REDUCING THE CORRECTING COST OF FAULTY NODES IN EXTERNALLY MANAGED OVERLAY NETWORKS

V.L.PAVANI, B.SATHYANARAYANA
Assistant Professor, Department of MCA, Intell Engineering College, ANANTAPUR.
Professor, Department of Computer Science, S.K. University, ANANTAPUR

ABSTRACT
An end-to-end approach of inferring probabilistic data forwarding failures is considered in an externally managed overlay network. The overlay nodes are independently operated by various administrative domains. The optimization goal is to minimize the expected cost of correcting all faulty overlay nodes that cannot properly deliver data. The correcting cost includes diagnosing and repairing. A candidate node should be first checked which is identified using a potential function instead of first checking the most likely faulty nodes as in conventional fault localization problems. Several efficient heuristics are proposed for inferring the best node to be checked in large-scale network. It is shown by extensive simulation that the best node can be inferred in at least 95 percent of time, and that first checking the candidate nodes rather than the most likely faulty nodes can decrease the checking cost of correcting all faulty nodes. The correct location of the candidate node that can be checked first is shown.

1. INTRODUCTION
Network components are prone to a variety of faults such as packet loss, link cut, or node outage. To prevent the faulty components from hindering network applications, it is important to diagnose (i.e., detect and localize) the components that are the root cause of network faults, as in [1], [2]. However, it is also desirable to repair the faulty components to enable them to return to their operational states. So the focus is on network fault correction, which means not only to diagnose, but also to repair all faulty components within a network. To diagnose (but not repair) network faults, recent approaches like [3], [4] use all network nodes to collaboratively achieve this. For instance, in hop-by-hop authentication [3], each hop inspects packets received from its previous hop and reports errors when packets are found to be corrupted. While such a distributed infrastructure can accurately pinpoint network faults, deploying and maintaining numerous monitoring points in a large-scale network introduces heavy computational overhead in collecting network statistics [5] and involves complicated administrative management [6]. It is difficult to directly monitor and access all overlay nodes in an externally managed network, whose routing nodes are independently operated by various administrative domains. So network condition is inferred from end-to-end information. Here, an end-to-end inference approach is considered which uses end-to-end measurements that infers components that are probably faulty in forwarding data in an application layer overlay network. A routing tree topology is taken from a large network with a set of overlay nodes, since a tree based setting is typically seen in destination-based routing where each overlay node builds a routing tree with itself as a root, as well as in multicast routing, where a routing tree is built to connect members in a multicast group. Then every root-leaf overlay path is monitored. If a path exhibits any “anomalous behavior” in forwarding data, then some “faulty” overlay node on the path must be responsible. In the above end-to-end solution, one can tell whether a path behaves anomalously, but cannot specifically tell which and how many overlay nodes on the path are faulty. Since externally managed overlay nodes cannot be directly accessed or monitored, in order to correct the faulty nodes, the administrators of the corresponding network should be contacted to manually check a sequence of potentially faulty nodes and fix any nodes that are found to be actually faulty. Given the anomalous paths in a tree, the main goal is to infer the best node (or the best set of nodes) that should be first checked so as to minimize the expected cost of correcting all faulty nodes.

2. LITERATURE SURVEY
A literature survey has been made on the overlay networks by researchers as follows.

2.1 EXTERNALLY MANAGED OVERLAY NETWORKS
In this paper, a faulty nodes in an externally managed overlay network is diagnosed and repaired, in which overlay nodes are independently operated by multiple administrative domains. In an externally managed network, the nodes cannot be directly accessed or monitored. Examples of externally managed overlay networks include Resilient Overlay Network (RON)[7], which provides routing resilience toward Internet path outages, and Service Overlay Network (SON)[8], which provides end-to-end quality-of-service guarantees. Both RON and SON deploy overlay nodes over multiple administrative domains that cooperatively accomplish certain network services. To ensure the availability of these network services, an effective network fault mechanism is, therefore, necessary. Researchers also advocate the notion of security-oriented overlay architectures, such as Secure Overlay Services (SOS) and Mayday[9], to defend against denial-of-service attacks. In both SOS and
Mayday, data is securely tunneled over an overlay network. As shown in, successful data delivery is preserved with a very high probability even if a subset of overlay nodes fail (e.g., shut down by attackers). While data may be rerouted to bypass failed nodes, robustness of data delivery will be degraded if the failed nodes are not immediately repaired because attackers can now devote resources to attacking the remaining non failed nodes.

Fig 1. End-to-end inference approach for a network fault correction scheme.

2.2 CONVENTIONAL FAULT LOCALIZATION PROBLEM
Automatic fault localization is a difficult problem, even for conventional programs. The most approach in the software engineering community is the Jones, Statsko and Harold’s algorithm that compares execution traces inorder to guess the source code statements that are more likely to contain a fault.

2.3 ANOMALOUS BEHAVIOR
The definition of anomalous behavior varies across applications. For example, a path is said to be anomalous if it fails to deliver a number of correct packets within a time window, and that some node on the path is faulty if it causes severe packet loss and delay.

2.4 RESILIENT OVERLAY NETWORK (RON)
A Resilient Overlay Network (RON) is an architecture that allows distributed Internet applications to detect and recover from path outages and periods of degraded performance within several seconds, improving over today’s wide-area routing protocols that take at least several minutes to recover. A RON is an application-layer overlay on top of the existing Internet routing substrate. The RON nodes monitor the functioning and quality of the Internet paths among themselves, and use this information to decide whether to route packets directly over the Internet or by way of other RON nodes, optimizing application-specific routing metrics.

2.5 MAYDAY: DISTRIBUTED FILTERING FOR INTERNET SERVICES
Mayday is an architecture that combines overlay networks with lightweight packet filtering to defend against denial of service attacks. The overlay nodes perform client authentication and protocol verification, and then relay the requests to a protected server. The server is protected from outside attack by simplifying in the substrate. The RON nodes monitor the functioning and quality of the Internet paths among themselves, and use this information to decide whether to route packets directly over the Internet or by way of other RON nodes, optimizing application-specific routing metrics.

3. PROBLEM FORMULATION
Each node in a logical tree T is classified as faulty or non faulty, depending on how we first define whether a (root-to-leaf) path exhibits any “anomalous behavior. Note that the inference approach only knows whether a path is anomalous, but does not specifically know which and how many nodes on the path are faulty. Each node in T is referred to as bad if it is faulty, or as good otherwise. A path is bad if it contains at least one bad node, and is, otherwise, good. Also, T is a bad tree if every path in the tree is a bad path. Since failure model focuses on fail-stop failures, it is further assumed that a node exhibits the same behavior across all paths upon which it lies. With this assumption, if a node lies on at least one good path, then we infer that it is a good node. Note that a good node may still lie on one or more bad paths, but this only means each such bad path contains some other bad node. On the other hand, if a node exhibits different behaviors across different paths, our analysis becomes more complicated, as a good path now Possibly contains bad nodes as well. We pose the analysis of different behaviors of a node as future work. Given a logical tree T, we determine whether a path is good or bad via end-to-end measurements that are carried out between the root and leaf nodes of T. For example, the root can send probes to the leaf nodes, from which we collect the measurement results. Since the focus is on the data forwarding failures, the probes should represent the regular data packets that can be forwarded by overlay nodes. Since a good path contains only good nodes that need not be checked, we only need to focus on the bad paths in T. Also, any node that lies on both good and bad paths is indicated to be good. To illustrate, Fig. 2 shows how to retain only the bad paths in T and indicate the good nodes. The resulting set of bad paths will then form a bad tree. With a slight abuse of notation, this bad tree is denoted as T as well. Then the bad tree T is
passed to the inference algorithm, which determines, from the set of nodes that are not indicated as good, the “best” node (or the “best” set of nodes) to be checked, and repaired, if necessary.

![Diagram of logical tree with bad and good nodes.](image)

Fig. 2. Given a logical tree, we retain only the bad paths and indicate any good node. Since path \{1; 3; 7\} is a good path, it is known that nodes 1, 3, and 7 are good. Nodes 3 and 7 can be pruned from the tree, and node 1 can be indicated as good. The resulting set of bad paths will lead to a bad tree.

A straightforward way to implement the inference algorithm is based on the brute-force approach as shown in Algorithm 1, which enumerates all possible diagnosis sequences in order to determine the best node.

**Algorithm 1. Brute-force inference algorithm**

Input: Bad tree \(T = (N; \{p_i\}; \{c_i\})\)
1: \(S^* = \emptyset\), \(c^* = \infty\)
2: for all diagnosis sequence \(S\) do
3: compute \(c = \text{the expected cost of } S\)
4: if \(c < c^*\) then
5: \(S^* = S\); \(c^* = c\)
6: return the first node in \(S^*\)

4.0 CANDIDATE NODES

A candidate node should be first checked, which is selected based on the maximization of a potential function as described below. Given a tree \(T\), ancestors of node \(i\) is defined to be the nodes (not including node \(i\)) on the path from the root of \(T\) to node \(i\), and descendants of node \(i\) to be the nodes that have node \(i\) as one of their ancestors. Let \(T\) be the event that \(T\) is a bad tree, and \(X_i\) be the event that node \(i\) is a bad node. Let \(A_i\) be the event that the ancestors of node \(i\) are all good. If node \(r\) is the root node, then let \(A_r\) be always true and \(Pr(A_r) = 1\). The potential of node \(I\) in a tree \(T\) is defined as the value returned by the potential function

\[ \Phi(I,T) = Pr(T|X_i,A_i)p_i c_i(1-p_i) \]

Intuitively, the best node should be a node with a high potential, since such a node, in general, has a small checking cost, a large failure probability, and a large likelihood of leading to a bad tree. Note that the term \(Pr(T|X_i,A_i)p_i\) denotes the potential of a node for a single-bad-path case. Therefore, the potential function \(\Phi(I,T)\) is to combine the potential of a single path

4.1 CANDIDATE BASED HEURISTICS

It is difficult to select a best node from a set of candidate nodes. In the following discussion, the optimality results are verified by the Brute force inference algorithm.

1. Given a bad tree, the best node is not the one with the highest potential.
2. Checking simultaneously all candidate nodes in a bad tree does not minimize the expected cost of correcting all faulty nodes.
3. The best node for a bad tree is not necessarily the best node for a subtree.

4.2 EVALUATION OF CANDIDATE-BASED HEURISTICS

Given the difficulty of finding the best node among a set of candidate nodes, we evaluate the performance of three candidate-based heuristics that approximate the best node selection of the inference algorithm. These heuristics are:

1) Cand-Prob, which selects the candidate node with the highest conditional failure probability given a bad tree;
2) Cand-Cost, which selects the candidate node with the least checking cost; and
3) Cand-Pot, which selects the candidate node with the highest potential.

CONCLUSIONS

The optimality results for an end-to-end inference approach to correct (i.e., diagnose and repair) probabilistic network faults at minimum expected cost are presented. One motivating application of using this end-to-end inference approach is an externally managed overlay network, where the nodes cannot be directly accessed and monitored that are independently operated by different administrative domains. First checking the node that is
most likely faulty or has the least checking cost does not necessarily minimize the expected cost of correcting all faulty nodes. A potential function is constructed for identifying the candidate nodes, one of which should be first checked by an optimal strategy. Due to the difficulty of finding the best node from the set of candidate nodes, several efficient heuristics are proposed that are suitable for correcting faulty nodes in largescale overlay networks. It is shown that the candidate node with the highest potential is actually the best node in at least 95 percent of time, and that checking first the candidate nodes can reduce the cost of correcting faulty nodes as compared to checking first the most likely faulty nodes.

REFERENCES

Authors
1. V.L.PAVANI, working as an Assistant Professor in the Department of MCA in Intell Engineering College, Anantapur and pursuing Ph.D (part-Time ) under the supervision of Prof. B.Sathyanarayana, Dept of Computer Science, SKU, Anantapur.
2. Dr. B. Sathyanarayana, Professor, Dept of Computer Science, Sri Krishnadevaraya University, ANANTAPUR.