The Boolean Solution to the Congested IP Link Location Problem: Theory and Practice

Hung X. Nguyen Patrick Thiran
School of Computer and Communication Sciences, EPFL
CH-1015 Lausanne, Switzerland
{hung.nguyen, patrick.thiran}@epfl.ch

***Abstract*— Like other problems in network tomography or traffic matrix estimation, the location of congested IP links from end-to-end measurements requires solving a system of equations that relate the measurement outcomes with the variables representing the status of the IP links. In most networks, this system of equations does not have a unique solution. To overcome this critical problem, current methods use the unrealistic assumption that all IP links have the same prior probability of being congested. We find that this assumption is not needed, because these probabilities can be uniquely identified from a small set of measurements by using properties of Boolean algebra. We can then use the learnt probabilities as priors to find rapidly the congested links at any time, with an order of magnitude gain in accuracy over existing algorithms. We validate our results both by simulation and real implementation in the PlanetLab network.**

I. Introduction

Many IP network inference problems are ill-posed: the number of measurements are not sufficient to determine uniquely their solution. For example, the traffic matrix estimation problem is finding the Origin-Destination (OD) pairs of traffic flows from the link counts. As the number of OD pairs far exceeds the number of links, the resulting system of equations is under-determined. Various heuristics, such as the gravity model, can then be used to reduce the set of possible solutions.

Another tomography problem, which we address in this paper, is the identification of poorly performing (congested) IP links from end-to-end path measurements. Diagnosing wide area IP networks is also difficult because the end-to-end measurements do not provide enough information to determine the link characteristics. To find the exact link loss rates, sophisticated infrastructures such as those in [1], [2] are needed. Even in these cases, the inverse problem of finding the link loss rates, given the end-to-end measurements, is not trivial.

In many settings of today's IP networks, there is one link along the path that is responsible for the majority of losses in the path. This link is called the *dominant* congested link [3], [4]. For most practical applications it is already sufficient to know the locations of these links. For example, VoIP calls that try to achieve the Public Switch Telephone Network (PSTN) quality only consider rerouting through alternative paths when the call rating factor R is below 70 [5]. If these calls use the G.729 codec, this requirement can be mapped to an end-to-end loss rate of less than 1%. It is therefore more important for these applications to identify the links that drop more than 1% of packets than to attempt to compute the exact link loss rates. The Boolean framework described in Section III, where variables representing links and paths quality take only two values, is therefore well suited to this setting.

In this work, we are interested in locating the dominant congested IP links, which we define as the links that drop more than a certain percentage of packets. Similarly to the linear relation between link and path loss rates, congested links and congested paths are linearly related in Boolean algebra [6]: a path is congested if and only if one of its constituent links is congested. Identifying the congested links therefore requires solving this system of Boolean linear equations. Although previous works, reviewed in Section II, have succeeded in doing so in a tree topology [4], [7], they need to rely on the assumption that all links have the same prior probability of being congested, because they infer network properties from only one full measurement (which we call *snapshot*) of the network, and hence face the fundamental problem that there is no unique solution for the inverse problem in Boolean algebra.

We take a different approach in this paper. We use multiple measurements (snapshots) over a period of time to learn about the congestion probabilities of the links, which we then use in a second stage to locate congested links in subsequent snapshots.

We first prove in Section IV that these link state probabilities are statistically identifiable from binary end-to-end measurements by using simple properties of Boolean vectors. This result shows that these probabilities can be correctly learnt from a sufficiently large set of end-to-end measurements if we have the proper learning method, but it does not tell us how to proceed. The solution is given in Section V. It combines information collected from measurements using very simple but powerful properties of Boolean algebra, which provide us with a sufficient number of additional constraints to make the congestion probabilities uniquely identifiable. These equations are not only independent from those obtained without combining measurements, but they are also linear. This nice feature makes it easy to find the solution using only a small number of snapshots.

At the end of this learning phase, we can use the estimated link congestion probabilities as *prior* information, together with the most recent measurements, to find the links that are actually congested using a *Maximum A-Posteriori (MAP)* estimator. We formulate this inference problem as a simple convex optimization problem and use a primal heuristic solution to

solve it in Section VI.

Finally, the simulations and Internet experiments of Section VII show that the algorithm is fast and significantly more accurate than existing algorithms.

II. Related work

The inference of internal link properties from end-to-end measurements is called network tomography. It requires solving a system of equations relating the end-to-end measurements with the link properties either in linear or in Boolean algebra (the *inversion* problem). Most end-to-end tomography methods fall into one of two classes: methods that require strong temporal correlation between probing packets in a multicast-like environment [1], [2], [8] and perform the inversion in linear algebra, and methods that use the distribution of congested links in the Internet [4], [7], [9] to solve the inversion problem in Boolean algebra.

The initial methods in the first class [1], [8] infer the loss rate of network links using multicast probing packets. As multicast is not widely deployed in the Internet, subsequent methods [2] emulate this approach using clusters of unicast packets. These methods are less accurate than their multicast counterparts and also require substantial development and administrative costs. Furthermore, the iterative algorithms used to compute link loss rates in these approaches are expensive for real-time applications in large networks.

Methods in the second class [4], [7], [9] use only uncorrelated end-to-end measurements for a simpler goal of identifying the congested links. These methods do so by finding the smallest set of links whose congestions can explain the observed measurements. They use two assumptions:

- All links have the same prior probability of being congested. Let us denote this probability by p_0.
- p_0 is small: In [4], [7] p_0 is less than 0.2.

The first assumption is unrealistic in real networks. The Internet is a heterogenous environment where links have different probabilities of being congested. For example, this probability is smaller for backbone links than for access links. Making the assumption that they are all the same can lead to an inaccurate diagnosis. It was reported in [7] that when p_0 is large, their algorithms perform poorly. For example, when $p_0 = 0.2$ the detection rate (percentage of congested links that are correctly identified) of their algorithms is only 30%.

The method in [10] also considers uncorrelated end-to-end monitoring traffic but performs the inversion in linear algebra. Because the inversion problem has multiple possible solutions, [10] tries to find the smallest sets of consecutive links whose combined loss rates can be determined. This method therefore cannot be used to identify the congested links at the granularity of each individual link.

There are also non-tomography techniques for calculating link loss rates such as [11]. Instead of using end-to-end measurements, these approaches use Internet Control Message Protocol (ICMP) echo requests to infer the loss rates. Unfortunately for security and performance issues, many routers do not respond or limit the responses to ICMP requests. Consequently, the Tulip tool of [11] needs more than ten minutes to diagnose a path.

III. The Network and Performance Model

We consider an overlay monitoring system that consists of end hosts that can send and receive UDP or TCP packets and perform traceroute [12]. We call these end hosts the *vantage points*. Each vantage point sends probing packets (either UDP or TCP) to a set of destinations. Loss rates are calculated at the receivers, based on packet sequence numbers. All measurement results are reported to a central server.

A. Network Topology

We model the network as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the set $\mathcal{V}$ of nodes denotes the network routers/hosts and the set $\mathcal{E}$ of edges represents the communication links connecting them. The number of nodes and edges is denoted by $n_v = |\mathcal{V}|$, and $n_e = |\mathcal{E}|$, respectively. The set of all vantage points is denoted by $\mathcal{V}_B$. Furthermore, we use $P_{s,t}$ to denote the path traversed by an IP packet from a source node s to a destination node t. Let $\mathcal{P}$ be the set of all paths between the vantage points and let $n_p = n_B(n_B - 1) = |\mathcal{P}|$.

For a known topology $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a set of paths $\mathcal{P}$, we compute the routing matrix D of dimension $n_p \times n_e$ as follows. The entry $D_{ij} = 1$ if the path $P_{s,t} \equiv P_i$, with $i = (s, t)$, contains the link e_j and $D_{ij} = 0$ otherwise. A row of D therefore corresponds to a path, whereas a column corresponds to a link. Clearly, if a column contains only zero entries, the quality of the corresponding link cannot be inferred from measurements of the paths in $\mathcal{P}$. Hence, we drop these columns from the routing matrix to obtain a matrix of dimensions $n_p \times n_c$, where $n_c \leq n_e$ is the number of links that are covered by at least one path in $\mathcal{P}$. We denote this set of covered links by $\mathcal{E}_c$, $|\mathcal{E}_c| = n_c$.

B. Performance Model

Given the network topology and end-to-end path loss rates, we can easily establish linear relationships between the link loss rates and the path loss rates as follows [13], for $1 \leq i \leq n_p$

$$\log(\phi_i) = \sum_{k=1}^{n_c} \log(\phi_{e_k}) D_{ik} \tag{1}$$

where ϕ_i is the transmission rate (i.e., one minus the loss rate) of path P_i and ϕ_{e_k} is the transmission rate of link e_k. To determine the link transmission rates, we need to solve the system of linear equations (1). Unfortunately, (1) does not have a unique solution because in most networks the matrix D is rank deficient, that is, $\text{rank}(D) < \min(n_p, n_c)$.

Our objective, however, is not to compute the link loss rates but rather to identify the links whose loss rates exceed the maximal threshold tolerated by the application. We call these links *congested links* and the other links *good links*. Let t_l be the threshold specified by the application. Link e_k is good if $\phi_{e_k} \geq t_l$, and it is congested if $\phi_{e_k} < t_l$. Note here that the threshold t_l can be changed by the applications depending on their performance requirements.

We define a path to be *congested* if it contains at least one congested link and we define a path to be *good* if it consists of only good links. We use the variable y_i to represent the state

of the path P_i: $y_i = 0$ if P_i is good and $y_i = 1$ otherwise; and the variable x_k to represent the state of link e_k: $x_k = 0$ if e_k is good and $x_k = 1$ otherwise. We can then establish a system of linear equations in Boolean algebra relating y_i and x_k as:

$$y_i = \bigvee_{k=1}^{n_c} x_k \cdot D_{ik} \tag{2}$$

where "$\vee$" denotes the binary max operation, and "$\cdot$" denotes the usual multiplication operation. We can recast (2) in vector form as:

$$\boldsymbol{y} = \bigvee_{k=1}^{n_c} x_k \boldsymbol{d}_k \tag{3}$$

where $\boldsymbol{d}_k$ is the kth column of D. In this paper, we use a bold letter to represent a vector.

Each input y_i to the equations in (2) is obtained from end-to-end measurements by comparing the path transmission rate ϕ_i against a threshold t_p. In [7], t_p is chosen to be t_l^d, where d is the length of the path. If $\phi_i \geq t_p$, we assume that the path is good, that is, all of its constituent links are good. Conversely, if $\phi_i < t_p$, we assume that the path is congested, that is, at least one of its constituent links is congested. Obviously, using the path threshold t_p to determine good and congested paths may lead to errors because in many cases the path transmission rate can be smaller than t_p even though there is no congested link in the path [7]. However, for many performance metrics such as connectivity (links are either up or down), dominant congested links, and delay spikes (sudden increase in the delay) the probability of making errors by using the path threshold is negligible [7].

The transformation from linear algebra to Boolean algebra is illustrated in the example of Figure 1. In the figure, Link $e_1 = SA$ is a good link with a transmission rate of 1. The two links $e_2 = AB$ and $e_3 = AC$ are congested with transmission rates of 0.8. The two end-to-end paths (P_1 from S to B and P_2 from S to C) are also congested. Similarly to the linear relationship between link and path transmission rates, link and path states are also linearly related in Boolean algebra. Moving from linear algebra to Boolean algebra offers several benefits. First, it is easier to obtain the quality of a path P_i (i.e., estimating the Boolean variable y_i) than to obtain the transmission rate of the path (i.e., estimating the continuous variable ϕ_i). Second, as we will see later, even though the equations (2) still have multiple solutions, there is a simple and accurate method to find the most probable solution for (2).

Our main goal in this paper is to solve (2). These equations have a unique solution is and only if all columns of D are linearly independent in Boolean algebra [6], which is rarely the case in practice. We need therefore additional information about the network links to find the most likely solution of (2). This information will be the probability p_k that a link e_k is congested. We denote by $\boldsymbol{p} = [p_1\ p_2 \ldots\ p_{n_c}]^T$ the vector of the link state probabilities, where T denotes transposition. Instead of making assumptions about these probabilities as in [7] and [4], we first show theoretically that it is possible to learn $\boldsymbol{p}$ from end-to-end measurements. Indeed, although the conditions on the routing matrix D needed to identify $\boldsymbol{x}$ are in general not satisfied, we will see in the next section that the conditions on D needed to identify $\boldsymbol{p}$ are much weaker. They are actually verified in practice, and we develop an algorithm that can estimate $\boldsymbol{p}$ from a small number of snapshots. Finally, using the resulting vector $\boldsymbol{p}$ as prior link state probabilities, we are equipped with sufficient information to propose an algorithm that can find the actual congested links accurately.

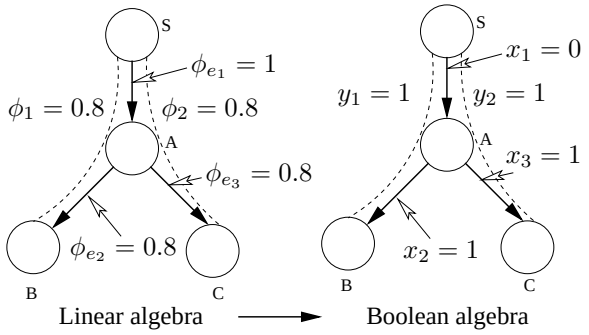

Fig. 1. The nodes are the network routers/hosts and the directed edges are the links connecting them. If we define that links with transmission rate below 0.9 and paths with transmission rate below 0.81 ($= 0.9^2$) as congested, then we can obtain an equivalent system of linear equations relating link qualities with path qualities in Boolean algebra.

IV. Identifiability of the link state probabilities

In this section, we show that the prior probabilities can be uniquely identified from end-to-end measurements in most networks. We call this property of the prior the *identifiability* property. Identifiability is critical, because it guarantees that a correct learning algorithm can give us the true link probability vector $\boldsymbol{p}$.

Let us first formulate mathematically this property. Let us denote by $\mathbb{P}_{\boldsymbol{p}}$ the probability measure on the set of network links when the link probability vector is $\boldsymbol{p}$. Let $\boldsymbol{X}$ denote the random binary vector of dimension n_c representing the states of the links. We have thus $\mathbb{P}_{\boldsymbol{p}}(X_k = 1) = p_k$, $1 \leq k \leq n_c$. Similarly, let $\boldsymbol{Y}$ denote the random vector of dimension n_p representing the resulting states of the paths. A statistical model is said to be identifiable, if the cumulative distribution function of the observable data is an injective function of the parameter(s). In other words, the link state probabilities are statistically identifiable if the property

$$\mathbb{P}_{\boldsymbol{p}}(\boldsymbol{Y} = \boldsymbol{y}) = \mathbb{P}_{\tilde{\boldsymbol{p}}}(\boldsymbol{Y} = \boldsymbol{y}) \text{ for any snapshot } \boldsymbol{y} \tag{4}$$

always implies that $\boldsymbol{p} = \tilde{\boldsymbol{p}}$.

We make the following assumptions about the network:

- the prior probabilities p_k and the routing matrix D remain unchanged during the measurement period when the m snapshots $\mathcal{Y} = \{\boldsymbol{y}^1, \boldsymbol{y}^2, ..., \boldsymbol{y}^m\}$ are obtained;
- $0 \leq p_k < 1$ for all $1 \leq k \leq n_c$;
- the link states X_k are independent from each other (but not identically distributed).

The first assumption can be violated in the Internet where routing changes can happen on any timescale. As a consequence, we will have some noise in our estimation of the

prior information. We show in our experiments in Section VII-B that the changes in network topologies do not significantly affect the accuracy of our algorithms. The stability of p_k has been observed in [14] where IP links can remain congested for hours.

When $p_k = 1$ for some k (i.e., link e_k is disconnected), we cannot uniquely identify the p_k from end-to-end measurements. Take the simple network of Figure 1. In this network, if link SA is disconnected ($p_{SA} = 1$), then the two end-to-end paths (from S to B and from S to C) are both disconnected. In this case, all measurement vectors have the same value $\boldsymbol{y}^1 = \boldsymbol{y}^2 = ... = \boldsymbol{y}^m = \mathbf{1}$. Furthermore, if both AB and AC are disconnected, we will also observe the same the end-to-end results. Therefore, even with multiple snapshots, we cannot ascertain if $p_{SA} = 1$ or $p_{AB} = p_{AC} = 1$. We show in [6] that in the "deterministic link failure scenario" where $p_k \in \{0, 1\}$, $\boldsymbol{p}$ is identifiable if and only if all columns of D are linearly independent in Boolean algebra. We will see below that the condition on D is weaker when $0 \leq p_k < 1$. Fortunately, IP routing algorithms can by-pass disconnected links very quickly. It is therefore reasonable to discard the value $p_k = 1$ for all k.

The third assumption can also be violated in the Internet. But previous works [7] and [4] have shown that the correlation of link loss rates is small and does not significantly effect the accuracy of the diagnosis.

Under the above assumptions, we can state and prove the following theorem.

Theorem 1: The link state probability vector $\boldsymbol{p}$ is identifiable if and only if the columns of the routing matrix D are all distinct.

The theorem says that if we have enough snapshots to estimate the probabilities of all possible measurement outcomes, then there is only one vector of link probabilities that can generate this sequence of snapshots. The proof of the theorem is given in the appendix, together with a short lemma.

We can make two important observations about Theorem 1. First, it holds for any topology, and therefore applies to all networks. Second, the idea of using multiple snapshots to learn about the link properties can be extended to many other link characteristics. Indeed, Vardi et al. [15] proved similar identifiability results for the rates of traffic counts with Poisson distributions. We keep this extension for future work.

V. An Efficient Algorithm to Estimate the Prior

In the previous section, we have shown that the prior probabilities can be learnt from a sufficiently large number of snapshots. However, the theorem does not tell us anything about the way to compute these probabilities.

In practice, even though it is easy to obtain many snapshots if we wait long enough, it is unlikely that the probabilities $\boldsymbol{p}$ or that the routing matrix D remain the same in all of these snapshots. It is therefore important to have an algorithm that can quickly estimate the prior from only a small number of snapshots. Here we provide an algorithm that does so.

Let us first begin with a precise description of the prior estimation problem. We use the same network topology and performance definitions as in Section III. We are given the following information:

- the routing matrix D as defined in Section III-A;
- m measured snapshots $\mathcal{Y} = \{\boldsymbol{y}^1, \boldsymbol{y}^2, ..., \boldsymbol{y}^m\}$.

Our objective is to find an algorithm that can quickly calculate prior probabilities $\boldsymbol{p}$ from the D and $\mathcal{Y}$.

A first possible approach is to use an iterative EM (Expectation-Maximization) algorithm such as the one in [8]. However, as demonstrated in [8], this method requires significant computational time and is only guaranteed to converge to a local optimum point.

A second, more direct approach is to take the expectations in (2), which gives for all $1 \leq i \leq n_p$

$$\begin{aligned}\mathbb{E}_{\boldsymbol{p}}[Y_i] &= \mathbb{E}_{\boldsymbol{p}}\left[\vee_{k=1}^{n_c} X_k D_{ik}\right] = \mathbb{P}_{\boldsymbol{p}}\left(\vee_{k=1}^{n_c} X_k D_{ik} = 1\right) \\ &= 1 - \mathbb{P}_{\boldsymbol{p}}\left(X_k D_{ik} = 0, 1 \leq k \leq n_c\right) \\ &= 1 - \prod_{k=1}^{n_c} (1 - p_k)^{D_{ik}}.\end{aligned}$$

Now, the expectation $\mathbb{E}_{\boldsymbol{p}}[Y_i]$ is computed by averaging all measured values of Y_i in $\mathcal{Y}$, which we denote by $\overline{y}_i$. Therefore, taking logarithms, we get a set of n_p linear equations

$$-\log(1 - \overline{y}_i) = \sum_{k=1}^{n_c} (-\log(1 - p_k)) D_{ik}, \tag{5}$$

with $1 \leq i \leq n_p$.

Unfortunately, equations (5) define in general a rank deficient system of linear equations because $\text{rank}(D) < \min(n_p, n_c)$, as mentioned earlier. We need to have more relations linking the measurements to the unknown $\boldsymbol{p}$. A first idea is to use spatial correlations via higher order moments. But this would result in strongly nonlinear equations, that are not easy to invert. For the example in Figure 1, the second moment between Y_1 and Y_2 is

$$\begin{aligned}\mathbb{E}_{\boldsymbol{p}}[Y_1 Y_2] &= \mathbb{P}_{\boldsymbol{p}}(Y_1 = 1, Y_2 = 1) \\ &= p_1(1 - p_2)(1 - p_3) + p_1 p_2 (1 - p_3) \\ &\quad + p_1(1 - p_2) p_3 + (1 - p_1) p_2 p_3.\end{aligned}$$

Boolean algebra will be of invaluable help to overcome this difficulty because it allows us to complete (5) with *independent linear* equations. Indeed, we cannot create new independent equations by linearly combining equations (5) in the conventional $(+, \times)$ algebra, but we can obtain new independent equations by linearly combining them in the Boolean $(\max, \times)$ algebra. The combination amounts to computing the maximum of the state of different paths, which is a linear operation in the Boolean $(\max, \times)$ but a nonlinear operation in the $(+, \times)$ algebra.

Let us consider the combination of two paths. Let Y_{il} be the binary random variable representing the event that both paths i and l are good: $Y_{il} = 0$ if both paths i and l are good and $Y_{il} = 1$ otherwise. Then

$$\begin{aligned}\mathbb{E}_{\boldsymbol{p}}[Y_{il}] &= \mathbb{P}_{\boldsymbol{p}}\left(\{\vee_{k=1}^{n_c} X_k D_{ik} = 1\} \cup \{\vee_{k=1}^{n_c} X_k D_{lk} = 1\}\right) \\ &= 1 - \prod_{k=1}^{n_c} (1 - p_k)^{D_{ik} \vee D_{il}},\end{aligned}$$

which becomes, with $\overline{y}_{il}$ being the average of Y_{il} computed over $\mathcal{Y}$, and taking logarithms,

$$-\log(1-\overline{y}_{il}) = \sum_{k=1}^{n_c}(-\log(1-p_k))\{D_{ik} \vee D_{lk}\}, \quad (6)$$

for all $1 \leq i < l \leq n_p$ pairs (we can take $i < l$ because $Y_{il} = Y_{li}$). There are $n_p(n_p-1)/2$ equations of this form.

Let

$$\begin{aligned} \boldsymbol{y}^{[1]} &= [-\log(1-\overline{y}_1) \ \ldots \ -\log(1-\overline{y}_{n_p})]^T \\ \boldsymbol{y}^{[2]} &= [-\log(1-\overline{y}_{12}) \ \ldots \ -\log(1-\overline{y}_{(n_p-1)n_p/2})]^T \end{aligned}$$

$$-\log(\mathbf{1}-\boldsymbol{p}) = [-\log(1-p_1) \ \ldots \ -\log(1-p_{n_c})]^T.$$

Combining the equations in (5) and (6) we have a system of $n_p(n_p+1)/2$ linear equations, which reads

$$\left[\begin{array}{c} \boldsymbol{y}^{[1]} \\ \boldsymbol{y}^{[2]} \end{array}\right] = \left[\begin{array}{c} D \\ D^{[2]} \end{array}\right] (-\log(\mathbf{1}-\boldsymbol{p}))^T, \quad (7)$$

where $D^{[2]}$ is the $n_p(n_p-1)/2 \times n_c$ matrix whose rows are indexed by $(i,l), 1 \leq i < l \leq n_p$, with the (i,l) row of $D^{[2]}$ being the element-wise max operation between the two rows i and l of D. For the network in Figure 1, we have

$$D = \left[\begin{array}{ccc} 1 & 1 & 0 \\ 1 & 0 & 1 \end{array}\right],$$

and thus

$$D^{[2]} = \left[\begin{array}{ccc} 1 & 1 & 1 \end{array}\right],$$

and $\boldsymbol{y}^{[2]}$ is a vector with a single entry: $\boldsymbol{y}^{[2]} = -\log(1-\overline{y}_{12})$.

The number of unknowns p_k in (7) is n_c. With the combinations of two paths, we can obtain up to $n_p(n_p+1)/2$ linear constraints. It was shown in [16] that if the underlying IP network has a power-law degree topology, then the number of links n_c between n_p vantage points scales as $O(n_p \log n_p)$, whence $n_p(n_p+1)/2 \gg n_c$. Consequently, although some of the equations (6) are linearly dependent, we found in all our simulations and experiments of Section VII that with real IP topologies, the combinations of two paths provided enough additional linearly independent equations (6) to make the augmented system (7) of full rank. If this had not been the case, we would have had to keep getting additional constraints by considering combinations of three or more paths.

After obtaining a full rank system of linear equations, we can then inverse $\left[\begin{array}{c} D \\ D^{[2]} \end{array}\right]$ to find vector $\boldsymbol{p}$. If the resulting system is not of full rank, we then need to use the LININPOS algorithm of [17] to perform the inversion. We refer to [17] for details of the LININPOS solution.

VI. Using the Prior to Identify Congested Links

The prior probabilities themselves provide useful information about the network and can be used in many applications. One such application is to combine the prior probabilities with the most recent measurement to in order to locate the links that are currently congested (and not just simply their probabilities of congestion). We focus on this problem in this section.

A. Problem Definition

Let us first begin with the detailed description of the congested link identification (CLINK) problem. We use the same network topology and performance definitions as in Section III. For the CLINK problem, we are given the following information:

- the routing matrix D as defined in Section III-A;
- a measured snapshot $\boldsymbol{y} = [y_1 \ y_2 \ldots \ y_{n_p}]^T$;
- the prior link state probabilities $\boldsymbol{p}$;

We then need to solve (2) to find $\boldsymbol{x} = [x_1 \ x_2 \ldots \ x_{n_c}]^T$. As (2) has multiple solutions, we look for the most likely one. In other words, we want to find the vector $\boldsymbol{x}$ that maximizes the conditional probability that all links e_k such that $x_k = 1$ are indeed congested given the measurements $\boldsymbol{y}$, i.e.,

$$\operatorname*{argmax}_{\boldsymbol{x}} \mathbb{P}_{\boldsymbol{p}}(\boldsymbol{X} = \boldsymbol{x} \mid \boldsymbol{Y} = \boldsymbol{y}). \quad (8)$$

Recall here that $\mathbb{P}_{\boldsymbol{p}}(\cdot)$ is the conditional probability, given the prior vector $\boldsymbol{p}$. The solution of (8) therefore gives a Maximum A-Posteriori (MAP) estimate of the set of congested links.

From Bayes' rule,

$$\mathbb{P}_{\boldsymbol{p}}(\boldsymbol{X} = \boldsymbol{x} \,|\, \boldsymbol{Y} = \boldsymbol{y}) = \frac{\mathbb{P}_{\boldsymbol{p}}(\boldsymbol{Y} = \boldsymbol{y} \,|\, \boldsymbol{X} = \boldsymbol{x})\mathbb{P}_{\boldsymbol{p}}(\boldsymbol{X} = \boldsymbol{x})}{\mathbb{P}_{\boldsymbol{p}}(\boldsymbol{Y} = \boldsymbol{y})}$$

and as $\mathbb{P}_{\boldsymbol{p}}(\boldsymbol{Y} = \boldsymbol{y})$ only depends on the network conditions and measurements, and not on the choice of $\boldsymbol{x}$, we are left with the equivalent maximization problem

$$\operatorname*{argmax}_{\boldsymbol{x}} \mathbb{P}_{\boldsymbol{p}}(\boldsymbol{Y} = \boldsymbol{y} \mid \boldsymbol{X} = \boldsymbol{x})\mathbb{P}_{\boldsymbol{p}}(\boldsymbol{X} = \boldsymbol{x}). \quad (9)$$

As the link states X_k are independent random variables,

$$\mathbb{P}_{\boldsymbol{p}}(\boldsymbol{X} = \boldsymbol{x}) = \prod_{k=1}^{n_c} p_k^{x_k}(1-p_k)^{(1-x_k)}.$$

For any path P_i, we compute from equations (2) that given $\boldsymbol{X} = \boldsymbol{x}$, $Y_i = 0$ if and only if, for all $1 \leq k \leq n_c$, $D_{ik}x_k = 0$ or equivalently $(1-D_{ik})^{x_k} = 1$. Consequently, the conditional probabilities

$$\mathbb{P}_{\boldsymbol{p}}(Y_i = 0 \mid \boldsymbol{X} = \boldsymbol{x}) = \prod_{k=1}^{n_c}(1-D_{ik})^{x_k}$$

and

$$\mathbb{P}_{\boldsymbol{p}}(Y_i = 1 \mid \boldsymbol{X} = \boldsymbol{x}) = 1 - \mathbb{P}_{\boldsymbol{p}}(Y_i = 0 | \ \boldsymbol{X} = \boldsymbol{x})$$

are either 1 or 0.

Let us define $\mathcal{P}_G$ as the set of paths measured as being good (i.e., $P_i \in \mathcal{P}_G$ when $y_i = 0$) and $\mathcal{P}_C$ the set of paths found congested (i.e., $P_i \in \mathcal{P}_C$ when $y_i = 1$). We can write

$$\begin{aligned} &\mathbb{P}_{\boldsymbol{p}}(\boldsymbol{Y} = \boldsymbol{y} \mid \boldsymbol{X} = \boldsymbol{x}) \\ &= \prod_{P_i \in \mathcal{P}_G} \mathbb{P}_{\boldsymbol{p}}(Y_i = 0 \,|\, \boldsymbol{X} = \boldsymbol{x}) \prod_{P_i \in \mathcal{P}_C} \mathbb{P}_{\boldsymbol{p}}(Y_i = 1 \,|\, \boldsymbol{X} = \boldsymbol{x}). \end{aligned}$$

We see that this probability is again 1 or 0.

Consequently, for the argument of (9) to be non-zero, its solution $\boldsymbol{x}$ must be such that (i) for each congested path $P_i \in \mathcal{P}_C$, it contains at least one link e_k (i.e., $x_k = 1$) that

satisfies $D_{ik} = 1$, and (ii) for each good path $P_i \in \mathcal{P}_G$, it does not contain any link e_k (i.e., $x_k = 0$) that satisfies $D_{ik} = 1$. In summary, all links in good paths must be good, and a congested path must go through at least one congested link.

Using the above observations, we can significantly simplify the CLINK problem as follows. Let R be the matrix obtained from D by removing all rows that correspond to good paths and all columns that correspond to links in the good paths. Each row of R now represents a congested path and each column of R represents a link that belongs to at least one congested path. Let us denote by $\mathcal{E}_R$ the set of links represented by columns of R. With known good links and paths excluded, R has dimension $|\mathcal{P}_C| \times |\mathcal{E}_R|$. The CLINK problem then amounts to finding a set of links $\mathcal{X} \subseteq \mathcal{E}_R$ such that all congested paths are covered by at least one link in $\mathcal{X}$, and such that

$$\underset{\mathcal{X} \subseteq \mathcal{E}_R}{\operatorname{argmax}} \mathbb{P}_{\boldsymbol{p}}(\mathcal{X}) = \underset{\mathcal{X} \subseteq \mathcal{E}_R}{\operatorname{argmax}} \prod_{k=1}^{|\mathcal{E}_R|} p_k^{x_k} (1 - p_k)^{(1-x_k)}. \qquad (10)$$

Taking the logarithm of (10) and eliminating the terms that do not depend on $\boldsymbol{x}$, we obtain the optimization problem

$$\begin{aligned} \underset{\mathcal{X} \subseteq \mathcal{E}_R}{\operatorname{argmax}} \mathbb{P}_{\boldsymbol{p}}(\mathcal{X}) &= \underset{\mathcal{X} \subseteq \mathcal{E}_R}{\operatorname{argmax}} \sum_{k=1}^{|\mathcal{E}_R|} x_k \log \frac{p_k}{1 - p_k} \\ &= \underset{\mathcal{X} \subseteq \mathcal{E}_R}{\operatorname{argmin}} \sum_{k=1}^{|\mathcal{E}_R|} x_k \log \frac{1 - p_k}{p_k} \qquad (11) \end{aligned}$$

subject to $\sum_{k=1}^{|\mathcal{E}_R|} R_{ik} x_k \geq 1$ for all $1 \geq i \geq |\mathcal{P}_C|$.

B. The Inference Algorithm

The optimization problem in (11) is indeed the weighted set cover problem (WSCP), a known NP-complete problem [18]. A number of optimal algorithms, typically based on tree-search procedures, are proposed in the literature to solve the WSCP. Most of these algorithms require extensive computational time (in the order of hours for a network with one thousand nodes) [19]. Given the current situation with respect to the optimal algorithms, we choose a computationally effective heuristic algorithm capable of producing good quality solutions. We define the *domain* of a link $e_k \in \mathcal{E}_c$, $Domain(e_k)$, as the set of paths that contain the link e_k. The algorithm uses a greedy heuristic that constructs a feasible solution set by a sequence of steps, each of which consists in selecting a link e_k (i.e., setting the variable x_k to 1) that minimizes $\log((1 - p_k)/p_k)/\,|Domain(e_k)|)$.

This algorithm is fast and yet is the best polynomial-time approximation algorithm for the WSCP (and hence the CLINK problem) in terms of worst-case performance [18]. The algorithm uses an auxiliary set variable $\mathcal{Q}_B$.

The CLINK algorithm is a $\log(n_c + 1)$-approximation algorithm with a computational complexity of $O(n_c n_p)$ [18].

The CLINK Algorithm

Input: The reduced routing matrix R, the sets of congested $\mathcal{P}_C$ paths.
Step 1:
1. Initialize $\mathcal{X}$ to an empty set, and $\mathcal{Q}_B := \mathcal{P}_C$.

Step 2: While $\mathcal{Q}_B \neq \emptyset$
1. Find a link $e_k \in \mathcal{E}_A$ that minimizes $\log \frac{1-p_k}{p_k} / |Domain(e_k)|$.
2. Add e_k to the solution $\mathcal{X}$: $\mathcal{X} := \mathcal{X} \cup \{e_k\}$, set $x_k := 1$.
3. Update the sets: $\mathcal{Q}_B := \mathcal{Q}_B \setminus Domain(e_k)$ and $Domain(e_j) := Domain(e_j) \setminus Domain(e_k)$ for all $e_j \in \mathcal{E}_A$.

Step 3: Output $\mathcal{X}$.

VII. Evaluation

We evaluate the performance of the congested link location algorithms using two metrics: the detection rate (DR), which is the percentage of links that are correctly diagnosed as congested, and the false positive rate (FPR), which is the percentage of links that are good but are diagnosed as congested. With $\mathcal{F}$ denoting the set of the actual congested links, and $\mathcal{X}$ the set of links identified as congested by a location algorithm, these two rates are given by:

$$\text{DR} = \frac{|\mathcal{F} \cap \mathcal{X}|}{|\mathcal{F}|} \; ; \; \text{FPR} = \frac{|\mathcal{X} \backslash \mathcal{F}|}{|\mathcal{X}|}.$$

A. Simulation Evaluation

We first evaluate the CLINK algorithm by simulations in different network topologies. Each link e_k in the network is congested with a probability p_k uniformly distributed between 0 and 1. The actual values of p_k are chosen such that the percentage of congested links equals a parameter f. f is varied in our simulations to evaluate the algorithm under different congestion levels of the network. We use the loss rates model LM1 of [4] where congested links have loss rates uniformly distributed in $[0.05, 1]$ and good links have loss rates in $[0, 0.01]$. Once each link has been assigned a loss rate, the actual losses on each link follow a Gilbert process, where the link fluctuates between good and congested states. When in a good state, the link does not drop any packet, when in a congested state the link drops all packets. The transition between good and congested states is chosen so that the average loss rate matches the loss rate assigned to the link. We also run simulations with Bernoulli losses, where packets are dropped on a link e_k with a fixed probability, but the differences are insignificant. Therefore, we only report results with Gilbert losses in this section.

For each network, we first take 30 snapshots to learn the prior probabilities $\boldsymbol{p}$. After learning $\boldsymbol{p}$, we can infer the congested links in subsequent measurements using the CLINK algorithm. Note here that we only learn the prior vector $\boldsymbol{p}$ once and use it for all subsequent snapshots. The path loss rate is calculated based on the transmissions of 1000 packets. The link threshold t_l is 0.99 whereas the path threshold t_p is t_l^d where d is the length of the path.

1) Results on tree topologies: We first compare the performance of our inference technique with the Smallest Consistent Failure Set (SCFS) algorithm of [7] on tree topologies of 1000 nodes with the maximum branching ratio of 10. We do **not** apply the scaling behavior for large networks as discussed in [7] in our simulations. We repeat each simulation configuration 10 times. The results are shown in Figure 2.

We observe that CLINK is significantly more accurate than SCFS with higher DRs and lower FPRs. The reasons for the

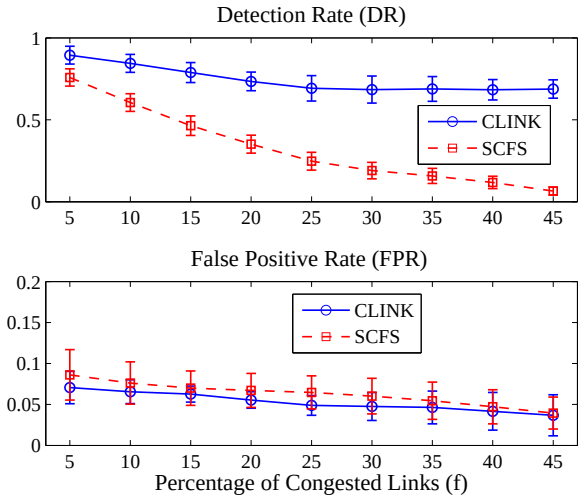

Fig. 2. DR and FPR for the two algorithms CLINK and SCFS in tree topologies of 1000 nodes. Links have different probabilities of being congested. The two figures have different scales on the y-axis.

better performance of the CLINK algorithm boil down to the one fact that it uses previous measurement snapshots to learn about the prior probabilities and therefore does not have the bias against links that belong to many congested paths that SCFS has.

2) Results for mesh topologies: We further compare the CLINK algorithm with the SCFS and the Bayesian approach using Monte Carlo Markov Chain (MCMC) simulation algorithm of [4]. We use different *mesh* topologies generated by BRITE [20] with 1000 nodes and one topology taken from the PlanetLab test-bed [21] with 250 vantage points. In each topology, we choose nodes with the least out-degree as the vantage points. The link loss model is LM1 with $f = 10\%$ congested links.

The SCFS algorithm works only with tree topologies. To evaluate this algorithm on a mesh topology, we first run it separately on each vantage point. The results obtained from each vantage point are the sets of identified congested and good links in the routing tree of that vantage point. We merge the results from all the vantage points as follows. For links that are diagnosed by only one vantage point we keep this diagnosis as the final result. For links that are diagnosed by multiple vantage points, we take the diagnoses given by more than 50% of these vantage points as the final results.

We apply the MCMC algorithm in the same way that it was used for tree topologies in [4]. The accuracy and running time of the MCMC algorithm depend on two parameters: the number of iterations and the number of bins for link loss rates. In this paper, we use 500 iterations and 100 bins (i.e., we divide the $[0, 1]$ interval into 100 intervals of equal length).

We repeat each simulation setting 10 times and report the average DR and FPR in Table I. All the DRs and FPRs have small confidence intervals. For clarity reasons, we omit them from Table I.

We observe again here that CLINK performs better than both SCFS and MCMC. MCMC is less accurate than CLINK because it uses an uninformative (p_k is the same for all links e_k) prior in its Bayesian updates. CLINK has the same running time as SCFS, which is an order of magnitude less than that of MCMC.

TABLE I
ACCURACY OF THE CLINK, SCFS AND MCMC ALGORITHMS.

Topology	CLINK DR	CLINK FPR	SCFS DR	SCFS FPR	MCMC DR	MCMC FPR
Babarasi-Albert	92.0%	0.8%	60.0%	0.9%	70.0%	5.0%
Waxman	91.2%	0.7%	62.0%	0.6%	73.4%	1.3%
Hierarchical	87.3%	0.9%	57.0%	1.1%	76.7%	1.9%
PlanetLab	90.3%	1.1%	61.0%	0.7%	69.4%	2.4%

B. Internet Experiments

We implement our prior learning and CLINK algorithms on 250 nodes of PlanetLab. These nodes are located mainly at universities around the world. 50% of them (124 nodes) are in the US, 40% (100 nodes) are in Europe and the other 10% are in South America, Asia and Australia.

1) Methodology: We first use traceroute to measure the network topology. Traceroute is performed sequentially from each vantage point to all other vantage points. The collected routes will then be combined to form a complete network topology. Using traceroute to build the network topology can lead to errors because of several factors. First, for security and performance reasons, many routers in the Internet do not respond or limit the rate of responses to ICMP queries. The paths to nodes behind these routers cannot be ascertained. According to our own traceroute measurements between PlanetLab hosts, 5 to 10% of routers do not respond to ICMP requests. Second, many routers have multiple interfaces and return different IP addresses to different traceroute requests. We use the sr-ally tool [22] to disambiguate multiple interfaces at a single node. We find that about 16% of routers between PlanetLab nodes have multiple interfaces.

We then measure the loss rate between each pair of vantage points. There are 26250 end-to-end paths among the 250 vantage points. A trivial measurement system will poll all these paths. However, using the technique in [6] to reduce the number of end-to-end measurements, each host only needs to measure approximately 100 end-to-end paths. We use simple UDP packets as probes. Each host sends 100 UDP packets of size 60 bytes to every other host. Time between probes follows an exponential distribution with a mean value of 0.2 seconds. To prevent overloading the nodes, each host sends probes to the other hosts in a random order. Each snapshot measurement therefore takes approximately 4 minutes.

Similarly to the simulations, in the Internet experiments, we also take 30 snapshots to estimate the prior probabilities $\boldsymbol{p}$. That is, we use the snapshots obtained in the previous two hours to learn the prior. We then use these probabilities to infer the congested links for the next measurement snapshot of the network. The following results are the average results of 10 experiments, each is separated by 4 hours from the previous one, starting at 10 a.m. on the 3rd of October, 2005.

TABLE II

PLANETLAB EXPERIMENTS.

	CLINK		SCFS		MCMC	
t_l	DR	FPR	DR	FPR	DR	FPR
0.96	91.8%	2.5%	52.7%	1.6%	73.1%	4.7%
0.98	95.3%	3.1%	58.4%	2.8%	79.6%	5.3%
0.99	97.4%	4.9%	62.9%	3.6%	85.4%	7.5%

TABLE III

RUNNING TIME (IN MINUTES).

	CLINK	SCFS	MCMC
Topology and Loss Rate Measurements	4	4	4
Location Algorithm	1	1	10

2) Results: In the Internet, we do not know the real loss rates of the network links as in the simulations of Section VII-A. Therefore we cannot validate our inference results by comparing them against the true values. We adopt the indirect validation method of [4], where the set of measurements is divided into two sets of equal size: the inference set and the validation set. The partition is done randomly. We run first the prior learning algoritm and then the CLINK algorithm on the inference set to identify the congested links. For each congested link that is identified, we check whether the end-to-end paths in the validation set that contain this link are congested or not. If all of these paths are congested, then the inference is correct. We also run our implementation of the SCFS and MCMC algorithms on the above traffic traces. To calculate the DRs and FPRs for the three algorithms, we first need to determine the set $\mathcal{F}$ of the actual congested links. As we do not know the true link loss rates, we take the heuristic that a link e_k is congested ($e_k \in \mathcal{F}$) if and only if it is identified by at least one of the three algorithms (CLINK, SCFS, and MCMC) on the validation set. We compare the three algorithms under different values of t_l (recall that t_l is the threshold we use to determine the congested links). The results are presented in Table II. Compared to the CLINK algorithm, the MCMC and the SCFS algorithms are again less accurate. The better performance of CLINK is due to the information provided by the priors.

The CLINK algorithm is very fast. Its average running time for 26250 end-to-end paths is 60 seconds on a computer with a 2Ghz processor running Perl scripts. The running time comparisons of the three algorithms are given in Table III. SCFS and CLINK have almost the same running time whereas MCMC takes 10 times longer. Note here that the CLINK algorithm also requires the prior $\boldsymbol{p}$, which can be learnt from snapshots obtained in the past 2 hours. The actual running time of the prior learning algorithm is short: approximately 1 minute in our experiments. The whole implementation of our algorithms on PlanetLab has approximately 1500 lines of Perl code. All data and code are available at [23].

VIII. CONCLUSIONS AND DISCUSSIONS

We have shown in this paper that a small number of snapshots of the network can significantly improve the accuracy of the congested link location problem. Our approach is fast and accurate. Our results are possible thanks to the properties of the Boolean algebra: (i) the prior probabilities can be uniquely identified from end-to-end measurements, (ii) linear combinations of measurements in Boolean algebra increase efficiently the rank of the system in conventional algebra, and (iii) computations in Boolean algebra are simple and fast. In addition to the future extensions discussed in Section IV, we also plan to investigate the relationships between the number snapshots and the learning errors.

ACKNOWLEDGEMENTS

This work is financially supported by grant ManCom 2110 of the Hasler Foundation, Bern, Switzerland. We would like to thank the anonymous reviewers for their stimulating discussions and useful suggestions.

APPENDIX: PROOF OF THEOREM 1

Let us introduce some notations first. $\boldsymbol{e_k}$ is the unit vector whose kth coordinate is 1 and all others 0. $|\boldsymbol{v}|$ is the Hamming weight of the vector $\boldsymbol{v} = [v_1\ v_2\ \ldots\ v_{n_p}]^T$, with T denoting transposition, and which is defined as $|\boldsymbol{v}| = \sum_{i=1}^{n_p} v_i$. Without loss of generality, assume that the columns $\boldsymbol{d}_i$, $1 \leq i \leq n_c$, of $D = [\boldsymbol{d}_1\ \ldots\ \boldsymbol{d}_{n_c}]$ are sorted in increasing values of their Hamming weights, that is,

$$1 \leq |\boldsymbol{d}_1| \leq |\boldsymbol{d}_2| \leq \ldots \leq |\boldsymbol{d}_{n_c}|\,. \tag{12}$$

Remember that all columns of D are distinct and nonzero. We denote by $\boldsymbol{u} \preceq \boldsymbol{v}$ the property that $u_i \leq v_i$ for all $1 \leq i \leq n_p$. Clearly, if $\boldsymbol{u} \preceq \boldsymbol{v}$, then $|\boldsymbol{u}| \leq |\boldsymbol{v}|$.

The following result of Boolean algebra will be useful to prove the theorem.

Lemma 1: With $\boldsymbol{d}_j$ denoting the jth column vector of matrix D, $1 \leq j \leq n_c$, any solution $\boldsymbol{x} = [x_1\ x_2\ \ldots\ x_{n_c}]^T$ of

$$\boldsymbol{d}_j = \bigvee_{k=1}^{n_c} x_k \boldsymbol{d}_k, \tag{13}$$

is such that $x_k = 0$ for all $j+1 \leq k \leq n_c$.

Proof: Clearly, (13) admits $\boldsymbol{x} = \boldsymbol{e}_j$ as a solution. Let $\boldsymbol{x}' \neq \boldsymbol{x}$ be another solution, and suppose that $x'_l = 1$ for some $j+1 \leq l \leq n_c$. Then, as

$$\bigvee_{k=1}^{n_c} x'_k \boldsymbol{d}_k \succeq x'_l \boldsymbol{d}_l = \boldsymbol{d}_l,$$

we get from (13) that $\boldsymbol{d}_j \succeq \boldsymbol{d}_l$ and thus that $|\boldsymbol{d}_j| \geq |\boldsymbol{d}_l|$. Combining this last result with (12), we have the equality $|\boldsymbol{d}_j| = |\boldsymbol{d}_l|$. All columns of D are distinct, hence $\boldsymbol{d}_j$ and $\boldsymbol{d}_l$ have the same number of entries equal to 1 but are different.

Since (13) is linear in Boolean algebra, $\boldsymbol{x} \vee \boldsymbol{x}'$ is also a solution. Hence

$$\boldsymbol{d}_j = \bigvee_{k=1}^{n_c} (x_k \vee x'_k)\boldsymbol{d}_k \succeq x_j \boldsymbol{d}_j \vee x'_l \boldsymbol{d}_l = \boldsymbol{d}_j \vee \boldsymbol{d}_l,$$

from which we get that $|\boldsymbol{d}_j| \geq |\boldsymbol{d}_j \vee \boldsymbol{d}_l|$. Now, we have just shown before that $|\boldsymbol{d}_j| = |\boldsymbol{d}_l|$, but that $\boldsymbol{d}_j$ and $\boldsymbol{d}_l$ differ by at least one entry. Therefore, $|\boldsymbol{d}_j \vee \boldsymbol{d}_l| \geq |\boldsymbol{d}_j| + 1$. But this

would yield in turn that $|\boldsymbol{d}_j| \geq |\boldsymbol{d}_j| + 1$, a contradiction. This proves that $x'_l = 0$ for all $j+1 \leq l \leq n_c$. ■

We can now prove Theorem 1. The necessary condition is trivial, because if two columns of the routing matrix are the same then the two corresponding links are traversed by the same set of paths. Therefore, end-to-end measurements cannot distinguish the states of these two links from each other, and their congestion probabilities are not identifiable. To prove the sufficient condition, we follow the same induction strategy as in Vardi's proof of the identifiability of Poisson traffic flows [17], but contrary to his proof, we need to work in Boolean algebra instead of conventional algebra. The proof has three steps. In Step 1, we show that $\prod(1-p_i) = \prod(1-\tilde{p}_i)$. In Step 2, we show that Equation (4) implies that $p_1 = \tilde{p}_1$. Finally, in Step 3 we use induction to complete the proof. In all three steps we calculate the probabilities of the possible solution of equations (3).

Step 1: Take $\boldsymbol{y} = \mathbf{0}$ (the all zeros vector) in (4), which then reads $\mathbb{P}_{\boldsymbol{p}}(\boldsymbol{Y} = \mathbf{0}) = \mathbb{P}_{\tilde{\boldsymbol{p}}}(\boldsymbol{Y} = \mathbf{0})$. Now, $\boldsymbol{Y} = \mathbf{0}$ if and only if $\boldsymbol{X} = \mathbf{0}$. It follows from the independence of the link states

$$\prod_{k=1}^{n_c}(1-p_k) = \prod_{k=1}^{n_c}(1-\tilde{p}_k). \tag{14}$$

Step 2: Now, take $\boldsymbol{y} = \boldsymbol{d}_1$ in (4). Then Lemma 1 yields that $\boldsymbol{e}_1 = [1\ 0\ \ldots\ 0]^T$ is the unique solution of (3). Hence

$$\mathbb{P}_{\boldsymbol{p}}(\boldsymbol{Y} = \boldsymbol{d}_1) = \mathbb{P}_{\boldsymbol{p}}(\boldsymbol{X} = \boldsymbol{e}_1) = p_1 \prod_{k=2}^{n_c}(1-p_k).$$

As the same equation is valid with $\boldsymbol{p}$ replaced by $\tilde{\boldsymbol{p}}$, we get that

$$p_1 \prod_{k=2}^{n_c}(1-p_k) = \tilde{p}_1 \prod_{k=2}^{n_c}(1-\tilde{p}_k).$$

Combining this relation with (14) yields that $p_1 = \tilde{p}_1$, because we assumed $p_k < 1$ for all $1 \leq k \leq n_c$.

Step 3 (induction): Pick $2 \leq j \leq n_c$, and suppose that $p_k = \tilde{p}_k$ for all $1 \leq k \leq j-1$. We want to show that $p_j = \tilde{p}_j$.

Pick $\boldsymbol{y} = \boldsymbol{d}_j$ in (4). Then (3) becomes (13), and Lemma 1 requires that $X_k = 0$ for $j+1 \leq k \leq n_c$. Denote the events A and A' by $A = \{\vee_{k=1}^{j-1} X_k \boldsymbol{d}_k = \boldsymbol{d}_j\}$ and $A' = \{\vee_{k=1}^{j-1} X_k \boldsymbol{d}_k \preceq \boldsymbol{d}_j\}$. As X_k are independent random variables, we can write that

$$\begin{aligned}
&\mathbb{P}_{\boldsymbol{p}}(\boldsymbol{Y} = \boldsymbol{d}_j) \\
&= \mathbb{P}_{\boldsymbol{p}}\left(\left\{\vee_{k=1}^{j} X_k \boldsymbol{d}_k = \boldsymbol{d}_j\right\} \cap \{X_{j+1} = \ldots = X_{n_c} = 0\}\right) \\
&= \mathbb{P}_{\boldsymbol{p}}\left(\vee_{k=1}^{j} X_k \boldsymbol{d}_k = \boldsymbol{d}_j\right) \prod_{k=j+1}^{n_c}(1-p_k) \\
&= \mathbb{P}_{\boldsymbol{p}}\left(\{A \cap \{X_j = 0\}\} \cup \{A' \cap \{X_j = 1\}\}\right) \prod_{k=j+1}^{n_c}(1-p_k) \\
&= [\mathbb{P}_{\boldsymbol{p}}(A)(1-p_j) + \mathbb{P}_{\boldsymbol{p}}(A')p_j] \prod_{k=j+1}^{n_c}(1-p_k) \\
&= \left[\mathbb{P}_{\boldsymbol{p}}(A) + \frac{p_j}{1-p_j}\mathbb{P}_{\boldsymbol{p}}(A')\right] \prod_{k=j}^{n_c}(1-p_k). \qquad (15)
\end{aligned}$$

Both events A and A' depend only on X_k for $1 \leq k \leq j-1$. Hence it follows from the induction assumption that $\mathbb{P}_{\boldsymbol{p}}(A) = \mathbb{P}_{\tilde{\boldsymbol{p}}}(A)$ and $\mathbb{P}_{\boldsymbol{p}}(A') = \mathbb{P}_{\tilde{\boldsymbol{p}}}(A')$. Combining the induction assumption with (14), we also get that $\prod_{k=j}^{n_c}(1-p_k) = \prod_{k=j}^{n_c}(1-\tilde{p}_k)$. Repeating the same reasoning for (15) with $\boldsymbol{p}$ replaced by $\boldsymbol{p'}$, we obtain therefore that

$$\frac{p_j}{1-p_j}\mathbb{P}_{\boldsymbol{p}}(A') = \frac{\tilde{p}_j}{1-\tilde{p}_j}\mathbb{P}_{\boldsymbol{p}}(A').$$

As $\mathbb{P}_{\boldsymbol{p}}(A') \geq \mathbb{P}_{\boldsymbol{p}}(X_1 = \ldots = X_{j-1} = 0) = \prod_{k=1}^{j-1}(1-p_k) > 0$, it follows that $p_j = \tilde{p}_j$. This concludes the proof of the induction step and the theorem.

REFERENCES

[1] A. Adams et al., "The use of end-to-end multicast measurements for characterizing internal network behavior," *IEEE Communications Magazine*, May 2000.

[2] M. Coates, A. Hero, R.Nowak, and B. Yu, "Internet tomography," *IEEE Signal Processing Magazine*, vol. 19, May 2002.

[3] W. Wei, B. Wang, D. Towsley, and J. Kurose, "Model-based identification of dominant congested links," in *Proceedings of the ACM Internet Measurement Conference*, 2003.

[4] V. N. Padmanabhan, L. Qiu, and H. J. Wang, "Server-based inference of internet performance," in *Proceedings of the IEEE INFOCOM'03*, San Francisco, CA, April 2003.

[5] I.-T. R. G.107, "The e-mode, a computational model for use in transmission planning," March 2003.

[6] H. X. Nguyen and P. Thiran, "Active measurement for failure diagnosis in IP networks," in *Proc. of PAM 2004*, Juan-les-Pins, France, 2004.

[7] N. G. Duffield, "Network tomography of binary network performance characteristics," *IEEE Transactions on Information Theory*, vol. 52, no. 12, pp. 5373–5388, Dec. 2006.

[8] T. Bu, N. Duffield, F. L. Presti, and D. Towsley, "Network tomography on general topologies," in *Proceedings of ACM Sigmetrics 2002*, Marina Del Rey, CA, 2002.

[9] A. Batsakis, T. Malik, and A. Terzis, "Practical passive lossy link inference," in *Proc. of PAM 2005*, 2005.

[10] Y. Zhao, Y. Chen, and D. Bindel, "Scalable deterministic overlay network diagnosis," in *Proceedings of ACM SIGCOMM*, Pisa, Italy, 2006.

[11] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson, "User-level internet path diagnosis," in *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP'03)*, 2003, pp. 106–119.

[12] V. Jacobson, "traceroute, ftp://ftp.ee.lbl.gov/traceroute.tar.z," 1989.

[13] Y. Shavitt, X. Sun, A. Wool, and B. Yener, "Computing the unmeasured: An algebraic approach to internet mapping," *IEEE J. on Selected Areas in Communications*, vol. 22, January 2004.

[14] Y. Zhang, N. Duffield, V. Paxson, and S. Shenker, "On the constancy of internet path properties," in *Proceedings of ACM SIGCOMM Internet Measurement Workshop*, San Francisco, 2001.

[15] Y. Vardi, "Network tomography: Estimating source-destination traffic intensities," *Journal of the American Statistical Association*, vol. 91, pp. 365–377, 1996.

[16] Y. Chen, D. Bindel, H. Song, and R. H. Katz, "An algebraic approach to practical and scalable overlay network monitoring," in *Proceedings of the ACM SIGCOMM*, Portland, August-September 2004.

[17] Y. Vardi and D. Lee, "From image deblurring to optimal investment: Maximum likelihod solutions for positive linear inverse problem," *Journal of Royal Statistical Society, Series B (Methodological)*, vol. 55, pp. 569–612, 1993.

[18] D. Hochbaum, *Approximation Algorithms for NP-Hard Problems*. PWS Publishing Company, Boston, MA, 1997.

[19] J. E. Beasley, "An Algorithm for Set Covering Problem," *European Journal of Operational Research*, vol. 31, pp. 85–93, 1987.

[20] A. Medina, I. Matta, and J. Byers, "On the origin of power-laws in internet topologies," *ACM Computer Communication Review*, pp. 160–163, 2000.

[21] www.planet lab.org.

[22] N. Spring, D. Wetherall, and T. Anderson, "Scriptroute: A public internet measurement facility," in *USENIX Symposium on Internet Technologies and Systems (USITS)*, 2003.

[23] http://netscope.epfl.ch.