

一种主动网络管理模型中的报文转发研究^{*}

李明

(重庆教育学院信息中心 重庆 400067)

摘要 主动网络是一个可编程的网络计算平台,它为用户提供了可编程的接口,允许用户或应用在网络内部定制自己的数据包处理方法。网络中传送的数据包是由用户数据和数据处理程序共同构成的主动包,主动包经过网络节点转发处理时,通过执行主动包中的程序代码来决定数据包的转发行为。传统网络管理不能发挥主动网络的分布式计算能力,不适应主动网络管理。本文讨论了一种基于节点的、用主动报文实现主动网管功能的主动网络管理模型,重点对管理报文的结构、转发机制和主动包路径转发算法进行了分析。

关键词 主动网络,网络管理,主动节点,主动报文

Researches on the Packet Transmission Model of Active Network Management System

LI Ming

(Network Information Center, Chongqing Educational College, Chongqing 400067)

Abstract Active network is a programmable platform which owns distributed calculating ability. It provides some interfaces for programmers to call. Within an active network, programmers can customize their own methods to process data packages. Data package in an active network contains both user information and process code. When an active data package arrives at a network node, it will be transmitted to next one, the whole course is controlled by the commands which exist in an active data package. Distributed calculating ability is not offered in a traditional network, and as a result, active network management is not available in such kind of environment. In this paper, we talk a network model over which is based on active message management, and the management message structure, transmission mechanism, arithmetic of transmission path choosing are adequately analyzed and discussed.

Keywords Active network, Network management, Active node, Network management system, Packet

传统的网络管理一般采用 C/S(Client/Server)技术架构,是一种集中式管理体系结构,在管理上面临通信瓶颈、带宽浪费、时效性差、消极被动和可扩展性差等问题。它不能利用主动网络中的节点的计算能力来管理网络,无法发挥和体现主动网络的优越性能,不能对主动网络实施有效的管理。主动网络是一种分布式、可编程的网络,其管理模式应能突破传统网络的非对称管理模式,使网络控制与管理工作站及主动节点之间达到一种对等的关系,从而克服传统网络管理中在管理端的瓶颈问题,减少带宽浪费,提高网络管理的时效性和主动性,便于业务的动态加载和动态 MIB 的管理与维护。

1 主动网络的管理特点及结构特征

为了适应主动网络的特点,主动网络的管理模式应能突破传统网络的非对称管理模式,使网络控制与管理工作站及主动节点之间达到一种对等的关系,从而克服传统网络管理中管理端出现的瓶颈问题,也便于业务的动态加载和动态 MIB 的管理与维护。主动网络管理结构如图 1 所示,主动节点是主动网管所要管理的主动对象。主动节点与控制管理工作站(NMS)之间的通信是一种对等的关系,而不像 SNMP 中客户端与服务端之间的非对等关系。ANMS(Active Network Management Serve)是主动网络管理的服务器,P 是由管理信息库 MIB 和代码服务器生成的一个管理任务。主动节点是网管系统的主要管理对象,负责处理主动包,它通过下

载 ANMS 上的管理应用 P 到本地执行。ANMS 通过定制 P 的转发例程,使 P 在各个主动节点上移动,并在访问节点时完成相应的计算。主动节点负责处理主动信包;执行环境(EE)提供了主动管理信包运行和处理所必需的环境;主动应用(AA)则执行主动管理信包中的代码。当管理节点需要执行某个管理任务时,它首先启动相应的管理程序,该管理程序创建一个封装体报文,然后将它注入到网络中。封装体报文到达主动网络中的节点时,节点根据管理节点发出和管理程序中的策略和计算规则执行相应的程序,并根据程序决定下面的动作,通过调用节点操作系统 OS 来访问各种资源来实施网络配置管理、性能管理、故障管理等功能。

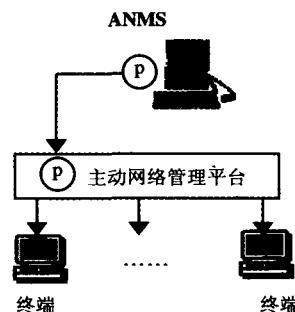


图 1 主动网络管理系统的结构

^{*}重庆市教委应用基础研究资助项目(041501)。李明 副教授,研究方向:计算机网络管理与应用,现代教育技术。

2 主动网络管理系统的结构

根据主动网络的管理特点及主动网络的结构特征,本文提出了一种基于节点为管理核心的主动网络管理模式,其框架结构如图 2 所示。对节点的管理程序和监视程序采用 ANEP 标准封装成一封装体并送到 ANEP 自适应数据鉴定

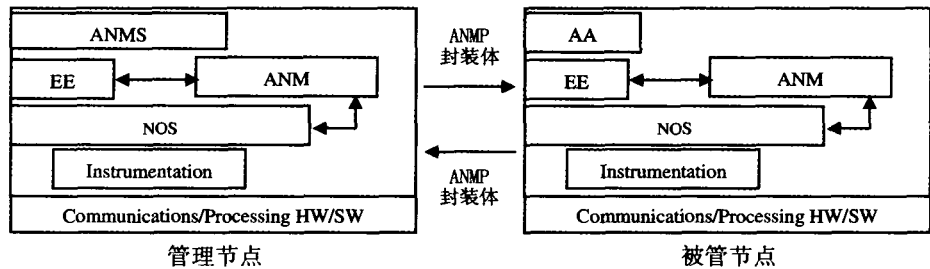


图 2 主网络管理系统总体结构

网络中的主动节点根据接收到的封装体的管理程序实现本地的管理功能。为发实现对节点的管理,在 NOS 与 EE 之间增加了主动节点管理器 ANM,在节点 NOS 与底层之间包含了指令适配器 (Instrumentation),实现对节点的管理。ANM 由一系列 SW (SoftWare) 组成,以实现对节点的监视、设置、分析与控制。ANM 通过节点 OS 的 API 发出指令来访问节点的数据、配置节点及操作事件,同时还为 EE 提供一套 API 接口,以使主动应用 (App) 可以动态地适应与配置网络资源、对网络性能进行监视,通过与 EE 的相互协作实现管理节点 EE 设置、性能并处理运行中出现的问题,调整 SW 使其能够动态地适应主动应用的变化,通过其它的 EE 或 AA 实现对节点配置对象的管理。

3 管理报文的结构与转发机制

3.1 管理报文的结构

主动管理报文是主动网管模型中的关键。在上述基于主动节点的网络管理框架中,所有管理功能都通过主动报文来实现。框架系统为节点的管理提供了一种动态可编程的管理模式。通过开发不同类型的管理主动报文,实现新管理策略的动态制定,使得网络管理软件也能循序地对新加载的业务进行管理,从而解决了目前网络难以把新的技术和标准引入现有网络中的困难。

在本系统设计中,主动报文封装在 UDP 和 ANEP 之中。主动报文由传统的 UDP 报头、ANEP 报头、主动报文主体和有效负载组成,如图 3 所示。主动报文主体遵循 ANTS 封装体的构造方式,头部有如下几个域:

Capsule/Protocol: 该字段用于说明本报文所属的代码段、代码组和相应的协议。

UDP 包头	UDP 包头		
	版本号	标志位	类型域 Type ID
UDP 包头	ANEP 头长度		ANEP 包长度
主动包	Capsule/Protocol	共享头	类型决定的头部信息
	有效载荷		

图 3 管理报文的封装格式

共享头:是不同类型封装体都具有的信息域,包含源地

器和监视器 Daemon (Data Adaptive Evaluator and Monitor),再由 Daemon 将包注入到网络中。当封装体到达被管理节点后,其转发例程被自动调用,根据管理节点中发出的管理程序中的转发策略和计算规则,在该节点执行一定的管理功能后根据结果决定后续动作。

址、目的地址、上一节点地址、版本信息等。

类型决定的头部:不同类型的主动包有不同的字段,字段的数目和大小也不相同。

在系统的设计中,将主动报文按照代码分发机制的不同分为直接执行报文和主动应用报文两种类型。直接执行报文采用“带内”的传输方式,即小程序代码直接封装在报文中进行传输;主动应用报文相对复杂,采用“按需索取”的代码分发机制,即主动报文中只携带代码标识。因此,可在头部信息中增加了两个固定字段,则系统中主动报文的统一格式为:“ANTS 固定头 + APTType + AppType + 类型决定部分”。其中,APTType 字段用来表示该主动报文的类型,0 表示直接执行报文,1 表示主动应用报文;AppType 字段表示该报文所属的应用,例如,0 表示 get 普通网管报文,1 表示 set 普通网管报文,2 表示巡逻报文。设立该字段的目的是为了使主动报文能够对传统节点实施操作。

3.2 管理报文的转发方式

在主动网络管理中,由管理站向节点发送具有管理功能的主动包,节点接收到主动包后可以执行包中的程序代码,通过执行程序代码完成对节点数据的采集和处理,从而实现管理的功能。在主动网络管理报文设计中,要充分考虑和利用主动网络的特点,采用恰当的包转发模型。

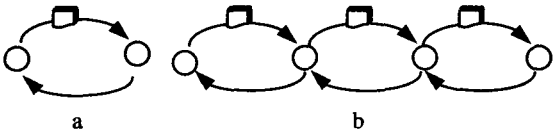


图 4 One-One 的转发模型

(1) One-One 模型

该模型是最简单和最常用的模型,它有两种结构:一种结构如图 4 中 a 所示,工作方式类似于传统网络中的“端到端”的通信模式,中间节点不执行接收到的管理报文中的具体管理程序,而是直接把该管理报文向目的节点转发,这种方式主要用于访问指定的节点,网管站只针对某一指定设备进行管理操作;另一种结构如图 4 中 b 所示,管理报文沿传输路径计算转发,管理报文每到一个中间节点时就要在该节点执行其携带的程序,通过这种执行方式,管理节点可以把集中的任务分发到沿整个传输路径上去执行。两种结构的最主要的区别

在于中间节点是否执行经过它的管理报文中携带的管理程序。

(2) BFST(Breadth First Search Traversing)转发模型

广度优先遍历转发模型采用并行控制模式,如图5所示。当管理报文到达一个主动节点后,它被转发到与当前节点直接相连的所有邻居节点。到达下一个节点时同样按照将该报文直接转发给相邻的节点。经过一次转发后,网络中将出现多个该管理报文的副本。当这些副本到达下一个节点后,它们继续被复制,并转发到它们的邻居节点,报文副本依次转发下去,直到遍历完整个网络。

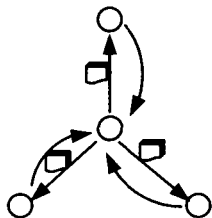


图5 BFST转发模型

(3) DFST(Depth First Search Traversing)转发模型

深度优先遍历转发模型是采用串行控制模式,如图6所示。在该模式中,管理报文到达一个主动节点后,它被转发到与当前节点直接相连的一个邻居节点。当它到达这个邻居节点后,它又被转发到该邻居节点的一个邻居节点,依次转发下去,直至遍历完整个网络。

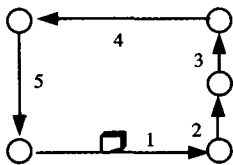


图6 DFST转发模型

4 主动管理报文径转发算法分析

图7是主动节点的抽象结构图。定义网络由连通图 $G = (V, E)$ 来描述, V 表示一组节点, E 表示一组节点间的双向连接。每个主动节点由 FF(Fast Forwarding) Unit、EE 等组成。每一个包通过每一跳都有转发延迟,它包括了传输延迟和队列延迟。在有些情况下,包可能会被 EE 处理,这样包还包括一个处理的延迟。假使通信链路对包的处理采用 FIFO 顺序,那么 EE 的服务程序也按 FIFO 顺序。一个包的头部主要包含了源、目的、应用标识等信息,FF 通过一组 Filters 进行包头的匹配,如果相匹配交给 EE 处理,否则直接向目的地址转发。每个包在节点上的这些延迟是有限的,为了分析简单,假定 FF 产生的延迟绑定一个常量 C ,代码在 EE 中执行的延迟看成其代码大小及算法的一个函数,表示为 $P(k)$ 。所以仅在节点上进行转发的包的延迟为 C ,通过 EE 的包的延迟为 $C + P(k)$ 。

忽略传输程序的代价,可以假定主动网络的代码大部分可以从节点缓存中获得。

函数 $P(k)$ 主要取决于包在 EE 中的计算,因为节点在处理时为了给 EE 发送消息它将对包进行拷贝,所以 $P(k)$ 至

少是线型的,可以假设:

$$P(k) = P_c + kP$$

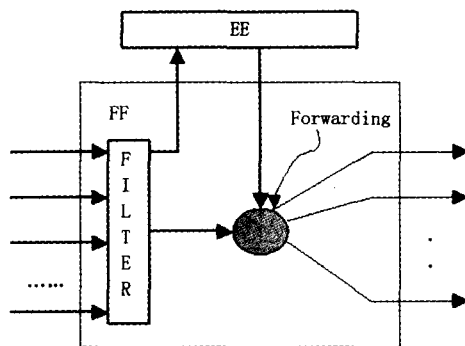


图7 主动节点的抽象结构图

其中 k 表示包的长度,在许多情况下 P_c 项可以忽略,所以上式简化为:

$$P(k) = kP \quad P \text{ 为一常量。}$$

定义 $TC(n)$ 表示时间复杂度,度量完成一项任务开始到结实的时间; $MC(n)$ 表示消息复杂度,度量完成一项任务主动包所经过的节点跳数。

管理报文常用的路径转发算法如图8所示,其中链路 A 看成管理中心或者注入相应主动包算法的节点。get-response 类似于 SNMP 中的请求回应; report-en-route 是请求到达节点后向源端发回响应同时向下一节点转发请求; collect-en-route 是请求到达节点后携带响应信息继续到下一节点,当到达目的节点后直接向源端返回; report-every-l 是按多域进行数据收集的算法,其在每个域中执行 collect-en-route 算法。

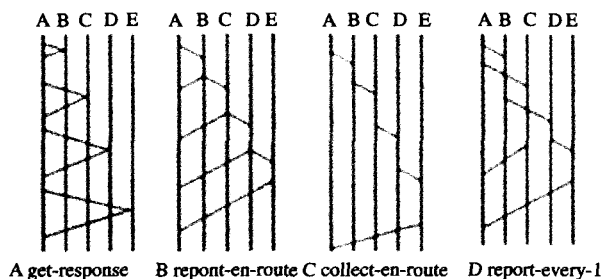


图8 主动包转发模式示意图

(1) 对于 get-response 算法有:

$$TC(n) = np + \sum_{i=1}^{n-1} 2iC; MC(n) = \sum_{i=1}^{n-1} 2i$$

其中 nP 是 n 个在 EE 中执行的动作延迟, $2i$ 为通过 $i = 1, 2, 3, \dots, n-1$ 跳的延迟。

(2) 对于 collect-en-route 算法有:

$$TC(n) = 2nC + \sum_{i=1}^n iP = 2nC + \frac{n(n+1)}{2}P$$

式中: iP 表示在节点 i 上的执行延迟,因为包的长度在经过一跳后会增加一个单元。很显然对于节点 n 的最大消息复杂度是不会超过 $2n$ 。

(3) 对于算法 report-en-route 有:

$$MC(n) = n + \sum_{i=1}^n i TC(n) = 2nC + nP$$

其中 nP 是在所有节点执行的延迟, $2nC$ 传输延迟,因为

(下转第 161 页)

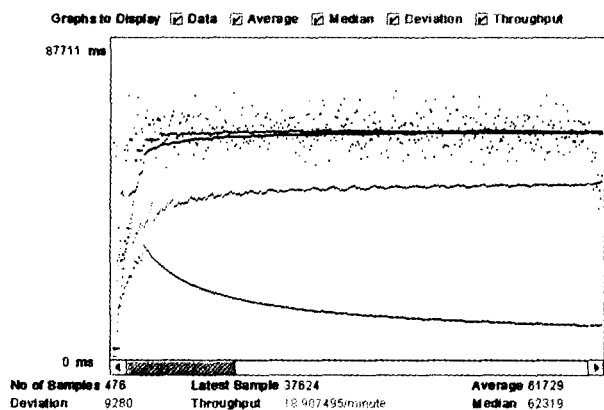


图 12 20 个线程不使用 Cache 的性能测试结果

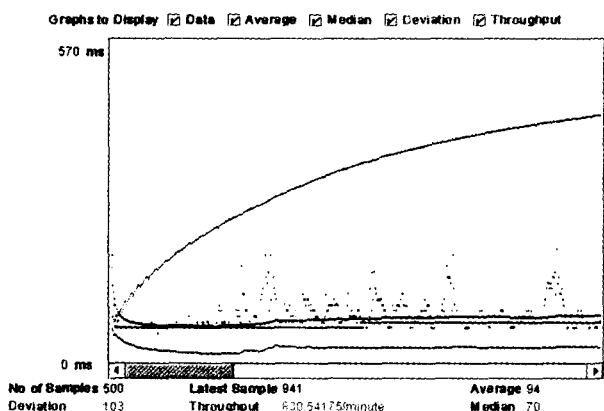


图 13 20 个线程使用 Cache 的性能测试结果

况下,系统的平均响应时间达 2077ms。使用 Cache 以后,系统的平均响应时间缩短为 60ms。由于第 1 组实验的用户 thread 只有 1 个,因此系统的吞吐量未达到满负荷,不能作为参考依据。

如图 12 和 13,在第 2 组实验中,当用户数达 20 个后,没有使用 DartCache 时系统的平均响应时间为 61729ms,使用 DartCache 后为 94ms。吞吐量也由 18.9 请求/分变成 830.5 请求/分。

实验证明,在使用了 DartCache 后,系统的性能和可能伸缩性都得到了巨大的提高。

结束语 采用 Cache 技术来提高数据库的访问性能是一种很常用的技术,已在工业界广泛使用。但是在 Cache 数据的分布性与 AOP 的设计上仍然有很多地方可以提高。

我们设计的 DartCache 是完整的分布式 Cache 解决方案,它在 Cache 数据分布与 AOP 设计上比以往的解决方案有了很大的改进。最后的实验结果证明,使用 DartCache 可使系统性能大幅度提高。

参考文献

- 1 Fowler M. Patterns of Enterprise Application Architecture. Addison Wesley, 2002
- 2 Deutsch P. 分布式计算八大谬论. <http://today.java.net/jag/Fallacies.html>. 2002
- 3 Neward T. Effective Enterprise Java. Addison Wesley, 2004
- 4 EhCache. <http://ehcache.sourceforge.net/>. 2004
- 5 OpenSymphony. OSCache. <http://www.opensymphony.com>. 2004
- 6 JBoss. JBoss Cache. <http://www.jboss.org>
- 7 Tangosol. Tangosol Coherence. <http://www.tangosol.com>
- 8 Ban B. Design and Implementation of a Reliable Group Communication Toolkit for Java. Cornell University, 1999
- 9 Apache. JMeter. <http://jakarta.apache.org/jmeter/>

(上接第 150 页)

包括发送到目的端以及目的端返回。

(4)对于算法 report-every-l:

report-every-l 算法是 collect-en-route 算法和 report-en-route 算法的折中。collect-en-route 算法具有线性的消息复杂度和二次项的延迟;而 report-en-route 算法具有线性的延迟和二次项的消息复杂度。综合这两种算法可以定义 report-every-l 算法,使消息、时间复杂度都比较理想,该算法优化了前面两种算法。算法 report-every-l 思想是将 n 分成 n/l 段,每段长度为 l ,为所有 n/l 段发送一个固定大小的初始化为 collect-en-route 算法的消息(可以在每段开始位置初始化一个赋值为 l 的计数器,每通过一个节点减一来实现)。所以对第 i 段来说其至少要经历 $(i-1)(C+P)$ 时间单元后才开始执行 collect-en-route,所以有:

$$TC(n) + (n-l)(C+P) + \sum_{i=1}^{n/l} (C+iP) = O(nC + (n+l^2)P)$$

$$MC(n) = O(n) + \sum_{i=1}^{n/l} (l+il) = O(\frac{n^2}{l})$$

如果选择 $l=\sqrt{n}$,就可以使得 $TC(\sqrt{n})$ 具有线性复杂度,而消息复杂度为 $O(n\sqrt{n})$ 。为了平衡两种复杂度,令 $l^2=n^2/l$,则两种复杂度都为 $O(n^{4/3})$ 。

综上,对本文的几种常用路径转发算法总结如下:

Algorithms	TC	MC
get-response	$O(np+n^2C)$	$O(n^2)$
report-en-rout	$O(np+nC)$	$O(n^2)$
collect-en-route	$O(n^2p+nC)$	$O(n)$
report-every-l	$O((n+l^2)p+nC)$	$O(n)$

结束语 基于主动网络技术的网络管理是当今网络管理界研究的一个热点,本文讨论了基于主动节点的网管模型,对

模型中的管理报文的结构、转发机制、转发算法进行了分析。主动网络是一种运行时可编程、可扩展的网络,提供了一种动态的运行环境。当网管报文到达网管主动节点时,网管主动节点将数据和程序代码分离,在执行环境中执行这些代码。网管报文由传统的 UDP 报头、ANEP 报头、主动报文主体和有效负载组成,报文主体遵循 ANTS 封装体的构造方式。在设计网络管报文中要充分考虑和利用主动网络的特点,采用恰当的包转发模型和算法。将具有网络管理功能的报文动态地分布在主动节点上,利用主动节点的计算能力,使节点能够自动发现、及时处理问题,是本系统实现网络优化管理的基本思想。把分布式计算模型引入到网络体系结构和网管体系结构是网管发展趋势,也是从根本上解决问题的方法。

参考文献

- 1 Fatta G D, Gaglio S, Re G L, Ortolani M [J]. Adaptive Routing in Active Networks. IEEE Openarch 2000, Tel Aviv Israel 23-24 March 2000
- 2 Fatta G D, Re G L. Active Networks [J]. an Evolution of the Internet. In: Proc. of AICA2001 - 39th Annual Conference, Cernobio, Italy, Sept, 2001
- 3 Munir S. Active Networks: A Survey [J]. <http://www.cse.ohio-state.edu/jain/cis788-97/ftp/activenets/index.htm>, 2000-07-02
- 4 Shaer E A. Active Management Framework for Distributed Multimedia Systems [J]. Journal of Networks and Systems Management, 2000, 8(1): 49~72
- 5 Kiwior D, Zabele S. Active Resource allocation in Active Networks [J]. IEEE JSAC, 2000, 19(3): 452~459
- 6 Brunner M, Stadler R. Service Management in Multi-Party Active Networks [J]. IEEE Communications Magazine, 2000, 38(3): 281~286
- 7 Calvert K L. Directions in Active Networks [J]. IEEE Communications Magazine, 1998, 36(1): 72~78
- 8 Kawamura R, Stadler R. Active Distributed Management for IP Networks [J]. IEEE Communications Magazine, 2000, 38(4): 114~121