAndTimer(); //处理网络报文和定时器
```

```
    ProcessWindowsMessage(); //处理 Windows 消息
}
```

由于测试组件的运行可能会使就绪测试组件队列发生变化,比如 PTC1 通过 CP1 发送了一个 CM 给 PTC2,因此原来阻塞的 PTC2 的状态应该发生变化,要由阻塞变为就绪,这就是在执行中重新计算就绪测试组件个数的原因.

6 可视化跟踪界面的设计

为了在测试发现错误的时候能够很好地找出错误位置,我们设计了一个可视化的跟踪界面.它有两种工作方式:自动切换树和跟踪特定树.每个跟踪窗口只跟踪一个测试组件,这是因为测试组件是并发执行的.由于 Attach 语句的存在使得一个测试组件对应的执行树在展开(expand)之后可能变得很大,不利于跟踪,因此我们将跟踪界面局限在某棵树上.在自动切换树的方式下,如果树 A 有一条语句 Attach B 得到了执行,则跟踪界面由原来的只显示树 A 变成只显示树 B.如果是跟踪特定树的方式则不由树 A 切换到树 B.

在实际实现的时候,我们使用 MFC 的 Grid 控件来进行可视化跟踪的显示输出,并用特定的颜色(红色)突出显示当前正在执行的语句.

7 结束语

经过一年多的努力,我们完整地实现了并发 TTCN 执行机,并已经投入到实际的应用中,受到了用户的好评.但是前面已经指出,该并发 TTCN 执行机中所有的测试组件都只能在同一个系统中并发执行,这是因为 PTC 分布在远程系统上会使测试组件之间的协调变得困难.因此,进一步的工作可以在解决分布式测试中的协调问题上展开.

References:

[1] Gong ZH. The Protocol Engineering of Computer Network. Changsha: National University of Defence Technology Press, 1993. 140~164 (in Chinese).

[2] ITU-T. OSI conformance testing methodology and framework for protocol recommendations for ITU-T applications——the tree and tabular combined notation (TTCN). Recommendation X.292, 1998.

[3] ISO/IEC. OSI conformance testing methodology and framework part 3: the tree and tabular combined notation (TTCN). ISO9646-3, 1997.

[4] Probert RL, Monkewich O. TTCN: the international notation for specifying tests of communications systems. Computer Networks and ISDN Systems, 1992,2:417~436.

[5] Hao RB, Wu JP, Toward formal TTCN-based test execution. In: Proceedings of the 16th IEEE Annual Conference on Computer Communications. INFOCOM, Part 1. 1997. 230~235.

[6] ITU-T. OSI conformance testing methodology and framework for protocol recommendations for ITU-T applications——abstract test suite specification. Recommendation X.291, 1998.

附中文参考文献:

[1] 龚正虎.计算机网络协议工程.长沙:国防科学技术大学出版社,1993.140~164.