文章编号:1001 - 9081(2005)11 - 2486 - 03

分布式网络性能监测代理自适应部署研究

曹献栋,夏德麟,晏蒲柳
(武汉大学 电子信息学院,湖北 武汉 430072)
(cxdtiger@ vip. sina. com)

摘 要:分布式网络性能监测的关键问题是如何在管理域中的适当位置放置适当数目的代理,才能达到提高监测效率,减小额外监测流量和响应时间的目的。本文提出的算法是在管理员改变网络拓扑结构或者网络节点或链路失效时被启动,能够针对网络的变化自动地进行局部监测代理分布的调整,使监测系统恢复到近似最优状态。该算法使本监测系统具有良好的可扩展性和适应能力,仿真实验证明了算法的可行性。

关键词:性能监测;代理;自适应部署
中图分类号: TP393.07 **文献标识码**:A

Self-adaption deployment of performance monitoring agent in distributed networks

CAO Xian-dong, XIA De-lin, YAN Pu-liu
(*College of Electronic Information, Wuhan University, Wuhan Hubei* 430072, *China*)

Abstract: One of the critical problems in distributed network performance monitoring is that how to place proper number of monitoring agents in proper locations in a managed network can improve monitoring efficiency, that is, decrease monitoring traffic and response time. Our algorithm was triggered when network topology changed, or nodes and links failures occurred. By adjusting some agents' deployment, it could draw the whole monitoring system back to proximate optimization state. Simulation shows that this algorithm can add flexibility and scalability to the proposed monitoring system.

Key words: performance monitoring; agent; self-adaption deployment

0 引言

由于网络日益大规模化和动态化,网络性能监测系统也应具备适应其动态特性的能力。网络状态的变化可以分为两种:一是网络拓扑结构不发生变化,只是某些链路的状态,如链路费用发生变化。二是网络拓扑结构的改变。发生此类变化时,该被管网络的路由器将会进行路由重算。当路由算法稳定后,管理域内路由器的路由表将发生相应的改变,某些路由表项出现增加、删减或者修改的情况。本文将拓扑变化概括为三种情况:1)拓扑扩展,即在被管网络中增加一个或多个节点; 2)拓扑删减,某些原本存在于被管网络中的节点被人为地从该管理域中删除,或者由于节点或链路失效造成这些节点不可达; 3)节点位置更改,某些原本就存在于管理域中的节点被人为地从原来的位置移动到新的位置,或者由于链路失效导致节点位置发生变化。

以前的研究者们对基于 Agent 的监测系统的自适应性进行了初步探讨,文献[1,2]采用周期性重新部署监测代理的方法来实现监测系统的自适应,这是最简单的一种方法。但是重新部署所有代理的开销非常大,对于大型的网络系统是不现实的。而且这种方案对于只存在本地或者局部网络变化的情况显得效率低下,因为经过重新计算以后,大多数的代理很可能仍被部署在与变化发生以前相同的位置上。文献[3,4]针对网络拓扑结构不发生变化,而链路状态发生变化的情况,采用基于自治策略的代理迁移实现自适应。分布在管理域中的每个监测代理周期性估算通过邻近候选节点进行监测的费用,当候选位置的费用显著减少到比当前位置的费用小时,当前节点上的代理迁移将被触发,代理将移动到这个位置较优的节点上。由于这种自适应策略是只基于本地决策的,因此存在没有考虑到全局最优化的缺点。本文引入了 Agent 间的协作机制,主要针对网络拓扑结构的变化对基于协作策略的代理自适应算法进行了探讨。

1 算法描述

本文提出了一种基于协作的代理自适应算法。当网络发生局部变化时,首先找到发生拓扑变化所在的区域,并由之确定区域所属的 MAs(Multi-Agent system)集合,通过 MA(Monitoring Agent)之间的协商和 MAs 与 MS(Management Station)之间的通信,找到该集合位于 MADT(MA Distributed Tree)中同一子树最上层的 MAs 的子集,这样就确定了受拓扑变化影响的 MAs 的最大范围。在该子集中的 MAs 上启动部署算法,在以这些 MAs 为根的子树上进行代理的重新部署。这样就能将 MAs 的调整和重新部署限制在局部,而网络中没有发生变化的区域中驻留的 MAs 则不受影响。

引理 给定一个加权有向图 $G = (V,E)$,权重函数为 $w: E \rightarrow R$,若 $p = < v_1, v_2, \cdots, v_K >$ 为顶点 v_1 到 v_K 的最短路径,对任意的 i 和 j,满足 $1 \leqslant i \leqslant j \leqslant K$,令 $p_{ij} = < v_i, v_{i+1}, \cdots, v_j >$ 为从顶点 v_i 到 v_j 的一条 p 的子路径,那么 p_{ij} 是从 v_i 到 v_j 的最短路径。

收稿日期:2005 - 05 - 12;**修订日期**:2005 - 07 - 13 **基金项目**:国家自然科学基金资助项目(90204008)
作者简介:曹献栋(1974 -),男,湖北武汉人,硕士研究生,主要研究方向:高速信息网络; 夏德麟(1940 -),男,湖北武汉人,教授,主要研究方向:数据挖掘; 晏蒲柳(1962 -),女,湖北武汉人,教授,博士生导师,主要研究方向:高速信息网络.

1.1　拓扑扩展的自适应算法

为了扩大网络规模,由网络管理员人为向被管网络中加入了一些新的节点 $\{v_1, v_2, \cdots, v_L\}$,此时被管网络将自动根据路由算法启动路由重算过程。当路由重算结束后,管理员通过管理界面在管理站上完成了节点的添加,此时代理自适应算法被启动。MS 发送 SNMP 消息与这些节点通信,遍历每个新增节点的路由表,找到每个新增节点到达 MS 的下一跳节点,由引理可知,该下一跳节点同时也是新增节点到达最近 MA 的下一跳节点。假设下一跳节点集合为 $U = \{u_1, u_2, \cdots, u_L\}$,则查找 MS 上保存的原来的每个 MA 的监测对象序列(Monitored Objects, MOs),找到每个 $u_i \in U$ 所属的 MA,假设找到的 MA 集合为 $S_0 = \{MA_1, MA_2, \cdots, MA_{L'}\}$,则判断需要对这些 MA 进行重新部署。为了避免在同一棵子树上重复运行代理部署,令 S_0 集合中的每个 MA 与代理协调器通信,由代理协调器根据全局 MADT 信息以共享信息的方式进行协商,找出该集合中位于每棵子树中最高层的 MA(即 level 属性值最小的 MA)集合,假定为 $S_1 = \{MA_1, MA_2, \cdots, MA_{L''}\}$($L'' \leqslant L'$),最后分别在每个 $MA_i \in S_1$ 上运行代理部署算法,在 MADT 中以 MA_i 为根的子树上重新部署代理。

算法伪码描述如下:

```
v1, v2, v3, ...vL : 增加的节点;
Add_nodes( v1, v2, ..., vL )
{
  For each vj in { v1, v2, ..., vL }
    MS ask vj for next_top uj from vj to MS
    MS find MAi of uj
    Send MAi to PS
  PS coordinates all reveived MA
  PS output a top level MA list { MA1, MA2, ..., MAl' }
  For each MAt in { MA1, MA2, ..., MAl' }
    HAD( MAt )
}
```

1.2　拓扑删减的自适应算法

拓扑删减可能由两种情况造成:1)网络管理员人为地从被管理网络中去掉或关闭某些节点;2)网络中某些节点或链路出现故障而失效。

第一种情况下,如果被删除的节点集合为 $\{v_1, v_2, \cdots, v_D\}$,MS 将查询本地保存的每个 MA 的 MOs 信息,分别找到这些节点在被删除前是由哪些 MA 负责监测的,并通知这些 MA 将相应 MO(Monitored Object)从各自的 MOs 中删除。假设找到的 MA 集合为 $S_2 = \{MA_1, MA_2, \cdots, MA_{D'}\}$,满足 $\{v_1, v_2, \cdots, v_D\} \in \bigcup_{i=1}^{D'} MOs_i$,则将 S_2 集合告知代理协调器。代理协调器通过 MADT 的历史信息,找到集合 S_2 中位于 MADT 的每棵子树最上层的 MA 子集,假设为 $S_3 = \{MA_1, MA_2, \cdots, MA_{D''}\} \subseteq S_2 (D'' \leqslant D')$,将每个以 $MA_i \in S_3$ 为根的子树中所有下层 MA 的 MOs 添加到 MOs_i 集合中,并停止这些子 MA。若 $MOs_i = \varnothing$,则将 MA_i 停止;若 $MOs_i \neq \varnothing$,则将其中所有 MO 作为输入参数,运行拓扑扩展自适应算法,进行 MA 的局部重新部署。

在第二种情况下,每个 MA 周期性地对本地路由器的路由表进行查询,当发现自己的 MOs 中以某些 MO 为目的的表项丢失,则判断这些 MO 失效,并向 MS 报告。当 MS 收到所有的故障节点的信息后(假设故障节点集合为 $S_4 = \{MO_1, MO_2, \cdots, MO_f\}$),以 S_3 为输入参数,运行与第一种情况相同的算法。

算法伪码描述如下:

情况一

```
设 v1, v2, ..., vD : 被删除的节点
Delete_manul( v1, v2, ..., vD )
{
  For each vj in { v1, v2, ..., vD }
    ask MS for vj's MAi
    MAi delete vj from its MO_list
    Send MAi to PS
  PS coordinates all reveived MAs
  PS output a top level MA list { MA1, MA2, ..., MAD' }
  For each MAt in { MA1, MA2, ..., MAD' }
    For each childagent of MAt
      Add MO_list of the childagent to MO_list of MAt
    If MO_list of MAt is empty then
      Kill MAt
    else
      Add MO_list of MAt to MOs
  Add_nodes( MOs )
}
```

情况二

```
Delete_auto()
{
  For each MAi in network
    MAi periodically inquire local route table
    Report lost MO to MS
  MS receive all lost MOs
  Delete_manul( MOs )
}
```

1.3　节点位置更改的自适应算法

节点位置的更改可能由两种情况造成:1)网络管理员人为地移动被管理网络中的某些节点;2)网络中出现链路故障导致节点位置发生变化。

第一种情况下,假设管理员指定被移动的节点集合为 $\{v_1, v_2, \cdots, v_M\}$,与节点删除自适应算法类似,MS 将查询本地保存的每个 MA 的 MOs 信息,分别找到这些节点在改变前是由哪些 MA 负责监测的,假设找到的 MA 集合为 $S_5 = \{MA_1, MA_2, \cdots, MA_{M'}\}$,满足 $\{v_1, v_2, \cdots, v_M\} \in \bigcup_{i=1}^{M'} MOs_i$,则将 S_5 集合告知代理协调器。代理协调器通过 MADT 的历史信息,找到集合 S_5 中位于 MADT 的每棵子树最上层的 MA 子集,假设为 $S_6 = \{MA_1, MA_2, \cdots, MA_{M''}\} \subseteq S_5 (M'' \leqslant M')$,将每个以 $MA_i \in S_6$ 为根的子树中的所有 MA 的 MOs 添加到一个待分配节点集合 S_7 中,以 S_7 中所有 MO 作为输入参数,执行拓扑扩展自适应算法类似的过程,进行 MA 的局部重新部署。

第二种情况下,当某个监测代理 MA_i 由本地路由表中发现,到达某些 MO 的下一跳节点改变了或者路由费用发生了明显变化,则向 MS 汇报这些变化。当 MS 收到所有的 MA 发送的节点变化信息后(假设发生变化的节点集合为 $S_8 = MO_1, MO_2, \cdots, MO_M\}$),以 S_8 为输入参数,运行与第一种情况相同的算法。

情况一

```
Change_manul( v1, v2, ..., vM )
{
  For each vj in { v1, v2, ..., vM }
    ask MS for vj's MAi
    Send MAi to PS
  PS coordinates all reveived MAs
  PS output a top level MA list { MA1, MA2, ..., MAM' }
  For each MAt in { MA1, MA2, ..., MAM' }
```

```
    For each childagent of MA_t
      Add MO_list of childagent to MO_list of MA_t
      Add MO_list of MA_t to MO_s
  Add_nodes( MO_s )
}
```

情况二

```
Change_auto()
{
  For each MA_i in network
    For each MO_j in MO_list of MA_i
      If MO's next_hop changed or cost changed greatly from MA_i to
        MO_j then
        Report MO_j to MS
  MS receive all MO_s changed
  Change_manul( MO_s )
}
```

2 算法实例分析

以图 1 所示的网络拓扑结构中链路(11,14)失效的情况为例,对代理自适应算法的计算过程加以说明。假设路由重算后,重新形成了如图 1(a)的路由树,系统中的 MAs 将自动发现链路失效的变化,并找到变化的区域,再对这些区域进行 MA 的重新部署。

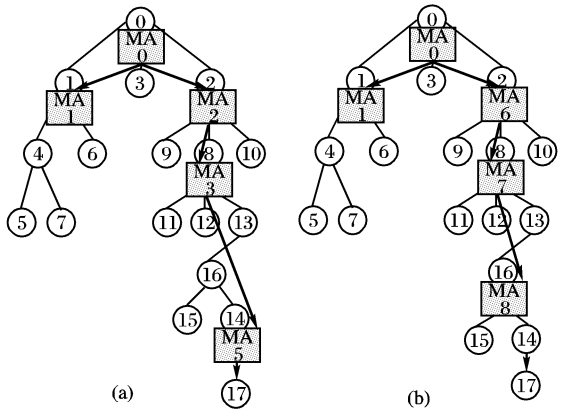

图 1 代理自适应过程示意图

本监测系统中的 MAs 周期性地访问本地路由器的路由表,监视路由表中它所管辖 MOs 的表项。发生链路失效后,MA_3 发现到达 14 号节点的路由表项改变了,其下一跳节点由原来的 11 号节点改变为 13 号节点,同时 MA_5 也发现到达 15 号节点的路由表项发生了变化,由原来的直接相连变为需要通过 16 号节点。此时,MA_3 将 14 号节点变化的信息告知 MS,MA_5 也将 15 号节点变化的信息向 MS 汇报。于是 MS 可确定发生变化的区域为 MA_3 和 MA_5 所管辖的区域,并将该信息传送给代理协调器。代理协调器根据 MADT 的历史信息,作出 MA_3 和 MA_5 位于同一棵子树,且 MA_3 的层次比 MA_5 高的判断。根据 MS 上保存的每个 MA 的 MOs 信息,代理协调器将所有 MADT 中位于以 MA_3 为根的子树中所有 MA 的 MOs 合并,此处为 MA_3 和 MA_5 的 MOs,得到合并后的 MO 集合 $S_{MO}=\{11-17\}$。之后对 S 中的 MO 进行重新划分。MS 首先访问 S 中每个节点的路由表,找到到达 MS 的下一跳节点集合 $S_{Next\_hop}=\{8,16,13,17\}$,并根据历史 MOs 信息,找到这些节点在链路失效前所归属的 MA 集合 $S_{MA}=\{MA_2, MA_3, MA_5\}$。同样,经过代理协调器的协商,发现 S_{MA} 中位于 MADT 最上层的 MA 为 MA_2,于是确定链路失效所影响的最上层监测代理为 MA_2。最后在 MA_2 所在节点重新运行代理部署算法,对 MADT 中位于以 MA_2 为根的子树进行代理的重新分配,产生新的监测代理 MA_6,MA_7 和 MA_8。得到的新的代理分布图如图 1(b)所示。这个自适应计算过程,只对受到链路失效影响的部分代理 MA_2,MA_3,MA_5 进行了重新部署,而没有受到影响的代理 MA_1 仍可继续正常运行。

3 算法性能分析

我们用仿真模拟了两条链路失效的情况。仿真参数的配置参见表 1。我们测量了本监测系统中出现该异常前后的平均稳态监测流量和响应时间与 MAs/MOs 的变化关系。

表 1 主要的仿真参数设置表

参数	取值
MO 数目	20 ~ 80
平均网络直径	7 ~ 8.5
MAs/MOs	0 ~ 1
轮询率	0.2 ~ 6/s
平均节点度数	2.6 - 11.7
跳步直径	6.3 ~ 10
平均跳数	5.07 ~ 7.92
轮询请求包大小	100bytes
轮询应答包大小	250bytes
代理通知包大小	50bytes
MA 大小	1000bytes
MA 序列化时间	100ms
MA 解序列化时间	200ms
MA 克隆时间	100ms

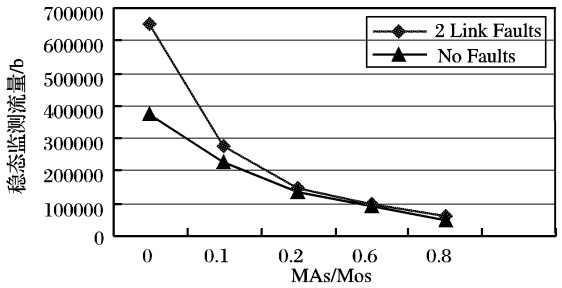

图 2 故障前后的稳态监测流量示意图

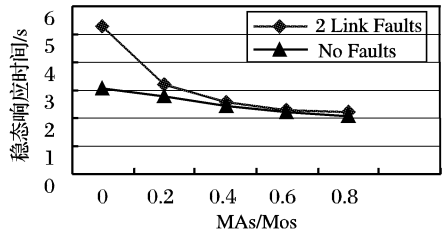

图 3 故障前后的稳态响应时间示意图

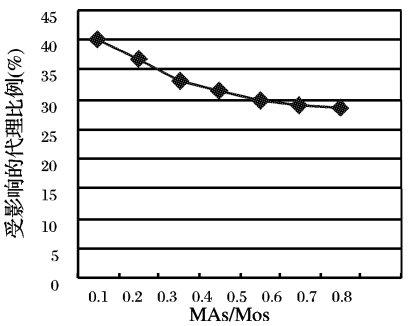

图 4 受影响代理的比例示意图

(下转第 2491 页)

}
}

2）解决别名归一,并删除子网广播地址

由于 Router_List < Ci, Pi, sysObjectID, ipAdEntAddr > 中可能存在代表同一个路由器几个 IP 地址,也可能存在响应 SNMP 报文的子网广播地址,因此在进行下一步工作之前需要对 Router_List < Ci, Pi > 进行处理。

首先根据网络拓扑结构分析规则 7 的判定依据将 Router_List < Ci, Pi, sysObjectID, ipAdEntAddr > 中子网广播地址过滤掉。

```
While( Router_List < Ci, Pi, sysObjectID,  ipAdEntAddr > 非空)
{
  取出一个 < Ci, Pi, sysObjectID, ipAdEntAddr >
  如果 Pi ∉ ipAdEntAddr
  则根据规则 7 可断定 Pi 是一个子网广播地址, 删除 < Ci, Pi,
  sysObjectID,  ipAdEntAddr >
}
```

根据规则一对 Router_List < Ci, Pi, sysObjectID, ipAdEntAddr > 进行别名归一。将 sysObjectID 相同的 < Ci, Pi, sysObjectID, ipAdEntAddr > 的项,只保留任意一个,其他的删除。

3）各管理域内的拓扑发现

```
While( Router_List < Ci, Pi, sysObjectID, ipAdEntAddr > 非空)
{
  取出一个 Router_List < Ci, Pi, sysObjectID, ipAdEntAddr > 进行
  广度优先搜索
  ① 获取每个搜索种子对应的∑NextHop, ∑Subnet, ∑RouteType
  和∑Mask
  ② 根据规则 4 的判定依据获取∑NextHop-Router
  ③ 根据规则 5 的判定依据∑ipRouteDest-Router
}
```

将∑NextHop-Router 和∑ipRouteDest-Router 合并成∑Router-Connection

4）管理域间的拓扑发现

直接运用规则 6,来判断管理域间的互联关系。

```
While( Router_List < Ci, Pi, sysObjectID, ipAdEntAddr > 非空)
{
  如果 Pi ∉ Pj 对应的∑NextHop, 而且 Pi ∉ Pj 对应的
  ipAdEntAddr, 则将 Pi 加入到 Pj 的∑Router-Connection 中
}
```

2.4　应用及测试效果

目前已经对该算法在远程科研教育网骨干网络环境中进行了测试,其具体的测试效果如图 2 所示。

图 2 是对科研教育网华东某省主干网络进行拓扑发现得到的网络拓扑图形。在具体实现的过程中,比较耗时的功能模块都采用了多线程设计,这样大大缩短了整个网络拓扑发现时间。用实际的网络拓扑图形与该图形进行拓扑比对发现,本来处于不同管理域的几个独立结构,已成为一个完整的网络拓扑结构,和真实网络拓扑结构基本上完全吻合。

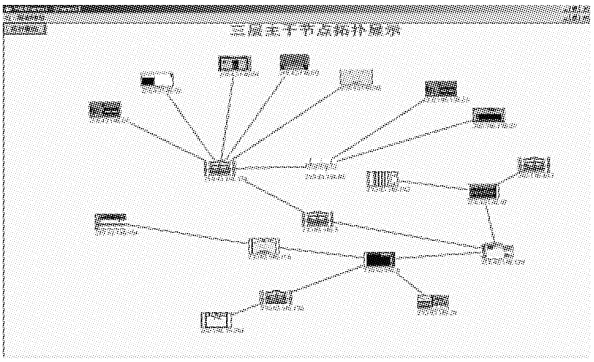

图 2　实际测试生成的网络拓扑结构图形

跨管理域的网络拓扑发现,使网络拓扑发现工作不再被限制在一个管理域内,得到完整的网络拓扑结构,达到了网络拓扑发现的三个指标:准确性、完整性和高效性[6]。

参考文献:

[1] DUAN Y, HUANG D. Networked System Management: Topology Discovery and Host Monitoring and Control[DB/OL]. http://eva-soft.com/millennium, 2004-11-15.

[2] LIN HC. An Algorithm for Automatic Topology Discovery of IP Networks[M]. IEEE, 1999.

[3] MANSFIELD G, OUCHI M, OHTA K. Techniques for automated network map gengrationusing SNMP[EB/OL]. http://www.cysols.com/contrib./pdf/InfoCom96.pdf, 2002-09-15.

[4] BREITBART Y, GAROFALAKIS M, MRTIN C, *et al*. Topology Discovery in Heterogeneous IP Networks[M]. IEEE, 2000.

[5] 孙德文,高儒振. 基于 SNMP 网络拓扑图自动构造实现[J]. 上海交通大学学报,1997:31(8):97-101.

[6] 施锋,吴秋峰. 网络多层拓扑发现算法分析[J]. 兵工自动化,2004,23(3):30-33.

(上接第 2488 页)

由图 2 和 3 可以看出,当发生链路失效时,采用集中式的轮询方案(当 MAs/MOs = 0 时),由于监测请求包和应答包均\会沿着更长的路径重新路由,因此造成监测流量和响应时间的显著增加,而运行了代理自适应算法的监测系统则通过部分代理的重新部署使失效的链路不会对整个监测系统的性能造成显著下降。本自适应算法的稳态监测流量与响应时间几乎是集中式算法的一半。

另外,部署时代理的数目的设计也是影响算法性能的一个重要因素。代理数目越多,虽然存在部署时由代理克隆和迁移所造成的额外开销较大的问题,但程度更高的分布化能够使监测系统得到更好的稳态监测性能,稳态监测流量和响应时间都能够相应减小。仿真试验表明随着代理数目的增加,出现相同网络异常时受到影响的代理数目呈非线性减少,这就意味着监测系统的稳定性越高。仿真结果如图 4 所示。

参考文献:

[1] ABDU H, LUTFIYYA H, BAUER M, *et al*. Adaptive Monitoring Configurations[A]. Proceedings of IEEEIM'99[C], 1999.371-384.

[2] LI H, YANG S, XI H. System Designs for Adaptive, Distributed Network Monitoring and Control[A]. Proceedings of IFIP/IEEE MMNS 2001[C], 2001.77-90.

[3] GAVALAS D, GREENWOOD D, GHANBARI M, *et al*. Implementing a Highly Scalable and Adaptive Agent-based Managemet Framework[A]. Proceedings of the IEEE Global Communications Conference[C]. San Francisco, USA, 2000, 3.1458-1462.

[4] TRIPATHI A, AHMED T, PATHAK S. Paradigms for Mobile Agent Based Active Monitoring of Network Systems, Network Operations and Management Symposium[A]. Proceedings of. IEEE/IFIP International Network Operations and Management Symposium (NOMS2002)[C], 2002.65-78.