Fault localization in Communication Networks

Vijay Akkineni
Department of Computer Science and Engineering
Georgia State University
vakkineni1@student.gsu.edu

Abstract

This paper is about fault localization techniques in communication networks worked for Phd qualifiers examination. Fault Localization is an important aspect of network fault management and is a process of deducing the source of a failure from the set of observed indicatations. It has been an important research area in the field of networking both communication networks and wireless sensor networks. As commuication networks grow in size and complexity it has been imposing new set of requirements on fault localization. Despite the amount of research done in this field we can still argue that it is an area of open for research. The paper essentially disucsses the work that has been done in this field in the pas few years with emphasis on the both the papers given for the examination.

Categories and Subject Descriptors

H.4 [**Communication Networks, Sensor Network Applications**]: Miscellaneous

Keywords

Fault Localization, Communication Networks, Root Cause Analysis, Causal Inference

1 Introduction

Fault diagnosis is an important part of networking. Faults are unavoidable in communication systems but their quick detection and isolation and repair is critical for the robustness, reliability and health of the system. When the networks get large and cumbersome automatic fault diagnosis and fault management is a crucial aspect.

A basic taxonomy in this field is mentioned below.

Event, defined as an exceptional condition occurring in the operation of hardware or software of a managed network, is a central concept pertaining to fault diagnosis.

Faults (also referred to as problems or root causes) constitute a class of network events that can cause other events but are not themselves caused by other events. Faults may be classified as: (1) permanent, (2) intermittent, and (3) transient. A permanent fault exists in a network until a repair action is taken. Intermittent faults occur on a discontinuous and periodic basis causing a degradation of service for short periods of time. However, frequently re-occurring intermittent faults significantly jeopardize service performance. Transient faults cause a temporary and minor degradation of service.

Error(Failure) is defined as a discrepancy between a computed, observed, or measured value or condition and a true, specified, or theoretically correct value or condition. Error is a consequence of a fault. Faults may or may not cause one or more errors. Errors may cause deviation of a delivered service from the specified service, which is visible to the outside world. Errors do not need to be directly corrected, and in many cases they are not visible externally. However, an error in a network device or software may cause a malfunctioning of dependent network devices or software. Thus, errors may propagate within the network causing failures of faultless hardware or software.

Symptoms are external manifestations of failures. They are observed as alarmsnotifications of a potential failure. These notifications may originate from management agents via management protocol messages (SNMP trap and CMIP EVENT-REPORT), from management systems that monitor the network status, e.g., using command ping, system logfiles or character streams sent by external equipments.

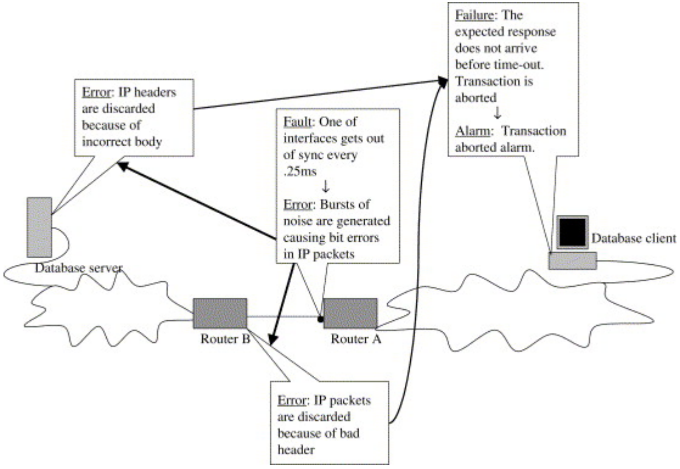
Figure 1 shows the concepts described above in an end to end perspective.

The process of fault diagnosis usually involves three steps as cited in [1]:

- Fault detection, a process of capturing on-line indications of network disorder provided by malfunctioning devices or fault detection agents in the form of alarms.
- Fault localization (also referred to as fault isolation, alarm/event correlation, and root cause analysis) is a set of observed fault indications is analyzed to find an explanation of the alarms.
- Testing is a process that, given a number of possible explanations, determines the actual faults.
- Fault Correction, by which we mean not only to diagnose, but also to repair all faulty components within a

network (This includes the testing component).

Figure 1. Distinction between fault, error and symptom


Failure: The expected response does not arrive before time-out. Transaction is aborted
↓
Alarm: Transaction aborted alarm.
Error: IP headers are discarded because of incorrect body
Fault: One of interfaces gets out of sync every .25ms
↓
Error: Bursts of noise are generated causing bit errors in IP packets
Database client
Database server
Router B
Router A
Error: IP packets are discarded because of bad header

The difficulty in the fault localization process arise from the ambiguity of the observed set of alarms as same alarm can be generated from multiple different faults and multiple alarms can correlate back to the same fault. The incompleteness of the alarm stems from the fact that the alarm doesn't have all the information or there is a loss of alarm. Inconsistency among the observed alarms results from device perception of the fault is different form device to device.

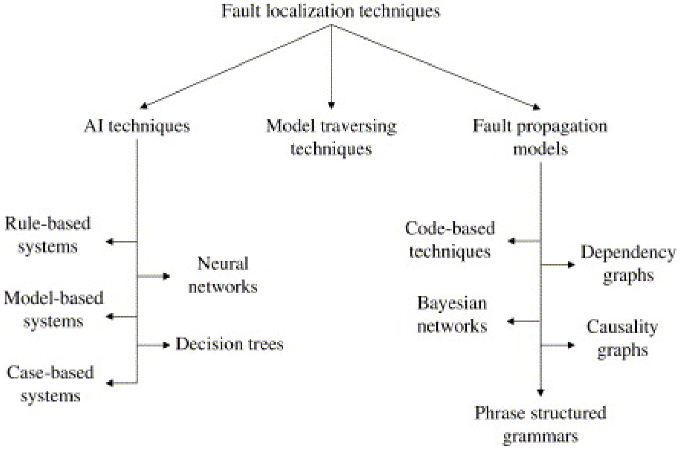
A set of alarms generated by a fault may depend on many factors such as dependencies among network devices, current configurations, services in use since fault occurrence, presence of other faults, values of other network parameters, etc. Due to this non-determinism the system knowledge may be subject to inaccuracy and inconsistency. Fault evidence may also be inaccurate because of spurious alarms, which are generated by transient problems or as a result of overly sensitive fault detection mechanisms. When spurious symptoms may occur, the management system may not be sure which observed alarms should be taken into account in the fault localization process. Event management systems should identify and eliminate multiple simultaneous related or unrelated root causes.

In large networks distributed fault localization process should be performed in distributed fashion. Many researches [2] and [3] have concluded that distributed fault localization using a set of event management nodes is a better approach than centralized process. The complexity arised from the nature of error propogation, they propogate either horizontally to the peers or vertically from bottom layer to upper layers or afecting higher level services. The errors also propogate from one management domain to a different management domain making the process even difficult. Therefore the distribution of complexity to distributed fault localization processes and inferring form the collective process makes the process computationally feasible and efficient.

An alarm can insinuate different type of faults that occurred in different communication devices where in fault localization process may not comeup with a definitive answer. Few approaches that will be discussed in this paper combine fault localization with testing and fault ocrrection process to validate the hypothesis. Therefore there should be some kind of optimality measure or confidence measure that should be employed in measuring the hypothesis that the localization process came up with and it could include the lowest cost or min failure probability or some heuristic function which optimizes the process of validating the hypothesis.

Numerous works have been proposed on fault localization process. The techniques are dervice from different areas of artificial intelligence, graph theory, neural networks, automata theory and many other approaches. Figure 2 broadly classifies the exisiting solutions. The two papers provided for the exam falls into the category of end to end testing scenarios and proabablistic reasoning and derive their roots from bayesian network analysis.

Figure 2. Classification of fault localization techniques



This paper presents the background research that has been done before paper 1 in section 2 - 4 and in section 5 presents the paper 1 [30].

2 Expert Systems techniques

Most widely used technique in the field of fault localization and diagnosis are expert systems as they try to mimic the actions of human expert. Most expert systems use rule based system as their inference engine. [4],[5] and [6] are examples of expert systems.

The expert systems developed differ in the knowledge they use. Rule based fault localization solely depend on the structure of the knowledge base as rule definition language. In [7] the knowledge base is divided to reusable knowledge modeled as core knowledge and customized knowledge. in [5] the rules are organized as composite events and an intervention of human expert is required to update the event base. The rule based systems can act as a powerful tool to eliminate least likely hypothesis. Although the RBR paradigm is appropriate for problem-solving tasks that are confined and well-understood its limitations are.

- an in-ability to learn from experience.
- Fan inability to deal with novel problems.
- the difxulty of updating the systems to keep up with rapidiy changing domains such as expanding heteroge-

neous network.

in Model bases expert systems like [6] conditions are ususally accosiated with rules which includes predicates referring to system model. These predicates test the syetem for existence of relationship among system component. [6] uses correlation tree skeletons describing cause and effect relation ships between event. regardless of what expert system is being used localization process is always driven by inference engine and correaltion rules between events.

[8] and [9] are examples of Case-based reasoning systems. The goals of CBR systems are (i) to learn from experience, (ii) to offer solutions to novel problems based on past experience, and (iii) to avoid expensive maintenance. The basic idea of CBR is to recall, adapt, and execute episodes of former problem-solving in an attempt to deal with a current problem. Former episodes of problem-solving are represented as cases in a case library. When confronted with a new problem, a CBR system retrieves a similar case and tries to adapt the case in an attempt to solve the outstanding problem. The disadvantages of such a system is the time complexity in retrieving a case that is the closest match to the event and the close tailoring of the application to the domain.

In addition to the above mentioned techniques there are other notable techniques are neural networks based approaches [10] [11] and decision tree based approaches [12]. Neural networks have parallel computing architecture and are very fast avoiding bottlenecks which commonly arises from serial processing. But the main disadvantage is that the learning process requires intesive training and in communication networks where all the alarms signatures are not readily available for pattern recognition.

Decision tree aproaches are simple and allows expressive representation of expert knowledge howeever they are limited by the dependencies sepcific applications nad degraded accuracy in noisy scenarios [13] [14].

3 Model traversing techniques

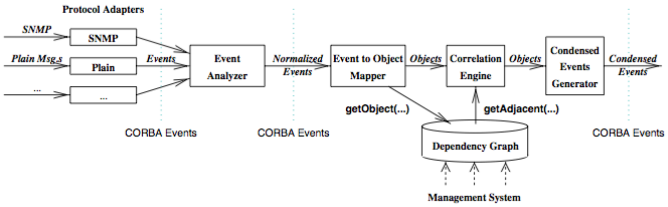
[16], [17], [18] and [19] are all Model Traversing that use formal representation of communication system with clearly marked relationships across network entities. This technique identifies faults by traversing a model of entities starting form the entity whic reported an alarm and also involves a fault identification process to identify and locate faulty network entities.

The model representation used by this technique is object-oriented representation of the given system. It is based on the OSI management framework and uses GDMO (Guidelines for Definition of Managed Objects) with extensions to the MO to model dependencies and services between the entities. This approach can make automated testing possible which can test for various testing entities availability and quality of service standards.

The event correlation in model traversing techniques is event driven as when an event occurs it is being mapped to the reported entity and the managed object representing that entity. For every event the search begins from the MO reporting the event and is searched recursively forllowing all relationship links between MO's using fault localization algorithms. Fault localization algorithms used in this technique can be of various flavours. models every event as singleton classes and merges them whenever they are traced to the same node. Typical proerties of such techniques involve

- Level at which a particular fault has occurred in the model.
- Event Type.
- Event severity.
- Event Origin - to identify if the event occurred is primary or secondary.

Figure 3. Architecture of a model traversing technique prototype



The MO's provide functions which lets the process test the entity for its operational status. The root cause is found when the process stops at an entity and validates that it is a malfucntioning object and doesn't depend on any other object. In a multi layer model vertical search is performed first and then horizontal search in the next lower layer to check its peer at the end of search process votes are collected to identify the faulty elements and the device that recieves the most votes is declared faulty and a testing process kicks in to test the validity of the hypothesis.

Model traversing techniques are pretty robust agianst network configuration changes and very useful in the scenario's where automated testing is a requirement and the depedency model is very natural as they are modeled as Trees or Graphs.

The biggest drawback is the Fault Propagation Patterns and when there fault is a logical combination of multiple devices and byzatine problems. It also incurs heavy testing costs.

4 Graph-theoretic techniques

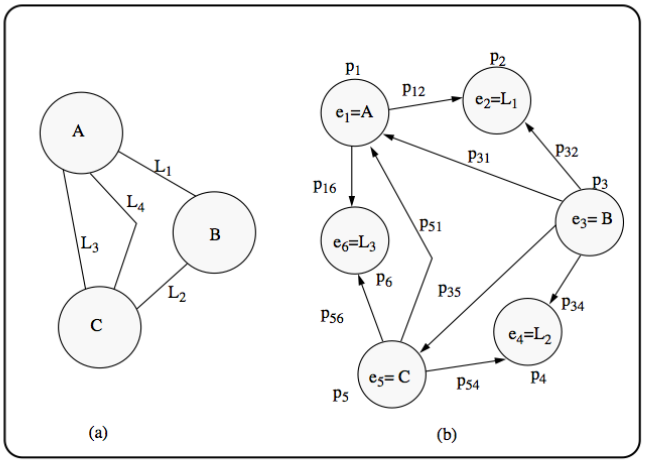
Grap theoretic techniques depend on gaphical model of a system called fault proppagation model(FPM). FPM represents fault and symptoms that occur in a system. The symptoms that are observed are mapped into nodes in the FPM and localization algorithms analyze the graph to infer the best explanation of the observed symptoms.

These techniques require a knowlede of depedencies among the abstract and physical system components and how these failures in one component are related to other components. The success of the fault localiztion algorithm depends on the the accuracy of this a priori specification.

FPM's are genrally modeled as (i) Cuasaultiy graph is is a DAG whose nodes are events and edges represent the causality implications i.e cause effect relationships between events. They are carry probabilties to the nodes and edges implying the probability of a certain event to happen. Dependency

Graph is a directed graph *G=(O,D)* where O represent a finite set of nodes and D are the depedency edges between Objects. The directed edge (o_i,o_j) ∈ D represents that when o_i fails o_j fails or emits some kind of error. The edges are also labeled with conditional probabilities. Figure 4 represents such a depedency graph.

Figure 4. A Network example along with the dependecy graph [24]



A lot of approaches using dependency graph, a critical assumption involves that an entity may fail in only one way. If that is the case the causal graph and dependecy graph would be the same. In the case of multiple failures associated to a single entity they can be divided into sets such as *complete failure, abnormal transmission delay, high packet loss, etc.*. The dependecy graph in this case would have multiple edges between failure modes and the Object(entity). Sethi et all in [23], [22], [21] have modeled using multiple failure modes principle.

A fault localalization algorithm, based on the provided FPM generally returns a number of hypothesis that best explain the observed symptoms. The problem of finding best explanation of the recieved alarms is *NP-hard* problem and proved in [24] by reducing the given problem to a Min Set-Cover problem.

4.1 Divide and conquer algorithm

The divide and conquer algorith uses a FPM assuming that one type of failure is allowed per object. It is a window based technique i.e gathers all the faults in a time window and analyzes them.

System model is a directed graph G = (E ,D) called the dependencygraph, where E is a finite nonempty set of active terminal objects e_i, and a directed edge (e_i, e_j) ∈ D denotes the fact that a fault in e_i has a side effect on e_j i.e., e_j is dependent on e_i. Each vertex e_i is assigned a weight p_i that is the probability that the object e_i fails independently of the state (failed or not failed) of any other active terminal object that is dependent on it. Each directed edge (e_j,e_i) is assigned a weight p_{ji} which represents the strength of dependency between the entities it connects. Specifically, p_{ji} , is the conditional probability p_{ji} = P(e_j fails|e_i fails) that entity e_j, fails as a result of the failure of entity e_i. In addition, the weight of a directed path is defined as the product of all the weights of the edges that compose the path. Finally, the strength of dependency p_{kl} between two non-adjacent vertices e_k and e_l, is defined as the maximum of the weights of all directed paths from e_k to e_l.

Let D(a_i) for every a_i alarm be the domain of alarms that defines as the set of objects that might have caused the alarm. The domain of an alarm calculation is a variation of single source problem that finds the min cost paths from a source vertex to all other graphs and the nodes are chosen such that the cost of path e_i to e_j is subject to restriction that we include only the minimum cost paths with cost ≤ log(W) where W is a parameter.

The outline of the algorithm run has been given below.

- Start with the set S of objects associated with the alarm cluster.
- Find the partitioning of the set S resulting in two disjoint sets so in each set the objects exhibit the maximum mutual dependency, i.e., for every object e_i and e_j, that belongs to the same set and e_k that does not belong to the set, the dependency weight $p_{ij}$is greater than the dependency weights p_{ik}, p_{ki}, p_{jk}, p_{kj}.
- Of the two sets, select the one that explains all the received alarms and has the maximum probability that at least one of the entities in the set is a fault. If there is no such set then find the subset of alarms that each set explains and select both.
- Apply the previous two steps of the algorithm recursively to the selected sets until the resulting sets of the current partitioning are singletons.

By partitioning with the maximum mutual dependency criterion we group together objects that are highly dependent on each other and when the result of partitionining, both the sets are capable of explaingin all the alarms then we select one set that has higher probability of having one of them at fault.

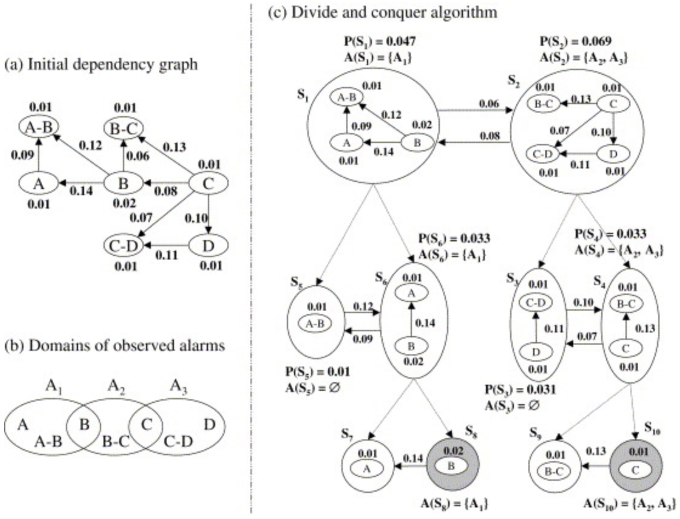
When determining atleast on memeber of the subset is faulty it is calcualted using.

$$P(S) = \sum_{\forall i: e_i \in S} [p_i + \sum_{\forall j: e_j \in S, j \neq i} p_{ij}.p_j]$$

In algorithm runs in two phases where in the first phase algorithm finds the maximum mutual partitionings of the entities in the set S. The result of Phase I is a hierarchical clustering of the entities in the set S. In the Phase II a fault localization algorithm is recursively invoked to find the subsets that are supposed to be used. The first subcluster contains all alarms that may be explained using the subset with the higher probability that one of its members was a primary source of a failure. The second subcluster contains the remaining alarms from the original alarm cluster. Its domain is the other of the two subsets. The recursive procedure is then invoked twice, taking the two subclusters and their respective domains as parameters. The recursion continues until the input alarm cluster domain is a singleton.

The algorithm mentioned above gives the explanation of the observed alarms but may not prouce the best output as

Figure 5. Divide and Conquer algorithm [24]


(a) Initial dependency graph

(b) Domains of observed alarms

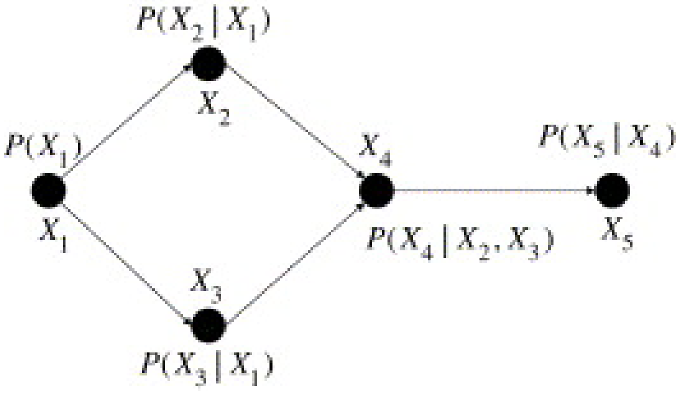
(c) Divide and conquer algorithm

it is a approximation algorithm based on maximum mutual dependency heuristic. The runtime complexity of the algorithm is O(N^3) and the partitioning phase runtime is O(logN). The drawback of the algorithm is it cannot handle the lost or spurious alarms and is a time window based determinsitic algorithm.

4.2 Bayesian Reasoning approach

Bayesian network or probabilistic graphical model represents a set of random varaibles and their conditional dependencies using a directed acyclic graph. In the context of fault localization the random variables represent the state of network entities and the occurrence of network events.

Figure 6. A example bayesian network



Belief networks are used to make four basic queries given evidence set e: belief assessment, most probable explanation, maximum a posteriori hypothesis, and maximum expected utililty [25]

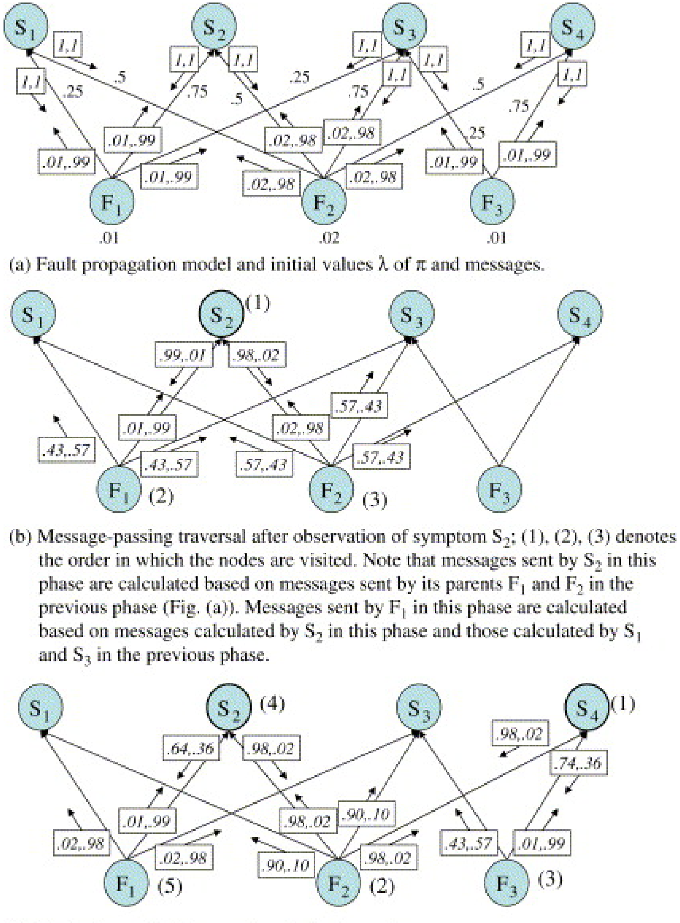
[26], [23], [22] and [21] all the papers try to model the FPM as a bayesian network using a layered fault model aparoach and in general fault component models are divided into services, protocols and functions. The recursive dependencies between services, protocols and functions constitue a dependenct graph. Uncertainty about dependencies between communication system entities is represented by assigning probabilities to the links and/or nodes in the dependency or causality graph.

The queries for belief assesment and most probable explanation are of particular interest in the approaches mentioned above. The belief assesment task is to compute $bel(V_i = v_i) = P(V_i = v_i|e)$. given an evidence e, *bel* gives the certainity of an event happening under the evidence. The most probable explanation task is to find an assignment that best explains the observed evidence e, i.e.,

$P(A_{max}) = max_A \overset{n}{\underset{i=1}{\pi}} P(V_i = v_i^A | Par(V_i)^A)$ [25].

It is known that the above tasks are NP-*hard* in the regular belief networks. A belief updating algorithm, polynomial with respect to |V|, is available for *polytrees*, i.e., directed graphs without undirected cycles. The exact calculation of the best explanatory hypothesis requires a number of steps that is exponential with respect to the number of graph nodes. One of the most popular exact algorithms bucket elimination is used as a reference algorithm against *iterative belief updating* algorithm and *iterative MPE query* algorithm.

Figure 7. Bayesian Network message passing algorithm example


(a) Fault propagation model and initial values λ of π and messages.

(b) Message-passing traversal after observation of symptom S$_2$; (1), (2), (3) denotes the order in which the nodes are visited. Note that messages sent by S$_2$ in this phase are calculated based on messages sent by its parents F$_1$ and F$_2$ in the previous phase (Fig. (a)). Messages sent by F$_1$ in this phase are calculated based on messages calculated by S$_2$ in this phase and those calculated by S$_1$ and S$_3$ in the previous phase.

(c) Graph traversal after symptom S$_4$ is observed.

The algorithms utilizes a message passing schema in which BN nodes exchange messages that encode certain conditional probabilities. The message that node X sends to its parent V_j for every valid V_j's value v_j is denoted by $\lambda_X(v_j)$. The message that node X sends to its child U_i for every valid value of X, x, is denoted by $\pi_{U_i}(x)$. Messages $\lambda_X(v_j)$ and $\pi_{U_i}(x)$ are calculated by node X based on messages it receives from its neighbors using the following equations (where β is any constant and α is a normalizing constant) based on messages the *bel(x)=*$\alpha\lambda(x)\pi(x)$.

The alogirthm proceeds in an event driven manner when a symptom is observed and applying a message passing iteration traversing the graph based on an order. Figure 7 shows an example bayesian network with fault nodes F_1, F_2, F_3. Once the algorithm ends every node is assigned a belief probability, the probability of its existence given the symptoms. The final hypothesis is chosen using the following heuristic: (1) a most-likely fault is chosen and placed in the final hypothesis, (2) the chosen fault node is considered a part of evidence, and (4) one iteration of message passing starting from the chosen fault node is performed. Steps (1)(3) are repeated as long as (1) the posterior distribution contains fault nodes whose probability is greater than 0.5, and (2) unexplained negative symptoms remain. An inherent property of the adapted algorithm is the capability to isolate multiple simultaneous faults even if their symptoms overlap.

However one limitation to the modeling of the domain as bayesian network is the model has to adopt canonical models used in the literature mentioned above like noisy-OR gates, AND gates to limit the algorithm runtime to polynomial time. The comparision of different algorithms in bayesian networks is mentioned in the table 1 below.

[29] Shrink a tool for failrue diagnosis in IP Networks. Shrink outputs the most likely explanation for the network's faulty state i.e., the collection of SRLGs whose failure is most likely given the IP link status. Shrink works on three modules (1) building the bayesian model (2) augemnting the model with guess edges (3) inferring the most likely explanation.

Figure 8. Shrink system setup

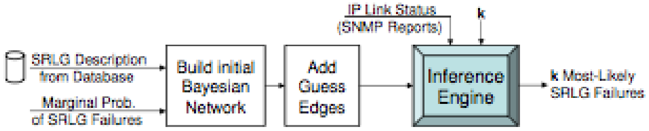
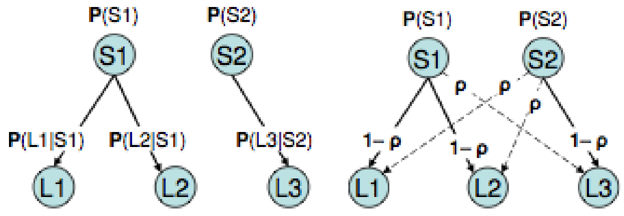


Figure 9. Shrink's network model



(a) Initial Bayesian Network **(b) With Guess Edges**

if E_k denotes the event that at most k SRLGs failed within any hour, there are exactly k combinations of n possible SRLG assignmentsthat satisfy E_k. Given the observed link status, shrink algorithm greedily picks the most likely SRLG assignment in E_k; i.e., it picks $(S_1, \ldots\ldots, Sn) \in \{0,1\}^n$ such that:

$arg \underset{S_1,\ldots\ldots,S_n}{max} P(S_1, \ldots\ldots, S_n|L_1, \ldots.L_n)$ subject to number of $\{S_i{=}1\} \leq$ k

There are fewer than n^k value assignments in E_k, and for each assignment $P(S_1, \ldots, S_n|L_1, \ldots, L_m)$ can be computed in O(m + n) time, where m is the number of IP links and n is the number of SRLGs. So, Shrinks running time is O(n^k (m+n)). k=3 is a valid case for a wide range of ISP network sizes and SRLG failure probabilities and when k≥3 the hypothesis performance is exponential.

4.3 Misc approaches

There are several other approaches to fault localization like codebook approach which uses hamming distances to find the fault and also context free grammer which allows xpressions to be built from subexpressions, may be effectively used to represent a hierarchically organized communication system. However these are not discussed in this papaer due to the poor applicability to the real world systems and poor performance of these approaches. [27], [28], [3] are papers that discuss about these techiniques.

5 Paper 1: Toward Optimal Network Fault Correction in Externally Managed Overlay Networks

[30] Paper 1 is an end-to-end approach of inferring probabilistic data-forwarding failures in an externally managed overlay network, where overlay nodes are independently operated by various administrative domains. The optimization goal is to minimize the expected cost of correcting (i.e., diagnosing and repairing) all faulty overlay nodes that cannot properly deliver data. Instead of first checking the most likely faulty nodes as in conventional fault localization problems it chooses a *candidate node* proceed based on a potential heuristic function and is bound to infer correctly atleast 95% of the time.

[24], [21] and [29] state of the art techniques that diagnose (i.e detect and localize) the components that are root causes of the network faults, however they do not focus on the *network fault correction* that is not only diagnosis but also considering the repair and checking costs. [30] conveniently outperforms all of them and tries to derive a strategy for efficient network fault correction at minimum cost.

Another set of research papers focus purely on the failure detection of network faults [31], [32] and [34] uses distributed techniques and all networks nodes to collaboratively achieve the fault detection. [34] uses inspection on packets from previous hop and reports when founded corruption. The problem with this approach is that an inspection logic or monitoring points have to be setup across the network and incurs heavy costs and different administration domains prove to be obstacle in the above cases too.

Paper1 considers and end-end inference approach using end-end measurements, infers components that are faulty in

Algorithm	Bucket Elimination	Iterative Belief Updating	Iterative MPE
Theoretical Bound	n^2exp(n)	n^5	n^6
Detection Rate	96-100%	93-98%	95-100%
False Positive Rate	0-4%	2-5%	0-8%
Max Network Size	10	50	25
Algorithm iterative?	no	yes	yes

Table 1. Comparision of algorithms

forwarding data. This applies perfectly to the scenario like CAN[35] and CHORD[36] where each overlay node builds a routing tree with itself as a root and in this setting every root to leaf path is monitored to find faulty path and anamolous behaviours on the path. One anamoulous behaviour could be number of correct packets bot being delivered in a fixed time period, such a validation could prove to be very important in QOS factor. The authors are particular interested in diagonize and repairing faulty nodes in overlay networks like RON[37], SON[38] and SOS[39]. The proposed fault correction mechanism suits well to this type of networks.

5.1 Problem Formulation

[30] considers a logical tree as T=$(N, \{p_i\}, \{c_i\})$, where N is the set of overlay nodes, p_i is the failure probability of node i∈N, and c_i is the checking cost of deciding if node i∈N is faulty. Overlay set node N provides a topology information and sequence of nodes. The construction of failure probabilities and cost probabilities depend on Failure model and Cost model.

Failure Model focus on the nodes that cannot forward data or fails to comply with a QOS because of which packets get dropped or fails to get delivered. Focus mainly stands on fail-stop failues like power outages, full transmission queues, hardware errors and route misconfigurations. The failure probabilities are calculated using statistical measurements of reliability indexes of node elements(vulnerability modelling). Authors also performed analysis of the mechanism under inaacurate probabilities and is provide in the results section of this paper. Important consideration to be made is that failure probabilities of nodes are independent of each other.

Cost Model provides the checking costs using personnel hours, wages and time effort that goes into debugging the problem or even the cost of the equipment. Important cirteria to be noted in the cost model the repair costs are not conisdered as eventually every fault has to be repaired. p_i and c_i can be any values between [0,1] and [0,∞].

Each node in a logical tree T is classified as fault or non-faulty. and definitions are being made on each root to leaf path that exhibits an anamalous behaviour. Since an end-to-end inference mechanism is being made we only know that something is wrong with the path rather than which node is at fault. Each node in T is referred to as *bad* if it is faulty and *good* if it is non-faulty and a path in T is bad if it has atleast one *bad* node. An important consideration is that the node behaviour remains the same across all the paths as only fail-stop considerations are being made.

Given a logical tree T we can infer the good path and bad paths by end to end inference mechanism, for example root can send a probe and collect all measurement results. We now create a bad Tree by pruning of all the good paths form the logical tree and also call it T with abuse of notation. Bad tree now becoms input to the inference algorithm which determines which set of nodes to be corrected(checked and repaired).

The optimization goal of this paper is to minimize the expected cost of correcting all faulty nodes in a given tree. repairing cost is not considered as eventually all the nodes have to be repaired and any strategy that is employed will eventually correct all the nodes. Here we focus only on the sequential case as checking the nodes in parallel doesnt improve the total expected checking cost so the inference algorithm only return one single node that is supposed to be the best node to check first. The best node is the first node that is from a diagnostic sequence $S = \{l_1, l_2, ..., l_N\}$ and gives us *N!* diagnositic sequences.

When a particular node l_i lies on a bad path we do no know whether it is a good node or a bad node but if it is on a good node we know for sure it is a good node. l_i gets *checked or skipped* based on the examination of the node. Thus we can summarize the total cost of checking a tree T is given by

$$\sum_{i=1}^{|N|} c_{l_i} Pr(node\ l_i\ is\ checked \mid nodes\ l_1, ..., l_{i-1}\ known\ to\ be\ good)$$

So the best node would be the first node in the diagnosis sequence and the inference works on the current topolgy to indetify this node and when the node is corrected the revised topology is fed into the inference algorithm to output the next node. The total complexity of calculating the expected cost of a diganosis sequence is O(n^3) and since there are n! differenct sequences, the brute force approach has a complexity of O(n^3n!). The complexity and the way to calculate the total cost for the diagnostic sequence is mentioned in Appendix A of [30]. In the interest of space and lenth only the highights are being mentioned here.

Conditional Failure Probabilties: the nodes in T by 1 to |N| in breadth- first-search order. T_i is the subtree rooted at node i. C_i is the set such that k∈C_i if node k is a child node of *i*. thus conditional failure probability is given by bayesian probability rule where Pr(X_i) is the independent failure probability of a node, and Pr(T_1) is the probability of the tree is bad at root, $Pr(T_1|X_i)$ is the probability that a tree is a bad tree upon the event that a node is bad.

$Pr(X_i|T_1) = \frac{Pr(T_1|X_i)Pr(X_i)}{Pr(T_1)}$ by baye's rule.

where

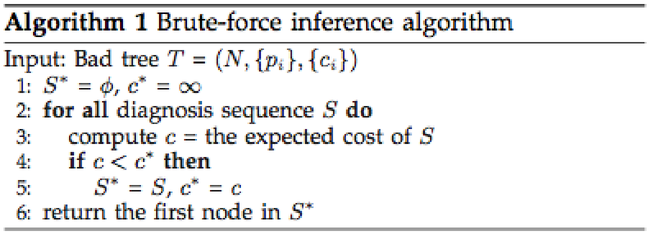
$Pr(T_i) = p_i + (1 - p_i) \underset{k \in C_i}{\pi} Pr(T_k), \forall i \in [1, |N|]$

$$Pr(T_i|X_j) = \begin{cases} Pr(T_i) & \text{if } i > j \\ 1 & \text{if } i = j \\ p_i + (1-p_i) \underset{k \in C_i}{\pi} Pr(T_k|X_j) & \text{if } i < j \end{cases}$$

The idea here is that a subtree T_i is bad if either the node itself is bad or node i is good and the subtree rooted at i is bad. Using dyanmic programming the above equation can be computed in O(N^2) which is the conditional failure probabitlity of a node.

Expected Cost of a Diagnosis Sequence: Let $T_{l_i}{}^D$ be the event that the subtree rooted at node l_i is a bad tree after nodes $l_1, ..., l_{i-1}$ have been examined (i.e., checked or skipped). Let $A_{l_i}{}^D$ be the event that every ancestor j of node l_i, such that j $\in \{l_{i+1}, ..., l_{|N|}\}$ (i.e., node j is examined after node l_i), is a good node. Since every node is examined in a sequential way in the diagnosis sequence, when l_i is examined, nodes $l_1, ..., l_{i-1}$ have been examined and they are good. Thus,
$Pr(T_{l_i}{}^D) = Pr(T_{l_i}|p_{l_1} = ... = p_{l_{i-1}} = 0)$, and
$Pr(A_{l_i}{}^D) = Pr(A_{l_i}|p_{l_1} = ... = p_{l_{i-1}} = 0)$
Where T_{l_i} is the event that the subtree rooted at node l_i is bad, and A_{l_i} is the event that all ancestors of node l_i are good. If the subtree rooted at node l_i remains a bad tree or at least one of the ancestors of node l_i is a bad node, then node l_i still lies on bad path and needs to be checked. Thus,
$Pr(node\ l_i\ is\ checked|T, nodes\ l_1, ..., l_{i-1} known\ to\ be\ good)$
$= Pr(T_{l_i}{}^D \cup \neg A_{l_i}{}^D|T)$
Thus the expected cost of S given that T is a bad tree is:
$\sum_{i=1}^{n} c_{l_i} Pr(T_{l_i}{}^D \cup \neg A_{l_i}{}^D|T)$
Brute force algorithm which inspects all the sequences is given in figure 10. This algorithm makes a good reference for comapring against the heuristic based ones.

Figure 10. Bruce Force Algorithm



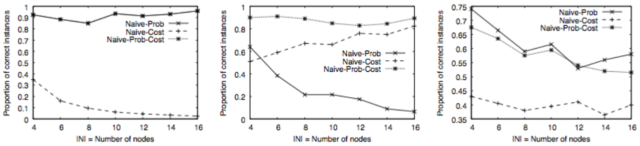
Algorithm 1 Brute-force inference algorithm

```
Input: Bad tree $T = (N, \{p_i\}, \{c_i\})$
$S^* = \phi$, $c^* = \infty$
for all diagnosis sequence $S$ do
    compute $c$ = the expected cost of $S$
    if $c < c^*$ then
        $S^* = S$, $c^* = c$
return the first node in $S^*$
```

5.2 Naive Heuristics for Inference Algorithm

Intuitively the best node returned by the inference algorithm can be either based on the highest conditional failure probabiltiy or checking cost. The authors provided a simple counter example in which neither the selection of node based on conditional failure probability nor the one selected based on checking cost gave the optimal solution. The three heuristics authors chose naively are (1) *Naive_Prob* (2) *Naive_Cost* (3) *Naive_Prob_Cost* . Based on the experimental setup where a 200 bad trees are tested with the three heuristic mentioned above with varying probabiltiy and cost distributions. Depending on the distributions of p_i and c_i, the proportion of instances where the best choice is made can be as low as 10% for Naive-Prob and Naive-Cost and less than 55% for Naive-Prob-Cost.

Figure 11. Naive-Prob, Naive-Cost, and Naive-Prob-Cost performance



(a) $p_i \sim U(0,1), c_i = 1$ (b) $p_i = 0.1, c_i \sim U(0,1)$ (c) $p_i \sim U(0,1), c_i \sim U(0,1)$

5.3 Candidate Node

The authors show that instead of selecting a node to test based on naive choices we should select a *candidate node* based on the maximization of a *potential function*. Let A_i be the event that the ancestors of node i are all good. If node r is the root node, then we let A_r be always true and $Pr(A_r) = 1$. The potential of a candidate node is calculated using the formula.
$\phi(i,T) = \frac{Pr(T|X_i,A_i)p_i}{c_i(1-p_i)}$

Intuitively the candidate node should have low conditaional failure probability and low checking cost and a higher likelihood of having a bad tree with respect to tree topology. The potential of node i can be obtained by first computing $Pr(T|X_i,A_i) = Pr(T|p_i = 1, p_j = 0 \forall j \in A_i)$, where A_i is the set of ancestors of node i. and the time complexity of comuting is O($|N|^3$) For each path W in a tree T we select node i that has the highest potential $\phi(i,T)$ among all the nodes. In case of a tie we choose the one closest to the root.

5.4 Optimality Results

The optimality results that were presented in the paper drive the portential function definition provided above.

Theorem 1: Given a bad tree T, there always exists an optimal diagnosis sequence that starts with a candidate node.

Proof of Theorem 1: Consider two diagnosis sequences $S_j = (j, k, l_3, , l_{|N|})\ and\ S_k = (k, j, l_3, , l_{|N|})$. Now two scenarios crop up (1) when j and k are ancestrally related i.e they are on the same path and (2) when j and k are on two different paths. In the second case when they are unrelated they have no impact on each other. thus S_j and S_k will have the same cost. When j and k are ancestrally related checking node j (resp k) if it reveals a good path then k (resp j) can be skipped checking and save c_k (resp c_j). But to validate the given statement we need certain conditions to hold good.

- node j (resp. k) is bad.
- node k (resp. j) is good.
- there exists a path W_{jk} containing nodes j and k such that all nodes other than nodes j and k on W_{jk} are good.

Thus, $E[cost\ of\ S_j|T] - E[cost\ of\ S_k|T]$
$= \frac{c_j c_k Pr(W_{jk}) Pr(X_j) Pr(X_k)}{Pr(T)} [-\phi(j,T) + \phi(k,T)]....(*)$
From (*) we can deduce that for expected cost of sequence starting with j whoudl be no greater than that of k when

$\phi(j,T) \geq \phi(k,T)$, therefore the node with the highest potential should be checked first.

Corollary 1: Given a bad tree T which is a single path, the best node is the one with the maximum $\frac{p_i}{c_i(1-p_i)}$.

Proof of Corollary 1: $Pr(T|X_i,A_i) = 1$ for all i. Thus, the best node will be the only candidate node whose potential is $max_i \frac{p_i}{c_i(1-p_i)}$

Corollary 2: Given a bad tree T, if every node i has p_i = p and c_i = c, where p and c are some fixed constants, then the best node is the root node of T.

Proof of Corollary 2: The above statement is true since the root node is the node with the highest conditional probability under this scenario.

Corollary 3: Consider a two-level tree T whose root node is attached to $|N|-1$ leaf nodes. If T is a bad tree and every node i has c_i = c for some constant c, the best node is the one with the maximum conditional failure probability.

Proof of Corollary 3: If root node has the maximum conditional probability then for all leaf nodes we can show that $Pr(X_r|T) \geq Pr(X_i|T) \leftrightarrow \frac{p_r}{Pr(T)} \geq \frac{p_r p_i + (1-p_r) \underset{j \neq i}{\pi} p_j}{Pr(T)} \leftrightarrow \frac{p_r}{c(1-p_r)} \geq \frac{\underset{j \neq i}{\pi} p_j}{c(1-p_i)} \leftrightarrow \phi(r,T) \geq \phi(i,T)$ where $Pr(X_r|T)$ is the maximum conditional probability for failure of root node. The above equations imply that root ndoe is the best candidate to check since it has the highest potential.

The second case for corollary 3 is when a non leaf node has the maximum conditional probability and let S^* is an optimal diagnosis sequence that doesnt start with k, so let $S^* = (l_1, l_2, l_3, ...k, l_{j+1}, ..., l_N)$. Here we need to show that moving node k to front doesnt increase the expected cost of S^*.

- Case 1: Node l_1 is the root node r: Let $S^1 = (k, r, l_2, l_3, ...l_{j-1}, l_{j+1}, ..., l_N)$ all other nodes except the root and node k are checked in both the sequences and doesnt effect the outcome of this proof. so we can safely ignore them. We skip node r(resp k) in S^1 (resp S^*) and save cost c in S^1 and S^*. E[cost of S^1|T]-E[cost of S^*|T] = -cPr($\overline{X_r}$|T)+cPr($\overline{X_i}$|T) is maximum.
- Case 2: Node $l_1 \neq r$ is a leaf node: In S^* let l_2 be the root node and when l_1 is checked all other leaf nodes potential drops to zero because of the conditional probability $p_i|p_{l_1}$ and root ndoe is the only candidate left. We can make similar argument as case 1 with $S^2 = (k, r, l_2, l_3, ...l_{j-1}, l_{j+1}, ..., l_N)$

There exists a diagnosis sequence that starts with node k and has no greater expected cost than does S^* , node k is a best node

5.5 Choosing a candidate node is difficult?

The authors have provided counter examples that violate the intuiton and why choosing a candidate node is difficult.

Counter-example 1: Given a bad tree, the best node is not the one with the highest potential.

Counter-example 2: Checking simultaneously all candidate nodes in a bad tree does not minimize the expected cost of correcting all faulty nodes.

Counter-example 3: The best node for a bad tree is not necessarily the best node for a subtree.

Because of the above counter examples we can show that the problem cannot be solved by recursively solving the suboptimal solutions which is proved by counter example 3 and counter example 2 is a nice example of how conditional probabilites change once an information of a node whether it is good or bad is revealed.

5.6 Evaluation of candidate based heuristics

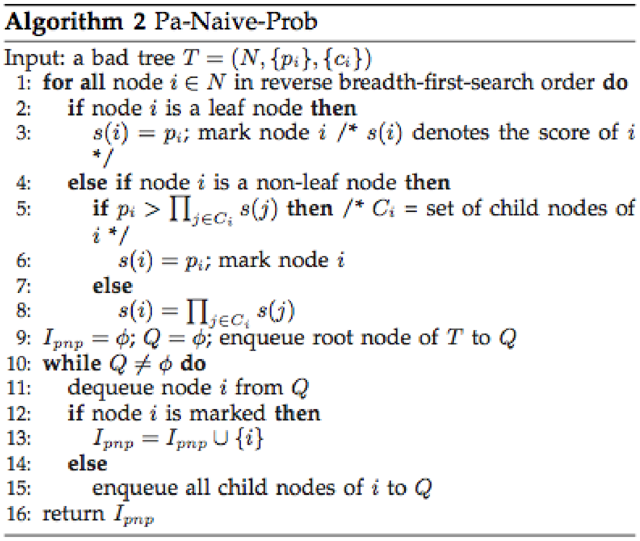
The heuristics that have been experimented and results are obtained are (1) *Cand-Prob* (2) *Cand -Cost* and (3) *Cand-Pot*. The three heuristics select the candidates based on the criteria of Cost, Probability and Potential. Results have shown that *Cand-Pot* has outperformed all other heuristics and produced results in 95% of the cases.

The potential of a node can be calulated in $O(|N|^2)$ time. However *Cand-Pot* which searches all the nodes for the one with highest potential and that would lead to a complexity of $O(|N|^3)$. This could lead to scalability issues when network grows, but generally only a subtree of network is bad so it is likely to be in a manageable size.

5.7 Heuristics for inference algorithm

Three different groups of efficient heuristics have been suggested for large scale networks. the heuristics belongs to one of these (1) naive heuristics which considers the maximum conditional probability (2) candiate based heuristics which considers both conditional failure probability and cost.

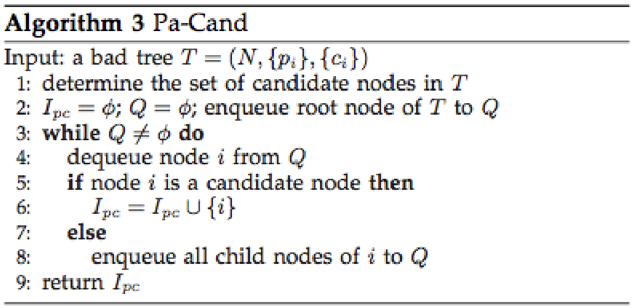
Single Node Inference for a Single Tree chooses two heuristics Naive-Prob and Cand-Pot. Multiple Node Inference for a Single Tree will try to check for multiple nodes in parallel to reduce the time needed to repair all the bad nodes. A parallel extension to naive-probability and cand-pot is given which returns the most likely subset of nodes that cover all the bad paths. They are names *Pa-Naive-Prob* and *Pa-Cand* respectively which returns subsets I_{pnp} and I_{pc} of faulty nodes that cover all the bad paths. An algorithm in Figure 12 determines I_{pnp} in $\theta(|N|)$ complexity. $I_{pnp} = argmax_I Pr(X_I|T) = argmax_I \underset{i \in I}{\pi} p_i$ since $Pr(T|X_I)$ is one.



Algorithm 2 Pa-Naive-Prob

```
Input: a bad tree $T = (N, \{p_i\}, \{c_i\})$
for all node $i \in N$ in reverse breadth-first-search order do
    if node $i$ is a leaf node then
        $s(i) = p_i$; mark node $i$ /* $s(i)$ denotes the score of $i$ */
    else if node $i$ is a non-leaf node then
        if $p_i > \prod_{j \in C_i} s(j)$ then /* $C_i$ = set of child nodes of $i$ */
            $s(i) = p_i$; mark node $i$
        else
            $s(i) = \prod_{j \in C_i} s(j)$
$I_{pnp} = \phi$; $Q = \phi$; enqueue root node of $T$ to $Q$
while $Q \neq \phi$ do
    dequeue node $i$ from $Q$
    if node $i$ is marked then
        $I_{pnp} = I_{pnp} \cup \{i\}$
    else
        enqueue all child nodes of $i$ to $Q$
return $I_{pnp}$
```

The algorithm of finding I_{pc} is shown in Figure 13, whose complexity is $\theta(|N|^3)$ due to the search of candidate nodes.



Algorithm 3 Pa-Cand

```
Input: a bad tree $T = (N, \{p_i\}, \{c_i\})$
determine the set of candidate nodes in $T$
$I_{pc} = \phi$; $Q = \phi$; enqueue root node of $T$ to $Q$
while $Q \neq \phi$ do
    dequeue node $i$ from $Q$
    if node $i$ is a candidate node then
        $I_{pc} = I_{pc} \cup \{i\}$
    else
        enqueue all child nodes of $i$ to $Q$
return $I_{pc}$
```

Multiple node inference for multiple trees involve which runs the inference algorithm on mutliple trees which form a directed acyclic graph. Since the problem of finsing nodes which best explains all the bad paths can be reduce to min set cover problem it is NP-*hard* and therefore we invoke *Mt-Pa-Naive-Prob and Mt-Pa-Cand* which internally invokes *Pa-Naive-Prob and Pa-Cand* on the each tree and the result in unioned.

Experiment results are presented in figure 14.

A References

[1] I. Katzela Fault diagnosis in telecommunications networks, Ph.D. Thesis. School of Arts and Sciences, Columbia University, New York, 1996.

[2] I. Katzela, A.T. Bouloutas, S.B. Calo Centralized vs distributed fault localization. A.S. Sethi, F. Faure-Vincent, Y. Raynaud (Eds.), Integrated Network Management IV, Chapman and Hall, London (1995), pp. 250263.

[3] S.A. Yemini, S. Kliger, E. Mozes, Y. Yemini, D. Ohsie. High speed and robust event correlation. IEEE Communications Magazine, 34 (5) (1996), pp. 8290.

[4] Peng Wu, Rajiv Bhatnagar, Lennie Epshtein, Malini Bhandaru, Zhongwen Shi. Alarm Correlation Engine (ACE). GTE Laboratories Incorporated.

[5] G. Liu, A.K. Mok, E.J. Yang. Composite events for network event correlation - JECTOR. Integrated Network Management VI, IEEE (1999), pp. 247260.

[6] Y.A. Nygate. ECXPERT: Event correlation using rule and object based techniques. Integrated Network Management IV, Chapman and Hall, London (1995), pp. 278289.

[7] K.-W.E. Lor. A network diagnostic expert system for Acculink multiplexers based on a general network diagnostic scheme. Integrated Network Management III, North-Holland, Amsterdam (1993), pp. 659669.

[8] L. Lewis. A case-based reasoning approach to the resolution of faults in communications networks. Integrated Network Management III, North-Holland, Amsterdam (1993).

[9] R.D. Gardner, D.A. Harle. Methods and systems for alarm correlation. Proc. of GLOBECOM, London, UK, November (1996), pp. 136140.

[10] R.D. Gardner, D.A. Harle. Alarm correlation and network fault resolution using the Kohonen self-organizing map. Proc. of IEEE GLOBECOM, Toronto, Canada, September (1997).

[11] R.D. Gardner, D.A. Harle. Pattern discovery and specification techniques for alarm correlation. NOMS98, Proc. Network Operation and Management Symposium, New Orleans, LA (1998), pp. 713722.

[12] G.D. Rodosek, T. Kaiser. Intelligent assistant: User-guided fault localization. Ninth Intl Workshop on Distributed Systems: Operations and Management, University of Delaware, Newark, DE, October (1998), pp. 119129.

[13] S. Russell. Machine learning. Artificial Intelligence, Handbook of Perception and Cognition (second ed.), Academic Press, New York (1996), pp. 89133 (Chapter 4).

[14] Daphne Koller, Nir Friedman Probabilistic Graphical Models. MIT Press.

[15] M. Steinder and A.S Sethi. The present and future of event correlation: A need for end to end service fault localization. In N. Callaos et al. World Multi-Conf. Systemics, Cybernetics and Informatics, Vol XII, Orlando, FL, 2001. pp. 124-129.

[16] S. Ktker. A modeling framework for integrated distributed systems fault management. In C. Popien (Ed.), Proc. IFIP/IEEE Internat. Conference on Distributed Platforms, pp. 187198, Dresden, Germany, 1996.

[17] S. Ktker, K. Geihs. A generic model for fault isolation in integrated management systems. Journal of Network and Systems Management, 5 (2) (1997), pp. 109130.

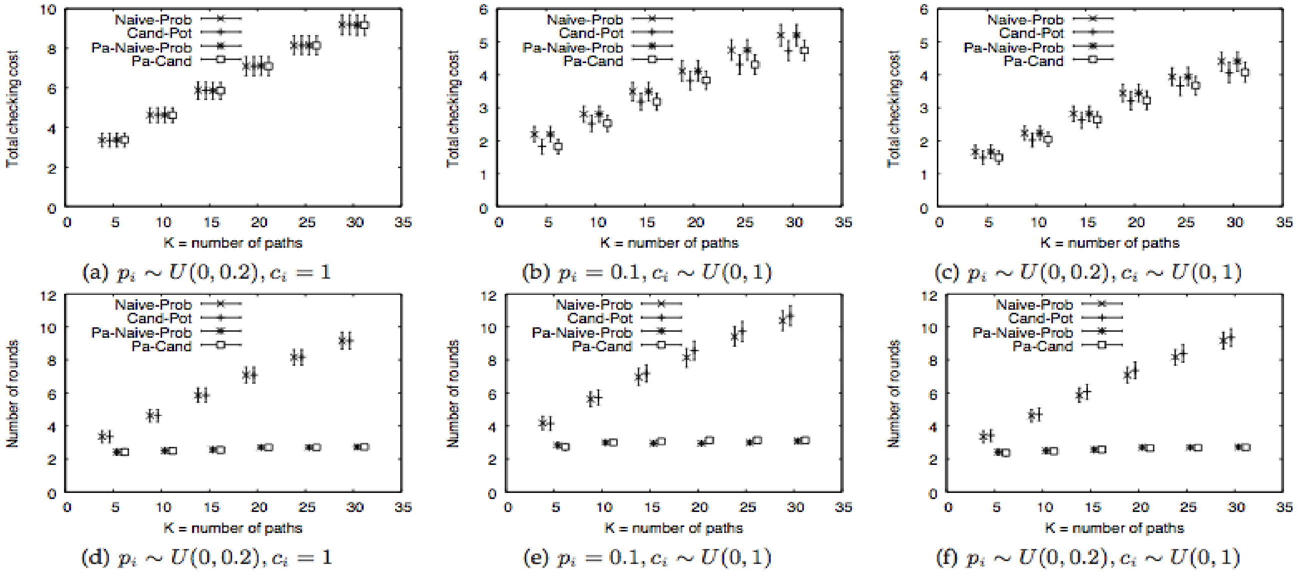
[18] S. Ktker, M. Paterok. Fault isolation and event correlation for integrated fault management. Integrated Network Management V, Chapman and Hall, London (1997), pp. 583596.

[19] B. Gruschke. Integrated event management: Event correlation using dependency graphs. A.S. Sethi (Ed.), Ninth Internat. Workshop on Distributed Systems: Operations and Management, University of Delaware, Newark, DE, October (1998) , pp. 130141.

[20] J.F. Jordaan, M.E. Paterok. Event correlation in heterogeneous networks using the OSI management framework. H.G. Hegering, Y. Yemini (Eds.), Integrated Network Management III, North-Holland, Amsterdam (1993), pp. 683695.

[21] M. Steinder, A.S. Sethi. Probabilistic fault diagnosis in communication systems through incremental hypothesis updating. Comput. Networks, 45 (2004), pp. 537562.

[22] M. Steinder, A.S. Sethi. Non-deterministic fault localization in communication systems using belief networks.



(a) $p_i \sim U(0, 0.2), c_i = 1$
(b) $p_i = 0.1, c_i \sim U(0, 1)$
(c) $p_i \sim U(0, 0.2), c_i \sim U(0, 1)$
(d) $p_i \sim U(0, 0.2), c_i = 1$
(e) $p_i = 0.1, c_i \sim U(0, 1)$
(f) $p_i \sim U(0, 0.2), c_i \sim U(0, 1)$

Figure 12. Comparison of Naive-Prob, Cand-Pot, Pa-Naive-Prob, and Pa-Cand in terms of the total checking cost

IEEE/ACM Transactions on Networking (2004) in press.

[23] M. Steinder, A.S. Sethi. End-to-end service failure diagnosis using belief networks. R. Stadler, M. Ulema (Eds.), Proc. Network Operation and Management Symposium, Florence, Italy, April (2002), pp. 375390.

[24] I. Katzela, M. Schwartz. Schemes for fault identification in communication networks. IEEE/ACM Transactions on Networking, 3 (6) (1995), pp. 733764.

[25] Judea Pearl. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference Morgan Kaufmann Publishers, San Mateo, CA (1988).

[26] Cynthia S. Hood, Chuanyi Ji. Proactive Network-Fault Detection. IEEE TRANSACTIONS ON RELIABILITY, VOL. 46, NO. 3, 1997.

[27] A.T. Bouloutas, S. Calo, A. Finkel. Alarm correlation and fault identification in communication networks. IEEE Transactions on Communications, 42 (24).

[28] S. Kliger, S. Yemini, Y. Yemini, D. Ohsie, S. Stolfo. A coding approach to event correlation. A.S. Sethi, F. Faure-Vincent, Y. Raynaud (Eds.), Integrated Network Management IV, Chapman and Hall, London (1995), pp. 266277.

[29] S. Kandula, D. Katabi, and J.P. Vasseur. Shrink: A Tool for Failure Diagnosis in IP Networks. In ACM SIGCOMM MineNet-05, Aug 2005.

[30] Patrick P. C. Lee, Vishal Misra, and Dan Rubenstein Toward Optimal Network Fault Correction in Externally Managed Overlay Networks. IEEE INFOCOM 07.

[31] S. Q. Zhuang, D. Geels, I. Stoica, and R. H. Katz. On Failure Detection Algorithms in Overlay Networks. In Proc. of IEEE INFOCOM, March 2005.

[32] A. T. Mizrak, Y.C. Cheng, K. Marzullo, and S. Savage. Fatih: Detecting and Isolating Malicious Routers. In Proc. of the IEEE Conference on Dependable Systems and Networks (DSN), June 2005.

[33] I. Avramopoulos, H. Kobayashi, R. Wang, and A. Krishnamurthy. Highly Secure and Efficient Routing. In Proc. of IEEE INFOCOM, March 2004.

[34] I. Avramopoulos, H. Kobayashi, R. Wang, and A. Krishnamurthy. Highly Secure and Efficient Routing. In Proc. of IEEE INFOCOM, March 2004.

[35] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A Scalable Content-Addressable Network. In Proc. of ACM SIGCOMM, 2001.

[36] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications. In Proc. of ACM SIGCOMM, 2001.

[37] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. Resilient Overlay Networks. In Proc. of the 18th ACM Symposium on Operating Systems Principles (SOSP), Oct 2001.

[38] Z. Duan, Z.-L. Zhang, and Y. Hou. Service Overlay Networks: SLAs, QoS, and Bandwidth Provisioning. IEEE/ACM Trans. on Networking, 11(6):870883, Dec 2003.

[39] A. Keromytis, V. Misra, and D. Rubenstein. SOS: An Architecture for Mitigating DDoS Attacks. IEEE JSAC, Special Issue on Service Overlay Networks, 22(1), January 2004.