Reliable and Fast Detection of Gradual Events in Wireless Sensor Networks

Liping Peng, Hong Gao, Jianzhong Li, Shengfei Shi, and Boduo Li

Harbin Institute of Technology, China
{lppeng,honggao,lijzh,shengfei,boduo}@hit.edu.cn

Abstract. Event detection is among the most important applications of wireless sensor networks. Due to the fact that sensor readings do not always represent the true attribute values, previous literatures suggested threshold-based voting mechanism which involves collecting votes of all neighbors to disambiguate node failures from events, instead of reporting an event directly based on the judgement of single sensor node. Although such mechanism significantly reduces false positives, it inevitably introduces false negatives which lead to a detection delay under the scenario of gradual events. In this paper, we propose a new detection method – the bit-string match voting (BMV), which provides a response time close to that of the direct reporting method and a false positive rate even lower than that of the threshold-based voting method. Furthermore, BMV is able to avoid repeated and redundant reports of the same event, thus prolongs the life of the network. Extensive simulations are given to demonstrate and verify the advantages of BMV.

1 Introduction

Wireless sensor networks (WSN) are of great significance in resolving many real-world problems, and have attracted increasing research interests in recent years. One of the most important applications of WSN is the detection of events, most of which in the real-world have the property of gradualness – we call them *gradual events*, such as fire and gas leakage[1]. The range of a gradual event changes slowly and the effect of a gradual event attenuates gradually as the distance from the event increases. It is very challenging to achieve a gradual-event-oriented detection with both short response time and high report reliability due to the node failures and the reading errors. In this paper, we focus on the reliable and fast detection of gradual events defined by thresholds, meanwhile guarantee a low energy cost by proposing an in-network method.

There are two existing methods to detect threshold-based events. The first one is a natural yet naive method – a sensor node directly reports an event whenever its reading exceeds the threshold[5]. We name it as *direct reporting* (DR) mechanism. Although this method can detect events in a short response time, it lacks a high report reliability since sensor readings do not always reveal the genuine attribute values due to the node failure and reading error caused by the intrinsic hardware constraints. For example, when a sensor node is damaged

Y. Li et al. (Eds.): WASA 2008, LNCS 5258, pp. 261–273, 2008.

or energy-exhausted, the sensor readings are likely to be constantly high[2] and may exceed the threshold upon which certain event is defined. If a node directly reports an event when its readings exceeds the threshold, the network, as a consequence, intends to report fake events, which is referred as *false positive*.

The second method is a *threshold-based voting* (TV) mechanism[1] proposed in [4]. Considering the fact that faulty nodes are likely to be uncorrelated, while environmental conditions are spatially correlated, TV mechanism disambiguates node failures and reading errors by examining the readings of nearby nodes to reduce false positives and thus to enhance the reliability of event detection. However, it causes the report of true events delayed or even missed inevitably under the scenario of a gradual event (we will formally describe the characteristics of gradual events in detail in Sect. 3.1). Name a node which is the first to detect an event and asks its neighbor nodes to vote for this event as a *reporting node*. Since its neighbors are farther away from the event and thus their readings may not reach the threshold, they will likely give negative votes. As a result, this event will not be reported to the base station until it has escalated to a certain scale, making at least half of neighbors' readings reach the threshold. This is referred as *false negative* in the event detection. As we can see, the false negatives lead to a delay in the detection of events, which need to be maximally eliminated.

Contour map matching is recently introduced to detect events with complex tempo-spatial patterns[1], which are quite different from the threshold-based events. However, it involves matching between globally constructed snapshots of a contour map and a user-specified event pattern, and thus consumes much energy. Actually, the detection of gradual events defined by thresholds can be accomplished locally.

In this paper, we propose a novel detection method for gradual events – *Bit-string Match Voting* (BMV). The intuition is that although readings of a certain node may not be dramatically affected by a gradual event to reach the threshold, there must be a trend in its recent readings. We use bit-strings to record trends and each neighbor of the reporting node votes by matching its own bit-string with that of the reporting node.

The contributions of this paper include:

- To the best of our knowledge, this is the first work to specifically put forward the concept of the gradual event.
- We propose a novel in-network detection method: Bit-string Match Voting (BMV), to quickly and reliably detect gradual events with little energy consumption. It provides a short response time close to that of the DR method and a report reliability which is even higher than that of the TV method.
- Extensive simulations are conducted, which validate the effectiveness of BMV.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 presents the framework of Bit-string Match Voting method, whose efficiency and reliability are corroborated by extensive simulative evaluations in Sect. 4. We conclude the paper in Sect. 5.

[1] This mechanism is named as Optimal Threshold Decision Scheme in [4].

2 Related Work

The definitions of events in previous literature fall into two categories: threshold-based[5,4,3] and non-threshold-based[1,6], the methods of which are discussed in the following two paragraphs correspondingly.

Events are defined as complex filters in [5], each of which can be considered as a special form of the threshold, then expressed as a table of conditions, which can be distributed throughout the network. Once a tuple is satisfied, a report is returned. [3] develops a local event detection method – exponentially weighted moving average(EWMA) without collecting neighbors' readings, while still maintaining certain reliability in the presence of reading errors. It calculates short-term and long-term moving averages with different gain parameters and compares the ratio of the short-term average to the long-term average with a predefined threshold to decide whether an event has occurred. However, the value of the ratio threshold is less intuitive compared with the straightforward attribute threshold, thus it needs more prior knowledge to set the threshold. Furthermore, false positives caused by sensor failures cannot be tackled by EWMA. [4] defines event directly upon attribute thresholds, and introduces Optimal Threshold Decision Scheme. The detection method involves the collection of 0/1 judge predicates provided by neighbors according to whether their readings exceed the threshold. However, within the context of gradual events, when the reading of the sensor node which is the nearest to the event exceeds the threshold, the readings on most of its neighbors may remain below the threshold, as a gradual event expands slowly and its impact on sensor readings attenuates with distance. Thus, the voting inclines to reject the event report when a gradual event just appears until the event has grown to certain scale. Although the event will finally be reported, the delay may be intolerable as such events may cause tremendous loss and become more uncontrollable every minute.

[1] defines event as time series with each element of the series representing a user-specified partial contour map of certain attribute, which is able to characterize the spatial-temporal patterns of event, and converts the event detection problem into a pattern matching one. The network builds or updates contour maps bottom-up at each time stamp and the matching between this globally constructed contour map and a predefined event is carried out at the base station. Since the method is not localized and the messages transmitted between two nodes are complicated, the cost of this method is higher compared with TV method proposed in [4]. [7,6] try to discover homogeneous regions to estimate the event boundary (which splits the whole region into event area and non-event area) to indirectly detect events. Our work differs with [7,6] since we focus on the direct detection and we adopt threshold-based definition of events.

3 Bit-String Match Voting

In this section, we discuss the problem of detecting gradual events and present the bit-string match voting (BMV) method. The goal is to report as soon as

possible when an event occurs and avoid reporting fake events caused by the inaccuracy of sensor readings. We assume that we have a reliable network layer for safe data transmission.

3.1 Preliminaries

Suppose that the surveillance area is a 2-dimensional space $\mathcal{R}$. The actual value of some attribute for a point $p \in \mathcal{R}$ at a time stamp t is $A(p,t)$.

For any t, $A(p,t)$ is spatially continuous. For ease of discussion in later parts, we introduce the symbol $\mathcal{D}(k,t)$ as the *k-region at time t*, which is defined as

$$\mathcal{D}(k,t) = \{p : A(p,t) \geq k\},$$

and the notation $|\mathcal{D}(k,t)|$ as the area of region $\mathcal{D}(k,t)$.

Definition 1 (Event). *Given an attribute value λ_e as a priori, if $\exists p \in \mathcal{R}$ such that $A(p,t) \geq \lambda_e$, we say an event exists at time t. We call λ_e the event value and $\mathcal{D}(\lambda_e,t)$ the event region at time t.*

λ_e is the character attribute value of an event and is obtained from natural observations. For example, $\lambda_e = 800$K in a fire event means that the fire flame temperature is 800K.

Events can be divided into two categories: the *upgrowth event* whose event value is greater than the normal attribute values, and the *downgrowth event* whose event value is less than the normal attribute values. Considering that these two cases are theoretically equivalent, we only discuss the upgrowth events in this paper without losing generality. The downgrowth events can be handled in the similar way.

In real applications, in order to detect events earlier and at the same time avoid false positives, the judgement of the occurrence of events is usually based on another attribute value λ_r which lies between λ_e and the normal attribute value when no event appears, rather than λ_e itself. For example, the temperature of a fire event is usually 600-1000K[8], the temperature under normal situations is 250-310K, and the threshold of reporting a fire event can be set to 380K.

Definition 2 (Report Region). *Given a predetermined threshold λ_r, $\mathcal{D}(\lambda_r,t)$ is called the report region at time t.*

In this paper, we focus on the detection of gradual events, which keep growing in a certain duration of time rather than disappear immediately after they happen. Now we describe the *gradual event* by listing its properties. Some properties of gradual events are widely used, but never clearly defined. A gradual event is an event that has the following two properties.

Property 1 (Spatial Gradualness). There exists a positive number α, $\forall p_1, p_2 \in \mathcal{R} - \mathcal{D}(\lambda_e,t)$, $|A(p_1,t) - A(p_2,t)| \leq \alpha \cdot ||p_1 - p_2||$, where $||p_1 - p_2||$ is the euclidian distance between p_1 and p_2.

α limits the geographical change of attribute values. Property 1 indicates the spatial correlation of attribute values. Generally, the effect on attribute values brought by an event is proportional to d^{-2} [9], where d is the distance from the event.

Property 2 (Temporal Gradualness). $\forall t$ when an event exists, $\mathcal{D}(\lambda_e, t) \subset \mathcal{D}(\lambda_e, t+1)$, and $E(|\mathcal{D}(\lambda_e, t+1) - \mathcal{D}(\lambda_e, t)|) = V_e \cdot f(\mathcal{D}(\lambda_e, t))$, with E representing the expectation and f representing a function that always returns a positive number. Different functions stand for different event models(will be further discussed soon). V_e is a positive number used to limit the growing speed.

Property 2 describes that an event keeps growing, which is quite common for physical events(e.g. fires). As most physical events grow irregularly, we use the expectation to describe the growing in Property 2, where V_e is the *growing speed* and f is the *growing model*. However, V_e can not be arbitrarily small – it has to grow apparently faster than the change of the natural environment.

As the first attempt to put forward the concept of the gradual event, we further introduce three basic growing models that we observe in the real world.

- *Linearly Growing Model*
  In this type, $f(\mathcal{D}(\lambda_e, t)) = 1$. Therefore $|\mathcal{D}(\lambda_e, t)|$ can be approximately considered as an arithmetic progression. A typical example is the gas leakage. At the very beginning of a gas leakage event, the source of the leakage uniformly emits gas, making the leakage area grows linearly.
- *Exponentially Growing Model*
  In this type, $f(\mathcal{D}(\lambda_e, t)) = |\mathcal{D}(\lambda_e, t)|$. Therefore $|\mathcal{D}(\lambda_e, t)|$ can be approximately considered as a geometric progression. A typical example is the eutrophication, a process where water bodies receive excess nutrients that stimulate excessive and exponential propagation of floating algae [10].
- *Boundary Growing Model*
  In this type, $f(\mathcal{D}(\lambda_e, t)) = |\mathcal{B}(\mathcal{D}(\lambda_e, t))|$, where $\mathcal{B}(\mathcal{D}(\lambda_e, t))$ is the boundary of region $\mathcal{D}(\lambda_e, t)$ and $|\mathcal{B}(\mathcal{D}(\lambda_e, t))|$ is its length. A typical example is the fire, where the flame at the boundary spreads and ignites the adjacent region.

The intuitive meaning of V_e differs for different models. For example, $V_e = 10$ for the first model means that the event extends 10 units of area per sample period; $V_e = 0.3$ for the second model means that the event extends 30% of current size in the next sample period, and for the third model means that the event extends 30% of the neighbor area of the current boundary.

In real applications, the readings of a sensor node may not accurately tell the actual attribute values mainly in two aspects. First, a sensor node may fail and keep reporting readings nowhere near the actual attribute values. Besides, even if a sensor node is properly working, its reading may still contain errors, though typically not far away from the actual attribute values. Let $A'(p, t)$ denotes a sensor reading at the position p and time stamp t, so the error is $A'(p, t) - A(p, t)$.

Since the inaccuracy of sensor readings may lead to both false positives and false negatives in the event detection, the immediate question is how to make the network reliable and at the same time responsive. We present BMV, which leverages the special properties of gradual events together with the voting mechanism, as a solution in Sect. 3.2.

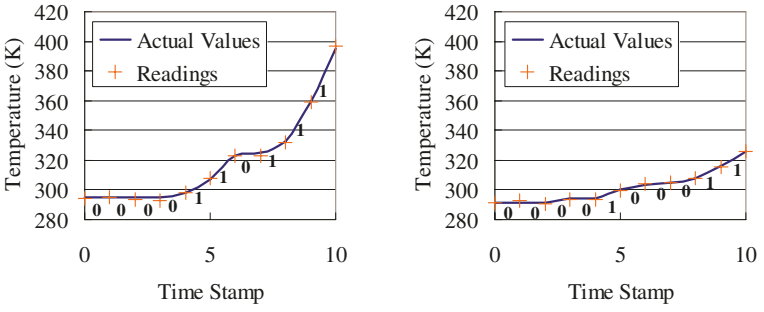


Fig. 1. Example of pattern encoding

3.2 Voting Strategy

We design a new strategy to give votes more accurately, rather than simply voting according to the event threshold. According to Property 1, when a node calls its neighbors to vote, although the readings of a neighbor node may not exceed the threshold, its recent sample readings should follow a similar pattern with that of the node that starts the voting, if both nodes are working properly. Since the communication in WSN is very costly, we encode the pattern into a bit-string on the basis of Property 2 to save the communication cost.

In our approach, each sensor node periodically samples the attribute values. A buffer is set on each node to keep several latest sample readings. When the current reading on a sensor node exceeds the event threshold (thus it becomes a *reporting node*), it encodes its buffered readings into a bit-string which records the growing pattern of these latest readings. Then this node calls for a voting process by sending its bit-string to its neighbors. If a neighbor's reading at current time stamp reaches the threshold, it returns a positive vote. If not, it encodes its buffered readings into a bit-string, and tries to match the two bit-strings by our matching method, which will be introduced later. If the matching succeeds, the neighbor node gives a positive vote. After the reporting node collects all votes from its neighbors, it reports the event to the base station only if positive voting rate exceeds an *voting percent* q. We name our method as Bit-string Match Voting (BMV). We first introduce the pattern encoding and bit-string matching method, then give the detailed voting process.

Pattern Encoding. Intuitively, if the values of two adjacent readings on a node differ significantly, it means the scale of the event has been escalated, which is called a *jump*. If such a *jump* can also be captured and verified by its nearby sensor nodes, then it is evidence that the reporting sensor node is working properly, and the event should be reported. Based on this intuition, we propose the encoding scenario as follows:

If a node's local buffer contains $n+1$ readings $A'(p, t-n)$, $A'(p, t-n+1)$, $\ldots$, $A'(p, t)$, then we can get an n-bit string $(b_0, b_1, \ldots, b_{n-1})$, where

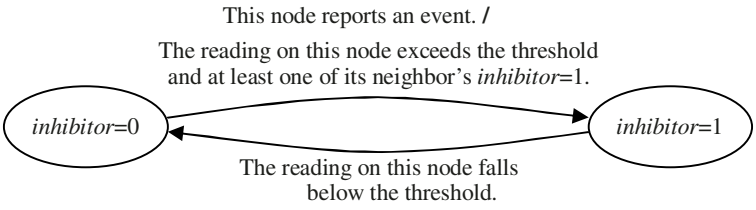


Fig. 2. Status transformation of inhibitor

$$b_i = \begin{cases} 1 & \text{if } A'(p, t-n+i+1) - A'(p, t-n+i) > \delta \\ 0 & \text{otherwise} \end{cases}.$$

δ is a positive number which should be determined according to the distribution of the reading error, and is typically set to be β times of the standard deviation of reading error's distribution. Correspondingly, we call δ the *encoding distance* and β the *relative encoding distance*. Recall that here what we consider is upgrowth events, therefore we do not record any decreasing trend.

Example 1. If a sensor node buffers the latest 11 temperature readings (294, 295, 293.4, 292.9, 297.8, 308.1, 322.7, 322.9, 332.3, 359.1, 396.3), as shown in Fig. 1 (a), the corresponding bit-string is 0000110111 for $\delta = 5$.

Bit-string Matching. We define two functions $AND(\omega_1, \omega_2)$ and $COUNT(\omega)$ where ω_1, ω_2 and ω are all bit-strings, and the length of ω_1 equals to that of ω_2. $AND(\omega_1, \omega_2)$ returns the bit-string which is the logic AND of ω_1 and ω_2. $COUNT(\omega)$ returns the number of 1's in ω.

When a voting node performs the bit-string matching, it should have already received a bit-string ω_r from the reporting node. To match ω_r with locally buffered readings, the voting node encodes its own readings to ω_v by the same pattern encoding methods. ω_r matches with the pattern on the voting node if and only if

$$AND(\omega_r, \omega_v) = \omega_v \text{ and } COUNT(\omega_v) > 0.$$

Example 2. If the reporting node is the same as in Example 1, and the buffered readings of a voting node is (291.1, 292.7, 290.9, 293.6, 293.8, 299.5, 304.3, 304.7, 308, 315.7, 325.9), thus ω_r is 0000110111 and ω_v is 0000100011 for $\delta = 5$. So ω_r matches the pattern on this voting node.

Voting Process. The voting process has been briefly introduced at the beginning of Sect. 3.2. But one event may be reported repeatedly for quite a long period, which will reduce the lifetime of the network. To solve this problem, we set a bit flag called *inhibitor* on each node, initialized as 0. The status of *inhibitor* is maintained according to Fig. 2. A nodes p's *inhibitor* is 1 at time t means that, $p \in \mathcal{D}(\lambda_r, t)$ and there exists a node $p' \in \mathcal{D}(\lambda_r, t)$ where p' satisfies:

1. p' reported an event before t.
2. There exists a multi-hop communication path $p' \rightarrow p$ such that all the nodes on the path are in $\mathcal{D}(\lambda_r, t)$.

Algorithm 1. `ExceedThreshold()`

Input: encoding distance δ, bit-string length n, voting percent q, locally buffered readings $readings[\,]$

```
 1: if inhibitor = 0 then
 2:    neighInhibitors[ ] = CollectNeighborInhibitor();
 3:    for all i is a neighbor do
 4:       if neighInhibitors[i] = 1 then
 5:          inhibitor = 1;
 6:    if inhibitor = 0 then
 7:       bitString = Encode(δ, n, readings[ ]);
 8:       voteResult = RequestVoting(bitString);
 9:       if voteResult.positiveVotePercent≥ q then
10:          ReportEvent();
11:          inhibitor = 1;
```

Algorithm 2. `ReceiveVotingRequest()`

Input: the bit-string received from the reporting sensor node $bitString$, threshold λ_r, encoding distance δ, bit-string length n, locally buffered readings $readings[\,]$

```
 1: if readings[ ].mostRecentReading≥ λr then
 2:    ReturnVote(POSITIVE);
 3: else
 4:    localString = Encode(δ, n, readings[ ]);
 5:    match = Match(localString, bitString);
 6:    if match =TRUE then
 7:       ReturnVote(POSITIVE);
 8:    else
 9:       ReturnVote(NEGATIVE);
```

Intuitively, if a node's *inhibitor* is 1, at least one node in the same connected subregion of the report region has already reported the event. Therefore, another report is unnecessary.

The detailed voting process is described in Algorithm 1 and 2. The procedure `ExceedThreshold()` is called on a node whenever its latest reading exceeds λ_r. Line 2 collects all its neighbor nodes' *inhibitor*s. Line 8 starts a voting, and broadcasts the bit-string to all neighbors. Line 10 reports an event to the base station. The procedure `ReceiveVotingRequest()` runs on a node whenever it receives the request for voting from its neighbor. Line 2, 7 and 9 send its vote to the reporting node.

4 Simulations

In this section we present simulative study of our approach compared with DR and TV. The response time and reliability of the three approaches are tested respectively.

4.1 Simulation Setup

We perform all simulations with C++ codes and some interface functions provided by MATLAB C Math Library. Each of our simulation results represents an average summary of 100 runs.

Event. Based on Sect. 3.1, we simulate a fire event whose event value λ_e is 800K. It firstly occurs at a random point in the network and then gradually grows at a velocity of V_e. At any time stamp t, the attribute value $A(p,t)$ is λ_e if p is inside the event area $\mathcal{D}(\lambda_e, t)$. If p is outside the event area,

$$A(p,t) = \lambda_e \cdot \left(1 + \frac{d(p)}{\sqrt{\frac{|\mathcal{D}(\lambda_e,t)|}{\pi}}}\right)^{-2}$$

where $d(p)$ represents the euclidean distance between p and $\mathcal{B}(\mathcal{D}(\lambda_e, t))$.

Network. In our simulation, 100 sensor nodes are uniformly distributed in a square area of 100×100 m^2. The communication radius is 20m. The readings of a normally working node fluctuate around the actual attribute values at corresponding time stamps due to the existence of errors, which are independent and obey the normal distribution $N(0, \sigma^2)$. Based on our data set collected from real sensor nodes as well as the Intel Lab Data[2], the readings of a failed node slowly and linearly grow until one reading meets the physical limitation of the sensor, and then the constant high readings are given out. The growing speed is much slower than an event. For our temperature data set, it is about 1-2K per minute, and we adopt this real phenomenon in the simulation.

Table 1. System Parameters

Parameter	Default Value
Growing speed	0.3
Error's standard deviation	2K
Event threshold	380K
Voting percent	50%
Encoding length	7
Relative encoding distance	2

Parameters and Metrics. We simulate two scenarios: one is that a fire event exists in the network, and the other is that there are some failed sensor nodes in the network but no event actually exists. The former scenario is used to demonstrate the response time of DR method, TV method and BMV method, whereas the latter one is to exhibit the report reliability of the TV method and BMV method.

In the first scenario, we test the response time (from the occurrence of an event till the reporting node sends the report to the base station) of the three

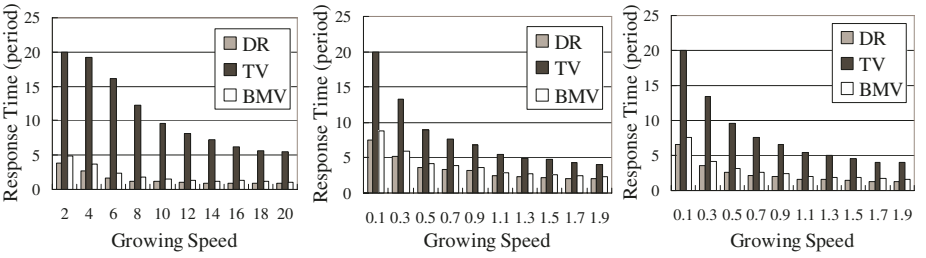
approaches with respect to several parameters, including the growing speed V_e, the error's standard deviation σ, the event threshold λ_r, the encoding length n, the relative encoding distance β and the voting percent q. The first three parameters impact the sensor readings and therefore they affect the response time of all the three methods. On the contrary, the other three parameters are the input of the BMV method, so they only influence the response time of the BMV method. In the second scenario, the only parameter that we care about is the percentage of the failed nodes in the network, and the metric is the report reliability. The default values are listed in Table 1 and in each simulation we examine only one parameter.

4.2 Response Time

Growing Speed. In this set of experiments, we test the three typical models of events discussed in Sect. 3.1 in order to exhibit the suitability of BMV under different growing models. The results are depicted in Fig. 3 (a)-(c). It can be observed that BMV's response time is quite close to DR's and better than TV's in all cases. This observation validates the effectiveness of our method. Figure 3 also shows that, as the growing speed increases, all three methods can detect events in less time, but the difference between BMV and TV are decreasing. This phenomenon indicates that although BMV always outperforms TV, it benefits more to use BMV in the scenario of gradual events.

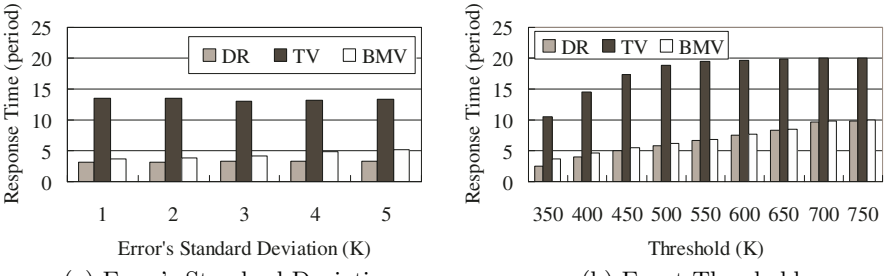
Error's Standard Deviation. We incrementally change the error's standard deviation σ and Fig. 4 (a) shows the effect of changing σ on DR, TV and BMV. The result reveals that the increase of σ does not affect DR and TV much yet raises the response time of BMV. The reason behind this phenomenon is the principles of the three methods. DR and TV just care about whether the absolute reading exceeds the threshold, therefore the error's standard deviation has less to do with these two methods. However, for BMV, since we fix the relative encoding distance as 2 (Table 1), the absolute encoding distance increases at a two times speed of the increasing of σ. Therefore, it is more possible for sensor nodes to generate all-zero bit-strings when the error's standard deviation is high, which will then result in the "dis-match" and finally lead to a slow response. However, thanks to current hardware design techniques, our test dataset well covers the possible values of the error's standard deviation of temperature sensors in real applications. As can be observed in Fig. 4 (a), our method outperforms TV significantly in all cases.

Event Threshold. In this simulation, we alter the event threshold from 350K to 750K in steps of 50K and Fig. 4 (b) displays the effects on the response time of the three methods brought by the threshold. As expected, a lower threshold brings a faster response. Again, the result suggests that BMV is much more efficient than TV in all cases.



(a) Linearly Growing Event (b) Exponentially Growing Event (c) Boundary Growing Event

Fig. 3. Response time v.s. growing speed for three event models



(a) Error's Standard Deviation (b) Event Threshold

Fig. 4. Response time w.r.t. error's standard deviation, event threshold

Voting Percent. As an input parameter for BMV, in this simulation, we vary the voting percent from 10% to 100% to see its effect on BMV's response time. Fig. 5 (a) shows the result. As expected, a higher voting percent brings a slower response.

Relative Encoding Distance and Encoding Length. These two parameters influence the response time of BMV by determining the bit-strings generated by sensor nodes. Figure 5 (b)-(c) shows the impact of relative encoding distance and encoding length on BMV's response time. In both figures, the response time first decreases and then increases. The reason behind this trend is as follows: It is likely to have the "inverse bit" between two bit-strings given a small value of relative encoding distance because it will easily consider the fluctuation brought by the reading error as a jump, but the reading error is independent. And a big value of relative encoding distance will likely lead to the all-zero bit-string; Quite similarly, a small value of encoding length is more likely to generate all-zero bit-strings while a big one is more likely to result in the "inverse bit" between two bit-strings. According to Sect. 3.2, both "inverse bit" and all-zero bit-strings will bring the "dis-match" and thus a slow response.

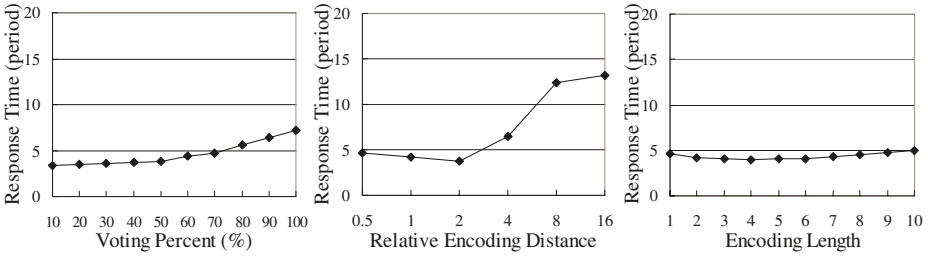


Fig. 5. Response time w.r.t. voting percent, relative encoding distance, encoding length

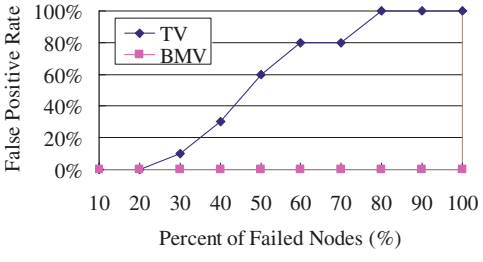


Fig. 6. False Positive Rate w.r.t. Failure Probability

4.3 Report Reliability

Fig. 6 shows that BMV never pass a fake report and the false positive rate of TV increases with the increasing of the percentage of failed nodes. Since the failed nodes always give out high readings above the threshold according to our data set collected by real nodes and the Intel Lab Data[2], the bit-strings generated by failed neighbor nodes are "all-zero", as a consequence, no neighbor will give a positive vote. On the contrary, as for the TV method, the failed neighbors are likely to wrongly pass the fake report due to the threshold-based judgement.

In summary, the BMV method on one hand provides a short response time close to that of the DR method, and on the other hand has a report reliability even higher than that of the TV method.

5 Conclusions

In this paper, we put forward the concept of the gradual event and design a new reliable and fast event detection method – the Bit-string Match Voting (BMV), which first encodes the readings in each sensor node's buffer into bit-strings and then determines whether to support a report by matching the bit-string of the reporting node with that of its neighbor nodes. This method on one hand maximally eliminates false negatives introduced by the threshold-based

voting (TV) method under the scenario of gradual events, thus provides a short response time close to that of the direct reporting (DR) method. On the other hand, considering the failed sensor nodes intend to present fixed reading pattern, the BMV method can avoid false positives as well even all neighbors of a failed reporting node are failed, thus provides the report reliability even higher than that of the TV method. Furthermore, BMV is able to avoid frequent requests for the voting process as well as the repeated and redundant reports of the same event, and thus prolong the life of the network.

Acknowledgements

This work is supported by the National Grand Fundamental Research 973 Program of China under Grant No. 2006CB303000, the Key Program of the National Natural Science Foundation of China under Grant No. 60533110, and the Program for New Century Excellent Talents in University under Grant No. NCET-05-0333.

References

1. Xue, W., Luo, Q., Chen, L., Liu, Y.: Contour map matching for event detection in sensor networks. SIGMOD (2006)
2. Intel Lab Data: `http://berkeley.intel-research.net/labdata/`
3. Werner-Allen, G., Johnson, J., Ruiz, M., Lees, J., Welsh, M.: Monitoring volcanic eruptions with a wireless sensor network. EWSN (2005)
4. Krishnamachari, B., Sitharama, I.: Distributed bayesian algorithms for fault-tolerant event region detection in wireless sensor network. IEEE Transactions on Computers (2004)
5. Abadi, D., Madden, S., Lindner, W.: Reed: Robust, efficient filtering and event detection in sensor networks. VLDB (2005)
6. Subramaniam, S., Kalogeraki, V., Palpanas, T.: Distributed real-time detection and tracking of homogeneous regions in sensor networks. RTSS (2006)
7. Nowak, R., Mitra, U.: Boundary estimation in sensor networks: Theory and methods. In: Zhao, F., Guibas, L.J. (eds.) IPSN 2003. LNCS, vol. 2634, Springer, Heidelberg (2003)
8. Lim, A., Chin Liew, S., Lim, K., Kwoh, L.: Retrieval of subpixel fire temperature and fire area in moderate resolution imaging spectrometer. IGARSS (2002)
9. Yu, X., Niyogi, K., Mehrotra, S., Venkatasubramanian, N.: Adaptive target tracking in sensor networks. CNDS (2004)
10. Eutrophication, `http://toxics.usgs.gov/definitions/eutrophication.html`