# Link Status Monitoring Using Network Coding

Mohammad H. Firooz, *Student Member, IEEE,* Sumit Roy, *Fellow, IEEE,*
Christopher Lydick, *Student Member, IEEE,* and Linda Bai, *Student Member, IEEE*

***Abstract*—In network tomography, we seek to infer link status parameters (such as delay) *inside a network* through *end-to-end* measurements at (external) boundary nodes. As can be expected, such approaches generically suffer from identifiability problems; i.e., even with the maximum amount of data obtainable from such end-to-end probes, status of links in a large number of network topologies is not identifiable. A basic result characterizing network topologies which are not identifiable under end-to-end probe sending is first derived. Then, we introduce an innovative approach based on linear network coding that overcomes this problem. We provide sufficient conditions on network coding coefficients and training sequence under which any *logical network* is guaranteed to be identifiable. In addition, we show that it is possible to locate any congested link inside a network during an arbitrary amount of time by increasing size of transmitted packets, leading to raise in complexity of the method. Further, given an appropriate choice of training sequence, a probability of success is provided for a random network. OPNET is used to implement the concept and confirm the validity of the claims - simulation results show that the network coding method correctly detects the congested link, while probing based algorithms fail.**

***Index Terms*—Network Tomography, Network Coding, Graph Theory, Finite Field**

## I. Introduction

Monitoring of link properties (delay, loss rates etc.) within the Internet has been stimulated by the demand for network management tasks such as fault and congestion detection. This would help network engineers and Internet Service Providers (ISP) to keep track of network utilization and performance. The need for accurate and fast network monitoring method has increased further in recent years due to the complexity of new services (such as video-conferencing, Internet telephony, and on-line games) that require high-level quality-of-service (QoS) guarantees. In 1996, the term *network tomography* was coined by Vardi [1] to encompass these class of approaches that seek to infer internal link parameters and identify link congestion status.

Current network tomography methods can be broadly categorized as follows:

- **Node-oriented:** These methods are based on cooperation among network nodes on an end-to-end route using *control* packets. For example, active probing tools such as ping or traceroute, measure and report attributes of the round-trip path (from sender to receiver and back) based on separate probe packets[2]. The challenges of such node-oriented methods arise from the fact that many service providers do not own the entire network and hence do not have access to the internal nodes[3].
- **Path-oriented:** In networks with a defined *boundary*, it is assumed that access is available to all nodes at the edge (and not to any in the interior). A boundary node sends probes to all (or a subset) of other boundary nodes to measure packet attributes on the path between network end-to-end points. Clearly, these *edge-based* methods do not require exchanging special control messages between interior nodes. The primary challenge of such end-to-end probe data [4],[5] to estimate *link level* status is that of *identifiability*, as will be discussed later.

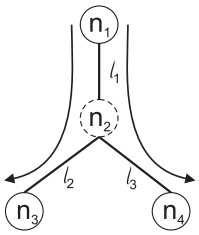



Fig. 1. In end-to-end measurement methods, probes are sent from one boundary node to the others.

As the Internet evolves towards decentralized, uncooperative, heterogeneous administration and edge-based control, node-oriented tools will be limited in their capability. Accordingly, in this work we only focus on path-oriented methods which have recently attained more attention due to their ability to deal with uncooperative and heterogeneous (sub)networks.

In path-oriented network tomography, probes are sent between two boundary nodes on pre-determined routes; typically these are the shortest paths between the nodes. For some parameters like delay (which is the main concern in this manuscript), an additive linear model adequately captures the relation between end-to-end and individual link delays, and can be written as [6], [7]

$$\mathbf{y} = \mathbf{R}\mathbf{x} \tag{1}$$

where $\mathbf{x}$ is the $L \times 1$ vector of individual link delays. The $J \times L$ *binary* matrix $\mathbf{R}$ denotes the routing matrix for the network graph corresponding to the measurements and $\mathbf{y} \in \mathbb{R}^J$ is the measured $J$-vector of end-to-end path delays. Solution approaches based on Eq. (1) can be largely categorized as follows:

1) **Deterministic models**: Here the link attributes, such as link delay, are considered as unknown but constant; the goal of network tomography is to estimate the value of those constants. Since the link delay is typically time varying in any network, this approach is suitable for periods of local 'stationarity' where such an assumption is valid.
2) **Stochastic model**: Here, it is supposed that the link vector $\mathbf{x}$ is specified by a suitable probability distribution. The goal of network tomography is to identify

the unknown parameters of the probability model. For example, many works assume the link attributes follow a Gaussian distribution or an exponential distribution [1-3]. Further, the observations are assumed to occur in the presence of an independent additive noise or interference term $\epsilon$ [8]; thus the observation equation is modified to $\mathbf{y} = \mathbf{A}\mathbf{x} + \epsilon$.

There exist challenges with both modeling approaches. Our work falls within the class of deterministic approaches. Stochastic approaches in the literature are Bayesian in nature, requiring a prior distribution. If incorrectly chosen, this lead to biases in the resulting estimates. Further, stochastic models are usually more computationally intensive than deterministic ones [4]. On the other hand, deterministic models suffer from generic *identifiability* problems; this will be discussed subsequently in more detail. In Eq. (1), typically, the number of observations $J \ll L$, because the number of accessible boundary nodes is much smaller than number of links inside the network. Thus the number of variables in Eq. (1) to be estimated is much larger than number of equations ($rank(\mathbf{R}) < L$)[8] in the linear model, leading to generic non-uniqueness for any solution to Eq. (1),i.e., inability to uniquely specify which links are congested [9].

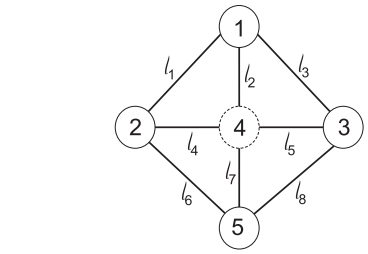
*Definition*: A network is said to be identifiable for a given monitoring scheme (choice of sources, receivers, whether intermediate nodes collaborate) if congestion status of all links inside the network can be inferred from the measurements at the receiver(s) [10].

For example consider the network in Figure 1. Throughout this manuscript, boundary nodes are depicted as solid circles while intermediate nodes are presented using dashed circle. Network in Figure 1 has three boundary nodes, one intermediate node and three links. Suppose probes are transmitted from $n_1$ to $n_3$ and $n_4$. A probe sent from $n_1$ to $n_3$ goes through links $l_1$ and $l_2$ and experiences a total delay $d_{l_1}+d_{l_2}$. Similarly, probe sent from $n_1$ to $n_4$ contains the delay of $d_{l_1}+d_{l_3}$. This can be written in the following set of linear equations:

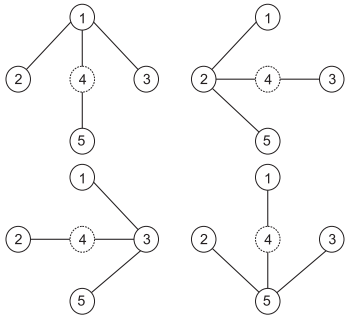
$$\underbrace{\begin{bmatrix} D_{n_1 \rightsquigarrow n_3} \\ D_{n_1 \rightsquigarrow n_3} \end{bmatrix}}_{\text{measurement vector}} = \underbrace{\begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}}_{\text{routing matrix}} \cdot \underbrace{\begin{bmatrix} d_{l_1} \\ d_{l_2} \\ d_{l_3} \end{bmatrix}}_{\text{delay vector}} \tag{2}$$

Let us assume that the only desired solutions in this framework are binary vectors $\mathbf{x}$ whereby $x_i = 1(0)$ indicates if the corresponding link is congested (not congested). While this is a simplifying assumption - that only congested link experience significant delay and is indicated by a corresponding entry of large magnitude in $\mathbf{x}$, whereas the other entries of $\mathbf{x}$ are relatively small, corresponding to low delays for non-congested links, it has been adopted for modeling in the literature, notably by [11].

Even for such an elementary network, it is not possible to disambiguate the following 'congestion' link states: (a) $\mathbf{x} = [1\ 0\ 0]^T$ where only link $l_1$ is congested from (b) $\mathbf{x} = [0\ 1\ 1]^T$ indicating both links $l_2, l_3$ are congested. For larger networks, it is evident that such lack of identifiability is generic.




**(a)**



**(b)**

$$\begin{array}{l} 1 \rightsquigarrow 2 \\ 1 \rightsquigarrow 3 \\ 1 \rightsquigarrow 5 \\ 2 \rightsquigarrow 3 \\ 2 \rightsquigarrow 5 \\ 3 \rightsquigarrow 5 \end{array} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

**(c)**

Fig. 2. An example of graph where end-to-end probe sending methods fail (a)Network topology (b)Spanning trees rooted at boundary nodes (c)routing matrix.

A potential solution to the above is to limit the maximum number of simultaneously congested links inside the network. This is reasonable in situations that the probability of having $k$ congested link in the network is proportional to $p^k$ where $p$ is the probability of a link being congested[1]. For a well-designed network, $p$ is sufficiently small, implying that $p^k$ can be considered negligible beyond some value of $k$. This allows an important side constraint to be imposed on any solution to Eq. 1 - namely, we are interested in binary vectors with at most $k$ non-zero entries in $\mathbf{x}$. In this work, we concentrate on $k = 1$; i.e. only a single congested link exists inside the network. This is the simplest possible class of identifiability problems (compared to the general and more difficult $k > 1$ case) and yet is sufficiently challenging as our investigations will show.

Xi et. al. in [12] provided conditions on the network routing matrix ($\mathbf{R}$) such that a single congested link is detectable by end-to-end delay measurements. We restate the theorem below for reference.

***Theorem** 1:* [12] End-to-end probe based measurements can detect a unique congested link in a network if and only if there are no two identical columns in the network routing matrix.

As illustration, consider the network graphs in Figure 2 and suppose that one of the links $l_2$ or $l_7$ is congested (and all

[1]This is consistent with the assumption that link failures are independent.

others are uncongested). Figure 2-b shows all the possible shortest paths to all other boundary nodes starting with a source (boundary) node in the network. Obviously, a probe either goes through both $l_1$ and $l_7$ or neither. It is now evident that it is not possible to identify which of $l_2$ or $l_7$ is congested link using end-to-end measurement. The routing matrix of the network is given in 2-C. Consistent with Theorem 1, it can be seen that this network is unidentifiable since columns 4 and 5 are identical. However suppose that in place of shortest path routing, we are allowed to route packets from $n_1$ to $n_2$ through a longer path that includes $l_2, l_4$. It is now possible to distinguish between congestion status of links $l_2$ and $l_7$. This exemplifies an intrinsic limitation for end-to-end measurement methods based on shortest path routes - probes transmitted along such paths contain only minimum information. If it was possible to exchange probes between boundary nodes via other (non-shortest) paths, it would improve network identifiability. However, this is often impractical in path-oriented methods that have fixed routing tables. Thus we accordingly propose a new approach - based on network coding - to achieve the same purpose of enhanced identifiability. Thanks to broadcast nature of network coding, a transmitted probe will traverse almost every paths between two boundary nodes.

We will show that employing network coding at intermediate nodes results in a novel link monitoring scheme that is able to identify a congested link in a network graph deemed unidentifiable using end-to-end probe based observation methods (such as network in Figure 2). In this method, a boundary (source) node sends a training sequence of $q$-bit packets to another boundary node (destination). These packets traverse over multiple paths between source and destination. By transmitting different probes at different periods of time and by sweeping over all boundary nodes, a collection of received packets is obtained that contains sufficient information about all internal network links. The primary design challenge is to determine a) the length of training sequence and b) size of network coding packets $q$ necessary to unambiguously identify any congested link in the network.

Network coding has received considerable attention in recent years for its potential for achieving the theoretical upper bound (max-flow, min-cut) of network resource utilization via the introduction of coding concepts at the network layer. It has been shown that with simple random linear coding in place of the usual forwarding, system throughput can be increased in several canonical network topologies. As a result, deployment of network coding in large scale networks is anticipated in future. Our intent in this work is different: to redirect network coding concepts towards a novel application, that of network status monitoring for graphs hitherto assumed un-identifiable by other means.

Our specific contributions in this work are summarized next:

1) We provide conditions under which network coding is able to locate a single congested link in a network. Specifically, we show that our approach succeeds for any *logical network* - i.e. a network containing no nodes with degree two [10].
2) We provide a relation between length of training sequence needed (time to identifiability) and size of network coding packets (complexity of method) to establish a fundamental speed/compexity tradeoff.
3) We provide a lower bound for probability of success of our method in a random graph.
4) We implement our proposed method within OPNET simulator - the first implementation of network coding within actual network simulator to the best of our knowledge - and demonstrate the validity of our ideas.

The paper is organized as follows: In Section II, we develop the principles for applying network coding to tomography for an ideal (delay-free) network. In Section III, we describe how the above scheme can be adapted to the practically important scenario of networks with finite delays. Implementation of linear network coding (LNC) for such a network in a commercially available simulator (OPNET) is described in Section IV for schemes proposed in Section II. The simulation results confirm that LNC correctly detects the congested link in situations where standard probing based algorithm fails. The paper concludes with reflections on future work in Section V. Appendix A provides a brief summary of relevant *Galois finite field* theory. Proofs of theorems can be found in Appendix B. A glossary of definitions of all symbols used in the manuscript is given in Appendix C.

*Notations*: We use bold capitals (e.g. $\mathbf{R}$) to represent matrices and bold lowecase symbols (e.g. $\mathbf{y}$) for vectors. The $i$-th entry of a vector $\mathbf{x}$ is denoted by $x_i$. A set of sets is denoted by a calligraphic capitalized symbol, e.g. $\mathcal{P}$ and the $i$-th element, which is itself a set, is denoted by regular capital symbol with $i$ as superscript (e.g. $P^i$).

## II. Tomography with Linear Network Coding

In principle, Linear Network Coding (LNC) is a block code operating on *IP layer* frames, implemented by routers inside the network. The coding is conducted over the finite field $\mathbb{F}_{2^q}$ whereby each coded symbol can be represented by $q$-bits within an IP layer frame. For the sake of simplicity, we initially consider a delay-free network as in [13], [14], [15] in which information reaches every node instantaneously; our method is readily adapted to a real network where links have finite (non-zero) delay. Linear Network Coding has been exploited in [16] and [17] to infer network topology. Consistent with these approaches, we assume that in addition to LNC coefficients at each node, destination nodes are aware of the entire network topology. Ho et. al. [18] consider network monitoring using network coding. However, their approach is based on multicast tree to find the congested link which implies that the method is restricted to probing using only a unique path between source-destination pairs. We shall relax this assumption in our work.

### A. Formulation

We model a communication network consisting of bidirectional links connecting transmitters, switches, and receivers as an undirected graph $G(V, E)$ where $V$ is a set of vertices and $E$ is a set of edges. Only networks with logical topology are considered here; i.e. degrees of all (interior) nodes in the network (except sources and destinations) are greater than or equal to three, since networks with degree 2 nodes are

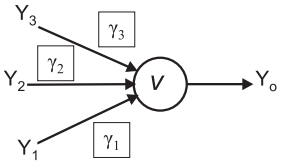



Fig. 3. In network coding, output signal is a linear combination of inputs: $Y_o = \gamma_1 Y_1 + \gamma_2 Y_2 + \gamma_3 Y_3$.

well-known to be un-identifiable by any end-to-end probing techniques.

In LNC [15], intermediate nodes linearly combine packets received on their incoming edges and broadcast the result over all outgoing edges. The coded symbols are transmitted as vectors of bits of length $q$, represented as an element of the finite field $\mathbb{F}_{2^q}$. The length of vectors is equal in all transmission and all links are assumed to be synchronized [19], [20] with zero delay.

As noted in [14], network coding presumes *directed acyclic graphs*. Gjoka et. al. in [10] propose an algorithm that converts a given graph $G(V, E)$ with a set of sources $S$ to an acyclic directed graph with the same sets of vertices and edges. However, this algorithm doesn't allow destination nodes to be pre-specified. This makes their algorithm inapplicable for our purpose, since in network tomography, we have access only to pre-determined boundary nodes. Converting an undirected graph to an acyclic directed graph with given sets of sources and destinations is beyond the scope of this paper. Hence, we shall thus presume that for each undirected graph $G(V, E)$, there exists a corresponding acyclic directed version; all results in this manuscript are assumed to be based on such a directed and acyclic graph with known topology.

### *B. Network Code Design*

Consider a source $s \in V$ and a destination $d \in V$ pair in the network. For a given network graph $G(V, E)$, all nodes apart from the source-destination pair $v \in V - \{s, d\}$, support network coding. The signal $Y_l$ on an outgoing link $l \in E$ for node $v$ is a linear combination (in finite field $\mathbb{F}_{2^q}$) of the signals $Y_j$ on the incoming links of $v$ (see Figure 3), i.e.,

$$Y_l = \sum_{\{j \in E | d(j) = v\}} \gamma_j Y_j \quad , v = o(l), l \in E \tag{3}$$

where $o(l)$ and $d(l)$ represent origin and destination nodes of link $l \in E$, respectively. Operations (addition and multiplication) in Eq. (3) are in finite field $\mathbb{F}_{2^q}$. A self-contained summary is given in Appendix-A; for more details, readers should refer, for example, to [21].

For each node $v \in V - \{s, d\}$, $\mathcal{P}(v)$ is defined as the collection of all paths from $s$ to $v$ in the directed graph $G(V, E)$; it's $i$-th element $P^i(v)$ is the $i$-th path between $s$ and $v$. Suppose there are total of $N$ paths from source $s$ to destination $d$, i.e., $N = |\mathcal{P}(d)|$. Further suppose that the source $s$ has $K$ outgoing links $e_1, e_2, ..., e_K$ and the symbol $\alpha_k$ is sent over the $k$-th outgoing link $e_k$, $k = 1, 2, \ldots, K$. Let $\mathcal{P}_{e_k}(d)$ denote the collection of paths from source to destination that share the $k^{th}$ outgoing link from the source, i.e.,

$$\mathcal{P}_{e_k}(d) = \{P^i(d) : e_k \in P^i(d) \; s.t. \; o(e_k) = s\} \tag{4}$$

In fact sets $\mathcal{P}_{e_k}$, $k = 1, ..., K$ are partitions of $\mathcal{P}(d)$; i.e. $\mathcal{P}(d) = \cup_{k=1}^{K} \mathcal{P}_{e_k}(d)$ and $\mathcal{P}_{e_i}(d) \cap \mathcal{P}_{e_j}(d) = \phi$, $i \neq j$.

If the source sends a symbol $\alpha \in \mathbb{F}_{2^q}$ over $P^i(d)$, the destination would receive

$$y[n] = \alpha \prod_{l \in P^i(d)} \gamma_l \tag{5}$$

$$= \alpha \beta_i(G) \tag{6}$$

where $\gamma_l \in \mathbb{F}_{2^q}$ is the coefficient of the link $l$ on path $P^i(d)$ and $\beta_i(G) \in \mathbb{F}_{2^q}$ is the product of LNC coefficients of all links lying on the $i$-th path from source to destination, $P^i(d)$. The argument $G$ in $\beta_i(G)$ highlights the dependency of $\beta_i$ on topology $G$. Now, suppose $s$ sends the symbol $\alpha_k[n]$ over the $k$-th outgoing link $e_k, k = 1, \ldots, K$ in time slot $n$ (which implies in turn that $\alpha_k[n]$ traverses over all paths $P^i(d) \in \mathcal{P}_{e_k}$). Since network coding is a linear operation, the destination receives, by (6) the following super-imposed symbol in time slot $n$ :

$$y[n] = \sum_{k=1}^{K} \alpha_k[n] . \sum_{P^i(d) \in \mathcal{P}_{e_k}} \prod_{l \in P^i(d)} \gamma_l \tag{7}$$

Eq. (7) can be rewritten as the vector product

$$y[n] = \boldsymbol{\alpha}^T[n] \, \boldsymbol{\beta}(G) \tag{8}$$

where

$$\boldsymbol{\alpha'}_k^T[n] = \overbrace{[\alpha_k[n] \, \alpha_k[n] \, \ldots \, \alpha_k[n]]}^{|\mathcal{P}_{e_k}| \text{ times}} \tag{9}$$

$$\boldsymbol{\alpha}^T[n] = [\boldsymbol{\alpha'}_k^T[n]]_{k=1}^{K} \tag{10}$$

and

$$\boldsymbol{\beta}^T(G) = [\prod_{l \in P^i(d)} \gamma_l]_{i=1}^{N} = [\beta_i(G)]_{i=1}^{N} \tag{11}$$

We call $\boldsymbol{\beta}(G)$ the *total network coding vector of the graph* $G$; the $i$-th entry of $\boldsymbol{\beta}$ is the product of LNC coefficients of all nodes lying on the $i$-th path from source to destination, $P^i(d)$. Note that $\boldsymbol{\alpha'}_k[n]$ is a repeated length-$|\mathcal{P}_{e_k}|$ vector comprising of the symbol $\alpha_k[n]$ (sent over the $k$-th outgoing link of the source at $n$-th time slot). If $M$ symbols that constitute a packet are sent in $M$ time slots, the destination receives:

$$\mathbf{y}_{M \times 1} = \mathbf{A}_{M \times N} \, \boldsymbol{\beta}(G)_{N \times 1} \tag{12}$$

where $\mathbf{A}$ is a $M \times N$ matrix whose $n^{th}$ row is $\boldsymbol{\alpha}^T[n]$, the training symbols sent in time slot $n$ (10). It follows by construction that columns of $\mathbf{A}$ corresponding to $\mathcal{P}_{e_k}$ are equal, for $k = 1, 2, ..., K$.

Now, if a link is severely congested, packets are significantly delayed and assumed lost at the destination. Hence, we can model the network with link $l$ in congestion state by its *edge deleted subgraph* denoted by $G_l(V, E_l)$ (for precious definition of an edge deleted subgraph refer to [22]). In the other words, for the original network $G(V, E)$, $G_l(V, E_l)$ represents the same network with link $l$ is congested such that packets sent on that link are dropped. The total network coding vector of the graph $G_l$, denoted by $\boldsymbol{\beta}(G_l)$, is related to the vector $\boldsymbol{\beta}(G)$, defined in (11) as follows. Clearly, if the congested link doesn't belong to $i$-th path from source to destination, $P^i(d)$, it will

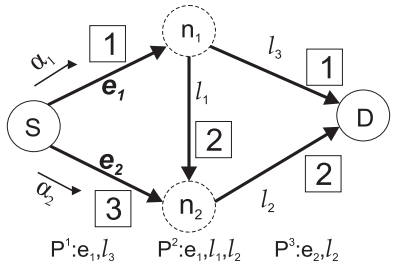



Fig. 4. A network with two boundary nodes and two interior nodes which contains three paths from source (node $S$) to destination (node $D$). Linear network coding coefficient of each link is shown in a box next to the link.

not affect packets going through those paths. That means the $i$-th entry of $\boldsymbol{\beta}(G_l)$ equals $i$-th entry of $\boldsymbol{\beta}(G)$ and it is zero if $P^i(d)$ includes the deficient link, i.e.,

$$\begin{aligned} \beta_i(G_l) &= \begin{cases} \beta_i(G) & if \;\; l \notin P^i(d) \\ 0 & o.w. \end{cases} \\ (\boldsymbol{\beta}(G_l))^T &= [\beta_i(G_l)]_{i=1}^{i=N} \end{aligned} \tag{13}$$

where $\beta_i(G)$ is the $i$-th entry of $\boldsymbol{\beta}(G)$ (or equivalently product of coefficients of all links on path $P^i(d)$) defined in (11) and $l$ is the congested link.

Now suppose, $s$ sends $\alpha_k[n]$ over path $\mathcal{P}_{e_k}$ in time slot $n$ given link $l$ is congested. The destination node receives

$$y^l[n] = \sum_{k=1}^{K} \alpha_k[n]. \sum_{P^i(d) \in \mathcal{P}_{e_k}} \beta(G_l)_i \tag{14}$$

in time slot $n$. Using (13), (10) and (14), the following vector form equation can be derived when link $l$ is in congestion:

$$y^l[n] = \boldsymbol{\alpha}^T[n].\boldsymbol{\beta}(G_l) \tag{15}$$

Sending probe packets in $M$ contiguous time slots results in $M$ linear equations which can be written as:

$$\mathbf{y}^l_{M\times 1} = \mathbf{A}_{M\times N}\, \boldsymbol{\beta}(G_l)_{N\times 1} \tag{16}$$

where $\mathbf{y}^l$ is vector of symbols observed at the destination in $M$ time slots with link $l$ congested. Comparing equations (12) and (16) shows that for given matrix $\mathbf{A}$, the received symbols change in response to link congestion. In the next subsection we will prove that this occurs if the network coding vector of the graph $G$ are chosen to satisfy certain conditions, leading to the potential for identifying the congested link. We next provide an illustrative example.

**Example 1:** Consider the topology in Figure 4 that consists of 4 nodes and 3 paths between the source and destination. The source has two output links, $e_1$ and $e_2$. Hence, using (13), $\mathcal{P}_{e_1}$ and $\mathcal{P}_{e_2}$ are:

$$\begin{aligned} \mathcal{P} &= \{P^1, P^2, P^3\} \\ \mathcal{P}_{e_1} &= \{P^1, P^2\} \\ \mathcal{P}_{e_2} &= \{P^3\} \end{aligned}$$

where the end-to-end paths $P^1, P^2, P^3$ between the source-destination are as defined in Figure 4.

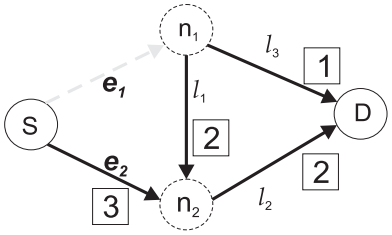



Fig. 6. $G_{e_1}$: link $e_1$ is congested and therefore node $n_1$ no longer receives packets from $e_1$. Dashed arrow represents deleted link from network $G$.

Suppose $\alpha_1$ and $\alpha_2$ traverse over $e_1$ and $e_2$ respectively. Further, suppose symbols are from $\mathbb{F}_{2^2}$ implying they are 2 bits long. In that case, the output of each intermediate node and the destination is depicted in Figure 5. Since the network is considered to be delay-free, all nodes receive their information instantaneously.

From Figure 5, the received symbol at destination is

$$\begin{aligned} y[n] &= (\alpha_1 \times (1 \times 1 + 1 \times 2 \times 2)) + (\alpha_2 \times (3 \times 2)) \\ &= 2\alpha_1 + \alpha_2 \end{aligned}$$

As defined in (6), $\beta_i(G)$ is defined as product of coefficients of all links on path $P^i(d)$. For example $\beta_2(G)$ is as follows:

$$\beta_2(G) = 1 \times 2 \times 2 = 3 \tag{17}$$

where the operation is over the finite field $\mathbb{F}_{2^2}$ (refer to Appendix A).

From Equations (11) and (10), the vector $\boldsymbol{\alpha}$ and *total network coding vector* $\boldsymbol{\beta}$ are obtained as:

$$\boldsymbol{\alpha}^T = \begin{bmatrix} \alpha_1 & \alpha_1 & \alpha_2 \end{bmatrix}$$

$$\boldsymbol{\beta}(G) = \begin{bmatrix} 1 & 3 & 1 \end{bmatrix}^T$$

where $\alpha_1$ and $\alpha_2$ are two symbols in $\mathbb{F}_{2^2}$.

For example, suppose at time slot 1, the source in Figure 4 sends symbols $\alpha_1 = 1 \in \mathbb{F}_{2^2}$ and $\alpha_2 = 2 \in \mathbb{F}_{2^2}$, respectively. At time slot 2, it transmits $\alpha_1 = 3$ and $\alpha_2 = 3$. In this case, (12) becomes:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 1 & 2 \\ 3 & 3 & 3 \end{bmatrix}}_{\text{training sequence } \mathbf{A}} \cdot \underbrace{\begin{bmatrix} 1 \\ 3 \\ 1 \end{bmatrix}}_{\boldsymbol{\beta}(G)} = \underbrace{\begin{bmatrix} 0 \\ 2 \end{bmatrix}}_{\text{received symbols}} \tag{18}$$

Now suppose link $e_1$ is congested; then $G_{e_1}$ depicted in Figure 6, represents the graph model for this case. As defined in (13), for edge deleted subgraph $G_{e_1}$, the total network coding vector is now given by

$$\boldsymbol{\beta}(G_{e_1}) = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$$

The first and second entries of $\boldsymbol{\beta}(G_{e_1})$ are zero because $e_1 \in P^1$ and $e_1 \in P^2$ (refer to Eq. (13)). For the same symbols sent by source in case of congested link $e_1$, the destination receivers

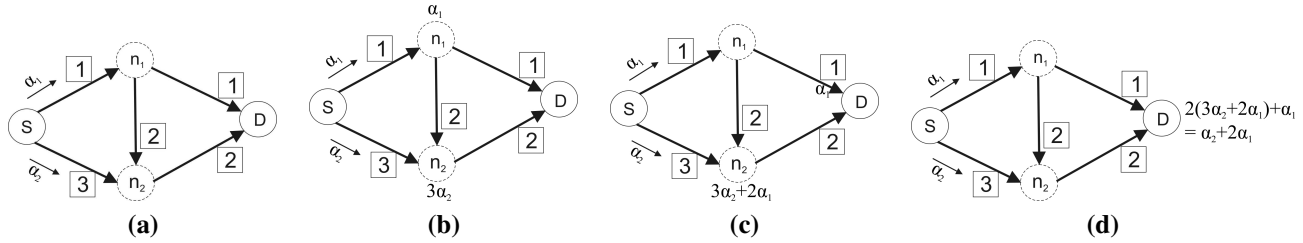



Fig. 5. Output of each intermediate node when source sends symbols $\alpha_1$ and $\alpha_2$ over its outgoing links. For delay-free network this is instantaneous.

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 1 & 2 \\ 3 & 3 & 3 \end{bmatrix}}_{\text{training sequence } \mathbf{A}} \cdot \underbrace{\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}}_{\boldsymbol{\beta}(G_{e_1})} = \underbrace{\begin{bmatrix} 2 \\ 3 \end{bmatrix}}_{\text{received symbols}} \tag{19}$$

which is different from the case of no link failure as in (18). For the same transmitted symbols, Table I contains the received symbols at destination for all cases of single link congestion in the network. Such a table may then be used as a lookup to locate/identify the congested link inside the network.

| Congested link | None | $e_1$ | $e_2$ | $l_1$ | $l_2$ | $l_3$ |
|---|---|---|---|---|---|---|
| 1st time slot | 0 | 2 | 2 | 3 | 1 | 1 |
| 2nd time slot | 2 | 3 | 1 | 0 | 1 | 3 |

TABLE I
SYMBOLS RECEIVED AT DESTINATION FOR TOPOLOGY IN FIGURE 4 WITH TRAINING SEQUENCE TRANSMITTED IN TWO TIME SLOTS GIVEN IN EQ.(18)

For $q = 2$, each received symbol at the destination belongs to the symbol set $\{0, 1, 2, 3\}$ representing the elements of $\mathbb{F}_{2^2}$. Therefore, for 2-bit long network coding, in each transmission (in one time slot) received symbol conveys only 4 different values. On the other hand, there are five links in the network which means receiver has to distinguish between six different states: single link congestion for each of the five links and no congestion case. These observations leads to the fact that for $q = 2$, the number of transmissions required in this simple network cannot be less than two time slots.

It follows that in general, the number of transmission time slots required depends on the choice of $q$, which can be treated as a design variable. For example with $q = 3$, the topology in Figure 5 is identifiable by transmission in just one time slot with $\alpha_1 = 1$ and $\alpha_2 = 4$. Table III shows received symbol at destination for each link congestion state in this case. We formalize this later in Theorem 4.

| Congested link | None | $l_1$ | $l_2$ | $l_3$ | $l_4$ | $l_5$ |
|---|---|---|---|---|---|---|
| 1st time slot | 6 | 4 | 2 | 5 | 7 | 1 |

TABLE II
SYMBOLS RECEIVED AT DESTINATION FOR TOPOLOGY IN FIGURE 5 WITH TRAINING SEQUENCE TRANSMITTED IN ONE TIME SLOT $\mathbf{A} = [1\ \ 1\ \ 4]$

### C. Link Identifiability Results

As explained via the example above, received symbols at the destination change in the event of a link failure. The following theorem describes the conditions on training sequence ($\mathbf{A}$) and LNC coefficients, under which there exists a one-to-one correspondence between link congestion states and received symbols at the destination.

***Theorem** 2:* Consider an acyclic, directed and connected graph $G(V, E)$, with a source- destination pair $s \in V$, $d \in V$. Let $N$ be the number of paths from node $s$ to $d$, and $K$ the number of outgoing links of $s$ (equivalently, the degree of $s$). Suppose each intermediate node $j \in V - \{s, d\}$ has fixed network coding coefficients and the total network coding vector for the graph $G$ is $\boldsymbol{\beta}_{N\times 1}$. Let $\mathbf{A}_{M\times N} = [a_{ij}]$ be constructed from the *training* symbols sent over $N$ different paths from source to destination over $M$ time slots. Then, for $M \geq K$, there exists a matrix $\mathbf{A}$ and vector $\boldsymbol{\beta}$ such that the following conditions hold:

1 For each subgraph $G_l(V, E_l)$, $\mathbf{A}\boldsymbol{\beta} \neq \mathbf{A}\boldsymbol{\beta}(G_l)$
2 For each pair of subgraphs $G_{l_1}(V, E_{l_1})$ and $G_{l_2}(V, E_{l_2})$, $\mathbf{A}\boldsymbol{\beta}(G_{l_1}) \neq \mathbf{A}\boldsymbol{\beta}(G_{l_2})$

*Proof:* See Appendix. □

The theorem provides *sufficient* conditions that guarantees identifiability of any logical topology $G(V, E)$ using network coding under certain conditions on the training sequence and total LNC vector. Under these sufficient conditions, the congestion states corresponding to different single link failures results in different symbols received at destination. Therefore, by having a simple lookup table, it is possible to identify the congested link in any logical network.

Corollary *3:* Given an acyclic, directed and connected graph $G(V, E)$ with source node $s$ and destination node $d$ where $s$ has $K$ outgoing links and the matrix $\mathbf{A}_{M\times N}$ of training symbols as above; then rank$(\mathbf{A}) = K$.

Theorem 2 talks about possibility of using network coding in the application of link monitoring when number of time slots ($M$) is greater or equal to number of outgoing links of the source ($K$). However, by increasing the number of bits assigned to network coding coefficients, it is possible to locate a failure link using fewer number of time slots.

***Theorem** 4:* Assume an acyclic, directed and connected graph $G(V; E)$ with source-destination pair $s \in V$ and $d \in V$ as before. Let the $K$ outgoing links of $s$ be represented by $e_1, e_2, \ldots e_K$. Let $N_i$ be total number of paths from $s$ to $d$ that starts with $e_i$, i.e., $\sum_{i=1}^{K} N_i = N$. Assume $q$ bits per symbol are used in network coding and $M$ is number of time slots used to send training sequence. Further, assume $Z = \{1, 2, \ldots, K\}$

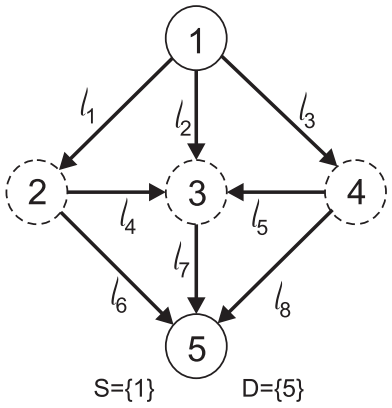



Fig. 7. A network with 5 nodes and five paths, N=5, paths from source to destination. Source has K=3 outgoing links. It is possible to locate a failure link in this network by sending training sequence in 2 time slots if q, number of bits in NC coefficients, are greater than 3.

and $\mathcal{Z}_M$ is the collection of all partitions of $Z$ with size $M$;

$$\mathcal{Z}_M = \{\, \{H_1, H_2, ..., H_M\} | \cup_{i=1}^{M} H_i = Z\}$$

Then $G(V; E)$ is identifiable using NC in field $\mathbb{F}_{2^q}$ if $q$ satisfies the following inequality:

$$q \geq \min_{\{H_i, i=1,...,M\} \in \mathcal{Z}_M} \max_i \sum_{j \in H_i} N_j \tag{20}$$

*Proof*: See Appendix.

Theorem 4 provides a tradeoff between number of time slots for training sequence (speed of the method) and size of network coding coefficient (complexity) to make a network $G(V, E)$ identifiable. In other words, by increasing complexity of the method (increasing $q$) it is possible to save some time slots (decrasing $M$). The following example further illustrates the above theorem.

**Example 2:** Consider the topology in Figure 7 that consists of 4 paths between the source and destination; i.e. $N = 4$. Source has 3 outgoing links, $K = 3$, and using the definition of $N_i$ we have, $N_1 = 2, N_2 = 1, N_3 = 1$. Suppose we want to locate a failure link by sending symbols from source to destination in 2 time slots, i.e. $M = 2$. Theorem 4 helps us find the minimum number of bits $q$, that guarantees identifiability in Figure 7.

Let $Z = \{1, 2, 3\}$. Since $M = 2$, we enumerate all the 2-partitions of $Z$ as given below:

$$\mathcal{Z}_2 = \{\, \{\, \{1\}, \{2, 3\}\}, \{\, \{2\}, \{1, 3\}\}, \{\, \{3\}, \{1, 2\}\}\, \} \tag{21}$$

The outer maximization in Theorem 4 is over $\{N_1 = 2, (N_2 + N_3) = 3\}$ which is 3; our network is found to be identifiable with $M = 2$ and $q \geq 3$.

□

As mentioned before, Theorem 4 implies that increasing the number of time slots used for identifiability of a network $G(V, E)$ decreases number of bits per symbol for NC coefficients. If number of bits of network coding coefficients are large enough, it is even possible to locate a congested link in one time slot. The following two corollaries states these observations more precisely:

Corollary *5:* Suppose $G(V, E)$ is identifiable using network coding with $q_1$ bits per symbol and training sequence in $M_1$ time slots. Then $G$ is identifiable in $M_2 > M_1$ time slots using NC values having $q_2$ bits per symbol where $q_2 \leq q_1$.

Corollary *6:* It is possible to locate a congested link in network $G(V, E)$ in one time slot, if $q$ number of bits assigned to LNC is greater than or equal to total number of paths between source and destination; i.e. $q \geq N$.

In all the results thus far, we have assumed that the network coding coefficients are fixed and chosen so as to satisfy the identifiability conditions. What if the nodes choose the LNC coefficients randomly? In that case, identifiability can be described as a random event. The following theorem provides a lower bound for probability of identifiability of a random graph in such a scenario.

***Theorem** 7:* Let graph $G(V, E)$ be a acyclic, directed and connected graph with two nodes $s \in V$ and $d \in V$ as source and destination respectively. If each intermediate node $j \in V - \{s, d\}$ choose their NC coefficients uniformly from the elements of $\mathbb{F}_{2^q}$, then the probability that all links in $G(V, E)$ are identifiable is bounded from below by $1 - |E|(|E| + 1)(\frac{1}{2^q})^M$

*Proof*: See Appendix.

### *D. Multi-source, Multi-destination Networks*

In previous sections, we established a novel linear network coding based approach which guarantees identifiability of a logical network but using probes between only a single source-destination pair. It may be possible to send probes between an arbitrary number of sources and destinations; the identifiability of a network in such scenarios is discussed next.

For a logical network $G(V, E)$ with sets $S \subset V$ and $D \subset V$ of sources and destinations, respectively, one approach to locate a congested link inside $G$ is to pick any source-destination pair $(s, d) \in S \times D$, and look at the identifiability of the corresponding subgraph. Denote the resulting subgraph by $G_{(s,d)}$. Clearly, If all $G_{(s,d)}$, $(s, d) \in S \times D$, are identifiable then $G$ is identifiable. However the inverse of this statement is not true. Surprisingly, it is even possible that none of $G_{(s,d)}$ are identifiable but still $G(V, E)$ is identifiable. In practice, many sub-graphs are often individually un-identifiable, and hence this approach is not very useful.

As an example, consider the network in Figure 8. In this graph, there are two sources, $S_1$ and $S_2$, and one destination $D$. Therefore there are two subgraphs, $G_{(S_1,D)}$ and $G_{(S_2,D)}$ which are depicted in Figure 9. Both $G_{(S_1,D)}$ and $G_{(S_2,D)}$ have intermediate nodes with degree two which make them individually unidentifiable. However, note that we have access to both $S_1$ and $S_2$ simultaneously and consequently we may consider the pair as a *supper node* as presented in Figure 10 . Hence, a multi-source multi-destination network can be studied as an equivalent single source, single destination network by substituting set of source and destinations with a single super node source and destination, respectively, represented by the new graph $G'(V, E)$. It follows that if $G'$ is identifiable (unidentifiable), then so is $G$ because every edge in G has 1:1 correspondence with an edge in $G'$. Therefore, results in

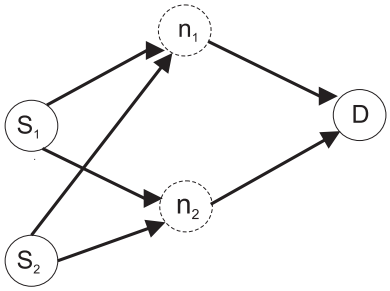



Fig. 8. A two source one destination network

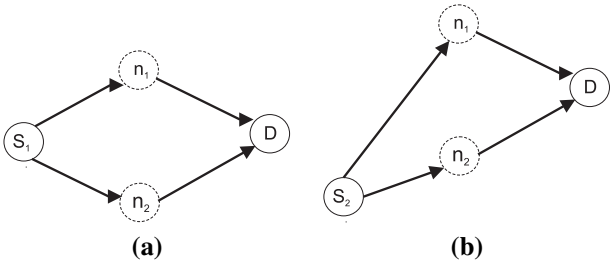



Fig. 9. One way to identify a multi-source multi-destination network is to look at each pair of source-destination combination and identify their corresponding subgraph (a)$G_{(S_1,D)}$ for source $S_1$ and destination $D$ (b)$G_{(S_2,D)}$ for source $S_2$ and destination $D$

previous section may be extended to a multi-source multi-destination network $G(V, E)$ in this way.

## III. Networks with Finite Delays

Given an acyclic network with delay, we can either operate the network in a continuous mode (where information is continuously injected into the network) or with burst mode. In the former, the same injected information can take differet routes causing different delays through the network. Koetter et al. in [14] treat this case by using memory at the receiver node(s). In burst-oriented mode, each node transmits information on an outgoing link only if an input has been observed on all incoming links or a timer times out. This approach is taken by Li et al. [15], and leads to a situation where a network with delay can be thought of being equivalent to a delay-free network and the results in previous subsection therefore apply. We use this method jointly with the buffering model proposed in [17], where an index called generation number is added to header of packets to distinguish between packets sent in successive time slots.

Let $\tau_d(l)$ be the delay of link $l$ when it is not congested. In that case, a packet traversing over path $P^i(d)$, defined

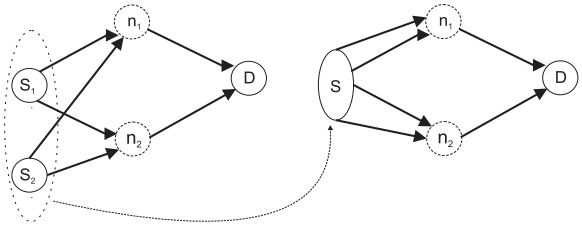



Fig. 10. In multi-source multi-destination network, sources $S$ can be though of a super node. Similarly set of destinations $D$ can be seen as a super node.

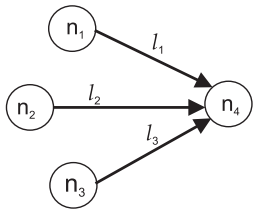



Fig. 11. waiting time of node $n_4$ is: $\tau_w(n_4) = \max_{i=1,2,3}\{\tau_w(n_i) + \tau_d(l_i)\}$.

in Section II-B, encounters net delay of $\sum_{l\in P^i(d)} \tau_d(l)$. We assume that the processing time in each node is negligible. We define $P^{max}(v)$ as the longest path between source and node $v \in V$:

$$P^{max}(v) \in \mathcal{P}(v) \text{ and } |P^{max}(v)| \geq |P^j(v)| \qquad (22)$$

Let $\tau_w(v)$ be the wait time of node $v$, *i.e.* the amount of time node $v$ has to wait before combining the received packets. In the other words, at the start of each time slot (every $T$ seconds), node $v \in V - \{s\}$ waits for $\tau_w(v)$, and then linearly combines all the received packets in that duration and broadcasts the result on all of its outgoing links. Packets received after $\tau_w(v)$ and before the start of the next time slot are dropped.

The waiting time of each node linearly depends on the following two parameters: 1- waiting time of its neighbors from which it receives packets, and 2- delays of its incoming links. This dependency can be written as below(see Figure 11):

$$\tau_w(v) = \max_{\{l\in E: v=d(l)\}} \{\tau_d(l) + \tau_w(o(l))\} \qquad (23)$$

where $o(l)$ and $d(l)$ represent origin and destination of link $l \in E$, respectively.

If the delay caused by congestion (which is ideally defined to be infinite) is greater than waiting time at the destination, the above buffering renders the delay network equivalent to a delay-free network. Note that by definition of waiting time at each node, if the congestion delay exceeds the waiting time at destination, it is greater than the waiting time of each interior node.

## IV. Simulation results

In this Section we describe our linear network coding simulator constructed within OPNET™14.5 aided by Matlab 7.1 that is used for finite field calculations necessary for network coding. The simulation results from applying the proposed link failure monitoring scheme to the network in Figure 2 (which is not identifiable by end-to-end measurements) is presented. Finally, we apply the method to the network graph for the University of Washington's Electrical Engineering department network of servers and present identifiability results.

### A. Simulation Environment

Ours is the first known implementation of Network Coding (NC) within OPNET. OPNET was selected because of its

wide acceptance as a network modeling tool within both the academic and commercial communities [23]. To implement network coding in OPNET, the primary need was a method for queueing packets within each router to account for variable transmission and reception times, and to ensure that packets were combined correctly. The routers model are modified to be able to distinguish between non-network coded and network coded packets through the use of a *flag bit* within the UDP header of received packets; thereafter, the intercepted NC (network-coded) packets are sent for separate processing while the non-NC packets are processed normally. In addition, because of the inherent necessity of network coding to operate on unidirectional links, each interface within a router model is designated as a SEND or RECEIVE interface only for the network coded packets while operating regularly with non-network coded packets (Figure 13). For interfaces which only SEND NC packets, any received NC packets were discarded. Conversely, the RECEIVE interfaces intercepted and processed the NC packets.

Network coding - by definition- is intended for implementation in Layer 3, over IP frames. For the purposes of this simulation, it was not necessary to implement those extensive features; rather it was convenient to develop an implementation which takes advantage of pre-existing OPNET functionality. Therefore, we employ network coding at transport layer (instead of IP layer) largely for convenience - it readily allows adding *hidden* data within the TCP/UDP frame in OPNET, which is invisible to the end-user and to the simulation statistics [24]. As mentioned in section II, any binary vector of length $q$, can be interpreted as an element in $\mathbb{F}_{2^q}$, the finite field with $2^q$ elements. In our network coding implementation, we assign a $q$-bit field called *LNC field* within the TCP/UDP header, for linear network coding. Only the contents in LNC field is used for network coding operation. In addition, a 1-bit flag within TCP/UDP header determines if the packet is a network coding packet. Once a router receives a packet, it identifies the packet type by looking at the 1-bit flag embedded in TCP/UDP header. If the packet is a network-coded packet, the data in LNC field is extracted and queued at a buffer for a predetermined amount of time (refer to Section III) after which they are combined and the router clears the buffer. The result of linear combining is written in LNC field of the outgoing packet which is forwarded on all of the outgoing links via unidirectional broadcast. Note that this network-coding approach is different from the bidirectional physical layer broadcast as described in Section II.

To simulate congestion within a link, the identified NC packets were either dropped or significantly delayed. Our implementation thus only affects the NC packets, and was transparent to all non-NC coded packets. Since NC involves finite field calculations which is not explicitly supported in OPNET, we implemented the Matlab API within OPNET [25] to use the Galois Field functions within Matlab's Communications Toolbox.

### *B. Validation*

Figure 14 demonstrates a test scenario that was run within OPNET to validate our findings - this topology (same as in Figure 2) is not identifiable using end-to-end probe monitoring. The arrows indicate the network coding graph which overlays the 100 Mbps Full Duplex connection between routers. Linear NC coefficient assigned to each link - $\gamma_l$ in Eq. (3)- are shown in a box next to the link. Table III presents received values at destination for different link congestion states when the training sequence given by elements of matrix $\mathbf{A}$ is used; it is the lookup table used by destination (node 5) to identify any congested link in the network.

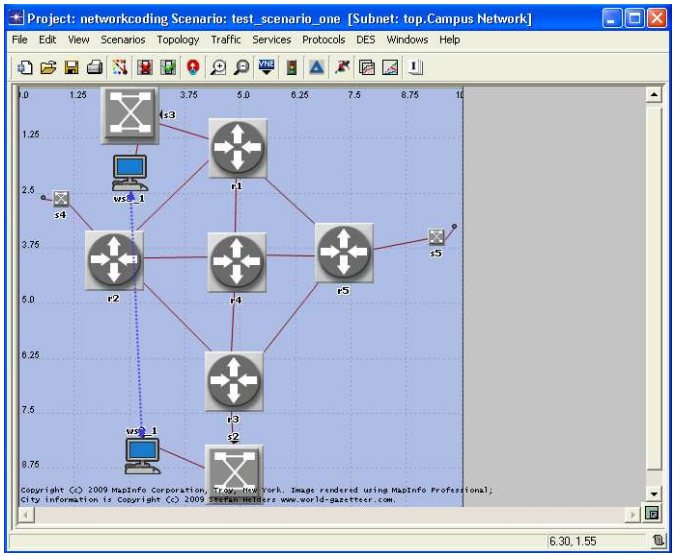



Fig. 12. OPNET implementation of proposed link failure monitoring for network in Figure 2 which is not identifiable by end-to-end measurements

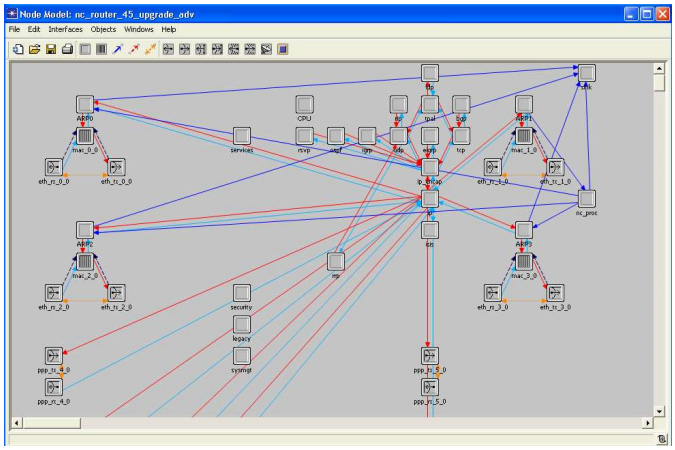



Fig. 13. An example of OPNET Network Coding router node model. Interface 0, 2, and 3 are SEND interfaces, while Interface 1 is a RECEIVE interface

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 & 2 & 2 \\ 1 & 1 & 2 & 1 & 1 \\ 1 & 1 & 3 & 3 & 3 \end{bmatrix} \tag{24}$$

Further tests of our approach was conducted on a larger graph resembling that of the University of Washington's Electrical Engineering network shown in Figure 15. Thirteen subnets (represented by the numbers 1-13) are all connected through Full Duplex Ethernet links to backbone routers (represented as A, B, C and D), which then connect to the rest of

| Congested link | None | $l_1$ | $l_2$ | $l_3$ | $l_4$ | $l_5$ | $l_6$ | $l_7$ | $l_8$ |
|---|---|---|---|---|---|---|---|---|---|
| 1st time slot | 2 | 0 | 0 | 0 | 3 | 3 | 1 | 0 | 1 |
| 2nd time slot | 0 | 2 | 3 | 1 | 1 | 3 | 3 | 1 | 2 |
| 3rd time slot | 0 | 2 | 1 | 3 | 1 | 2 | 3 | 2 | 1 |

TABLE III
SYMBOLS RECEIVED AT DESTINATION FOR TOPOLOGY IN FIGURE 14 WITH TRAINING SEQUENCE GIVEN IN EQ. (24)

| + | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 0 | 1 | 2 | 3 |
| 1 | 1 | 0 | 3 | 2 |
| 2 | 2 | 3 | 0 | 1 |
| 3 | 3 | 2 | 1 | 0 |

| * | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 2 | 3 |
| 2 | 0 | 2 | 3 | 1 |
| 3 | 0 | 3 | 1 | 2 |

TABLE IV
ADDITION AND MULTIPLICATION IN $\mathbb{F}_{2^2}$. IRREDUCIBLE POLYNOMIAL IS $R(x) = x^2 + x + 1$

campus. Figure 15-(a) depicts the network coding coefficients of each node and the training sequence used is given below.

$$\mathbf{A} = \begin{bmatrix} 5 & 5 & 2 & 5 & 2 & 1 & 5 & 3 & 5 & 4 & 3 & 6 & 7 \\ 4 & 7 & 4 & 1 & 2 & 7 & 7 & 1 & 2 & 2 & 6 & 4 & 2 \end{bmatrix}$$

Figure 15-(b) shows the received symbols at destination (node $A$) for different link congestion scenarios. Symbols next to each link represent values at the destination when that link is congested. Destination node $A$ uses these symbols to uniquely locate any congested link in the network; *i.e.* UW EE network is seen to be identifiable under proposed network coding monitoring.

## V. CONCLUSION

This work has presented a novel approach to link status monitoring based on a deterministic approach that exploits linear network coding at the internal nodes in a network. The key problem of identifiability for such approaches was highlighted and various insights provided regarding this concept. New sufficient conditions were derived for successfully identifying a congested link in any *logical network*, and trade-offs between length of training slots and size of the network coding alphabet established. Finally, the method is verified by implementation within OPNET to confirm the validity of the claims.

## APPENDIX A
## FINITE FIELD DEFINITIONS

For completeness, we provide a summary of the basic facts concerning finite (Galois) fields, over which the network code is defined. A field with $2^n$ elements, denoted by $\mathbb{F}_{2^n}$, is a finite field (the extension finite field) with elements represented as polynomials of degree strictly less than $n$ over $\mathbb{F}_2$; i.e. elements of $\mathbb{F}_{2^n}$ have the form $a_{n-1}x^{n-1}+a_{n-2}x^{n-2}+...+a_0$ where $a_i \in \mathbb{F}_2$, the binary field. Thus the four elements of $\mathbb{F}_{2^2}$ can be represented as $\{0, 1, x, x+1\}$. However, it is conventional to express the elements of $\mathbb{F}_{2^n}$ by an equivalent $n$ bit binary representation. For example, elements in $\mathbb{F}_{2^2}$ can be represented as $\{00, 01, 10, 11\}$ or $\{0, 1, 2, 3\}$.

By definition, a finite field supports two operations; for $\mathbb{F}_2$, these are modulo-2 addition (XOR) and multiplication (AND). These operations (Addition and multiplication) in extension fields $\mathbb{F}_{2^n}$ is best understood in terms of polynomial addition and multiplication which are performed modulo $R(x)$ where $R(x)$ is an *irreducible polynomial* of degree $n$ over $\mathbb{F}_2$. The addition of two polynomials $P(x)$ and $Q(x)$ is done as usual; multiplication may be done as follows: compute $W(x) = P(x).Q(x)$ as usual, then compute the remainder modulo $R(x)$, using polynomial long division.

| + | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 2 | 3 | 0 | 1 | 2 | 3 |
| 1 | 1 | 0 | 3 | 2 | 5 | 4 | 7 | 6 |
| 2 | 2 | 3 | 0 | 1 | 6 | 7 | 4 | 5 |
| 3 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 |
| 4 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 |
| 5 | 5 | 4 | 7 | 6 | 1 | 0 | 3 | 2 |
| 6 | 6 | 7 | 4 | 5 | 2 | 3 | 0 | 1 |
| 7 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

| * | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 2 | 0 | 2 | 4 | 6 | 3 | 1 | 7 | 5 |
| 3 | 0 | 3 | 6 | 5 | 7 | 4 | 1 | 2 |
| 4 | 0 | 4 | 3 | 7 | 6 | 2 | 5 | 1 |
| 5 | 0 | 5 | 1 | 4 | 2 | 7 | 3 | 6 |
| 6 | 0 | 6 | 7 | 1 | 5 | 3 | 2 | 4 |
| 7 | 0 | 7 | 5 | 2 | 1 | 6 | 4 | 3 |

TABLE V
ADDITION AND MULTIPLICATION IN $\mathbb{F}_{2^3}$. IRREDUCIBLE POLYNOMIAL IS $R(x) = x^3 + x + 1$

For example, consider two elements in $\mathbb{F}_{2^8}$, which has an irreducible polynomial as $x^8+x^4+x^3+x+1$ - $(001010011)$ and $(011001010)$. They can also be represented in polynomial form as $x^6 + x^4 + x + 1$ and $x^7 + x^6 + x^3 + x$ respectively. The sum of these two symbols results in $x^7 + x^4 + x^3 + 1$ by rules of modulo-2. Multiplication of these two symbols is calculated as in Figure 16.

It is convenient to have a lookup table for the two operations of summation and multiplication for an extension field (especially when implementation is in concern). Table IV and V provide these for $\mathbb{F}_{2^2}$ and $\mathbb{F}_{2^3}$, respectively.

## APPENDIX B
## PROOF OF THEOREMS

*Proof of theorem 2*: Let $e_1, e_2, ..., e_K$ be $K$ outgoing links of source $s$. Recall that $\mathcal{P}_{e_i}$ is defined as set of paths from source to destination that have $e_i$ in common and $\mathcal{P}_{e_1}(d), ..., \mathcal{P}_{e_K}(d)$ is a K-partition of $\mathcal{P}(d)$. Let $P^k_{e_i}$ represent the $k$-th element of $\mathcal{P}_{e_i}(d)$; i.e. $P^k_{e_i}$ is the $k$-th path among all paths from $s$ to $d$ passing through $e_i$. Let $N_i = |\mathcal{P}_{e_i}|$, total number of paths from source to destination starting at $e_i$. Consider the

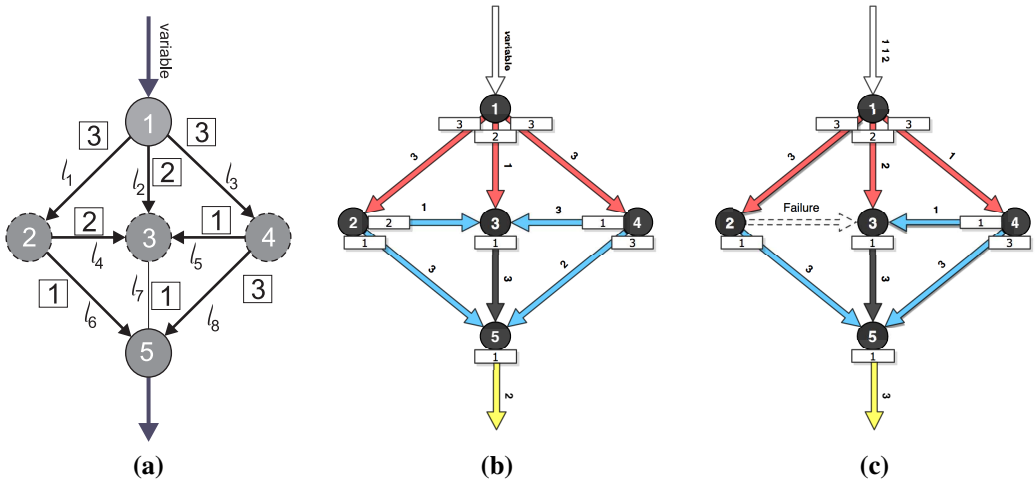



Fig. 14. Topology given in Figure 2 which is not identifiable using end-to-end measuring (a)Network Coefficient values in $\mathbb{F}_{2^2}$ ($\gamma_l$) (b)Snapshot of network coding simulation (c)Snapshot of simulation when $l_3$ is congested.

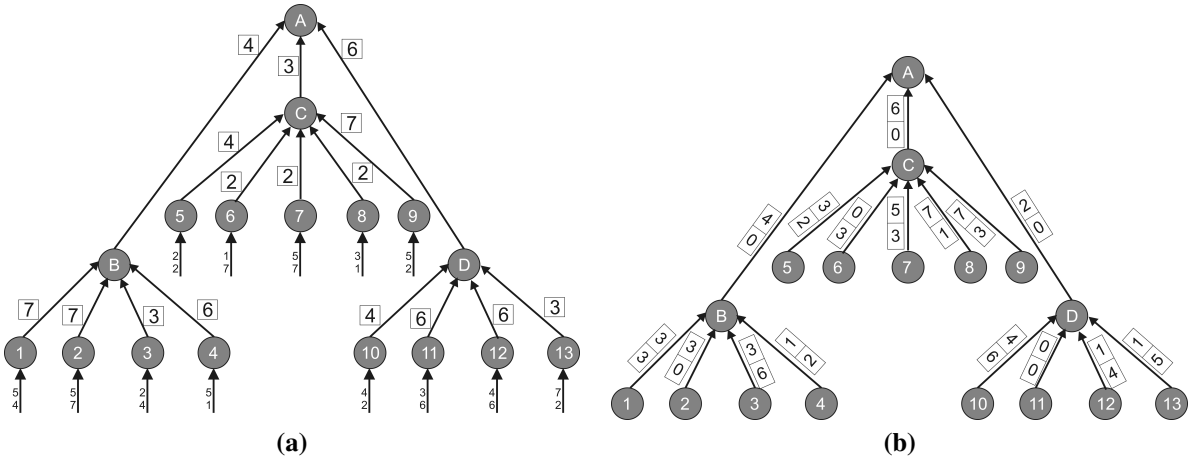



Fig. 15. (a) UW Electrical Engineering network topology, network coding coefficients in $\mathbb{F}_{2^3}$ ($\gamma_l$) and training sequence (b) numbers next to each link shows received symbols at destination when the link is congested.

$$(x^6+x^4+x+1)(x^7+x^6+x^3+x) \text{ modulo } x^8+x^4+x^3+x+1 = \\ x^{13}+x^{12}+x^{11}+x^{10}+x^9+x^8+x^6+x^5+x^4+x^3+x^2+x \text{ modulo } x^8+x^4+x^3+x+1 = 1 \quad (25)$$

Fig. 16. An example of multiplication of two elements in $\mathbb{F}_{2^8}$ with irreducible polynomial $x^8+x^4+x^3+x+1$

following definitions:

$$\beta_{k,e_i} = \prod_{j\in P^k_{e_i}} \gamma_j \quad (26)$$

$$\boldsymbol{\beta}_{e_i} = [\beta_{k,e_i}]_{k=1}^{N_i} \quad (27)$$

$$(\boldsymbol{\beta}(G))^T = [\beta_{e_i}]_{i=1}^{K} \quad (28)$$

$\beta_{k,e_i}$ is the product of linear network coding coefficients of all links on $k$-th path in $\mathcal{P}_{e_i}$. Vector $\boldsymbol{\beta}(G)$ defined above is a rearranged version of $\boldsymbol{\beta}(G)$ in (11). Let $\beta_{k,e_i}$ be elements in field $\mathbb{F}_{2^q}$ with the following property for each $i=1,\ldots,K$:

$$\sum_{k=1}^{N_i} \zeta_k \beta_{k,e_i} = 0 \Leftrightarrow \zeta_k = 0,\ \forall k \quad (29)$$

We shall show that $\boldsymbol{\beta}(G)$ with above property satisfies both conditions in Theorem 2.

Since in the extended binary field $\mathbb{F}_{2^q}$ addition and subtraction are equivalent, $\mathbf{A}\boldsymbol{\beta}(G) \neq \mathbf{A}\boldsymbol{\beta}(G_l)$ implies $\mathbf{A}(\ \boldsymbol{\beta}(G) + \boldsymbol{\beta}(G_l)\ ) \neq 0$. Define $\tilde{\boldsymbol{\beta}}(G_l) = \boldsymbol{\beta}(G) + \boldsymbol{\beta}(G_l)$. According to (13), $\tilde{\boldsymbol{\beta}}(G_l)$ can be written as

$$\tilde{\beta}_{k,e_i} = \begin{cases} \beta_{k,e_i} & if\ \ l \in P^k_{e_i}(d) \\ 0 & o.w. \end{cases} \quad (30)$$

$$\tilde{\boldsymbol{\beta}}_{e_i} = [\tilde{\beta}_{k,e_i}]_{k=1}^{N_i} \quad (31)$$

$$\tilde{\boldsymbol{\beta}}(G_l) = [\tilde{\boldsymbol{\beta}}_{e_i}]_{i=1}^{K} \quad (32)$$

where $\tilde{\boldsymbol{\beta}}(G_l)$ contains network coding coefficient of the paths which go through link $l$ and is zero otherwise. According to broadcast nature of network coding, there is a path from

source to destination which goes through link $l$. Therefore, there is at least one $\beta_{k,e_i}$ which is not zero, say $\beta_{k^*,e_{i^*}} \neq 0$. On the other hand, $\forall i \in \{1,2,...,K\}$ $\beta_{k,e_i}, k = 1,2,...,N_i$ are selected as in (29). Now, define matrix $\mathbf{A} = [a_{i,j}]$ as:

$$a_{i,j} = \begin{cases} 1 & \sum_{k=1}^{j-1} N_k < i \leq \sum_{k=1}^{j} N_k \\ 0 & o.w. \end{cases} \quad (33)$$

corresponding to the following - in time slot $i$, source sends symbol $\mathbf{1} \in \mathbb{F}_{2^q}$ over $i^{th}$ outgoing link and zero over others. An example of matrix $\mathbf{A}$ for network in Figure 7 is given below. As it is explained in Example 2, for that network we have $N = 5$, $K = 3$ and $N_1 = 2$, $N_2 = 1$, $N_3 = 2$.

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Now let look at the $i^*$ entry of vector $\mathbf{A}\ \tilde{\boldsymbol{\beta}}(G_l)$. Since in time slot $i^*$ source sends $\mathbf{1}$ over $e_{i^*}$ and zero otherwise, the $i^*$ entry of $\mathbf{A}\ \tilde{\boldsymbol{\beta}}(G_l)$ is calculated as follows:

$$\sum_{k=1}^{N_{i^*}} \tilde{\beta}_{k,e_{i^*}} \quad (34)$$

which can be rewritten as

$$\sum_{k=1}^{N_{i^*}} \zeta_k \beta_{k,e_{i^*}} \quad (35)$$

where $\zeta_k$ is one if $\tilde{\beta}_{k,e_{i^*}} = \beta_{k,e_{i^*}}$ and is zero otherwise. Since $\tilde{\beta}_{k^*,e_{i^*}} \neq 0$, or equivalently $\tilde{\beta}_{k^*,e_{i^*}} = \beta_{k^*,e_{i^*}}$ (according to Eq. (32), $\forall i,k$ $\tilde{\beta}_{k,e_i}$ is either zero or equal $\beta_{k,e_i}$), in Eq. (35), $\zeta_{k^*} \neq 0$. For that reason, Eq. (29) guarantees that Eq. (35) is not zero which means $i^*$ entry of $\mathbf{A}\ \tilde{\boldsymbol{\beta}}(G_l)$ is not zero and consequently $\mathbf{A}\ \tilde{\boldsymbol{\beta}}(G_l) \neq 0$

Now we prove matrix $\mathbf{A}$ defined in (33) and $\boldsymbol{\beta}$ defined in (29) also satisfy the second condition of the theorem. With the same argument, it is sufficient to show that $\mathbf{A}(\boldsymbol{\beta}(G_{l_1}) + \boldsymbol{\beta}(G_{l_2})) \neq 0$. Let $\tilde{\boldsymbol{\beta}}(G_{l_1}, G_{l_2}) = \boldsymbol{\beta}(G_{l_1}) + \boldsymbol{\beta}(G_{l_2})$, using definition of $\boldsymbol{\beta}(G_l)$ in (13) it is easily seen that $\tilde{\boldsymbol{\beta}}(G_{l_1}, G_{l_2})$ has the following structure:

$$\tilde{\beta}_{k,e_i} = \begin{cases} 0 & \text{if } (l_1 \notin P^i(d) \text{ and } l_2 \notin P^i(d)) \text{ or} \\ & (l_1 \in P^i(d) \text{ and } l_2 \in P^i(d)) \\ \beta_{k,e_i}(G) & o.w. \end{cases} \quad (36)$$

$$\tilde{\boldsymbol{\beta}}_{\boldsymbol{e_i}} = [\tilde{\beta}_{k,e_i}]_{k=1}^{N_i} \quad (37)$$

$$\tilde{\boldsymbol{\beta}}(G_{l_1}, G_{l_2}) = [\tilde{\boldsymbol{\beta}}_{\boldsymbol{e_i}}]_{i=1}^{i=K} \quad (38)$$

Basically, $\tilde{\boldsymbol{\beta}}(G_{l_1}, G_{l_2})$ is non-zero over paths which go through one of the links but not both. On the other hand, we have assumed that the network is logical which means there is a path which goes through one of $l_1$ or $l_2$ and not both of them. Hence, $\tilde{\beta}(G_{l_1}, G_{l_2})$ has at least one non-zero element, say $\tilde{\beta}_{k^*,e_{i^*}} \neq 0$. By definition of $\tilde{\beta}_{k,e_i}$, if $\tilde{\beta}_{k^*,e_{i^*}}$ is not zero it is equal $\beta_{k,e_i}$. By the same argument as before the $i^*$ entry of matrix production $\mathbf{A}\tilde{\boldsymbol{\beta}}(G_{l_1}, G_{l_2})$ is not zero and consequently $\mathbf{A}\tilde{\boldsymbol{\beta}}(G_{l_1}, G_{l_2}) \neq 0$ which implies $\mathbf{A}\boldsymbol{\beta}(G_{l_1}) \neq \mathbf{A}\boldsymbol{\beta}(G_{l_2})$.

■

*Proof of Theorem 4*: Let $H = \{H_1, H_2, ..., H_M\}$ be a $M$-partition of $Z = \{1,2,\ldots,K\}$; i.e. $Z = \cup_{i=1}^{M} H_i$ and $H_i \cap H_j = \phi, i \neq j$. Suppose for each $j$, $1 \leq j \leq M$, NC coefficients $\beta_{k,e_i}$, $k = 1,...,N_i$ and $i \in H_j$, have following property:

$$\sum_{i \in H_j} \sum_{k=1}^{N_i} \zeta_{i,k} \beta_{k,e_i} = 0 \Leftrightarrow \zeta_{i,k=0},\ i \in H_j, k = 1,2,...,N_i \quad (39)$$

Now, define matrix $\mathbf{A} = [a_{i,j}]$ as the following; In time slot $j$, source sends symbol $\mathbf{1} \in \mathbb{F}_{2^q}$ over $i^{th}$ outgoing link where $i \in H_j$ and zero over others. By same argument as given in proof of theorem 2, matrix $\mathbf{A}$ satisfies the following two inequalities:

- $\mathbf{A}\boldsymbol{\beta}(G_l) \neq \mathbf{A}\boldsymbol{\beta}, \forall l_1 \in E$
- $\mathbf{A}\boldsymbol{\beta}(G_{l_1}) \neq \mathbf{A}\boldsymbol{\beta}(G_{l_2}), \forall l_1, l_2 \in E$

which means $G(V,E)$ is identifiable using $\mathbf{A}$ as training sequence.

In finite filed $\mathbb{F}_{2^q}$ there are $q$ numbers $\varphi_k$, $k = 1,\ldots,q$ such that [26]

$$\sum_{k=1}^{q} \zeta_k \varphi_k = 0 \Leftrightarrow \zeta_k = 0, \forall k \quad (40)$$

This property and the condition given in Eq. (39) for $\beta_{k,e_i}$ contribute to the fact that, for a given $H$, $G(V,E)$ is identifiable if $q \geq \sum_{i \in H_j} N_i$ for all $1 \leq j \leq K$ or equivalently $q \geq \max_j \sum_{i \in H_j} N_i$. In the other words, for a given partition of $Z$, $G(V,E)$ is identifiable using $M$ time slots and $q$ bits for NC coefficients if $q \geq \max_j \sum_{i \in H_j} N_i$. Therefore, $G(V,E)$ is identifiable if $q$ is the smallest number amount all partitions of $Z$; i.e. if $q$ satisfies the following min-max inequality:

$$q \geq \min_{\{H_i, i=1,\ldots,M\} \in \mathcal{Z}_M} \max_i \sum_{j \in H_i} N_j \quad (41)$$

■

*Proof of Theorem 7*:

We prove the theorem by using Schwartz-Zippel Lemma[27], given as follows:

**Schwartz-Zippel Lemma**: If f, g are two different m-variate polynomial of degree at most d over $\mathbb{F}$, then $P(f = g) \leq \frac{d}{|\mathbb{F}|}$.

There is a status ambiguity between link $l_1 \in E$ and $l_2 \in E$ if received symbols at the destination in case of $l_1$ congestion be the same as received symbols when $l_2$ is congested, $\mathbf{y}^{l_1} = \mathbf{y}^{l_2}$. Using above lemma this can happen with the following probability:

$$\begin{aligned} & P(\mathbf{y}^{l_1} = \mathbf{y}^{l_2}) \quad (42)\\ = \;& P(\mathbf{A}\boldsymbol{\beta}(G_{l_1}) = \mathbf{A}\boldsymbol{\beta}(G_{l_2}))\\ = \;& P(\boldsymbol{\alpha}^T[i]\boldsymbol{\beta}(G_{l_1}) = \boldsymbol{\alpha}^T[i]\boldsymbol{\beta}(G_{l_2}), i = 1, 2..., M)\\ = \;& \prod_{i=1}^{M} P(\boldsymbol{\alpha}^T[i]\boldsymbol{\beta}(G_{l_1}) = \boldsymbol{\alpha}^T[i]\boldsymbol{\beta}(G_{l_2}))\\ \leq \;& \prod_{i=1}^{M} \frac{1}{2^q} = (\frac{1}{2^q})^M \end{aligned}$$

where in the last inequality we use Schwartz-Zippel lemma where $d = 1$ and $|\mathbb{F}_{2^q}| = 2^q$.

link $l_1 \in E$ is not identifiable if there is a link $l_2 \in E \cup \{\phi\}$ ($\{\phi\}$ stands for no congestion in the network) such that there is status ambiguity between them. The probability that $l_1$ is not identifiable is calculated as below:

$$\begin{aligned} P(l_1 \text{ is not identifiable}) &= P(\mathbf{y}^{l_1} = \mathbf{y}^{l_2},\, l_2 \in \tilde{E}) \quad (43)\\ &= \sum_{l_2 \in \tilde{E}} P(\mathbf{y}^{l_1} = \mathbf{y}^{l_2})\\ &\leq (|E| + 1)(\frac{1}{2^q})^M \end{aligned}$$

where $\tilde{E} = E \cup \{\phi\}$.

Graph $G(V, E)$ is identifiable if all links are identifiable. Using above probabilities for identifiability of link $l_1 \in E$, probability of $G$ be identifiable is given as follows:

$$\begin{aligned} P(G \text{ is identifiability}) &= 1 - P(G \text{ is not identifiability}) \quad (44)\\ &= 1 - P(l \text{ is not identifiable},\, l \in E)\\ &= 1 - \sum_{l \in E} P(l \text{ is not identifiable})\\ &\geq 1 - \sum_{l \in E} (|E| + 1)(\frac{1}{2^q})^M\\ &= 1 - |E|(|E| + 1)(\frac{1}{2^q})^M \end{aligned}$$

■

## APPENDIX C
## GLOSSARY

- $\gamma_l$ is the network coding coefficient of the link $l$
- $\mathcal{P}(v)$ collection of all paths from $s$ to $v$
- $P^i(v)$ is the $i$-th element of $\mathcal{P}(v)$ or equivalently $i$-th path between $s$ and $v$
- $N$ is the total number of paths from source to destination
- $K$ number of outgoing links of the source
- $M$ number of time slots in which source sends symbols to destination
- $e_k$ is the $k$-th outgoing link of source
- $\beta_i(G)$ is the product of LNC coefficients of all links lying on the $i$-th path from source to destination in network $G$; i.e. $P^i(d)$
- $\beta_i(G_l)$ is the product of LNC coefficients of all links lying on the $i$-th path from source to destination in network $G_l$, edge deleted subgraph of $G$.
- $\alpha_k[n]$ is the symbol sent by source $s$ over the $k$-th outgoing link, $e_k$, in time slot $n$.
- $y[n]$ received symbols at destination in time slot $n$ in network $G$
- $y^l[n]$ received symbols at destination in time slot $n$ in network $G_l$
- $\boldsymbol{\beta}(G)$ is the total network coding vector of graph $G$. Its $i$-th entry is $\beta_i(G)$.
- $\mathbf{A}_{M\times N}$ is a $M \times N$ matrix whose $n$-th row is the training symbols sent in time slot $n$
- $\mathbf{y}_{M\times 1}$ received symbols at destination in $M$ time slots in network $G$
- $\mathbf{y}_{M\times 1}$ received symbols at destination in $M$ time slots in network $G_l$

## ACKNOWLEDGMENT

This work was supported in part by a grant from NSF under ECCS 0801997.

## REFERENCES

[1] Y. Vardi, "Network tomography: Estimating source-destination traffic intensities from link data," *Journal of the American Statistical Association*, vol. 91, no. 433, 1996.

[2] S. W. Richard, "TCP/IP illustrated," *Addison-Welsey Publishing Company*, 1994.

[3] Y. Bejerano and R. Rastogi, "Robust monitoring of link delays and faults in IP networks," in *Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'03)*, vol. 1, 2003, pp. 134–144.

[4] R. Caceres, N. Duffield, J. Horowitz, and D. Towsley, "Multicast-based inference of network-internal loss characteristics," *IEEE Trans. Information theory*, vol. 45, no. 7, pp. 2462–2480, 1999.

[5] Y. Tsang, M. Coates, and R. Nowak, "Passive network tomography using em algorithms," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing(ICASSP'01)*, vol. 3, 2001.

[6] A. Coates, A. Hero III, R. Nowak, and B. Yu, "Internet tomography," *IEEE Signal processing magazine*, vol. 19, no. 3, pp. 47–65, 2002.

[7] T. Bu, N. Duffield, F. Presti, and D. Towsley, "Network tomography on general topologies," *ACM SIGMETRICS Performance Evaluation Review*, vol. 30, no. 1, pp. 21–30, 2002.

[8] R. Castro, M. Coates, G. Liang, R. Nowak, and B. Yu, "Network tomography: Recent developments," *Statistical Science*, vol. 19, no. 3, pp. 499–517, 2004.

[9] Y. Xia and D. Tse, "Inference of link delay in communication networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 12, pp. 2235–2248, 2006.

[10] M. Gjoka, C. Fragouli, P. Sattari, and A. Markopoulou, "Loss tomography in general topologies with network coding," in *Proc. IEEE Conf. Global Telecommunications(GLOBECOM'07)*, 2007, pp. 381–386.

[11] N. Duffield, "Network tomography of binary network performance characteristics," *IEEE Trans. Information Theory*, vol. 52, no. 12, pp. 5373–5388, 2006.

[12] B. Xi, G. Michailidis, and V. Nair, "Estimating network loss rates using active tomography," *Journal of the American Statistical Association*, vol. 101, no. 476, pp. 1430–1448, 2006.

[13] T. Ho, M. Medard, J. Shi, M. Effros, and D. Karger, "On randomized network coding," in *Proc. Annual Allerton Conf. on Communication Control and Computing*, vol. 41, no. 1, 2003, pp. 11–20.

[14] R. Koetter and M. Medard, "An algebraic approach to network coding," *IEEE/ACM trans. Networking*, vol. 11, no. 5, pp. 782–795, 2003.

[15] S. Li, R. Yeung, and N. Cai, "Linear network coding," *IEEE Trans. Information Theory*, vol. 49, no. 2, pp. 371–381, 2003.

[16] G. Sharma, S. Jaggi, and B. Dey, "Network tomography via network coding," in *Information Theory and Applications Workshop, 2008*, 2008, pp. 151–157.

[17] P. Chou, Y. Wu, and K. Jain, "Practical network coding," in *Proc. the annual allerton Conf. Communication control and computing*, vol. 41, no. 1. The University; 1998, 2003, pp. 40–49.

[18] T. Ho, B. Leong, Y. Chang, Y. Wen, and R. Koetter, "Network monitoring in multicast networks using network coding," in *Proc. Int. Symp. Information Theory (ISIT'05)*, 2005, pp. 1977–1981.
[19] T. Ho, R. Koetter, M. Medard, D. Karger, and M. Effros, "The benefits of coding over routing in a randomized setting," in *Proc. IEEE Int. Symposium on Information Theory*.
[20] T. Ho, M. Medard, R. Koetter, D. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *IEEE Trans. Information Theory*, vol. 52, no. 10, pp. 4413–4430, 2006.
[21] E. Horowitz, "Modular arithmetic and finite field theory: A tutorial," in *Proceedings of the second ACM symposium on Symbolic and algebraic manipulation*, 1971, pp. 188–194.
[22] J. Clark and D. Holton, *A first look at graph theory*. World Scientific Publishing Company, 1991.
[23] D. Cavin, Y. Sasson, and A. Schiper, "On the accuracy of manet simulators," in *Proceedings of the second ACM international workshop on Principles of mobile computing*, 2002, pp. 38–43.
[24] C. Lydick, "Tutorial: Network coding using opnet and matlab," *Thechnical report*, May 2009.
[25] [Online]. Available: www.cse.psu.edu/ venkates/index_files/opnet_tips.html
[26] R. McEliece, *Finite fields for computer scientists and engineers*. Springer, 1987.
[27] T. Ho and D. Lun, *Network Coding: an Introduction*. Cambridge University Press, 2008.

PLACE PHOTO HERE

**Mohammad H. Firooz** Mohammad Hamed Firooz received his B.Sc. degree in Electrical Engineering from the Isfahan University of Technology, Isfahan, Iran, in 2004, his M.Sc. degree in Communication Systems from the University of Tehran, Tehran, Iran, in 2007. As a research assistant, he worked in Electrical and Computer Engineering department of University of Utah in 2007-2008. Since September 2008, Hamed is a Ph.D. candidate at the University of Washington, Seattle. His current research interest is cooperative networks, network coding, and network management for ad-hoc and sensor networks.

PLACE PHOTO HERE

**Sumit Roy** Sumit Roy received the B. Tech. degree from the Indian Institute of Technology (Kanpur) in 1983, and the M. S. and Ph. D. degrees from the University of California (Santa Barbara), all in Electrical Engineering in 1985 and 1988 respectively, as well as an M. A. in Statistics and Applied Probability in 1988. Presently he is Prof. of Electrical Engineering, Univ. of Washington where his research interests include analysis/design of wireless communication and sensor network systems with a current emphasis on wireless LANs (802.11) and wireless MANs (802.16), multi-standard wireless inter-networking and cognitive radio platforms, and sensor networking involving RFID technology. He is also exploring applications and extensions of networking technologies and principles to new domains such as social, energy and biological networks.

He spent 2001-03 on academic leave at Intel Wireless Technology Lab as a Senior Researcher engaged in systems architecture and standards development for ultra-wideband systems (Wireless PANs) and next generation high-speed wireless LANs. His activities for the IEEE Communications Society (ComSoc) includes membership of several technical and conference program committees (most recently, Vice Chair of Technical Program Committee for IEEE WCNC05 conference). He has served as Associate Editor for IEEE Trans. Communications and IEEE Trans. on Wireless Communications and currently serves on the Editorial Board for IEEE Trans. Communications, IEEE Intelligent Transportation Systems and Wiley J. Wireless Communications and Mobile Computing. He was elevated to IEEE Fellow by Communications Society in 2007 for his "contributions to multi-user communications theory and cross-layer design of wireless networking standards."

PLACE PHOTO HERE

**Linda Bai** Linda Bai is a Ph.D. candidate at the Electrical Engineering Dept., Univ. of Washington. She received her B.E. degree from the Electronic Engineering Dept., Tsinghua Univ., Beijing, China, in 2008. Her current research focuses on network tomography.