Inter-Process Correlation Model based Hybrid Framework for Fault Diagnosis in Wireless Sensor Networks

Amna Zafar[1*], Ali Hammad Akbar[1] and Beenish Ayesha Akram[1]
[1] Department of Computer Science and Engineering, University of Engineering and Technology Lahore, Pakistan
[e-mail: amnazafar@uet.edu.pk]
[e-mail: ahakbar@uet.edu.pk]
[e-mail: beenish.ayesha.akram@uet.edu.pk]
* Corresponding author: Amna Zafar

Received April 6, 2018; revised June 30, 2018; accepted September 2, 2018; published February 28, 2019

Abstract

Soft faults are inherent in wireless sensor networks (WSNs) due to external and internal errors. The failure of processes in a protocol stack are caused by errors on various layers. In this work, impact of errors and channel misbehavior on process execution is investigated to provide an error classification mechanism. Considering implementation of WSN protocol stack, inter-process correlations of stacked and peer layer processes are modeled. The proposed model is realized through local and global decision trees for fault diagnosis. A hybrid framework is proposed to implement local decision tree on sensor nodes and global decision tree on diagnostic cluster head. Local decision tree is employed to diagnose critical failures due to errors in stacked processes at node level. Global decision tree, diagnoses critical failures due to errors in peer layer processes at network level. The proposed model has been analyzed using fault tree analysis. The framework implementation has been done in Castalia. Simulation results validate the inter-process correlation model-based fault diagnosis. The hybrid framework distributes processing load on sensor nodes and diagnostic cluster head in a decentralized way, reducing communication overhead.

Keywords: Correlation, process, error classification, fault diagnosis, wireless sensor network

http://doi.org/10.3837/tiis.2019.02.004 **ISSN : 1976-7277**

1. Introduction

WSN autonomous deployment in unattended and harsh environment leads to high frequency of faults due to internal and external factors. Faults can be attributed as hard and soft based upon their impact on sensor node behavior [1]. A soft faulty node continues to operate with erroneous behavior. Further classification of soft faults may then be based upon frequency and continuity as intermittent and transient [2]. Intermittent faults occur inconsistently and their impact is not visible always. Software malfunction are primary sources of such faults. Transient faults are temporary in nature and are caused by external sources such as environmental noise and interference. Communication errors such as loss of connectivity, routing loops and broadcast storms result in network partition, delay, throughput reduction and congestion etc. Likewise, wireless channel errors and radio interference are external causes of signal fading, collisions and packet loss etc. Therefore, timely detection and root cause analysis of such faults i.e., fault diagnosis, is important to ensure reliable and fault tolerant operation of WSNs.

The processes implementing the protocol stack for communication constitute software layer of sensor nodes. These processes communicate to exchange information through service access points and therefore, need to be observed for failures due to soft faults [3]. Therefore, analysis of process execution is important to identify process misbehavior and observe the channel effect as being erroneous. Consequently, it is pertinent to timely and correctly classify erroneous behavior as alerts, warnings and critical errors. Likewise, an analysis of propagation impact of one process failure on stacked or peer layer processes execution can also be applied for fault diagnosis. However, current fault diagnosis techniques do not analyze the systemic effect of communication errors on protocol execution at node, link and network levels. Furthermore, the impact of vertical and horizontal dissemination of a process failure on correlated processes has not been analyzed in depth. Therefore, motivated by the need to analyze correlations of stacked and peer layer processes, this work proposes an inter-process correlation model for fault diagnosis.

Architecturally, WSN fault diagnosis techniques are classified as centralized, distributed and hybrid depending upon location of decision center. In centralized architecture, sink node is responsible for network wide fault diagnosis. However, sink being a central entity is unable to analyze stacked process failures at individual nodes and link levels. In distributed architecture, each sensor node collects and analyzes responses from neighboring nodes to build local diagnostic view. Local views are propagated throughout the network. However, individual sensor nodes cannot capture and analyze peer process failures due to network wide errors. Hybrid architecture implements fault diagnosis at both node and sink/cluster head levels. Individual sensor nodes are instrumented for self-monitoring and fault detection. Whereas the cluster head handles network wide errors. Therefore, hybrid architecture is applicable to diagnose stacked process failures at node level and peer process failures at the cluster head.

The contributions of this work are as follows. A novel error classification mechanism is defined to detect and classify process failures. An inter-process correlation model is proposed to analyze the impact of a process failure on correlated processes. The concept of inter-process correlations has been applied in software engineering to detect and classify defects [4]. However, this concept has not been applied in context of WSNs for fault diagnosis. The proposed model is realized through local and global decision trees. A hybrid framework is

proposed to implement local decision tree on sensor nodes and global decision tree on diagnostic cluster head. The rest of the paper is organized as such. Section 2 presents a review of current fault diagnosis schemes for WSNs. In Section 3, the hybrid framework and modules are explained in detail. In Section 4, the inter-process correlation model-based fault diagnosis is analyzed using fault tree analysis. In Section 5, the framework implementation and performance evaluation in Castalia [5] are discussed. Finally, Section 6 concludes the paper with research findings.

2. Related Work

2.1 Fault Diagnosis Schemes

Many centralized fault diagnosis schemes for WSNs have been proposed. A node tracing and probing scheme to construct a belief network based inference engine at sink is presented in [6]. After observing network symptoms, sink reconstructs network topology involving the problematic region. The observed network exceptions are fed as input to the inference engine to analyze correlated symptoms. The inference engine generates multiple candidate fault sets. Incremental probing is then performed to collect information about un-observed symptoms for final fault diagnosis. The inference engine creates a data report vertex for each sensing node. For a large data centric network, complexity increases linearly with addition of data report vertices for all sensing nodes.

Sensor nEtwork DEfect Localization (SEDEL) scheme is proposed to handle topology changes due to transient faults [7]. The routing topology of each processing stage is modelled as a graph. A centralized sink stores the routing topology. A graph-mining approach is used to detect frequent subgraphs database. Afterwards, information gain for all nodes is computed. Lastly, to identify faulty node, information gain is used to rank each node. Performance evaluation exhibited that this method is able to localize faulty node's location in the routing table to maximum two neighbors.

A distributed fault diagnosis scheme for soft faults is presented in [8]. The network health is sampled periodically through 'heartbeat' messages. Afterwards, through comparison of heartbeat messages exchanged between neighbors and propagation of local diagnostic views, distributed fault diagnosis is implemented. The impact of transient faults in wireless communication and intermittent faults in sensing was investigated. Diagnostic complexity for intermittent faults turns out to be higher since multiple tests are required, resulting in high detection latency.

To handle network faults due to node failures, an agent based fault detection technique is presented in [9]. An agent packet is periodically transmitted to the neighbors by sink. The agent establishes a query path to reach the faulty node. The sender id, message id, and number of active nodes are stored by the neighbors in their query list. Afterwards, an Ack is transmitted back to the sender. A random decision about further transmission of the agent packet is made by receiver. In case of no Ack received within a certain time, the neighbor is considered dead. A packet consisting of dead node id and time to live (TTL) information is broadcasted.

A hybrid two-phased fault detection scheme using majority voting is proposed in [10]. In the first phase, local decision for fault state by each node is transmitted to sink. The sink takes a further decision by processing the data received from all nodes up to a certain time. A cluster based hybrid fault diagnosis scheme is proposed in [11]. The scheme defines three phases i.e., cluster formation, fault detection and classification. Time out mechanism is used for hard fault

detection. To detect soft, intermittent and transient faults, analysis of variance method (ANOVA test) is used.

Diagnosis traffic overhead, rapid energy depletion and delays make centralized architecture unsuitable for large scale WSN deployment [12]. Distributed fault diagnosis produces lesser overhead yet it suffers from achieving a good balance between detection accuracy and latency. Hybrid architecture overcomes these issues at the cost of additional components i.e., cluster head nodes. However, robustness, energy efficiency and minimal traffic overhead make hybrid architecture feasible for WSN fault diagnosis.

2.2 Communication Protocols Analysis for Fault Diagnosis

Several network monitoring systems for fault diagnosis in WSNs have been proposed. Sympathy [13] is a monitoring system that deploys agent code on each node to periodically collect and transmit node metrices to sink. The sink analyzes these metrics and executes a decision tree for fault diagnosis. A localized source is assigned to each detected failure i.e., self, path or sink. Z-Monitor is a protocol analyzer and monitoring tool to debug IEEE 802.15.4 based WSNs [14]. Multiple sniffer nodes capture and transmit data traffic to base station. Z-Monitor buffers incoming packets, perform decoding and parsing to evaluate network behavior. Fault diagnosis is performed to identify potential root causes of communication problems.

An in-situ diagnostic system for low power IPv6 WSNs is presented in [15]. The system consists of traffic monitor to capture IEEE 802.15.4 data frames, a frame decoder and a decision tree. The frame decoder extracts and decodes network and MAC layer headers to detect network problems including unresponsive node, network partition and intermittent dis-connectivity. Afterwards, the decision tree is employed to identify potential causes of these problems.

SNIF, a distributed framework to inspect and debug WSNs [16] employs multiple sniffers to overhear network traffic. Each sniffer node implements the receiving side of WSN protocol stack i.e., medium access control (MAC) and physical (PHY) layers. The captured data streams are transmitted to sink for analysis. The sink employs a decision tree to diagnose network malfunctions including node failures, link loss, routing loss and network partitions.

Passive monitoring system in wireless sensor networks (PMSW) [17], employs sniffer nodes to capture and transmit data packets to gateways. Local packet trace is generated by each gateway. The trace record from multiple gateways are merged to generate global trace at a server. Afterwards, the global trace is analyzed for fault detection. PMSW is based on analysis of data packets only. However, control packets, such as routing packets are not analyzed.

A hybrid debugging framework (HDF) to monitor WSNs is presented in [18]. HDF implements dynamic tracing through device and program agents on each sensor node. A monitor node is used for managing remotely deployed nodes. However, HDF requires specialized hardware equipment increasing deployment cost.

Existing monitoring systems perform WSN traffic analysis for fault diagnosis. However, these monitoring systems do not analyze effect of communication errors and faults on protocol execution and process failure. In **Table 1**, a comparison of monitoring systems for WSNs is presented. Therefore, there is need to further explore inter-process correlations of routing, MAC and PHY layer protocols for fault diagnosis. Moreover, hybrid architecture is suitable for inter-process correlations based fault diagnosis in WSN.

Table 1. Comparison of monitoring systems for fault diagnosis in WSN

Monitoring Systems	Architecture	Protocols Analysis			Inter-Process Correlations for Fault Diagnosis
		Routing	MAC	PHY	
Sympathy [13]	Centralized	Yes	No	No	No
Z-Monitor [14]	Centralized	Yes	Yes	No	No
In-situ Diagnostic [15]	Centralized	Yes	Yes	No	No
SNIF [16]	Distributed	No	Yes	Yes	No
PMSW [17]	Distributed	Yes	No	No	No
HDF [18]	Hybrid	Yes	No	No	No
Proposed Framework	Hybrid	Yes	Yes	Yes	Yes

To the best of authors' knowledge, most of the current fault diagnosis schemes do not substantiate relationships between fault diagnosis and inter-process correlations of functional peers. In [19], the authors presented a hybrid fault diagnosis architecture based on inter-process communication of routing and MAC layer processes. However, PHY layer processes were not handled. The node level diagnostic agent was defined for failure detection and classification only. Fault diagnosis module was defined on cluster head. The diagnostic communication cost and reliability were analyzed analytically. The architecture was defined on conceptual level only. This work extends by proposing an inter-process correlation model for routing, MAC and PHY layer processes. The proposed hybrid framework encompasses fault diagnosis modules for both sensor nodes and cluster head. The framework has been implemented and evaluated in Castalia.

3. Hybrid Framework

The proposed hybrid framework is aimed at analysis of both the communication protocols specifications and implementations through online process execution tracking as shown in **Fig. 1**. Firstly, typical processes that execute on WSN communication protocol stack are identified. IEEE 802.15.4 MAC, PHY layer protocols [20] and AODV [21] have been selected. The process flows of these protocols build foundation for the hybrid framework. For investigation of process executing on network, MAC and PHY layers, a periodic probing scheme is defined. Analysis code examines probe results to detect process failure and classify error level. Finally, fault diagnosis is performed through local decision tree (LDT) and global decision tree (GDT). LDT and GDT are based upon inter-process correlations of both stacked and peer layer processes at node, link and network levels.

Partial and deployment specific realization of the hybrid framework can be attained through LDT on sensor nodes and GDT on diagnostic cluster head (DCH) respectively, as shown in **Fig. 2**. DCH is a specialized entity to handle network wide errors. LDT is implemented on each sensor node to diagnose critical failure caused by errors in stacked processes. GDT is executed on DCH to diagnose critical failures due to errors in peer layer processes.

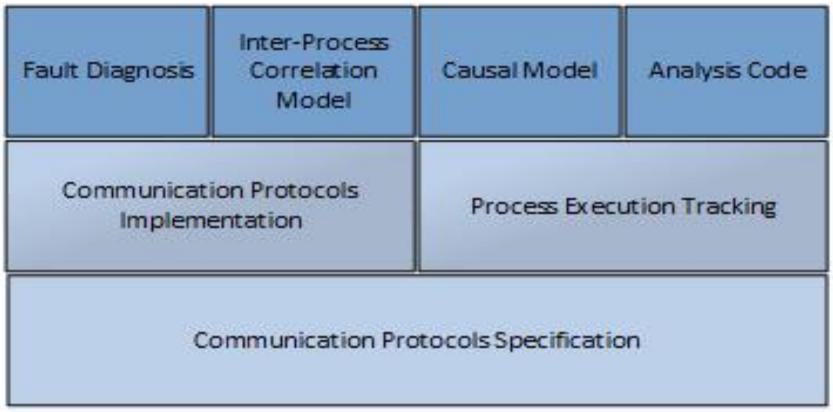

Fig. 1. Inter-process correlation model based hybrid framework for fault diagnosis in WSN

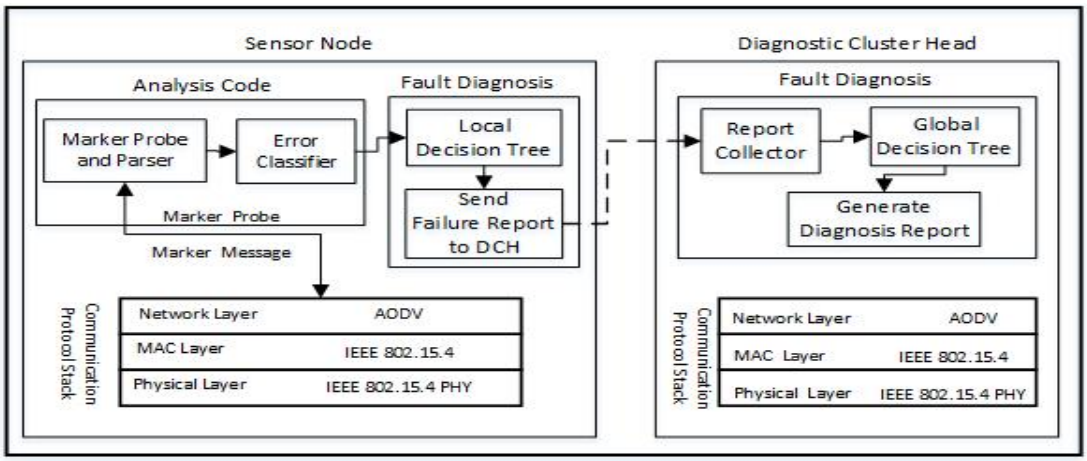

Fig. 2. Modular architecture for deployment of the hybrid framework on sensor nodes and DCH

3.1 Communication Protocols Specification and Implementation

IEEE 802.15.4 standard defines a communication protocol for wirelessly interconnected devices in a personal area network (PAN) [20]. The standard defines both PHY and MAC layers. The processes executed by PHY layer include channel selection, energy management, signal modulation and signal transmission /reception. The MAC sub-layer is based upon PHY layer. The processes executed by MAC layer include association, synchronization with PAN coordinator, channel sense and frame transmission. MAC layer performs channel access using carrier sense multiple access with collision avoidance (CSMA-CA) algorithm. The network layer is responsible for topology specific routing. AODV is a reactive routing protocol that executes route discovery process on demand [21]. Standard implementations of routing, MAC and PHY layers protocols are available in simulators such as Castalia.

3.2 Process Execution Tracking

The processes and procedures running on stacked and peer layers of the protocol stack form software components of WSN as shown in **Fig. 3.** The process execution status is functionally dependent upon state of the internal procedures. For online process tracking, the state of each procedure within a process is stored as a marker. All procedures collectively form a marker

pattern which represents execution status of the process on a particular layer. The procedure execution state is encoded for two states only i.e. normal and error. Under regular conditions, procedures within a process execute without any error and return a normal marker. Exception handling code in protocol implementation is also provided to generate warnings and alerts, that triggers error markers creation on multiple layers. Process execution sequence and markers are stored in real time in the form of *Process Execution Stack (PES)*, as shown in **Fig. 4**.

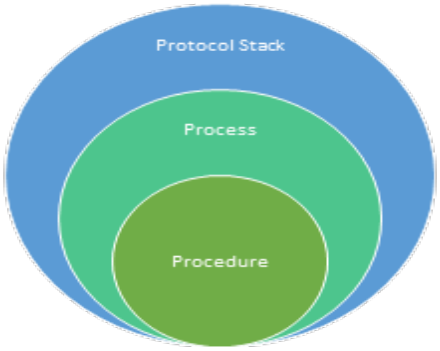

Fig. 3. Software components of WSN

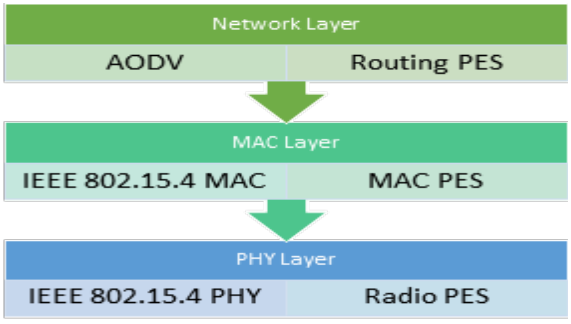

Fig. 4. Process execution stack on each layer of communication protocol stack

3.3 Causal Model

Traditionally, causal models are represented as a relationship between fault, error and failure [22]. We define causal model as the impact of communication related phenomena on process execution. The key terms that connect the chain of events from a phenomenon (cause) to a malfunction (effect) that appears at a protocol layer are defined as such.

- Fault is an erroneous state of a software component. Fault manifests itself through occurrence of errors.
- Error represents an incorrect procedure state. The error propagation across multiple layers of the protocol stack results in process failure.
- Failure is deviation from normal process behavior and is a service disruption.

The causal model relates errors of a specific type with a fault that resulted in a failure. The model maps process failure to procedural errors. The errors are mapped to underlying faults as shown in **Fig. 5**.

Fig. 5. Causal model to analyze process failure

3.4 Analysis Code

Analysis code comprises of *Marker Probe and Parser* and *Error Classifier* modules. A marker probe is sent periodically to retrieve markers by traversing respective PES after a certain interval, designated as probe period. Parser decodes error markers from probe results that may be embodying process failures. Each process failure is mapped to procedural errors according to the causal model. Subsequently, procedural error counters for each process are generated.

The error classification is performed based upon temporal impact and continuity of the procedural errors on process execution and node functional behavior. Three error levels are defined as such *critical*, *warning* and *alert. Critical* errors are main source of process failure which disrupt communication and may cause sensor node dis-connectivity. An example of such error could be the inability of a sensor node to associate/remain associated with PAN coordinator. The errors which have a temporary effect on sporadic occurrences may cause complete process failure if they persist. Such errors are classified as *warning* e.g., radio buffer overflow for *send data* process. *Alert* level is assigned for those errors that result in communication interruptions such as radio non-receiving (RX) state for *channel sense* process.

Error Classifier module performs top down comparison of error counters to detect process failures and classify error levels against thresholds. After failure detection and error classification, inter-process correlation model based decision tree is employed for fault diagnosis.

3.5 Inter-Process Correlation Model

Process execution sequence is based upon inter-process communication in the form of up/down calls (stacked processes) and to/from calls (peer processes) as depicted in **Fig. 6**. According to down call sequence, network layer *route discovery* process on an originating node is dependent on MAC layer *transmit frame*. Similarly, MAC *association*, *synchronization (PAN SYNC)*, *channel sense* and PHY layer *send data* processes affect *transmit frame* process. *Association* process depends on *synchronization*. PHY layer *receive data process* receives and sends beacon frames to MAC layer i.e, an up call to *receive frame* process. Therefore, *synchronization* is dependent on these two correlated processes. The failure of any of these correlated processes may cause the dependent *transmit frame* process to fail. The failure effect is propagated upward the protocol stack for a down call.

Similarly, peer routing processes i.e., *process RREQ*, *generate RREP* and *process RREP* on intermediate and destination nodes are dependent on correlated stacked processes. Based on to/from calls sequence, *route discovery* process is dependent on these peer routing processes. Therefore, on peer layers of protocol stack, failure effect may propagate horizontally. The inter-process correlations are therefore inferred from these calls and represented as a model. The components in the model are both stacked and peer layer processes on each sensor node as depicted in **Fig. 7**. The arrows denote conditional dependency of upper layer processes on lower layer and/or peer layer processes. These arrows are directed from correlated processes to the dependent process e.g., from *transmit frame* to *route discovery* process.

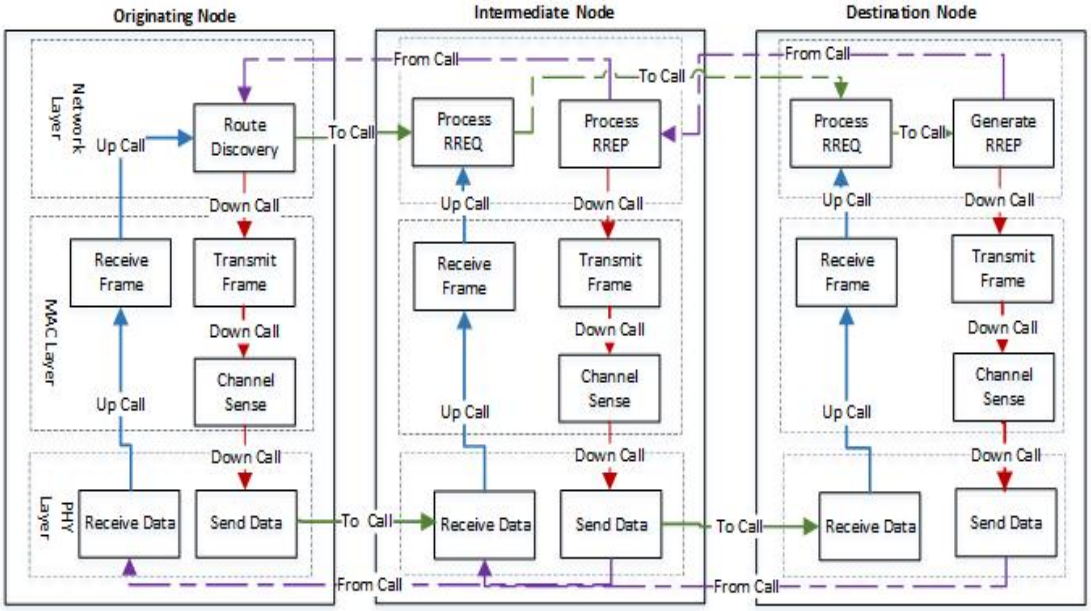

Fig. 6. Down /up calls between stacked processes and to/from calls between peer layer processes

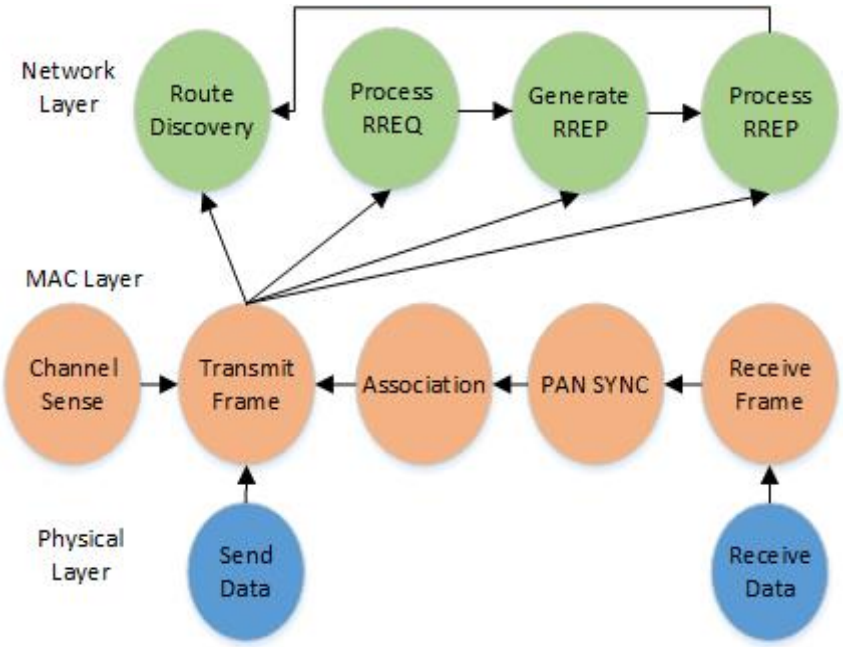

Fig. 7. Inter-process correlation model for node level stacked processes and peer layer processes

3.6 Fault Diagnosis

Decision tree is found to be a natural contender for root cause analysis in WSNs [15] due to inherent relationships of network entities and data emulating a tree structure. In the hybrid framework, the decision tree utilizes inter-process correlation model for fault diagnosis. For each critical failure, LDT is employed to identify critical errors, warnings and alerts due to

primary faults according to the causal model, as shown in **Fig. 8**. LDT is only partially parsed on multiple sensor nodes according to currently running processes at each sensor node in a specific probe round. Each decision test tallies error counters of correlated processes (critical errors, warnings and alerts) against the respective down call/up call counters.

For instance, in case of *route discovery* failure, error counters of *transmit frame, association, synchronization, channel sense* and *send data* processes are successively compared with *route discovery* down call counter. If error counters of these correlated processes are greater than the down call counter, primary faults are added to list of potential root causes. *Transmit frame* process is affected by both *association* failure i.e., a critical error and *channel sense* failure.

Association process may fail due to *synchronization* failure that is further correlated with *receive frame* and *receive data* processes. *Receive data* process fails due to the following alerts and warnings i.e., (a) radio state error (b) low power signal (c) maximum bit errors (d) maximum interference. In case of *synchronization* failure i.e., a critical error, error counters of correlated *receive data* process are tested against up call counter to *receive frame*. If these error counters are greater than the up-call counter, primary faults are diagnosed as potential root causes.

Channel sense process is based on CSMA-CA algorithm to transmit MAC frame. The algorithm implementation is based on time units called back off periods. After each attempt to perform clear channel assessment (CCA), the algorithm randomly back offs before next CCA. If number of back offs are greater than maximum number of back offs allowed, *channel sense* process fails. Moreover, *channel sense* process is delayed if radio is not in RX state while attempting CCA. In both cases, LDT executes decision tests to compare these error counters against down call counter of *transmit frame* process. In case of, these error counters being greater than the down call counter, primary faults are inserted into the list of potential root causes.

Transmit frame process may also fail due to *send data* failure. In this case, LDT executes decision test to compare *transmit frame* down call counter against error counter of *send data* process. If this error counter is greater than the down call counter, LDT infers radio buffer overflow as a potential root cause. Similarly, peer routing process may fail due to errors in correlated stacked processes. Therefore, similar decision tests are executed to diagnose peer routing process failure.

If partial traversal of LDT successfully diagnoses primary faults, a diagnosis report containing critical process failure and root causes is produced. Otherwise, the failure source may be external. The external sources are not observable on this node; therefore, this scenario is deemed as a peer layer process failure. In this case, a failure report containing partial diagnosis results is generated and sent to DCH for detailed analysis. Similar reports are generated for peer layer routing process failures and sent to DCH after each probe round.

The *Report Collector* module on DCH parses received failure reports. GDT on DCH analyzes impact of errors in peer layer routing processes on *route discovery* failure as shown in **Fig. 9**. After executing decision tests to diagnose *route discovery* failure, a diagnosis report is generated.

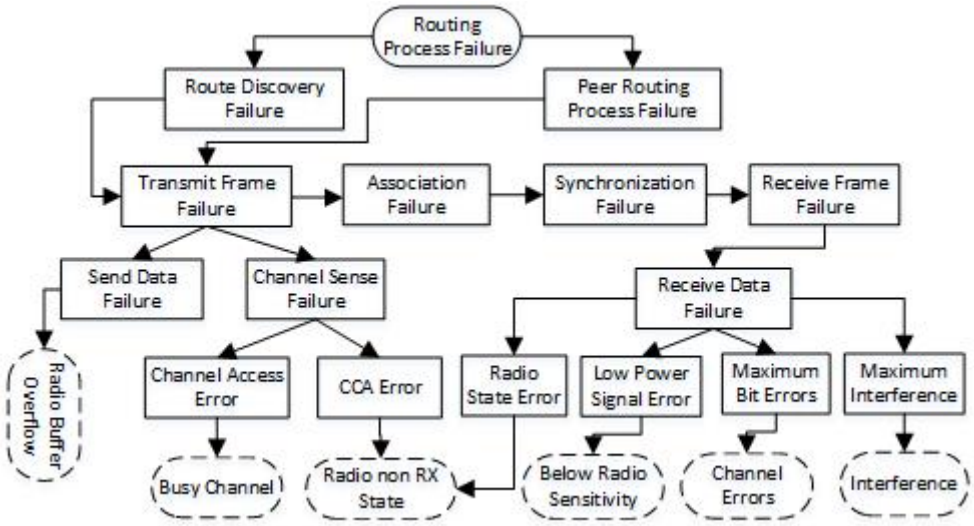

Fig. 8. LDT to diagnose routing process failure: each rectangle denotes a decision test, dotted ovals (terminal nodes) represent primary faults

According to AODV specification [21] on unidirectional links, a node executing *process RREQ* (to-call) places source of incoming RREQ in a black list resulting into error. If this error counter is greater than the to-call counter, failure effect propagates on peer layers of protocol stack causing *route discovery* failure. RREP packet is generated by destination node and intermediate nodes transmit back on reverse route to the originating node i.e., from-call. In both cases, if reverse route to the next hop does not exist in routing table of the node, *generate RREP* or *process RREP* failure occur. If this error counter is greater than the from-call counter, *route discovery* process fails due to dissemination of failure effect on peer layers of protocol stack.

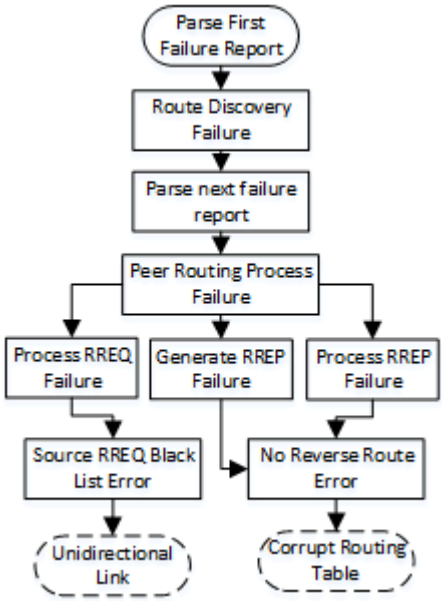

Fig. 9. GDT on DCH to diagnose route discovery failure due to errors in peer routing processes

4. Mathematical Analysis

In this section, we analyze inter-process correlation model based decision tree for fault diagnosis. For this purpose, the decision tree is alternatively represented as a fault tree.

A fault tree (FT) is a directed acyclic graph comprising of two types of nodes: events and gates [23]. An event is an incidence within the system, generally failure of a subsystem down to a specific component. Events are divided into basic events (BEs) and intermediate events. A basic event is a primary fault that does not need any additional development, represented as leaf node of the tree. An intermediate event is triggered by logical combinations of basic events. The event at root of fault tree i.e., top event models system failure.

Gates represent logical combinations of component failures eventually leading to an overall system failure. Logical AND, OR, NOT and INHIBIT gates are used. INHIBIT is a special case of AND gate with single input and a conditioning event. In this case, output event occurs if the input event occurs along with the conditioning event.

4.2 Fault Tree Analysis

Fault tree analysis (FTA) is a deductive technique that identifies all possible causes of a particular system state in terms of state of components within the system [24]. The qualitative FTA considers structure of FT and is used to detect deviations from normal system behavior. An important qualitative measure is finding minimal cut sets. A cut set is a set of components that can together cause the system to fail. A minimal cut set (MC) is the smallest combination of component failures that will cause the top event to occur.

The minimal cut sets of a fault tree can be determined by first converting the tree to its equivalent Boolean expression. Then either top-down or bottom-up approach is applied [25]. Each gate is represented as a Boolean expression of BEs and/or other gates. Through algebraic manipulation, an expression is deduced to relate the top event to BEs excluding gates. Generalized form of minimal cut set expression for the top event can be written as.

$$TE = MC_1 + MC_2 + MC_3 + \cdots + MC_k \tag{1}$$

In equation 1, TE is the top event and MC_i is a minimal cut set. The expression for n-component minimal cut set can be written as

$$MC_i = F_1 \cdot F_2 \cdot \ldots \cdot F_n \tag{2}$$

In equation 2, F_1, F_2, etc., represent basic component failures in a fault tree.

4.3 Local Decision Tree to Fault Tree

After formal introduction of FTs, we build an equivalent local fault tree for LDT as shown in **Fig. 10**. The network, MAC and PHY layer processes represent components in the system. The set of BEs comprises of terminal nodes of LDT from **Fig. 8**. Intermediate events are set of correlated stacked processes failures, according to the inter-process correlation model. The top event is routing failure to be analyzed. We determine minimal cut sets that represent necessary conditions for the routing failure.

Definition 1: Necessary conditions for routing failure are set of errors in correlated stacked processes at node level.

According to the inter-process correlation model, *route discovery* process on an originating node is also correlated with peer routing processes executing on intermediate or destination nodes. This condition is modeled as an undeveloped event in the local fault tree, representing an external failure. Because, the information related to peer routing process failure at network level is not known at node level.

We use $GateTypes = \{OR, INHIBIT\}$ only. We use INHIBIT gate to represent a conditioning event that is essentially an error counter of a process exceeding the down-call/up-call counters of correlated process(es). **Table 2** contains a list of basic events and corresponding conditioning events. In **Table 3**, a list of conditioning events for intermediate events is defined.

Table 2. List of notations for basic events and corresponding conditioning events

Basic Events	Description	Conditioning Event	Description
OF	Radio Buffer Overflow	E_{OF}	Error counter E_{OF} > down-call counter of *send data* process
NRX	Radio non-RX State	E_{NRX}	Error counter E_{NRX} > up-call counter of *receive data* process
		E_{NRX2}	Error counter E_{NRX2} > down-call counter of *channel sense* process
BRS	Below Radio Sensitivity	E_{BRS}	Error counter E_{BRS} > up-call counter of *receive data* process
CE	Channel Errors	E_{CE}	Error counter E_{CE} > up-call counter of *receive data* process
INF	Interference	E_{INF}	Error counter E_{INF} > up-call counter of *receive data* process
BC	Busy Channel	E_{BC}	Error counter E_{BC} > down-call counter of *channel sense* process

Table 3. List of notations used for conditioning events for intermediate events

Notation	Description	Conditioning Event
E_{TF}	*transmit frame* error counter	E_{TF} > down-call counter of *route discovery* process
E_{SD}	*send data* error counter	E_{SD} > down-call counter of *transmit frame* process
E_{AS}	*association* error counter	E_{AS} > down-call counter of *transmit frame* process
E_{RF}	*receive frame* error counter	E_{RF} > up-call counter of *receive frame* process
E_{RD}	*receive data* error counter	E_{RD} > up-call counter of *receive data* process
E_{CS}	*channel sense* error counter	E_{CS} > down-call counter of *receive frame* process
EF	External failure	*route discovery* failure due to external failure
N_O	Originating node	*route discovery* process execution on No
N_{ID}	Intermediate or destination node	peer routing process execution on N_{ID}

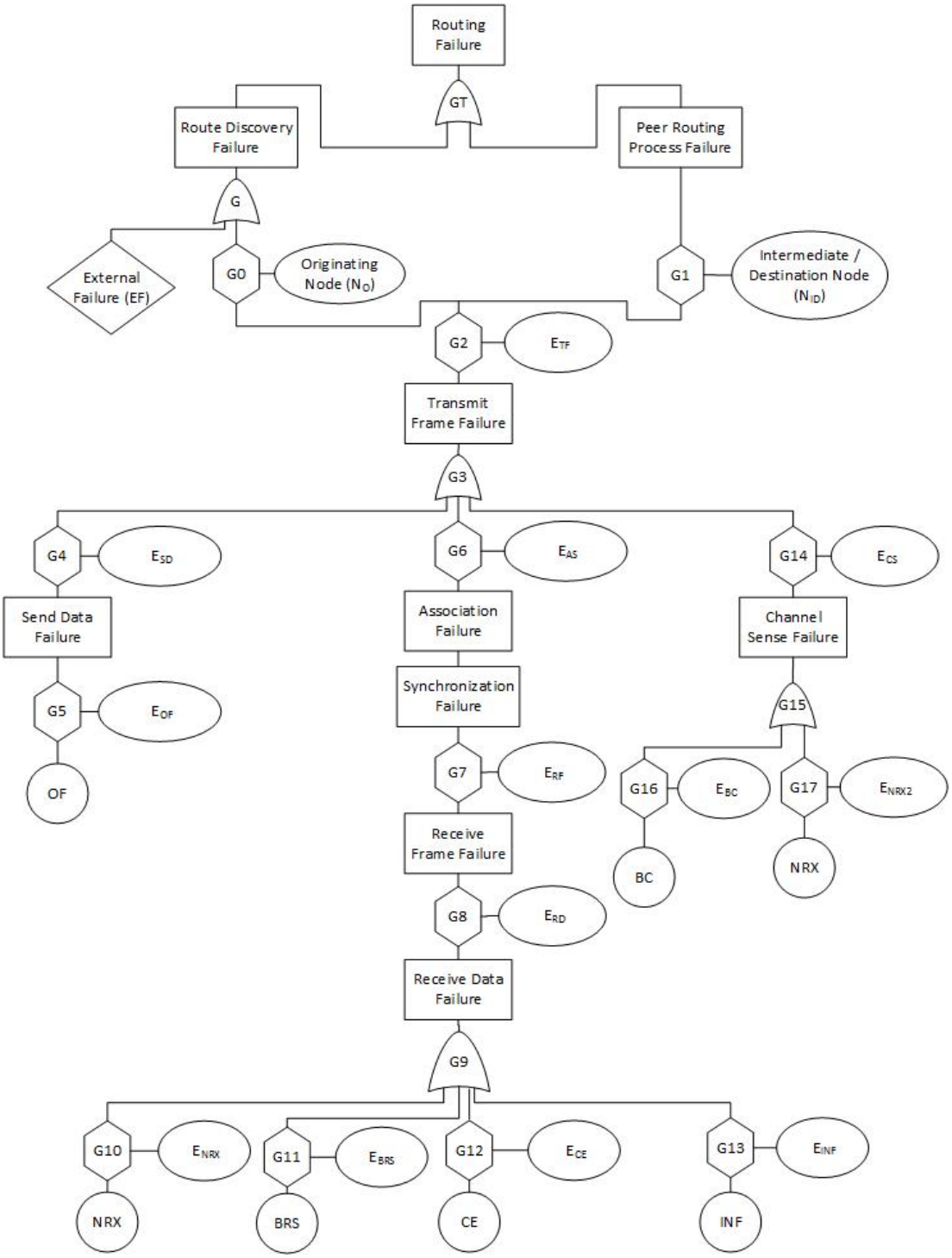

Fig. 10. Equivalent local fault tree for LDT: circles represent basic events, rectangles represent intermediate events, hexagons denote INHIBIT gates, ellipses denote conditioning events, diamond represents an undeveloped event

4.4 Minimal Cut Set Analysis

Formally defining, let the set of basic events *BE1* for local fault tree be represented as such.

$$BE1 = \{OF, NRX, BRS, CE, INF, BC\} \tag{3}$$

The set of gates used is divided into *OG*, *IG* representing OR and INHIBIT gates respectively.

$$OG = \{GT, G, G3, G9, G15\} \tag{4}$$
$$IG = \{G0, G1, G2, G4, G5, G6, G7, G8, G10, G11, G12, G13, G14, G16, G17\} \tag{5}$$

The equivalent Boolean equations for each gate are defined as follows.

$$GT = G + G1 \tag{6}$$
$$G = G0 + EF \tag{7}$$
$$G0 = G2 \cdot N_o \tag{8}$$
$$G1 = G2 \cdot N_{ID} \tag{9}$$
$$G2 = G3 \cdot E_{TF} \tag{10}$$
$$G3 = G4 + G6 + G14 \tag{11}$$
$$G4 = G5 \cdot E_{SD} \tag{12}$$
$$G5 = OF \cdot E_{OF} \tag{13}$$
$$G6 = G7 \cdot E_{AS} \tag{14}$$
$$G7 = G8 \cdot E_{RF} \tag{15}$$
$$G8 = G9 \cdot E_{RD} \tag{16}$$
$$G9 = G10 + G11 + G12 + G13 \tag{17}$$
$$G10 = NRX \cdot E_{NRX} \tag{18}$$
$$G11 = BRS \cdot E_{BRS} \tag{19}$$
$$G12 = CE \cdot E_{CE} \tag{20}$$
$$G13 = INF \cdot E_{INF} \tag{21}$$
$$G14 = G15 \cdot E_{CS} \tag{22}$$
$$G15 = G16 + G17 \tag{23}$$
$$G16 = BC \cdot E_{BC} \tag{24}$$
$$G17 = NRX \cdot E_{NRX2} \tag{25}$$

We use top-down approach to determine minimal cut sets for the top event i.e., routing failure. Starting from top event equation, we substitute and expand until minimal cut set expression for the top event is deduced. Recalling equation 6, $GT = G + G1$ substitute for G, G1 from equation 7 and 8 respectively.

$$GT = (G0 + EF) + (G2 \cdot N_{ID}) \tag{26}$$

However, to simplify we ignore EF and expand G0, G2 from equation 9, 10.

$$GT = (G3 \cdot E_{TF} \cdot N_O) + (G3 \cdot E_{TF} \cdot N_{ID}) \tag{27}$$

Now substitute for G3 from equation 11,

$$GT = \big((G4 + G6 + G14) \cdot E_{TF} \cdot N_O\big) + \big((G4 + G6 + G14) \cdot E_{TF} \cdot N_{ID}\big) \tag{28}$$

Converting into disjunctive normal form (DNF) so that each conjunction is a MC.

$$GT = (G4 \cdot E_{TF} \cdot N_O) + (G6 \cdot E_{TF} \cdot N_O) + (G14 \cdot E_{TF} \cdot N_O) + (G4 \cdot E_{TF} \cdot N_{ID}) + (G6 \cdot E_{TF} \cdot N_{ID}) + (G14 \cdot E_{TF} \cdot N_{ID}) \quad (29)$$

Substituting for G4 (equation 12), G6 (equation 14) and G14 (equation 22), we get.

$$GT = (G5 \cdot E_{SD} \cdot E_{TF} \cdot N_O) + (G7 \cdot E_{AS} \cdot E_{TF} \cdot N_O) + (G15 \cdot E_{CS} \cdot E_{TF} \cdot N_O) + (G5 \cdot E_{SD} \cdot E_{TF} \cdot N_{ID}) + (G7 \cdot E_{AS} \cdot E_{TF} \cdot N_{ID}) + (G15 \cdot E_{CS} \cdot E_{TF} \cdot N_{ID}) \quad (30)$$

Now replace G5, G7 and G15 by equation 13, 15 and 23 respectively.

$$GT = (OF \cdot E_{OF} \cdot E_{SD} \cdot E_{TF}.N_O) + (G8 \cdot E_{RF} \cdot E_{AS} \cdot E_{TF} \cdot N_O) + \big((G16 + G17) \cdot E_{CS} \cdot E_{TF} \cdot N_O\big) + (OF \cdot E_{OF} \cdot E_{SD} \cdot E_{TF} \cdot N_{ID}) + (G8 \cdot E_{RF} \cdot E_{AS} \cdot E_{TF} \cdot N_{ID}) + \big((G16 + G17) \cdot E_{CS} \cdot E_{TF} \cdot N_{ID}\big) \quad (31)$$

Simplifying to DNF, we get.

$$GT = (OF \cdot E_{OF} \cdot E_{SD} \cdot E_{TF}.N_O) + (G8 \cdot E_{RF} \cdot E_{AS} \cdot E_{TF} \cdot N_O) + (G16 \cdot E_{CS} \cdot E_{TF} \cdot N_O) + (G17 \cdot E_{CS} \cdot E_{TF} \cdot N_O) + (OF \cdot E_{OF} \cdot E_{SD} \cdot E_{TF} \cdot N_{ID}) + (G8 \cdot E_{RF} \cdot E_{AS} \cdot E_{TF} \cdot N_{ID}) + (G16 \cdot E_{CS} \cdot E_{TF} \cdot N_{ID}) + (G17 \cdot E_{CS} \cdot E_{TF} \cdot N_{ID}) \quad (32)$$

Now substitute for G8 (equation 16), G16 (equation 24) and G17 (equation 25).

$$GT = (OF \cdot E_{OF} \cdot E_{SD} \cdot E_{TF}.N_O) + (G9 \cdot E_{RD} \cdot E_{RF} \cdot E_{AS} \cdot E_{TF} \cdot N_O) + (BC \cdot E_{BC} \cdot E_{CS} \cdot E_{TF} \cdot N_O) + (NRX \cdot E_{NRX} \cdot E_{CS} \cdot E_{TF} \cdot N_O) + (OF \cdot E_{OF} \cdot E_{SD} \cdot E_{TF} \cdot N_{ID}) + (G9 \cdot E_{RD} \cdot E_{RF} \cdot E_{AS} \cdot E_{TF} \cdot N_{ID}) + (BC \cdot E_{BC} \cdot E_{CS} \cdot E_{TF} \cdot N_{ID}) + (NRX \cdot E_{NRX2} \cdot E_{CS} \cdot E_{TF} \cdot N_{ID}) \quad (33)$$

Now replace G9 from equation 17, we get equation 34.

$$GT = (OF \cdot E_{OF} \cdot E_{SD} \cdot E_{TF}.N_O) + \big((G10 + G11 + G12 + G13) \cdot E_{RD} \cdot E_{RF} \cdot E_{AS} \cdot E_{TF} \cdot N_O\big) + (BC \cdot E_{BC} \cdot E_{CS} \cdot E_{TF} \cdot N_O) + (NRX \cdot E_{NRX} \cdot E_{CS} \cdot E_{TF} \cdot N_O) + (OF \cdot E_{OF} \cdot E_{SD} \cdot E_{TF} \cdot N_{ID}) + \big((G10 + G11 + G12 + G13) \cdot E_{RD} \cdot E_{RF} \cdot E_{AS} \cdot E_{TF} \cdot N_{ID}\big) + (BC \cdot E_{BC} \cdot E_{CS} \cdot E_{TF} \cdot N_{ID}) + (NRX \cdot E_{NRX2} \cdot E_{CS} \cdot E_{TF} \cdot N_{ID}) \quad (34)$$

After conversion to DNF, the resulting expression is equation 35.

$$GT = (OF \cdot E_{OF} \cdot E_{SD} \cdot E_{TF}.N_O) + (G10 \cdot E_{RD} \cdot E_{RF} \cdot E_{AS} \cdot E_{TF} \cdot N_O) + (G11 \cdot E_{RD} \cdot E_{RF} \cdot E_{AS} \cdot E_{TF} \cdot N_O) + (G12 \cdot E_{RD} \cdot E_{RF} \cdot E_{AS} \cdot E_{TF} \cdot N_O) + (G13 \cdot E_{RD} \cdot E_{RF} \cdot E_{AS} \cdot E_{TF} \cdot N_O) + (BC \cdot E_{BC} \cdot E_{CS} \cdot E_{TF} \cdot N_O) + (NRX \cdot E_{NRX} \cdot E_{CS} \cdot E_{TF} \cdot N_O) + (OF \cdot E_{OF} \cdot E_{SD} \cdot E_{TF} \cdot N_{ID}) + (G10 \cdot E_{RD} \cdot E_{RF} \cdot E_{AS} \cdot E_{TF} \cdot N_{ID}) + (G11 \cdot E_{RD} \cdot E_{RF} \cdot E_{AS} \cdot E_{TF} \cdot N_{ID}) + (G12 \cdot E_{RD} \cdot E_{RF} \cdot E_{AS} \cdot E_{TF} \cdot N_{ID}) + (G13 \cdot E_{RD} \cdot E_{RF} \cdot E_{AS} \cdot E_{TF} \cdot N_{ID}) + (BC \cdot E_{BC} \cdot E_{CS} \cdot E_{TF} \cdot N_{ID}) + (NRX \cdot E_{NRX2} \cdot E_{CS} \cdot E_{TF} \cdot N_{ID}) \quad (35)$$

Substituting G10, G11, G12 and G1 from equation 18, 19, 20 and 21 respectively.

$$GT = (OF \cdot E_{OF} \cdot E_{SD} \cdot E_{TF}.N_O) + (NRX \cdot E_{NRX} \cdot E_{RD} \cdot E_{RF} \cdot E_{AS} \cdot E_{TF} \cdot N_O) + (BRS \cdot E_{BRS} \cdot E_{RD} \cdot E_{RF} \cdot E_{AS} \cdot E_{TF} \cdot N_O) + (CE \cdot E_{CE} \cdot E_{RD} \cdot E_{RF} \cdot E_{AS} \cdot E_{TF} \cdot N_O) + (INF \cdot E_{NRF} \cdot E_{RD} \cdot E_{RF} \cdot E_{AS} \cdot E_{TF} \cdot N_O) + (BC \cdot E_{BC} \cdot E_{CS} \cdot E_{TF} \cdot N_O) + (NRX \cdot E_{NRX} \cdot E_{CS} \cdot E_{TF} \cdot N_O) + (OF \cdot E_{OF} \cdot E_{SD} \cdot E_{TF} \cdot N_{ID}) + (NRX \cdot E_{NRX} \cdot E_{RD} \cdot E_{RF} \cdot E_{AS} \cdot E_{TF} \cdot N_{ID}) + (BRS \cdot E_{BRS} \cdot E_{RD} \cdot E_{RF} \cdot E_{AS} \cdot E_{TF} \cdot N_{ID}) + (CE \cdot E_{CE} \cdot E_{RD} \cdot E_{RF} \cdot E_{AS} \cdot E_{TF} \cdot N_{ID}) + (INF \cdot E_{NRF} \cdot E_{RF} \cdot E_{AS} \cdot E_{TF} \cdot N_{ID}) + (BC \cdot E_{BC} \cdot E_{CS} \cdot$$

$E_{TF} \cdot N_{ID}) + (NRX \cdot E_{NRX2} \cdot E_{CS} \cdot E_{TF} \cdot N_{ID})$
(36)

Equation 36 gives the minimal cut sets for routing failure. Each cut set contains a combination of correlated stacked processes failures that together lead to routing failure under conditioning events. The top event thus contains 14 minimal cut sets with the smallest comprising of 5 events. These cut sets define the necessary conditions for routing failure to occur on a sensor node.

4.5 Global Decision Tree to Fault Tree

Similarly, GDT defined on DCH is modeled as a global fault tree representing *route discovery* failure due to peer routing process(es) failure. In global fault tree, peer layer routing processes are the system components as shown in **Fig. 11**. The set of BEs contains terminal nodes of GDT from **Fig. 9**. The set of intermediate events comprises of correlated peer routing processes failures. The top event is *route discovery* failure to be analyzed. We determine minimal cut sets that represent sufficient conditions for *route discovery* failure.
Definition 2: Sufficient conditions for *route discovery* failure are set of errors in peer layer routing processes at network level.

In **Table 4,** a list of notations used for BEs and their description is given. **Table 5** enlists notations used to represent conditioning events for intermediate events.

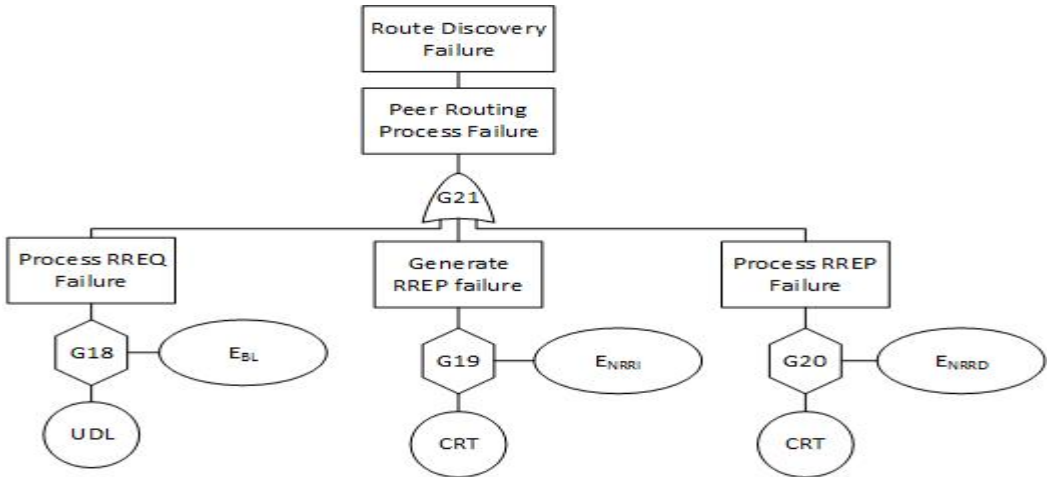

Fig. 11. Equivalent global fault tree representation for GDT

Table 4. Basic events and their description

Basic Events	Description
UDL	Uni-directional link
CRT	Corrupt routing table

Table 5. List of notations for conditioning events

Notation	Description	Conditioning Event
E_{BL}	Source-RREQ-black-list error counter	E_{BL} > to-call counter of *process RREQ*
E_{NRRI}	No reverse route error counter	E_{NRRI} > from-call counter of *process RREP* on intermediate node
E_{NRRD}	No reverse route error counter	E_{NRRD} > from-call counter of *generates RREP* on destination node

Formally, let the set of BEs, *BE2* for global fault tree be represented as such.

$$BE2 = \{UDL, CRT\} \tag{37}$$

The equivalent Boolean expressions for gates in global fault tree are defined as follows.

$$G21 = G18 + G19 + G20 \tag{38}$$
$$G18 = UDL \cdot E_{BL} \tag{39}$$
$$G19 = CRT \cdot E_{NRRI} \tag{40}$$
$$G20 = CRT.E_{NRRD} \tag{41}$$

Now, we use top-down approach to determine minimal cut sets for the top event. Referring equation 38, $G21 = G18 + G19 + G20$, substituting G18, G19 and G20 from equation 39, 40 and 41 respectively, we get.

$$G21 = (UDL \cdot E_{BL}) + (CRT.E_{NRRI}) + (CRT \cdot E_{NRRD}) \tag{42}$$

The minimal cut sets in equation 42 represent the basic events that together cause the top event under the presence of conditioning events. These minimal cut sets represent the sufficient conditions for *route discovery* failure. The fault tree analysis qualitatively evaluates the proposed inter-process correlation model based fault diagnosis. In the next section, the hybrid framework is experimentally evaluated.

5. Framework Implementation and Evaluation

5.1 Framework Implementation

The proposed framework has been implemented in Castalia under OMNET++ on ubuntu. The communication module in Castalia has been modified to store routing, MAC and PHY layers markers in stack data structure. To collect markers, analysis code on application layer sends a marker probe to the communication module.

On network layer, handleNetworkControlCommand () function has been overridden to process the marker probe by collecting and reporting routing markers. Simultaneously, network layer sends the marker probe to MAC layer. IEEE 802.15.4 implementation has been enhanced by overriding handeControlCommand () function for markers collection and reporting. Afterwards, MAC sends the marker probe to PHY layer as a radio control command. The radio module has been modified to recognize and process the probe command. The framework defines a failure report packet format that extends default application packet in Castalia.

5.2 Performance Evaluation

To evaluate performance of the hybrid framework, extensive simulation experiments have been done in Castalia.

5.2.1 Simulation Environment

In simulation setup, network consists of 20 nodes uniformly distributed in a two-dimensional grid of size 40 × 60 meters. Both sensor nodes and DCH are static. For selection of DCH, the algorithm proposed in [26] can be used. Constant bit rate (CBR) traffic model has been used with packet rate of 1 packet per second. On network layer, AODV routing protocol has been used. On MAC layer, IEEE 802.15.4 protocol with beacon enabled mode has been used. For contenion based channel access, slotted CSMA-CA is used. The maxCSMABackOffs defines maximum number of back-offs for channel access. This parameter is set to range 1- 4 specified in IEEE 802.15.4 standard. On PHY layer, CC2420 radio model has been used with PHY data rate of 250 Kbps. Receiver sensitivity parameter defines minimum signal strength receivable by the radio. This parameter has been set to -95 dbm for CC2420. Radio transmitting output power has been set at -5 dbm according to CC2420 model.

For wireless channel, log normal shadowing model has been used. In this model, path loss PL(d) at distance d from a transmitting node is calculated based on path loss exponent and distance between two nodes. The path loss exponent defines rate at which signal attenuates. This has been set to default value in Castalia.

Radio computes signal to noise ratio (SNR) of the received signal using signal power and radio noise floor. Based on SNR, modulation and PHY data rate, bit errors of received packet are computed. To calculate interference, signal to interference ratio (SIR) is computed according to the interference model. Castalia defines simple and additive interference models. In former case, simultaneous transmission by two senders always result in a collision at a receiver. In case of later, for simultaneous transmisison by two senders, a collision occurs or the receiver receives stronger signal based on SIR value. Complete simulation parameters are given in **Table 6**.

Table 6. Simulation Parameters

Parameter	Value
Simulation Time	3000 seconds (sec)
Number of Nodes	20
Field Area	40 x 60 meters
Probe Period	100, 300, 500 sec
Traffic Model	CBR
Packet Rate	1 packet/sec
Payload	100 bytes
Routing Protocol	AODV
MAC Protocol	IEEE 802.15.4 (beacon enabled)
Channel Sense	Slotted CSMA-CA
maxCSMABackOffs	1,2,3,4
Physical Model	CC2420 radio
Physical data rate	250 Kbps
Radio Output Power	-5 dbm
Receiver Sensitivity	-95 dbm

Interference Model	Simple Interference Additive Interference
Wireless Channel Model	Log Normal Shadowing
Path Loss Exponent	2.4

5.2.2 Experimental Methodology

The experimental methodology is aimed at validating the protocols operation under normalcy and analyze errors in correlated processes that trigger critical process failure. The performance hypothesis is based on accurate diagnosis of critical failure with minimum overhead. To test this hypothesis, impact of critical errors, warnings and alerts in stacked processes on routing failure is analyzed. We validate all possible root causes of routing failure. For this purpose, outputs produced by LDT and GDT are evaluated by varying channel conditions and interference model.

To evaluate diagnosis communication overhead, packet rate and probe period parameters are varied. For performance comparison, diagnosis communication overhead is compared with Sympathy. Each set of experiments is repeated three times with different random seeds and average values are analyzed.

5.2.3 Error Classifier

The error classification scheme is evaluated by analyzing impact of probe period and radio interference model. Error classification gives an insight into long term (steady state) behavior of WSN under natural and implied faults. The impact of probe period variation can be best described by first understanding the relationship between critical errors, warnings or alerts. The relationship between them is a function of the communication protocols implementations and the exception handling code for error reporting. The spatial-temporal frequency of alerts and warnings with respect to the frequency of critical errors themselves demonstrates the adaptability of the code implemented through timers and counters. Varying the probe period either suppresses or intensifies the short-term manifestations of alerts and warnings through critical errors. An extended probe period simply dilates the effects thereof as shown in **Fig. 12**. In case of shorter probe period of 100 sec, higher number of warnings and alerts are manifested as critical errors. This is due to cumulative effect of warnings and alerts on critical errors over a shorter period. However, for a longer probe period of 500 sec, the network becomes stable as protocols converge. Consequently, the frequency of warnings and alerts is reduced. As a result, smaller number of critical errors are classified.

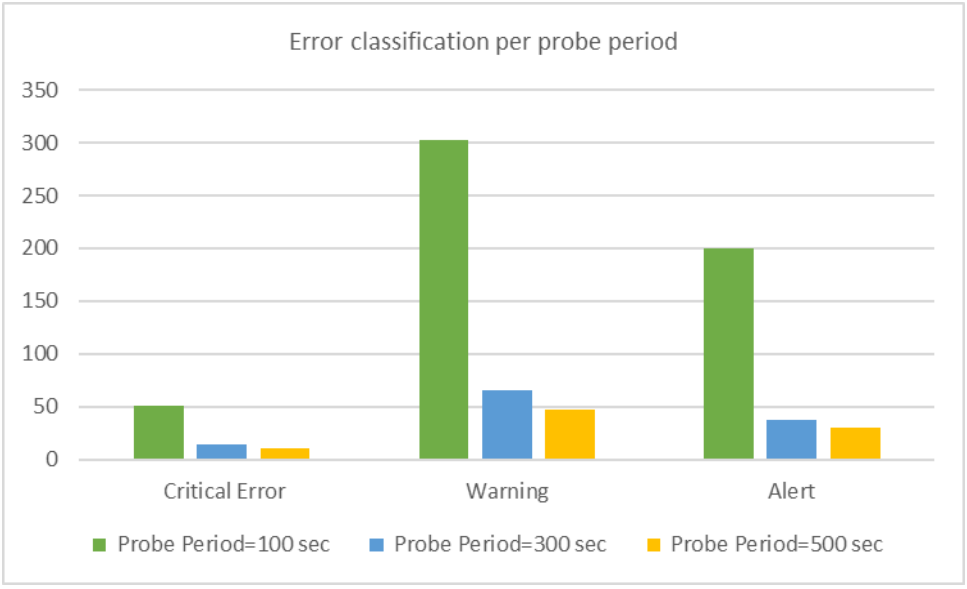

Fig. 12. Network wide behavior shows spatial aspects of critical error-warning-alert classification: higher frequency of warnings for probe period of 100 sec is observed.

Fig. 13 shows that for simple interference model, higher number of warnings and alerts are classified due to collisions. However, in case of additive interference, slightly larger number of critical errors are classified due to high interference caused by additive impact of concurrent signals reception at receiver nodes. Because in this case, transmissions from other nodes are calculated as interference by linearly adding their effect at the receiver.

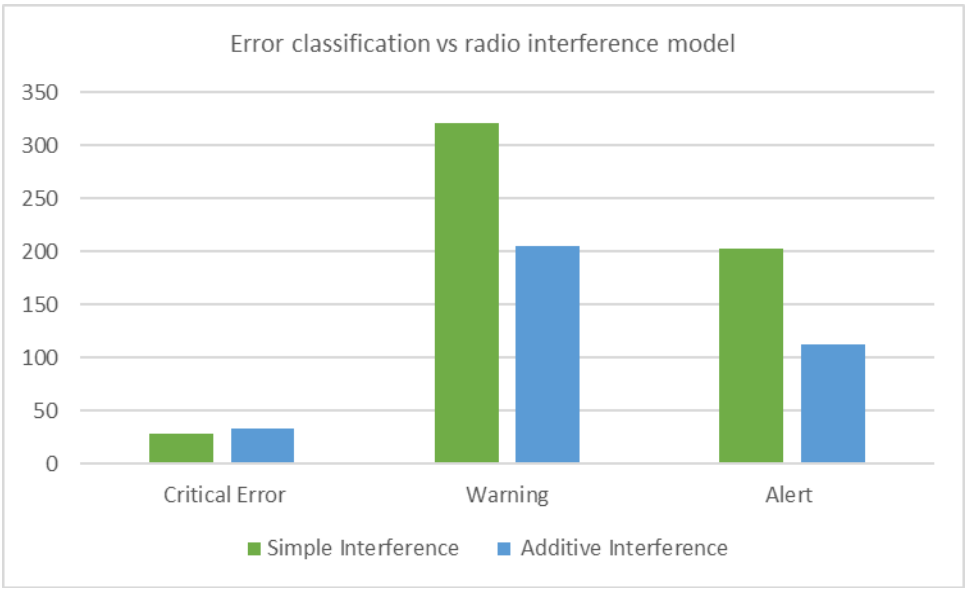

Fig. 13. Impact of radio interference model: higher number of warnings and alerts for simple interference model due to collisions.

5.2.4 Local Decision Tree

LDT is evaluated by analyzing impact of errors in correlated stacked processes on routing failure.

5.2.4.1 Impact of Channel Sense Errors

In this simulation, maxCSMABackOffs parameter is varied to evaluate the impact of errors in *channel sense* process on *route discovery* failure. For smaller value of maxCSMABackOffs, the frequency of errors in *channel sense* increases. According to the inter-process correlation model, errors in *channel sense* process triggers *transmit frame* failure. Due to dissemination of failure effect on network layer, correlated *route discovery* process fails. Accordingly, LDT diagnoses busy channel as primary root cause for maxCSMABackOffs equal to 1, 2, and 3 as shown in **Fig. 14**. However, frequency of synchronization loss due to interference and channel errors is reduced due to simple interference model used. Similarly, for peer routing process failure, LDT infers busy channel and radio non-RX state for CCA as dominating root causes for maxCSMABackOffs equal to 1 as shown in **Fig. 15**. For higher values of maxCSMABackOffs, frequency of channel access error decreases.

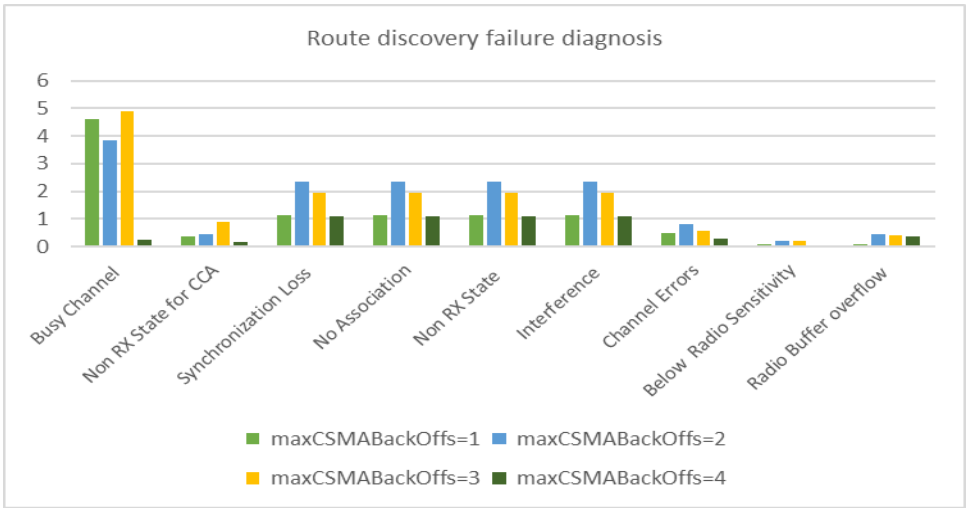

Fig. 14. Impact of errors in channel sense on route discovery failure: LDT diagnoses channel access error due to busy channel as dominating root cause for smaller values of maxCSMABackOffs

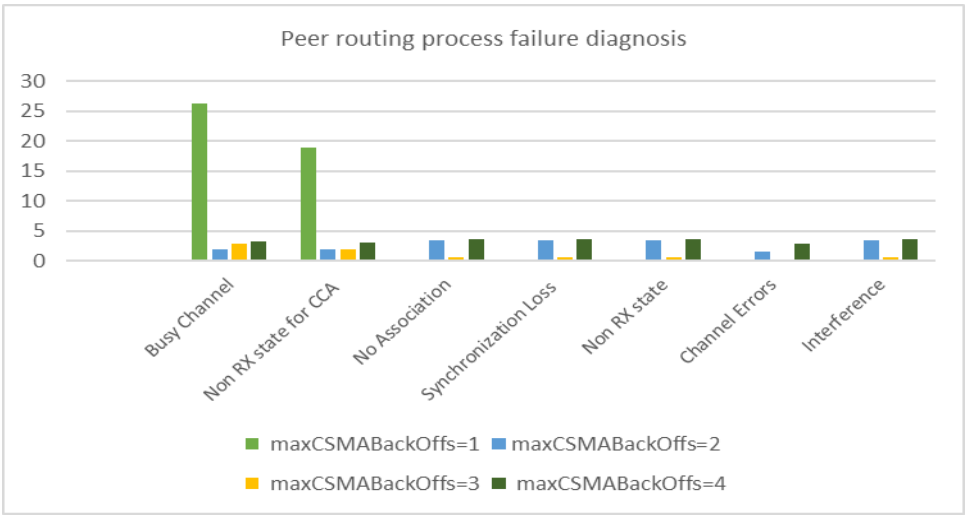

Fig. 15. Impact of channel sense errors on peer routing process failure: LDT infers busy channel and radio non-RX state for CCA as dominating root causes for maxCSMABackOffs equal to 1

5.2.4.2 Impact of Radio Interference Model

In this experiment, radio interference model has been varied while keeping probe period fixed at 100 sec. The *receive data* process fails due to radio non-RX state, low power signal below radio sensitivity, channel errors and interference. According to the inter-process correlation model, upward propagation of failure triggers errors in MAC *receive frame* process causing beacon frame loss. Multiple beacon frame loss results in synchronization loss i.e. a critical error. Consequently, a chain of correlated process failures is initiated i.e., *association* and *transmit frame*. The failure effect is propagated to network layer instigating *route discovery* failure. In case of additive interference model, combined impact of multiple signals reception at the receiver results in high interference. Accordingly, LDT diagnose synchronization loss as primary root cause due to interference and channel errors as shown in **Fig. 16**. Peer routing process failure diagnosis produced similar results as shown in **Fig. 17**. However, in both cases, LDT also infers busy channel as a secondary root cause. This is due to concurrent channel access by multiple nodes either to broadcast RREQ or unicast RREP packets.

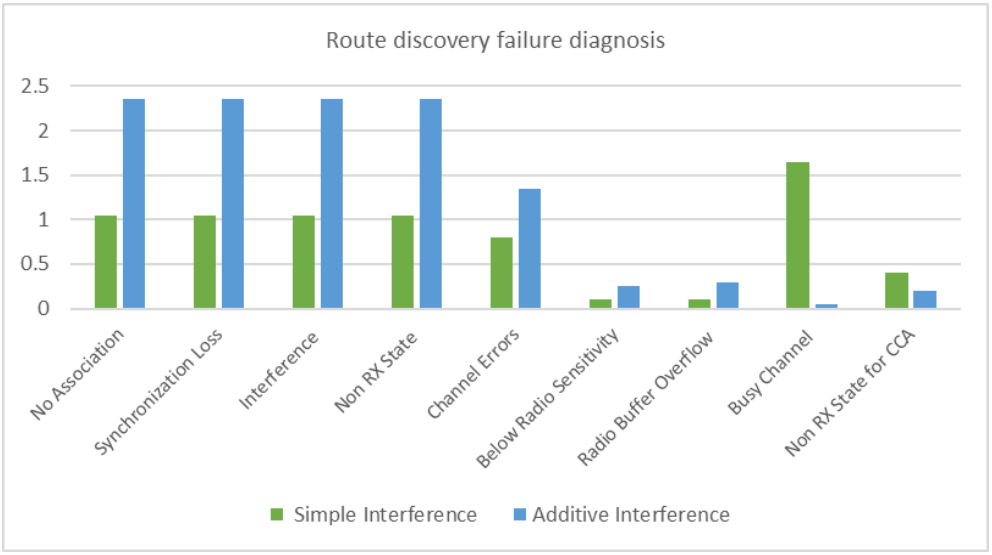

Fig. 16. Route discovery failure diagnosis: LDT infers synchronization loss and no association as primary root causes.

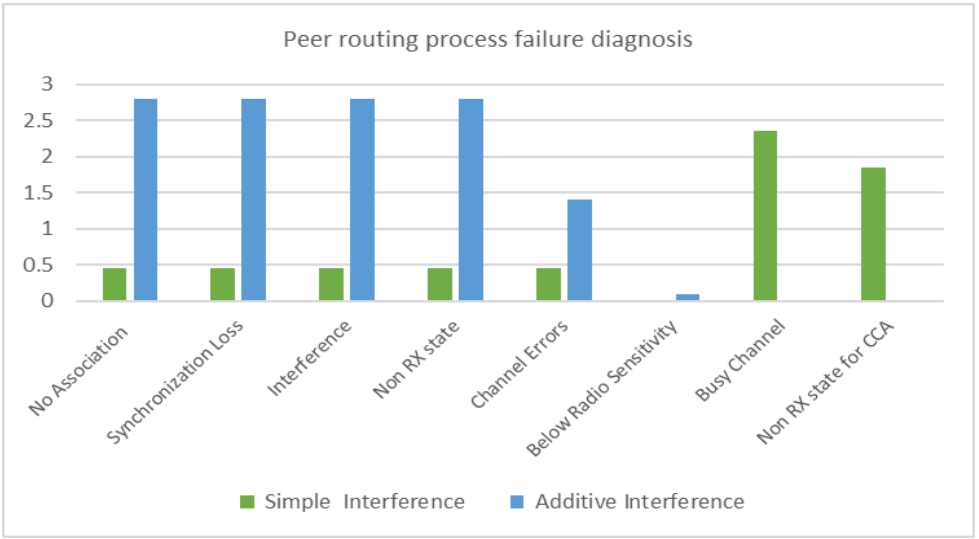

Fig. 17. Peer routing process failure diagnosis: LDT infers synchronization loss and no association as dominant root causes for additive interference model due to high interference.

5.2.5 Global Decision Tree

The output produced by GDT on DCH is evaluated to analyze impact of errors in peer routing processes on *route discovery* failure. This experiment is based on ideal radio communication model. In this scenario, all nodes perfectly receive signals from a transmitter and all communication links are bidirectional within a particular range. In this case, procedural errors in peer routing processes leads to failure. According to the inter-process correlation model, propagation of failure effect on peer layers of protocol stack results in *route discovery* failure. Therefore, external failure source is inferred by LDT on originating nodes. Subsequently, failure reports containing partial diagnosis results are sent to DCH.

Failure reports are collected and parsed by *Report Collector* module on DCH. GDT deduces no reverse route error in *Generate RREP* and *Process RREP* as probable root cause of *route discovery* failure as shown in **Fig. 18**. The potential root cause for this error is corrupt routing table entry for next hop on reverse route to the originating node. In ideal communication model, all links in a particular disk range are bidirectional. However, on a fewer nodes outside the range, RREQ packet reception over a unidirectional link results in error and source of RREQ is placed in a black list. Consequently, *process RREQ* failure occurs. However, GTD infers source-RREQ-blacklist error as a secondary root cause for *route discovery* failure.

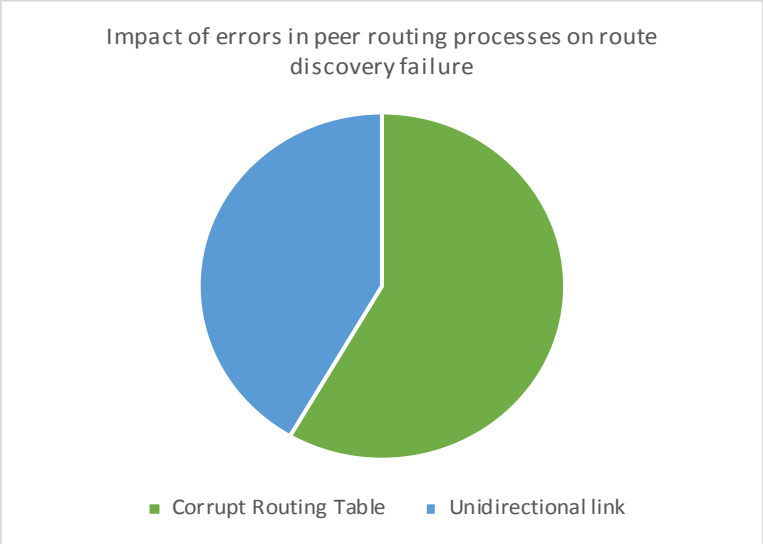

Fig. 18. Impact of errors in peer routing processes: GDA infers corrupt routing table as major source of route discovery failure

5.2.6 Diagnosis Traffic Overhead

To estimate diagnosis traffic overhead, empirical cumulative distribution function (ECDF) is used. Let $(X_1,\ldots,X_n)$ represents a sample of independent random variables. Let F(x) defines the common cumulative distribution function of the sample. Then ECDF is defined as such.

$$F_n(t) = \frac{1}{n}\sum_{i=1}^{n} 1_{x_i \le t} \tag{43}$$

Where the step function $F_n(t)$ at a particular point represents fraction of observations that are less than or equal to the specified value i.e., *t*. ECDF function in MATLAB has been used. Probe period and packet rate have been varied to investigate their impact on diagnosis traffic overhead. In **Fig. 19**, x-axis represents ratio of failure report bytes to overall bytes transmitted. Y-axis denotes corresponding ECDF of the fraction of network traffic consumed for diagnosis communication. Lesser diagnosis traffic overhead is represented by smaller ratios. Considering graph area as a square box, s-curve of ECDF closer to upper left corner represents smaller overhead. In this case, slope of s-curve is high. However, decreasing slope of s-curve represents comparatively higher overhead. Diagnosis traffic overhead depends on network dynamics and periodic probing activity. Overall, the hybrid framework implemented through local and global decision trees significantly reduces diagnosis traffic overhead.

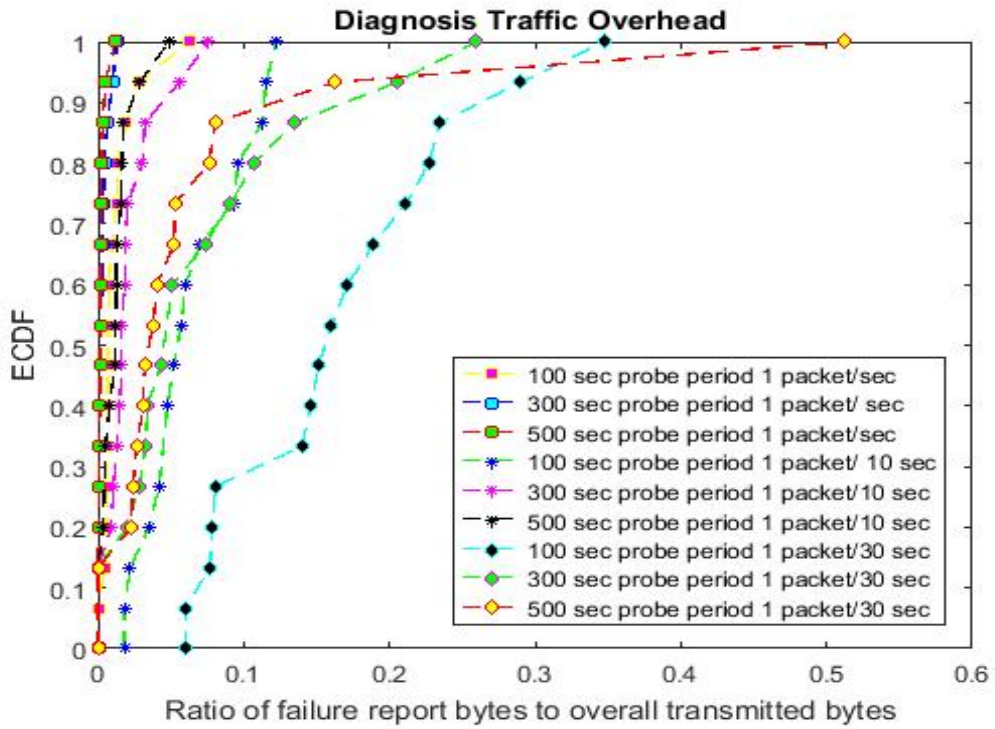

Fig. 19. Diagnosis traffic overhead. Each line represents a particular simulation execution and each point represent fraction of failure report bytes transmitted by a node

5.2.7 Comparison with Related Work

Diagnosis traffic overhead of hybrid framework has been compared with Sympathy [13] that periodically collects metrices from all nodes at sink even in case of no network exception. However, the proposed hybrid framework requires diagnosis communication with DCH only if LDT infers external failure source i.e, a peer process failure. The framework distributes diagnostic workload in a decentralized way that is energy efficient and produces lesser communication overhead. As, more energy is spent in wireless communication than in processing [27]. For overhead comparison, ECDF function in MATLAB has been used. In **Fig. 20**, x-axis represents ratio of diagnosis traffic to overall network traffic and y-axis represents the corresponding ECDF. For example, a point (5, 0.5) on graph indicates that in 50 percent of the time, less than 5 percent of the total traffic is dominated by diagnosis traffic overhead. The probe period has been varied against different values of Sympathy's metric period. As shown in **Fig. 20**, hybrid framework significantly outperforms Sympathy. In all cases, failure reports are transmitted on need basis only, reducing diagnosis communication overhead.

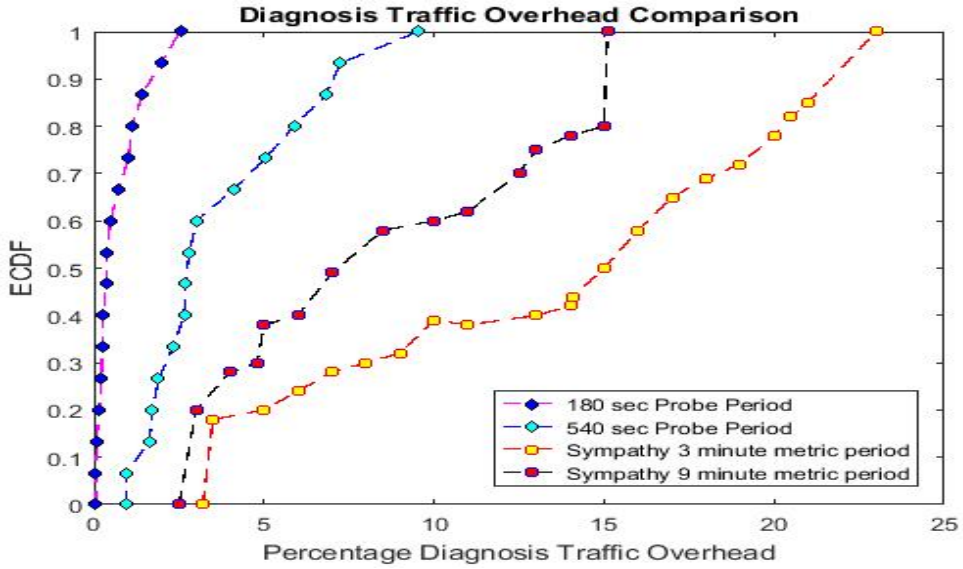

Fig. 20. Comparison of diagnosis traffic overhead with Sympathy for different values of probe period and metric period of Sympathy

6. Conclusion

This work presents an inter-process correlation model based hybrid framework for fault diagnosis in wireless sensor networks. The inter-process correlation model represents dependencies of stacked and peer layer processes on node and network levels. The hybrid framework implements local and global decision trees for inter-process correlation model based fault diagnosis. Using fault tree analysis, the proposed model has been qualitatively analyzed. Simulation results show that network wide fault diagnosis depends upon multiple factors. The first being how fine grained the error classification is, based upon extent and complexity of exception handling code in protocols implementation. Secondly, inter-process correlations of stacked and peer layer processes determine the ability to carry out robust root cause analysis of process failures. Lastly, the hybrid framework implementation through local and global decision trees on sensor nodes and DCH reduces diagnosis communication overhead.

References

[1] Z. Zhang, A. Mehmood, L. Shu, Z. Hou, Y. Zhang, and M. Mukherjee, "A Survey on Fault Diagnosis in Wireless Sensor Networks," *IEEE Access*, vol. 6, pp. 11349–11364, 2018. Article (CrossRef Link)

[2] A. Mahapatro and P. M. Khilar, "Fault Diagnosis in Wireless Sensor Networks: A Survey," *IEEE Commun. Surv. Tutorials*, vol. 15, no. 4, pp. 2000–2026, 2013. Article (CrossRef Link)

[3] H. M. A. Fahmy, "Protocol Stack of WSNs," *Wireless Sensor Networks: Concepts, Applications, Experimentation and Analysis*, pp. 55–69, 2016. Article (CrossRef Link)

[4] R. Chillarege, I. S. Bhandari, J. K. Chaar, M. J. Halliday, B. K. Ray, and D. S. Moebus, "Orthogonal Defect Classification—A Concept for In-Process Measurements," *IEEE Trans. Softw. Eng.*, vol. 18, no. 11, pp. 943–956, 1992. Article (CrossRef Link)

[5] D. Pediaditakis and Y. Tselishchev, "Performance and Scalability Evaluation of the Castalia Wireless Sensor Network Simulator," in *Proc. of the 3rd International ICST Conference on*

Simulation Tools and Techniques., p. 53:1–53:6, 2010. Article (CrossRef Link)

[6] W. Gong, K. Liu, and Y. Liu, "Directional Diagnosis for Wireless Sensor Networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 5, pp. 1290–1300, 2015. Article (CrossRef Link)

[7] C. OBner, E. Buchmann, and K. Bohm, "Identifying defective nodes in wireless sensor networks," *Distrib. Parallel Databases*, vol. 34, no. 4, pp. 591–610, 2016. Article (CrossRef Link)

[8] A. Mahapatro and P. Khilar, "Online fault diagnosis of wireless sensor networks," *Open Comput. Sci.*, vol. 4, no. 1, pp. 30–44, 2014. Article (CrossRef Link)

[9] M. Y. M. Sabrigiriraj, "Fault detection and recovery scheme for routing and lifetime enhancement in WSN," *Wirel. Networks*, vol. 23, no. 1, pp. 267–277, 2017. Article (CrossRef Link)

[10] J. Wu, D. Duh, T. Wang, and L. Chang, "Fast and Simple On-Line Sensor Fault Detection Scheme," in *Proc. of International Conference on Embedded and Ubiquitous*, pp. 444–455, 2007. Article (CrossRef Link)

[11] R. R. Swain, P. M. Khilar, and S. K. Bhoi, "Heterogeneous fault diagnosis for wireless sensor networks," *Ad Hoc Networks*, vol. 69, pp. 15–37, 2018. Article (CrossRef Link)

[12] A. Munir, J. Antoon, and A. Gordon-Ross, "Modeling and Analysis of Fault Detection and Fault Tolerance in Wireless Sensor Networks," *ACM Trans. Embed. Comput. Syst.*, vol. 14, no. 1, pp. 1–43, 2015. Article (CrossRef Link)

[13] N. Ramanathan, K. Chang, R. Kapur, L. Girod, E. Kohler, and D. Estrin, "Sympathy for the sensor network debugger," in *Proc. of the 3rd international conference on Embedded networked sensor systems - SenSys '05*, pp. 255–267, 2005. Article (CrossRef Link)

[14] S. Tennina, O. Gaddour, A. Koubâa, F. Royo, M. Alves, and M. Abid, "Z-Monitor: A protocol analyzer for IEEE 802.15.4-based low-power wireless networks," *Comput. Networks*, vol. 95, pp. 77–96, 2016. Article (CrossRef Link)

[15] D. Rodenas-Herraiz, P. R. A. Fidler, T. Feng, X. Xu, S. Nawaz, and K. Soga, "A handheld diagnostic system for 6LoWPAN networks," in *Proc. of 2017 13th Annual Conference on Wireless On-Demand Network Systems and Services, WONS 2017*, pp. 104–111, 2017. Article (CrossRef Link)

[16] M. Ringwald, R. Kay, and A. Vitaletti, "SNIF: Sensor Network Inspection Framework," *Technical Report 535*, 2011. Article(CrossRefLink)

[17] I. G. Siqueira, L. B. Ruiz, and A. a. F. Loureiro, "Coverage area management for wireless sensor networks," *Int. J. Netw. Manag.*, no. October 2005, pp. 17–31, 2014. Article (CrossRef Link)

[18] Y. J. Kim, S. Song, and D. Kim, "HDF: Hybrid debugging framework for distributed network environments," *ETRI J.*, vol. 39, no. 2, pp. 222–233, 2017. Article (CrossRef Link)

[19] A. Zafar, B. Wajid, and B. A. Akram, "A Hybrid Fault Diagnosis Architecture for Wireless Sensor Networks," in *Proc. of International Conference on Open Source Systems & Technologies (ICOSST)*, pp. 7–15, 2015. Article (CrossRef Link)

[20] IEEE Standards Associations Official Website, "802.15.4-2015 - IEEE Standard for Low-Rate Wireless Networks," Web: http://standards.ieee.org/findstds/standard/802.15.4-2015.html. Article (CrossRef Link)

[21] C. E. Perkins and E. M. Royer, "Ad-hoc on-demand distance vector routing," in *Proc. of - WMCSA'99: 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pp. 90–100, 1999. Article (CrossRef Link)

[22] D. Rodopoulos *et al.*, "Classification Framework for Analysis and Modeling of Physically Induced Reliability Violations," *ACM Comput. Surv.*, vol. 47, no. 3, pp. 1–33, 2015. Article (CrossRef Link)

[23] S. Kabir, "An overview of fault tree analysis and its application in model based dependability analysis,"*Expert Syst. Appl.*, vol. 77, pp. 114–135, 2017. Article (CrossRef Link)

[24] E. E. Hurdle, L. M. Bartlett, and J. D. Andrews, "System fault diagnostics using fault tree analysis," in *Proc. of Inst. Mech. Eng. Part O J. Risk Reliab.*, vol. 221, no. 1, pp. 43–55, 2008. Article (CrossRef Link)

[25] E. Ruijters and M. Stoelinga, "Fault tree analysis: A survey of the state-of-the-art in modeling, analysis and tools," *Comput. Sci. Rev.*, vol. 15, pp. 29–62, 2015. Article (CrossRef Link)

[26] S. Farooq, A. H. Akbar, A. Raza, A. Waheed, and K. H. Kim, "Application-oriented Re-Clustering and Cluster Head Re-Election scheme for Wireless Sensor Networks," in *Proc. of IWCMC 2011 - 7th International Wireless Communications and Mobile Computing Conference*, pp. 1087–1092, 2011. Article (CrossRef Link)

[27] Y. Zhang, N. Dragoni, and J. Wang, "A Framework and Classification for Fault Detection Approaches in Wireless Sensor Networks with an Energy Efficiency Perspective," *Int. J. Distrib. Sens. Networks*, vol. 2015, no. 1, 2015. Article (CrossRef Link)

Amna Zafar received B.Sc (Hons.) in Computer Science and M.Sc degree in Computer Science from University of Engineering and Technology (UET) Lahore, in 2004 and 2011 respectively. Currently, she is a PhD scholar in Computer Science at UET Lahore. From 2004 to 2005, she worked as Research Assistant at Al-Khawarzmi Institute of Computer Science, UET Lahore. From Dec 2005 to Oct 2006, she worked in a Govt. organization as IT teacher. She joined UET Lahore as a lecturer in Nov 2006. Since 2012, she has been working as an Assistant Professor with the Department of Computer Science and Engineering, UET Lahore. Her research interests include wireless sesnor networks, IoT, WBAN and machine learning

Ali H. Akbar received the bachelor's degree (Hons.) in telecommunications from NUST, Pakistan, in 1997, the M.S. degree from UNSW Australia in 1999, and the Ph.D. degree from Ajou University in 2008. He is currently an Associate Professor with the University of Engineering and Technology at Lahore, Lahore, Pakistan. His topics of interest include wireless networks, such as MANETs and sensor network, M2M networks, and distributed systems. He was a consultant with the Al-Khawarizmi Institute of Computer Science for government organizations, such as Lahore Transport Company and RESCUE 1122.

Beenish A. Akram received the B.Sc. degree (Hons.) in Computer Engineering and the M.Sc. degree in Computer Science from University of Engineering and Technology (UET) Lahore, in 2006 and 2010, respectively. She is currently pursuing the Ph.D degree in computer science from UET Lahore. From 2006 to 2007, she was a Software Design Engineer with MicroTech Industries (Pvt.) Ltd., Lahore. From 2007 to 2012, she was a Lecturer with UET Lahore. Since 2012, she has been an Assistant Professor with the Department of Computer Science and Engineering, UET Lahore. Her research interests include machine learning, IoT, and indoor localization and embedded systems.