文章编号: 1001- 2486(2008) 03- 0053- 06

基于分布式转发交换的并行路由器关键技术研究*

戴 艺, 孙志刚, 苏金树, 管剑波
(国防科技大学 计算机学院, 湖南 长沙 410073)

摘 要: 随着 Internet 链路速率和 IP 前缀数目的不断增长, 对路由器的转发和交换能力提出了更高的要求。提出一种基于分布式转发交换的并行路由器体系结构, 采用多个低速的能够独立转发和交换报文的功能部件构成多级流水线, 以流水的方式执行报文转发和交换。对该结构实现关键技术——基于子树映射的 IP 流水查找机制进行了深入的研究, 提出了相应的解决方案, 并指出了下一步的研究方向和思路。

关键词: 并行路由器体系结构; 路由表分解; 基于子树的 IP 查找
中图分类号: TP393 文献标识码: A

Research of Key Techniques for Parallel Router Based on Distributed Forwarding and Switching

DAI Yi, SUN Zhi-gang, SU Jin-shu, GUAN Jian-bo
(College of Computer, National Univ. of Defense Technology, Changsha 410073, China)

Abstract: Continuing growth in link speeds and the number of advertised IP prefixes places increasing demands on the performance of Internet routers. In view of this fact, a parallel router architecture based on partial forwarding and pipelining switching is proposed. The architecture consists of multi-stage, lower speed nodes performing IP-lookups and switching independently, thus IP-lookups and switching for multiple packets have been pipelined. We investigate the key technologies of this architecture including the logical mapping from subtries to function components as well as the pipelining IP-lookup mechanism based on subtrie. Finally, future directions and possible open problems are discussed.

Key words: parallel router architecture; partition of routing table; IP-lookup based on subtrie

Internet 网络流量、网络规模和上层应用的快速发展, 对 Internet 互联的核心设备——路由器提出了越来越高的要求。目前路由表前缀数目已经达到 24 万并在继续增长中, 不久的将来就会达到 100 万[1], 在链路速率为 160 Gbps、最小报文为 40 个字节的情况下, IP 路由查找速率必须达到 2 ns/packet。随着 IPv6 协议的应用, 前缀长度将从 32 位增长到 128 位, 这无疑对 IP 查表技术提出了更大的挑战。基于 TCAM 的 IP 路由查找策略能够满足目前路由器对 IP 路由查找性能的需求, 但 TCAM 成本高、功耗大、可扩展性差, 难以支持日益庞大的路由表。基于算法的 IP 路由查找策略可扩展性好、性价比高, 但速度慢, 难以适应高速链路的发展。另一方面, 随着 WDM 通道速率的不断提高, 链路速率已经从 OC192 (10Gb/s) 到 OC768(40Gb/s), 甚至达到了 OC3072(160Gb/s), 在路由器交换结构中缓冲报文已经变得很困难甚至是不可能的。例如, 链路速率为 160Gb/s 时, 缓冲存储器必须在不到 1ns 的时间里完成对 40 字节信元的写入和读出操作, 而商业 DRAMs 的随机访问时间为 50ns[2]。

目前已有的路由器采用转发与交换分离的体系结构, 即首先由转发部件对报文进行精确查表, 确定其转发决策, 然后由多级交换网络将报文发送到目的端口。在这种体系结构中, 转发阶段与交换阶段在逻辑上是分离的; 转发部件按照特定的 FIB 容量及 FIB 处理能力进行设计, IP 转发机制对链路速率及 FIB 容量的增长不具备可扩展性; 交换部件则难以匹配高速链路, 提供延迟保证。必须同时开发转发和交换操作的并行度, 实现转发延迟与交换延迟的相互隐藏。

本文提出一种基于分布式转发交换的并行路由器体系结构, 路由器由多个结点构成, 每个结点是独

* 收稿日期: 2007- 12- 20
基金项目: 国家自然科学基金资助项目(90604006)
作者简介: 戴艺(1980—), 女, 博士生。

立的具有一定转发和交换能力的功能部件, 称之为转发交换结点 FSN(Forwarding and Switching Node) 。通过分解路由表, 构建路由表到 FSN 结点的映射, 每个 FSN 结点仅保存转发表的一部分, 执行部分转发操作, 大大降低了 IP 查找复杂度; 多个 FSN 结点组成多级流水线结构, 对报文转发交换执行流水操作。这种体系结构通过同时开发转发和交换操作的并行度, 降低对单个转发交换部件的要求, 提高路由器的整体转发和交换能力。

1 基于分布式转发交换的并行路由器体系结构

在如图 1 所示的基于分布式转发交换的并行路由器体系结构实例中, 4 级 2× 2 的转发交换结点 FSN 构成 4 级流水线, 到达报文被分派到 2 个 FSN 结点并行处理, 每个 FSN 结点执行部分 IP 查找之后确定下一级可用的 FSN 集合, 将报文缓冲到相应的输出队列中等待交换到下一级 FSN 结点。在该体系结构下, 流水深度、资源复用度(每一级 FSN 结点的个数) 、FSN 节点端口数目均可自由配置, 因而具有很好的灵活度和可扩展性。通过深入研究基于路由表分解的部分转发和流水交换技术, 提出了可行的能够正确转发和交换报文、显著提高 IP 转发速率及报文交换能力的并行路由器实现方案。该体系结构的实现取决于以下三大关键技术:

(1) IP 路由表的分解技术: 如何合理地分解 IP 查找树, 有效降低报文转发复杂度。

(2) 子树到 FSN 结点的映射技术: 研究分解后的路由转发表在多个 FSN 结点的映射算法, 保证报文部分转发的正确性和路由转发表在 FSN 结点中分布的均衡性。

(3) FSN 结点的 IP 查找和交换机制: 设计单个 FSN 结点的报文转发和交换策略, 以高效的查表和简单的报文调度机制实现多路径负载均衡及报文流的顺序。

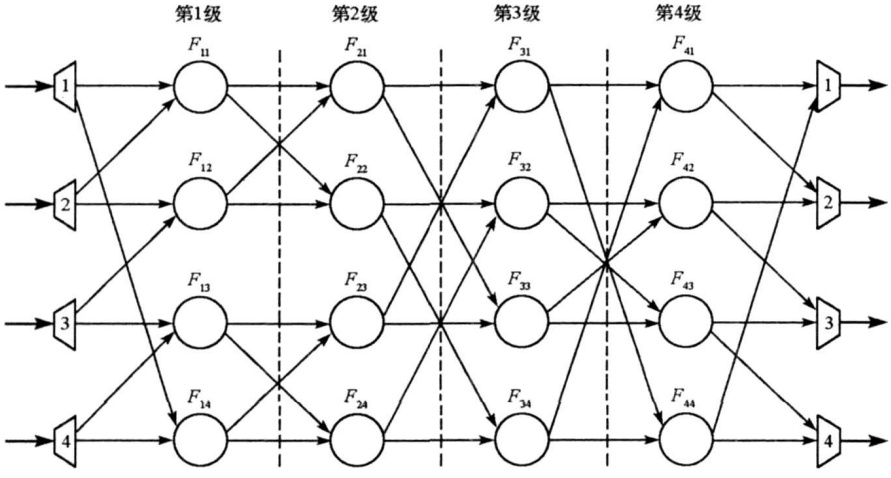

图 1 基于分布式转发交换的路由器体系结构

Fig. 1 A router architecture based on distributed forwarding and switching

为了降低单个 FSN 结点的 IP 查表复杂度, 需要对原始的 IP 路由表进行分解。IP 路由查找的经典算法是基于二叉 Trie 树的软件算法, 如 Radix Trie、Patricia 算法以及 Sklower 对 Patricia 算法的改进算法等。因此我们将 IP 转发表表示为二叉 Trie 树的形式, 在此基础上将原始的 IP 查找树自上而下分解成 S 层, 其中 S 等于路由器流水线的级数。图 2 显示了一个最大级数为 7、包含 9 个前缀的 IP 查找树经过分解后得到的子树结构, 子树的最大级数为 2。需要指出的是, 每一层子树的最大级数可以不同, 为了实现原始 IP 查找树分布的均衡性, 可以根据 IP 查找树的稀疏状况动态调整每一层子树的最大级数。

子树到 FSN 结点的映射算法用于建立子树到 FSN 结点的逻辑映射关系, 它一方面要保证报文流水交换的正确性(报文能够被交换到正确的目的端口) , 另一方面要实现转发表在 FSN 结点的均衡分布。提出一种基于输出端口集合的子树映射算法, 根据子树输出端口集合和 FSN 结点输出端口集合的包含关系来映射子树。子树的输出端口集合定义为在原始的 IP 查找树结构中, 该子树的根结点可到达的所有前缀结点对应的输出端口集合。如图 2 所示, 第 i 层的第 j' 棵子树记做 $ST_{ij'}$, 其输出端口集合表示为

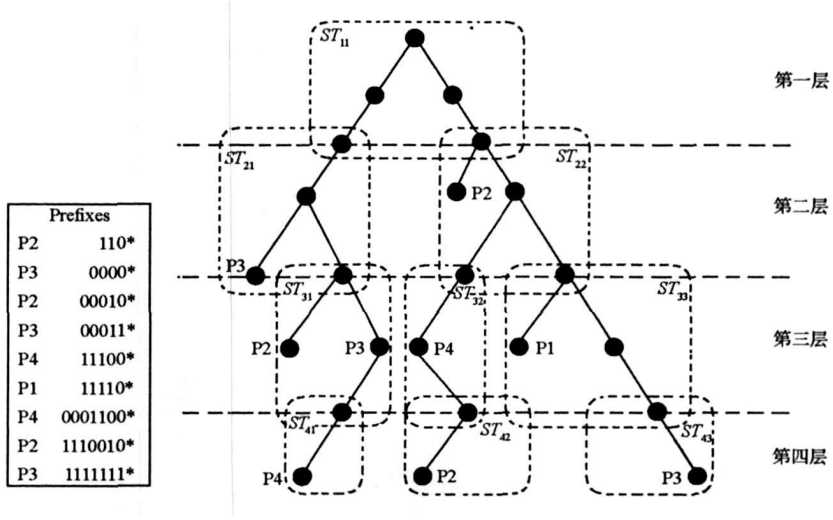

图 2　IP 查找树分解后得到的子树结构

Fig. 2　The trie look-up structure after being partitioned

$STO_{ij'}$, 其中, $1 \leqslant i \leqslant S, j' = 1, 2, 3, \ldots$; 转发交换节点的输出端口集合则定义为位于第 i 级的 FSN 结点可通过第 $i+1$ 级 ……第 N 级的 FSN 交换可达的输出端口集合, 第 i 级的第 j 个 FSN 结点记做 F_{ij}, 其输出端口集合表示为 FO_{ij}, $1 \leqslant j \leqslant K$, K 为每级 FSN 结点的个数。子树到 FSN 结点的映射算法可描述如下:

从第 i 级开始执行以下步骤, 直到 $i = S$, 变量 i 初始化为 1:

(1) 对于第 i 级的每一个转发交换结点 F_{ij}, 将那些输出端口集合 STO_{ij} 包含于 FO_{ij} 的子树映射到 F_{ij}。第 i 层所有子树的集合表示为 Ω_i, 映射到 F_{ij} 的子树集合表示为 Ω_{ij}。如果 $\Omega_i \subseteq \Omega_{i1} \cup \ldots \cup \Omega_{iK}$, 则映射成功, 将变量 i 加 1 返回到步骤 1; 否则, 第 i 层存在不能映射到第 i 级任何转发交换节点的子树, 跳转至步骤(2)。

(2) 采取子树迁移的办法来解决映射失败的问题: 将不能映射的子树迁移到其父子树所在的 FSN 结点, 由于在原始的 IP 查找树中, 子树的输出端口集合具有可包含性, 即任何子树的输出端口集合必定包含其孩子子树的输出端口集合, 因此父子树所在的 FSN 结点的输出端口集合也一定包含被迁移的孩子子树的输出端口集合, 故该迁移操作符合子树的映射条件。

① 将被迁移的孩子子树从第 i 层的子树集合 Ω_i 中删除;

② 将被迁移子树的所有后代子树添加到其上一层的子树集合, 变量 i 加 1 返回到步骤 1。

被迁移的孩子子树并不作为独立的子树而存在, 也不为其分配子树标识, 父子树和所有被迁移的孩子子树作为一个整体构成一棵新的子树。因为任何查找过程都只会先访问父子树, 由后继目的 IP 地址的位信息来触发对孩子子树的访问, 父子树中保存指向孩子子树块的指针。报文到达 FSN 结点时, FSN 结点根据其目的 IP 地址相应的位信息搜索子树, 决定其交换路径。每棵子树包含有效前缀的下一跳信息以及下一级可用的 FSN 结点信息。

2　分布式转发交换实现机制

2.1　树位图算法

FSN 结点采用基于子树的 IP 查找机制, 以子树为逻辑单位进行流水化查找。大多数基于算法的 IP 查找策略均采用树的遍历法, 即从根节点开始搜索, 遍历树的各级, 最后结束于叶节点。它们当中大部分的查找算法既不是以子树为逻辑单位进行搜索也不利于流水实现。通过深入研究各种基于树的查找算法, 我们发现位图表示法能够以固定尺寸存储子树, 并有利于各级 FSN 结点的流水查找。假设子树高度为 n, 那么其孩子子树的数目至多为 2^n, 最大前缀数目为 $\sum_{i=0}^{n-1} 2^i$, 可分别用 2^n bits 和 $\sum_{i=0}^{n-1} 2^i$ bits 的向量

表示孩子子树和有效前缀的分布信息。

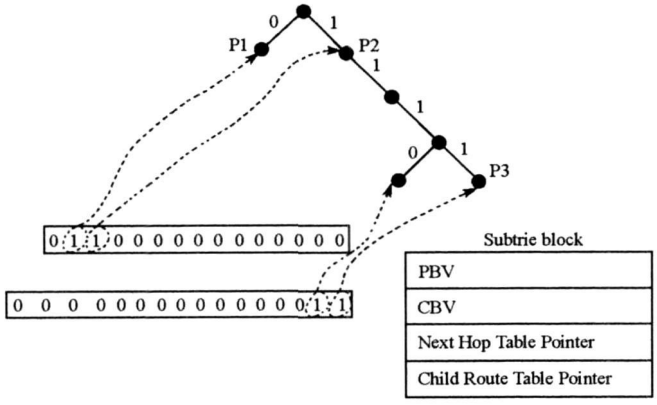

图 3 子树位图表示法

Fig. 3 The bitmap presentation of a subtrie

图 3 显示了高度为 4 的子树位图表示法,任何形式的高度为 4 的子树都可以保存在固定尺寸的子树块中。前缀位向量 PBV(Prefix Bit Vector)描述了子树内与前缀相关的节点。PBV 将原始子树表示为线性格式:按照从上到下从左至右的顺序记录前缀信息。PBV 中的每一位依次对应前缀:* ,0* ,1* ,00* ,01* ,10* ,11* ,…,111* 。图 3 中 PBV 中的两个"1"分别对应前缀 P1(0*)和 P2(1*)。子树中所有前缀的下一跳信息存储在下一跳表连续的空间里,下一跳表指针(Next Hop Table Pointer)记录了第一个有效前缀下一跳信息的位置,通过偏移量计算其他前缀下一跳信息地址。孩子位向量 CBV(Child Bit Vector)描述了孩子子树的分布状态。CBV 长度为 2^4 = 16bits,末尾 2 个"1"分别对应根节点为 1110 和 1111 的孩子子树。需要注意的是,前缀 P3 作为根节点前缀(*)保存在下一级孩子子树中,在这里 P3 作为孩子子树根节点而不是前缀而存在。孩子子树标识及其可用的 FSN 结点的集合保存在孩子路由表一片连续的空间里,子树块保存访问这些信息的基地址即孩子路由表指针(Child Router Table Pointer)。对于给定的 FSN 结点,孩子子树可用的 FSN 结点集合就是那些与其相连且包含该孩子子树的 FSN 结点。

采用类似于文献[3]所提出的树位图算法对子树块进行搜索,假设子树高度为 h,树位图算法以 h 为步长,读取目的 IP 地址对应的信息位执行以下操作:

(1)读位向量 CBV,若第 $S_3S_2S_1S_0$ 位为"1",计数命中位左边"1"的个数(包括命中的"1"本身),以此为偏移量,根据孩子路由表指针读取孩子路由表,获取孩子子树标识及其可用的 FSN 结点的集合。

(2)读位向量 PBV,按照 $S_3S_2S_1$* ,S_3S_2* ,S_3* ,* 的顺序匹配 PBV,一旦命中 PBV,计数命中位左边"1"的个数(包括命中的"1"本身),以此为偏移量根据下一跳表指针读取匹配前缀的下一跳信息。

步骤(1)和步骤(2)并行执行可提高查找速率。为了避免回溯并实现最长前缀匹配,需要记录最近匹配前缀的下一跳信息,若下一级子树没有匹配的前缀也没有匹配的孩子子树,那么最近匹配前缀就是最长匹配前缀。步骤(2)在搜索匹配前缀时,从最长前缀开始搜索,一旦命中则结束本次匹配,记录命中前缀的下一跳信息。

2.2 IP 查找优化策略

2.2.1 初始阵列优化

几乎所有的 IP 查找算法都可以通过初始阵列优化来提高查找速率[4-5]。初始阵列优化的基本思想是,通过在搜索树的顶端牺牲少量的存储开销达到大大减少查找次数的目的。若初始阵列为 8bits,则表示目的 IP 地址的前 8 位将用于检索包含 2^8(= 256)个条目的初始阵列。每个条目至多存储一个前缀信息和一个孩子子树标识。初始阵列优化的缺点是更新性能差,例如更新前缀 0000* 时,需要检查 00000000 到 00001111 的所有条目,其中所有前缀长度小于 4(更新前缀 0000* 长度为 4)的前缀都要被更新。在硬件实现中,可以采用三端口 RAM 存储初始阵列,一个读端口用于查表,另外一读一写两个端口

用于更新操作, 查找和更新操作可同时进行, 查找性能不会受到慢速更新的影响。

2.2.2　子树存储优化

有统计显示, 大多数的子树只包含少量的前缀。文献[6] 对子树内前缀数目的分布信息进行统计, 初始步长为8, 子树高度为4, 统计结果显示: 超过50% 的子树不包含任何前缀信息; 大多数子树只包含少量的前缀, 其中含1个前缀的子树又比含多个前缀的子树的总和还要多。因此, 树位图算法采用特殊的存储方式来保存那些含少量前缀的子树, 这就是CAM 块。CAM 块由CAM 条目构成, 每个CAM 条目由匹配位、匹配长度和下一跳信息构成。以步长为4 的子树为例, 匹配位为4bits, 匹配长度需要2bits 来存储。这样前缀的下一跳信息可以从下一跳表中移除, 避免了子树查找的第二次访存, 提高了查找效率。采用CAM 块存储末端子树时, 超过50% 的下一跳信息从下一跳表中转移到CAM 块中。

IP 查找结构中存在既不包含前缀也不包含孩子子树的"单路径"子树。为了进一步降低存储开销, 通过在子树块中引入跳步长度的方法避免对"单路径"子树的存储, 这就是路径压缩策略。子树高度为4、跳步长度为1 时, 直接忽略目的IP 地址中对应本级的4bits 地址信息, 使用下一级4bits 地址信息进行子树搜索操作。

2.3　FSN 转发引擎设计

子树步长的选择对IP 查找性能有着极大的影响, 步长越大, 存储空间利用率越低, 存储管理和前缀更新机制也越复杂。综合多方面因素考虑, 我们将子树步长定为4, 初始步长为12。如图4 所示, 第一级FSN 结点采用初始步长为12 的初始阵列, 初始阵列的每个条目保存: 前缀长度, 前缀下一跳信息(目的端口), 下一级孩子子树标识。为了进一步降低存储开销, 孩子子树标识只在其所属层保持唯一特性, 而不是全局唯一, 既然子树查找操作是分级流水进行的, 因而这样做并不会影响IP 查找的正确性。假设端口数目 $N = 1024$, 那么初始阵列大小为 $(4 + 10 + 12) \times 2^{12}$ bits $\approx$ 13KB, 而文献[6] 中12bits 的初始阵列大小为28KB。为了保持子树在FSN 结点分布的均衡性, 我们将第二层子树全部映射到第一级FSN 结点(不违反基于输出端口集合的子树映射规则, 因为第一级的每个FSN 结点一定能够到达所有的输出端口)。第一级IP 查找结构如图4 所示: 转发引擎首先根据目的IP 地址的高12 位访问初始阵列, 若命中前缀则记录下一跳信息; 然后以孩子标识为地址读取相应的子树块, 计算访问孩子路由表和下一跳表的地址; 若命中CBV, 读孩子路由表, 获取下一级孩子子树标识及其可用的FSN 结点集合; 若命中PBV, 读下一跳表, 记录匹配前缀的下一跳信息, 孩子路由表和下一跳表可并行访问。这样, 经过三次访存完成了第一级报文转发操作。第二级和第三级FSN 结点采用相同的报文转发机制, 结构如图4 红色虚线所示。与第一级FSN 结点转发引擎不同的是, 二、三级FSN 结点转发引擎需要维护子树标识到子树块的映射表, 根据映射表访问子树块, 而不是通过子树标识直接访问。

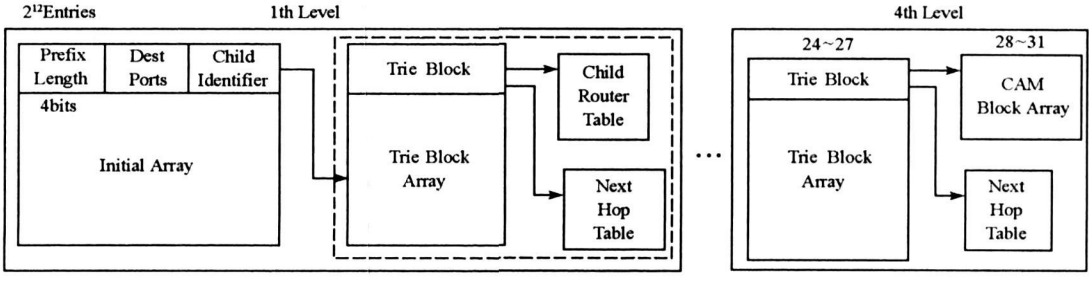

图4　4 级FSN 结点流水转发结构

Fig. 4　The pipelining IP-lookup structure for four stages of FSN nodes

28~ 31 地址段全是末端子树, 即不含孩子子树的子树, 只需保存其前缀信息; 另一方面, 含4 个以上前缀的末端子树较少, 约占全部末端子树的20%, 因此我们采用CAM 块存储28~ 31 地址段的末端子树并将其合并到第4 级转发结构中(注: 在执行基于输出端口集合的子树映射算法时, 将24~ 31 地址段的子树集合映射到第4 级FSN 结点)。如图4 所示, 转发引擎读取子树块, 若命中前缀, 获取下一跳信息; 若命中孩子子树, 则获取下一级子树CAM 块地址。每个CAM 块保存3 个前缀及其下一跳信息, 可

开辟额外的存储空间保存少量的溢出前缀,为提高查找性能,子树的前 3 个最长前缀保存在 CAM 块中。

每一级 FSN 结点采用 CAM 块存储前缀数目小于等于 3 的末端子树,既降低了存储开销,又减少了访存次数,CAM 块和子树块设有跳步长度,用于路径压缩操作,消除了“单路径”子树开销。为简化存储管理,CAM 块和子树块具有相同尺寸。

2.4 交换部件设计

在报文流水转发的过程中,一旦匹配最长前缀,则根据最长匹配前缀的下一跳信息为报文附上目的端口,后继 FSN 结点根据报文的目的端口交换报文,无须再进行子树查找操作。目的端口对应的下一级可用 FSN 结点就是那些输出端口集合中包含了该输出端口的下一级 FSN 结点,这些路由信息完全由 FSN 结点的互联拓扑结构决定,与子树的映射结果无关。为了确保报文能够被发送到正确的目的端口,FSN 结点需保存其输出端口集合中每个输出端口的下一级可用 FSN 结点集合,这些和输出端口相关的路由信息保存在 FSN 结点交换部件的交换表中(Switching Table)。经转发引擎处理后的报文可分为两种:命中 CBV 还需进一步匹配前缀的报文;没有命中 CBV 已完成最长前缀匹配的报文。对于第一种报文,交换部件根据交换表来确定下一级 FSN 结点;对于第二种报文,交换部件需根据孩子子树下一级可用的 FSN 结点集合确定下一级 FSN 结点。

接下来的问题是:如何从多个可用的 FSN 结点中选择有利于降低报文延迟、提高系统吞吐率的下一级 FSN 结点。我们采用了一种基于信用的流控机制:下一级 FSN 结点向上一级 FSN 结点发送信用值,通知可用缓冲区容量;上一级 FSN 结点交换部件选择信用值大的 FSN 结点发送报文,如果报文长度大于最大信用值则等待。基于信用的流控机制能有效提高缓冲区利用率、系统吞吐率,并在一定程度上避免了缓冲区溢出。

3 总结

本文提出了一种分布式转发交换的并行路由器体系结构,是目前唯一将转发技术和交换技术有机融合在一起的新型路由器体系结构。随着该体系结构关键技术的突破,路由器大量的硬件资源将从繁重的 FIB 处理中释放出来,专注于组播、QoS、安全等方面的应用。下一步的研究内容包括:结合网络报文流的特性,分析 FSN 结点的拓扑连接关系对负载均衡和报文乱序的影响,研究路由器内部报文保序和流量集中问题及如何在 FSN 结点中支持 IPv6 转发。

参考文献:

[1] Huston G. CIDR Report[R/OL]. http://www.cidr-report.org/, 2008.

[2] Iyer S, McKeown N. Making Parallel Packet Switches Practical[C]//Proc. of IEEE Infocom., 2001: 1680- 1687.

[3] Baboescu F, Rajgopal S, Huang L B, et al. Hardware Implementation of a Tree Based IP Lookup Algorithm for OC768 and Beyond[C]//Proc. of DesignCon., 2005: 1680- 1687.

[4] Degermark M, Brodnik A, Carlsson S, et al. Small Forwarding Tables for Fast Routing Lookups[C]//Proc. of ACM SIGCOMM' 97, Cannes (14- 18 September 1997), 1997.

[5] Waldvogel M, Varghese G, Turner J, et al. Scalable High Speed IP Routing Lookups[J]. ACM Transactions on Computer Systems, 2001, 19(4): 440- 482.

[6] Eatherton W. Hardware-based Internet Protocol Prefix Lookups. Hardware-based Internet Protocol Prefix Lookups[D]. Washington University Electrical Engineering Department, 1999.