第42卷 第10期 西 安 交 通 大 学 学 报 Vol.42 №10
2008年10月 JOURNAL OF XI'AN JIAOTONG UNIVERSITY Oct. 2008

一种可演化网络的研究与实践

刘涛[1]，钱德沛[1]，王锐[2]，栾钟治[2]，黄泳翔[1]，许大炜[1]
（1.西安交通大学电子与信息工程学院，710049，西安；2.北京航空航天大学计算机学院，100083，北京）

摘要：针对目前的互联网难以满足不断发展的应用需求问题，提出了一种新的可演化网络的模型.可演化网络可以根据网络运行的状态和网络应用的特征进行自主调整，由此改变自身的算法，以提供不同的服务质量，从而使得网络能够应对多变的应用环境.该网络的演化分为3个层次：模块运行参数的自调节、网络模块的自配置和网络节点间的协同工作.同时，设计了可演化网络的体系结构，实现了基于软件的原型系统，讨论了系统实现的关键技术和提高系统性能的优化方法.应用实验表明，可演化网络能够及时适应网络状况的变化，更加平稳、高效地提供网络服务.

关键词：可演化网络；自主通信；自配置；自适应；服务质量
中图分类号：TP393 **文献标志码**：A **文章编号**：0253-987X(2008)10-1193-11

Research and Practice on Evolutionary Networks

LIU Tao[1], QIAN Depei[1], WANG Rui[2], LUAN Zhongzhi[2], HUANG Yongxiang[1], XU Dawei[1]
(1. School of Electronics and Information Engineering, Xi'an Jiaotong University, Xi'an 710049, China; 2. School of Computer Science and Engineering, Beihang University, Beijing 100083, China)

Abstract: To tackle the problem that the internet can not adapt to the requirements of ever-changing and wide applications, a novel evolutionary network model is proposed. The evolutionary network can adapt its behaviors according to the network conditions and status by adaptively changing its control parameters and control algorithms to achieve different QoS for different applications. Three levels of evolution including parameter adaptation, network module reconstruction, and collaboration among network nodes are presented. The architecture of the evolutionary networks is designed and a software-based prototype is developed. Key technologies and optimization methods for performance are studied. Experiment results show that the evolutionary network can adapt to the dynamic changes in network conditions and provide stable and efficient network service.

Keywords: evolutionary network; autonomic communication; self-configuration; adaptive; quality of service

自20世纪60年代末分组交换网络概念的提出，互联网(Internet)的发展已经经历了近40年的历史.从一个纯粹研究性的网络，成长为今天承载各类应用和业务的全球性基础设施，互联网经历了创新、完善、成熟、普及的发展道路.互联网的成功之处主要是：①基于分组交换，以无连接的形式实现了数据包的传输和交换，有效改善了网络的鲁棒性；②分层的协议栈采用单一网络层协议，为网络提供了简洁的编程接口；③通过路由器实现的网络层协议，将各个局部的网络互联起来，由此实现了互联网的灵活扩展；④复杂功能(可靠传输，拥塞控制)置于端点，网络中间结点尽可能简化，从而高效地完成了路由和数据转发.这些正是互联网在短时间内遍布整个世界，成为不可或缺的信息基础设施的主要

收稿日期：2008-01-23. 作者简介：刘涛(1980—)，男，博士生；钱德沛(联系人)，男，教授，博士生导师. 基金项目：国家自然科学基金资助项目(69973036，90104022，60673180)；国家重点基础研究发展规划资助项目(G199932710).

原因.

随着网络规模的不断扩大,承载业务和应用日趋多样化,互联网也暴露出其不足之处,原本一些特点和优点反而成为制约发展的不利因素,主要表现在以下几个方面.

(1)协议一旦部署,难以改变.协议制定和部署的速度远远低于应用程序的发展,不能满足飞速发展的应用要求.

(2)难以适应不同应用的 QoS 要求.在目前的 QoS 机制中,集成服务的可扩展性比较差,区分服务分级粗糙且依赖于设备功能,这将难以以灵活的方式根据应用的性质提供所需质量的服务.同样,网络设备的 QoS 配置很难动态地适应和满足用户的不同要求.

(3)难以应对复杂的网络管理.网络协议、服务的多样性,有线、无线、移动等各类异质网络的互联,不同自治域之间策略的差异性,网络性能和安全管理的新需求等,都对网络管理提出了新的挑战.

(4)复杂功能置于端点,中间结点只完成简单数据转发的这一特点,在一定程度上又限制了网络的性能和控制.例如,拥塞控制依赖于发送端点对拥塞状态的感知能力,从而降低了发送速率.这种状态传递比较迟缓,不能使端点及时了解拥塞发生的时间,也难以在早期化解网络拥塞.由于控制参数难以动态调整,所以网络中间节点的拥塞控制效果往往不尽人意.

(5)目前的互联网只提供了基本的网络连通性,而在安全性、移动性等方面却未见更有效的措施,所以它不适用于诸如无线、传感器等网络环境.

为了解决上述问题,新一代互联网研究已经受到广泛关注.下面是一些具有代表性的内容.

(1)主动网[1].这个概念是 20 世纪 90 年代中期提出的,是一种允许用户为网络中间节点编程的新型网络.具有计算能力的主动网节点不再是简单地转发流经的数据,而是针对应用敏感性、根据流经的数据包性质执行不同的代码,由此产生各异的效果,同时允许用户按需创建自己的服务并部署于网络.主动网能自我复制、自我再生、自我发展、自我保护,其结构和行为不再取决于静态的参数设置,而是根据场景动态变化.

(2)层叠网络.它是当前网络研究的热点之一,其试图解决网络基础设施发展缓慢与网络应用日新月异的矛盾[2].主要思想是:从应用的特点和需求出发,通过一定的计算,在现行的网络之上构建出满足特定应用需求的虚拟网络.然而,构建虚拟网络并不能解决承载网络本身的基本问题,如果承载网络出现错误,构建虚拟网络只会增加问题的复杂性.

(3)自主通信.它是在 WAC 会议上提出的一种通信系统.该系统可以感知网络环境,发现网络环境的变化,并理解这种变化的含义[3-4],再根据变化实施网络的控制和管理,生成服务,进行服务重组等.相关的研究工作多数侧重于服务的生成和新服务的组合[5]等,而对底层的网络传输研究甚少.

综上所述,本文提出了一种可演化网络的概念,其主要思想是:网络具有演化和自我发展的能力,可以根据运行的状态和应用特征进行自我调整,改变自身的算法、协议和控制机制,根据应用的需求提供不同的服务质量,发现并对抗安全威胁,使得网络能够自动应对不断变化的运行环境、安全状态和应用需求,提供最佳的网络服务.

1 可演化网络模型

以路由器为核心的互联网的能力和性质取决于路由器的构造和功能,所以可以对传统路由器作如下抽象.

定义 1 一个路由器可以定义为一个五元组

$$R=\{r,\ T_R, f_Q, f_R,\ f_M\}$$

式中:r 为路由函数,决定下一跳路径;T_R 是路由表;f_Q 是转发过程中的 QoS 控制;f_R 是路由表更新管理功能,由路由协议决定;f_M 是路由器的管理部件,通常称为管理代理,它完成路由器的状态采集和配置.

路由器对数据包的处理动作可表示为 $D\_\mathrm{Proc}=r(T_R, f_Q)$,即根据路由表和 QoS 控制设置完成数据的转发.

更新后的路由表可表示为 $T'_R=f_R(T_R)$.

定义 2 在可演化网络中,一个可演化路由器可表示为

$$R_E=\{r,\ T_R, f_{EQ}, P,\ f_{ER},\ f_{PR},\ f_{EM}\}$$

式中:f_{EQ}是可扩展的控制函数集合,可以动态地部署于路由器,或者由数据包来携带,它决定着数据包的 QoS 和拥塞控制等动作;P 是 QoS、拥塞等控制参数集合,f_{EQ}通过 P 作用于数据包;f_{ER}是可变的路由表管理参数,意味着路由器支持的路由协议可以动态变化;f_{PR}是反射部件,它感知或接收网络的状态,并修改参数集合 P;f_{EM}是可扩展的管理功能,在运行时可以动态部署和变化.

在这种可演化路由器中,数据包处理的动作可

以表示为 $D\_\mathrm{Proc}'=r(T_\mathrm{R},f_\mathrm{EQ}(P))$,即根据路由表,并在控制函数和对应参数的控制下,完成数据转发动作.控制参数 $P'=f_\mathrm{PR}(P)$可通过反射部件来调整.更新后的路由表 $T'_\mathrm{R}=f_\mathrm{ER}(T_\mathrm{R})$.

可演化网络的路由器与传统路由器的最大区别在于,它具有可动态部署、动态修改的控制函数集合,具有路由管理和网络管理功能.这些功能可以在数据包的驱动下或在外部控制下变化,从而改变路由器的能力和行为,进而改变数据包处理的效果.此外,控制函数所使用的控制参数也可以随感知网络状态而变,或者在外部控制下进行修正.

通过这种可演化路由器,可以构造可演化网络.

定义 3　一个可演化网络 N_E 可以由一个五元组来表示

$$N_\mathrm{E}=\{S_{R_\mathrm{E}},L,\mathrm{ext\_}q,\mathrm{ext\_}r,\mathrm{ext\_}m\}$$

式中:S_{R_E}表示可演化路由器集合,$S_{R_\mathrm{E}}=\{R_{\mathrm{E}i}\mid i=0,\cdots,n\}$;$L$ 为连接 R_E 的链路集合;$\mathrm{ext\_}q$ 是 QoS 等网络模块及控制函数的扩展机制,完成路由器的 f_EQ的动态部署和修改,即 $f'_\mathrm{EQ}=\mathrm{ext\_}q(f_\mathrm{EQ})$;$\mathrm{ext\_}r$ 是路由协议和路由表管理的扩展和修改机制,决定着路由器所支持的路由协议和路由表的管理功能,即 $f'_\mathrm{ER}=\mathrm{ext\_}r(f_\mathrm{ER})$;$\mathrm{ext\_}m$ 是网络管理功能的扩展机制,可改变路由器的管理功能,即 $f'_\mathrm{EM}=\mathrm{ext\_}m(f_\mathrm{EM})$.

可演化网络的演化能力体现在以下 3 个方面.

(1)网络算法的自适应性.这种自适应性主要表现在参数可根据网络状态的变化、时间的推进自适应地变化,以适用于不同的网络环境.对比拥塞控制中的 DropTail、ARED 算法可以清楚地看出自适应算法与固定算法的不同之处,固定算法 DropTail 在缓冲区无运行空间时会丢弃新接收到的数据包,而自适应算法 ARED 则会根据缓冲区的大小、数据包的多少以及丢包策略进行计算,从而获得新的参数,最后通过不断地调整参数来满足不断变化的网络运行需求.显然,可演化网络中的各种算法应该采用类似于 ARED 的自适应算法.

(2)网络节点软件的反射性.这种反射性主要体现在网络软件或模块可以动态变更,以适应于广泛变化的网络环境.具有反射性的软件可以在运行时动态调节自身代码,所以在网络节点的设计中,应考虑软件的某个功能由多个算法来完成,不同的算法具有不同的适用环境,这样当网络状态发生变化时,通过一个机制可以使得网络节点能够选择并切换到适合当前网络状态的算法.例如,RED、BLUE、SFB 这 3 种拥塞控制算法各有其优缺点,但在可演化网络节点中,可以根据外界环境和需求适当地选择、切换它们,以达到更好的拥塞控制效果.这种结果在传统的网络环境下,利用单一算法或经管理员手工调节参数也难以实现.为了达到算法或模块可切换的目的,在网络节点上需要额外添加信息收集监测、模块切换等手段.如图 1 所示,信息收集模块负责收集软件模块的信息和网络状态,并由信息处理与反射模块通过计算来切换使用不同的软件模块.

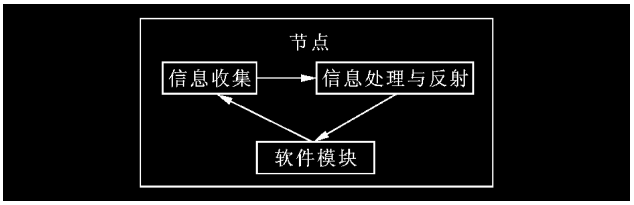

图 1　网络软件的反射性

(3)网络节点的协同性.这种协同性主要表现在可演化网络软件能够协同工作,相互修复.传统网络路由器之间可以相互交换路由信息,而在可演化网络中,网络节点之间不但可以交换路由信息,而且可以交换安全信息和预警信息等.也就是说,在网络中应部署多个分布式管理系统,这些系统可以综合分析网络节点发来的网络事件信息,并做出合适的反应,如网络软件的安装和更换、软件参数的配置等.同时,这些反应会及时作用到相应的网络节点之上,通过调整多个网络节点的软件与参数,达到调整整个网络协同运行的目的.如图 2 所示,2 个网络节点将收集到的数据在服务器上进行综合处理,再根据处理结果对这 2 个网络节点分别进行管理.例如,一个具有协同性的分布式攻击检测系统,其网络节点可定时将流量信息发送到分析节点上,分析节点通过分析流量数据、判断攻击发生的地点和情况、修改攻击点附近的防火墙规则来阻止攻击进一步发生.这种分布式的检测与分析能力就需要各个网络节点相互协同,并具有高性能计算能力,这也是对一个可演化网络的最高级层次的要求.

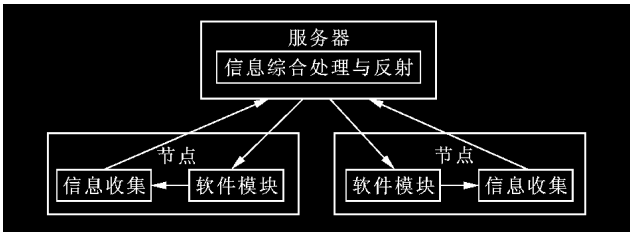

图 2　网络软件的相互协同性

综上,实现可演化网络的关键是可演化路由器

的控制功能、路由功能和管理功能如何动态变化、扩展和部署，也就是如何实现上述模型中的 ext_q、ext_r和 ext_m 等机制，这就是本文要讨论的主要问题.

2 可演化网络体系结构

针对上述模型及实现中所涉及到的问题，提出了一种可演化网络的系统结构，如图3所示. 该体系结构由可演化路由器、网络事件存储节点、网络事件分析与响应节点、网络管理中心和管理界面构成.

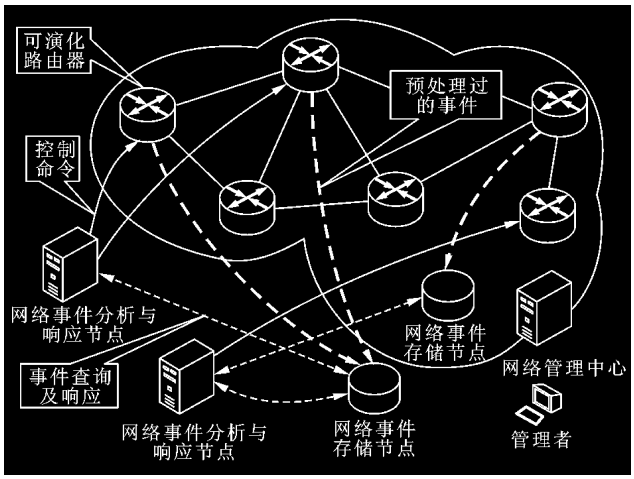

图3 可演化网络的体系结构图

2.1 可演化路由器

可演化网络的核心是可演化路由器，其控制功能、路由功能和管理功能根据环境和应用可动态地变化. 为了适应不断变化的网络应用需求和网络运行环境，可演化路由器的控制和管理必须具有高度的动态性.

可演化路由器的结构如图4所示，其主要构件包括本地节点管理器、执行环境、可加载的网络软件模块、事件收集模块、事件预分析模块、管理代理、路由模块等.

可加载的网络软件模块是可以变化的网络软件，通过它可扩展网络协议，提高服务和控制功能. 可演化路由器除了具有高速数据转发的硬件以外，还可部署软件来实现不同的控制功能，改善其QoS、拥塞控制、安全控制等. 通过灵活地部署网络软件模块，可演化路由器可以针对网络流量和网络环境的突然变化，灵活地修改自身的功能与控制参数，以适应更加广泛的网络应用和网络环境.

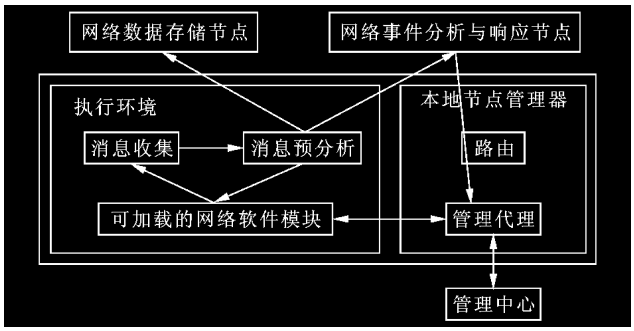

图4 可演化路由器的结构图

事件收集模块负责采集可演化路由器中的信息，包括部件和网络软件模块的运行状态和网络过滤器截获的各种数据包中的信息. 事件收集模块可采用不同的包过滤器和不同的过滤规则，过滤规则和网络事件的接收者可以由网络管理者和网络事件分析与响应节点动态地添加和删除.

事件预分析模块可对收集到的事件做预先分析，包括数据包的预分析与统计及特征提取，攻击事件的识别等. 预分析的结果将存储于网络事件存储节点，并由网络事件分析与响应节点做进一步的全局性分析. 通过对多个网络路由器的预分析，可以获得比较准确的分析结果. 如果仅通过本地的网络事件就可以在某种程度上下定论的话，那么事件预分析模块就可以直接对网络软件模块的相关参数进行调整. 如果发现网络软件模块中的算法通过调整后仍无法满足需求，就可以加载更适合当前网络需求的新算法. 通过这种局部反射机制，可以促进网络节点的进一步演化.

执行环境是运行可加载网络软件、事件收集、事件预分析等模块的程序执行环境，它具有动态加载、卸载、删除程序代码的能力. 用户可以根据网络控制需求，在执行环境中开发和部署新的网络软件模块，以达到改变可演化路由器特性的目的.

可演化路由模块在具备传统网络路由模块功能的同时，还可进行接插、变换，它不仅可以采用现有的路由协议，如RIP、OSPF、BGP等，也可以通过管理代理动态地加载新的路由协议.

管理代理是用来管理可演化路由器的. 网络节点中的所有网络软件、事件收集、事件预分析等模块都可在管理代理的管辖之下动态地加载和替换. 当网络节点需要某个新功能时，管理代理就部署新的功能模块.

本地节点管理器的主要功能是对可演化路由器的运行进行管理，它接收管理中心发来的管理命令并做出回应，具体功能包括：移动代码的下载、存储、安装、加载、维护及运行时的监控和管理；通过身份认证、数字签名、加密、基于角色的权限控制等手段，

实现移动代码的分发、加载、执行等环节的安全管理.

2.2　网络事件存储节点

网络事件存储节点的结构如图 5 所示. 该节点根据网络事件的功能特性、数据量大小、复杂程度、时间特性、访问特性和操作特性等存储网络事件,从而获得最佳的访问效率和时空性能. 网络事件存储节点提供了事件查询接口,允许网络事件分析与响应节点来查询网络事件.

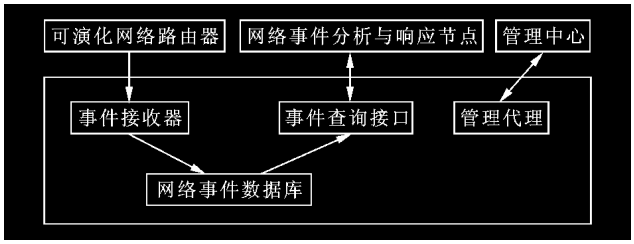

图 5　网络事件存储节点的结构图

在网络事件存储节点中,事件接收器接收其他路由器发来的事件,经过初步分析后存入网络事件数据库.

事件查询接口接收网络事件分析与响应节点的查询请求,返回符合查询的网络事件结果集.

网络事件存储节点可定期对事件数据库进行维护,删除旧数据,优化查询性能. 管理中心可以利用管理代理来管理网络事件存储节点.

2.3　网络事件分析与响应节点

网络事件分析与响应节点可以获取网络路由器采集到的相关网络事件信息和定期采集到的网络状态数据,通过统计、计算、分析,全面了解网络的具体运行情况,并对已知和潜在的网络攻击等行为做出分析和预警. 该节点可根据分析结果采取相应措施,并以软件模块的形式由响应模块直接部署于合适的网络路由器,由此达到调整网络路由器的目的,实现分布式的、主动的网络控制与管理.

网络事件分析与响应节点的结构如图 6 所示,其主要构件包括本地节点管理器、代理执行环境、分析模块、响应模块、事件接收器、本地事件数据库、知识库等. 在可演化网络系统中,这种节点的作用是控制、管理服务器.

事件接收器接收来自网络事件存储节点或其他网络节点(路由器)的网络事件,并将事件存入本地事件数据库,再依据这些数据的类型及标识将事件送给特定的分析模块.

本地事件数据库将网络事件按照事件的发送时间、发生地点、类型等参数有组织地进行缓存.

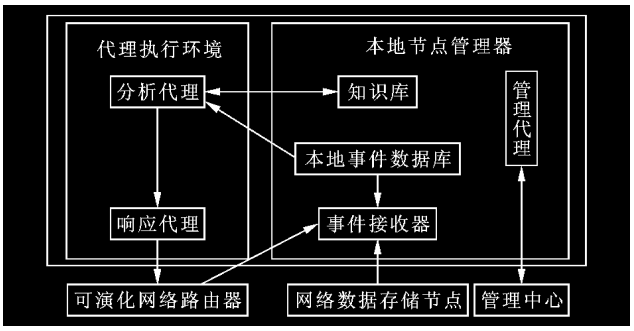

图 6　网络事件分析与响应节点的结构图

本地节点管理器的主要功能与可演化路由器中的本地节点管理器相类似,负责接受管理中心对网络事件分析与响应节点的管理.

代理执行环境应支持分析模块和响应模块,为了达到此目的,节点应具有动态加载新程序代码的能力. 可将需要动态添加、删除的分析模块和响应模块当作移动代理来处理,而这些移动代理能够动态地部署在各个网络事件分析与响应节点的代理环境之中. 用户可以根据网络控制需要,开发或购买新的网络事件分析模块和响应模块,并部署于网络事件分析与响应节点之中,从而达到动态管理、控制网络的目的.

知识库与分析模块相关,它为分析模块中的算法提供所需的知识信息. 本地节点管理器在更新分析模块之时,通过管理中心对知识库进行升级.

2.4　可演化网络管理中心

可演化网络管理中心[6]负责管理整个系统,包括管理代码及网络模块代码的动态分发,管理功能、界面的动态构造,以及系统的动态扩展. 该中心的结构如图 7 所示,其主要由管理服务器、代码服务器、应用模式库、被管对象集、界面生成器、注册管理器等构成.

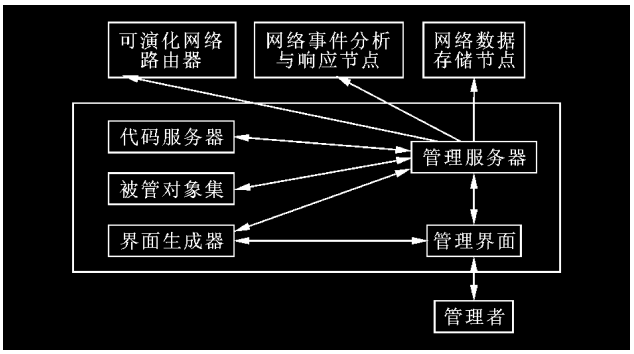

图 7　网络管理中心的结构图

管理服务器包括实现核心管理功能的 Web 服

务器、数据库服务器等，并为网管系统提供基础支持.

代码服务器用于存放管理功能代码和网络软件模块代码，并将不同的网络软件模块代码，以及网络事件分析模块、网络事件响应模块对应的管理代码发往相应的网络节点和网络事件分析与响应节点.

被管对象集可提供管辖对象的定义及描述，建立管理功能关联. 该集合采用类似管理信息库(MIB)的结构，与传统网管的 MIB 库兼容，并描述 IP、TCP、UDP、RSVP、ICMP、IGMP 等协议，描述网络的各种硬件设备，描述服务、事件，以及与系统应用相关的被管对象. 通过动态编译可以生成及扩展这种被管对象集.

界面生成器可动态构造用户的管理界面，即针对网络管理员、设备厂商、应用开发者提供的内容界面，根据网络软件模块的动态变化情况，随时生成相应的管理界面.

3 可演化网络原型及实现

基于以上模型和体系结构，依托现有 IP 网络，以移动代理和层叠网的形式实现了可演化网络的实验原型. 在可演化网络中，软件功能模块均由移动代理实现，基于主动网络的移动代理平台[7]支持各种网络软件模块，网络中的数据和控制包在各个移动代理之间可交互传递，而移动代理能动态部署，代理之间也可动态交互，由此实现了网络功能模块的动态部署、升级和演化，达到了整个网络演化的目的.

可演化路由器的编程、软件模块的动态部署和执行，是可演化网络要解决的核心问题，其关键技术阐述如下.

3.1 可演化网络的编程模式

可演化网络的演化能力取决于其动态可编程的能力. 可演化网络原型支持 2 种类型的程序：代理程序和服务程序. 这 2 种程序的功能如下.

服务程序 它是驻留在网络节点中长期运行的程序. 一旦该程序加载到某个节点并作为服务程序，网络数据包在该节点通过访问服务程序就可以获得相应的服务. 服务程序可以通过管理命令动态地加载到需要提供服务的节点上.

代理程序 它负责完成特定的传输任务或设置用户的传输参数. 代理程序可以在网络节点间移动.

在可演化网络原型中，服务程序和代理程序均依托于移动代理技术来实现.

可演化网络原型包括 6 种不同的可编程单元，即位于可演化路由器上的可加载网络模块、事件收集代理和事件预分析代理，位于网络事件分析与响应节点上的分析代理和响应代理，以及各个网络节点上的管理代理. 这 6 种可编程单元均可由开发者自由编写，再由网络管理员动态地加载到网络之中.

可加载网络模块 该模块一旦在可演化路由器上得以注册，将负责过滤数据包，并将过滤结果发到网络模块进行处理.

事件收集代理 它负责与可加载的网络模块进行连接，并收集在可加载网络模块中产生的各种事件.

事件预分析代理 对事件收集代理发来的事件进行基本的预先分析操作.

分析代理 对接收的消息进行详细分析.

响应代理 依据事件分析结果做出响应，即调整各模块与代理的参数，部署新模块、删除旧模块等.

管理代理 对各种功能模块实施管理.

3.2 程序代码的部署

在可演化网络中，各类代理及网络模块的部署场景有以下 3 种.

(1)管理员通过界面操作，从管理中心将可加载的网络模块、事件收集代理、事件预分析代理等部署到可演化路由器之中.

(2)管理员通过界面操作，从管理中心将分析代理和响应代理部署到具体的网络分析与响应节点之中，并将事件预分析代理、分析代理、网络事件存储节点连接起来，由此达到在线分析、响应网络事件的目的.

(3)通过分析计算，在无人干预的情况下，响应代理自动将新的可加载网络模块部署到可演化路由器之中，修正网络的功能，也可以自动部署新的事件收集代理和事件预分析代理，由此获得更详细、全面的事件数据.

代码的部署可采用主动和被动 2 种方式：以主动方式，则节点上的管理代理接收其他代理发来的代理程序的名称、下载地址等信息，然后主动地从指定代码服务器下载代码并安装到本地节点上；以被动方式，则管理代理直接接收其他代理发送来的移动代理程序并安装到本地节点上. 在可演化网络中，主要采用主动的方式来部署、扩展和更新网络的功能模块.

以主动方式部署代码的优点是：可操作的程序代码量比较大；代码的来源比较明确，有利于安全性检查，从而保障代码的完整性；可以较好地解决代码中各个程序块间的依赖关系，有效防止恶意代码.

3.3　可动态部署代码的执行环境

可演化路由器和网络事件分析与响应节点的各个软件功能模块,均依托于移动代理技术来实现.为了支持移动代理的部署与执行,本文在网络节点上实现了基于主动网络的移动代理执行环境平台[8-9],如图 8 所示.

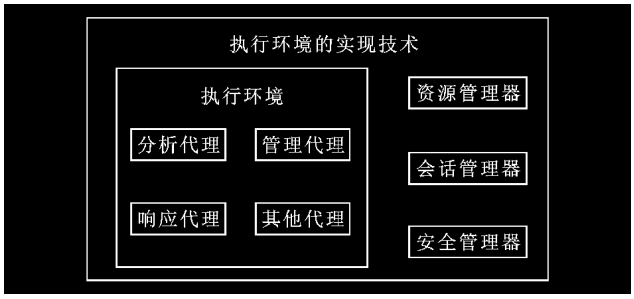

图 8　执行环境的结构图

移动代理的执行环境是一种程序执行环境,它为动态部署的管理代理、事件分析模块、事件响应模块的执行提供了必要的条件.在执行环境中,代码的划分和安全性是保证执行环境的重点.执行环境的主要功能包括接收移动代理,验证、运行、删除移动代理程序.

执行环境采用 Java 语言来实现.各个移动代理均直接继承一个移动代理的基类.通过扩展 Java 语言中的类加载器,可构造执行环境的代码加载器.对于每一个移动代理类,需生成一个移动代理执行环境,而在该执行环境中均采用独立的代码加载器,由此可将各个移动代理的实际运行环境分配到不同的空间之中.网络节点中的代码加载器可采用树型结构来组织,如图 9 所示.根代码加载器加载最基本的软件模块,如 Java 自带的类和加密解密算法的类等.通用代码加载器应加载可演化网络系统中用到的 API 类和移动代理基类.每个节点需设定一个特殊的执行环境(系统执行环境),以为网络节点中的系统软件提供执行环境,并负责加载系统管理程序

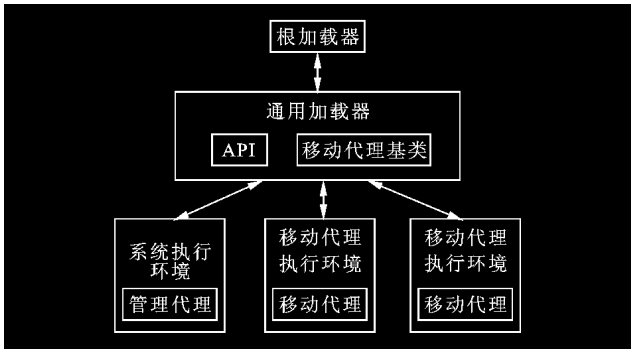

图 9　代码加载器的层次结构图

和管理代理,而在其他移动代理执行环境中可分别加载不同的移动代理.每一个类可以使用本类代码加载器及其父加载器加载的代码,即叶子节点可以访问父节点及根节点加载的代码,由此保证移动代理使用节点提供的 API 资源.叶子节点之间不能相互访问,这样才能保证移动代理之间不受影响,由此提高安全性.

3.4　移动代理的执行性能

可演化网络通过动态部署、执行程序可以改变网络的行为,但同时也带来了性能上的问题.在可演化网络中,移动代理的部署和执行虽不是经常发生的事件,但是其执行性能对网络的整体性能仍有显著影响.

如图 10 所示,移动代理在可演化网络节点中的执行过程可以分为 7 步,其时间消耗模型为

$$T_{ma} = T_{in} + T_{parse} + T_{createMAEE} + T_{loadMA} + T_{run} + T_{queue} + T_{out}$$

式中:T_{ma}表示移动代理通过网络节点的总时间;T_{in}表示移动代理导入网络节点时系统执行环境所用时间;T_{parse}表示移动代理包的解析时间;$T_{createMAEE}$表示系统执行环境为移动代理生成临时执行环境的时间;T_{loadMA}表示装载移动代理程序的时间;T_{run}表示移动代理程序的执行时间;T_{queue}表示移动代理执行完毕后至转发的等待时间;T_{out}表示包的发送时间.

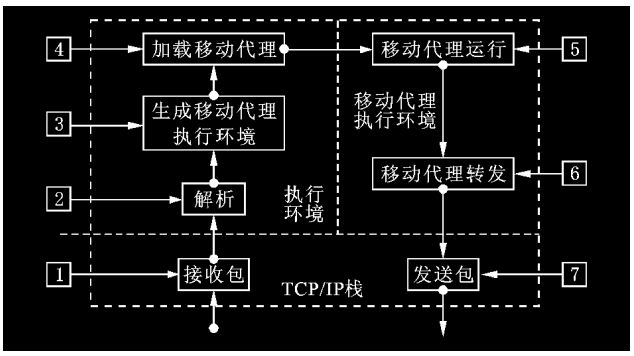

图 10　移动代理执行的 7 个步骤

上述 7 步所需时间可以分成以下 2 类.

被动性时间　指接收、存贮、转发包的时间.传统网络中的每个包的转发时间为

$$T_{ip} = T_{in} + T_{route} + T_{queue} + T_{out}$$

可以看出,它与图 10 的类似之处是都包括了 T_{in}、T_{queue}、T_{out},而不同之处是,传统网络节点还需要一个路由时间,即查找路由表的时间 T_{route}.

主动性时间　指引入主动性而产生的额外延时,包括包的解析、执行环境的生成、加载程序和程序执行的时间,即

$$T_{active}=T_{parse}+T_{createMAEE}+T_{loadMA}+T_{run}$$

可以看出，可演化网络节点与传统网络节点转发包的时间之差正是主动性时间

$$\Delta T=T_{ma}-T_{ip}=T_{active}-T_{route}$$

由此可见，影响可演化网络节点性能的主要原因是引入移动代理后需要解析包、生成移动代理执行环境、加载移动代理程序和运行移动代理程序的时间开销.

在原型系统中，可以采用2种移动代理程序来测试移动代理转发的主动性时间：Ping，即将Ping转发到下一个节点，功能是查找路由表；Ping＋Comp，它比Ping多了一个300次字符串加操作的计算时间. 表1给出了一次移动代理转发的平均主动性时间测试结果.

表1 部分主动性时间比较

测试程序	T_{parse} /ms	$T_{parse}\cdot T_{active}^{-1}$/%	$T_{createMAEE}$ /ms	$T_{createMAEE}\cdot T_{active}^{-1}$/%	T_{loadMA} /ms	$T_{loadMA}\cdot T_{active}^{-1}$/%	T_{run} /ms	$T_{run}\cdot T_{active}^{-1}$/%	T_{active} /ms
Ping	0.04	0.3	4.15	30.3	9.39	68.4	0.14	1.0	13.72
Ping＋Comp	0.04	0.2	4.15	19.5	9.39	44.2	7.67	36.1	21.25

从表1可以看出，对于很简单的移动代理程序Ping，$T_{createMAEE}$和T_{loadMA}占据了约98.7%的主动性时间，而对于较为复杂的主动代理程序Ping＋Comp，$T_{createMAEE}$和T_{loadMA}也占据了约63.7%的主动性时间，也就是说，为移动代理程序运行所做的准备工作消耗了较多的时间，这就是节点的性能瓶颈. 为了提高节点的性能，本文在原型系统中采用了以下优化技术.

快速激活及加载技术 通过优化类加载时的查找顺序，可提高加载效率，优化加载算法如图11所示. 对于与移动代理主程序同属于一个包的所有类，均先查找本地代码库，并直接加载主程序，这样就省去了查找父加载器甚至是根加载器的过程；对于非同一包的类，采用传统的类查找法和加载法；对于无法加载的类，则需要在远程代码服务器上进行查找. 这种快速的激活及加载技术可使移动代理主程序的平均加载时间从原来的9.39 ms减至6.13 ms，代理执行时的其他类的加载时间也得到了相应的优化.

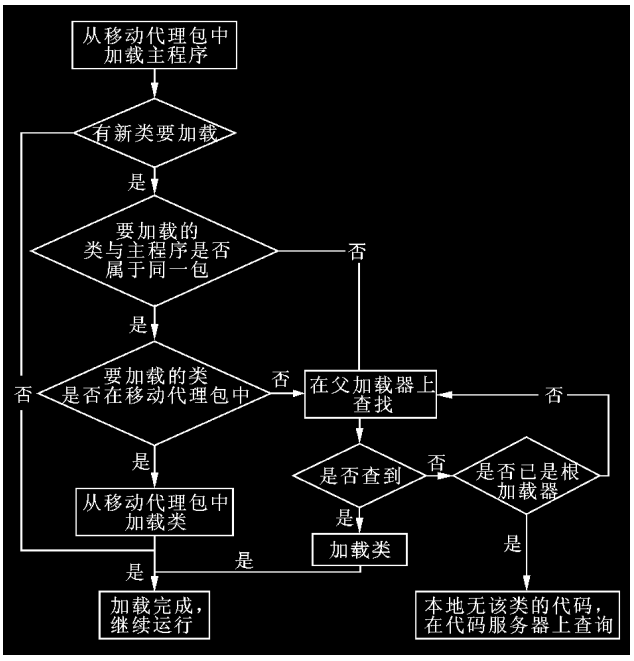

图11 优化的加载技术

执行环境池技术 执行每个移动代理均需要生成一个新的执行环境，而生成一个执行环境的准备工作需要消耗时间. 在最初的原型系统中，该时间达到了4.15 ms，而生成线程的时间约为2 ms. 为此，本文采用了执行环境池技术，即用空间换取时间. 具体做法是，在网络节点空闲时预先生成一定数量的移动代理执行环境，并放入执行环境池中. 当网络节点接收到移动代理的程序包时，便从池中选择一个空闲的执行环境，再通过一定的配置就可以装载、执行移动代理程序. 移动代理程序执行完毕之后，执行环境被系统自动回收，并可重复利用. 显然，选取一个未使用的移动代理执行环境比生成一个新的移动代理执行环境的速度更快. 在本文的原型系统中，该优化在整体时间上约节省了2 ms.

执行环境池的容量选择与性能有关. 池太大，其占用的内存空间就越大，并行运行的线程则较多，这样会影响效率；池太小，很多移动代理程序则无法获得执行环境，等待队列就会加长，这样会使丢包率增加. 本文采用动态的方法来控制池的大小，即当池长期空闲时逐步删除部分执行环境，减小执行环境池；当池使用频繁时逐步生成新的执行环境，加大执行环境池. 当然，执行环境池包括内存、CPU等多种资源，它与传统意义上的包队列有本质区别，所以池的

管理仍然是一个需要继续研究的问题.

4　可演化网络的应用实验

下面以 2 种典型应用场景来验证可演化网络的功能及有效性.

第 1 个应用是自适应的 BLUE 算法[10],即在现有的算法中添加自适应模块,解决算法参数难以设定的问题. BLUE 算法是一种典型的主动队列管理算法,其思想是根据丢包和链路空闲事件进行拥塞控制. BLUE 具有一个概率变量 p_{mark},当数据包进入队列时,根据 p_{mark} 来标记数据包. 如果队列溢出而导致连续丢包,则通过增加 p_{mark} 来增加发送拥塞通知信息的概率;如果出现了空队列或链路空闲,则减小 p_{mark}. BLUE 算法虽然采用了 2 个参数 Δ_in、Δ_de 来表示 p_{mark} 增大和减小的幅度,但仍然存在参数设定困难的问题,因为 Δ_in、Δ_de 的不同取值对平均队列长度的性能影响较大.

采用 ns2 软件进行了模拟实验,其网络拓扑如图 12 所示. 图中,发送和接收节点数均为 100,每对节点之间生成一个 TCP 连接,节点 1 和节点 2 之间是瓶颈链路. 设 $\Delta=0.000\ 25$,$\Delta\_in=\Delta N$, $\Delta\_de=10\times\Delta N$,当 N 分别取 1～16 时,测得的稳定状态下的平均队列长度如图 13 所示. 从图中可知,不同的参数设置对算法性能的影响非常大,当 $N=10$ 时,算法性能最好,队列的平均长度较短,抖动也较小. 然而,当系统的其他参数发生变化时,该最佳取值范围不复存在了. 由此可见,固定参数的 BLUE 算法难以适应不同的环境.

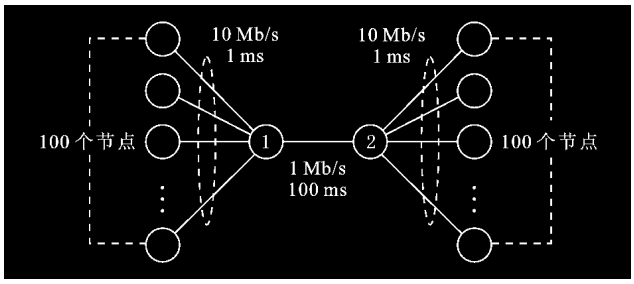

图 12　模拟实验的网络拓扑图

针对算法参数的选择问题,本文为 BLUE 算法添加了一个自适应调整算法 A. 算法 A 可动态调整 Δ 来增加 BLUE 算法的自适应性,但并不修改其他内容. 算法 A 如图 14 所示.

算法 A 的主要思想是:如果队列过长,增加 Δ,这样 p_{mark} 会较快增长,但为了避免 p_{mark} 增长过快,在 p_{mark} 大于 0.5 以后,不再增加 Δ,以防止 p_{mark} 过度

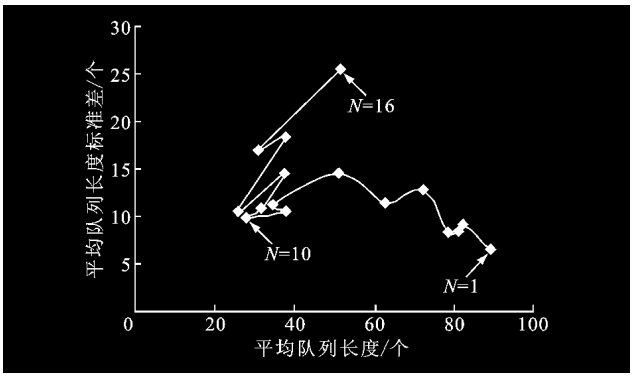

图 13　平均队列长度和方差与参数 N 的关系

```
Enque:
if (q_avg>q_limit/3.0)
  if (p_mark<0.5)
     Δ*=1.005;
     if (Δ>maxΔ)  Δ=maxΔ;
     p_mark*=1.005;
Deque:
if (q_avg<q_limit/3.1)
  p_mark/=1.000 5;
  Δ/=1.005;
  if (Δ<minΔ)  Δ=minΔ;
```

图 14　自适应的调整算法 A

振荡;如果队列过短,减少 Δ,当然也要保证 p_{mark} 减小的速度小一些. 自适应 BLUE 算法的实验结果如图 15 所示. 从图中可以看到,在给定任意一个很小的初始 Δ 后,p_{mark} 值迅速攀升到 0.7,随后在 0.45 附近小幅振荡,并在性能最好的参数下小幅振荡. 在平均队列长度方面,自适应 BLUE 算法的结果与 BLUE 算法在固定参数 $N=10$ 时的情况基本相同,见图 16.

由此可见,加入算法 A 可以在较短的时间内将 BLUE 算法调整到接近最优的状态,基本上解决了手工调节参数带来的问题. 其主要原因是,算法 A 采用了可演化网络的自适应思想,为网络算法带来了自我调整的能力.

第 2 个典型应用是路由器的队列管理算法的动态切换,在这里重点考察了网络的公平性问题. 队列管理算法 Tail-Drop 计算简单,但会影响网络的公平性. 某些恶意的、对丢包不做响应的 UDP 流会严重影响 TCP 流的性能. SFB 算法[11]试图通过鉴别、控制非响应流来保证响应流的正常使用. SFB 算法采用了多次 Hash 的鉴别算法,只需要花费 $O(NL)$ 的空间就可以处理 N^L 个流,也具有较好的公平性.

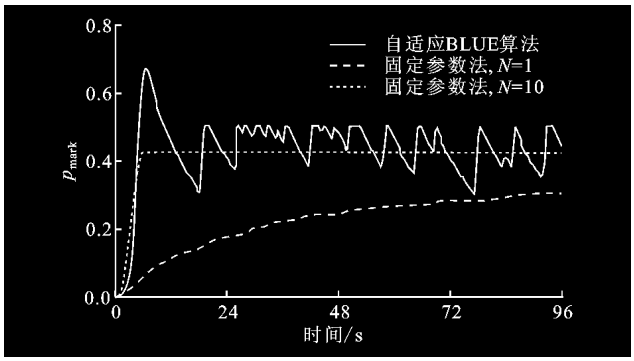

图15 p_{mark}随时间变化的关系

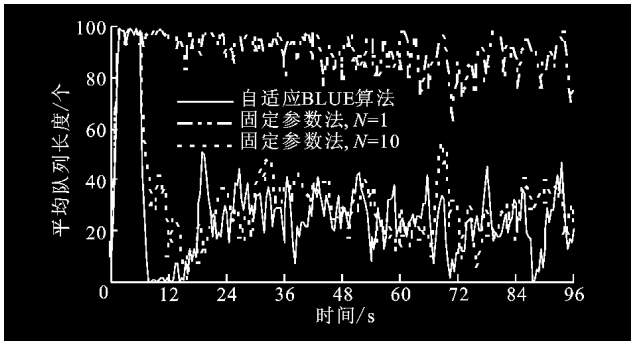

图16 队列长度随时间变化的关系

但是,其对每个包需进行 L 次 Hash 运算,这明显增大了包的延时,也耗费了较多的 CPU 资源.

事实上,任何一种队列管理算法都只能适用于一定的环境,如果能在网络环境与状态发生变化时及时更换不同的队列管理算法,则可以更好地管理路由器中的队列. 在实验中,用 TCP 来模拟响应流,用 UDP 来模拟非响应流,用一个包检测程序来统计一段时间内的 TCP、UDP 流量. 如果 UDP 流量达到了一个较高的水平,启用 SFB 算法来限制该流量;如果 UDP 流量回落到一个较低的水平,则停用 SFB 算法,此处采用最简单的 Tail-Drop 算法. 这样,开销较大的 SFB 算法只在网络环境较差的时候才被激活.

包检测程序(算法 B)如下:如果队列已满,且队列内 UDP 包的个数是 TCP 的 2 倍,则启动 SFB 算法,如果 5 s 内 SFB 算法未检测出非响应流,则启动 Tail-Drop 算法.

在模拟实验中,仍然采用图 12 所示的网络拓扑,瓶颈链路带宽设为 1 Mb/s,延时设为 10 ms. 在 0～1 s 中的随机时间点启动 100 个 TCP 流和一个 CBR 非响应流,在 85 s 时停止 CBR 流. 那么,实际测得的 CBR 流量随时间的变化如图 17 所示. 从图中可看出,算法 B 很快检测到了 CBR 流,并且对每次检测到的流进行了 2 s 的有效屏蔽. 在第 86 s 之后,由于 CBR 流的残余流量已经很小,则其被当作正常数据包. 再经 5 s 之后,由于未检到非响应流,队列管理算法在第 91 s 时切换到 Tail-Drop 算法. 在第 5 s～第 75 s 的各个 TCP 流的平均速率如图 18 所示,通过计算得知 TCP 流的平均速率为 9.98 kb/s,而 CBR 的速率为 3.77 kb/s,可见算法有效地抑制了非响应流的流量,保证了 TCP 流的正常运行.

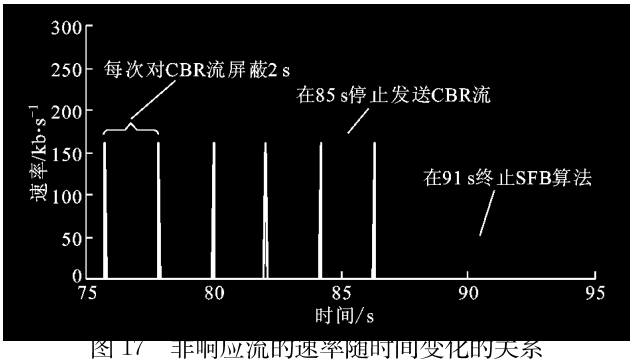

图17 非响应流的速率随时间变化的关系

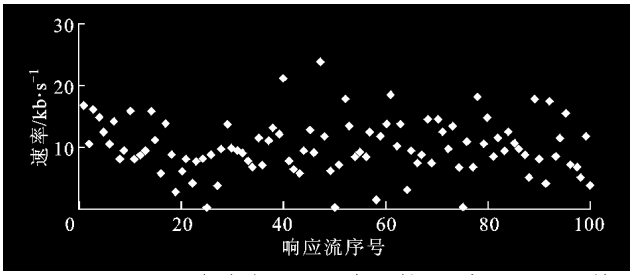

图18 100 个响应流的速率比较(速率的理论平均值为 10 kb/s)

5 结 论

随着互联网规模的不断扩大以及网络承载的业务和应用的日趋多样化,互联网的弱点和不足也越来越明显,甚至成为制约发展的不利因素. 针对这些不足,本文提出了一种新的可演化网络模型,它支持网络功能的动态扩展、网格协议的动态更新和网络控制管理的动态调整,使得网络呈现出可演变、可进化的特点.

基于移动代理技术实现了一种可演化网络的原型,它揭示了可演化网络实现中可能遇到的性能和安全性等问题,探索了问题的解决途径和方案,并开展了可演化网络的应用实验. 原型系统以及典型应用的模拟仿真表明,可演化网络对高度变化的环境能够做出快速反应,并自动地进行状态演化,更好地满足了服务质量的需求.

下一步的工作是,继续研究可演化网络的实现

策略,例如基于硬件支持的、采用并行结构的可演化路由器,并在可演化网络上开展更多的应用实验,进一步验证可演化网络的有效性和实用性.

参考文献:

[1] TENNENHOUSE D L, SMITH J M, SINCOSKIE W D, et al. A survey of active network research [J]. IEEE Communication Magazine, 1997, 35(1): 80-86.

[2] ANDERSEN D G, BALAKRISHNAN H, KAASHOEK M F, et al. Resilient overlay networks[C]// Proceedings of The 18th ACM Symposium on Operating Systems Principles. New York, USA: ACM, 2001: 131-145.

[3] ZAMBONELLI F, MAMEI M. Spatial computing: an emerging paradigm for autonomic computing and communication [M]// Lecture Notes in Computer Science 3457. Berlin, Germany: Springer, 2005: 44-57.

[4] YAMAMOTO L, TSCHUDIN C. Experiments on the automatic evolution of protocols using genetic programming [M]// Lecture Notes in Computer Science 3854. Berlin, Germany: Springer, 2006: 13-28.

[5] KAPPLER C, MENDES P, PREHOFER C, et al. A framework for self-organized network composition [M]// Lecture Notes in Computer Science 3457. Berlin, Germany: Springer, 2005: 139-151.

[6] LUAN Zhongzhi, QIAN Depei, ZHANG Xingjun, et al. A novel model and architecture on NMS-dynamically constructed network management [M]// Lecture Notes in Computer Science 2834. Berlin, Germany: Springer, 2003: 398-403.

[7] 陆月明, 钱德沛. Softnet: 一个基于移动代理的主动网络 [J]. 计算机学报, 2001, 24(11): 1310-1314.
LU Yueming, QIAN Depei. Softnet: an active network based on mobile agents [J]. Chinese Journal of Computer Science, 2001, 24(11): 1310-1314.

[8] 陆月明, 钱德沛, 徐斌, 等. 基于可编程移动软设备的主动网络执行环境 [J]. 软件学报, 2002, 13(2): 227-234.
LU Yueming, QIAN Depei, XU Bin, et al. Execution environments for active network based on programmable mobile soft devices [J]. Journal of Software. 2002, 13(2): 227-234.

[9] 徐斌, 钱德沛, 张文杰, 等. 主动网络管理代理的执行环境 [J]. 计算机研究与发展, 2002, 39(11): 1478-1483.
XU Bin, QIAN Depei, ZHANG Wenjie, et al. Active network management agent execution environment [J]. Journal of Computer Research and Development, 2002, 39(11): 1478-1483.

[10] FENG W, KANDLUR D, SAHA D, et al. BLUE: a new class of active queue management algorithms, CSE2 TR2387299 [R]. Ann Arbor, MI, USA: University of Michigan, 1999.

[11] FENG W, KANDLUR D, SAHA D, et al. Stochastic fair BLUE: a queue management algorithm for enforcing fairness[C]// Proceedings of the Conference on Computer Communications. Piscataway, NJ, USA: IEEE, 2001: 1520-1529.

(编辑　苗凌)

西安交通大学教师和博士生荣获2008年度IEEE通信学会“通信系统集成与建模”最佳论文奖

2008年5月22日,西安交通大学智能网络与网络安全教育部重点实验室的博士生秦涛、王平辉和他们的指导教师:电信学院系统工程研究所所长、长江学者管晓宏教授以及中国教育科研网西北中心主任李卫副教授获得2008年度IEEE通信学会“系统集成与建模”最佳论文奖.他们合著的论文“Dynamic Features Measurement and Analysis for Large-Scale Networks”是IEEE通信学会通信系统集成与建模委员会在第13届IEEE International Workshop on Computer-Aided Modeling, Analysis and Design of Communication Links and Networks选出的惟一最佳论文.该会议是2008年度国际通信大会(International Conference on Communications)的一部分,来自加拿大、意大利、英国、日本、葡萄牙、中国等国家和地区的学者参加了会议.会议主席向获奖人颁发了IEEE通信学会奖牌.

(来源:西安交大科技在线 http://www.xjtust.com)