第33卷 第10期 计 算 机 工 程 2007年5月
Vol.33 No.10 Computer Engineering May 2007
·安全技术· 文章编号：1000—3428(2007)10—0172—03 文献标识码：A 中图分类号：TP393

基于网络处理器的防火墙优化设计与研究

沈 健，周兴社，张 凡，於志勇
（西北工业大学计算机科学学院，西安 710072）

摘 要：提出了基于网络处理器的状态检测型防火墙设计方案，并针对IXP2400的硬件结构，对访问控制列表和状态会话表的存储结构及表项查找等关键技术进行了优化，发挥了IXP2400内部各硬件单元的优点，系统达到线速处理的能力，使其性能得到了较大的提交。
关键词：网络处理器；防火墙；访问控制列表；状态会话表

Optimized Design and Research of Firewall Based on Network Processor

SHEN Jian, ZHOU Xingshe, ZHANG Fan, YU Zhiyong
(College of Computer Science, Northwestern Polytechnical University, Xi'an 710072)

【Abstract】 This paper proposes the design scheme of state packet inspection firewall based on network processor, and describes the optimized design of key technologies for IXP2400's hardware structure, including the storage structure and item searching of the access control list and the status session table. The proposed approach will enhance the performance of firewall observably.
【Key words】 Network processor; Firewall; Access control list(ACL); Status session table(SST)

利用网络处理器(Network Processor, NP)构建各种网络平台是当前研究的热点。在网络安全越来越受到威胁的情况下，防火墙技术日益受到人们的重视。基于NP技术的防火墙比通用处理器(GPP)技术在性能上有了很大提高，而且节省了专用集成电路(ASIC)技术所需的大量资金和较长的开发周期，因此备受国内外信息安全厂商关注，成为国内厂商实现高端千兆位防火墙的热门选择。

访问控制列表(Access Control List, ACL)和状态会话表(Status Session Table, SST)的查找效率是影响防火墙整体性能的重要因素。ACL/SST在GPP或ASIC上的应用已较成熟，而结合NP进行设计及优化方面的研究却很少。

本文针对IXP2400的具体硬件结构，给出了状态检测型防火墙系统的整体设计方案，对其中ACL/SST的建立及查找进行了优化。

1 防火墙整体设计

1.1 IXP2400简介

本文选用的NP是Intel IXP2400 [3]，它主要面向速率为2.5Gbps的网络应用，具有编程环境灵活、代码可重用、易于实现快速应用等特点。它主要包括1个600MHz的XScale处理器和8个可编程微引擎ME(MicroEngine)，每个ME支持8个硬件实现的线程。

ME不仅能够访问其内部的多种存储器，还可以对片外的SRAM、DRAM进行访问。此外，IXP2400还包含媒体交换接口MSF用于接收和发送数据包，Hash Unit提供48位、64位、128位的硬件散列算法。

1.2 功能设计

整个防火墙系统分为数据平面和控制平面，其中数据平面工作在ME上，用于对数据包进行ACL、SST的匹配；网络地址转换(NAT)等操作。而控制平面工作在XScale上，主要工作包括对各种表的建立与维护，对特殊包的处理，日志管理等。

1.3 流程设计

将使用7个ME完成数据平面的处理，每个ME作为一个功能模块，完成特定的功能，具体分配如表1所示。模块间通过Scratch Ring进行数据传递。

表1 微引擎功能分配

模块	微引擎	功能说明
接收模块	ME[0：0]	从端口接收数据，并将数据放入相应队列
接入处理模块	ME[0：1]	对从外到内的数据进行各种预处理，然后将其传给ME[0：2]微引擎
	ME[0：2]	根据ACL和SST对从ME[0：0]传来的数据进行过滤
接出处理模块	ME[0：3] ME[1：0]	对从内向外的数据进行NAT，过滤，MAC层处理等操作。这两个微引擎是并行模式
VPN模块	ME[1：1]	处理VPN数据流
发送模块	ME[1：2]	接收从各处理模块传递过来数据，并将数据根据相应的端口发送出去

2 ACL、SST的设计与优化

状态检测型防火墙在传统基于ACL静态过滤防火墙基础上增加了基于SST的状态检测功能。对ACL与SST表项的查找效率是影响防火墙性能的重要因素。

基金项目：国家“863”计划基金资助项目(2003AA1z2100)；西北工业大学青年科技创新基金资助项目(M016213)；西北工业大学研究生创业种子基金资助项目(Z2000643)
作者简介：沈 健(1983－)，男，硕士，主研方向：嵌入式系统；周兴社，教授、博导；张 凡，讲师、博士生；於志勇，硕士生
收稿日期：2006-06-10 **E-mail：**kavenyz@hotmail.com

2.1 ACL

2.1.1 ACL 原理

ACL 功能即匹配预先设定的规则来允许或拒绝数据包，从而实现对数据流的控制。它是一组由 permit 或 deny 语句组成的条件列表，可以让防火墙管理员以报文的源、目的 IP 地址和协议类型等条件来控制网络的数据流。

2.1.2 ACL 表项的设计

ACL的规则可以让管理员通过控制平面输入指令的方式进行配置[5]，需要将管理员输入的指令映射到存储在内存中的ACL表中。表项格式如图 1 所示。

Listnumber (8bits)	P (1bit)	Vid (1bit)	Protocol (6bits)	Icmp type (8bits)	Icmp code (8bits)
source ip address (32bits)					
source mask (32bits)					
source port1 (16bits)			source port2 (16bit)		
dest port1 (16bits)			dest port2 (16bits)		
dest IP address (32bits)					
dest mask (32bits)					
Log (32bits)					
source Operator		dest Operator		Next entry (24bit)	

图 1 ACL 表项格式

2.1.3 ACL 表的存储设计与优化

ACL 表只在新连接建立前被查找，可以放在 SRAM 中。由于 ME 对 SRAM 一次访问会产生 90 周期延时，而 ACL 每个表项为 9 个长字，因此进行一次匹配表项的操作超过 810 个周期。如果线性遍历 ACL 表进行匹配，产生的延时将是十分惊人的。为减少延时，设计了以下方式进行优化。

将 ACL 细分成 TCP-ACL、UDP-ACL、ICMP-ACL、OTHER-ACL 4 大类。应用层的大多数服务都是基于 TCP 的，在企业网中，HTTP、FTP、SMTP 和 POP3 被称为 4 大协议，而对校园网流量进行评估一般采用的流量模型是 72%HTTP+20%FTP+4%UDP。将 TCP-ACL 分为 HTTP-ACL、FTP-ACL、SMTP-ACL 及 OTHER-ACL 4 个子表。第 1 步优化就是将完整的 ACL 表被分成这 7 个子表，匹配时根据接收包类型，进入对应的 ACL 子表，逐一匹配。在最坏的情况下，也只匹配整个 ACL 项数的 2/7，在较大程度上提高了效率。

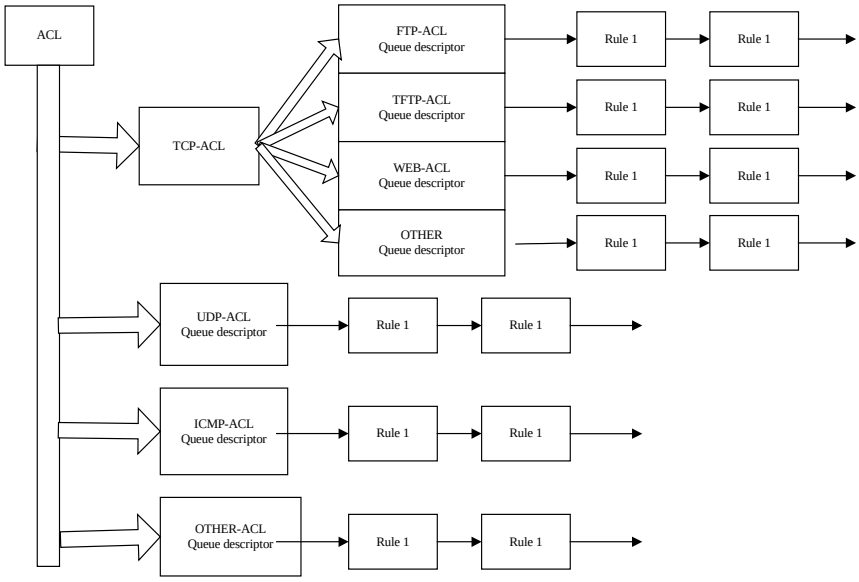

图 2 ACL表存储结构

传统 ACL 是一个整表，只需静态分配一段足够大的存储区。ACL 表被细分后，这种分配方法不再适用，因为如果子表存储区域固定，其大小将很难确定，太小可能导致子表空间不够，太大就造成资源浪费。为此，第 2 步优化就是动态分配子表表项。每个子表为链表方式。系统给整个 ACL 模块分配的存储空间是固定的，某个子表需添加一项时，就从总存储空间中取出一项，添加到该子表链表中，如图 2 所示。

这样优化后，效率还有提升的余地。因为查找子表仍然是线性访问 SRAM 的方式。第 3 步优化是将经常匹配的表项存放在 LM(Local Memory)中。在 IXP2400 中，每个 ME 都含有一个 2 560B 的 LM，对它的访问延时小于 3 个周期。LM 最多可以存放 71 条规则，而一般 ACL 的规则至多 200 条，因此从容量来看，LM 作为 Cache 已足够。置换算法采用最久未使用算法(LRU)，大大减少了对 SRAM 的访问次数，效率得到很大的提高。

综上所述，对 ACL 表项的匹配过程如下：先遍历 LM 中已有的表项，若匹配成功，则完成；否则根据包的类型再到 SRAM 中的对应子表进行匹配；匹配成功后，将该表项复制到 LM 中；如果 LM 已满，则采用 LRU 进行置换。

与IXP2400 相结合，其硬件结构大大增强了这种设计的实际可行性。首先，ACL存储在SRAM中，SRAM控制器[1]是IXP2400 与SRAM的接口单元，它有着复杂的功能结构，除了支持一般的数据读写之外，还支持原子操作、RMW操作，另外，它还集成Q-Array[1]等功能强大的硬件结构，支持多达64 个链表队列，使得ACL表等复杂的数据结构在运用时十分简单方便；其次，每个ME都有自己的LM。如果有两个ME分别处理两个来自不同接口的数据，那么它们内部LM中存放的都是各自经常访问的规则，彼此相互独立。

2.1.4 代码实现

为了更好地描述 ACL 查找过程，下面结合 ME 部分伪代码进行关键步骤分析：

```
Lm*Index=base;[2]
//该项是否有效
Begin#: Alu[--, index, and,0x00400000 ]
Beq[next-entry]
//类型匹配
    If (type!=protocal)
     //目的和源地址匹配
    Compare[src-ip, index1, index2, match]
    If (!match) Beq[next-entry]
    Compare[dest-ip, index5, index6, match]
    If (!match-not) Beq[next-entry]
    //目的和源端口匹配
    Compare[source-port, index3, index8, match]
    If(!match) Beq[next-entry];
    Compare[dest-port, index4, index8, match]
    If (!match) Beq[next-entry]
    //next-entry#处伪代码
    base=base+9
    If (base<lm_max) Index=base Br[begin#]
    Endif
    //LM 匹配失败，转到 SRAM 中
     Base=type-base;
    …
```

根据代码量可以看出匹配 LM 中一项的时间为 20~30 个周期。如果在 LM 中匹配不成功，需要转入 SRAM 的话，延时就会增加 20 倍以上。但对 ACL 匹配只发生在有新连接到来时，而且 LM 能存放约 1/3 的全部表项，匹配不成功的概率很小。经过以上 3 步优化后，总体性能显著提高。

2.2 SST

2.2.1 SST 原理

动态包过滤是一种分析包的方法。当通过防火墙建立一个从内部主机访问外部主机的 TCP/UDP 会话时，有关该会话的信息将被自动记录在 SST 中。会话状态信息包括源和目的地址、端口号、生存周期。对于 TCP 会话还包括 TCP 序列号以及确认号。经过基于会话状态建立的过滤规则对入网包进行过滤，只有当存在一个能确认该包可以通过的相关会话时，才允许通过防火墙。

2.2.2 SST 的设计与优化

由此看出，影响动态包过滤性能的最主要因素就是 SST 的查找效率。由于每一个数据包在进入防火墙过滤模块后都必须先进行 SST 的匹配，与 ACL 相比，SST 的查找更加频繁，因此它的查找效率对整个系统的性能更为关键。

传统的 SST 的查找是最普通的线性查找，这种查找方式的缺点很明显，尤其当 SST 存放在 SRAM 中时，效率十分低下。基于此，设计了两种优化的方法。

(1)采用与先前 ACL 一样的 Cache 方式。但是 SST 是一个全局表，存放在 LM 中就使其变成了局部表，而且 SST 表项在被成功匹配后，相应元素会动态变化，要做到在每一个 LM 中对其修改同步是很难的。如果一个 ME 同时进行 ACL 和 SSL 的查找，那么 LM 已用于 ACL，不可能再用于 SST。

(2)基于散列方式。每个连接在 SST 中的存储信息，只有序列号、确认号以及生存周期是不断更新的，而源/目的 IP 地址、端口是不变的，可以根据一个会话的不变信息将它的存储位置确定。进行查找的时候，可以直接定位该项地址，而无须从表头开始进行线性查找。

在具体实现这种方法之前，先设计出 SST 的表项格式(图 3)。表项中的信息应包括源/目的 IP 地址、源/目的端口、TCP 序列号、TCP 确认号以及该会话的生存时间。Next entry 指向下一项，它的具体值是所对应项在冲突表中的偏移；Vid 表示该项是否有效；live time 规定了该连接的生存时间，该值最大为 3 600s。

<table>
<tr><td>Next entry (16bit)</td><td>Vid (1bit)</td><td>Live time (15bits)</td></tr>
<tr><td colspan="3">source IP address (32bits)</td></tr>
<tr><td colspan="3">dest IP address (32bits)</td></tr>
<tr><td colspan="2">source port (16bits)</td><td>dest port (16bits)</td></tr>
<tr><td colspan="3">Sequence number (32bits)</td></tr>
<tr><td colspan="3">Acknowledge number (32bits)</td></tr>
</table>

图 3 SST 表项格式

给 SST 分配的空间是 2MB，其中前 1.5MB 是主表，后 0.5MB 作为冲突表。一个表项为 24B，因此主表最大可支持 1.5M/24=64K，即 65536 个连接。而冲突表则最大可解决超过 2 万个冲突项。在向 SST 中添加一条会话时，先得到该会话的源 IP

地址的低 24 位，目的 IP 地址的低 24 位，源端口的低 8 位，目的端口的低 8 位，如表 2 所示，然后对其进行 Hash 操作，得到 64bits Hash 值，按一定规则取其中 16bits 作为 key，再将该 key 作为该会话在主表中地址的偏移。找到对应项后，如果该项中的有效位为 0，则表示没有发生冲突，这时将该条会话写进即可；如果该项中的有效位为 1，则表示该项中已经有值，这时查看该项中的 Next entry，如果不为空，则沿着 Next entry 往下找，直到为空的那一项，从冲突表中分配一项，写入该条会话的信息，然后用前一项的 Next entry 指向该项。

表 2 Hash 操作的数据位

Source_addr (low24bits)	Source_port (low8bits)
Dest_addr(low24bits)	Dest_port(low8bits)

具体结合 IXP2400，它提供了 48bits、64bits、128bits 的 Hash 操作。由于是由硬件完成，因此延时极短，而且 Hash 单元在算法上做了优化，产生冲突的可能性很低，显著地提高了 SST 查找的效率。

2.2.3 代码实现

下面结合 ME 部分伪代码对 SST 查找进行关键步骤分析：

```
//构造出 hash 二元组
Hash[0]=source_ip<<8+src_port&&0x0000001
Hash[1]= dest_ip<<8 + dest_port&&0x00000011
//进行 hash 操作
Hash_io_64[hash[0]]
//得到 key
Extract(hash[0],hash[1],key)
//得到 sram 中的对应项
Adderss=sst_base + key*6
Entry(0,1,2,3,4,5)=Address(0,1,2,3,4,5)
//匹配失败，进行下一项
Compare(Entry,cur,match)
If(!match) Beq[next-entry]
//下一项无效，则匹配失败，将该报丢弃
If(!next_entry_valid)    Drop[curr_packet]
```

3 结束语

NP 被认为是推动下一代网络向高性能、灵活性方向发展的核心技术，通过合理的软件体系提供硬件级的处理性能是其关键。本文依据 IXP2400 网络处理器的软硬件特点，在实际实现状态检测型防火墙过程中，结合 IXP2400 中的各种硬件单元，对 ACL 和 SST 的存储，表项的建立、查找进行了优化，充分发挥了 IXP2400 内部各硬件单元的优点，使系统达到线速处理的能力。

参考文献

1 Intel Corp. IXP2400 Network Processor Hardware Reference Manual[Z]. 2003-09.

2 Intel Corp. IXP2400/2800 Network Processor Programmer’s Reference Manual[Z]. 2003-09.

3 Intel Corp. Intel Internet Exchange Architecture(IXA) Portability Framework Developer’s Manual[Z]. 2003-11.

4 陈红林. 基于 Intel IXA 架构的防火墙设计[D]. 成都: 电子科技大学, 2003-04.

5 范　萍, 李罕伟. 基于 ACL 的网络层访问权限控制技术研究[J]. 华东交通大学学报, 2004, 21(4): 89.