文章编号:1007-5321(2019)03-0037-06 DOI:10.13190/j.jbupt.2018-114

基于任务转移概率的感知节点异常运行状态检测方法

马峻岩, 张 特, 王 瑾
(长安大学 信息工程学院, 西安 710064)

摘要: 针对无线传感器网络中感知节点异常状态检测困难问题,提出了一种基于感知节点任务转移概率的节点状态特征描述方式,利用该特征判断感知节点的运行状态,实现节点异常检测. 基于任务转移概率的异常检测方法(T2PAD),根据感知节点运行任务的一步转移概率特征,对节点的运行状态进行分析,通过对转移概率向量相似性进行异常检测,识别出导致异常的任务,缩小并定位异常范围,为修正异常提供依据. 传感器网络开源代码库中的缺陷实例验证了 T2PAD 对于异常检测的有效性.

关 键 词: 无线传感器网络; 感知节点异常; 转移概率矩阵; 节点状态特征
中图分类号: T311 **文献标志码**: A

Task Transition Probability Based Anomaly Detection Method for Sensor Nodes

MA Jun-yan, ZHANG Te, WANG Jin
(School of Information Engineering, Chang'an University, Xi'an 710064, China)

Abstract: The anomaly detection of sensor nodes is a great challenge to wireless sensor networks. A feature based on task transition probability is therefore proposed to model running states of sensor nodes, and the feature can be further used for anomaly detection. Task transition probability based anomaly detection (T2PAD) analyzes states of sensor nodes based on the one-step transition probability of running tasks within the nodes, and then performs anomaly detection by comparing similarities between transition probability vectors. T2PAD can identify tasks that caused the anomaly and narrow down the scope of problematic code, which provides clues to deal with the anomaly. Case studies on defects from a sensor network open source project are carried out to verify the effectiveness of T2PAD.

Key words: wireless sensor network; anomaly of sensor node; transition probability matrix; node state feature

随着越来越多无线传感器网络(WSN, wireless sensor network)应用的部署,感知节点运行状态监测和异常检测成为保障 WSN 应用正常工作、提高 WSN 应用可用性的关键技术之一. WSN 具有应用定制的特性,感知节点的状态与行为在不同应用中具有较大的差异,为感知节点运行时的状态监测和异常检测带来了巨大的挑战. Zhou 等[1]提出对节点程序的执行轨迹进行区间划分,并将区间内指令周期数作为程序动态运行时的特征,应用分类支持向量机,对由瞬态错误导致的异常区间进行检测; Zhou 等[2]从节点执行函数调用轨迹中挖掘并建立工作流模型,将程序运行时的行为与已知工作流模

收稿日期: 2018-06-04
基金项目: 国家自然科学基金项目(61402050, 61303041); 国家重点研发计划项目(2017YFC0804806, 2018YFB1600802)
作者简介: 马峻岩(1982—), 男, 副教授, E-mail: majy@chd.edu.cn.

型对比检测异常;马峻岩等[3]提出了一种基于有限状态机模型断言检查的故障检测算法,并借助轻量级有限状态签名算法检测运行时状态异常变化;Khan 等[4]提出了一种改进的关联规则算法实现日志事件序列频繁判别模式挖掘,并基于挖掘模式实现异常检测;Dong 等[5]应用主成份分析和 t 检验方法对多个时间窗口观测的节点函数调用频数进行统计分析,实现节点的异常检测与诊断;Lu 等[6]以嵌入式系统事件序列与序列事件之间间隔时间为特征,提出了一种硬件辅助的嵌入式异常检测方法.

与现有研究大多从程序函数、事件序列角度出发对感知节点状态进行建模不同,笔者提出的基于任务转移概率的异常检测方法(T2PAD, task transition probability based anomaly detection)从感知节点任务运行的周期性与关联性特征出发,提出了基于节点任务转移概率的运行状态特征描述方法,并基于该特征实现异常检测.

1 感知节点状态模型

感知节点使用的操作系统通常采用事件驱动执行模型和轻量级线程的结构设计,这种设计可以在单栈开销下执行多并发任务. 笔者所述任务对应感知节点中操作系统内核调度器的调度实体. 以 TinyOS 为例对异常检测方法进行阐述,但 T2PAD 同样适用于其他具有类似调度机制的 WSN 操作系统平台.

1.1 任务转移概率

感知节点所有的任务构成一个任务集合. 假设感知节点有 u 个任务,则其任务集合表示为

$$E=\{T_1,T_2,T_3,\cdots,T_u\} \tag{1}$$

定义 1(任务转移) 从当前任务执行结束,到下一任务开始执行的过程构成一次任务转移.

定义 2(转移频数) 转移频数的形式为

$$M\{X(n)=t_{i+n}\mid X=t_i\},t_i,t_{i+n}\in E,n\geqslant 1 \tag{2}$$

在指定的统计窗口内,t_i 为第 i 个被执行的任务. 节点从处于执行任务 t_i 状态,经过 n 次任务转移处于执行任务 t_{i+n}状态的频数称为从 t_i 到 t_{i+1}的 n 步转移频数,其中当 $n=1$ 时称为一步转移频数.

定义 3(转移概率) 转移概率的形式为

$$P\{X(n)=t_{i+n}\mid X=t_i\},t_i,t_{i+n}\in E,n\geqslant 1 \tag{3}$$

在指定的统计窗口内,节点从处于执行任务 t_i 状态,经过 n 次任务转移处于执行任务 t_{i+n}状态的观测概率称为从 t_i 到 t_{i+1}的 n 步转移概率,其中当 $n=1$ 时称为一步转移概率.

1.2 状态模型

为了方便表示,当 $t_i=T_i$, $t_{i+1}=T_j$, M_{ij}表示 T_i 到 T_j 的一步转移频数,即由定义 2 有

$$M_{ij}=M\{X(1)=T_j\mid X=T_i\}$$

P_{ij}表示 T_i 到 T_j 的一步转移概率,由定义 3 有

$$P_{ij}=P\{X(1)=T_j\mid X=T_i\}$$

在指定的统计窗口内,感知节点的状态定义为如下一步任务转移概率矩阵:

$$\boldsymbol{S}=\begin{Bmatrix}P_{11}&P_{12}&P_{13}&\cdots&P_{1u}\\P_{21}&P_{22}&P_{23}&\cdots&P_{2u}\\P_{31}&P_{32}&P_{33}&\cdots&P_{3u}\\\vdots&\vdots&\vdots&\vdots&\vdots\\P_{u1}&P_{u2}&P_{u3}&\cdots&P_{uu}\end{Bmatrix} \tag{4}$$

状态 $\boldsymbol{S}$ 为节点在时间窗口 w 内所有任务的一步转移概率. 当 T_i 到 T_j 之间没有观测到转移时,$P_{ij}=0$.

转移概率 P_{ij}的计算是取一定长度的时间窗口 w,统计该窗口内每个任务 T_i 到任务 T_j 一步转移频数 M_{ij},由式(5)可计算得到转移概率:

$$P_{ij}=\frac{M_{ij}}{M_i},\quad M_i=\sum_{j'=0}^{u}M_{ij'} \tag{5}$$

其中 M_i 为所有从任务 T_i 出发的一步转移频数之和. 通常一个感知节点有数个正常的状态,在不同的情况下节点表现出不同的状态,节点工作的状态序列在时间轴上表现为状态流.

定义 4(状态流) 节点在连续时间窗口下得到的状态序列定义为状态流 Γ, Γ 的形式为

$$\Gamma=\langle\boldsymbol{S}_1,\boldsymbol{S}_2,\boldsymbol{S}_3,\boldsymbol{S}_4,\cdots\boldsymbol{S}_i,\cdots\rangle \tag{6}$$

其中 $\boldsymbol{S}_i$ 表示节点在时间窗口 w_i 的状态.

1.3 状态相似性判定

T2PAD 采用余弦相似度来评估 2 个状态的相似程度. 设状态 $\boldsymbol{S}_1$、$\boldsymbol{S}_2$ 矩阵中 2 个对应的行向量为 $\boldsymbol{V}_i^1$、$\boldsymbol{V}_i^2$,夹角为 θ_i,满足如下关系:

$$\cos\theta_i=\frac{\boldsymbol{V}_i^1\boldsymbol{V}_i^2}{\|\boldsymbol{V}_i^1\|\ \|\boldsymbol{V}_i^2\|} \tag{7}$$

定义集合 φ 为 2 个状态 $\boldsymbol{S}_1$、$\boldsymbol{S}_2$ 中所有对应行向量夹角的集合,即 $\varphi=\{\theta_1,\theta_2,\theta_3,\cdots,\theta_u\}$. 规定对于 $\forall\theta_n>Q,\theta_n\in\varphi$,则认为 $\boldsymbol{S}_1\neq\boldsymbol{S}_2$,即 $\boldsymbol{S}_1$ 与 $\boldsymbol{S}_2$ 不相似,其中 Q 为阈值. 即当 2 个状态中任意一组对应的行向量夹角 $\theta>Q$ 时,T2PAD 判定 2 个状态为不同的状态. Q 的取值由具体的检测需求确定,Q 的值越小,状态分类的粒度越细,对异常的敏感性也越高.

2　任务特征分析

图 1 所示为 TinyOS 某应用中一个任务的一步转移频数变化情况. 可以看出,该任务后会有 7 个不同的任务可能被执行,且分别与这 7 个任务的一步转移概率呈现一种稳定的特征. 这主要由于 WSN 应用多为周期性的重复任务,且任务之间具有一定的相关性,比如完成传感器读取任务后,会执行数据发送任务.

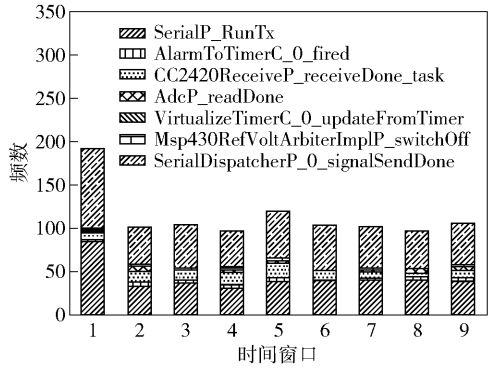

图 1　某任务在不同时间窗口的一步转移频数

为了进一步验证关于感知节点任务转移概率的特征,实验使用 9 个不同规模的 TinyOS 官方应用,构建任务执行序列数据集,并对数据集进行分析得到表 1. 可以看出,其任务执行具有如下特点:1)少量的任务占据了大部分的任务转移;2)系统长时间运行过程中会出现多个状态,通常代码规模越大状态数越多;3)在不同的负载情况下,通过设定一个合适的状态相似度阈值(实验中阈值 $Q=15$),可以将由节点任务执行序列导出的状态流合并成规模较小的有限状态.

表 1　应用任务运行特征

TinyOS 应用	应用代码规模/行	发生大量转移的任务占总任务数量的比例/%	发生大量转移任务的转移数量占总转移数量的比例/%	状态数量	阈值/(°)	任务执行序列数据集规模/条
MultihopOscilloscope	42 021	11.3	92.4	5	15	109 925
RadioSenseToLeds	24 174	25.0	99.9	4	15	202 089
TCPEcho	33 683	17.1	98.6	4	15	92 987
LplBroadcastCountToLeds	23 126	7.8	90.0	3	15	188 224
LplBroadcastPeriodicDelivery	23 172	7.0	99.4	5	15	23 807
RssiToSerial	24 804	12.6	99.0	2	15	2 528 988
TxThroughput	26 686	26.0	98.0	2	15	239 410
RadioCountToLeds1	23 074	16.7	99.9	3	15	143 899
Block	17 581	20.0	99.9	3	15	233 539

上述实验结果表明,感知节点任务转移概率可以作为构建节点运行状态模型的一种特征描述方式. 同时,借助状态相似度度量公式和相似度判别阈值,可以对节点随时间变化的状态特征进行分析.

3　异常状态检测方法

感知节点工作实质是一系列任务的顺序执行. T2PAD 根据指定的时间窗口 w 将节点任务执行序列划分为若干任务执行子序列. 基于任务转移概率,计算出该时间窗口内节点的状态矩阵. 从一系列的时间窗口,T2PAD 得到节点运行的状态流.

T2PAD 通过对节点实际运行状态与正常状态的相似度评估来实现异常状态的检测. 图 2 为 T2PAD 系统结构. 通过部署前的测试构建初始状态表,并以表中的状态对异常进行检测,检测步骤如下.

步骤 1　获得任务序列

T2PAD 通过在系统内核调度器调度任务之前捕获任务 ID 进而获得任务序列.

步骤 2　任务序列转换为状态流

将任务序列按照时间窗口 w 划分为子序列,对于每个子序列依据 1.2 节所述方法将其转换为一个状态.

步骤 3　异常状态检测

将状态流中的每个状态与状态表中正常状态依据 1.3 节所述方法进行相似性判断,判断的阈值为 Q. 如果与状态表中所有状态的相似性均大于 Q,则认为该状态为异常状态,报告异常状态并提交导致

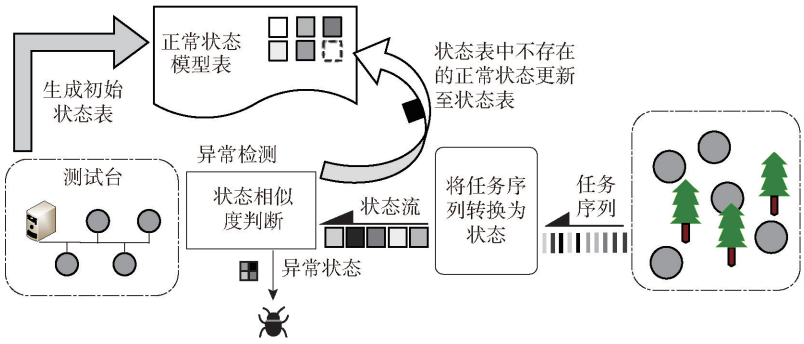

图2 T2PAD 系统结构

相似性大于 Q 的任务,为寻找异常的原因与异常的定位提供代码级帮助.

步骤4 状态表更新

对于报告为异常的状态经过检查后发现其是一个在状态表中不存在的正常状态,则将其添加至状态表.

T2PAD 有2个可调参数:w、Q. 其中,w 为统计任务转移概率的时间窗口大小;Q 为利用余弦相似度评估状态矩阵间差异的阈值. w 的值越大,对持续较短时间的异常越不敏感,但其对于长期异常的检测的准确性越高;Q 的值的大小与异常检测的灵敏度相关,Q 值越大漏报率会升高,误报率会下降,反之漏报率下降,误报率越高.

4 实例分析

感知节点异常检测研究领域目前没有标准的异常测试集,现有研究大多通过实例分析验证其所提方法有效性. 实例1来自 TinyOS 开源代码库,实例2为笔者开展实验过程中遇到的真实问题,实例3来自文献[5]. 实例1和实例2的异常与软件缺陷有关,实例3的异常则与外部存储器硬件故障有关. 在实例2中,发现了 TinyOS 网络协议中一个新的缺陷,利用 T2PAD 给出的异常信息找到了异常程序在源码中的位置,并对其进行了改善.

实验使用 TinyOS2.1.x 嵌入式操作系统和 Telosb 节点作为程序编译目标平台,使用 Cooja 仿真器运行网络程序[7]. 通过对 TinyOS 内核调度器代码的修改,实现了任务的记录. 通过修改 Cooja 底层代码,在实例3中模拟硬件故障.

4.1 实例1:分发协议定时器溢出

实例1中 TinyOS 的 Trickle 协议定时器代码存在溢出缺陷(该版本在 GitHub 网站 TinyOS 主仓库 commit 号为11ff964):当分发节点丢失一段时间(约10 min),与分发节点通信的接收节点会持续执行某一个任务而无法进入低功耗模式. 实验使用 TinyOS 例程 TestDissemination 对该缺陷进行测试. 设置时间窗口参数 $w=4.5$ min,阈值 $Q=13$.

实验包含1个分发节点、3个接收节点,分发节点在135 min 后死亡 实验中1个接收节点正常状态模型矩阵如下:

$$\boldsymbol{S}_0=\begin{Bmatrix}0.37 & 0.539 & 0 & 0 & 0 & 0.091\\ 0 & 0 & 0 & 0.003 & 0.074 & 0.923\\ 0 & 0 & 0 & 0 & 0 & 0\\ 0.429 & 0.536 & 0 & 0 & 0 & 0.036\\ 0 & 0.018 & 0 & 0.482 & 0 & 0.5\\ 0.448 & 0.483 & 0 & 0 & 0.069 & 0\end{Bmatrix} \tag{8}$$

$$\boldsymbol{S}_1=\begin{Bmatrix}0.356 & 0.548 & 0 & 0.006 & 0 & 0.09\\ 0.005 & 0 & 0 & 0 & 0.064 & 0.931\\ 0 & 0 & 0 & 0 & 0 & 0\\ 0.577 & 0.423 & 0 & 0 & 0 & 0\\ 0 & 0.021 & 0 & 0.479 & 0 & 0.5\\ 0.455 & 0.483 & 0 & 0.002 & 0.06 & 0\end{Bmatrix} \tag{9}$$

图3所示为感知节点的状态流以及对状态的检测结果,图中数值为状态的编号. 可以看出,T2PAD 准确地检测到了程序的异常行为. 表2给出了相应的异常任务. 该异常原因为 TrickleTimerImplP 组件的代码缺陷,而由异常任务 0x000f 可以为定位该代码缺陷的范围提供线索.

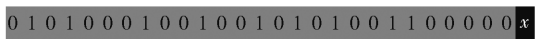

图3 节点状态流

进一步研究了 T2PAD 参数 w、Q 与误报率的关系. 由图4可知,随着时间窗口的增大,误报率将呈下降趋势,但时间窗口过大,T2PAD 对于持续时间

表 2　异常任务 ID

ID	任务名
0x0002	CC2420CsmaP_sendDone_task
0x0008	CC2420ReceiveP_receiveDone_task
0x000a	VirtualizeTimerC_0_updateFromTimer
0x000f	TrickleTimerImplP_0_timerTask

较短的异常状态敏感性降低. 由图 5 可知,随着阈值的增大,误报率呈下降趋势.

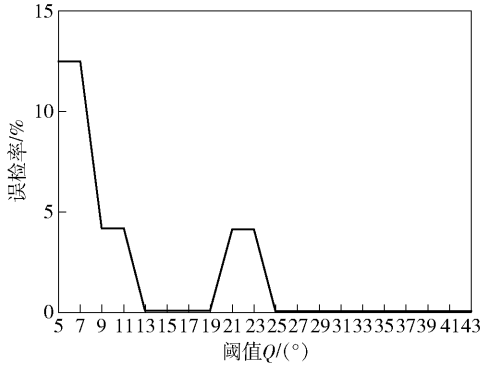

图 4　T2PAD 时间窗口 w 与误报率关系

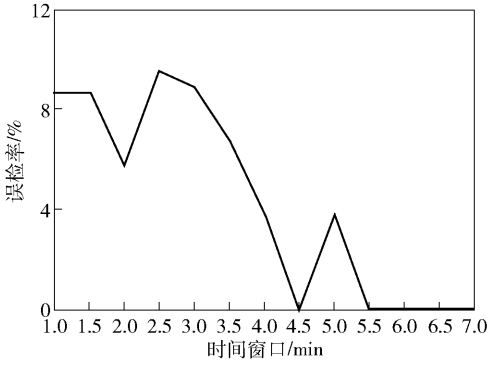

图 5　T2PAD 阈值 Q 与误报率的关系

4.2　实例 2:节点移动导致网络传输异常

实例 2 由 10 个运行 MultihopOscilloscope 应用的节点组成,该应用使用 TinyOS 汇聚树协议进行网络通信. 实验测试了节点移动性对网络的影响,并使用 T2PAD 对这种影响进行监测. 实验包含 10 个节点,其中 1 号节点为根节点,2 号节点和 3 号节点与 1 号节点 1 跳相连,4 号节点在实验开始时仅能通过 3 号节点路由到根节点. 实验仿真时间 2 h 后,4 号节点由于位置移动,离开了 3 号节点的通信范围,进入了 2 号节点的通信范围,仅能通过 2 号节点路由到根节点,整个实验过程持续 4 h. 设置 T2PAD 的时间窗口参数 w = 4.5 min,阈值 Q = 13. T2PAD 检测到 2 号节点、3 号节点、4 号节点发生异常,其状态流图分别如图 6、图 7 和图 8 所示. 图中数值为状态的编号,深色为检测出的异常状态.

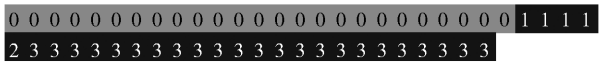

图 6　2 号节点状态流

图 7　3 号节点状态流

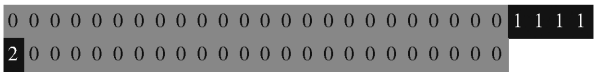

图 8　4 号节点状态流

2 号节点异常状态 1 出现的原因是任务 0x0011 (CC2420ReceiveP_receiveDone_task) 的转移概率发生变化. 从任务 0x0011 到 0x0011 的转移概率由原来的 0.5 变为 0.96. 此异常表明 2 号节点频繁处理接收事件,这是由于 4 号节点在此时频繁向 2 号节点发送数据包. 2 号节点异常状态 2 是 4 号节点与 2 号节点建立连接时出现的状态,出现的原因是任务 0x0021 (UARTDebugSenderP_sendTask) 的转移概率发生变化,这是因为建立连接时节点频繁发送 Debug 调试信息. 2 节点异常状态 3 主要是由一些与发送和接收有关的任务导致,通过分析可知状态 3 是网络结构重新达到稳定后的正常状态,因此将状态 3 添加至正常状态表.

3 号节点出现异常状态 1 ~ 5 的异常任务主要是与发送和接收数据包相关的任务,由于网络经历了从突然改变结构到新的网络结构逐步稳定的动态变化过程,所以与发送和接收相关的任务也会经历一个动态的变化过程,其从异常状态 1 逐步达到异常状态 5 并最终达到稳定. 因此需要将状态 5 添加至正常状态表.

4 号节点出现的异常任务与发送数据、更新路由 0x0016 (CtpRoutingEngineP_0_updateRouteTask) 相关. 4 号节点断开与 3 号节点的连接导致 4 号节点丢失父节点,4 号节点需要重新评估链路质量并更新自己的路由表. 实际上,4 号节点从丢失父节点并进入 2 号节点通信范围内到其与 2 号节点建立路由经历了 20 min,远大于一般情况下路由建立的期望时间. 4 号节点丢失父节点之后任务 0x0016 出现异常,而 2 号节点的 0x0016 任务未出现异常,进一步分析 CtpRoutingEngineP 组件代码分析发现:

Router 定时器 BeaconTimer 在网络稳定之后,广播 Beacon 数据包的周期上限为 512 000 ms,较长的广播周期导致了 4 号节点不能及时获取到新的邻居信息,进一步导致 4 号节点建立新的路由时间较长. 这一异常将使得网络在节点位置发生变化之后不能很快建立新的最佳链路,网络性能变差. 将 Beacon 广播周期上限设置为 1 000 ms 后,TinyOS 汇聚树协议程序很好地满足了项目在节点具有移动性的情况下快速建立路由的需求.

4.3 实例 3:外部存储器失效

Telosb 节点使用 STM25P 作为外部存储器,当外部存储器件失效而无法响应读写请求时程序执行会出现异常. 通过修改 Cooja 底层 STM25P 相关的模块的代码,使其忽略 TinyOS 节点发出的任何读写操作,可以在 Cooja 中重现这一故障.

利用 T2PAD 可以准确地检测到该故障导致的异常. 节点内所有运行的任务如表 3 所示. T2PAD 检测到任务 0x0007 现异常:在正常状态下任务 0x0007 到任务 0x0000 的一步转移概率为 0.035,异常状态下该任务的一步转移概率变为 0.411. 根据 T2PAD 给出的异常信息,开发人员可以确定该故障与 TinyOS 资源仲裁器代码相关. 对 0x0007 任务的进一步分析可以发现:STM25P 相关驱动代码没有对 STM25P 是否正常工作进行检查,进而导致后续异常.

表 3 实例 3 节点内运行任务

ID	任务名
0x0007	ArbiterP_1_grantedTask
0x0000	Stm25pSectorP_signalDone_task
0x0005	DeferredPowerManagerP_0_timerTask
0x0004	DeferredPowerManagerP_0_startTask
0x0003	VirtualizeTimerC_0_updateFromTimer
0x0001	ArbiterP_0_grantedTask

5 结束语

提出了一种新的感知节点状态描述方法. 该方法利用观测窗口内节点执行任务一步转移概率矩阵分析节点运行状态特征,通过矩阵向量的相似性建立节点运行状态序列模型,进而实现对节点异常状态的检测. 9 个典型应用测试表明,任务转移概率可以较好地表征节点运行状态;3 个代表性实例分析,验证了 T2PAD 异常检测方法的有效性.

参考文献:

[1] Zhou Yangfan, Chen Xiny, Lyu M R, et al. Sentomist: unveiling transient sensor network bugs via symptom mining[C]// 2010 IEEE 30th International Conference on Distributed Computing Systems. New York: IEEE Press, 2010: 784-794.

[2] Zhou Yangfan, Chen Xinyu, Lyu Michael R, et al. T-Morph: revealing buggy behaviors of tinyOS applications via rule mining and visualization[C]//2012 ACM SIGSOFT Symposium on the Foundations of Software Engineering. New York: ACM Press, 2012: 1-11.

[3] 马峻岩, 周兴社, 李士宁. 基于 FSM 的感知节点软件故障检测[J]. 北京邮电大学学报, 2013, 36(2): 107-112.
Ma Junyan, Zhou Xingshe, Li Shining. FSM-based fault detection for sensor node software[J]. Journal of Beijing University of Posts and Telecommunications, 2013, 36(2): 107-112.

[4] Khan M M H, Le H K, Ahmadi H, et al. Troubleshooting interactive complexity bugs in wireless sensor networks using data mining techniques[J]. ACM Transactions on Sensor Networks, 2014, 10(2): 1-35.

[5] Dong Wei, Luo Luyao, Chen Chun, et al. Post-deployment anomaly detection and diagnosis in networked embedded systems[J]. IEEE Transactions on Parallel and Distributed Systems, 2016, 27(12): 3588-3601.

[6] Lu Sixing, Roman L. Time and sequence integrated runtime anomaly detection for embedded systems[J]. ACM Transactions on Embedded Computing Systems, 2018, 17(2): 1-27.

[7] Eriksson J, Österlind F, Finne N, et al. COOJA/MSPSim: interoperability testing for wireless sensor networks[C]//2009 International Conference on Simulation Tools and Techniques. Brussels: ICST, 2009: 27.