文章编号：1005－8893（2006）04－0036－03

网络拓扑的存储和变换*

王　涛，史书明，陆建德
（苏州大学 计算机科学与技术学院，江苏 苏州 215006）

摘要：目前不少网管软件都具有拓扑发现功能，但很少提供将当前拓扑与以往拓扑比较的功能，也很少提供对拓扑进行变换的功能，该研究的目的是实现这两项功能。利用二维表存储拓扑，并在此基础上提出了拓扑变换的算法，从而实现了拓扑的存储和变换。

关键词：拓扑存储；拓扑表；拓扑变换
中图分类号：TP 393. 02　　文献标识码：A

Store and Turn Network Topology

WANG Tao，SHI Shu－ming，LU Jian－de
（School of Computer Science & Technology，Soochow University，Soochow 215006，China）

Abstract：Much of software of network management has function of topology discovery，but less of it provides comparison between current and past topology，nor has function of turning network topology. The purpose of this study is completing those two functions by using table to store TOP. They show the algorithm of turning topology，also.

Key words：network topology storage；topology table；network topology turning

作为网络管理的重要部分——网络拓扑的发现一直受到关注。发现当前网络的拓扑固然重要，同时网管员还经常需要将当前的拓扑与以往的拓扑作比较，这就涉及到了拓扑的存储；另外，得到的拓扑，有时需要切换不同的角度观察，这就涉及到了拓扑的变化。下面就这两方面逐一讨论。

1　拓扑的存储

从网络管理员的角度来看“拓扑发现”这项功能，应该说极少有网管员纯粹为了发现而发现——通常情况下，网管员对自己辖区的网络拓扑是清楚的，网管员使用拓扑发现的目的往往是为了查验网络的当前状况，从而发现问题，而发现问题最简便的途径就是拿当前的拓扑与以往的拓扑作比较，这就需要将以往的拓扑保存下来，即拓扑的存储。

这里所说的“拓扑的存储”是指如何把拓扑信息存储到数据库，也就是二维表中。存储除了要保证正确，还要考虑到今后提取和使用的方便，所以拓扑的存储结构必然受到拓扑发现算法的影响。考虑到目前多数网络拓扑发现的算法都是从一个种子路由器（一般就是本机所在子网的网关）开始，通过路由信息，按广度优先的顺序发现指定深度内的设备（显然用“广度优先”是与“发现指定深度内的设备”的需求密切相关的）[1]。所以这个二维表的表结构就见表 1 所示（设：指定的发现深度为 n）。

* 收稿日期：2006－05－10
作者简介：王涛（1977－），男，江苏无锡人，硕士研究生。

为了描述方便，下文中将 Router 缩写为 R；Level*n* 缩写为 L（*n*）；Deep 缩写为 D。

表 1 “拓扑表”结构

Table 1 Structure of TOP table

字段	描述
ID	序号，充当主键，可用自动编号类型
Router(R)	节点(设备)标识
Level1(L(1))	访问到该设备时经过的第 1 个设备，默认值为空
Level2(L(2))	访问到该设备时经过的第 2 个设备，默认值为空
…	…
Leveln(L(*n*))	访问到该设备时经过的第 *n* 个设备，默认值为空
Deep(D)	所在深度

按照这种表结构创建的表被称为“拓扑表”。见图 1 的拓扑（种子路由器是 R0，指定深度为 4），其拓扑表即为表 2 的样子。

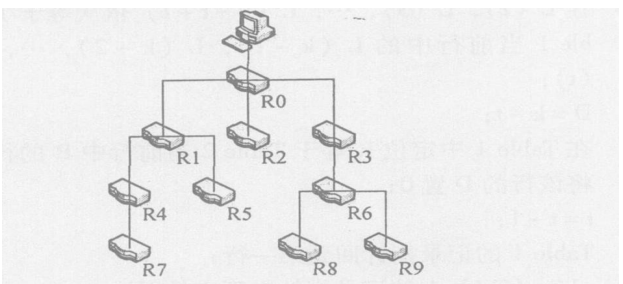

图 1 拓扑 1

Fig 1 TOP 1

表 2 拓扑 1 的拓扑表

Table 2 TOP table for TOP 1

ID	R	L(1)	L(2)	L(3)	L(4)	D
1	R0	R0	○	○	○	1
2	R1	R0	R1	○	○	2
3	R2	R0	R2	○	○	2
4	R3	R0	R3	○	○	2
5	R4	R0	R1	R4	○	3
6	R5	R0	R1	R5	○	3
7	R6	R0	R3	R6	○	3
8	R7	R0	R1	R4	R7	4
9	R8	R0	R3	R6	R8	4
10	R9	R0	R3	R6	R9	4

按照被算法发现的先后次序，各个设备的信息以及从种子路由器到达该设备的路径信息被依次记入表中。用这样的表信息，自上而下体现着广度优先的特征，可以很容易地加以应用，譬如根据表信息利用 tree 控件建立树模型[2]，因为通常在构建这个模型时也是逐层构建的，见图 2 所示。

解决了拓扑的存储，实现当前拓扑与以往拓扑的比较，接下来讨论如何对当前的拓扑进行变换。

2 拓扑的变换

在网络管理的实际工作中，从不同角度观察网络拓扑的功能还是很有用的。这里所谈的拓扑变化正是指将得到的拓扑以拓扑内别的节点为根节点来重新呈现。譬如图 1 的拓扑，若以 R9 为根节点，则应呈现如图 3 的面貌，而若以 R3 为根节点，则应呈现图 4 的面貌。

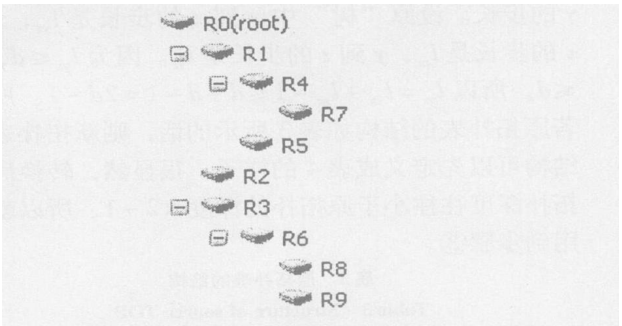

图 2 拓扑 1 的树

Fig 2 Tree for TOP 1

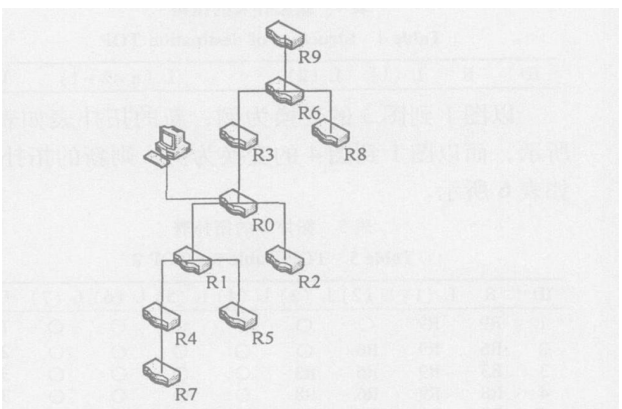

图 3 拓扑 2（拓扑 1 以 R9 为根的形态）

Fig 3 TOP 2 (Shape of TOP 1 when R9 as root)

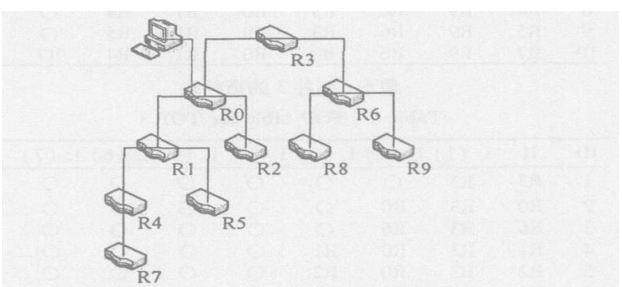

图 4 拓扑 3（拓扑 1 以 R3 为根的形态）

Fig 4 TOP 3 (Shape of TOP 1 when R3 as root)

借助前面的拓扑存储可以通过一定的算法实现拓扑变换，实现流程其实就是依据原有的拓扑表信息，以指定的节点为新的根构建一个新的拓扑表。

步骤：①创建一个新的拓扑表；②将原拓扑表中的信息，按照新的根节点转换到新的拓扑表中；③删除无用的列。具体如下：

步骤①：首先是创建新的拓扑表——不难发现，新拓扑的深度≤原拓扑的深度×2－1。原因如

下：设原拓扑的深度是 D，根节点是 x。新拓扑的含义就是以原拓扑中其他节点（假设是 y）为新的根，重新梳理这颗“树”，这颗新“树”的最大深度就是原“树”中离 y 最远的节点（假设是 z）到 y 的步长。设原“树”中 y 到 x 的步长是 l_{yx}，z 到 x 的步长是 l_{zx}，y 到 z 的步长是 l_{yz}。因为 $l_{yx}\leqslant d$，$l_{zx}\leqslant d$，所以 $l_{yz}=l_{yx}+l_{zx}-1\leqslant d+d-1=2d-1$。所以若原拓扑表的结构如表 3 所示的话，则新拓扑表的结构可以先定义成表 4 的样子。很显然，转换后的拓扑深度往往小于原拓扑的深度×2－1，所以就会用到步骤③。

表 3 原拓扑表的结构

Table 3 Structure of source TOP

ID	R	L (1)	L (2)	…	L (n)	D

表 4 新拓扑表的结构

Table 4 Structure of destination TOP

ID	R	L (1)	L (2)	…	L ($n\times 2-1$)	D

以图 1 到图 3 的变换为例，新的拓扑表如表 5 所示，而以图 1 到图 4 的变换为例，则新的拓扑表如表 6 所示。

表 5 拓扑 2 的拓扑表

Table 5 TOP Table for TOP 2

ID	R	L (1)	L (2)	L (3)	L (4)	L (5)	L (6)	L (7)	D
1	R9	R9	○	○	○	○	○	○	1
2	R6	R9	R6	○	○	○	○	○	2
3	R3	R9	R6	R3	○	○	○	○	3
4	R8	R9	R6	R8	○	○	○	○	3
5	R0	R9	R6	R3	R0	○	○	○	4
6	R1	R9	R6	R3	R0	R1	○	○	5
7	R2	R9	R6	R3	R0	R2	○	○	5
8	R4	R9	R6	R3	R0	R1	R4	○	6
9	R5	R9	R6	R3	R0	R1	R5	○	6
10	R7	R9	R6	R3	R0	R1	R4	R7	7

表 6 拓扑 3 的拓扑表

Table 6 TOP table for TOP 3

ID	R	L (1)	L (2)	L (3)	L (4)	L (5)	L (6)	L (7)	D
1	R3	R3	○	○	○	○	○	○	1
2	R0	R3	R0	○	○	○	○	○	2
3	R6	R3	R6	○	○	○	○	○	2
4	R1	R3	R0	R1	○	○	○	○	3
5	R2	R3	R0	R2	○	○	○	○	3
6	R8	R3	R6	R8	○	○	○	○	3
7	R9	R3	R6	R9	○	○	○	○	3
8	R4	R3	R0	R1	R4	○	○	○	4
9	R5	R3	R0	R1	R5	○	○	○	4
10	R7	R3	R0	R1	R4	R7	○	○	5

步骤②：表 5、表 6 中的数据是由步骤②生成，具体算法如下：

目标：设 Table 1 为以 R0 为根，深度为 n 的拓扑表。现在要改以 R_x 为根，变换拓扑，并将变换的结果同样以拓扑表的形式保存到 Table 2 中。

描述：

首先，创建如表 4 的 Table 2 空表，并在 Table 2 中插入第一条记录：R＝Rx；L（0）＝Rx；C＝1；

通过 Table 1 中的列 R，定位节点 Rx 的所在行，将该行的 D 置为 0；

//在 Table 1，用 D＝0 来标识该行所代表的节点已经过转换处理

获取 Rx 的所在深度，即 D 的值（假设为 k）；

t＝k－1； //定义一个循环控制变量 t

while（t＞0）

{在 Table 2 中插入新的空白行；

Table 2．R＝Table 1．L（t）；Table 2．L（1）＝Rx；

让 L（2），L（3），…，L（k－t＋1）依次等于 Table 1 当前行中的 L（k－1），L（k－2），…，L（t）；

D＝k－t；

在 Table 1 中定位 R 等于 Table 2 当前行中 R 的行，将该行的 D 置 0；

t＝t－1；}

Table 1 的记录指针回到第一行；

while（Table 1 的记录指针未到文件尾）

{If（Table 1 当前行的 D＝＝0）

{Table 1 的记录指针向下移 1；continue；}

在 Table 2 中插入新的空白行，Table 2．R＝Table 1．R；Table 2．L（1）＝Rx；

在 Table 2 中查找 R＝Table 1．L（Table 1．D－1）的行（假设为第 j 行，该行的 D＝m）；

//这样的行，是必定能找到的，因为 Table 1 中的记录是按照深度（即 D）递增存放的

让新插入行的 L（2），L（3），…，L（m）依次等于 Table 2 第 j 行中的 L（2），L（3），…，L（m）；L（m＋1）＝本行的 R；

Table 1 的记录指针向下移 1；}

最后，对 Table 2 中的记录按照列 D 递增排序

步骤③：删除新拓扑表中的无用空白列。

本文讨论了如何将依据广度优先算法发现的网络拓扑保存在二维表中，阐述了如何基于拓扑表实现网络拓扑的变换，举例说明了拓扑变换的含义。

参考文献：

[1] 邱建林，何鹏．一种改进的网络拓扑发现方法［J］．计算机应用，2005，25（4）：891－893.

[2] 李玉鹏，王换招，田海燕．基于 SNMP 和 JAVA 的网络拓扑发现［J］．计算机工程与应用，2004，（5）：152－154.