基于网络演算的网络故障检测方法

章子游[1], 赵千川[1†], 杨 文[2]

(1. 清华大学 自动化系和北京信息科学与技术国家研究中心, 北京 100084;

2. 航天发射场可靠性技术重点实验室, 海南 海口 570100)

摘要: 在计算机网络系统中, 由于存在外界攻击, 设备寿命有限等原因, 网络设备不可避免地会发生故障. 本文提出了一种基于网络演算的故障检测方法, 并证明了该方法能在确定的时延范围内检测出网络中节点发生故障及恢复正常的时间, 以及节点的故障种类. 在仿真算例上测试发现, 本文的方法能及时检测出节点故障和正常的时间, 且检测的平均准确率比当前常用的基于LSTM的方法高很多.

关键词: 计算机网络; 故障检测; 网络演算; 时延

引用格式: 章子游, 赵千川, 杨文. 基于网络演算的网络故障检测方法. 控制理论与应用, 2019, 36(11): 1861 – 1870

DOI: 10.7641/CTA.2019.90496

Network faults detection based on network calculus

ZHANG Zi-you[1], ZHAO Qian-chuan[1†], YANG Wen[2]

(1. Department of Automation and BNRist, Tsinghua University, Beijing 100084, China;

2. Key Laboratory of Space Launching Site Reliability Technology, Haikou Hainan 570100, China)

Abstract: In computer networks, faults occurring to devices are common due to external interference or limited service life of devices. In this paper, we propose a fault detection method based on network calculus. We show that the method can detect the time of faults occurring or returning to normal and the type of fault in predetermined delay. Simulation show that our algorithm can detect fault or normal in time, and the accuracy is much higher than algorithm based on LSTM.

Key words: computer networks; fault detection; network calculus; delay

Citation: ZHANG Ziyou, ZHAO Qianchuan, YANG Wen. Network faults detection based on network calculus. *Control Theory & Applications*, 2019, 36(11): 1861 – 1870

1 引言

随着计算机及网络技术的不断发展, 通信网络的规模和复杂性也逐渐增加. 但由于外界攻击, 设备损坏等原因, 网络系统中的故障是不可避免的[1]. 为保证系统的正常运行, 需要快速、准确地检测网络系统中的故障. 目前网络故障检测的方法有很多, 主要包括基于知识的方法[2–4]、基于模型的方法[5–7]、基于图论的方法[8–10]、基于数据的方法[11–13]等.

基于知识的方法常常通过基于规则的推理实现, 如故障诊断的专家系统[3]. 对输入的信息, 专家系统通过规则库内的知识进行推理, 进而输出故障检测结果. Liu等[2]使用Java实现了该方法并对其进行了讨论. 王伟等[4]使用故障收集、故障过滤、专家系统和解释器4个模块构建了故障管理系统. 但基于知识的方法效率较低, 且通常只针对特定系统, 当系统的复杂度逐渐提高时规则可能不再适用.

基于模型的方法用数学模型来描述系统的行为, 通过模型预测的系统行为和系统的实际情况进行对比, 进而判断系统是否发生故障. Dupuy等[5]通过对网络对象和关系的定义设计了用于检测故障的网络管理模型. Katker等[6]提出了一种网络中的故障隔离和事件关联算法. 蒋康明等[7]使用主动探测技术, 提出了故障检测探测选择和故障定位探测选择算法来检测网络故障. 因为基于模型的方法建立在对系统性质的了解之上, 这种方法通常具有可扩展性. 但现有的基于模型方法难以检测设备联合故障的情况.

收稿日期: 2019−06−28; 录用日期: 2019−09−11.

†通信作者. E-mail: zhaoqc@tsinghua.edu.cn; Tel.: +86 10-62783612.

本文责任编委: 陈增强.

国家“十三五”重点研发项目(2017YFC0704100, 2016YFB0901900), 国家自然科学基金项目(61425027), “111”引智计划(BP2018006), 北京国家信息科学与技术研究中心项目(BNR2019TD01009)资助.

Supported by the National Key Research and Development Project of China (2017YFC0704100, 2016YFB0901900), the National Natural Science Foundation of China(61425027), the 111 International Collaboration Program of China (BP2018006) and the BNRist Program (BNR2019TD01009).

基于图论的方法需要先获取系统网络组件之间的依赖关系, 再通过故障传播模型 (fault propagation model, FPM)[8]定位网络中的故障源. Kandula等[8]提出了通过依赖关系图进行故障定位的两种算法, 分别用于解决对象独立和对象间存在依赖关系的故障情况. Steinder等[9]使用贝叶斯网络作为故障传播模型, 并提出了多项式复杂度的近似算法. Bennacer等[10]将案例推理与贝叶斯网络结合, 对故障诊断的过程进行了进一步简化. 基于图论的方法主要研究系统报警后的故障定位, 通常不包含故障发现这一步骤.

随着近年来模式识别、神经网络技术的迅速发展, 基于数据的方法开始被广泛使用. 这种方法的好处是具有泛用性, 对于不同的系统, 不需分析系统的结构和具体的模型, 均可通过大量的历史数据进行训练进而进行故障检测. Gardner等[11]使用神经网络与传统方法相结合, 进行通信网络中的故障管理. 尚志信等[12]使用粗糙集处理网络故障, 再使用处理后的规则训练神经网络. Kim等[13]使用长短期记忆网格(long short term memory, LSTM)检测网络威胁, 发现LSTM比传统的神经网络方法准确率更高. 模式识别方法需要很长的训练时间和大量的训练数据, 且只适用于特定种类的网络.

为解决目前网络故障检测方法适应性差、复杂度高的问题, 本文中, 借鉴文献[14]中运用网络演算进行交换机故障隔离的思想, 考虑网络中的数据流, 提出一种基于网络演算的信息网络系统的故障检测方法. 相比于文献[14], 本文的结果把网络演算分析故障的范围从单一通信设备拓展到了更一般的通信网络. 网络演算是一种用来处理计算机网络中排队系统的理论[15–16], 该理论的优势在于可以得到系统在最坏情况下的延迟上界, 因而常被用于具有硬实时性要求的系统性能分析, 数据流整形等. 相应地, 延迟上界估计的紧致性[17], 就成为成功应用网络演算解决实际问题有效性的关键, 也是近期网络演算理论研究的热点[18–19]. 本文通过分析流入流出设备的数据, 利用网络演算理论, 建立针对网络中的设备故障检测方法. 本文从理论上证明了该方法的有效性, 并将LSTM网络与本文的方法进行了对比实验.

2 问题描述

参考网络演算[15]的建模方式, 将计算机网络抽象成数据流和网络元素两个组成部分. 其中, 网络元素可以表示网络中的服务器、路由器、传输线路等各种网络设备, 为网络中的数据流提供服务. 由此, 可以将网络建模成有向图$G=(V,E)$. 其中: $V=\{v\}$为图中的节点集, 表示网络元素, $E=\{e_{ij}|v_i,v_j\in V\}$为图中从$v_i$指向$v_j$的边, 表示存在从节点$v_i$发送到节点$v_j$的数据流. 记从节点$v_i$发送到节点$v_j$的累计数据为$R_{ij}(t)$, 即在时间$[0,t]$内从$v_i$发送到$v_j$的总比特数. 设发送数据到节点$v_i$的节点集为$N_{i,\text{in}}=\{v_j|e_{ji}\in E\}$, 接收节点$v_i$发出数据的节点集为$N_{i,\text{out}}=\{v_j|e_{ij}\in E\}$. 则在时间$[0,t]$内, 节点$v_i$收到的累计数据为

$$R_{i,\text{in}}(t)=\sum_{v_j\in N_{i,\text{in}}}R_{ji}(t), \tag{1}$$

将$R_{i,\text{in}}(t)$称为节点v_i的输入过程. 节点v_i发送的累计数据为

$$R_{i,\text{out}}(t)=\sum_{v_j\in N_{i,\text{out}}}R_{ij}(t), \tag{2}$$

将$R_{i,\text{out}}(t)$称为节点v_i的输出过程. 在t时刻, 若节点v_i的发送数据少于接收数据, 即$R_{i,\text{out}}(t)<R_{i,\text{in}}(t)$时, 称此时节点$v_i$处存在数据积压.

为引入节点的服务模型, 需要用到网络演算中的一些基本定义.

定义 1 最小加卷积$\otimes$.

函数$a(x)$和$b(x)$之间的最小加卷积定义为

$$(a\otimes b)(x)=\inf_{0\leqslant y\leqslant x}[a(y)+b(x-y)]. \tag{3}$$

定义 2 服务曲线$S(t)$.

若节点的输入过程为$R_{\text{in}}(t)$, 节点的输入过程为$R_{\text{out}}(t)$, 则当存在函数$S(t)$满足

$$R_{\text{out}}(t)\geqslant(R_{\text{in}}\otimes S)(t) \tag{4}$$

时, 称$S(t)$为节点的服务曲线.

假设网络中的节点正常工作时为数据流提供如图1所示的服务曲线, 服务曲线$S_i(t)$表示节点服务能力的下界.

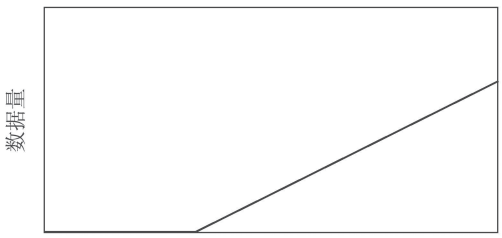

图 1 服务曲线

Fig. 1 Service curve

图1中: $S_i(t)=\max(0,r(t-\tau))$, τ为节点开始服务的最大时延, r为节点服务的最低速率. 假设网络中的节点均有该类型的服务曲线, 但不同节点的参数可能不同.

在故障检测中, 考虑以下3类故障:

1) 节点停止服务: 节点不具有对数据流提供服务的能力. 其行为特征约定如下: 若节点v_i在时间$(s,s+t]$内停止服务, 则对任意形式的输入过程和任意积压, 均有

$$R_{i,\text{out}}(s+t)-R_{i,\text{out}}(s)=0,$$

且在停止服务的时间内, 节点不保存接收的数据. 若节点在时刻$s+t$重新开始服务, 节点仅对时刻$s+t$后输入的数据提供服务.

2) 节点服务减慢: 节点对数据流提供服务, 但服务能力远低于严格服务曲线所描述的下界. 其行为特征约定如下: 若节点v_i在时间$(s, s+t]$内服务减慢, 则对任意形式的输入过程和积压, 均有

$$R_{i,\text{out}}(s+t) - R_{i,\text{out}}(s) \leqslant rt.$$

与停止服务不同的是, 在服务减慢时节点仍保存所有收到的数据, 并对其依次进行服务.

3) 节点错误输出: 节点不为接收的数据流提供服务, 且大量发送与输入过程无关的数据. 其行为特征约定如下: 若节点v_i在时间$(s, s+t]$内错误输出, 则对任意形式的输入过程和积压, 均有

$$R_{i,\text{out}}(s+t) - R_{i,\text{out}}(s) = C(t),$$

其中$C(t)$为节点在错误发送时的发送数据流, 与输入过程无关. 同时, 假设

$$C(t) \geqslant R_{i,\text{in}}(s+t) - R_{i,\text{in}}(s),$$

且$C(t) \geqslant rt$与停止服务相同, 节点错误发送时不保存接收的数据.

图2–5中给出了节点正常与发生不同种类故障时的输入输出曲线示例.

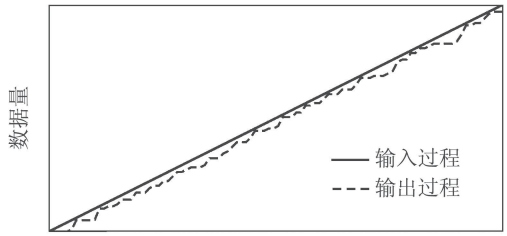

图 2　正常输入输出过程

Fig. 2　Input process and output process when normal

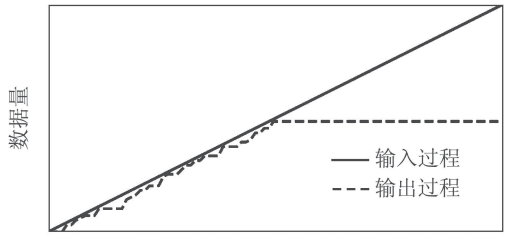

图 3　停止服务时输入输出过程

Fig. 3　Input process and output process when no service

假设这些故障不会同时发生, 且节点不同种类的故障不会连续出现, 即故障状态的节点只可能保持当前的故障状态或恢复正常. 假设节点在故障或正常状态下持续的时间均不小于τ, 即节点不会出现频繁的状态切换. 在实际过程中由于τ很小, 该假设是合理的. 故障检测的目标是通过网络中每个节点的输入过程与输出过程, 获取每个节点发生故障与恢复正常的时间及故障种类.

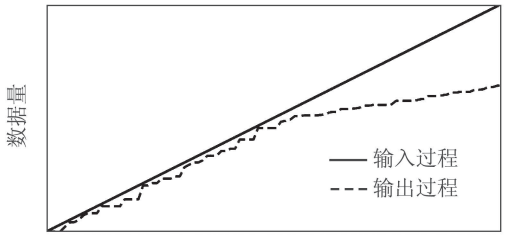

图 4　服务减慢时输入输出过程

Fig. 4　Input process and output process when service rate is slow

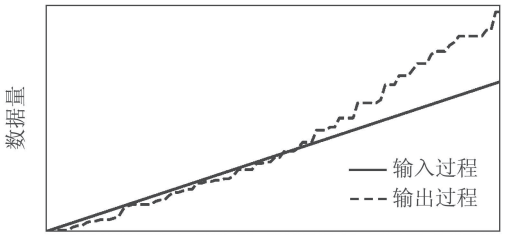

图 5　错误输出时输入输出过程

Fig. 5　Input process and output process when output error

3　主要结果

3.1　故障检测原理

为进行故障检测, 先假设网络中每个节点v的输入过程$R_{\text{in}}(t)$与输出过程$R_{\text{out}}(t)$均可测量, 且服务曲线$S(t)$已知.

从定理1中, 可以得到服务曲线与服务曲线的关系. 而服务曲线通过节点的输入过程, 对节点输出过程进行了限制. 由此, 可以得到节点正常服务时, 输入过程与输出过程之间的关系.

$$R_{\text{out}}(t) \geqslant R_{\text{in}}(t) \otimes S(t), \tag{5}$$

同时, 节点正常服务时, 发送的累计数据应不超过接收的累计数据, 因此有

$$R_{\text{out}}(t) \leqslant R_{\text{in}}(t). \tag{6}$$

对网络中的任意一个节点v, 其正常工作时输入过程$R_{\text{in}}(t)$与输出过程$R_{\text{out}}(t)$应满足式(5)–(6). 因此, 若已知其输入过程$R_{\text{in}}(t)$与输出过程$R_{\text{out}}(t)$, 则可通过式(5)–(6)判断其是否正常工作.

推论 1　若节点v的输入过程$R_{\text{in}}(t)$与输出过程$R_{\text{out}}(t)$在时刻t不满足式(5)或式(6), 则节点v在时间$[0, t]$内存在故障.

式(5)或式(6)不满足是节点故障检测的充分条件, 但并非必要条件. 即使能确定节点发生故障, 还需判断具体的故障时段, 并根据输入过程和输出过程的关系, 判断具体的故障类型. 为在理论上保证故障的可检测性, 在本节中对输入过程进行一些限制. 需要注意的是, 在实际的网络系统中, 这些限制通常不一定会被满足. 在算法设计中将对此进行进一步讨论.

为保证节点正常工作时不会出现过大的数据积压, 引入如下假设:

假设 1 节点正常服务时, 数据输入速率不大于最低服务速率, 即节点v在时间$(s, s+t]$内正常服务意味着

$$\forall a \in (s, s+t], b \in (a, s+t],$$
$$R_{\rm in}(b) - R_{\rm in}(a) \leqslant r(b-a). \tag{7}$$

为检测节点是否停止服务, 引入如下假设:

假设 2 输入过程$R_{\rm in}(s+t)$是严格单调递增的, 即

$$\forall s, t > 0, R_{\rm in}(s+t) - R_{\rm in}(s) > 0. \tag{8}$$

为检测节点是否服务减慢, 引入如下假设:

假设 3 节点服务减慢时, 服务速率远小于数据输入速率, 即节点v在时间$(s, s+t]$内服务减慢意味着存在$r_0 > 0$满足

$$\forall a \in (s, s+t], b \in (a, s+t],$$
$$R_{\rm in}(b) - R_{\rm in}(a) >$$
$$R_{\rm out}(b) - R_{\rm out}(a) + r_0(b-a). \tag{9}$$

为检测节点是否错误输出, 引入如下假设:

假设 4 节点错误输出时, 输出速率远大于数据输入速率, 即节点v在时间$(s, s+t]$内错误输出意味着存在$r_1 > 0$满足

$$\forall a \in (s, s+t], b \in (a, s+t],$$
$$R_{\rm in}(b) - R_{\rm in}(a) + r_1(b-a) <$$
$$R_{\rm out}(b) - R_{\rm out}(a). \tag{10}$$

在假设1和假设2的保证下, 可以得到节点停止服务的检测方法.

定理 1 假设1–2满足时, 若节点v的输入过程$R_{\rm in}(t)$与输出过程$R_{\rm out}(t)$在时刻s不满足式(5), 即$R_{\rm out}(s) < (R_{\rm in} \otimes S)(s)$且有$R_{\rm out}(s+t) - R_{\rm out}(s) = 0$. 则节点$v$在时刻$s$停止服务.

证 若$R_{\rm out}(s) < R_{\rm in}(s) \otimes S(s)$, 则节点的服务能力低于严格服务曲线所描述的下界. 且$R_{\rm out}(s) < R_{\rm in}(s) \otimes S(s) < R_{\rm in}(s)$, 故时刻$s$节点$v$处有数据积压. 但$R_{\rm out}(s+t) - R_{\rm out}(s) = 0$. 则节点$v$未对积压的数据提供服务, 即节点$v$在时刻$s$停止服务.

证毕.

定理 2 假设1–2满足时, 若节点v在时间$(s, s+t]$内停止服务, 则节点v的输入过程$R_{\rm in}(t)$与输出过程$R_{\rm out}(t)$在时间$(s+\tau, s+t]$内不满足式(5).

证 若节点v在时间$(s, s+t]$内停止服务, 则

$$\forall a \in (s+\tau, s+t],\ R_{\rm out}(a) - R_{\rm out}(s) = 0.$$

由假设2可知

$$R_{\rm out}(a) = R_{\rm out}(s) \leqslant R_{\rm in}(s) < R_{\rm in}(a-\tau).$$

又有

$$(R_{\rm in} \otimes S)(a) = \inf_{0 \leqslant p \leqslant a} [R_{\rm in}(p) + S(a-p)],$$
$$S(a-p) = \max(0, r(a-p-\tau)).$$

由假设1可知, $0 \leqslant p < a-\tau$时, 有

$$R_{\rm in}(a-\tau) - R_{\rm in}(p) \leqslant r(a-\tau-p) = S(a-p),$$

即

$$R_{\rm in}(a-\tau) \leqslant R_{\rm in}(p) + S(a-p).$$

又$a-\tau \leqslant p \leqslant a$时, 有$S(a-p) = 0$, 即

$$R_{\rm in}(p) + S(a-p) = R_{\rm in}(p) \geqslant R_{\rm in}(a-\tau).$$

综上, 有$(R_{\rm in} \otimes S)(a) = R_{\rm in}(a-\tau)$. 故$R_{\rm out}(a) < R_{\rm in}(a) \otimes S(a)$, 即不满足式(5). 证毕.

在假设1和假设3的保证下, 可以得到节点服务减慢的检测方法.

定理 3 若节点v的输入过程$R_{\rm in}(t)$与输出过程$R_{\rm out}(t)$在时刻s不满足式(5), 即$R_{\rm out}(s) < R_{\rm in}(s) \otimes S(s)$且有$R_{\rm out}(s+\tau) - R_{\rm out}(s) > 0$. 则节点$v$在时刻$s$服务减慢.

证 若$R_{\rm out}(s) < R_{\rm in}(s) \otimes S(s)$, 则节点的服务能力低于服务曲线所描述的下界. 又知$R_{\rm out}(s+\tau) - R_{\rm out}(s) > 0$, 故节点$v$在时刻$s$仍提供服务, 即节点$v$在时刻$s$服务减慢. 证毕.

由于服务曲线$S(t) = \max(0, r(t-\tau))$中允许存在开始服务的时延, 节点服务减慢的检测可能存在延迟. 若节点v在时间$(s, s+t]$内服务减慢, 假设在服务减慢开始的时刻有$R_{\rm in}(s) = R_{\rm out}(s)$, 且输入速率满足$\forall a \in (s, s+t],\ R_{\rm in}(s+a) - R_{\rm in}(s) = ra$, 服务速率满足$\forall a \in (s, s+t]$,

$$R_{\rm in}(s+a) - R_{\rm in}(s) =$$
$$R_{\rm out}(s+a) - R_{\rm out}(s) + (r_0 + \delta)a,$$

即$\forall a \in (s, s+t]$,

$$R_{\rm out}(s+a) = R_{\rm in}(s+a) - (r_0+\delta)a =$$
$$R_{\rm in}(s) + ra - (r_0+\delta)a,$$

则可检测的时间a_0需满足

$$(R_{\text{in}} \otimes S)(s+a_0) =$$
$$R_{\text{in}}(s+a_0-\tau) =$$
$$R_{\text{in}}(s)+r(a_0-\tau) > R_{\text{out}}(s+a_0) =$$
$$R_{\text{in}}(s)+ra_0-(r_0+\delta)a_0.$$

解得$a_0 > \dfrac{r\tau}{r_0+\delta}$, 令$\delta$趋于0, 此时有$a_0 > \dfrac{r\tau}{r_0}$. 故在该情况下, 检测延迟最长可达$\dfrac{r\tau}{r_0}$.

由于未对节点的服务能力上界做出约束, 节点恢复正常时可能立即处理完所有积压数据. 在该情况下, 若节点在$s+t+\delta$时刻恢复正常, 则此时已满足式(5). 令δ趋于0, 则不满足式(5)的时间$a_0 \leqslant s+t$.

下面的定理证明了对于一般情况, 在时间$(s+\dfrac{r\tau}{r_0}, s+t]$内均可检测节点服务减慢的故障.

定理 4　假设1, 3满足时, 若节点v在时间$(s, s+t]$内服务减慢, 则节点v的输入过程$R_{\text{in}}(t)$与输出过程$R_{\text{out}}(t)$在时间$(s+\dfrac{r\tau}{r_0}, s+t]$内不满足式(5),

证　若节点v在时间$(s, s+t]$内服务减慢, 由假设3, 有

$$\forall a \in (s+\frac{r\tau}{r_0}, s+t],$$
$$R_{\text{out}}(a)-R_{\text{out}}(s) < R_{\text{in}}(a)-R_{\text{in}}(s)-r_0(a-s).$$

又由假设1可知

$$(R_{\text{in}} \otimes S)(a) =$$
$$\inf_{0\leqslant p\leqslant a}[R_{\text{in}}(p)+S(a-p)] =$$
$$R_{\text{in}}(a-\tau).$$

则

$$R_{\text{out}}(a) < R_{\text{in}}(a)-R_{\text{in}}(s)+R_{\text{out}}(s)-r_0(a-s) \leqslant$$
$$R_{\text{in}}(a)-r_0(a-s) \leqslant R_{\text{in}}(a)-r\tau \leqslant R_{\text{in}}(a-\tau).$$

即不满足式(5).　　证毕.

在假设1, 4的保证下, 可以得到节点错误输出的检测方法.

定理 5　若假设1, 4满足时, 若节点v的输入过程$R_{\text{in}}(t)$与输出过程$R_{\text{out}}(t)$在时刻s不满足式(6), 即$R_{\text{out}}(s) > R_{\text{in}}(s)$, 则节点$v$在时刻$s$错误输出.

证　若$R_{\text{out}}(s) > R_{\text{in}}(s)$, 则节点发送的累计数据超过了接收的累计数据, 节点错误输出.

证毕.

由于服务曲线$S(t)=\max(0, r(t-\tau))$中允许存在开始服务的时延, 节点错误输出的检测可能存在延迟. 若节点v在时间$(s, s+t]$内错误输出, 假设在错误输出开始的时刻有$R_{\text{in}}(s) = R_{\text{out}}(s)+r\tau$, 且输出速率满足

$$\forall a \in (s, s+t],$$
$$R_{\text{in}}(s+a)-R_{\text{in}}(s)+(r_1+\delta)a =$$
$$R_{\text{out}}(s+a)-R_{\text{out}}(s),$$

即

$$\forall a \in (s, s+t],$$
$$R_{\text{out}}(s+a) = R_{\text{in}}(s+a)+(r_1+\delta)a-r\tau,$$

则可检测的时间a_0需满足

$$R_{\text{out}}(s+a_0) =$$
$$R_{\text{in}}(s+a_0)+(r_1+\delta)a_0-r\tau > R_{\text{in}}(s+a_0).$$

解得$a_0 > \dfrac{r\tau}{r_1+\delta}$, 令$\delta$趋于0, 此时有$a_0 > \dfrac{r\tau}{r_1}$. 故在该情况下, 检测延迟最长可达$\dfrac{r\tau}{r_1}$.

下面的定理证明了对于一般情况, 在时间$(s+\dfrac{r\tau}{r_1}, s+t]$内均可检测节点错误输出的故障.

定理 6　假设1, 4满足时, 若节点v在时间$(s, s+t]$内错误输出, 则节点v的输入过程$R_{\text{in}}(t)$与输出过程$R_{\text{out}}(t)$在时间$(s+\dfrac{r\tau}{r_1}, s+t]$内不满足式(6).

证　若节点v在时间$(s, s+t]$内错误输出, 由假设4, 有

$$\forall a \in (s+\frac{r\tau}{r_1}, s+t],$$
$$R_{\text{in}}(a)-R_{\text{in}}(s)+r_1(a-s) < R_{\text{out}}(a)-R_{\text{out}}(s).$$

又由假设1可知

$$(R_{\text{in}} \otimes S)(s) =$$
$$\inf_{0\leqslant p\leqslant s}[R_{\text{in}}(p)+S(s-p)] =$$
$$R_{\text{in}}(s-\tau),$$

则

$$R_{\text{out}}(a) > R_{\text{in}}(a)-R_{\text{in}}(s)+R_{\text{out}}(s)+r_1(a-s) \geqslant$$
$$R_{\text{in}}(a)-R_{\text{in}}(s)+R_{\text{in}}(s-\tau)+r_1(a-s) \geqslant R_{\text{in}}(a).$$

不满足式(6).　　证毕.

由定理1–6, 可以在一定的延迟容忍下检测出正常状态的节点进入故障状态的时刻, 并对节点的故障种类做出判断.

在实际系统中, 节点的故障有时是由外界干扰导致的. 在干扰消除后, 节点又会恢复正常. 因此, 除了检测节点发生故障的时间与故障种类外, 在节点进入故障状态后, 还需要判断节点恢复正常的时间.

为判断节点恢复正常的时间, 并对节点进行后续的故障检测, 定义对输入过程, 输出过程的初始化.

定义 3　设在时刻s对输入过程$R_{\text{in}}(t)$和输出过

程$R_{\rm out}(t)$进行初始化, 则有

$$R'_{\rm in}(t)=R_{\rm in}(s+t)-R_{\rm in}(s), \tag{11}$$

$$R'_{\rm out}(t)=R_{\rm out}(s+t)-R_{\rm out}(s). \tag{12}$$

在假设2的保证下, 可以得到节点从停止服务到恢复正常的检测方法.

定理 7 假设2满足时, 若s时刻处于停止服务状态的节点v在$s+t$时刻发送数据, 即$R_{\rm out}(s+t)-R_{\rm out}(s)>0$, 则节点在$s+t$时刻已恢复正常.

证 由于节点在停止服务时不发送数据, 而$R_{\rm out}\ (s+t)-R_{\rm out}(s)>0$意味着节点发送了数据, 故节点在$s+t$时刻已恢复正常. 证毕.

定理 8 假设2满足时, 若节点v在时刻s从停止服务到恢复正常, 且在时间$(s,s+t]$内均正常, 则节点v的输出过程$R_{\rm out}(t)$满足$\forall a\in(s+\tau,s+t]$, $R_{\rm out}(a)-R_{\rm out}(s)>0$.

证 若节点v在时刻s从停止服务到恢复正常, 由于节点仅对时刻时刻s后的数据重新开始服务, 在时刻s对输入过程$R_{\rm in}(t)$和输出过程$R_{\rm out}(t)$进行初始化, 则$\forall a\in(s+\tau,s+t]$,

$$R_{\rm out}(a)-R_{\rm out}(s)=$$
$$R'_{\rm out}(a-s)>(R'_{\rm in}\otimes S)(a-s).$$

由假设2可知

$$R'_{\rm in}(a-\tau-s)=R_{\rm in}(a-\tau)-R_{\rm in}(s)>0.$$

故

$$\forall a\in(s+\tau,s+t],\ R_{\rm out}(a)-R_{\rm out}(s)>0.$$

证毕.

在假设1, 3的保证下, 可以得到节点从服务减慢到恢复正常的检测方法.

定理 9 假设1, 3满足时, 若s时刻处于服务减慢状态的节点 v 满足 $\exists t>0$, $R_{\rm out}(s+t)-R_{\rm out}(s)>rt$, 则节点在$s+t$时刻已恢复正常.

证 若节点在$s+t$时刻仍服务减慢, 由假设1, 3可知

$$R_{\rm out}(s+t)-R_{\rm out}(s)<$$
$$R_{\rm in}(s+t)-R_{\rm in}(s)<rt.$$

故$\exists t>0$, $R_{\rm out}(s+t)-R_{\rm out}(s)>rt$, 则节点在$s+t$时刻已恢复正常. 证毕.

定理 10 假设3满足时, 若节点v在时刻s从服务减慢到恢复正常, 则节点v的输出过程$R_{\rm out}(t)$满足

$$\exists a\in(s,s+\tau],\ b>a,$$
$$R_{\rm out}(b)-R_{\rm out}(a)>r(b-a),\ t>0.$$

证 若节点v在时刻s_0进入服务减慢状态, 在时刻s从服务减慢到恢复正常, 则由假设3可知

$$R_{\rm in}(s)-R_{\rm out}(s)>$$
$$R_{\rm in}(s_0)-R_{\rm out}(s_0)+r_0(s-s_0)\geqslant r_0(s-s_0).$$

故时刻s存在数据积压, 节点恢复正常后, 开始服务的时延不超过τ, 速度不低于r, 故

$$\exists a\in(s,s+\tau],\ b>a,$$
$$R_{\rm out}(b)-R_{\rm out}(a)>r(b-a),\ t>0.$$

证毕.

在假设1的保证下, 可以得到节点从错误输出到恢复正常的检测方法.

定理 11 假设1满足时, 将处于错误输出状态的节点v在s时刻初始化, 若$\exists t>0$, 使得初始化后的输入过程$R'_{\rm in}(t)$和输出过程$R'_{\rm out}(t)$在时间$(s,s+t]$内满足式(6), 则节点在s时刻后已恢复正常.

证 若节点v在时刻s仍处于错误输出状态, 则$\exists a\in(s,s+t]$, $R_{\rm out}(a)-R_{\rm out}(s)>r(a-s)$. 由假设1可知

$$R_{\rm out}(a)-R_{\rm out}(s)>$$
$$r(a-s)>R_{\rm in}(a)-R_{\rm in}(s),$$

即初始化后的输入过程$R'_{\rm in}(t)$和输出过程$R'_{\rm out}(t)$在时刻$a\in(s,s+t]$不满足式(6), 矛盾. 证毕.

定理 12 若节点v在时刻s从错误输出到恢复正常, 且在时间$(s,s+t]$内均正常, 则在s时刻初始化后的输入过程$R'_{\rm in}(t)$和输出过程$R'_{\rm out}(t)$在时间$(s,\ s+t]$内满足式(6).

证 若节点v在时刻s从错误输出到恢复正常, 由于节点仅对时刻时刻s后的数据重新开始服务, 故s时刻初始化后的输入过程$R'_{\rm in}(t)$和输出过程$R'_{\rm out}(t)$在正常时间$(s,\ s+t]$内均满足式(6).

证毕.

综上, 本文已经给出了在网络中每个节点v的输入过程$R_{\rm in}(t)$与输出过程$R_{\rm out}(t)$均可测量, 严格服务曲线$S(t)$已知且假设1–4满足时, 节点的故障检测方法.

此时, 节点停止服务与恢复的检测延迟不超过τ, 节点服务减慢的检测延迟不超过$\dfrac{r\tau}{r_0}$, 速率恢复的检测延迟不超过τ, 节点错误输出的检测延迟不超过$\dfrac{r\tau}{r_1}$, 正确输出的检测延迟不超过检测的最短间隔.

3.2 检测原理假设的可满足性讨论

在此, 集中讨论为了得出上一节中理论结果所做的主要假设在实际系统中可满足性, 及其对故障检测方法的影响, 以便设计符合实际的故障检测算法.

在实际系统中, 部分节点v的输入过程$R_{\rm in}(t)$与输出过程$R_{\rm out}(t)$可能无法测量. 此时, 可以将无法测量输入或输出过程的节点与相邻的节点组成一张子图, 若子图的总输入输出可测量, 则可以对子图内是否有节点发生故障做出判断.

图6中给出了使用子图进行故障检测的一个示例. 若节点v_2的输入与输出过程无法测量, 可将节点v_1, v_2, v_3 看作一个子图, 通过测量节点 v_1 的输入过程$R_{1,\rm in}(t)$和节点v_3的输出过程$R_{3,\rm out}(t)$, 检测子图的故障情况.

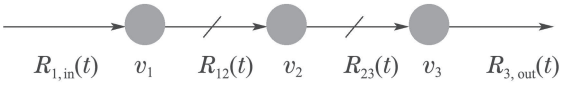

图 6 使用子图进行故障检测

Fig. 6 Fault detection by subgraph

本文假设不会发生子图中的节点故障恰好抵消的情形, 即子图中节点恰好同时发生不同种类的故障, 这些故障导致输入输出过程满足式(5)与式(6), 进而故障无法检测的情况. 这种情况虽然理论上可能发生, 但概率极小.

假设1–4要求输入过程和输出过程均为时间的连续函数. 但在实际系统中, 由于节点接收与发送的数据通常是一个个数据包的形式, 输入过程和输出过程是离散的. 进行故障检测时, 以一个固定的采样周期, 对节点的输入过程和输出过程进行采样, 且采样周期远大于连续发送的两个数据包之间的时间间隔. 因此输入过程和输出过程可以被近似当作连续来处理, 对检测结果不会产生影响.

为进行故障检测, 需要先得到节点或子图的服务曲线. 服务曲线可以从节点的历史输入输出中获得, 但在这里本文使用一种更简单的方法. 为测量节点的最大时延, 只需向节点逐个发送数据包, 在节点完成对一个数据包的服务后再发送下一个. 统计节点发送数据包的时延, 即可得到节点的最大时延τ. 为测量节点的服务最低速率, 向节点提供一个速率较大的输入过程$R_{\rm in}(t)=Vt$, 使节点的输出过程满足$R_{\rm out}(t)<V(t-\tau)$. 统计节点发送数据的速率, 即可得到节点的服务最低速率r.

假设1要求数据输入速率不大于最低服务速率. 但实际系统中, 节点的服务能力通常只保证这个假设在长期成立. 即$\exists a>0, \forall s>a, R_{\rm in}(s)>rs$. 短期来看, 节点的输入过程中, 常常会有突然到达的大量数据引起暂时性的积压, 进而不满足假设1. 假设1不满足时, 节点的各类故障检测与故障结束时的检测均可能发生更大的延迟.

假设2要求数据输入速率不为0. 但实际系统中, 数据通常是间断性输入到节点的, 不可避免地会出现数据输入速率为0的时期. 当数据输入速率为0时, 由于节点正常和停止服务时都无输出, 无法检测节点是否从正常状态进入停止服务状态或从停止服务状态恢复正常. 直到开始有数据输入时才能进行检测. 因此假设2不满足时, 节点停止服务与恢复的检测均可能发生更大的延迟.

假设3要求数据输入速率比节点服务减慢时的服务速率大r_0. 由于数据的间断性输入, 假设3同样无法保证. 当节点正常时, 若数据输入速率小于节点服务减慢时的服务速率, 则数据从到达节点到服务完成的时延不会超过τ, 故障无法检测. 但此时即使节点服务减慢, 也并不会对系统的整体性能造成影响, 因此故障无法检测也是可以接受的. 若数据输入速率大于节点服务减慢时的服务速率, 但差值很小, 由定理4可知检测延迟上界为$\dfrac{r\tau}{r_0}$, 故检测延迟可能很大. 当节点服务减慢时, 若数据输入速率小于节点服务减慢时的服务速率, 则节点无论是否恢复正常, 服务速率都可能不超过r, 无法判断恢复. 为防止系统持续错误报警, 引入一个新的判断规则.

若处于服务减慢状态的节点v在时间$(s, s+T]$内接收到了数据, 即$R_{\rm in}(s+T)-R_{\rm in}(s)>0$, 且满足式(5). 则认为节点$v$在$s+T$时刻已恢复正常. 其中$T>\tau$, 该规则的含义是如果系统连续收到数据且正常工作了时间T, 则认为系统可能已恢复正常. 具体的T值需要根据系统的实际情况选取, 较大的T值会导致检测时延较长, 较小的T值可能会导致误判. 但即使发生了误判, 当数据输入速率大于节点服务减慢时的服务速率后, 系统也能重新检测出故障.

假设4要求错误输出时的数据输出速率比数据输入速率大r_1. 由于数据存在突发到达, 假设4无法保证. 当节点处于正常状态时, 若数据输入速率大于错误输出时的数据输出速率, 无法检测节点是否进入错误输出状态. 直到数据输入速率小于错误输出速率时才能进行检测. 若数据输入速率小于错误输出时的数据输出速率, 但差值很小, 由定理6可知检测延迟上界为$\dfrac{r\tau}{r_1}$, 故检测延迟可能很大. 当节点处于错误输出状态时, 若数据输入速率大于错误输出时的数据输出速率, 可能会出现误判, 即通过节点在s时刻初始化后的输入输出过程满足式(6)判断节点恢复正常. 因此, 需要引入新的判断规则.

将处于错误输出状态的节点v在s时刻初始化, 若$\forall t\in(s, s+T]$, 初始化后的输入过程$R'_{\rm in}(t)$和输出过程$R'_{\rm out}(t)$满足式(6), 则节点在$s+T$时刻已恢复正常. 该规则的含义是如果系统输出小于输入持续了时间T, 则认为系统可能已恢复正常. 具体的T值需要根

据系统的实际情况选取, 较大的T值会导致检测时延较长, 较小的T值可能会导致误判. 但即使发生了误判, 当数据输入速率小于节点错误输出时的服务速率后, 系统也能重新检测出故障.

经过讨论发现, 引入新的判断规则后, 假设1–4不满足带来的主要影响是检测时延变长.

3.3 故障检测算法

本节基于检测原理和假设在实际当中的可满足性讨论, 设计如下基于网络演算的故障检测算法, 该算法适用于网络中的一个可测量输入输出过程的节点或子图. 在算法中, 输出状态state值表示节点或子图状态, 0为正常, 1为停止服务, 2为服务减慢, 3为错误输出.

算法 1 基于网络演算的3类故障检测算法.

步骤 1 获得采样后的输入过程$R_{\rm in}$, 输出过程$R_{\rm out}$, 服务曲线$S(t)=\max(r(t-\tau),0)$, 采样周期δt, 正常持续上限T.

步骤 2 初始化采样后延迟delay $=\lceil\dfrac{\tau}{\delta t}\rceil$, 采样后正常持续上限$T_0=\lceil\dfrac{T}{\delta t}\rceil$, 延迟输出过程下界lowbound $=0$, 当前时刻$t=0$, 当前状态 state $=0$, 正常持续时间con $=0$.

步骤 3 若$t\leqslant$ delay, 到步骤21.

步骤 4 若$t>$delay, 则state$==0$到步骤5, state $==1$ 到步骤10, state $==2$到步骤11, state $==3$且con $==0$ 到步骤16, state $==3$且con >0 到步骤17.

步骤 5 更新下界 lowbound $=\min($lowbound $+r\delta t, R_{\rm in}[t-$delay$])$.

步骤 6 若 lowbound $>R_{\rm out}[t]$, 且 $R_{\rm out}[t]==R_{\rm out}[t-1]$, 更新状态state $=1$, 到步骤21.

步骤 7 若 lowbound $>R_{\rm out}[t]$, 且 $R_{\rm out}[t]>R_{\rm out}[t-1]$, 更新状态state $=2$, 到步骤21.

步骤 8 若$R_{\rm out}[t]>R_{\rm in}[t]$, 更新状态state $=3$, 到步骤21.

步骤 9 若$R_{\rm out}[t]<R_{\rm in}[t]$且lowbound$<R_{\rm out}[t]$, 到步骤21.

步骤 10 若$R_{\rm out}[t]>R_{\rm out}[t-1]$, 初始化$R_{\rm in}=R_{\rm in}-R_{\rm in}[t], R_{\rm out}=R_{\rm out}-R_{\rm out}[t]$, lowbound $=0$, 更新状态state $=0$, 到步骤21.

步骤 11 更新下界lowbound $=\min($lowbound $+r\delta t, R_{\rm in}[t-$delay$])$.

步骤 12 若$R_{\rm out}[t]-R_{\rm out}[t-1]>r\times$ delay, 更新状态state $=0$, 到步骤21.

步骤 13 若lowbound $<R_{\rm out}[t]$, 到步骤14, 否则重置正常持续时间con $=0$, 到步骤21.

步骤 14 若con $\geqslant T_0$, 更新状态state $=0$, 重置正常持续时间con $=0$, 到步骤21.

步骤 15 若con $<T_0$, 更新正常持续时间con $+=1$, 到步骤21.

步骤 16 测试初始化输入过程, 输出过程$R'_{\rm in}=R_{\rm in}-R_{\rm in}[t], R'_{\rm out}=R_{\rm out}-R_{\rm out}[t]$, lowbound $=0$, con $=1$, 到步骤21.

步骤 17 更新下界lowbound $=\min($lowbound $+r\delta t, R'_{\rm in}[t-$delay$])$.

步骤 18 若$R'_{\rm in}[t]<R'_{\rm out}[t]$, 到步骤19, 否则重置正常持续时间con $=0$, 到步骤21.

步骤 19 若con $\geqslant T_0$, 初始化$R_{\rm in}=R'_{\rm in}, R_{\rm out}=R'_{\rm out}$, 更新状态state $=0$, 重置正常持续时间con $=0$, 到步骤21.

步骤 20 若con $<T_0$, 更新正常持续时间con $+=1$, 到步骤21.

步骤 21 输出当前状态state, 更新时间$t+=1$, 到步骤3.

设网络中节点数为n, 采样后的输入过程$R_{\rm in}$与输出过程$R_{\rm out}$长度为t, 则算法的时间复杂度为$\mathrm{O}(nt)$.

4 仿真算例与对比

4.1 仿真算例

在本节中, 作者使用网络模拟器 NS3 (network simulator 3)搭建仿真算例来验证算法的可行性.

在实际系统中, 对网络中的每个节点而言, 发生故障的时间应当远小于正常工作的时间. 但在仿真算例中, 为了检验故障检测算法的效果, 对网络中的节点v, 假设节点状态正常持续的时间在$[T_{\rm n}-\delta T_{\rm n}, T_{\rm n}+\delta T_{\rm n}]$内均匀分布, 节点故障持续的时间在$[T_{\rm e}-\delta T_{\rm e}, T_{\rm e}+\delta T_{\rm e}]$内均匀分布, 且节点发生各类故障的概率相等. 假设节点间断性地收到数据, 周期为T_1+T_2, 每个周期内的前T_1时间未收到数据, 后T_2时间以固定速率ρ收到数据, 即输入过程为

$$R_{\rm in}(t)=\lfloor\frac{t}{T_1+T_2}\rfloor\rho T_2+\max(0,\rho(t-T_1-(T_1+T_2)\lfloor\frac{t}{T_1+T_2}\rfloor)). \tag{13}$$

实验中, 取 $T_{\rm n}=40\,$s, $\delta T_{\rm n}=10\,$s, $T_{\rm e}=40\,$s, $\delta T_{\rm e}=10\,$s, $T_1=7\,$s, $T_2=7\,$s, $\rho=1\,$Mbit/s. 节点正常时的服务速率为$r_{\rm n}=4\,$Mbit/s, 服务减慢的服务速率为$r_s=0.4\,$Mbit/s, 错误输出时的输出速率为$r_{\rm e}=4\,$Mbit/s. 在故障检测中, 需要对输入输出过程进行采样, 取采样周期为$\delta T=0.1\,$s. 设置节点服务速率下界的估计为$r=2\,$Mbit/s, 时延为$\tau=0.1\,$s. 此处节点错误输出

时的输出速率虽然不大于服务速率, 但大于输入速率ρ, 因此不会出现将正常服务与错误输出状态混淆的情况. 算法中设置节点故障时正常工作时间上限$T = 0.2$ s.

仿真时间共500 s, 即采样后序列长度为5000, 实验结果如下.

如图7–8所示, 在仿真时间500 s内, 节点共发生8次故障, 故障检测过程中未发生漏检和误报.

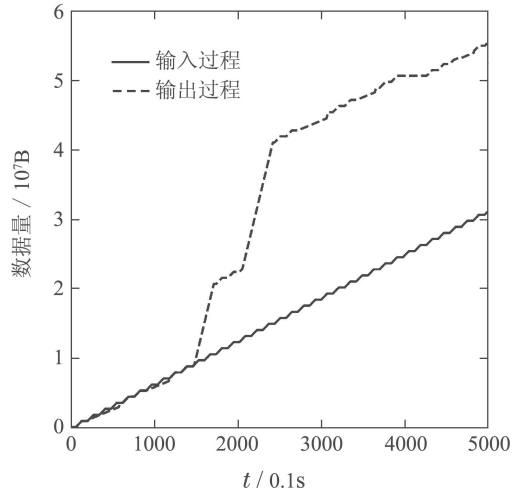

图 7　节点的输入输出过程

Fig. 7　Input process and output process

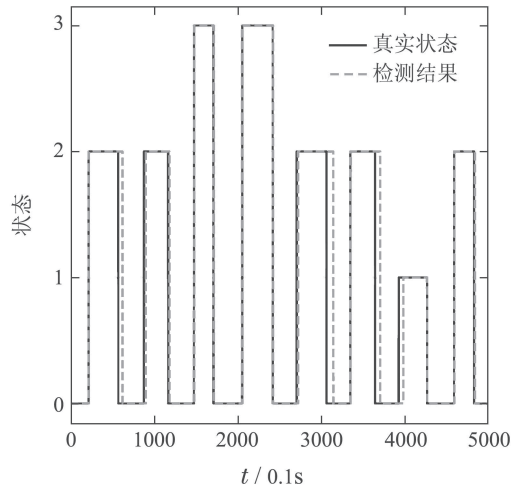

图 8　节点状态及检测结果

Fig. 8　Node states and detection results

从检测延迟来看, 节点停止服务与服务减慢故障的检测出现了一定的延迟. 这是因为节点刚停止服务或服务减慢时, 若未收到数据, 此时无论节点是否发生故障, 都有相同的表现, 从理论上无法检测. 节点从服务减慢状态到恢复正常的检测也出现了延迟, 这也是因为节点刚恢复正常时, 若未收到数据, 则无法判断节点是否恢复, 收到数据后才可判断.

本文分别考虑算法在所有采样点上和收到数据时的检测准确率, 结果如表1所示, 在考虑所有采样点时, 节点正常状态和停止服务状态的检测准确率相对较低.

表 1　检测准确率

Table 1　Detection accuracy

状态	正常	停止服务	服务减慢	错误输出
全部	0.913	0.837	0.969	0.998
部分	0.975	0.986	0.983	0.997

如上所述, 这是由节点未收到数据导致的. 准确率低的另一个原因是在仿真时故障和正常时间均较短, 实际系统中时间较长, 准确率会高很多.

在仅考虑节点收到数据时的检测准确率时, 节点各状态的检测准确率均很高. 事实上, 此时无论是节点发生故障还是恢复正常, 都只有数个采样周期的延迟, 均不超过1 s. 同样, 在实际系统中故障和正常时间较长, 准确率也会有提升.

4.2　效果对比

在本节中, 作者使用时序数据故障检测常用的LSTM与基于网络演算的方法进行对比.

由于不同节点的服务曲线不同, 对网络中的每种节点, 均需要训练一个故障检测模型. LSTM模型的输入为节点的输入过程和输出过程采样后的二维时间序列, 输出为每个时间点上的状态.

在训练时, 由于一次仿真中输入过程和输出过程采样后的长度均很大, 需要对其进行切割. 从每个序列中随机取出一段长度为1500的子序列进行训练, 每个LSTM单元隐层状态的维数为16, 再通过全连接层输出维数为4的预测值, 使用Adam算法进行训练优化. 训练集使用了10次采样后序列长度为5000的仿真数据, 在每个序列上进行了10次采样, 采样后子序列数共 100. 为了防止过拟合, 各仿真数据的 $T_{\mathrm{n}}, \delta T_{\mathrm{n}}, T_{\mathrm{e}}, \delta T_{\mathrm{e}}, T_1, T_2, \rho$值均不同.

在测试的过程中, 将输入拆分成多段长度为1500的子序列输入模型. 由于在子序列的起始点可能会发生误判, 令相邻序列的重合长度为750, 最终结果中, 除第一个子序列外, 只选取每个子序列的后750个结果作为对应时间上的预测结果. 测试集使用了5次采样后序列长度为 5000 的仿真数据. 且各仿真数据的 $T_{\mathrm{n}}, \delta T_{\mathrm{n}}, T_{\mathrm{e}}, \delta T_{\mathrm{e}}, T_1, T_2, \rho$值均在训练集中对应值的上下限范围内.

在每个采样点上, 两种算法的检测准确率如表2所示.

如表2所示, 在各状态下, 基于网络演算的方法的检测准确率均高于LSTM. 且值得注意的是, 在仅考虑节点接收数据的采样点下, 基于网络演算的方法准确率与考虑所有采样点时相比提升了很多. 这是因为节

点未接收数据时, 理论上无法检测停止服务和服务减慢两类故障. 但LSTM在两者上效果相差不大, 这可能是因为LSTM难以学习到“节点未接收数据时前两类故障无法检测”这一特性, 反而对未接收数据的故障状态进行了错误的学习, 进而导致LSTM的检测准确率低.

表 2 检测准确率对比

Table 2 Comparison of detection accuracy

	状态	正常	停止服务	服务减慢	错误输出
网络演算	全部	0.959	0.957	0.923	0.999
	部分	0.980	0.992	0.982	0.999
LSTM	全部	0.836	0.783	0.578	0.945
	部分	0.852	0.768	0.570	0.921

除此之外, 基于网络演算的方法检测耗时略低于LSTM, 且服务曲线获取只需要很简单的操作, 远低于LSTM的训练时长.

5 结论

本文提出了一种基于网络演算的故障检测方法, 并证明了该方法能在确定的时延范围内检测出网络中节点发生故障及恢复正常的时间与故障种类. 仿真算例显示, 该方法能及时检测出节点故障和正常的时间, 且在各采样点上的平均准确率比LSTM高很多.

然而, 该检测方法需要节点为数据流提供的服务曲线满足特定形式. 虽然网络中的大部分节点具有该形式的服务曲线, 但仍存在部分节点的服务曲线不满足该形式. 因此, 希望未来能将该方法扩展到更一般的情况.

参考文献:

[1] DUSIA A, SETHI A S. Recent advances in fault localization in computer networks. *IEEE Communications Surveys & Tutorials*, 2016, 18(4): 3030 – 3051.

[2] LOR K W E. A network diagnostic expert system for Acculink multiplexers based on a general network diagnostic scheme. *Proceedings of the IFIP TC6/WG6.6 Third International Symposium on Integrated Network Management with participation of the IEEE Communications Society CNOM and with support from the Institute for Educational Services*. Washington DC: IEEE, 1993: 659 – 669.

[3] LIU G, MOK A K. An event service framework for distributed real-time systems. *IEEE Workshop on Middleware for Distributed Real-Time Systems and Services*. San Francisco: IEEE, 1997: 1 – 8.

[4] WANG Wei, LU Dongxin, TANG Ying. Management of network barrier based on expert systems. *Computer Engineer and Design*, 2005, 26(11): 173 – 175.
(王伟, 芦东昕, 唐英. 基于专家系统的网络故障管理系统的设计. 计算机工程与设计, 2005, 26(11): 173 – 175.)

[5] DUPUY A, SENGUPTA S, WOLFSON O, et al. Design of the Netmate network management system. *Integrated Network Management*. North-Holland: Elsevier Science Publishers, 1991: 639 – 650.

[6] KATKER S, PATEROK M. Fault isolation and event correlation for integrated fault management. *International Symposium on Integrated Network Management*. Boston: Springer, 1997: 583 – 596.

[7] JIANG Kangming, LIN Bin, QIAO Yan. An efficient fault detection and localization method based on active probing. *Journal of Beijing University of Posts and Telecommunications*, 2012, 35(1): 36 – 40.
(蒋康明, 林斌, 乔焰. 基于主动探测的高效故障检测与定位方法. 北京邮电大学学报, 2012, 35(1): 36 – 40.)

[8] KANDULA S, MAHAJAN R, VERKAIK P, et al. Detailed diagnosis in enterprise networks. *ACM SIGCOMM Computer Communication Review*, 2009, 39(4): 243 – 254.

[9] STEINDER M, SETHI A S. Probabilistic fault diagnosis in communication systems through incremental hypothesis updating. *Computer Networks*, 2004, 45(4): 537 – 562.

[10] BENNACER L, AMIRAT Y, CHIBANI A, et al. Self-diagnosis technique for virtual private networks combining Bayesian networks and case-based reasoning. *IEEE Transactions on Automation Science and Engineering*, 2014, 12(1): 354 – 366.

[11] GARDNER R D, HARLE D A. Alarm correlation and network fault resolution using the Kohonen self-organising map. *IEEE Global Telecommunications Conference*. Phoenix: IEEE, 1997, 3: 1398 – 1402.

[12] SHANG Zhixin, ZHOU Yu, YE Qingwei, et al. Network fault diagnosis based on rough sets and BP neural network. *Journal of Ningbo University(NSSE)*, 2013, 26(2): 45 – 48.
(尚志信, 周宇, 叶庆卫, 等. 基于粗糙集和BP神经网络算法的网络故障诊断模型研究. 宁波大学学报(理工版), 2013, 26(2): 45 – 48.)

[13] KIM J, KIM J, THU H L T, et al. Long short term memory recurrent neural network classifier for intrusion detection. *The 2016 International Conference on Platform Technology and Service (PlatCon)*. Jeju: IEEE, 2016: 1 – 5.

[14] BRAHIMI B, AUBRUN C, RONDEAU E. Network calculus based FDI approach for switched ethernet architecture. *IFAC Proceedings Volumes*, 2006, 39(13): 312 – 317.

[15] LE BOUDEC J Y, THIRAN P. *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. New York: Springer Science & Business Media, 2001.

[16] JIANG Y, LIU Y. *Stochastic Network Calculus*. London: Springer, 2008.

[17] LONG Yanchen, SHEN Haibin, LU Zhonghai. Analysis and evaluation of delay bounds for multiplexing models based on network calculus. *Acta Electronica Sinica*, 2018, 46(8): 1815 – 1821.
(龙彦辰, 沈海斌, 鲁中海. 基于网络演算的聚合模型分析方法及其评估. 电子学报, 2018, 46(8): 1815 – 1821.)

[18] BONDORF S. Better bounds by worse assumptions – improving network calculus accuracy by adding pessimism to the network model. *IEEE International Conference on Communications (ICC)*. Paris: IEEE, 2017: 1 – 7.

[19] BONDORF S, NIKOLAUS P, SCHMITT J B. Catching corner cases in network calculus – flow segregation can improve accuracy. *Measurement, Modelling and Evaluation of Computing Systems (MMB)*. Erlangen: Springer, 2018: 218 – 233.

作者简介:

章子游 博士研究生, 目前研究方向为网络化动态系统故障检测, E-mail: zbw6596@163.com;

赵千川 教授, 第9届“关肇直奖”(2003年)获奖论文作者, 目前研究方向为网络化动态系统性能优化与安全控制及在建筑、能源及通信等领域的应用, E-mail: zhaoqc@tisnghua.edu.cn;

杨 文 工程师, 目前研究方向为工业智能, E-mail: whutyw@126.com.