ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.20, No.7, July 2009, pp.1885−1894
doi: 10.3724/SP.J.1001.2009.03272

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

传感器网络中健壮数据聚集算法*

吴中博[1,2,3+], 张重生[1,2], 陈 红[1,2], 秦 航[4]

[1](中国人民大学 信息学院,北京 100872)

[2](中国人民大学 数据工程与知识工程教育部重点实验室,北京 100872)

[3](襄樊学院 计算机科学与技术系,湖北 襄樊 441053)

[4](武汉大学 软件工程国家重点实验室,湖北 武汉 430072)

Robust Aggregation Algorithm in Sensor Networks

WU Zhong-Bo[1,2,3+], ZHANG Chong-Sheng[1,2], CHEN Hong[1,2], QIN Hang[4]

[1](Information School, Renmin University of China, Beijing 100872, China)

[2](Key Laboratory of Data Engineering and Knowledge Engineering (Ministry of Education), Remin University of China, Beijing 100872, China)

[3](Department of Computer Science and Technology, Xiangfan University, Xiangfan 441053, China)

[4](State Key Laboratory of Software Engineering, Wuhan University, Wuhan 430072, China)

+ Corresponding author: E-mail: rucwzb@163.com

Wu ZB, Zhang CS, Chen H, Qin H. Robust aggregation algorithm in sensor networks. *Journal of Software*, 2009,20(7):1885−1894. http://www.jos.org.cn/1000-9825/3272.htm

Abstract: Saving energy to prolong network life is a big challenge for WSNs (wireless sensor networks) research. In-Network query can reduce the number or size of packets through processing data in intermediate nodes so as to consume energy effectively. Present aggregation algorithms suppose all the sample data are correct. The existing outlier detection algorithms regard detection rate as the primary object and do not consider energy consumption and query characteristic. So the simple combination of the two aspects can not bring good performance. By analyzing the influence of faulty and outlier readings to aggregation results, this paper puts forward a robust aggregation algorithm RAA (robust aggregation algorithm). RAA improves traditional aggregation query using reading vector to judge whether a faulty or outlier has happened. RAA deletes faulty readings, aggregates normal readings and reports outliers. Thus, customers can know the networks condition clearly. Finally, this paper compares RAA and TAGVoting which uses tiny aggregation algorithm to complete aggregation and the Voting algorithm to realize outlier detection at the same time. Experimental results show that RAA outperforms TAGVoting in terms of both energy consumption and detection rate.

Key words: sensor network; query processing; data aggregation; outlier detection; reading vector

* Supported by the National Natural Science Foundation of China under Grant Nos.60603046, 60673138 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant No.2008AA01Z120 (国家高技术研究发展计划(863)); the Program for New Century Excellent Talents in University of China (新世纪优秀人才支持计划)

Received 2007-11-15; Revised 2008-01-10; Accepted 2008-02-04

摘 要: 节约能量以提高网络寿命是传感器网络研究面临的重要挑战.网内聚集查询在中间节点对数据进行预处理,可以减少消息传送的数量或者大小,从而实现能量的有效利用,但是,目前的聚集查询研究假设采样数据都是正确的.而目前的异常检测算法以检测率作为首要目标,不考虑能量的消耗,也不考虑查询的特点.所以将两方面的研究成果简单地结合在一起并不能产生很好的效果.分析了错误和异常数据可能对聚集结果造成的影响,提出了健壮聚集算法 RAA(robust aggregation algorithm).RAA 对传统聚集查询进行了改进,在聚集的同时利用读向量相似性判断数据是否发生了错误或异常,删除错误数据,聚集正常数据并报告异常,使用户可以对网络目前状况有清晰的理解.最后,比较了 RAA 和 TAGVoting(在使用 TAG(tiny aggregation)算法聚集的同时利用 Voting 算法进行异常检测),实验结果表明,RAA 算法在能量消耗和异常检测率方面都优于 TAGVoting.

关键词: 传感器网络;查询处理;数据聚集;异常检测;读向量

中图法分类号: TP393 **文献标识码**: A

无线传感器网络(wireless sensor networks,简称传感器网络)由部署在感知区域的大量微型传感器节点组成,这些节点通过无线通信方式形成一个多跳的自组织网络,协同地感知、处理和传输采集到的数据,使人们能够远程地获取需要的信息[1−3].目前,传感器网络正被广泛地应用于国防军事、环境监测、交通管理、医疗卫生、制造业、反恐抗灾等领域.

传感器的电源能量极其有限,网络中的传感器由于电源能量的原因经常失效或废弃,如何在网络工作过程中节省能源,最大化网络的生命周期,是传感器网络研究所面临的重要挑战[4].由于节点计算所消耗的能量要远远小于数据传输消耗的能量,网内查询处理成为提高网络寿命的重要手段,其核心思想是尽可能地减少数据传送的能量消耗,利用节点对数据进行预处理,可以减少消息传送的数量或者大小,从而实现能量的有效利用[5].文献[6]提出了一个梳子-针模型(comb-needle),该模型将每个数据源的数据缓存在周围的节点上,形成一个针状缓存区,而查询消息则形成类似梳子状的查询转发区.该方法权衡了 Pull 和 Push 方法,减少了通信代价.文献[7]提出基于模型的数据采集算法,在传感器节点和基站分别保存模型,当节点采样数据符合模型时就可以不用传输数据,以此来节约通信开销.另外,网内聚集查询受到广泛重视[8−10],文献[8]提出了 TAG(tiny aggregation)算法,它在网内构造树结构,然后从叶子节点到根节点逐层进行聚集操作,减少能量消耗.文献[10]研究了聚集过程中节点的同步问题.文献[11]设计了一种可以缓存不同粒度聚集结果的层次缓存模型,借助缓存结果可以直接回答某些聚集查询,从而节省了查询开销.但是目前的聚集查询研究假设采样数据都是正确的.

复杂的环境、低廉的价格,导致传感器节点易于失效[12].失效的节点可能会报告任意的读数,不能反映环境的状况;而且,由于受到干扰,传感器节点也经常会产生噪音[13].任意的读数和噪音都是错误数据,会影响查询的结果.所以,过滤错误读数以保证查询结果的准确性是非常重要的.最简单的异常检测办法就是所有节点将数据传送到基站,然后在基站进行统计分析,但是这样会浪费过多的能量.文献[13]利用统计模型进行异常检测,该方法具有较高的错误检测率,但是需要全局的信息,所以并不适用.分布式的异常检测是目前研究的热点[14−16].文献[15]认为空间关系临近的节点往往会产生相似的数据,利用空间关系可以进行异常数据的判断.当一个节点 S_i产生了一个不正常的数据时,S_i将该数据发送给所有邻居节点.邻居节点 S_j将收到的数据和自己当前的采样数据进行比较,如果两个数据的差低于预先给定的阈值,S_j 就认为该数据是正常的,就给 S_i 投正票,否则就投负票.在收到所有邻居的投票后,S_i 进行统计,如果结果为正数,S_i 就认为该数据是正常的,否则就是错误的.文献[16]对此算法进行了改进,提出了基于权重的投票算法,认为距离近的邻居的投票应该具有更高的权重.但是当网络中错误数据增多时,投票算法就不适用了.能量消耗过多仍然是目前异常检测算法所面临的主要问题.

1 研究动机

1.1 问题1

传感器大多布置在恶劣的环境中,在这种情况下,传感器容易失效.失效的传感器节点可能报告任意的读

数,这种读数并不能反映被监控地区的正常状况.而且传感器在受到干扰的情况下会报告一些噪音.

如图 1 所示,在区域 A 内布置了传感器节点以监视当前的温度状况,节点 1 和节点 2 正常,采样数据分别为 25 和 29;节点 3 由于受到干扰,报告了错误的读数 78.如果我们对该区域求平均值,*Avg*(*A*)=*Avg*(25,29,78)=44,则我们可以看到,结果并不能反映真实的环境状况.

1.2 问题2

如图 2 所示,在区域 B 布置了一组传感器以监视当前的温度状况,这些节点自组织成树状结构.某处着火了,离着火处较近的两个节点观测到了这个状况.按照传统的聚集查询,我们计算 *Avg*(*B*)=*Avg*(20,70,23,72,22,21)=38.这里存在两个问题,一方面,38 不能够反映网络目前的真实状况;另一方面,从这一个结果我们并不能知道网络中某个地方着火了.

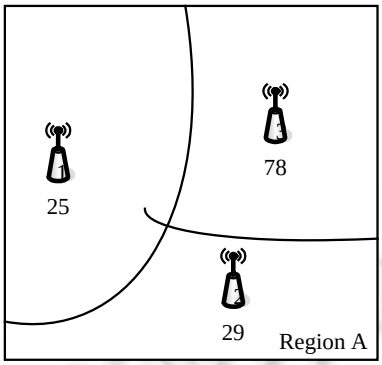

Fig.1 Appearance of faulty reading

图 1 出现错误数据

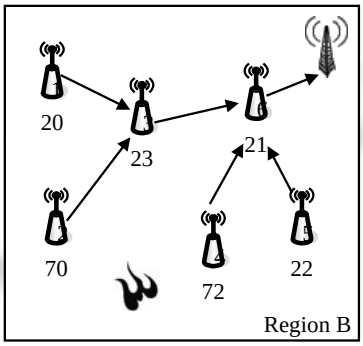

Fig.2 Appearance of outlier

图 2 出现异常数据

对于问题 1,如果我们可以判断出 78 是一个错误读数,那么我们计算 *Avg*(*A*)=*Avg*(25,29)=27,结果就是令人满意的;对于问题 2,虽然网络中的数据都是正确的,但是把异常数据和正常数据放在一起聚集是没有意义的,只有分别报告给用户,用户才能明白网络目前的真实状况.如果我们可以判断出发生了异常,然后把异常数据和正常数据分开计算:*Avg*(*B*)=*Avg*(20,23,21,22)=21.5;*Avg*(*B*′)=*Avg*(70,72)=71.用户得到结果 21.5 和 71,那么他就可以清楚地知道网络目前的状况.通过以上分析,我们可以发现,用户所关心的事情是对传感器网络中的正常读数的聚集结果和对局部异常的报告.

如果我们简单地将网内聚集查询和异常检测的研究成果结合在一起,并不能产生很好的效果.比如使用前面介绍的投票算法,噪音发生了就要进行投票,这样会浪费过多的能量;而且对于之前的异常检测方法,错误数据和异常数据是无法区分开来的,异常数据有可能就被当作错误数据被删掉了,没有报告给用户.如图 2 所示,节点 2 和节点 4 都检测到了一个局部火灾事件,但是它们之间超出了 1 跳的通信范围,也就是说,它们不是邻居.按照投票算法,它们都会被各自的邻居投负票,最终它们的采样数据被当作错误数据删掉了,用户也就无法知道区域中发生了这样一个事件.

从图 2 我们观察到这样一个现象,节点 2 和节点 4 的采样数据如果不被删掉,而是依次在聚集的过程中传给父节点,那么这两个数据会在节点 6 汇合.当节点 6 发现这两个数据很相似,而且它们距离很近时,节点 6 就可以判断出这两个数据是异常的而不是错误的.基于这一观察,我们提出了健壮聚集算法 RAA(robust aggregation algorithm),在聚集的同时进行错误和异常的判断,这样就不会比传统的聚集算法耗费更多的能量;而且 RAA 能够区分出错误数据和异常数据,删除错误数据,并将异常报告给用户,可以使用户对网络目前的真实状况有更加清晰的理解.

RAA 算法对传统聚集方法进行了改进,每个节点保存中间聚集结果,错误数据集合和异常数据集合;节点采样数据后首先判断该数据是否正常,如果节点判断该数据正常,就将该数据合并到中间聚集结果中,否则就放

入到错误数据集合中;对于错误数据集合中的元素,如果能够证明它为异常,就将它从错误数据集合中拿出来放到异常数据集合中;每个节点合并自己所有孩子节点的聚集结果,错误数据集合和异常数据集合,并将合并后的结果传给自己的父节点;最终 Sink 节点将得到正常数据的聚集结果和异常数据集合.另外,我们还提出了读向量的概念,通过比较读向量相似性来进行错误数据和异常数据的判断.

2 聚集框架

在我们的聚集框架中,考虑如下形式的聚集查询:

SELECT *AggFun*(*s.value*)

FROM Sensors *s*

WHERE condition

SAMPLE PERIOD *e* FOR *t*

我们把传感器网络看成一张虚表 sensors.其中 SELECT 子句表示对虚表进行聚集操作,AggFun 表示聚集函数,我们考虑的聚集函数包括 MAX,MIN,SUM 和 AVG;WHERE 子句表示进行聚集操作的数据应满足的条件;SAMPLE 子句中的参数 *e* 表示传感器节点采样数据的频率,参数 *t* 表示这个查询持续的时间,除非用户终止这个查询,否则,这个查询将运行 *t* 个单位时间.我们假设已经使用文献[5]中的算法建立了如图 2 所示的融合树.查询执行过程中,每个节点使用 TAG 协议,对其子树上的节点数据进行聚集.节点将自己的采样数据和子树的聚集结果融合后传给自己的父节点.

在我们的聚集框架中,考虑 3 种数据:正常数据、错误数据和异常数据.

(1) 正常数据.正常数据应该是有规律的,有趋势的,渐变的.比如说我们观测室外的温度,从早上到中午,温度会升高,而且是逐渐升高的,然后在中午保持一段时间,到了下午,温度又逐渐回落.

(2) 错误数据.错误数据一般由硬件的质量、节点的失效或者恶劣的环境等因素造成.正常的节点会产生“噪音”,偏离正常读数;失效的节点会产生任意的读数,这两种数据我们都称其为错误数据.这种数据的特点就是没有规律,任意性,偏离正常读数.

(3) 异常数据.异常数据一般由突发事件引起,比如着火会引起温度的突然升高,下雨会导致温度在短时间内降低等等.这种数据的特点是突发性,短时间内数据趋势发生突变.异常包括全局异常和局部异常.

在我们的聚集框架中,所关心的异常数据是局部的异常数据,就是说那种突发的却只被少数几个节点观测到的异常,而不是那种全局的异常.比如我们观测室外温度,突然下雨了,所有节点都会观测到这个异常,我们通过对所有数据的聚集可以知道发生了这一事件.但是如果是局部着火了,只有少数几个节点观测到这一事件,那么局部的异常会影响全局的聚集结果,全局的聚集结果也反映不了局部的异常情况.我们的目标是区分出传感器网络中的正常数据、错误数据和局部异常数据,将错误数据删掉,将正常数据和局部异常分别聚集后的结果反馈回用户.

3 RAA 算法

3.1 基本定义

定义 1(传感器网络). 一个传感器网络可以用无向图 $G(V,E)$来表示.其中 V 表示所有传感器节点和 Sink 节点的集合,E 表示所有传感器节点之间的一跳路由以及传感器节点和路由节点之间的路由集合,形式化描述为

$$V=V_{sensor}\cup V_{sink},\ E=\{(u,v)\,|\,u,v\in V_{sensor}\}\cup\{(u,v)\,|\,u\in V_{sensor},v\in V_{sink}\} \tag{1}$$

我们用 V_i 表示节点编号为 i 的节点.

定义 2(读向量). 传感器节点 V_i 的读向量是由一个滑动窗口 Δt 内的一系列读数组成的.V_i 的读数可以被表示为 $b_i(t)=\{x_i(t-\Delta t+1),x_i(t-\Delta t+2),...,x_i(t)\}$,其中 $x_i(t)$ 表示节点 V_i 在 t 时刻的采样值.节点 V_i 在一个滑动窗口内的读数就代表它的一个读向量.

定义 3(中间聚集结果). 传感器节点 V_i 的中间聚集结果用于记录其子树上所有正常读数的聚集结果,用 A_i 表示.A_i={*answer*,*count*},其中 *answer* 记录的是聚集的中间结果,*count* 表示该聚集结果中所包含的节点个数.对于聚集查询 MAX,MIN 和 SUM,*count* 是用不着的;对于 AVG,最终的聚集结果等于 *answer* 除以 *count*.

定义 4(错误数据集合). 传感器节点 V_i 的错误数据集合用 F_i 表示,用于记录其子树上的所有的有可能是错误的数据,形式化描述为 $F_i=\{M_1,M_2,\ldots,M_t\}$.

其中,M_t={*nodeId*,*readingVector*,*location*},*nodeId* 表示采样该数据的节点号,*readingVector* 表示该节点的当前读向量,*location* 表示该节点所在位置.因为被节点认为是错误的数据还可能是一个异常,该数据可以被邻近节点的数据证明为异常,*location* 就用于判断节点的邻近关系.

定义 5(异常数据集合). 传感器节点 V_i 的异常数据集合用 Q_i 表示,用于记录其子树上所有的异常数据,形式化描述为 $Q_i=\{N_1,N_2,\ldots,N_t\}$.

其中,N_t={*nodeId*,*readingVector*,*location*},*nodeId* 表示采样该数据的节点号,*readingVector* 表示该节点的当前读向量,*location* 表示该节点所在位置.因为一个异常可以被多个邻近的节点观测到,该异常可以证明邻近的节点的错误数据为异常.

定义 6(RAA 聚集). 节点 V_i 在收到其孩子节点$\{V_j,V_K,\ldots,V_n\}$的中间聚集结果、错误数据集合和异常数据集合后,分别予以合并以形成自己的中间聚集结果、错误数据集合和异常数据集合.

性质 1(中间聚集结果的合并). 节点 V_i 的中间聚集结果为 A_i,

如果聚集查询是 MAX, $A_i.answer=\mathrm{MAX}\{A_j.answer,A_k.answer,...,A_n.answer\}$;

如果聚集查询是 MIN, $A_i.answer=\mathrm{MIN}\{A_j.answer,A_k.answer,...,A_n.answer\}$;

如果聚集查询是 SUM, $A_i.answer=A_j.answer+A_k.answer+...+A_n.answer$;

如果聚集查询是 AVG, $A_i.answer=A_j.answer+A_k.answer+...+A_n.answer$, $A_i.count=A_j.count+A_k.count+\ldots+A_n.count$.

性质 2(错误数据集合的合并). 节点 V_i 的错误数据集合 F_i 为其所有孩子节点$\{V_j,V_K,\ldots,V_n\}$的错误数据集合的并集, $F_i=F_j\cup F_k\cup...\cup F_n$.

性质 3(异常数据集合的合并). 节点 V_i 的异常数据集合 O_i 为其所有孩子节点$\{V_j,V_K,\ldots,V_n\}$的异常数据集合的并集, $O_i=O_j\cup O_k\cup...\cup O_n$.

3.2 RAA算法

RAA 算法的基本思想是:每个节点保存中间聚集结果、错误数据集合和异常数据集合;节点采样数据后首先判断该数据是否正常,如果节点判断该数据正常,就将该数据合并到中间聚集结果中,否则就放入到错误数据集合中;对于错误数据集合中的元素,如果能够证明它为异常,就将它从错误数据集合中拿出来放到异常数据集合中;每个节点合并自己所有孩子节点的聚集结果、错误数据集合和异常数据集合,并将合并后的结果传给自己的父节点;最终 Sink 节点将得到正常数据的聚集结果和异常数据集合.

1) 叶子节点 V_l 的处理

叶子节点先判断自己的采样数据是否正常,若是,则放入到集合 A_l 中;若不是,则放入到 F_l 中,将 A_l 和 F_l 都上传给父节点.

2) 中间节点 V_i 的处理

a) V_i 在收到孩子节点的中间聚集结果、错误数据集合和异常数据集合后,执行 RAA 聚集;

b) V_i 重复执行 a),直到所有孩子都处理完毕,最终形成自己的 A_i,F_i 和 O_i;

c) 中间节点判断自己的采样数据,如果是正常数据就和 A_i 合并;如果不是,就放入到 F_i 中;

d) 对于 F_i 中的每个元素 M_t,依次与 F_i 中的地理位置邻近的元素 M_s 进行相似性比较,如果相似,则把 M_t 和 M_s 从 F_i 中删除,添加到 O_i 中;

e) 对于 F_i 中的每个元素 M_t,依次与 O_i 中的地理位置邻近的元素进行相似性比较,如果相似,则把 M_t 加入到

O_i 中;

f) 将 A_i,F_i 和 O_i 上传给自己的父节点.

3.3 错误数据的判断

正常数据是有趋势的,渐变的.错误数据会使当前的数据变化趋势发生改变或者会使当前的数据发生跳跃.为了捕捉这种变化,我们定义了读向量关系.

定义 7(读向量关系).

$$corr_{i,j}=\frac{b_i(t)\cdot b_j(t)}{\|b_i(t)\|_2^2+\|b_j(t)\|_2^2-b_i(t)\cdot b_j(t)},\ \|b_i(t)\|_2^2=|x_i(t-\Delta t+1)|^2+...+|x_i(t)|^2 \tag{2}$$

如果两个读向量 $b_i(t)$和 $b_j(t)$没有相同的趋势,也不在同一范围内,那么 $corr_{i,j}$ 将趋于 0;当 $b_i(t)$和 $b_j(t)$相同时,$corr_{i,j}$ 就等于 1.

性质 4(读向量的相似性). 两个读向量 $b_i(t)$和 $b_j(t)$是相似的当且仅当 $corr_{i,j}\geq\theta$,θ是预先给定的阈值.

节点新采样数据后首先要进行自我检测,为此需要在每个节点上保持两个读向量:

1) 在当前时间 t 下的读向量,表示为 $b_i(t)$;

2) 最后一次正确的读向量,设其时间为 t_p,表示为 $b_i(t_p)$.

我们计算这两个读向量的相似性,如果这两个读向量不相似,则当前采样数据就被认为是一个错误数据.

3.4 异常数据的判断

在节点进行自我检测之后,一个被认为是错误数据的采样,实际上有可能是一个异常.一个异常发生了,周围相近的几个节点都观测到了这一现象,那么它们应该有近似的趋势或取值,所以我们可以通过比较它们的当前读向量的相似性来判断是否发生了异常.

一个错误数据可以被一个异常数据证明为异常,两个错误数据也可以相互证明为异常.对于错误数据和异常数据我们需要保存它们当前的读向量.但是,如果网络规模比较大,多个节点都发生了错误,那么这些错误数据有可能会比较近似,从而相互证明为异常.为了避免这种误判,我们规定只有地理位置接近的节点才能够相互证明.不失一般性,假设异常的影响范围是一个半径为 r 的圆形区域,那么观测到同一异常的两个节点相隔最远为 $2r$,所以我们认为距离在 $2r$ 之内的两个节点是邻近的.我们可以通过计算两个节点坐标之间的距离来判断两个节点是否接近.为了避免将错误数据的读向量从产生节点一直带到 Sink 节点,我们需要对此作一些优化,优化算法基于性质 5.

性质 5. V_i 表示某个节点,$L(V_i)$表示 V_i 邻近的节点的集合;$P(V_i)$表示 V_i 的一个祖先节点,,而且满足 $L(V_i)\subset B(P(V_i))$,其中 $B(P(V_i))$表示节点 $P(V_i)$子树上所有节点的集合,我们称 $P(V_i)$控制 V_i.如果节点 V_i 产生了一个错误数据,该错误数据直到 $P(V_i)$还未被证明为异常,那么该数据就一定是一个错误数据,可以从 F_i 中删除.

我们的目标就是要发现距离 V_i 最近的满足 $L(V_i)\subset B(P(V_i))$的祖先,这样,错误数据就可以被最早删除,从而消耗最少的能量.当路由树已经建立后,我们使用以下方法确定每个节点的 $P(V_i)$:

(1) 每个节点依次广播自己的信息〈id,location〉;

(2) 每个节点 V_j 将收到的节点信息的 location 和自己的 location 相比较,确定一个节点的集合 V,对于 V 中的每个节点 V_i 满足 $V_j\subset L(V_i)$;

(3) 从叶子节点开始,每个节点 V_j 对于 V 中的每个节点 V_i 产生一条元组〈V_i,1〉,同时为自己产生一条元组 $\langle V_j,1,|L(V_j)|\rangle$,并向父节点发送,其中 $|L(V_j)|$ 表示 $L(V_j)$包含的节点的个数;

(4) 中间节点将收到的元组聚集,最早产生 $\langle V_i,|L(V_i)|,|L(V_i)|\rangle$ 的节点就是 $P(V_j)$.

比如,〈1,1〉和〈1,1〉聚集就产生〈1,2〉,〈1,2〉和〈1,1,5〉聚集就产生〈1,3,5〉.使用这一算法后,每个节点会生成一个自己可以控制的节点的列表 $List(V_i)$.在 RAA 聚集的过程中,中间节点 V_i 判断 F_i 和 O_i 中的节点是否属于 $List(V_i)$,如果属于,对于 F_i 中的节点就可以删掉了;对于 O_i 中的节点应该加上一个标志位,表示它不能够再用来去证明其

他错误数据了.

4 实 验

我们的实验采用 OMNet++进行设计.54 个传感器节点分布在一个 400×700 的空间中,节点的通信距离设置为 80,每个节点有 2~5 个邻居.在实验中,节点的 location 都是已知的;只考虑通信的开销,忽略计算的开销;还假设线路都是可靠的,不考虑线路 fail 的情况.

节点的拓扑结构和采样数据来自于 Berkeley 实验室的采样数据集[17].原数据集中包括温度、湿度、照度等采样数据,由于维度对于实验效果并没有影响,为简单起见,我们只提取了温度进行实验.每一条采样数据中包含节点号、时间戳和采样值.由于原数据集的采样频率比较快,数据变化比较缓慢,不利于进行实验,我们取每 10 轮的平均值作为 1 轮的采样数据,共构建了 1 000 轮采样数据.

我们比较 RAA 和 TAGVoting 两种算法的执行情况.其中 RAA 是本文提出的健壮聚集算法;TAGVoting 是在使用 TAG 算法进行数据聚集时使用文献[15]中的 Voting 算法进行异常检测.

4.1 实验参数的设置

滑动窗口要反映数据的变化趋势,当然滑动窗口越大就越能反映数据变化的趋势,但是滑动窗口越大,传输中耗费的能量越多.所以数据滑动窗口的原则应该是在反映数据变化趋势的基础上越小越好.

数据变化的趋势和数据变化的幅度是相关的,而数据变化的幅度和采样的频率以及采样的属性是相关的,所以滑动窗口大小的选择应该根据实际应用来确定.以实验中的场景为例,室内的温度采样数据一般在 15°C~30°C 之间,所以正常数据的数据变化幅度一般在 2 倍以内.如果发生异常,采样值也应该在 30°C~100°C 之间,所以我们认为数据变化的最大幅度最大为 4 倍(近似).

表 1 中的数据是,当数据发生变化时,对于不同的数据窗口,其阈值的变化情况.横向表示数据窗口的变化,纵向表示数据变化率.我们可以看到对于一个给定的数据变化率,阈值随着数据窗口的增大而增大,但当数据窗口达到一定程度时,阈值的变化就不明显了.对于错误数据的判断,我们取数据变化率为 2;对于异常数据的判断,我们取数据变化率为 4.

Table 1 Change of threshold

表 1 阈值的变化情况

	1	2	3	4	5	6
1.3	0.935 252	0.978 471	0.989 39	0.993 705	0.995 838	0.997 046
1.6	0.816 327	0.922 766	0.957 864	0.973 547	0.981 871	0.986 808
1.9	0.701 107	0.849 699	0.910 124	0.940 375	0.957 609	0.968 337
2.2	0.604 396	0.772 794	0.853 073	0.897 436	0.924 443	0.942 069
2.5	0.526 316	0.699 707	0.792 811	0.848 69	0.884 789	0.909 419
2.8	0.463 576	0.633 701	0.733 549	0.797 662	0.841 273	0.872 25
3.1	0.412 783	0.575 564	0.677 709	0.746 971	0.796 201	0.832 438
3.4	0.371 179	0.524 924	0.626 414	0.698 314	0.751 342	0.791 606
3.7	0.336 67	0.480 966	0.579 983	0.652 646	0.707 914	0.751 016
4	0.307 692	0.442 777	0.538 289	0.610 396	0.666 667	0.711 566

综上所述,实验参数设置见表 2.

Table 2 Experimental parameters

表 2 实验参数

Signal	Value	Description
θ_1	0.9	Threshold of fault detection
θ_2	0.6	Threshold of outlier detection
Δt	4	Length of reading vector

4.2 能量消耗的比较

首先我们比较了 RAA 和 TAGVoting 算法在数据错误率发生改变时的能量消耗情况.我们在每轮采样数据中随机地将一些正常数据改变为错误数据,使错误数据达到一定的比例.如图 3 所示,横轴表示网络中错误数据

的比例,纵轴表示在该错误率下节点平均收到的总包数.从图 3 中我们可以看出,RAA 算法在数据错误率发生改变时,节点的平均收包数维持在 1 000,没有改变,而 TAGVoting 算法随着数据错误率的增长,节点的平均收包数不断增加.RAA 算法在聚集的过程中完成错误检测,当网络中错误数据增多时,路由结构并不会发生改变,聚集过程也就不会发生改变,所以通信的次数也没有发生改变.而对于 TAGVoting 算法,被节点认为是错误的数据需要邻居节点投票加以确认,随着错误率的增加,需要邻居投票的次数也相应增加,从而导致网络能量消耗增加.

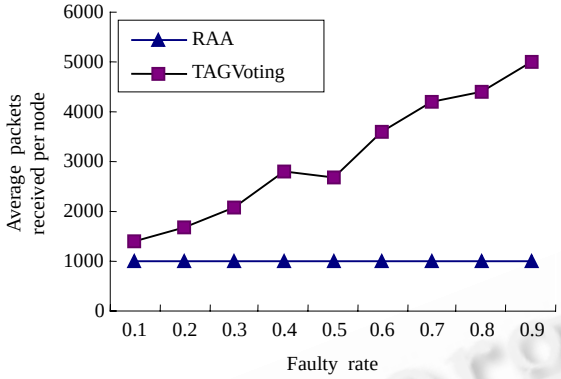

Fig.3 Comparison of energy consumption

图 3 能量消耗的比较

4.3 检测率的比较

为了比较两种算法的错误检测效果,我们定义了检测率和检测失效率.

定义 8. 检测率.我们用 X_B 表示采样的数据集,用 Y_B 表示 X_B 中的错误数据集.在算法执行过程中,Y_B 的一部分数据被检测出来并被过滤掉了,用 Y'_B 表示.检测率就记作 $\frac{|Y_B \cap Y'_B|}{|Y_B|}$.

定义 9. 检测失效率.检测率失效记作 $\frac{|(Y_B \cup Y'_B)-(Y_B \cap Y'_B)|}{|X_B|}$,是错误读数被当作正常读数的比例.

首先,我们只考虑错误数据,不考虑异常数据.与实验 1 一样,我们在每轮采样数据中随机地将一些正常数据改变为错误数据,使错误数据达到一定的比例.从图 4 和图 5 我们可以看出,RAA 算法在检测率和检测失效率方面都优于 TAGVoting 算法.这是因为 Voting 算法仅仅以值的大小来判断节点读数的相似性,而 RAA 算法不仅考虑到值的因素,还考虑到趋势的变化.

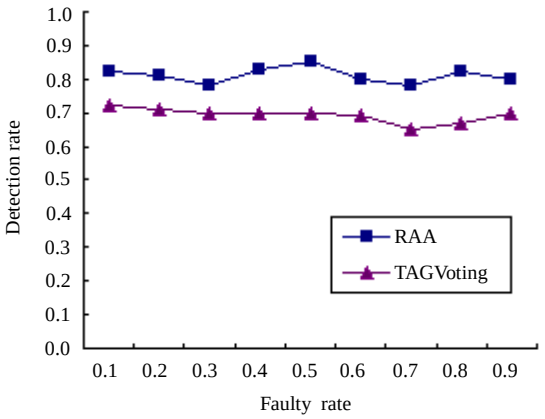

Fig.4 Comparison of detection rate

图 4 错误检测率的比较

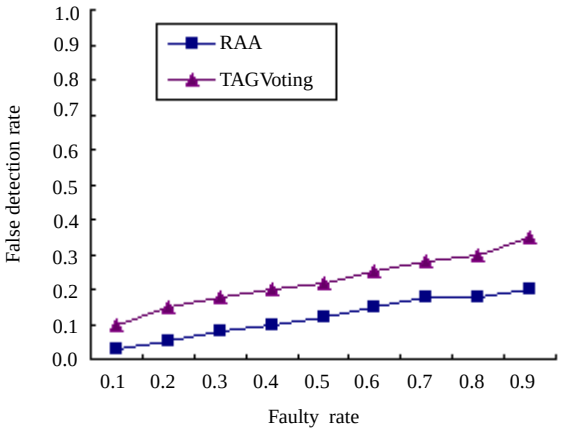

Fig.5 Comparison of false detection rate

图 5 检测失效率的比较

其次,我们在错误数据中插入一些异常.我们将数据错误率设置为 10%,然后在 1 000 轮采样中随机抽取了 100 轮,在其中插入局部异常,该异常被 2~5 个节点观测到,然后比较了 RAA 和 TAGVoting 对于异常的检测效果,RAA 算法对于异常的检测达到 90%,而 TAGVoting 只有 70%.分析结果表明,在异常被较多节点(5 个)观测到的时候,RAA 和 TAGVoting 检测效果是差不多的;但是当异常仅被少数节点(2 个)观测到的时候,RAA 算法的检测效果就明显优于 TAGVoting.

4.4 读向量长度对检测率的影响

最后,我们分析了滑动窗口的大小对于 RAA 算法检测率的影响.如图 6 所示,横轴表示Δt 的变化,纵轴表示随着Δt 的变化,RAA 算法检测率的变化情况.从图中我们可以看出,Δt 从 1 变为 5 的过程中检测率有比较明显的提高,而当Δt 大于 5 之后,检测率变化就不明显了.当Δt 等于 1 时,RAA 就和投票算法的原理是一样的,仅仅通过值的大小来判断两个读数是否相似.Δt 较小的时候,体现不出数据变化的趋势,当Δt 增大到一定程度时就足够反映当前数据变化的趋势了.所以,Δt 具体的取值是与环境的变化以及采样的频率密切相关的.

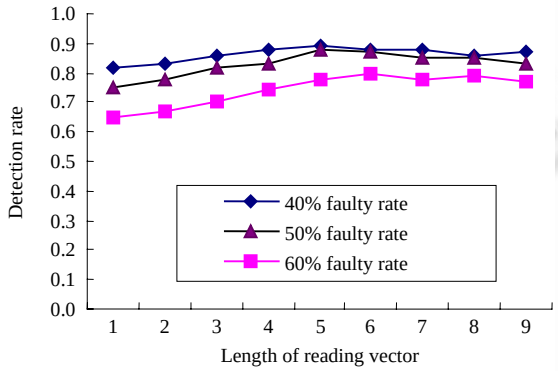

Fig.6 Impact of Δt

图 6 Δt 的影响

5 结束语

本文分析了错误数据和异常数据可能对传感器网络中聚集结果造成的影响,提出了健壮聚集算法 RAA.RAA 对传统聚集查询进行了改进,在聚集的同时利用读向量相似性判断数据是否发生了错误或异常,删除错误数据,聚集正常数据并报告异常,使用户可以对网络目前状况有清晰的理解.我们比较了 RAA 和 TAGVoting(在使用 TAG 算法聚集的同时利用 Voting 算法进行异常检测),实验结果表明,RAA 算法在能量消耗上明显低于 TAGVoting.在异常检测方面,在节点邻居较多时,两种算法的准确率是近似的;在节点邻居较少时,RAA 算法要明显优于 TAGVoting 算法.RAA 算法由于要判断节点位置和读向量的相似性会造成一定的延迟,由于运算中涉及的都是简单的除法和乘法,所以本文并没有考虑延迟的影响.我们目前正在搭建真实的传感器网络环境,将在真实的环境中来验证 RAA 算法的延迟.

References:

[1] Kahn JM, Katz RH, Pister KSJ. Next century challenges: mobile networking for “Smart Dust”. In: Proc. of the 5th Annual Int’l Conf. on Mobile Computing and Networks. New York: ACM Press, 1999. 271−278. http://bnrg.eecs.berkeley.edu/~randy/Papers/mobicom99.pdf

[2] Ren FY, Huang HN, Lin C. Wireless sensor networks. Journal of Software, 2003,14(7):1282−1291 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/14/1282.htm

[3] Li JZ, Li JB, Shi SF. Concepts, issues and advance of sensor networks and data management of sensor networks. Journal of Software, 2003,14(10):1717−1727 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/14/1717.htm

[4] Akyildiz IF, Su WL, Sankarasubramaniam Y, Cayirci E. A survey on sensor networks. IEEE Communications Magazine, 2002,40(8):102−114.

[5] Krishnamachari B, Estrin D, Wicker S. The impact of data aggregation in wireless sensor networks. In: Proc. of the Int'l Workshop on Distributed Computing Systems. Washington: IEEE Press, 2002. 575−578. http://lecs.cs.ucla.edu/Publications/papers/krishnamacharib_aggregation.pdf

[6] Liu X, Huang Q, Zhang Y. Combs, needles, haystacks: Balancing push and pull for discovery in large scale sensor networks. In: Proc. of the 2nd ACM Conf. on Embedded Networked Sensor Systems. New York: ACM Press, 2004. 183−194. http://www2.parc.com/spl/projects/ecca/pubs/comb.pdf

[7] Chu D, Desphande A, Hellerstein J, Hong W. Approximate data collection in sensor networks using probabilistic models. In: Proc. of the Int'l Conf. on Data Engineering. Washington: IEEE Press, 2006. 48. http://www.cs.umd.edu/~amol/papers/icde06.pdf

[8] Madden SR, Franklin MJ, Hellerstein JM, Hong W. TAG: A tiny aggregation service for ad-hoc sensor networks. In: Proc. of the ACM Symp. on Operating System Design and Implementation. New York: ACM Press, 2002. 131−146. http://www.cs.berkeley.edu/~franklin/Papers/madden_tag.pdf

[9] Sharaf A, Beaver J, Labrinidis A, Chrysanthis P. Balancing energy efficiency and quality of aggregate data in sensor networks. VLDB Journal, 2004,13(4):384−403.

[10] Yao Y, Gehrke J. The cougar approach to in-network query processing in sensor networks. SIGMOD Record, 2002,31(3):9−18.

[11] Li Y, Ramakrishna MV, Loke SW. Approximate query answering in sensor networks with hierarchically distributed caching. In: Proc. of the 20th Int'l Conf. on Advanced Information Networking and Applications. Washington: IEEE Press, 2006. 281−285. http://portal.acm.org/citation.cfm?id=1129263

[12] Elnahrawy E, Nath B. Online data cleaning in wireless sensor networks. In: Proc. of the Int'l Conf. on Embedded Networked Sensor Systems (SenSys). New York: ACM Press, 2003. 294−295. http://cens.ucla.edu/sensys03/proceedings/p294-elnahrawy.pdf

[13] Ding M, Chen DC, Xian K, Cheng XZ. Localized fault-tolerant event boundary detection in sensor networks. In: Proc. of the IEEE INFOCOM. Washington: IEEE Press, 2005. 902−913. http://www.seas.gwu.edu/~cheng/Publication/PID48574.pdf

[14] Branch J, Szymanski B, Giannella C, Wolff R. In-Network outlier detection in wireless sensor networks. In: Proc. of the 26th IEEE Int'l Conf. on Distributed Computing Systems. Washington: IEEE Press, 2006. 51. http://www.cs.technion.ac.il/~ranw/papers/wolff06icdcs.pdf

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

[16] Krasniewski MD, Varadharajan P, Rabeler B, Bagchi S, Hu YC. Tibft: Trust index based fault tolerance for arbitrary data faults in sensor networks. In: Proc. of the Int'l Conf. on Dependable Systems and Networks. Washington: IEEE Press, 2005. 672−681. http://cobweb.ecn.purdue.edu/~dcsl/publications/papers/2005/tibfit_DSN05.pdf

[17] Intel Berkeley Research Laboratory. http://berkeley.intel-research.net/labdata/

附中文参考文献:

[2] 任丰原,黄海宁,林闯.无线传感器网络.软件学报,2003,14(7):1282−1291. http://www.jos.org.cn/1000-9825/14/1282.htm

[3] 李建中,李金宝,石胜飞.传感器网络及其数据管理的概念、问题与进展.软件学报,2003,14(10):1717−1727. http://www.jos.org.cn/1000-9825/14/1717.htm

吴中博(1980－),男,湖北襄樊人,博士,讲师,主要研究领域为数据库,数据仓库,传感器网络.

陈红(1965－),女,博士,教授,博士生导师,CCF 高级会员,主要研究领域为数据库,数据仓库,传感器网络.

张重生(1982－),男,硕士,主要研究领域为数据库,数据挖掘.

秦航(1980－),男,博士,讲师,主要研究领域为软件工程,传感器网络.