On-line Fault Diagnosis for Distributed Communication Networks

[1]Arunanshu Mahapatro, [2]Pabitra Mohan Khilar
Department of Comp. Sc. & Engg, NIT, Rourkela, India, Pin-769008
Email: [1]arun227@gmail.com, [2]khilarpm@yahoo.com,

ABSTRACT

This paper proposes an on-line two phase (TPD) fault diagnosis algorithm for distributed communication networks. Intermediate nodes communicate the messages between different source destination pairs. The algorithm addresses a realistic fault model considering crash and value faults in the nodes. The algorithm is shown to produce a time complexity of O(1) and message complexity of O(n.e) respectively. The results such as diagnostic latency, message complexity and energy consumption shows that the proposed diagnosis algorithm is feasible for design of different fault tolerant wired and wireless communication networks.

Key Words: On-line diagnosis, two phase diagnosis, value faults, dynamic fault environment

I. INTRODUCTION

The distributed communication networks such as mobile ad hoc network, sensor network etc, are becoming popular due to their extensive use in social, commercial and scientific applications. As the computing power and the physical complexity of these systems increase, they become more susceptible to component failures. Incorporating correct and timely fault diagnosis capability to the system is essential to improve the system reliability and availability. An important problem known as distributed self-diagnosis provides software based diagnosis solution for achieving fault tolerance without using redundancy. In distributed self-diagnosis, every node in the distributed communication networks needs to record the status of all other nodes.

Motivated by the need of an on-line distributed diagnosis, we develop and evaluate a two-phase fault diagnosis and recovery strategies in a dynamic fault environment i.e., the nodes may change their status during execution of the diagnosis procedure. The diagnostic latency and message complexity is used as the performance measure in order to evaluate the proposed fault diagnosis algorithm. The paper outline as follows: Section 2 presents the related work on distributed diagnosis. The system and fault model is specified in section 3. The TPD along with its formal analysis has been described in section 4 followed by the simulation results in section 5. Finally the section 6 concludes the paper.

II. RELATED WORKS

An excellent survey on fault diagnosis has appeared by authors Barborak, Malek and Dahbura in paper [1]. Most classical approaches assume permanent or crash fault model and regular network topologies [2]. Previous work in this field has almost exclusively dealt with static fault situations, i.e., status of the nodes (faulty or fault free) does not change during execution of the diagnosis procedure. These assumptions are neither realistic nor suitable for on-line environment where nodes may fail and recover during execution of the diagnosis procedure. Recently, Subbiah and Blough [4] propose one instance of distributed diagnosis in dynamic fault situation both for completely and not completely connected systems but assumes traditional crash fault model. The algorithms considering a combination of crash and value failures for fully connected network has been proposed by authors C.J.Walter et. el., in [3]. Very small number handles a combination of crash and value failures for arbitrary network topologies. These algorithms rely on a particular testing strategy such as test based or heartbeat based. Since tests are difficult to obtain in practice, various comparison-based approaches have been proposed. Tasks are duplicated on multiple processors and their results are compared to identify faulty ones [5]. The comparison-based approach is believed to be most practical diagnosis strategy for system diagnosis. The heartbeat-based schemes are of low cost usually few bytes and needs a single message transmission to achieve diagnosis between a pair of nodes. Here, in this paper an algorithm Two Phase Diagnosis (TPD) has been proposed which considers a combined heartbeat and comparison testing strategy under more realistic fault model for arbitrary network topologies. The author has also proposed a number of fault diagnosis algorithms and energy efficient fault diagnosis protocols for distributed wireless and wired networks suitable for general purpose and real-time wireless and wired distributed communication networks [6-10].

III. SYSTEM AND FAULT MODEL

A System Model

The communication network is assumed to be error-free, and deliver messages reliably. We consider a round-based communication model, which implies that periodically, i.e. , at the period boundaries, messages are sent by system nodes. A synchronous, message-passing framework is assumed where the communication delay is bounded. All processors execute the same workload (For example temperature sensing from the environment) and determine the output value V_i. This value is communicated to all other nodes. An arbitrary network with connectivity k has been assumed where a node issues a single heartbeat message, which is broadcast to all nodes via intermediate nodes. Every node is assigned with a node-id, and can detect the absence or time deviance for an expected message. Nodes may be faulty and links are fault-free The transmission time t_x, is the time between a node initiating a communication and the last bit of the message being injected into the network. The minimum and maximum propagation delays for delivering all of the message bits between any two neighboring nodes are denoted by $t_{pro\text{-}min}$ and $t_{pro\text{-}max}$, respectively.

B Fault Model

We consider crash and value faults in nodes. Links are assumed to be fault free. The network delivers messages reliably. A faulty node performs its computation like a non-faulty node but it may fail to send its value (crash fault) or it may send an erroneous value to another node (value fault). For example, a faulty mobile in adhoc network may be faulty due to it is switched off, out of range or an adversary node has corrupted the header of the message. The combined heartbeat and comparison based testing mechanism is followed to detect the faults in nodes. A state machine with two states, failed (1) and working (0) models the status of a node. The node subjected to crash fault simply ceases functioning without sending any message and the failed nodes, which may send valid but incorrect result to the receiver, indicate the value fault. The *state holding time* is the minimum time that a node remains in one state before transitioning to the other state.

IV. THE ALGORITHM

A Terminology

We consider n as the number of nodes in the system and $mess_y$ representing a message sent by node Y. We consider two fault categories: 1) The set of *missing messages,* are those messages which X believes Y failed to issue during round m, and 2) The set of *improper logical messages*, are those messages which are correctly delivered but disagree with V_X, the result of X's own voting process on inputs received. The syndrome Syndrome(Y), $\forall$ X, Y represents the union of test outcomes due to improper logical message and missing message. $Syndrome^m(Y)$ is represented in vector form for each value of X, with vector entries corresponding to all Y values from which X receives messages. The vector entry corresponding to any node Y is a binary input: 0 corresponds to a fault-free input received from Y as perceived by X, and 1 represents a fault being perceived by X. Each node maintains its perception of the system state using a system level error report in the form of a vector, $F_X{}^m(Y)$ (as received by X from node Y over round m), consisting of an ordered quadruple < X,Y,m, $Syndrome_X{}^m(Y)$ >. The function $F^m_{tot}(Y) = |U_X \in n, {}_{X \neq Y} F_X{}^m(Y)|$ is used to count the number of accusations on a processor Y by all other processors during period m. Thus $F^m_{tot}(Y)$ is an integer where $0 \leq F^m_{tot}(Y) \leq (n-1)$.

B Algorithm Assumptions

For arbitrarily connected network, a restricted case is considered where fewer than k nodes are in the failed state at any given instant of time so that the network remains connected. Each node periodically executes the diagnostic workload and initiates a round of message transmissions. Assume an arbitrary node X initiates a round of heartbeat transmissions at real time t and remains in the working state indefinitely afterward. X will initiate another round of heartbeat transmissions no later than real time $t+(1+\rho)\pi$, where π is the heartbeat period and ρ is the maximum drift rate of the clock with $\rho << 1$. The message can be quite sophisticated and can encapsulate within it information about the health of the node being monitored. A message consists of the following fields (1) Node_id: The ID of the node that initiated the message, (2) Seq_no: The sequence number of the message, (3) Val: The value associated with message, (4) Delay: Heartbeat delay stored in the buffer of a node. The Seq_no field serves to distinguish successive messages from the same node. The sequence number is incremented by 1 during every period starting from sequence number 0. When the maximum sequence number (MAX_SEQ_NUM) that can be stored in this field is exceeded, the sequence number is wrapped around to 0. Thus it is used as a logical sequence number. The Node_id and Seq-no fields together identify the message. Val field is used to check for value fault in the received message. Sequence numbering has been employed to differentiate the old diagnostic messages from the most up-to-date versions of them. Delay field stores the minimum time a heartbeat is stored at a particular node. Each node maintains an array of n entries, where n is the total number of nodes in the network. The i^{th} entry in array stored in node X contains: (1) Status: Node X's view of node i. (state 0 or 1), (2) Last_seq_no: The Seq_no field of the latest heartbeat received from node i by node X.

C The Proposed Algorithm (TPD)

Out of the two phases of the algorithm, during the first phase each node periodically execute the diagnostic work load and initiates the results by a round of message transmissions to all other nodes. A node detects a crash or missing message fault without receiving a heartbeat for duration T_{time_out}. If the message delivery and it's arrival at a receiving node is valid but incorrect (i.e., value does not match with it's own computed value), the message is recorded as *improper logical message* and the node is *value faulty*. This phase of diagnosis we call local diagnosis phase. In this phase a node identifies other node's validity by comparing it's own message with that of received message. The comparison need not seek for an exact value in the message rather can choose to consider range or deviance check. If the received message is well within the range of it's own value, it accepts as a correct message otherwise records as incorrect message. For example, in a mobile adhoc network, a faulty mobile node may not respond due to either physical damage or out of range. An adversary node may also send an erroneous message in its header, which may not be detectable during this phase. We show that, these faults are detected by the second phase of diagnosis. In the second phase these local results available at each node are further exchanged with other nodes and a counter maintained at every node is incremented by one for every positive diagnosis. If the counter value at a particular node for another node is greater than half of the nodes, it means that more number of nodes detected that node as faulty and all other nodes that recorded this event as fault free is accused as faulty. If the accusation against a node is recorded as faulty in the previous round, this node is considered as faulty in the current round. This phase of diagnosis is called global diagnosis phase and mainly used for accurate diagnosis of the system. Both the phases of two-phase diagnosis procedure are executed in a pipelined manner to improve diagnostic latency.

D Basic Analysis of Algorithm

The formal analysis of algorithm involves satisfying the two important properties as follows: (P1): Correctness: every node diagnosed to be faulty by a non-faulty node is indeed faulty (P2): Completeness: Every faulty node is identified. First, we consider correctness, which states that if a good processor accuses some other processor, the accused processor is indeed faulty.

Lemma 1. If "X" is good, then $F_X{}^m(Y) \neq 0$, implies that "Y" is faulty.

Theorem 1. (Correctness). If a good "X" declares "Y" faulty, „Y" is indeed faulty.

Theorem 2. (Completeness). If "Y" is faulty, then all good nodes diagnose "Y" as faulty.

The proofs for the lemma 1, theorem 1 and 2 is given in Appendix. The upper bound for recovery event and failure detection event is computed as follows. First we compute the upper bound between a node initiating a heartbeat and a heartbeat with the same or a higher sequence number being received by all other nodes is $D_{maxn} = d(k-1)(n-1)t_x + (n+k-2)(t_x + t_{prop\text{-}max} + c) - (k-1)\varepsilon$, if the failed state holding time, SHT_f is at least D_{maxn} where, d is the maximum number of neighbors of any node, k is connectivity of the network, n is number of nodes, ε = is a small positive constant and is the smallest time such that a heartbeat send in $t_x - \varepsilon$ time is an invalid heartbeat and c is the comparison time between received value and the node's own computed value. D_{maxn} is the theoretical latency in detecting a recovery event and was derived by taking a particular sequence of events to represent an worst case scenario. A crash or missing message fault is detected by time out. Upon receipt of a new heartbeat, the period of time a node that is continuously in the working state waits for the next new heartbeat before detecting a fault event, called the timeout period $T_{timeout}$ is $(1+2\rho)\pi + (1+\rho)(D_{maxn} -$ heartbeat.delay) where heartbeat.delay is the value in the delay field of the last heartbeat received. Due to the existence of multiple paths between nodes, it is possible that a node times out thereby detecting a fault event and then receives a stale heartbeat. Arrival of this stale heartbeat does not indicate a recovery event. Hence, when nodes time out, they discard any heartbeats they may receive from the node just diagnosed to be faulty for a predetermined amount of time T_{exist}. T_{exist} refers to the maximum possible time the heartbeat can exist in the network and is $= (1+2\rho)\pi + (1+\rho)\, D_{maxn} + n(t_{pro\text{-}max} - t_{pro\text{-}min} + c)$. A heartbeat is said to exist in the network if either the heartbeat is propagating in the network or the heartbeat is stored in some node's buffer. The value of T_{exist} is the theoretical failure detection latency. A node needs to stay for large enough time in the failed state such that all other fault free nodes detect it's state. So after T_{out} a node keeps on rejecting the heartbeat for a maximum amount of time called heartbeat rejection time $T_{reject} = 2\rho m + 2\rho D_{maxn} + n(1+\rho)(t_{pro\text{-}max} - t_{pro\text{-}min} + c)$. The heartbeat rejection time is the period of time a node X discards heartbeats from a node Y after diagnosing node Y to be faulty. Since arrival of a stale heartbeat is not a recovery event, for a genuine recovery event to be detected, the failed state holding time should be made sufficiently large to guarantee that no new heartbeat message arrives during the rejection time. The failed state holding time, denoted by SHT_f, is the minimum time a node remains in the failed state before transitioning to the working state. The failed state holding time (SHT_f) is $T_{exist} + (1+\rho)\, T_{reject} - t_x$. The derived expressions are closely related with the expressions that appears in paper [4]. It can be observed that T_{exist} is the theoretical latency in detecting a failure event.

V. SIMULATION RESULTS

, respectively, for all simulations. Simulations were performed on networks of sizes 8, 16, 32, 64, 128, and 256 using discrete event simulation techniques. The dynamic nature of the system was modeled using a Poisson process. When an event occurs on a node, the next event time is computed by adding the state holding time and the time as given by the poison process. In all simulations, the minimum state holding times for both failed and working states were set to the failed state holding time.

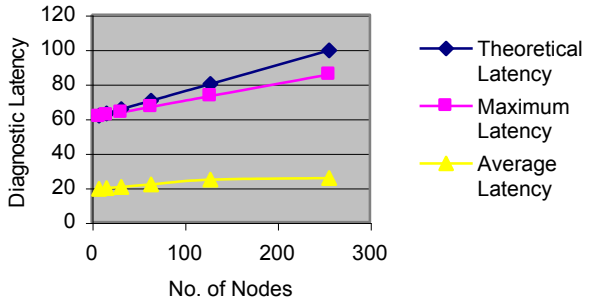

Figure 1. Diagnostic Latency versus n.

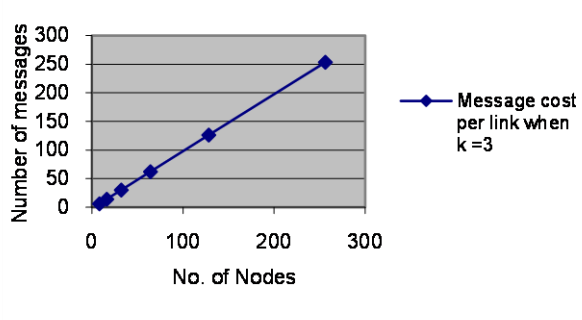

Fig. 2 Number of messages per link versus n for different values of connectivity

A

Diagnostic Latency and Message Complexity

Fig. 1 shows the theoretical diagnostic latency i.e., in detecting either a failure or recovery event which ever is higher. The diagnostic latency is variable due to the difference in the actual propagation time of the last heartbeat and the delay tracked by the intermediate nodes. Hence, the more links that are traversed by the last heartbeat that is sent by a node before it could fail, the more varied the latency will be. Fig 2 shows the number of messages exchanged per heartbeat period per link in an n-node e-link network. The number of messages increases linearly with n which shows the algorithm is linearly scalable. The message complexity is O(n.e).

Algorithm was simulated on randomly generated networks. t_x, $t_{pro\text{-}min}$, $t_{pro\text{-}max}$, c, Poisson mean, ρ ,k and π were kept fixed at 0.002, 0.008, 0.08, 0.05, 1, 0.001,3 and 40 seconds

VI. CONCLUSION

The paper proposed an on-line two-phase diagnosis algorithm considering a realistic fault model based on monitoring the system message traffic generated due to execution of normal workload. The algorithm propagates status information as quickly and through as many redundant paths as possible to allow it to effectively handle dynamic situations. All nodes can change state at the same time or a cascade of status changes can occur, while maintaining a notion of correctness at all times. The proposed algorithm TPD has been able to handle these behaviors effectively in arbitrary-connected networks with diagnostic latency of O(1) and message complexity of (n.e). Our future works include the validation of results using network life time as the main parameter which will address the resource constraint in wireless adhoc network.

REFERENCES

[1] M. Barborak, Malek, and Dahbura, " The Consensus Problem in Fault-Tolerant Computing", ACM Computing Surveys, vol. 25, pp. 171-220, June 1993.

[2] P.M.Khilar, " Algorithms For Distributed System Level Diagnosis ", M.Tech Thesis, 1999, NIT, Rourkela, India

[3] C.J.Walter et al., " Formally Verified On-Line Diagnosis, IEEE Trans. Software Engg., vol. 23, no. 11, pp. 684-721, Nov. 1997.

[4] A. Subbiah et al., "Distributed Diagnosis in Dynamic Fault Environments ", IEEE Trans on Parallel and Distributed System, Vol 15, No. 5, May 2004.

[5] D.M. Blough et al, " The Broadcast Comparison Model for On-Line Fault Diagnosis in Multicomputer Systems: Theory and Implementation ", IEEE Transactions on Computers, Vol. 48, 1999.

[6] P.M.Khilar "Algorithms for Fault Diagnosis in Wireless Adhoc Networks and Distributed Embedded Systems", 2009, Ph.D Thesis, IIT, Kharagpur, India.

[7] P.M.Khilar and S.Mahapatra, "A Novel Hierarchical Clustering Approach For Diagnosing Large-Scale Wireless Adhoc Systems" International Journal of Computer and Applications, Vol. 31, No.4, 2009, pp. 260-267.

[8] M. N. Sahoo and P. M. Khilar, "Survivability of IEEE 802.11 Wireless LAN against Access Point Failure", International Journal of Applications in Engineering, Technology and Sciences, Vol 1, No. 2, 2009, pp. 424-428.

[9] M. Panda and P.M.Khilar**,** "Energy Efficient Search in Dense Wireless Sensor Networks", In proc. of International Conference on Computational Intelligence and Communication Networks (CICN – 2010), Bhopal, India, November 12-15, 2010.

[10] M. Panda, P.M.Khilar, T.Panigrahi and G.Panda, "Learning with Distributed Data in Wireless Sensor Networks", In the proc. of First International Conference on Parallel, Distributed and Grid Computing, (PDGC-2010), Jaypee University of Information Technology, Waknaghat, Solan (H.P), India, October. 28-30, 2010, pp.241-245.

APPENDIX

Lemma 1. If "X" is good, then $F_X{}^m(Y) \neq 0$, implies that "Y" is faulty.

Proof: We prove this by complete induction on the number of rounds of algorithm we use. If $F_X^m(Y) \neq 0$, then some message sent from Y to X was bad. If the message was simply missing or was value faulty, then Y is in fact bad. The two remaining reasons that X would record an accusation of Y is if Y failed to accuse some bad processor, or if Y accused some good processor in an earlier frame. By induction we can assume that the previous majority votes on the fault status of other processors are correct. Since all faults are assumed to be symmetrically dispersed, if Y failed to accuse a bad processor Z, then Y is bad, as all processors including Y must have received the same bad message from Z. If Y accused a good processor Z, then Y must be bad since no majority of good processors received a bad message from Z.

Theorem 1. (Correctness). If a good "X" declares "Y" faulty, "Y" is indeed faulty.

Proof. We prove this by complete induction on the number of rounds used by algorithm. The only situation in the algorithm that a good node could declare another node faulty is when $F_{tot}^m(Y) \geq \lceil n/2 \rceil$. This can arise only if over half of all nodes send an accusation to X. Since we assume that over half of all nodes are nonfaulty, some good node Z must have contributed by sending an accusation of Y to X. By Lemma 1, we thus know that Y is indeed faulty.

Theorem 2. (Completeness). If "Y" is faulty, then all good nodes diagnose "Y" as faulty.

Proof. We prove this by complete induction on the number of rounds used in algorithm. For round m, we assume that for smaller number of rounds this property holds. If Y is faulty in earlier rounds then by induction we can assume that X declared Y faulty earlier, and thus since declarations are monotone, X declares Y faulty. If Y is faulty in this round, it must be either benign faulty, in which case all messages are benign, or symmetric-value faulty. If the symmetric-value faulty nodes produces a good value then it is not a fault. Thus even symmetric-value faulty nodes must produce bad values. In either case all good processors will accuse Y of being faulty, sending $F_X^m(Y)$ to all other processors. Since we assume there are a majority of good processors, there will be a majority of votes to condemn Y, and all good processors will declare Y faulty.