MBD Techniques for Internet Delay Diagnosis

Roni Stern, Meir Kalech [1]

[1] *Department of Information Systems Engineering, Ben Gurion University, Beer-Sheva, Israel roni.stern@gmail.com, kalech@bgu.ac.il*

ABSTRACT

Internet service providers (ISP) and administrators of Local Area Networks (LAN) aim to provide a certain level of service to end-users. However, network elements such as routers and switches may malfunction, resulting in abnormal communication delays. When such delays are observed, network administrators try to diagnose which network elements cause the abnormal delays. While most common techniques use additional measurements to identify the faulty device, we propose a non-intrusive model-based approach. The first approach translates classical MBD terms to this problem and shows a complete and sound solution. The second approach uses linear programming to produce a single minimum cardinality diagnosis in polynomial time. Both approaches are analyzed and evaluated empirically using the standard NS2 network simulator, and are able to find diagnoses or the minimal cardinality diagnosis in less than half a second for network models with up to 1,000 nodes.

1 INTRODUCTION

With the rise of the Internet and Local Area Networks (LAN), computer networks are growing in size and complexity. As a result, much effort is exerted in maintaining the quality of service provided by the network. This is performed by Internet Service Providers (ISP) in the Internet, and by network administrators in LANs. As a result, much academic and commercial work has been done on diagnosis of abnormal components in computer networks (Kandula *et al.*, 2009).

Network quality of service can be measured by many attributes, such as bandwidth, latency and stability. One of the most important attributes is the delay experienced by the network end-users. This is the time required for data to pass from the source computer to the destination computer. In modern computer networks, data sent from a computer will usually pass via a sequence of network components (e.g. routers) until reaching the destination computer. The components through which data is passed are called the *route* of the data. The delay experience by end-users is affected by several attributes of the components in the route. One of these attributes is the bandwidth of the routers on the route. For example, 200 bytes passing through a router with a bandwidth of 10^6 bits per seconds will be delayed for $\frac{200\times 8}{10^6}$=1.6 milliseconds.

There is much literature on how to infer a network model from measurements (Spring *et al.*, 2002; Shavitt and Shir, 2005; Wei *et al.*, 2003). The result of this inference is a model of the network, including nodes and edges as well as the delays incurred by each edge and node. However, events may occur that disrupt the inferred model, causing unexpected changes in the network behavior. For example, a router may become congested or slightly damaged, resulting in increased delay in all network routes passing through this router.

In this work, we discuss how to diagnose abnormal delays experienced by several end-users. The goal is to identify the set of network components that causes the abnormal delays. Finding these nodes is important to ISPs and LAN administrators, in order to maintain the quality of service to the customers. In addition, such abnormal delays may suggest man-in-the-middle attacks, where the abnormal delay is caused by the processing done by the man-in-the-middle.

The most common technique for detecting which nodes have abnormal delays in a network is to actively measure nodes on delayed routes (Spring *et al.*, 2002; Shavitt and Shir, 2005). Such measurements can be done using network control protocols such as ICMP (ping, traceroute). While this technique is very effective, it has two shortcomings. First, it requires active measurements of the network nodes, adding to the network load. Second, not all of the network components respond to active measurement. For example, it is known that some routers do not respond to ICMP request and other control protocols. In addition, layer 2 network nodes (e.g. ATM switches) usually cannot be measured directly by higher level protocols. Thus in this paper we address the problem of diagnosing delays without intrusive network measurements.

A model-based diagnosis system has been previously proposed for discovering network faults (Sandro T. Fontanini and Maragon, 2002). The diagnosis engine that was proposed queries the network nodes along the path to a network node that returned abnormal behavior, and assumes that there is only a single faulty node in the system. We propose a non-intrusive approach which discovers the diagnosis without actively querying the network nodes. Our approach is orthogonal to active measurement algorithms, and thus could be viewed as a complementing approach. Other work related to model-based diagnosis in networks (Bahl *et al.*, 2007; Kandula *et al.*, 2009) assume knowledge of the network components and try to learn the relationship between them (i.e. which components are affected by every component). In this paper we assume that a model of the network and the connections between its components is given (possibly by a network discovery system), and research how to diagnose network behavior that is inconsistent with this model. In this paper we propose a linear programming based approach to MBD. This is similar to previous work that formulated and MBD problem as an integer programming problem (Fijany and Vatan, 2005). A linear program was even used to bound the search for the solution of the integer programming problem. For the problem described in this paper, we show that a linear programming approach can be used. This is significant as the complexity of solving a linear program is polynomial, while solving an integer program has an exponential complexity.

The contribution of this paper is as follows. First, we show how this problem can be solved by adjusting classical MBD terms and using a corresponding diagnosis engine. Second, we propose a diagnosis algorithm that uses the normal observation to prune candidate network nodes, resulting in substantial speedup in runtime. Then we propose a second solution that uses a linear programming solver to produce the minimal cardinality diagnosis. Finally, we analyze these solutions theoretically and empirically, suggesting that the presented approaches are applicable. Empirical evaluations were run using a standard network simulator NS2 (McCanne *et al.*, 2001) on networks with up to 1,000 nodes. Results show that using our algorithms a diagnosis can be found in less than a second.

2 PROBLEM DESCRIPTION

We consider a high-level model of a packet-switching computer network, consisting of computers that are connected via a graph of switching components (e.g. routers, switches). We use the term *computers* to denote the start and end points of the data transfer in the network. This includes end-user computers as well as application and web servers. Computers and switching devices are connected by *links*, which correspond to communication channels such as fiber optic wires or satellite channels.

A natural way to describe this model is by a graph $G = (V, E)$. The computers and switching components are the nodes of the graph, and the links are the edges. The nodes of the graph are divided into three sets: V_s, V_i and V_r. V_s are the nodes that send information and V_r are the nodes that receive the information (these two sets of nodes correspond to the computers in the network). V_i is the set of intermediate nodes

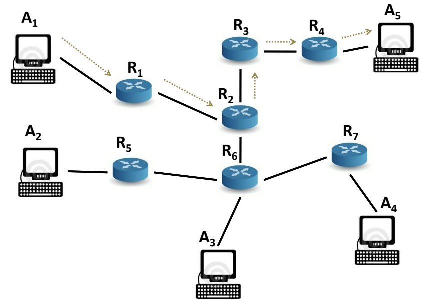


Figure 1: An example of a model of a simple network

through which the information passes (corresponding to the switching devices). We call the transfer of data from a node in V_s to a node in V_r a *flow*.

Definition 1 *[Flow and route] A flow F is described by the tuple $F = \langle s, r, R, b\rangle$ where $s \in V_s$, $r \in V_r$, $R = (v_1, ..v_k) \in V_i^k$ is a vector of switching devices connecting s to r, and $b \in \mathbb{R}$ is the number of bytes that are transfered. R is denoted as the* **route** *of F.*

Figure 1 displays a simple network for example. The circles are the switching devices and the lines are the links. The dashed line represents a possible flow from node $A_1 \in V_s$ to $A_5 \in V_r$ via route $R = \{R_1, R_2, R_3, R_4\} \in V_i^{|R|}$.

Every link in E has various attributes that affect the expected time required to pass b bytes through it. This expected time is called the *delay* of the link. For example, the delay of a link is affected by the physical length of a communication channel, due to the time required to propagate the data through it. Similarly, every switching device has an expected delay of passing b bytes through it. This delay corresponds to the time required to process the data and transfer it to the next link in the route. For example, switching devices have bandwidth, i.e. the number of bytes it can process in a time unit (usually a second). The delay of passing b bytes via a device with bandwidth w will incur a delay of $w \times b$.[1]

Definition 2 *[Expected delay] The expected delay of a node is a function $\delta(v, b) \to \mathbb{R}$ that corresponds to the expected time in seconds that is required for passing b bytes via node v.*

In this paper, we focus on the delays caused by the nodes and not by the links, as they are more probable to be abnormal.[2] Therefore we will omit the delays caused by the links from all the following definitions and calculations, although it should be added as a constant delay for every link in practice. Consequently, we define the expected delay of flow $F = \langle s, r, R, b\rangle$

[1]The delay may even be larger than $w \times b$ due to queues that may form in a switching device.

[2]Link delays are mostly caused by the physical propagation of data in the channel. Thus changes in delay may be caused by changes in the physical infrastructure or strong electromagnetic influence, both of which are seldom.

as the sum of the expected delays of the components in its route: $\delta(F) = \sum_{v \in R} \delta(v_i, b)$. Notice that the expected delay of a node is function of its bandwidth as well as the expected traffic load on that node. As a result, a node may abnormal if its bandwidth has reduced due to some malfunction or if it is experiencing an unexpected increased traffic load.

Definition 3 *[Network model] The network model $M = (G, \delta)$ is composed of the network graph G and an expected delay function δ, enabling the calculation of expected delay for all the possible flows over G.*

During the runtime of the network we are able to observe and measure the actual time it takes data to pass through a flow.

Definition 4 *[Observation] An observation is a pair $\langle F, obs \rangle$, corresponding to an observed flow F and its observed delay $obs \in \mathbb{R}$. This is the* **actual time** *in seconds it took b bytes to path through R.*

Each observed delay is the average delay measured over several flows between the same pair of computers passing the same amount of bytes. We define the difference between an observed delay and an expected delay as the abnormal delay of a flow:

Definition 5 *[Abnormal delay] The abnormal delay of an observation $\langle F, obs \rangle$, is a function $\gamma(\langle F, obs \rangle) = obs - \delta(F)$.*

For brevity, we use $\gamma(o)$ to denote the abnormal delay in observation o and $R(o)$ to denote the route of the observed flow. Given a network model $M = (G, \delta)$ and a set of observations OBS, a diagnosis problem arises when the observed delays are not equal to the expected delays. This means that there exists at least a single observation $o \in OBS$ such that $\gamma(o) > 0$. Since the delay of a flow is the sum of the delays of the nodes on its route, then $\gamma(o) > 0$ implies that there is at least a single node in the route that added more than its expected delay (as calculated by δ).

Definition 6 *[Abnormal node] A node v is called* abnormal *if there exists a flow of b bytes such that the actual delay added by v to this flow is greater than the expected delay $\delta(v, b)$. The difference between the expected delay the actual delay added by v, is called the* abnormal delay *of v, denoted by $\gamma(v)$.*

It is clear that for any observation o, $\gamma(o) = \sum_{v \in R(o)} \gamma(v)$. We call the problem of finding the set of *abnormal nodes*, the *Network Delay Diagnosis Problem* (NDDP). A valid diagnosis is a set of abnormal nodes that will explain the observed delays.

In this paper we add several assumptions. First, we assume that the abnormal delay is non-negative, i.e. $\forall v \;\; \gamma(v) \geq 0$, in order to focus on the case where the end-users are experiencing degraded performance. Second, in order to simplify the problem we assume that every abnormal node adds the same delay. In other words, we limit the abnormal delay function γ to be $\gamma : V_i \rightarrow \{0, C\}$ for some constant C. Later we show how this assumption can be relaxed, producing diagnoses for scenarios where $\gamma : V_i \rightarrow \mathbb{R}$. Without loss of generality we will assume $C = 1$ hereinafter. We also assume a static world, i.e. observing identical flows (same source, destination and number of bytes) will yield the same observed delay, and will pass through the same route. Thus we assume that there will not be more than a single observation per any pair of computers.[3] We leave dynamic routing (e.g. due to load balancing) to future work.

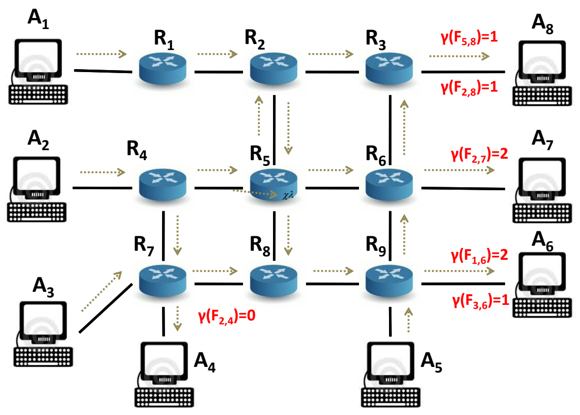


Figure 2: Example of MBD-based network diagnosis

Flow	Route	$\gamma(F_{i,j})$
$F_{1,6}$	R_1, R_2, R_5, R_8, R_9	2
$F_{2,8}$	R_4, R_5, R_2, R_3	1
$F_{2,7}$	R_4, R_5, R_6	0
$F_{2,4}$	R_4, R_7	0
$F_{3,6}$	R_7, R_8, R_9	1
$F_{5,8}$	R_9, R_6, R_3	1

Table 1: Routes of observed flows in Figure 2

3 MBD APPROACH

Using a classical model-based diagnosis approach, NDDP can be described by the tuple $\langle SD, COMP, OBS \rangle$. The system description SD is the network model, i.e. $M = (G, \delta)$ (Definition 3). The components $COMP$ are the switching devices V_i. The observations OBS are the set of observed flows as defined above (Definition 4). A *conflict* is a set of nodes $S \subseteq V_i$ such that there exists an observation $o \in OBS$ that has an observed delay that cannot be explained if all the nodes in S are not abnormal. For every observation o with $\gamma(o) > 0$, every subset of $R(o)$ of size $|R(o)| - \gamma(o) + 1$ is a minimal conflict.

For example consider the graph displayed in Figure 2. $A_1, .., A_8$ are the computers, $R_1, ..R_9$ are the switching devices and $F_{i,j}$ is the flow from computer A_i to computer A_j. The route of each flow is marked by gray arrows, ending with the abnormal delay of the flow (γ). Table 1 lists the six observations displayed in Figure 2, and Table 2 lists the minimal conflicts for every observed flow. For example, the minimal conflicts

[3] As explained above, the observations will be the average delay over several delay measurements.

Flow	Minimal conflicts	$\gamma(F_{i,j})$
$F_{1,6}$	(R_1, R_2, R_5, R_8), (R_1, R_2, R_5, R_9), (R_1, R_2, R_8, R_9), (R_1, R_5, R_8, R_9), (R_2, R_5, R_8, R_9)	2
$F_{2,8}$	(R_4, R_5, R_2, R_3)	1
$F_{2,7}$	$\emptyset$	0
$F_{2,4}$	$\emptyset$	0
$F_{3,6}$	(R_7, R_8, R_9)	1
$F_{5,8}$	(R_9, R_6, R_3)	1

Table 2: Minimal conflict set in Figure 2

of flow $F_{1,6}$ is all $\binom{5}{4}$ possibilities to choose 4 nodes out of the 5 nodes in the route $\{R_1, R_2, R_5, R_8, R_9\}$.

Notice that generating minimal conflicts in NDDP is very straightforward, and does not require any complex diagnosis engine. Next we present an algorithm for NDDP that exploits this and finds all valid diagnoses.

3.1 NDDE

Under the assumption that $\gamma(v) = \{0, 1\}$, it is easy to see that any observation o has exactly $\gamma(o)$ abnormal nodes in $R(o)$. Thus a diagnosis is consistent with the observations if and only if for every observation o it selects from $R(o)$ exactly $\gamma(o)$ nodes as abnormal. We refer to this selection of $\gamma(o)$ nodes from $R(o)$ as *satisfying* the observation. Therefore, a set of nodes is a diagnosis if the assumption that they are abnormal satisfies all the observations. This leads to the following greedy algorithm for finding all the diagnoses, that we call NDDE (Network Delay Diagnosis Engine).

The complete pseudo code of NDDE is presented in Algorithm 1. CND is the set of candidate nodes which may be abnormal. ABS is the set of nodes that are assumed to be abnormal. In the first iteration, CND is initialized with all the nodes in $COMP$ while ABS is an empty set. In every iteration, an observation o is selected and removed from OBS (line 1). Then the procedure $FIND\_ABS$ returns all the possible subsets of $R(o)$ that satisfies o (line 2). If all observations has been satisfied then ABS is a valid diagnosis (lines 6-7). Otherwise try to satisfy the next observation, updating CND' and ABS' as necessary (lines 4-5 and 9).

Note that the procedure $FIND\_ABS$ receives as input ABS, CND and o. This is because the satisfying sets of nodes returned by $FIND\_ABS$ must also be consistent with the observations that have been previously satisfied. This means that nodes in ABS that are also in $R(o)$ are assumed to be abnormal. In addition, only nodes from CND can be selected to satisfy o. This verifies that observations that have been previously satisfied (by assuming that the nodes in ABS are abnormal) remain satisfied. For example, if $|ABS \cap R(o)|$ is already larger than $\gamma(o)$ then with the current choice of abnormal ABS it is not possible to be consistent with the observed delay of o. In such a case $FIND\_ABS$ will return an empty set.

As an example, consider the model displayed in Figure 2. Initially $CND = \{R_1, .., R_9\}$ and $ABS = \{\}$. Let the observation for flow $F_{2,8}$ be the first observation chosen ($\gamma(F_{2,8})$=1). The set of satisfying sets for this observation will be $\{\{R_4\}, \{R_5\}, \{R_2\}, \{R_3\}\}$. Assume that R_4 is chosen first as a satisfying set for this observation ($F_{2,8}$). Then $ABS' = \{R_4\}$ and $CND' = CND \setminus \{R_5, R_2, R_3\}$. CND' is updated because $\gamma(F_{2,8}) = 1$, and thus if R_4 is abnormal, then all other nodes in the same route (R_5, R_2, R_3) must not be abnormal. If the next observation chosen to satisfy is $F_{1,6}$ ($\gamma(F_{1,6} = 2)$), then the set of satisfying sets generated for this observation is only $\{\{R_1, R_8\}, \{R_1, R_9\}, \{R_8, R_9\}\}$. Assume that (R_1, R_8) was chosen, the $ABS' = \{R_4, R_1, R_8\}$ and CND' is updated with R_9. The process continues until all consistent diagnoses are found. It is easy to see that NDDE is sound and complete, i.e. all diagnoses are consistent with the model (this is how they are built), and all consistent diagnoses will be returned.

Algorithm 1: NDDE algorithm

```
   Input: $SD$, the network model
   Input: $CND$, the nodes in the network
   Input: $OBS$, the set of observations
   Input: $ABS$, The set of abnormal nodes
   Output: $\Omega$ the set of diagnoses
 1 Remove observation $o$ from $OBS$
 2 $NewAbs \leftarrow$ FIND_ABS(CND,ABS,o)
 3 for $abs \in NewAbs$ do
 4     $ABS' \leftarrow ABS \cup abs$
 5     $CND' \leftarrow CND \setminus R(o)$
 6     if $OBS$ is empty then
 7         Add $ABS'$ to $\Omega$
 8     else
 9         NDDE($SD$,$CND'$,$OBS$,$ABS'$,$\Omega$)
10     end
11 end
12 return $\Omega$
```

It is clear that the runtime of NDDE is dominated by the runtime required to generate all the possible combinations of satisfying sets for all the observations. Calculating all the satisfying sets of an observation o requires $O(\binom{|R(o)|}{\gamma(o)}) \approx O(|R(o)|^{\gamma(o)})$. If m is the longest route in OBS, and k is the largest abnormal delay measured in an observation, then the total runtime of generating all the satisfying sets combinations for all the observations is:

$$\prod_{o \in OBS} |R(o)|^{\gamma(o)} = (m^k)^{|OBS|} = m^{k \cdot |OBS|}$$

Since we assume that the abnormal delay of a node is either one or zero, k corresponds to the number of abnormal nodes in a route. In non-crisis scenario, the number of faulty switching devices in a route will be very low. Additionally, due to the "‘small world"’ effect observed in Internet routing topologies (Barabasi and Albert, 1999), the length of most Internet routes is very small. Routes with more than 20 nodes are relatively rare. In addition, since most of the networks nowadays are relatively stable the number of abnormal delays on a single path will usually be very small.

Thus is is expected that even in large network models, both m and k will remain relatively small, resulting in the feasibility of this approach for a small number of observations. The experimental results presented further on in this paper support this claim.

3.2 Exploiting normal observations

In the algorithm described above, there is no differentiation between normal and abnormal observations. Each such observation will generate a satisfying set and update CND and ABS. While this result is sound and complete, there is a more effective way to exploit normal observations, as follows. Let OBS_n be the set of all the normal observations, i.e. $\forall o \in OBS_n \;\; \gamma(o) = 0$, and OBS_a be the set of abnormal observations. Let V_n be the set of all the nodes that are on routes of observed normal flows, i.e. $V_n = \cup_{o \in OBS_n} R(o)$, and let V_a be the set of remaining nodes ($V_a = COMP \setminus V_n$). Since we assume that there is no negative abnormal delays, we can safely infer that all the nodes in R_n do not have abnormal delays. Therefore, nodes from V_n should never be used as part of a satisfying set. Thus we propose the following preprocessing step to NDDE: Remove from CND all the nodes in V_n, and remove from OBS all the normal observations OBS_n. This reduces the size of the candidate list, resulting in reduction in runtime. We call this variant of the NDDE algorithm NDDE+.

For example, consider again the model and observations displayed in Figure 2. The observations of flows $F_{2,7}$ and $F_{2,4}$ are normal. Consequently, $V_n = \{R_4, R_5, R_6, R_7\}$ should not be in any satisfying set. As a result, the number of satisfying sets generated for $F_{1,6}$ is reduced from $\binom{5}{2}$ to $\binom{4}{2}$.

For observation $o \in OBS_a$, let $R_a(o)$ be the set of nodes in $R(o)$ that are in V_a, i.e. $R_a(o) = R(o) \cap V_a$. Clearly $|R(o)| \geq |R_a(o)|$. Thus, in the worst case there are no normal observations the computational complexity of NDDE+ remains exactly the same as that of NDDE. However if there are normal observations, the described above preprocessing has the following effects on the computational complexity of NDDE+. First, there are less satisfying sets - instead of $\binom{|R(o)|}{\gamma(o)}$ satisfying sets for every observation o, we have $\binom{|R_a(o)|}{|\gamma(0)}$. Second, there are less observations to satisfy - instead of iterating over all the observations in OBS, only the observations in OBS_a are considered. If m_a be the longest observed route, ignoring nodes from V_n, i.e. $m_a = \max_{o \in OBS_a} R_a(o)$, then the total runtime complexity after the preprocessing step is:

$$\prod_{o \in OBS_a} |R_a(o)|^{\gamma(o)} = (m_a^k)^{|OBS_a|} = m_a^{k \cdot |OBS_a|}$$

We therefore achieve a substantial reduction in runtime in comparison to the analysis in Section 3.1, since $m_a \leq m$ and $|OBS_a| < |OBS|$. The experimental results that we have performed (described in Section 5) also show very large reduction in runtime.

4 LINEAR PROGRAMMING APPROACH

As discussed in the previous section, exploiting the normal observations significantly reduces the diagnosis engine runtime. However, it is still exponential in the number of components. When all possible diagnoses cannot be returned, we would like to return the most probable diagnoses. In the absence of probabilistic knowledge, a common approach is to rank the diagnoses with their cardinality. A cardinality of a diagnosis is the number of (abnormal) component it contains. Unfortunately, it has been proven that finding the minimal cardinality diagnosis is NP-Hard (Garey and Johnson, 1979).

In addition, the previous approach depends on the assumption that the abnormal delays are binary, i.e. the abnormal delay of a node is either zero or one (or some constant C). This allowed relatively simple generation of satisfying sets for every observation, because for every observation o the number of abnormal nodes in the path is exactly $\gamma(o)$. However, in real-life this may not be the case, since nodes can have abnormal delays of various sizes. For example, the observed delay of flow $F_{1,6}$ in Figure 2 is two. This might be caused by two abnormal nodes, e.g. $\gamma(R_1) = 1$ and $\gamma(R_2) = 1$, or a single abnormal node with an abnormal delay of 2, e.g. $\gamma(R_1) = 2$. Furthermore, a network delay is measure in time, and is therefore even not discrete. For example, the observed delay of $F_{1,6}$ can also be explained by $\gamma(R_1) = \frac{2}{3}, \gamma(R_2) = \frac{2}{3}, \gamma(R_5) = \frac{2}{3}$.

When all the abnormal delays are of the same size, a diagnosis is simply the set of nodes with abnormal delays. However, when various sizes of abnormal delays are possible, it is also important to know the amount of abnormal delay added to every abnormal node. For example, the node with the highest abnormal delay should probably be repaired or replaced first. Therefore in such cases we extend a *diagnosis* in NDDP to be a mapping of nodes to their abnormal delays.

Fortunately, it is possible to allow various sizes of abnormal delays and find the minimal cardinality diagnosis by formulating this problem as a linear program and applying a linear programming solver. We associate a variable ab_v for every node $v \in V_a$ (every node that may be abnormal). These variables represent a possible abnormal delay of the nodes in V_a. A constraint is added for every observation $o \in OBS_a$ verifying that the observed abnormal delay is the sum of the abnormal delays of the nodes in $R_a(o)$. Additional constraints are added for every node $v \in R_a$ to verify that abnormal delays are only positive. The target function is to minimize the sum of abnormal delays. This results in the following linear program for an NDDP instance with observation set OBS:

$$\min \sum_{v \in V_a} ab_v \qquad (1)$$

subject to:

$$\begin{array}{ll} ab_v \geq 0 & :\forall v \in V_a \\ \sum_{v \in R_a(o)} ab_v = \gamma(o) & :\forall o \in OBS_a \end{array}$$

As an example, consider yet again the model and observations from Figure 2. Table 3 lists the equation expressing every observation, where variable ab_i corresponds to node R_i. Minimizing the target function $ab_1 + ab_2 + ab_3 + ab_8 + ab_9$ results in the minimal cardinality diagnosis $ab_9 = ab_2 = 1$. Next we prove that this approach is sound:

Flow	Constraint	
$F_{1,6}$	$ab_1 + ab_2 + ab_8 + ab_9$	=2
$F_{2,8}$	$ab_2 + ab_3$	=1
$F_{3,6}$	$ab_8 + ab_9$	=1
$F_{5,8}$	$ab_9 + ab_3$	=1

Table 3: LP constraints for observations in Figure 2

Theorem 1 *[Linear programming soundness] A solution to the linear program is a sound diagnosis, i.e. for every observation $o \in OBS$, $\gamma(o) = \sum_{v \in R_a(o)} ab_v$.*

Proof: By definition, all the nodes in V_a are not included in the route of any normal observation, and $R_a(o) \subseteq V_a$. Thus for every normal observation $o \in O_n$ we have $\sum_{v \in R_a(o)} ab_v = 0 = \gamma(o)$, as required. For every abnormal observation $o \in O_a$ only the nodes in $R_a(o) \subseteq R(o)$ may cause the abnormal delay and be consistent with the normal observations (recall that we assume that there is no negative abnormal delay). Therefore satisfying the constraints in Equation 1 are necessary and sufficient to explain all the observations.□

Since abnormal nodes can have abnormal delays of various sizes, we modify the concept of minimal cardinality diagnosis to the diagnosis which minimizes the sum of abnormal delays (over all abnormal nodes). For example, if diagnosis A consists of a single node with a delay of 100 while diagnosis B consists of two nodes with a delay of 5, the linear program will prefer diagnosis B. We believe that this is reasonable when it is possible to have various abnormal delays in every node. Note that every diagnosis is a solution to the linear program. Since the linear program minimizes the sum of the abnormal delays, the diagnosis produced by the linear program is that which minimizes the sum of the abnormal delays, corresponding to our modified concept of "'minimal cardinality"'.[4]

Unfortunately a linear program returns only a single diagnosis. Thus it is not complete. For example, while $ab_9 = ab_2 = 1$ is a sound diagnosis for the problem described in Figure 2, $ab_3 = ab_1 = ab_8 = 1$ is also a diagnosis. It is possible to extend the linear programming approach to produce more diagnoses by adding a custom constraint after every diagnosis, forbidding the algorithm from returning the previous diagnosis. For example, if the first diagnosis generated was $ab_v = 1$, $ab_w = 2$, then in order to find more diagnoses we will add the constraint $|ab_v - 1| + |ab_w - 2| > 0$. By adding a new constraint that forbids returning to the same diagnosis twice, we can create a complete diagnosis algorithm. The algorithm will continue to diagnose and add constraints until there is no solution that satisfies all the constraints. However, adding such a constraint ($|ab_v - 1| + |ab_w - 2| > 0$) causes the program to be not linear. This requires a more complex solver, which may not run in polynomial time. In addition the number of diagnoses itself can be exponential, in which case the runtime of returning all the diagnoses will also be exponential.

[4]It is possible to define a target function that tries to minimize the sum of abnormal delays and the number of abnormal nodes, by penalizing diagnoses with large cardinality.

5 EXPERIMENTAL RESULTS

In order to evaluate the applicability of the proposed approaches, we have run simulations on computer network models using NS2 (McCanne *et al.*, 2001), which is a standard well-known network simulator. Models were generated randomly using the Barabasi-Albert (Barabasi and Albert, 1999) scale-free random graph model. This model is often used as a model for computer networks, Internet router topology and Internet Autonomous Systems topology.

Table 4 presents the average runtime in milliseconds of running the different diagnosis engines. NDDE+ denotes NDDE with the pruning of normal nodes and observations that was described in Section 3.2. LP denotes the linear programming approach described in Section 4. While NDDE and NDDE+ results show the runtime of finding all diagnoses, LP results are for finding the minimal diagnosis (as described in the previous section). This comparison is reasonable as we have found that finding the minimal cardinality diagnosis with NDDE and NDDE+ required approximately the same runtime as finding all the diagnoses.

The table columns represent the number of observations, and the table rows are the number of nodes in the network model. Each table cell contains the average runtime in milliseconds over 25 instances. We simulated faults in the network by adding a delay of 100ms to 5% of the nodes which were randomly selected. Cells marked with an asteriks (*) denote values of NDDE that were not significantly different than the corresponding values with NDDE+ (using t-test with $\alpha = 0.05$).

As can clearly be seen, NDDE+ is much more efficient than NDDE. This shows that the pruning of nodes that are in normal paths is extremely effective in reducing the diagnosis runtime. For example, with a model of 1000 nodes and 70 observations, NDDE+ is 200 times faster than NDDE. Surprisingly, NDDE+ is also faster than LP. This is because running LP requires the overhead of initializing LP solver and building all the equations. Since NDDE+ finds all the diagnoses in less than a second, the LP overhead is significant.

Next we consider the effect of the different problem parameters. For all the algorithms, as the number of nodes in the network models grows, the total runtime increases. This is reasonable, since larger networks can have longer routes. Longer routes enable more diagnoses, and each diagnosis may contain more nodes (since there may be more abnormals in a single route). Thus generating the diagnoses requires more time.

Increasing the number of observations has a similar effect for NDDE. More available observations entail more observations that need to be satisfied. In addition, more observations increase the number of observed abnormal nodes, resulting in larger diagnoses. Thus the runtime of NDDE grows with the number of observations (see Section 3.1 and the results in Table 4). However, this is not the case for NDDE+, where the runtime increases with the number of observations up to a limit, after which the runtime decreases. For example, consider the line in Table 4 for a network with 700 nodes. The runtime increases up to 5.9 for

	NDDE				NDDE+				LP			
Nodes	5	15	50	70	5	15	50	70	5	15	50	70
100	0.2*	0.5	1.6	1.5	0.2	0.3	0.5	0.7	3.7	4.0	4.3	10.6
300	0.4*	3.2	5.8	25.9	0.4	1.6	1.0	1.3	14.5	16.1	16.5	16.9
500	0.8	1.6	40.0	176.9	0.7	0.7	2.0	2.0	75.1	78.1	79.9	80.2
700	0.5*	4.5	132.7	150.3	0.5	2.6	5.9	3.1	182.7	186.4	189.1	188.8
1000	0.4*	3.9	8517.5	2038.9	0.4	2.8	95.2	10.4	398.3	402.4	402.1	383.8

Table 4: Diagnosis engines runtime in milliseconds. 5% abnormal nodes

	NDDE+			LP		
Nodes	5	15	70	5	15	70
100	0	1	1	3	3	4
300	1	3	11	10	11	14
500	1	25	353	49	50	56
700	1	10	3656	123	124	130
1000	3	142	21203	270	271	281

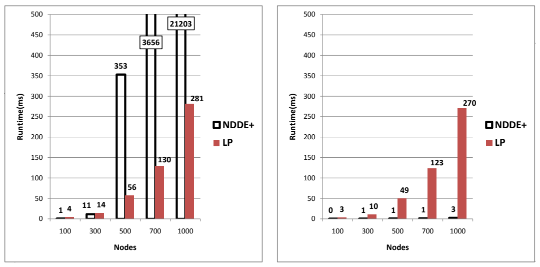
Table 5: NDDE+ and LP with 20% abnormal nodes

5,15 and 50 observations, after which it decreases to 3.1 for 70 observations. This is because more observations increase the number of normal observations. Normal observations increase the pruning of normal nodes, resulting in a reduction in runtime. Note that the runtime of LP remains almost unaffected by the number of observations, since the runtime mostly depends on the number of equations. This number is mainly affected by the size of the network, as in our experiments the size of network was much larger than the number of observations.

Interestingly, the probability that a node will be abnormal have a crucial effect on the runtime of the different algorithms. Table 5 presents the effect of increasing the abnormal probability to 20%. Results for NDDE are omitted as they were much worse than both NDDE+ and LP. Unlike the results in Table 4 (with 5% abnormals), here LP is superior to NDDE+ when there are 70 observations. For example, for a network with 1000 nodes and 70 observations, LP is approximately 75 times faster. Figure 3 emphasizes the runtime difference of LP and NDDE+ for network models with 20% abnormal probability. Each data point is the average runtime of 25 instances running on network models with 1000, 700 and 500 nodes. The y-axis represents the algorithm runtime. Figure 3(a) and (b) show the results for 70 and 5 observations respectively. As can be seen, when there are 20% abnormal nodes, LP is superior to NDDE+ with many observations, while NDDE+ is still better when a small number of observations are given.

The impact of adding abnormal nodes can be explained as follows. Adding abnormal nodes reduces the number of normal observation. Since NDDE+ utilizes normal observations to prune nodes, then increasing the probability of a node to be abnormal increases the runtime of NDDE+. In addition, increasing the number of abnormal nodes enlarges the size of the corresponding diagnoses, as more abnormal nodes are required to explain larger abnormal delays. This is supported by the data presented in Figure 4, that shows the average size of the minimal cardinality diagnosis found by NDDE+. As can be seen, increasing the abnormal probability to 20% significantly increases the size of the minimal cardinality diagnosis.

In addition, more abnormal nodes also allow more possible diagnoses, which also results in longer runtime for NDDE+. Data supporting this claim is presented in Figure 5, which shows the average number of diagnoses found by NDDE+. As can be seen, increasing the abnormal probability to 20% caused NDDE+ to return much more diagnoses. For a



(a) 70 Observations (b) 5 Observations

Figure 3: LP Vs. NDDE+ with 20% abnormals

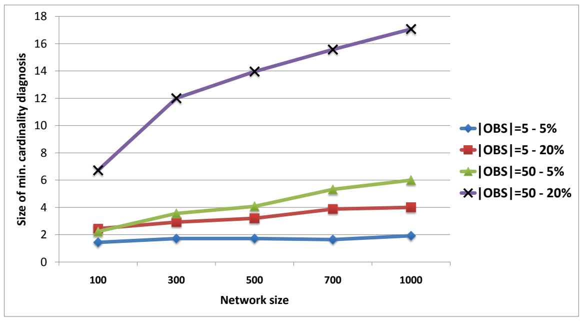


Figure 4: Effects of abnormal prob. on min. cardinality

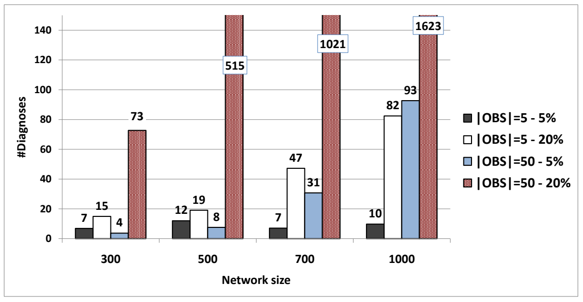


Figure 5: Effects of abnormal prob. on Num. of diagnoses

network with 1,000 nodes, with 5% abnormals the average number of diagnoses was 93 and with 20% abnormals it was 1,623.

6 DISCUSSION AND FUTURE WORK

We have presented two approaches to diagnose and find bottlenecks in computer networks, based on a set of delay observations and prior knowledge of the network model. The first approach builds on classical model-based diagnosis techniques. In order to remain consistent with both normal and abnormal observations, satisfying sets are generated instead of conflict sets. This results in a complete solution, returning all possible minimal diagnoses. Based on simulations of standard network models, we have seen that this approach is very efficient, although in a worst complexity analysis the runtime is exponential. The second approach formulates the problem as a linear program, finding diagnosis with minimum cardinality in polynomial time.

A significant advantage of the first approach is that it is complete - all consistent diagnoses are returned. Simulations show that this can be done very quickly even for network models with 1000 nodes. However, the first approach is applicable when every node is either normal or abnormal, with some constant abnormal delay. This is reasonable if the network nodes are relatively the same, and there is a common type of abnormality that can be abstracted as a binary abnormality. For example, if network routers have a common fault that causes a common delay, this can be modeled as a binary attribute (normal/abnormal). On the other hand, when this is not the case (e.g. when node faults may have a range of values), the LP approach is more appropriate. In addition, if the goal is to find the minimal cardinality diagnosis, then NDDE requires exponential time while the linear programming requires only polynomial time to find the diagnosis.

A major assumption of the linear programming approach is that the best diagnosis is the one that minimizes the sum of abnormal delays. This is reasonable when there is no probabilistic model of the faults of the nodes. However, when a fault model exists, the target function of the linear program will need to be changed. Ideally, if a probabilistic fault model exists then the target function to maximize for finding the most probable diagnosis will be $\prod\limits_{i \in COMP} Pr(ab_i)$. Whether this can be formulated as a linear program target function depends on the prior function ($Pr(ab_n)$). A reasonable assumption is that small abnormal delays are much more common than very large abnormal delays. An example of a fault model that encapsulate this is $Pr(ab_n = x) = e^{-x}$, or any other model where the probability of a delay diminishes exponentially fast with the size of the abnormal delays. For such a fault model , it is even possible to formulate the target function to still be a valid linear program target function:

$$\max \prod_{n \in COMP} Pr(ab_n) = \max \prod_{n \in COMP} e^{-ab_n} = \max \; ln(\prod_{n \in COMP} e^{-ab_n}) = \min \sum_{n \in CMP} ab_n$$

Notice that this is exactly the same as the original target function for the LP approach.

There are many challenges left for future work. Due to complexity issues of the network simulator (NS2), we have performed simulation on relatively network models with up to 1000 nodes. This corresponds to most Local Area Networks (LAN). However, we believe that it is possible to use the described approaches to much larger real-world networks. We therefore intend to evaluate the proposed approaches on real networks, using data measured by Internet mapping projects, e.g. (Spring *et al.*, 2002; Shavitt and Shir, 2005). Using the proposed approaches in real computer networks presents several challenges. Real networks may have dynamic routing, incomplete network models and various types of faults. Dynamic routing for example, introduces a non-deterministic aspect to the diagnosis problem. Using active measurements of the network nodes is another interesting future direction. This is especially feasible when network control protocols such as ICMP (e.g. ping) are available.

REFERENCES

(Bahl *et al.*, 2007) Victor Bahl, Ranveer Ch, Albert Greenberg, Srikanth K, David A. Maltz, and Ming Zhang. Towards highly reliable enterprise network services via inference of multi-level dependencies. In *In SIGCOMM*, pages 13–24, 2007.

(Barabasi and Albert, 1999) A. L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

(Fijany and Vatan, 2005) A. Fijany and F. Vatan. New high performance algorithmic solution for diagnosis problem. In *IEEE Aerospace Conference (IEEEAC05)*, pages 3863 – 3873, 2005.

(Garey and Johnson, 1979) M. R. Garey and D. S. Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)*. W. H. Freeman, January 1979.

(Kandula *et al.*, 2009) Srikanth Kandula, Ratul Mahajan, Patrick Verkaik, Sharad Agarwal, Jitendra Padhye, and Paramvir Bahl. Detailed diagnosis in enterprise networks. *SIGCOMM Comput. Commun. Rev.*, 39(4):243–254, 2009.

(McCanne *et al.*, 2001) Steven McCanne, Sally Floyd, and Kevin Fall. ns2 (network simulator 2), 2001.

(Sandro T. Fontanini and Maragon, 2002) Volnys Bernal Sandro T. Fontanini, Jacques Wainer and Silvio Maragon. Model based diagnosis in lans. In *IEEE Workshop on IP Operations and Management (IPOM)*, 2002.

(Shavitt and Shir, 2005) Yuval Shavitt and Eran Shir. Dimes: let the internet measure itself. *Computer Communication Review*, 35(5):71–74, 2005.

(Spring *et al.*, 2002) Neil Spring, Ratul Mahajan, and David Wetherall. Measuring ISP topologies with rocketfuel. *SIGCOMM Comput. Commun. Rev.*, 32(4):133–145, 2002.

(Wei *et al.*, 2003) Wei Wei, Bing Wang, Don Towsley, and Jim Kurose. Model-based identification of dominant congested links. In *Proceedings of ACM Internet Measurement Conference*, 2003.