基于网络微处理器包过滤硬件防火墙的研究

胡成伟
(广州民航职业技术学院 通讯工程系，广东 广州 510403)

摘 要：从网络处理器NP（Network Processor）对数据包接收、处理和发送的角度，讨论在NP架构下多微引擎、多线程并行处理网络数据包，实现基于包过滤方式防火墙的原理。

关键词：网络微处理器；防火墙；包过滤；微引擎；并行处理
中图分类号：TP393.08 **文献标识码**：A

Research of filter packet hardware firewall based on network micro processor

HU Chen Wei
(Department of Communication Civil Aviation, College of Guangzhou, Guangzhou 510403, China)

Abstract: From the angle of data transmission, data procession and data receiving, the article discusses multiple micro-engines and multi-thread process data packets in parallel. It also analyses the theory of filter packet hardware firewall based on network processor.

Key words: network micro processor; firewall filter packet; micro-engine; process in parallel

防火墙是当今计算机网络安全的主要设备之一。随着网络数据量的增大、速度的增高，软件防火墙已不能胜任，硬件防火墙成为大流量、高品质防火墙产品的首选。硬件防火墙的实现方案有采用网络处理器ASIC技术实现芯片级防火墙和采用网络处理器NP（Net Processor）技术实现微处理芯片防火墙两种。对两种方案进行比较，ASIC处理速度快，数据吞吐量大，但开发难度大，开发周期长，很难适应快速变化的网络环境而升级换代；NP技术实现方案灵活，也较ASIC开发难度小，开发周期短，可以根据情况快速升级。NP技术是适合我国国情的硬件防火墙产品的实现方案，是目前我国硬件防火墙产品主要采用的技术[1]。

防火墙可根据其处理数据的层次分为：包过滤防火墙、状态检测防火墙、代理服务器防火墙和核处理防火墙。对于要求高速处理的硬件防火墙（处理能力大于2.5 Gb/s），无法完成代理服务防火墙功能和核处理防火墙功能。基于包过滤功能硬件防火墙是高速防火墙能采用的基本实现策略。以下将就NP实现包过滤的方法进行研究。

1 IXP2400网络处理器

NP是专门为处理网络数据包而设计的可编程处理器，它能够并行、高速完成网络数据处理。IXP2400是INTEL公司的第二代NP产品，它采用多内核并行结构，由1个XScale Core作为核心处理器，以及8个32位独立可编程、支持多线程的微引擎（MicroEngine）构成。NP的体系结构可分为两个层面：控制层面和数据层面。

控制层面由XScale Core处理，它的主要任务是完成对整个NP各部件的初始化，运行操作系统，完成复杂而非实时运算。控制层面一般安装的操作系统是：VxWorks或LINUX等。由于控制层面是构建在一个强大的XScale Core之上，其开发与一般的嵌入式系统开发相类似。

数据层面主要负责对数据包的收发和实时数据处理，由微引擎来完成。微引擎是32位的采用RISC技术实现的微型MCU，对它的编程与传统嵌入式系统开发不同，由INTEL提供的MicroC和MicroASM支持。微引擎和XScale Core在工作时关系如图1所示。

2 包过滤的规则

包过滤是基于一系列规则对进出防火墙的数据包

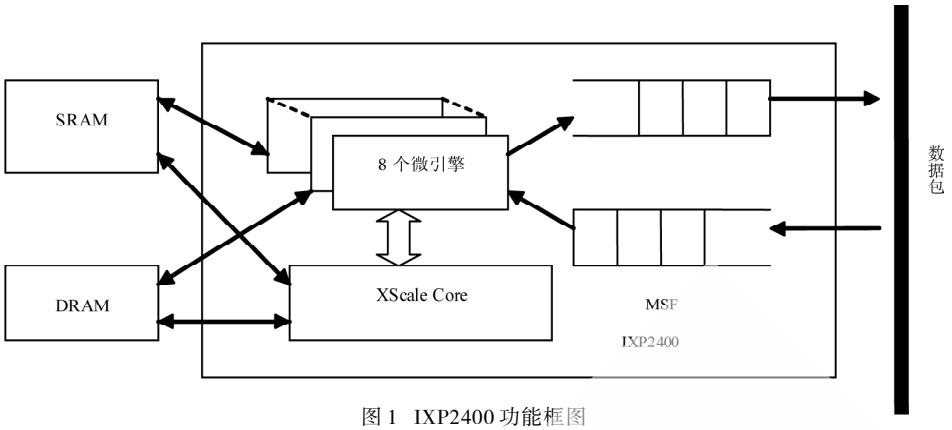

图 1 IXP2400 功能框图

表1 包过滤规则表

规则	目的地址（地址/掩码）	源地址（地址/掩码）	传输层目的	传输层协议	动作
R1	152.163.190.69/255.255.255.255	152.163.80.11/255.255.255.255	*	*	禁止
R2	152.168.3.0/255.255.255.0	152.163.200.157/255.255.255.255	Eq www	UDP	禁止
R3	152.163.198.4/255.255.255.255	152.163.160.0/255.255.255.0	Gt 1023	TCP	允许
R4	0.0.0.0/0.0.0.0	0.0.0.0/0.0.0.0	*	*	允许

表2 包过滤的结果

数据包	目的地址	源地址	传输层目的	传输层协议	规则与动作
P1	152.163.190.69	152.163.80.11	www	Tcp	R1，禁止
P2	152.168.3.21	152.163.200.157	www	UDP	R2，禁止
P3	152.163.198.4	152.163.160.10	1024	TCP	R3，允许

进行过滤。规则其实是一个“if conditions then action”的判断，一组规则构成一个规则表（如表1）。当IP数据包通过这个规则表的检查后，允许通过的IP包就转发，禁止通过的IP包就被拦截下来，其结果如表2所示。对于一个小型的网络，其规则一般有几百个，对于一个中型的ISP服务网络，其规则一般有数千个，对于一个大型的网络其规则一般超过2万个。在进行包过滤时，最重要的就是进行规则的匹配。如何快速查找匹配规则，减少存储器容量成为包过滤算法的研究重点，参考文献[2]中详细地讨论了一些包过滤算法。

本文主要是讨论NP在整个过程中各部分的工作，为叙述简单将不涉及复杂的规则查找，采用线性搜索，即从上到下对规则表进行查找，在制定规则表时，优先级高的规则将被安排在表的前面。

3 基于NP的包过滤

IP数据包过滤处理由IXP2400中的微引擎完成。微引擎是一个个独立的MCU，有自己的寄存器、存储器，执行各自的指令序列，互不干扰，而且每个微引擎支持8个硬件线程。在数据包处理的过程中，IXP2400的8个微引擎可以采用串行流水线方式（如图2）工作，或以并行处理方式（如图3）工作。在参考文献[3]中指出并发处理比串行流水线处理的效率要高25%。

在串行流水线工作方式中，每一个微引擎（ME）完成的工作不同，当一个微引擎完成其工作后，将数据包交给下一个微引擎继续后续工作，所以每一个微引擎执行的代码不同。在并发处理方式中，每一个微引擎完成的任务相同，每一个微引擎所执行的代码相同。本文在安排微引擎时，包的接收采用ME0，当它处理完后将结果交给ME1～ME6，这6个微引擎并发对数据包进行规则匹配，匹配结束后将包交由ME7完成发送任务。整个包过滤处理结构既有串行方式，又有并发方式，这是因为在整个处理过程中，进行规则匹配所需的时间和运算都比接收和发送的要多，将更多的处理能力部署在这一环节，可以消除整个系统处理的瓶颈。

3.1 接收处理的实现

MSF（Media Switch Fabric Interface）是IXP2400连接网络的接口。它具有8 KB的RBUF（接收缓冲）和8 KB的TBUF（发送缓冲）。当外部以太网帧（Packet C）进入MSF时，MSF

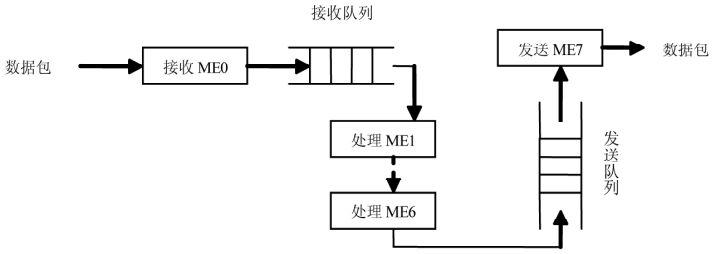

图 2 串行流水线方式

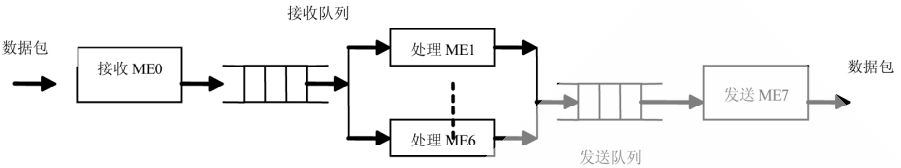

图 3 并行处理方式

将数据帧分割成若干个大小为 64 B（或 128/256 B）的 mpacket，mpacket 存放在 RBUF 中，每一个 mpacket 占据 RBUF 的一条目。当 RBUF 中有有效条目时，MSF 将发出一个 RBUF 有效信号。在 MSF 中，有一个 RX_THREAD_FREELIST 数据结构，该结构登记了用于处理接收任务的空闲线程。本文将 ME0 中空闲线程登记在这里，并按顺序排列成接收线程链。当一个空闲线程接收到一个 RBUF 有效信号的触发时，便会进入忙状态，进行一次 mpacket 的接收循环；当循环结束后，这个线程将重新回到空闲状态，并重新链入 RX_THREAD_FREELIST。各线程在接收 mpacket 时，以无限循环方式进行，一次循环中完成如下工作：

（1）检查 mpacket 是否 SOP（Packet C 进入 MSF 时，被分割成多个 mpacket，其中第一 mpacket 就是 SOP）或 EOP（Packet C 中最后一个 mpacket）；

（2）如果是 SOP，线程将在 DRAM 中开辟一个新 Buffer，并将 mpacket 拷贝至其间，如果不是 SOP 则紧接着前面拷贝的内容进行拷贝。这样可以将一个被分割的包重新组合起来；

（3）如果是 EOP，意味着 Packet C 的结束，在拷贝完此 mpacket 后，线程将在 SRAM 中的接收队列中将此 Packet C 的包句柄（PC）插入到队列的尾部。接收过程如图 4 所示。

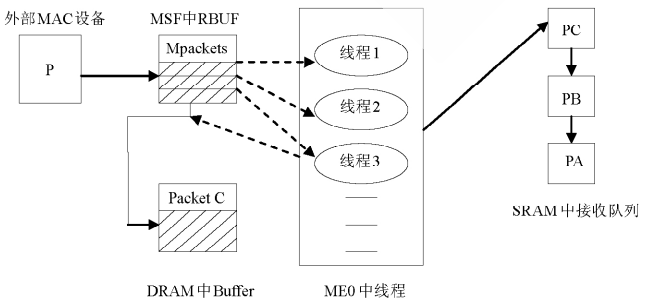

图 4 接收过程

每个线程一次循环只处理一个 mpacket，如果一个包分成几个 mpacket，则由几次这样的循环完成接收。由于每个 IXP2400 的微引擎都有 8 个硬件支持的线程，在接收包处理时，可以出现多个线程并发接收多个 mpacket，这样效率当然会很高，但同样也可能打乱 Packet C 的重组。为避免这种情况的发生，各线程的工作次序是严格规定的。各空闲线程在 RX_THREAD_FREELIST 登记时，就按顺序登记，线程 1 在最前，线程 8 在最后。当第一个触发到来时，则触发线程 1，线程 1 在处理时，若再接收到触发时，线程 2 接收触发，如此类推，当线程 1 处理完后，它跟在线程 8 后面，如此形成一个闭合的线程处理链。在将数据从 RBUF 拷贝至 DRAM 的过程中，线程要经过一个对线程序号敏感的微处理块，以保证多线程在拷贝过程中是按顺序进行的。

3.2 规则匹配处理的实现

本文所涉及的规则表中规则数目较少，搜索匹配规则的方法也相应简单，采用的是线性搜索方法。处理的流程如图 5 所示。

如前所述，采用 ME1 ~ ME6 共 6 个微引擎同时对数据包进行包过滤，每个微引擎有 8 个线程，所以可用于包过滤的线程有 6 × 8 = 48 个，每个线程都采用无限循环方式。当接收队列中有有效元素，便发出处理信号。在线程池中，处在等待状态的某一线程便会从 SRAM 中的接收队列取出头元素。接着，线程根据所取得元素中的包句柄从 DRAM 中将帧首部读进来，然后判断是否是一个有效的以太网帧。如果不是，则丢弃包，并返回等待状态。如果是有效帧，则从 DRAM 中读出 IP 首部，对在 SRAM 中的规则进行匹配。包过滤规则由 XScale Core 在 SRAM 中建立一个规则表，并可根据实际情况对表进行增加和删除。线程从 SRAM 中将一条规则读进来，进行匹配运算。若匹配，则根据规则中允许 / 禁止进行后

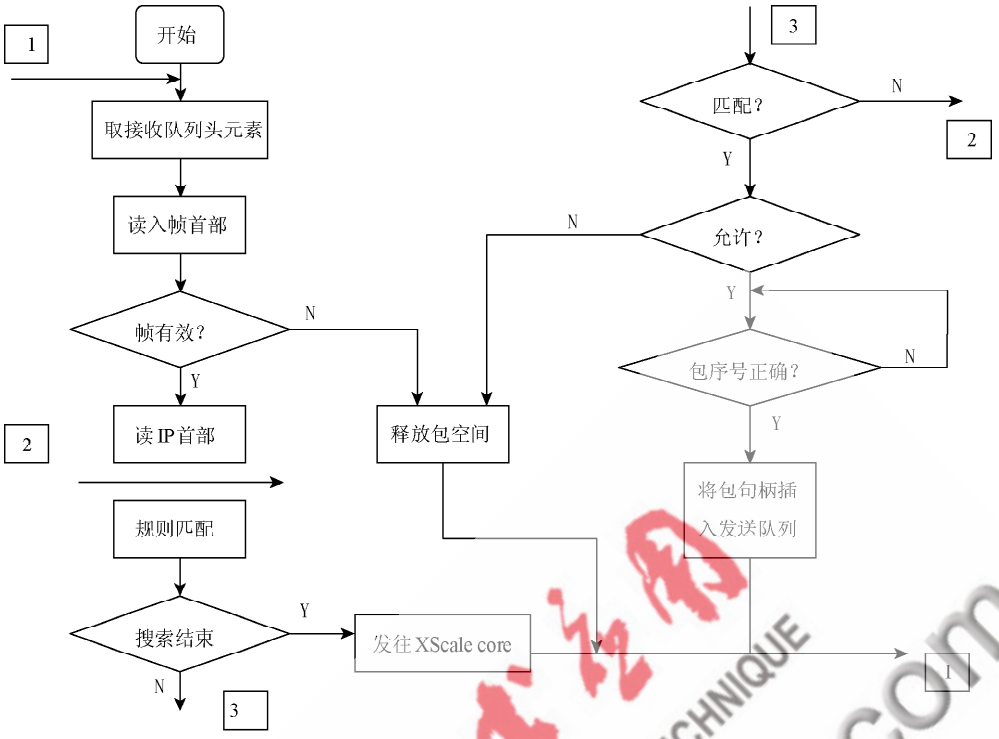

图 5 包过滤线程处理流程

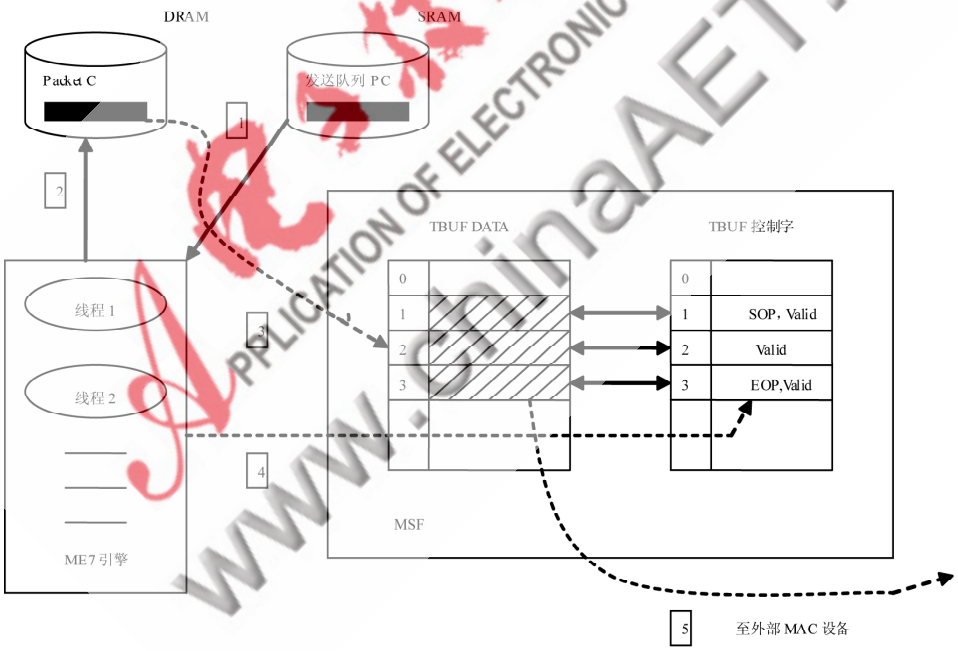

图 6 发送线程工作原理

续工作；若允许，则将把 DRAM 中对应的包的句柄作为新的元素插入 SRAM 中的发送队列中；若禁止，则释放 DRAM 中对应的包空间，并返回线程的等待状态。若不匹配，则从 SRAM 中读入下一条规则，重复以上工作，直到最后一条；如果仍找不到匹配规则，则将接下来的工作交给 XSale Core 完成。

每个包过滤线程完成一个包的过滤，从接收队列中将包句柄一一取出，处理完后，再在发送队列中将其插入到队列的尾部。但可能因为有些包处理得快，有些包处理得慢，使原来的接收顺序因为处理速度的不同而打乱。为了使发送队列句柄的顺序保持与接收队列一致，采用了阻塞式顺序包算法（Blocking Packet-ordering Athorithm）[4]。

3.3 发送处理的实现

发送任务由 ME7 来完成，ME7 有 8 个线程，每个线

(下转第 43 页)

表3是在LPR和XUE两种算法下，链路的利用率状况(为直观只列出所选路径中涉及的链路)。

表3 两种算法下链路利用率

链路	XUE算法下链路利用率/%	LPR算法下链路利用率/%
1-6	83	83
6-7	100	72
4-6	67	67
6-5	65	83
5-7	57	82
7-9	26	26
6-8	5	24
8-7	12	29

图4是在SPF和LPR算法下链路6-7带宽的利用率情况。

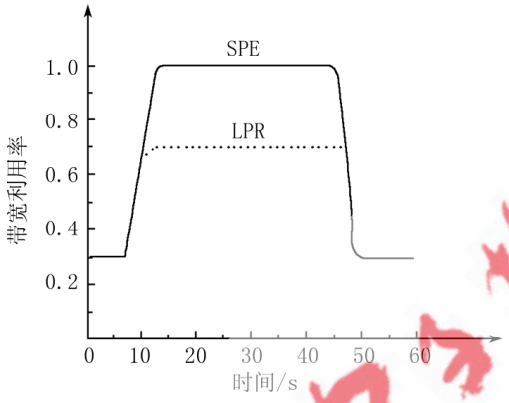

图4 链路6-7在两种算法下的带宽利用率

由仿真实验可知，LPR算法较SPF和XUE算法更加有效：

（1）LPR算法是XUE算法的补充，对于缓解由于流量对资源竞争引起的包丢失具有显著的效果；

（2）LPR算法更加有效地降低了资源利用率，避免瓶颈链路的产生；

（3）LPR算法能更好地调节流量在整个网络上的分布，使网络资源得到均衡分布；

（4）LPR算法大大提高了连接请求的通过率，增加了网络的吞吐量。

参考文献：

[1] 薛希俊，孙雨耕，刘振肖. 基于带宽和跳数的流量工程动态路由选择算法研究[J]. 电子学报，2002，30(2):274-278.

[2] 贾艳萍，孟相如. 基于MPLS流量工程的多路径约束负载均衡方法[J]. 计算机应用，2008，27(3)：522-524.

[3] 朱明英，叶梧. 基于最小干扰机制的MPLS流量工程动态路由算法[J]. 科学技术与工程，2008，8(19)：5394-5398.

[4] HUANG He，LI Wei-Qin. Optimization of MPLS traffic engineering architecture[J]. Journal of Beijing University of Aeronautics and Astronautics，2003，29(3):221-224.

[5] 刘郁恒，张光昭. MPLS流量工程技术的研究[J]. 数据通信，2000(2)：1-4.

[6] WANG Y F，WANG Z. Explicit routing algorithms for internet traffic engineering[A]. IEEE ICCCN'99[C]. Boston，MA. 1999:582-588.

（收稿日期：2008-12-24）

(上接第39页)

程完成如下工作：(1)线程发现发送队列中的有效包句柄则从SRAM的发送队列中将队列头元素取下来；(2)计算每个mpacket在包中的位置，把包从DRAM中以mpacket大小拷贝到TBUF中，其中TBUF是MSF中的发送缓冲区；(3)写入TBUF的单元控制字，表明TBUF包含有效数据；(4)当MSF收到EOP标志的mpacket时，表明该包结束，此后该包将交由外部的MAC设备传输。其过程如图6所示。

一个发送线程一次循环只负责一个mpacket的操作，周而复始。如同接收线程那样，发送线程排好队，如流水线般将发送队列中的元素对应的包，分解为mpacket单元，并逐个按顺序搬运到TBUF缓冲区。

在上述包过滤规则匹配时，微引擎会多次访问DRAM，以及在SRAM中进行搜索。当规则表中有较多规则时，查找规则的算法会变得相当复杂，将严重影响防火墙的处理速度。要使防火墙能快速地完成包过滤功能，可采用2个层次的手段：其一，改进查找算法，比如使用基于状态的动态包过滤算法；其二，充分应用NP内部的并行处理架构，安排好各微引擎工作内容，协调好微引擎内各线程的工作，使NP能高效并行地运行。

参考文献

[1] 宋斌，程勇，刘科全. NP架构千兆线速防火墙的体系结构与关键技术，信息安全与通信保密，2004(8)：22-25.

[2] PANKAJ G，KEOWN M. Algorithms for packet classification. New York:IEEE，March/April 2001：24-32.

[3] DEEPA S，FANG，Wu Chang. Performance analysis of multi-dimensional packet classification on programmable network processors. New York： IEEE，Proceeding of the 29th Annual IEEE International Coference on Local Computer Networks (LCN'04).

[4] ERIK J J，AARON R K. IXP2400/2800 Programming. INTEL PRESS.

(收稿日期：2008-12-24)