文章编号:1001 - 9081(2005)01 - 0192 - 04

分布式网络性能测试系统的设计与实现

蒋 涛,张 彬,王世普
(云南大学 软件学院,云南 昆明 650091)
(jiangtao_guido@yahoo.com.cn)

摘 要:网络性能测试是网络测试中的一个重要部分。介绍了一种分布式测试和集中控制相结合的网络性能测试模型,并对该模型的测试目标、测试方法、体系结构及实现机制进行了深入的分析和讨论,在此基础上进行了系统的设计和实现。实验结果表明,系统具有良好的可扩展性,适合对大规模的网络进行性能测试。

关键词:分布式测试模型;网络性能;SNMP;Console;Agent
中图分类号: TP393 **文献标识码**:A

Design and implementation of distributed network performance testing system

JIANG Tao, ZHANG Bin, WANG Shi-pu
(Software Institute, Yunnan University, Kunming Yunnan 650091, China)

Abstract: Network Performance Testing is an important component in network testing. A distributed test model integrated with centralized control was introduced. The test goal, test method, architecture and working mechanism for the model were discussed thoroughly and the system design and implementation were developed based on them. The final experiment results show that the system is scalable and is suitable for network performance measurement.

Key words: distributed testing model; network performance; SNMP; Console; Agent

0 引言

随着网络新技术的不断出现、网络新业务的不断开展,对网络的性能要求越来越高。同时,随着网络规模的进一步扩大,保持网络高性能运转变得越来越困难。因此,对网络性能的测试和评价越来越受到人们的重视。然而,网络性能测试是一项非常复杂的任务。网络性能测试软件的开发在我国还处于起步阶段,国外开发的非常成熟的网络性能测试软件也不多。很多时候网络性能测试常常用一些简单的网络测试命令,如 ping、tracert/traceroute 等,进行基本的通断测试,而现在一般的网络性能测试软件基本上也是基于传统的网络测试模型:集中式测试模型或分层式测试模型。对于大型网络系统,这种测试都很难达到令人满意的效果。

没有良好的网络性能测试系统,就无法对网络性能进行深入的测试,也无法依据测试获得的网络性能指标定量描述网络性能。网络性能测试是探讨网络行为和运行规律的基础、网络管理的前奏和基础、网络服务质量(QoS)的验证和控制的基础。因此,为了更好地提高网络性能测试的效率、准确性,有必要设计一个能满足网络测试要求的系统。本文提出了一种分布式测试和集中控制相结合的网络性能测试模型,并对一个实际的网络性能测试系统进行了设计与实现。

1 传统测试模型的局限性

传统的测试模型主要是指集中式测试模型和分层式测试模型,这两种测试模型与分布式模型相比存在明显的缺陷。

(1) 在集中式测试模型中:由一个中心结点负责与其他各个分支结点的测试,并由中心结点采集各分支结点的性能测试数据。中心结点承担了大量的处理工作,因而要求它具有很好的性能,如果它的性能不能满足测试要求,那么它将成为测试的瓶颈。另一方面,由于中心结点与各分支结点存在大量的数据交换,必然增加网络的负载,并且可能使得由测试数据流导致的网络负载极不均衡。

(2) 在分层式测试模型中:网络被分成若干域,每个域中有一中心结点,域中的测试模型类似于集中式测试模型,整个网络中有一个域的管理结点即超级中心结点。这种测试模型在一定程度上提高了网络的测试性能,但是它只不过是集中式测试模型的变形,因而它的测试效率不可能有根本性的提高。随着网络规模的不断扩大和设备能力的不断提高,它仍不能满足当前网络高速发展的需要。

2 分布式网络测试系统设计

本文分布式网络测试系统的性能测试指标主要有:端到端(End-to-End)的吞吐量、响应时间、丢包率、可达性、网络设备参数等。网络性能测试指标的定义符合 RFC1242。

在该系统中采用了主动测试方法:即在网络的一端向测试网络发送特定类型的数据包或网络应用,并从测试网络输出端接收响应的信息流,然后对接收内容进行分析,或主动地去获取网络设备中的性能参数信息。

2.1 系统体系结构

系统体系结构如图 1 所示。

收稿日期:2004 - 07 - 12;**修订日期**:2004 - 10 - 10

作者简介:蒋涛(1973 -),男,湖南衡阳人,硕士,主要研究方向:网络安全; 张彬(1978 -),女,江苏徐州人,硕士,主要研究方向:网络安全; 王世普(1958 -),男,云南人,教授,主要研究方向:计算机网络与通信、网络安全.

系统采用分布式测试和集中控制的体系结构和 Client/Server 工作方式,主控端称为 Console,被控端称为 Agent。在一个网络系统内可以部署一个或多个 Console,每一个测试结点部署一个 Agent,运行 Agent 的操作系统可为异构系统。Console 负责测试脚本的配置、测试结点的部署、测试指令发送、测试任务的管理、测试数据分析、测试数据显示以及网络性能评价。Agent 按照 Console 的指令要求,产生特定强度、目的地及内容的测试数据包,并收集来自其他节点的通信流,或采集支持 SNMP 的网络设备 MIB 性能参数。

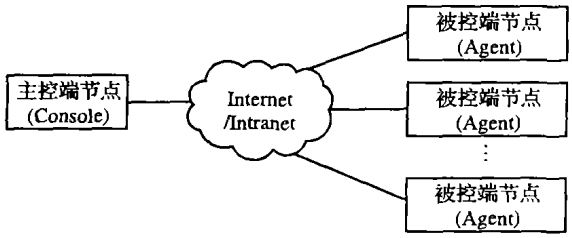

图 1　系统体系结构图

本系统的体系结构试图避免集中式、层次式模型的缺点,把测试工作和测试流量分布到了测试网络的各个节点,使得测试的网络流量比较均衡地分布在网络的各个分支上,不但测试数据的准确度提高,而且灵活、方便、易于扩展。

2.2　系统测试过程

系统测试过程如图 2 所示。

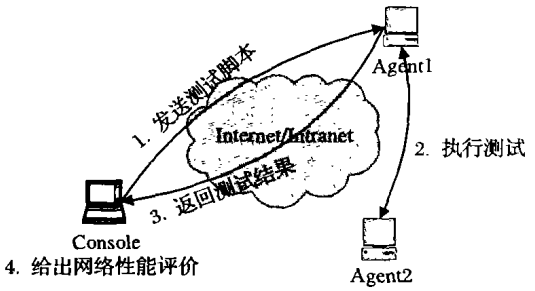

图 2　系统测试过程示意图

Console 把测试结点参数、系统配置信息和定义的测试信息转换成测试脚本指令并发送给测试网络部署的被控端 Agent1。然后由 Agent1 解析测试脚本指令,之后它根据解析的测试类型向 Agent2 发送测试包,执行要求的测试。Agent1 和 Agent2 完成测试过程之后,Agent1 将测试结果数据发回给 Console。最后 Console 将对测试数据进行分析并对整个的网络整体性能进行评价,给出相应的修改建议或方案。

上面的测试模型属于一对一的测试映射,为最简单的网络测试拓扑结构。在实际测试过程中,网络测试拓扑结构为一对一、一对多及多对一测试映射的混合体,都可由此演化而来。对于大型的、非常复杂的网络拓扑结构,可以在测试网络中部署多个 Console,每个 Console 为一个独立的实体,单独管理测试网络的某个区域,就像 Internet 网络中的自治区域一样。

2.3　消息代码定义

Console 向 Agent 下达的测试指令使用自定义的消息,其中消息代码(Message Code,MC)。MC 格式定义如下:

表 1　MC 结构

消息头 (2 字节)	长度 (1 字节)	测试对地址 (8 字节)	测试对端口 (4 字节)	测试参数 (24 字节)	校验 (1 字节)	消息尾 (1 字节)

测试对(test pair)是指两个相互进行测试的 Agent。下文中关于测试对的引用不再说明。

消息长共 41 字节。其中:

消息头:2 字节,F4H,F5H;

长度:1 字节,从测试对地址到校验的字节数(包括校验位),目前固定为 25H;

测试对地址:8 字节,前 4 字节为源 IP,即发起方 IP,后 4 字节为目的 IP,即接收方 IP;

测试对端口:4 字节,前 2 字节为源端口,后 2 字节为目的端口;

测试参数:24 字节,从前往后依次定义为:

1) 测试脚本类型:1 字节,01H-吞吐量,02H-响应时间,03H-丢包率等;

2) 测试开始时间:8 字节,表示从 1970:00:00:00 到现在的毫秒数;

3) 数据包尺寸:2 字节,例如 IP 数据包一般不超过 576 字节;

4) 数据包数量:1 字节,例如 50 表示为 32H;

5) 测试重复次数:1 字节,例如 100 表示为 64H;

6) 测试文件字节数:3 字节,范围为 1k ~ 1024k;

7) 保留字节:8 个字节。

校验:1 字节,从长度到参数所有数据的异或运算,得到的值作为校验字节;

消息尾:1 字节,FBH。

3　分布式网络测试软件设计

为了使该系统具有跨平台的特性,Console 和 Agent 分别使用 VC. NET 和 Java 开发语言;为了在异构平台中交换数据,我们采用 XML 统一数据格式;为了系统测试的方便性,Console 同时被配置成 Web 服务器,Agent 代码可以从该服务器下载,当需要对测试网络中的某一个结点进行测试时,只需在该结点上下载并安装就可以了,并且以后 Agent 作为一个守护进程随操作系统启动而启动,停止而停止。

3.1　系统结构

整个系统主要由 5 个子系统组成:测试控制子系统、网络评价子系统、数据子系统、网络测试子系统和设备数据采集子系统。系统结构如图 3 所示。

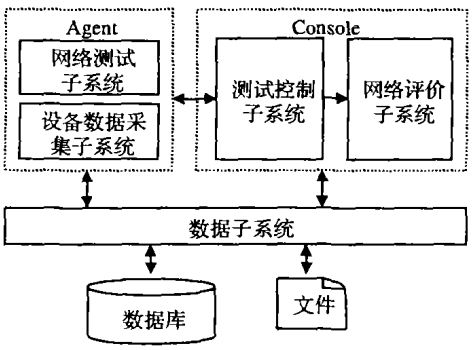

图 3　系统结构图

Agent 的网络测试子系统负责网络性能参数的测试,而设备数据采集子系统负责网络设备性能参数的采集。Console 测试控制子系统负责测试参数、测试脚本、网络测试任务的管

理,同时负责与 Agent 的交互(如测试指令的发送)以及数据子系统的交互;网络评价子系统负责测试的分析、显示和性能评价。数据子系统是其他子系统的数据接口,其中数据库采用 SQL Server 7.0,也可选其他数据库系统,而数据文件可采用文本文件格式。

3.2 数据子系统

数据子系统是系统的数据接口,它管理系统中各种数据的输入/输出。系统中存在三种数据库:① 配置数据库,用于存放测试所需要的各种测试配置信息,如测试结点、测试间隔时间、是否需要详细路径信息等;② 测试数据库,用于存放测试数据;③分析数据库,用于存放分析后的统计信息。

数据子系统与数据库的接口在 Console 中采用 ODBC 而在 Agent 中使用 JDBC。不过须注意的是:在 Java 中访问 SQL Server 还须安装驱动程序 SqlServer2000 for JDBC。Console 使用 ODBC 中的 Recordset、CRecordView 等类来实现数据的查询(Select)、插入(Insert)、更新(Update);Agent 利用了 Java 的 rmi(远程方法调用)技术,定义了从 Remote 类派生的远程接口类(AgentDataInterface)及其实现类(AgentDataInterfaceImpl),其方法被 Agent 中的子系统所调用。数据库表单的组织和设计依据网络设备进行,一个网络设备对应一个数据库。而在该设备数据库中表单的设计又按时间进行分类,如按小时进行分类。数据文件格式设计如下:固定每一行的字节数(包括回车、换行)以及行中的字段顺序和字段的字节数,如该字段无值则填空格(ASCII 码为 20H),字段不同的数据类型采用不同的对齐方式。

3.3 测试控制子系统

测试控制子系统是整个测试的控制中心,它根据用户预先设定的有关参数及测试脚本向 Agent 代理发送相应的测试指令消息(MC),指定 Agent 代理端在一定的时间进行要求类型的测试,并接收测试返回的数据,最后把数据交给网络评价子系统。它主要包括三个组成部分:测试结点设计器、测试脚本定义器、测试任务管理器。

(1) 测试结点设计器

测试结点设计器在整个网络中部署测试结点(即指定有关结点的参数及结点之间的关系)。在结点设计器中有 Single Agent、Group Agent、Connector 三类结点实体。Single Agent 包括了 IP 地址、测试脚本、测试对编号等信息。Group Agent 与 Single Agent 的不同之处在于它包括至少两个 IP 地址。Connector 指定了测试的发起方和测试的脚本。结点设计图中包括三种 End-to-End 的测试映射:一对一的映射、一对多的映射和多对一的映射。整个结点设计图为上述三种映射的混合体。该模块把设计的测试结点信息转换成数据库要求的数据形式,保存在配置数据库中。结点设计图的信息采用 XML 文件格式存储,并可输出(Export)成测试实例(一个实际的测试用例)文件保存;

(2) 测试脚本定义器

测试脚本定义器为测试脚本的用户接口。可以定义的主要测试脚本有 Throughput. src、Response-Time. src、Loss-Packet-Rate. src 等;测试脚本中包含的主要参数有:源端口、目的端口、记录产生次数、记录重复次数、记录发送延迟时间、发送缓冲区大小、接收缓冲区大小、发送数据速率、发送文件尺寸等。

(3) 测试任务管理器

测试任务管理器统一管理系统的测试任务。它首先把用户定义的任务插入任务队列,然后不断从队列取出任务,每一个任务对应生成一个子线程(线程并发执行),子线程完成数据的发送/接收,最后当测试完成之后向网络评价子系统发出控制指令,要求其对系统进行分析、评价。它的主要任务有:① 参数管理;② 数据发送/接收;③ 测试任务调度。其中②中包括发送测试指令和接收 Agent 返回的测试结果两方面的功能。

3.4 网络评价子系统

网络评价子系统分析测试数据后,把分析后的数据用图形的方式进行显示,最后给出整个网络的性能评价,评价依据一定的评价模型进行。它主要包括三个模块:数据分析、数据显示、性能评价。各个模块之间的联系通过数据子系统进行。

(1) 数据分析

数据分析模块主要对测试获得的数据进行分析,分析参数主要包括可用性、延时和丢包率,分析后的结果送数据子系统处理。分析的内容包括:① 端到端(End-to-End)的延时或丢包率随时间的变化规律;②端到端的延时或丢包率的空间分布规律;③ 端到端的路径变化规律。数据分析模块被设计成一个单独运行的进程,它按照设定时间间隔定时分析。

(2) 数据显示

数据显示模块从分析数据库中取得分析数据,并将数据显示成曲线图、曲线区域图、饼图和二维直方图等形式。具体的要求符合 RFC2544。系统的绘图功能的实现可以利用 MFC 设备场景 CDC 类和基本的绘图工具的派生类。

(3) 性能评价

完成系统评价模型的定义,并依据该模型对整个系统的性能进行评价;评价时的数据来自分析数据库。网络性能评价模型涉及许多方面,包括:建立关于网络体系结构、网络协议模型和网络协议控制算法的量化模型,评测环境的仿真以及与这些结构、模型和算法相关的典型行为的模拟。本文提出的评价模型是一个简化的模型,主要考虑了网络的一些关键性能参数,并把参数值划分等级,选择不同的参数组合来反映不同网络的性能,进而制定不同的评价等级。

3.5 Agent 子系统

Agent 在功能上分为主动测试点 Agent1 和被动测试点 Agent2,实际上是同一个程序实体。Agent1 在指定的端口接收 Console 发送的消息代码,解析后生成测试任务插入测试任务队列,然后任务执行模块不断从队列中取出任务执行,任务执行模块根据任务要求,向 Agent2 发送测试帧,并接收 Agent2 返回的应答帧,计算测试结果,最后把结果发送给 Console。Agent2 在指定的端口接收测试帧,并对接收到的测试包进行预处理,计算出结果后返回应答帧。Agent1 和 Agent2 根据消息代码可分为:

(1) 网络测试子系统

网络测试子系统对网络的性能指标进行测试,它被设计成一个单独的守护进程,同时也被设计成总控其他线程的 Java 应用程序类(NetPerfTestingApp)。被控线程主要包括通断测试(PingThread)、吞吐量测试(ThroughputThread)、响应时间(ResponseTimeThread)等线程。另外远程访问接口实现类为它的成员变量,是该子系统数据的接口,它在数据子系统中

实现。被控线程被设计成公用(public)访问方式,它们从 Thread 类中派生并实现 Runnable 接口。测试过程采用了异步输入输出流技术(须包含 java. nio 包)。

(2) 设备数据采集子系统

设备数据采集子系统利用 SNMP 简单网络管理协议获取支持 SNMP 协议的网络设备的 MIB 参数,它类似于 SNMP 代理,运行在不同的网络工作站上,收集性能数据,然后把采集过来的原始数据统计分析后保存在测试数据库中。本子系统开发中使用了 AdventNet 公司的 Java 版 SNMP 类库:即 com. adventnet. snmp. beans. SnmpTarget,它提供了 get( )、getnext( )、getbulk( )、set( )、trap( )等成员函数来完成网络设备性能参数的采集。

4　测试实例分析

本实例中实验环境为:中型星型局域网,测试中 Console、Agent1、Agent2 各使用一台机器,Agent1 和 Agent2 都使用 10M 交换端口,测试文件尺寸为 163 885 字节,每次的测试结果为测试 100 次的平均值。我们主要分析了吞吐量随其他因素的变化情况。图 4 中测试包大小为 1 024 Bytes、发送速率无限制的情况下,吞吐量随时间的变化情况;图 5 中发送速率无限制,测试包大小变化的情况下,吞吐量随着包大小的变化情况;图 6 中测试包固定为 1 280 Bytes,发送速率变化的情况下,吞吐量随速率的变化情况(说明:吞吐量计算中延迟时间没有计算在内)。

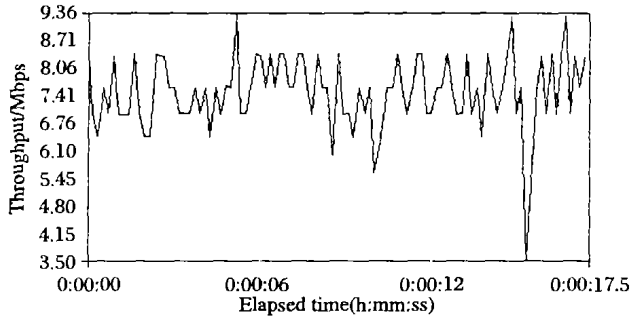

图 4　吞吐量随时间的变化

可以从图 4 中看出整个网络运行基本稳定,吞吐量保持在 8.00 Mbps 左右,只在 15 秒时刻吞吐量急剧下降,最低达到 3.50Mbps,这说明网络出现了严重的拥塞,性能下降。

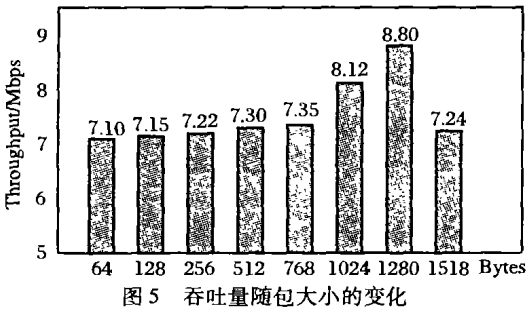

图 5　吞吐量随包大小的变化

由图 5 可知测试包的大小对吞吐量有比较大的影响。当包大小由 64 Bytes 逐渐增加时,吞吐量也随之缓慢增加;当包增大到 1 024 字节时,增速增快,吞吐量达到 8. 12 Mbps;当包增大到 1 280 Bytes 时,吞吐量达到最大 8. 80 Mbps;当包大小为 1 518 Bytes 时,吞吐量回到原来水平,下降到 7. 24 Mbps。因此,吞吐量测试时包大小选 1 280 Bytes 比较合适。

由图 6 可知测试包延迟对吞吐量有比较大的影响,当包延迟为 1ms 时,吞吐量小幅减少,下降到 6. 50 Mbps;当包延迟在 2ms ~ 9ms 之间,吞吐量与前相比继续减少,但维持在 4. 40 Mbps左右;当包延迟为 10ms 时,吞吐量最低达到 1. 56 Mbps,大幅减少。这反映了吞吐量测试中发送速率应符合一定的要求。其实 RFC2544 就给出了 Ethernet 中最大的帧速率参考,比如:1 280 字节的帧,1 s 须发送 961 帧。

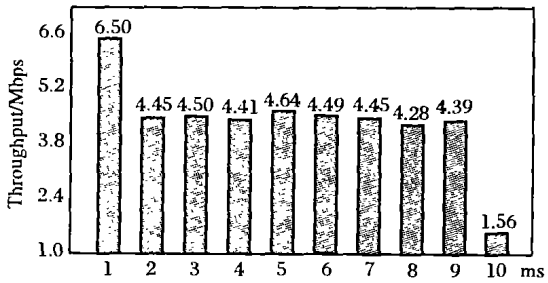

图 6　吞吐量随着包延迟的变化

在以上的实验中,我们还观察了测试系统对待测系统的影响。结果表明测试系统对待测系统影响非常少,另外使用 Java 开发虽然运行效率比 C 低,但没有影响我们测试的效率,也不影响待测系统的性能。

由于篇幅的原因,其他的测试实例就不在文中分析了。必须指出的是,以上各项实验结果都是在 Ethernet 环境下得到的,具有一定的局限性,还应当在广域网中开展实验,进行下一步的研究。

5　结语

网络性能测试是一项复杂的工作。为了得到准确可靠的测试结果,应考虑网络测试广播、网络测试流量对测试结果的影响,并尽量使这种影响减少到最低。大型复杂的网络性能测试还要考虑网络测试流的统计特征,比如测试流服从 Poisson 分布,各个测试点时钟的同步问题等。在对网络评价时,评价模型将直接影响评价的结果,在开发整个系统时还应充分运用多线程以及并发机制来减少系统的测试响应时间。实验结果表明,该系统具有很强的扩展性,适合于对大规模的网络进行实时性能测试。

参考文献:

[1] BRADNER S. RFC 1242 Benchmarking Terminology for Network Interconnection Devices[S], 1991.

[2] BRADNER S, MCQUAID J. RFC 2544 Benchmarking Methodology for Network Interconnect Devices[S], 1999.

[3] PAXSON V, ALMES G, MAHDAVI J, *et al.* RFC 2330 Framework for IP Performance Metrics[S], 1998.

[4] RAISANEN V, GROTEFELD G, MORTONA. RFC3432 Network performance measurement with periodic streams[S], 2002.

[5] AdventNet. AdventNet Agent Toolkit Java Edition [EB/OL]. http://snmp. adventnet. com, 2004.

[6] PAXSON V. An Architecture for Large-Scale Internet Measurement [EB/OL]. http://www. ncne. nlanr. net/nimi/papers/nimi-IEEE-081998. pdf.

[7] PAXSON V. Measurements and Analysis of End-to-End Internet Dynamics[M]. Berkeley: Computer Science Division University of California, 1997.

[8] BEYAH R, SIVAKUMAR R, COPELAND J. A Measurement-based Study of End-to-End Internet Traffic Dynamics[M]. School of Electrical and Computer Engineering Georgia Institute of Technology Atlanta, Georgia, 2002.