(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 106416152 B
(45)授权公告日 2019.09.27

(21)申请号 201480079291.6

(22)申请日 2014.06.10

(65)同一申请的已公布的文献号
申请公布号 CN 106416152 A

(43)申请公布日 2017.02.15

(85)PCT国际申请进入国家阶段日
2016.11.28

(86)PCT国际申请的申请数据
PCT/CN2014/079624 2014.06.10

(87)PCT国际申请的公布数据
WO2015/188319 ZH 2015.12.17

(73)专利权人 华为技术有限公司
地址 518129 广东省深圳市龙岗区坂田华为总部办公楼

(72)发明人 张文勇

(74)专利代理机构 北京同达信恒知识产权代理有限公司 11291
代理人 黄志华

(51)Int.Cl.
H04L 12/745(2006.01)

(56)对比文件
EP 1434148 A3,2008.01.09,
US 2014003436 A1,2014.01.02,
US 2004100960 A1,2004.05.27,
EP 1063827 A2,2000.12.27,

审查员 张述照

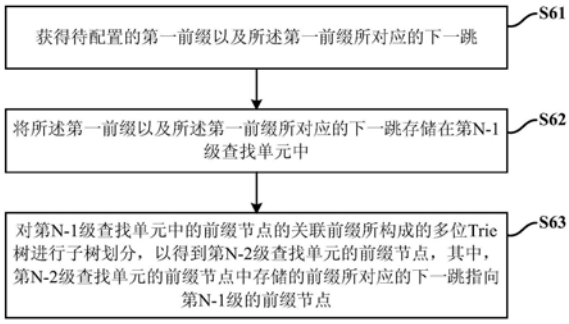
权利要求书4页 说明书20页 附图11页

(54)发明名称

一种查找装置、查找配置方法和查找方法

(57)摘要

一种查找装置、查找配置方法和查找方法，所述查找装置包括N个流水线级，每个流水线级包括一个查找单元，每级查找单元中配置前缀节点，第N-1级查找单元配置的前缀节点是通过对查找表构成的多位Trie树进行子树划分得到的，第N-2级查找单元中配置的前缀节点是通过对第N-1级查找单元中配置的前缀节点的关联前缀构成的多位Trie树进行子树划分得到的，通过多次迭代进行前缀节点配置。采用本发明提供的查找装置，可以减少内存资源的占用和流水线级数，进而减少查找延迟并降低实现难度。



CN 106416152 B

1.一种查找装置,其特征在于,包括:

N个流水线级,其中N>1;

每个流水线级中包括一个查找单元,按照所述N个流水线级执行的先后顺序,分别为第0级至第N-1级查找单元;每级查找单元中包括前缀节点,用于存储前缀以及与所述前缀所对应的下一跳,其中:

第i+1级查找单元中配置有第一前缀节点,第i级查找单元中配置有第二前缀节点,其中0<=i<=(N-2);所述第二前缀节点中存储的前缀所对应的下一跳指向所述第一前缀节点,所述第二前缀节点通过对所述第一前缀节点的关联前缀所构成的第一多位Trie树进行子树划分得到;

所述第二前缀节点通过对所述第一前缀节点的关联前缀所构成的第一多位Trie树进行子树划分得到包括:将所述第一多位Trie树划分为M个子树,M>=1,每个子树包括所述第一多位Trie树的一个Trie节点以及所述Trie节点的K个分支,根据所述M个子树得到所述第二前缀节点,其中0<=K<=stride,所述stride为所述第一多位Trie树的步长。

2.如权利要求1所述的装置,其特征在于,所述第二前缀节点包括1至2^{stride}个间接前缀节点,或者包括0至2^{stride}个间接前缀节点和一个直接前缀节点;其中,直接前缀节点用于存储落入该子树的Trie节点内的前缀,间接前缀节点用于存储落入该子树的Trie节点的一个分支内的前缀。

3.如权利要求2所述的装置,其特征在于,所述第i+1级查找单元中还配置有所述第一多位Trie树中包含有前缀节点的Trie节点。

4.如权利要求1所述的装置,其特征在于,所述第一前缀节点中存储第一数组或所述第一数组的地址,所述第一数组中存储所述第一前缀节点的每个前缀所对应的下一跳指针;和/或

所述第二前缀节点中存储第二数组或所述第二数组的地址,所述第二数组中存储所述第二前缀节点的每个前缀所对应的下一跳指针。

5.如权利要求1所述的装置,其特征在于,所述第0级查找单元为一个三态内容寻址存储器TCAM。

6.如权利要求1-5中任一项所述的装置,其特征在于,所述N个流水线级用于执行以下步骤:

S91:接收查找关键字,转入S92;

S92:设置i=0,转入S93;

S93:第i级查找单元用所述查找关键字与第i级查找单元中的第一前缀节点容纳的前缀进行匹配,若匹配到前缀,则将匹配到的前缀的长度以及匹配到的前缀所对应的下一跳输出给第i+1级查找单元,转入S94,否则,转入S96;

S94:设置i=i+1,并转入S95;

S95:第i级查找单元根据第i-1级查找单元输出的前缀的长度,从所述查找关键字中提取相应长度的分段关键字,根据第i-1级查找单元输出的下一跳确定所述第i级查找单元中对应的第二前缀节点,用提取出的分段关键字与所述第二前缀节点所存储的前缀的相应分段关键字进行匹配;

若匹配到前缀,当i=N-1时,输出匹配到的前缀所对应的下一跳;否则,将匹配到的前

缀的长度以及匹配到的前缀所对应的下一跳输出给第i+1级查找单元，转S94；

若没有匹配到前缀，且所述第二前缀节点有最长前缀匹配PLPM，则输出所述PLPM对应的下一跳；如果所述第二前缀节点没有PLPM，则转入S96；

S96：输出未查找到匹配前缀的提示信息。

7.一种查找配置方法，其特征在于，所述方法包括：

S61：获得待配置的第一前缀以及所述第一前缀所对应的下一跳，转入S62；

S62：将所述第一前缀以及所述第一前缀所对应的下一跳存储在第N-1级查找单元中，转入S63；其中，N为流水线级的数量，N>1，每个流水线级中包括一个查找单元，按照所述N个流水线级执行的先后顺序，分别为第0级至第N-1级查找单元；每级查找单元中包括前缀节点，用于存储前缀以及与所述前缀所对应的下一跳；

S63：对第N-1级查找单元中的前缀节点的关联前缀所构成的多位Trie树进行子树划分，以得到第N-2级查找单元的前缀节点，其中，第N-2级查找单元的前缀节点中存储的前缀所对应的下一跳指向第N-1级的前缀节点；所述对第N-1级查找单元中的前缀节点的关联前缀所构成的多位Trie树进行子树划分，以得到第N-2级查找单元的前缀节点，包括：将所述多位Trie树划分为M个子树，M>＝1，每个子树包括所述多位Trie树的一个Trie节点以及所述Trie节点的K个分支，其中0<＝K<＝stride，所述stride为所述多位Trie树的步长。

8.如权利要求7所述的方法，其特征在于，

所述S62包括：

判断第N-1级查找单元中的前缀节点的容量是否允许存储所述第一前缀以及所述第一前缀所对应的下一跳，若是，则将所述第一前缀以及所述第一前缀所对应的下一跳存储于所述第N-1级查找单元的前缀节点中；否则，重新配置所述第N-1级查找单元的前缀节点，将所述第一前缀以及所述第一前缀所对应的下一跳存储于所述第N-1级查找单元重新配置后的前缀节点；

所述S63包括：

根据第N-2级查找单元中的前缀节点的容量判断是否允许存储第N-1级查找单元重新配置的前缀节点的关联前缀，若是，则将第N-1级查找单元重新配置的前缀节点的关联前缀存储于第N-2级查找单元的前缀节点；否则，转入S64；

S64：设置j＝N-2，转入S65；

S65：通过对第j+1级查找单元中的前缀节点的关联前缀所构成的多位Trie树进行子树划分来重新配置第j级查找单元的前缀节点，将第j+1级查找单元中配置的前缀节点的关联前缀存储于第j级查找单元重新配置后的前缀节点，转入S66；

S66：根据第j-1级查找单元的前缀节点的容量判断是否允许存储第j级查找单元重新配置的前缀节点的关联前缀，若是，则将第j级查找单元重新配置的前缀节点的关联前缀存储于第j-i级查找单元的前缀节点，并结束流程，否则转入S67；

S67：设置j＝j-1，并转入S68；

S68：若j＝0则结束流程，否则，转入S65。

9.如权利要求8所述的方法，其特征在于，所述重新配置第N-1级查找单元的前缀节点，包括：

将所述第一前缀插入第N-1级查找单元对应的第一多位Trie树；

若第N-1级查找单元中存在拆分目标前缀节点，则根据所述第一前缀在第N-1级查找单元对应的第一多位Trie树中的位置、所述第一多位Trie树的步长以及所述拆分目标前缀节点所存储的前缀在所述第一多位Trie树中的位置，将所述拆分目标前缀节点拆分为多个前缀节点，拆分得到的一个前缀节点中存储有所述第一前缀以及所述第一前缀所对应的下一跳，否则，在第N-1级查找单元中新配置一个前缀节点以存储所述第一前缀以及所述第一前缀所对应的下一跳；其中，所述目标前缀节点满足第一条件且是所有满足所述第一条件的前缀节点中距离所述第一前缀的步长最少的前缀节点，所述第一条件为：前缀节点所对应的子树的分支上包含所述第一前缀；

所述通过对第j+1级查找单元中的前缀节点的关联前缀所构成的多位Trie树进行子树划分来重新配置第j级查找单元的前缀节点，包括：

若第j级查找单元中存在拆分目标前缀节点，则根据所述第一前缀在第j级查找单元所对应的第二多位Trie树中的位置、所述第二多位Trie树的步长以及所述拆分目标前缀节点所存储的前缀在所述第二多位Trie树中的位置，将所述拆分目标前缀节点拆分为多个前缀节点，拆分得到的一个前缀节点中存储有第j+1级查找单元中新配置的前缀节点的关联前缀以及对应的下一跳，否则，在第j级查找单元中新配置一个前缀节点以存储所述第j+1级查找单元中新配置的前缀节点的关联前缀以及对应的下一跳；其中，所述目标前缀节点满足第一条件且是所有满足所述第一条件的前缀节点中距离所述第j+1级查找单元中新配置的前缀节点的关联前缀的步长最少的前缀节点，所述第一条件为：前缀节点所对应的子树的分支上包含所述第j+1级查找单元中新配置的前缀节点的关联前缀。

10.如权利要求7至9中任一项所述的方法，其特征在于，还包括：

S71：获得待删除的第二前缀以及所述第二前缀对应的下一跳；

S72：从第N-1级查找单元中存储有所述第二前缀以及所述第二前缀对应的下一跳的前缀节点中释放所述第二前缀以及所述第二前缀的下一跳；

S73：若释放所述第二前缀后的前缀节点中，已不再存储有前缀以及所述前缀的下一跳，则释放该前缀节点，并转入S74；

S74：设置j＝N-2，转入S75；

S75：针对第j+1级查找单元中被释放的前缀节点，从第j级查找单元中的前缀节点中释放相应前缀节点的关联前缀，若释放关联前缀后的前缀节点中已不再存储有关联前缀，则释放第j级查找单元中已不再存储有关联前缀的前缀节点，并转入S76；

S76：设置j＝j-1；转入S77；

S77：若j<0，则结束流程，否则转入S75。

11.一种查找方法，其特征在于，包括：

S91：查找装置接收查找关键字，转入S92；其中，所述查找装置包括N个流水线级，N>1；每个流水线级中包括一个查找单元，按照所述N个流水线级执行的先后顺序，分别为第0级至第N-1级查找单元；

S92：设置i＝0，转入S93；

S93：第i级查找单元用所述查找关键字与第i级查找单元中的第一前缀节点中存储的前缀进行匹配，若匹配到前缀，则将匹配到的前缀的长度以及匹配到的前缀所对应的下一跳输出给第i+1级查找单元，转入S94，否则，转入S96；

S94:设置i＝i+1,并转入S95;

S95:第i级查找单元根据第i-1级查找单元输出的前缀的长度,从所述查找关键字中提取相应长度的分段关键字,根据第i-1级查找单元输出的下一跳确定所述第i级查找单元中对应的第二前缀节点,用提取出的分段关键字与所述第二前缀节点所存储的前缀的相应分段关键字进行匹配;

若匹配到前缀,当i＝N-1时,输出匹配到的前缀所对应的下一跳;否则,将匹配到的前缀的长度以及匹配到的前缀所对应的下一跳输出给第i+1级查找单元,转入S94;

若没有匹配到前缀,且所述第二前缀节点有最长前缀匹配PLPM,则输出所述PLPM所对应的下一跳;如果所述第二前缀节点没有PLPM,则转入S96;

S96:输出未查找到匹配前缀的提示信息。

一种查找装置、查找配置方法和查找方法

技术领域

[0001] 本发明涉及网络通信领域，尤其涉及一种查找装置、查找配置方法和查找方法。

背景技术

[0002] 路由器的主要任务是进行网际协议(Internet Protocol，简称IP)报文转发，也就是根据报文头中的目的IP地址将到达路由器输入端口的报文转发到正确的输出端口。路由查找就是根据报文的目的IP地址，查找路由器中的路由表，得到报文下一跳信息的过程。

[0003] 路由查找采用最长前缀匹配(longest prefix match，简称LPM)原则。当有多个前缀与输入的IP地址匹配时，与该IP地址匹配的前缀中掩码最长的前缀对应的下一跳信息为最终的查找结果。

[0004] 常用的路由查找算法包括三态内容可寻址存储器(Ternary Content Addressable Memory，简称TCAM)算法和基于Trie树的算法等。

[0005] TCAM算法可以将输入的IP地址同时与所有的路由表项进行匹配。TCAM的价格昂贵，用TCAM实现路由查找的成本较高。TCAM的集成度比一般的存储器低很多，容量比较有限，很难满足百万量级路由表的处理。另外，TCAM功耗也比较大。因此目前更为常用的是多位Trie算法。

[0006] 基于Trie树(也称前缀树)的算法根据前缀中的位串建立一棵二叉树或多叉树。如果每次考虑一位，则建立一棵二叉树，也称为单位Trie树。图1示出了一棵单位Trie树，其中11个前缀，图1中左侧p0～p11，在单位Trie树中对应的节点用黑色圆圈表示，连接点用白色圆圈表示，每个节点用唯一编号标识。如果每次考虑多位，则建立一棵多位Trie树。在多位Trie树中，每次考虑的位数一般是固定的，称为Trie树的步长(英文:stride)。

[0007] 多位Trie树可以看作按stride把一个单位Trie树分成多个子树，并为每个子树创建一个Trie节点。每个Trie节点均有一个关联前缀，一个Trie节点的关联前缀是该Trie节点对应的子树的根节点上的前缀值。如果某个子树内分布有前缀，还要为该子树创建一个前缀节点，位于在该子树内的前缀都保存在该前缀节点中，每个前缀对应一条下一跳信息。一般在多位Trie树中只保存下一跳指针，也就是指向下一跳信息的指针。根据stride可将每个前缀分成很多段，每个分段对应的位串称为该前缀在该分段的分段键值(key)。比如，在stride等于3的情况下，前缀‘10101*’的第一个分段键值为‘101’，第二个分段键值为‘01*’。

[0008] 图2示出了基于图1中的前缀建立的stride等于3的多位trie树。该多位Trie树包括7个Trie节点，如图2中所示的Trie Node T1～Trie Node T7，每个Trie节点配置有一个前缀节点，如图2中所示Prefix Node。每个Prefix Node中保存有该Prefix Node对应的Trie节点中的各个前缀的下一跳指针，例如，图2中Trie Node T4对应的Prefix节点(Prefix节点表示前缀节点)中保存有前缀p3的下一跳指针“RE Index of P3”。除了Trie Node T1对应的前缀节点以外的每个前缀节点中，还保存一个最长前缀匹配(present longest prefix match，简称PLPM)。一个Prefix节点的PLPM为覆盖该Prefix节点对应的子

树的最长前缀。图2中Prefix节点中的“RE Index of PLPM”表示该Prefix节点的PLPM的下一跳指针。

[0009] 为了提高路由查找性能，路由查找一般用硬件实现的查找装置。由硬件实现的查找装置一般采用多级的流水线结构。基于多位Trie树的算法可以通过将多位Trie树结构存放在流水线结构的各级中来实现。

[0010] 图3是将图2的多位Trie树放在一个5级的流水线结构中的示意图。如图3所示，流水线结构中的前3级(stage1～stage3)保存3层Trie节点，第4级(stage4)保存7个Trie节点对应的Prefix节点，第5级(stage5)保存每个Prefix节点中的各个前缀对应的RE Index。基于上述流水线结构的查找装置在进行路由查找时，将待匹配的IP地址(即查找关键字)沿着流水线结构一级一级进行查找，最终得到查找结果。比如，查找关键字为111 010 011，则路由查找过程包括：

[0011] 在stage1，访问T1，由于T1中的节点1对应的前缀与任意查找关键字匹配，因此保存T1对应的Prefix节点的指针，并从查找关键字中提取第一个分段key‘111’；

[0012] 在stage2，根据分段key‘111’分别访问T1的子节点T2、T3、T4，由于T4中的节点9对应的前缀p3(即111*)的第一个分段key与查找关键字的第一个分段key‘111’最长匹配，因此更新保存的Prefix节点的指针为T4对应的Prefix节点的指针，并从查找关键字中提取第二个分段key‘010’；

[0013] 在stage3，根据分段key‘010’访问T4的子节点T7，由于T7中的节点21对应的前缀p10(1110100*)的第二个分段key‘010’与查找关键字的第二个分段key‘010’最长匹配，因此更新保存的Prefix节点指针为T7对应的Prefix节点的指针，并从查找关键字中提取第三个分段key‘011’；

[0014] 在stage4，由于T7没有Trie子节点，因此使用最近一次保存的T7对应的Prefix节点指针访问T7对应的Prefix节点，用查找关键字的第三个分段key与T7对应的Prefix节点中各前缀的第三个分段key进行匹配，其中，前缀p10的第三个分段key‘0*’与该查找关键字的第三个分段key‘011’最长匹配，因此将T7对应的Prefix节点中“RE Index of p10”(前缀p10对应的下一跳指针)对应的RE Index数组中的位置作为stage4的输出；

[0015] 在stage5，根据stage4中得到的RE Index数组中的位置得到前缀p10对应的RE Index。

[0016] 最终，根据查找装置返回的RE Index可以得到上述查找关键字对应的下一跳信息。

[0017] 现有基于多位Trie算法实现的路由查找算法，将整个多位Trie树结构放在查找装置中，会占用大量内存资源；并且由于硬件实现的查找装置的流水线结构的级数与多位Trie树的级数对应，因此对于掩码较长的IPv6路由，多位Trie树的级数往往较多，相应流水线结构的级数也较多，导致查找延迟大，也增加了硬件实现的难度。

发明内容

[0018] 本发明实施例提供了一种查找装置、查找配置方法和查找方法，用以减少查找装置中的流水线级数以及存储开销，进而减少查找延迟、降低实现难度，并减少资源占用。

[0019] 第一方面，提供一种查找装置，所述查找装置包括：

[0020] N个流水线级,其中N＞1;

[0021] 每个流水线级中包括一个查找单元,按照所述N个流水线级执行的先后顺序,分别为第0级至第N-1级查找单元;每级查找单元中包括前缀节点,用于存储前缀以及与所述前缀所对应的下一跳,其中:

[0022] 第i+1级查找单元中配置有第一前缀节点,第i级查找单元中配置有第二前缀节点,其中0＜＝i＜＝(N-2);所述第二前缀节点中存储的前缀所对应的下一跳指向所述第一前缀节点,所述第二前缀节点通过对所述第一前缀节点的关联前缀所构成的第一多位Trie树进行子树划分得到;

[0023] 所述第二前缀节点通过对所述第一前缀节点的关联前缀所构成的第一多位Trie树进行子树划分得到包括:将所述第一多位Trie树划分为M个子树,M＞＝1,每个子树包括所述第一多位Trie树的一个Trie节点以及所述Trie节点的K个分支,根据所述M个子树得到所述第二前缀节点,其中0＜＝K＜＝stride,所述stride为所述第一多位Trie树的步长。

[0024] 结合第一方面,在第一方面的第一种可能的实现方式中,所述第二前缀节点包括1至2^{stride}个间接前缀节点,或者包括0至2^{stride}个间接前缀节点和一个直接前缀节点;其中,直接前缀节点用于存储落入该子树的Trie节点内的前缀,间接前缀节点用于存储落入该子树的Trie节点的一个分支内的前缀。

[0025] 结合第一方面的第一种可能的实现方式,在第一方面的第二种可能的实现方式中,所述第i+1级查找单元中还配置有所述第一多位Trie树中包含有前缀节点的Trie节点。

[0026] 结合第一方面,在第一方面的第三种可能的实现方式中,所述第一前缀节点中存储第一数组或所述第一数组的地址,所述第一数组中存储所述第一前缀节点的每个前缀所对应的下一跳指针;和/或

[0027] 所述第二前缀节点中存储第二数组或所述第二数组的地址,所述第二数组中存储所述第二前缀节点的每个前缀所对应的下一跳指针。

[0028] 结合第一方面,在第一方面的第四种可能的实现方式中,所述第0级查找单元为一个三态内容寻址存储器TCAM。

[0029] 结合第一方面以及第一方面的第一种至第四种可能的实现方式中的任一种,在第一方面的第五种可能的实现方式中,所述N个流水线级用于执行以下步骤:

[0030] S91:接收查找关键字,转入S92;

[0031] S92:设置i＝0,转入S93;

[0032] S93:第i级查找单元用所述查找关键字与第i级查找单元中的第一前缀节点容纳的前缀进行匹配,若匹配到前缀,则将匹配到的前缀的长度以及匹配到的前缀所对应的下一跳输出给第i+1级查找单元,转入S94,否则,转入S96;

[0033] S94:设置i＝i+1,并转入S95;

[0034] S95:第i级查找单元根据第i-1级查找单元输出的前缀的长度,从所述查找关键字中提取相应长度的分段关键字,根据第i-1级查找单元输出的下一跳确定所述第i级查找单元中对应的第二前缀节点,用提取出的分段关键字与所述第二前缀节点所存储的前缀的相应分段关键字进行匹配;

[0035] 若匹配到前缀,当i＝N-1时,输出匹配到的前缀所对应的下一跳;否则,将匹配到的前缀的长度以及匹配到的前缀所对应的下一跳输出给第i+1级查找单元,转S94;

[0036] 若没有匹配到前缀，且所述第二前缀节点有最长前缀匹配PLPM，则输出所述PLPM对应的下一跳；如果所述第二前缀节点没有PLPM，则转入S96；

[0037] S96：输出未查找到匹配前缀的提示信息。

[0038] 第二方面，提供一种查找配置方法，所述方法包括：

[0039] S61：获得待配置的第一前缀以及所述第一前缀所对应的下一跳，转入S62；

[0040] S62：将所述第一前缀以及所述第一前缀所对应的下一跳存储在第N-1级查找单元中，转入S63；其中，N为流水线级的数量，N＞1，每个流水线级中包括一个查找单元，按照所述N个流水线级执行的先后顺序，分别为第0级至第N-1级查找单元；每级查找单元中包括前缀节点，用于存储前缀以及与所述前缀所对应的下一跳；

[0041] S63：对第N-1级查找单元中的前缀节点的关联前缀所构成的多位Trie树进行子树划分，以得到第N-2级查找单元的前缀节点，其中，第N-2级查找单元的前缀节点中存储的前缀所对应的下一跳指向第N-1级的前缀节点；所述对第N-1级查找单元中的前缀节点的关联前缀所构成的多位Trie树进行子树划分，以得到第N-2级查找单元的前缀节点，包括：将所述多位Trie树划分为M个子树，M＞＝1，每个子树包括所述多位Trie树的一个Trie节点以及所述Trie节点的K个分支，其中0＜＝K＜＝stride，所述stride为所述多位Trie树的步长。

[0042] 结合第二方面，在第二方面的第一种可能的实现方式中，

[0043] 所述S62包括：

[0044] 判断第N-1级查找单元中的前缀节点的容量是否允许存储所述第一前缀以及所述第一前缀所对应的下一跳，若是，则将所述第一前缀以及所述第一前缀所对应的下一跳存储于所述第N-1级查找单元的前缀节点中；否则，重新配置所述第N-1级查找单元的前缀节点，将所述第一前缀以及所述第一前缀所对应的下一跳存储于所述第N-1级查找单元重新配置后的前缀节点；

[0045] 所述S63包括：

[0046] 根据第N-2级查找单元中的前缀节点的容量判断是否允许存储第N-1级查找单元重新配置的前缀节点的关联前缀，若是，则将第N-1级查找单元重新配置的前缀节点的关联前缀存储于第N-2级查找单元的前缀节点；否则，转入S64；

[0047] S64：设置j＝N-2，转入S65；

[0048] S65：通过对第j+1级查找单元中的前缀节点的关联前缀所构成的多位Trie树进行子树划分来重新配置第j级查找单元的前缀节点，将第j+1级查找单元中配置的前缀节点的关联前缀存储于第j级查找单元重新配置后的前缀节点，转入S66；

[0049] S66：根据第j-1级查找单元的前缀节点的容量判断是否允许存储第j级查找单元重新配置的前缀节点的关联前缀，若是，则将第j级查找单元重新配置的前缀节点的关联前缀存储于第j-i级查找单元的前缀节点，并结束流程，否则转入S67；

[0050] S67：设置j＝j-1，并转入S68；

[0051] S68：若j＝0则结束流程，否则，转入S65。

[0052] 结合第二方面的第一种可能的实现方式，在第二方面的第二种可能的实现方式中，所述重新配置第N-1级查找单元的前缀节点，包括：

[0053] 将所述第一前缀插入第N-1级查找单元对应的第一多位Trie树；

[0054] 若第N-1级查找单元中存在拆分目标前缀节点，则根据所述第一前缀在第N-1级查

找单元对应的第一多位Trie树中的位置、所述第一多位Trie树的步长以及所述拆分目标前缀节点所存储的前缀在所述第一多位Trie树中的位置，将所述拆分目标前缀节点拆分为多个前缀节点，拆分得到的一个前缀节点中存储有所述第一前缀以及所述第一前缀所对应的下一跳，否则，在第N-1级查找单元中新配置一个前缀节点以存储所述第一前缀以及所述第一前缀所对应的下一跳；其中，所述目标前缀节点满足第一条件且是所有满足所述第一条件的前缀节点中距离所述第一前缀的步长最少的前缀节点，所述第一条件为：前缀节点所对应的子树的分支上包含所述第一前缀；

[0055] 所述通过对第j+1级查找单元中的前缀节点的关联前缀所构成的多位Trie树进行子树划分来重新配置第j级查找单元的前缀节点，包括：

[0056] 若第j级查找单元中存在拆分目标前缀节点，则根据所述第一前缀在第j级查找单元所对应的第二多位Trie树中的位置、所述第二多位Trie树的步长以及所述拆分目标前缀节点所存储的前缀在所述第二多位Trie树中的位置，将所述拆分目标前缀节点拆分为多个前缀节点，拆分得到的一个前缀节点中存储有第j+1级查找单元中新配置的前缀节点的关联前缀以及对应的下一跳，否则，在第j级查找单元中新配置一个前缀节点以存储所述第j+1级查找单元中新配置的前缀节点的关联前缀以及对应的下一跳；其中，所述目标前缀节点满足第一条件且是所有满足所述第一条件的前缀节点中距离所述第j+1级查找单元中新配置的前缀节点的关联前缀的步长最少的前缀节点，所述第一条件为：前缀节点所对应的子树的分支上包含所述第j+1级查找单元中新配置的前缀节点的关联前缀。

[0057] 结合第二方面以及第二方面的第一种和第二种可能的实现方式中的任一种，在第二方面的第三种可能的实现方式中，还包括：

[0058] S71：获得待删除的第二前缀以及所述第二前缀对应的下一跳；

[0059] S72：从第N-1级查找单元中存储有所述第二前缀以及所述第二前缀对应的下一跳的前缀节点中释放所述第二前缀以及所述第二前缀的下一跳；

[0060] S73：若释放所述第二前缀后的前缀节点中，已不再存储有前缀以及所述前缀的下一跳，则释放该前缀节点，并转入S74；

[0061] S74：设置j＝N-2，转入S75；

[0062] S75：针对第j+1级查找单元中被释放的前缀节点，从第j级查找单元中的前缀节点中释放相应前缀节点的关联前缀，若释放关联前缀后的前缀节点中已不再存储有关联前缀，则释放第j级查找单元中已不再存储有关联前缀的前缀节点，并转入S76；

[0063] S76：设置j＝j-1；转入S77；

[0064] S77：若j＜0，则结束流程，否则转入S75。

[0065] 第三方面，提供一种查找方法，包括：

[0066] S91：查找装置接收查找关键字，转入S92；其中，所述查找装置包括N个流水线级，N＞1；每个流水线级中包括一个查找单元，按照所述N个流水线级执行的先后顺序，分别为第0级至第N-1级查找单元；

[0067] S92：设置i＝0，转入S93；

[0068] S93：第i级查找单元用所述查找关键字与第i级查找单元中的第一前缀节点中存储的前缀进行匹配，若匹配到前缀，则将匹配到的前缀的长度以及匹配到的前缀所对应的下一跳输出给第i+1级查找单元，转入S94，否则，转入S96；

[0069] S94:设置i＝i+1,并转入S95;

[0070] S95:第i级查找单元根据第i-1级查找单元输出的前缀的长度,从所述查找关键字中提取相应长度的分段关键字,根据第i-1级查找单元输出的下一跳确定所述第i级查找单元中对应的第二前缀节点,用提取出的分段关键字与所述第二前缀节点所存储的前缀的相应分段关键字进行匹配;

[0071] 若匹配到前缀,当i＝N-1时,输出匹配到的前缀所对应的下一跳;否则,将匹配到的前缀的长度以及匹配到的前缀所对应的下一跳输出给第i+1级查找单元,转入S94;

[0072] 若没有匹配到前缀,且所述第二前缀节点有最长前缀匹配PLPM,则输出所述PLPM所对应的下一跳;如果所述第二前缀节点没有PLPM,则转入S96;

[0073] S96:输出未查找到匹配前缀的提示信息。

[0074] 本发明的上述实施例中,查找装置的N个流水线级中,每级查找单元中包括前缀节点,用于存储前缀以及与所述前缀对应的下一跳,其中:第i+1级查找单元中配置有第一前缀节点,第i级查找单元中配置有第二前缀节点,所述第二前缀节点中存储的前缀对应的下一跳指向所述第一前缀节点,所述第二前缀节点通过对所述第一前缀节点的关联前缀所构成的第一多位Trie树进行子树划分得到。划分得到的一个子树的覆盖范围包括多位Trie树中的一个Trie节点以及所述Trie节点的K个分支,划分得到的一个子树对应一组前缀节点,因此,一方面由于仅将前缀节点配置在查找单元中,与现有技术相比减少了内存资源的占用;另一方面,由于前一级的查找单元中的前缀节点是基于后一级的查找单元中的前缀节点的关联前缀构成的多位Trie树进行子树划分得到的,使得一个前缀节点所对应的子树的覆盖范围较大,减少了流水线级数,进而减少查找延迟并降低实现的难度。

附图说明

[0075] 为了更清楚地说明本发明实施例中的技术方案,下面将对实施例描述中所需要使用的附图作简要介绍,显而易见地,下面描述中的附图仅仅是本发明的一些实施例,对于本领域的普通技术人员来讲,在不付出创造性劳动性的前提下,还可以根据这些附图获得其他的附图。

[0076] 图1为背景技术中的单位Trie树的示意图;

[0077] 图2为背景技术中的多位Trie树的示意图;

[0078] 图3为基于图2的多位Trie树的查找结构示意图;

[0079] 图4为本发明实施例中多位Trie结构创建过程的示意图;

[0080] 图5为本发明实施例中查找配置过程示意图;

[0081] 图6a、图6b为本发明实施例中增加表项时的配置流程示意图;

[0082] 图7为本发明实施例中删除表项时的配置流程示意图;

[0083] 图8a、图8b、图8c、图8d为本发明实施例中增量配置实例示意图;

[0084] 图9a、图9b为本发明实施例提供的查找流程示意图。

具体实施方式

[0085] 为了使本发明的目的、技术方案和优点更加清楚,下面将结合附图对本发明作进一步地详细描述,显然,所描述的实施例仅仅是本发明一部份实施例,而不是全部的实施

例。基于本发明中的实施例，本领域普通技术人员在没有做出创造性劳动前提下所获得的所有其它实施例，都属于本发明保护的范围。

[0086] 针对现有多位Trie算法中存在的流水线级数多等问题，本发明实施例提出了一种基于多位Trie树迭代的查找装置，查找配置方法以及查找方法，由于流水线级中仅保存前缀节点，可以减少对内存资源的占用，通过多位Trie树迭代可以减少流水线级数、降低对查找装置的带宽占用，减小查找延迟。

[0087] 本发明实施例提出的基于多位Trie树迭代的查找配置方法，在一次迭代过程中，针对一个流水线级配置前缀节点。具体当前迭代过程对应的多位Trie树是由前一次迭代过程创建的前缀节点的关联前缀构成，并且该多位Trie树中的叶子节点都是前缀节点，根据该多位Trie树可以创建当前迭代过程的前缀节点，将创建的前缀节点配置到当前迭代过程对应的流水线级中。最初一次迭代过程对应的多位Trie树是由待配置前缀，例如路由表中的IPv4/IPv6前缀构成，且每一次迭代过程创建的前缀节点对应一个流水线级，即，一个迭代过程对应的多位Trie树中的前缀节点保存在同一流水线级中。这样，流水线结构中可以仅保存前缀节点。

[0088] 为了更清楚地理解本发明，下面从以下几个方面对本发明进行详细说明：(一)查找装置、(二)查找配置流程、(三)查找流程、(四)其他可选方案。

[0089] (一)查找装置

[0090] 本发明实施例提供的查找装置可以是位于路由器或交换机等用于实现报文转发的网络设备中的装置，例如路由查找模块，处理器，或硬件实现的查找引擎；或者也可以是路由器或交换机等用于实现报文转发的网络设备。

[0091] 具体的，本发明实施例提供的查找装置中包括N个级流水线级，其中N＞1；

[0092] 每个流水线级中包括一个查找单元，按照所述N个流水线级执行的先后顺序，分别为第0级至第N-1级查找单元；每级查找单元中包括前缀节点，所述前缀节点用于存储前缀以及与所述前缀所对应的下一跳，其中：

[0093] 第i+1级查找单元中配置有第一前缀节点，第i级查找单元中配置有第二前缀节点，其中0＜＝i＜＝(N-2)；所述第二前缀节点中存储的前缀所对应的下一跳指向所述第一前缀节点，所述第二前缀节点通过对所述第一前缀节点的关联前缀所构成的第一多位Trie树进行子树划分得到。具体地，所述第二前缀节点通过对所述第一前缀节点的关联前缀所构成的第一多位Trie树进行子树划分得到可包括：将所述第一多位Trie树划分为M个子树，M＞＝1，每个子树包括所述第一多位Trie树的一个Trie节点以及所述Trie节点的K个分支，根据所述M个子树得到所述第二前缀节点，其中0＜＝K＜＝stride，所述stride为所述第一多位Trie树的步长。

[0094] 特别地，第N-1级查找单元中的前缀节点中的前缀所对应的下一跳指向下一跳的具体内容，比如所述下一跳的内容可以是下一跳IP地址。

[0095] 查找单元中配置的前缀节点实质上表现为一种数据结构，其中存储有前缀以及所述前缀对应的下一跳。

[0096] 每级查找单元中的前缀节点的数量可以是一个也可以是多个。

[0097] 每级查找单元对应一个多位Trie树。其中，第0级查找单元对应的多位Trie树是根据查找表中的前缀生成的，查找表中的前缀对应于该树中的相应节点。第1级查找单元对应

的多位Trie树是由第0级查找单元中的前缀节点的关联前缀构成的，即第0级查找单元中的前缀节点的关联前缀对应于该树中的相应节点；第2级查找单元对应的多位Trie树是由第1级查找单元中的前缀节点的关联前缀构成的。以此类推，除第0级查找单元以外的每相邻的两级查找单元中，前一级查找单元对应的多位Trie树由后一级查找单元中的前缀节点的关联前缀构成。

[0098] 根据一个多位Trie树划分得到的每个子树对应一个或多个前缀节点。比如，上述第二前缀节点中包括1至2^{stride}个间接前缀节点，或者包括0至2^{stride}个间接前缀节点和一个直接前缀节点，用于存储落入该第二前缀节点对应的子树内的前缀，其中，stride为多位Trie树的步长。此处，对于划分得到的每个子树，将该子树的顶点位置上的Trie节点称为该子树的Trie节点。在一个子树对应的前缀节点中，直接前缀节点用于存储落入该子树的Trie节点内的前缀，一个间接前缀节点用于存储落入该子树的Trie节点的一个分支内的前缀。

[0099] 每级查找单元中的每一个前缀节点对应于该查找单元所对应的多位Trie树的一个子树。如同传统的多位Trie算法中，每个Trie节点都有一个关联前缀一样，类似的，在本发明实施例中，每个前缀节点也有一个关联前缀。一个前缀节点的关联前缀为该前缀节点对应的子树的根节点对应的前缀。关联前缀也称为顶点前缀或子树根节点前缀。

[0100] 所述查找表可以是用于指导报文转发的查找表。根据不同的业务需求，可以配置相应类型的查找表。比如，对于三层转发业务，该查找表为IPv4/IPv6路由表，该路由表中包含IPv4/IPv6前缀及其对应的下一跳信息。对于不同的业务，使用的查找算法可能有所不同，比如对于路由查找，使用最长前缀匹配算法，因此可根据不同业务类型配置查找算法。

[0101] 可选地，所述第一前缀节点中存储第一数组或所述第一数组的地址，所述第一数组中存储所述第一前缀节点的每个前缀所对应的下一跳指针；和/或，所述第二前缀节点中存储第二数组或所述第二数组的地址，所述第二数组中存储所述第二前缀节点的每个前缀所对应的下一跳指针。

[0102] 也就是说，前缀节点可以存储一个数组，所述数组可以称为下一跳指针数组；前缀节点也可以存储该下一跳指针数组的地址。前缀节点存储的下一跳指针数组中保存了该前缀节点存储的每个前缀对应的下一跳指针。对于下一跳指针数组中的每个元素，其下标用于标识前缀，其元素值为该元素的下标所标识的前缀对应的下一跳指针。

[0103] 进一步地，前缀节点中还可以存储PLPM。如果一个前缀节点包括PLPM，则可以将该前缀节点的PLPM对应的下一跳指针存储到该前缀节点的下一跳指针数组中，比如可作为该数组的第一个元素。优选地，下一跳指针数组中存储的PLPM对应的下一跳指针，指向所述PLPM对应的下一跳信息。一个多位Trie树中，对于中间的子树，覆盖该子树的最长前缀即为该子树对应的前缀节点的PLPM。

[0104] 可选地，所述第i+1级查找单元中还配置有所述第一多位Trie树中包含有前缀节点的Trie节点。

[0105] 上述流水线结构中，级数较高的几级查找单元中配置的前缀节点数量一般较少，也就是说存储的前缀数量较少，因此可以使用TCAM来实现，一方面不会因前缀过多导致TCAM成本增加，另一方面还可利用TCAM高效查找的性能。用TCAM实现的查找单元的数量，可根据查找表的容量以及查找表中前缀的长度等因素来确定。优选地，第0级查找单元(即流

水线结构中最先执行的流水线级的查找单元)用TCAM实现。

[0106] 基于上述N个流水线级实现的查找配置过程以及查找过程,将在后续内容中详述。

[0107] 为了更清楚起见,下面以图4中(a)所示的多位Trie树为例,描述配置后的N个流水线级中的前缀节点的数据结构。

[0108] 图4中(a)所示的多位Trie树的结构与图2所示的多位Trie树结构相同。如果前缀节点的容量比较大,则T1以及T1的所有分支都可以保存在T1'对应的一组前缀节点中,因此创建如图4中(b)所示的Trie结构。

[0109] 如图4中(b)所示,该多位Trie树只有一个Trie节点T1',T1'对应的一组前缀节点中包括4个前缀节点P1～P4。P1、P2、P3和P4分别与图4中(a)中的T1、T2、T3和T4对应。P1中保存的是T1中的前缀;P2中保存的是从T2开始的分支上的所有前缀,也就是T2、T5和T6中的前缀;P3中保存的是T3中的前缀;P4中保存的是从T4开始的分支上的所有前缀,也就是T4和T7中的前缀。P1称为T1的直接前缀节点,P2、P3、P4称为T1的间接前缀节点。

[0110] 如果前缀节点的容量比较小,则创建如图4中(c)所示的Trie结构。

[0111] 如图4中(c)所示,该多位Trie树中有3个Trie节点,其中T2、T4是T1的Trie子节点。T1"对应的一组前缀节点中包括2个前缀节点P1和P3,T2"对应的一组前缀节点中包括3个前缀节点P2、P5和P6,T4"对应的一组前缀节点中包括P4和P7。Pi和Ti与图4中(a)中的Ti相对应。P1中保存的是T1中的前缀;P3中保存的是T3中的前缀;P2中保存的是T2中的前缀;P5中保存的是T5中的前缀;P6中保存的是T6中的前缀;P4中保存的是T4中的前缀;P7中保存的是T7中的前缀。P1称为T1的直接前缀节点,P3称为T1的间接前缀节点;P2称为T2的直接前缀节点,P5和P6称为T2的间接前缀节点;P4称为T4的直接前缀节点,P7称为T4的间接前缀节点。

[0112] 对于图4中(c)中的Trie结构,可将所有前缀节点保存在流水线结构中形成一级。所有前缀节点的关联前缀,包括:P1的关联前缀"*",P2的关联前缀"000*",P3的关联前缀"100*",P4的关联前缀"111*",P5的关联前缀"000010*",P6的关联前缀"000011*",P7的关联前缀"111010*",.组成了一个小的路由表,采用相同方式基于该小的路由表对应的多位Trie树创建前缀节点,最终可得到如图5所示的四级流水线结构。

[0113] 在图5所示的四级流水线结构中,第四级(stage3)的查找单元中配置有前缀节点PNODE30、PNODE31、PNODE32、PNODE33、PNODE34、PNODE35、PNODE36,其中,PNODE30～35分别对应于图4(c)中的Trie结构中的P1、P3、P2、P5、P6、P4、P7;第三级(stage2)的查找单元中配置有前缀节点PNODE20、PNODE21、PNODE22,其中,PNODE20～22分别对应于stage1的Trie结构中的T1、T2、T4;第二级(stage1)的查找单元中配置有前缀节点PNODE10,对应于stage1的Trie结构中的T1;第一级(stage0)的TCAM中配置PNODE10对应的下一跳指针。

[0114] 如果前缀节点中的下一跳指针数组表示为如下结构:

[0115] {PLPM对应的RE Index,第一个前缀节点对应的RE Index,第二个前缀节点对应的RE Index,……}

[0116] 则图5中各级查找单元中前缀节点的下一跳指针数组可表示为:

[0117] Stage3:

[0118] PNODE30:{p0对应的RE Index,p1对应的RE Index,p2对应的RE Index},其中:

[0119] p0对应的RE Index指向前缀p0对应的RE;

[0120] p1对应的RE Index指向前缀p1对应的RE;

[0121] p2对应的RE Index指向前缀p2对应的RE；

[0122] PNODE31：{PLPM对应的RE Index，p5对应的RE Index}，关联前缀为011*，PLPM为10*，其中：

[0123] PLPM对应的RE Index指向前缀10*对应的RE；

[0124] P5对应的RE Index指向前缀p5对应的RE；

[0125] PNODE32：{PLPM对应的RE Index，p4对应的RE Index}，关联前缀为000*，PLPM为0*，其中：

[0126] PLPM对应的RE Index指向前缀0*对应的RE；

[0127] P4对应的RE Index指向前缀p4对应的RE；

[0128] PNODE33：{PLPM对应的RE Index，p9对应的RE Index}，关联前缀为000010*，PLPM为0000*，其中：

[0129] PLPM对应的RE Index指向前缀0000*对应的RE；

[0130] P9对应的RE Index指向前缀p9对应的RE；

[0131] PNODE34：{PLPM对应的RE Index，p8对应的RE Index}，关联前缀为000011*，PLPM为0000*，其中：

[0132] PLPM对应的RE Index指向前缀0000*对应的RE；

[0133] p8对应的RE Index指向前缀p8对应的RE；

[0134] PNODE35：{PLPM对应的RE Index，p3对应的RE Index，p6对应的RE Index，p7对应的RE Index}，关联前缀为111*，PLPM为*，其中：

[0135] PLPM对应的RE Index指向前缀*对应的RE；

[0136] P3对应的RE Index指向前缀p3对应的RE；

[0137] P6对应的RE Index指向前缀p6对应的RE；

[0138] P7对应的RE Index指向前缀p7对应的RE；

[0139] PNODE36：{PLPM对应的RE Index，p10对应的RE Index}，关联前缀为111010*，PLPM为11101*，其中：

[0140] PLPM对应的RE Index指向前缀11101*对应的RE；

[0141] P10对应的RE Index指向前缀p10对应的RE。

[0142] Stage2：

[0143] PNODE20：{PNODE30的关联前缀对应的RE Index，PNODE31的关联前缀对应的RE Index}，关联前缀为*，其中：

[0144] PNODE30的关联前缀对应的RE Index指向PNODE30；

[0145] PNODE31的关联前缀对应的RE Index指向PNODE31；

[0146] PNODE21：{PLPM对应的RE Index，PNODE32的关联前缀对应的RE Index，PNODE33的关联前缀对应的RE Index，PNODE34的关联前缀对应的RE Index}，关联前缀为000*，PLPM为*，其中：

[0147] PLPM对应的RE Index指向前缀*对应的RE；

[0148] PNODE32的关联前缀对应的RE Index指向PNODE32；

[0149] PNODE33的关联前缀对应的RE Index指向PNODE33；

[0150] PNODE34的关联前缀对应的RE Index指向PNODE34；

[0151] PNODE22:{PLPM对应的RE Index,PNODE35的关联前缀对应的RE Index,PNODE36的关联前缀对应的RE Index},关联前缀为111*,PLPM为*,其中:

[0152] PLPM对应的RE Index指向前缀*对应的RE;

[0153] PNODE35的关联前缀对应的RE Index指向PNODE35;

[0154] PNODE36的关联前缀对应的RE Index指向PNODE36;

[0155] Stage1:

[0156] PNODE10:{PNODE20的关联前缀对应的RE Index,PNODE21的关联前缀对应的RE Index,PNODE22的关联前缀对应的RE Index},关联前缀为*,其中:

[0157] PNODE20的关联前缀对应的RE Index指向PNODE20;

[0158] PNODE21的关联前缀对应的RE Index指向PNODE21;

[0159] PNODE22的关联前缀对应的RE Index指向PNODE22;

[0160] Stage0:

[0161] PNODE00:{PNODE10的关联前缀对应的RE Index},PNODE10的关联前缀对应的RE Index指向PNODE10。

[0162] 通过以上描述可以看出,本发明的上述实施例中,查找装置的N个流水线级中,每级查找单元中包括前缀节点,用于存储前缀以及与所述前缀对应的下一跳,其中:第i+1级查找单元中配置有第一前缀节点,第i级查找单元中配置有第二前缀节点,所述第二前缀节点中存储的前缀对应的下一跳指向所述第一前缀节点,所述第二前缀节点通过对所述第一前缀节点的关联前缀所构成的第一多位Trie树进行子树划分得到。划分得到的一个子树的覆盖范围包括多位Trie树中的一个Trie节点以及所述Trie节点的K个分支,划分得到的一个子树对应一组前缀节点,因此,一方面由于仅将前缀节点配置在查找单元中,与现有技术相比减少了内存资源的占用;另一方面,由于前一级的查找单元中的前缀节点是基于后一级的查找单元中的前缀节点的关联前缀构成的多位Trie树进行子树划分得到的,使得一个前缀节点所对应的子树的覆盖范围较大,减少了流水线级数,进而减少查找延迟并降低实现的难度。

[0163] 需要说明的是,本发明实施例提供的查找装置中,所述N个流水线级可由硬件实现,所述硬件可以是现场可编程门阵列(Programmable Gate Array,简称:FPGA),专用集成电路(Application Specific Integrated Circuit,简称ASIC);所述N个流水线级也可由软件实现,例如处理器中多个线程实现所述N个流水线级。

[0164] (二)查找配置过程

[0165] 本发明实施例中的查找配置过程,是指查找表的配置过程,即,将查找表的表项配置到N个流水线级中的过程,查找表的每个表项中包括前缀以及所述前缀对应的下一跳。

[0166] 查找配置过程可包括初始配置过程和增量配置过程。在初始配置过程中,可以根据已有的整个查找表配置所述N个流水线级,在运行过程中,可根据接收到的路由更新命令获取新增的待配置前缀和该前缀对应的下一跳,增量配置所述N个流水线级。

[0167] 下面分别对初始配置过程和增量配置过程进行详细描述。

[0168] (1)初始配置过程

[0169] 首先定义前缀节点的容量。前缀节点的容量是指前缀节点的数据结构的大小,其单位为比特。前缀节点的容量与如下任一或多个因素相关:查找装置的最大带宽,查找表中

前缀的长度等。查找装置的最大带宽表示在一个时钟周期内能够并行处理的最大比特数。前缀节点可存储的前缀的个数与前缀节点的容量和前缀长度相关。优选地，为了减少流水线级数，可将前缀节点的容量设置得较大，比如对于路由查找可设置为256比特。

[0170] 优选地，为了进一步使一个前缀节点能够存储更多的前缀以减少流水线级数，可以对前缀节点进行编码以减少存储开销。

[0171] 在初始配置过程中，可根据查找表创建多位Trie树，从第N-1级查找单元至第0级查找单元，通过N次迭代过程配置查找单元，配置后流水线结构如前所述。具体配置过程如下所述。

[0172] 在第一次迭代过程中，首先，根据查找表以及预定的stride，创建多位Trie树，并根据前缀节点的容量对该多位Trie树进行子树划分。前缀节点的容量越大，子树的覆盖范围也就越大。在根据前缀节点的容量对多位Trie树进行划分时，尽可能使每个子树覆盖较多的前缀，只有当超过前缀节点的容量时，才将该子树拆分。

[0173] 根据划分得到的子树创建对应的前缀节点中包括一个或多个前缀节点。对于每个子树来说，落入该子树内的前缀被存储到该子树对应的前缀节点所容纳的前缀中。

[0174] 最后，将通过上述过程得到的前缀节点配置在第N-1级查找单元中。进一步地，可将该多位Trie树的结构保存到查找装置内置的或外接的存储器中，并标识为第N-1级查找单元对应的多位Trie树。

[0175] 第N-1级查找单元对应的多位Trie树中，划分得到的每个子树对应一个Trie节点，该Trie节点有两组子节点，第一组用于保存该Trie节点的子Trie节点；第二组用于保存该Trie节点对应的前缀节点，其中可包括间接前缀节点，还可进一步包括直接前缀节点。在子树中未包括Trie节点的任何分支时，该Trie节点的第一组子节点为空，第二组子节点中的间接前缀节点为空。由于一个Trie节点最多有2^{stride}个Trie子节点，因此一个子树对应的Trie节点所对应的前缀节点最多为$2^{stride}+1$个。

[0176] 在第二次迭代过程中，首先，根据第一次迭代过程得到的前缀节点的关联前缀创建多位Trie树，并根据前缀节点的容量对该多位Trie树进行子树划分。根据前缀节点的容量以及第一次迭代过程创建的前缀节点的关联前缀在多位Trie树上的分布，划分得到的一个子树的覆盖范围可能包括一个Trie节点或者一个Trie节点及其一个或多个分支。在子树划分时，尽可能使每个前缀节点存储较多的前缀。对于划分得到的每个子树来说，落入该子树内的前缀是前一次迭代过程得到的前缀节点的关联前缀。然后，根据划分得到的子树创建对应的一组前缀节点，每个子树对应的一组前缀节点中包括最多2^{stride}个间接前缀节点，还可包含一个直接前缀节点。最后，将通过上述过程得到的前缀节点配置在第N-2级查找单元中，并可进一步将该多位Trie树的结构保存到查找装置内置的或外接的存储器中，并标识为第N-2级查找单元对应的多位Trie树。

[0177] 后续每次的迭代过程，基本与第二次迭代过程类似。每次迭代过程完成后，将该次迭代过程得到的前缀节点配置在相应一级的查找单元中，并可进一步在查找装置的内置或外接存储器中保存该级查找单元对应的多位Trie树，直到第0级查找单元配置完成。

[0178] 如前所述，一个前缀节点可通过数组或其他数据结构来保存该前缀节点中存储的前缀所对应的下一跳。比如，前缀节点中可存储一数组，该数组称为下一跳指针数组，其中每个元素的下标用于标识前缀，每个元素的取值为与该元素的下标所标识的前缀对应的下

一跳指针(即RE Index)。采用这种数据结构,可以减少流水线级数,降低硬件实现的难度。

[0179] 为了进一步减少查找装置的内存开销,前缀节点中可以仅存储该数组的地址(即起始地址),或指向该数组的指针,该数组的内容可存储在查找装置的内置或外接存储器中。

[0180] 优选地,对于每级查找单元中配置的每个前缀节点,如果该前缀节点存在对应的PLPM,则还要在该前缀节点中保存该PLPM。优选地,对于一个具有PLPM的前缀节点,可将该PLPM对应的下一跳指针(RE Index)作为该前缀节点对应的下一跳指针数组中的第一个元素进行保存。

[0181] 如前所述,对于第N-1级查找单元,每个前缀节点对应的下一跳指针数组中,一个前缀对应的下一跳指针指向该前缀对应的下一跳的具体内容,该下一跳的具体内可存储在查找装置的内置或外接存储器中。对于除第N-1级以外的其他级查找单元,比如以第i个查找单元为例,该查找单元中的每个前缀节点对应的下一跳数组指针中,一个前缀对应的下一跳指针指向第i+1级查找单元中保存该前缀的前缀节点。

[0182] 以上仅为前缀节点的数据结构的优选实现方式,本发明对前缀节点的数据结构不作限制。

[0183] 当然,初始配置过程也可采用增量配置过程的方式来实现,比如,首先从查找表中取出一个表项,采用增量配置过程对各级查找单元进行配置,然后再取出另一个表项,采用增量配置过程对各级查找单元进行配置,以此类推,直到取出查找表中的最后一个表项并采用增量配置过程完成对各级查找单元的配置,至此,完成整个查找表的配置过程。增量配置过程具体如下所述。

[0184] (2)增量配置过程

[0185] 查找表通常会定时或不定时进行更新,比如增加查找表的表项或删除查找表的表项。

[0186] 图6a示出了增加查找表项时的增量配置过程,可包括以下步骤:

[0187] S61:获得待配置的第一前缀以及所述第一前缀所对应的下一跳,转入S62。

[0188] 通常,用于指导报文转发的查找表允许人工配置或更新,或者可根据相关协议进行更新。比如对于路由查找,可通过人工方式配置静态路由或者根据边界网关协议(Border Gateway Protocol,简称:BGP)等路由选择协议动态更新路由表,包括增加新的路由表项或者删除已有的路由表项。本实施例中,当查找装置接收到查找表更新命令后,能够根据该命令确定出需要增加或删除的查找表的表项,以便对流水线级进行配置更新。

[0189] S62:将所述第一前缀以及所述第一前缀所对应的下一跳存储在第N-1级查找单元中,转入S63。其中,N为流水线级的数量,N>1,每个流水线级中包括一个查找单元,按照所述N个流水线级执行的先后顺序,分别为第0级至第N-1级查找单元;每级查找单元中包括前缀节点,用于存储前缀以及与所述前缀所对应的下一跳。

[0190] S63:对第N-1级查找单元中的前缀节点的关联前缀所构成的多位Trie树进行子树划分,以得到第N-2级查找单元的前缀节点,其中,第N-2级查找单元的前缀节点中存储的前缀所对应的下一跳指向第N-1级的前缀节点。

[0191] 其中,所述对第N-1级查找单元中的前缀节点的关联前缀所构成的多位Trie树进行子树划分,以得到第N-2级查找单元的前缀节点,包括:将所述多位Trie树划分为M个子

树，M＞＝1，每个子树包括所述多位Trie树的一个Trie节点以及所述Trie节点的K个分支，其中0＜＝K＜＝stride，所述stride为所述多位Trie树的步长。

[0192] 上述配置过程中，若新增加的查找表项可容纳于第N-1级查找单元中配置的前缀节点中，则可以仅对第N-1级查找单元中的前缀节点进行更新，其他级的查找单元中配置的前缀节点无需更新。若新增加的查找表项无法容纳于第N-1级查找单元中配置的前缀节点中，则需要通过图6b所示的以下过程调整Trie树结构，新增加前缀节点，以容纳新增加的查找表项：

[0193] 在S62中，判断第N-1级查找单元中的前缀节点的容量是否允许存储所述第一前缀以及所述第一前缀所对应的下一跳，若是，则将所述第一前缀以及所述第一前缀所对应的下一跳存储于所述第N-1级查找单元的前缀节点中；否则，重新配置第N-1级查找单元的前缀节点，将所述第一前缀以及所述第一前缀所对应的下一跳存储于所述第N-1级查找单元重新配置后的前缀节点；

[0194] 在S63中，根据第N-2级查找单元中的前缀节点的容量判断是否允许存储第N-1级查找单元重新配置的前缀节点的关联前缀，若是，则将第N-1级查找单元重新配置的前缀节点的关联前缀存储于第N-2级查找单元的前缀节点；

[0195] 若所述S63中的判断结果为否，则转入S64；

[0196] S64：设置j＝N-2，转入S65；

[0197] S65：通过对第j+1级查找单元中的前缀节点的关联前缀所构成的多位Trie树进行子树划分来重新配置第j级查找单元的前缀节点，将第j+1级查找单元中配置的前缀节点的关联前缀存储于第j级查找单元重新配置后的前缀节点，转入S66；

[0198] S66：根据第j-1级查找单元的前缀节点的容量判断是否允许存储第j级查找单元重新配置的前缀节点的关联前缀，若是，则将第j级查找单元重新配置的前缀节点的关联前缀存储于第j-i级查找单元的前缀节点，并结束流程，否则转入S67；

[0199] S67：设置j＝j-1，并转入S68；

[0200] S68：若j＝0则结束流程，否则，转入S65。

[0201] 优选地，上述流程中，所述重新配置第N-1级查找单元的前缀节点，包括：

[0202] 将所述第一前缀插入第N-1级查找单元对应的第一多位Trie树；

[0203] 若第N-1级查找单元中存在拆分目标前缀节点，则根据所述第一前缀在第N-1级查找单元对应的第一多位Trie树中的位置、所述第一多位Trie树的步长以及所述拆分目标前缀节点所存储的前缀在所述第一多位Trie树中的位置，将所述拆分目标前缀节点拆分为多个前缀节点，拆分得到的一个前缀节点中存储有所述第一前缀以及所述第一前缀所对应的下一跳，否则，在第N-1级查找单元中新配置一个前缀节点以存储所述第一前缀以及所述第一前缀所对应的下一跳；其中，所述目标前缀节点满足第一条件且是所有满足所述第一条件的前缀节点中距离所述第一前缀的步长最少的前缀节点，所述第一条件为：前缀节点所对应的子树的分支上包含所述第一前缀。

[0204] 优选地，上述流程中，所述通过对第j+1级查找单元中的前缀节点的关联前缀所构成的多位Trie树进行子树划分来重新配置第j级查找单元的前缀节点，包括：

[0205] 若第j级查找单元中存在拆分目标前缀节点，则根据所述第一前缀在第j级查找单元所对应的第二多位Trie树中的位置、所述第二多位Trie树的步长以及所述拆分目标前缀

节点所存储的前缀在所述第二多位Trie树中的位置，将所述拆分目标前缀节点拆分为多个前缀节点，拆分得到的一个前缀节点中存储有第j+1级查找单元中新配置的前缀节点的关联前缀以及对应的下一跳，否则，在第j级查找单元中新配置一个前缀节点以存储所述第j+1级查找单元中新配置的前缀节点的关联前缀以及对应的下一跳；其中，所述目标前缀节点满足第一条件且是所有满足所述第一条件的前缀节点中距离所述第j+1级查找单元中新配置的前缀节点的关联前缀的步长最少的前缀节点，所述第一条件为：前缀节点所对应的子树的分支上包含所述第j+1级查找单元中新配置的前缀节点的关联前缀。

[0206] 类似的，当需要删除查找表项时，其流程可如图7所示：

[0207] S71：获得待删除的第二前缀以及所述第二前缀对应的下一跳；

[0208] S72：从第N-1级查找单元中存储有所述第二前缀以及所述第二前缀对应的下一跳的前缀节点中释放所述第二前缀以及所述第二前缀的下一跳；

[0209] S73：若释放所述第二前缀后的前缀节点中，已不再存储有前缀以及所述前缀的下一跳，则释放该前缀节点，并转入S74；

[0210] S74：设置j＝N-2，转入S75；

[0211] S75：针对第j+1级查找单元中被释放的前缀节点，从第j级查找单元中的前缀节点中释放相应前缀节点的关联前缀，若释放关联前缀后的前缀节点中已不再存储有关联前缀，则释放第j级查找单元中已不再存储有关联前缀的前缀节点，并转入S76；

[0212] S76：设置j＝j-1；转入S77；

[0213] S77：若j＜0，则结束流程，否则转入S75。

[0214] 通过以上描述可以看出，本发明实施例能够以最小代价对原有查找结构进行更新，以尽量减少避免对报文转发造成的影响，从而保证报文转发性能。与传统的多位Trie算法相比，本发明实施例提供的方案的增量更新的时间复杂度为O(W/k)，其中W表示前缀的长度，k表示Trie stride。当插入或删除一个前缀时，只有该前缀所在的分支会受影响，其它分支都不变。

[0215] 图6a或图6B或图7仅示出了查找表增量配置的优选实现方式，本发明并不仅限于图中示出的增量配置方式。基于前述已经介绍过的查找表流水线结构，只要能够使配置完成的流水线符合上述结构，本发明并不对具体实现方式做任何限制，比如在路由表更新时，可根据更新后的路由表更新第N级查找单元对应的多位Trie树，并基于此配置第N级查找单元，然后逐级向上配置各级查找单元。

[0216] 为了更清楚地对增量配置过程进行说明，下面以一个简单的例子介绍查找配置的过程。

[0217] 在下面的例子中，依次向一个空路由表中增加6个前缀，将这6个前缀依次插入到路由表中，创建查找结构。这6个前缀为：p0(*)、p1(0*)、p2(10*)、p3(111*)、p4(0000*)、p5(1001*)。

[0218] 查找装置的流水线级中包括三级，第一级(stage0)为TCAM。

[0219] 完整的多位Trie结构被存储于存储器中，当接收到新的前缀时更新该多位Trie结构，并生成更新指令发送给查找装置，所述更新指令可以是写入一个前缀节点的指令、删除一个前缀节点的指令，或者刷新一个前缀节点的指令。

[0220] 如图8a所示，当插入前缀p0时，先在stage2对应的多位Trie树中添加前缀p0，并根

据该多位Trie树创建一个前缀节点PNODE20，前缀p0和它对应的RE Index都保存在PNODE20中；然后将PNODE20封装之后写入stage2对应的查找单元。接下来，将stage2对应的多位Trie树中的前缀节点的关联前缀p10插入stage1对应的多位Trie树，并创建一个前缀节点PNODE10，并在封装之后将PNODE10写入stage1对应的查找单元。Stage1对应的多位Trie树中每个前缀对应stage2的多位Trie树中的一个前缀节点。将stage1对应的多位Trie树中的前缀节点的前缀放在stage0中的TCAM中。这样从stage0到stage2的查找结构就被建立起来了。

[0221] 如图8b所示，当插入前缀p1和p2时，由于前缀p1、p2和p0都在stage2对应的多位Trie树中的一个Trie节点内(即在一个stride内)，将p1和p2添加到PNODE20后不会超出该前缀节点的容量，因此只需要更新PNODE20即可，即，将p1和p2对应的下一跳指针存到PNODE20中。TCAM和stage1的查找单元中的前缀节点都不用更新。

[0222] 如图8c所示，当插入前缀p3和p4时，虽然它们与前缀p0、p1和p2不在stage2对应的多位Trie树中的一个Trie节点内(即不在一个stride内)，但是将前缀p3和p4添加到PNODE20后不会超出该前缀节点的容量，因此将p1和p2对应的下一跳指针存到PNODE20中。

[0223] 如图8d所示，当插入前缀p5时，如果将前缀p5的下一跳指针添加到PNODE20后将会超出该前缀节点的容量，因此此时需要对stage2对应的多位Trie树中的前缀节点进行拆分。

[0224] 拆分时，首先，根据前缀p5在该多位Trie树中的位置以及该多位Trie树中原有的前缀节点PNODE20所容纳的前缀在该多位Trie树中的位置，确定需被拆分的目标前缀节点，此处由于该多位Trie树只有一个前缀节点PNODE20，因此将其作为需要被拆分的前缀节点。接着，根据PNODE20中容纳的前缀以及前缀p5在多位Trie树中的位置，按照步长(stride)将该前缀节点拆分成多个小的前缀节点。从图中可以看到，6个前缀按步长划分之后，可以分成4部分，每个部分可以作为一个新的前缀节点，如图中所示的PNODE21～24。为了尽可能的利用访存带宽，一般采用最小切分策略，也就是切分后的子节点所覆盖的范围尽可能大。可以使用与前缀节点关联的多位Trie树的根节点进行切分。在本例中，从根节点切分可以把原来的PNODE20切分成4个小的前缀节点PNODE21～24，并且创建一个Trie节点T1作为这4个前缀节点的父节点。

[0225] 拆分完成之后，产生了新的前缀节点，将新的前缀节点PNODE21～24配置到查找装置中的stage2对应的查找单元，并将它们的关联前缀插入到stage1对应的多位Trie树中。由于PNODE21的关联前缀与原来的PNODE20的关联前缀相同，而该前缀已经在stage1对应的多位Trie树中了，所以不需要重复插入该前缀节点对应的关联前缀，只需要更新PNODE21即可。在更新过程中，先将新的前缀节点PNODE22、PNODE23和PNODE24配置到stage2对应的查找单元中，然后再将原来的PNODE20更新为PNODE21，以避免对查找过程造成影响。

[0226] 如果PNODE10所容纳的前缀节点的下一跳指针连续存放，则可只在PNODE10中保存下一跳指针数组的基址，当需要对PNODE10进行更新时，只需要先将新的数组元素写入，然后更新PNODE10的下一跳指针数组的基址，最后再删除旧的节点就可以了。

[0227] 在stage1对应的多位Trie树中，当插入stage2的前缀节点的关联前缀后，如果没有超出stage1的前缀节点的容量，则只需要在stage2对应的多位Trie树中更新PNODE10。

[0228] 如前所述，本发明实施例中的查找装置可以是位于路由器或交换机等用于实现报

文转发的网络设备中的装置，例如路由查找模块，处理器，或硬件实现的查找引擎等。在这种情况下，所述查找配置流程由所述路由器或交换机等用于实现报文转发的网络设备中的控制平面执行，比如，控制平面通过执行查找配置流程，将路由表中的前缀以及所述前缀所对应的下一跳配置到该查找装置。本发明实施例中的查找装置也可以是路由器或交换机等用于实现报文转发的网络设备，这种情况下，所述查找配置流程由该网络设备中的控制平面执行，即，控制平面通过执行查找配置流程，将路由表中的前缀以及所述前缀所对应的下一跳配置到该网络设备中的数据平面。

[0229] (三)查找流程

[0230] 本发明实施例提供的查找流程是基于前述查找装置实现的。

[0231] 参见图9a，为本发明实施例提供的查找流程示意图，该流程可包括：

[0232] S91：查找装置接收查找关键字，转入S92；其中，所述查找装置包括N个流水线级，N＞1；每个流水线级中包括一个查找单元，按照所述N个流水线级执行的先后顺序，分别为第0级至第N-1级查找单元；

[0233] S92：设置i＝0，转入S93；

[0234] S93：第i级查找单元用所述查找关键字与第i级查找单元中的第一前缀节点中存储的前缀进行匹配，若匹配到前缀，则将匹配到的前缀的长度以及匹配到的前缀所对应的下一跳输出给第i+1级查找单元，转入S94，否则，转入S96；

[0235] S94：设置i＝i+1，并转入S95；

[0236] S95：第i级查找单元根据第i-1级查找单元输出的前缀的长度，从所述查找关键字中提取相应长度的分段关键字，根据第i-1级查找单元输出的下一跳确定所述第i级查找单元中对应的第二前缀节点，用提取出的分段关键字与所述第二前缀节点所存储的前缀的相应分段关键字进行匹配；

[0237] 若匹配到前缀，当i＝N-1时，输出匹配到的前缀所对应的下一跳；否则，将匹配到的前缀的长度以及匹配到的前缀所对应的下一跳输出给第i+1级查找单元，转入S94；

[0238] 若没有匹配到前缀，且所述第二前缀节点有PLPM，则输出所述PLPM所对应的下一跳；如果所述第二前缀节点没有PLPM，则转入S96；

[0239] S96：输出未查找到匹配前缀的提示信息。

[0240] 图9b中以三层流水线结构为例描述了沿流水线执行顺序的查找过程，该流程可包括：

[0241] S1021：在流水线stage0对应的查找单元中，用查找关键字与本级查找单元中配置的前缀节点所容纳的前缀进行匹配。若匹配到前缀，则将匹配到的前缀的长度以及匹配到的前缀对应的下一跳指针输出给stage1对应的查找单元，该下一跳指针指向stage1对应的查找单元中配置的前缀节点，然后转入S1022。

[0242] 若S1021中，若未匹配到前缀，则表明查找失败，转入S1026。

[0243] S1022：在流水线stage1对应的查找单元中，根据stage0输出的前缀的长度，从查找关键字中提取相应长度的分段key，用提取出的分段key与stage0输出的下一跳指针所指示的前缀节点所容纳的前缀进行匹配，其中，所述“相应长度的分段key”是指从stage0输出的前缀长度开始的所有剩余key。若匹配到前缀，则输出该前缀的长度以及该前缀对应的下一跳指针给stage2，该下一跳指针指向stage2对应的查找单元中配置的前缀节点，然后转

入S1023。若若未匹配到前缀，则转入S1024。

[0244] S1023：在流水线stage2对应的查找单元中，根据stage1输出的前缀的长度，从查找关键字中提取相应长度的分段key，用提取出的分段key与stage1输出的下一跳指针所指示的前缀节点所容纳的前缀进行匹配，其中，所述“相应长度的分段key”是指从stage1输出的前缀长度开始的所有剩余key。若匹配到前缀，则输出该前缀对应的下一跳指针，该下一跳指针指向下一跳信息，然后转入S1025。若未匹配到前缀，则转入S1024。

[0245] S1024：根据前一级查找单元输出的下一跳指针，获取该指针所指示的前缀节点对应的PLPM，获取该PLPM对应的下一跳指针，该下一跳指针指向下一跳信息，然后转入S1025。若前一级查找单元输出的下一跳指针所指示的前缀节点没有PLPM，则转入S1026。

[0246] S1025：根据输出的下一跳指针获取下一跳信息，将该下一跳信息作为该查询关键字的查询结果并返回，结束查找流程。

[0247] S1026：输出查找MISS的信息(即未找到匹配前缀的信息)，结束查找流程。

[0248] 上述流程的S1024中，由于前缀节点对应的PLPM有多种不同的存储方式，因此该根据前缀节点对应的PLPM得到下一跳指针的方式也有所不同。如果PLPM以及该PLPM对应的下一跳指针保存到前缀节点对应的下一跳指针数组中，则可以从该前缀节点对应的下一跳指针数组中获取该PLPM对应的下一跳指针。前缀节点的PLPM也可以是以指针形式存储的，该指针包括容纳该PLPM的前缀节点的指针以及该PLPM在该前缀节点的下一跳指针数组中的位置。

[0249] 上述流程的S1025中，可从前缀节点的下一跳指针数组中获取该前缀节点所容纳的前缀对应的下一跳信息。具体地，在查找单元中存储有前缀节点对应的下一跳指针数组的情况下，可直接从前缀节点对应的下一跳指针数组中获取下一跳指针，并根据该指针获取对应的下一跳信息；在查找单元中未存储下一跳指针数组而是存储下一跳指针数组的基址的情况下，可根据该下一跳指针数组的基址访问对应的下一跳指针数组，从该数组中获取对应的下一跳指针，并根据该指针获取对应的下一跳信息。

[0250] 如前所述，流水线级中，还可包含Trie节点，或者可以是如前所述的其他改进结构，但无论哪种流水线结构，其查找过程基本与上述查找过程类似，在此不再详述。

[0251] 以上流程仅以三层流水线为例描述，其他数量的流水线级数，查找过程与此类似。比如，如果流水线级中除最上一级和最下一级以外，有多个中间层级的stage，则对于中间层级的每个stage，其处理流程与上述流程中的S1022类似。

[0252] (四)其他可选方案

[0253] 在以上描述的实施例中，在配置查找装置的流水线结构时，只将各流水线级对应的前缀节点配置在流水线结构中。在另一种优选方案中，还可以在此基础上将所有带有前缀节点的Trie节点配置到流水线结构中，比如，将第i级查找单元对应的多位Trie树中带有前缀节点的Trie节点以及该Trie节点所对应的前缀节点都配置到第i级查找单元中。

[0254] 在配置流水线结构时，对于每级查找单元对应的多位Trie树，将该多位Trie树中所有带有前缀节点的Trie节点放在一级，所有的前缀节点放在Trie节点的下一级。也就是说，对于每一个多位Trie树，在查找装置的流水线结构中产生两级，其中一级用于放置Trie节点，另一级用于放置前缀节点。如前所述，每个Trie节点可以有2组子节点，一组用于保存Trie子节点，一组用于保存前缀节点。每组子节点单独索引，但在查找装置的流水线级中可

以只保存对前缀节点的索引。

[0255] 具体实施时，每个查找单元中包含两个子单元，每个子单元对应流水线结构中的一级；按照流水线自前向后的顺序，一个查找单元中的第一子单元配置有该查找单元对应的多位Trie树中的所有带有前缀节点的Trie节点，第二子单元配置有该查找单元对应的多位Trie树中的所有前缀节点。

[0256] 与只在流水线级中保存前缀节点的技术方案相比，这里描述的可选方案可以使得每次迭代产生的关联前缀更少，因为一般情况下，一个Trie节点都有多个前缀节点，所以Trie节点的关联前缀数量要小于前缀节点的关联前缀数量，从而有利于提高更新速度。

[0257] 进一步地，还可以将两种配置方案结合使用，比如对于某些查找单元，只将多位Trie树中的前缀节点配置在这些查找单元中，对于另外的一些查找单元，将所有带有前缀节点的Trie节点以及前缀节点本身都配置到这些查找单元中。

[0258] 如前所述，本发明实施例提供的方案除了可以用在路由查找中，还可以用在精确查找等查找算法中，比如用于处理媒体接入控制(media access control，简称:MAC)查找，或者线性表查找。对于精确查找，只需要将查找表中的每个表项看做是满长前缀即可，在流水线级的结构方面，以及配置过程和查找过程等方面，基本与前述的实施例相似，在此不再详述。

[0259] 综上所述，本发明实施例针对现有多位Trie算法中存在的流水线级数多、各级内存占用不确定以及数据结构比较复杂等问题，提出了一种通过多Trie迭代来配置查找装置的方案。在创建多位Trie树的过程中，一方面通过划分子树使每个子树尽可能多的覆盖前缀并为每个子树创建对应的前缀节点以容纳该子树所覆盖的前缀，从而使多位Trie树结构“扁平化”，然后通过多位Trie迭代，用下一级前缀节点的关联前缀创建新的多位Trie树以确定上一级的前缀节点，从而与现有多位Trie算法相比，减少了流水线的级数，并较大降低了访存带宽和查找延迟。

[0260] 另外，所有的Trie节点或者不带有前缀节点的Trie节点不需要保存在查找装置中，可以减少内存资源的占用。流水线各级的内存占用完全独立，按照流水线执行顺序各级内存占用量逐渐减少，有明显的规律性，非常利于软件/硬件实现。大部分的内存占用都在流水线最后一级，对于带有片外内存的硬件架构(即查找装置外接有存储器的架构)，可以大大减少片上资源(即查找装置内置存储器资源)的占用。片外内存一般都是价格低廉的动态随机存取存储器(Dynamic Random Access Memory，简称:DRAM)，所以本发明实施例提供的方案还可以降低硬件实现的成本。

[0261] 再一方面，本发明实施例对于IPv4和IPv6路由的处理完全相同，而且通过将软硬件的实现分离，数据结构也大大简化，降低了实现的复杂度和难度。本发明实施例提供的方案对前缀的长度不敏感，可扩展性好，可以支持较大规模的路由表。

[0262] 本领域内的技术人员应明白，本发明的实施例可提供为方法、系统、或计算机程序产品。因此，本发明可采用完全硬件实施例、完全软件实施例、或结合软件和硬件方面的实施例的形式。而且，本发明可采用在一个或多个其中包含有计算机可用程序代码的计算机可用存储介质(包括但不限于磁盘存储器、CD-ROM、光学存储器等)上实施的计算机程序产品的形式。

[0263] 本发明是参照根据本发明实施例的方法、设备(系统)、和计算机程序产品的流程

图和/或方框图来描述的。应理解可由计算机程序指令实现流程图和/或方框图中的每一流程和/或方框、以及流程图和/或方框图中的流程和/或方框的结合。可提供这些计算机程序指令到通用计算机、专用计算机、嵌入式处理机或其他可编程数据处理设备的处理器以产生一个机器，使得通过计算机或其他可编程数据处理设备的处理器执行的指令产生用于实现在流程图一个流程或多个流程和/或方框图一个方框或多个方框中指定的功能的装置。

[0264] 这些计算机程序指令也可存储在能引导计算机或其他可编程数据处理设备以特定方式工作的计算机可读存储器中，使得存储在该计算机可读存储器中的指令产生包括指令装置的制造品，该指令装置实现在流程图一个流程或多个流程和/或方框图一个方框或多个方框中指定的功能。

[0265] 这些计算机程序指令也可装载到计算机或其他可编程数据处理设备上，使得在计算机或其他可编程设备上执行一系列操作步骤以产生计算机实现的处理，从而在计算机或其他可编程设备上执行的指令提供用于实现在流程图一个流程或多个流程和/或方框图一个方框或多个方框中指定的功能的步骤。

[0266] 尽管已描述了本发明的优选实施例，但本领域内的技术人员一旦得知了基本创造性概念，则可对这些实施例作出另外的变更和修改。所以，所附权利要求意欲解释为包括优选实施例以及落入本发明范围的所有变更和修改。

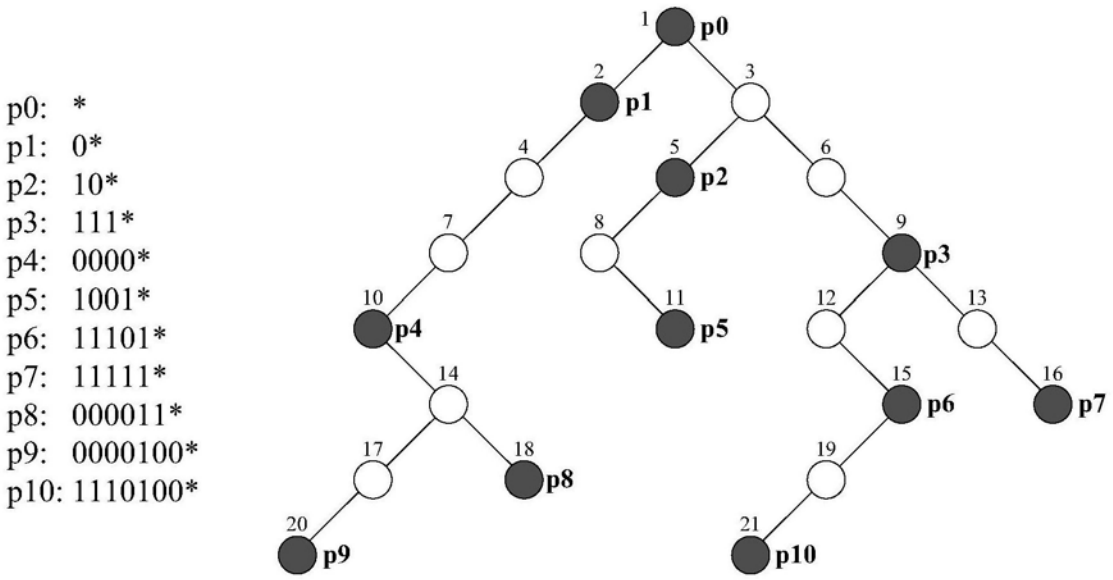

p0: *
p1: 0*
p2: 10*
p3: 111*
p4: 0000*
p5: 1001*
p6: 11101*
p7: 11111*
p8: 000011*
p9: 0000100*
p10: 1110100*
p0
p1
p2
p3
p4
p5
p6
p7
p8
p9
p10

图1

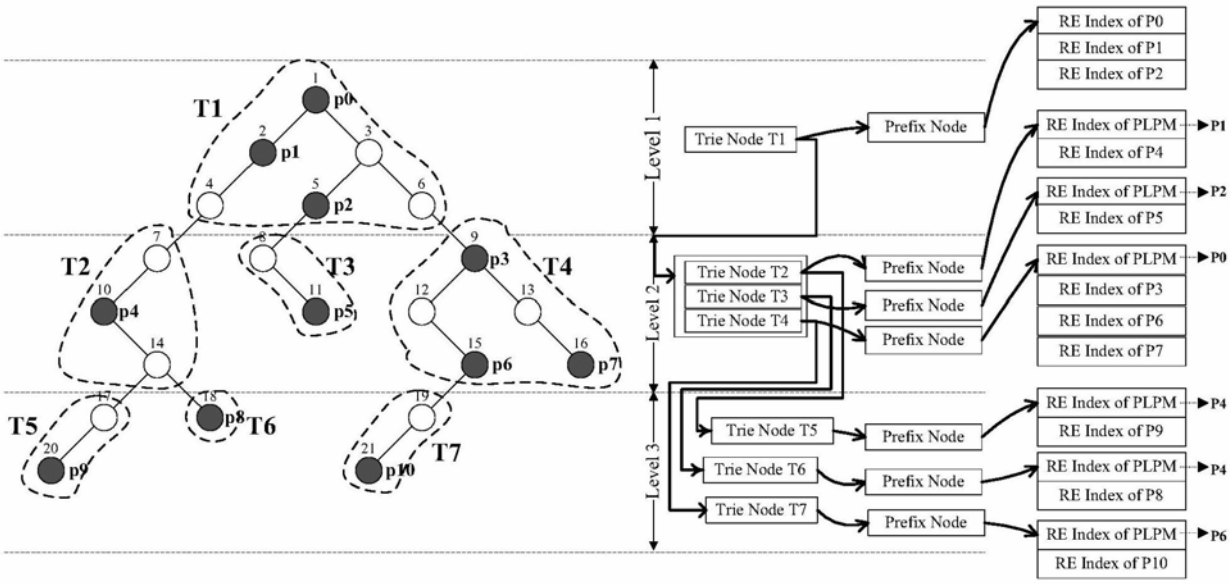

T1
T2
T3
T4
T5
T6
T7
Level 1
Level 2
Level 3
Trie Node T1
Trie Node T2
Trie Node T3
Trie Node T4
Trie Node T5
Trie Node T6
Trie Node T7
Prefix Node
RE Index of P0
RE Index of P1
RE Index of P2
RE Index of PLPM
RE Index of P4
RE Index of P5
RE Index of P3
RE Index of P6
RE Index of P7
RE Index of P9
RE Index of P8
RE Index of P10
P0
P1
P2
P4
P6

图2

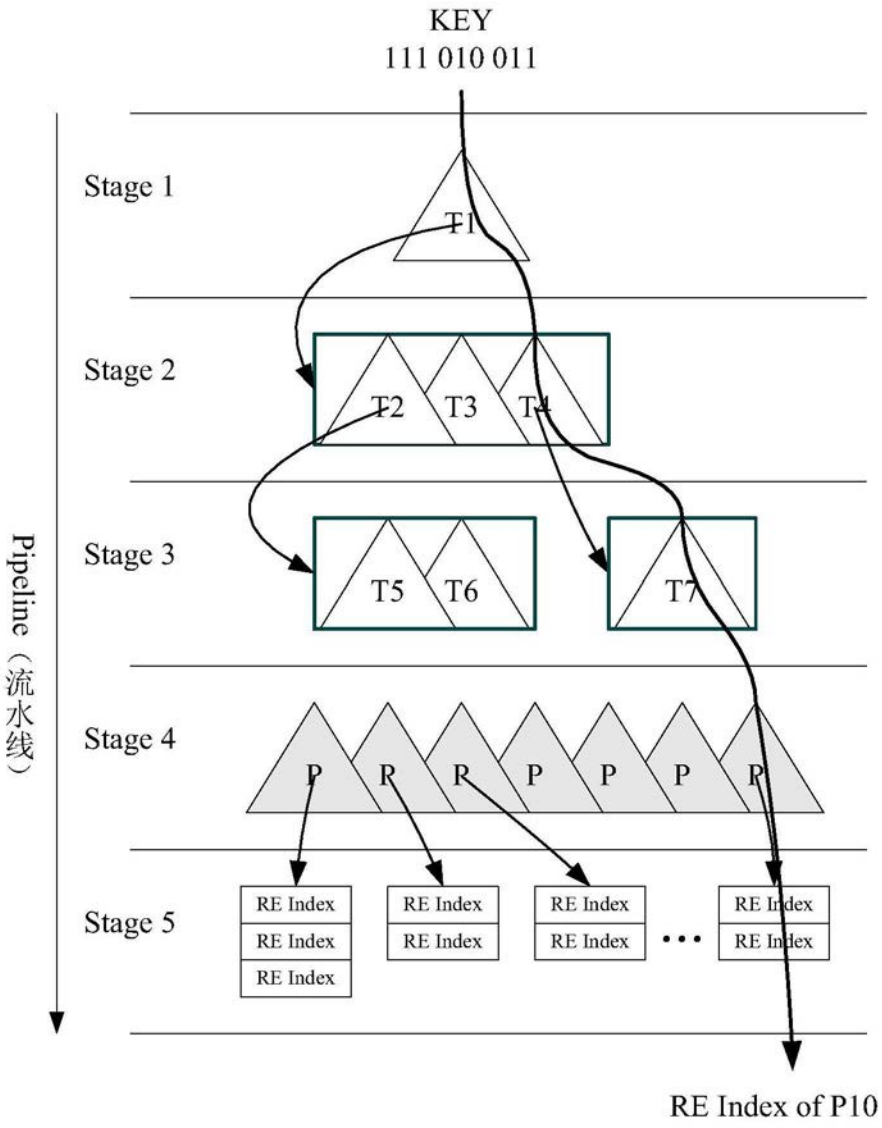

KEY
111 010 011
Pipeline（流水线）
Stage 1
Stage 2
Stage 3
Stage 4
Stage 5
T1
T2
T3
T4
T5
T6
T7
P
RE Index
RE Index of P10

图3

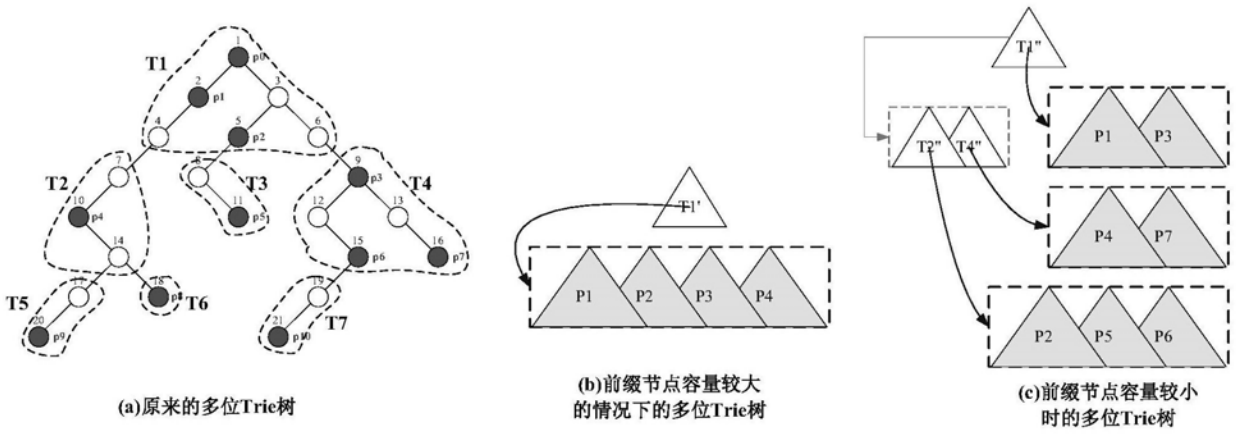

T1
T2
T3
T4
T5
T6
T7
(a)原来的多位Trie树
T1'
P1
P2
P3
P4
(b)前缀节点容量较大
的情况下的多位Trie树
T1"
T2"
T4"
P1
P3
P4
P7
P2
P5
P6
(c)前缀节点容量较小
时的多位Trie树

图4

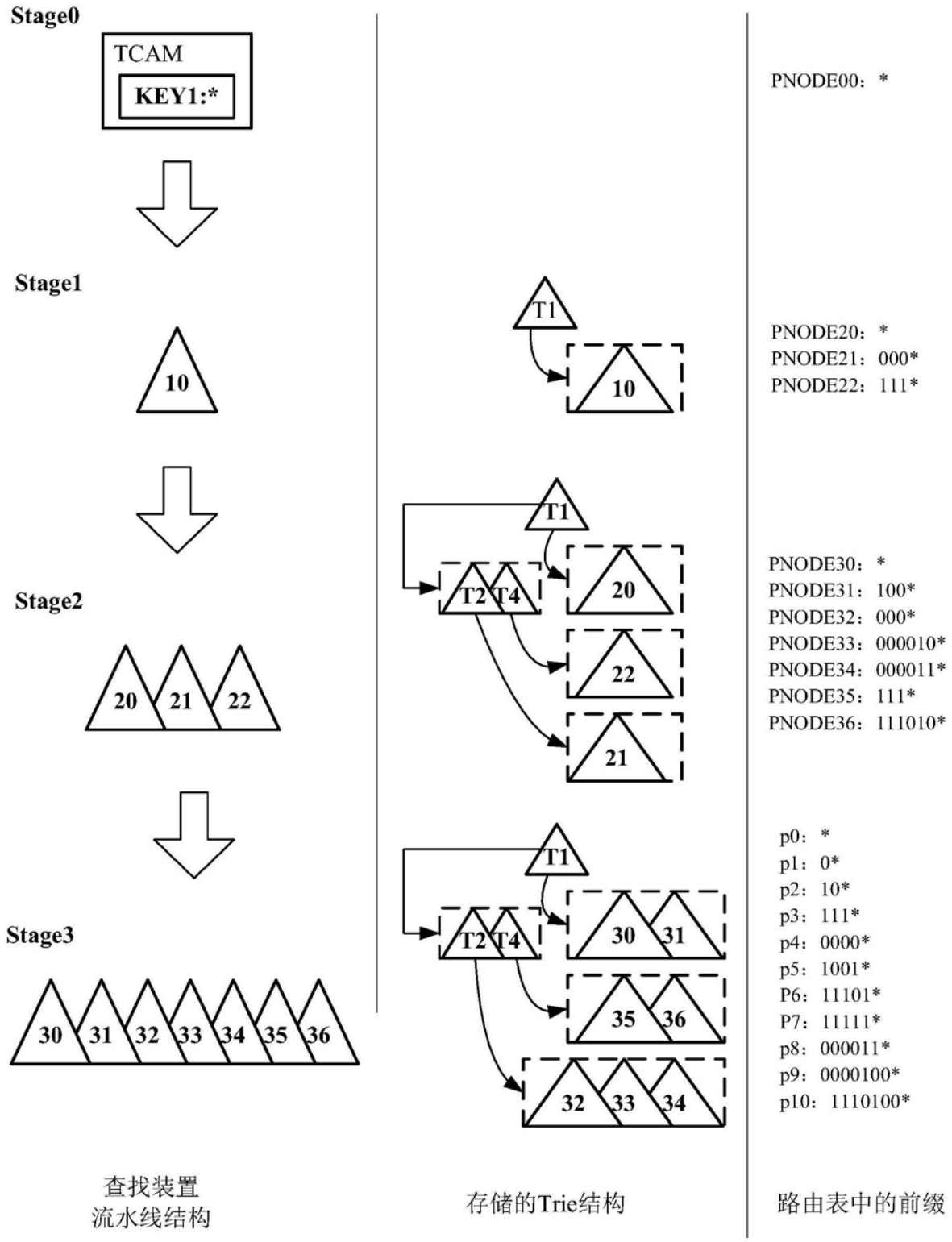

Stage0
TCAM
KEY1:*
Stage1
10
Stage2
20
21
22
Stage3
30
31
32
33
34
35
36
查找装置
流水线结构
T1
10
T1
T2
T4
20
22
21
T1
T2
T4
30
31
35
36
32
33
34
存储的Trie结构
PNODE00：*
PNODE20：*
PNODE21：000*
PNODE22：111*
PNODE30：*
PNODE31：100*
PNODE32：000*
PNODE33：000010*
PNODE34：000011*
PNODE35：111*
PNODE36：111010*
p0：*
p1：0*
p2：10*
p3：111*
p4：0000*
p5：1001*
P6：11101*
P7：11111*
p8：000011*
p9：0000100*
p10：1110100*
路由表中的前缀

图5

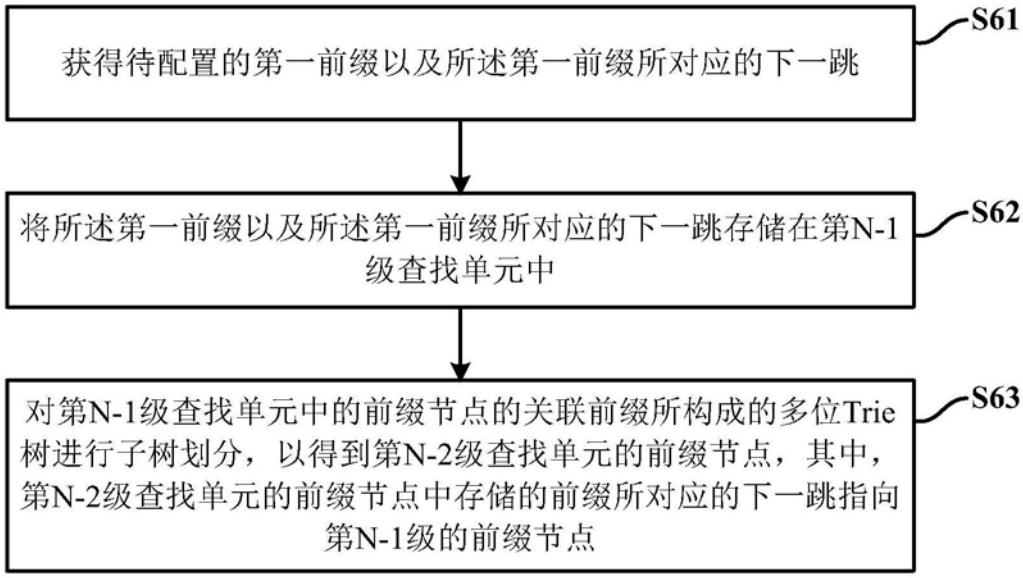

获得待配置的第一前缀以及所述第一前缀所对应的下一跳
S61
将所述第一前缀以及所述第一前缀所对应的下一跳存储在第N-1级查找单元中
S62
对第N-1级查找单元中的前缀节点的关联前缀所构成的多位Trie树进行子树划分，以得到第N-2级查找单元的前缀节点，其中，第N-2级查找单元的前缀节点中存储的前缀所对应的下一跳指向第N-1级的前缀节点
S63

图6a

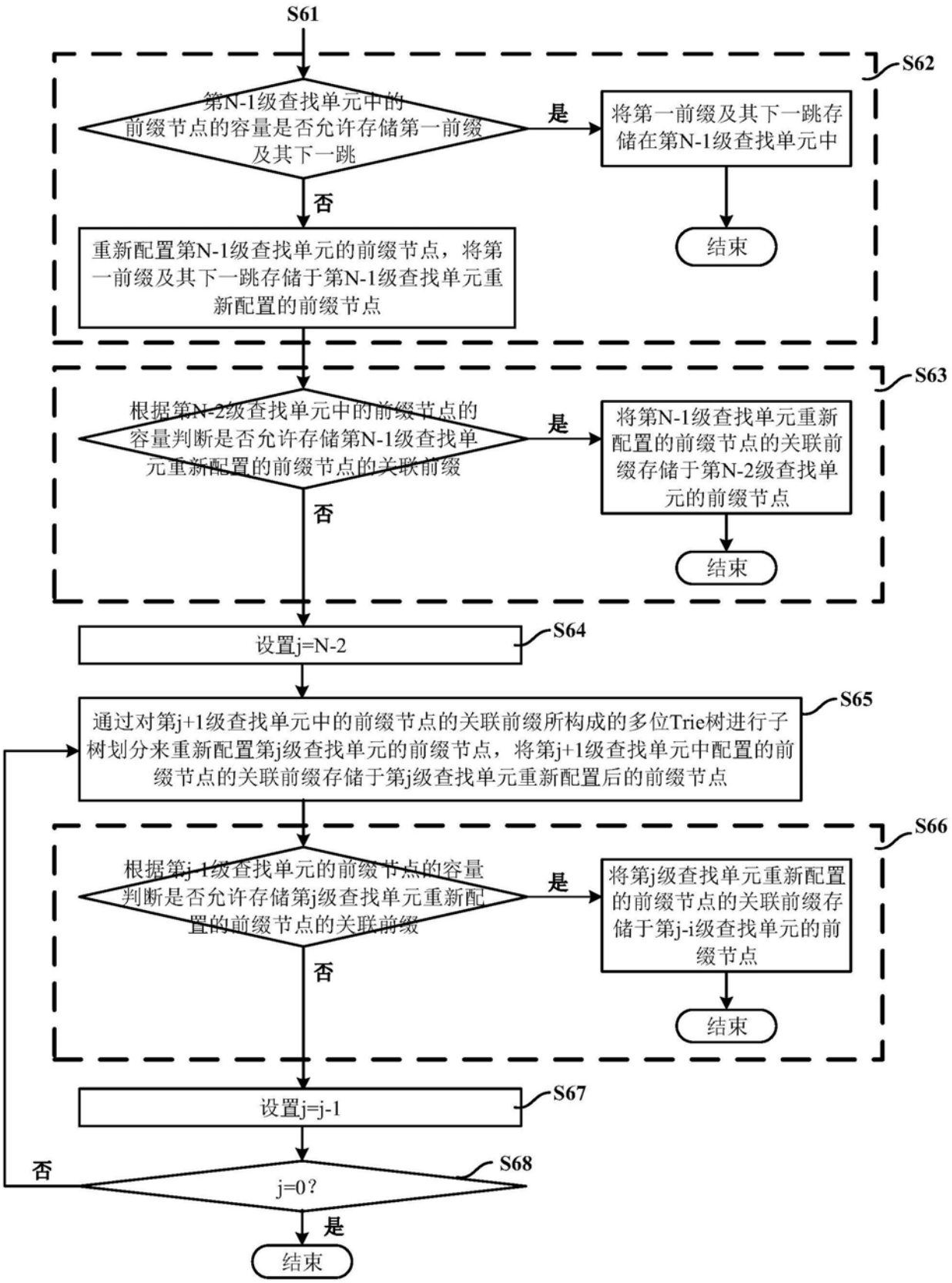

S61
S62
第N-1级查找单元中的前缀节点的容量是否允许存储第一前缀及其下一跳
是
将第一前缀及其下一跳存储在第N-1级查找单元中
结束
否
重新配置第N-1级查找单元的前缀节点，将第一前缀及其下一跳存储于第N-1级查找单元重新配置的前缀节点
S63
根据第N-2级查找单元中的前缀节点的容量判断是否允许存储第N-1级查找单元重新配置的前缀节点的关联前缀
是
将第N-1级查找单元重新配置的前缀节点的关联前缀存储于第N-2级查找单元的前缀节点
结束
否
S64
设置j=N-2
S65
通过对第j+1级查找单元中的前缀节点的关联前缀所构成的多位Trie树进行子树划分来重新配置第j级查找单元的前缀节点，将第j+1级查找单元中配置的前缀节点的关联前缀存储于第j级查找单元重新配置后的前缀节点
S66
根据第j-1级查找单元的前缀节点的容量判断是否允许存储第j级查找单元重新配置的前缀节点的关联前缀
是
将第j级查找单元重新配置的前缀节点的关联前缀存储于第j-i级查找单元的前缀节点
结束
否
S67
设置j=j-1
S68
否
j=0?
是
结束

图6b

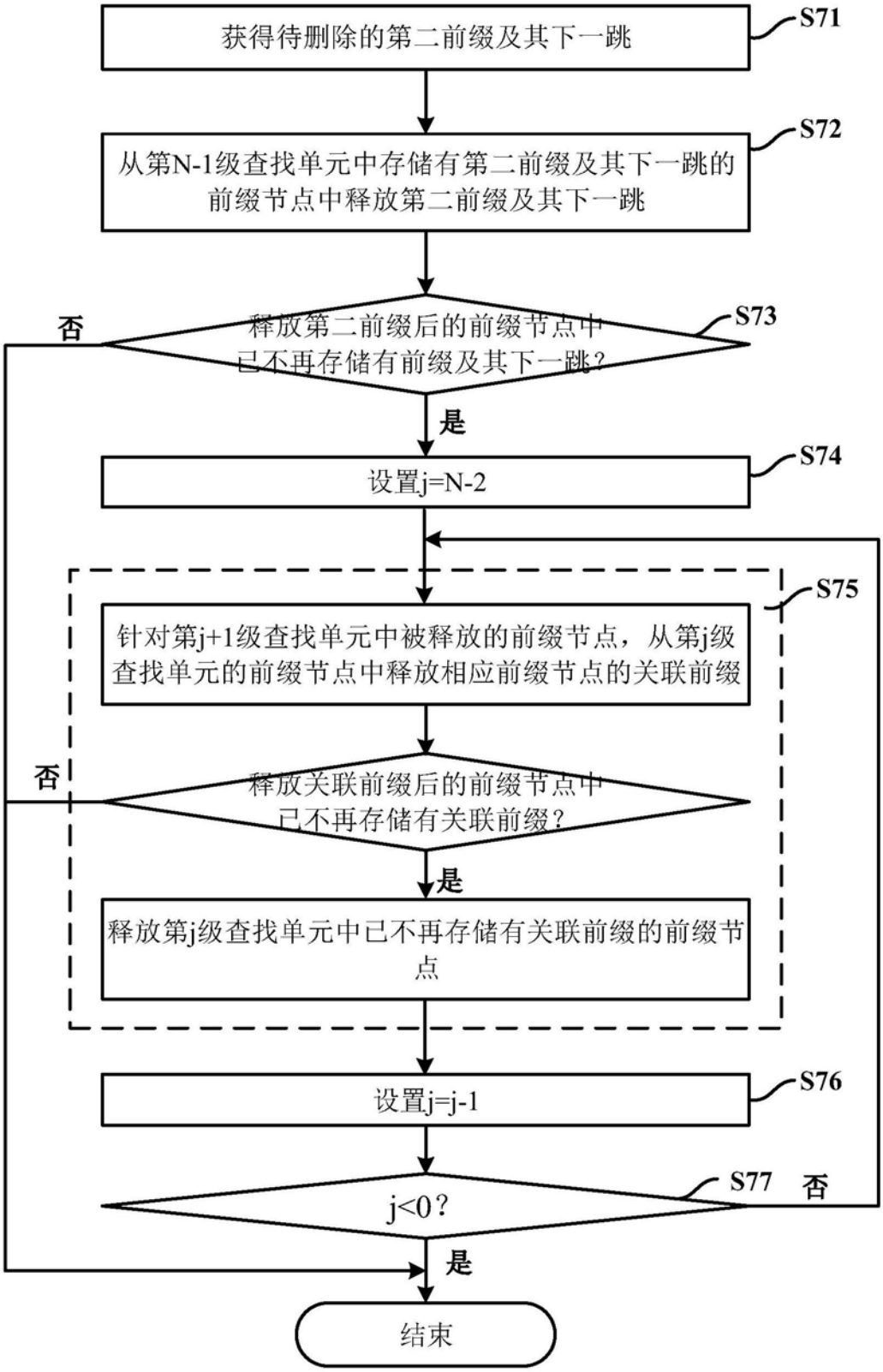

获得待删除的第二前缀及其下一跳
S71
从第N-1级查找单元中存储有第二前缀及其下一跳的前缀节点中释放第二前缀及其下一跳
S72
释放第二前缀后的前缀节点中已不再存储有前缀及其下一跳？
S73
否
是
设置j=N-2
S74
针对第j+1级查找单元中被释放的前缀节点，从第j级查找单元的前缀节点中释放相应前缀节点的关联前缀
S75
释放关联前缀后的前缀节点中已不再存储有关联前缀？
否
是
释放第j级查找单元中已不再存储有关联前缀的前缀节点
设置j=j-1
S76
j<0？
S77
否
是
结束

图7

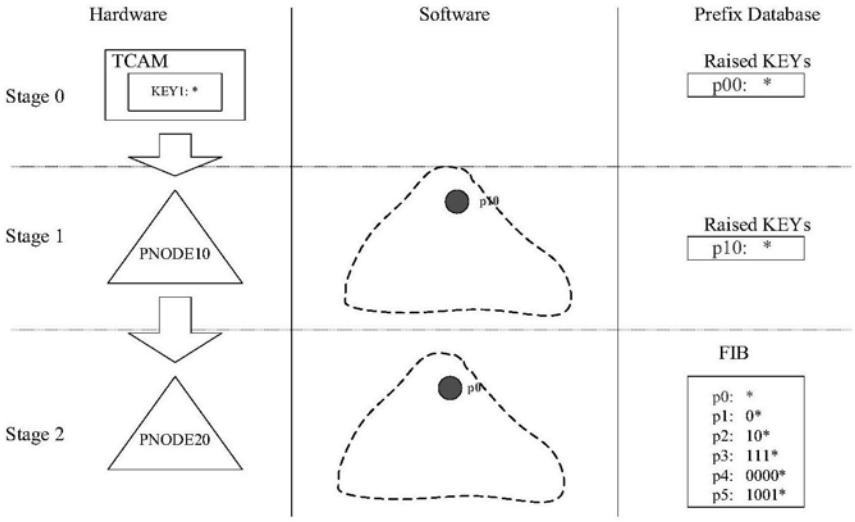

Hardware
Software
Prefix Database
TCAM
KEY1: *
Stage 0
Raised KEYs
p00: *
Stage 1
PNODE10
Raised KEYs
p10: *
FIB
Stage 2
PNODE20
p0: *
p1: 0*
p2: 10*
p3: 111*
p4: 0000*
p5: 1001*

图8a

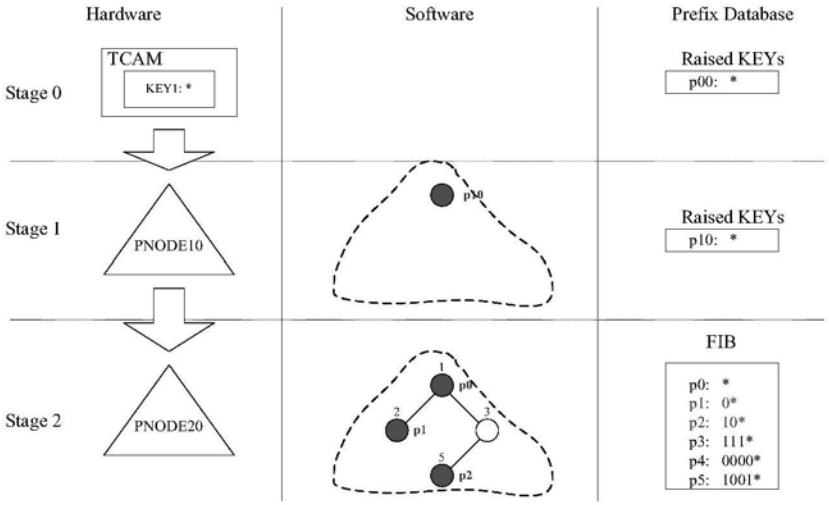

Hardware
Software
Prefix Database
TCAM
KEY1: *
Stage 0
Raised KEYs
p00: *
Stage 1
PNODE10
Raised KEYs
p10: *
FIB
Stage 2
PNODE20
p0: *
p1: 0*
p2: 10*
p3: 111*
p4: 0000*
p5: 1001*

图8b

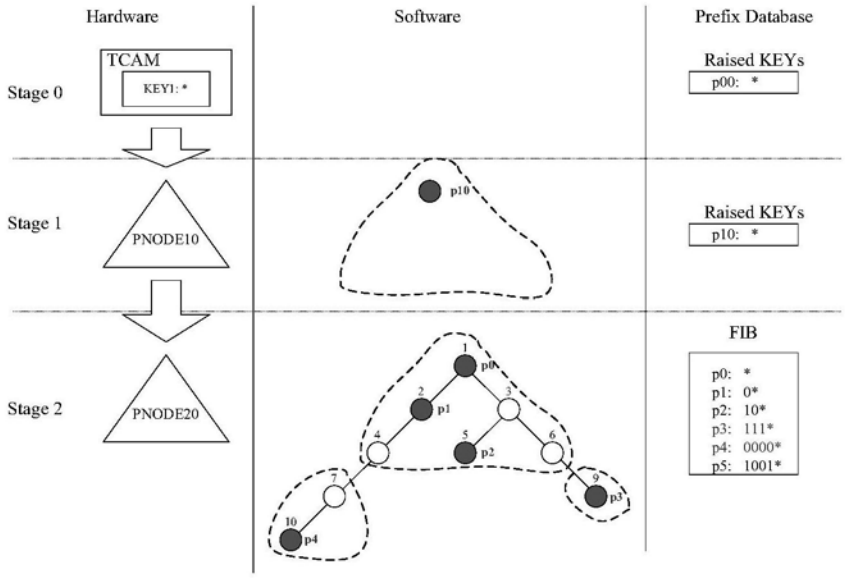
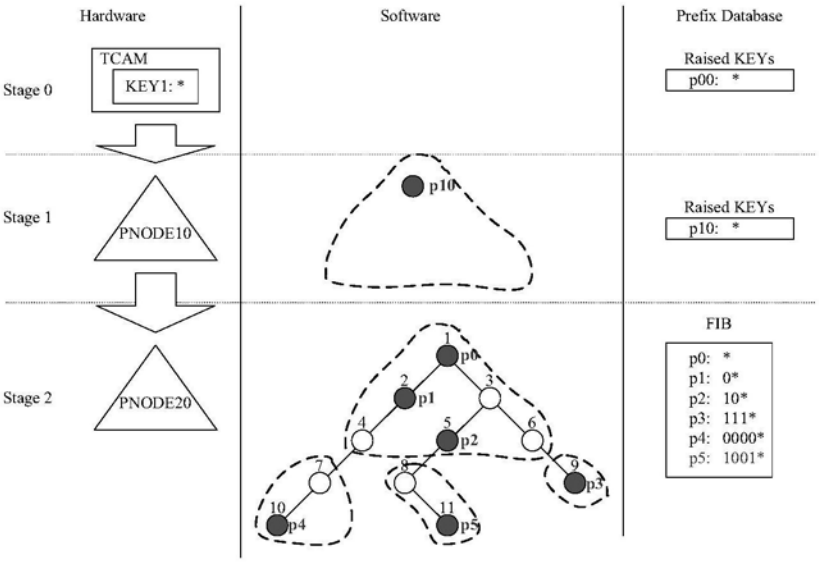
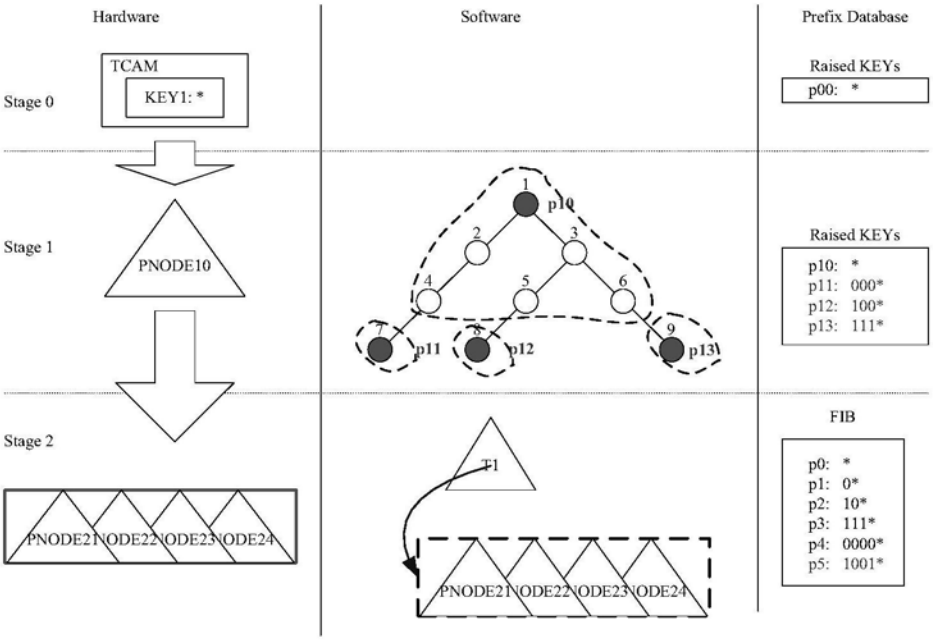

Hardware
Software
Prefix Database
TCAM
KEY1: *
Stage 0
Raised KEYs
p00: *
Stage 1
PNODE10
p10
Raised KEYs
p10: *
Stage 2
PNODE20
FIB
p0: *
p1: 0*
p2: 10*
p3: 111*
p4: 0000*
p5: 1001*
p0
p1
p2
p3
p4

图8c



(1) 插入一前缀触发 PNODE拆分



(2) 在软件中拆分PNODE并相应更新硬件中的PNODE

图8d

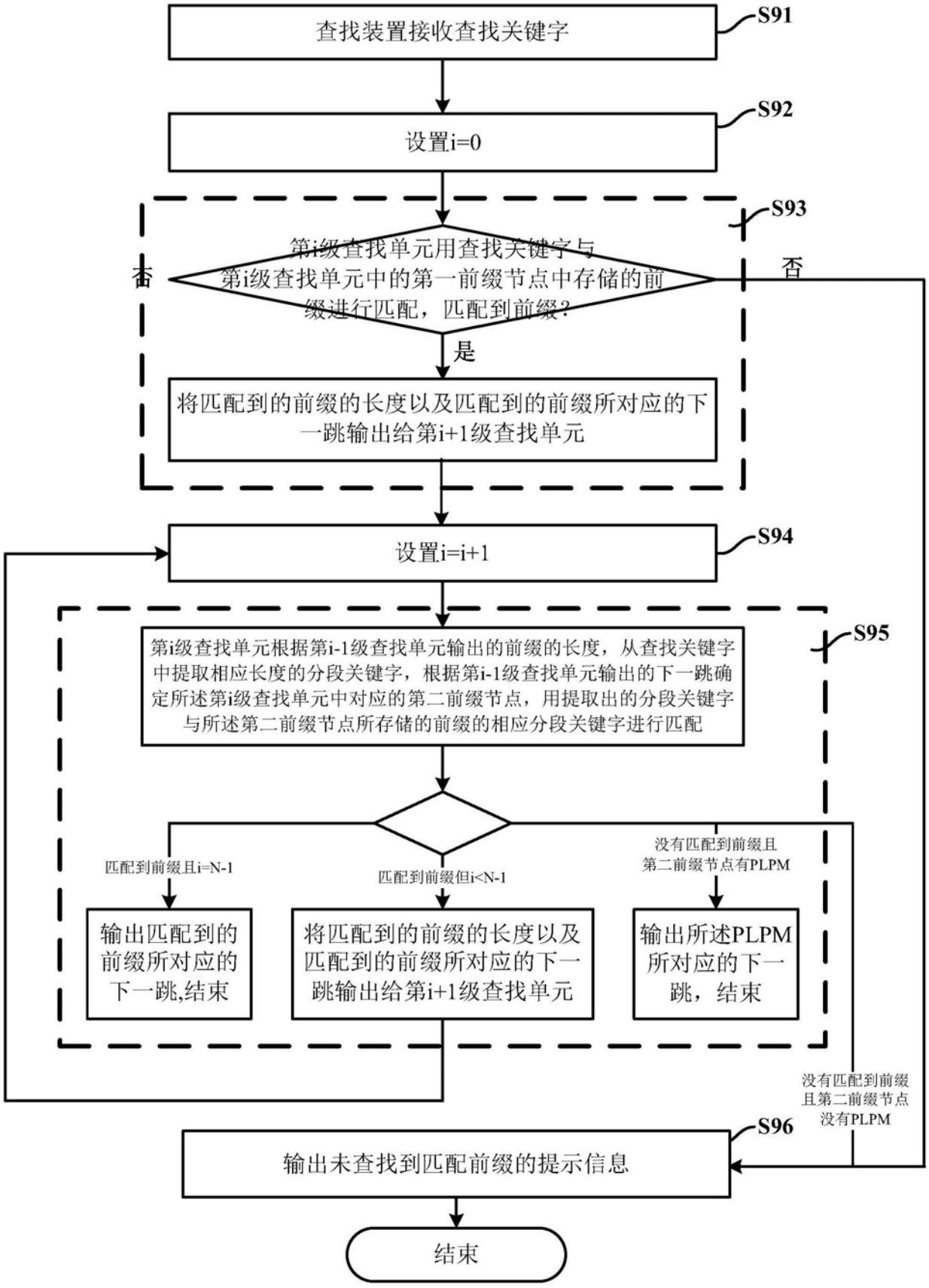

查找装置接收查找关键字
S91
设置i=0
S92
S93
第i级查找单元用查找关键字与第i级查找单元中的第一前缀节点中存储的前缀进行匹配，匹配到前缀?
否
是
否
将匹配到的前缀的长度以及匹配到的前缀所对应的下一跳输出给第i+1级查找单元
设置i=i+1
S94
S95
第i级查找单元根据第i-1级查找单元输出的前缀的长度，从查找关键字中提取相应长度的分段关键字，根据第i-1级查找单元输出的下一跳确定所述第i级查找单元中对应的第二前缀节点，用提取出的分段关键字与所述第二前缀节点所存储的前缀的相应分段关键字进行匹配
匹配到前缀且i=N-1
匹配到前缀但i<N-1
没有匹配到前缀且第二前缀节点有PLPM
输出匹配到的前缀所对应的下一跳,结束
将匹配到的前缀的长度以及匹配到的前缀所对应的下一跳输出给第i+1级查找单元
输出所述PLPM所对应的下一跳，结束
没有匹配到前缀且第二前缀节点没有PLPM
S96
输出未查找到匹配前缀的提示信息
结束

图9a

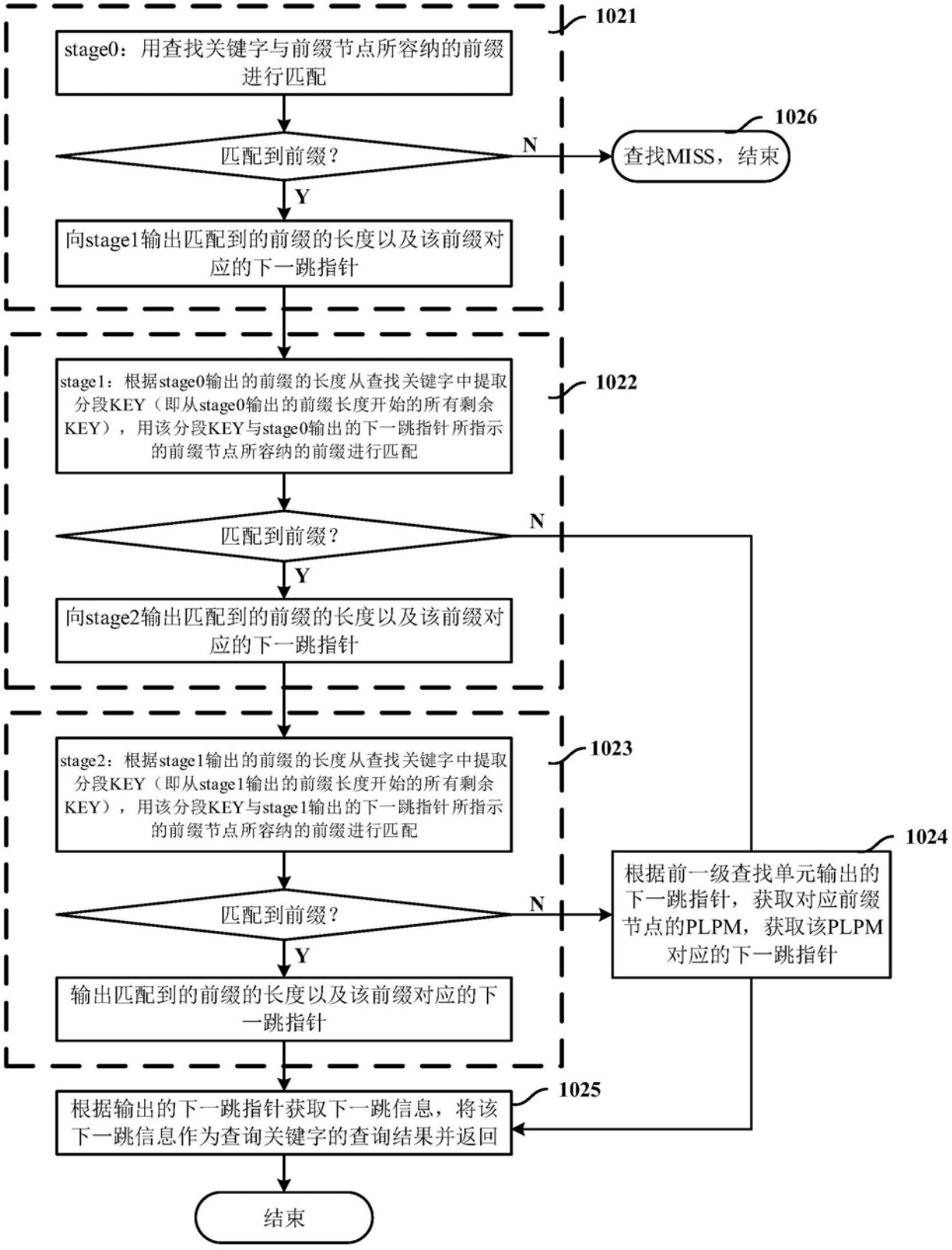

1021
stage0：用查找关键字与前缀节点所容纳的前缀进行匹配
匹配到前缀？
N
1026
查找MISS，结束
Y
向stage1输出匹配到的前缀的长度以及该前缀对应的下一跳指针
1022
stage1：根据stage0输出的前缀的长度从查找关键字中提取分段KEY（即从stage0输出的前缀长度开始的所有剩余KEY），用该分段KEY与stage0输出的下一跳指针所指示的前缀节点所容纳的前缀进行匹配
匹配到前缀？
N
Y
向stage2输出匹配到的前缀的长度以及该前缀对应的下一跳指针
1023
stage2：根据stage1输出的前缀的长度从查找关键字中提取分段KEY（即从stage1输出的前缀长度开始的所有剩余KEY），用该分段KEY与stage1输出的下一跳指针所指示的前缀节点所容纳的前缀进行匹配
1024
匹配到前缀？
N
根据前一级查找单元输出的下一跳指针，获取对应前缀节点的PLPM，获取该PLPM对应的下一跳指针
Y
输出匹配到的前缀的长度以及该前缀对应的下一跳指针
1025
根据输出的下一跳指针获取下一跳信息，将该下一跳信息作为查询关键字的查询结果并返回
结束

图9b