A trust-based formal model for fault detection in wireless sensor networks

Na Wang
Shanghai Polytechnic University

Jiacun Wang
Monmouth University

Xuemin Chen
Texas Southern University

sensors

Article

A Trust-Based Formal Model for Fault Detection in Wireless Sensor Networks

Na Wang [1,*,†], Jiacun Wang [2,†] and Xuemin Chen [3,*,†]

1 Department of Computer and Information Engineering, Shanghai Polytechnic University, Shanghai 201209, China
2 Department of Computer Science and Software Engineering, Monmouth University, West Long Branch, NJ 07764, USA; jwang@monmouth.edu
3 Department of Engineering, Texas Southern University, Houston, TX 77004, USA
* Correspondence: wangna@sspu.edu.cn (N.W.); xuemin.chen@tsu.edu (X.C.)
† These authors contributed equally to this work.

Received: 24 January 2019; Accepted: 19 April 2019; Published: 23 April 2019

Abstract: Wireless Sensor Networks (WSNs) are prone to failures and malicious attacks. Trust evaluation is becoming a new method for fault detection in WSNs. In our previous work, a comprehensive trust model based on multi-factors was introduced for fault detection. This model was validated by simulating. However, it needs to be redeployed when adjustment to network parameters is made. To address the redeployment issue, we propose a Trust-based Formal Model (TFM) that can describe the fault detection process and check faults without simulating and running a WSN. This model derives from Petri nets with the characteristics of time, weight, and threshold. Basic structures of TFM are presented with which compound structures for general purposes can be built. The transition firing and marking updating rules are both defined for further system analysis. An efficient TFM analysis algorithm is developed for structured detection models. When trust factor values, firing time, weights, and thresholds are loaded, precise assessment of the node can be obtained. Finally, we implement TFM with the Generic Modeling Environment (GME). With an example, we illustrate that TFM can efficiently describe the fault detection process and specify faults in advance for WSNs.

Keywords: formal model; fault detection; multi-factors; Petri nets; wireless sensor networks

1. Introduction

Wireless Sensor Networks (WSNs) consist of distributed sensors that can monitor the environment, communicate with each other, and transmit information. It can continuously and automatically monitor a given field or event without any human presence. The working process of the sensors is to take measurements of the surrounding environment and transmit data to a base station for further data processing. Currently, the applications of WSNs are popular in wide areas such as intelligent industry, health care monitoring, environment monitoring, home automation, smart transportation, natural disaster relief, etc.

Due to the inherent characteristics and natural environments, WSNs are prone to various types of attacks such as black hole attack, eavesdropping, etc. [1]. The emergence of new data handling technologies and analytic enabled the organization of big data in processes as an innovative aspect in WSNs. The big data paradigm, combined with the WSN technology, involves new challenges that are necessary to resolve in parallel [2]. Therefore, it is crucial to detect faults, which will enhance the overall performance by monitoring network activities, minimizing risk, and ensuring the network activities of the entity such as data gathering and data processing. In [3], fault diagnosis in WSNs through various fault detection algorithms was given. In [4–10], the authors proposed various fault detection techniques. These techniques focused on how to detect and deal with faults by deploying a real or virtual WSN. In order to get precise detection, the authors above proposed effective methods from different perspectives such as energy consumption and sensor circuits. However, in wireless sensor networks, the trust model has played an important role in identifying misbehaving nodes and providing collaboration among trustworthy nodes [11]. The authors of [12] proposed a protocol layer trust-based intrusion detection scheme for wireless sensor networks, and the work in [13] proposed an evaluation model and data fusion mechanism based on trust. They focused on different aspects to measure trust values. In [14,15], the authors showed that trust models can provide a metric for routing, aggregation, and faulty detection. In our former work [16], we introduced a comprehensive trust model, which is able to assess the trust values of nodes by private trust and interactive trust. We classified trust into two parts as interactive trust and private trust. Interactive trust describes the trust of a node's interaction with its neighbor nodes based on interactive attributes. Private trust focuses on describing a node's reputation based on private attributes. Both trust values are used for detecting abnormal events and faulty nodes. All of the methods above were proposed for fault detection focusing on trust assessment. However, they were all implemented by simulations, and each detection process was run on a real deployment. Therefore, adjustment may be needed when the detection result cannot meet the requirement. In order to avoid frequently employing WSNs, a formal model can help check the detection method instead of simulation.

Petri nets are a powerful formal approach in computer science and system engineering [17]. Petri nets combine a well-defined mathematical model with an intuitive graphical model. The theoretical aspect of Petri nets allows precise modeling and analysis of system behavior, while the graphical aspect enables visualization of the state changes of the modeled system. Petri nets are a superior choice for specifying the concurrency and competitiveness of systems. These advantages allow Petri nets to find application in various kinds of event-driven systems such as embedded systems, communication systems, manufacturing plants, networks, real-time computing systems, and so on. Timed Petri nets, in which event times are specified, are able to catch the time-related performance or real-time properties of a system [18]. With timed Petri nets, we can precisely specify the fault detection mechanism of complex wireless sensor networks.

In this paper, inspired by the modeling power of Petri nets, we propose a trust-based formal model (TFM) for fault detection in WSNs. Based on TFM, the process of fault detection is built. To demonstrate the effectiveness of TFM-based trust evaluation, we apply the analysis algorithm to our previous multi-factor trust model [16].

The rest of the paper is organized as follows. Related works are introduced in Section 2. The definitions of TFM is provided in Section 3. The analysis of TFM for the detection process is depicted in Section 4. The implementation of TFM is given in Section 5. The concluding remarks are drawn and future work is discussed in Section 6.

2. Related Works

There are several methods on different levels that can assess trust performances. They are paper proof, simulation, and formalization. The paper proof method, however, is prone to human error and is not scalable to deal with large systems [19]. Due to the inherent incompleteness of simulation coupled with the rounding errors of computers, results cannot be considered as 100% accurate, which is a serious limitation for WSNs [20]. Using rigorous mathematical techniques, formal methods can overcome the limitations of simulation and have been used to validate a wide range of hardware and software systems [21]. Formal methods have been explored for analyzing WSNs, but most of the existing work is focused on analyzing their functional aspects only. However, with the wide application of WSNs in safety and mission-critical domains, there is an emergent need to assess their performances accurately as well. Formal methods sometimes provide languages such as Z language with strict semantics and syntax, corresponding techniques for the construction of models of systems under development, and verification of these models against selected requirements [22]. As a consequence, quantitative and qualitative properties, such as trust value or event detection rate, can be evaluated.

Recently, a few formal methods have been used for event detection in sensor networks. Some methods utilized the traditional model to check and validate special aspects of WSNs. In [23], the authors performed the formal analysis of the optimal geographical density control algorithm in real-time Maude to verify the network coverage intensity and lifetime. Real-time Maude is a language and tool for specification and analysis in real-time and hybrid systems [24]. It is based on re-writing logic that can provide analysis ability, but cannot provide performance evaluation. The authors of [25] suggested the use of P-Maude to enhance probabilistic analysis. Probabilistic model checking methods and tools [26] such as Prism have also been used for probabilistic analysis of wireless systems. Nevertheless, the accuracy of probabilistic model checking is very limited for validating statistical properties. In [27], an algebraic approach to the fault detection for parabolic distributed parameter systems was described. The modulation functions approach was used to obtain an algebraic fault detection equation, which only depends on known signals and the fault.

In [28,29], the authors showed that most of the popular formal approaches are based on theoretical models such as finite state machine, timed automata and process algebra. Finite state machine approaches have been pointed out for the difficulty in dealing with hierarchical models. Timed automata are an extension of finite state machines by incorporating real-valued clocks. Many specification methods are based on timed automata, and one of the well-known ones is UPPAAL. However, timed automata are deterministic finite state machines, so they inherit the limitations of finite state machines such as state explosions in large and complex distributed systems. Several approaches based on process algebra and composition logic were presented [30]. However, these approaches were mainly developed for database systems, so the sensing activities and spatial and temporal properties of WSNs were not addressed.

Timed Petri nets are a powerful extension to regular Petri nets [18]. Petri nets have advantages to describe events in network applications. In [31], the authors proposed a Petri net-based approach for resource requirements analysis and introduced Resource-Oriented Workflow Nets (ROWN). Petri nets have graphical support for users to operate easily. If the models can be improved for complex attributes, Petri nets can be a powerful tool to evaluate the performances in WSNs. In [32], the authors proposed a fault detection method modeled with Partially-Observed Timed Hybrid Petri nets (POTHPNs). Discrete faults that affect continuous processes have been considered. The marking of some continuous places and the firing of some discrete transitions were assumed to be measured on-line. Abrupt faults were considered as unexpected firings of some discrete silent transitions. This method is suitable for the class of hybrid systems that concerns continuous processing driven by discrete controllers. In [33], the authors used stochastic Petri nets to build a hierarchical model for the trust evaluation. The model focused on the location and energy of nodes. In [34], the goal of the trust model was to provide the authenticity of public keys. The trust model presented in this paper is based on the modeling technique of colored Petri nets. Colored Petri nets are a compact description of regular Petri nets. They do not increase the modeling and analytic capability of Petri nets. Therefore, colored Petri nets are mainly used for reachability, deadlock, and invariants' analysis.

In [35], based on Petri nets, the authors described a compact event description and analysis language for wireless sensor networks, namely MEDAL, for simultaneous monitoring of multiple events in a single network. MEDAL is a modified Petri net, which provides a more compact formal language for event description. It can capture the structural, spatial, and temporal properties of a complex event detection system, so as to assist system designers in identifying inconsistencies and potential problems. MEDAL is an improvement of formalization in WSNs, but it can only describe private attributes of events. In [36], we proposed a formal model for temporal-spatial event detection in Internet of Vehicles (IoV) based on Petri net. In IoV, the events are detected based on attributes such as location, speed, and arriving time. The model proposed in [36] focused on describing the relationship of location and time to get an event detection result.

3. A Trust-Based Formal Model

In WSNs, data collected by each sensor device are passed to the detection and processing module. This process is enforced during network operation by the processing program in a monitoring system. If the detection model can be validated in advance before being embedded into the system, the trial running cost of the system will be significantly reduced, and the accuracy of the detection will be improved. As shown in Figure 1, in a detection system, the user can describe the concerned event in formalized sentences and send it to the monitoring system. After receiving the formal statement, the monitoring system conveys it into the fault detection model. Then, the processing program generates the result based on the detection model, which will be fed back to the user. The detection process is described as in Figure 2.

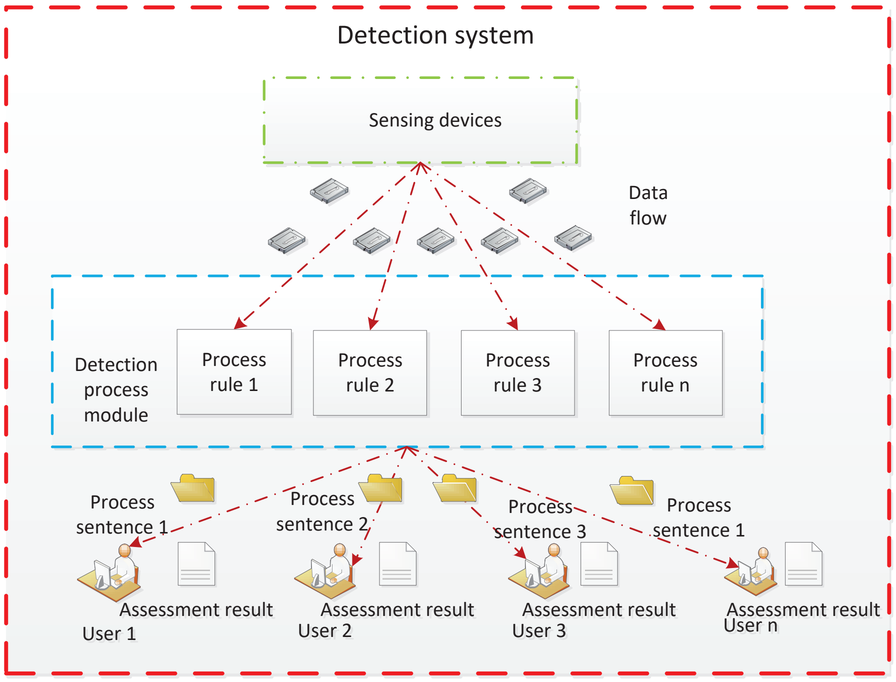

Figure 1. Detection system structure.

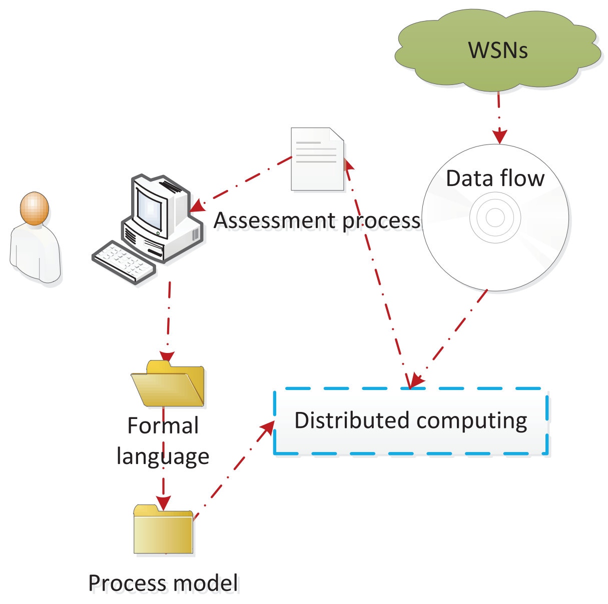

Figure 2. Detection process.

Since most events in WSN applications are concurrent, asynchronous, distributed, and non-deterministic in nature, we use Petri nets as a base model to specify WSN operation. The basic structure of Petri nets consists of places (P), transitions (T), arcs, and tokens. In the graphic representation of Petri nets, circles represent states or conditions, dots are used to model instances or objects, rectangles model various kinds of actions, and arcs represent changes between states. When a token represents an object with a variety of attributes, the token has a value (color) that represents the specific characteristics of the object, such as a token representing a student (name, age, gender). For the sake of analysis, when time or latency needs to be modeled, each transition may have a time stamp that specifies the duration of its firing. Weights associated with Petri nets can also be set as the attribute of an arc. For specific applications, we can extend Petri nets into most functional ones. According to the requirements in WSNs, it is necessary to have a formal description of abstract items as follows:

- The time of data sensed
- The weight of each factor
- The threshold for decision

3.1. Definition of the TFM

Formally, a Petri net is defined as $PN = (P, T, I, O, M_0)$ where
$P = \{p_1, p_2, \ldots p_m\}$ is a finite set of places;
$T = \{t_1, t_2, \ldots t_n\}$ is a finite set of transitions, $P \cup T \neq \emptyset, andP \cap T \neq \emptyset$;
$I = T \times P \rightarrow N$ is an input function that defines directed arcs from places to transitions;
$O = T \times P \rightarrow N$ is an output function that defines directed arcs from transitions to places;
$M_0 = P \rightarrow N$ is the initial marking.

A marking in a Petri net is an assignment of tokens to the places of a Petri net. Tokens reside in the places of a Petri net. The number and location of tokens may change during the execution of a Petri net. The tokens are used to define the execution of a Petri net. A place containing one or more tokens is said to be marked [31].

The TFM can be described as an eight-tuple structure $(P, T, I, O,\ M_0, \mu, \delta, \theta)$ based on Petri nets, where $P, T, I, O,$ and M_0 are classic definitions of Petri nets. In order to describe trust-based detection, we extend the basic Petri net with three items.

- μ is the weight on arcs, which represents the probability or importance of factors of a transition. $\mu : A \rightarrow [0, 1]$ and $\sum_{i=1}^{n} \mu_i = 1$ where n is the number of input arcs into a transition. For example, if there is an arc (p, t), $\mu(p, t) = w$ means there is a probability of $\mu(p, t)$ inducing the token entering t from p. If the token has a capacity c, the new capacity will be $c * w$.
- δ is a time guard for T , δ: $T \rightarrow [t_1, t_2]$, and $t_1 \leq t_2$. $\delta(T) = (a, b)$ means transition T can only fire during t_1 and t_2. Especially, if $t_1 = t_2$, that means the transition can only happen during t_1.
- θ is the threshold of token capacity in P, θ: $P \rightarrow R$, and R is a real type data. $\theta(P) = r_2$, means when the capacity of the token in P is greater than or equal to r_2, P can reach a new station.

Tokens are abstract representations of sensed data. When a transition fires, the values of a token will be updated according to the rules. In the TFM, the main data are the trust of different factors.

3.2. Trust Modeling

There are several types of factors that can have an impact on trust in WSNs, and the value of each type of factor can be represented by a non-negative real number [16]. An evaluation process will consume factors for aggregating a new trust value. We use F^{in} to describe the input factors consumed and use F^{out} to describe the aggregation trust value. For an evaluation process TP_k, $F^{in}(TP_k)$ is associated with the input place and $F^{out}(TP_k)$ is associated with the output place. For example, considering Figure 3, assume P_4 is a state of aggregation output, and P_1, P_2, P_3 are states of input.

When there is a process TP_4, which has three inputs, $F^{in}(TP_4)$ will be the factors' values and $F^{out}(TP_4)$ will be the aggregation value of inputs according to special operations, which will be introduced later.

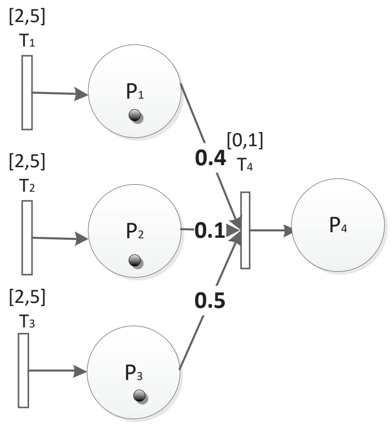

Figure 3. A simple Trust-based Formal Model (TFM).

3.3. Rules in TFM

Rules for firing transitions are described as below:

A transition T_k under the marking M_i can be fired if and only if:

$$t_k \in \delta(T_K) \tag{1}$$

$$F^{in}(TP_k) > 0 \tag{2}$$

$$F^{out}(TP_k) \geq \theta(P_i) \tag{3}$$

$$M_i \geq I(t_k) \tag{4}$$

where $\delta(T_k)$ is the valid sensing time in WSNs and $F^{in}(TP_k)$ is the current value of the token in the input place. Note that for a process TP_k, there may be more than one input. $F^{out}(TP_k)$ is the current value of the token in the output place, and θ is the threshold for entering P_i.

Condition (1) stands for time limit satisfaction; conditions (2) and (3) stand for valid value being available; and condition (4) stands for control readiness. These conditions must be met by trust factors simultaneously.

Rules for markings are described as below:

A token value may be changed when a transition fires, and it will be held in a new place due to the threshold.

$$F^{out}(TP_k) = \sum_{j=0}^{n} \mu_j * F_j^{in}(TP_k) \tag{5}$$

where n is the number of factors.

Then, the new marking will be updated as below:

$$M_{j+1} = M_j - I(T_k) + O(T_k) \tag{6}$$

3.4. An Example

In order to explain the TFM intuitively, we give an example here. Still considering Figure 3, it represents a system as follows:

$$\delta(T_1) = \delta(T_2) = \delta(T_3) = [2,5], \delta(T_4) = [0,1];$$

$$F_3^{in}(TP_4) = (0.9, 0.09, 0.8);$$

$$M_0 = (1, 1, 1, 0);$$

$$\mu(P_1, T_4) = 0.4, \mu(P_2, T_4) = 0.1, \mu(P_3, T_4) = 0.5;$$

$$\theta(P_4) = 0.6, \theta(P_1) = \theta(P_2) = \theta(P_3) = 0(default);$$

T_1, T_2, T_3 represent the three transitions that can fire after two and must fire before five time units. After the three transitions fire, new tokens are built. Since P_1, P_2, and P_3 receive tokens unconditionally, their thresholds are assigned as zero. Then, in one time unit duration, T_4 fires and $F^{out}(TP_4) = \sum_{i=1}^{3} \mu_i * F_i^{out}(TP_4) = 0.774$. Since 0.774 is greater than the threshold of P_4, P_4 can be reached.

3.5. Structures of the TFM

In order to evaluate a trust value based on multi-factors so as to assess the state of a node, we use two sequential places to model its evaluation process. One place stands for the sensing data, and the other stands for the assessment result. The evaluation action is modeled with a transition in between. The single logic unit is shown as in Figure 4.

The sequential structure is shown in Figure 5. P_1 is the initial place; after T_1 is fired, P_2 is marked. If T_2 is fired, P_3 will be marked. In this case, P_2 is the shared place for P_1 and P_3. We describe the sequence as $T_1P_2 \rightarrow T_2P_3$.

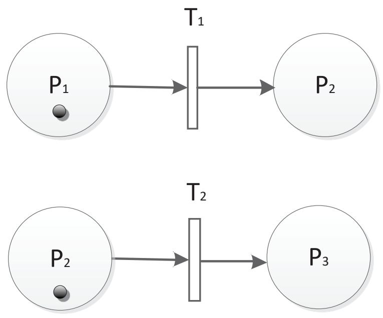

Figure 4. Two single logic units.

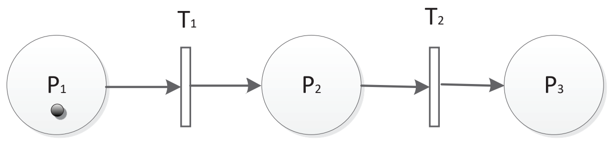

Figure 5. Sequential structure.

The second is the parallel structure, which is shown in Figure 6. There are two parallel units T_1P_2 and T_3P_4 that will fire T_3. T_2 can be fired if and only if there are tokens in both P_2 and P_4. Then, P_5 will be marked if the time and threshold conditions are also met. The parallel structure can be described as $T_1P_2||T_3P_4$. The parallel structure can also be chained as shown in Figure 7.

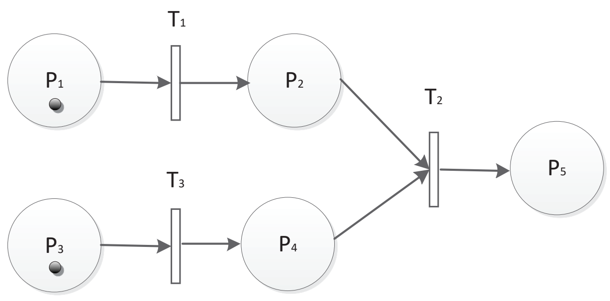

Figure 6. Parallel structure.

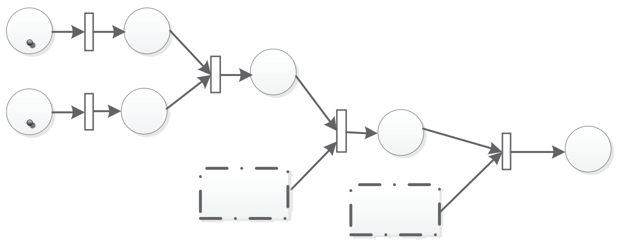
Figure 7. Chained parallel structure.

The third is the choice of the structure that is shown in Figure 8. There are two choice units T_2P_2 and T_3P_3. Once there are tokens in P_1, T_2 or T_3 will be fired. Then, P_2 or P_3 will be marked if time and threshold conditions are also met. The choice structure can be described as $T_2P_2 \oplus T_3P_3$. The choice structure can also be chained as shown in Figure 9.

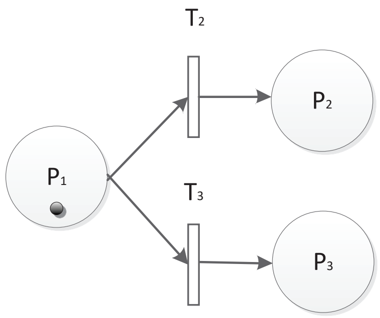

Figure 8. Choice structure.

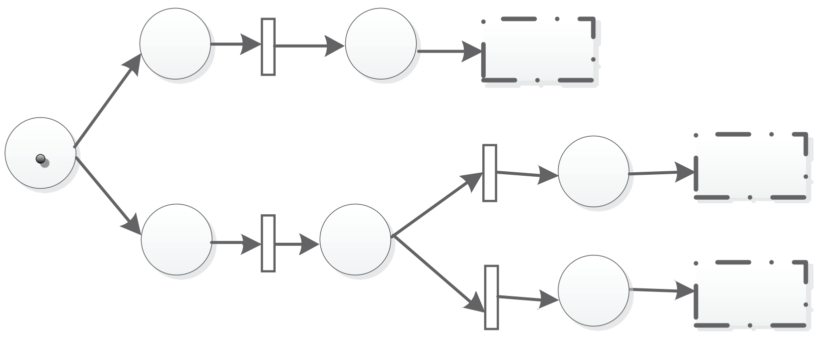
Figure 9. Chained choice structure.

4. Analysis of the TFM

In our previous work [16], we defined a hierarchical network with a number of sensors. These sensors were placed in an area and transmit information within a certain radius. Each node maintained its identified number, sensing data, and location. There were three kinds of nodes including sink node, cluster heads, and member nodes. The member nodes sensed data and communicated with their heads directly. The cluster heads aggregated data sent by their member nodes and forwarded them to the sink node through other cluster heads by hops. The sink node is a central control node that can schedule the whole network. The running process of the network can be depicted as follows:

Event-driving stage: When there is a request for detecting in a certain field from the sink node, the sink node will send a sensing order to its neighbor nodes.

Self-organizing stage: The nodes receiving the request will act as the first-level cluster heads. The cluster heads will select their member nodes according to the cluster protocol.

Detecting stage: The member node senses data after receiving the request from its cluster head. The sensing action is frequent according to the sampling period.

Communication stage: The member nodes send data to the cluster head and its neighbors. In this stage, interaction and data aggregation are crucial to the trust value.

Data aggregation: After data transmitting, the cluster head has the information of its member nodes. In order to reduce information and energy consumption, data from the member nodes will be aggregated by the cluster head and be sent to the higher level cluster head.

Convergence stage: Data from different levels of cluster heads will be aggregated and sent hierarchically till the sink.

According to the behaviors of the nodes, we divided them into two types as normal nodes and outlier nodes. Normal nodes provided data that changed gradually and regularly. An outlier appeared to deviate markedly from other members in the same group [37]. Outlier nodes that did not perform normally were mainly caused by malicious activity, instrumentation error, human error, and a change in the environment [38]. In the network we have defined, the outlier nodes included faulty nodes providing fault data and event nodes providing event data that can reflect the change in the environment. In order to recognize normal nodes, faulty nodes and event nodes, we defined two types of trust: private trust and interactive trust. Private trust was mainly evaluated based on three factors:

(1) A node's private trust in the last cycle.
(2) The number of times it deviated from the aggregation value in the current cycle.
(3) The number of times it sensed the same data consecutively.

It is clear that only the second factor is related to the aggregation data in a cluster. Therefore, even in such a cluster that a group of nodes has sensed firing and the others have not, the node that keeps its private trust above the normal threshold will be regarded as a normal one. If it does not, then it will be regarded as an outlier node. However, an outlier node may be either a faulty node or a node detecting an event such as firing. In order to decide whether the outlier node is a faulty node, we used interactive trust [16]. Interactive trust was mainly evaluated based on the data similarity and communication between a node and its neighbors. When the interactive trust of an outlier node is greater than event threshold, the outlier node should be recognized as an event node. Otherwise, the outlier node will be regarded as a faulty node [39].

For example, a set of nodes were deployed to monitor fire in a forest as shown in Figure 10. There were two faulty nodes F_1 and F_2. If there was no fire, normal sensors except F_1 and F_2 would keep a private trust greater than the normal threshold. Once there is a fire, the sensors in the fire area such as N_4 and N_5 would sense fire data. Meanwhile, the sensors on the fire border such as E_1 and E_2 would also sense fire data. However, the sensors far away from the fire area such as N_1, N_2, and N_3 would still sense normal data. If E_1 is in the same cluster with N_1, N_2, and N_3, but not in the same cluster with N_4 and N_5, its sensing data will deviate far from the aggregation value. If its private trust has not been remarkably reduced by the deviation, it will still be recognized as a normal one though its current sensing data are different from others'. If the deviation reduced its private value to under the normal threshold, it will be recognized as an outlier. Next, its interactive trust will be used to decide whether it is a faulty node or an event one. In this case, different from F_1 and F_2, E_1 and E_2 may be normal nodes or outlier nodes that just provide event data, but not faulty data.

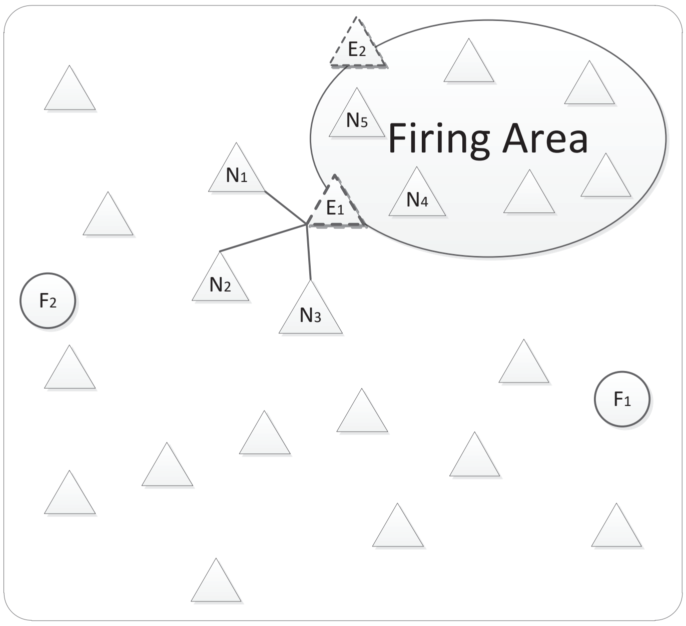

Figure 10. A distribution with four outliers F_1, F_2, E_1, and E_2.

In [16], as is shown in Tables 1 and 2, interactive factors were abstracted as ITC (Interactive trust based on valid communication), ITD (Interactive trust based on data similarity) and ITT (Interactive trust based on clock synchronization), which are crucial when computing the trust value between nodes. Private factors depend on the node's PTD (Private trust based on previous data), PTE (Private trust based on remaining energy), PTR (Private trust based on the misreading) and PTF (Private trust based on consecutive same sensing). The nodes must be penalized when sensing data deviate far from the aggregation value; they also should be awarded if sensing data are in normal distributed range consecutively. In order to depict the relations and importance of factors, we used a reciprocal matrix with a right characteristic root to calculate weight vectors. With interactive and private evaluations, we can detect fault nodes and event nodes, as shown in Figure 11. If the private trust value of a node is greater than the normal threshold, it will be regarded as being normal. Otherwise, if its interactive trust value is greater than the event threshold, it should be recognized as a event node. Otherwise, it will be regarded as a faulty node.

Table 1. Interactive factors.

ITC	ITD	ITT
valid communication	data similarity	clock synchronization

Table 2. Private factors.

PTD	PTE	PTR	PTF
History data	Remaining energy	Penalty of misreading	Consecutive same sensing

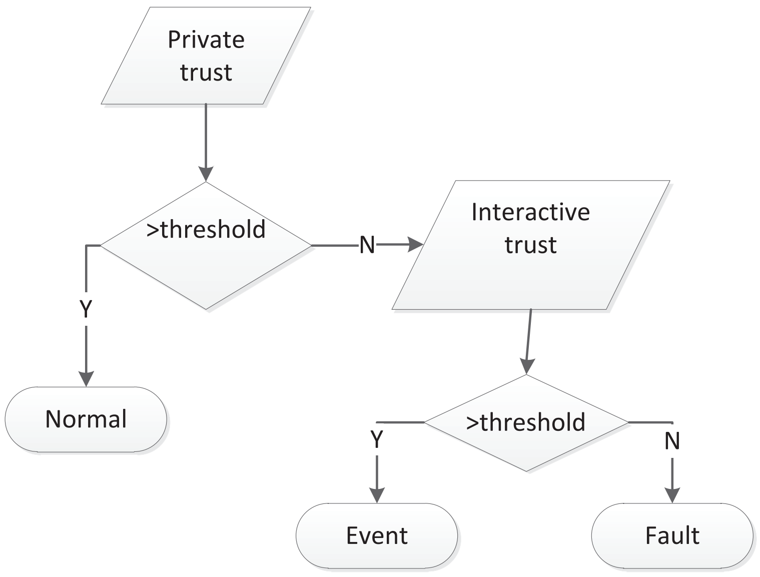

Figure 11. Fault detection based on trust [16].

Using the TFM, we focus on assessing the status of nodes according to the time constraints and thresholds.

4.1. Analysis

With regard to time constraints, δ is used to control the transition duration to guarantee the validity of values in WSNs. If there is no requirement on the time limit, the default of δ is $(-\infty, +\infty)$. For sequential and choice structures, the firing time of each transition must be in δ. For parallel structures, the firing role about time must be described specially.

Assume there are two transitions T_1 and T_2. The firing duration of T_1 is $\delta(T_1) = [t_1, t_2]$, and for T_2, $\delta(T_2) = [t_3, t_4]$. T_1 is enabled at time τ_1 where $t_1 \leq \tau_1 \leq t_2$. If it fires at time $\tau_1 + \phi_1$, according to the time rules in a timed Petri net, $\delta(T_2) = [max\{0, t_3 - \phi_1\}, t_4 - \phi_1]$.

If there are more parallel transitions in a system, the firing time of a transition will shift for times according to the number of transitions firing before it, which is shown in (7). If we use D to denote the set of time constraints δ, then where $D = \{\delta(T_j), j = 0 \ldots n\}$, n is the number of places.

$$\delta(T_j) = [max\{0, t_{j,1} - \sum_{i=2}^{j} \varphi_{i-1}\}, t_{j,2} - \sum_{i=2}^{j} \varphi_{i-1}] \tag{7}$$

With regard to markings, consider a detecting procedure including $TP_1, TP_2, \ldots TP_j$. According to different structures, it can be described as:

If they are sequential:

$$F^{out}(TP_j) = F^{out}(TP_{j-1}) * \mu_{j-1} \tag{8}$$

If they are parallel:

$$F^{out}(TP_j) = \sum_{k=1}^{j-1} \mu_k * F^{out}(TP_k) \tag{9}$$

4.2. The TFM for the Trust Model Based on Multi-Factors

According to the logic description implied in Figure 11, when nodes are not regarded as normal, they may be outliers or abnormal nodes. If the interactive trust is greater than its threshold while private trust is less than its threshold, it may be the case that the node is located on the edge of the event area and detects an event. Otherwise, it can be treated as a fault node. To illustrate how to apply the TFM for fault detection, we built an example based on multi-factors, as shown in Figure 12. The numbers used in Figure 12 are defined in Tables 3 and 4.

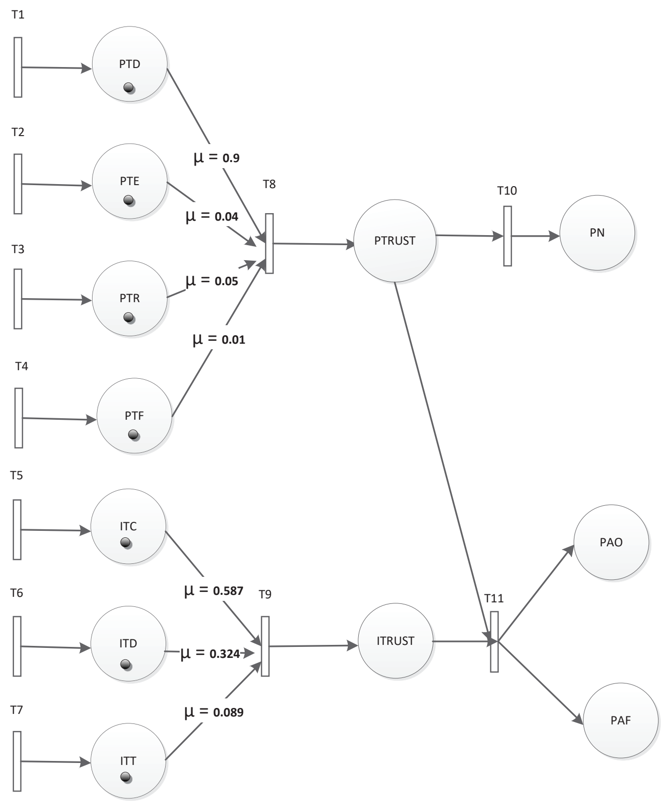

Figure 12. TFM for fault detection.

The trust model is described as follows:

P is the set for all places including factors and events. Factors represent the evaluation factors in the trust model; they are {PTD, PTE, PTR, PTF, ITC, ITD, ITT, PTRUST, ITRUST}. The first seven factors are explained in Tables 1 and 2. PTRUST is a comprehensive trust deduced from PTD, PTE, PTR, and PTF which means private trust. ITRUST is a comprehensive trust deduced from ITC, ITD, and ITT, which means interactive trust. Events represent detection result; they are {PN, PAO, PAF}. PN represents a normal status of a node. PAO represents an abnormal status of a node, which is called an outlier. PAF represents an abnormal status of a node, which is called fault. We used interactive and private factors calculated by the method introduced in [16]. Current tokens are indicated in Table 3. In Table 3, there are three groups of factors, which are $Value^1$, $Value^2$, and $Value^3$. Different groups will get different assessments.

μ is the weight of each arc. If there are no branches, μ is one by default. Otherwise, μ is set according to the importance of a factor. In the trust model of [16], we used the reciprocal matrix with a right characteristic root to calculate weight vectors, as is shown in Table 4.

Since the data sensing and evaluation occurred periodically, we set $\delta(T_{i=1...4})$ for private factors' relative transition, which can sense real-time factor values. Meanwhile, we set $\delta(T_{i=5...7})$ for interactive factors' relative transition since interactive valuation will be executed in the case that private evaluation is less than the threshold. For other transitions, we left δ as $(-\infty, +\infty)$, which means there were no time limitations for them. For this case, they are indicated in Table 5.

Table 3. Initial marking in different cases.

Place	*Value*[1]	*Value*[2]	*Value*[3]
PTD	0.89	0.29	0.29
PTE	0.93	0.93	0.93
PTR	0.7	0.7	0.7
PTF	0.88	0.88	0.88
ITC	0.94	0.94	0.64
ITD	0.82	0.82	0.82
ITT	0.77	0.77	0.77

Table 4. Weights of factors.

ITC	ITT	ITD	PTD	PTE	PTR	PTF
0.587	0.324	0.089	0.9	0.04	0.05	0.01

According to time constraint analysis, D will be:
$D_0 = \{1 \leq \delta(T_1) \leq 4, 2 \leq \delta(T_2) \leq 4, 2 \leq \delta(T_3) \leq 4, 3 \leq \delta(T_4) \leq 4\}$
$D_1 = \{0.5 \leq \delta(T_2) \leq 2.5, 0.5 \leq \delta(T_3) \leq 2.5, 1.5 \leq \delta(T_4) \leq 2.5\}$
$D_2 = \{0 \leq \delta(T_2) \leq 2, 1 \leq \delta(T_4) \leq 2\}$
$D_3 = \{0 \leq \delta(T_4) \leq 1\}$
$D_4 = \{1 \leq \delta(T_8) \leq 2\}$
$D_5 = \{2 \leq \delta(T_5) \leq 3, 1 \leq \delta(T_6) \leq 3, 2 \leq \delta(T_7) \leq 3\}$
$D_6 = \{1 \leq \delta(T_5) \leq 2, 1 \leq \delta(T_7) \leq 2\}$
$D_7 = \{0 \leq \delta(T_7) \leq 0.5\}$
$D_8 = \{2 \leq \delta(T_9) \leq 4\}$
$D_9 = \{2 \leq \delta(T_10) \leq 4\}$
$D_{10} = \emptyset$

Table 5. The time for transition.

Transition	δ	Enabled Time	Firing Time	Firing Order
T1	[1, 5]	1	1.5	1
T2	[2, 4]	2	1	3
T3	[2, 5]	1	0.5	2
T4	[3, 6]	2	1	4
T5	[2, 3]	2	1.5	2
T6	[1, 4]	2	1	1
T7	[2, 4]	1	0.5	3
T8	[1, 2]	1	1	1
T9	[2, 4]	2	2.5	1
T10	[2, 4]	2	3	1

θ is the threshold that controls whether the token can enter place P. In a trust model, when the private trust reaches 0.8, a normal status is reached. Here, we set both $\theta(PN)$ = 0.8 and $\theta(PAO)$ = 0.8 according to our former work [16].

Using Algorithm 1, i.e., TFM analysis algorithm, we can evaluate the detection process. Suppose in the initial marking, weights and time constraints are set as shown in Tables 3–5, the attributes updating in processes of each branch reaching PN, PAO, and PAF are shown in Table 6.

In WSNs, it is important to detect fault nodes. In our previous trust model [16], we modeled the trust and evaluated the fault detection by simulation. Different from simulation, the analysis of the TFM does not need the detailed real-time data, but focuses on modeling time constraints and other limitations combining with trust values to determine the status of a node. Therefore, it is convenient for users to develop a detection method and validate its correctness by adjusting inputs. Furthermore, the detection rate based on the TFM can be achieved by building interpreters in the future.

Algorithm 1 TFM analysis algorithm.

```
Input:
  Input:
  Multi-factors;
  Thresholds;
  Weights;
  Lower time;
  Upper time;
Output:
  Node status;
  j=0;
  while $(M_j)$
  {
  for (i=0; i<n; i++)
  {
  if $(D_i)$
  calculate $F^{out}(TP_j)$ using Equation (9);
  }
  calculate $F^{out}_{j+1}$ using Equation (8);
  if $(F^{in}_{j+1} \geq \theta_j)$
  j++;
  }
```

Table 6. Attributes in each branch.

	PN	PAO	PAF
D	$\{D_0, D_1, D_2, D_3, D_4, D_{10}\}$	$\{D_5, D_6, D_7, D_8, D_{10}\}$	$\{D_5, D_6, D_7, D_9, D_{10}\}$
M	$\left\{\begin{array}{l}(0.89, 0.93, 0.7, 0.88, 0.94,\\ 0.82, 0.77, 0, 0, 0, 0, 0),\\ (0, 0, 0, 0, 0.94, 0.82,\\ 0.77, 0.882, 0, 0, 0, 0),\\ (0, 0, 0, 0, 0.94, 0.82,\\ 0.77, 0, 0.822, 0, 0, 0)\end{array}\right\}$	$\left\{\begin{array}{l}(0.29, 0.93, 0.7, 0.88, 0.94,\\ 0.82, 0.77, 0, 0, 0, 0, 0),\\ (0, 0, 0, 0, 0.94, 0.82,\\ 0.77, 0.35, 0, 0, 0, 0),\\ (0, 0, 0, 0, 0, 0, 0, 0.35,\\ 0.886, 0, 0, 0),\\ (0, 0, 0, 0, 0, 0, 0, 0.35,\\ 0, 0, 0.886, 0)\end{array}\right\}$	$\left\{\begin{array}{l}(0.29, 0.93, 0.7, 0.88,\\ 0.64, 0.82, 0.77, 0, 0, 0, 0, 0),\\ (0, 0, 0, 0, 0.64, 0.82,\\ 0.77, 0.35, 0, 0, 0, 0),\\ (0, 0, 0, 0, 0, 0, 0, 0.35, 0,\\ 0.71, 0, 0),\\ (0, 0, 0, 0, 0, 0, 0, 0.35, 0,\\ 0, 0, 0, 0.71)\end{array}\right\}$
F^{out}	$\left\{\begin{array}{l}0.89 * 0.9 + 0.93 * 0.04+\\ 0.7 * 0.05 + 0.88 * 0.01 = 0.882,\\ 0.882 * 1 = 0.882\end{array}\right\}$	$\left\{\begin{array}{l}0.94 * 0.587 + 0.82*\\ 0.324 + 0.77 * 0.089 = 0.886,\\ 0.886 * 1 = 0.886,\\ 0.886 * 1 = 0.886\end{array}\right\}$	$\left\{\begin{array}{l}0.64 * 0.587 + 0.82*\\ 0.324 + 0.77 * 0.089 = 0.71,\\ 0.71 * 1 = 0.71,\\ 0.71 * 1 = 0.71\end{array}\right\}$
θ	$0.882 > 0.8$	$0.886 > 0.8$	$0.71 < 0.8$

5. Implementation of the TFM

In order to provide a convenient environment for users, we used a Generic Modeling Environment (GME) as a platform to build the framework for the TFM applications, as shown in Figure 13. After the framework was built, registration and interpretation were executed. Then, different TFM applications can be constructed. The implementation part of the full environment is visualized for modeling and analysis in Figure 14.

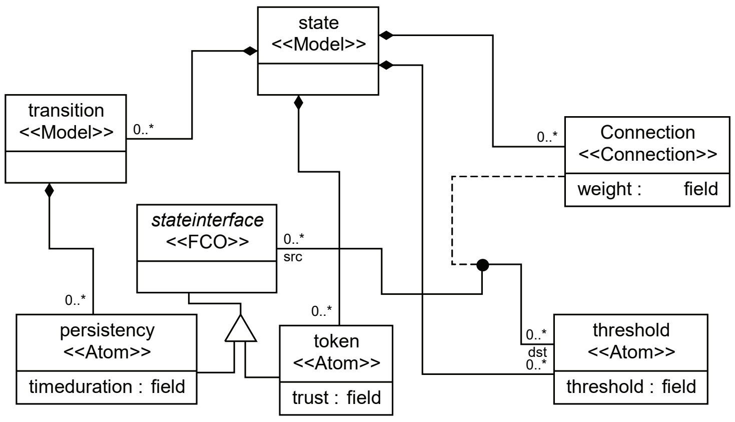

Figure 13. Model process for the TFM.

For each place, there are properties such as token and threshold. For each transition, there are properties such as low time and high time. For each arc, there is a property named weight. We can set values for properties according to the TFM application requirements. The values can be modified easily on the property browser window. With model checking support, we can perform model checking for the TFM applications to validate detection methods, etc.

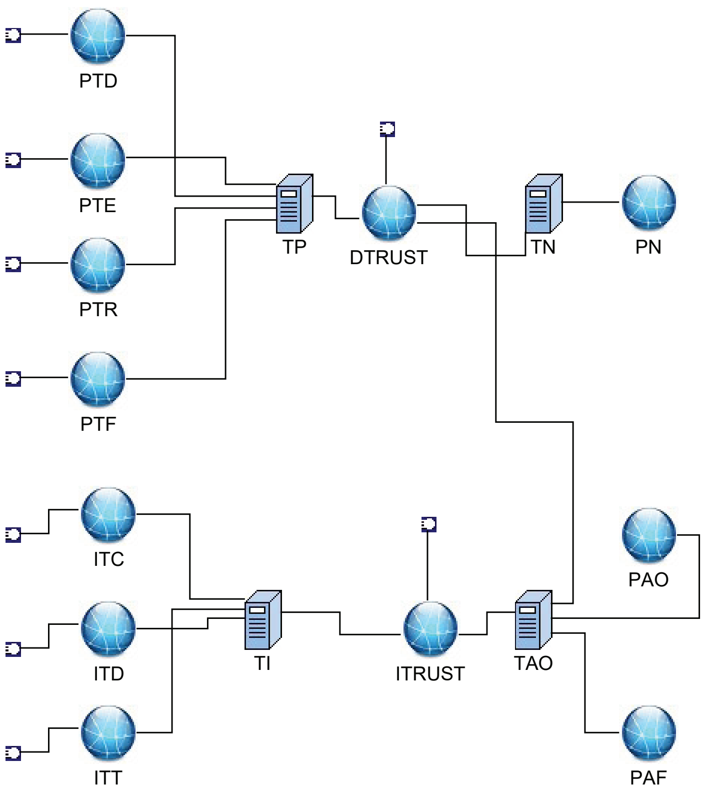

Figure 14. Model for trust in the TFM.

In our former work, we compared the trust model with a Trust Management Scheme (TMS) [39]. The result showed that the detection rate of our model was higher than TMS during the running time because we used private trust to confirm a fault node rapidly. However, the fluctuation of the detection rate was larger than TMS due to the temporary malicious judgment of event nodes.

In the simulation, we tested forty sets of data that were the same as the former experiment. Each node had two to five neighbors in the experiment, and the node's location was already known. The weights of factors were set as shown in Table 4 according to the reciprocal matrix in [16]. The thresholds for fault and event nodes were both set as 0.8 according to the statistics shown in Figure 15. In [16], we proved that the trust values were real numbers between zero and one. The trust values of normal nodes vibrated near 0.85. If the thresholds are set too low, some fault nodes cannot be detected. If the thresholds were set too high, some normal nodes may have been recognized as fault nodes. We assumed there was an event during the running time. Our former result is shown in Figure 16. It shows the numbers of different types of nodes including normal and fault ones. Within the running time, the normal ones will be detected more precisely, and the fault ones will be detected from outliers. Executing the TFM analysis algorithm, we evaluated both private trust values and the interactive trust value to detect whether the outlier was a fault or event node. The result in Figure 17 shows that the detecting of the normal nodes was stable, and the detecting of the event nodes from outliers was rapid.

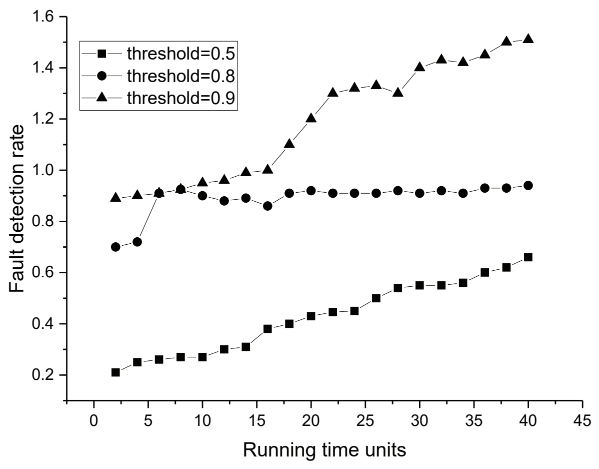

Figure 15. Fault detection rate with different thresholds.

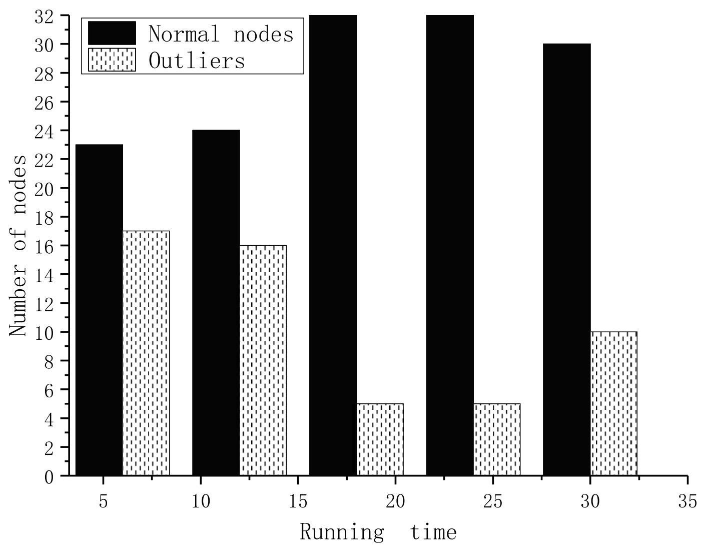

Figure 16. Nodes detected in our former work.

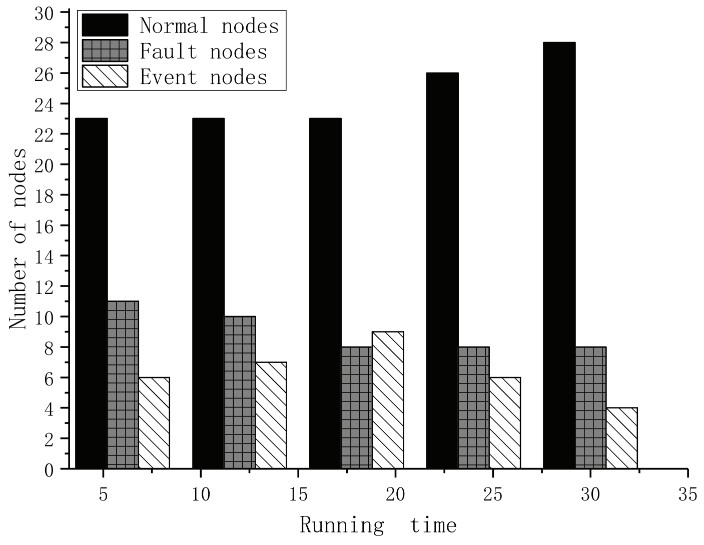

Figure 17. Nodes detected in the current work.

Referring to Figure 12, there were three resulting states, PN, PAO and PAF. Transition TN related to PN had priority over TI related to PAO and PAF. When the token in state DTRUST was greater than 0.8, the node would be assessed as PN, which means normal. Otherwise, state ITRUST would be selected for assessment. When token in state ITRUST was greater than 0.8, the node would be assessed as PAO, which means event node. Otherwise, the node would be assessed as PAF, which means fault node. Once the event nodes had been recognized, the malicious judgment in our former model would decline. Then, the fault nodes could be excluded earlier, and the detection rate of fault nodes would become stable. The detection result comparison between this formal model and our former model is shown in Figure 18.

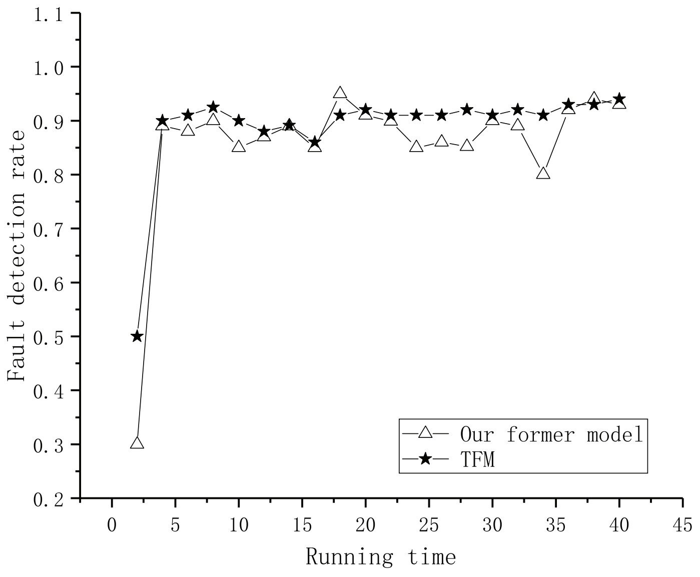

Figure 18. Comparison of the detection rate between two models.

6. Conclusions and Future Work

In this paper, we used Petri nets to build a trust-based formal model that can describe the fault detection process. Compound model structures can be built from the basic TFM structures such as sequential, parallel, and choice structures. According to the TFM analysis algorithm, the fault detection process can be described by using the basic TFM structures, and the status of nodes can be assessed. With the implementation of the TFM, users can deploy a TFM application more visually and conveniently. The example and the result demonstrated that introducing time, weight, and threshold into Petri nets is suitable for the TFM. Once the places, thresholds, and transitions are set, the time constraints and the weights calculated from reciprocal matrix can make a precise evaluation of a node's status so as to detect fault in advance.

In WSN applications, the TFM can be used to describe a system in a structured way. Meanwhile, the TFM allows users to modify existing designs quickly and re-evaluate updated designs conveniently without considering the network size.

However, we also need a software tool to process data from the model automatically. Typical processing tasks include running queries, generating program code, and building models automatically from information. The information may be provided by another data source (e.g., a database). In the future, we will develop a TFM interpreter to meet more application requirements.

Author Contributions: N.W. conceived of the idea and developed the algorithm; N.W. and J.W. designed the model; N.W. and X.C. performed the data analysis; N.W. wrote the paper; J.W. and X.C. made critical revision to the paper.

Funding: The paper is supported in part by the National Natural Science Foundation of China under Grant No. 61672022, Key Disciplines of Computer Science and Technology of Shanghai Polytechnic University under Grant No. XXKZD1604, and the U.S. National Science Foundation under Grant 1137732.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

WSNs	Wireless Sensor Networks
TFM	Trust-based Formal Model
GME	Generic Modeling Environment
ROWN	Resource-Oriented Workflow Nets

References

1. Butun, I.; Morgera, S.D.; Sankar, R. A survey of intrusion detection systems in wireless sensor networks. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 266–282. [CrossRef]
2. Boubiche, S.; Boubiche, D.E.; Bilami, A.; Toral-Cruz, H. Big Data Challenges and Data Aggregation Strategies in Wireless Sensor Networks. *IEEE Access* **2018**, *6*, 20558–20571. [CrossRef]
3. Munir, A.; Gordon-Ross, A.; Ranka, S. Modeling and Analysis of Fault Detection and Fault Tolerance in Embedded Wireless Sensor Networks. In *Modeling and Optimization of Parallel and Distributed Embedded Systems*; John Wiley & Sons, Ltd: Hoboken, NJ, USA, 2016. [CrossRef]
4. Muhammed, T.; Shaikh, R.A. An analysis of fault detection strategies in wireless sensor networks. *J. Netw. Comput. Appl.* **2017**, *78*, 267–287. [CrossRef]
5. Zidi, S.; Moulahi, T.; Alaya, B. Fault detection in wireless sensor networks through SVM classifier. *IEEE Sens. J.* **2018**, *18*, 340–347. [CrossRef]
6. Chanak, P.; Banerjee, I. Fuzzy rule-based faulty node classification and management scheme for large scale wireless sensor networks. *Expert Syst. Appl.* **2016**, *45*, 307–321. [CrossRef]
7. Jia, S.; Ma, L.; Qin, D. Fault Detection Modelling and Analysis in a Wireless Sensor Network. *J. Sens.* **2018**, *2018*, 7935802. [CrossRef]
8. Sharma, K.P.; Sharma, T.P. rDFD: Reactive distributed fault detection in wireless sensor networks. *Wirel. Netw.* **2017**, *23*, 1145–1160. [CrossRef]
9. Cheng, Y.; Liu, Q.; Wang, J.; Wan, S.; Umer, T. Distributed Fault Detection for Wireless Sensor Networks Based on Support Vector Regression. *Wirel. Commun. Mob. Comput.* **2018**, *2018*, 4349795. [CrossRef]

10. Hidoussi, F.; Toral-Cruz, H.; Boubiche, D.E.; Lakhtaria, K.; Mihovska, A.; Voznak, M. Centralized IDS Based on Misuse Detection for Cluster-Based Wireless Sensors Networks. *Wirel. Pers. Commun.* **2015**, *85*, 207–224. [CrossRef]
11. Rani, V.S. Review of Trust Models in Wireless Sensor Networks. *Int. J. Comput. Inf. Eng.* **2014**, *8*, 371–377.
12. Wang, J.; Jiang, S.; Fapojuwo, A.O. A Protocol Layer Trust-Based Intrusion Detection Scheme for Wireless Sensor Networks. *Sensors* **2017**, *17*, 1227. [CrossRef]
13. Chen, Z.; Tian, L.; Lin, C. Trust Model of Wireless Sensor Networks and Its Application in Data Fusion. *Sensors* **2017**, *17*, 703. [CrossRef] [PubMed]
14. Gu, X.; Qiu, J.; Wang, J. Research on Trust Model of Sensor Nodes in WSNs. *Procedia Eng.* **2012**, *29*, 909–913.
15. Yim, S.J.; Choi, Y.H. An Adaptive Fault-Tolerant Event Detection Scheme for Wireless Sensor Networks. *Sensors* **2010**, *10*, 2332–2347. [CrossRef]
16. Wang, N.; Chen, Y. A Comprehensive Trust Model Based on Multi-factors for WSNs. *Int. J. Comput. Commun. Control* **2015**, *10*, 257–271. [CrossRef]
17. Wang, J. Petri Nets for Dynamic Event-Driven System Modeling. In *Handbook of Dynamic System Modeling*; Fishwick, P., Ed.; CRC Press: Boca Raton, FL, USA, 2007; Chapter 24.
18. Wang, J. *Timed Petri Nets: Theory and Application*; Kluwer Academic Publishers: Boston, MA, USA, 1998.
19. Qasim, M.; Hasan, O.; Elleuch, M.; Tahar, S. Formalization of Normal Random Variables in HOL. In *Intelligent Computer Mathematics*; Kohlhase, M., Johansson, M., Miller, B., de Moura, L., Tompa, F., Eds.; Springer International Publishing: Cham, Switzerland, 2016; pp. 44–59.
20. Elleuch, M.; Hasan, O.; Tahar, S.; Abid, M. Formal probabilistic analysis of detection properties in wireless sensor networks. *Formal Aspects Comput.* **2015**, *27*, 79–102. [CrossRef]
21. Riaz, S.; Afzaal, H.; Imran, M.; Zafar, N.A.; Aksoy, M.S. Formalizing Mobile Ad Hoc and Sensor Networks Using VDM-SL. *Procedia Comput. Sci.* **2015**, *63*, 148–153. [CrossRef]
22. Man, K.L.; Krilavičius, T.; Vallee, T.; Leung, H.L. TEPAWSN: A Formal Analysis Tool for Wireless Sensor Networks. *Int. J. Res. Rev. Comput. Sci.* **2009**, *1*, 24.
23. Ölveczky, P.C.; Thorvaldsen, S. Formal Modeling and Analysis of the OGDC Wireless Sensor Network Algorithm in Real-Time Maude. In *Formal Methods for Open Object-Based Distributed Systems*; Bonsangue, M.M., Johnsen, E.B., Eds.; Springer: Berlin/Heidelberg, Germany, 2007; pp. 122–140.
24. Ölveczky, P.C.; Meseguer, J. Specification and Analysis of Real-Time Systems Using Real-Time Maude. In *Fundamental Approaches to Software Engineering*; Wermelinger, M., Margaria-Steffen, T., Eds.; Springer: Berlin/Heidelberg, Germany, 2004; pp. 354–358.
25. Agha, G.; Meseguer, J.; Sen, K. PMaude: Rewrite-based Specification Language for Probabilistic Object Systems. *Electron. Notes Theor. Comput. Sci.* **2006**, *153*, 213–239. [CrossRef]
26. Zayani, H.; Barkaoui, K.; Ayed, R.B. Probabilistic verification and evaluation of Backoff procedure of the WSN ECo-MAC protocol. *Int. J. Wirel. Mob. Netw.* **2010**, *2*, 156–170. [CrossRef]
27. Fischer, F.; Deutscher, J. Algebraic fault detection and isolation for parabolic distributed–parameter systems using modulation functions. *IFAC-PapersOnLine* **2016**, *49*, 162–167. [CrossRef]
28. Silva, D.S.; Resner, D.; de Souza, R.L.; Martina, J.E. Formal Verification of a Cross-Layer, Trustful Space-Time Protocol for Wireless Sensor Networks. *Lect. Notes Comput. Sci.* **2016**, *10063*, 426–443.
29. Testa, A.; Cinque, M.; Coronato, A.; De Pietro, G.; Augusto, J.C. Heuristic strategies for assessing wireless sensor network resiliency: An event-based formal approach. *J. Heuristics* **2015**, *21*, 145–175. [CrossRef]
30. Babich, F.; Deotto, L. Formal methods for specification and analysis of communication protocols. *IEEE Commun. Surv. Tutor.* **2002**, *4*, 2–20. [CrossRef]
31. Wang, J.; Li, D. Resource Oriented Workflow Nets and Workflow Resource Requirement Analysis. *Int. J. Softw. Eng. Knowl. Eng.* **2013**, *23*, 677–693. [CrossRef]
32. Lefebvre, D.; Aguayo-Lara, E. A Discussion on Fault detection for a class of Hybrid Petri Nets. *IFAC-PapersOnLine* **2017**, *50*, 6837–6842. [CrossRef]
33. Cho, J.H.; Chang, M.; Chen, I.R.; Swami, A. A Provenance-Based Trust Model for Delay Tolerant Networks. In *Trust Management VI*; Dimitrakos, T., Moona, R., Patel, D., McKnight, D.H., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; pp. 52–67.
34. Lory, P. A colored Petri net trust model. In Proceedings of the 14th International Workshop on Database and Expert Systems Applications, Prague, Czech Republic, 1–5 September 2003; pp. 415–419. [CrossRef]

35. Kapitanova, K.; Son, S.H. MEDAL: A coMpact event description and analysis language for wireless sensor networks. In Proceedings of the 2009 Sixth International Conference on Networked Sensing Systems (INSS), Pittsburgh, PA, USA, 17–19 June 2009; pp. 1–4. [CrossRef]
36. Wang, N.; Chen, X. A Formal Model for Temporal—Spatial Event in Internet of Vehicles. *Lect. Notes Comput. Sci.* **2018**, *11280*, 222–234.
37. Hodge, V.J.; Austin, J. A Survey of Outlier Detection Methodologies. *Artif. Intell. Rev.* **2004**, *22*, 85–126. [CrossRef]
38. Chandola, V.; Banerjee, A.; Kumar, V. *Outlier Detection: A Survey*; Technical Report 07-017; University of Minnesota: Minneapolis, MN, USA, 2007.
39. Feng, R.; Che, S.; Wang, X.; Yu, N. Trust Management Scheme Based on D-S Evidence Theory for Wireless Sensor Networks. *Int. J. Distrib. Sens. Netw.* **2013**, *2013*, 130–142. [CrossRef]