A HIERARCHICAL APPROACH TO FAULT DIAGNOSIS IN LARGE-SCALE SELF-DIAGNOSABLE WIRELESS ADHOC SYSTEMS

[1]P.M.Khilar, [2]S.Mahapatra
[1]Department of CSE, NIT, Rourkela- 769 008
[2]Department of E & ECE, IIT, Kharagpur- 721 302
Email: khilarpm@yahoo.com

ABSTRACT

This paper proposes a multi-cluster hierarchical distributed system-level diagnosis approach for large-scale self-diagnosable distributed systems, such as one formed by overlaying mobile adhoc networks (MANETs) over another class of adhoc network, known as wireless sensor networks (WSN). The proposed diagnosis strategy assumes multiple numbers of initiators that initiate the diagnosis process in contrast to a diagnosis strategy having one initiator node, which creates a bottleneck in the system. The diagnosis strategy also avoids a costly distributed diagnosis algorithm where every node is an initiator of the diagnosis process. The approach enables the diagnosis at the host level where some external operator can collect all the diagnostic information accessing any active node after issuing a command to the initiator nodes at the highest layer in the hieararchy. The sensor nodes in the lowest layer in the hierarchy are static tested only nodes and thus do not maintain any kind of diagnosis information. Key results of this paper include an adaptive heartbeat-comparison based testing mechanism and fault models and an efficient and scalable distributed diagnosis algorithm using clustering that provides every active node a global diagnostic view of all the nodes. The correctness proof of the algorithm has been given. The analysis of algorithm has shown that the diagnosis latency and message complexity of the algorithm are $O(l_c(T_x + T_f + \max(T_{out1,}T_{out2})) + T_{xcg}$) and $O(n_cC_s)$ respectively. The simulation results show that the diagnostic latency, message complexity and diagnosability of the proposed clustering approach is better than the equivalent non-clustering approach.

Keywords: *Ad-hoc networks, clustering, diagnosability, diagnosis latency and message complexity*.

1. INTRODUCTION

Wireless Adhoc Networks such as mobile adhoc networks (MANETs) and wireless sensor networks (WSNs) are fully autonomous distributed systems and can be set up anywhere and at any time without either a wired backbone or centralized control. The typical distributed systems comprising of mobile nodes in MANETs and static nodes in WSNs form an arbitrary network topology and are of great use in many application scenarios such as in battlefield/emergency/rescue operations for monitoring the environment. Fault diagnosis of wireless adhoc networks with static and mobile nodes is an important problem in designing a fault tolerant system for smooth running of the applications. The static and mobile nodes are subjected to crash (hard) and value (soft) faults. A mobile node is subjected to a crash fault, when it goes out of range or is physically damaged or due to a battery failure. A value faulty mobile node may produce erroneous result due to incorrect computation. Similarly, a static sensor node is subjected to crash fault due to either physical damage or due to battery failure. The sensor nodes are subjected to value fault due to incorrect computation.

The existing diagnosis algorithms for wireless distributed systems result in diagnosis overhead such as high diagnostic latency and message complexity [1-3]. They assume a single initiator node that initiates the diagnosis process thus creating a centralized bottleneck. In case of failure of the initiator, all other nodes wait for diagnosis information and this results in starvations. The assumptions of a static fault environment i.e., the nodes do not change their status during execution of the algorithm is rather impractical. Particularly, the algorithms are not suitable for MANET because mobile nodes move from one place to another place during the execution of the diagnosis procedure. Therefore, the existing diagnosis algorithms cannot handle such situations. The diagnosability of the network is limited by its connectivity [14-15]. Moreover, the existing algorithms assume that a node can test another node directly, which is not feasible for wireless distributed systems due to their time varying topology.

In order to address the above problems, this paper proposes a diagnosis approach for large-scale self-diagnosable wireless distributed systems such as an overlayed MANET on a WSN. Our goal is to allow every active node to learn the status of every other active node in the system. The passive sensor nodes are tested only nodes and need not maintain the global diagnostic view.

The main contributions of this paper are as follows: (i) it provides a hierarchical clustering method that resolves the scalability issue over a non-clustering method; (ii) both static as well as dynamic fault environments are assumed where the static and mobile nodes in different clusters are subjected to crash and value faults; (iii) the presence of multiple initiator nodes avoids the centralized bottleneck of having a single initiator node; (iv) As all the nodes do not act as initiators, the diagnosis overhead is significantly reduced compared to traditional distributed diagnosis techniques where every node needs to learn the status of every other node.

The paper is organized as follows: in section 2, we describe system and fault model that exactly describes the adhoc networks where the nodes either remain stationary or allowed to move from one place to another. The different notations used are also given in this section. Section 3 describes the distributed diagnosis algorithm using clustering approach for wireless ad-hoc systems. The correctness proof and complexity analysis of the algorithm have been presented in section 4 where its cost is computed in terms of diagnosis latency and message complexity. The simulation results are presented in section 5 and end the paper with concluding remarks in section 6.

2. PRELIMINARIES

In this section, we describe the system model, fault model, diagnostic model and the basic definitions with different notations and their meanings. The system model specifies the topological structure of the underlying network with different system parameters. The fault model specifies the types of faults the system components are subjected to. The diagnostic model presents the mechanism used to detect the specified faults.

2.1 System Model

We consider the system of n wireless nodes (static or mobile), which communicate via radio transceivers. The wireless nodes are either switches or endpoints. In MANET or WSN, all devices deployed are not identical, and hence we treat some nodes as switches and some other nodes such as sensor nodes as endpoints. Each switch is equipped with transceivers and processing logic within it whereas each endpoint has only sensing and transceivers logic without any processing logic as such. A synchronous system is assumed where the execution time of diagnosis tasks and communication delays are bounded. Generic parameters are assumed for executing the diagnosis tasks, send initiation time, and propagation time of the heartbeat and the diagnostic messages. Each node is assigned with a unique identifier and can be encoded using log(n) bits. Each node knows its identifier and the identifiers of its neighbors.

We assume that all the transmissions from any node u are omni-directional. Thus, any node in its neighborhood, i.e., within its transmitting range, can receive a message sent by u. The neighborhood of node u is denoted by N(u). The topology of the system can be described as a directed graph G = (V,E), called the communication graph, where V is the set of nodes and E is the set of edges connecting the nodes. Given any $u,v \in V$, directed edge $(u,v) \in E$ exists if and only if $v \in N(u)$. We assume that there exists a link level protocol providing the following services: a MAC protocol is executed to solve contentions; the protocol provides two communication primitives: 1-hop reliable broadcast (1_hB) and Selective Send (SelSend). Primitive 1_hB(m) delivers a message m to all the nodes in the senders neighborhood, while primitive SelSend(v,m) delivers m only to neighboring node v. It is assumed that the underlying network protocol handles the message delivery and message collisions. The communication graph is assumed to remain connected at all times. The network is divided into a number of clusters at various layers in the hierarchy. Every cluster in the network may have one or more mobile nodes along with a set of stationary nodes.

2.2 Fault Model

The wirelessly connected nodes in the network are subjected to crash and value faults. Crash faulty nodes are unable to communicate with the rest of the system, due to either physical damage, or battery depletion or being out of range. Value faulty nodes produce and communicate erroneous values while processing the data packet [16]. A crash fault model captures not only the faulty nodes due to physical damage or battery depletion but also the mobile nodes that are out of range. Both static and mobile nodes are subjected to value faults. New faults may occur during the execution of the diagnosis protocol (viz, other than those already present before the start of the diagnosis session), which leads to a dynamic fault environment. The dynamic fault model also captures the faulty cluster heads and faulty initiator nodes during execution of the diagnosis algorithm. The set of fault-free and faulty nodes are denoted by FF and F, with $|FF| = ff \geq 0$ and $|F| = f \geq 0$ respectively.

2.3 Diagnostic Model

The algorithm assumes a heartbeat based testing mechanism to detect faulty nodes in a cluster. A node x can test another node y if y is a neighbor of x. Testing is assumed to be error-free. The diagnostic model that is used to identify the crash and value faulty nodes is based on an adaptive heartbeat [18] and comparison-based technique as follows. A time-out mechanism is used to detect a crash fault. If a testee node does not respond within certain timeout period T_{out1} or T_{out2}, we say the node is crash faulty. Whereas, to detect a value fault, we assume that each wireless node has an estimated value (may be the remaining energy value or value of a diagnosis task) already known to it. The observed values obtained from the tested nodes are compared with the estimated value and their difference is computed. If the deviation is greater than certain threshold θ, we say that the node is value faulty. The time-out mechanism can be implemented using an interrupt by the tester nodes such that it generates a high-observed remaining energy value [17]. When this time-out value is compared with the estimated remaining energy value, the difference between estimated and observed remaining energy value will be automatically beyond the threshold θ, thus recording a crash fault event. The main reason for incorporating this fault model is to reduce the model to the state of the art diagnosis mechanism called the comparison-based diagnosis. Therefore, both crash and value faults in wireless nodes are detected by the comparison outcomes between tester and testee nodes [19-20]. The set of comparison outcomes or test results is known as a syndrome. The fault-free nodes achieve the diagnosis based on the syndromes collected so far.

The algorithm assumes that the diagnosis process is initiated by a set of fault-free nodes at the highest layer of clusters, henceforth called initiators, in response to an explicit request of an external operator. There are two types of messages exchanged during the diagnosis execution: fixed size heartbeat messages, and variable size diagnostic messages. The heartbeat messages are of two types such as initiation heartbeat message and response heartbeat message. The format of the heartbeat message sent by a node u contains (u, v, diagnostic value, message-code), where v is the identifier of the node from which u received the first heartbeat message (u itself if u is the initiator), diagnostic value is the result of the testing task and the message code is used to identify the type of heartbeat message. Heartbeat messages with comparison-based mechanism are used to detect the crash and value faulty nodes. The diagnostic messages exchanged during the execution of the algorithm are of two types: the local diagnostic message and the global diagnostic message. The format of a diagnostic message sent by a node u contains (u, $F_{u,}$ message code), where F_u is the set of identifiers of the nodes currently diagnosed as faulty by node u and message code is the code to identify the type of message. The local diagnostic messages are used to contain the identities of faulty nodes within a subtree rooted at a cluster head. Whereas, global diagnostic messages are used to contain the identities of faulty nodes in the entire network. A variable diagnostic message size was chosen to accommodate the faults that would occur during the diagnosis process as well as the faults that were present before the start of a diagnosis session. Since the messages keep only the status of the faulty nodes, the message size is reduced.

In order to maintain the status of the nodes about the entire network, each cluster head maintains a vector known as Status_Table[u] which stores the status of each node u in the network. An entry 0 or 1 in this vector indicates that the status of node u is fault free or faulty respectively, and x denotes that the status of a node u is unknown. The corresponding entry in this vector is updated as and when any event is known to occur in that node. Note that, the main aim of the algorithm is to achieve distributed diagnosis, i.e., every node in the system needs to learn the status of every other node in the system. We will show that our diagnosis approach allows every node to obtain a global diagnostic image without testing all the system nodes.

2.4 Definitions and Notations

We present the following definitions, which are common in diagnosis literature. We again include them here for the sake of completeness of the paper.

Definition 1: The connectivity k(G) of a graph G is the minimum number of nodes whose removal results in a disconnected graph.

Definition 2: The cluster connectivity k_c is the minimum number of nodes in the cluster whose removal makes the clusters a disconnected graph.

Definition 3: A wireless network described by the communication graph G is said to be t-diagnosable if correct diagnosis is always possible provided the number of faults does not exceed t. The largest integer t for which G is t-diagnosable is called the diagnosability of G, denoted t_G.

Definition 4: A diagnosis session means the time elapsed between the initiations of the diagnosis session till all other nodes receive the global diagnostic message of the entire network. This is known as diagnosis in on-demand basis.

The notations used in this paper are summarized in Fig. 1.

3. THE DESCRIPTION OF DIAGNOSIS APPROACH

In this section we introduce the diagnosis approach for wireless adhoc networks using a hierarchical clustering method. Clustering or partitioning methods have been suggested to provide scalable solutions to the diagnosis problem in many large networking systems [4-13]. The nodes in wireless systems are expected to autonomously group themselves into clusters, each of which functions as a multi-hop packet radio network. A hierarchical structure allows the clusters to be organized into different layers. The lowest layer contains all the nodes in the network. Some nodes in the upper layers are the cluster representatives of the clusters at the immediate lower layer. The representatives of the clusters in a layer form the next higher layer. The instance of the clustering scheme operating at each layer places the nodes into multiple clusters.

The wireless network clustering problem is defined as follows: Let C_s be a positive integer, such that, $1 \leq C_s \leq |V|$. For each connected component, find a collection of subsets $V_1, \ldots, Vn_c$ of V, so that the following conditions are met.

1. $\cup V_i = V$ for i = 1 to n_c. All vertices are part of some cluster.
2. $G[V_i]$, the subgraph of G induced by V_i is connected.
3. $1 \leq |V_i| < C_s$. This is the size bound for the clusters.
4. $|V_i \cap V_j| \sim O(1)$. Two clusters should have up to a small constant number of common nodes.

We consider the case when all the nodes in the network have the same transmission range. In this case, the underlying communication graph consist of a value $R \geq 0$ and an embedding of the vertices in the plane, such that (u, v) is an edge if and only if $d(u, v) \leq R$. For such graphs, it can be seen that if a node has many neighbors, i.e., a vertex has very high degree, then all these vertices have to be within its transmission radius. These neighboring nodes will, therefore, be relatively close to each other. As a consequence many of these neighboring nodes will be within transmission range of each other and will have edges between themselves in the communication graph. Since the transmission range depends on the power available at the node, in general, for a homogeneous set of nodes, the transmission radii would be close to each other. We do not consider here the nodes having different transmission ranges.

Figure 1. The different notations and their meaning

Cluster formation consists of the discovery of a Breadth First Search (BFS) tree and then forming clusters. We use a BFS tree because its radius is bounded by the diameter of the graph. Fig. 3 gives the algorithm for the discovery of a BFS tree and cluster creation. Fig. 4 outlines the distributed diagnosis technique, whereas Fig. 5 and Fig. 6 present the procedure to merge the smaller partial clusters and to split larger clusters respectively. The proposed diagnosis approach using hierarchical clustering has two phases: Cluster creation and cluster diagnosis. The cluster creation phase is invoked when the existing clustering falls below a quality threshold. Cluster diagnosis is the second phase that handles node mobility and other usual dynamics such as crash and value faults in the network.

3.1 Discovery of a BFS tree and Clustering

The BFS tree is discovered for the undirected graph G(V,E). The procedure assumes the node S as the starting vertex, or the source node. The purpose of a BFS tree is to discover all the nodes reachable from the source node S, or to measure the number of hops between S and any reachable node. In an unweighted graph, the paths in this BFS tree are identical to the shortest paths between S and the reachable nodes. At last, the parent of each node in the BFS tree is found and recorded. The algorithm GRAPH-CLUSTER can be initiated by any node in the network. There are two parts in cluster creation: Tree Discovery and Cluster Formation.

The messages for Cluster Formation are piggybacked on the messages for the tree Discovery component. This is a distributed implementation of creating a BFS tree. Each node, u, transmits a *tree discovery beacon,* which indicates its shortest hop-distance to the root r. The beacon contains the following fields: {*src-Id, Parent-Id, root-Id, root-seq-no, root-distance*}. If any neighbor, v of u on receiving this beacon, discovers a shortest path to the root through u, it will update its hop-distance to the root appropriately and will choose u to be its parent in the tree. The *parent-Id* field will be initially NULL, and change as the appropriate parent in the BFS tree is discovered. The root-distance field reflects the distance in hops from the root of the tree. The root-Id is used to distinguish between multiple simultaneous initiators of the Cluster Creation phase of which only one instance is allowed to proceed. The root-seq-no is used to distinguish between multiple instances of the Cluster Creation Phase initiated by the same root node at different time instants.

To create the clusters on the BFS tree, each node needs to discover its subtree size and the adjacency information of each of its children in the BFS tree as explained below. A *cluster formation message* is piggybacked onto the tree discovery beacon by each node. This has the following fields: {subtree-size, node adjacency}. The subtree size information is aggregated on the tree from the leaves to the root. This subtree size at a node u is given by $[1 + \sum_{v \in \text{Children}(u)} \text{subtree-size}(v)]$. When a node w detects that its subtree size has crossed the size parameter C_s, it initiates cluster formation on its subtree. If the entire subtree T(w) is of size less than C_s it creates a single cluster for the entire subtree. Otherwise, it will create a set of clusters by appropriately partitioning its sub trees into these clusters. This information is subsequently propagated down the child subtrees as cluster assignment messages to the relevant nodes.

The partitioning of the subtrees into clusters is implemented as specified in the graph cluster algorithm (Fig. 3). To do this, node w needs to know the adjacency information of its children in the tree. This is available as the neighborhood information, N(u) carried in the node-adjacency field of the cluster formation message from each child, u. In its subsequent cluster formation messages to its parent, node w does not include all the nodes, that it has assigned to different clusters. This is equivalent to the deletion of these nodes from the tree in the algorithm GRAPH CLUSTER and MERGEPARTIALCLUSTERS.

Let T be this rooted BFS tree, and T(v) denote the subtree of T rooted at vertex v. We use |T(v)| to denote the size of the subtree rooted at v. Let C(v) be the set of children of v in T. We assume that $|V| \geq C_s$, else we can treat the entire graph as one cluster. First, we identify a node u such that $|T(u)| \geq C_s$. For each $v \in C(u)$ we have $|T(v)| < C_s$. Let C(u) consist of *l* nodes $v_1, \ldots, v_l$. The clustering can be achieved by taking the set of subtrees $\{T(v_1), \ldots, T(v_p)\}$ where $v_1, \ldots, v_p$ are the children of u in the tree. The clusters are, by definition, disjoint and each partition consists of a set of sub-trees such that the number of all vertices in the subtrees comprising the clusters lies between 1 to C_s. Each cluster is created by adding subtrees sequentially to it until the size lies between 1 to C_s. Addition of a single subtree can increase the partition size to more than C_s. In such case, the nodes beyond C_s are taken out to be part of a new cluster.

The algorithm is implemented via post-order traversal of T. When we are visiting a vertex, we can check the size of the subtree rooted at that vertex. If the subtree has size $\geq C_s$ then we can trigger the above scheme. Once we output a set of clusters and consequently delete the vertices that belong to these clusters, we can update the size of the current subtree and continue with the post-order traversal at the parent of this vertex.

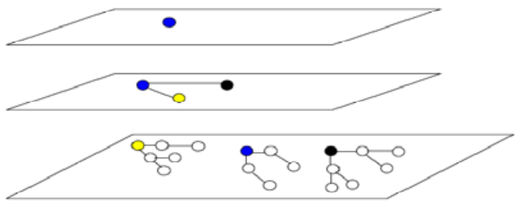

Figure 2. Hierarchical Clustering

3.2 Leader Election

In case a cluster head fails, then the children nodes have to cooperate to elect a new cluster head. The fault free node within a cluster is elected as the cluster head if the current cluster head fails. The main advantage of electing a leader using node ids is that these node ids are also exchanged for diagnosis purposes. Each fault-free node in the cluster sends its node id to all other nodes in a cluster. The node ids within a cluster are arranged in ascending order and first node is the cluster head. The decisions about the identity of the cluster head can be made locally and consistently. The assignment of the representative tasks is dynamic in that the identities of the representatives may change after each diagnosis session due to failures or repairs. Since every cluster may have one or more mobile nodes along with the static sensor nodes, the leader election procedure identifies a mobile node as the cluster head.

3.3 Testing and Hierarchical Diagnosis

Once the cluster creation phase generates a set of clusters, the diagnosis phase is invoked to diagnose the nodes with crash or value faults as specified in the fault model. The following distributed diagnosis procedure performs testing and diagnosis at every layer in the hierarchy in each cluster. Our clustering method divides the system into clusters and into a number of layers depending on the size of the system. The number of layers l_c is computed as: $l_c = \lceil \log_{C_s}(n/N_{init}) \rceil$, where N_{init} is the number of initiator nodes. A specialized number of nodes from one or more clusters, consisting of initiators of diagnosis process at the top layer in the hierarchy, initiate the testing and diagnosis process. The cluster size C_s and the number of layers l_c are two variables used to tune the clusters such that a balance is achieved between these two.

At the start of the diagnosis process, each initiator node executes the following:

1. **Forward dissemination of heartbeat messages.** Each imitator node initiates testing by sending a heartbeat initiation message top-down in the BFS tree, rooted at the initiator node, to its neighbors using the 1_hB primitive. Each cluster head down the tree, upon receiving the heartbeat message, forwards it to all its children nodes. Heartbeat messages are propagated throughout the network, and a tree spanning all fault-free nodes corresponding to each initiator is built during propagation. Once a node has propagated the message, it waits for the reply heartbeat messages from its children nodes, diagnosing as fault-free the sender of any message received. After the timeout period T_{out1}, nodes that did not reply with their heartbeat message are diagnosed as crash faulty. If the message arrives but does not match with the actual value, it is considered as value faulty. Thus, if a cluster head exists, it will record a fault event in its Status_Table irrespective of whether the node is crash or value faulty. If a node was found to be fault-free, the diagnosis information is later obtained from that fault-free node about the entire sub-tree formed by the node. The proposed algorithm assumes 100% test coverage [21-25].

2. **Backward propagation of local diagnosis information.** Each cluster head v achieves its local diagnosis independently, i.e., it knows the state of all of its neighbors, based on the heartbeat message received. Once these diagnosis information have been received, node v combines all of them as well as its own local diagnosis into a unique diagnostic message m containing the identities of all the faulty nodes adjacent to at least one fault-free node in the sub tree rooted at v, and selectively sends m backwards to its parent in the spanning tree. It is noted that the diagnosis algorithm executed at the node v captures the status of both crash and value faulty nodes. This process is repeated for each of the cluster heads in the hierarchy until the initiator nodes achieve the complete diagnosis about the nodes in the spanning tree rooted at the initiator node. It is noted that, the spanning trees formed by each of the

initiator nodes are independent of each other. In order to preserve this property, when a node moves from one cluster to another cluster, it becomes a member of the later cluster and the cluster head of the destination cluster will assign a new node-id to this migrating node and consider this node as a member of its own cluster.

3. **Exchange of the diagnosis information among initiator nodes and Global diagnosis broadcast:** Once the initiators receives diagnostic messages from all its children in the spanning tree, it combines the diagnostic information of these messages with its own diagnosis in order to obtain a global diagnosis view of the system, i.e., the identities of all the faulty nodes in the system. It is noted that when a child node does not receive a heartbeat message within the time out period T_{out2} or receives an erroneous message from its cluster head, it detects the cluster head to be faulty and elects another cluster head from the nodes in that cluster. For implementing this, the child nodes also maintain another time out period. Therefore, at any time, a fault-free nodes would act as the head of a cluster.. This property of the algorithm satisfies the dynamic fault environment in the sense that when a cluster head becomes faulty during execution of the diagnosis process, the child nodes detect this and elects another cluster head. This ensures that cluster heads and initiator nodes towards higher layer in the hierarchy except the nodes in the lowest layer contain only fault-free nodes. Since the initiator nodes keep the status of entire network, they are fault-free and thus reliable. The initiator nodes exchange the diagnosis information among themselves and therefore achieve the global diagnosis of the entire network i.e., identities of all the faulty nodes in the system. The global diagnosis is then disseminated downward in the spanning tree using a broadcast protocol.

Each active node in the network maintains the status of every other node in the network in a diagnosis session. The diagnosis session captures the events that were previously present in the network and the events that occur within a diagnosis session. A diagnosis session consists of executing the above three phases by the mobile and the active nodes of the wireless network. The nodes are subjected to any of the events such as: a new node joins, an existing nodes leaves and the nodes become faulty or fault-free. It is noted that the messages that contain only fault status of some nodes are exchanged.

4 THE CORRECTNESS PROOF AND THE COMPLEXITY ANALYSIS

In this section, we shall first prove the correctness of our diagnosis approach and then perform the complexity analysis of the algorithm in terms of diagnosis latency and message complexity. The diagnosis latency is the duration between the inception and the end of the diagnosis procedure, whereas the message complexity is defined as the total number of messages exchanged during a diagnosis session.

For a distributed self-diagnosis algorithm to be correct it should guarantee that at the end of each diagnosis session each fault-free node correctly diagnoses not only the state of all its nodes within cluster, but also that of all the nodes in the system. In the following we prove that our algorithm is correct and complete. The correctness proof is based on the following two main properties: *Partial correctness* i.e., the fault status of any node connected to the network is correctly diagnosed by at least one fault-free node and *Completeness,* i.e., local diagnostic views generated by fault-free nodes are correctly received by any other fault-free node in a finite time. The first property certifies that cluster heads know the fault status of all nodes within the cluster that are connected to the network at the end of the local testing/diagnosis phase. While, the second property ensures that after propagating the partial diagnostic views by cluster heads i.e., at the end of the dissemination of global diagnostic messages, correct diagnosis is achieved in a finite time. The first property is proved in the Lemma 2, while Lemma 3 proves the second property. Assume the network topology is time varying, and the communication graph G is the connected communication graph of the system at time t during the diagnosis session, with $t_{start} \leq t \leq t_{end.}$

Lemma 1. Once a diagnosis session has been initiated, the nodes in the hierarchy will receive the first heartbeat message in at most $T_x + (l_c\text{-}1)_*$ 1_hrB_delay time where T_x is the send initiation time of the heartbeat message and 1-hrB_delay is the one hop reliable broadcast delay within the cluster.

Proof: The initiator node generates a heartbeat message at most in time $T_{x.}$ This heartbeat message reaches the cluster heads of the next layer in the hierarchy in time T_x+1_hrB_delay. And, the leaf nodes in the spanning tree rooted at the initiator node will receive this message at the latest by $T_x+(l_c-1)_*$ 1_hrB_delay. As the leaf nodes will be the last ones to receive the message, it follows that in at most $T_x+(l_c-1)_*$1_hrB_delay, all the nodes in spanning tree will receive the heartbeat message sent by initiator nodes.

Lemma 2. If a diagnosis session has been initiated, then the fault status of each node connected to the network during the diagnosis session is correctly diagnosed by at least one fault-free node.

Proof: Assume that node u is in the network. u may be either fault-free or faulty. If node u is fault-free, irrespective of it is static or mobile, it will receive a heartbeat message in a finite time according to Lemma 1. Hence, if it receives a heartbeat message then it will report about its status to its cluster head by sending a reply heartbeat message. Since we are considering diagnosable adhoc network (i.e., the diagnosability bound in each cluster is preserved), node u will always have its cluster head in the cluster at any point of time, which will correctly diagnose its state. Now if node u is faulty, there are four scenarios. The first is that u is a static value-faulty node. In this case, the same reasoning applies, i.e, u will receive a heartbeat message and will reply by sending a reply heartbeat message carrying an erroneous result within a bounded time allowing its cluster head to diagnose its faulty state. However, if u is static crash faulty node (second scenario), then its cluster head will be able to diagnose its state once a timeout occurs. The third scenario is that node u is a dynamic value-faulty node. In this case, node u may not receive the initiation heartbeat message from its cluster head since it has moved to some other clusters. However, it will receive a heartbeat message from the cluster head for the cluster where it has joined. In this case, it will send an erroneous reply heartbeat message to its new cluster head. The new cluster head can diagnose this node value faulty node. The last scenario is that node u is a dynamic crash-faulty node, and hence, the state of u will be correctly identified once the timer in the cluster head times out.

Now, we prove the second property, i.e., all nodes are correctly diagnosed in a finite time.

Corollary 1. If a diagnosis session has been started, then the last cluster head node to transmit its local diagnostic view will do so in at most $T_x + (l_c-1)_*[1\_hb\_delay + \max(T_{out1}, T_{out2})]$.

The proof of this corollary follows trivially from that of Lemma 1 given that after receiving its first heartbeat message the node will transmit its heartbeat message to all its children nodes, which are the leaf nodes, and then waits for at most T_{out1} before starting preparation of local diagnostic message. In case any parent node fails, including the initiator nodes, the children nodes will be able to identify by timeout T_{out2}. Hence, the last cluster head to compute its local diagnostic view will do so in $T_x + (l_c-1)_*[1\_hb\_delay + \max(T_{out1}, T_{out2})]$.

Lemma 3. Let G be the graph representing the adhoc systems during a diagnosis session. If G is connected and the total number of faulty nodes is at most $\delta_c = (k_c–1)$ in a cluster, then the local diagnostic view generated by the cluster head is correctly received by any other fault-free node connected to the system during the diagnosis session in a finite time.

Proof: The graph G is connected means there is a path between every pair of nodes in the entire graph. If a cluster does not exceed its diagnosability bound δ_c, it will remain connected and its cluster head will acquire the status of its children nodes within the time out period T_{out1}, stored at the cluster head. The cluster head prepares a local diagnostic message containing the fault status of only faulty nodes in this cluster within a bounded time. The local diagnostic message is transmitted to the cluster head bottom up in the hierarchy to reach the initiator nodes in the spanning tree using the SelSend primitive.

If the number of faulty nodes in a cluster exceeds its diagnosability bound, the cluster heads in the next higher layer in the hierarchy determine this within a time out period of T_{out1}. If a cluster head becomes faulty, it will be detected by its children nodes that elect another node in the cluster as cluster head within a time out T_{out2} maintained at the children nodes. The diagnosis tasks executed at the top layer in the hierarchy exchange the local diagnostic messages. It is noted that if an initiator node is faulty, it will be detected by its fault-free children node that elect another initiator dynamically in the same diagnosis session. Therefore, the initiator nodes are always fault-free and achieve the global diagnosis within a bounded time. Once, the global diagnosis message is prepared by an initiator node, it is broadcast to all the nodes in its spanning tree. It is easy to observe that the complete diagnostic message sent by respective initiators reach all the leaf nodes within a bounded time according to Lemma 1. This shows that all the nodes get the global diagnosis information within a finite time.

Theorem 1: Let G be the graph representing the system during a diagnosis session. If G is connected and the total number of faulty nodes in each cluster is at most $\delta_c = k_c - 1$, then every fault-free node connected to the system during a diagnosis session correctly diagnoses the state of all the nodes in the system in a finite time.

Proof: The proof of Theorem 1 follows trivially from lemma 2 and lemma 3. In fact, once all the nodes have been diagnosed at the end of the diagnosis session within a cluster, the local diagnostic view from the cluster heads is sent bottom up towards the initiator nodes. The initiator nodes then exchange the local diagnosis messages and prepare the global diagnosis message by combining the local diagnosis messages within a finite time. This global diagnosis message is then sent to every node in the respective spanning trees of each of the initiator nodes. This message is correctly received by all the fault-free nodes connected to the system. Given that G_t is connected and $k_c \leq d_{min}$, where d_{min} is the minimum node degree in G_t, it follows that each node is adjacent to at least one fault-free neighbor, and hence, it is correctly diagnosed by at least one fault-free node.

Now, we evaluate the performance of our algorithm in the worst-case by analyzing its time and message complexity. The time complexity refers to the duration of a diagnosis session whereas the message complexity refers to the total number of 1-hop reliable broadcast messages transmitted during a diagnosis session. To determine the performance improvement achieved by clustering the nodes, we look at the time complexity and message count of the clustering as compared with single level or non-clustering method. The following theorems evaluate the time and message complexities of the proposed diagnosis algorithm.

Theorem 2: If $l_c < \Delta_{nc}$, the diameter of the network without clustering, then the proposed hierarchical clustered diagnosis algorithm has the time-complexity of $O(l_c(T_x + T_f + \max(T_{out1}, T_{out2})) + T_{xcg})$.

Proof: Any initiator node originates the initiation heartbeat message in time T_x. This means that in time at most $l_c (T_x + T_f)$, the furthest node in the network receives the heartbeat message. The nodes in the lowest level in the network respond to this heartbeat message by sending response heartbeat messages bottom up in the spanning tree in at most time $(T_x + l_c(T_x + T_f))$. Each cluster head receives the diagnosis information and prepares the local diagnostic message by aggregating this with the diagnosis information received from the other nodes in the cluster latest by the time: $(l_{c\,+}\,1)(T_x + T_f)+ \max(T_{out1}, T_{out2})$. The time for the initiator nodes to receive the local diagnostic messages from the different cluster heads is at most $2(l_c-1)(T_x + T_f)+ (l_c-1)\max(T_{out1}, T_{out2})$. The initiator nodes prepare the global diagnostic message by exchanging these local diagnostic messages since the start of the diagnosis session in at most time: $2(l_c - 1)(T_x + T_f)+ (l_c-1)\max(T_{out1}, T_{out2}) + T_{xcg}$ where T_{xcg} is the time needed to exchange the diagnosis information by all the initiators at the top layer in the hierarchy in order to form the global diagnosis message. This global diagnosis message is then sent downwards in the spanning tree. The last non-leaf node in the hierarchy receives the global diagnosis information at the latest by the time $3(l_c - 1)(T_x + T_f)+ (l_c-1)\max(T_{out1}, T_{out2}) + T_{xcg}$. It is noted that the leaf nodes are passive wireless devices and do not keep any diagnosis information. This follows by observing that this is the worst-case time at which any fault free node receives the global diagnosis information once the diagnosis session is initiated. Therefore, the worst case time complexity of the proposed algorithm is $O(l_c(T_x + T_f + \max(T_{out1}, T_{out2})) + T_{xcg})$.

Theorem 3: The message complexity of the hierarchical cluster based diagnosis approach is $O(n_c C_s)$.

Proof: The worst-case scenario is when all the nodes are fault-free. In this case, one heartbeat message goes from each initiator to every other node in the network for a total of $(n_c C_s - N_{init})$ messages. The other nodes send one reply heartbeat message each to their respective cluster heads. Each cluster head sends one local diagnostic message upwards in the hierarchy. These two operations lead to $(n_c-1)(C_s+1)$ messages. Finally, the initiators broadcast the global diagnosis message in the tree rooted at the initiators. The total number of global diagnosis messages will be (n_c-1) where n_c is the number of clusters in the spanning tree rooted at the initiators. Hence, the total number of messages exchanged can be estimated as $(n_cC_s + (n_c-1)(C_s+1)+ (n_c-1)-N_{init})$. Here, we ignore the number of local diagnosis messages exchanged among the initiator nodes to formulate the global diagnosis message. In the order notation, this can be expressed as $O(n_cC_s)$.

Theorem 4: The percentage of the diagnosability improvement of the proposed clustering approach over the non-clustering approach with cluster connectivity k_c is $((\alpha -\beta)/\alpha)*100$ where α and β are diagnosability of clustering and nonclustering approach respectively given by $\alpha= n_c(k_c-1)$ and $\beta=k(G)-1$, $k(G)$ being the connectivity of the network.

Proof: It is observed from the clustering method that union of set of all clusters produced by the clustering algorithm covers all the nodes in the network. The node or cluster connectivity of each cluster is k_c. During the diagnostic phase of the cluster, the cluster heads in each cluster initiate test procedures and all clusters conduct their tests simultaneously. The results are exchanged among the initiators at the top layer in the hierarchy. It can be observed that the diagnosability of the non-clustering algorithms is given by $\beta=k(G)-1$.
The cluster-based diagnosis algorithm proposed here allows the diagnosis process to proceed independently in each of the clusters. Clustering the whole system into almost equal size clusters allows each cluster to diagnose up to a number of $(k_c – 1)$ faults and the local diagnosability of each cluster is (k_c-1). The global diagnosability of the clustering method is $\alpha = n_c(k_c – 1)$. The percentage improvement of the diagnosability of clustering approach is $((\alpha - \beta)/\alpha)*100$. Thus, follows the theorem.

5. SIMULATION RESULTS

A simulator has been developed using language C++ for evaluating the diagnosis latency and message complexity of the proposed distributed diagnosis approach. Graphs were randomly generated for a given n and k. The links were randomly introduced such that every node has at least k neighbors to ensure k-connectivity. The connectivity of this network is then found by running the Ford_Fulkerson algorithm [9]. When the connectivity of the network is below k, n links are randomly introduced into the network. To investigate the suitability of the networks, five different networks were generated for every value of (n, k). Simulations were performed on arbitrary network topologies of sizes 16, 32, 64, 128, 256 and 512 nodes of cluster size 4, 8, and 16 using discrete event simulation techniques. The arbitrary network topologies were generated with connectivity value $k(G) = 3$.

The tree discovery beacons are transmitted by each node once every π units of time, over the duration of the cluster creation phase. The period π is chosen depending on the average connectivity of a node in the network. The average number of beacons sent by a node for the entire cluster creation phase is approximately bounded by the diameter of the network. As the cluster creation phase ends, only the cluster information needs to be retained by the clusters. The BFS tree does not need to be maintained any further.

We simulated the operations of our clustering scheme assuming that the number of mobile nodes in a cluster is at least one and the remaining nodes are static nodes keeping their state holding time of 60 plus a random number generated by poison process with mean equal to 1 and 200 units to represent the status of mobile and static nodes in the system. This models the dynamic nature of the proposed approach. When an event occurs on a node, the time at which the next event occurs on the same node is the state holding time for the current state plus an additional time as given by the Poisson process. If a failure event is not possible to occur because the number of failed nodes in a cluster is greater than (k_c-1), then the failure event is rescheduled to a later time again according to a Poisson process. In all simulations, the minimum state holding times is different for both the static sensor and mobile nodes. This reflects the situation that the mobile nodes are more prone to faults than static sensor nodes. The mobile nodes are subjected to three types of crash faults: out of range, physically damaged and low battery power. Therefore, the state holding time is computed assuming mean 1 unit whereas sensor nodes will not have mobility and therefore the state holding time was chosen to be 200 unit.

The time spent due to initial clustering of the BFS tree is not taken into account. In the experiments reported here, we choose the average time gap between successive initiations of diagnosis sessions by an initiator node to be 60 units. The send initiation time T_x, diagnosis task execution time, propagation time were assumed to follow an uniform distribution between lower and upper limit as follows:

T_x: 0.002 units

1-hB-delay: 0.008 to 0.08 units

Diagnosis Task Execution Time: 0.01 to 0.05 units

Clustering Task execution Time: 0.01 units

Diagnosis sessions: 1000 simulation run

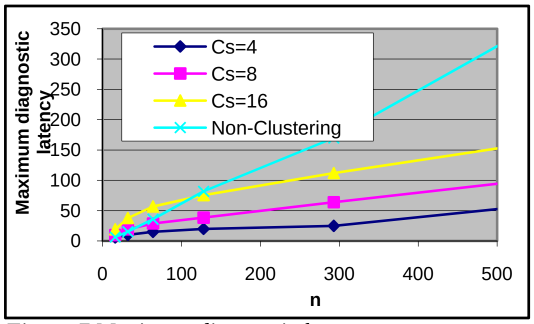

Figure.7 Maximum diagnostic latency vs. n

5.1 Diagnosis Latency Vs Number of Nodes (Clustering & Non-Clustering with varying cluster size)

It is observed that, if we increase the number of nodes, the diagnosis latency does not increase much. This is due to clustering the system graph into almost equal sized clusters and executing the diagnosis process in each cluster in parallel. In fact, the diagnosis latency depends on the number of layers in the hierarchy, cluster size, and message size. Figure 7 and Figure 8 compares the diagnosis latency of clustering method over non-clustering method for different number of nodes 16, 32, 64, 128, 256 and 512 with cluster size 4, 8, and 16.

Another observation in the result is that the diagnosis latency increases lenearly but slowly as the number of nodes increases keeping the cluster size fixed. This is due to the fact that, since the diagnosis is parallel in each cluster, increasing the network size does not affect much on the diagnosis time. Because, the diagnosis time depends on the cluster size and the number of layers in the hierarchy. When the network size is increased and cluster size is constant, the diagnosis time varies linearly and slowly due to the fact that, increase in number of nodes results in increase in number of layers if cluster size is fixed. It is noted that, if cluster size is fixed, the number of initiators are not changed unless more than one cluster is considered as number of initiators. Figure shows this observation.

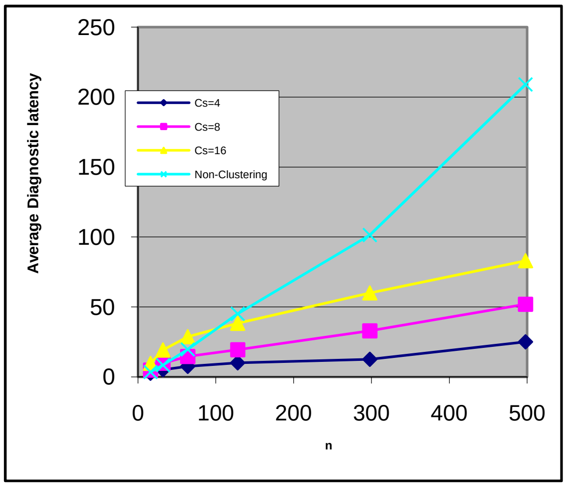

Figure. 8 Average diagnostic latency vs. n

5.2 Message Complexity Vs No. of Nodes (Clustering & Non-Clustering with varying cluster size)

Figure 9 compares the number of messages required by the clustering method versus the number required for the non-clustering algorithm. The communication complexity of our algorithm depends on number of messages and the message size. The number of messages in clustering scheme is reduced due to the limited number of initiators which is less than C_s (cluster size at the highest layer in the hierarchy) as compared to conventional non-clustering method which assumes all nodes to become an initiator in the diagnosis process as a result of which the number of messages are of O(n.e) where n is the number of nodes and e is the number of links in the network. However, the number of messages generated by the diagnosis process using clustering is $O(n_cC_s)$ due to spanning tree formed by a limited number of initiators. Our diagnosis approach allows every node to acquire a global diagnostic view about the entire system using a spanning tree with the exchange of minimum number of messages as compared to any non-clustering method that uses flooding.

The diagnostic latency and message complexity increases with increase in the cluster size. This is because, as the cluster size increases, the diameter of the network also increases. We show the diagnostic latency and message complexity for the number of nodes such as 16. 32, 64, 128, 256 and 512 node in an arbitrary network topology of different cluster sizes such as 4, 8, and 16. In general, it is desirable to have clusters of low diameter so that maximum parallelism can be achieved.

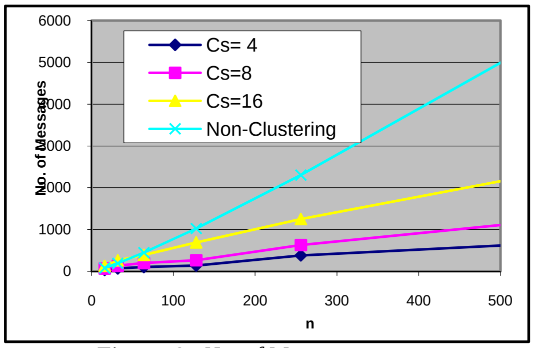

Figure. 9. No. of Messages vs. n

5.3 Diagnosability improvement vs. number of nodes

In this section, we show that there is an improvement in the diagnosability of our algorithm assuming five initiator nodes. The result shows that, the diagnosability increases with increase in transmission radius of individual clusters. Because, the increase in transmission range increases the connectivity of each cluster. We show the diagnosability improvement for 16, 32, 64, 128, 256 and 512 nodes in an arbitrary network topology with average cluster connectivity $k_c = 3$ with cluster size of 4 for clustering method. The results are compared with non-clustering method considering the connectivity value k(G) = 3. Figure 10 shows the percentage of diagnosability improvement of the clustering over the non-clustering approach versus the number of nodes in the network.

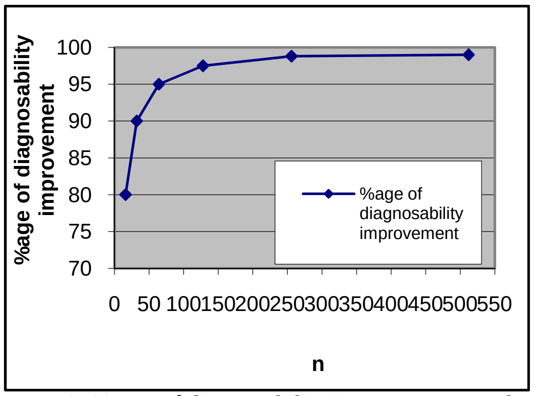

Figure.10. % age of diagnosability improvement vs. the number of nodes

6. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a hierarchical clustering method for large-scale self-diagnosable wireless distributed systems exploiting the topological properties of these classes of distributed systems. The approach was analyzed and validated through exhaustive simulation. The analysis shows that the diagnosis algorithm has diagnosis latency and message complexity of $O(l_c(T_x+T_f+\max(T_{out1},T_{out2}))+T_{xcg})$ and $O(n_cC_s)$ respectively. The proposed clustering approach outperforms the non-clustering method in terms of diagnosis overhead. It is assumed that there is an upper limit to the state holding time. The multi-cluster approach is shown to provide better performance in terms of reduced diagnosis time, message complexity and diagnosability while increasing the admissible network size by an order of magnitude. The proposed distributed diagnosis approach presented in this paper has the following advantages: (i) improved scalability, (ii) reduced message overhead, (iii) reduced diagnosis latency and (iv) improved diagnosability.

The proposed diagnosis approach can be extended in many possible ways providing the solutions to several problems such as adding more heterogeneity to the existing systems. Our future work includes the investigation of roving diagnosis where some of the clusters may run the application while other clusters in the system may execute the diagnosis algorithm simultaneously. We are interested to know the kind of distribution for interleaved/simultaneous execution of applications and diagnosis processes in a network. Any heterogeneous hierarchically organized distributed system such as the Internet can use our diagnosis approach to acquire a global diagnostic view any time and anywhere.

REFERENCES

[1] Stefano Chessa and Paolo Santi, "Comaprisom-based system level fault diagnosis in ad hoc networks, Proc. SRDS' 01, New Orleans, LA (October-2002)

[2] Stefano Chessa and Paolo Santi, "Crash faults identification in wireless sensor networks ", Computer Communications, 00 (2002) 000-000

[3] M.Barborak, M.Malek, A.Dahbura, " The Consensus Problem in Fault-Tolerant Computing," ACM Computing Survey, vol.

25, No.2, pp. 171-220, June 1993.

[4] Barborak, M. Malek, M, " Partitioning for Efficient Consensus", In proc. of the IEEE 26th Hawaii International Conference on System Technology Sciences, 1993, pp. 438-446.

[5] O.Benkahla, C.Aktouf & C.Robach, "System-Diagnosis of Cluster-Based Parallel Architectures", In proceedings of IEEE conference on Parallel and Distributed Processing'96, 1996 pp. 305-309.

[6] Duarte E.P.Jr. and Nanya T., " A Hierarchical Adaptive Distributed System-Level Diagnosis Algorithm", IEEE Trans. Computers, vol.47, pp.34-45, Jan. 1998.

[7] Duarte E.P Jr. and Nanya T. "Multi-Cluster Adaptive Distributed System-Level Diagnosis Algorithms, " IEICE Technical Report FTS 95-73, 1995.

[8] A. Brawerman, E.P.Duarte Jr. "An Isochronous Testing Strategy for Hierachical Adaptive Distributed System-Level Diagnosis", Journal of Electronic Testing: Theory and Applications 17, 185-195, 2001.

[9] Su M.S., Thulasiraman K. and Das A., "A Scalable On-line Multilevel Distributed Network Fault Detection/Monitoring System based on the SNMP protocol", IEEE GlobeComm 2002, November 2002.

[10] Su M.S., Thulasiraman K., and Goel V., "The Multi-Level Paradigm for Distributed Fault Detection in Network with Unreliable Processors", ISCAS'03, Proc. of the 2003 Intl symposium on Circuits and Systems, vol.3, pp 862-865, May 2003.

[11] Paritosh Chandrapal and Padam Kumar, "A Scalable Multi-level Distributed System-Level Diagnosis", Proc. of ICDCIT 2005, LNCS 3816, pp. 192-202, 2005, Springer-Verlag Berlin Heidelberg 2005.

[12] G.Jeon and Y.Cho, "A Partitioning Method for Efficient System-level Diagnosis", The Journal of Systems and Software 63 (2002) 1-16.

[13] Sailesh. Chutani,"A Framework For Fault diagnosis of Networks", Ph.D Dissertation, University of North Carolina, Chapel Hill, USA, 1996.

[14] M.Stahl, R.Buskens and R.P.Bianchini, "On-Line Diagnosis in General Topology Networks", Workshop on Fault Tolerant Parallel and Distributed Systems, (July 1992).

[15] Rangarajan. S, Dahbura. A. T, Ziegler, E.A., " A distributed system-level diagnosis algorithm for arbitrary network topologies", IEEE Trans. Computers. 44(2), 312-334,1995.

[16] K-F. Ssu, Ch.H.Chou, H.C.Jian, W-T Hu, "Detection and diagnosis of data inconsistency failures in wireless sensor networks", Computer Networks, 50, (2006) 1247-1260.

[17] D. Blough and H. W. Brown, The Broadcast comparison model for on-line fault diagnosis in multicomputer systems: Theory and implementation, IEEE Trans. Computers. Vol. 48, no. 5, pp. 470-493, May 1999.

[18] Arun Subbiah, and D. M. Blough, "Distributed Diagnosis in Dynamic Fault Environments", IEEE Trans on PDS, Vol 15, No. 5, May 2004.

[19] J.Maeng and M.Malek, "A Comparison Connection Assignment for Self diagnosis of Multi-processor Systems," Digest 11th Int'l Symp. Fault-Tolerant Computing, pp. 173-175 -1981.

[20] E. P. Duarte Jr., A. Brawerman, and L. C. P. Albini, "An Algorithm for Distributed Hierarchical Diagnosis of Dynamic Fault and Repair Events," *Proc. IEEE ICPADS'00*, pp. 299-306., 2000.

[21] L. C. E. D Bona & E. P. Duarte Jr. "A flexible Approach for Defining Distributed Dependable Tests in SNMP-Based Network Management Systems," in Journal of Electronic Testing: Theory and Applications 20, 447-454, 2004.

[22] A. Bagchi and S. L. Hakimi, "An Optimal Algorithm for Distributed System Level

Diagnosis," Proc. Digest of the 21st Int'l Symp. Fault Tolerant Computing, pp.214-221, 1991.

[23] Sailesh Chutani and Henri J. Nussbaumer, " On the Distributed Fault Diagnosis of Computer Network", Proceedings of IEEE Symposium on Computers and Communications, 1995, 27-29 June 1995, Pages: 71-77.

[24] Xuanwen Luo, Ming Dong and Yinlun Huang, "On Distributed Fault-Tolerant Detection in Wireless Sensor Networks", IEEE Trans. on Computers, vol. 55, no. 1, January 2006.

[25] Sampath Rangarajan and Don Fussell "A Probabillistic Method for Fault Diagnosis of Multiprocessor Systems", Proceedings of the 18th International Symposium on Fault-Tolerant Computing, Tokyo, Japan, pp. 278-283, June 1988.

G	(V,E) be the communication graph of the wireless network
G_t	The topology of the communication graph at time t
θ	Represents the maximum deviation between estimated and observed remaining energy/diagnostic value
n	Number of nodes in the system
N_{init}	The number of initiator nodes;
t_{start} and t_{end}	The starting and finishing times of the diagnosis session.
Δ_{nc}	The diameter of the network without clustering;
l_c	The number of layers in the hierarchy or depth of the spanning tree
C_s	The Cluster size i.e., the number of nodes in a cluster;
n_c	Number of clusters;
T_f	An upper bound to the time needed to propagate a diagnostic message and accounts for the time needed to aggregate the diagnostic information and to send it to the parent node or to the children in the spanning tree.
d_{min}	The minimum of node degrees;
T_x	Send initiation time;
1_hrB_delay	One hop reliable broadcast delay;
T_{out1}	The time out value of the timer maintained by the initiators and cluster heads;
T_{out2}	The time out value of the timer maintained by the children node;
k_c	The connectivity of the cluster;
k(G)	The connectivity of the entire network;
δ_c	Diagnosability of the cluster;
α	Diagnosabilty of system using clustering;
β	Diagnosability of system without clustering;
Init-HB-msg	Init-HB-msg: Initiator heartbeat message;
Reply-HB-msg	Reply-HB-msg: reply heartbeat message;
Loc-diag-msg	local diagnostic message;
Global-diag-msg	global-diagnostic-message;
N(u)	The Neighborhood of a vertex u, is the set of vertices that have an edge to the vertex u in the graph.
R	The common transmission radius of all the wireless nodes;
d(u,v) :	The distance or proximity between two nodes u and v;

Figure 1. The different notations and their meaning

Each node in the arbitrary graph executes the following.

Proc. 1: GRAPHCLUSTER(G,C_s)

T: A BFS tree of graph G
root(T): Root of the BFS tree
T(x): subtree of T, rooted at vertex x.

|T(x)| denote the size of the subtree rooted at x.
ClusterSet: The set of clusters created by the algorithm
RemChildren: Variable used to store the set of remaining children (i.e., that has not been deleted) that are yet to be processed at a vertex.
C_s is the cluster size;
PartialClusterSet: Set of temporary clusters those have size < C_s.
Children(u): the set of children of u in T; u is the parent node;
N(u) is the set of adjacent or neighbor nodes of u.
Empty set is denoted by ϕ;
Temp is a temporary set;
Tree-discovery-beacon: The beacon contains the following fields: {*Src-Id, Parent-Id, Root-Id, Root-seq-no, Root-distance*}
cluster-form-msg: {subtree-size, node adjacency}
Root-distance: the field reflects the distance in hops from the root of the tree.
Root-Id: is used to distinguish between multiple simultaneous initiators of the cluster creation phase of which only one instance is allowed to proceed. The node having lowest id can initiate.
Root-seq-no: is used to distinguish between multiple instances of the cluster creation phase initiated by the same root node at different time instants.
{Root-Id, Root-seq-no}: is used to uniquely identify cluster creation phase instance;
Parent-Id = NULL

```
Step 1: T = BFS tree of G; ClusterSet = $\phi$
Step 2: For each node u do
            transmits the tree discovery beacon to all the nodes; this indicates it's shortest
            hop-distance to the root r.
            N(u) receives the tree discovery beacon;
            If [shortest-path (N(u) and root)] == "FOUND"
                update the hop-distance to the root
                update the root-distance;
                Change u = parent of $v \in T(u)$;
            Endif
        Endfor

Step 3: Piggyback the cluster formation messages on the tree discovery beacon
        If (T(u) < $C_s$) then
            Create one single cluster for the entire subtree;
            ClusterSet = ClusterSet $\cup$ T(u);
        Else
            Continue until a single partial cluster remains in the tree;
        endif;
```

Step 4: Create Clusters using the post-order traversal of BFS tree T: Create the set of disjoint clusters or subtrees $\{T(v_1), \ldots, T(v_p)\}$ where $v_1, \ldots, v_p$ are the children of u in the tree, from subtree T(u) such that the number of all vertices in the subtrees comprising the clusters lies between 1 to C_s.

```
4.1 For $u \in G$, in post-order traversal of T do   // Left-Right-Root
      discover the |T(u)| and N(u) of each of its children $v \in$ Children(u) in the BFS tree.
      Aggregate subtree size information on the tree from the leaves to the root.
      $|T(u)| = 1 + \Sigma_{v \in Children(u)}$ subtree-size(v)
         Let Children(u) = $\{v_1, \ldots, v_l\}$
         Let |Children(u)| = l
         // Initiate cluster formation on its subtree;
4.5   if ($|T(u)| \geq 1$) and $|T(v)| < C_s \; \forall v \in$ Children(u)) then
            Receive Cluster-form-msg from each child node $v_i$ from Children(u)
            Find the Cluster-form-msg.node-adjacency information of Children(u) in the tree;
         PartialClusterSet = $\phi$
         RemChildren = Children(u)
```

```
4.6    While $\exists v \in$ RemChildren do
           TempCluster = T(v);
           Remove v from RemChildren;
4.7        While (|TempCluster|< $C_s$) $\wedge$ ($\exists x \in$ RemChildren, s.t x has an edge to w $\in$
                                                   Children(u) $\cap$ TempCluster) do
4.8            TempCluster = TempCluster $\cup$ T(x)  // inclusion of a children node into cluster
4.9            Remove x from UnpChildren
4.10       endwhile
4.11       if (|TempCluster|< $C_s$) then
                   PartialClusterSet = PartialClusterSet $\cup$ TempCluster;
               //Add each subtree T(v) to Cluster(u) sequentially until 1≤ |Cluster(u)|≤ $C_s$.
4.12       Remove all subtrees from the TempCluster;
4.13     endif
4.14     endwhile
4.15     MERGEPARTIALCLUSTERS(u, $C_s$, PartialClusterSet, ClusterSet)
4.16     if (Children(u) = $\phi$) $\wedge$ (u has been assigned to some cluster) then
         Remove u from the tree  // T(u) is already processed and put into some cluster
         endif
         endif
4.17   Propagate the cluster assignment information down the child subtrees T(u).
4.18   Endfor
4.19   if PartialClusterSet $\neq \phi$ then
4.20     {|PartialClusterSet| = L; Let P $\in$ PartialClusterSet}
4.21     ClusterSet = ClusterSet $\cup$ {P U {root(T)}}
4.22   endif
```

Figure 3. Proc 1 for Creating clusters of size 1-C_s

A set of nodes forms a cluster till the cluster size remains less than C_s. Once cluster size exceeds C_s, it is included in TempCluster. A set of clusters are included in the PartialClusterSet.

Proc. 2: DISTRIBUTED-DIAG-ADHOC()

We use the following notation in the diagnosis algorithm:

f: Number of faulty nodes;
ff: Number of fault-free nodes;
δ_c : diagnosability of the cluster;
Init-HB-msg : Initiator heartbeat message;
Reply-HB-msg: reply heartbeat message;
Loc-diag-msg: local diagnostic message;
Global-diag-msg: global-diagnostic-message;
Init-Node-Id: initiator node identity;
Clust-head-Id: Cluster head identity;
Leaf-node: the nodes without any children node;
T(x): Subtree of T, rooted at vertex x;
Status_Table[Node-Id]: status of all nodes in the network maintained at every node
N(u): Neighbor of node u;
CLUSTER(u): All the nodes in a cluster with cluster head u
Terminate = False;   // boolean variable used to terminate the protocol
Start_Diagnosis = True; // Boolean variable used to initiate the protocol
FF = ϕ;   // Set of neighbor nodes diagnosed as fault-free
F = ϕ; // Set of nodes diagnosed as faulty
D = ϕ; // Set of nodes which sent their diagnosis to the initiator
Fu = set of the identifiers of the nodes currently diagnosed as faulty by node u.

```
Parent = -1;        // variable Parent is used to detect the sender of the first heartbeat message received
                    by v, which is the parent of u in the spanning tree;
ToSend = False; // Boolean variable ToSend is True only if node v received at least one heartbeat
                    message but still has to send its heartbeat message
C(v) = $\phi$;            // Set of the children of v in the spanning tree
```

Each initiator node executes the following.

```
Repeat
   Choice {
   [] Start_Diagnosis:
                    1-hB(Init-HB-msg, Initnode-Id);
                     Set_Timeout($T_{out1}$, $T_{out2}$);
                     Start_Diagnosis = false;
   [] receive (u, Initnode-Id):
                    // the sender of the heartbeat message is diagnosed as fault-free
                    Status_Table[u] = fault-free;
                    FF = FF U {u};
                    // the nodes that replied an erroneous message are diagnosed as faulty
                    F = N(Initnode-Id) – FF;
                    If (F = N(Initnode-Id)) Then
                              // if all of its neighbors are faulty then the diagnosis is complete
                              Terminate = True;
                    Endif;
   [] Timeout:
                    // the nodes that did not reply within time $T_{out1}$ are diagnosed as faulty
                    F = N(Initnode-Id) – FF;
                    If (F = N(Initnode-Id)) Then
                              // if all of its neighbors are faulty then the diagnosis is complete
                              Terminate = True;
                    Endif;
                    If (children(Init-id) times out)
                              Initiate elect();  // Elects a New Initiator
                    Update entry in Status_Table[u];
    [] receive-local-diag-msg(u, Fu):
                    // Upon receiving a local diagnostic message local-diag-msg, the diagnosis contained
                   in  local-diag-msg is used to extend the initiators diagnosis
                    F = F U Fu;
                    D = D U {u};
                    If (D = N(Initnode-Id) – F) Then
                          // If all fault-free neighbors sent their diagnosis, the initiator’s diagnosis is
                       // complete and is sent to all the fault-free nodes in the network using a
                       // broadcast protocol
                              1_hB(Initnode-Id, F);
                              Terminate = True;
                Endif;
                 Update entry in Status_Table[u];
    [] global-diag-Intiators(local-diag-msg, Initnode-Id):
                              // All initiator nodes exchange local diagnostic message among themselves to
                           prepare global diagnosis message
                    1-hB(local-diag-msg, Initnode-Id);
                    Initiator node prepares a global diagnostic message;

    [] cluster-diagnosability(children(Initnode-Id), Initnode-Id):
                    If ($f < \delta$) then
                    If (a cluster-head is not present) then
                    Elect()    // Elect a cluster head from Fault-free members;
                Else
                    Continue
                Endif
        Else
        This implies that its cluster has exceeded the diagnosability bound, and repairs or
            reconfiguration is necessary. The cluster does not have a head and merge the fault-
            free nodes into another cluster.
        Endif

}
Until Terminate;
```

Any arbitrary active node (cluster head) v executes the following:

```
 Node v:
 Repeat
  Choice {
 [] receive-HB(u,w):
                // the sender of the heartbeat message is diagnosed as fault-free
                Status_Table[u] = fault-free;
              FF = FF U {u};
                If (Parent = -1) then
                    // the first heartbeat message is received
                    // the sender of this message is the v's parent in the spanning tree
                    Parent = u;
                    ToSend = true;
                Else
                  If v = w then
                  C(v) = C(v) U {u}; // node u is a child of v in the spanning tree
                Endif;

  [] ToSend:
         1_hB(v, Parent); // node v sends its heartbeat message
         Set_Timeout($T_{out1,}$ $T_{out2}$);
         ToSend = False;
  [] TimeOut:
        // nodes that did not reply within time $T_{out1}$are diagnosed as faulty
         F = N(v) – FF;
        If (C(v) =  $\phi$) Then
                // if node v has no fault-free children it sends its diagnosis to the parent
              SelSend(Parent, (V, F));
         Endif;
         Update entry in Status_Table[u];

  [] Receive-HB(u, v):
        // the nodes that replied an erroneous heartbeat message are diagnosed as faulty
                F = N(v) – FF;
                If (C(v) = $\phi$) Then
                    // if node v has no fault-free children it sends its diagnosis to the parent
              SelSend(Parent, (V,F));
                Endif;

 [] Cluster-Head-Timeout:
        // the cluster heads that did not reply within $T_{out2}$ are diagnosed as faulty
        // invoke election to elect a new cluster head
```

```
[] Receive-local-diag-msg(u, Fu):
    // upon receiving a local diagnostic message local-diag-msg, the diagnosis contained in
        local-diag-msg is used to extend the diagnosis of node v
          F = F U Fu;
        D = D U {u};
      If (D = C(v)) Then
            // If all the children in the spanning tree sent their diagnosis, the diagnosis of v is sent
            to its parent in the tree using selective send
            SelSend(Parent, (V,F));
      Endif;

[] receive-global-diag-msg(u, (Initiator, $F_{Initiator}$)):
      F = $F_{Initiator}$;
      // the global diagnosis message global-diag-msg from the Initiator is propagated to the
       children
      For Each u $\in$ C(v) do
            SelSend(u, (Initiator, $F_{Initiator}$));
            // the protocol for node v terminates when the complete diagnosis sent by the
          // initiator is received and is propagated downward in the tree
            Terminate = true;
          update-status(Node-Id):

[] leave-cluster(u, v):
    // If any cluster exceeds the diagnosability bound then repairs or reconfiguration is necessary.
      At each layer, the diagnosis is among fault-free cluster heads, and therefore, testing is not
        required. If all the mobile nodes from a cluster move to some other cluster, the former
      cluster may not have a cluster head.

      If ($f < \delta$) then
       If (a cluster-head is not present) then
            Elect a cluster head from fault-free mobile nodes using Elect();
         Else
            Continue;
         Endif
       Else
            MergePARTIALCLUSTERS (u, $C_s$, P, ClusterSet);

      // This implies that its cluster has exceeded the diagnosability bound, and repairs or
       reconfiguration is necessary. The cluster does not have a head and merge the nodes
       into another cluster.
     Endif
[] join-cluster(u, v);
      // If a new node joins into a cluster, the cluster size increases
      If  (CLUSTER(v) > $C_s$) then
            Call SPLIT-CLUSTERS(u, $C_s$, P, ClusterSet);
      Endif
}
Until_Terminate;
Any static node w executes
 Node w:
 Repeat
   Choice {
   [] receive-HB(w,u):
            If (Parent = -1) then
                  // the first heartbeat message is received
                  // the sender of this message is the w's parent in the spanning tree
```

```
                        Parent = u;
                      ToSend = true;
                  Endif;
   [] ToSend:
         1_hB(w, Parent); // node w sends its reply heartbeat message to node u.
        ToSend = false;

  [] join-cluster(w,u);
        // If a new node joins into a cluster, the cluster size increases
        If  (|CLUSTER(u)| > $C_s$) then
                Call SPLIT-CLUSTERS(u, $C_s$, P, ClusterSet);
        Endif
  }
Until_Terminate;
```

The diagnostic algorithm executing at the initiator nodes terminate after sending the global diagnostic message. The diagnostic algorithm executing at the cluster heads and leaf nodes terminate when they receive the global diagnostic message.

Figure. 4. The Distributed Diagnosis algorithm

The following procedure MergePARTIALCLUSTERS merges the partial clusters using subtrees;

Proc. 3: MergePARTIALCLUSTERS (u, C_s, P, ClusterSet)

```
1:  Temp = $\phi$
2:  While ($P \neq \phi$) do
3:      Pick an arbitrary partial cluster p from P
4:      Temp = Temp $\cup$ p; Remove p from P
5:      if (|Temp| $\leq C_s$) then
6:          ClusterSet = ClusterSet $\cup$ {Temp $\cup$ {u}}
7:          Remove all subtrees in Temp; Temp = $\phi$
8:      endif
9:  end while
```

Figure 5. Proc 3 for Merging two clusters into a single cluster of size 1-C_s

The following procedure **SPLIT-CLUSTERS** splits the larger cluster whose size is greater than C_s into clusters using subtrees of cluster size between 1 and C_s.

Proc. 4: SPLIT-CLUSTERS(u, C_s, Q, ClusterSet)

```
1:  if (|ClusterSet| $\geq C_s$) then
2:      Form a subtree TempCluster such that |TempCluster $\cup$ {u}}| = $C_s$;
3:      {Let Q $\in$ Remaining nodes of the ClusterSet}
4:      PartialClusterSet = Q $\cup$ TempCluster;
5:      Remove all subtrees in TempCluster;
6       Root(T) = PartialClusterSet $\cup$ Root(T);
7:  endif
```

Figure 6. Proc 4 for splitting a cluster into two clusters of size 1-C_s