Fault-Tolerant Routing of Network Management Messages in the Internet

Jaime Cohen
Universidade Estadual de Ponta Grossa
Departamento de Informática
R. C. Cavalcanti, 4748 CEP 84031-900
Ponta Grossa PR Brazil
jaime@convoy.com.br

Elias Procópio Duarte Jr.
Universidade Federal do Paraná
Departamento de Informática
P.O. Box 19081 CEP 8131-990
Curitiba PR Brazil
elias@inf.ufpr.br

Abstract

There is a pressing need for dependable network management systems, as network applications and operations have become critical for organizations and individuals. However, considering current approaches, a network fault may cause a partial collapse of the management system. For instance, consider two management entities communicating over the Internet using SNMP (Simple Network Management Protocol) messages. If there is a fault along the IP route that is being used to deliver a message, the communication between the management entities may fail even if they are fault-free. The situation does not change until the routing protocols at the network layer recover from the network fault. Alternatively, the application can choose by itself another route, bypassing the network layer fault. An SNMP routing proxy is an agent that is capable of acting as a bridge between the two management entities. In this paper we present a new fault-tolerant routing strategy based on the use of alternative routes that pass through the routing proxies. To locate the proxy to be used, we propose a fast heuristic that takes into account connectivity concepts of Graph Theory. We present a procedure to measure the fault coverage of the new approach in terms of the fraction of network management communication packets that reach their destination in the presence of network route faults. Finally we describe the implementation of the routing proxy that allows any agent to become a routing proxy at the cost of adding a MIB to the agent.

Keywords: Network Management, SNMP, Graph Theory, Internet Routing, Fault Tolerance.

1 Introduction

Computer network applications have become critical for organizations and individuals. At the same time, networks are becoming increasingly larger and more complex. It is thus necessary to have systems and tools that allow effective network monitoring and control, i.e., integrated network management systems [Ros94]. The Simple Network Management Protocol version 3 (SNMPv3) is the Internet standard management architecture. An SNMPv3 system is composed of management entities which communicate using the management protocol. The architecture defines a Management Information Base (MIB) as a collection of related management objects which are kept in order to allow management applications to monitor and control the managed nodes [HPW98].

SNMPv3 entities have traditionally been called managers and agents. Managed nodes contain an agent, which is a management entity that have access to management instrumentation. Each system has at least one Network Management Station, which runs at least one manager entity. Managers are collections of user-level applications, which may aim at performance evaluation or fault diagnosis, among others. There is currently a very large number of SNMP-based systems available, both commercial and on the public-domain.

One of the most important components of network management systems is the fault management subsystem. The purpose of fault management is to allow the quick discovery, isolation and solution of network faults [LFC95]. It is absolutely essential that a network fault management system be fault-tolerant, being able to work correctly even in the presence of faults in the network over which it is run [JMNM94]. However this is not the case today for most systems [Jr.97, JN98, JMNN97].

Consider a pair of network management entities, a manager and an agent communicating over the Internet using SNMP messages. As SNMP is an application layer protocol, if there is a fault along the IP route that is being used between the entities, their communication may fail even if the two entities are fault-free. Actually these entities will not be able to communicate until the routing protocols at the network layer recover from the failure. It is known that routing in the Internet does present a number of problems and instabilities [LAJ98, LMJ99, Pax98, Pax96].

A solution to this problem is to allow the application itself to re-route messages, bypassing the network layer fault. An *application route* is defined as a route computed at the application layer, and is a concatenation of *network routes*, which are computed at the network layer [JMNM94]. An SNMP *routing proxy* is an entity that is capable of acting as a bridge between the two management entities. The route from a manager to the proxy then to the agent is an application route; the route from the manager to the proxy is a network route, the route from the proxy to the agent is also a network route.

An algorithm to locate proxies on a given backbone was introduced in [JMNN98]. The algorithm begins considering the network routes from a fixed manager to each agent. In the first step, the algorithm marks all nodes that can reach both manager and agent if one link is removed from the graph, i.e., if one link is not operational, it finds which nodes can reach both agent and manager through a route that does not employ the removed link. The algorithm keeps for each node a counter of the number of alternative routes that the node provides. Nodes are ranked as proxy candidates according to the value of these counters.

In this paper we present a new fault-tolerant routing strategy to locate routing proxies.

The advantage of the new strategy is that it does not use the ever-changing network routes as input, being based on the network topology. The main criterion of the strategy is to find nodes contained in high connected parts of the graph in order to maximize the chances that the newly create application routes do not use the fauty link. We present a procedure to measure the fault coverage of the new approach in terms of the fraction of network management communication packets that reach their destination in spite of network route failures.

The rest of the paper is organized as follows. Section 2 presents an overview of the Internet's SNMP framework, including a description of the routing proxy. In that section we also describe de proposed criteria to find good proxies and some definitions. Section 3 presents the algorithm for locating the proxies. Section 4 presents a procedure to measure the fault coverage of the new approach. Section 5 present experimental results. Section 6 concludes the paper. The Appendix includes an ASN.1 description of the routing proxy's main objects.

2 SNMP Routing Proxies

The Simple Network Management Protocol version 3 (SNMPv3) is the Internet standard management architecture. An SNMPv3 system is composed of management entities which communicate using the management protocol. SNMP entities have traditionally been called managers and agents [Ros94]. Managed nodes contain an agent, i.e. a management entity which have access to management instrumentation. A manager queries the agents for management information describing the state of links, devices, protocol entities and nodes. Agents may also send event information to the manager by using alarms called *traps*. The manager takes decisions related to fault diagnosis, performance management, and network configuration, among others, based on the collected information.

There is a pressing need for network management systems capable of handling errors. Although network management systems are in principle responsible for fault management, current systems often fail as a consequence of the faults they should instead be helping to solve. If a communication link along the path from the manager to an agent or to a managed network is down at some point, there will be a collapse of network management, as the NMS won't be able to determine the state of part of the managed network.

To make the network management system resilient to network failures there has to be alternative means of accessing agents. The network is in general a mesh-type structure, there are multiple potential paths between two communication nodes. However, since network management systems are application layer entities, these have little or no control over the paths that will be chosen by the network layer for routing the management queries. So, alternative paths for management communication have to use application layer entities which relay the management query and the replies along adequate communication routes.

Using the concept of a *routing proxy* [JMNN98], the NMS has a simple application routing engine to implement a fault tolerant routing system. An SNMP proxy is an entity used by an SNMP manager entity to access another device, i.e., the proxy receives the query, transmits it to the agent, gets the reply and sends it back to the manager.

2.1 Application Routes

Consider the simple network topology in figure 1, where the manager (NMS) is connected to an agent (Ag) and also to two gateways, G1 and G2. Considering communications involving the NMS and the Ag, suppose that routing is such that the direct link is used to communicate the queries and replies, as shown in part A of the figure. If the link between the NMS and agent fails, network management queries will be delayed until the network layer recovers from the failure. The delay may be significant as a new route for the agent must be discovered. A proxy could relay the queries from NMS to Ag and the corresponding replies from Ag to NMS, as shown in part B of the figure. The condition to obtain this solution is that the routes used by the proxy be available when a failure occurs in the network route between manager and agent. In the example, G2 can be used as proxy in such a situation.

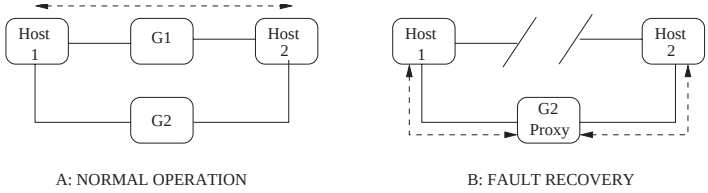

Figure 1: Management communication routes.

An *application route* is a concatenation of one or more *network routes*, which are joined by an application. Thus, the network route from the NMS to the proxy and the network route from the proxy to the agent result in an application route from the NMS to the agent when concatenated. Network routes are not transitive, so if there is a network route from node A to node B, and another network route from node B to node C, the concatenation of these two network routes may be different from the network route from node A to node C. Thus, the application route can be used as an alternative when there is a fault along the corresponding network route.

For a simple network topology like that of figure 1 the position of the proxy is quite obvious, but for a more complex network it is not a simple decision.

If any network route from the manager to an agent is not available, the agent will become unreachable to the manager. A set of proxies should be determined such that whenever an agent becomes unreachable an application route will be established to reach that node. In the next section a proxy location algorithm is presented to improve the availability of the network management system.

2.2 The Criteria and Ideas Behind the Algorithm

The development of an algorithm to position routing proxies in a network requires the establishment of well defined criteria to specify an ordering of the nodes that are candidates to became routing proxies for each pair of communicating nodes. Bellow we propose a set of criteria and their justification.

The algorithm receives as input the network topology and the nodes in which the manager and the agent are running. The network topology is given as an undirected

graph with possible multiple edges. Multiple links between two nodes actually occurs in practice. Possibly, the network can be weighted. The weights can represent the bandwidth of the connections or some reliability function of the links.

The main criterion used for sorting the routing proxies is based on connectivity concepts of Graph Theory. Those nodes of the network that belong to subgraphs with high edge-connectivity have precedence to be a routing proxy. In this way, the chances that the application route and the IP route are different is increased and also the proxy is likely to be reused when there are other faulty connections.

To avoid the use of proxies located too far from the communicating nodes, a bound on the length of the newly created application route is imposed.

To summarize the criteria used to find a good proxy, we enumerate them: (1) The proxy must belong to a component with large edge-connectivity. (2) The distance of the proxy must be bounded so that the the length of the application route does not exceed a certain size.

Beside the above ideas, two rules are used to break ties: (1') The size of the component containing the candidate routing proxy should be maximized. Proxies located in large components also maximize the chances that the network route and the application route are disjoint. (2') The distance of the application route resulting from the use of the routing proxy should be minimized.

For an example of a good proxy that follows our criteria, see figure 2. Assuming that the network route between the nodes representing the server and client has been broken, a good choice for the routing proxy is node A.

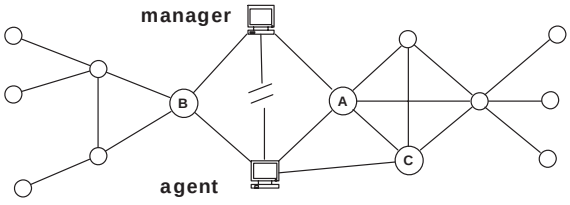

Figure 2: The direct link from the client to the server is faulty. Following our criteria, the first routing proxy to be tried would be node A.

2.3 Definitions

In this section we define some concepts about connectivity and distance on graphs that will be used in the description of the algorithms in the next section.

Applictation Route Length. The length of the application route formed by a routing proxy is defined below:

Definition 2.1 *For nodes proxy, manager and agent, define new-length*$(proxy, manager, agent)$ *as the sum of the length of the network path from the manager to the proxy and the length of the network path from the proxy to the agent.*

Neighborhood. The concept of a neighborhood will define those nodes that when used as a routing proxy will form an application route that is not too long. The definition follows:

Definition 2.2 *For an integer d and nodes manager and agent of the graph, define d-neighborhood$(manager, agent)$ as the set of nodes v such that the application route using v as proxy has length d or less.*

Vertex Numbering and Components with Maximum Edge-Connectivity. The connectivity parameters used to sort the candidate proxies are given below:

Definition 2.3 *Let $\#C(v)$ (with respect to a graph G) be the edge-connectivity of a non-trivial subgraph of G containing v and such that the cardinality of any cut separating nodes of this subgraph is maximized. Also, define $MCC(v)$ as the largest subgraph containing v and such that any pair of vertices of this subgraph cannot be separated by a cut of size less than $\#C(v)$. See figure 3 for an exemple.*

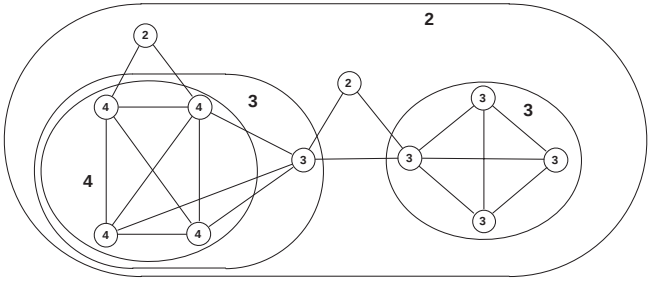

Figure 3: This figure shows the connectivity numbers of the nodes and the components $MCC(v)$.

In the definitions of $\#C()$ and $MCC()$, note that by non-trivial we mean a subgraph containing two or more vertices. If trivial cuts were allowed in the definition, the degree of the vertex would be its connectivity number. But the degree of a node is not a good measure for our problem. One of the reasons is that there may be many similar paths passing through the node and this situation happens exactly when the node belongs to a subgraph with low connectivity. For example, consider an induced subgraph that has a star topology. All IP routes between pairs of those nodes may pass through the center of the star and a good routing proxy may need to avoid passing through this center. Note also that the degree of a vertex v is an upper bound for $\#C(v)$.

3 The Algorithm for the Routing Proxy Selection

The main algorithm for proxy location first finds a good routing proxies inside some neighborhood of the nodes that are trying to communicate. The connectivity criteria is

used and ties are broken by the size of the maximum connected component containing the node and by the distance of the new route.

The most difficult step to be accomplished by the algorithm is to find the connectivity numbers of the nodes. This problem is discussed in a separate subsection. Below we describe the main algorithm:

```
Algorithm Find_Proxies (Network N, Node manager, Node agent)
Output: a list of candidate proxies ordered from the best to the worst
begin
  Let L be an empty list;
  Find the Edge-Connectivity Numbering of each node, #C(v), and
  the sizes of their correspondent maximum connected components MCC(v);
  Find new-length(v_proxy, manager, agent) for each node v_proxy;
  Let P be the IP path used for the manager-agent communication;
  Let d = 2;
  while |L| <= n-2 do
     Sort the nodes in V-L and in the (d*|P|)-neighborhood(manager,agent)
     in non-increasing order, using as key
     <#C(v),|MCC(v)|, - new-length(v,manager,agent)>;
     Append the sorted nodes to the rear of L;
     d := d + 1; // the neighborhood is augmented to reach new nodes
  end_while;
  return L;
end.
```

Time Complexity The time complexity of the above algorithm is given by the time to find the connectivity numbers and the distances in the network. The distances can be computed by two applications of a single source minimum distance algorithm, one having the manager as source and the other having the agent as source. Using Dijkstra's algorithm the time taken is $\Theta(n.\log(n))$, assuming that the network is sparse.

The running time complexity of the entire algorithm depends on how the functions $\#C()$ and $MCC()$ are computed. Both functions can be computed in time $O(n^4)$ using the graph theoretical concept of Cut Trees, see [CJ01] for a complete description of the algorithm. To avoid the high time complexity of the exact algorithm, we propose in this paper an heuristic to compute the functions $\#C(v)$ and $MCC(v)$. The running time of the heuristic drops to $O(n+m)$, assuming that the networks are sparse. The whole complexity of the algorithm is improved to $\Theta(n.\log(n)+m)$. We also show experimentally that the heuristic gives a good approximation for $\#C(v)$.

3.1 A Linear Time Heuristic to Compute #C(v) and MCC(v)

The heuristic we propose is based on an idea suggested by an algorithm described in [KR96] for another problem related to connectivity in graphs. The heuristic we describe here seeks, at each step, to remove a small number of edges that form a set of two edge-

connected subgraphs possibly connected by simple paths. The vertices of the 2-edge-connected subgraphs have their estimatives of their $\#C(v)$ increased by 2 and the others by 1. To that end, the heuristic removes, from each connected component and at each step, a depth first search tree plus a set of edges to form 2-edge-connected subgraphs. The extra edges are called *cover edges*. The cover edges are found using the following heuristic: to cover an edge, find the edge not in the tree that goes as high as possible. In this way, the edge tends to cover a large number of tree edges.

A pseudo-code of the algorithm follows:

```
Algorithm Heuristic_Compute_#C(Graph G)
Output: An approximation of the function #C(v), for all vertices v.
begin
  for all v in G
      #C[v] = 0;
      if degree(v) = 0
      then remove v from G;
  end_for;
  while the set of edges is not empty do
     for each connected component c of G do
         T = a DFS tree of c;
         for each edge of T do
             find the cover edge of e; // e may not exist if e is a bridge
         for each node v of c do
             if v belongs to a cycle of (T + the cover edges)
             then #C[v] = #C[v] + 2;
             else #C[v] = #C[v] + 1;
         end_for;
         delete T and the cover edges from G;
         remove isolated nodes;
     end_for;  // end of the each component for
  end_while;   // end of the outer while
  return #C();
end.
```

The algorithm runs in worst case of $O(m/n(m+n))$. In practice the graphs are sparse, so the expected running time is linear because m/n is O(1).

To compute the function $MCC(v)$, given the exact connectivity number of the nodes, we need to transverse the graph starting at vertex v and visiting just the neighbors with $\#C()$ greater than or equal to $\#C(v)$. Since we computed the function $\#C(v)$ approximately, the same algorithm will give an approximation for the function $MCC()$.

4 Fault Coverage Measurement

The previous section introduced a new approach to locate SNMP routing proxies on a given network. These proxies are used to implement a fault-tolerant routing strategy for

network management messages. Consider a pair of management entities communicating over an Internet backbone. Consider, for instance, that a manager application has sent a message to an agent. Whenever this message is not delivered because of a fault on the IP route, e.g. a faulty link, a new application route is tried, which is the concatenation of the IP route from the manager to the routing proxy, and the IP route from the routing proxy to the agent.

In this section we evaluate the impact of using the Routing Proxies on the dependability of the network management system. We give a measure of the fault coverage considering single link faults. The system is fully operational when all management messages are properly delivered. The system has failed if a message cannot be delivered. A failed system can be repaired if there is a Routing Proxy that is able to bridge the communication between manager and agent.

To obtain our fault coverage measurement for a given network, we fix the manager, and consider that all other nodes in the backbone are agents. Every node can be considered to be the manager, in this way all possible manager-agent communications over the network are taken into consideration.

We define the *Link Vulnerability*, v_i, for a given link l_i, as the number of agents that become unreachable to the manager if l_i is faulty divided by the total number of agents. In the presence of a proxy placement algorithm, the Link Vulnerability defines reachability taking into account possible paths going through the chosen proxy.

Network Vulnerability, V, for a given network is the weighed summation of link vulnerabilities, for all L links in that network, i.e.: $V = \frac{\sum_{i=1}^{L} v_i}{|L|}$.

The fault coverage c, of a system, gives the probability that the system will recover given the occurrence of a fault in the network. In this context it refers to the probability that the network management system will stay operational if one link becomes faulty. The measure can be obtained from the previously introduced vulnerability. For the network management case, whenever an alternative route exists as an option for the communications that use a given link, the coverage of the system is improved, as the system remains operational. This is how the measure is useful for analyzing the placement of proxies over the network. We can easily generalize the fault coverage concept to consider a set of managers by summing the fault coverage for each manager and dividing the result by the number of managers.

5 Experimental Results

All graphs used in the experiments were generated using the Waxman Method described in [ZCD97], a paper dedicated to the study of methods to model Internet backbones. We used graphs with different number of nodes and average degree around 4.

First we compared the exact value of the function $\#C(v)$ with the value computed by

the heuristic. We show experimentally that on the average the difference for each node is less than one. The table 1 shows the results. One column shows the sum of the differences over all nodes and the other is the average difference for the nodes.

n	sum of differences	average error per node
50	39.2	0.78
100	64.2	0.64
200	155.0	0.77
500	428.0	0.86

Table 1: Comparison between the exact values of $\#C()$ and the values found by the heuristic.

The table 2 shows that the routing proxies have a good chance to recover the communication in case of random errors in the network. The values are the percentages of pairs of nodes that can recover their communication after some link in their IP route has failed. The results presented are the average computed considering the failure of every link, one at a time, and all pairs of nodes, what is equivalent to the *coverage* measure defined in the previous section. The column entitle "one proxy" represents the percentage of times that the first routing proxy was able to restore the communication between the pair of communicating nodes and the next column shows the results when a second routing proxy is tried when the first one fails.

n	one proxy	two proxies
10	73%	93%
20	75%	93%
30	62%	87%
50	59%	81%

Table 2: Percentage of node pairs that recover their communication using the method proposed in the paper.

6 Conclusions

As network management runs at the application level, it fails whenever management entities employ a network layer route that is faulty. In this paper we propose a fault-tolerant routing strategy for network management messages over the Internet. This approach is based on using alternative application routes that pass through a special SNMP entity called routing proxy, which acts as a bridge between the pair of communicating entities. Any SNMP agent may have the functionality of the routing proxy with the cost of adding

a new MIB. This approach allows network management systems, which include a fault management system to be fault-tolerant.

We propose an algorithm to locate routing proxies that receives as input the network topology, the manager and the agent whose network route has become faulty. The main criterion to select proxies is to find the maximal connected subgraphs of the network. In this way, the probability that the proxy can be used when there are faulty links is maximized. The algorithm returns a set of proxies that we call *candidate proxies*. The algorithm runs in worst case of $O(m/n(m+n)+n.\log(n))$, where n is the number of vertices and m the number of edges in the graph. In practice as the graphs corresponding to backbones are sparse, the expected running time of the algorithm is $O(n.\log(n)+m)$. We gave a procedure to evaluate the fault coverage of the new approach which gives the percentage of management messages that are correctly delivered in spite of network level route faults.

Future work include adapting the algorithm for weighted graphs modeling the bandwidth of the links, and comparing different approaches to find routing proxies.

References

[CJ01] J. Cohen and E. P. Duarte Jr. Algoritmos para o problema da conectividade em grafos com aplicações ao gerenciamento integrado de redes de computadores. Research report, Universidade Estadual de Ponta Grossa, Brazil, March 2001.

[HPW98] D. Harrington, R. Presuhn, and B. Wijnen. An architecture for describing snmp management frameworks. *Request for Comments 2271*, January 1998.

[JMNM94] E.P. Duarte Jr., G. Mansfield, S. Noguchi, and M. Miyazaki. Fault-tolerant network management. *Proceedings of ISACC'94*, 1994. Monterrey, Mexico.

[JMNN97] E.P. Duarte Jr., F. Mansfield, T. Nanya, and S. Noguchi. Non-broadcast network fault-monitoring based on system-level diagnosis. *Proc. IFIP/IEEE International Symposium on Integrated Network Management IM'97*, pages 597–609, 1997.

[JMNN98] E.P. Duarte Jr., G. Mansfield, T. Nanya, and S. Noguchi. Improving the dependability of network management systems. *International Journal of Network Management*, 8(4), July-August 1998.

[JN98] E.P. Duarte Jr. and T. Nanya. A hierarchical adaptive distributed system-level diagnosis algorithm. *IEEE Transactions on Computers*, 47(1):34–45, January 1998.

[Jr.97] E.P. Duarte Jr. *Fault Tolerant Network Monitoring.* PhD thesis, Dept. of Computer Science, Tokyo Institute of Technology, 1997.

[KR96] S. Khuller and B. Raghavachari. Improved approximation algorithms for uniform connectivity problems. *Journal of Algorithms*, 21, 1996.

[LAJ98] C. Labovitz, A. Ahuja, and F. Jahanian. Experimental study of internet stability and backbone failures. *Proceedings of the FTCS-29*, 1998. Madison, Wiscosin.

[LFC95] A. Leinwand and K. Fang-Conroy. *Network Management: A Practical Perspective.* Addison-Wesley, Reading, MA, 1995.

[LMJ99] C. Labovitz, G. R. Malan, and F. Jahanian. Internet routing instability. *IEEE Transactions on Networking*, 7(3), June 1999.

[Pax96] V. Paxson. End-to-end routing behavior in the internet. *Proceedings of the ACM SIGCOMM*, 1996.

[Pax98] V. Paxson. End-to-end internet packet dynamics. *IEEE Transactions on Networking*, 6(5), October 1998.

[Ros94] M.T. Rose. *The Simple Book - An Introduction to Internet Management.* Prentice-Hall, Englewood Cliffs, second edition, 1994.

[ZCD97] E. W. Zegura, K. L. Calvert, and M. J. Donahoo. A quantitative comparison of graph-based models for internet topology. *IEEE/ACM Transactions on Networking*, 1997.