内核级透明代理TPF的设计与实现*)

蔡圣闻　黄　皓　谢俊元

(南京大学计算机科学技术系　南京210093)

摘　要　代理防火墙存在着性能不高、自身安全性无法保证、可扩展性差等诸多弱点。针对这些问题，本文提出了在定制安全操作系统基础上，将代理程序与OS内核协议栈一体化设计及代理程序分层实现的思想。介绍了报文分类标记、策略树机制、协议栈快速通道等实现内核级代理的关键技术，以及在安全OS内核中实现透明代理防火墙TPF的过程。TPF在性能、自身安全性和易扩展性方面较之传统代理防火墙有了显著的提高。

关键词　内核代理，安全操作系统，策略树，协议栈快速通道，自身安全性

A Dedicated Security OS Based Transparent Proxy Implemented In Kernel

CAI Sheng-Wen　HUANG Hao　XIE Jun-Yuan

(Department of Computer Science and Technology, Nanjing University, Nanjing 210093)

Abstract　In this paper, disadvantages of traditional proxy firewall are analyzed, and then, TPF, a dedicated security OS based transparent proxy applied in kernel, is introduced. The conceptions and key technologies of TPF, such as integration in security OS kernel of proxy code, interruption slack off, packet classified and tagged, policy tree mechanism, protocol stack fast channel, layered software architecture, etc, make a remarkable improvement in autoimmunity, performance and expansibility of the proxy firewall.

Keywords　Kernel proxy, Security OS, Policy tree, Protocol stack fast channel, Autoimmunity

1. 前言

防火墙是目前技术最为成熟、应用最为广泛的网络安全保护手段。防火墙产品大致可以分为三类，即包过滤防火墙、代理防火墙和兼具包过滤和代理功能的混合型防火墙。包过滤防火墙通常在三层以下实现，其实现简单且性能较好，但因不能对应用协议内容进行检查，以及无法防范穿透防火墙攻击、应用协议欺骗等攻击行为，安全防护能力有限而不能满足当前网络安全形式发展的需要。传统的代理防火墙在应用层实现，能够对应用协议和数据进行过滤检查，具有较高的安全保护能力，但受实现技术的制约，存在着诸多不足。主要表现在系统资源消耗大，性能较差，并发连接数不高，通信时延较长，支持的应用协议数量有限，扩展困难。而且往往代理防火墙作为独立的应用程序开发，运行在普通操作系统至上，导致防火墙自身安全性无法得到保证。

针对传统代理防火墙的诸多弱点，本文提出了采用定制安全操作系统，将代理防火墙和操作系统内核进行一体化设计的思想。通过B1级安全操作系统和对协议栈的安全加固提高代理防火墙本身的抗攻击能力。通过在定制操作系统中引入无效中断抑止机制，对协议栈的优化和建立支持透明代理的协议栈快速通道、代理主程序的内核线程方式实现等措施，大幅度提高透明代理的吞吐量和并发连接数，降低了代理的处理时延。在代理程序结构方面提出了分层实现的思想，将支撑程序、协议报文解析和内容安全区别开来，形成一个开放的系统构架，便于代理支持的应用协议数量的扩充和渐进式开发。

本文第2节分析了传统应用层代理和透明代理的实现技术及存在的弱点，并介绍专用安全操作系统ESK的设计及其对内核透明代理的支持等相关背景知识。在第3、4、5部分分别介绍了内核级透明代理防火墙TPF的设计思想，程序结构与工作流程以及实现TPF的关键技术。第6节给出了TPF的一组性能测试数据。最后对基于定制操作系统的内核级透明代理的特点做了简要总结。

2. 应用代理的实现及问题

传统的代理防火墙的工作原理如图1。客户端需要将代理网关设置为代理防火墙。在客户端访问服务器时，先和代理防火墙建立连接(图1中连接1)，并将访问请求发送到代理防火墙。代理防火墙根据安全策略对客户端进行身份认证和访问许可检查，若检查通过，则代理程序和目的服务器建立连接(图1中连接2)，通过连接2将客户端的访问请求发送到目的服务器。其后，客户端和目的服务器通过代理防火墙进行数据交换，代理程序对交换的数据及访问请求进行安全检查。

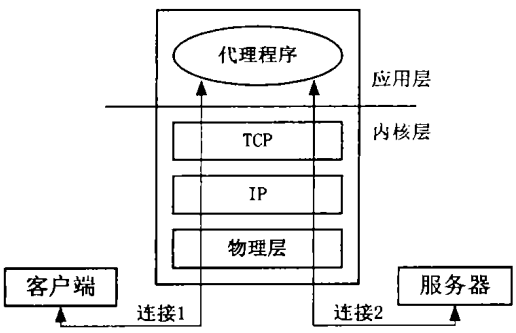


图1

传统透明代理防火墙的典型实现如图2。透明代理位于内外网的网关位置，内外网间的通信必须经过代理防火墙转发。

*)本文得到国家863计划资助(课题编号2001AA144010)和江苏省软件和集成电路专项《高安全高性能的网络防火墙》项目资助。**蔡圣闻**　博士研究生，主要研究方向为计算机网络、信息安全。**黄　皓**　教授，博士生导师，主要研究方向为信息安全，分布式系统。**谢俊元**　教授，博士生导师，主要研究方向为人工智能，信息安全。

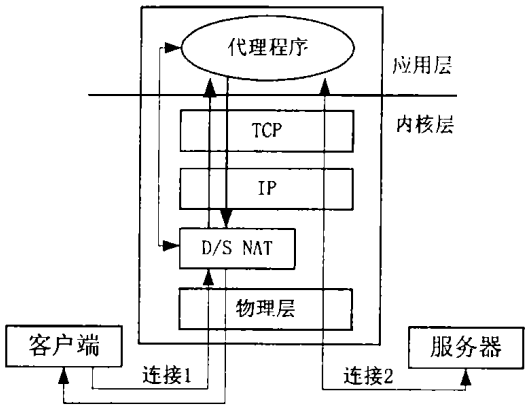


图2

透明代理工作时，客户端无需将透明代理防火墙设置为应用网关。客户端直接向目的服务器发起连接请求，连接请求及其后的通信数据包流经代理防火墙时被运行于内核协议栈介于链路层与IP层之间的透明代理支持程序截获。该支撑程序根据安全策略决定是否对该连接做应用协议安全检查。对需要交给代理程序做应用协议安全检查的连接的数据包进行目的地址、端口转换——将目的地址转换为本机地址，将目的端口转换为代理程序监听的端口，同时记录原始连接信息。然后将数据包交给IP层处理。这样客户端发送到服务器的数据包均被重定向到代理程序。代理程序在收到一个连接请求后，向内核中的支持程序查询该连接请求的目的服务器地址，并根据安全策略对连接合法性进行检查。若检查通过，则代理程序与目的服务器建立连接2。透明代理程序通过连接1和连接2为客户端和目的服务器交换数据，同时进行应用协议安全检查。为使客户端程序运行正常，代理程序发送到客户端的数据包，在IP层与链路层间做源地址转换——将源地址从透明代理防火墙本机地址转换为服务器的地址。

传统代理程序在应用层利用多线程或轮询机制实现，以提供并发服务。代理防火墙的应用层实现机制存在诸多的弊端。在性能方面，由于代理程序需要为每一个合法访问连接会话维持两条连接，系统资源消耗很大。且应用协议内容安全检查以外的开销过多，造成代理防火墙的处理效率不高。因系统中断具有比应用层进程更高的优先级，在网络访问量较大时，系统频繁地执行中断调用，代理程序难以对接收到的数据包进行及时处理，且不能对无效中断进行抑制，导致丢包率上升，CPU资源浪费，整体性能较差[2]。同时，受操作系统对进程拥有的套接字及文件描述符数量的限制，代理支持的并发连接数非常有限[3,4]。

在代理防火墙的自身安全性方面，代理程序在应用层实现，无法为操作系统提供保护，同时防火墙自身安全完全依赖于操作系统本身。当应用代理防火墙运行于非安全商用操作系统之上时，自身安全无法得到保证[5]。此外，代理防火墙对访问连接请求的合法性判断在应用层进行，在一条连接请求被判定非法之前，客户端与代理防火墙的连接已经建立，此时再切断该连接，代价太高，且易使代理防火墙遭受拒绝服务(DOS)攻击。传统代理防火墙的实现机制对其结构的开放性和可扩充性存在一定的制约。对每一种应用协议均需要相应的代理程序程序进行处理。应用代理程序的开发与应用协议本身关联非常紧密，不利于应用协议代理数量的扩充，新的应用协议代理的开发和升级发布代价较高[6]。

3. TPF的设计思想

内核透明代理防火墙TPF的主要设计思想是，通过对协议栈处理流程及传统代理防火墙缺陷的分析，将操作系统和代理程序作为一个有机整体，在解决代理防火墙自身安全性问题，提高处理性能和结构的可扩充性的基础上，实现代理的应用层安全功能。

TPF从两方面解决代理防火墙的自身安全问题：一是采用安全程序设计方法学，减少网络数据包处理程序(主要是协议栈和代理程序)本身的安全漏洞；二是提高操作系统安全性、安全保护能力及免疫能力，确保在黑客利用漏洞对防火墙攻击时，危害得到有效抑制，不会扩散且不会破坏安全策略的机密性和完整性。

为了提高处理性能，TPF利用报文分类及标记技术来区分网络数据包的应用协议类型和所属的会话连接。利用会话状态信息表记录每条会话连接的状态信息。消除了代理并发连接数对套节字描述符的依赖。利用内核线程方式实现代理主程序，消除了内核与应用层间的数据拷贝。通过特殊任务队列将协议栈和代理程序紧密结合起来。利用透明代理对网络数据报文处理的特性，建立协议栈快速通道，加快网络报文在协议栈中的处理过程。利用报文的有序重组机制，保证代理程序对会话连接的有序处理。

此外，运行于代理防火墙主机之上的任务单一，除配置程序外，系统主要是对网络数据包的处理和检查。因此，可以根据任务的特点，对操作系统的调度、中断等机制进行优化，提高运行特定任务时的性能。

对于代理的可扩充性问题，TPF引入了分层结构，将支撑程序，协议报文解析和内容安全区别开来，形成一个开发的系统构架，便于代理支持的应用协议数量的扩充和渐进式开发。

4. TPF的结构及工作过程

内核透明代理防火墙TPF由远程控制界面程序、运行于客户机的认证客户端程序、运行于防火墙主机应用层的认证服务器程序和配置管理程序、内核代理程序和OS内核与协议栈支持程序等六个部分组成。其总体结构如图3。

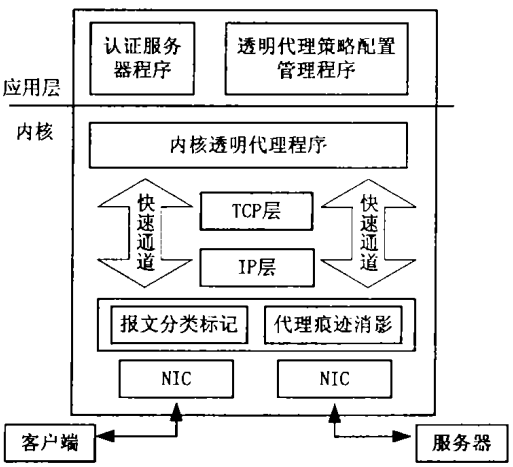


图3

认证服务程序与认证客户端程序协作完成对客户端身份(客户机IP与当前访问用户绑定信息)的认证，并将认证结果交给内核代理程序的用户认证模块。用户身份认证模块根据认证结果完成对当前连接是否为身份认证合法连接的检查。

策略配置管理程序将管理员经由管理界面程序发送来的安全策略翻译后组织成静态策略树的形式交给内核协议栈中的报文分类程序。报文分类程序将静态策略树转换为动态策略树，并依据策略树的内容对网络报文进行分类标记。

内核支持程序包括报文截获程序、报文分类标记程序、代理痕迹消影和协议栈快速通道。其负责应用协议报文的标记和策略查找，并通过协议栈快速通道将该报文和其对于安全策略传递给内核代理程序。若在应用协议网络报文到来时，内核代理程序为睡眠状态，则将其唤醒并在当前软中断结束后即调度其运行，以对网络报文进行及时处理。

内核透明代理程序由一个与CPU绑定的内核线程实现。其维护一个网络数据报文接收队列和一个记录当前所有连接状态信息的队列。当新的数据报文到达时，代理程序根据报文被标记的代理类型和所属的连接，调用应用协议报文分析程序对报文进行解析。对解析出来的语法单元和数据调用应用协议内容安全进行合法性检查和内容过滤。根据检查的结果进行数据转发、终止或重定向连接及日志记录等动作。

5. 关键技术与实现

内核透明代理防火墙的设计和实现包括专用安全OS、内核支撑机制和代理程序三部分。

5.1 专用安全操作系统ESK

ESK(Embedded Security Kernel)是在B1级安全增强操作系统SoftOS基础上设计开发的，满足网络安全工具特殊需求专用操作系统内核。

SoftOS是在Linux内核2.4.x基础上开发的服务器专用安全操作系统。它在操作系统安全方面具有以下特性：采用了内核安全保护技术、安全存储器访问控制技术、进程通信和文件系统的自主和强制访问控制。提供对主体、客体安全标记、禁止客体重用、隐蔽信道检测、基于策略的强制访问控制、数据完整性保护和特殊文件(如安全程序文件、安全策略配置文件等)保护功能[7]。

ESK在继承SoftOS的B1级安全特性的基础上，对系统进行了合理的剪裁，剔除了不必要的应用服务和内核功能(如虚拟文件系统，等等)，保留了一个能支持内核透明代理及其管理程序运行的最小完备集。同时，对网络协议栈进行了安全加固，消除了已知的各种协议栈安全漏洞，使其能够有效地抵御针对协议栈的攻击。此外增加了自身安全威胁检测及反馈机制。在发现遭受安全威胁时，能够及时报警并根据威胁类型采取调整策略、关闭服务等针对性措施，防止威胁的加剧和扩散。这些措施极大提高了ESK自身安全保障能力和可靠性，并能够确保即使是在黑客攻击成功的情况下，也不可能修改系统的安全策略，或以当前主机作为进一步攻击的跳板。ESK在一定程度上解决了操作系统与安全程序间"who protect who"[6]的问题。对ESK的详细介绍请参看文[8]。

5.2 内核支撑机制

可配置的调度策略及无效中断抑制 运行于网络安全工具专用操作系统之上的任务有限而且确定(对透明代理防火墙而言，任务有两类，一是安全管理和策略配置，二是对网络数据流的安全分析和处理)，因而可以根据任务的性质选择合适的操作系统调度周期以及对调度算法进行优化，以提高系统的处理性能。

防火墙作为网络处理设备，同样存在中断过载问题(interrupt overhead)。当网络流量较大时，系统被频繁中断，接收到的数据包因系统无法及时做进一步的处理而丢弃，网络吞吐率急剧下降。为此，在ESK中引入可配置的调度策略支持及无效中断抑制机制。

在透明代理防火墙TPF1.0的开发过程中，我们根据大量性能测试数据的统计分析，提出了一个针对特定硬件系统的最优化调度和中断抑制算法[8]。算法的主要思想为：

策略配置程序、网络硬中断、软中断和内核代理程序具有同等优先级；

硬中断、软中断均以原子方式调度，即调度程序不打断一次已响应的硬中断和已调度运行的软中断的运行；

调度程序监控硬中断、网络处理软中断和代理进程的CPU占用率，以及网络报文接收队列，代理接收报文队列和网络报文发送队列的长度。当队列长度超出设定阈值，且硬中断频率较高时，对中断作适度抑止。

优化的调度策略和中断抑止显著地改善了防火墙系统网络处理性能，特别是能够确保出现网络拥塞时的吞吐率。

网络报文分类及标记 为了便于安全程序对网络数据包的处理，ESK在链路层提供网络报文的分类及标记功能(图3)。安全程序可以根据需要，定义标记类型和注册报文分类函数。对于透明代理而言，可以根据定义于策略树(见5.3节)中的安全策略，对网络数据报文的源地址、目的地址、源端口、目的端口及应用层协议等进行合法性检查。并根据检查的结果将报文标记为丢弃(DROP)、转发(FORWARD)或交给代理程序(TOPROXY)，同时注明该报文的代理类型(如http代理，ftp代理等等)，所属连接和对该报文的应用协议安全检查策略。标记、所属连接、代理类型、应用协议内容安全策略及路由信息(在动态策略树生成时，查找路由表，将源地址和目的地址的路由信息填写在策略树中，避免对同一连接的每个网络数据包都进行一次路由查询，以节约系统资源)同时被记录在该报文的sk_buff[4]结构中。标记为TOPROXY的报文可以通过协议栈快速通道(以下介绍)传递到内核中的代理程序做进一步的安全检查。链路层的报文分类和标记功能可以在第一时间阻断非法连接，避免协议栈和代理程序对其做进一步的处理，节省了系统资源，同时也使得代理程序可以专注于应用协议内容安全。

非法连接阻断及DOS过滤 ESK在链路层提供DOS(Denial of Service)攻击检测和攻击源地址过滤功能。对被标记为DROP的报文做丢弃处理。对内核透明代理程序依据应用层安全策略判定为非法的访问，可短时间内在链路层阻断类似连接的建立，既减少了系统对违反应用协议安全策略访问的处理开销，又可避免黑客利用此类非法访问对代理防火墙进行拒绝服务攻击。

协议栈快速通道 对目的地址不为本机的网络报文，协议栈对其做路由转发处理。为了将客户段和目的服务器间通信的数据报文交由内核透明代理程序处理，我们在ESK协议栈中引入了快速通道机制。快速通道只对接收到的网络报文做差错检查，和有序重组(以下介绍)。对标记为TOPROXY的报文，在经过快速通道后，被加入到内核代理的报文接收队列中，而不经过系统协议栈的处理。对经过代理程序检查合法的网络报文，同样经由协议栈快速通道直接挂接到发送设备的输出队列。此时，快速通道所做的工作只是根据sk_buff结构中的路由信息选择正确的发送设备。

网络数据包的有序重组 为了保证透明代理程序能够对通信内容进行完整、有序的分析，必须对接收到的报文根据序列号(传输层协议为TCP的报文)或应用协议的规范(传输层协议为UDP)对报文进行有序重组。为此，ESK中引入了网络数据包的有序重组功能。在网络数据包经协议栈快速通道加入透明代理的接收队列时对接收队列做有序排列。

5.3 代理程序

策略树机制 策略树是透明代理防火墙TPF安全策略的组织形式。

通常，防火墙的安全策略是一个以一维表形式存储的优

先级有序规则集。安全策略检查表现为从规则集第一条规则开始的顺序匹配查找，直至找到一条与当前检查的数据报文或连接信息(如源、目地址，协议，端口等等)相匹配的规则。规则表的组织形式和有序查找限制了防火墙性能的提高。特别是在规则表规模增大时性能下降尤为明显。

TPF 将安全策略检查要素(如地址、协议、端口、有效时间等)在内核中组织成改进的"Grid of Tries"[9]树形式的动态策略树。消除了多条安全策略间的优先级限制。安全策略查找表现为走一条从策略树根节点到叶节点的路径。

策略树的节点内容除检查要素外，还包含一些减少重复操作的辅助信息，如在地址节点中包含了到该地址的路由信息等。一个策略树的示意如图4。

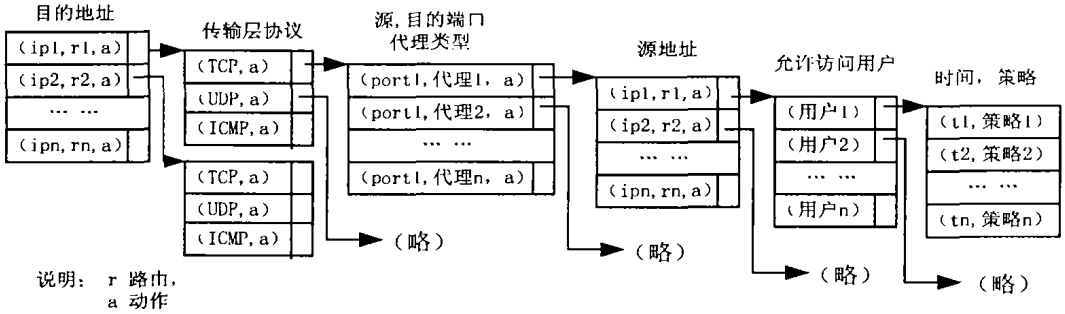


图4

会话状态信息队列 在 TPF 中，我们用会话状态信息队列取代传统代理防火墙中由客户机-防火墙通信套节字和防火墙-服务器通信套节字组成的套节字对。客户机向服务器发出的每条连接的信息记录在会话状态信息队列中。连接状态信息包含如下内容：

应用层协议 如 HTTP，FTP 等；传输层协议 TCP 或 UDP；源地址、源端口、目的地址、目的端口；应用协议内容安全策略指针；客户机发出报文接收队列；服务器发出报文接收队列；连接闲置时间；协议分析程序跳转状态；统计指标，如敏感词汇出现次数等；用户，通信数据量，连接建立时间，结束时间等审计信息。

当一次应用协议连接会话的首个数据报文达到时，内核代理程序为其建立一个会话状态信息节点，并将该节点加入会话状态信息队列。属于该次会话的所有数据报文在有序重组后加入报文接收队列，有应用协议分析程序进行处理。

会话状态信息队列机制使 TPF 消除了传统代理防火墙对操作系统套接字描述符的依赖，突破了代理进程拥有的套节字描述符数量对代理支持的并发连接数的限制。同时 TPF 无需为一次应用会话建立两条连接，消除了内核与应用层间的两次数据拷贝，节约了存储资源和系统处理开销。

代理的分层结构 TPF 的内核代理程序采用分层结构实现(图5)。

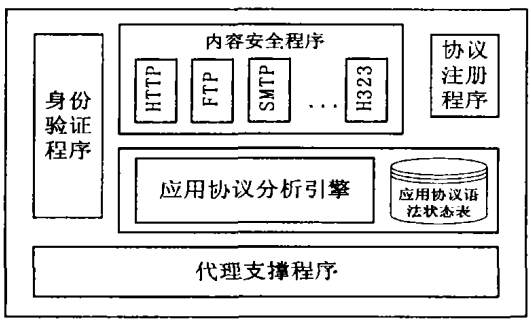


图5

支撑程序是整个内核代理程序的基础，负责接收和发送数据、建立和维护应用协议连接会话状态信息、用户身份认证、调用协议分析程序对会话内容进行检查并根据检查结果对会话连接执行控制动作、在会话结束时记录日志。

处于支撑程序之上的是应用协议分析程序。应用协议分析程序由一个基于有限状态机的语法分析引擎和数个应用协议语法状态表组成[10]。分析引擎对不同的应用协议使用该协议的语法状态表对其报文进行语法分析，解析出应用协议语法单元和数据，并调用最上层的协议内容安全程序对解析出的内容进行安全检查和过滤。

协议注册程序提供应用协议语法状态表和协议内容安全检查程序的注册机制。在向 TPF 增加一种应用协议代理功能时，只需要生成该协议的语法状态表及开发该协议内容安全检查程序即可。在对已有的应用协议代理安全检查功能升级时，只需替换协议内容安全检查程序。

TPF 的代理程序分层结构较好地解决了代理防火墙支持的应用协议数量的易扩充性问题，降低了升级开发的成本。

6. TPF 的性能

我们对 TPF 在百兆网络环境下的性能进行了测试。测试环境如图6。

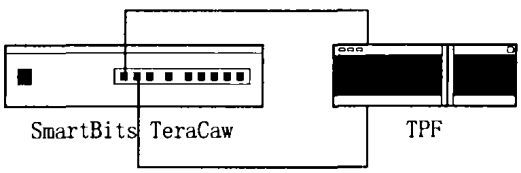
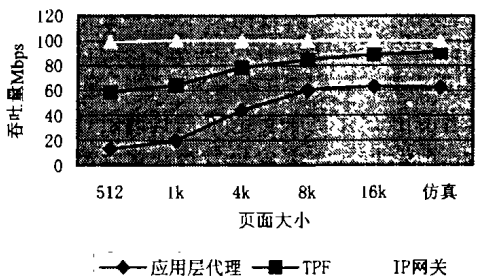


图6



(注：网络仿真测试时，被访问页面大小的分布情况为256字节 7.64%，512字节9.44%，1k 字节7.95%，2k 字节16.77%，3k 字节11.68%，4k 字节7.28%，6k 字节9.86%，8k 字节5.47%，16k 字节10.40%，32k 字节6.69%，64k 字节4.26%)

图7

代理防火墙的主机配置为：440BX 主板/PIII700处理器/256M 内存/3块 Intel pro100 s 网卡。测试设备为 SmartBits

600，测试软件为思博伦通信公司的应用代理专用测试软件TeraCaw。TPF策略配置为最小规模策略集。测试时长为120秒。对http协议代理的测试结果如下：

在50%吞吐量下网路仿真测试下平均处理延时为150微秒。

最大并发连接数为32.8万。在追求吞吐量的仿真测试中，每秒实际处理完成的最大请求数为989条。

TPF与代理的应用层实现及IP网关的最大吞吐量的对比测试数据见图7。

结束语 TPF的内核级实现方式，克服了存在于传统代理防火墙的诸多弱点，提高了自身安全性，很好地改善了代理防火墙的网络处理性能，同时也使得代理支持的应用协议数量扩充变得更加容易。下一步，我们将在ESK中引入代理程序与内核隔离机制，同时对协议栈快速通道的处理流程做进行优化，并改进协议报文分析程序的效率，以期进一步提高TPF的自身安全性和性能。

感谢 南京大学计算机系研究生赵静凯、卜宏、郭晓芳和软件工程中心职工朱佳来、李论等人在TPF的原型开发中承担了大量的编码工作。在此对他们的辛勤劳动表示衷心的感谢！

参考文献

1 Herrin G. Linux IP Networking. May 2000. http://kernelnewbies.org/documents/ipnetworking/linuxipnetworking.html
2 Anand V, Hartner B. TCP/IP Network Stack Performance in Linux Kernel 2.4 and 2.5 2002. http://www.linuxsymposium.org/2002/view_txt.php?text=abstract&talk=91
3 Stevens W R. TCP/IP Illustrated, Volume 2: Implementation. Addison Wesley Longman, Inc, 2000
4 毛德操，胡希明. Linux 2.4内核源代码情景分析. 浙江大学出版社，2001
5 Payne C, Markhan T. Articeture and Application for a Distributed Embedded Firewall. 2002. www.acsac.org/2001/papers/73.pdf
6 Cisto Systems, Evolution of the Firewall Industry 2002 http://www.cisco.com/univercd/cc/td/doc/product/iaabu/centri4/user/scf4ch3.htm
7 茅兵，等. 863计划《基于Linux的操作系统安全增强技术的研究与开发》课题技术报告，2003
8 蔡圣闻，等. 江苏省软件和集成电路专项《高安全高性能的网络防火墙》项目技术报告，2003
9 Srinivasan V. Fast and efficient internet lookups[D]: [Ph. D Thesis]. Washington University, 1999
10 卜宏，黄皓. 一种应用层协议报文解析算法. 计算机应用研究. 2003
11 Gallatin A, Chase J, Yocum K. Trapeze/IP: TCP/IP at Near-Gigabit Speeds, 1999 USENIX Annual Technical Conference
12 Epstein J, Thomas L, Monteith E. Using Operating System Wrappers to Increase the Resiliency of Commercial Firewalls. In: 16th Annual Computer Security Applications Conf. 2000
13 Kang J-M, et al. Extended BLP Security Model Based on Process Reliability for Secure Linux Kernel. 2001. http://www.computer.org/proceedings/prdc/1414/1414toc.htm
14 Barford P, Crovella M. Critical Path Analysis of TCP Transactions. In: Proc. of the 2000 ACM SIGCOMM Conf. Sep. 2000

（上接第60页）

较高。而ECN对弃尾队列的报文丢弃率没有影响。

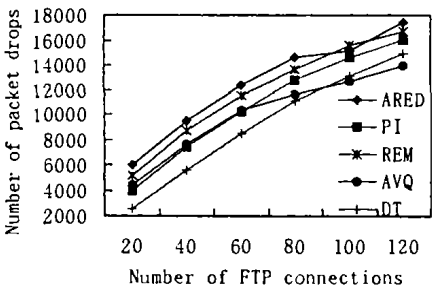


图6 报文丢弃率

结论 本文通过仿真对最近提出的几种重要AQM算法的性能进行了分析和比较，仿真结果表明：

(1)响应速度：ARED算法的响应速度最快，PI，REM，AVQ算法的响应速度随着负载的增加而减慢，其中PI的响应速度最慢。

(2)队列长度的控制能力：这四种算法都能够将队列长度控制在特定的水平，消除了稳态队列长度与负载的耦合。ARED在负载较重时，队列动荡大。PI和REM在收敛到稳态以后，能够将队列长度控制在参考点附近。AVQ只有在链路利用率较低的情况下，才能够将队列长度控制在较低的水平。PI，REM，AVQ队列动荡小。

(3)链路利用率：ARED，PI和REM的链路利用率比较高，AVQ的链路利用率与目标利用率的设置有关，需要在利用率和队列长度之间取折衷。

(4)报文丢弃率：标记策略能够大大降低AQM算法对TCP流的丢弃率。PI和REM在稳态时报文丢弃率为0。AVQ报文丢弃率一直为0，ARED在负载较轻时，报文丢弃率为0，负载较重时，由于队列长度大于最大阈值，导致报文被丢弃。

另外，ARED的实现复杂度是最高的，PI，REM和AVQ都比较容易实现。

在文中所讨论的四种算法中，ARED算法是动态调节参数的典型算法。由于ARED算法是对RED算法的增强和改进，所以算法的参数配置复杂，而且参数配置主要依靠经验方法，而不是基于系统的、科学的分析方法。PI是运用控制论设计的AQM控制器，它的参数选择基于系统的稳定性要求，能够比较容易地确定符合网络状况的参数。但是，PI控制器本身是一种滞后控制，所以不可避免具有响应速度慢的特点，而且PI控制器的响应速度与负载相关。因此，基于控制论的研究应该侧重于提高系统的响应速度和鲁棒性。REM算法是通过求解最优化问题得到的，其形式和性能均与PI相似。一般情况下，基于最优化方法设计的AQM机制需要与端节点的流控机制相结合，才能得到系统的最优性能。AVQ算法本质上是一种基于速率的AQM算法，由于报文到达速率很难反映拥塞的持续状况，所以AVQ算法对队列长度的控制能力有限。如何将报文到达速率与队列长度相结合来判断网络拥塞，是值得研究的问题。

参考文献

1 Braden B, et al. Recomedations on Queue Management and Congestion Avoidance in the Internet, RFC2309, April 1998
2 Floyd S, Jacobson V. Random Early Detection Gateways for Congestion Avoidance, IEEE/ACM Transactions on Networking, 1993, 1(4): 397～413
3 Floyd S, Gummadi R, Shenker S. Adaptive RED: an algorithm for increasing the robustness of RED's Active Queue Management. http://www.icir.org/floyd, Aug. 2001
4 Hollot C, Misra V, Towsley D, Gong W. On designing Improved Controllers for AQM Routers Supporting TCP Flows. In: Proc. of IEEE INFOCOM'01, Anchorage, Alaska, USA, 2001. 1726～1734
5 Lapsley D, Low S. Random early marking: An optimization approach to internet congestion control. In: Proc. of IEEE ICON '99, Brisbane, Australia, 1999. 67～74
6 Kunniyur S, Srikant R. Analysis and Design of an Adaptive Virtual Queue (AVQ) Algorithm for Active Queue Management. In: Proc. of ACM SIGCOMM 2001, San Diego, California, USA, 2001. 123～134
7 Ramakrishnan K K, Floyd S, Black D. The Addition of Explicit Congestion Notification (ECN) to IP. RFC3168, 2001