Pa-GFDP: an Algorithm Enhancing Reliability of WSNs

GUIPING WANG [1], SHUYU CHEN [2], HUAWEI LU [1] AND MINGWEI LIN [1]
[1] College of Computer Science
Chongqing University
No. 174 Shazheng Street, Shapingba, Chongqing, 400044
CHINA
w_guiping@163.com
[2] School of Software Engineering
Chongqing University
No. 174 Shazheng Street, Shapingba, Chongqing, 400044
CHINA

Abstract: - Connectivity of Wireless Sensor Networks (WSNs) is a minimal requirement for their functionality. However, their distributed and self-organizing nature creates the presence of critical nodes, whose failures may partition the system or create communication bottlenecks. This paper focuses on enhancing reliability of WSNs, through detecting critical nodes and protecting them. The classical centralized algorithms of detecting critical nodes, which are based on DFS, require global topological knowledge. However, there are some dynamic factors in WSNs, such as frequent joining in and departure of nodes, unexpected failure of nodes due to running out of energy, and changes in network connections, etc. Consequently, the topology of WSNs is dynamic. Therefore, centralized algorithms are impractical. This paper extends the studies on GFDP (Grouping Fault Detection Protocol) and proposes a Partitioning-avoidance GFDP (Pa-GFDP) to enhance reliability of WSNs. Pa-GFDP cost-effectively detects critical nodes in WSNs and protects them. Without global information, the accurate detection of critical nodes can be accomplished with little traffic overhead and within limited time threshold. Pa-GFDP is verified to be correct and effective through simulation and experiments.

Key-Words: - Wireless Sensor Networks (WSNs); Reliability; Connectivity; Critical Node; Fault Detection; Pa-GFDP

1 Introduction

A wireless sensor network (WSN) consists of a set of (from several to thousands) sensor nodes which are able to collect data and communicate with each other by radio links [1]. In recent years, WSNs have been widely applied to habitat monitoring [2], real-time target tracking, environment surveillance and healthcare, etc. Previous researches in WSNs mainly focus on routing protocol [2], congestion detection and avoidance [3], energy-efficient architecture [4], etc. This paper addresses critical nodes detection and protection aiming at enhanceing reliability of WSNs.

In a WSN, a small proportion of the nodes are likely to be more critical to the system's reliability than others. Failures of these nodes may partition the system or create communication bottlenecks. Therefore, it is meaningful to detect and protect critical nodes from reliability point of view.

Based on graph theory, a WSN can be modelled as an undirected graph $G = (V, E)$; where the vertex set V represents nodes, that is, sensors, and the edge set E represents physical links. Intutively, all the cut vertices in a graph are critical nodes. Therefore, the terms "cut vertex" and "critical node" are interchangeable in the remainder of this paper. Cut vertex is an elementary concept of connectivity in graph theory. A node v is a cut vertex if its removal will disconnect the graph into two (or more) separate components, assuming the graph is a connected one initially.

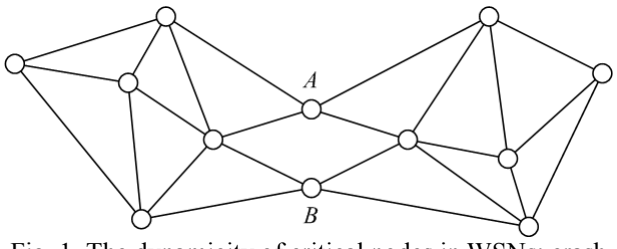


Fig. 1 The dynamicity of critical nodes in WSNs: crash of node A causes node B to become a critical node.

Due to dynamic topological structure of WSNs, the concept "critical node" has deeper meaning. For

example, in Fig. 1, both node *A* and *B* are not a critical node initially. But if node *A* crashes or runs out of energy, node *B* becomes a critical node.

Tarjan [5] first proposed a DFS (Depth First Search) based algorithm to detect critical nodes and links. It is a centralized algorithm, and can be also implemented in globalized distributed manner. A centralized algorithm requires that a node should be aware of global topology [6].

However, in WSNs, due to their dynamic and distributed nature, detecting critical nodes based on global topological information is impractical. The main reasons are summarized as follows.

(1) Dynamicity: such as frequent joining in and departure of nodes, unexpected failure of nodes due to running out of energy, and changes in network connections. This dynamicity makes it is difficult to collect global topological information.

(2) Traffic overhead: collecting global information usually consumes a large amount of energy and comparatively high traffic overhead, which is not tolerable in WSNs.

(3) Time consumption: collecting global information also usually costs comparatively long time. During this time, global information may have changed, which makes it is meaningless to collect global information.

Therefore, in WSNs, solutions where nodes declare themselves as critical ones based on limited locally available knowledge are needed [6], which means nodes can only use relatively limited information to estimate the probability of being a critical node.

For critical node detection in WSNs, two wrong results may occur after the detection process.

(1) False positive (FP): a normal node wrongly identified as a critical one.

(2) False negative (FN): a truly critical node has not been detected.

Considering trade-off between traffic overhead and preciseness, a 100% precise solution is not necessary as long as all the cut vertices are detected and the number of wrongly identified cut vertices is small [7]. That is, a cost-effective solution should meet three requirements: traffic lightweight and energy efficient, zero false negative rate, false positive rate confined to a low constant.

Failure of critical nodes is usually a fatal strike for WSNs. Therefore, critical nodes detection and protection is an effective method for enhancing reliability of these systems. To this end, based on our early work [8] in engineering applications of Graph Theory, we design an enhanced Grouping Fault Detection Protocol (GFDP [9]) called Partitioning avoidance GFDP (Pa-GFDP) in this paper. The highlights of Pa-GFDP are summarized as follows:

1) Local: Without global knowledge, a node can identify whether it is a critical one based solely on local information;

2) Adaptive accuracy: Using limited information to estimate a critical node with high probability;

3) Traffic lightweight: In most cases any extra traffic does not need to insert into the network, additional but acceptable traffic overhead is needed if/when necessary;

4) Dynamic: As nodes could join in and leave frequently, Pa-GFDP is able to run at each node at any time to identify itself whether it becomes a critical one at the given time and protect it.

The remainder of this paper is organized as follows. Section 2 discusses related work. System model and definitions are introduced in section 3. Section 4 presents the proposed algorithm, namely, Pa-GFDP. Simulation and experiments are presented in section 5. Lastly, this paper is concluded in Section 6.

2 Related Work

Reliability refers to the property that a system can run continuously without failure [10]. It is an important attribute of a more extensive concept, i.e., dependability. Dependability subsumes the usual attributes of reliability, availability, safety, integrity, maintainability, etc. Avižienis et al. [11] give the main definitions relating to dependability, some useful comments, and serval supplementary definitions, which address the threats of dependability, their attributes and the means for ehancing dependability.

Over the past few years, numerous methods and techniques have been proposed to detect and monitor failures, and various schemes and mechanisms have been developed to improve the reliability of systems [9, 12-14]. For example, the authors in [13] propose a methodology based on an automatic generation of a fault tree to evaluate the reliability and availability of Wireless Sensor Networks, when permanent faults occur on network devices.

In large-scale distributed systems, including WSNs, Ad hoc, P2P, etc. a large class of approaches enhances reliability through predicting, analysing, avoiding or delaying partition. These approaches fall into two categories.

(1) "Post-partitioning" approaches [15, 16]: working on existing partitions or after the partitioning has occurred.

(2) "Pre-partitioning" approaches [1, 6, 7, 17, 18]: trying to avoid or delay partitioning by identifying critical node or links in the system and reinforcing them. It works before the partitioning can occur.

For the former approaches, routing protocols like gateway routing [15] address the issue in terms of quick and efficient recovery from partition. Other works, like [16], attempt to improve the communication without avoiding or delaying partitioning. This can, however, be too late in many scenarios, and the goal of this paper is to detect critical nodes and protect them so that possible partitions are avoided.

For the latter approaches, authors in [6] survey existing prediction concepts and discuss their scalability, simplicity, correctness and communication overhead. The concepts of p-hop neighbor and p-hop critical node, which are restated in Section 4.1, are especially useful to this paper. Authors in [7] propose a novel completely distributed scheme where every single node can determine whether it is a cut vertex or not. They focus on the P2P systems and assume the communication between peers is mainly by flooding. Since flooding may cause local network congestion, this paper adopts gossiping instead of flooding.

Sheng et al. [17] present a critical node detection algorithm - DMCC (Detection algorithm based on Midpoint Coverage Circle), which is only applied to Ad hoc network. The algorithm classifies the nodes into three categories, that is, global critical nodes, local critical nodes and ordinary nodes. Goyal et al. [18] focus on critical links detection. The algorithm in [1] travels all the nodes of a network in parallel, colors the edges based on the interval-coded spanning tree, and then determines the cut vertices by counting the edge colors. Therefore, it is a centralized algorithm.

The role of high degree node in [19] is similar to that of critical node in this paper. High degree nodes are more vulnerable to attacks. But detecting high degree nodes is less difficult than locating critical nodes. The method in [19] tries to reduce high degree nodes and is invoked only when an attack occurs. However, this method cannot improve the reliability of topology for the normal cases when there are no attacks.

One of the most efficient ways to enhance the system's reliability is failure detection and monitoring. Freiling et al. [20] survey the failure detector as a fundamental abstraction in distributed computing. Two main approaches have been used to implement failure detectors: the *heartbeat* mechanism and the *pinging* mechanism [21]. The former is considered as a more practical approach compared to the latter. Therefore, this paper adopts heartbeat mechanism. The classical heartbeat approach is based on the continuous monitoring of a node to know whether it is alive or not. Hence, each node transmits an "*I Am Alive*" message periodically to nodes that monitor its state. The heartbeat message is given by a *Beat Counter*. After the expiration of a timeout, if a node u does not receive such a message from one of the neighbors, say v, u is in charge of detecting v's failure, then u starts suspecting v as being faulty. Node u adds v to a list of suspected nodes.

3 Model and Definitions

Several strict definitions are given in this section, which describe the model of a WSN system, its failure-detection system and some related concepts.

3.1 Models

Definition 1(model: a WSN system): a WSN system is modelled as an undirected graph $G = (V, E)$, where each node in vertex set $V = \{ v_1, v_2, \ldots, v_n \}$ represents a sensor, and the edge set E represents physical links between nodes. G is usually a connected (or initially connected) graph.

Any node in a WSN system may join in, crash or run out of energy randomly.

Definition 2(model: a failure-detection system): a failure detector, which is an abstract presentation of a module or a process, is deployed on each node, therefore the detectors themselves form a distributed a failure-detection system, which can be represented by $O = \{ d_1, d_2, \ldots, d_n \}$. Any node v_i in V has a corresponding detector d_i in O, d_i is called a failure detector attached to v_i.

An active node means its attached detector participates in the failure-detection system; while a failed node means its attached detector may be crashed or detached.

For ease of expression, the terms "node" and its attached "detector" are interchangeable in the remainder of this paper.

3.2 Graph-theoretical definitions

Definition 3(Component): A component of a disconnected graph is a maximally connected subgraph.

For a connected graph, it contains one component, that is, itself. While in a disconnected graph each separate part is a component.

Definition 4(Critical Node or CN for short): CN is also referred as cut vertex in graph theory. A CN is a vertex of a graph such that its removal causes an increase in the number of components.

Definition 5(Critical Link or CL for short): CL is also referred as cut edge or bridge in graph theory. Similarly, a CL is an edge whose removal causes an increase in the number of components.

Definition 6(*k*-connected): For a connected graph, if it is always possible to establish a path between any pair of vertices after removing **any** (*k*-1) vertices, while after removing **some** *k* vertices the graph is disconnected, then the graph is said to be *k*-connected. And vertex connectivity degree of this graph is said to be *k*.

Definition 7(Block, bi-connected component): A block of a connected graph, also referred as a bi-connected component, is a maximally connected subgraph having no CN.

Note that, 2-connected is not equivalent to bi-connected. A 2-connected graph or component means its vertex connectivity degree is 2. While a bi-connected component means it has no CN. Its vertex connectivity degree is greater than or equal to 2.

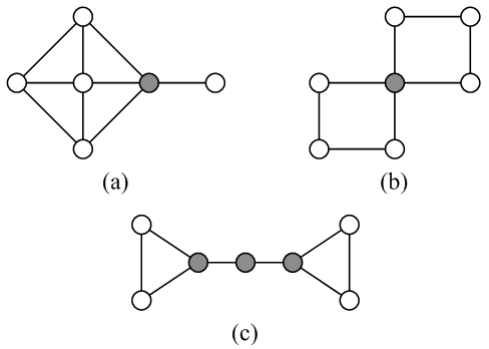


Fig. 2 Three cases of CN: (a), the joint of a bi-connected component and a bridge; (b), the joint of two bi-connected components; (c), the joint of some bridges.

Three possible cases are illustrated in Fig. 2, and the CNs are coloured in grey. In Fig. 2(a), the CN is the joint of a bi-connected component and a bridge. In Fig. 2(b), the CN is the joint of two bi-connected components. In Fig. 2(c), the middle CN is the joint of some (two, here) bridges. Indeed, all CNs fall into these three cases.

To simplify discussion, this paper focuses on critical node detection in this paper. Critical link will not be considered.

4 Pa-GFDP

This section describes the proposed localized algorithm, i.e., Pa-GFDP, to identify and protect the critical nodes under a large-scale distributed WSN environment. We first introduce three extended definitions, which are important to Pa-GFDP. Then the main thoughts and concrete processes of Pa-GFDP are described. Lastly, we analyse its traffic overhead.

4.1 Preliminaries – Extended definitions

The definition of critical node (Definition 4) is based on global topology of a graph. This concept should be extended to p-hop critical node in WSNs since limited locally information is available.

Definition 8(p-hop neighbor): Two nodes are considered to be p-hop neighbors if and only if the shortest route between them has p or less hops [6].

Definition 9(p-hop critical node, phCN for short): For each node *v*, considering a subgraph of p-hop neighbors of *v*, where *v* and all its incident edges are excluded, *v* is a p-hop critical node if the corresponding subgraph of p-hop neighbors of *v* is disconnected [6].

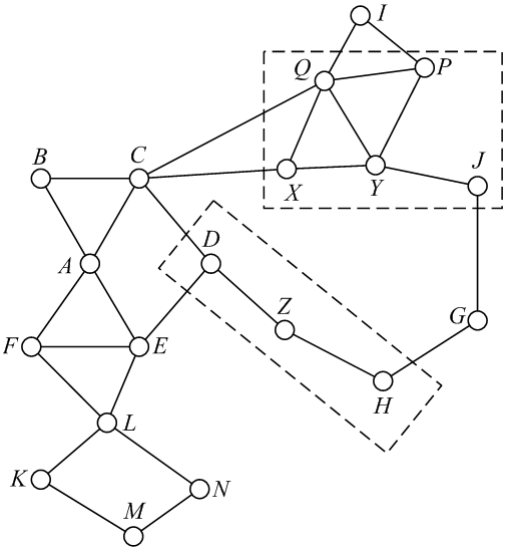


Fig. 3 Node G is not a global cirtical node, but is 3-hop critical one. (cited from [6] with minor modification)

A global critical node is always a p-hop one, while a p-hop critical node is not always a global one.

For example, in Fig. 3, Node *G* is not a global critical node, but it is a 3-hop one since its 3-hop neighbors are divided in two sub-graphs with vertices {*H*, *Z*, *D*} and {*J*, *Y*, *X*, *Q*, *P*} (these two sets are enclosed by dashed polygons), which are disjoint.

Note that, *G* is also a 1-hop and 2-hop critical node but not a 4-hop one since node *C* joints the aforementioned two sub-graphs.

Definition 10(a localized algorithm): As opposed to centralized algorithms, a localized algorithm discovers critical nodes based on limited locally available information. In other words, the critical nodes discovered by a localized algorithm are p-hop ones, not global ones.

With smaller communication overhead, localized algorithms provide faster and often more reliable partition warnings for possible timely replication or protection decisions.

However, localized algorithms may detect some nodes as p-hop critical although they may not be globally critical, that is, false positives occur. This is unavoidable since local knowledge only is used, and with such restriction it is impossible for a node to learn about alternate connections in different parts of the network. But the false positives mostly occur when alternative routes exist but are relatively long, and therefore may not provide satisfactory service in applications. Thus, a localized method may provide even more useful decisions, which is verified by simulation and experiments in Section 5.

4.2 Algorithm Specifications

Pa-GFDP enhances reliability of WSNs through detecting phCNs and protecting them. There are two phases, i.e., *phCN detection* and *phCN protection*. The former consists of two processes, *passive detection* and *active detection*.

With the help of information interchanged between nodes, this paper designs a zero-overhead passive detection method to identify phCNs. The accuracy of passive detection can be reinforced by an active detection method with a fairly low cost. Both passive and active detections are new functions added into Pa-GFDP, and both detections will be executed periodically to reflect the topologically change of failure-detection system. When phCNs are detected, they are protected to enhance system's reliability.

4.2.1 Passive phCN Detection

In the failure-detection system which running Pa-GFDP, a message is propagated from one detector to others hop-by-hop using the gossiping mechanism. The forwarding message contains the path information from the starting detector to the current one. This paper proposes to utilize such information to find phCNs by sorting the neighbors of a detector into one or more blocks. According to the aforementioned definition (definition 9), a detector is denoted as a phCN if its p-hop neighbors belong to multiple blocks.

In this scheme, the following characteristics of gossiping are assumed in the failure-detection system.

(1) A detector forwards the received message to part of its local view neighbors, that is, direct neighbors, except the one where the message came from. The number of forwarding destination nodes is determined by the pre-configured gossiping fan-out, which is denoted as *c* in this scheme.

(2) Each message is assigned a globally unique message ID. Since every detector in the failure-detection system only maintains a local view of its own, it does not have global knowledge of the system. As every detector in the system has a unique ID $d_i.id$, and it has an aforementioned *Beat Counter*, a message could be generated by combining $d_i.id$ with *Beat Counter*, which is denoted as *Msg.id* in this scheme.

(3) A detector gossips each message only once. If it receives the same message later, it simply drops the duplicates.

In the passive detection, each node keeps track of recently received messages. A list of records is cached on each node, called *MsgList*, with each entry representing a received message in the format of < *Msg.id*, list of $d_i.id$ that the message has traversed nodes >. A node also records the block flag of its neighbors. At the beginning of a passive detection period, *MsgList* is empty and all neighbors of a node are assumed to be in different blocks.

During the periodical execution of passive detection, a node randomly receives messages from its neighbors. By examining the path information of the received messages, it could discover circles formed by its neighbors. Thus a node could deduce

that all neighbors on a circle belong to a block including itself. When two circles shares one or more nodes then it could deduce that the nodes on both circles are in the same block. This is how blocks merging and flags changing in a node's local view. As more messages arrive, node keeps merging the blocks and changing the block flags of its neighbors. The pseudo code of function *On_Recieving*( ) is illustrated in Algorithm 1.

Algorithm 1(Pseudo code of function *On_Receiving*( ))

```
/* msg: a gossiping message;
   msg.TTL: pre-configured hops times msg can be forwarded;
   msg.id: the message ID of msg;
   N: the detector receiving msg.*/
1.  On_Receiving( )
2.  {
3.    if( N receives msg for the first time )
4.    {
5.      N creates a new entry in N.MsgList;
6.      if( msg.TTL > 0 )
7.        Gossip(msg);  //forwarding the msg
8.    }
9.    else  //N has received the msg before
10.   {
11.     N creates a new entry in N.MsgList;
12.     FindEntry( N.MsgList, Msg.id ); //Find entries
13.     MergeBlk( N.LocalView );  //Merge blocks
14.     N drops msg;
15.   }
16.}
```

In the above-mentioned function, when a detector *N* receives a message for the first time (Line 3-8), it creates a new entry in *N.MsgList*. Then it judges whether or not to gossip the message according to the remaining *TTL* (time to live) value. Otherwise (Line 9-15), it finds entries in *N.MsgLIst* with same *Msg.id* of the received message. If any such entry is found, it merges blocks containing the nodes which deliver message with same *Msg.id*.

After running a determined period of passive phCN detection, all the neighbors of a node will probably be merged into a few blocks. If there are only one block flag in a node's local view, it is not a phCN. Otherwise, an active detection process is triggered for further determination, which is introduced below.

Note that the passive detection is executed passively during the process of message dissemination. All the information is attached in the normal messages, so it would not incur any additional traffic overhead.

4.2.2 Active phCN Detection

With the passive detection, a node knows for sure that it is not a phCN if all the neighbors belong to a single block. However, having two or more blocks remaining at the end of passive detection does not determinately mean a node is a phCN. For example, a node might not be able to receive messages containing all possible paths from its neighbors because the messages are forwarded in a manner of gossip, or the message *TTL* threshold is reached.

In order to further identify phCNs, an active detection is necessary. Compared to the passive detection, the active detection achieves shorter convergence time at the cost of additional but acceptable traffic overhead.

If a node's block flags are not consistent after a long enough period of passive detection, which means that the neighbors of that node are sorted into two or more blocks, it regards itself as a suspect phCN and immediately starts an active detection process.

At first, it randomly selects a neighbor from each block and numbers each of these neighbors with a unique *Block-index* (e.g. 1, 2, 3...). Then the node sends probe messages to these neighbors. The format of the probe message is < d_i*.id*, *Msg.id*, *TTL*, *Block-index* >, where d_i*.id* is the suspect node's ID, *Msg.id* keeps the records the time the probe message is generated, *TTL* is a pre-configured number of hops that the message can be forwarded, and *Block-index* denotes the index of the block to which the suspect sends the probe message. Each node keeps a probe message list. There is one entry for each suspect in prob message list with the format of <suspect's d_i*.id*, *Msg.id*, *Block-index* 1, *Block-index* 2, …>.

Upon receiving a probe message, one of the following situations may arise.

(1) The node has already received the message, or the *Msg.id* of the message is smaller than that stored in the corresponding probe message list. The node just drops the message.

(2) There is no entry for the suspect that issues this probe message. The node creates a new entry for it.

(3) The *Msg.id* in the received message is larger than that stored in the corresponding probe message list. The suspect replaces the older *Msg.id* and

Block-indexes stored in the probe message list with the new one.

(4) The *Msg.id* of the received message is the same as the one stored in the corresponding probe message list, but the *Block-index* of the message is not the same. The node adds the new *Block-index* to the corresponding entry and sends an arrival message back to the suspect. The arrival message therefore contains $d_i.id$ of the current node, two or more *Block-indexes*, and the *Msg.id* stored in the entry. A node does not send any arrival messages until it receives at least two probe messages with different *Block-indexes*.

In this way, the neighbors of a phCN suspect can be merged into fewer and fewer blocks. If only one block remains in its local view, the suspect is not a phCN. Otherwise; the suspect must be a phCN. Once a node has been verified as a phCN, a progress of phCN protection would be trigger, which is discussed below.

Since the initial *TTL* value of a probe message is usually small, a conclusion can be made that active detection is much sooner than the passive detection. In other words, the active detection can be applied as a useful complement to the passive detection. It can also be utilized as an independent approach to identify phCNs if speed is valued over cost.

4.2.3 phCN Protection

The goal of phCN protection is to enhance the system's reliability from topology connectivity point of view. For a WSN network, phCN protection is relatively easy to achieve. Serval choices are provided. On the one hand, a phCN node *P* may be given a higher priority in competing channels, or be released from unnecessary sensing jobs to save its power and hence results in a longer lifetime. On the other hand, if possible, an extra node *Q* may be added beside each phCN *P*. Consequently, all nodes in the WSN system will get bi-connected and the phCNs become normal nodes.

4.3 Traffic Overhead

In Pa-GFDP, the communication between nodes is derived from GFDP [9], mainly by gossiping. This paper evaluates the traffic overhead by counting the messages delivered due to the phCN detections. Note that gossiping is adopted in GFDP as the basic mechanism for data dissemination. Even if there is no passive detection, the traffic overhead of gossiping does exist as an element of system running. Hence the passive detection does not incur any additional traffic overhead because it only utilizes the information extracted from the existing messages.

For the active detection, suppose the system has n nodes, let c be the gossiping fan-out and t be the initial *TTL* value.

Note that a detector will not forward the probe message if it has already sent an arrival message back to the corresponding suspect. This paper defines the traversal set as the nodes that are traversed by the same block number of a suspect's probe message. And the traversal sets of different blocks of a suspect will not overlap. As a result, the total traffic overhead of probe messages is $O(n^2c/2)$, where $nc/2$ is the number of edges in the whole WSN. On the other hand, the traffic overhead is also limited by the initial *TTL* value. It can never exceed $O(nc^t)$. Therefore the total traffic overhead of forwarding probe messages is $\min(O(n^2c/2), O(nc^t))$. In the failure-detection system, the value of c is much smaller than that of n. The inequality $c^t \leq n$ holds when the initial *TTL* value t is limited to save traffic cost. Thus it can conclude that the total traffic overhead of active detection is $O(nc^t)$.

5 Simulation and Experiments

In this section, Pa-GFDP is verified through simulation and experiments. The symbols used in this section are listed below.

n: the size of a WSN system.

d: the average degree of all nodes.

m: the number of global critical nodes in a WSN. The concepts of global critical node and truly critical node are equivalent.

p: the number of p-hop critical nodes detected by Pa-GFDP.

c, t: the gossip fan-out and *TTL* value in Pa-GFDP.

5.1 Simulation

Due to dynamic topology of real WSN systems, the centralized Tarjan's algorithm [5] is impractical. Therefore, in order to compare Pa-GFDP with Tarjan's algorithm and test FP rate and FN rate of Pa-GFDP, a program is written to generate random but static WSNs, where n is controlled and pre-defined. Pa-GFDP and Tarjan's algorithm then run upon these WSNs respectively.

Three representatives of random WSNs are shown in Tab. 1. The number of global critical nodes, that is, m, is counted by Tarjan's algorithm

since it is proven to be a correct centralized algorithm.

Tab. 1 Three representatives of random WSNs

WSN no.	*n*	*d*	*m*
1	1000	4.17	57
2	3000	4.42	181
3	5000	4.63	316

Except for FP and FN introduced in Section 1, the other two indices are list below.

True positive (TP): a truly critical node is correctly identified as a critical one.

True negative (TN): a normal node is correctly identified as a normal one.

The value of *c* and *t* in Pa-GFDP are fixed to 3 and 5 respectively, since the real experiments in Section 5.2 show that these values are most appropriate. The running results of Pa-GFDP on the above three representative WSNs are listed in Tab. 2. FP rate is the rate of the number of FPs to the number of normal nodes. Normal nodes include FPs and TNs. Therefore, FP rate = FP / (FP+TN).

Tab. 2 Running results of Pa-GFDP

WSN no.	*p*	TP	FP (rate)	TN	FN
1	63	57	6 (0.64%)	937	0
2	203	181	22 (0.78%)	2797	0
3	350	316	34 (0.73%)	4650	0

The results in Tab. 2 show that the FP rate of Pa-GFDP is fairly low and can be confined to 1% when the system size is enough big and the values of *c* and *t* are appropriate. Just as the results show, FN rate is always 0%. This is because a global critical node is always a p-hop one, and can always be detected by Pa-GFDP algorithm.

Tab. 3 Running time of Pa-GFDP and Tarjan's algorithm

WSN	Tarjan (ms)	Pa-GFDP (ms)
1	286	195
2	817	572
3	1383	912

The configuration of the test machine is: CPU, 2.26 GHz; memory, 3 GB. The running time of Pa-GFDP and Tajan's algorithm on the above three representative WSNs are listed in Tab. 3. These results show that, Pa-GFDP outperforms Tarjan's algorithm in running time since Pa-GFDP is based on limited locally available information.

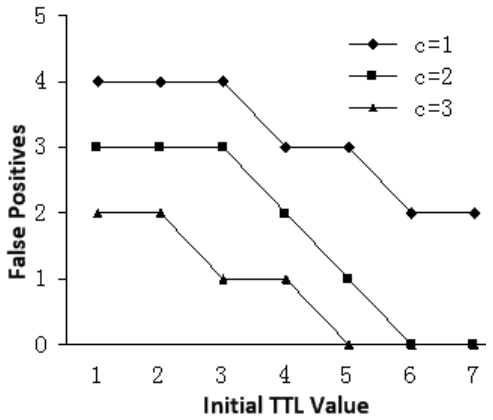
5.2 Experiments

In order to evaluate the effect of the values of *c* and *t* to the accuracy of Pa-GFDP, real WSNs are constructed. The real WSN networks are composed of 30-50 nodes, and each node's sensing range is set at 25 m. The WSN networks are deployed in the No. 9 Laboratory Building of Chongqing University.

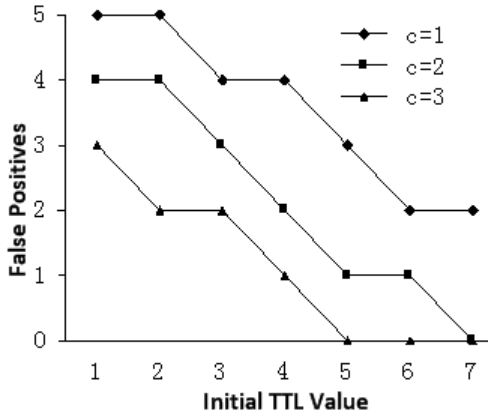
The initial configurations of two representatives of real WSNs are listed in Tab. 4. The number of global critical nodes, that is, *m*, is also counted by Tarjan's algorithm.

Tab. 4 Real WSNs constructed in Experiments

WSN no.	*n*	*d*	*m*
1	30	3.73	6
2	50	4.11	8



(a) WSN 1



(b) WSN 2

Fig. 4 Effects of the values of *c* and *t*.

Effect of the values of *c* and *t* is evaluated in terms of the number of false positives. The results

are shown in Fig. 4. Note that, Fig. 4(a) and Fig. 4(b) correspond to WSN 1 and WSN 2 in Tab. 4 respectively.

The results in Fig. 4 indicate that both c and t effect false positive rate. Smaller values cause more false positives. While larger values incur much higher traffic overhead. The results indicate that c = 3 and t = 5 are most appropriate values, which make best trade-off between traffic overhead and preciseness.

6 Conclusions

The localized Pa-GFDP algorithm enhances system's reliability from topological connectivity point of view, through detecting critical nodes and protecting them. The detection phase consists of two processes, namely, *passive detection* and *active detection*. The former does not incur any extra overhead, the latter costs additional but relatively small overhead. Since there is a trade-off between traffic overhead and preciseness, Pa-GFDP is a more practical solution to WSNs compared with centralized algorithms.

There are some normal nodes wrongly identified as critical ones in Pa-GFDP. Nevertheless, on the one hand, these false positive nodes cause no harms to the system's reliability since protecting these nodes is superfluous but harmless. On the other hand, the FP rate in Pa-GFDP is fairly small and can be confined to a low constant.

Acknowledgements

We are grateful to the editor and anonymous reviewers for their valuable comments on this paper.

The work is supported by Research Fund for the Doctoral Program of Higher Education of China (Grant No. 20110191110038), National Natural Science Foundation of China (Grant No. 61272399) and Fundamental Research Funds for the Central Universities (Grant No. CDJXS12180001).

References:

[1] S.G. Xiong, J.Z. Li, An Efficient Algorithm for Cut Vertex Detection in Wireless Sensor Networks, *In: Proc. of International Conference on Distributed Computing Systems (ICDCS)*, June 2010, pp. 368-377.

[2] S.S. Chiang, C.H. Huang, K.C. Chang, A minimum hop routing protocol for home security systems using wireless sensor networks, *IEEE Transactions on Consumer Electronics*, Vol. 53, No. 4, Nov. 2007, pp. 1483-1489.

[3] L.Q. Tao, F.Q. Yu, ECODA: Enhanced Congestion Detection and Avoidance for Multiple Class of Traffic in Sensor Networks, *IEEE Transactions on Consumer Electronics*, Vol. 56, No. 3, Aug. 2010, pp. 1387-1394.

[4] M.A. Lopez-Gomez, J.C. Tejero-Calado, A Lightweight and Energy-Efficient Architecture for Wireless Sensor Networks, *IEEE Transactions on Consumer Electronics*, Vol. 55, No. 3, Aug. 2009, pp. 1408-1416.

[5] R. Tarjan, Depth First Search and Linear Graph Algorithms, *SIAM Journal on Computing*, Vol. 1, No. 2, 1972, pp. 146-160.

[6] I. Stojmenovic, D.S. Ryl, A. Nayak, Toward Scalable Cut Vertex and Link Detection with Applications in Wireless Ad Hoc Networks, *IEEE Network*, Vol. 25, No. 1, Feb. 2011, pp. 44-48.

[7] Y. He, H. Ren, Y.H. Liu, B.J. Yang, On the reliability of large-scale distributed systems - A topological view, *Computer Networks*, Vol. 53, No. 12, Aug. 2009, pp. 2140-2152.

[8] Guiping Wang, Yan Wang, LDAG: A New Model for Grid Workflow Applications, *WSEAS Transactions on Computers*, Vol. 10, No. 6, June 2011, pp. 179-188.

[9] G.H. Chang, H.W. Lu, S.Y. Chen, I. Shih, Grouping Fault Detection Protocol under Dynamic Network Environments, *In: Proc. of International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)*, July 2010, pp. 151-155.

[10] A.S. Tanenbaum, R. Catherman, *Distributed Systems: Principles and Paradigms*, 2nd ed., pp. 322, Pearson Education, 2007.

[11] A. Avižienis, J.C. Laprie, B. Randell, C. Landwehr, Basic concepts and taxonomy of dependable and secure computing, *IEEE Transactions on Dependable and Secure Computing*, Vol. 1, No. 1, Jan. 2004, pp. 11-33.

[12] C. M. Jeffery, R.J.O. Figueiredo, A Flexible Approach to Improving System Reliability with Virtual Lockstep, *IEEE Transactions on Dependable and Secure Computing*, Vol. 9, No. 1, Jan. 2012, pp.2-15.

[13] I. Silva, L.A. Guedes, P. Portugal, F. Vasques, Reliability and Availability Evaluation of Wireless Sensor Networks for Industrial Applications, *Sensors*, Vol. 12, No. 1, Jan. 2012, pp. 806-838.

[14] H.W. Lu, S.Y. Chen, X.Q. Zhang, G.H. Chang, Byzantine-Tolerant Grouping Fault Detection Protocol under High Churn Networks, *In: Proc.*

of International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA), July 2011, pp. 624-628.

[15] S. Raghunathan, S. Venkatesan, and R. Prakash, Gateway routing: a cluster based mechanism for recovery from mobile host partitioning in cellular networks, *In Proc. of 3rd IEEE Symposium on Application-Specific Systems and Software Engineering Technology (ASSSET)*, Apr. 2000, pp. 135-39.

[16] A. Vahadat and D. Becker, Epidemic routing for partially connected ad hoc networks, Technical Report CS-200006, Duke University, Apr. 2000.

[17] M. Sheng, J.D. Li, Y. Shi, Critical nodes detection in mobile Ad Hoc network, *In Proc. of 20th International Conference on Advanced Information Networking and Applications (AINA)*, Apr. 2006, pp. 336-340.

[18] D. Goyal, J. Caffery, Partitioning avoidance in mobile ad hoc networks using network survivability concepts, *in Proc. of 7th IEEE International Symposium on Computers and Communications (ISCC)*, July 2002, pp. 553-558.

[19] P. Keyani, B. Larson, M. Senthil, Peer pressure: distributed recovery from attacks in peer-to-peer systems, *In Proc. of IFIP Workshop on Peer-to-Peer Computing*, May 2002, pp. 306-320.

[20] F.C. Freiling, B. Guerraoui, P. Kuznetsov, The Failure Detector Abstraction, *ACM Computing Surveys*, Vol. 43, No. 2, Article 9, Jan. 2011.

[21] M. Elhadef, A. Boukerche, A failure detection service for large-scale dependable wireless ad-hoc and sensor networks, *In Proc. of 2nd International Conference on Availability, Reliability and Security (ARES)*, Apr. 2007, pp. 182-189.