On Algebraic Traceback in Dynamic Networks

Abhik Das, Shweta Agrawal and Sriram Vishwanath
Department of Electrical & Computer Engineering
University of Texas, Austin, USA
Email: {akdas, shweta.a}@mail.utexas.edu, sriram@austin.utexas.edu

***Abstract*—This paper discusses the concept of *incremental traceback* for determining changes in the trace of a network as it evolves with time. A distributed algorithm, based on the methodology of algebraic traceback developed by Dean et al., is proposed that can determine a path of d nodes using $O(d)$ marked packets, and subsequently determine the changes in it using $O(\log d)$ marked packets. The algorithm is established to be order-wise optimal i.e. no other distributed algorithm can determine changes in the path topology using lesser order of bits (i.e. marked packets). The algorithm is shown to have a computational complexity of $O(d \log d)$, which is significantly less than that of any existing non-incremental algorithm for algebraic traceback. The extension of traceback mechanisms to systems deploying network coding is also considered.**

***Index Terms*—Incremental traceback, MANETs.**

I. Introduction

Given the increasing number and forms of attacks on networks in recent years, developing efficient counter-measures, such as *traceback*, is of significant value. In simple words, traceback refers to determination/tracking of the paths traversed by packets in a network. In this paper, we focus on developing efficient traceback mechanisms for networks with time-varying topologies. Settings such as mobile ad-hoc networks (MANETs) are of particular interest in which we desire to use traceback towards network management and defence against attacks. For the Internet and wired networks, *IP traceback* is a popular mechanism for countering denial-of-service (DoS) attacks [1], [2]. Similarly, generalized (not necessarily IP-based) traceback proves to be useful in determining the origin of attacks and monitoring network behavior for MANETs. There are many other purposes that traceback can serve. For instance, it can be used for network maintenance and monitoring [3], source/route verification and determining the location of faulty nodes in the network.

In this paper, we present an algebraic *incremental traceback* approach for time-varying networks. Our traceback mechanism is suitable for networks where the topology changes at a rate much slower than the rate of data transmission, such as wireless networks. We analyze the performance of this mechanism using a framework similar to random coding arguments. Traceback mechanisms have been traditionally studied for IP-based networks under the name of IP traceback [4], [5], [6], [7], [8], but all these approaches have been *non-incremental* in a dynamic network setup, which do not utilize existing knowledge of network topology in computing changes to the same. In contrast, our approach to traceback is *incremental* and the information about network topology, obtained from prior traceback operations, is actually utilized. We base our incremental traceback mechanism on the framework of algebraic traceback, developed by Dean et al [8].

We consider traceback being performed in a continuous manner, with the goal of ensuring that the destination(s) stay well informed of the path(s) traversed by the received packets. We show that, once a path of d routers/nodes in a network is learnt using the non-incremental version of algebraic traceback, $O(\log d)$ marked packets and a traceback algorithm with a computational complexity of $O(d \log d)$ (operations per execution) are sufficient to *track the changes* (node addition and deletion) in the path. Note that, if we were to use the non-incremental traceback process each time there is a change in the path, $O(d)$ marked packets would be required to perform traceback. Next, we argue that our incremental traceback process is order-wise optimal in terms of the number of marked packets (number of bits) required.

We limit our incremental traceback approach to track single node addition and deletion in a path of the network. This is deliberate, as conventionally, in wireless networks, the time-scale at which routes/paths change in an ad hoc network (order of seconds) is many orders of magnitude greater than the time-scale of data transmission (order of milliseconds or less). Thus, any one change can be detected before additional changes occur in a path. Our algorithm and analysis framework can be naturally extended to scenarios when multiple nodes can enter or leave a path.

II. System Model

We consider a directed graph, that represents the connectivity model of the network considered. The nodes in the graph have unique identifiers (IDs) that come from the finite field $GF(p)$, for some large, suitable prime number p. A directed edge between a pair of nodes in the graph represents an error-free channel. We assume that the transmissions across different edges do not interfere with each other in any way.

We focus our attention on a source and a destination in the network, represented in the graph by nodes r_1 and D respectively. The source node wants to transmit data to the destination node using the path $\mathcal{P} = (r_1, r_2, \ldots, r_d, D)$. However, this path may change over the course of the transmission, and we consider the case when exactly one node is added

We acknowledge the support of the DARPA IAMANET program, the AFOSR and the ARO for this research.

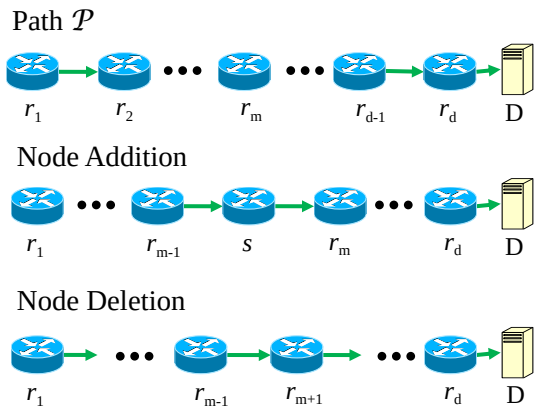

Fig. 1. Dynamic behavior of $\mathcal{P}$

to or removed from path $\mathcal{P}$ (Figure 1). We want to develop an incremental algebraic traceback mechanism that enables destination D to figure out this change in path $\mathcal{P}$.

We assume that there is the possibility of node-ID spoofing, i.e., a malicious node in path $\mathcal{P}$ misreporting its ID. We include a flag-bit field and hop-count field (with initial values 0) in each packet and assume that the hop-count field cannot be altered by any malicious node. For the case when the hop-count field is attackable, please refer to [8].

III. Review: Algebraic Traceback

In this section, we present the relevant aspects of algebraic traceback, developed by Dean et al [8]. The main idea behind this traceback scheme is that a polynomial of degree n in $GF(p)$ is completely determinable using $(n+1)$ of its evaluations at distinct points in $GF(p)$.

A. Deterministic Path Encoding

The deterministic path encoding scheme is used when no node-ID spoofing is suspected. The encoding or packet marking process is initiated by the first node (source node) that encounters the packet (r_1 for $\mathcal{P}$). The flag-bit and hop-count values are set to 1 when a packet is marked, otherwise the flag-bit value remains unchanged and each node following the source node just increments the hop-count by 1. In path $\mathcal{P}$, when node r_1 initiates the packet marking process (with some fixed probability), it encodes a value-pair (x, y) into it, where x is chosen uniformly at random from $GF(p)$ and $y = r_1$. When node r_i $(i = 2, \ldots, d)$ encounters a marked packet, it uses the values x, y, r_i to update y as:

$$y \leftarrow y \cdot x + r_i. \tag{1}$$

If destination D gets d distinct value-pairs $(x_i, y(x_i))$, $i = 1, 2, \ldots, d$, from the marked packets, then path $\mathcal{P}$ can be reconstructed by solving the following matrix equation:

$$\begin{bmatrix} 1 & x_1 & x_1^2 & \ldots & x_1^{d-1} \\ 1 & x_2 & x_2^2 & \ldots & x_2^{d-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_d & x_d^2 & \ldots & x_d^{d-1} \end{bmatrix} \begin{bmatrix} r_d \\ r_{d-1} \\ \vdots \\ r_1 \end{bmatrix} = \begin{bmatrix} y(x_1) \\ y(x_2) \\ \vdots \\ y(x_d) \end{bmatrix}.$$

The resulting matrix in the equation is a full-rank Vandermonde matrix, which is solvable in $O(d^2)$ operations.

B. Randomized Path Encoding

The deterministic path encoding scheme may be infeasible if node-ID spoofing is possible and/or the first node to receive a packet is unsure if it is the source node (for example, if r_1 does not know it is the source node in path $\mathcal{P}$). This situation can be addressed by using a probabilistic traceback mechanism. For path $\mathcal{P}$, node r_1 initiates marking of a packet as before (with probability, say q_1), but now each intermediate node r_i clears an existing marking on a packet, if any, and re-marks it with probability q_i, $i = 2, \ldots, d$. Otherwise, with probability $(1 - q_i)$, node r_i follows the update mechanism as given by Equation (1). Note that the hop-count and flag-bit values are reset to 1 every time a packet gets remarked.

If the values of marking probabilities q_i, $i = 1, 2, \ldots, d$, are non-trivial, apart from marked packets with value-pairs corresponding to path $\mathcal{P}$, there are marked packets with value-pairs corresponding to sub-paths $\mathcal{P}_i = (r_{i+1}, r_{i+2}, \ldots, r_d)$, $i = 1, 2, \ldots, (d-1)$. Then, a marked packet received by D has value-pair of the form $(x, y(x))$ where

$$y(x) = \sum_{i=0}^{k} r_{d-i} x^i, \;\; k = 0, 1, \ldots, d-1.$$

These marked packets can be segregated in terms of the sub-paths their value-pairs correspond to, on the basis of their hop-count values, as hop-count value of $i (< d)$ means that the value-pair is for sub-path $\mathcal{P}_{d-i}$ and hop-count value of d means that the value-pair is for path $\mathcal{P}$. Using this, the sub-paths and therefore the path $\mathcal{P}$ can be reconstructed after getting sufficient number of marked packets.

Let f_i, $i = 1, 2, \ldots, d$, be defined as the fraction of packets whose marking was initiated by node r_i and received by destination D. f_i can be expressed in terms of q_i, $i = 1, 2, \ldots, d$, as

$$f_i = \begin{cases} q_i \prod_{j=i+1}^{d} (1 - q_j) & i \neq d \\ q_d & i = d \end{cases}.$$

The fraction of unmarked packets received by destination D is given by $f_0 = 1 - \sum_{i=1}^{d} f_i = \prod_{i=1}^{d} (1 - q_i)$. It can be shown that an average of $d\lceil (1 - f_0)/f_1 \rceil$ or $O(d(1 - f_0)/f_1)$ marked packets need to be received by destination D to determine the path $\mathcal{P}$ with a computational complexity of $O(d^2)$.

IV. Incremental Algebraic Traceback: Deterministic Path Encoding

In this section, we present an incremental traceback approach based on the methodology of deterministic path encoding. We adopt the same packet encoding/marking procedure as in Section III-A, i.e. the source node initiates the packet marking process with some fixed probability.

As discussed earlier, the path $\mathcal{P}$ can be ascertained using $O(d)$ marked packets with a computational complexity of $O(d^2)$. Our interest lies in the case when path $\mathcal{P}$ has been determined by destination D, and then its topology changes

due to node addition or deletion. The conventional way to determine the change(s) would be to repeat the traceback process, i.e. destination D waits until it receives $O(d)$ marked packets again, reconstructs the modified path and detects the change(s) in topology of path $\mathcal{P}$. This scheme is very inefficient – the number of marked packets and computational load incurred remains the same despite the fact that destination D has knowledge of path $\mathcal{P}$. The proposed incremental traceback mechanism makes use of this knowledge to determine the correct change in topology of path $\mathcal{P}$ accurately using $O(\log d)$ marked packets with a complexity of $O(d \log d)$.

The change in topology of path $\mathcal{P}$ involves either addition or deletion of a single node, which can be detected using the hop-count value of a marked packet – it changes from d to $(d+1)$ for node addition and to $(d-1)$ for node deletion. Hence, we examine these two cases separately.

A. Node Addition

Suppose a node with identity s gets added to path $\mathcal{P}$ in the mth position, $1 \leq m \leq (d+1)$ (1st position refers to the position before node r_1 and $(d+1)$th position refers to the position after node r_d). Then the new packets have value-pairs of the form $(x, z(x))$ encoded in them, where

$$z(x) = a_m(x) + x^{d-m+1}(s + x b_m(x)). \tag{2}$$

$a_k(x)$ and $b_k(x)$ are polynomials given by

$$a_k(x) = \begin{cases} r_d + r_{d-1}x + \ldots + r_k x^{d-k} & k \neq d+1 \\ 0 & k = d+1 \end{cases} \tag{3}$$

$$b_k(x) = \begin{cases} r_{k-1} + r_{k-2}x + \ldots + r_1 x^{k-2} & k \neq 1 \\ 0 & k = 1 \end{cases} \tag{4}$$

for $k = 1, 2, \ldots, (d+1)$. These polynomials are known to destination D from the usual traceback performed previously, which gives $r_1, r_2, \ldots, r_d$. The polynomials also satisfy

$$y(x) = \sum_{i=0}^{d-1} r_{d-i} x^i = a_k(x) + x^{d-k+1} b_k(x)$$

for all k, where $y(x)$ refers to the y-value of the marked packet received by D prior to entry of node s in path $\mathcal{P}$.

Suppose (x_i, z_i), $i = 1, 2, \ldots, l$, be the value-pairs encoded in l marked packets received after the addition of s in the path $\mathcal{P}$. We consider the following set of equations:

$$z_j = a_k(x_j) + x_j^{d-k+1}(s + x_j b_k(x_j)),\ 1 \leq j \leq l. \tag{5}$$

From relation (2), the above set of equations is consistent for $k = m$. For $k \neq m$, the set of equations is not consistent with high probability (established by Theorem 1 below). We make use of this property to design an incremental traceback algorithm, for the case of node addition, as follows:

Algorithm I

(i) Construct a $(d+1) \times l$ matrix $\hat{S} = [\hat{s}_{kj}]$ where

$$\hat{s}_{kj} = \frac{z_j - a_k(x_j)}{x_j^{d-k+1}} - x_j b_k(x_j).$$

(ii) If there exists a unique row in $\hat{S}$ with equal elements, say the $\hat{m}$th row, declare that the new node is in $\hat{m}$th position with identity $\hat{s} = \hat{s}_{\hat{m}j}$, $1 \leq j \leq l$.

(iii) If there exists more than one row in $\hat{S}$ with equal elements, declare that an error has occurred. Wait for more value-pairs to arrive through marked packets, say (x_i, z_i), $i = l+1, \ldots, l+\epsilon$, where ϵ is an integer of smaller order compared to l. Repeat the algorithm using the value-pairs (x_i, z_i), $i = \epsilon+1, \ldots, l+\epsilon$. Theorem 1 below shows that the algorithm terminates with high probability, giving the correct node identity.

Theorem 1: A newly added node in path $\mathcal{P}$ can be determined by destination D using *Algorithm I* and $l = \lceil \log_p d + \delta \rceil$ marked packets ($\delta \in \mathbb{N}$) with an error probability of atmost $p^{-\delta}$ and a computational complexity of $O(d \log d)$.

Proof: From (5), it is clear that all elements of the mth row of $\hat{S}$ will be equal. If this is the only such row, we have the correct new node position and identity $s = \hat{s}_{mj}$, $1 \leq j \leq l$. An error occurs if there exists another row $i \neq m$ such that all elements of the ith row are equal as well. To determine the probability of this happening, we note that x_j is chosen uniformly over $GF(p)$. This makes $\hat{s}_{kj}$ uniform for any $k \neq m$, since each $\hat{s}_{kj}$ is purely a function of x_j. So, $\hat{s}_{ij}$, $j = 1, 2, \ldots, l$ is an i.i.d. uniform random process. This gives

$$Pr(\hat{s}_{ij} = \hat{s}_{ij'}) = \frac{1}{p} = 2^{-\log_2 p}$$

for any $1 \leq j, j' \leq l$ and $j \neq j'$. Let E_i be the event that all elements of the ith row of $\hat{S}$ are same. Then we have $Pr(E_i) = 2^{-l \log_2 p}$ for $i \neq m$, since there are l elements in each row. The probability of error is

$$P_e = Pr(\cup_{i \neq m} E_i) \leq d Pr(E_i) = 2^{\log_2 d - l \log_2 p}$$

where the inequality above is due to the union bound. P_e can be made arbitrarily small if $\log_2 d - l \log_2 p$ can be made as negative as possible. If we require that $l > \frac{\log_2 d}{\log_2 p}$, then this can be satisfied. Thus, we choose $l = \lceil \frac{\log_2 d}{\log_2 p} + \delta \rceil$, where $\delta \in \mathbb{N}$ is a small constant. Then P_e gets upper-bounded as

$$P_e \leq 2^{\log_2 d - l \log_2 p} = \frac{1}{p^\delta} 2^{\log_2 d - \log_2 p \lceil \frac{\log_2 d}{\log_2 p} \rceil} \leq \frac{1}{p^\delta}$$

where the second inequality follows from the fact that $a - b\lceil \frac{a}{b} \rceil \leq 0\ \forall a, b \in \mathbb{R}, b \neq 0$. By choosing a large enough value for p, P_e can be bounded above by any arbitrary small positive value. In other words, $l = O(\log d)$ is sufficient for determining the newly added node correctly.

Since the algorithm relies on the computation of $\hat{S}$ which has $(d+1)l$ entries, we get a complexity of $O(d \log d)$ (since $l = O(\log d)$). This completes our proof.

B. Node Deletion

The case of node deletion is handled analogously using an incremental traceback algorithm named *Algorithm II* – for details please refer to [9]. We have the following result for node deletion in $\mathcal{P}$:

Theorem 2: A deleted node in path $\mathcal{P}$ can be determined by

destination D using *Algorithm II* and $l = \lceil \log_p d + \delta \rceil$ marked packets ($\delta \in \mathbb{N}$) with an error probability of atmost $p^{-\delta}$ and a computational complexity of $O(d \log d)$.

Thus, be it node addition or deletion, $O(\log d)$ marked packets are always sufficient for D to determine the correct change in $\mathcal{P}$. Note that the order-optimality of *Algorithms I* and *II* is fairly straightforward. The entropy of a uniform source with an alphabet of size k is $\log_2 k$ bits. Thus, even if a centralized mechanism existed that could communicate the location of the node being inserted/deleted in $\mathcal{P}$, it would require $O(\log_2 d)$ bits to do so, as there are d equally likely positions for the change to occur. Our distributed mechanism uses $\lceil \log_p d + \delta \rceil$ packets or $2(\log_2 d + \delta \log_2 p)$ bits. Thus, in terms of the order of growth of network overhead in d, the incremental traceback algorithms are optimal.

V. Incremental Algebraic Traceback: Randomized Path Encoding

In this section, we present an incremental traceback approach, based on the methodology of randomized path encoding. This is useful in scenarios when node-ID spoofing is suspected or there are multiple attacks. We use the same packet marking procedure as in Section III-B, i.e. each node clears the encoded data in a marked packet with some fixed probability and re-initiates the marking process.

For path $\mathcal{P}$, the packet remarking probability for node r_i is q_i. As a result, multiple nodes in path $\mathcal{P}$ act as source nodes and the marked packets carry value-pairs corresponding to path $\mathcal{P}$ and sub-paths $\mathcal{P}_i$, $i = 1, 2, \ldots, (d-1)$. As described earlier, path $\mathcal{P}$ is completely determinable using an average of $O(d(1 - f_0)/f_1)$ marked packets with a computational complexity of $O(d^2)$. Once $\mathcal{P}$ is known to D, we show that it possible to detect the correct changes in $\mathcal{P}$, using smaller number of marked packets.

Due to the random nature of packet-marking, we cannot immediately ascertain if node addition or node deletion has occurred from the hop-count value of the marked packets. A change occurring at the first position in path $\mathcal{P}$, i.e. either node r_1 getting deleted or a new node getting added before it, can be detected only using those packets that are marked by the first node on the new path, which we call path $\mathcal{P}'$. In fact, these marked packets give information about any change occurring anywhere in path $\mathcal{P}$. Let f_i' denote the fraction of packets received by destination D and marked by the ith node of path $\mathcal{P}'$. Then, the fraction of marked packets coming from the first node of path $\mathcal{P}'$ is $f_1'/(1-f_0')$, where $f_0' = 1 - \sum_{i \geq 1} f_i'$ is the fraction of unmarked packets. This means that, from a average of $l\lceil (1-f_0')/f_1' \rceil$ new marked packets received by destination D after a change in path $\mathcal{P}$ (addition or deletion of a node), the l marked packets with highest hop-count values are most likely to come from the first node of path $\mathcal{P}'$.

Suppose a new node with identity s gets added at the mth position in path $\mathcal{P}$ ($1 \leq m \leq d+1$). Then a marked packet with hop-count value h, where $d-m+2 \leq h \leq d+1$, contains information about the new node. The value-pair $(x, z(x))$ encoded in the marked packet satisfies

$$z(x) = a_m(x) + x^{d-m+1}(s + x b_{m,h}(x)). \tag{6}$$

$a_m(x)$ is defined as in (3) and $b_{k,h}(x)$ is defined as

$$b_{k,h}(x) = r_{k-1} + r_{k-2}x + \ldots + r_{d-h+2}x^{k-d+h-3}$$

for $k = d-h+2, \ldots, d+1$ and $b_{k,h}(x) = 0$ for $k = d-h+2$. Similarly, if node r_m ($1 \leq m \leq d$) gets deleted from path $\mathcal{P}$, a marked packet with hop-count h, where $d - m + 1 \leq h \leq d-1$, contains information about the deletion. The value-pair $(x, w(x))$ of the marked packet satisfies

$$w(x) = a_m(x) - x^{d-m}(r_m - b_{m,h+2}(x)). \tag{7}$$

Depending upon whether a node gets added or deleted in path $\mathcal{P}$, the new path $\mathcal{P}'$ has $d+1$ or $d-1$ nodes respectively. If there is no change in path $\mathcal{P}$, we have $\mathcal{P}' = \mathcal{P}$. So, (f_0', f_1') can take three possible values, one is (f_0, f_1) when there is no change, the other two values result from node addition or deletion in path $\mathcal{P}$. Let (F_0, F_1) be that value of (f_0', f_1') which maximizes $(1-f_0')/f_1'$ over the three choices. Suppose (x_i, z_i), $i = 1, 2, \ldots, l$, be the value-pairs in the marked packets with the highest hop-count values, say h_i, $i = 1, 2, \ldots, l$, among $l\lceil (1-F_0)/F_1 \rceil$ marked packets received by destination D. By an average/expected value arguement, these l packets are marked by nodes close to the first node of new path $\mathcal{P}'$ and they possess information about any change in path $\mathcal{P}$. If $h_i = d+1$ for some i, it means there has been node addition but if $h_i \leq d$ for all i, we cannot conclude anything and have to consider both the possibilities of node addition and node deletion. We propose the following incremental traceback algorithm for destination D to determine any change in path $\mathcal{P}$ accurately:

Algorithm III

(i) Construct a $(d+1) \times l$ matrix $\hat{S} = [\hat{s}_{kj}]$ where

$$\hat{s}_{kj} = \frac{z_j - a_k(x_j)}{x_j^{d-k+1}} - x_j b_{k,h_j}(x_j)$$

for $k \geq d - h_j + 2$ and $\hat{s}_{kj} = 0$ otherwise.

(ii) If there exists a unique row in $\hat{S}$, say the $\hat{m}$th row, such that all non-zero elements (there should be atleast two non-zero elements) of the row are equal, declare that there is a new node added in $\hat{m}$th position with identity $\hat{s}$ equal to the non-zero element value.

(iii) If there exists more than one row in $\hat{S}$ with equal non-zero elements, declare that an error has occurred. Wait to get more value-pairs with high hop-count values through marked packets. Repeat (i), (ii) using these and some of the earlier value-pairs (l value-pairs in all).

(iv) If there exists no row in $\hat{S}$ with equal non-zero elements, construct a $d \times l$ matrix $\hat{R} = [\hat{r}_{kj}]$ where

$$\hat{r}_{kj} = b_{k,h_j+2}(x) - \frac{z_j - a_k(x_j)}{x_j^{d-k}}$$

for $k \geq d - h_j + 1$ and $\hat{r}_{kj} = 0$ otherwise.

(v) If there exists a unique row in $\hat{R}$, say the $\hat{m}$th row, such that all non-zero elements of the row are equal, declare that the node in $\hat{m}$th position has been deleted with identity equal to the non-zero element value.

(vi) If there exists more than one row in $\hat{R}$ with equal non-zero elements, declare that an error has occurred. Wait to get more value-pairs with high hop-count values through marked packets. Repeat (iv), (v) using these and some of the earlier value-pairs (l value-pairs in all).

(vii) If there exists no row in $\hat{R}$ with equal non-zero elements, declare that there has been no change in $\mathcal{P}$.

Theorem 3: Any change in path $\mathcal{P}$ can be determined by destination D using $l = O(\log d)$ marked packets, which contain information about the change encoded in them, and *Algorithm III* with a computational complexity of $O(d \log d)$.
Proof: See proof of Theorem 3 in [9].

Thus, $O(\log d)$ marked packets, with the information about the change in $\mathcal{P}$ encoded in them, or an average of $O((\log d)(1 - F_0)/F_1)$ marked packets in general, are sufficient to determine the correct change in path $\mathcal{P}$. An intelligent choice of marking probabilities can significantly reduce the number of marked packets required for the incremental traceback algorithm based on randomized path encoding. The basic idea is to make F_1 converge to a constant as $d \to \infty$. Thinking of the marking probabilities as functions of d ($q_i = q_i(d)$), a necessary and sufficient condition for this to happen is that $\sum_i q_i < \infty$, i.e., the marking probabilities decrease along path $\mathcal{P}$. Two such schemes are described in [9] which give a requirement of an average of $O(\log d)$ marked packets.

VI. Traceback with Network Coding

In this section, we extend traceback to networks that deploy network coding. The main difference between traceback in conventional networks and that in networks deploy network coding is that in the former, any packet traverses a *path* from source to destination, whereas in the latter, a packet is now a combination of different packets and hence traverses a *subgraph* (with potentially multiple sources) to reach the destination. In such a network, traceback can be performed if every node that performs coding randomly chooses a particular path to encode. Thus, a subgraph can be traced by tracing each of the individual paths that comprise it.

Specifically, let (x_i, y_i), $i = 1, 2, \ldots, m$, be the value-pairs received by node A from other nodes. Then A chooses one of the value-pairs with some probability, say (x_i, y_i), and updates it using its own ID a, to get (x_i, y_i'), where $y_i' \leftarrow y_i \cdot x_i + a$. To ensure that the same path is not chosen every time, node A may change the probability of selection in every time-slot. When the chosen value-pair is received by the other nodes, the same policy as traditional marking is followed. Thus, a destination receives multiple types of value-pairs corresponding to the multiple paths in a subgraph, with a value identifying the encoded path (so that decoding can be performed). Note that incremental traceback can be performed for each path and hence for the subgraph.

A. Faulty/Malicious Nodes in Network-Coded Systems

As described above, a receiver in a network-coded system traces a subgraph instead of a path traversed by a packet. Here, we describe an approach to identify a malicious or faulty node in such a network. We restrict our attention to the case in which a single node in the network is faulty or malicious; this approach can be extended easily to the more general case.

The broad idea is that routing can be performed in such a way that the subgraph traversed by packets from a set of sources to a given destination evolves over time. More precisely, if at time t_1, the subgraph G_1 traversed by packets originating at sources S_1 and S_2 and ending at a destination D is *different* from the subgraph G_2 traversed between sources S_1, S_2 and destination D at time t_2, then the intersection of G_1 and G_2 is *small*. So, if this subgraph evolves so that it is different at different time-slots, then for each time-slot that decoding fails (due to some node in the subgraph being malicious or faulty), the subgraph traversed during that time-slot can be isolated and intersected with subgraphs of other such time-slots (when decoding failed). This will enable the receiver to identify a small set of nodes (in the intersection) as candidates for the malfunctioning node.

The subgraph creation needs to be done carefully, so that every k subgraphs (for some chosen k) have a nonempty but not too large intersection. We defer the details of such a construction to a future version of the paper.

VII. Conclusion and Remarks

We present a way of performing incremental algebraic traceback in a network whose topology changes at a slower time-scale compared to the rate of data exchange and consider an extension to network coding. Note that our proof mechanisms closely resemble the random coding proofs for discrete additive memoryless channels. *Algorithms I* through *III* can be viewed as "achievability" proofs while, in this case, the converse is straightforward. We also consider the extension of traceback in network-coded systems.

References

[1] A. Belenky and N. Ansari, "On IP Traceback," *IEEE Communications Magazine*, Vol. 41, Issue 7, pp. 142-153, July 2003.

[2] H. Burch and B. Cheswick, "Tracing Anonymous Packets to their Approximate Source," Unpublished paper, Dec. 1999.

[3] I.Y. Kim and K.C. Kim, "A Resource-Efficient IP Traceback Technique for Mobile Ad-hoc Networks Based on Time-Tagged Bloom Filter," *ACM International Conference on Convergence and Hybrid Information Technology (ICCIT)*, Vol. 2, pp. 549-554, 2008.

[4] D. Song and A. Perrig, "Advanced and Authenticated Marking Schemes for IP Traceback," *IEEE INFOCOM*, Vol. 2, pp. 878-886, Apr. 2001.

[5] S.M. Bellovin, M. Leech and T. Taylor, "The ICMP Traceback Message," *http://tools.ietf.org/html/draft-ietf-itrace-04*, Oct. 2001.

[6] S. Savage, D. Wetherall, A. Karlin and T. Anderson, "Practical Network Support for IP Traceback," *ACM SIGCOMM*, Aug. 2000.

[7] A.C. Snoeren, C. Partridge, L.A. Sanchez, C.E. Jones, F. Tchakountio, S.T. Kent, and W.T. Strayer, "Hash-Based IP Traceback," *IEEE/ACM Transactions on Networking (TON)*, Vol. 10, Issue 6, Dec. 2002.

[8] D. Dean, M. Franklin and A. Stubblefield, "An Algebraic Approach to IP Traceback," *ACM Transactions on Information and System Security (TISSEC)*, Vol. 5, Issue 2, pp. 119-137, May 2002.

[9] A. K. Das, S. Agrawal and S. Vishwanath, "On Algebraic Traceback in Dynamic Networks," arXiv:0908.0078v3.