MPLS网的防止循环技术

Technology of Loop Prevention in MPLS Networks

陈跃斌[1-3] 梁 虹[2] 林孝康[3]

(云南民族学院物理系 昆明650031)[1] (云南大学信息与电子科学系 昆明650091)[2]

(清华大学电子工程系微波与数字通信国家重点实验室 北京100084)[3]

Abstract This paper introduces the two algorithms to prevent loop in MPLS networks, and their some of specialities are briefly analysed.

Keywords MPLS, LSP, Loop prevention, PD, CT, Hop count

一、引言

在MPLS(多协议标签交换)网中,标签交换路径(LSP)是由使用标签交换来转发控制消息或数据的一组相互链接的标签交换路由器(LSR)构成的,当某些消息或数据在LSP上发生路由循环时,由于它们长期占用路由资源,造成对其他数据的路由拥挤,严重时甚至形成网络崩溃,因此要加以防止。从防止循环的角度,可将LSP分为两类:一是非融合的LSP:这是指LSP上的每个LSR的每条输入链路都存在着对应的输出链路,但这并不意味着不允许标签融合。二是可融合的LSP:是指在LSP上某个(或多个)LSR处,其多条输入链路可融合到一条输出链路上。其结果,多输入标签就融入了单输出标签。

对非融合LSP,防止循环的一个简单方法是:在每条标签请求消息中都包含一张路由器R的标识符(含地址)ID表,当R转发此消息时,须将自己的ID填入该表中;若R从所收到的消息中发现自己的ID已在这张表中,则检测出路由循环。防止的方法是将造成循环的消息予以丢弃。

而在可融合的LSP中,防止循环却没有这样简单;这是因为在LSP中不同的R上,可能同时发生多个(处)融合的情形。下面介绍的两种方法:"路径向量/扩散法"(Path-Vector/Diffusion Algorithm,PD)和"线程着色法"(Color Thread Algorithm,CT)不仅适于非融合情形,而且适合于融合情形。

二、PD算法

图1给出了此法的一个示例。注意在图中的R_3、R_5处存在融合。图中ACK表示对询问的"确认"应答,NAK表示"非确认"应答。图1(a)~(f)给出了整个流程。

PD算法的流程如图2所示,对照图1的示例,可清楚地理解该流程。

1)当下游节点R_{i+1}收到自上游节点(可能有多个)R_i的ACK时,对R_{i+1}的要求是:

(1)若所收到的ACK不是对尚未完结的询问的应答(称不相匹配),则R_{i+1}丢弃该ACK。

参照图1(c),说明一下"尚未完结"的意思。R_5收到来自R_4的ACK,该ACK仅是R_4对R_5的应答。但因R_5还有另一路上游R_3,R_3又还有其上游R_1、R_2,因而R_5还不能立即向R_5返回应答,它还要继续向其上游发出询问(即询问尚未完结),因此R_5此时收到的ACK不是对R_3这个方向上尚未完结的询问的应答。该ACK仅表明在R_4方向不会发生循环,却不能表明在其他方向也不能发生循环。因此,R_5将此ACK丢弃。

如何判断返回的ACK是从哪个上游R发出的?采用的方法是在ACK中须加入此R的标识符。若收到的ACK(NAK应答也同样)包含的路径向量与发出去的询问包含的路径向量相同(但路由器标识符ID顺序相反),则称此应答与询问是匹配的。

(2)R_i继续将尚未完结的询问扩散到上游,若询问完结,表明已到达最终(上游)的入口路由器R_1,R_1回送ACK(最后的ACK)到其下游,直至到达询问的源端(图1的R_5)收到这个ACK后,才表明全部路径没有循环发生,入标签与出标签才开始标签交换。

在图1(c)中,若没有虚线所示的路径,则当R_3收到来自R_1及R_2的ACK后,便判定此LSP没有循环发生。

2)当R_{i+1}收到自R_i的NAK时,对R_{i+1}的要求是:

陈跃斌 硕士,副教授,主要研究兴趣为宽带网络技术。梁 虹 讲师。林孝康 教授,博士生导师。

(1)若收到的NAK与尚未完结的询问不相匹配，则R_{i+1}丢弃该NAK。

这是因为尽管收到一个NAK，说明有循环，但是我们现在要判断的是在某节点(如图1的R_5)处由于路径发生变化(R_5的下一跳指向R_3)是否会引起循环。因此，从R_5发出的询问，一直要传送到全部上游节点(R_3→R_1→R_5)才结束，这时返回的NAK才是针对这个询问所作的应答。而在询问结束前就收到的NAK不是对这个询问的应答，也就是说，这个NAK所表明的循环，就是存在，也不是因为由R_5的下一跳路径改变所产生的。

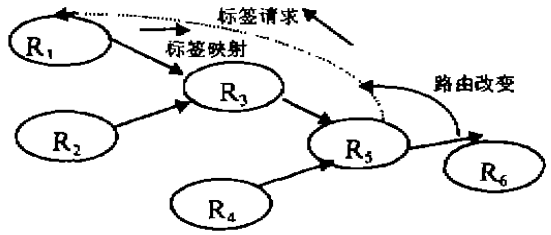
(2)当询问完结，收到NAK时，该NAK返回下游节点，直至询问的发端(图1的R_5)，之后结束扩散算法。

3)R_{i+1}可能同时收到ACK和NAK的情况。如图3所示，若R_i的其中一条输入链路来自一个入口路由器R_2，即会发生R_i同时收到ACK和NAK的情况，不过，如前所述，无论是ACK或NAK应答，都包含有发出此应答的路由器的标识符，即R_i能分辨ACK和NAK是分别由哪个路由器发出的，并将两个应答都返回下游，最后由询问的发端R_5来判断哪条路由有循环，哪条没有。

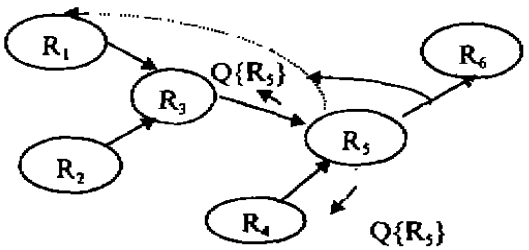
三、CT算法

1. 基本概念

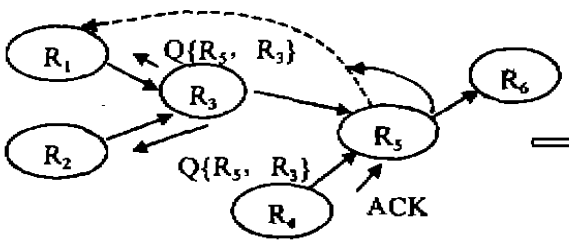
·颜色(color)：实际上是一组代码，由长度固定的两部分组成：产生颜色的节点地址及在生成节点内是唯一的本地标识符。



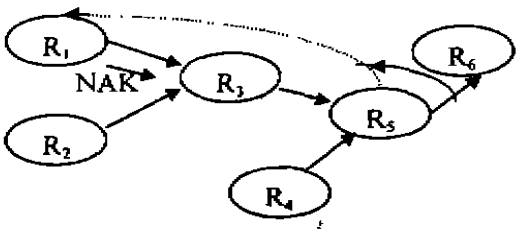
(a) 路由在R_5处发生改变，R_5向R_1发出标签请求，R_1向R_5回送标签映射。



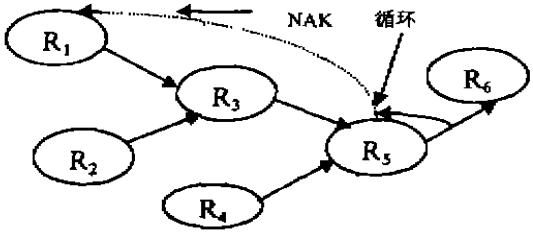
(b) R_5启动扩散算法并向其上游R_3，R_4发出含有自己标识{R_5}的询问$Q\{R_5\}$。



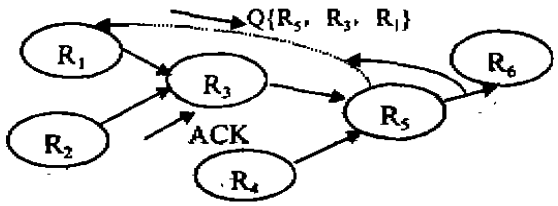
(c) R_3将自己的标识{R_3}加入所收到的$Q\{R_5\}$中，并继续向其上游R_1，R_2转发询问$Q\{R_5, R_3\}$，入口路由器R4向R5返回ACK，R5等待自R3的另一个应答。



(f) R_4向R_3回送NAK，R_3又将NAK转发到R_5，R_5收到NAK后，扩散算法结束。



(e) R_5发现其标识符位于所收到的询问$Q\{R_5, R_3, R_1\}$中，因此向R_1回送NAK。



(d) R_1将{R_1}加入所到的$Q\{R_5, R_3\}$中，并继续向其上游R_3转发$Q\{R_5, R_3, R_1\}$，入口路由器R2向R3回送ACK，R3等待自己R1的另一个应答。

图1 PD算法的一个示例

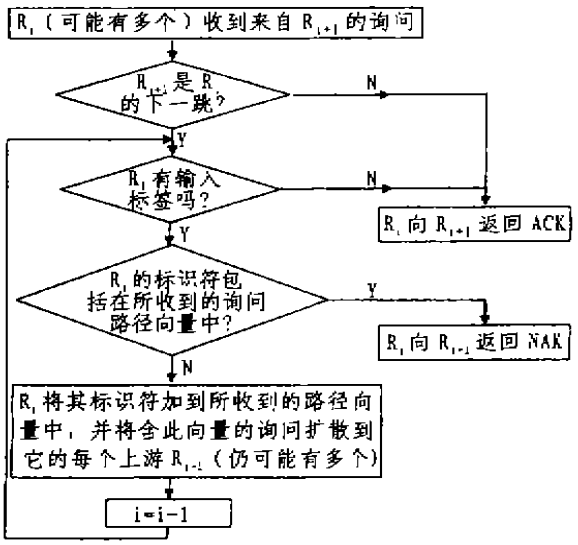


图2 PD算法流程

·链路的跳数(hop count)：LSP上的最上游的节点与该条链路之间的链路数，也简称跳数。

·线程(thread)：包含(颜色、跳数、生存时间TTL)的一组参数集。

2. 着色规则及示例

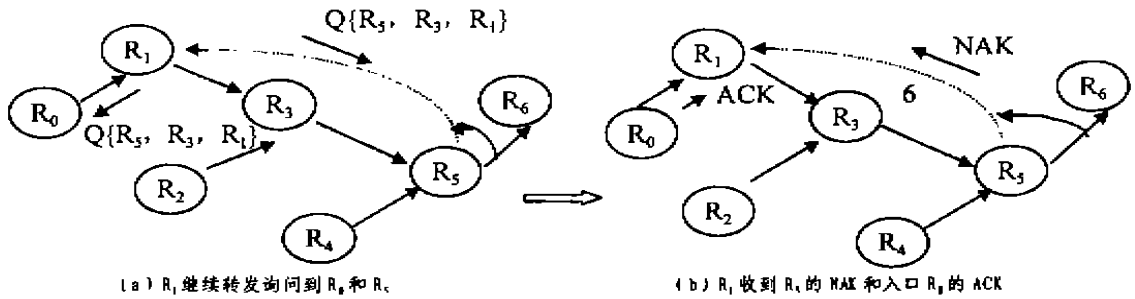
CT算法就是沿着LSP的下游方向给每一条链路着色，将线程参数进行扩散并由此判断循环发生与否的过程。对链路着色意味着通过链路传递节点的地址及标识符信息。

图4给出了一个示例。

由发现其下一跳发生改变的节点(图4(a)中的R_5)生成颜色(即R_5生成其地址及标识符，假定用红色表示，下同)并由其输出链路传到新的下一跳(图4(a)的R_3)，称为启动着色过程。链路旁的数字表示跳数。

节点从其输入链路收到颜色后，对该颜色的处理须服从以下规则：

(1)当正在接收颜色的节点的输入链路是一条新



(a) R_1继续转发询问到R_3和R_5　　(b) R_1收到R_5的NAK和入口R_3的ACK

图3 R_1同时收到ACK和NAK的例子

链路(图中的那条虚线)且该节点存在输出链路时，则该节点生成一种新颜色(蓝)并将它从输出链路扩散到下一跳如图4(b)。

(2)否则(即输入链路不是新的)，将已收到的颜色扩散到下一跳。如图4(c)。

着色进程一直向下游扩散，直至LSP确认是否发生循环为止。若没有循环，则以一跳接一跳(hop-by-hop)的方式向上游返回ACK应答。若发生循环，则着色进程停止，且不返回ACK。

3. 循环判断及处理

(1)当且仅当某节点发现两条或多条输入链路具有相同的颜色、或者其中的一条链路的颜色是由该节点生成时(如图4(d)中的R_1)，则LSP发生循环；若如此，除非输出链路的跳数H已经是未知的，否则，该节点生成一新颜色，并与跳数H=U(未知)的刷新消息一起由输出链路扩散到新的下一跳(见图4(d))。未知跳数被认为是大于任何其他的已知跳数，它被用于防止控制消息进入循环检测。

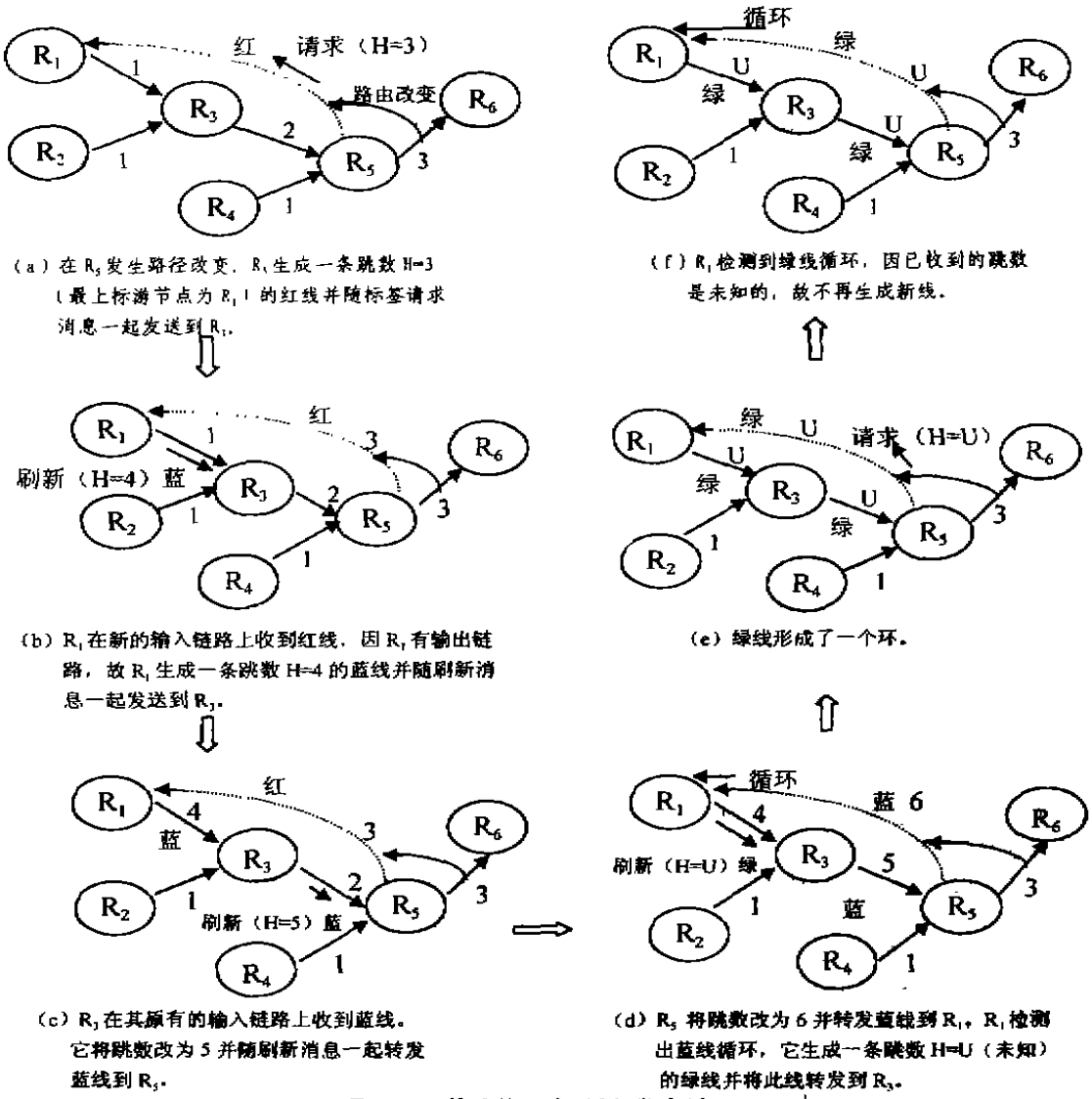
(2)当节点收到ACK应答时，说明没有循环，节点将所有的输入、输出链路的颜色变为透明色，以表明不再进行着色过程，并扩散此ACK到上游节点。同时，链路的跳数也要随之改变。若H_{max}表示最大输入链路跳数，H_0表示刚收到ACK的节点的输出链路跳数，则如果$(H_{max}+1)<H_0$，那么用$(H_{max}+1)$刷新H_0，且既不启动着色进程又不等待ACK，就将此刷新消息向下游扩散。

4. 线程参数中 Hop Count 和 TTL 的作用

只有当着色进程到达LSP的出口节点时，才能认为此LSP无循环。利用链路的跳数，可以不必将着色进程自始自终地进行，就能提前判新，方法如下：

(1)对于一条已经建立了的但尚未进行着色进程的LSP，当节点给一条被融入到这条LSP中的输入链路L着色时，若L的跳数小于这条LSP输出链路的跳数，则该LSP没有循环。

(2)对于一条正被着色的LSP，当节点给一条被融入到该LSP中的输入链路L着色时，若L的跳数小于



（a）在 R_5 发生路径改变，R_5 生成一条跳数 H=3（最上标游节点为 R_1）的红线并随标签请求消息一起发送到 R_1。

（b）R_1 在新的输入链路上收到红线，因 R_1 有输出链路，故 R_1 生成一条跳数 H=4 的蓝线并随刷新消息一起发送到 R_3。

（c）R_3 在其原有的输入链路上收到蓝线。它将跳数改为 5 并随刷新消息一起转发蓝线到 R_5。

（d）R_5 将跳数改为 6 并转发蓝线到 R_1，R_1 检测出蓝线循环，它生成一条跳数 H=U（未知）的绿线并将此线转发到 R_3。

（e）绿线形成了一个环。

（f）R_1 检测到绿线循环，因已收到的跳数是未知的，故不再生成新线。

图4　CT 算法的一个示例(发生循环)

这条 LSP 的输出链路的跳数，则该节点不扩散着色进程，它等待 ACK 应答。

TTL 的作用是避免着色进程的无限执行，其初值在生成颜色时设置。每当颜色扩散到一个下游节点，该值就减1；当 TTL 为0时，着色进程终止。

结束语　本文介绍的两种防止循环的算法，对循环是否发生的判断，本质上还是根据 LSP 上路由器(节点)的地址及标识符信息来作出的，这是两者的共同点。不过它们也有不同之处：PD 法要先以上游方式发出询问，之后要等到收到以下游方式传送的应答信息后才能作出判断；CT 法的着色进程(实际上是传 R 的地址及标识符信息)是以下游方式进行，应答信息以上游方式传送。因而可以推断，两种方法对以不同标签发布方式(标签有序下游节点按需分配方式(DoD)及标签有序下游节点分配方式(DU))建立的 LSP 会产生不同的效果。CT 与 PD 孰优孰劣，不能一概而论，这取决于具体的网络结构和节点位置，情况较为复杂，需要编制专门软件来模拟。

参 考 文 献

1　IETF，draft-ietf-mpls-arch-02，txt. Multiprotocol Lable Switchung Architecture，1999

2　IETF，draft-ietf-mpls-ldp-06，txt，LDP Specification，2000

3　IFTF，draft-ietf-mpls-loop-prevention-02. txt，MPLS Loop Prevention Mechanism，1999