Network Coding Tomography for Network Failures

Hongyi Yao[1], Sidharth Jaggi[2], and Minghua Chen[2]
[1] Tsinghua University [2] The Chinese University of Hong Kong

***Abstract*—Network Tomography (or *network monitoring*) uses end-to-end measurements to characterize the network, such as estimating the network topology and localizing random or adversarial glitches. Under the setting that all nodes in the network perform random linear network coding, this work provides a comprehensive study of passive network tomography in the presence of network failures, in particular adversarial/random errors and adversarial/random erasures. Our results are categorized into two classes: 1.** *Topology Estimation.* **In the presence of both adversarial/random failures, we prove it is both necessary and sufficient for all nodes in the network to share *common randomness*, *i.e.,* the receiver knows the random codebooks of other nodes. Without such common randomness, we prove that in the presence of adversarial or random failures it is either theoretically impossible or computationally intractable to estimate topology accurately. With common randomness, we present the first set of algorithms for characterizing topology exactly. Our algorithms for topology estimation in the presence of random errors/erasures have polynomial-time complexity. 2.** *Failure Localization.* **Given the topology, we present the first polynomial time algorithms to localize random errors and adversarial erasures. For the problem of locating adversarial errors, we prove that it is intractable.**

I. Introduction

The goal of *network tomography* (or *network monitoring*) is to use *end-to-end* measurements across a network to infer the network topology, estimate link statistics such as loss rate, and locate network failures [1].

Random linear network coding [2][3] (*i.e.*, each node independently and randomly mixes its receiving packets and outputs their linear combinations) is proved to attain optimal multicast throughput [4]. In addition to its desirable distributed nature, such scheme also has low complexity [2][3].

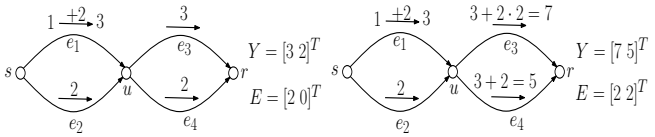
The main observation driving this work is that the linear transforms arising from random linear network coding have specific relationships with the network structure, and these relationships can significantly aid tomography. Similar observation was obtained in several previous works [5][16].

To see it concretely, consider the illustrating example shown in Figure 1. Source s desires to transmit *probe symbols* 1 and 2 to receiver r via intermediate node u. Suppose edge e_1 is erroneous and adds 2 to every symbol transmitted over it. Receiver r knows the probe symbols a prior and wants to detect and locate errors.

The case where node u performs routing only is shown in Figure 1(a). Source s sends symbols 1 and 2 to node u over edges e_1 and e_2 respectively. Due to the error introduced in e_1, node u receives symbols 3 and 2 from edges e_1 and e_2 respectively. Node u then forwards them to r. Node r receives two symbols from e_3 and e_4, denoted as a vector $Y = [3\ 2]^T$. Since r knows the probe symbols 1 and 2, and it can computes the error vector to be $E = Y - [1\ 2]^T = [2\ 0]^T$. Using E, r can see that error happens in the routing path $\{e_1, e_3\}$, but can not figure out whether it is from e_1 or e_3.

In Figure 1(b) network coding is used by u to send coded packets. In this particular case, $\mathbf{x_3} = \mathbf{x_1} + 2\mathbf{x_2}$ and $\mathbf{x_4} = \mathbf{x_1} + \mathbf{x_2}$ are the coding schemes of u, where $\mathbf{x_1}$ and $\mathbf{x_2}$ are the symbols received from e_1 and e_2, and $\mathbf{x_3}$ and $\mathbf{x_4}$ are the symbols to be sent over e_3 and e_4. With error $\mathbf{e} = 2$ injected into e_1, r receives $Y = [7\ 5]^T$. Knowing the probe symbols to be 1 and 2 and the coding scheme node u is using, r can compute the error vector $E = Y - [1 + 2 \cdot 2, 1 + 2]^T = [2\ 2]^T$. For an additive error $\mathbf{e}$ at e_1, e_2, e_3 and e_4, it must result in error vector $\mathbf{e}[1\ 1]^T$, $\mathbf{e}[2\ 1]^T$, $\mathbf{e}[1\ 0]^T$ and $\mathbf{e}[0\ 1]^T$ respectively. Observing $E = \begin{bmatrix} 2 & 2 \end{bmatrix}^T$, r can figure out the error is at e_1 and must be $\mathbf{e} = 2$. While in this toy example it seems that the coding scheme needs to be designed delicately, our results in this paper show random linear coding is sufficient.

Note that if network error correcting code [13][14] is in place, r can decode the source symbols in the presence of network errors. Thus network tomography can be implemented in a passive manner, *i.e.*, no probe packet is needed.



(a) Routing Scheme (b) Coding Scheme

Fig. 1. Example for locating an error at edge e_1. In Figure 1(a) the information of error vector $E = [2\ 0]^T$ is not enough to distinguish the error locations e_1 and e_2. In Figure 1(b), since network coding is used by u, the information of $E = [2\ 2]^T$ is enough to locate the error edge e_1.

In this paper, we consider a network in which all nodes inside perform random linear network coding. To simplify discussion let a source s send messages to a receiver r. Beside receiving the message, the receiver r wants to recover the network topology between s and r, and then detect and locate adversaries activities and random glitches.

We carry out a comprehensive study of passive network tomography in the presence of network failures, under the setting of random linear network coding. In particular, we seek answers to the following fundamental questions:

- Under what conditions is it ever possible to estimate the topology of general networks in the presence of random or adversarial failures, and locate the failures.
- Under what conditions do there exist corresponding computationally tractable algorithms?

For network coding tomography on other settings, such as active tomography on deterministic designed network coding [6]-[9] and network inference using all forwarders information [10]-[12], more details can be found in [15].

A. Our Contributions

We define the *impulse response vector* (IRV) $\mathbf{t}'(\mathbf{e})$ for an edge $e \in \mathcal{E}$ as the transform vector from edge e to the receiver. The $\mathbf{t}'(\mathbf{e})$ can be treated as a fingerprint of e upon failures, and is exposed to the receiver when errors or failures happen on e. This allows us to detect and locate the failing edges.

Our results are categorized into two classes and summarized in Table I.

1) *Topology Estimation*. For the network suffering *random errors* or *random erasures*, we provide the *first* polynomial time algorithm estimating the exact topology. For the network suffering *adversarial errors* or *adversarial erasures*, we provide the *first* existence result for accurate topology estimation. It is assumed that the randomness of each node is chosen from its random code-book known by the receiver. Without such knowledge, we prove that in the presence of adversarial or random failures it is either theoretically impossible or computationally intractable to estimate topology accurately.

2) *Failure Localization*. We provide the *first* polynomial time algorithm for locating the edges experienced random errors and random (and adversarial) erasures. For the network suffering adversarial errors, we provide the *first* proof for intractability of the problem.

Due to space limitation, we only present the topology estimation and error location results for the case of random errors. For the complete results for all cases, please refer to our technical report [15].

TABLE I
SUMMARY OF OUR RESULTS AND PREVIOUS WORKS

Objective	Failure model[1]	Complexity of our algorithms	Complexity of previous works
Topo-Estimation	A,C	Exponential	-
	B,D	Polynomial	-
Error-Localization	A	Hardness Proof	Exponential [5]
	B,C,D	Polynomial	Exponential [5]

II. PROBLEM FORMULATION

A. Notational convention

Scalars are in lower-case (*e.g.* z). Matrices are in upper-case (*e.g.* X). Vectors are in lower-case bold-face (*e.g.* $\mathbf{e}$). Column spaces of a matrix are in upper-case bold-face (*e.g.* $\mathbf{E}$). Sets are in upper-case calligraphy (*e.g.* $\mathcal{Z}$).

B. Settings

For ease of discussion, we consider an acyclic and delay-free network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of edges. Each node has a *unique identification number* known to itself[2]. The capacity of each edge is normalized to be one symbol of $\mathbb{F}_q$ per unit time. Edges with non-unit capacity are modeled as parallel edges. We denote $e(u, v, i)$ as the ith parallel edge between nodes u and v.

We focus on the unicast scenario where a single source s communicates with a single receiver r over the network. In principle, our results can be generalized to any communication scenario for which random linear network coding suffice (for instance, multicast communications). It is assumed each node in $\mathcal{V}$ is connected to r; Otherwise an isolated node would never effect the network communication and then be detected by r.

Let C be the *min-cut* (or *max-flow*) from s to r. Without loss of generality, we assume that both the number of edges leaving the source s and the number of edges entering the receiver r equal C.

C. Network Transmission on Random linear network Coding

We consider the following popular distributed random linear coding scheme [2].

Source encoder: The source s arranges the data into a $C \times n$ *message matrix* X over $\mathbb{F}_q$. It then takes C independently and uniformly random linear combinations over $\mathbb{F}_q$ of the rows of X to generate respectively the packets transmitted on each edge outgoing from s (recall that exactly C edges leave the source s). Note that X contain a short head (*i.e.*, $C \times C$ identity matrix) to record the linear transforms performed along the path from s to r.

Network encoders: Each internal node takes linear combinations of the packets on incoming edges to generate packets transmitted on outgoing edges. Let $\mathbf{x}(u, v, i)$ represent the packet traversing edge $e(u, v, i)$, which is a row vector in $\mathbb{F}_q^{1\times n}$. An internal node v generates its outgoing packet as

$$\mathbf{x}(v, w, j) = \sum_{(u,i):e(u,v,i)\in\mathcal{E}} \beta(u, v, w, i, j)\mathbf{x}(u, v, i).$$

All nodes independently and uniformly choose the local coding coefficients $\{\beta(u, v, w, i, j)\}$ at random. For notational simplicity, in places where it causes no confusion, we write $\beta(u, v, w, i, j)$ as $\beta(e, v, e')$ where $e = e(u, v, i)$ and $e' = e'(v, w, j)$.

Receiver decoder: The decoder constructs the $C \times n$ matrix Y over $\mathbb{F}_q$ by treating the received packets as consecutive length-n row vectors of Y (recall that exactly C edges reach the receiver r). The network internal operations induce a linear transform between X and Y as $Y = TX$, where T is the overall transform matrix. With high probability T is invertible [15]. The receiver extracts T from received packet headers and decode X from Y by $X = T^{-1}Y$.

D. Network failure models

Networks may experience disruption as a part of normal operations. To be concrete a set of edges $\mathcal{Z} \subseteq \mathcal{E}$ suffer errors. Note that an error on edge $e \in \mathcal{Z}$ means that a length-n vector $\mathbf{z}(e)$ is added to the the packet $\mathbf{x}(e)$ carried by e. A random

[1]Failure model notation: A = Adversarial errors; B = Random Errors; C = Adversarial Erasures; D = Random Erasures.

[2]Such a label could correspond to the node's GPS coordinates, or its IP address, or a factory stamp.

error on e means that at least one randomly chosen symbol of the packet $\mathbf{x}(e)$ is changed to a uniformly random symbol in $\mathbb{F}_q$, *i.e.*, $\mathbf{z}(e)$ has at least one symbol chosen uniformly at random from $\mathbb{F}_q$. [3]

E. Network error-correcting codes(ECC)

Consider the scenario where a faulty set of edges $\mathcal{Z}$ of size z *injects* faulty packets. The network transform becomes $Y = TX + T'(\mathcal{Z})Z = TX + E$. Note that Z is a $z \times n$ matrix whose rows are respectively the faulty packets injected by edges in $\mathcal{Z}$, and $T'(\mathcal{Z}) \in \mathbb{F}_q^{C\times z}$ is the linear transform from $\mathcal{Z}$ to r defined later in Section III-A, and the $C \times n$ *error matrix* E is given by $T'(\mathcal{Z})Z$.

It turns out that the receiver r, upon receiving Y can still reconstruct the source message X, by using end-to end network error-correcting codes (ECC) [13][14]. ECC attains the optimal throughput in the presence of network errors.

III. Linear transforms in the network

A. Linear transforms: GEV and IRV

For each edge e there exists a length-C row vector over $\mathbb{F}_q$ called the *global encoding vector* (GEV) $\mathbf{t}(\mathbf{e})$ such that the packet carried by e equals $\mathbf{t}(\mathbf{e})X$. Each $\mathbf{t}(\mathbf{e})$ can be inductively calculated in terms of the linear operations carried out by each node of the network as $\mathbf{t}(\mathbf{e}) = \sum_{1\le j\le d} \beta(e_j, v, e)\mathbf{t}(\mathbf{e_j})$, where e_j $(1 \le j \le d)$ are the incoming edges of v while e is an outgoing edges of v.

For each edge $e \in \mathcal{E}$ we also define its length-C *impulse response vector* (IRV) $\mathbf{t}'(\mathbf{e})$ from e to r. In particular, let s transmit the *all-zero packet* $\mathbf{0} \in \mathbb{F}_q^{1\times n}$ on all outgoing edges, and edge e inject a *nonzero packet* $\mathbf{z}(e) \in \mathbb{F}_q^{1\times n}$. Then Y received by r is $Y = \mathbf{t}'(\mathbf{e})\mathbf{z}(e)$, where $\mathbf{t}'(\mathbf{e})$ is the IRV of e in $\mathbb{F}_q^{C\times 1}$. Thus all the nonzero columns of Y are equivalent to $\mathbf{t}'(\mathbf{e})$ up to a scalar multiple. For a set of edges $\mathcal{Z} \subseteq \mathcal{E}$ with cardinality z, the columns of the $C \times z$ *impulse response matrix* $T'(\mathcal{Z})$ consists of the vectors $\{\mathbf{t}'(\mathbf{e}) : e \in \mathcal{Z}\}$.

Note that both the GEV and the IRV of any edge e are independent of the length of the packet, and GEV are in some sense dual to IRV – the former represent the linear transform from the source s to the edge e, while the latter are the linear transform from the edge e to the receiver r. IRVs can be reversely computed from the incoming edges of the receiver. For example, assume e is an incoming edge of v while e_i $(1 \le i \le d)$ are the outgoing edges of v, then we have $\mathbf{t}'(\mathbf{e}) = \sum_{1\le i\le d} \beta(e, v, e_i)\mathbf{t}'(\mathbf{e_i})$. An illustrating example for edge IRVs can be found in Figure 2.

We normalize GEVs and IRVs so that any two vectors that are scalar multiples of each other are said to be equivalent. Thus, unless otherwise specified, the GEV and the IRV of e are both one-dimensional subspaces in $\mathbb{F}_q^C$.

[3]Note the difference of this model from the usual model of *dense random errors*, wherein each packet is replaced with another packet uniformly at random. The model described in this work is more general. It includes the dense random errors model as a special case. It is also of more practical relevance [15].

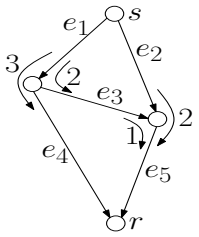


Fig. 2. An example of IRVs of a network. The local encoding coefficients are as shown, and the IRVs are as follows: $\mathbf{t}'(\mathbf{e_4}) = [1, 0]$, $\mathbf{t}'(\mathbf{e_5}) = [0, 1]$, $\mathbf{t}'(\mathbf{e_3}) = \mathbf{t}'(\mathbf{e_5}) = [0, 1]$, $\mathbf{t}'(\mathbf{e_2}) = 2\mathbf{t}'(\mathbf{e_5}) = [0, 2]$ and $\mathbf{t}'(\mathbf{e_1}) = 3\mathbf{t}'(\mathbf{e_4}) + 2\mathbf{t}'(\mathbf{e_3}) = [3, 2]$. Edges e_2 and e_3 are not out-independent, so $\mathbf{t}'(\mathbf{e_2})$ equals $\mathbf{t}'(\mathbf{e_3})$ (up to a scalar multiple). Conversely, e_1 and e_5 are out-independent, so $\mathbf{t}'(\mathbf{e_1})$ is linearly independent from $\mathbf{t}'(\mathbf{e_5})$.

B. Relationships between topology and linear transforms

We demonstrate some correspondences between graph properties of the network and IRVs arose from random linear network coding. We only discuss IRVs here as it suffices for network tomography in random error model. The results for GEVs are necessary for network tomography in erasure model, and can be found in [15].

Any set of z edges $e_1, e_2, ..., e_z$ is said to be *out-independent* if there is one path from the tail node of each edge to r, and these z paths are edge-disjoint. The *out-rank* of an edge set $\mathcal{Z}$ equals the size of a subset of $\mathcal{Z}$, which is comprised of out-independent edges and of maximum size. A collection of edge sets $\mathcal{Z}_1, \mathcal{Z}_2, ..., \mathcal{Z}_n$ is said to be *out-independent* if *out-rank*$(\cup_{i=1}^n \mathcal{Z}_i) = \sum_{i=1}^n$ *out-rank*$(\mathcal{Z}_i)$. For the set $\mathcal{Z} \subseteq \mathcal{E}$ with out-rank z, the *out-extended set* $\mathcal{E}_{OUT}(\mathcal{Z})$ is the set that is of out-rank z, includes $\mathcal{Z}$ and is of maximum size. Note that $\mathcal{E}_{OUT}(\mathcal{Z})$ is well-defined and unique [15].

The relationships between the edges and their IRVs are:

Lemma 1: 1). The rank of the impulse response matrix $T'(\mathcal{Z})$ of an edge set $\mathcal{Z}$ with out-rank z is at most z.

2). The probability that the IRVs of an out-independent edge set are linear independent is at least $1 - |\mathcal{E}|/q$. Thus the probability $rank(T'(\mathcal{Z})) =$ out-rank$(\mathcal{Z})$ is at least $1 - |\mathcal{E}|/q$.

Thus for a large enough field-size q, *out-rank*$(\cup_{i=1}^n \mathcal{Z}_i) = \sum_{i=1}^n$ *out-rank*$(\mathcal{Z}_i)$ if and only if $rank(T'(\cup_{i=1}^n \mathcal{Z}_i)) = \sum_{i=1}^n rank(T'(\mathcal{Z}_i))$.

IV. Topology estimation for random errors

A. Common randomness

Common randomness means that all local coding coefficients $\{\beta(u, v, w, i, j)\}$ are chosen using a common random code-book $\mathcal{R}$, known *a priori* to r. Each internal node, say v, needs to know only the part of the common randomness in $\mathcal{R}$ belonging to v, *i.e.*, the local random code-book of v.

The code-book $\mathcal{R}$ comprises of a list of elements chosen independently and uniformly at random from $\mathbb{F}_q$. This common random code-book can be securely broadcasted by the source before communication using a common public key signature scheme such as RSA, or part of network design.

To be concrete recall that the local coding coefficient $\beta(u, v, w, i, j)$ transforms information from nodes u to v to

w, via the ith and jth parallel edge respectively. Node v uses the element $\mathcal{R}(u,v,w,i,j)$ in the code-book $\mathcal{R}$ as its local coding coefficient $\beta(u,v,w,i,j)$. In this way, each distinct (u,v,w,i,j) tuple indexes a distinct element in $\mathcal{R}$.

Common randomness is both sufficient and necessary for topology estimation. On one hand the sufficiency is a corollary of our topology estimation schemes; On the other hand, the necessity is shown in the following theorem [15].

Theorem 2: 1) Without common randomness, there exist cases where it is statistically impossible to distinguish the actual topology from a class of topologies that lead to the same probability distribution (over the choices of local coding coefficients) of the transform matrix observed by the receiver.

2) Without common randomness, estimating the accurate topology of the network with random network faults is computational intractable[4].

B. Topology estimation algorithm

We provide a polynomial-time scheme to recover the topology of the network that suffers random network errors. The receiver r proceeds in two stages. In the first stage (**Algorithm FIND-IRV**), r recovers the IRVs during several rounds of network communications suffering random errors. In the second stage (**Algorithm FIND-TOPO**), r uses the IRV information obtained to recover the topology. An interesting feature of the algorithms proposed is that random network failures actually make it *easier* to efficiently detect the topology.

Assumptions and Justifications [15]:

1) Multiple "successful" communication rounds[5]. The protocol runs for t independent 'successful" communication rounds, where t is a design parameter chosen to trade off between the probability of success and the computational complexity of the topology estimation protocol. Let $X(i)$ be the source messages transmitted, $\mathcal{Z}(i)$ be the edge set suffering errors and $Y(i)$ be the received matrix in the ith communication round.

2) Weak connectivity requirement. It is assumed that each node (except r) has out-degree no less than 2. Note it implies each edge is *distinguishable* from every other edge, *i.e.*, any pair of edges have out-rank at least 2.

3) For each communication round, each edge e independently has random errors with non-negligible probability, *i.e.*, probability bounded away from 0.

4) The network is not "noodle like" (*i.e.*, high-depth and narrow-width)[6]. To be precise for any distinct $i,j \in [1,t]$ let the random variable $\mathcal{D}(i,j)$ be 1 if and only if $\mathcal{Z}(i)$ is out-independent to $\mathcal{Z}(j)$, *i.e.*, *out-rank*$(\mathcal{Z}(i) \cup \mathcal{Z}(j)) =$ *out-rank*$(\mathcal{Z}(i))+$*out-rank*$(\mathcal{Z}(j))$. Since the edges suffer independent random errors, $Pr(\mathcal{D}(i,j)=1)$ has no dependence on (i,j) and is assumed non-negligible.

Stage I: Find candidate IRVs

[4]It is as hard as the minimum-codeword-problem in random linear code[17].

[5]A "successful" round here means the number of errors does not exceed the bound and r can decode the source message correctly using network error-correcting-code (ECC)

[6]At a high-level, the problem lies in the fact that such networks have high description complexity (dominated by the height), but can only support a low information rate (dominated by the width).

Recall the source message is formed as $X(i)=[I_C, M(i)]$, where I_C is a $C\times C$ identity matrix and $M(i)\in \mathbb{F}_q^{C\times(n-C)}$ is the message. Let $Y(i)_h$ (and $Y(i)_l$) be the matrix containing the first C columns (and last $n-C$) of $Y(i)$. Then the algorithm that finds a set of candidate IRVs is as follows:

- **Algorithm FIND-IRV**: The input is $\{Y(i), i\in[1,t]\}$. The output is $\mathcal{I}_{IRV}$ which is a set of dimension-one subspaces in $\mathbb{F}_q^C$ and initialized as an empty set.
- Step A: For $i\in[1,t]$, r computes $M(i)$ using ECC and then $E(i)_r = Y(i)_\ell - Y(i)_h M(i)$.
- Step B: The intersection of the column-spaces $\mathbf{E(i)_r}\cap \mathbf{E(j)_r}$ is computed for each pair $i,j\in\{1,\ldots,t\}$. If $rank(\mathbf{E(i)_r}\cap\mathbf{E(j)_r})=1$ for any (i,j) pair, $\mathbf{E(i)_r}\cap\mathbf{E(j)_r}$ is added into $\mathcal{I}_{IRV}$.
- Step C: End **FIND-IRV**:

Theorem 3 characterizes when all IRVs would join in $\mathcal{I}_{IRV}$:

Theorem 3: The probability that $\mathcal{I}_{IRV}$ contains all edge IRVs is $1-o(1)$ when $t=\mathcal{O}(|\mathcal{E}|)$.

The theorem is based on the following Lemma 4, which is arisen from the properties of random errors and is a core lemma for network tomography in random network errors.

Lemma 4: For random error model, $\mathbf{E(i)_r}=\mathbf{T}'(\mathcal{Z}(\mathbf{i}))$ with probability at least $(1-z/q)[1-2C^2/(n-C)]$, which can arbitrarily close to one for large enough q and n.

Remarks: 1) r can estimate how many communication rounds t is needed so that $\mathcal{I}_{IRV}$ has the desired probability to include all edge IRVs [15]. 2) $\mathcal{I}_{IRV}$ can also include some "fake" IRVs [15]. In the next stage for topology estimation, these fake IRVs will be filtered out automatically.

Stage II: Topology recovery via candidate IRVs

We now describe **Algorithm FIND-TOPO** that determines the network topology, using $\mathcal{I}_{IRV}$.

Note that $\mathcal{I}_{IRV}$ is merely a set of one-dimensional subspaces, and as such, individual element may have no correspondence with the actual IRV of any edge in the network. At any point in **FIND-TOPO**, let $\bar{\mathcal{G}}$ denote the network topology recovered thus far. Let $\bar{\mathcal{V}}$ and $\bar{\mathcal{E}}$ be the corresponding sets of nodes and edges respectively in $\bar{\mathcal{G}}$, and $\bar{\mathcal{I}}_{IRV}$ be the set of IRVs of the edges in $\bar{\mathcal{E}}$, which are computed from $\bar{\mathcal{G}}$ and the random code-book $\mathcal{R}$. We note that the IRVs in $\bar{\mathcal{I}}_{IRV}$ are vectors rather than one-dimensional subspaces.

We describe algorithm estimating the network topology as follows.

- **Algorithm FIND-TOPO**: The input is $\mathcal{I}_{IRV}$ and the common random code-book $\mathcal{R}$. The output is $\bar{\mathcal{G}}=(\bar{\mathcal{V}},\bar{\mathcal{E}})$.
- Step A: The set $\bar{\mathcal{V}}$ is initialized as the receiver r, all its upstream neighbors, and the source s. The set $\bar{\mathcal{E}}$ is initialized as the set of edges incoming to r. Hence $\bar{\mathcal{G}}=(\bar{\mathcal{V}},\bar{\mathcal{E}})$. The initial set of $\bar{\mathcal{I}}_{IRV}$ are the IRVs of the incoming edges of r, *i.e.*, a set of distinct unit vectors. The state STATE(New-Edge) is initialized to be "False".
- Step B: For each node $v\neq s$ in $\bar{\mathcal{V}}$, call function $FindNewEdge(v)$ (Step C). If
  - STATE(New-Edge) is "True", set STATE(New-Edge) be "False" and return to the beginning of Step B.

– STATE(New-Edge) is "False", go to Step E.

- Step C: (Function $FindEdge(v)$) Let $e_1, \ldots, e_d$ be the outgoing edges of v in $\bar{\mathcal{G}}$. If $\{\bar{\mathbf{t}}'(\mathbf{e_1}), \ldots, \bar{\mathbf{t}}'(\mathbf{e_d})\}$ from $\bar{\mathcal{I}}_{IRV}$ has
  - rank 1, step-back to the loop in Step B.
  - rank greater than 1, for each candidate incoming edge of v, say $e = (u, v, i)$, if $e \notin \bar{\mathcal{E}}$, call function $CheckIRV(v, e)$ (Step D). Step-back to the loop in Step B.
- Step D: (Function $CheckIRV(v, e)$) Use $\mathcal{R}$ to compute the IRV of e as $\bar{\mathbf{t}}'(\mathbf{e}) = \sum_{j=1}^{d} \beta(e, v, e_j)\bar{\mathbf{t}}'(\mathbf{e_j})$. Check whether $\bar{\mathbf{t}}'(\mathbf{e})$ is in one of $\bar{\mathcal{I}}_{IRV}$. If so,
  1) Set STATE(New-Edge) be "True".
  2) If $u \notin \bar{\mathcal{V}}$, add u to $\bar{\mathcal{V}}$.
  3) Add $e = e(u, v, i)$ to $\bar{\mathcal{E}}$.
  4) Update $\bar{\mathcal{I}}_{IRV}$ from $\bar{\mathcal{G}} = (\bar{\mathcal{V}}, \bar{\mathcal{E}})$ and $\mathcal{R}$[7].

  Step-back to the loop in Step C.
- Step E: End **FIND-TOPO**.

If $\mathcal{I}_{IRV}$ contains all edge IRVs which is supported by Theorem 3, we show correctness of **FIND-TOPO** as:

Theorem 5: With probability $1 - \mathcal{O}(|\mathcal{E}|^4|\mathcal{V}|)/q$, **FIND-TOPO** recovers the accurate topology by performing $\mathcal{O}(|\mathcal{E}|^4|\mathcal{V}|C)$ operations over $\mathbb{F}_q$.

Sketch of idea: Since the elements of code-book $\mathcal{R}$ are randomly and independently chosen, the probability that function $CheckIRV(v)$ accepts a non-existed edge is negligible.

V. Error Localization

To locate the set of edges $\mathcal{Z}$ suffering random errors, we assume the IRVs of the edges are known to r. The information can be estimated by the scheme in Section IV, or is given a priori in the process of network design. We also assume network ECC is used to transmit information from s to r.

Since $\mathbf{T}'(\mathcal{Z}) = \mathbf{T}'(\mathcal{E}_{\mathbf{OUT}}(\mathcal{Z}))$ (see the definition of $\mathcal{E}_{OUT}(\mathcal{Z})$ in Section III-B for reference), the receiver can not distinguish whether the errors are from $\mathcal{Z}$ or $\mathcal{E}_{OUT}(\mathcal{Z})$. So rather than finding $\mathcal{Z}$, we provide a computationally tractable algorithm to locate $\mathcal{E}_{OUT}(\mathcal{Z})$, a proxy for $\mathcal{Z}$. The algorithm that finds $\mathcal{E}_{OUT}(\mathcal{Z})$ is as follows:

- **Algorithm LOCATE**: The input is the matrix Y received by r, and the IRV of each edge. The output is an edge set $\mathcal{Z}'$ initialized as an empty set.
- Step A: r computes E_r as the Step A of **Algorithm FIND-IRV**.
- Step B: r checks for each IRV $\mathbf{v}$ whether it lies in $\mathbf{E_r}$. If so, the edge corresponding to $\mathbf{v}$ is added into $\mathcal{Z}'$.
- Step C: End **LOCATE**.

The correctness of **LOCATE** is followed.

Theorem 6: If z is no more than $C - 1$, $\mathcal{Z}' = \mathcal{E}_{OUT}(\mathcal{Z})$ with probability at least $1 - 3|\mathcal{E}|^2/q - C^2/(n - C)$. The computational complexity is $\mathcal{O}(|\mathcal{E}|C^2)$ operations over $\mathbb{F}_q$.

[7]The reason that $\bar{\mathcal{I}}_{IRV}$ needs to be updated is that: when e is found as a new edge in $\bar{\mathcal{G}}$, the IRVs of the edges upstream of e in $\bar{\mathcal{G}}$ will change.

The high level idea is that $\mathbf{E_r} = \mathbf{T}'(\mathcal{Z})$ with high probability from Lemma 4, where $\mathbf{T}'(\mathcal{Z})$ is spanned by $\{\mathbf{t}'(\mathbf{e}), e \in \mathcal{E}_{OUT}(\mathcal{Z})\}$.

For a short glance at adversarial error model, the hardness of locating z adversarial edges arises when the adversary carefully choose Z such that Z has low rank and then $\mathbf{E_r} \neq \mathbf{T}'(\mathcal{Z})$, *i.e.*, Lemma 4 is not true. In such case r needs to try every edge set $\mathcal{Z}'$ with size z and check whether $\mathbf{E_r} \subseteq \mathbf{T}'(\mathcal{Z}')$. Unfortunately this brute-force searching would increase the computational complexity exponentially and be proved unavoidable.

VI. Acknowledgement

This work was supported by National Natural Science Foundation of China Grant 60553001, the National Basic Research Program of China Grant 2007CB807900 and 2007CB807901, RGC GRF grant 412608, 411008, and 411209, RGC AoE grant on Institute of Network Coding, established under the University Grant Committee of Hong Kong, CUHK MoE-Microsoft Key Laboratory of Human-centric Computing and Interface Technologies, Direct Grant (Project Number 2050397) of The Chinese University of Hong Kong, and two gift grants from Microsoft and Cisco.

References

[1] R. Castro, M. Coates, G. Liang, R. D. Nowak, and B. Yu, "Network tomography: recent developments," *Statistical Science*, 2004.
[2] T. Ho, M. Mdard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *IEEE Trans. on IT*, vol. 52, no. 10, pp. 4413–4430, 2006.
[3] P. A. Chou, Y. Wu, and K. Jain, "Practical network coding," *Allerton 2003*.
[4] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. on IT*, vol. 46, no. 4, pp. 1204–1216, 2000.
[5] T. Ho, B. Leong, Y. H. Chang, Y. G. Wen, and R. Koetter, "Network monitoring in multicast networks using network coding," *ISIT 2005*.
[6] C. Fragouli and A. Markopoulou, "A network coding approach to network monitoring," *Allerton 2005*.
[7] C. Fragouli, A. Markopoulou, and S. Diggavi, "Topology inference using network coding," *Allerton 2006*.
[8] M. Gjoka, C. Fragouli, P. Sattari, and A. Markopoulou, "Loss tomography in general topologies with network coding," *Globecom 2005*.
[9] P. Sattari, A. Markopoulou, and C. Fragouli, "Multiple source multiple destination topology inference using network coding," *NetCod 2009*.
[10] M. J. Siavoshani, C. Fragouli, S. Diggavi, and C. Gkantsidis, "Bottleneck discovery and overlay management in network coded peer-to-peer system," *SIGCOMM workshop on Internet Network Management*, 2007.
[11] J. M. Siavoshani, C. Fragouli, and S. Diggavi, "Subspace properties of randomized network coding," *IEEE ITW 2007*.
[12] M. J. Siavoshani, C. Fragouli, and S. Diggavi, "On locating byzantine attackers," *Network Coding Workshop: Theory and Applications*, 2008.
[13] S. Jaggi, M. Langberg, S. Katti, T. Ho, D. Katabi, and M. Medard, "Resilient network coding in the presence of byzantine adversaries," *INFOCOM 2007*.
[14] D. Silva, F. R. Kschischang, and R. Kotter, "A rank-metric approach to error control in random network coding," *IEEE Trans. on IT*, vol. 54, no. 9, pp. 3951–3967, 2008.
[15] H. Yao, S. Jaggi, and M. Chen, "Network tomography for network failurs," submitted to IEEE Trans on IT, available at `http://arxiv.org/abs/0908.0711`.
[16] G. Sharma, S. Jaggi, and B. K. Dey, "Network tomography via network coding," *Information Theory and Applications Workshop*, 2008.
[17] A. Vardy, "The intractability of computing the minimum distance of a code," *IEEE Trans. on IT*, vol. 43, no. 6, pp. 1757–1766, 1997.