

# 分布式网络故障监测的探针部署方法<sup>①</sup>

郭振兴, 黎文伟

(湖南大学 信息科学与工程学院, 长沙 410082)

**摘要:** 在基于主动探针的网络故障管理中, 首要的是确定监测探针的部署。不确定性会对监测探针的选取带来影响。为了降低这种影响, 采用概率依赖模型为主动探针选取路由, 提出了不确定环境下的监测探针部署方案。利用该方案在网络中部署探针, 对网络进行主动探测, 能以较小的代价尽早定位网络中的故障。实验表明本文的探针选择算法可以有效节约探针开销, 且部署的探针集具有较好的故障定位能力。

**关键词:** 主动探针; 网络监测; 故障定位; 探针部署

## Probe Deployment of Distributed Network Fault Monitor

GUO Zhen-Xing, LI Wen-Wei

(School of Information Science and Engineering, Hunan University, Changsha 410082, China)

**Abstract:** In the network fault management based on active probe, the first job is to determine the deployment of monitoring probe. Non-deterministic environment will impact the monitoring probe selection. In order to reduce the influence, a probe station deployment approach is proposed with select routing for active probe based on probability dependence model. The faults in the network can be localized sooner and cost-effectively using the active probes sent from the probe stations placed in the network. The simulation indicates that the algorithm has strong ability of fault location and cost-effectively.

**Key words:** active probe; network monitoring; fault localization; probe deployment

网络故障管理作为网络管理系统中重要的组成部分, 是维护网络可靠性、保障网络正常运行的关键。目前解决故障管理问题的方法主要分为被动探测和主动探测两类。采用被动监听故障监测方法的网管系统无法在用户感知到故障前监测和定位故障, 且可能由于目标系统告警信息的延迟或丢失而无法快速准确地定位故障, 无法满足日益复杂的网络需求。基于主动探测的方法具有主动、高效及自适应的特性, 能以很小的代价尽早地获得网络、系统故障、服务失败及性能衰退等症状, 从而为根源故障的分析提供依据。在基于主动探测的故障监测方法中, 首要的是确定监测探针的部署。监测探针的部署直接会对故障定位的能力、探测的有效性以及探针的配置开销造成很大影响, 是网络故障管理中至关重要的环节。

针对监测探针的部署问题, 文献[1,2]假设网络中

同时发生故障的链路个数最多为 1, 在监测探针发送的探测能够覆盖所有链路的前提下, 通过部署最小的探针集合来监测和定位网络中发生的单个链路故障。该方法主要考虑链路故障的情况, 并没有考虑网络中节点出现故障的情况。针对节点故障, 文献[3]提出了一种基于条独立路径的监测探针部署算法, 支持网络中多故障定位。该算法认为, 当网络中的每个节点都具有从监测探针集合发出的条独立路径时, 只要有一条独立路径无故障, 节点都可以被有效监测, 并且监测探针集合发出的探测能对网络中同时发生的个故障进行定位。但在实际网络拓扑结构中, 这种算法的有效性受到大量存在的度数小于的节点的极大制约。上述方法均是在确定环境下<sup>[4]</sup>给出的一种监测探针部署方案, 其探针选择算法基于故障和探针为确定性联系的假设, 且实际中很难精确获取每个探针经过的路由,

① 基金项目: 国家自然科学基金(60703097); 国家 973 计划(2007CB310702); 东莞市高等院校科技计划(201010815603)

收稿时间: 2011-03-22; 收到修改稿时间: 2011-05-12

并不能满足现实网络中的不确定环境的要求。文献[5]使用沿特定路径发送的源路由探针监测路径性能,这种方法虽然避免了动态路由的影响,但是源路由机制需要网络提供额外的支持。本文结合实际网络拓扑结构,通过为每个探针包以概率选路,给出了一种在不确定环境下,支持多故障监测的探针部署方法,利用该方法部署的探针较少,且具有较好的故障定位能力。

### 1 问题阐述

现实中管理网络可以抽象成为一个无向图  $G=(V,E)$ , 其中  $V=\{v_1,v_2,\dots,v_i\}$  表示图中顶点的集合,代表网络节点  $v_i$  的集合,  $E=\{e_1,e_2,\dots,e_i\}$  表示图中边的集合,代表网络中通信链路  $e_i$  的集合。边  $e_k$  是由无序节点对  $(v_i,v_j)$  构成,且  $v_i$  和  $v_j$  称为边  $e_k$  的端点。若  $(v_i,v_j)\in E$  则称  $v_i$  为  $v_j$  的邻接点。节点  $v_i$  的度数用  $D(v_i)$  表示。 $P_{s,t}=(s,v_1,v_2,\dots,v_i,t)$  表示一次探测中 IP 数据包从源节点  $s$  经过  $v_1,v_2,\dots,v_i$  到目标节点  $t$  所经历的路径,  $v_1,v_2,\dots,v_i$  称为路径  $P_{s,t}=(s,v_1,v_2,\dots,v_i,t)$  的中间节点。

基于主动探测的故障监测算法首先在网络中选择一些节点部署为具有发送探测能力的监测探针。然后监测探针  $s$  向网络中目标节点  $t$  沿着探测路径  $p_{s,t}$  发送探针包进行探测,根据探测返回的结果分析定位网络中出现故障的节点。在不考虑网络噪声的情况下认为,当 1 个探测成功时,该探测经过的节点均正常工作。反之,当 1 个探测失败时,该探测经过的所有节点(症状节点)均可能是故障节点。

一次探测失败只能说明探测路径所经过的节点有故障发生,并不能确定网络中有多少个故障以及哪些节点故障。为了准确定位出这些故障节点,在已知网络拓扑的情况下,首要的是找出症状节点集合。在确定环境下,针对特定节点发送的探针包选路固定,症状节点的信息能够较为方便的获取。如图 1 网络拓扑中,节点 1 被部署为监测探针,若要监测节点 2 的故障情况,需发送一个到目标节点 2 的探针包进行探测,若探测成功,说明节点 2 正常工作。若探测失败,因探针包发送路径没有经过其他节点,此时可以直接判定节点 2 是故障节点。

同理,若要监测节点 5 的故障情况,需要向目标节点 5 发送一个经过节点 4 的探针包,若探测成功,说明节点 1、4、5 均无故障。反之,节点 4 和 5 均可能发生故障(至少有一个故障发生,但不能确定是什

么节点)。若要准确定位这些故障节点,必须发送更多的探针包进行探测,但这种方法不一定可行。考虑图 1 中节点 2、4 和 5 均出现故障的情况,虽然节点 2 和 4 均可以被探针站 1 准确定位,但节点 5 的故障却无法被定位。这类问题的解决只能依靠于部署更多的监测探针来增加对网络中故障的定位能力。但过多的探针又会增加部署开销。

一种理想的探针部署方案就是,部署最少的探针且通过这些探针能够准确定位网络中的多个故障。考虑到不确定环境下探针包选路的随机性(如探针对于节点 5 的探测,其探测路径可以经过节点 4 也可以经过节点 2、3 到达 5),因此在不确定性网络环境下,如何部署一系列探针,能够较准确的获取每个探针经过的路由,从而确定症状节点,以进一步准确定位网络中的多个故障,且部署的探针个数最少,就成为探针部署需要解决的问题。

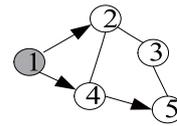


图 1 节点 2 和 5 的故障监测

### 2 求解方法

若要实现对被管理网络中任意  $k$  个故障的准确监测,假设网络中被监测节点最多同时发生  $k$  个故障,只要为每一个被监测节点找取条独立路径<sup>[3]</sup>即可。不确定环境下独立路径的选取是随机的,为克服这一困难,本文通过建立概率依赖模型<sup>[6]</sup>为探针到被监测节点所有可能选路的中间节点指定一个概率值(较短选路上的节点将会赋予更高的概率值),通过相应的计算来求取独立路径。最后通过计算现有探针集到相应被监测节点的监测能力来选取监测探针。

#### 2.1 监测能力

若把被管网络的所有节点都看成潜在的故障节点,为量化探针集对每个节点的监测概率,下面将给出监测能力值  $DQ$  (detection quality)的定义。给定故障集  $V$ , 探针集  $SP$ , 如果当前探针子集为  $SP_n \in SP$ ,  $SP_n$  对任意潜在故障节点的监测能力定义为:当潜在故障节点  $v_i$  发生故障时,  $SP_n$  到该节点存在至少有  $k$  条独立路径的概率。监测能力定义如下:

$$DQ(SP_n, v_i) = 1 - p(\sum_{i=1}^n (P_{SP_n, v_i}) < k)$$

其中,  $p(\sum_{i=1}^n(P_{SP_n}, v_i) < k)$  表示现有探针集到节点  $v_i$  的独立路径个数小于  $k$  的概率,  $\sum_{i=1}^n(P_{SP_n}, v_i)$  表示节点  $v_i$  现有的独立路径个数。

不确定环境下独立路径的求取方法如下:

$V(I(p_1, p_2)) = 1 - \cup_{v \in nodes(p_1) \cap nodes(p_2)} P(p_1, v) \cdot P(p_2, v)$ , 其中  $\cup_{v \in nodes(p_1) \cap nodes(p_2)} P(p_1, v) \cdot P(p_2, v)$  表示路径  $p_1$  和  $p_2$  中间节点有重复的概率,  $V(I(p_1, p_2))$  表示  $p_1$  和  $p_2$  相互独立的概率, 当该值大于指定阈值时, 则认为  $p_1$  和  $p_2$  是相互独立的。

在首个监测探针选取后, 备选探针  $C$  到节点  $v_i$  的独立路径求取如下:

$$V(I(path(SP_n, v_i), path(C, v_i))) = 1 - \cup P(path(SP_n, j)) \cdot P(path(C, j))$$

其中,  $path(SP_n, v_i)$  表示监测探针集合  $SP_n$  到节点  $v_i$  的探测路径,  $j$  表示路径  $path(SP_n, v_i)$  和路径  $path(C, v_i)$  中的重复节点, 即  $j \in nodes(path(SP_n, v_i) \cap path(C, v_i))$ 。

### 2.2 探针选择算法

基于以上分析, 下面给出一个基于贪婪策略的启发式算法来解决监测探针的选择问题, 该算法依据弱连接点求取独立路径, 能够快速找取部署下一个监测探针的最优位置, 并通过指定那些部署为监测探针的节点会发生的故障总数的上限值为  $k_b (k_b < k)$ , 即此类节点需选取的独立路径个数为  $k_b$ , 来减少部署的监测探针个数。

定义 1 (弱连接点) 在独立路径选取过程中, 满足式子  $k - NP(v_i) = 1$  或  $D(v_i) - NP(v_i) = 1$  的节点  $v_i$  称为弱连接点。其中  $NP(v_i)$  表示现有探针集到节点  $v_i$  的独立路径个数。

弱连接点待找独立路径个数较少这一特征可以极大缩小下一个最优监测探针部署位置的范围。此外,  $D(v_i) < k$  的节点无法找取条独立路径, 这类节点在独立路径选取中若满足:

(1)  $D(v_i) = NP(v_i)$ ;

(2) 有  $k_b + 1$  个邻接点是监测探针或者有  $k_b + 1$  个独立路径。则认为节点  $v_i$  满足  $DQ(SP_n, v_i) = 1$ 。

若首个监测探针的部署位置未指定, 则选取度数最高的节点部署为首个监测探针,  $\bar{V}$  表示弱连接点集合,  $num(SP)$  表示现有监测探针个数,  $l$  表示部署探针个数上限。算法的伪代码如下:

initialize  $SP = \phi, \bar{V} = \phi$

input:  $k, k_b, l$

```

while num(SP) < l or V ≠ φ
do
    find  $\bar{V}$ 
    find SP for  $\bar{V}$ 
    update  $\sum_{i=1}^n(P_{SP_n}, v_i)$ 
    if  $DQ(\sum_{i=1}^n(P_{SP_n}, v_i)) = 1$  remove  $v_i$  from V
end while
output SP
    
```

### 3 仿真分析

实验采用自治域(AS)级拓扑产生器 Inet<sup>[7]</sup>生成模拟网络拓扑图。Inet 的特点是利用 route-views.oregon-ix.net 上从 1997 年 11 月到 2002 年 2 月间的大量 BGP 数据来确定拓扑度量的值, 可信性很高, 能较准确的模拟真实网络的情况, 是网络仿真试验中常用的拓扑生成工具。为模拟不确定环境下的探针包选路, 实验通过建立概率依赖模型为监测探针到被探测节点间所有可能经过的节点指定一个概率值, 该值可以通过专家分配或者根据历史统计值评估<sup>[8]</sup>。通过 Inet 生成点规模从 40~200 的 5 组数据, 使用 C 语言对本文算法进行仿真实现, 并将该算法与算法, 在算法执行时间和探针部署以及故障定位能力方面分别作以比较。

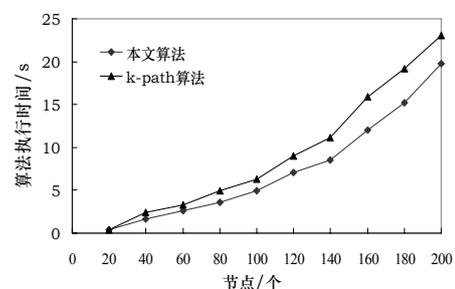


图 2 算法执行时间比较

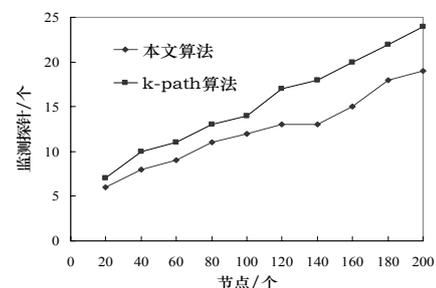


图 3 部署探针个数比较

本文算法采用基于弱连接点求取独立路径, 而网络中大量弱连接点的存在将使得本文算法的优势更为突出。图 2 显示出了在不同规模的网络下, 本文算法

$k$ -path 和算法在执行时间上的差异。从结果中可以看出两个算法的运行时间之差随着网络规模的扩大越来越大。

本文通过为监测探针集限定较小的故障个数,降低了探针本身对独立路径个数的要求,从而降低了监测探针的选取个数。图3显示出了在相同网络规模下,本文算法部署的监测探针的数目少于  $k$ -path 算法。

为考察两种算法部署探针的故障定位能力,实验通过在相同网络规模下,100次随机指定网络中的个故障,然后通过统计两种算法部署的探针每次成功定位的故障个数和的比值的平均值,来考察两种算法部署的探针对网络故障的定位能力。图4给出了在节点个数为40,节点平均度数为2.6的网络中,不同  $k$  值下,两种算法部署的探针的故障定位能力比较。

从结果中可以看出,随着  $k$  值的增大,两种算法部署的探针能够准确定位出的故障个数的平均百分比均有下降趋势,这是因为网络中大量的叶子节点的故障,在其唯一的一条独立路径中出现故障节点后,将变的不可探测。此外,当  $k$  值小于网络中节点平均度数值时,不难看出两种算法部署的探针对整个网络的故障均有较高的探测能力。但当值  $k$  大于节点的平均度数时,  $k$ -path 算法部署的探针的故障定位能力已明显不如本文算法。

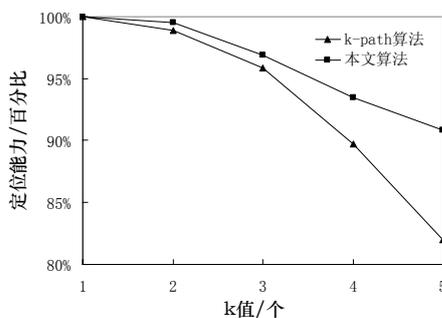


图4 故障定位能力比较

## 4 结语

本文在非确定性环境下,给出了一种支持多故障定位的探针部署方法。该方法较好的解决了度数小于  $k$  的节点的监测问题,并通过利用弱连接点求取独立路径的方式,降低了探针选择算法的执行时间。仿真结果显示,本文的探针选择算法可以有效节约探针开销,且部署的探针集具有较好的故障定位能力。

本文算法假设网络中不存在噪声,而现实网络中的噪声是普遍存在的。因此,找取在概率环境下具有抗噪能力的监测探针集合将是下一阶段工作的重点。

## 参考文献

- Horton J, Lopez-Ortiz A. On the number of distributed measurement points for network tomography. In IMC'03, 2003
- Bejerano Y, Rastogi R. Robust monitoring of link delays and faults in IP networks. Proc. of IEEE Infocom, Apr. 2003.
- Natu M, Sethi AS. Probe Station Placement for Fault Diagnosis. Global Telecommunications Conference, 2007, 113-117.
- 褚灵伟,邹仕洪,程时端,等.概率和噪声环境下基于主动探针的Internet服务故障管理.中国科学E辑:信息科学,2008, 38(10):1733-1746.
- Li F, Thottan M. End-to-end service quality measurement using source-routed probes. IEEE INFOCOM, 2006
- Natu M, Sethi AS. Probabilistic fault diagnosis using adaptive probing. IFIP/IEEE International Workshop on Distributed Systems: Operations and Management. San Jose: IEEE, 2007, 38-39.
- Winick J, Jamin S. Inet-3.0: Internet topology generator. University of Michigan, 2002.
- Huang XH, Zou SH, Wang WD, et al. Fault management for internet service: modeling and algorithms. IEEE International Conference on Communications, 2006