当收到一个数据包后，路由器需要根据其目的地址所匹配的路由表项来决定将该数据包送往哪个端口。无类别域间路由技术（CIDR，classless interdomain routing）将这一做法改为寻找与数据包目的地址拥有相同前缀的路由表项，但这样做可能会出现一个数据包同时与多条路由表项相匹配的情况，此时我们认为拥有最长前缀匹配的路由表项具有最高的优先级。随着网络带宽的不断提升，路由器处理数据包的延时便成为了提升网络整体性能的瓶颈。为提高路由器处理数据包的效率，人们开始尝试将简单的线性搜索、关联搜索等算法在硬件上实现，其中，三态内容寻址存储器（TCAM，ternary content-addressable memory）得到了广泛的商业应用。TCAM中的每个晶元都拥有三种状态：0,1，或者 “不关心”。利用第三种状态，TCAM可以灵活设置每条路由表项待匹配前缀的长度，或者直接将其设置为通配表项。TCAM将所有的路由表项依照优先级的顺序存储，这样首次匹配成功的路由表项即是数据包的最终匹配表项。采用这种方法，TCAM很好地解决了CIDR匹配中的优先级问题，这使得TCAM在进行数据包的路由表查询时非常快，但也带来了新的问题，当路由表发生改变，如增加或者删除了某些路由表项时，TCAM需要维护路由表项之间的优先级。关于这个问题，最简单的解决办法是在找到新表项的正确位置后移动所有的后续表项：

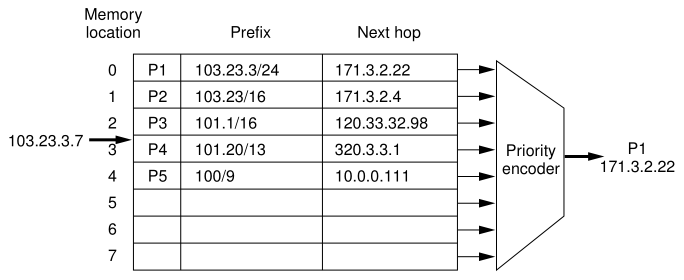


图 1

如图 1，当新的路由表项插入3号内存区后，P4、P5两条表项均需要向下移动，显然，这种方法需要次（N为路由表的总条数）内存拷贝操作。在此基础上的一种改进措施是将预留空间分散化，如图2

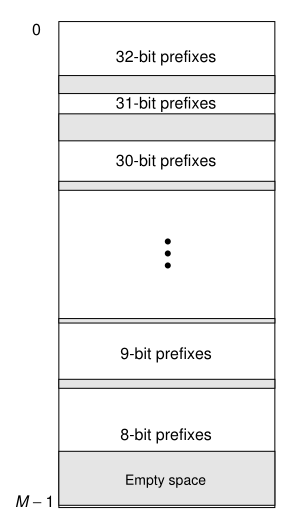


图2

但分散地预留空间会浪费大量的内存空间，而且这种做法在最坏情况下也需要移动次内存。一个更好的做法是，我们可以将预留的内存空间放置在中部，如图3

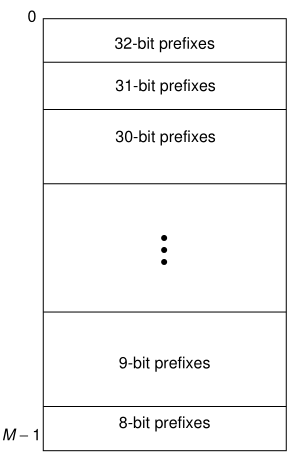


图3

这样一来我们需要的内存移动次数仅为图1中的一半，将集中化的预留空间也避免了内存空间的大量浪费。这或许对TCAM提升维护路由表的有序性的性能有所帮助，但还不够。其实我们只需要维护具有相同前缀的路由表项间的优先级。基于这一发现，D Shan等人提出了CAO\_OPT算法[1]。该算法利用里字典树（Trie）来组织路由表项。而且仍旧将预留内容置于中部，如图4

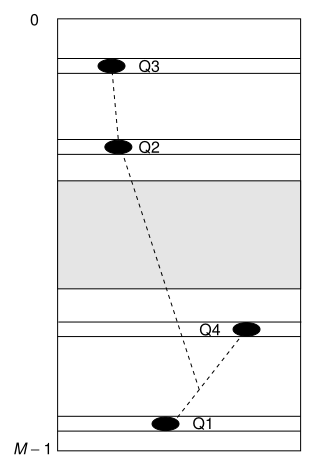


图4

鉴于字典树结构的特性，具有相同前缀的结点均在同一条路径上，因此当发生路由表项更改时，我们只需要维护字典树中一条上的结点顺序。下面来看一下具体的操作过程：

1）插入新的路由表项

当插入的新表项位于预留区上方（较小地址区）时

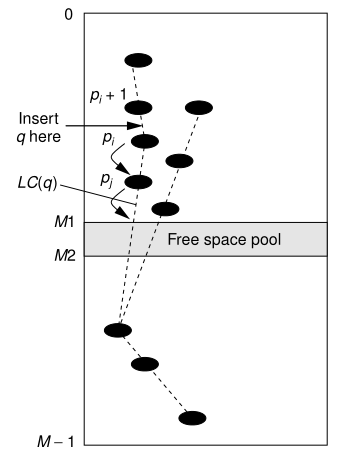


图5

我们只需要将从该位置原结点到其根路径上最邻近预留内存区的结点范围内的路由表项进行移动（如图5），便可以完成路由表项优先级的维护工作。

当插入的新表项位于预留区下方（较大地址区）时

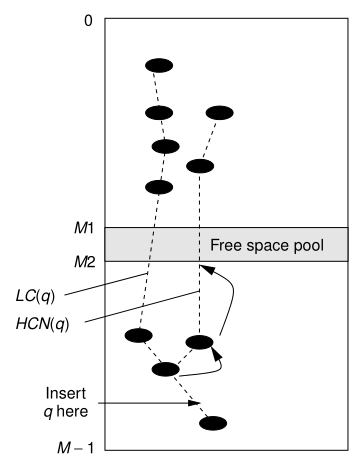


图6

在找到新结点应该插入的位置后，我们每次选取一个离此结点最近的子结点，以此后推，直至遇到预留的内存区域，然后调整这些结点的位置以维护结点的有序性（如图6）。

2）删除旧的路由表项。

删除路由表项可以看做在对应的位置插入一条“空白”的路由表项，再将该“空白”路由表项依照上述插入过程移动至中间的预留内存区域。

可以看到，利用字典树结构组织起来的路由表项在维护有序性时只需要移动次内存（D是字典树中根到叶节点的最大路径长度）。

使用字典树来应对以最长前缀匹配为优先规则的路由表项的更新问题看起来是一个不错的办法。但这种优先规则并不通用，在应对具有一般性的优先级的路由表更新问题上，Haoyu Song等给出了一些解决方案[2]。同样的，这里也将 路由表项在TCAM中存储的位置作为其优先级，存储在地址编号更小的路由表项具有更高的优先级，在进行路由查找时会被先检索到。注意，即使是考虑一般化的优先级，优先规则仍旧只会在具有相同前缀的路由表项间有意义，只不过相比于最长前缀匹配，此时较短的前缀匹配可能具有更高的优先级。Haoyu Song等给出的方法是将路由表项依照优先级划为若干集合，统一集合内部的路由表项不存在匹配冲突，可以随意调换位置，而不同的集合之间存在匹配冲突，不能调换位置。图7所示的情况是一个具体的例子：

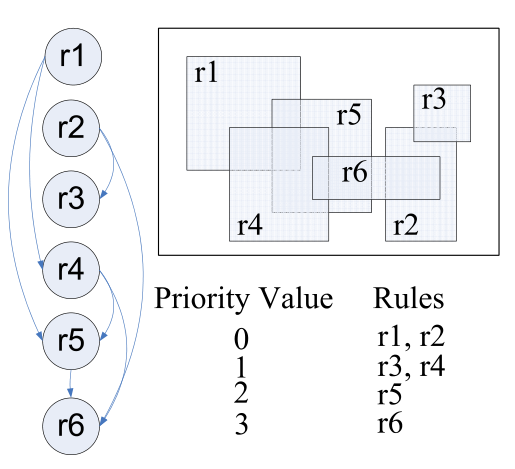


图7

图7中每个结点代表一条路由表项，用带箭头的线连接的两个结点具有匹配冲突，其箭头指向优先级更低的结点，如r1与r4有相同的前缀，存在匹配冲突，且r1的优先级高于r4，r1存储在地址编号更小的内存段。标识出路由表中所有的冲突关系后，我们可以根据这幅有向无环图给所有的路由表项设置一个具体的优先级的值。在图7中，先给没有任何入度的结点赋值，即将r1、r2的优先级赋值为0，然后将r3、r4的优先级赋值为1，依次类推。给所有的路由表项的优先级都赋值后，具有相同优先级值的路由表项便划分为一个集合。实践发现，优先级不同的值的个数远远小于路由规则的条数，如图8：

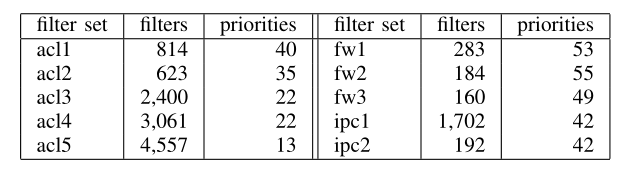


图8

因此，将处理不同的路由表项转变为处理不同的优先级将带来很大的方便。Haoyu Song等还给出了针对优先级的搜索方案，图9描述了如何搜索一个具有32个不同的优先级值的路由表：

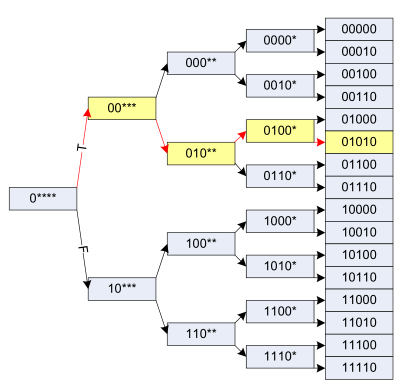


图9

先行搜索优先级范围（0b0xxxx）。若发现匹配的路由表项，则继续搜索（0b00xxx）；否则搜索（0b10xxx），依次类推。

在处理一般化的优先级问题上，Tania Mishra等提出了基于DUOS改进的PC-DUOS[3]方法。这里同时使用了两块TCAM，如图10

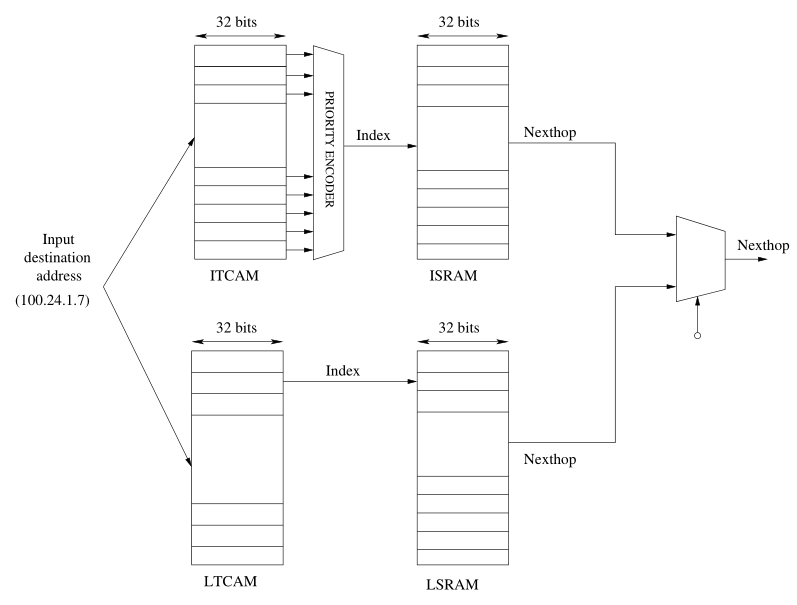


图10

每块TCAM均有一块SRAM与之相连，TCAM中存贮的是路由表项中待匹配的前缀，而SRAM中存储是与之相对应的动作。两块TCAM分别为ITCAM（Interior TCAM）和LTCAM（Leaf TCAM）。DUOS也使用字典树来管理路由表项的前缀，将字典树中的叶节点对应的路由表项存储在LTCAM中，将其余的叶节点对应的路由表项存储在ITCAM中。因此，LTCAM中存储的路由表项间不存在前缀匹配冲突，所以LTCAM中的路由表现也不需要优先级。在处理数据包时，同时查询ITCAM和LTCAM，如果在LTCAM中查找到了相匹配的路由表项，则直接将该路由表项最为最终匹配结果，终止ITCAM查询。

一旦我们在LTCAM中找到了相匹配的路由表箱，则立即将其作为最终匹配结果，立即终止在ITCAM中的查找。为了保证这种查找策略的正确性，我们必须保证如下两点：

1. 每个数据包在LTCAM中多有只能有一条相匹配的路由表项。
2. 若一个数据包在LTCAM中匹配到一条路由表项，这条路由表项必须是该数据包在路由表中能匹配到的最高优先级路由表项。

对于一条由以下域组成的路由表项

PC-DUOS按照每个域对应一颗字典树，将路由表组织成一棵多级字典树。为了使查找性能达到最优，应该将路由表中的最大前缀不相交集放入LTCAM中，但寻找这一集合并不容易，权衡之下，PC-DUOS改为寻找一个较大的前缀不相交集——“叶叶集”，如图12

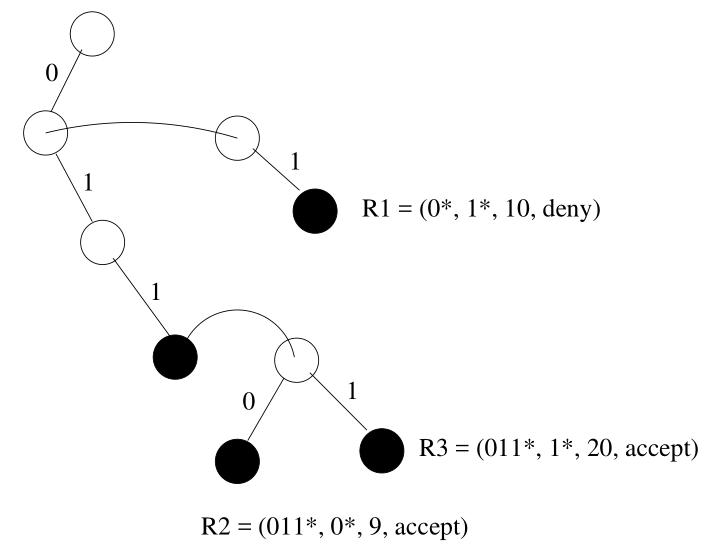


图12

图12中描述了路由表项的两个域，目的地址和源地址。R1、R2、R3在源地址字典树中位于叶子节点，但为了保证条件1的满足，PC-DUOS将R1对应的路由表项置于ITCAM中，由于其所在字典树的根节点不是从上一级字典树的叶节点派生出来的。另外，为了保证条件2的满足，PC-DUOS将R3对应的路由表项也置于ITCAM中，尽管其所在的字典树的根节点是从上一级字典树的叶节点派生出来的，但由于其优先级小于已经置于ITCAM中的R1节点，且与R1有前缀匹配冲突。故最终只有R2被置于LTCAM中。

LTCAM中的路由表项无需优先级编号，对ITCAM中的表项，PC-DUOS采取了和[2]中同样的方法。

[1] fast updating algorithms for tcams

[2] Fast Filter Updates for Packet Classification using TCAM

[3] PC-DUOS: Fast TCAM Lookup and Update for Packet Classifiers