基于 SNMP 物理网络拓扑发现算法①

A Physical Network Topology Discovery Algorithm Based on SNMP Protocol

陈　锟　王正华　(国防科学技术大学 计算机学院 湖南 长沙 410073)

摘　要：对网络进行准确的拓扑发现，对于网络的故障检测、性能分析具有重要意义。通过对目前网络拓扑发现算法的研究、综合和改进,本文提出了一个完整并且通用的物理网络拓扑发现算法。该算法基于SNMP协议，使得算法具有广泛的适应性,能够完全准确的发现网络中设备以及它们之间的连接关系。

关键词：网络拓扑 SNMP

1　引言

随着网络的不断变化，使得网络的管理和维护变得越来越困难。然而，一个好的网络管理系统首先要掌握的就是整个管理网络的拓扑结构 。网络管理中的拓扑发现主要目的就是获取和维护网络中元素的信息以及它们之间的连接关系信息，最终实现对它们的有效管理。通常情况下，IP 网络可划分成主干网和子网二级结构，他们分别对应于网络结构的第 3 层(网络层，如图 1 所示)和第 2 层(数据链路层，如图 2 所示)。主干网通过路由器把各个子网连接起来，而子网通常包括交换机、网桥、Hub 和大量的主机。

物理网络拓扑指的是一个通信网内部实体的实际物理连接(对应网络结构第 2 层的连接)。由于交换机是基于网络第 2 层的设备,所以用传统的基于 TCP/IP 的方法难以发现其相互之间关系,那就必须借助另外的方法。目前，很多网络设备供应商在其网络设备中都支持 SNMP 协议。因此,我们可以将每个以太网交换机认为是一个多口透明网桥(bridge)或者叫生成树网桥(spanning tree bridge),遵守 RFC1493 协议。SNMP Bridge MIB-II 就实现了 RFC1493,它提供了以太网桥的状态信息,其中我们感兴趣的是地址转发表。它存储了网桥转发与过滤数据帧的信息。从以太网交换机工作原理上看我们不难发现地址转发表并未直接提供交换机之间的连接信息,这就必须采用另外的方法来找出它们之间的连接关系。

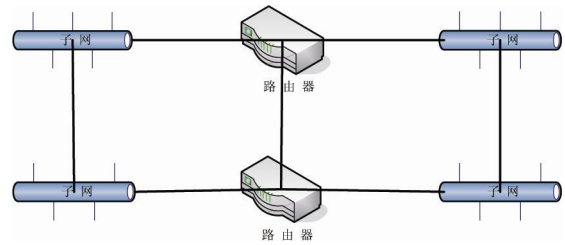

图 1　第三层网络拓扑图

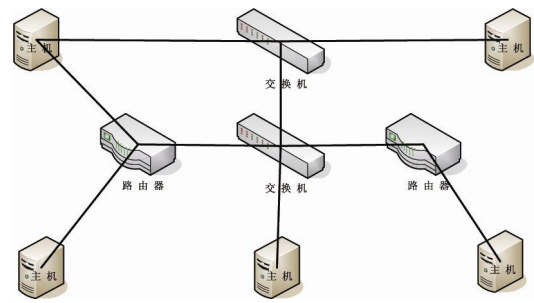

图 2　第二层网络拓扑图

2　SNMP 简介

SNMP 是一种基于 TCP/IP 的网络管理协议,用于监视和控制各种可管理的网络设备。SNMP 采用 MIB-II,利用无连接的 UDP 协议在管理者和代理之间进行管理信息的传递,借助于检查参数或监督特定的网络状态来进行网络设备的管理。SNMP 管理模型由四部分组成:1)管理者(manager) :驻留在管理站中的软件进程,接收来自网管用户的服务请求,并通过 SNMP

① 收稿时间:2008-08-18

向指定代理发送网管请求以监控被管理的网络资源，如交换机、路由器等的运行情况;2) 管理代理(agent) :驻留在被管设备(主机、路由器、交换机、打印机等) 中的软件进程,负责收集被管网络资源的状态信息,并应管理者的网管请求，返回相应的操作结果; 3)MIB :存放被管理对象(managed object,MO) 的信息,由代理维护、由管理器进行查询和设置。4) 管理协议(SNMP) :用于在管理者和代理间传递管理信息。

关于 MIB 有二个基本而又重要的规范: MIB-II 和 Bridge MIB。MIB2II 规范定义了所有网络设备必须支持的一些管理信息,如设备基本信息、接口统计信息、IP 统计信息等;而 Bridge MIB 规范定义了网桥、交换机等设备必须支持的桥管理信息。

3 物理网络拓扑发现需要解决的问题

目前对网络进行物理拓扑发现需要解决下面的问题:

(1) 前提假设太多。拓扑发现算法应该对被研究的网络没有限制或者只有很少的假设，于是我们应该利用绝大多数被管对象最长保存的信息。但是，由于第二层网络设备并不知道(或只知道很少)和它们连接的其他设备的情况，因此在网络的第二层推断网络的物理连接情况是很困难的。

(2) 二层设备对高层协议是透明的。目前网络设备经常工作在 ISO 协议规定的不同层次，算法应当对各个网络设备都建立正确的互连关系，这一点也有一定的困难.因为在交换子网中，对于第三层的路由器来说，第二层设备是完全透明的。

(3) 网络设备的厂商比较多的问题。算法应当能在异构的网络设备中收集网络拓扑信息，保证在不同厂家的产品中收集的相关数据读取正确而且含义理解正确。解决这一点依赖于厂家规范自己的产品，实现 RFC 对交换机提出的标准。

4 拓扑发现算法分析

4.1 简单网络分析

假定我们网络满足：(1)每个交换区域中只包含一个子网；(2)其中不存在虚拟网(VLAN)；(3)地址转发表是完整的，也就是说，地址转发表中包含了在这个子网内该交换机可达的所有 MAC 地址的信息。

首先，主要描述如何发现交换机和路由器集合，然后描述网络的拓扑发现算法.其中，R1 是我们已知的路由器的 IP 地址：

```
routerSet ={ R1 }
  routerVisited = φ
  while routerSet ≠ φ  do {
    从 routerSet 中选择一个路由器 R
    routerSet = routerSet {R}
    if {R} ∈ routerVisited
      continue
    routerVisited = routerVisited ∪{R}
    neighbor {R} = R 的下一跳路由器
    routerSet = routerSet ∪ neighbor{R}
}
```

网络中的 IP 地址是否为交换机，则要通过检验在该 IP 地址上是否存在网桥 MIB，但是路由器和交换机都存在网桥 MIB,因此还有必要使用 ipForwarding 变量作最后的决定。如果 ipForwarding 的值是 1，那么这就是一个路由器，否则就是一台交换机。接下来要确定路由器和交换机之间的连接情况。在介绍具体算法之前，首先来讨论交换机链接的情况，下面的定理提供了交换机互连的充要条件。

定理 1(直连定理).用 N 表示子网内所有交换机和路由器的 MAC 地址集合,假设 F_i^x 和 F_j^y 都是完备的，那么交换机 S_i 和 S_j 是通过 S_i^x 和 S_j^y 直接连接的，当且仅当 $F_i^x \cap F_j^y = \phi$ 和 $F_i^x \cup F_j^y = N$ 同时成立。

证明：

“$\Rightarrow$”：如果交换机 S_i 和 S_j 是直连，根据定理，再考虑到交换机都采用生成树的协议来避免环路重复连接，所以 $F_i^x \cap F_j^y = \phi$。由于这两个交换机可以将消息送到所在网络内的所有节点，所以也有 $F_i^x \cup F_j^y = N$。

“$\Leftarrow$”：已知界 $F_i^x \cap F_j^y = \phi$, $F_i^x \cup F_j^y = N$，我们反过来假设 S_i 和 S_j 不是直接相连的。那么在生成树之中，在 S_i 和 S_j 之间一定会有一条路径 P 存在。对于这条路径 P 有如下三种情况:

(1) 路径 P 中同时包括 S_i 和 S_j 两点，由于 S_i 和 S_j 不是直接相连的，因此在 S_i 和 S_j 之间一定存在第三点 S_k。但我们已知 $F_i^x \cap F_j^y = \phi$，因此这种情况不可能存在。

(2) 路径 P 中只含有 S_i 和 S_j 其中的一个交换机，和情况 A 相同，也会和条件 $F_i^x \cap F_j^y = \phi$ 产生矛盾，

因此不成立。

(3) 路径 P 中 S_i 和 S_j 两个都不包括，这种情况下显然于 $F_i^x \bigcup F_j^y$ =N 的条件对立，因此 $F_i^x \bigcup F_j^y$ =N 的集合不同时包含 S_i 和 S_j。

定理 1 为发现交换机之间的互连提供了基础。但是还需要考虑路由器和交换机之间的互连情况。下面来介绍路由器和交换机互连的条件。

首先介绍一个定义：叶子接口，交换机 S_i 的叶子接口是和其它任意交换机都不相连的接口。显然，对于一个交换机接口 S_i^x，不存在另一个叶子接口 S_j^y 使得 F_i^x 和 F_j^y 满足定理 1 的条件。

定理 2. 一个路由器 R 和一个交换机的接口 S_i^x 互连，当且仅当 S_i^x 是一个叶子接口，并且 F_i^x 中包含 R 的地址。

证明：

"$\Rightarrow$"：如果交换机的接口 S_i^x 和一个路由器相连，那么这个接口一定不能再与另一个交换机的接口连接，因此端口 S_i^x 一定是一个叶子节点。同时根据 F_i^x 的定义，有 F_i^x 包含路由器 R 的地址。

"$\Leftarrow$"：已知这个交换机端口 S_i^x 是一个叶子接口，那么这个端口的连接情况就只有下面两种：

(1) 和所有的其它设备(包括路由器)都不相连。

(2) 和某个路由器 R 相连。

假如这个端口是情况 1，那么它的 F_i^x 因该是 ϕ，但是我们又已知 F_i^x 至少有路由器 R 的地址这样的一个元素，所以出现矛盾，因此不可能是 A 种情况。

那么这个端口一定是和一个路由器相连，而在 F_i^x 中包含 R 的地址，因此知道是和路由器 R 相连。

下面算法的功能是找出交换机和交换机或者交换机和路由器的互连情况。其中 $S_1, S_2, \cdots\cdots, S_n$ 代表子网 S 中的交换机，$R_1, R_2, \cdots\cdots, R_n$ 是子网 S 中的路由器。

```
for each S_i do
   for each S_i^x do
      if S_i^x 已经知道和其它设备连接
         continue
      else{
         if(F_i^x ⋃ F_j^y =N and F_i^x ⋂ F_j^y = φ)
               记录 S_i^x 和 S_j^y 互连
}
for each R_k do
   for each S_i do
      for each S_i^x do
         if(S_i^x 和其它接口没有互连 and R_k ∈ F_i^x)
               记录 S_i^x 和 R_k 互连
```

4.2 补充定理

可以看到前面的分析区域中只包含一个子网的情况，交换机的拓扑发现是很简单的。前面的定理已经给出了充要条件。但是在更多的情况里，这样的假设是不能满足的，因此我们有必要对这个算法进行扩充使它适应更多情况。对于包含多个子网的情况，我们的拓扑发现方法是消去不能连接的接口。在下面的定理中，我们会研究存在两个接口无法相连的条件。这个定理利用了包含多个子网的交换域的性质：

引理 1. 假设 S_i 和 S_j 是处于两个不同子网的交换机，那么 F_i^x 包含 S_j 当且仅当存在另一个和 S_j 处在同一个子网的交换机 S_k，使得 S_k，…，S_i，…，S_j 是生成树中的一条路径。

定理 3. S_i^x 和 S_j^y 是存在于两个不同子网里的交换机的接口，如果 $F_i^x \bigcap F_j^y = \phi$，那么接口 S_i^x 和 S_j^y 就不能相连。

证明：不失一般性，存在一个，$S_k \in F_i^x \bigcap F_j^y$ 并且反过来假设接口 S_i^x 和 S_j^y 是互相连接的。那么从 S_k 经过 S_j 到 S_i，和从 S_k 经过 S_i 到 S_k 都有各自的一条通路。既然两条通路都是生成树中合理的路径，那么这两条路径实际上构成了一条环路，这在在生成树中是不允许的。定理得证。

定理 4. 设 t 是一个子网，该子网内至少包含两个交换机 S_k 和 S_i。如果 $F_i^x \bigcap F_j^y = \phi$，并且 $F_i^x \bigcup F_j^y$ 中包含 S_k 和 S_i 中的任意一个，但没有同时包含这两个交换机，那么 S_i^x 和 S_j^y 不能相连。

证明：还是反着假设 S_i^x 和 S_j^y 是相连的。不失一般性，可以假设 $S_k \in F_i^x$，因此 S_1，…，S_i，…，S_k 是生成树中的一条路径，这时会有两种情况：

(1) 从 S_1 到 S_i 的路径中不包括 S_j，这是的 S_1 应该是属于 F_j^y 的，因此在从 S_j 到 S_k 的路径中会通过 S_1 到 S_i，而我们已知 S_k 和 S_1 是属于同一子网的。

(2) 从 S_1 到 S_i 的路径中包括 S_j。这时，由于 $S_k \in F_i^x$，一定存在属于子网 t 的一个交换机 S_m，使得 S_k，…，S_k, S_i，…，S_m 为一条合理路径。同样，$S_1, \cdots, S_j, S_1, \cdots, S_m$ 也应该是一条合理的路径，因此 $S_1 \in F_i^x$ 成立。因此如果 S_i^x 和 S_j^y 是相连的，那么我们可以得出 S_k 和 S_1，一定是同时属于 $F_i^x \bigcup F_j^y$ 的。这与已知结果矛盾，因此 S_i^x 和 S_j^y 是不相连的。

定理 5. 已知条件 $F_i^x \bigcap F_j^y = \phi$ 和 $F_i^x \bigcap F_k^z = \phi$。如果 $F_i^x \bigcup F_j^y = F_i^x \bigcup F_k^z$，并且 S_i 和 S_j 属于同一个子网，S_k 单独属于另一个子网，那么 S_i 和 S_j 不能相连。

证明：假设 S_i 和 S_j 是相连的，由于 $F_i^x \bigcap F_j^y = \phi$

和 $F_i^x \cap F_k^z = \phi$，$F_i^x \cup F_j^y = F_i^x \cup F_k^z$，因此 $F_j^y = F_k^z$。又因为 S_i 和 S_j 属于同一子网，所以 $S_i \in F_i^x$，因此 $S_i \in F_k^z$。因此一定存在一个交换机 S_m 与 S_i 和 S_j 处于同一子网，使得 S_i，S_j，…，S_k，…，S_m 是生成树中的一条路径。但是由于 S_i 和 S_j 属于同一子网，因此 $S_j \in F_k^z$，所以有 $S_j \in F_j^y$，因此矛盾。

通过上面三个定理，可以消去一些交换机不可能的连接情况，因而在拓扑发现的时候就可以减小运算量，仅当在对三个定理无法断定的时候来检验连接情况。

从上面三个定理，我们知道对任意的两个互连的交换机接口 S_i^x 和 S_j^y，以下的情况一定成立。

(1) $F_i^x \cap F_j^y = \phi$

(2) 对于任意的子网 t，$F_i^x \cup F_j^y$ 包含子网 t 内的所有交换机，或者 $F_i^x \cup F_j^y$ 不包括子网 t 内的任意一个交换机。

(3) 如果 S_i^x 和 S_j^y 属于同一个子网，一定不存在子网外的某个交换机 S_k，使得 $F_i^x \cup F_j^y = F_i^x \cup F_k^z$，$F_i^x \cap F_k^z = \phi$。

对于满足上述条件的可能连接的接口 S_i^x 和 S_j^y，我们称 $F_i^x \cup F_j^y$ 为一个有效转发地址集合。对于一个有效转发地址集合 $F_i^x \cup F_j^y$，如果 S_j^y 不属于其它任何有效转发地址集合，那么可以断定 S_i^x 和 S_j^y 相连的，同时，因为 S_i^x 只可能有一条连接，因此也可以消去其它包含 S,的有效转发地址集合.而对于两个交换机，因为它们之间只可能有一条直接连接，因此如果我们确定 S_i^x 和 S_j^y 是相连的，那么所有其它同时包含 S_i^x 和 S_j^y 的有效转发地址集合就应该被放弃，不予考虑。

5 拓扑算法的实现

算法的实现中,需要采用 3 个队列,一个是待检测队列,初始情况下队列中的成员为当前生成树中所有的结点。另一个是已检测队列，队列中的成员为已经知道自己的连接关系的交换机结点。第 3 个队列是中间的叶交换机队列，它的成员是指已经知道自己是叶交换机,但是还没有进行判断的交换机结点。

算法实现步骤如下:

(1)获取子网内所有的交换机的列表;

(2)ping 子网内所有交换机;

(3)依次读取每台交换机的地址转发表;

(4)从各交换机地址转发表中构造每台交换机的上行端口和下行端口集合,同时将子网内所有交换机结点放入待检测队列;

(5)从待检测队列中取出一个交换机结点,判断是否是叶交换机.如果是叶交换机,则放入叶交换机队列中.如果不是叶交换机,则到(6);

(6)检查交换机的每个接口的地址转发表,如果地址转发表中含有的元素都是叶交换机的话,可以判断出此接口与所含地址表示的交换机的上行端口相连.如果地址转发表中含有的不只是叶交换机的 MAC 地址,将这些叶交换机的MAC地址删除.如果待检测队列中还有交换机没有检查,转到(5);

(7)遍历叶交换机队列中的交换机,如果一个交换机已经确定了自己的上行端口的连接端口,将此交换机移入已检测队列;

(8)检测待生成队列,如果不为空,转到(5)。

实现过程中,每个设备使用其 IP 地址唯一标识,并保存了设备的所有 IP 地址用于同一性的判定,对于发现的,没有反应的设备使用 IP 地址形式为 “0. x. x. x” 来进行标识。算法在实验室的局域网中得到验证，实验结果表明,该算法能够完全准确的发现网络中所有设备、主机以及它们之间的连接关系。

6 小结

现在，随着对 IP 网络管理要求的提高，物理网络拓扑的重要性日趋明显。早期关于网络的研究的注意力主要集中在逻辑网络的拓扑上，无法捕捉到网络二层中的信息。该文利用 SNMP 协议，结合 MIB-II 及其扩展 Bridge-MIB 中的相关数据表信息，分析并改进了物理网络拓扑发现的算法。这样的算法所构造的网络拓扑能够较真实地反映网络拓扑情况，效果比较理想。

参考文献

1 Bejerano Y, Breitbart Y, Garofalakis M, Rastogi R. Physical topology discovery for large multi-subnet networks. Proc. IEEE INFOCOM, 2003:342－352.

2 Bejerano Y, Garofalakis M, Jai B, Rastogi R, Topology Discovery in Heterogeneous IP Networks: The NetInventory System. IEEE/ACM Trans. Networking, 2004,12(4).

3 Case J, Fedor M, Schoffstall M, et al.RFC1157-1999,Simple Network Management Protocol (SNMP).

4 郑海,张国清.物理网络拓扑发现算法的研究.计算机研究与发展, 2002,39(3):264－268.