一种主动网络中报文转发机制的研究

钟国祥
(重庆教育学院 重庆 400067)

摘 要 主动网络是一种新型的网络体系,不仅可以传送数据包,而且还可以执行数据包中特定的运算任务。主动网络为用户提供了可编程的接口,用户可通过网络中的节点动态地注入所需的服务。而传统网络管理不适应主动网络管理,因而不能发挥主动网络的分布式计算能力。讨论了一种基于节点的主动网络管理模型,分析了该模型的结构、管理机制和设计要点,并对网络拓扑发现和流量进行了分析。

关键词 主动网络,网络管理,主动节点
中图法分类号 TP393 **文献标识码** A

Study of the Hierarchy Management Model Based on Active Network Node

ZHONG Guo-xiang
(Chongqing Education College, Chongqing 400067, China)

Abstract Active network is a new framework where network nodes not only forward packets, but also perform customized computation on the packet flowing through them. It provides a programmable interface to the user where users dynamically inject services into the intermediate nodes. However, the traditional prototype of network management does not accommodate to the management of active networks, it cannot utilize the distributed copulation capabilities that active networks provides. This paper analysed the structure and mechanism of the active network management system, introduced a pattern of active network management, and studied the structure, management mechanism, design outline and each connection of the management system. The paper also studied the network topology discovery and traffic.

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

传统的网络管理由于采用集中式管理,无法利用主动网络中节点的计算能力来管理网络。因此,它们不可能对主动网络实施有效的管理,无法发挥和体现主动网络的优越性能。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Manager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。

1 主动网络管理

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。由图1可知,主动网络管理(ANM)体系结构中主动节点是主动网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。

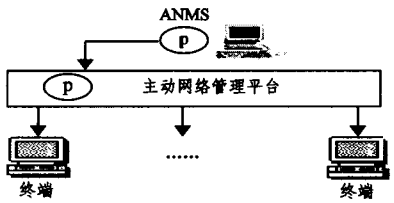

图1 主动网络管理系统的结构

在主动网络管理ANM(Active Network Management)系统中,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站(ANMS)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。主动节点是网管系统的主要管理对象,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与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,实施网络配置管理、性能管理、故障管理等功能。

到稿日期:2008-11-25
钟国祥(1969—),男,副教授,主要研究方向为人工智能。

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，我们提出了一种主动网络的管理模式。该管理以节点为管理核心，充分利用了主动网络的主动性、分布性和智能性，以实现主动网络的分布式智能管理，其框图如图 2 所示。

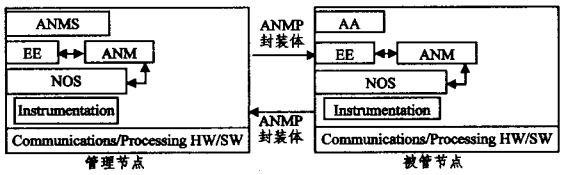

图 2 主动网络管理系统总体结构

图 2 中主动节点执行环境相当于 e，主动管理的执行与应用相当于 a，f 相当于主动管理部署与策略。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面的动作。

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(ANet Node Mgr)，在节点 NOS(Net OS)与底层之间包含了指令适配器(Instrumentation)，实现了对节点的管理[2]。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，同时为 EE 提供一套 API 接口，以使主动应用(App)可以动态地适应于配置网络资源，对网络性能进行监视。并通过与 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

在我们所设计的基于主动网络技术的网络管理系统中，所有管理功能都是通过主动报文来实现的，它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，将主动报文封装在 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。下面介绍本系统中主动报文主体各字段的用途和含义。

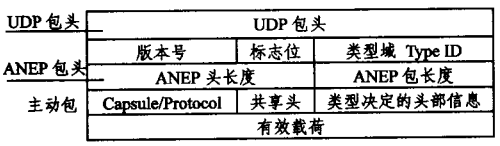

图 3 管理报文的封装格式

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。

共享头：是不同类型封装体都具有的信息域，包含源地址、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，它采用“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，我们在头部信息中增加了两个固定字段。系统中主动报文的统一格式如图 4 所示。其中，APType 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。

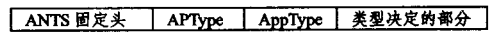

图 4 管理报文的格式

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

3.2.1 One-One 模式

One-One 的转发模式是一种最简单的转发模式，如图 5 所示。它又根据包是否在所经过节点分成两种结构：一是所经中间节点不执行，其方式与目前的端到端通信方式类似，这种方式主要用于访问指定的节点，如图 5 中(a)所示；二是沿传输路径计算的转发模式，在这种转发模式中，封装体报文按照指定每到一个中间节点时，就要在该节点执行其携带的程序，如图 5 中(b)所示。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。

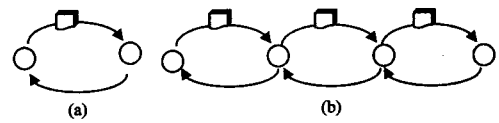

图 5 One-One 的转发模式

3.2.2 BFST(Breadth First Search Traversing)转发模式

这是一种并行控制模式，如图 6 所示。当封装体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。显然，经过一次转发后，网络中将出现很多该封装体报文的副本。当这些副本到达下一个节点后，它们又被复制，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。

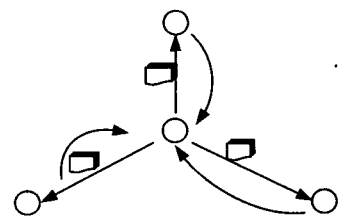

图 6 BFST 转发模式

3.2.3 DFST(Depth First Search Traversing)转发模式

这是一种串行控制模式，如图 7 所示。在该模式中，封装体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。

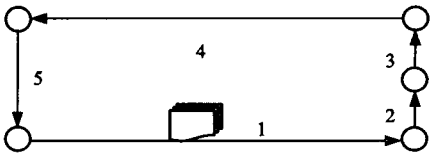

图 7　DFST 转发模式

4　主动包路径转发算法分析

我们定义网络由连通图 $G=(V,E)$ 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。如图 8 所示，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。我们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，我们可以假定 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 $C+P(k)$。

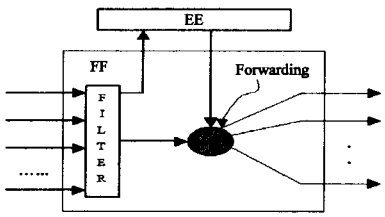

图 8　主动节点的抽象结构图

在这里忽略了传输程序的代价，我们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，它将对该包进行拷贝，所以 $P(k)$ 至少是线性的，因此我们假设：

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

其中 k 表示包的长度。在许多情况下可以忽略 P_C 项，所以上式简化为

$$P(k)=kP$$

P 为一常量。

为了性能分析，我们定义

$TC(n)$：时间复杂度，度量完成一项任务开始到结束的时间；

$MC(n)$：消息复杂度，度量完成一项任务主动包所经过的节点跳数。

下面总结本文中的几种常用算法，并进行分析。其中 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 算法的应用，比如节点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。为了更清楚地描述各算法，下面用图形表示，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，各算法如图 9 所示。

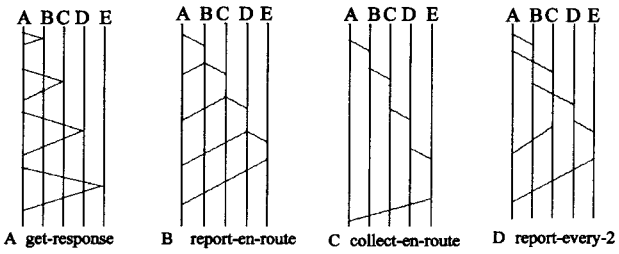

图 9　主动包转发模式示意图

对于 get-response 算法，有

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

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

对于 collect-en-route 算法，有

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

其中 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 的最大消息复杂度不会超过 $2n$。

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，$2nC$ 为传输延迟，因为包括发送到目的端以及目的端返回。

经过以上分析，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得到两者的折中，定义 report-every-l 算法，该算法优化了前面两种算法，使消息、时间复杂度都比较理想。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，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：

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

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

如果选择 $l=\sqrt{n}$，就可以使得 $TC(n)$ 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 $O(n^{\frac{4}{3}})$

对本文的几种常用路径转发算法总结如表 1 所列。

表 1　路径转发算法比较

算法	时间复杂度	消息复杂度
get-response	$O(nP+n^2C)$	$O(n^2)$
report-en-route	$O(nP+nC)$	$O(n^2)$
collect-en-route	$O(n^2P+nC)$	$O(n)$
reort-every-l	$O((n+l^2)P+nC)$	$O(n^2/l)$

结束语 主动网络管理体现了主动网络的思想，将一部分网络管理功能动态地分布在主动节点上，充分利用了主动节点的计算能力，使节点能够自动发现、解决问题，从而极大地优化了网络管理。本文讨论了一种主动网络的管理模型，该模型中各个模块相互独立、任务明确，而且在每个层都可以动态更新以适应主动网络中主动节点的易变性和主动应用的扩展性，因此网络管理的稳定性和扩展性都大为提高，适应了现代网络管理的需要。

参 考 文 献

[1] Di Fatta G, Gaglio S, Lo Re G, et al. Adaptive Routing in Active Networks// IEEE Openarch 2000. Tel-Aviv Israel, March 2000

[2] Di Fatta G, Lo Re G. Active Networks[J]. An Evolution of the Internet// Proc. of AICA2001-39th Annual Conference. Cernobbio, Italy, Sept. 2001

[3] Munir S. Active Networks: A Survey[EB/OL]. http://www.cse.ohio-state.edu/jain/cis788-97/ ftp/ activenets/index.htm, 2000-07-02

[4] Al Shaer E. Active Management Framework for Distributed Multimedia Systems [J]. Journal of Networks and Systems Management, 2000, 8: 49-72

[5] Brunner M, Stadler R. Service Management in Multi-party Active Networks [J]. IEEE Communications Magazine, 2000, 38 (3): 281-286

[6] Calvert K L. Directions in Active Networks [J]. IEEE Communications Magazine, 1998, 36 (1): 72-78

(上接第 76 页)

响应者端进行实验的验证结果

使用 AAAP 算法对 NSL 协议响应者端的验证结果如图 7 所示。

```
Protocol: Needham-Schroeder-Lowe
This is the Responder's Guarantee
Goal Set;
goal is B;
goal is Nb;
goal is A;
goal is Na;
We use Outgoing Test
First, we use Outgoing Test Part One;
Second, we use Outgoing Test Part Two;
We have proved Na by Outgoing Test Part One;
We have proved B by Outgoing Test Part One;
We have proved A by Outgoing Test Part One;
We have proved Nb by Outgoing Test Part Two;
The protocol is right, every component has been proved!
```

图 7 NSL 协议响应者端的验证

从以上验证结果可知，AAAP 算法证明了 NSL 协议能够满足完整的身份认证和消息保密的属性。

3.3 AAAP 算法的效率分析

AAAP 算法通过实验验证了 NSL 协议的正确性，将其算法效率同 Athena 算法和 Murφ 算法进行比较，表 1 列出了 3 种算法在对 NSL 协议进行证明过程中产生的状态数量。可以看出，当 NSL 协议仅有一对主体参与的情况下，AAAP 算法仅遍历 10 个状态就能够证明 NSL 协议所满足的认证属性，算法收敛性较好。

表 1 NSL 协议验证过程状态遍历情况

验证工具	Murφ	Athena	AAAP
遍历状态	1706	19	10

结束语 本文在改进的认证测试方法的基础上提出了安全协议自动化验证算法——AAAP 算法。该算法以 3 条改进的认证测试规则为依据实现了对于安全协议非单射一致性属性的正面。此外，本文还提出了测试分量唯一性属性，从而增强了 AAAP 算法检测重放攻击的能力。最后，通过对 Neuman-Stubblebine 协议和 NSL 协议的实验验证证明了 AAAP 算法的有效性，并通过将 AAAP 算法与 Athena 算法、Murφ 算法的遍历空间数进行比较，说明了 AAAP 算法在协议验证中的高效性。

参 考 文 献

[1] Meadows C, Syverson P F, Cervesato I. Formal specification and analysis of the Group Domain of Interpretation Protocol Using NPATRL and the NRL Protocol Analyzer. Journal of Computer Security[J], 2004, 12(6): 893-931

[2] Mitchell J C, Mitchel M, Stern U. Automated analysis of cryptographic protocols using Murφ // Proceedings of the 1997 IEEE Symposium on Security and Privacy[C]. Okland: IEEE Computer Society Press, 1997: 141

[3] Thayer F J, Herzog J C, Guttman Joshua D. Strand space : why is a security protocol correct [A] // Proceedings of the 1998 IEEE Symposium on Security and Privacy[C]. California, USA: IEEE Computer Society Press, 1998: 160-171

[4] Guttman J D, Fabrega T F J. Authentication tests [A] // Proceedings of 2000 IEEE Symposium on Security and Privacy [C]. Oakland CA: IEEE Computer Society Press, 2000: 96-109

[5] Song D X. Athena: A new efficient automatic checker for security protocol analysis // Proceedings of the 1999 IEEE Computer Security Foundations Workshop [C]. Washington DC, USA: IEEE Computer Society Press, 1999: 192-202

[6] Liu D X, Li X Y, Bai Y C. An attack-finding algorithm for security protocols. Journal of Computer Science and Technology[J], 2002, 17(4): 450-463

[7] Li Xiehua, Yang Shutang, Li Jianhua, et al. Security protocol analysis with improved authentication tests[A] // Proceedings of the 2nd Information Security Practice and Experience Comference [C]. vol. 3909 of Lecture Notes in Computer Society[C]. Hangzhou: Springer Verlag, 2006: 123-133

[8] Lowe G. Breaking and fixing the Needham-Schroeder public-key protocol using FDR // Proceedings of TACAS[C]. volume 1055 of Lecture Notes in Computer Science. Springer Verlag, 1996: 147-166

[9] Lowe G. A hierarchy of authentication specifications[A] // Proceedings of the 10th Computer Security Foundations Workshop [C]. Washington DC: IEEE Computer Society, 1997: 31-43

[10] Neuman B C, Stubblebine S G. A note on the use of timestamps as nonces[J]. Operating Systems Review, 1993, 27(2): 10-14