DOI 第51卷 第5期 武汉大学学报(理学版) Vol.51 No.5
2005年10月 J. Wuhan Univ.(Nat.Sci.Ed.) Oct. 2005, 594~598

文章编号: 1671-8836(2005)05-0594-05

一种基于移动代理的网络性能管理模型研究

吴黎兵[1], 吴产乐[1], 崔建群[2], 熊 卿[1]
(1.武汉大学 计算机学院, 湖北 武汉 430072; 2.华中师范大学 计算机科学系, 湖北 武汉 430079)

摘 要: 对传统网络性能管理系统的缺陷进行分析后, 提出了一种基于移动代理的网络性能管理模型. 该模型通过将网络性能管理需要采集的对象和指令封装在移动代理内, 利用Grasshopper移动代理平台进行支撑, 实现了网络性能数据的自主采集, 并利用被管网元的本地资源即时计算出最终的性能指标数据. 通过对传统性能管理和文中提出的基于移动代理性能管理进行分析和比较, 发现基于移动代理的网络性能管理可以减轻管理站的计算负荷, 减少网络管理数据流量, 从而降低因网络管理而带来的额外开销.

关 键 词: 移动代理; 网络性能管理; 性能分析
中图分类号: TP 319 文献标识码: A

0 引 言

随着网络规模的不断扩大, 复杂性不断增加, 对网络管理尤其是网络性能管理的要求也越来越高. 传统的集中式网管: 基于SNMP(simple network management protocol)和基于CMIP(common management information protocol)的网络管理, 由于采用定时轮询方式, 占用了网络中较多的带宽, 增加了网络的额外开销, 已越来越难适应网络的新发展. 因而一些新的分布式技术正逐步被应用于网络管理中, 如CORBA、主动网络、Web技术、移动代理技术等[1]. 其中移动代理MA(mobile agent)技术因其具有良好的分布、灵活、易扩展和容错等特性, 成为网络管理研究领域的热点之一.

1 传统的网络性能管理

网络性能管理主要处理与服务质量有关的网络运行状态, 它的任务主要包括: 收集和传送与资源性能当前水平有关的数据; 检查和维护性能记录, 以进行分析和计划. 一般来说, 以Manager/Agent模式运行的网管系统需要采集大量的性能数据进行分析处理. 由于大数据量的传送, 因此对Manager/Agent之间的连接质量提出了较高的要求; 同时, 由于Agent缺乏计算分析和自主决策的能力, 也引发了Manager端的计算瓶颈等问题, 这导致了Agent具备过滤、计算、自治等功能的需求[2].

性能管理的核心是性能数据的采集, 性能数据采集的准确、高效与否是性能管理质量的保证[3]. 目前基于SNMP的性能管理无论是集中式网管还是客户端/服务器模式的分布式网管, 在数据采集方面基本采用的都是定时轮询模式: 由管理者向分布在各个被管设备上的SNMP代理发出Get请求, SNMP代理接收到该请求后, 将被请求的MIB(management information base)变量值返回给管理者, 再由管理者对数据进行存储、计算、分析, 然后显示在图形化界面上. 这种方法实现起来比较简单, 但是却带来了不可避免的性能问题.

① 由于每隔一段很短的时间, 管理者就要向各个被管设备发出轮询, 同时接到请求的SNMP代理都要向管理者返回相应数据, 因此导致网络中增加很多SNMP报文, 占用了网络的带宽, 缩小了用户实际用到的有效带宽.

② 由于通常所需的性能数据并不是单独一个MIB变量就可以确定的, 如端口流入错误率, 需要得到ifInErrors、ifInUcastPkts和ifInNUcastPkts 3个变量值, 才能统计出真正所需的性能数据, 而这些计算工作都集中在管理者一方, 如果网络中的设备

收稿日期: 2004-12-19
基金项目: 国家863引导项目(2003AA001032); 软件工程国家重点实验室开放基金资助项目
作者简介: 吴黎兵(1972), 男, 副教授, 现从事网络管理, 网络通信, 网格计算研究. E-mail: wu@whu.edu.cn

过多, 则会导致管理者的负荷过重, 降低网络管理的效率, 甚至导致错误的发生.

将移动代理应用到网络性能管理中可以弥补上述传统网络性能管理的缺陷.

2　基于移动代理的网络性能管理模型

移动代理是一种能在异构计算机网络的主机间自主迁移, 代表其他实体的计算机程序, 它能从网络中的一个节点移动到另一个节点并继续运行, 必要时可以进行自我复制以及生成子代理. 每一个节点上的移动代理可以直接同本地服务资源进行交互, 待任务完成后再将结果集传送回管理站[4].

移动代理迁移的目的是为了使程序尽可能地接近数据源, 并在数据源端处理数据和实施管理操作, 从而可以降低网络的通信开销, 平衡负载, 提高完成任务的效率. 移动代理"迁移 计算 迁移"的工作模式及代理间的通信和协作能力为网络管理提供了全新的整体解决方案.

一些科研作者提出了基于移动代理网络管理的初步框架, 如图 1 所示[5,6].

其中 NMS(network management system)为网络管理工作站, 与传统网管中的工作站类似; NE(network element)为被管网元, 由于要在其上运行移动代理, 因此必须加载移动代理执行环境 MAE(mobile agent environment); MA 具有本地处理能

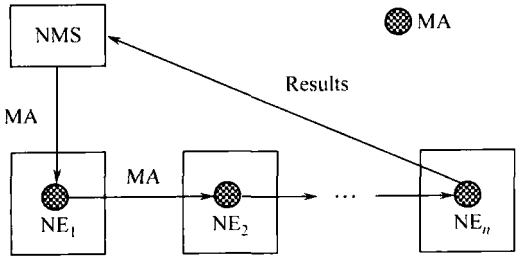

图 1　基于移动代理网络管理的初步框架

力和移动性, 可以处理数据并传送到其它网元或 NMS 上[7].

基于移动代理的网络管理工作原理并不复杂, 首先由 NMS 产生相应功能的 MA, 将其发送到第 1 个被管设备 NE_1 上, MA 在 NE_1 上执行相应的网管功能后, 带着可能存在的反馈信息移动到 NE_2, NE_3, …, 直到 NE_n. MA 在经过的这些网元上执行相应的网管功能, 并得到 NMS 所需的数据后, 回到 NMS, 完成一次管理操作.

在基于移动代理的网络性能管理中, NMS 不再像传统网管那样对每一个网元都发出大量轮询命令, 并等待网元上的 SNMP Agent 返回计算性能指标所需的单个 MIB 变量数据, 而是将采集命令封装在 MA 中, 将 MA 发送至第 1 个网元, 剩下的工作都由 MA 自主来完成. 根据这一原理, 本文提出一种基于移动代理的性能管理模型, 如图 2 所示.

NMS 首先在代理生成处生成一个新的基本 MA, 将其发送至代理封装处, 在代理封装处, 将基本 MA 要经历的网元地址集、采集的 MIB 变量集、要得到的性能指标集及其相应算法和存放结果集等相关信息封装在 MA 的属性中, 然后将其发送至第 1 个需采集的网元上.

封装好的 MA 移动到 NE_1 上后, NE_1 上的 MAE 负责解释并执行 MA 的代码, 其主要任务就是根据 MA 所携带的上述信息, 从传统的 SNMP 代理处获得要采集的 MIB 变量, 并根据 MA 提供的算法, 直接在网元本地计算出性能指标的结果, 存放在 MA 的结果集中.

MA 完成在 NE_1 上的性能数据采集和计算后, 按照 NMS 加载给自己的网元地址集列表, 移动到下一个设备进行同样的采集和处理, 直至地址集中的所有网元都处理完毕. 最后 MA 返回到 NMS 处, 将保存在结果集中的性能指标数据传递给 NMS 的性能数据处理模块, NMS 只需将其进行相应的存储

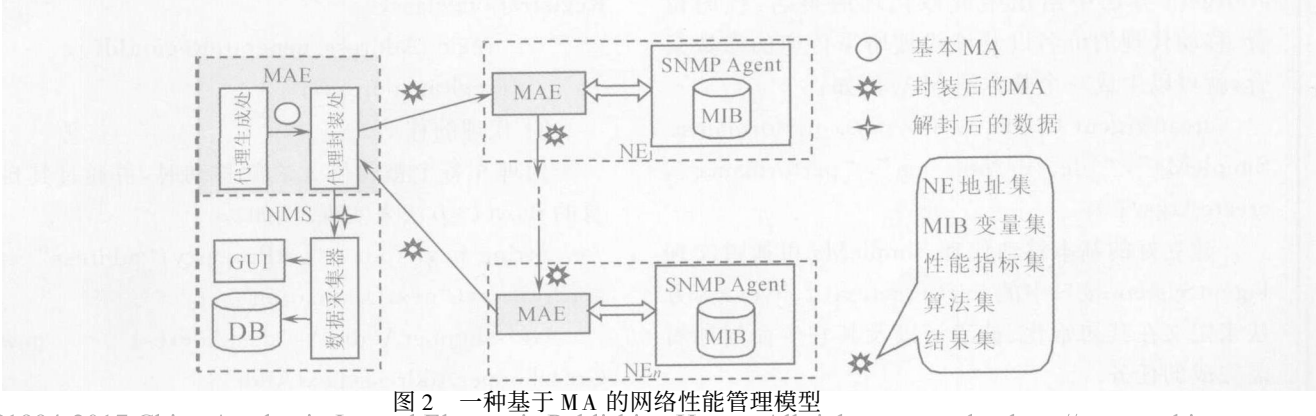

图 2　一种基于 MA 的网络性能管理模型

或显示,就完成了对整个网络被管设备的一次性能数据的采集和处理.

3 利用 Grasshopper 实现性能管理模型

3.1 Grasshopper 移动代理平台

Grasshopper 是建立在分布处理环境下的纯 Java 实现的移动代理平台. 分布处理环境使得将传统的客户端/服务器模式与移动代理相结合成为可能,而 Java 语言的平台无关性,则使得移动代理可以在各种支持 Java 虚拟机的异构网络环境下自由移动,适应了网络被管设备多样性的要求.

Grasshopper 移动代理平台的执行环境主要由域(Region)、域注册器(Region Registry)、代理处(Agency)和各种不同功能的代理(Agent)组成[8]. 可以将隶属于同一个团体的代理处和代理定义在相同的域里,并通过域注册器来登记该域所有的相关组件. Grasshopper 通过域来管理代理处的代理. 当代理移动到别处时,域注册器会自动更新与该代理相关的注册信息. 代理处为代理提供了实时运行环境,即 MAE,因此每一台需要执行代理操作的主机都需要运行代理处. 代理处包括核心代理处和泊位,由通讯服务、管理服务、保存服务、注册服务、安全服务和传输服务等执行代理操作所必需的服务构成;泊位则将逻辑上功能相类似的代理聚集在一起,通常处于同一泊位的代理共同完成某项功能.

3.2 实现方法

按照前面介绍的原理及性能数据采集流程,利用 Grasshopper 移动代理平台来实现性能管理,需经过以下几个步骤.

① 生成基本移动代理

Grasshopper 在 IagentSystem 接口里提供了 createAgent(…)方法来生成一个基本的移动代理对象. 只要实现了 IagentSystem 接口,并在 createAgent 方法中给出生成该代理的类名、代码位置、移动代理泊位名以及该代理所需传递的参数集合,就可以生成一个基本的 MA. 例如:

```
createAgent("edu.whu.nms.performance.SimpleMa", "file:/e:/nms/ma", "performance", createArgs);
```

建立好的基本移动代理 SimpleMa 可通过实现 IagentSystem 接口中的 init()、action()、live()等方法来定义在其初始化、被激活以及其它生命周期所需完成的任务.

② 封装基本代理

生成基本代理后,NMS 要对其进行封装,以便为其提供移动到网元之后所需的各种参数,如目的地址集、MIB 变量集、性能指标集、算法集等. 通过 setProperty()或 setProperties()方法来将这些参数加载到基本移动代理上. 例如:

```
setProperty("address", "socket://202.114.66.223:7002/wufox/place");
setProperty("OID", ".1.3.6.1.2.1.2.2.1.14");
```

③ 启动移动代理注册器

构造好所需的 MA 后,由于所有处在同一个域中的移动代理要和移动代理处进行通信,因此必须启动移动代理注册器,为两者提供通信的桥梁. 如要启动采用默认的 profile、端口为 7020、注册器名为 MARegistry 的代理注册器,可在 DOS 方式下直接输入以下命令:

```
Grasshopper r -n MARegistry
```

④ 启动移动代理处

驻留在被管网元设备上的移动代理处必须在 MA 开始移动前启动,才能为 MA 提供执行环境. 启动默认的移动代理处,并在移动代理注册器上注册,可使用下述命令:

```
Grasshopper a -r socket://<ip>:7020/MARegistry
```

⑤ 注册代理

代理要在代理注册器上注册才能完成各种移动操作. 只需在程序中生成 IregionRegistration 实例,就可以实现代理在注册器上的注册. 例如:

```
GrasshopperAddress regionAddress = new GrasshopperAddress(registryAddress);
//registryAddress 为代理注册器的地址
IRegionRegistration regionAgent = (IRegionRegistration)
ProxyGenerator.newInstance(IRegionRegistration.class,
regionAddress.generateRegionId(),
regionAddress);
```

⑥ 代理的移动

代理在各个被管网元之间移动时,可通过其自身的 move()方法来实现. 例如:

```
String nextAddr = getProperty("address" + getProperty("nextDestination"));
GrasshopperAddress destAddress = new GrasshopperAddress(nextAddr);
move(destAddress);
```

4　性能分析

采用上述移动代理方式采集网络性能管理数据, 相对传统的SNMP 轮询方式, 可以减少网络管理通信开销, 即减小由于网络管理而占用的额外带宽, 同时还可以缩短代理对 NMS 总的响应时间.

传统的SNMP 轮询方式, 需要对每一个被管网元设备都发出 get/getnext 请求, 然后等待每个网元上的SNMP Agent 将结果以 get_response 报文的形式返回. 如果要获取一个 MIB 变量表的数据, 则需要不断进行这种 getnext_ request get_ response 报文的交互, 因为要得到表中的下一个数据, 必须从上一个返回的 get_response 报文中获得下一个数据的 OID 标识, 这必然导致网络中的管理通信数据流量的增加. 此外, 很多性能指标只靠获取一个 MIB 变量, 无法完成指标的计算, NMS 通常要进行几次的查询才能得到计算性能指标的所有 MIB 变量. 例如要计算某端口的流入错误率, 需要采集 ifInErrors、ifInUcastPkts 和 ifInNUcastPkts 3 个 MIB 变量, 这使得网络通信数据流量翻倍[9].

假定网络中有 n 台被管设备, NMS 向每台设备发出一条SNMP get/getnext request 请求报文的平均大小为 K, NE_i 向 NMS 返回的响应报文大小为 A_i, 则 NMS 对网络中所有设备进行一次 MIB 变量的获取, 其数据流量为:

$$R_{SNMP} = \sum_{i=1}^{n}(K + A_i) \tag{1}$$

若计算某一性能指标, 需要采集 S 个 MIB 变量, 则计算一个性能指标所产生的数据流量为:

$$R'_{SNMP} = S \cdot \sum_{i=1}^{n}(K + A_i) \tag{2}$$

而基于移动代理的性能数据采集, 由于 NMS 将要采集的请求封装在 MA 中, 由 MA 将其带到网元上, 在网元本地进行各种查询和计算操作, 并将结果保存在 MA 中, 直到 MA 依次移动过所有被管设备, 再一次性的将结果返回给 NMS, 从而节省了 NMS 与每个设备进行通信的数据流量, 这种方式对于大数据量的 MIB 表变量的获取效率更高.

假定与上面相同的环境, 若 NMS 向第一个网元设备发出的 MA 大小为 M_1, MA 经过 NE_1 处理后, 其大小变为 M_2, 移动到 NE_2 上, 以此类推, 最后回到 NMS 处的数据包大小为 M_{n+1}, 则 NMS 对网络中所有设备进行一次 MIB 变量的获取, 其数据流量为

$$R_{MA} = M_1 + M_2 + \cdots + M_n + M_{n+1} = \sum_{i=1}^{n+1} M_i \tag{3}$$

若计算某一性能指标, 需要采集 S 个 MIB 变量, 移动代理方式的数据流量并不是(3)式中的 S 倍, 而是在封装基本代理时, 将要采集的 S 个 MIB 变量要求加载在 MA 上, 当 MA 移动到 NE_i 上时, 一次性地获取这 S 个 MIB 变量的值, 并计算出结果, 当它移动到下一个网元上时, 所携带的数据量还是只有一个, 因此只是数据包的大小略有增大, 而移动的次数不变. 计算一个性能指标所产生的数据量为

$$R'_{MA} = M_1(S) + M_2(S) + \cdots + M_n(S) + M_{n+1}(S) = \sum_{i=1}^{n+1} M_i(S) \tag{4}$$

对比公式(2)和(4), 当 n 较大时, 可以忽略一个数据包 $M_{n+1}(S)$的大小, 而得到以下公式:

$$R'_{SNMP} - R'_{MA} = \sum_{i=1}^{n}(S \cdot (K + A_i) - M_i(S)) \tag{5}$$

从公式(5)中可以看出, 只要 $M_i(S) < S \cdot (K + A_i)$, 采用移动代理方式的数据流量就会小于采用传统 SNMP 方式的数据流量.

在实际实验中, 选取 $\max(n) = 80$, 假定对每台设备仅采集前面提到的计算端口流入错误率的数据对象, 即 ifInErrors、ifInUcastPkts 和 ifInNUcastPkts, 因此 $S = 3$. 在网络管理运行过程中, 通过 Sniffer 软件获取数据包的大小并计算网络流量. 通过实验得到的数据流量对比如图 3 所示.

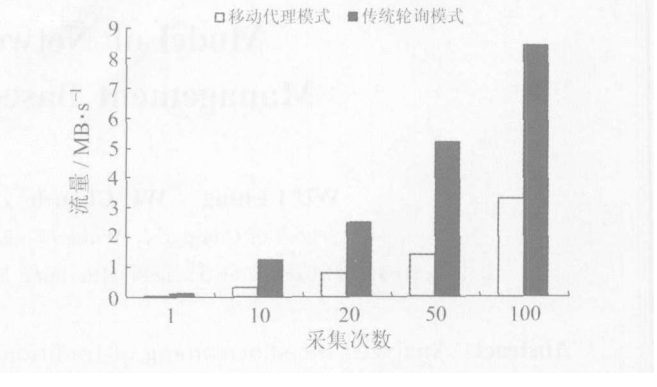

图 3　数据流量对比图

从实验数据来看, 采用 Grasshopper 移动代理平台实现的数据采集满足 $M_i(S) < S \cdot (K + A_i)$ 条件, 因此产生的性能管理数据流量明显小于采用传统轮询方式产生的数据流量. 当网络中被管设备越多时, 产生的数据流量 R'_{MA} 与 R'_{SNMP} 的差值越大, 可见采用移动代理的网络性能管理可以节省网络带

宽,降低因网络管理而带来的额外开销.

对于响应时间的分析,与上面类似,由于篇幅原因,在此不作赘述.

5 结束语

通过以上分析和讨论,可以得到如下结论:文中提出的基于移动代理的网络性能管理方法,可以减少网络管理的数据流量,缩短管理响应时间,避免传统SNMP轮询带来较大的网络管理通信开销,以及由此引起的网络抖动甚至短暂的阻塞.除此之外,还增强了网络中各节点的计算能力,使网络管理的计算功能不再集中在管理者一方,而是将这些计算功能合理地分配到网络中的其他网元设备,减轻了管理者的负载,从而提高整个网络管理的可靠性和分布性.模型在设计与实现中主要考虑系统性能问题,对移动代理安全性考虑得较少,这是整个系统原型转化为实际应用时需要进一步研究的问题.

参考文献:

[1] Bohoris C, Pavlou G, Cruickshank H. Using Mobile Agents for Network Performance Management [A]. *Proceeding of Network Operations and Management Symposium* [C]. Honolulu: IEEE Press, 2000. 637-652.

[2] Li Ye wen, Meng Luo ming, Qi Feng. The Study and Perspective of Mobile Agent Applications in Network Management Environment [J]. *Acta Electronica Sinica*, 2002, **30**(4): 564-569.

[3] Pagurek B, Wang Y, White T. Integration of Mobile Agents with SNMP: Why and How [A]. *Proceeding of Network Operations and Management Symposium* [C]. Honolulu: IEEE Press, 2000. 609-622.

[4] Damianos G, Dominic G. Implementing a Highly Scalable and Adaptive Agent Based Management Framework [A]. *Proceeding of Global Telecommunications Conference* [C]. San Francisco: IEEE Publishing, 2000. 1458-1462.

[5] Satoh I. A Mobile Agent Based Framework for Active Networks [A]. *Proceedings of IEEE Systems* [C]. Ichiro Satoh: IEEE Press, 1999. 1-5.

[6] 张普含,孙玉芳.一种基于移动代理的网络管理系统及性能分析[J].软件学报,2002,**13**(11):2090-2098.
Zhang Pu han, Sun Yu fang. Evaluating the Performance of a Network Management System Based on Mobile Agents [J]. *Journal of Software*, 2002, **13**(11): 2090-2098(Ch).

[7] Papaioannou T, Minar N. Mobile Agents in the Context of Competition and Cooperation [A]. *Proceeding of MAC3 Workshop* [C]. Seattle: Springer Verlag, 1999. 1251-1260.

[8] Grasshopper Programmer's Guide Release 2.2 [DB/OL]. *http://213.160.69.23/grasshopper website/download/doc/pguide2.2.pdf*, 2002-11.

[9] Rubinstein R, Duarte C. Evaluating Tradeoffs of Mobile Agents in Network Management [J]. *Networking and Information Systems Journal*, 1999, **2**(2): 237-252.

Model of Network Performance Management Based on Mobile Agents

WU Li bing[1], WU Chan le[1], CUI Jian qun[2], XIONG Qing[1]

(1. School of Computer, Wuhan University, Wuhan 430072, Hubei, China;
2. Department of Computer Science, Huazhong Normal University, Wuhan 430079, Hubei, China)

Abstract: Analyzed the shortcoming of traditional network management system, then, a model of network performance management based on mobile agents was proposed. Model suggested encapsulating the objects and instructions with mobile agents. Mobile agents can independently collect performance MIB objects node by node and compute the last performance data in local network entities in the model. We presented the architecture, design and implementation of such a model, compared and contrasted it to traditional poll mechanism implementation with experiment data. The data prove the model can decrease network overload and scatter compute tasks. We implemented the model by Grasshopper mobile agent platform.

Key words: mobile agent; network performance management; performance evaluation