ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.18, No.10, October 2007, pp.2584−2594
DOI: 10.1360/jos182584

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

Internet服务故障管理:分层模型和算法*

黄晓慧, 邹仕洪+, 褚灵伟, 程时端, 王文东

(北京邮电大学 网络与交换国家重点实验室,北京 100876)

Internet Services Fault Management: Layering Model and Algorithm

HUANG Xiao-Hui, ZOU Shi-Hong+, CHU Ling-Wei, CHENG Shi-Duan, WANG Wen-Dong

(State Key Laboratory of Networking and Switching, Beijing University of Posts and Telecommunications, Beijing 100876, China)

+ Corresponding author: Phn: +86-10-62246844, Fax: +86-10-62247638, E-mail: zoush@bupt.edu.cn, http://bnrc.cs.bupt.cn

Huang XH, Zou SH, Chu LW, Cheng SD, Wang WD. Internet services fault management: Layering model and algorithm. *Journal of Software*, 2007,18(10):2584−2594. http://www.jos.org.cn/1000-9825/18/2584.htm

Abstract: In window-based Internet service fault management, improper time window size setting will affect the fault diagnosis algorithm. In order to reduce the impact, challenges of Internet service fault management are analyzed in this paper, and a layering model is recommended. Bipartite graph is chosen to be the fault propagation model (FPM) for each layer. A window-based fault diagnosis algorithm MFD (multi-window fault diagnosis) is proposed for the bipartite FPM. MFD takes the correlation of adjacent time windows into account. As a result, it can reduce the impact of improper time window size setting. Simulation results prove the validity and efficiency of MFD.

Key words: fault management; layering model; fault propagation model; fault diagnosis; bipartite graph

摘 要: 在基于时间窗口的 Internet 服务故障管理中,时间窗口大小设置不合适会给算法准确度带来影响.为了降低这种影响,分析了 Internet 服务故障管理中存在的问题,提出了分层故障管理模型,采用图论技术进行故障诊断,选择二分图作为各层的故障传播模型.提出了基于时间窗口的故障诊断算法——多窗口故障诊断(multi-window fault diagnosis,简称 MFD),该算法通过综合考虑相邻时间窗口之间的关联关系,在一定程度上降低了因时间窗口大小设置不合适而给算法准确度带来的影响.仿真结果证明了 MFD 算法的有效性和效率.

关键词: 故障管理;分层模型;故障传播模型;故障诊断;二分图

中图法分类号: TP393 **文献标识码**: A

近年来,Internet 已逐渐成为大多数人日常生活中不可缺少的一部分.服务提供商(service provider,简称 SP)意识到在 Internet 上提供增值服务能带来潜在的高额利润,并开发了各种各样的 Internet 服务.然而,我们在使用 Internet 服务的过程中也遇到了很多问题,服务不可用或性能降级都将影响 SP 的信誉.故障管理对于 Internet 服务运营非常重要.

* Supported by the National Basic Research Program of China under Grant Nos.2003CB314806, 2006CB701306 (国家重点基础研究发展计划(973)); the National Natural Science Foundation of China under Grant Nos.60603060, 90604019, 60472067 (国家自然科学基金)

Received 2006-04-26; Accepted 2007-02-15

本文提出了一个分层故障管理模型,将故障管理任务划分到多个独立的层次,每一层只关注相应的症状和故障,从而简化了故障诊断过程,层与层之间的交互进一步提高了故障定位算法的准确度.

本文研究采用图论技术(包括依赖图[1]、二分图[2,3]、贝叶斯网络[4]等)的非确定性故障诊断,选择二分图(bipartite graph,简称BG)作为故障传播模型(fault propagation model,简称FPM),其原因在于:

一方面,在构建复杂模型(如依赖图模型和贝叶斯网络)时具有知识工程瓶颈(knowledge engineering bottleneck)[5],尽管一些研究人员致力于系统组件依赖关系的分析[6],但要动态地维护依赖图模型仍存在一定的难度.

另一方面,复杂模型中,故障诊断的计算复杂度较高[7],为降低复杂度,通常将复杂模型简化为二分图模型.

在先前的工作中,我们为二分图故障传播模型设计了一种基于时间窗口的故障诊断算法MCA+(最大覆盖算法)[8].然而,基于时间窗口的算法存在着固有缺陷,即时间窗口大小设置不合适将会影响算法的准确度.为了解决此问题,本文考虑相邻时间窗口的关联关系,提出了多窗口故障诊断(multi-window fault diagnosis,简称MFD)算法.仿真结果表明:即使在时间窗口大小设置不合适的情况下,MFD仍能获得比MCA+更高的故障检测率和更低的误判率.

本文第1节简要回顾Internet服务故障管理的相关研究工作.第2节描述分层故障管理模型、各层的管理任务以及存在的困难.MFD算法在第3节进行详细介绍.第4节给出仿真模型和实验结果分析.第5节总结全文并提出未来的研究工作.

1 Internet服务故障管理相关工作

Internet服务可能由多个应用程序组件共同提供,应用层故障是造成服务故障的主要原因.Emre和Armando在文献[9]中提出了检测Internet服务应用层故障的工具Pinpoint,通过观察组件之间的交互以及用户请求所经历的路径学习应用程序的正常行为,并根据该参考模型检测服务出现的异常.Lingkun等人在文献[10]中指出:由于服务组件之间存在复杂的调用关系,某些部分发生故障或某个组件过载都可能引起整个系统的性能降级,为此,他们提出“依赖容器(dependency capsule)”的概念,在基于线程的服务中,通过切断组件之间的依赖关系来解决该问题.

操作系统宕机是造成Internet服务故障的严重错误,将引起服务完全不可用.在这些情况下,最直接的恢复方法是重启系统.然而,重启会破坏正在运行的事务、造成会话状态丢失并引起服务中断,对紧急和事务型的服务并不适用.为此,Sultan等人提出了会话迁移机制,包括用于迁移传输层会话的迁移TCP(migrated TCP,简称M-TCP)[11],以及在操作系统宕机情况下用于迁移系统级会话的“后门(backdoor)”技术[12].

Internet服务依赖于IP网络来传输其流量,而网络异常是无法避免的,会对服务造成影响.传统网络故障管理是一个较为成熟的研究领域,研究人员提出了许多故障诊断技术[1−4,13,14].然而,面向服务与面向网络的故障管理之间存在着许多差异[15].Steinder和Sethi在文献[4]中在贝叶斯推理算法基础上设计了故障检测算法;他们在文献[16]中扩展了先前的工作,为跨多个自治域的服务提出了分布式故障诊断算法.

上述文献都表明,Internet服务会受不同层故障的影响.服务故障管理的完整解决方案必须考虑应用层软件、操作系统以及网络的故障对服务产生的影响.分层模型是服务故障管理的恰当选择,可以将复杂的故障管理任务分解和简化.分层的思想早已被广泛运用在网络技术上:Rajeev在文献[17]中提出了支持故障隔离和恢复的分层模型.在文献[18]中,Srikanth等人开发了IP网络故障诊断工具“Shrink”.Shrink包含两层,可用于分析IP层故障和WDM(wavelength division multiplexing)网络中链路中断之间的关联关系.然而,这两个分层模型都是面向网络而非面向服务的,只着眼于网络故障检测,缺乏对Internet服务本身特性的综合考虑.

2 Internet服务分层故障管理模型

2.1 为什么要分层?

服务运营价值链上存在多个角色,不同角色对于故障管理的需求有所不同.用户关注服务是否可用这个事

实;服务组件开发商关注软件错误(bug)和执行逻辑故障;运营服务的服务提供商(SP)则需要检测执行平台的故障,为软件提供稳定而可靠的执行环境.分层模型是简化故障诊断的有效方法:通过分层,故障管理可以根据不同角色的需求、角度和粒度进行剪裁,每一层只关注特定症状和故障.当故障诊断在高层进行时,低层故障被汇聚和隐藏,从而简化了故障诊断任务.

2.2 服务的分层故障管理模型

本文提出的Internet服务分层故障管理模型,如图1左侧所示.每一层都有其相应的关注者:服务用户关注服务交互层;服务组件开发商关注服务软件层;SP 关注执行平台故障;而 NP 则关注网络故障.

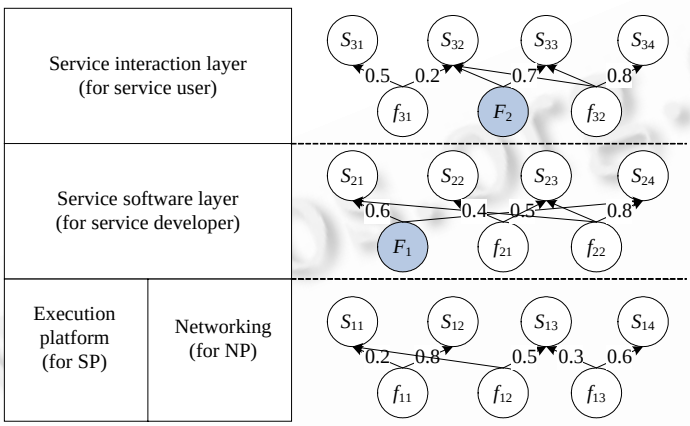


Fig.1 Layering fault management model and bipartite FPM

图 1 分层故障管理模型和二分图故障传播模型

服务交互层

主要检测软件组件是否可用,不探究故障的具体原因,所有低层故障被看作是单一起因.组件可用性检测可借助一些测试手段,如模拟用户调用、使用心跳或轮询机制,以确定组件是否发生故障.

了解组件之间的交互关系有助于故障根源的确定,这些关系可以在服务创建时得到.以Web服务为例,BPEL(business process execution language)[19]提供了描述合成服务的工具.

服务软件层

软件工程中通常采用的软件测试技术可用于发现软件源码中存在的大部分错误,保证服务软件的可靠性,但却无法修正所有错误.在软件交付使用以后,还需要对软件行为进行监测.

平台层

执行平台故障.所谓执行平台,主要是指 Internet 服务所驻留的计算机或服务器.该层故障分为软件故障和硬件异常.驻留在主机上的 Agent 能够帮助检测执行平台故障,监控平台的资源使用率,并在资源使用率超过特定门限时发出告警信息.

网络故障.主要关注造成服务传输问题的协议软件异常和网络设备故障.协议故障包括如 IP 地址配置错误引起的间歇性网络故障、核心网络服务(如 DHCP(dynamic host configuration protocol)和 DNS(domain name system)服务)配置错误造成的连接问题等等.网元设备的故障模式更为复杂,如本地网络接口、电缆、集线器、交换机、网桥、路由器的物理故障等等.

2.3 构建二分图故障传播模型

在进行故障诊断之前,需要为各层建立二分图FPM.二分图FPM的形式化定义为多元组$BG=(F,S,P_F,P_{F,S})$,F为可能发生的故障集合,S为由F引起的症状集合.在现实网络中,由于配置错误或是症状被其他组件所掩盖,只有一部分S可被观察到[20].$P_F=\{p(f)|f\in F\}$是F集合中故障发生的概率,由专家分配或从历史数据估算得

到.$P_{F,S}=\{p(s|f)|s\in S,f\in F\}$是一个具有$|F|\times|S|$个元素的二维条件概率矩阵,其中,$p(s|f)$是故障$f$引起症状$s$被观察到的概率.基于二分图的故障诊断适用于网管日志中保存有大量先验知识的情况,这些知识也可以通过故障注入(fault injection)技术[21]来获得.根据日志信息,条件概率$p(s|f)$可以通过公式(1)计算得到.一个样本定义为$Sample=\{F_X,S_X\}$,其中,S_X是在该样本中观察到的症状构成的集合,而F_X是诊断结果中包含的故障集合.

$$p(s\mid f)=\frac{Number\ of\ \ Sample\ in\ which\ \{s\in S_X,f\in F_X\}}{Number\ of\ \ Sample\ in\ which\ \{f\in F_X\}} \tag{1}$$

当某一层接收到告警信息时,该层将启动故障诊断过程.在高层进行故障诊断时,所有低层故障将被看作单一的故障起因.当在高层上进行的故障诊断认为低层故障是问题根源,则在低层上进一步查找故障源.

3 基于时间窗口的故障诊断算法的改进

在先前的工作中,我们为二分图故障传播模型设计了一种基于时间窗口的故障诊断算法MCA+(最大覆盖算法)[8].然而,如Steinder在文献[3]中指出的那样,基于时间窗口故障诊断算法的准确度受时间窗口大小设置值的影响.实际中很难将时间窗口大小设置得完全准确,因此,需要算法本身能够适应不准确的时间窗口设置值.为此,本文在MCA+算法基础上提出了MFD(多窗口故障诊断,multi-window fault diagnosis)算法,并通过仿真证明了算法的有效性.

在介绍MFD算法之前,首先给出故障定位算法的形式化定义,并分析不准确的时间窗口设置值对于算法准确度的影响.

3.1 故障诊断算法形式化定义

输入:

1. 故障传播模型$BG=(F,S,P_F,P_{F,S})$
2. 可观察的症状集合S_O
3. 症状可观察率$OR=|S_O|/|S|$,该值由系统的可管理程度决定(即系统中嵌入的管理代码量)
4. 在某个时间窗口内观察到的症状信息$S_N\subseteq S_O$
5. 症状丢失率$LR(s),s\in S_O$
6. 症状虚假率$SSR(s),s\in S_O$

假设:

1. Noisy-OR 模型,即引起同一症状的各个故障相互独立,并使用逻辑操作符“OR”进行连接;
2. 故障之间相互独立;
3. 多个故障同时发生的概率比单个故障发生的概率要低(这是故障诊断的常用假设,由于故障假设结果通常需要进一步测试以确定确切的故障,为节省测试成本,故障假设中包含的故障数目越少越好).

输出:最可能故障假设 $H=\bigcup_{i=1}^{n}\{f_i\mid f_i\in F\}$,具有以下性质:

1. 可被故障假设H中的至少一个故障$f_i\in H$所解释;
2. 故障假设所包含的故障数量$|H|=n$越小越好.

3.2 时间窗口算法的根本缺陷

当时间窗口大小设置不准确时,故障诊断算法将会由于排除了某些症状或包含过多的症状而得出不准确的故障假设[3].图 2 分析了时间窗口设置值小于理论值的情况.二分图FPM描述了故障和症状之间的关联关系.时间窗口的理论值为$W_T,W_1=W_2=W_T,SW_1=SW_2<W_T$.在时间窗口$W_1$,发生了两个故障$f_1$和$f_2$,其相应的告警信息是$s_1-s_3$和$s_4-s_6$.在时间窗口$W_2$,故障$f_1$和$f_2$消失,新故障$f_3$出现,其相应的告警信息是$s_7-s_9$.如果时间窗口的设置值小于$W_T$,则由故障$f_1$引起的所有症状可能分布在两个时间窗口中.如图 2 所示,症状s_3在时间窗口SW_2而非SW_1中被观察到.由于症状s_3是由f_1引起的,因此,时间窗口SW_2的故障假设应该为$\{f_1,f_3\}$.然而,根据MCA+算法最大覆盖的性质,得出的故障假设将为$\{f_3,f_4\}$.

同理,当时间窗口设置值大于W_T时也会影响算法的准确度.

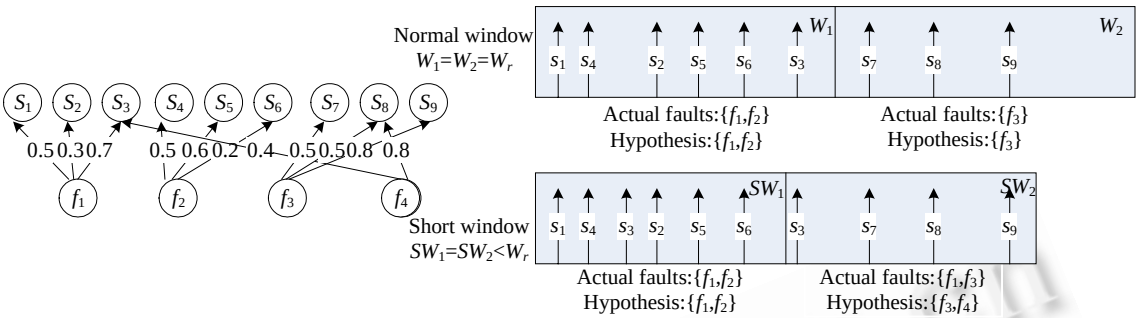


Fig.2 Impact of improper time windows size

图 2 时间窗口设置不准确对算法的影响

3.3 多窗口故障诊断算法MFD

考虑到时间窗口大小设置不准确带来的影响,本文提出了MFD算法来解决此问题.与MCA+类似,MFD对于故障的"效用"$G(f,S_N)$定义如下:

如果 $LR(s)\neq 0$ 或 $SSR(s)\neq 0$,即当存在丢失或虚假症状时,

$$G(f,S_N)=\frac{\sum_{s\in d(f)\cap S_N}(1-LR(s))\times p(s,f)}{\sum_{s_l\in d(f)\setminus S_N}LR(s_1)\times p(s_l,f)+\sum_{S_p\in d(f)\cap S_N}SSR(s_P)\times p(s_p,f)} \tag{2}$$

否则,

$$G(f,S_N)=\sum_{s\in d(f)\cap S_N}p(s,f) \tag{3}$$

在 MFD 算法中,故障 f 的最终"效用"需要根据历史故障假设进行修改.如果一个故障没有出现在上一个时间窗口中,该故障的"效用"将被降低.MFD 算法描述如下(其中,i 表示时间窗口的序号):

算法. MFD.

- 对于第 1 个时间窗口(i=0),执行以下步骤:

1. 设$H_0=\varnothing$
2. 找出S_N中每一个症状$s_k\in S_N$的可能故障源,构成待选故障集合 F_{S_N}

$$F_{S_N}=\{f\mid f\in F,s\in d(f)\cap S_N\}$$

3. 对于 F_{S_N} 中的每一个故障,计算该故障的效用$G(f,S_N)$
4. 选择效用值最大的故障f并将其加入故障假设H_0中
5. 从S_N中删除f能够解释的所有症状,$S_N=S_N-d(f)\cap S_N$
6. 返回步骤 2,直到$S_N=\varnothing$

- 对于其他时间窗口(i>0),执行以下步骤:

1. 设$H_i=\varnothing$
2. 找出S_N中每一个症状$s_k\in S_N$的可能故障原因,构成待选故障集合 F_{S_N}

$$F_{S_N}=\{f\mid f\in F,s\in d(f)\cap S_N\}$$

3. 对于 F_{S_N} 中的每一个故障

{

计算该故障的效用$G(f,S_N)$;

如果f在H_{i-1}中,$G(f,S_N)=G(f,S_N)$;

否则,$G(f,S_N)=G(f,S_N)\times 0.5$;

}

4. 选择效用值最大的故障f并将其加入故障假设H_i中,$H_i=H_i\cup\{f\}$
5. 从S_N中删除所有f能够解释的症状,$S_N=S_N-d(f)\cap S_N$
6. 返回步骤2,直到$S_N=\varnothing$

3.4 计算复杂度

MFD算法在最坏情况下的时间复杂度与MCA+相同,都为$O(|S_O|^2|F|)$.在最坏情况下,所有可被观察的症状都出现了,即$S_N=S_O$,并且每一次迭代选择的故障都只能解释一种症状,即$|F_{S_N}|=|F|$,因此,算法对于步骤2~步骤6需要执行$|S_N|$次.每次迭代时,步骤2被执行$|S_N|$次;在计算故障的"效用"时,外循环需要执行至多$|F|$次,而内循环需要执行最多$|S_N|$次,以获得条件概率值,即步骤3的循环次数为$|F|\times|S_N|$.因此,整个算法的复杂度为

$$O(|S_N|^2|F|)=O(|S_O|^2|F|).$$

4 仿真结果和分析

4.1 仿真模型

本文的仿真参考了我们先前工作中用于验证MCA+算法的仿真模型[8].由于各层的二分图故障传播模型对于MFD算法来说没有区别,因此,仿真只在网络层进行.仿真程序运行在配置为2.40GHz Intel Pentium (R) 4 CPU和512M内存的计算机上.

在仿真开始时,首先随机产生包含n个节点的树状或网状网络,并使用最短路径优先算法来计算节点之间的路径.在路径确定后,根据文献[8]中提出的建模方法构建故障传播模型$BG=(F,S,P_F,P_{F,S})$.故障发生概率P_F和故障-症状间的条件概率$P_{F,S}$都随机产生,分别服从参数为[0.001,0.01]和(0,1)的均匀分布.

每一个仿真场景由4个参数确定:症状观察率OR,症状丢失率LR,虚假症状率SSR以及网络大小n.对于每个仿真场景,运行500~1 000个仿真案例.

在每一个案例中,产生100个连续正常时间窗口nw情况下的故障-症状映射样本$\{F_{ci},S_{ci}\}(i=1,2,\ldots,100)$,其中,样本$\{F_{ci},S_{ci}\}$表示在第$i$个时间窗口中发生的故障构成的集合为$F_{ci}$,其相关的症状信息构成的集合为$S_{ci}$.每一个时间窗口的样本的产生过程如下:随机选择故障集合$F_{ci}\subseteq F$作为发生的故障,包含1,2,3,4和4个以上故障的案例比例为5:5:3:2:1;从所有症状中随机选择$OR|S|$个症状作为可观察症状,形成集合S_O,随后从S_O中选取与F_{ci}中故障相对应的症状,构成症状集合S_N;为了模拟症状丢失,从S_N中随机选择$LR\times|S_N|$个症状作为已被丢失的症状,并从S_N中删去;从S_O中选取$SSR\times|S_O|$个症状作为虚假症状加入到S_N中.因此,在最后的样本$\{F_{ci},S_{ci}\}$中,症状集合S_{ci}包含有$(1-LR)\times|S_N|+SSR\times|S_O|$个症状.

在产生样本后,100个正常时间窗口的故障-症状映射样本$\{F_{ci},S_{ci}\}(i=1,2,\ldots,100)$将被输入到交织模块.图3给出了相邻2个时间窗口的交织实例,实际时间窗口SW_i的大小等于正常时间窗口W_i的2/3.

与文献[8]相同,本文使用了4个指标来评估故障定位算法:检测率(detection rate,简称DR)、误判率(false positive rate,简称FPR)、检测率方差(variance of detection rate,简称VDR)以及故障诊断时间.前3个指标的定义如下:

$$DR=\frac{|F_C\cap H|}{F_C} \tag{4}$$

$$FPR=\frac{|H-F_C|}{|H|} \tag{5}$$

$$VDR=\frac{\sum_{i=1}^{N}(DR_i-\sum_{i=1}^{N}DR_i/N)^2}{N} \tag{6}$$

其中,F_C表示实际发生的故障,H表示由故障定位算法产生的故障假设,而N表示在同一个故障场景中仿真案例的个数.

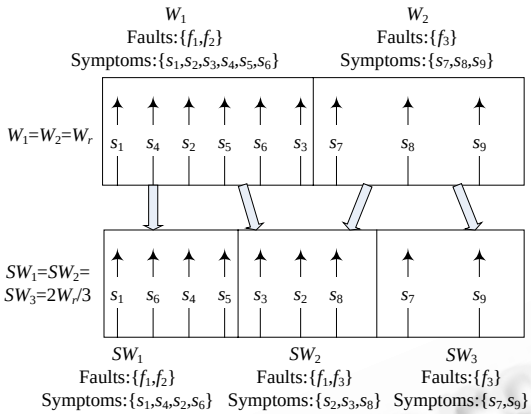


Fig.3 A weave example

图 3 交织实例

4.2 结果和分析

图 4 和图 5 比较了 MCA+和 MFD 算法的故障检测率.仿真场景的参数为 *OR*=0.5,*LR*=0.1 和 *SSR*=0.01,小时间窗口 *sw*=2/3,而大时间窗口 *lw*=4/3.图 4 给出了 MFD 和 MCA+算法故障检测率随时间的变化曲线;而图 5 比较了不同网络大小中两种算法的故障检测率.从图 4 可以看出:当时间窗口大小设置得不合适时(SW_MFD 和 SW_MCA+),随着时间的推移,检测率将产生抖动,但 MFD 算法的故障检测率曲线比 MCA+算法的曲线震荡幅度要小,说明 MFD 的诊断结果较为稳定.图 5 表明,MFD 在时间窗口大小设置准确的情况下(NW_MFD 和 NW_MCA+),其故障检测率与 MCA+相当;而当时间窗口设置不准确时,MCA+算法的故障检测率将下降(NW_MCA+,SW_MCA+和 LW_MCA+);而由于 MFD 考虑了相邻时间窗口的关联关系,因此,无论在小窗口或大窗口情况下,其故障检测率都比 MCA+算法要高.

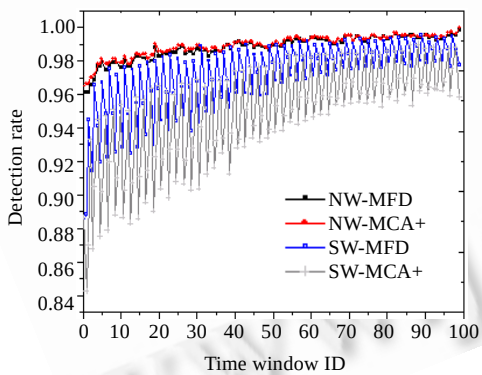


Fig.4 Detection rate comparison (*X*=time window ID) (*OR*=0.5,*LR*=0.1,*SSR*=0.01)

图 4 检测率比较(*X*=时间窗口 ID)

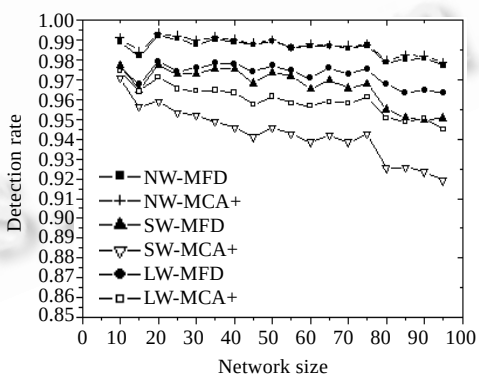


Fig.5 Detection rate comparison (*X*=network size) (*OR*=0.5,*LR*=0.1,*SSR*=0.01)

图 5 检测率比较(*X*=网络大小)

图 6 比较了两种算法的误判率.当时间窗口设置准确时,MFD 算法的误判率接近于 MCA+算法;但当时间窗口大小设置不准确,尤其是在小时间窗口的情况下,MFD 算法的误判率比 MCA+算法要低.这主要是由于 MFD 算法认为,在上一个时间窗口中出现的故障在当前时间窗口出现的概率更大,从而避免了在故障假设中引入虚假故障.

图 7 比较了故障检测率方差 *VDR*.*VDR* 描述了故障检测率的分散程度,可用于衡量算法的稳定性.*VDR* 的值

越低,说明算法的稳定性越好.图 7 中的曲线表明:在时间窗口设置准确的情况下,MFD 和 MCA+算法具有相似的稳定性;但当时间窗口设置不准确时,MFD 算法比 MCA+算法要稳定.

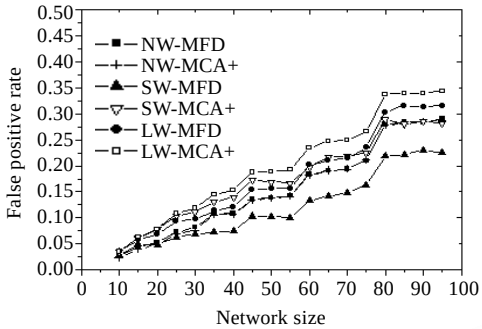


Fig.6 False positive rate comparison (*OR*=0.5,*LR*=0.1,*SSR*=0.01)

图 6 误判率比较

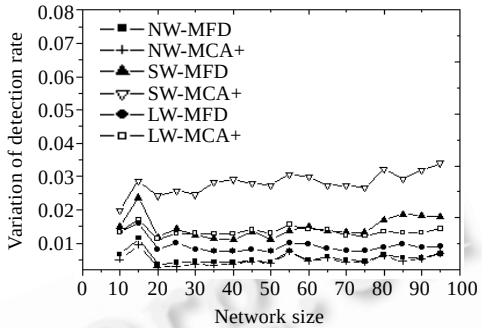


Fig.7 Variation of detection rate comparison (*OR*=0.5,*LR*=0.1,*SSR*=0.01)

图 7 检测率方差比较

如第 3.4 节所述,MFD 算法和 MCA+算法的计算复杂度相同,图 8 验证了上述分析结果.从图 8 可以看出:无论时间窗口大小设置准确与否,MFD 算法的故障诊断时间曲线和 MCA+算法的时间曲线有着类似的走向.但对于某些特定场景,MFD 稍快于 MCA+算法,这主要是由于 MCA+算法的循环次数较多所造成的.

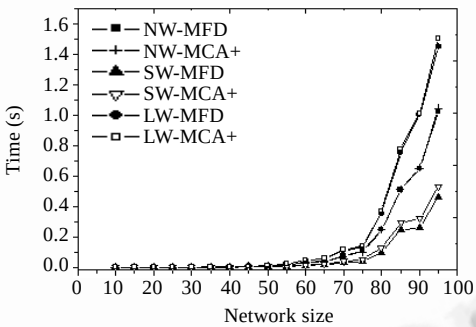


Fig.8 Computation time comparison (*OR*=0.5,*LR*=0.1,*SSR*=0.01)

图 8 计算时间比较

图 9~图 14 给出了症状可观察率 *OR* 分别为 0.5,0.3 和 0.1 情况下的部分仿真结果.受篇幅所限,本文只给出这些仿真场景中故障检测率和误判率的比较.图 9 比较了两种算法在正确时间窗口情况下的故障检测率,由图 9 可以看出:在具有相同 *OR* 的正常时间窗口情况下,MFD 算法的检测率与 MCA+算法大致相等.图 10 的曲线说明了在具有相同 *OR* 的小窗口情况下,MFD 比 MCA+算法准确.

由图 11 可以看出,在具有相同 *OR* 的大窗口情况下,MFD 算法的检测率高于 MCA+算法.此外,图 9~图 11 还说明了随着 *OR* 的下降,故障检测率也将随之降低.图 12 的曲线表明了在正常时间窗口情况下,MFD 算法的误判率与 MCA+相当;而图 13 和图 14 则描述了当时间窗口大小设置不合适时,MFD 算法可以获得比 MCA+算法较低的误判率.

仿真结果证明:当时间窗口设置准确时,MFD 算法的性能与 MCA+算法类似;而由于引入了对相邻时间窗口关联关系的分析,在时间窗口大小设置不合适的情况下,MFD 仍能获得比 MCA+算法更好的性能.此外,MFD 算法的计算复杂度与 MCA+算法相同,在某些特定场景中,甚至能比 MCA+算法更快地得出诊断结果.在实际网络中,由于时间窗口的大小不可能在任何时候都设置准确,因此,MFD 算法比 MCA+算法更优.

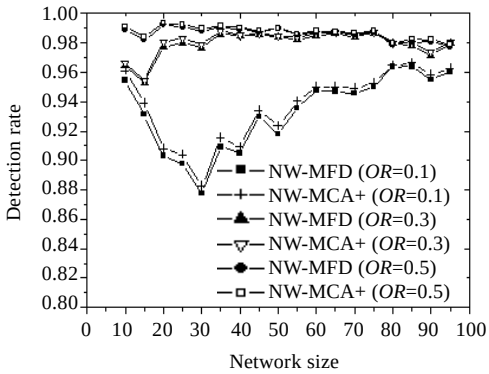


Fig.9 Detection rate comparison (normal time window) (*LR*=0.1,*SSR*=0.01)

图 9 检测率比较(正常时间窗口)

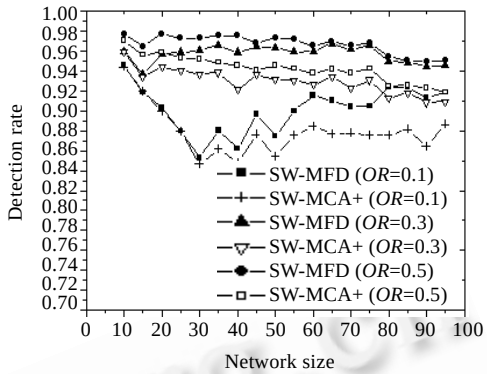


Fig.10 Detection rate comparison (small time window) (*LR*=0.1,*SSR*=0.01)

图 10 检测率比较(小时间窗口)

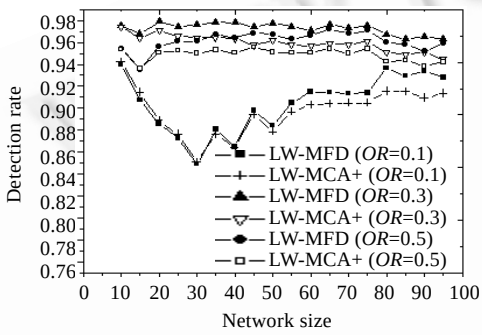


Fig.11 Detection rate comparison (large time window) (*LR*=0.1,*SSR*=0.01)

图 11 检测率比较(大时间窗口)

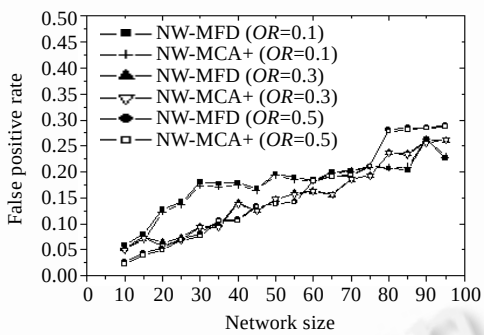


Fig.12 False positive rate comparison (normal time window) (*LR*=0.1,*SSR*=0.01)

图 12 误判率比较(正常时间窗口)

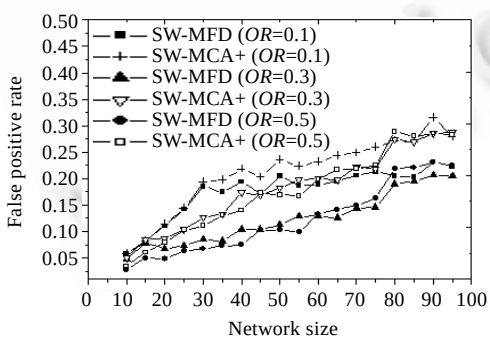


Fig.13 False positive rate comparison (small time window) (*LR*=0.1,*SSR*=0.01)

图 13 误判率比较(小时间窗口)

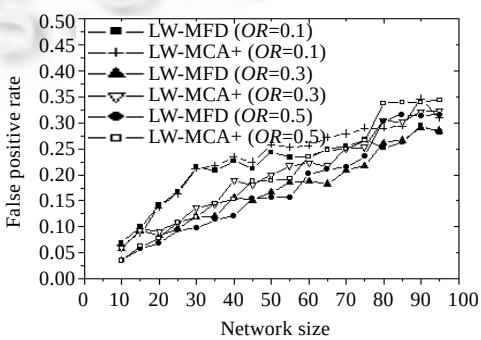


Fig.14 False positive rate comparison (large time window) (*LR*=0.1,*SSR*=0.01)

图 14 误判率比较(大时间窗口)

5 结束语

本文分析了Internet服务故障管理中存在的困难,提出了一个分层故障管理模型,并选择二分图作为各层的故障传播模型.另外,本文提出了MFD算法,该算法是MCA+算法[8]的改进版本,通过考虑相邻时间窗口的关联关系,在一定程度上克服了基于时间窗口的故障诊断算法的固有缺陷,即时间窗口大小设置不准确造成算法准确度下降.仿真结果表明:当时间窗口设置不合适时,MFD算法可以获得比MCA+算法更高的故障检测率和更低的误判率.此外,MFD算法和MCA+算法具有相同的时间复杂度,并且比MCA+算法更稳定.

本文在算法设计时假设使用 Noisy-OR 模型,这在实际网络中是不充分的,在未来工作中,我们将综合考虑 AND 模型和其他规范模型.此外,对于实际运营的 Internet 服务,故障和症状之间的关系非常复杂,我们计划研究故障注入技术以获得故障传播模型,并通过实验进一步验证 MFD 算法.

References:

[1] Katzela I, Schwartz M. Schemes for fault identification in communication networks. IEEE/ACM Trans. on Networking, 1995,3(6):753−764.

[2] Yemini SA, Kliger S, Mozes E, Yemini Y, Ohsie D. High speed and robust event correlation. Communications Magazine, 1996, 34(5):82−90.

[3] Steinder M, Sethi AS. Probabilistic event-driven fault diagnosis through incremental hypothesis updating. In: Proc. of the IFIP/IEEE Symp. on Integrated Network Management. 2003. 635−648. http://www.comsoc.org/confs/im/2003/index.html

[4] Steinder M, Sethi AS. Probabilistic fault localization in communication systems using belief networks. IEEE/ACM Trans. on Networking, 2004,12(5):809−822.

[5] Korb KB, Nicholson AE. Bayesian Artificial Intelligence. Boca Raton: Chapman & Hall/CRC Press, 2003. 225−226.

[6] Brown A, Kar G, Keller A. An active approach to characterizing dynamic dependencies for problem determination in a distributed environment. In: Proc. of the IEEE/IFIP Int'l Symp. on Integrated Network Management, IM2001. 2001. 377−390. http://www.comsoc.org/confs/im/2001/

[7] Dagum P, Luby M. Approximating probabilistic inference in Bayesian belief networks. IEEE Trans. on Pattern Analysis and Machine Intelligence, 1993,15(3):246−255.

[8] Huang XH, Zou SH, Wang WD, Cheng SD. Fault management for Internet service: Modeling and algorithms. In: Proc. of the IEEE Int'l Conf. on Communications, ICC 2006. 2006. http://www.ieee-icc.org/2006/

[9] Kiciman E, Fox A. Detecting application-level failures in component-based Internet services. IEEE Trans. on Neural Networks, 2005,16(5):1027−1041.

[10] Chu L, Shen K, Tang H, Yang T, Zhou J. Dependency isolation for thread-based multi-tier Internet services. In: Znati T, Knightly E, Makki K, eds. Proc. of the IEEE INFOCOM, Vol. 2. New York: IEEE Press, 2005. 796−806. http://www.ieee-infocom.org/2005/

[11] Sultan F, Srinivasan K, Iyer D, Iftode L. Migratory TCP: Connection migration for service continuity in the Internet. In: Proc. of the 22nd Int'l Conf. on Distributed Systems. 2002. 469−470. http://icdcs2002.di.fc.ul.pt/

[12] Sultan F, Bohra A, Smaldone S, Pan Y, Gallard P, Neamtiu I, Iftode L. Recovering Internet service sessions from operating system failures. IEEE Internet Computing, 2005,9(2):17−27.

[13] Jakobson G, Weissman M. Alarm correlation. IEEE Network, 1993,7(6):52−59.

[14] Lewis L. A case-based reasoning approach for the resolution of faults in communication networks. In: Proc. of the 3rd IFIP/IEEE Symp. on Integrated Network Management. Amsterdam: North-Holland Publishing Co., 1993.

[15] Huang XH, Zou SH, Wang WD, Cheng SD. MDFM: Multi-domain fault management for Internet services. In: Royo JD, Hasegawa G, eds. Proc. of the 8th Int'l Conf. on Management of Multimedia Networks and Services, MMNS 2005. LNCS 3754, New York: Springer-Verlag, 2005. 121−132.

[16] Steinder M, Sethi AS. End-to-End service failure diagnosis using belief networks. In: Proc. of the Network Operations and Management Symp. (NOMS). Florence, 2002. 375−390. http://www.noms.org/2002/

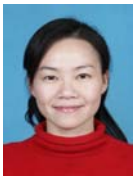
[17] Gopal R. Layered model for supporting fault isolation and recovery. In: Proc. of the Network Operations and Management Symp. 2000. 2000. 729−742. http://www.noms.org/2000/

[18] Kandula S, Katabi D, Vasseur JP. Shrink: A tool for failure diagnosis in IP networks. In: Proc. of the Sigcomm2005 MineNet Workshop. 2005. http://www.sigcomm.org/sigcomm2005/

[19] Weerawarana S, Francisco C. Business process with BPEL4WS: Understanding BPEL4WS, Part 1. Research Report, IBM DeveloperWorks, 2002. http://www-106.ibm.com/developerworks/webservices/library/ws-bpelcol1/

[20] Mas C, Thiran P. An efficient algorithm for locating soft and hard failures in WDM networks. IEEE Journal on Selected Areas in Communications, 2000,18(10):1900−1911.

[21] Bagchi S, Kar G, Hellerstein J. Dependency analysis in distributed systems using fault injection: Application to problem determination in an e-commerce environment. In: Proc. of the 12th Int'l Workshop on Distributed Systems: Operations and Management, DSOM 2001. Nancy, 2001. http://www.loria.fr/~festor/DSOM2001/



黄晓慧(1979－),女,广东深圳人,博士,主要研究领域为服务故障管理,Web 服务技术,QoS.

程时端(1940－),女,教授,博士生导师,主要研究领域为网络性能评估,QoS.

邹仕洪(1978－),男,博士,副教授,CCF 会员,主要研究领域为移动无线网络,网络服务质量与服务管理.

王文东(1963－),男,教授,主要研究领域为高速网络理论技术.

褚灵伟(1982－),男,博士生,主要研究领域为服务故障管理,Web 服务技术.

中国科学院软件研究所筹建国内首家软件博物馆

近日，中国科学院软件研究所发起建设我国首家以计算机软件为主题的软件博物馆。

软件博物馆旨在记录软件发展历程，展示软件发展成就，传播软件科技知识，宣传软件科学文化。软件博物馆将以丰富而翔实的史料及珍贵的实物,将计算机软件从起步到现在的发展状况以及未来发展趋势生动、直观地展示给大众。通过各种展示手段，追溯软件的发展历程，发掘软件文化内涵，弘扬科学精神，普及科技知识。

软件博物馆计划于 2008 年中期向公众开放。目前，正面向社会各界广泛征集能够反映国内、外软件发展历程和软件发展成就的实物、照片、回忆文章、模型、成果展示材料等。有捐赠意向的单位及个人请与软件博物馆建设办公室联系。

地址:北京市中关村南四街 4 号中科院软件园区 5 号楼 202 室　　邮编:100080

电话:86-10-62661035　　传真:86-10-62661035

联系人:李洁　　E-mail: rjbwg@iscas.ac.cn