Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2014, Article ID 709390, 16 pages
http://dx.doi.org/10.1155/2014/709390

Research Article

Spatial-Temporal Correlative Fault Detection in Wireless Sensor Networks

Zhiping Kang,[1,2,3] Honglin Yu,[1] Qingyu Xiong,[2] and Haibo Hu[2,3]

[1] *Key Laboratory of Optoelectronic Technology and System of Ministry of Education, Chongqing University, Chongqing 400044, China*
[2] *School of Software Engineering, Chongqing University, Chongqing 400044, China*
[3] *Key Laboratory of Ministry of Education for Dependable Service Computing in Cyber Physical Society, Chongqing University, Chongqing 400044, China*

Correspondence should be addressed to Zhiping Kang; kzp@cqu.edu.cn

Received 12 May 2014; Revised 26 October 2014; Accepted 18 November 2014; Published 18 December 2014

Academic Editor: Qi Han

Wireless sensor networks (WSNs) have been used extensively in a range of applications to facilitate real-time critical decision-making and situation monitoring. Accurate data analysis and decision-making rely on the quality of the WSN data that have been gathered. However, sensor nodes are prone to faults and are often unreliable because of their intrinsic natures or the harsh environments in which they are used. Using dust data from faulty sensors not only has negative effects on the analysis results and the decisions made but also shortens the network lifetime and can waste huge amounts of limited valuable resources. In this paper, the quality of a WSN service is assessed, focusing on abnormal data derived from faulty sensors. The aim was to develop an effective strategy for locating faulty sensor nodes in WSNs. The proposed fault detection strategy is decentralized, coordinate-free, and node-based, and it uses time series analysis and spatial correlations in the collected data. Experiments using a real dataset from the Intel Berkeley Research Laboratory showed that the algorithm can give a high level of accuracy and a low false alarm rate when detecting faults even when there are many faulty sensors.

1. Introduction

A wireless sensor network (WSN) consists of spatially distributed autonomous sensors. A WSN operated in self-organization and multihop mode can be used to perceive physical or environmental conditions (such as the temperature, sound level, or pressure) in a target region and to acquire, process, and transmit data describing these conditions. Wireless sensor networks were developed for military applications, but they have been used to monitor and control industrial processes, to monitor machine health, and in other applications [1, 2].

Low-cost and large-scale sensor nodes are often deployed in uncontrolled or even harsh environments, so they are prone to developing faults and becoming unreliable. Faults may occur at different levels of the sensor network for different reasons, such as the depletion of batteries or the failure of a physical component, and faults can lead to incorrect readings being included in a dataset, packet losses during communication, and errors occurring in the middleware and software [3, 4]. From a "data-centric" point of view, faulty sensors may cause dust data to be generated, and this may cause limited resources to be wasted, shorten the network lifetime, affect the analysis of the data and the decisions that are made, and even lead to the failure of the entire network. It is therefore desirable to exclude data from faulty sensors to ensure that the quality of the service is maintained. Faults in WSNs are defined as observations that are not consistent with expected behaviour. Fault tolerance and detection in systems controlling machines and distributed systems have previously been studied intensively [5–8]. Traditional detection methods cannot be directly applied to WSNs because of the limited resources available and the need for the large scale deployment of sensors. It is challenging to develop an accurate detection method that has the characteristics required for use in WSNs.

The aim of the work presented here was to develop a system for locating faulty sensors in a WSN. We propose a localized fault detection algorithm for identifying faulty sensors. The algorithm uses spatial and temporal correlations in WSN data to define normal behaviour and then identifies faults as they occur. The main benefits of our proposed method are outlined below.

(1) The system identifies temporal outliers by performing a time series analysis of each node and performing spatial correlations using neighbouring sensor data and local majority voting. Four new simple and effective spatial strategies are proposed for revising the anomalies. Performance evaluations showed that different strategies and thresholds may be applicable in different situations.

(2) The localized decision-making strategies in the proposed methods are aimed at decreasing communication times and avoiding detection status diffusion, thereby saving energy consumption.

(3) Simulations showed that our detection algorithm does not need to be given the physical positions of the sensors and that it works well even when there are many faulty sensors.

The algorithm was found to be able to successfully identify faulty sensors even when half of the neighbouring sensors failed or when there were few neighbouring sensors. Even if a node had no neighbouring sensors the algorithm degenerated into a time series analysis method but still worked well.

The remainder of this paper is organized as follows. The literature on fault detection in WSNs is reviewed in Section 2. The network model and data model are defined in Section 3. The proposed spatial-temporal fault detection scheme is described in detail in Section 4. Performance evaluations are presented in Section 5 and our conclusions are drawn in Section 6.

2. Related Work

Fault detection in WSNs is an attractive field of research. Krishnamachari and Iyengar [9] developed a Bayesian fault recognition algorithm for disambiguating binary fault features. The algorithm they developed was simple and consumed little energy, but the authors assumed that measurement faults were equally likely to occur at every sensor node and used the binary mode to represent the measurements made by the sensors, and these aspects of the algorithm may limit the scope of its applications. Ni and Pottie [10] used hierarchical Bayesian space-time modelling to detect faults. This method was more complex than the first order linear AR modelling method, but the spatial and temporal correlations were not explicitly calculated in either method, and the existence of such correlations was simply assumed.

A distributed fault detection scheme in which the collected data were spatially correlated was proposed by Chen et al. [11]. In this method, each sensor uses its neighbours to identify its initial status, determines whether its status is either “good” or “faulty” from its neighbours’ statuses, and then sends its final state to the adjacent sensor nodes. The false alarm rate using this method was low when the probability of sensor faults occurring was low. Jiang [12] improved the scheme described above by defining new detection criteria and increased the fault detection accuracy for different average numbers of neighbouring nodes and node failure ratios. However, in both of the methods described [11, 12] each sensor needs to communicate with neighbouring nodes at least three times, and such frequent communication between adjacent nodes consumes a large amount of energy. Lee and Choi [13] proposed a fault detection algorithm in which comparisons between neighbouring nodes were made, and the decision made at each node was disseminated. The use of the dissemination process meant that the network would have a relatively high level of energy consumption.

Some researchers have used artificial intelligence approaches to detect faults. Rassam et al. [14] assessed PCA-based anomaly detection models. Siripanadorn et al. [15] integrated SOM and DWT data to detect anomalies. Nandi et al. [16] introduced two different detection probabilities and used a model selection approach, a multiple model selection approach, and Bayesian model averaging methods to solve the detection problem. Such algorithms consume relatively large amounts of resources.

Sharma et al. [17] described and analysed four qualitatively different classes of fault detection method (rule-based, LLSE, time series forecasting, and HMMs) using real-world datasets. They found that each of the four method types sat at a different point on the accuracy/robustness spectrum. Their evaluation showed that no single method perfectly detected the different types of faults and that using hybrid methods could help eliminate false positives or false negatives. They also found that using two methods in sequence could allow more low-intensity faults to be detected, at the expense of a slightly higher number of false positives, than could be detected using one method alone.

Zhang et al. [18] introduced five different detection methods that were based on either spatial or temporal correlations or on both. Out of these algorithms, the temporal and spatial outlier detection method, the spatial predicted outlier detection method, and the spatial and temporal integrated outlier detection method used both spatial and temporal correlations, but they were complex methods, and each node required rather large amounts of computing resources so that it could receive a number of parameters to allow it to predict its neighbours’ observations from its previous observations. In this paper we will attempt to identify simple and efficient strategies using spatial correlations of time series analyses that have previously been used [17, 18] for the distributed and online detection of faults in a WSN.

3. Network Model and Data Model

We assume that sensors are randomly deployed or placed in predetermined locations and that every sensor has the same transmission range. Each sensor node is able to locate its neighbours within its transmission range using a broadcast/acknowledge protocol. The communication graph for a WSN can be represented as digraph $G(V, E)$, in which V is the set of sensor nodes in the network and E is the set of edges connecting the sensor nodes. Two nodes v_i and v_j are said to have an edge $e_{ij} = (v_i, v_j)$ in the graph if the distance between them is less than the transmission range R_c. In this paper, faulty sensors are defined as those for which the observations do not match the expected behaviour. Nodes with faulty sensors remain capable of receiving, sending, and processing data. Only sensor nodes with permanent communication faults (including a lack of power) are removed from the network.

We define the data model for a sensor network using spatial and temporal correlations. Individual faulty sensor nodes are not relevant to this model. A spatial-temporal correlation will imply that adjacent sensor nodes have made similar observations and that each node has similar values to its previous values in the time series. We let v_i and v_j be neighbouring nodes and let x_i^t and x_i^{t+1} be the sensed data at node v_i at time t_t and t_{t+1}, respectively. The conditions that need to be satisfied by v_i and v_j are then $|x_i^t - x_j^t| \le \theta_1$ and $|x_i^t - x_i^{t+1}| \le \theta_2$, in which θ_1 and θ_2 may vary depending on the application.

4. Spatial-Temporal Correlative Fault Detection

4.1. Definitions. The notation used in this paper is shown below.

(i) S: the set of all sensors,

(ii) N: the total number of sensors,

(iii) v_i: sensor node i,

(iv) Neighbour(v_i): the set of neighbours of v_i,

(v) x_i^t: the measured value for node v_i at time t_t,

(vi) $\tilde{x}_i^t$: the predicted value for node v_i at time t_t,

(vii) C_{ij}: a comparison test between v_i and v_j, only performed if $(v_i, v_j) \in E$. $C_{ij} = \{0, 1\}$, $C_{ij} = C_{ji}$,

(viii) DS_i: the detection state of node v_i, $\mathrm{DS}_i = \{\mathrm{LS}_i, \mathrm{FS}_i\}$. The initial DS_i is NULL,

(ix) LS_i: the likelihood detection state of node v_i, $\mathrm{LS}_i = \{\mathrm{LG}, \mathrm{LF}\}$,

LG: the likelihood state of node v_i is good,

LF: the likelihood state of node v_i is faulty,

(x) FS_i: the final detection state of node v_i, $\mathrm{FS}_i = \{\mathrm{FG}, \mathrm{FF}\}$,

FG: the final state of node v_i is good,

FF: the final state of node v_i is faulty,

(xi) NeigTab(v_i): each node v_i maintains a neighbour table that contains details of its neighbours, their comparison test values c_{ij}, and their detection statuses DS_i,

(xii) the threshold test for the time series: for each node v_i, if $|\tilde{x}_i^t - x_i^t| \le \theta_1$ then $\mathrm{LS}_i = \mathrm{LG}$. Otherwise $\mathrm{LS}_i = \mathrm{LF}$,

(xiii) the threshold test for spatial neighbours: for each node v_i, if $|v_i^t - v_j^t| \le \theta_2$ then $C_{ij} = 0$. Otherwise $C_{ij} = 1$,

(xiv) θ_1, θ_2: two predefined threshold values.

4.2. Algorithm. The proposed fault detection algorithm is based on spatial-temporal correlations and uses the definitions given above. The algorithm can be broken down into four steps.

Step 1. The value for each node v_i is predicted from an analysis of its time series.

A node will have similar values throughout its time series, so temporal correlations can be used to construct an efficient time series model to allow changes in the values to be modelled and forecast. The multiplicative seasonal autoregressive integrated moving average (SARIMA) time series model is a general model for analysing time series. The SARIMA can be written explicitly as [19]

$$\Phi_p(L) A_p(L^s) \nabla^d \nabla_s^D Y_t = \Theta_q(L) B_Q(L^s) e_t, \quad (1)$$

where Y_t is a time series, e_t is a residual sequence, L is a lag operator, $\nabla = 1 - L$ is a “no seasonal difference” operation, $\nabla_s = 1 - L^s$ is a “seasonal difference” operation, $\Phi_p(L)$ and $A_p(L^s)$ are “no seasonal difference” and “seasonal difference” autoregressive polynomials, respectively, $\Theta_q(L)$ and $B_Q(L^s)$ are “no seasonal difference” and “seasonal difference” moving average polynomials, respectively, parameters P, Q, p, and q are polynomial coefficients for the four polynomials just described, respectively, and d and D are nonseasonal difference and seasonal difference frequencies, respectively.

The time analysis involves four substeps. (1) Model identification: the choice of a time series model is focused on the selection of parameters P, D, Q, p, d, q, and s by observing the sample autocorrelations and sample partial autocorrelations. For example, if $s = P = D = Q = 0$, the model degenerates into the autoregressive integrated moving average, but if $s = P = D = Q = d = 0$, the model is called an autoregressive moving average. (2) Parameter estimation: the historical data are used to estimate the parameters for the tentatively selected model. (3) Diagnostic checking: various diagnostic tests are used to check the suitability of the tentatively selected model. (4) Model forecasting: the selected model is used, with the previous observations, to predict the next observation.

The choice of a time series model for the sensor measurements is determined by the nature of the phenomenon being evaluated. It is possible that a complex seasonal model could be the most appropriate for making the predictions, but using a complex model means that more parameters and more computationally intensive training will be required. Determining the best-fitting time series model for modelling the phenomena of interest is an important task for a resource-constrained WSN, but it is not the focus of this work. Our results obtained using real-world datasets showed that the simple AR(p) model used here allows faults in a time series of temperature measurements to be detected effectively.

Step 2. Run a threshold test to determine the preliminary state of each node v_i.

Once each node v_i knows its measured value x_i^t and predicted value $\tilde{x}_i^t$ at time t, we can obtain the likelihood state LS_i of node v_i using a threshold test. If $|\tilde{x}_i^t - x_i^t| \le \theta_1$ then LS_i is set to LG. The state LS_i, based on temporal correlations, is a preliminary test result. Sensors can be determined to be either good or faulty in this phase.

Step 3. Compute a comparison value between each node v_i and each member of Neighbour(v_i).

A comparison test result C_{ij} is generated by sensor v_i based on the measurements for its neighbour v_j using a predefined threshold value θ_2. If $|v_i^t - v_j^t| \leq \theta_2$ then $C_{ij} = 0$. Obviously, a faulty sensor can generate arbitrary measurements, and the comparison test result C_{ij} will then exceed θ_2.

Step 4. Analyse the results using different judging rules to determine the final state of node v_i.

Each sensor sends its preliminary state to all of its neighbours. There are four possible relationships between node v_i and its neighbours, and they are defined in (2). Consider

$$\begin{gathered} g_i^t = \text{number}\left(\sum_{v_j \in \text{Neighbours}(v_i) \text{ and } \text{DS}_j = \text{LG}} C_{ij} = 0\right), \\ h_i^t = \text{number}\left(\sum_{v_j \in \text{Neighbours}(v_i) \text{ and } \text{DS}_j = \text{LF}} C_{ij} = 1\right), \\ y_i^t = \text{number}\left(\sum_{v_j \in \text{Neighbours}(v_i) \text{ and } \text{DS}_j = \text{LG}} C_{ij} = 1\right), \\ w_i^t = \text{number}\left(\sum_{v_j \in \text{Neighbours}(v_i) \text{ and } \text{DS}_j = \text{LF}} C_{ij} = 0\right). \end{gathered} \tag{2}$$

Obviously, g_i^t and h_i^t imply that node v_i may be good, and y_i^t and w_i^t imply that node v_i may be faulty. The relationship between node v_i and its neighbours can follow the pattern shown in Figure 1. The node is marked with its preliminary state LG or LF, and the edge is marked with the comparison test result C_{ij}.

The final state of node v_i depends on its neighbours' states and their comparison test results. According to the voting strategy, if the majority of the neighbours conclude that v_i is good, it is considered to be fault-free. We use the rules shown in (3)–(6) to determine the final state of node v_i to illustrate different detection effects.

Rule 1a. Consider

$$\text{FS}_i = \begin{cases} \text{FG}, & (g_i^t - y_i^t) + (h_i^t - w_i^t) > 0 \\ \text{ToFS}(\text{LS}_i), & \text{otherwise.} \end{cases} \tag{3}$$

Rule 1b. Consider

$$\text{FS}_i = \begin{cases} \text{FG}, & (g_i^t - y_i^t) + (h_i^t - w_i^t) > 0 \\ \text{ToFS}(\text{LS}_i), & g_i^t = h_i^t = y_i^t = w_i^t = 0 \\ \text{FF}, & \text{otherwise.} \end{cases} \tag{4}$$

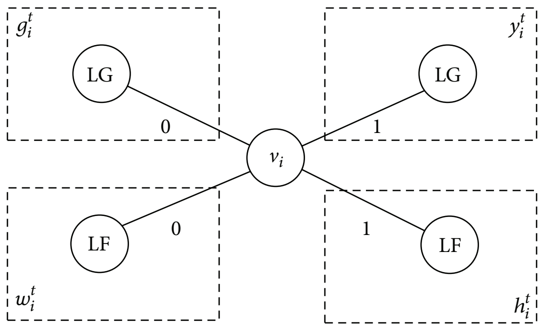


Figure 1: Relationships between node v_i and its neighbours.

Rule 2a. Consider

$$\text{FS}_i = \begin{cases} \text{FG}, & g_i^t - y_i^t > 0 \\ \text{ToFS}(\text{LS}_i), & \text{otherwise.} \end{cases} \tag{5}$$

Rule 2b. Consider

$$\text{FS}_i = \begin{cases} \text{FG}, & g_i^t - y_i^t > 0 \\ \text{ToFS}(\text{LS}_i), & g_i^t = h_i^t = y_i^t = w_i^t = 0 \\ \text{FF}, & \text{otherwise.} \end{cases} \tag{6}$$

Function ToFS(LS$_i$) will convert LG (or LF) into FG (or FF). The criteria shown above mean that there is a common implied condition: if $g_i^t = h_i^t = y_i^t = w_i^t = 0$ then $\text{FS}_i = \text{ToFS}(\text{LS}_i)$. This means that the final state of node v_i is based on its own preliminary state LS$_i$ when node v_i does not have any neighbouring nodes. This condition may decrease the likelihood of false alarms occurring if the majority of the network nodes are faulty.

The threshold values θ_1, θ_2 and the time series model parameters are preset in the node process unit at the time of deployment. The consistency of the diagnosis is not checked by propagating states because such a validation process will consume a large amount of communication or computing resources and even cause detection errors. The fault detection algorithm described above is summarized below.

Spatial-Temporal Correlative Fault Detection Algorithm (STCFD). We have the following.

Step 1. Consider the following.

(1) Each sensor node v_i establishes its own NeigTab(v_i) and sets predefined thresholds θ_1 and θ_2.

(2) Parameters p, d, q, P, D, Q, and s are set for the time series model for the sensor nodes.

(3) The predicted value for every node v_i is estimated using $\bar{x}_i^t = \text{SARIMA}(p, d, q)(P, D, Q)s$.

Step 2. Consider the following.

(1) The difference between the value for each node v_i and its predicted value at time t, $|\bar{x}_i^t - x_i^t|$, is calculated.

(2) IF $|\tilde{x}_i^t - x_i^t| \le \theta_1$ THEN LS_i = LG ELSE LS_i = LF.

(3) The values x_i^t and LS_i are sent to Neighbour(v_i) and NeigTab(v_i) is updated.

Step 3. Consider the following.

(1) The differences between the value for node v_i and the values for its neighbours at time t, $|x_i^t - x_j^t|$, are calculated.

(2) IF $|x_i^t - x_j^t| \le \theta_2$ THEN $C_{ij} = 0$ ELSE $C_{ij} = 1$.

(3) The C_{ij} value for each node v_i in NeigTab(v_i) is updated.

Step 4. Consider the following.

(1) g_i^t, h_i^t, y_i^t, and w_i^t are calculated for each node v_i.

(2) *Rule 1a*

(a) IF $(g_i^t - y_i^t) + (h_i^t - w_i^t) > 0$ THEN FS_i = FG, ELSE.

(b) FS_i = ToFS(LS_i).

Rule 1b

(a) IF $(g_i^t - y_i^t) + (h_i^t - w_i^t) > 0$ THEN FS_i = FG.

(b) IF $g_i^t = h_i^t = y_i^t = w_i^t = 0$ THEN FS_i = ToFS(LS_i).

(c) OTHERWISE FS_i = FF.

Rule 2a

(a) IF $(g_i^t - y_i^t) > 0$ THEN FS_i = FG, ELSE.

(b) FS_i = ToFS(LS_i).

Rule 2b

(a) IF $(g_i^t - y_i^t) > 0$ THEN FS_i = FG.

(b) IF $g_i^t = h_i^t = y_i^t = w_i^t = 0$ THEN FS_i = ToFS(LS_i).

(c) OTHERWISE FS_i = FF.

(3) Output all nodes with DS_i = FF.

4.3. Example. In this section we will present an example to illustrate our algorithm. A partial set of sensor nodes in a wireless sensor network with some faulty nodes is shown in Figure 2. We examined nodes v_1–v_{12}. If two nodes are neighbours they are connected by a line. For convenience, each edge is marked with the value C_{ij} and each node v_i is marked with its preliminary state LS_i.

The results of performing the detection algorithm are shown in Table 1. The final detected states and the preliminary states of the nodes were consistent except for nodes v_4 and v_8. Nodes $v_1, v_2, v_3, v_5, v_6, v_9, v_{10}, v_{11}$, and v_{12} were found to be good, and node v_7 was faulty. Node v_4 was found to be faulty in the time series analysis but it was found to be good using all four of our algorithm rules because most of its neighbouring nodes believed that it was good.

Node v_8 was first considered to be good but was then found to be faulty under some of the rules or good under other rules. LS_8 = LG and $(g_8^t + h_8^t) - (y_8^t + w_8^t) = 2 - 3 = -1 < 0$ is considered to be good using Rule 1a but faulty using Rule 1b. Similarly, it is considered to be good using Rule 2a but faulty using Rule 2b. The differences between the results found using different rules may affect the detection accuracy of the algorithm, and this will be discussed in detail in Section 5.

Table 1: Detection process and results for the algorithm used on the wireless sensor network shown in Figure 2.

v_i	g_i^t	h_i^t	y_i^t	w_i^t	LS_i	Rule 1a FS_i	Rule 1b FS_i	Rule 2a FS_i	Rule 2b FS_i
v_1	3	1	0	0	LG	FG	FG	FG	FG
v_2	2	0	0	1	LG	FG	FG	FG	FG
v_3	4	0	0	0	LG	FG	FG	FG	FG
v_4	2	1	1	0	**LF**	**FG**	**FG**	**FG**	**FG**
v_5	2	1	1	0	LG	FG	FG	FG	FG
v_6	2	0	0	1	LG	FG	FG	FG	FG
v_7	0	1	3	0	LF	FF	FF	FF	FF
v_8	2	0	3	0	**LG**	**FG**	**FF**	**FG**	**FF**
v_9	3	0	0	0	LG	FG	FG	FG	FG
v_{10}	2	1	0	0	LG	FG	FG	FG	FG
v_{11}	2	1	1	0	LG	FG	FG	FG	FG
v_{12}	2	0	1	0	LG	FG	FG	FG	FG

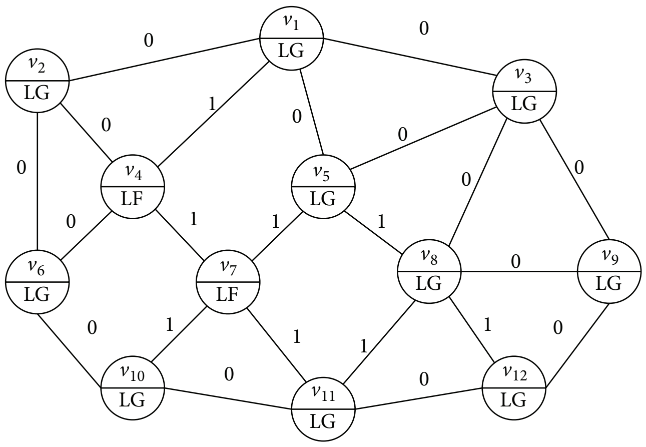


Figure 2: Illustration of a fault detection algorithm for a wireless sensor network.

5. Performance Evaluation

The proposed STCFD depends on a number of parameters (the threshold values θ_1 and θ_2, the average number of neighbours, and the number of faulty nodes in a target area). The performance of the STCFD was evaluated using a real-world dataset with different values for the parameters. We assumed that faults were independent of each other in the experiments.

We used the detection rate (DR) and the false positive rate (FPR) to evaluate the detection performance. The DR is the ratio of the number of detected faulty sensor nodes to the total number of faulty nodes. The FPR is the ratio of the number of fault-free sensor nodes that are diagnosed as faulty to the total number of fault-free nodes. An effective detection technique should achieve a high DR and a low FPR. Here, nodes with some transient faults were treated as fault-free nodes. In the time series analysis we forecast the sensor measurements for time $t + 1$ from the measurements up to time t. We used EViews 6.0 for the time series analysis and Matlab 7.0 as a computing tool.

5.1. Real-World Dataset from the Intel Berkeley Research Laboratory. The Intel Berkeley Research Laboratory (IBRL) dataset was used as a real world dataset for testing the proposed model. The data were collected from 54 Mica2Dot sensors that were deployed in the IBRL between 28 February and 5 April 2004. The position of each node in the deployment area is shown in Figure 3. The Mica2Dot sensors had weather boards and collected time-stamped topology information and humidity, temperature, light, and voltage values once every 31 s. The data were collected using a TinyDB in-network query processing system, which was built in the TinyOS platform. The IBRL dataset included a log of about 2.3 million readings from these sensors [20].

The temperature dataset for two consecutive days (from 00:00 to 24:00), 28 February and 29 February, was selected from the experimental data. The data from 28 February were used to estimate the SARIMA model parameters. The fault detection methods were evaluated using the data from 29 February. The transmission range of the nodes was chosen to ensure that the sensors had different average numbers of neighbours in the simulation runs. An example of the network topology is shown in Figure 4. The relationships between the transmission range R_c, the average number of neighbours, and the maximum difference between neighbours are shown in Table 2. We used the transmission range R_c (rather than the average number of neighbours) for the mapping relationships in the following discussion.

5.2. Data Preprocessing. The proposed STCFD is a distributed parallel algorithm, and it requires time synchronized data to be input. However, different nodes cannot collect data at the same time because the Mica2Dot sensor would miss some of the data packages. The IBRL dataset could not therefore be directly used in the STCFD, and we had to preprocess the data before it was input to the algorithm. We used a smoothing window to modify the original data to keep the gathered data synchronized. The smoothing window outputs the average value, giving samples at specified time intervals. The smoothing window size could be set at 10, 20, 30, or 60 min, as required.

The raw IBRL data are shown in Figure 5(a), and the data processed using smoothing window of 30 min is shown in Figure 5(b). The smoothing window was 30 min in the analysis described below. Each sensor node could acquire 48 time series samples between 00:00 and 24:00 on 29 February.

Table 2: Relationships between the transmission range R_c and the average number of neighbours and the maximum difference between neighbours.

Transmission range (R_c)	Average number of neighbours	Maximum difference
6 m	3.37	2.06°C
8 m	5.67	2.39°C
10 m	8.19	2.39°C
12 m	10.56	2.49°C

Analysing the 48 samples from each sensor node on 28 February easily showed that the samples satisfied AR(2), and the desired prediction data were easily obtained. For example, the sampling data and their forecasts for node 1 on 29 February are shown in Figure 6.

Some artificial anomalies with slightly deviating statistical characteristics were inserted to allow the anomaly detection models to be evaluated using these samples. The maximum temperature in the 2592 samples from 54 nodes on 29 February was 23.0°C and the minimum was 14.9°C. Taking the maximum difference between neighbouring sensors into account, random measured values were generated using the temperature ranges (0, 12) and (26, 40) as faults, and they were inserted into the normal dataset.

5.3. Experimental Results. The DR and FPR values produced in simulations using different parameters and rules were compared. We assumed that the network was available when the number of faulty nodes was less than half the total number of nodes in the WSN. Sensors were randomly chosen to be faulty, and the number of faulty sensors was set in the range 1–27. We will now discuss the effects on the algorithm of using different parameters in the experiments described below. Each experimental result was the average of 30 independent runs.

5.3.1. Experiment I: Variable θ_1, $\theta_2 = 2.5$, and $R_c = 10\,m$. In experiment I we determined the faulty node detection rate and false alarm rate using different θ_1 values and algorithm Rule 1a, Rule 1b, Rule 2a, and Rule 2b. The θ_1 parameter was set at 0.5, 1.5, 3, 5, 7, or 9, and θ_2 and R_c were set at 2.5 and 10 m, respectively.

As can be seen in Figures 7 and 8, increasing the number of faulty nodes caused the DR using Rule 1a and Rule 1b to decrease gradually. The DR using Rule 1a was higher when θ_1 was 1.5 or 3 than when it was 0.5, 5, or 7 (Figure 7). Increasing θ_1 caused the FPR using Rule 1a to decrease. The FPR decreased from 6.26% to 2.39% when θ_1 was 0.5, but the FPR using Rule 1a was only 0% when θ_1 was 7.

In Figure 8, larger θ_1 values gave better DR values but not larger FPR values using Rule 1b. Increasing θ_1 from 1.5 to 3 caused the FPR to fall sharply. Increasing θ_1 from 3 to 5 and then to 7 caused the FPR to change little. Increasing θ_1 to 9 caused the FPR using Rule 1b to increase rather than decrease.

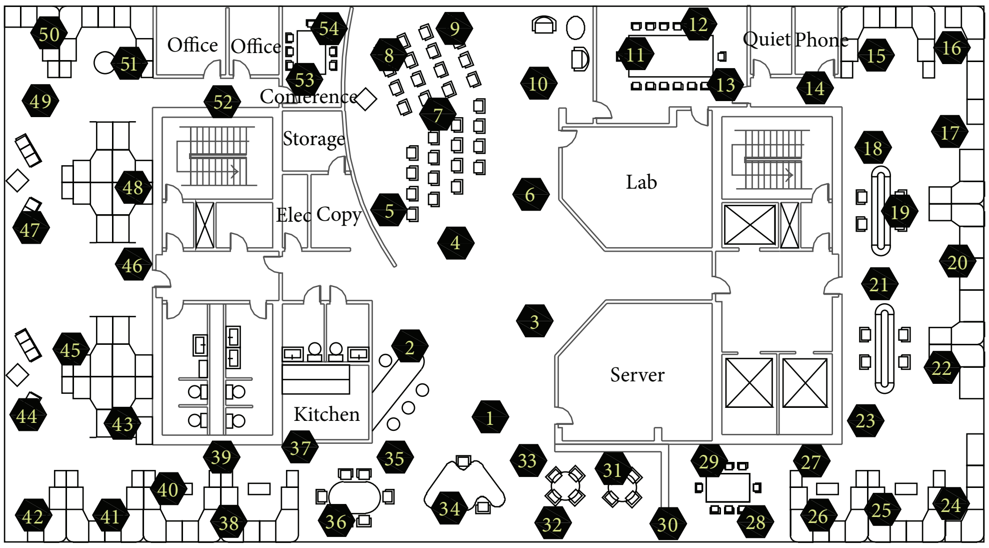


FIGURE 3: Sensor nodes deployed to collect the Intel Berkeley Research Laboratory dataset.

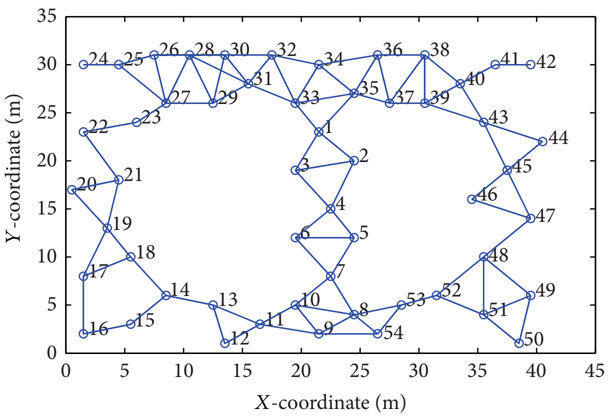


FIGURE 4: An illustration of the network topology when R_c = 6 m in the Intel Berkeley Research Laboratory dataset.

It can be seen from Figures 9 and 10 that the DR using Rule 2a and Rule 2b almost stayed the same as the number of faulty nodes was changed. However, the DR using Rule 2a was affected by the θ_1 value and increasing θ_1 decreased the detection rate. The DR using Rule 2a was about 99.96% when θ_1 was 0.5, but the DR was only about 92% when θ_1 was 7. Fortunately, increasing θ_1 decreased the FPR. The FPR was approximately 0% when θ_1 was 7, and when θ_1 was 0.5 the FPR using Rule 2a continued to increase (from 2.2% to 3.6%).

The θ_1 value hardly affected the DR using Rule 2b (Figure 10), and the DR was around 99.95% when θ_1 was 0.5, 1.5, 3, 5, or even 7. However, the FPR using Rule 2b was influenced by the θ_1 value. The FPR using Rule 2b was lower when θ_1 was 1.5 or 3 than when it was 0.5, 5, or 7.

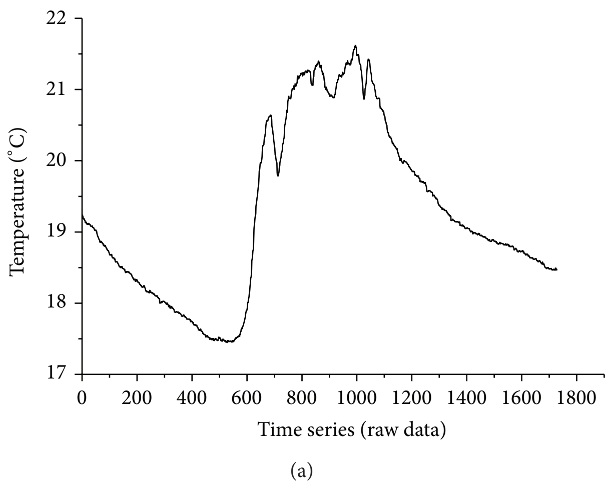
Two main conclusions were drawn from Experiment I. (1) θ_1 can affect the DR and FPR using all four rules, but it is impossible to find a θ_1 value that gives both a high DR and a low FPR. The θ_1 value that gives an optimal effect will depend on the application. (2) The DRs found using algorithm Rule 1a and Rule 1b are related to the numbers of faulty nodes. The DRs found using algorithm Rule 2a and Rule 2b are almost independent of the numbers of faulty nodes. However, the FPRs using Rule 2a and Rule 2b increase as the number of faulty nodes increases.

5.3.2. Experiment II: Variable θ_2, θ_1 = 1.5, and R_c = 10 m. Experiment II gave the faulty node detection rate and false alarm rate using different θ_2 values and algorithm Rule 1a, Rule 1b, Rule 2a, and Rule 2b. The θ_2 values were 1.5, 2.5, 3.5, 4.5, 5.5, and 9, and θ_1 and R_c were set to 1.5 and 10 m, respectively.

It can be seen from Figures 11 and 12 that the DRs obtained using Rule 1a and Rule 1b were similar. The θ_2 value was relatively small and the DR was relatively high when the number of faulty nodes was relatively low. The θ_2 value was relatively small and its DR was relatively low when the number of faulty nodes was relatively high. Increasing the number of faulty nodes increased the θ_2 value and the corresponding DR decreased more slowly.

The DR using Rule 1a was higher when θ_2 was 1.5 than when θ_2 was 3.5 when the number of faulty nodes was ≤11. The DR when θ_2 was 1.5 decreased from 99.638% to 98.808%, but the DR when θ_2 was 3.5 decreased from 99.384% to 98.762%. When the number of faulty nodes was >11, the DR was lower when θ_2 was 1.5 than when θ_2 was 3.5, and the rate decreased more quickly when θ_2 was 1.5 than when θ_2 was 3.5. The DR when θ_2 was 1.5 decreased from 98.43% to 70.078%, but the DR when θ_2 was 3.5 only decreased from 98.436% to 80.725%. It was difficult to find a θ_2 value that gave a better DR than other values gave for all of the different faulty node patterns.

The best θ_2 value was approximately the same for the FPRs using Rule 1a or Rule 1b. The FPR was lower when θ_2 was 2.5, 3.5, 4.5, or 5.5 than when θ_2 was 1.5 or 9 (Figure 11(b)). The FPR was lower when θ_2 was 3.5, 4.5, or 5.5 than when θ_2 was 1.5, 2.5, or 9 (Figure 12(b)).



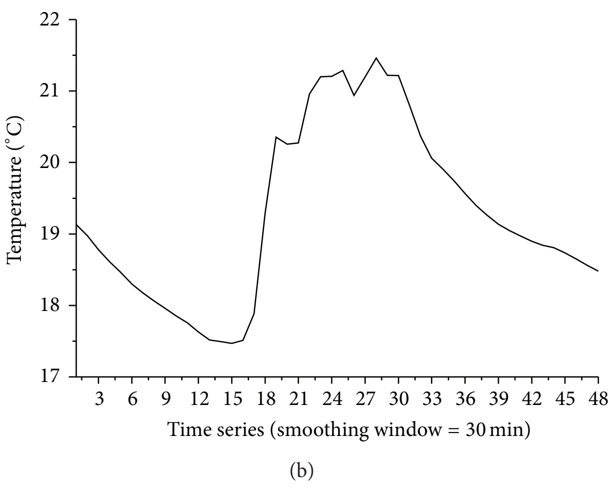


Figure 5: (a) Raw data for node 1 from 00:00 to 24:00 on 29 February and (b) time series samples for node 1 from 00:00 to 24:00 on 29 February produced using a smoothing window of 30 min.

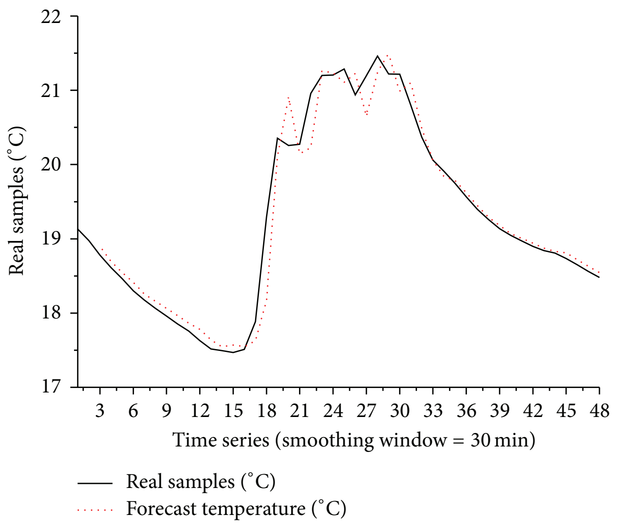


Figure 6: Sampling data and forecasts for node 1 on 29 February (using a smoothing window of 30 min).

The effects on the DR and FPR using Rule 2a and Rule 2b are clearly shown in Figures 13 and 14. Smaller θ_2 values gave higher DRs (up to 100%). Larger θ_2 values gave lower FRPs. The DR was about 100% when θ_2 was 1.5 but only about 81.5% when θ_2 was 9, and the FPR when θ_2 was 1.5 increased from 3.01% to 6.20% but when θ_2 was 9 the FPR increased from 0% to 0.88% (Figure 13).

Two main conclusions were drawn from Experiment II. (1) The best θ_2 value gave optimal effects, but the optimal value depends on the application. (2) The DRs achieved using algorithm Rule 1a and Rule 1b are related to the number of faulty nodes. The DRs achieved using algorithm Rule 2a and Rule 2b are almost independent of the number of faulty nodes but the FPRs increase as the number of faulty nodes increases.

5.3.3. Experiment III: Variable R_c, $\theta_1 = 1.5$, and $\theta_2 = 2.5$. The faulty node detection accuracies achieved using Rule 1a, Rule 1b, Rule 2a, and Rule 2b and the numbers of faulty nodes for different neighbours are shown in Figures 15 and 16. R_c was set to 8 m, 10 m, and 12 m (from Table 2), implying that the average number of neighbours was 5.67, 8.19, and 10.56, respectively.

Increasing R_c from 8 m to 12 m caused the DRs using both Rule 1a and Rule 1b to increase but the FPR to decrease (Figure 15). Increasing the number of faulty nodes caused both the DR and the FPR to decrease, but if the R_c was increased the corresponding FPR decreased more slowly.

The DRs using Rule 2a and Rule 2b were almost independent of the average number of neighbours (Figure 16). Increasing the average number of neighbours caused the FPRs using Rule 2a and Rule 2b to decrease, but the FPRs increased as the number of faulty nodes increased. Even when half of the total number of sensors were faulty the DRs using Rule 2a and Rule 2b were still about 99.95%. The FPR using Rule 2a was 0.13% when R_c was 8 m, 0.04% when R_c was 10 m, and only 0.03% when R_c was 12 m. The FPR using Rule 2b was 4.47% when R_c was 8 m, 1.22% when R_c was 10 m, and only 0.58% when R_c was 12 m.

In brief, Experiment III showed that increasing the number of neighbours may improve the detection performances of algorithm Rule 1a, Rule 1b, Rule 2a, and Rule 2b.

5.3.4. Experiment IV: Comparing the Accuracies of Detection Achieved Using Different Algorithms. In the following experiments we compared the detection performances of our STCFD, a single time series model (STM) [18], and a distributed fault detection (DFD) algorithm [11].

(1) STCFD and STM. We used two scenarios to compare the performances of the STCFD and STM. (1) $\theta_1 = 0.5$, $\theta_2 = 2.5$, and $R_c = 10$ m; (2) $\theta_1 = 5$, $\theta_2 = 2.5$, and $R_c = 10$ m. When $\theta_1 = 0.5$, the STM gave a high detection rate and a high false

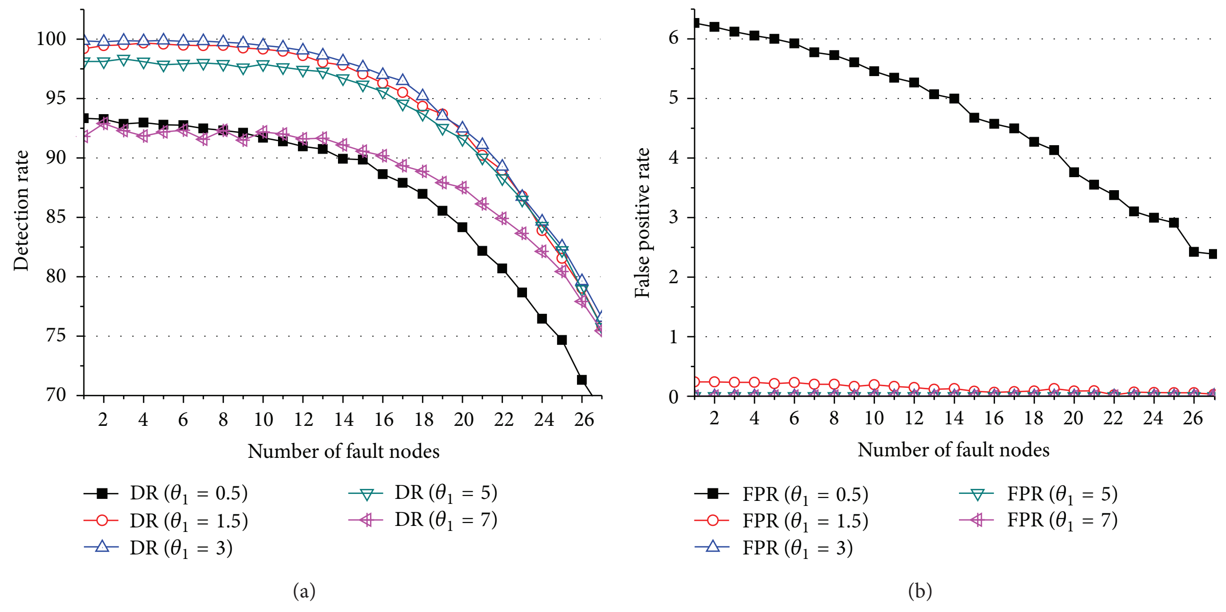


FIGURE 7: (a) Detection rate (DR) for algorithm Rule 1a when $\theta_2 = 2.5$ and $R_c = 10$ m and (b) false positive rate (FPR) for algorithm Rule 1a when $\theta_2 = 2.5$ and $R_c = 10$ m.

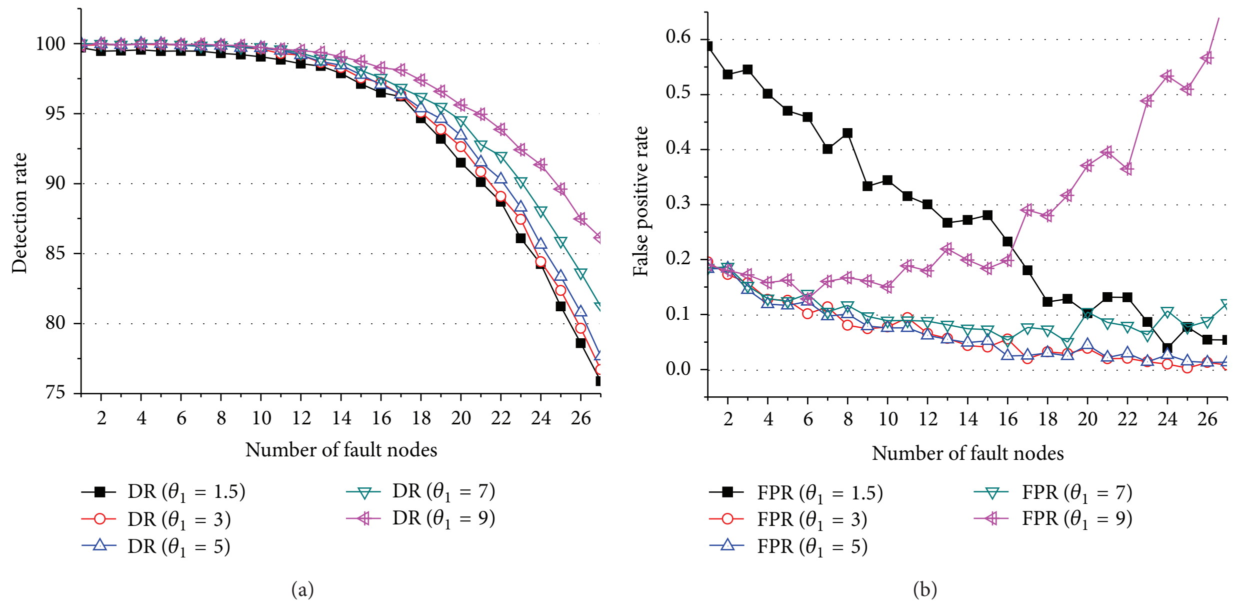


FIGURE 8: (a) Detection rate (DR) for algorithm Rule 1b when $\theta_2 = 2.5$ and $R_c = 10$ m and (b) the false positive rate (FPR) for algorithm Rule 1b when $\theta_2 = 2.5$ and $R_c = 10$ m.

positive rate. When $\theta_1 = 5$, the STM gave a low detection rate and a low false positive rate.

The STM DR was about 100% and the FPR remained at about 8%, and neither was affected by the number of faulty nodes (Figure 17). Increasing the number of failed nodes caused the DRs and FPRs using Rule 1a and Rule 1b to decrease nonlinearly (Figure 17(a)). The DRs decreased from about 93% to about 70%, the FPR using Rule 1a decreased from about 6.27% to 2.39%, and the FPR using Rule 1b decreased from about 7.68% to 2.56%. The DRs using Rule 2a and Rule 2b were about 99.95% and were not related to the number of faulty nodes (Figure 17(b)), the FPR using Rule 2a increased from about 2.19% to 3.63%, and the FPR using Rule 2b increased from about 2.47% to 4.96%, and both of the FPRs were less than the STM FPR (8%). Rule 1a and Rule 1b gave lower false alarm rates than the STM, but at the expense of a lower detection rate. Rule 2a and Rule 2b obviously gave lower false alarm rates than the other methods and maintained a high detection rate that was equivalent to that achieved using the STM.

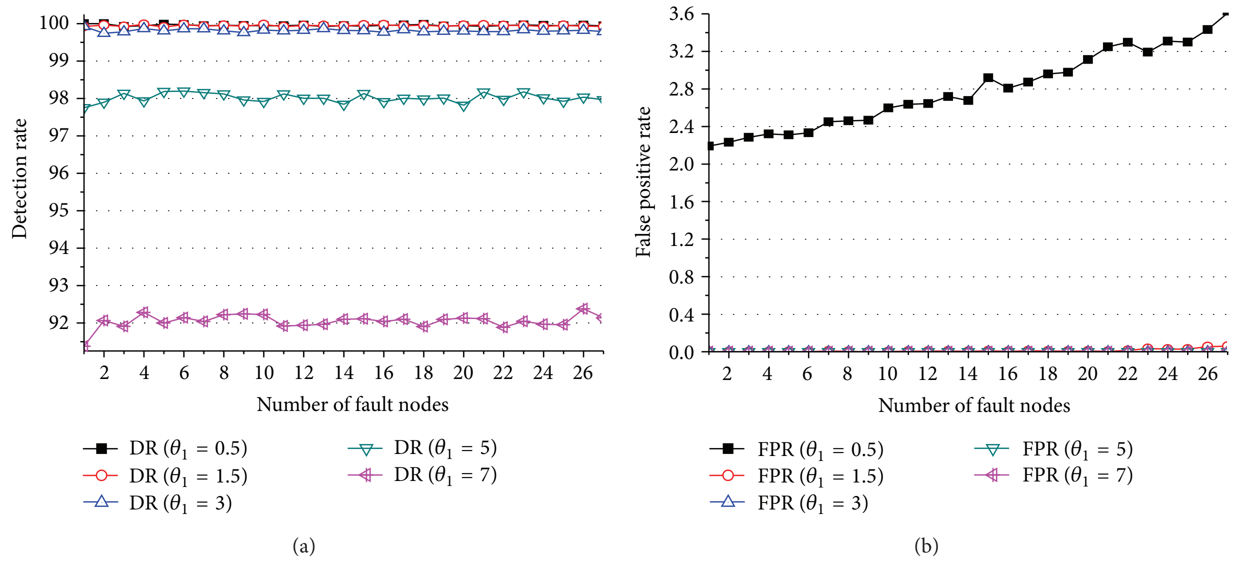


Figure 9: (a) Detection rate (DR) for algorithm Rule 2a when $\theta_2 = 2.5$ and $R_c = 10$ m and (b) the false positive rate (FPR) for algorithm Rule 2a when $\theta_2 = 2.5$ and $R_c = 10$ m.

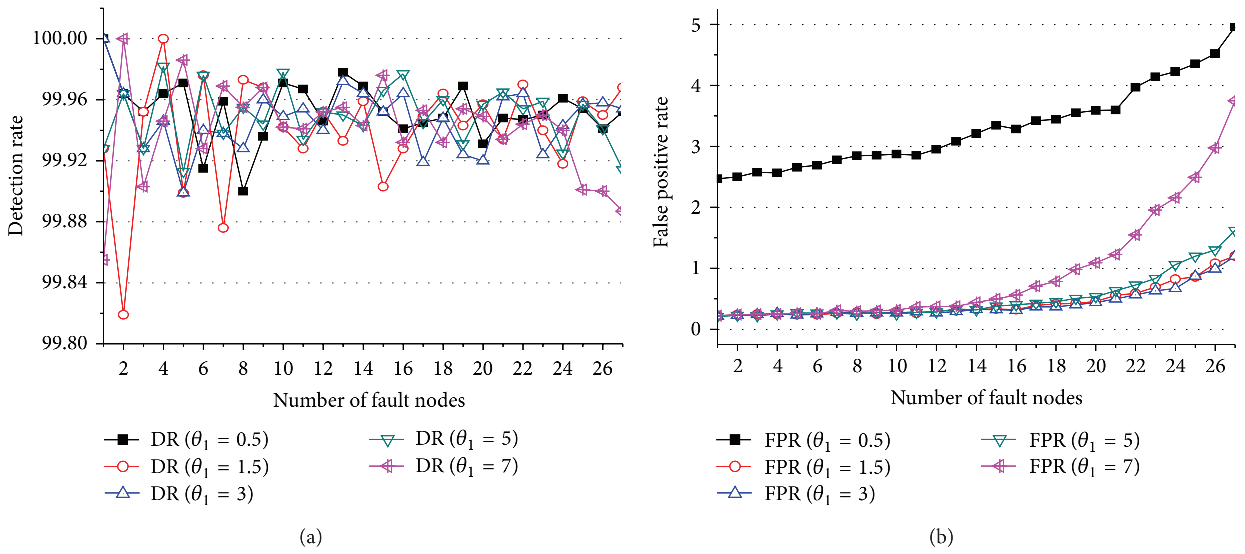


Figure 10: (a) Detection rate (DR) for algorithm Rule 2b when $\theta_2 = 2.5$ and $R_c = 10$ m and (b) the false positive rate (FPR) for algorithm Rule 2b when $\theta_2 = 2.5$ and $R_c = 10$ m.

It can be seen from **Figure 18** that the FPR using the STM was only 0% but the DR using the STM was about 98%. The DR using Rule 1a decreased from 98.11% to 75.76%, and the DR using Rule 2a was about 98%. Neither Rule 1a nor Rule 2a improved the detection accuracy because their DRs were not higher than the DR achieved using the STM.

The DR using Rule 2b was about 99.95%, which was higher than was achieved using the STM (2%). The FPR using Rule 2b only increased from 0.18% to 1.62%. When the number of faults was <16, the DR using Rule 1b was higher than was achieved using the STM (1.90%–0.03%), and the corresponding FPR was also higher than was achieved using the STM (0.18%–0.01%). These results imply that the FPRs were increased less than the DRs using algorithm Rule 2b and Rule 1b relative to using the STM, so Rule 2b and Rule 1b gave better detection performances than the STM. Experiment IV showed that we can produce an STCFD rule that will increase the detection rate or decrease the false positive rate under different conditions, to improve the detection accuracy over that achieved using the STM.

(2) STCFD and DFD. Different types of parameters are available, so it is difficult to compare the different algorithms exactly. We chose relatively moderate values for appropriate parameters so that we could compare the STCFD and DFD algorithms in general terms. We used $\theta_1 = 1.5$, $\theta_2 = 2.5$,

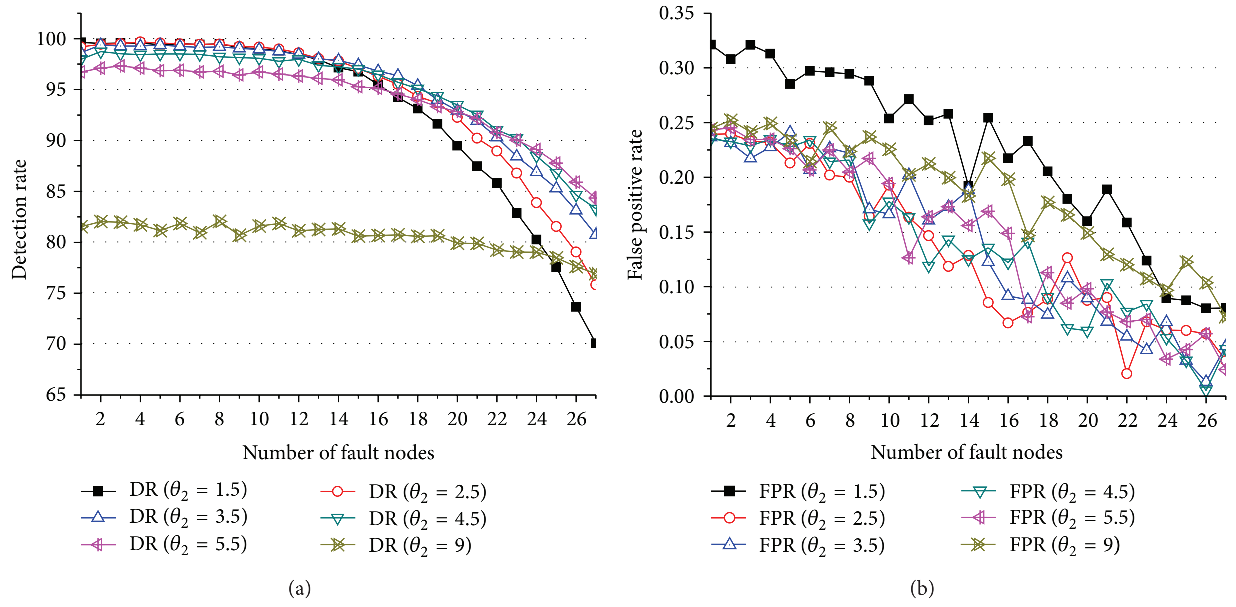


FIGURE 11: (a) Detection rate (DR) for algorithm Rule 1a when $\theta_1 = 1.5$ and $R_c = 10$ m and (b) the false positive rate (FPR) for algorithm Rule 1a when $\theta_1 = 1.5$ and $R_c = 10$ m.

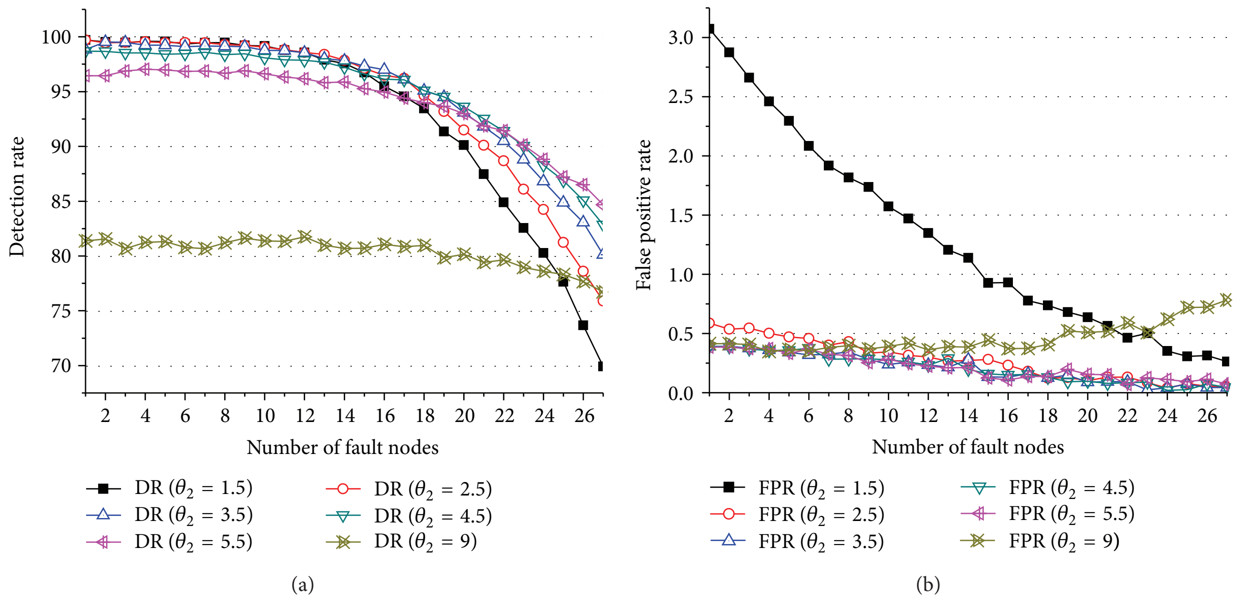


FIGURE 12: (a) Detection rate (DR) for algorithm Rule 1b when $\theta_1 = 1.5$ and $R_c = 10$ m and (b) the false positive rate (FPR) for algorithm Rule 1b when $\theta_1 = 1.5$ and $R_c = 10$ m.

and $R_c = 10$ m for the STCFD and $\theta_1 = 2.5$, $\theta_2 = 0.1$, and $R_c = 10$ m for the DFD.

The DRs and FPRs achieved using the STCFD and DFD methods plotted against the numbers of faulty sensors are shown in Figure 19. The DR using DFD, Rule 1a, and Rule 1b decreased as the number of faulty nodes increased. When the number of faulty nodes was <12, the DR using the DFD was >99.0% and was slightly higher than or at least equal to the DRs achieved using Rule 1a and Rule 1b. When the number of faulty nodes increased from 12 to 24, the DR using the DFD decreased sharply until it fell below 70%, but the DRs using Rule 1a and Rule 1b only decreased to about 84%. Some sensors may not have enough neighbours for a correct and complete analysis to be achieved, so faulty sensors were not diagnosed as faulty using the DFD. Increasing the number of faulty sensors caused the FPR using the DFD to increase from 0.65% to 2.4% and then to decrease to 1.62%.

The DRs and FPRs using Rule 2a and Rule 2b were higher than the DR using the DFD. The DRs using Rule 2a and Rule 2b were still about 99.95% when 27 sensors were faulty.

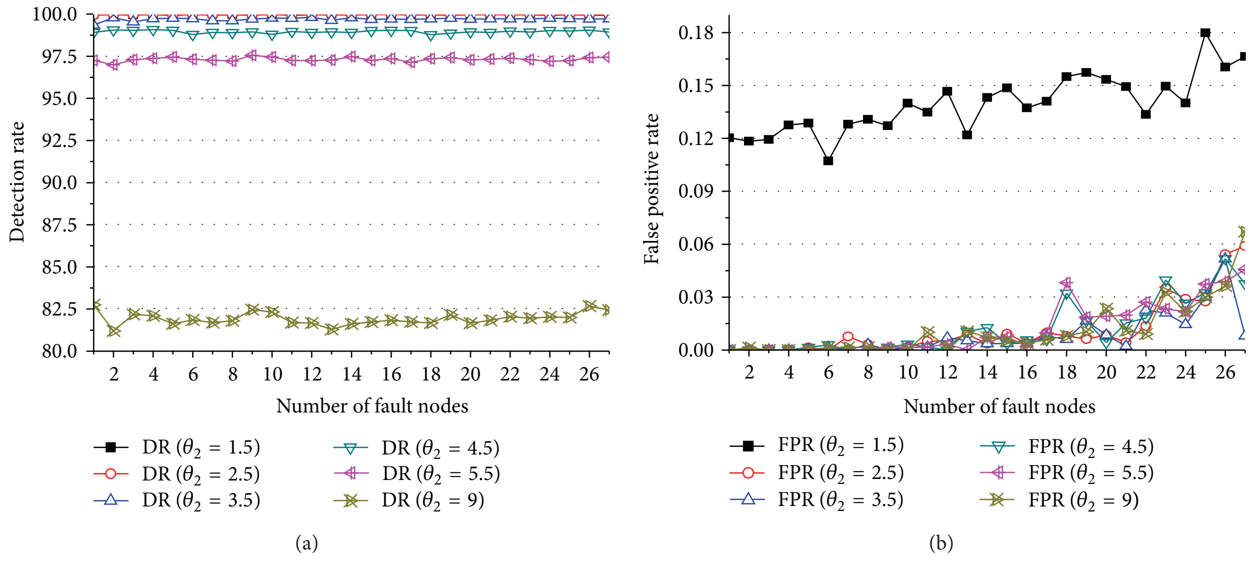


FIGURE 13: (a) Detection rate (DR) for algorithm Rule 2a when $\theta_1 = 1.5$ and $R_c = 10$ m and (b) the false positive rate (FPR) for algorithm Rule 2a when $\theta_1 = 1.5$ and $R_c = 10$ m.

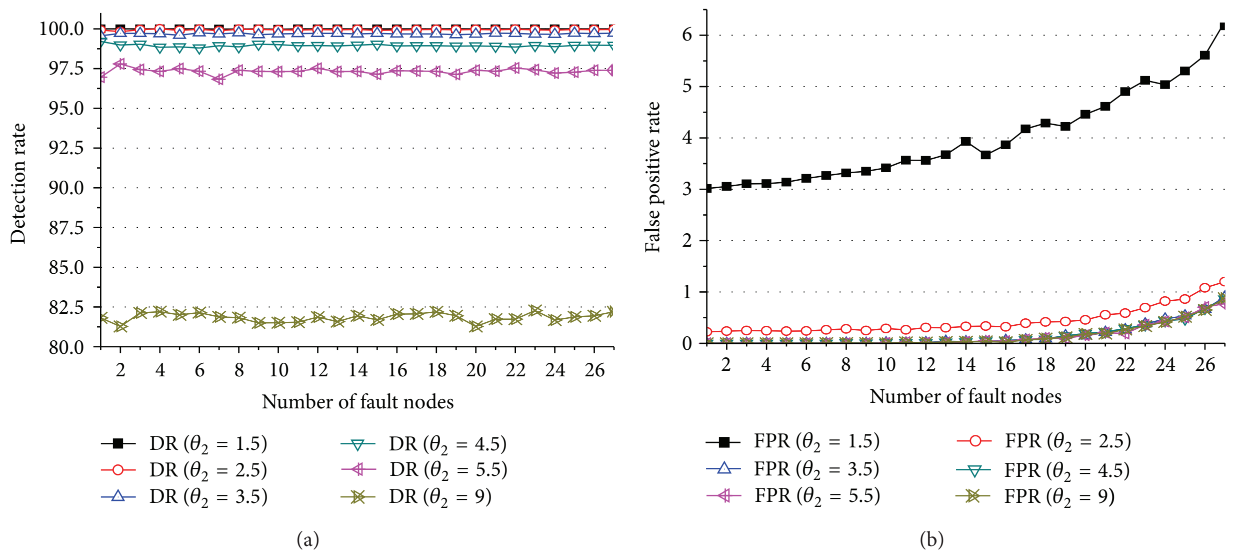
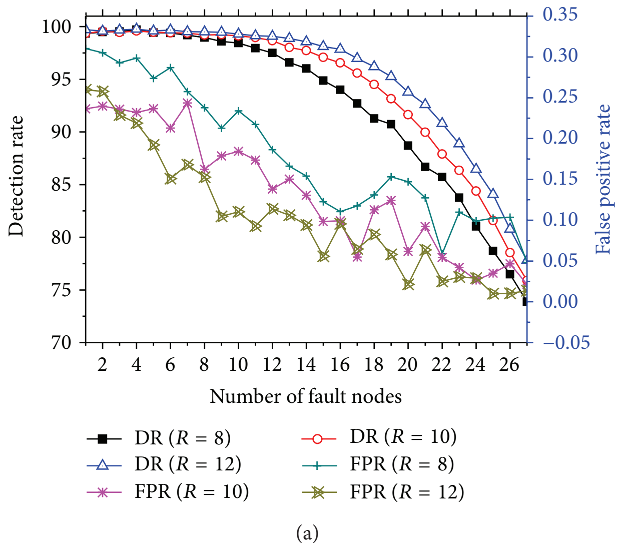


FIGURE 14: (a) Detection rate (DR) for algorithm Rule 2b when $\theta_1 = 1.5$ and $R_c = 10$ m and (b) the false positive rate (FPR) for algorithm Rule 2b when $\theta_1 = 1.5$ and $R_c = 10$ m.

Furthermore, the FPR using Rule 2a increased from 0% to 0.06% and the FPR using Rule 2b increased from 0.22% to 1.20%. It is clear that our STCFD gives a better fault detection accuracy than does the DFD, especially when the number of faulty nodes is higher than in the target network.

5.4. Complexity Analysis. There are usually strict constraints on resources when WSN algorithms are performed, and this is a significant difference from other scenarios in which such algorithms are used. A very accurate but resource-hungry method is hardly applicable to WSNs. We analysed the complexity of the STCFD method, including the communication, computation, and memory demands.

The computational complexity was found to depend mainly on the time series model used to calculate C_{ij}, g_i^t, h_i^t, y_i^t, and w_i^t for the spatial neighbours. The SARIMA model gave a different level of complexity. Considering AR(p) in our experiment, the computational complexity was $O(p)$, where p is the regression coefficient. The maximum computational complexity at each node was, therefore, $O(p + c \cdot q)$, where c is a constant and q is the number of neighbours. The communication complexity of the STCFD was low, each node only sending one of its own observations to its neighbours to allow a spatial comparison to be performed in each run period, which is less than the communication times required in previous algorithms [11–13]. The memory complexity came mainly from keeping



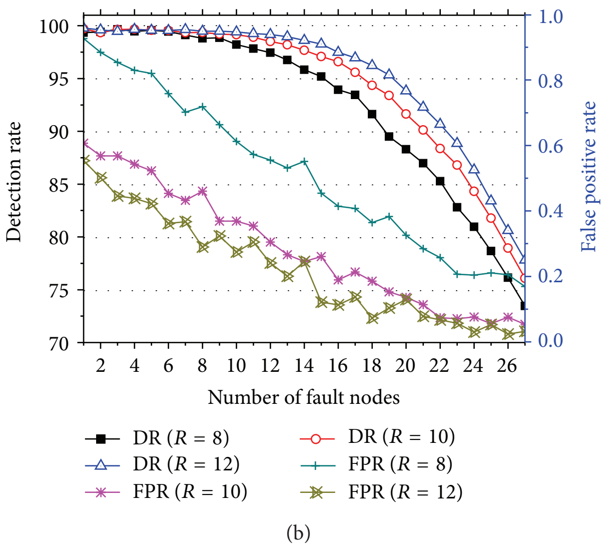
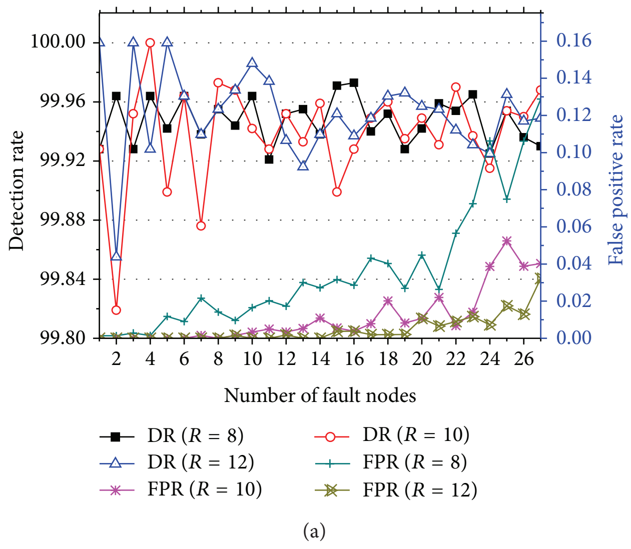


Figure 15: (a) Detection rate (DR) and false positive rate (FPR) for algorithm Rule 1a when $\theta_1 = 1.5$ and $\theta_2 = 2.5$ and (b) DR and FPR for algorithm Rule 1b when $\theta_1 = 1.5$ and $\theta_2 = 2.5$.



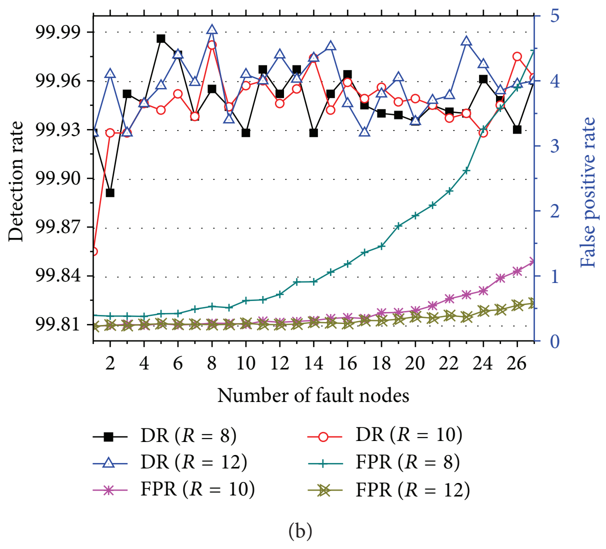


Figure 16: (a) Detection rate (DR) and false positive rate (FPR) for algorithm Rule 2a when $\theta_1 = 1.5$ and $\theta_2 = 2.5$ and (b) DR and FPR for algorithm Rule 2b when $\theta_1 = 1.5$ and $\theta_2 = 2.5$.

previous observations and algorithm parameters (selected SARIMA parameters, $\tilde{x}_i^t, \mathrm{DS}_i, \theta_1, \theta_2, C_{ij}, \ldots$) in memory. This may be represented as $O(p + d)$, where d is the number of variables required. The overheads involved in storing the temporal and spatial correlation parameters were found to be negligible.

6. Conclusions

In this paper we present a method for detecting distributed faults in coordinate-free WSNs. The method is based on spatial and temporal correlations in WSN data. Firstly, we used a time series analysis method to determine the preliminary detection state for each node. Secondly, we used spatial correlations of adjacent nodes to obtain a comparison test result C_{ij} for the neighbours of each node. Finally, we used four rules (Rule 1a, Rule 1b, Rule 2a, and Rule 2b) to determine the final detection results. Our algorithm involves three parameters, a temporal threshold value θ_1, a spatial threshold value θ_2, and the average number of neighbours.

Experimental results obtained using a real database gave the following results. (1) The four rules in our algorithm give different detection accuracies. Rule 1a and Rule 1b use the inequality $(g_i^t - y_i^t) + (h_i^t - w_i^t)$, and the DRs and FPRs they yield decrease nonlinearly. They may work better when there are only small numbers of faulty nodes. Rule 2a and Rule 2b use $(g_i^t - y_i^t)$ and give DRs that are independent of the number of faulty nodes. However, the FPRs they produce may rise as the

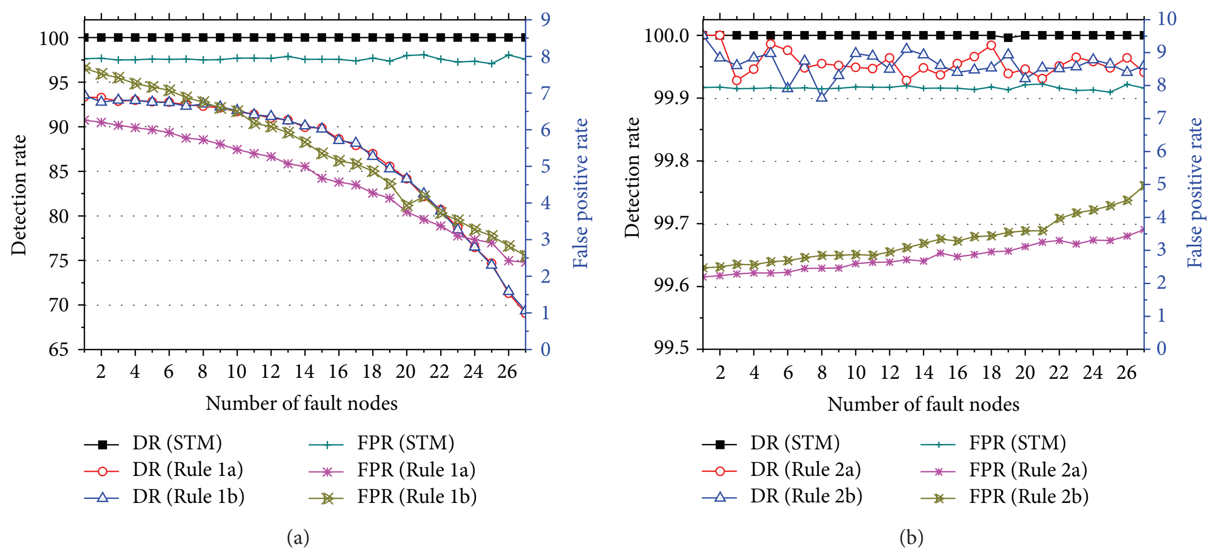


Figure 17: (a) Detection rate (DR) and false positive rate (FPR) for the single time series model (STM) and the algorithm Rule 1a and Rule 1b when $\theta_1 = 0.5$, $\theta_2 = 2.5$, and $R_c = 10$ m and (b) DR and FPR for the STM and the algorithm Rule 2a and Rule 2b when $\theta_1 = 0.5$, $\theta_2 = 2.5$, and $R_c = 10$ m.

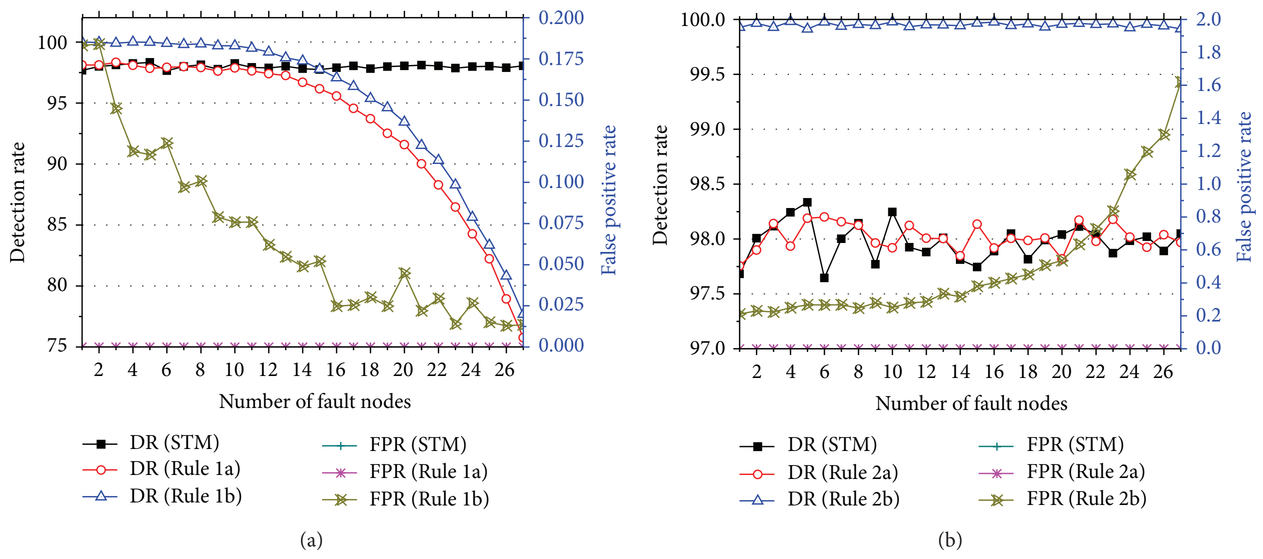


Figure 18: (a) Detection rate (DR) and false positive rate (FPR) for the single time series model (STM) and the algorithm Rule 1a and Rule 1b when $\theta_1 = 5$, $\theta_2 = 2.5$, and $R_c = 10$ m and (b) DR and FPR for the STM and the algorithm Rule 2a and Rule 2b when $\theta_1 = 5$, $\theta_2 = 2.5$, and $R_c = 10$ m.

number of faulty nodes increases. (2) The values of θ_1 and θ_2 depend on the specific application, and the larger the average number of neighbours each node has the better. (3) Overall, our localized fault detection algorithm gives a high detection accuracy and a low false alarm rate even when there is a large set of faulty sensors, and our algorithm gives better detection accuracies than the STM and DFD algorithms.

Our algorithm is promising, and we intend to extend it to see how it behaves in extremely large deployments. In further research and simulations, we will use our proposed algorithms in real situations. We will focus on the following four aspects: (1) using more complex data structures, with seasonal, cyclical, and other trends, (2) inserting different fault types (short faults, noise faults, and constant faults) or fault intervals into real data for detection tests, (3) using different SARIMA values for the forecast for each node, rather than uniform values as used in the experiments described above, and (4) optimizing the analysis process to decrease the complexity of the algorithm.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

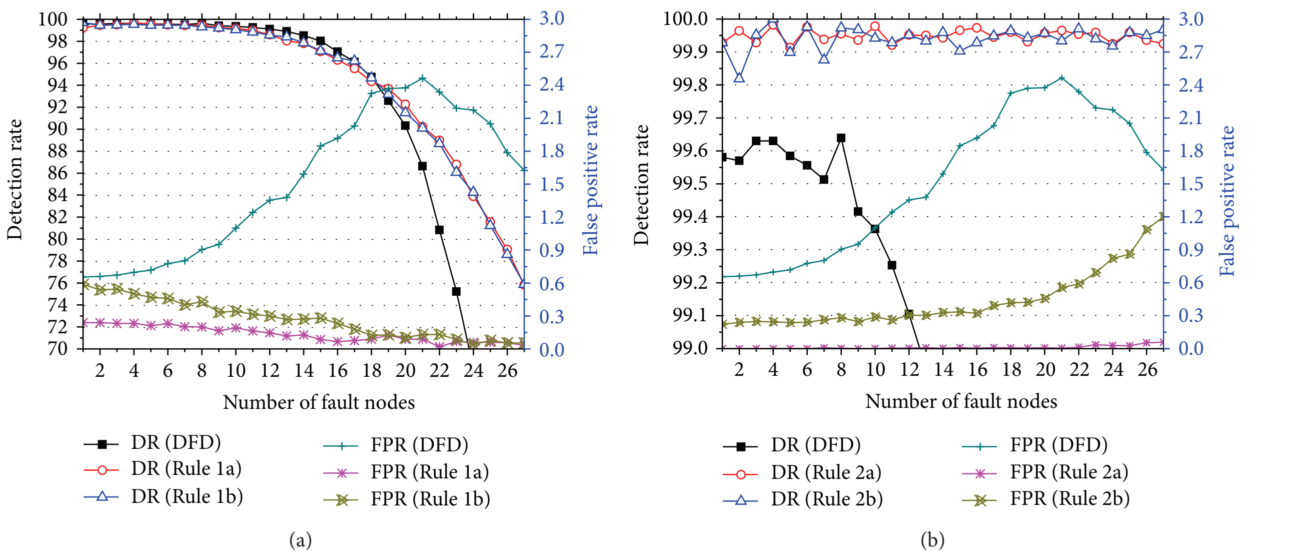


FIGURE 19: (a) Detection rate (DR) and false positive rate (FPR) for the distributed fault detection method (DFD) and the algorithm Rule 1a and Rule 1b and (b) DR and FPR for the DFD and the algorithm Rule 2a and Rule 2b.

Acknowledgments

The work was supported by the key National Natural Science Foundation of China under Grant 91118005, the key Projects in the National Science & Technology Pillar Program during the Twelfth Five-year Plan Period under Grant 2011BAJ03B13, the Fundamental Research Funds for the Central Universities of China under Grant 106112014 CDJZR098801. The real-world dataset was provided by Intel Berkeley Research lab, thanks for their hard work.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “Wireless sensor networks: a survey,” *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002.

[2] D. Puccinelli and M. Haenggi, “Wireless sensor networks: applications and challenges of ubiquitous sensing,” *IEEE Circuits and Systems Magazine*, vol. 5, no. 3, pp. 19–29, 2005.

[3] L. Paradis and Q. Han, “A survey of fault management in wireless sensor networks,” *Journal of Network and Systems Management*, vol. 15, no. 2, pp. 171–190, 2007.

[4] K. Ni, N. Ramanathan, M. N. H. Chehade et al., “Sensor network data fault types,” *ACM Transactions on Sensor Networks*, vol. 5, no. 3, pp. 1–29, 2009.

[5] B. W. Johnson, *Design & Analysis of Fault Tolerant Digital Systems*, Addison-Wesley Longman, 1988.

[6] F. Cristian, “Understanding fault-tolerant distributed systems,” *Communications of the ACM*, vol. 34, no. 2, pp. 56–78, 1991.

[7] M. Blanke and J. Schröder, *Diagnosis and Fault-Tolerant Control*, vol. 115, Springer, New York, NY, USA, 2003.

[8] I. Koren and C. M. Krishna, *Fault-Tolerant Systems*, Morgan Kaufmann, Boston, Mass, USA, 2010.

[9] B. Krishnamachari and S. Iyengar, “Distributed Bayesian algorithms for fault-tolerant event region detection in wireless sensor networks,” *IEEE Transactions on Computers*, vol. 53, no. 3, pp. 241–250, 2004.

[10] K. Ni and G. Pottie, “Sensor network data fault detection using hierarchical Bayesian space-time modeling,” Tech. Rep. TR-69, University of California, 2009.

[11] J. Chen, S. Kher, and A. Somani, “Distributed fault detection of wireless sensor networks,” in *Proceedings of the Workshop on Dependability Issues in Wireless Ad Hoc Networks and Sensor Networks (DIWANS ’06)*, pp. 65–71, September 2006.

[12] P. Jiang, “A new method for node fault detection in wireless sensor networks,” *Sensors*, vol. 9, no. 2, pp. 1282–1294, 2009.

[13] M.-H. Lee and Y.-H. Choi, “Fault detection of wireless sensor networks,” *Computer Communications*, vol. 31, no. 14, pp. 3469–3475, 2008.

[14] M. A. Rassam, A. Zainal, and M. A. Maarof, “One-class principal component classifier for anomaly detection in wireless sensor network,” in *Proceedings of the 4th International Conference on Computational Aspects of Social Networks (CASoN ’12)*, pp. 271–276, Sao Carlos, Brazil, November 2012.

[15] S. Siripanadorn, W. Hattagam, and N. Teaumroong, “Anomaly detection in wireless sensor networks using self-organizing map and wavelets,” *International Journal of Communications*, vol. 4, pp. 74–83, 2010.

[16] M. Nandi, A. Dewanji, B. Roy, and S. Sarkar, “Model selection approach for distributed fault detection in wireless sensor networks,” *International Journal of Distributed Sensor Networks*, vol. 2014, Article ID 148234, 12 pages, 2014.

[17] A. B. Sharma, L. Golubchik, and R. Govindan, “Sensor faults: detection methods and prevalence in real-world datasets,” *ACM Transactions on Sensor Networks*, vol. 6, no. 3, article 23, 2010.

[18] Y. Zhang, N. A. S. Hamm, N. Meratnia, A. Stein, M. van de Voort, and P. J. M. Havinga, “Statistics-based outlier detection for wireless sensor networks,” *International Journal of Geographical Information Science*, vol. 26, no. 8, pp. 1373–1392, 2012.

[19] P. J. Brockwell and R. A. Davis, *Introduction to Time Series and Forecasting*, vol. 1, Taylor & Francis, London, UK, 2002.

[20] "Intel Berkeley Research Laboratory (IBRL) dataset," 2014, http://db.csail.mit.edu/labdata/labdata.html.