Locating Faults in a Systematic Manner in a Large Heterogeneous Network

Sanjoy Paul
AT&T Bell Laboratories
Holmdel, NJ 07733

Raymond E. Miller
Department of Computer Science
University of Maryland
College Park, MD 20742

Abstract

In this paper, we present several techniques for locating faults in a network which is modeled as an undirected graph. The novelty of these techniques lies in combining the process of collecting status information (from each network element) with the process of locating a fault, which is completely different from the existing techniques in which fault location begins after all the status information has been collected. The techniques presented in this paper have different characteristics and hence they are applicable in different networking environments. However, an intelligent combination of the techniques can be used in a large scale heterogeneous network in an efficient manner for locating faults.

1 Introduction

Fault management is a very important aspect of network management in which the network manager tries to determine if there is a fault in the network and if so, tries to locate the fault. Although there has been a tremendous amount of work in the area of network management, it was not until recently that some of these problems have been stated formally and solved to a certain extent [BHS90][RH92][WS93a][WS93b]. The fault management problem can be formulated in different ways. For example, [WS93a] modeled each layer of the network as a finite state machine running a protocol for the corresponding layer and modeled the network manager as a collection of much simpler finite state machines which observe the interactions of the protocol with the environment. Based on this observation, the network manager concludes if the protocol is working properly or if there is a fault in the behavior of the protocol. [WS93b] however does a different modeling for the problem in which they model the whole network as an undirected graph with nodes and edges representing hosts and transmission links respectively. They assume that the network manager has information available in the form of reachability of nodes from a designated node. Based on this information, the problem of fault location reduces to ranking the links in a descending order of probability of failure.

[DAP93] also has a similar formulation of the problem except that they assume that each node (which can be a host, a router or a gateway) sends its status periodically to a DCD (data collecting device) and based on that information, the Network Manager (NM) tries to conclude if any of the nodes and/or links is down at any time. In this paper, we represent the network as an undirected graph exactly like [WS93b], [DAP93]. However, we make the network manager more intelligent in the sense that it can exert control on the devices in the way they send their status information to the network manager. This is different from both [WS93b] and [DAP93] because they start with the assumption that the status information is available right when the network manager starts the algorithm for the location of faults while we start with the assumption that no status information is available to the network manager when it starts to diagnose the fault but it tries to locate the fault as it keeps collecting the status information from the various network elements. This paper is organized as follows. We discuss the model of the network in section 2 followed by the description of a multicastrating approach, an investigator tour approach, a hybrid approach and a combined approach for fault location in section 3 and the conclusion in section 4.

2 Model of the Network

We assume in this paper that there are two types of network elements in any network, namely, active (or processing) elements and passive (or propagating) elements. Active elements are those which have computing power and the capability to send their own status, for example, the hosts, bridges, routers, gateways and any other component with the ability of sending and receiving information. Passive components are those which do not have computing power or the capability of sending their own status, for example, the transmission lines. Based on this assumption, a network, small (LAN) or large (WAN) or in between (MAN),

\footnote{The terms down, dead and faulty are used as synonyms and so are the terms up and live.}
can be represented by an undirected graph \( G = (V, E) \) with a set of vertices \( V \) which correspond to the active elements and a set of edges \( E \) which correspond to the passive elements connecting the active ones. Thus, an edge in the graph \( G \) is not necessarily a single physical link but may be a sequence of physical links (Fig. 1).

![Figure 1: Model of the Network](image1)

With this model in mind, a fault in the network corresponds to the failure of a node or an edge. When such a fault occurs, alarms may be sounded to inform the network manager of a malfunction in the network. Then it is the responsibility of the network manager to locate the fault. In this paper, we present several approaches that the network manager can take to solve the problem. We also assume that the network manager is running on top of a reliable transport protocol like TCP [C90] so that it does not have to worry about retransmissions and/or timeouts. We assume that the network manager resides at one of the nodes of the graph.

3 Techniques to locate faults

In this section, we present two basic approaches to solving the fault location problem followed by a hybrid scheme which combines the notions from the basic schemes and finally we show how the two basic schemes can be used in a complementary role in a large heterogeneous network. The first basic scheme is called the **multicasting** scheme and it is described next.

3.1 Multicasting Scheme

In this scheme, the network manager requests the lower layer to set up a multicast tree (which can be a virtual connection in case of ATM networks and a strict source route in case of IP networks) rooted at the network manager node, which is the source, to all the other nodes, which are destinations (Fig. 2). Thus, some of these nodes are at distance one from the source (that is, there is an edge in the tree connecting the source and the node in question), some are at distance two (that is, there are two edges in sequence in the tree between the source and the given nodes) and so on until there are some nodes of the tree which are at distance \( m \) from the source, \( m \) being the depth of the tree.

![Figure 2: A Multicast Tree rooted at the NM](image2)

Note that such a multicast tree is nothing but a spanning tree rooted at the network manager node spanning all the other nodes. Such a spanning tree does not cover all the edges of the graph in general. We can assume that a multiple number of such multicast trees can be computed rooted at the network manager node such that all the edges are covered (Fig. 3). We also introduce the notion of minimum number of multicast (spanning) trees to cover all the edges of the graph later on in the paper. We concentrate on a single such multicast tree for the sake of discussion.

Our multicasting scheme has two variations, namely, the **incremental** approach and the **logarithmic** approach, although the fundamental notion is the same. We describe the incremental approach first.

3.1.1 Incremental Multicasting Scheme (IMS)

In this scheme, the network manager (NM), multicasts a status request message to all the nodes at distance one and each node sends its reply back to the NM along the same multicast tree but in the reverse direction. For example, NM sends a status request message to nodes \( n2 \) and \( n3 \) along edges \( e1 \) and \( e2 \) respectively in MT1 (Fig. 2). After that, the NM multicasts a status request message to all the nodes at distance two and the nodes send back the reply exactly along the same path the status request message from the NM came in, except in the reverse direction. For example, NM at node \( n1 \) sends a status request message to nodes \( n4, n5, n6 \) and \( n7 \) along \( (e1, e3), (e1, e4), (e2, e5) \) and \( (e2, e6) \) respectively in MT1 (Fig. 2) and \( n4, n5, n6 \) and \( n7 \) send back replies along \( (e3, e1), (e4, e2, e5) \) and so on.
Figure 3: Multicast Trees covering all the edges of the Graph

c1), (e5, e2) and (e6, e2) respectively. Thus the NM proceeds incrementally along the same multicast tree.

In order to describe the analysis done by NM after it gets back the reply from each node at a given depth of the multicast tree, we assume that the NM sends a status request message to all the nodes at distance i. We assume that the reply is either 1 (the node is alive and a reply comes back) or 0 (no reply comes back). If the NM receives a 1, it knows that the node is alive and so are the links along the multicast tree from the NM to the node in question. However, if the reply is 0, there are two possibilities: (1) the node in question is dead or (2) the last link on the path from the NM to the node is dead. For example, consider nodes at distance two from NM along MT1 in Fig. 2 and assume that all but n6 sends a reply 1. Since n6 did not send any reply, either it is dead or the link e5 is dead. We know e2 is not dead because n3 sent a reply through e2 in the first round.

In order to resolve this ambiguity, the NM has to use a different multicast tree and probe the same node. If the reply is 1, then the link used to reach the node in the last multicast tree is dead but not the node itself. However, if the reply is still 0, the controversy still remains as to whether the node is dead or both the links to the node used in the two different multicast trees are dead. For example, NM can send a status request message to n6 using e8 in MT2 (Fig. 3). If no reply is obtained from n6, either n6 is dead or both e5 and e8 are dead. If there are more links leading to the node in question, the same procedure can be repeated with different multicast trees if all of them have not been used already. In case all the multicast trees have been used but all the links ending in (or originating from) the node have not been used for sending or receiving messages, then the unused edges are marked and the testing continued with one of the multicast trees until a node n2 is reached which is connected to a marked edge through live edges.

Let us continue with the example to clarify the ideas. Recall that no reply has been obtained from n6 and the edges not yet considered are e13 and e14. They have not been used so far in MT1 because MT1 does not contain them. They have not been used in MT2 because e13 and e14 are both at distance 3 from the NM in MT2 and the testing at distance 2 is not yet complete. Therefore, in this example, the marked edges will be e13 and e14. In this case, when testing is continued with MT1, node n8 will be reached through e15 (assuming e15 is up) when multicasting is done to nodes at distance three and n8 is connected to e14 (which is a marked edge). Similarly, n9 will also be reached in MT1 through e11 (assuming e11 is up) and n9 is connected to the marked edge e13. In this example, n8 and n9 are the so called n2 nodes which are connected to marked edges e14 and e13 respectively. Note, however, that in general, a marked edge is not incident on a node n2. That is, a marked edge is not necessarily represented as (-,n2) or (n2,-) and that there may be several edges between n2 and a marked edge.

Once a node n2 is reached, NM sends a status request message to the node, from which no reply has been obtained so far, along the path from n2. If a response is not obtained, no conclusion can be drawn if there are marked edges which have not yet been tested. In that case, testing is continued until all the marked edges have been used. If a response is not obtained even after this, the ambiguity remains unresolved. Let us continue with the example to illustrate these ideas.

When testing is continued with MT1 and n9 is reached along (e1, e4, e11), then a status request message is sent to n6 along (e1, e4, e11, e13). If no response is obtained, the marked edge e14 must be used before anything is concluded. If n8 is also reached in MT1, then NM must send a status request message to n6 using the path (e2, e6, e15, e14). If a response is obtained, then e5, e8 and e13 are down while n6 is up. Otherwise, the ambiguity about whether n6 is down or all of e5, e8, e13 and e14 are down remains unresolved. However, only local tests can be used to resolve the ambiguities that remain following the incremental multicasting test method. In fact, testing a single link (say e5 in this example) is enough to find out which one of the two cases is true.

**Lemma 1** If a undirected graph \( G = (V, E) \) has no bridge edge and if there is a single fault in the network and that single fault is a faulty edge, IMS will definitely locate it.

**Proof:** Suppose there is a faulty edge \( e_1 = (v_j, v_k) \) and that is the only fault in the network.

Since each edge is part of at least one multicast tree, we can assume that \( e_1 \) is a part of multicast tree MT1. Assume that \( v_j \) is at a distance \( i \) from the NM while \( v_k \) is at a distance \( i + 1 \) from the NM in MT1. If \( e_1 \) is the only fault in the network, the NM will receive a 1 from node \( v_j \) when multicasting a status request
message to all nodes at a distance i from it. Then NM will multicast status request message to all nodes at a distance i+1 from it and in particular will enquire the status of v_k using e_2 as the last link on the path. Obviously there will be no response. Since there is no bridge edge in the graph, there exists another edge e_2 = (v_k, v_1) connecting node v_k which can be reached without passing through e_1. There are two possibilities at this point.

1. e_2 is a part of MT1 and e_2 is reached through e_1.
2. e_2 is a part of MT2 and is reached without going through e_1.

In case (2), v_k will be reached through e_2 and NM will receive a response from v_k. This will indicate that e_1 is faulty. In case (1), v_k will be reached through live edges from v_k (which is reached in MT1) and the response from v_k will reach NM. Thus e_1 will be detected as faulty.

Therefore, e_1 will be detected as a faulty edge in any case.

Lemma 2 If an undirected graph G = (V, E) has no bridge edge and if there is a single fault in the network and that single fault is a faulty node, IMS with a local test of a single edge connected to the node in question will definitely locate the fault.

Proof: Using reasoning similar to the proof of Lemma 1, it is easy to see that in case of a single faulty node, IMS by itself will fail to resolve the ambiguity between the case of a faulty node and the case of all faulty edges incident on the node. Assuming that the network has only a single fault and that is a faulty node, all the edges incident on it must be live. A local test performed on any one of the edges will indicate that the edge is live and hence will negate the proposition of all the edges (incident on the node) being down. Thus IMS and the local test will locate the faulty node.

Theorem 1 If an undirected graph G = (V, E) has no bridge edge and if there is a single fault (faulty node or faulty edge) in the network, IMS with at most a single local test will definitely locate the fault.

Proof: Follows directly from Lemmas 1 and 2.

Lemma 3 The NM needs to do O(d) multicasts per multicast tree on an average to detect a single fault in the network, where d is the depth of the multicast tree.

Proof: If there is a single fault in the network and that is a faulty edge at a distance i from the NM on a multicast tree, IMS needs at least i multicasts (one for each level) before the faulty edge is reached. If the edge is at a distance i in each multicast tree, i multicasts will be needed in each multicast tree, in the worst case (when the faulty edge belongs to the last multicast tree), to locate the fault.

If the only fault in the network is a faulty node and it is at a distance i from the NM in each multicast tree, i multicasts are needed in each multicast tree before a local test is performed to locate the fault. Thus, in either case, IMS needs to do ≤ i multicasts per multicast tree to detect a single fault. Assuming that each link or node is equally likely to fail, the probability that exactly one edge (node) fails in level i is f* probability that exactly one edge fails in level (i - 1), where f is the fan out of a node. This is true because there are f times more edges in level i than in level (i - 1), assuming a balanced tree. If the probability that exactly one edge fails in level 1 of the multicast tree is p, then the NM has to multicast once with probability p to locate the fault. By the same logic, IMS has to multicast twice with probability f*p, three times with probability f^2 * p and in general, i times with probability f^i-1 * p to locate a single fault in the multicast tree. Thus the average number of times IMS has to multicast is N_M = \sum_{i=1}^{d} (i * f^{(i-1)} * p). To avoid complex formulas, we choose to show the result for f = 2 and state that the reader can verify the result for general f. For f = 2, N_M = p * (d-1) * 2^d + 1. Since \sum_{i=1}^{d} (p^i) = 2, p = 1/(2^d-1) and N_M = (d-1) * 2^d + 1 / (2^d-1) = O(d). This has to be done for each multicast tree in the worst case.

The network management load generated by O(d) multicasts on an average by the NM and the responses from the nodes at any given level after the corresponding multicast, may considerably affect the performance of the system during the fault location phase. Thus the network management traffic which can be substantially high may not be desirable in all cases. The next scheme which we call logarithmic multicasting scheme is capable of reducing this traffic.

3.1.2 Logarithmic multicasting scheme (LMS)

In this scheme, we assume that the NM has the statistics of failure of the various network components and has a precomputed table of probability of failures for each node and each edge. Just as in the previous scheme, the NM requests the lower layer to set up a multicast tree from the NM node to all the other nodes of the graph. The NM uses the precomputed table of failure probabilities to compute a set S_0 of intermediate points on the multicast tree such that the probability of failure from the NM node to each intermediate point is approximately 0.5. We assume that at each such point there is a node. If not, we can choose the one closest to that point. The NM then multicasts the status request message to all the nodes at the intermediate points. The nodes respond to that request exactly as in IMS. In general, a reply will be received from a subset S_1 of the nodes and no reply will be received from the other subset S_2. At this point, we have two choices. First, we can discard subset S_1 and continue with subset S_2 implying that locating a fault is good enough for the NM. The second choice is to continue with both S_1 and S_2 implying that the NM prefers locating as many faults as...
possible.
We describe the general scheme in which none of the subsets are discarded. For subset $S_1$ (no faults), compute a subset $S'_1$ with new intermediate points between the old intermediate points and the leaf nodes. For subset $S_2$ (faulty), compute a subset $S'_2$ with new intermediate points between the NM node and the old intermediate points. This completes the computation of the set of nodes to which the NM will send the status request message next. This process of dividing the paths on the multicast tree continues until a single link is isolated or the situation of local testing (either a node is bad or all the links connected to the node are bad) is reached. Consider the multicast tree in Fig. 2 and assume that each link and each node has 0.15 probability of failure. Then the probability of failure $P(n_l, n_a)$ between $n_l$ and $n_a$ may be computed as follows.

$$P(n_l, n_a) = 1 - \left( 1 - P(n_l, n_d) \right) \cdot \left( 1 - P(n_d, n_a) \right)$$

Similarly, $P(n_l, n_0) = P(n_l, n_5) = P(n_5, n_7) = 0.48$. Thus, the set $S_0 = \{ n_4, n_5, n_6, n_7 \}$ and the NM multicasts a status request message to each node in $S_0$. Suppose replies are obtained from $n_6$ and $n_7$ but not from $n_4$ and $n_5$. In that case, $S_1 = \{ n_6, n_7 \}$ and $S_2 = \{ n_4, n_5 \}$. Now, we want to compute a set $S'_1 = \{ n_i | P(n_1, n_i) = 0.75 \}$ and a set $S'_2 = \{ n_i | P(n_1, n_i) = 0.25 \}$.

$$P(n_1, n_0) = 1 - (0.85)^2 = 0.67$$

Thus, $S'_1 = \{ n_8 \}$ and $S'_2 = \{ n_2 \}$. The NM multicasts a status request message to $n_0$ and $n_4$. If no reply is obtained from $n_2$, the faulty region $(n_2 or e_1 or all edges incident on n_2)$ is located. At this point, resolving the ambiguity reduces to sending status request messages to $n_2$ using the edges incident on it, namely $e_8, e_4, e_3$, exactly as in IMS.

This scheme is called logarithmic multicasting scheme because it reduces the number of multicasts (needed to locate a fault) logarithmically. Note that, the logarithmic multicasting scheme is similar to the concept of binary search in a more general environment. A definition is in order here.

**Definition:** A set of nodes $N$ is said to be at a $x$ ($0 \leq x \leq 1$) probability of failure distance from the NM along a specific multicast tree if the total probability of failure of (the links and nodes) along a path from the NM to each node $n \in N$ is $x$.

**Lemma 4** The NM needs to do $\log_2 d$ multicasts on an average to detect a single fault in the network.

**Proof:** LMS does the first multicast to all the nodes at 0.5 probability of failure distance from the NM along a given multicast tree. If faults are detected (no response from a subset $S_1$ of nodes) during this multicast, the portion of the multicast tree between $S_1$ and the leaves is discarded. Then a set of nodes $S'_1$ between the NM and $S_1$ is chosen such that the nodes in $S'_1$ are at 0.25 probability of failure distance from the NM. If no fault is detected in the first multicast, a new set of nodes $S'_2$ is chosen on the multicast tree such that the nodes in $S'_2$ are at 0.75 probability of failure distance from the NM. Thus, on an average, links in half of the depth of a multicast tree are discarded after each multicast (assuming each link is equally likely to fail). The depth of a multicast tree being $d$, the logad divisions are enough, on an average, to locate a fault. That is, LMS requires $\log_2 d$ multicasts on an average.

**Theorem 2** The network management load generated on an average in a network for locating a fault is higher for IMS than for LMS.

**Proof:** Follows from lemmas 3 and 4.

The performance of both the multicasting schemes IMS and LMS in terms of how quickly they can locate the faults, depends on two factors, namely the number of multicasting trees for a given graph and the depth of each tree. Ideally one would like to minimize both the number of multicast trees and the depth of each such tree. However, that may not always be possible. This needs further investigation.

The second basic scheme is called **investigator propagation scheme** which is motivated by the fact that the network management traffic load can be high in the multicasting approach, thereby affecting the network performance. We describe the second technique next.

### 3.2 Investigator Propagation Scheme (IPS)

In this scheme, the NM sends a special message encapsulating a program which is called the investigator along a precomputed route (using source routing in datagram networks) in the network leading back to the NM when the network exploration is complete. The function of the investigator, upon reaching a node, is to activate the corresponding node to collect the status of the links (edges) and nodes directly connected to it. As mentioned earlier, the status of a link is either 1 (up) or 0 (down). The investigator examines the status of each link and reports to the NM if any of them is down. Otherwise, it continues to the next node specified in its route and repeats the process. Consider the graph $G$ in Fig. 1. An investigator from the NM can be sent along the route $e_1, e_4, e_{10}, e_{17}, e_{20}, e_{21}, e_{12}, e_{14}, e_5$ and $e_2$. Note that the investigator collects the status of edges $e_3, e_4, e_8$ and nodes $n_2, n_4, n_5, n_6$ when it is at node $n_2$ on its path. Similarly, when it reaches node $n_5$, it collects the status of nodes $n_2, n_3, n_5, n_9, n_10$ and of edges $e_4, e_7, e_{10}, e_{11}$. In this way, when it returns to $n_1$, it has explored the entire network. Note that several edges and nodes are tested more than once, which may or may not be desirable. If it is not desirable to check a network entity more than once for whatever reason (may be performance), then the edges and nodes, which have been tested already can be marked. These marked edges (nodes) will not be tested again. For example, the edge $e_4$ and node $n_2$ are checked when the investigator is in node $n_2$. If we do not want to recheck them, we can mark them
as checked. Thus the investigator, when in node $n_5$, need not check the status of $e_4$ or $n_2$.

Note that the load generated by the investigator at any time is local to the node being activated and the overall network management load is distributed over time. We assume that each node sends a status request message (can be as simple as a "ping" in an IP network) to its neighboring nodes and based on whether a reply is received or not, the status of the corresponding link is marked 1 or 0. Strictly speaking, the absence of a reply does not ensure that the link is down. It may be that the corresponding node is down. A local testing can be done in such a case to resolve the ambiguity.

In order for the investigator to collect information about all the links and nodes, a route must be precomputed at the NM. Ideally, the route would be a minimum length tour such that the tour contains at least one node of each link. Note that such a tour is not a postman tour because the postman tour requires that each node is visited. In this case, the investigator does not need to visit each node because in some cases, it can collect the status of a node (without visiting the node) from its neighboring node. To state the problem formally, we need to define a few terms. Assume that an undirected graph $G = (V, E)$ is given where $V$ is a set of vertices and $E$ is a set of edges. An edge $e \in E$ is represented as $(v_i, v_j)$ where $v_i \in V$ and $v_j \in V$.

**Definition:** An edge $e = (v_i, v_j)$ in $G$ is said to be covered if either $v_i$ is visited or $v_j$ is visited.

**Definition:** A path in $G$ from vertex $v_i$ to vertex $v_k$ is defined as a sequence of edges in $E$ connecting $v_i$ and $v_k$.

**Definition:** A tour in $G$ is defined as a path starting and ending at the same vertex.

**Problem:** Given an undirected graph $G$, find a minimum length tour which covers each edge of $G$.

The above problem, which we call the Investigator Tour Problem (ITP), can be shown to be NP-hard. In the following paragraph, we show how the Traveling Salesman Problem (TSP) can be reduced to ITP in polynomial time.

**Theorem 3** Investigator Tour Problem (ITP) is NP-hard.

**Proof:** Given an undirected graph $G = (V, E)$, the minimum distance between each pair of nodes in $V$ can be computed. We refer to the minimum distance between nodes $n_i$ and $n_j$ as $d(i,j)$. Now, for each node $n_i \in V$, add an edge $(n_i, n_j) \in E$ with an weight $d(i,j)$. The edge $(n_i, n_j)$ is not added to the new set if it has already been included as $(n_j, n_i)$. This generates a new complete graph $G' = (V, E')$ where $E'$ contains the new weighted edges in addition to the edges in $E$. Note that the weight corresponding to an edge in $E$ is 1. Now, add a node $n'_i$ for each node $n_i \in G'$ and add an edge $(n_i, n'_i)$ to generate a new graph $G'' = (V', E'')$. The computation of $d(i,j)$ from $G'$ can be done in polynomial time. The addition of edges to $G$ to make it complete and addition of edges to $G''$ to convert it to $G''$ can also be done in polynomial time because the process is repeated at most as many times as the number of vertices $|V|$. Thus the transformation from the original graph $G$ to graph $G''$ can be performed in polynomial time. If there is a polynomial time algorithm to solve the ITP on $G''$, the same algorithm can be used to solve the TSP on the graph $G''$ (and effectively on $G$). That is, an ITP tour on $G''$ is a TSP tour on $G''$ (or $G$). Now, since TSP, which is an NP-complete problem, can be reduced to ITP in polynomial time, ITP is NP-hard.

Since ITP is NP-hard, we concentrate on finding an efficient heuristic solution to ITP. In this paper, we present one heuristic solution to ITP. The steps followed in the heuristic solution are:

1. Use an efficient heuristic algorithm [BB94][CLR90] to compute the minimum vertex cover set $VC$ of $G = (V, E)$.

2. Compute the minimum distance between each pair of nodes in $G$. Let us denote such a distance between $n_i$ and $n_j$ by $d(i,j)$.

3. Compute a new graph $G' = (VC, E')$ where an edge $(n_i, n_j) \in E'$ is the same as the edge $(n_i, n_j)$ in $G$ if one exists in $G$. Each such edge is given a weight of one. If $(n_i, n_j)$ does not exist in $G$, generate a new edge $(n_i, n_j)$ between nodes $n_i, n_j \in VC$ such that the weight on the edge is $d(i,j)$. The edge $(n_i, n_j)$ is not generated if $(n_j, n_i)$ has already been constructed.

4. Use an efficient heuristic algorithm [HK70][LK73][CLR90] for TSP on $G'$.

In some cases, the NM may be interested in finding the status of only the nodes and in such a case, the investigator needs to cover every node and not every edge. This is a special case of ITP and has been studied in the literature as the covering salesman problem (CSP) [CS89]. Interestingly enough, the CSP has been solved in [CS89] by using efficient heuristics for the Set Cover Problem (SCP) and TSP, which is an approach very similar to ours.

The advantage of the investigator propagation scheme is the low load it generates on the network at any given instant of time. However, the main drawback is the time it takes for the investigator to make a tour of the network. Although the investigator reports a fault to the NM the moment it detects the fault, it may not be until it reaches the second last node of the tour that a fault is detected. This may mean a long delay if the route is very long. One possible way to retain the advantages of the investigator approach while reducing the long delay is to have a hybrid scheme which is described next.

### 3.3 Hybrid Scheme

This scheme combines notions from the multicasting scheme and the investigator scheme in a way to
overcome the poor response time of the investigator scheme while reducing the network load generated by the multicasting scheme. In fact, the decrease in response time is achieved by trading network load. In the hybrid scheme, the graph $G = (V, E)$ representing the network is divided into multiple sub-graphs $G_i = (V_i, E_i)$ for $i = 1, \ldots, n$ where $E = \bigcup E_i$ and $V = \bigcup V_i$ and the sub-graphs $G_i$'s form a tree rooted at the NM node. Note that a subgraph can be linear and in that case it will be a simple branch of a multicast tree rooted at the NM node. Once this structure is formed, the NM can send $n$ investigators parallelly to the $n$ sub-graphs and each of the investigators can individually report faults in each sub-graph. Consider graph $G$ in Fig.1. We divide $G = (V, E)$ into sub-graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ where

$$V_1 = \{n_1, n_2, n_3, n_4, n_5, n_6, n_7\},$$

$$V_2 = \{n_2, n_4, n_5, n_9, n_10, n_11, n_13\},$$


$$E_2 = \{e_1, e_3, e_4, e_7, e_9, e_{10}, e_{16}, e_{17}, e_{18}, e_{20}\}.$$  

The path taken by the investigator in $G_1$ may be $e_2, e_6, e_{15}, e_{19}, (e_{13}, e_5, e_2)$ while the path taken by the investigator in $G_2$ may be $e_1, e_3, e_7, e_{17}, (e_{13}, e_9, e_3, e_1)$ where the edges in the parenthesis indicate the investigator does not stop on the nodes along those edges to ensure the status of the edges or nodes adjacent to them. It just uses those edges as a return path.

![Figure 4: Graph $G = (V, E)$ whose structure is suitable for Hybrid Scheme](image)

The advantage of the hybrid scheme depends heavily on the structure of the underlying graph. For example, consider a graph $G = (V, E)$ shown in Fig 4 which can be easily subdivided into graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ such that

$$V_1 = \{n_1, n_2, n_3, n_4, n_5, n_6, n_7\},$$

$$V_2 = \{n_8, n_9, n_{10}, n_{11}, n_{12}, n_{13}\},$$

$$E_1 = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}\},$$

$$E_2 = \{e_{11}, e_{12}, e_{13}, e_{14}, e_{15}, e_{16}, e_{17}, e_{18}, e_{19}\}.$$  

Note that $V = V_1 \cup V_2$ and $E = E_1 \cup E_2$. In this example, the NM located at node $n_1$ will send two investigators parallelly to $G_1$ and $G_2$ along $(e_4, e_8, e_5, e_1)$ and $(e_{19}, e_{11}, e_{14}, e_{13}, e_{12}, e_{19})$ respectively. Intuitively, the hybrid scheme will be faster than either the multicasting scheme or the investigator propagation scheme considered separately.

In the hybrid scheme, as the number of sub-graphs increases, response time decreases but the load increases. If a cost function can be defined in terms of response time and load, it is possible to design an algorithm which will sub-divide the graph $G$ in an optimal way.

Keeping in mind the advantages of the multicasting scheme and the investigator scheme, it is possible to use them in a complementary way in managing a heterogeneous network consisting of wide area networks (WANs), metropolitan area networks (MANs) and local area networks (LANs). We describe the combined scheme next.

### 3.4 Combined Scheme

The multicasting approach has the advantage of quick response time because the nodes at the leaves of the tree send their status messages simultaneously. However, generating a multicast tree across all the individual LANs and MANs may be cumbersome for the NM and the overall network management load may slow down the whole network during the fault management phase. On the other hand, the investigator approach has the advantage of keeping the network load to a minimum. However, such an approach is impractical for a WAN simply because of the time the investigator takes to explore the whole network. It seems that the investigator approach will work well in LANs and possibly in MANs (with the hybrid scheme) as well, partly because the time spent in exploring these relatively small networks will be small. On the other hand, the multicasting approach will work well in the WANs because faster response is more important in that case. Also the backbone network in a WAN has much higher capacity in contrast to the backbone in a LAN and hence the network management load generated in a WAN will not have significant effect on the overall network performance. Thus, we envision the LANs and MANs as single nodes in the multicast tree spanning the WAN rooted at the NM node. A single node on the multicast tree representing a LAN, for example, will serve as the NM for the LAN and will be responsible for reporting the status of the LAN as a whole to the NM.

Consider, for example, the network in Fig.5 which consists of several LAN’s and MAN’s connected to a backbone WAN. The nodes $n_2, n_3, n_4$ and $n_5$ may contain the network manager for the wireless ATM LAN, FDDI, DQDB and Ethernet respectively and they may use the IPS to locate faults in their respective local networks while the backbone network may define a multicast tree rooted at $n_1$ spanning $n_2, n_3, n_4, n_5, n_6, n_7$, etc., and may use either IMS or LMS to locate faults in the global network.
4 Conclusion

We have described in this paper several novel techniques for locating faults in networks in which the finding of faults and determining the status of various links and nodes in the network are carried out simultaneously. These techniques give rise to some interesting graph problems, which we have only touched on. In our investigator tour approach, we showed that finding an optimal tour is an NP-hard problem, but showed that good heuristics exist to obtain a near optimal tour. The various multicasting approaches lead to other graph theoretic optimization problems which we have only hinted at. In addition, it should be clear that our two multicasting approaches can be combined in several ways and could be used in combination with other network management techniques. Thus this work should be viewed as only a first step towards developing techniques for effective network management.

References


