# 分布式无线传感器网络故障检测算法综述*

徐小龙[1,2]，耿卫建[1]，杨 庚[3]，李玲娟[1]，杨 震[3]
(1. 南京邮电大学 计算机学院，南京 210003；2. 中国科学院软件研究所 信息安全国家重点实验室，北京 100190；3. 宽带无线通信与传感网技术教育部重点实验室，南京 210003)

**摘 要**：阐述了传感器网络中节点发生故障的原因并建立了故障模型。将当前主要的分布式节点故障检测算法分成了基于多数投票策略、基于中值策略、基于决策扩散策略、基于加权和基于分簇的算法五大类，详细阐述了分布式无线传感器网络故障检测算法的原理和步骤，并指出了各个算法的优势与不足。最后，对各个算法的性能进行了分析与比较，讨论了算法存在的问题，并指出了进一步的研究方向。

**关键词**：分布式；无线传感器网络；故障检测
**中图分类号**：TP393 **文献标志码**：A **文章编号**：1001-3695(2012)12-4420-06
doi:10.3969/j.issn.1001-3695.2012.12.004

## Survey of distributed fault detection algorithms for wireless sensor networks

XU Xiao-long[1,2], GENG Wei-jian[1], YANG Geng[3], LI Ling-juan[1], YANG Zhen[3]
(1. *College of Computer, Nanjing University of Posts & Telecommunications, Nanjing* 210003, *China*; 2. *State Key Laboratory of Information Security, Institute of Software, Chinese Academy of Sciences, Beijing* 100190, *China*; 3. *Key Laboratory of Broadband Wireless Communication & Sensor Network Technology of Ministry of Education, Nanjing* 210003, *China*)

**Abstract**: This paper introduced the reasons of WSN node faults, established the fault models and divided the existing popular distributed fault detection algorithms into five types: the majority voting based algorithm, the middle value based algorithm, the decision diffused based algorithm, the weight based algorithm and the cluster based algorithm. It described the basic principle and steps of the algorithms in detail and pointed out the advantages and disadvantages of each algorithm. In the end, it analyzed and compared the performance of each algorithm, discussed the existing algorithms' problems, and specified the further research direction.
**Key words**: distributed; wireless sensor network; fault detection

无线传感器网络是由部署在监测区域内大量的传感器节点组成，通过无线通信方式形成的一个多跳自组织网络[1]。无线传感器网络可用于环境的监测与保护、医疗护理、目标跟踪及军事侦测等领域[2~4]。无线传感器网络中的故障检测是指当传感器节点发生故障时，节点所采集到的数据可能是不正确的，从而导致无线传感器网络对监测信息的错误判断，特别是在检测精确度要求较高的应用领域，信息的准确性是至关重要的，所以对无线传感器网络的故障检测是必不可少的，也是值得研究的。所谓分布式[5~8]是指算法在每个节点上执行，每个节点在本地就可判断出节点自身的故障状态，不需要将所有节点的信息集中到某一个节点上判断。本文详细讨论了现有的分布式无线传感器网络故障检测算法的基本原理，在归纳总结的基础上提出了需要完善的地方，使得算法能够适用的范围更广，算法的故障检测能力更强，能量消耗更低。

## 1 相关模型

### 1.1 故障模型

由于传感器网络自身存在以下的特点：

a) 自身所携带的能量是有限的[9]。大部分节点都是有源的，由于节点之间频繁通信导致网络消耗大量的能量，而节点可能部署在恶劣的环境中(如在高山或沙漠中)，不能更换电池或充电，整个传感器网络工作一段时间后，就会由于能量的逐渐耗尽而导致节点电气特性的不稳定，可能会对采集的数据造成影响，使得采集的数据与真实的数据不相符。

b) 由于节点自身的物理特点(如节点廉价，使用的物理材料较差)及所处的工作环境(如在恶劣环境中)，节点可能会很容易发生硬件故障，但此类故障往往会影响节点之间的通信，或者直接使得整个节点失效。

c) 节点可能会由于软件出现某些漏洞或者受到周围环境的影响发生软件故障，造成节点采集到的数据不正确，或者采集到了正确的数据，但由于传输错误，造成其他节点接收到的数据不正确等。

d) 可能由于节点的信号强度较弱、天气因素或者信号遇到障碍物等原因，造成节点发送数据时一部分数据丢失，或者发送的数据不正确等通信故障。

e) 由于传感器网络中的节点通常直接部署在暴露的环境

**收稿日期**：2012-05-02；**修回日期**：2012-06-15 **基金项目**：国家“973”重点计划资助项目(2011CB302903)；国家自然科学基金资助项目(60873231)；国家教育部高等学校博士学科点专项科研基金资助课题(20093223120001)；中国博士后科学基金资助项目(2011M500095)；江苏省博士后科研资助计划项目(1102103C)；江苏省自然科学基金资助项目(BK2011754，BK2009426)；信息安全国家重点实验室开放课题(03-01-1)

**作者简介**：徐小龙(1977-)，男，江苏盐城人，副教授，博士(后)，主要研究方向为计算机软件、分布式计算、信息安全、agent 技术等(xuxl@njupt.edu.cn)；耿卫建(1987-)，男，江苏连云港人，硕士研究生，主要研究方向为分布式计算、无线传感网技术；杨庚(1961-)，男，江苏盐城人，教授，博导，博士，主要研究方向为计算机应用、网络计算技术、信息安全技术等；李玲娟(1963-)，女，江苏南京人，教授，博士，主要研究方向为网络软件、数据挖掘和物联网技术等；杨震(1961-)，男，教授，博导，博士，主要研究方向为无线通信与网络信号处理、语音处理与现代语音通信、信息安全等.

中,节点可能会轻易地遭受到恶意攻击,节点采集的数据很容易被窃取、窜改,进而使得整个无线传感器网络的信息泄露或不能正常工作。

在分布式的节点故障检测算法中,要求发生故障的节点仍然具有接收、发送、采集及处理数据的能力,只是节点采集的数据是错误的。其中,根据节点所采集数据的异常情况,可将传感器节点可能发生的故障具体分为以下几种类型:

a)固定读数故障,即节点采集到的数据不受环境的影响,一直采集到相同的数据。

b)随机读数故障,即节点采集到的数据是随机的,同样不受环境的影响。

c)瞬时读数故障,即节点在采集数据的过程中,可能由于环境的突然变化或者其他原因造成节点在某一时刻或某几个时刻采集到的数据不正确;但在其他时刻,节点采集到的数据仍然是正确的。

d)偏移读数故障,即节点读数偏离正常值,但仍然随环境的变化而变化。

### 1.2　网络模型

为了方便描述算法,本文假设 $N$ 个传感器节点随机分布在某一特定的区域中,这些传感器节点具有相同的通信半径 $R$,每个传感器节点都可以通过一跳或多跳的方式与其他节点通信。用 $s_i$ 表示无线传感器网络中第 $i$ 个传感器节点,将处在节点 $s_i$ 通信半径内的节点称为节点 $s_i$ 的邻节点(包括节点 $s_i$ 本身),用 num(neighbor($s_i$))表示节点 $s_i$ 邻节点的个数,令 $k$ = num(neighbor($s_i$)),用 neighbor($s_i$)表示节点 $s_i$ 所有邻节点的集合,即 neighbor($s_i$) = $\{x_1, x_2, \cdots, x_i, \cdots, x_k\}$。用 $x_i$ 表示在节点 $s_i$ 所采集到的数据。

## 2　故障检测算法

关于节点故障检测算法的研究已经有很多,大致可分为五类:基于多数投票策略的算法、基于中值策略的算法、基于决策扩散策略的算法、基于加权的算法、基于分簇的算法。五种类型的算法往往不是单独的,而是相互交叉综合运用的。

无线传感网络中的节点所采集的数据具有空间相关性[5~8]与时间相关性[10,11]的特点。所谓的空间相关性,即相邻或相近的传感器节点具有相同或相近的测量值;而时间相关性则是指节点在相近的时刻所采集的数据相同或相近。

算法往往利用无线传感器网络具有空间相关性与时间相关性、而传感器节点所发生的故障是不相关的这一特点对无线传感器网络中节点可能出现的各种故障进行检测。

### 2.1　基于多数投票策略的算法

网络中每个节点首先根据自己的测量值作出决策,然后利用其邻节点的决策来判断自己决策的正确与否,进而确定节点是否发生故障。若大部分邻节点与该节点的决策相同,则认为该节点为正常节点,否则为故障节点。或者节点将其测量值与其邻节点的测量值作比较,大部分的邻节点与该节点的测量值相同或相近,则认为该节点为正常节点,否则为故障节点。如容错的贝叶斯事件检测算法[5]中的最优门限判决策略就是典型的多数投票策略。

基于多数投票策略的算法具有简单、复杂度较低、能耗低的优点,但算法的故障检测性能一般,特别是在节点故障率较高的情况下,算法的检测精确度将会迅速降低,所以算法一般会与其他算法一起使用来提高算法的适应能力,如 MFD 算法[12]是综合利用多数投票策略与决策扩散策略进行故障检测。

#### 2.1.1　容错的贝叶斯事件检测算法

算法用二进制变量表示节点所处的区域以及传感器节点的测量值,用 $T_i = 0$ 表示节点实际处在正常区域中,$T_i = 1$ 表示节点实际处在事件区域中;用 $S_i = 0$ 表示节点测量值正常,用 $S_i = 1$ 表示节点测量值异常。若 $S_i \neq T_i$ 表明节点发生了故障。

假设节点发生故障的概率为 $p$,且 $p$ 是对称不相关的,即

$$P(S_i = 0 \mid T_i = 1) = P(S_i = 1 \mid T_i = 0) = p \tag{1}$$

用 $E_i(a,l)$ 表示节点 $s_i$ 的邻节点中有 $l$ 个节点与节点 $s_i$ 的测量值 $a$ 相同,则

$$P(R_i = a \mid E_i(a,l)) = l/N \tag{2}$$

其中:$R_i$ 表示根据节点 $s_i$ 邻节点的测量值对节点 $s_i$ 评估后的值。则正确的表示节点实际状态的概率为

$$P_{aak} = P(R_i = a \mid S_i = a, E_i(a,l)) = \frac{(1-p)\frac{l}{N}}{(1-p)\frac{l}{N} + p(1-\frac{l}{N})} = \frac{(1-p)l}{(1-p)l + p(N-l)} \tag{3}$$

算法得出了三种策略:

a)随机判决策略。产生一个随机数 $u \in (0,1)$,如果 $u < P_{aak}$,则 $R_i = S_i$,否则 $R_i = \neg S_i$。

b)门限值判决策略。设置一个门限值 $\theta$,如果 $\theta < P_{aak}$,则 $R_i = S_i$,否则 $R_i = \neg S_i$。

c)最优门限判决策略。当 $k_i \geqslant 0.5\text{neighbor}(s_i)$ 时,$R_i = S_i$,否则 $R_i = \neg S_i$。其中,$k_i$ 为与节点 $s_i$ 具有相同测量值的邻节点个数。

理论分析与仿真实验表明,最优门限判决策略在容错、纠错等方面都要优于其他两种策略。

容错的贝叶斯事件检测算法具有简单、能耗低的优点;然而算法假设每个传感器故障的概率相同,并且用二进制变量表示节点所处的区域以及传感器节点的测量值,限制了算法的适用范围,而且在故障率较高的情况下,算法的容错能力将会迅速降低。

#### 2.1.2　MFD 算法

算法首先将节点的初始状态置为 $T_i = 1$,并通过式(4)得到节点 $s_i$ 与其所有邻节点的测试状态 $c_{ij}$:

$$c_{ij} = \begin{cases} 0 & \text{if } (|x_i - x_j| \leqslant \xi) \\ 1 & \text{otherwise} \end{cases} \tag{4}$$

其中:$\xi \geqslant 0$ 表示测量值之间允许的误差范围,$j = 1, \cdots, k$。

对于节点 $s_i$ 的故障状态的判断用式(5)表示:

$$T_i = 0 \quad \text{if}(\sum_{j=1}^{k} c_{ij} < \theta) \tag{5}$$

其中:$T_i = 1$ 表示节点发生故障,$T_i = 0$ 表示节点正常,$\theta$ 的取值一般大于等于 $k/2$。

若节点 $s_i$ 的状态为正常节点,则将其状态扩散到其邻节点,若在其邻节点中存在节点 $s_j$ 使得 $c_{ij} = 0$,则 $T_j = 0$。对于剩下未处理的节点 $s_m$,若 $c_{im} = 0$,$T_i = 0$ 并且 $T_m = 1$,则 $T_m = 0$,否则令 $T_m = 1$。

MFD 算法首先利用节点与其邻节点测量值的差值得到节点的测试状态,再利用贝叶斯算法的最优门限判决策略对节点的故障状态进行判断,最后将已确定故障状态的节点的状态扩散给其邻节点,从而最终确定全网络节点的故障状态。MFD 算法对传感器测量数据的要求不再局限于二进制数据,算法的适用范围较广,而且由于算法结合了扩散策略,使得算法故障

检测能力得到了进一步的提高,但是由于需要将节点的故障状态扩散给其他节点,从而增大了网络的能量开销。

### 2.2 基于中值策略的算法

算法首先将节点 $s_i$ 的邻节点测量值按值大小进行排序,然后取出排好序之后的数据的中间值,若 $k$ 为偶数,取中间两个数的平均值;然后将节点 $s_i$ 的值与其邻节点的中值作比较,若差值较大,则认为节点 $s_i$ 发生了故障,否则为正常节点。典型的有 LEBDF 算法[6]、基于双邻域中值的算法[13]等。

基于中值策略的算法在很大程度上减小了错误的邻居测量值对测量精度的影响,具有较高的检测精确率和较低的误判率,而且节点的能量消耗也较低。采用中值而不采用平均值的好处为:均值不能很好地代表样本的实际中心,特别是当样本中部分值严重偏离中心值时,平均值和中值相差很大。中值能够较好地反映实际的数值,只要传感器数据一半以上是正常的,故障节点对中值的影响就会很小。

#### 2.2.1 LEBDF 算法

节点故障状态的判断如下:

若节点 $s_i$ 的邻节点测量值排序后的数据为 $x_1^i \leqslant x_2^i \leqslant \cdots \leqslant x_i^i \leqslant \cdots \leqslant x_k^i$,则根据式(6)求出节点 $s_i$ 的中值 $\mathrm{med}_i$:

$$\mathrm{med}_i = \begin{cases} x^i_{(k+1)/2} & k \text{ 为奇数} \\ (x^i_{(k/2)} + x^i_{(k/2+1)})/2 & k \text{ 为偶数} \end{cases} \tag{6}$$

计算节点 $s_i$ 与其邻域内节点测量值的差值 $d_i$:

$$d_i = |x_i - \mathrm{med}_i| \tag{7}$$

$N^*(s_i)$ 表示包含 $n$ 个节点的一个区域(包含节点 $s_i$)的节点集合。$D = \{d_1, \cdots, d_n\}$,若 $d_i$ 的值为极值点,则说明节点 $s_i$ 为故障节点,节点状态由下式判断:

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} d_i \tag{8}$$

$$\hat{\sigma} = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(d_i - \hat{\mu})^2} \tag{9}$$

将集合 $D$ 标准化:

$$y_i = \frac{d_i - \hat{\mu}}{\hat{\sigma}} \tag{10}$$

若 $|y_i| \geqslant \theta$,则认为 $s_i$ 为故障节点,否则为正常节点,其中 $\theta > 1$。

LEBDF 算法是典型的基于中值策略的算法,它耗能低,在节点故障率低时,故障检测性能较好,而且算法的适用性也较强;但在节点故障率较高的情况下,算法的故障检测性能将会急剧下降。

#### 2.2.2 基于双邻域的中值算法

算法利用节点的邻域及邻域的邻域中值对事件区域进行节点故障检测,采用了符合实际事件驱动的算法模型,模型的实用性很强。此外,算法利用的邻节点的信息较多,特别是在邻节点较少的情况下,算法的适用性依然很强。

假设 $x_{ij}$ 表示节点 $s_i$ 邻节点中数据排序后,排行第 $j$ 个,则节点 $s_i$ 所有单邻域内的中值为

$$\mathrm{med}(s_i) = \begin{cases} x_{i\frac{k+1}{2}} & \text{if } (k \text{ is odd}) \\ (x_{i\frac{k}{2}} + x_{i\frac{k+2}{2}})/2 & \text{if } (k \text{ is even}) \end{cases} \tag{11}$$

根据单邻域获取的中值,计算双邻域中值,即

$$\mathrm{med\_}D(s_i) = \begin{cases} \mathrm{med}(s_{i\frac{Ni+1}{2}}) & \text{if}(k \text{ is odd}) \\ [\mathrm{med}(s_{i\frac{Ni}{2}}) + \mathrm{med}(s_{i\frac{Ni+2}{2}})]/2 & \text{if}(k \text{ is even}) \end{cases} \tag{12}$$

计算各传感器节点 $s_i$ 的测量值 $x_i$ 与其双邻域中值 med_$D$($s_i$)的差值

$$g_i = |x_i - \mathrm{med\_}D(s_i)| \tag{13}$$

将式(13)标准化处理:

$$\begin{cases} \hat{\mu} = \dfrac{1}{n}\sum_{i=1}^{n} g_i \\ \hat{\sigma} = \sqrt{\dfrac{1}{n-1}\sum_{i=1}^{n}(g_i - \hat{\mu})^2} \end{cases} \tag{14}$$

$$y_i = \frac{g_i - \hat{\mu}}{\hat{\sigma}} \tag{15}$$

若 $y_i \geqslant \theta$,则说明节点 $s_i$ 为故障节点,其测量值是错误的,需要纠正;否则 $s_i$ 为正常节点。

基于双邻域的中值算法是在 LEBDF 算法基础上改进而来的,改进后的算法的故障检测性能得到了进一步的提高,但由于算法不仅需要节点单邻域的中值,还需要双邻域的中值,不仅增大了算法的复杂度,而且增加了网络的能量消耗。

### 2.3 基于决策扩散策略的算法

算法一般使用某种策略首先确定某些节点的故障状态,然后将这些节点的测量值与其邻居节点比较,若两者的值相近,则将其邻节点的状态设定为该节点的故障状态。算法通过节点状态扩散的方式确定网络中每一个节点的故障状态,如 FD[14]、DFD[15]、改进的 DFD[16]、MDF 等算法。

基于决策扩散策略的算法具有较好的检测性能,即使在节点故障率较高的情况下,算法仍然具有较高的检测精确度与较低的误判率,但是算法在决策的扩散中需要增加额外的通信量,造成较大的能量消耗。

#### 2.3.1 FD 算法

算法利用连续 $q$ 个时刻的邻节点的测量值来实现对瞬时测量值故障的容错,并根据节点在相同时刻的差值信息构建矩阵 $\boldsymbol{M}$,通过矩阵 $\boldsymbol{M}$ 确定节点间的测试状态 $c_{ij}$,进而确定节点的最终状态,并把正常节点的状态扩散到邻居节点。

首先将节点的初始状态置为 $F_i = 1$,通过式(16)初始化矩阵 $\boldsymbol{M} = [m_{jk}]$ 得到 $m_{jk}$ 的值与节点 $s_i$ 与其所有邻节点的测试状态 $c_{ij}$。

$$m_{jt} = \begin{cases} 0 & \text{if}(|x_i^t - x_j^t| \leqslant \xi) \\ 1 & \text{otherwise} \end{cases} \tag{16}$$

其中:$t = 1, \cdots, q$;$\xi \geqslant 0$ 表示测量值之间允许的误差范围;$j = 1, \cdots, k$。

$$c_{ij} = \begin{cases} 0 & \text{if}(\sum_{t=1}^{q} m_{jk} \leqslant q - \theta_2) \\ 1 & \text{otherwise} \end{cases} \tag{17}$$

对于节点 $s_i$ 的故障状态的判断用式(18)表示为

$$F_i = 0 \quad \text{if}(\sum_{j=1}^{k}(1 - c_{ij}) \geqslant \theta_1) \tag{18}$$

其中:$F_i = 1$ 表示节点发生故障,$F_i = 0$ 表示节点正常;$\theta_1 \geqslant 0$。

若节点 $s_i$ 的状态为正常节点,则将其状态扩散到其邻节点,若在其邻节点中存在节点 $s_j$ 使得 $c_{ij} = 0$,则 $T_j = 0$。对于剩下未处理的节点 $s_m$,若 $c_{im} = 0$,$F_i = 0$ 并且 $F_m = 1$,则 $F_m = 0$,否则令 $F_m = 1$。

FD 算法采用了时空相关性的原理,实现了对瞬时测量值故障的检测,使得算法的适用范围更广,故障检测能力更强。然而算法在确定节点的故障状态后,还需要将其状态扩散给其邻节点,增大了网络的能量消耗。

#### 2.3.2 DFD 算法

算法采用相邻节点互相测试的方法来确定节点的初步状

态,再根据节点的初步状态与相邻节点互相测试的结果进一步确定节点的最终状态,并将结果扩散到其他节点。

若 $x_j^t$ 表示节点 $s_i$ 的邻节点 $s_j$ 在 $t$ 时刻采集到的数据($j=1,\cdots,k$),$x_i^t$ 表示节点 $s_i$ 在 $t$ 时刻采集到的数据。根据空间相关性原理,利用式(19)(20)可得到节点 $s_i$ 与其所有邻节点的测试结果 $C_{ij}$。

$$\begin{cases} d_{ij}^t = x_i^t - x_j^t \\ d_{ij}^{t+1} = x_i^{t+1} - x_j^{t+1} \\ \Delta d_{ij}^t = d_{ij}^t - d_{ij}^{t+1} \end{cases} \tag{19}$$

$$C_{ij} = \begin{cases} 1 & \text{if}(d_{ij}^t > \theta_1 \text{ or } \Delta d_{ij}^t > \theta_2) \\ 0 & \text{otherwise} \end{cases} \tag{20}$$

用 $h_i^t$ 表示节点 $s_i$ 与其邻节点的测试值 $C_{ij}$ 等于 1 的个数。利用测试结果,可初步确定节点 $s_i$ 的状态 $NS_i$。

$$NS_i = \begin{cases} \text{LT} & \text{if}(h_i^t > \text{num}(\text{neighbor}(s_i))/2) \\ \text{LG} & \text{otherwise} \end{cases} \tag{21}$$

其中:LT 表示可能故障,LG 表示可能正常。

对 neighbor($s_i$)中的任意一个节点 $s_j$ 而言,其实际状态可能正常也可能故障,因此依据测试结果 $C_{ij}$ 得到的 $s_i$ 初步诊断状态也可能是错误的,不能据此断言 $s_i$ 正常或故障。需要进一步判断该节点的最终状态 $\text{FS}_i$:

$$\text{FS}_i = \text{GD if}((g_i^t - y_i^t) \geqslant \text{num}(\text{neighbor}(s_i))/2) \tag{22}$$

其中:$g_i^t$ 表示节点 $s_i$ 与其初始状态为 LG 的邻节点的测试值 $C_{ij}$ 等于 0 的个数,$y_i^t$ 则表示节点 $s_i$ 与其初始状态为 LG 的邻节点的测试值 $C_{ij}$ 等于 1 的个数。

若节点 $s_j$ 的状态为 LT 或者 LG,在其邻节点中若存在节点 $s_i$ 的状态为 GD(表示节点正常)且 $C_{ij}=0$,则将节点 $s_i$ 的状态置为 GD,否则置为 FT,表示节点为故障节点,即

$$\text{FS}_i = \begin{cases} \text{GD} & \text{if}(\text{FS}_j = \text{GD and } C_{ij} = 0) \\ \text{FT} & \text{otherwise} \end{cases} \tag{23}$$

DFD 算法利用时空相关性原理首先对节点的初步状态进行判断,然后利用节点的初步状态再进行最终状态的判断,最后将节点的最终状态扩散给其邻节点。显然算法需要节点与其邻居通信三次,才能确定节点的最终状态,所以算法将会消耗大量的能量,但算法具有较高的故障检测能力,即使在故障率较高的情况下仍可适用。

### 2.3.3 DFD 改进算法

DFD 改进算法认为 DFD 算法将节点的最终状态判断为正常节点的条件太苛刻,对节点状态的最终判断条件式(23)进行了修改,进一步提高了算法的性能。改进后的判断条件为

$$\text{FS}_i = \begin{cases} \text{GD} & \text{if}(g_i^t - y_i^t > 0) \\ \text{FT} & \text{otherwise} \end{cases} \tag{24}$$

由改进的 DFD 算法可知,当 neighbor($s_i$)中,初步诊断状态为 LG 的节点个数较少(网络的节点故障率较大)时,改进的 DFD 算法亦能准确地判断节点的状态。改进的 DFD 算法也适用于网络中待诊断节点 $s_i$ 的邻居节点数较少的传感器网络。但改进的 DFD 与 DFD 算法仍然存在网络能量消耗大的问题。

## 2.4 基于加权的算法

算法为每个传感器节点赋予了权值(或可信度),而且权值的大小不是固定不变的,算法一般会将节点的权值随着算法的检测效果动态地变化。当然基于加权的故障检测算法也是和其他算法综合在一起使用的,如与基于中值的算法结合的 WMFDS 算法[17]、与节点平均值结合的 DFDBW 算法[5]、DWF 算法[18]等。

相对于没有加权的算法,在检测精确度与误判率方面,该算法都有很好的检测性能,正是因为节点权值的动态变化,使得算法具有更优的检错性能。而且该算法的能量消耗也不大,但其往往忽视对事件边界节点的处理以及没有考虑到节点可能发生的瞬时故障情况。

### 2.4.1 WMFDS 算法

WMFDS 算法是一种基于加权中值的算法,它首先根据 $\lambda_i$ 与节点 $s_i$ 邻节点的测量值计算出节点 $s_i$ 的加权中值 $\hat{x}_i$。

$$\hat{x}_i = \text{MED}\left\{ \lambda_j \in x_j \,\middle|\, {}^{N}_{j=1} \right\} \tag{25}$$

其中 $\lambda_j \in x_j$ 表示为

$$\lambda_j \in x_j = \overbrace{x_j, x_j, \cdots, x_j}^{\lambda_j \text{次}} \tag{26}$$

MED 表示为

$$\text{MED}\left\{ x_j \,\middle|\, {}^{N}_{j=1} \right\} = \begin{cases} x_{\frac{N+1}{2}} & N \text{ 是奇数} \\ \dfrac{x_{\frac{N}{2}} + x_{\frac{N}{2}+1}}{2} & N \text{ 是偶数} \end{cases} \tag{27}$$

其次根据判别函数 $f(x_i, \hat{x}_i)$ 判断节点 $s_i$ 的状态,即

$$f(x_i, \hat{x}_i) = \begin{cases} 1 & \text{if} \left| \dfrac{x_i - \hat{x}_i}{\hat{x}_i} \right| > \xi \\ 0 & \text{otherwise} \end{cases} \tag{28}$$

若 $f(x_i, \hat{x}_i) = 1$ 则表示节点 $s_i$ 为故障节点,并将其可信度 $\lambda_i$ 减 1,直到减到 0 为止,并通知基站,该节点为故障节点。

该算法与中值算法相比具有较高的检测准确度与较低的误判率,而且只需与其邻节点通信一次,能量消耗不大。

### 2.4.2 DFDBW 算法

DFDBW 算法是一种基于加权平均值的算法,它为无线传感网中的每个传感器节点定义了一个可信度 $\lambda_i$,并根据可信度与节点测量值的乘积的平均值综合判断节点是否发生了故障,定义了判定函数 $f(x_i, \bar{x}_i)$。若均值满足特定的条件,就将 $f(x_i, \bar{x}_i)$ 设置为 1,否则为 0。

假设 $x_j$ 表示传感器节点 $s_i$ 的邻节点 $s_j$ 的测量值,每个节点都有一个可信度 $\lambda_j$ 与之对应($j=1,\cdots,k$),并将 $\lambda_j$ 的初值设置为 $\lambda_{\max}$。

首先算法根据 $\lambda_i$ 与节点 $s_i$ 邻节点的测量值计算出节点 $s_i$ 的平均权值 $\bar{x}_i$:

$$\bar{x}_i = \text{AVG}\left\{ x_j \,\middle|\, {}^{k}_{j=1} \right\} = \frac{\sum_{i=1}^{k} (\lambda_j x_j)}{\sum_{i=1}^{k} \lambda_j} \tag{29}$$

其次根据判别函数 $f(x_i, \bar{x}_i)$ 判断节点 $s_i$ 的状态,即

$$f(x_i, \bar{x}_i) = \begin{cases} 1 & \text{if} \mid x_i - \bar{x}_i \mid > \delta \\ 0 & \text{otherwise} \end{cases} \tag{30}$$

若 $f(x_i, \bar{x}_i) = 1$ 则表示节点 $s_i$ 为故障节点,并将其可信度 $\lambda_i$ 减 1,直到减到 0 为止,并通知基站,该节点为故障节点。其中参数 $\delta$ 是可由不同的应用场合定义的正整数。

与 WMFDS 算法相比,DFDBW 算法会略逊一筹,因为采用中值能够较好地反映实际的数值,不过两者在能量的消耗方面都很低。

## 2.5 基于分簇的算法

基于分簇的算法[19~21]首先将无线传感器网络中的所有节点划分为簇,然后利用空间或时间相关性对簇内的节点进行故障检测。由于算法可能不需要网络中所有的节点与其邻节点

进行通信,在一定程度上节约了网络的能量,而且算法的故障检测性能也较好。

刘凯等人[20]提出了一种基于分簇的故障检测算法。算法通过 LEACH 协议[22]对传感器节点进行分簇,然后利用 DFD 算法对节点进行故障检测。算法可按以下步骤进行:

a)簇头的选举。无线传感器网络中每个节点随机地产生 0 ~ 1 之间的随机数,如果节点产生的随机数小于阈值 $T(n)$,则该节点就成为簇头节点,其中 $T(n)$ 的值由式(31)决定。

$$T(n)=\begin{cases}\dfrac{p}{1-p(r \bmod \dfrac{1}{p})} & n\in G\\ 0 & \text{otherwise}\end{cases} \tag{31}$$

其中:$p$ 表示网络中簇头所占的比例;$r$ 表示当前进行的轮数;$G$ 表示在过去$\frac{1}{p}$轮中没有成为簇头的节点并且在前轮故障检测中确定为正常节点的集合。

b)分簇。簇头向其他节点广播信息,告知它们簇头节点的信息,其他节点接收到广播信息后,会根据广播信号的强度决定加入哪个簇,并把该信息发送给簇头节点。

c)计算簇头节点与该簇内所有节点的数据差值 $d_{ij}^t$,若 $d_{ij}^t>\theta_1$,则 $C_{ij}=1$,然后计算下一个节点;若 $d_{ij}^t\leqslant\theta_1$,计算 $\Delta d_{ij}^t$,若 $\Delta d_{ij}^t>\theta_2$,则 $C_{ij}=1$,然后计算下一个节点,直到该簇内所有的节点都计算完毕。

d)选择合适的阈值对簇内所有终端节点的故障状态进行判断。

e)对剩下的没有经过检测的簇进行检测,直到完成网络中所有簇的检测。

f)为了达到节点能量均匀消耗的效果,避免簇头节点因频繁通信造成能量消耗过多而过早死亡的热点效应,在网络经过稳定工作阶段后,算法将重新对网络中所有的节点进行分簇。

与 DFD 算法相比,算法具有较低的能耗与较高的检测精确度,因为算法在故障检测过程中只需簇头节点与簇内的其他节点交换数据,而在 DFD 算法中,每个节点都需要与其邻节点通信,进而消耗大量的能量。

为了防止网络中出现节点能量消耗不均的问题,算法需要不时地更换簇头节点,从而造成额外的开销。而且算法还需要保证簇头节点为正常节点,否则算法会对簇内的其他正常节点造成误判。

## 3 算法比较分析

### 3.1 检测性能

基于多数投票策略的算法的检测性能较差,特别是在节点概率较高时,算法的检测性能将会很低,算法适合在节点故障概率较低时使用。基于决策扩散策略的算法具有很好的检测性能,因为算法首先根据某些方法将部分正常的节点(或者故障节点)确定下来,然后将其状态传递给其邻近的节点,若两者的测量值相差不大,则认为其邻节点与该节点的状态一致。其他算法的检测性能基本上位于两者之间。基于加权的算法要优于基于决策扩散策略的算法与基于中值策略的算法,基于分簇的算法和基于决策扩散策略的算法的性能相当。一般情况下,综合性较强的算法与时间复杂度较高的算法所具有的检测性能也较高。

### 3.2 能量消耗

由于传感器节点的相当一部分能量主要消耗在与其他节点的通信上[23],一般认为,算法要求节点的通信次数越多,算法所消耗的能量就越大。在故障检测算法中,基于决策扩散策略的算法的能耗是最高的,因为算法至少需要两次节点与其邻节点的通信才能确定节点的最终状态;而在大部分的其他算法中,节点只需与其邻节点通信一次便可确定节点的故障状态。

### 3.3 时间复杂度

基于多数投票策略的算法的时间复杂度最低,基于决策扩散策略的算法时间复杂度最高。这是由于基于多数投票策略的算法仅依靠邻节点的状态信息就可判断出节点自身的状态;而基于扩散策略的故障检测算法一般要首先进行节点状态的初步判断,然后再根据初步判断的结果对节点的最终状态进行进一步的判断。一般而言,综合性较强的算法,时间复杂度也较高。

### 3.4 适用性

大部分的算法考虑节点故障类型不足,如节点可能发生的瞬时测量值故障,只有少数算法考虑了此类故障,如 FD 算法。此外,大部分算法忽略了对事件边界节点的检测,认为事件边界节点占少数而不予以考虑,但在事件发生范围较大时,处在事件边界的节点数目也是相当可观的。

各个算法的性能比较如表 1 所示。

表 1 算法性能总结与比较

| 算法 | 能耗 | 检测性能 | 时间复杂度 | 适用性 | 基本原理 |
|---|---|---|---|---|---|
| 贝叶斯 | 低 | 低 | $O(n)$ | 低 | 空间相关 |
| MDF | 中 | 中 | $O(n^2)$ | 中 | 空间相关 |
| LEBDF | 低 | 中 | $O(n^2)$ | 中 | 空间相关 |
| 双邻域中值 | 高 | 高 | $O(n^2)$ | 高 | 空间相关 |
| FD | 中 | 中 | $O(n^2)$ | 高 | 时空相关 |
| DFD | 高 | 高 | $O(n^2)$ | 高 | 时空相关 |
| 改进 DFD | 高 | 高 | $O(n^2)$ | 高 | 时空相关 |
| WMFDS | 低 | 中 | $O(n)$ | 中 | 空间相关 |
| DFDBW | 低 | 低 | $O(n)$ | 中 | 空间相关 |
| 分簇 | 高 | 高 | $O(n^2)$ | 高 | 时空相关 |

## 4 结束语

本文分析了现有的分布式无线传感器网络故障检测算法,针对各种算法分别从它的基本思想、算法原理及主要步骤几个方面进行了论述与分析。它们主要的不同之处在于算法基于的思想不同、适用的场合不同,某些算法可能在节点故障率高的情况下仍然具有较好的性能。它们都是基于无线传感器网络具有的空间相关性或时间相关性对节点故障进行检测的,然而大部分的算法都忽视了事件边界节点的故障检测,因为此类节点不具有空间相关性的特点;而且还存在检测性能好的算法往往需要消耗较多能量的问题。算法考虑的传感器故障类型也不全面,从而导致当某一种特定类型的故障增多时,算法的检测能力将会下降,如很多算法没有考虑瞬时测量值故障。所以在进一步的研究中,需要考虑如何在较低能耗的基础上,全面地考虑节点可能发生的各种故障,使得算法不仅在事件区域或正常区域中具有较好的检测性能,而且在事件边界区域中仍然具有较好的性能。

**参考文献:**

[1] KAPOOR N, BHATIA N, KUMAR S, *et al*. Wireless sensor networks: a profound technology [J]. International Journal on Computer Science and Technology, 2011, 2(2): 211-215.

[2] VIJAY G, Ben Ali BDIRA E, IBNKAHLA M. Cognition in wireless sensor networks: a perspective [J]. IEEE Sensors Journal, 2011, 11(3): 582-592.

[3] SINGHAL S, GANKOTIYA A K, AGARWAL S, *et al*. An investigation of wireless sensor network: a distributed approach in smart environment[C]//Proc of the 2nd International Conference on Advanced Computing & Communication Technologies. [S. l.]: IEEE Press, 2012:522-529.

[4] 任丰原,黄海宁,林闯. 无线传感器网络[J]. 软件学报, 2003,14(7):1282-1291.

[5] KRISHNAMACHARI B, IYENGAR S. Distributed Bayesian algorithms for fault-tolerant event region detection in wireless sensor networks [J]. IEEE Trans on Computers, 2004, 53(3): 241-250.

[6] DING Min, CHEN De-chang, XING Kai, *et al*. Localized fault-tolerant event boundary detection in sensor networks [C]//Proc of IEEE INFOCOM. Washington DC: IEEE Computer Society, 2005: 902-913.

[7] JI Sai, YUAN Shen-fang, MA Ting-huai, *et al*. Distributed fault detection for wireless sensor based on weighted average[C]//Proc of the 2nd International Conference on Networks Security, Wireless Communications and Trusted Computing. Washington DC: IEEE Computer Society, 2010: 57-60.

[8] BEHNKE R, SALZMANN J, SIMANOWSKI S, *et al*. Efficient localized detection of erroneous nodes (ELDEN)[C]//Proc of the 10th IEEE International Conference on Computer and Information Technology. Washington DC: IEEE Computer Society, 2010: 284-289.

[9] LIU Hai, NAYAK A, STOJMENOVIC I. Fault-tolerant algorithms/protocols in wireless sensor networks[M]//Guide to Wireless Sensor Networks. Berlin: Springer, 2009: 265-295.

[10] YAO Bo, CHEN Qing-chun. On the temporal-spatial correlation based fault-tolerant dynamic event region detection scheme in wireless sensor networks[C]//Proc of the 3rd International Conference on Mobile Ad hoc and Sensor Networks. Berlin: Springer, 2007: 511-523.

[11] 季赛,袁慎芳,吴键,等. 基于时空特性的无线传感器网络节点故障诊断方法[J]. 传感器与微系统,2009,28(10):117-120.

[12] 季赛,袁慎芳,马廷淮,等. 无线传感器网络中节点故障诊断方法的研究[J]. 计算机工程与应用,2010,46(23):95-97.

[13] 李平,李宏,吴敏. WSNs 分布式事件区域容错算法[J]. 计算机工程,2009,35(14):142-145.

[14] LEE M H, CHOI Y H. Fault detection of wireless sensor networks [J]. Computer Communications, 2008, 31(14): 3469-3475.

[15] CHEN Jin-ran, KHER S, SOMANI A. Distributed fault detection of wireless sensor networks[C]//Proc of ACM International Conference on Mobile Computing and Networking. New York: ACM Press, 2006: 65-72.

[16] JIANG Peng. A new method for node fault detection in wireless sensor networks[J]. Sensors, 2009, 9(2): 1282- 1294.

[17] 高建良,徐勇军,李晓维. 基于加权中值的分布式传感器网络故障检测[J]. 软件学报,2007,18(5):1208-1217.

[18] 李宏,谢政,陈建二,等. 一种无线传感器网络分布式加权容错检测算法[J]. 系统仿真学报,2008,20(14):3750-3755.

[19] AKBARI A, BEIKMAHDAVI N, KHOSROZADEH A, *et al*. A survey cluster-based and cellular approach to fault detection and recovery in wireless sensor networks [J]. World Applied Sciences Journal, 2010, 8(1): 76-85.

[20] 刘凯,彭力. 分簇式无线传感器网络节点故障诊断算法研究[J]. 传感器与微系统,2011,30(4):37-41.

[21] VENKATARAMAN G, EMMANUEL S, THAMBIPILLAI S. Energy-efficient cluster-based scheme for failure management in sensor networks [J]. IET Communications, 2008, 2(4): 528-537.

[22] HEINZELMAN W, CHANDRAKASAN A, BALAKRISHNAN H. Energy-efficient communication protocol for wireless microsensor networks[C]//Proc of the 33rd Hawaii International Conference on System Sciences. Washington DC: IEEE Computer Society, 2000: 1-10.

[23] MADDEN S, FRANKLIN M J, HELLERSTEIN J M, *et al*. TAG: a tiny aggregation service for Ad hoc sensor networks[C]//Proc of the 5th Symposium on Operating Systems Design and Implementation. New York: ACM Press, 2002: 131-146.

(上接第 4406 页)

[32] PAN Xing-hao, TAN Jia-qi, KAVULYA S, *et al*. Ganesha: black-box diagnosis of MapReduce systems[J]. ACM SIGMETRICS Performance Evaluation Review, 2009, 37(3): 8-13.

[33] RINGS T, GRABOWSKI J, SCHULZ S. On the standardization of a testing framework for application deployment on grid and cloud infrastructures[C]//Proc of the 2nd International Conference on Advances in System Testing and Validation Lifecycle. 2010: 99-107.

[34] CHAN W K, MEI Li-jun, ZHANG Zhen-yu. Modeling and testing of cloud applications[C]//Proc of IEEE Asia-Pacific Services Computing Conference. 2009: 111-118.

[35] KING T M, GANTI A S. Migrating autonomic selftesting to the cloud [C]//Proc of the 3rd International Conference on Software Testing, Verification, and Validation. Washington DC: IEEE Computer Society, 2010: 438-443.

[36] MATHEW R, SPRAETZ R. Test automation on a SaaS platform[C]// Proc of International Conference on Software Testing, Verification and Validation. Washington DC: IEEE Computer Society, 2009: 317-325.

[37] GU Lin, CHEUNG S C. Constructing and testing privacy-aware services in a cloud computing environment: challenges and opportunities[C]//Proc of the 1st Asia-Pacific Symposium on Internetware. New York: ACM Press, 2009: 1-10.

[38] DU Juan, WEI Wei, GU Xiao-hui, *et al*. Runtest: assuring integrity of dataflow processing in cloud computing infrastructures[C]//Proc of the 5th ACM Symposium on Information, Computer and Communications Security. New York: ACM Press, 2010: 293-304.

[39] MENZEL M, WARSCHOFSKY R, THOMAS I, *et al*. The service security lab: a model-driven platform to compose and explore service security in the cloud[C]//Proc of the 6th World Congress on Services. Washington DC: IEEE Computer Society, 2010: 115-122.

[40] RIUNGU L M, TAIPALE O, SMOLANDER K. Research issues for software testing in the cloud[C]//Proc of the 2nd IEEE International Conference on Cloud Computing Technology and Science. 2010: 557-564.

[41] PARVEEN T, TILLEY S. When to migrate software testing to the cloud[C]//Proc of the 3rd International Conference on Software Testing, Verification and Validation. Washington DC: IEEE Computer Society, 2010: 424-427.

[42] RIUNGU L M, TAIPALE O, SMOLANDER K. Software testing as an online service: observations from practice[C]//Proc of the 3rd International Conference on Software Testing, Verification and Validation. Washington DC: IEEE Computer Society, 2010: 418-423.

[43] Cloud testing[EB/OL]. [2010-04]. http://en. wikipedia. org/wiki/Cloud_testing.

[44] Amazon Web services[EB/OL]. http://aws. amazon. com.

[45] Apache Hadoop[EB/OL]. http://hadoop. apache. org/core.

[46] DEAN J, GHEMAWAT S. MapReduce: simplied data processing on large clusters[C]//Proc of the 6th Symposium on Operating Systems Design and Implementation. 2004: 137-150.

[47] Apache Hbase[EB/OL]. http://hadoop. apache. org/hbase.

[48] Apche Zookeeper[EB/OL]. http://hadoop. apache. org/zookeeper.

[49] Google App Engine[EB/OL]. http://appengine. google. com.

[50] Microsoft Azure[EB/OL]. http://www. microsoft. com/azure.

[51] Salesforce. com[EB/OL]. http://www. force. com.

[52] 冯登国,张敏,张妍,等. 云计算安全研究[J]. 软件学报,2011,22(1):71-83.