Topology Inference using Network Coding

Christina Fragouli, Athina Markopoulou and Suhas Diggavi

***Abstract*— The network coding paradigm is based on the idea that independent information flows can be linearly combined throughout the network to give benefits in terms of throughput, complexity etc. In this paper, we explore the application of the network coding paradigm to topology inference. Our goal is to infer the topology of a network by sending probes between multiple sources and receivers at the edge of the network, while intermediate nodes locally combine incoming probes before forwarding them. In previous tomography work, the correlation between the observed packet loss patterns has been used to infer the underlying topology. In contrast, our main idea behind using network coding is to introduce correlations among probe packets in a topology dependent manner and also develop algorithms that take advantage of these correlations to infer the network topology from end-host observations. Preliminary simulations illustrate the performance benefits of this approach. In particular, in the absence of packet loss, we can deterministically infer the topology, with very few probes; in the presence of packet loss, we can rapidly infer topology, even at very small loss rates (which was not the case in previous tomography techniques).**

I. Introduction

The seminal work in [1], [2] showed that for multicast networks, if intermediate nodes can do simple local XOR-operations on packets coming on its incoming links, then one can achieve the min-cut of the network to each receiver. These linearly combined packets can then be utilized at the end receivers to recover the original information symbols by solving a set of linear equations over the finite field [3]. This breakthrough idea has spawned a significant effort in applying network coding to other network topologies, developing practical algorithms that achieve this performance as well as quantifying the throughput benefits of network coding. Many implementations (such as the Microsoft Avalanche project [4], [5]) use random linear combinations for decentralized operations that guarantee such recovery with high probability. In terms of applications, the network coding idea is well-matched to content distribution over peer-to-peer networks as seen by ongoing projects for this application.

Motivated by the fact that, in the future, network coding can be deployed in large scale networks, we explore how we can utilize it for tomographic applications such as topology inference. We explore this idea in the context of overlay networks [6], since (i) topology inference and performance monitoring [7] are of particular importance for overlay routing and (ii) network coding could be deployed incrementally on overlay nodes (rather than at the routers). However, our approach is applicable to any network where network coding is deployed. In this paper, we offer a first approach on how to use network coding to improve network tomography.

The main insight in utilizing network coding for topology discovery is that when we do local XOR-operations, the observations seen at the end-hosts depend on the network topology. Therefore, we can develop algorithms that utilize this to infer the network topology *deterministically* without any further active participation by the internal nodes. In the presence of packet losses, just one successfully received probe per network path is sufficient, without the need to collect packet loss statistics. This property enables rapid discovery of the underlying topology. Moreover, these ideas can be further combined with characteristics of packet loss patterns for highly lossy networks.

In this paper, we formulate the problem of topology inference for a network where (i) multiple sources and receivers are used at the edge and (ii) network coding can be used in intermediate nodes. In Section 3, we propose three algorithms for solving the problem depending on the scenario. In Section 3.1, we develop an algorithm that *deterministically* discovers the topology, in one pass, for tree networks without packet losses. In Section III-B, we show that even in the of presence

This work was supported by the Fonds National Suisse Award No. 200021-103836/1 (for C.Fragouli) and by the SNSF supported center on wireless sensor networks (for S.Diggavi).

C. Fragouli is with the School of Computer and Communication Sciences, EPFL, Lausanne, Switzerland, christina.fragouli@epfl.ch

A. Markopoulou is with the EECS Department., University of California at Irvine, USA, athina@uci.edu

S. Diggavi is with the School of Computer and Communication Sciences, EPFL, Lausanne, Switzerland, suhas.diggavi@epfl.ch

of packet losses, this technique allows rapid discovery of the underlying topology, after only a few probes. In Section III-C, we look at the special case of a single receiver, where the network coding strategy develops a sink tree (reverse multicast tree). In Section IV, we apply our techniques to an example tree with packet loss, and we demonstrate that we correctly infer the correct topology with high success probability and fast convergence. In Section II, we discuss related work and compare our approach to other topology inference techniques. In Section V, we conclude with a short discussion about ongoing extensions of these ideas.

II. Related work and positioning

Over the past decade significant developments have been made in topology inference using only measurements at the network edge. The insight behind the algorithm proposed in [8] was that the correlation between end-to-end (multicast) packet loss patterns can be used to infer the network topology. If a pair of nodes have a significant overlap of packet loss and success patterns, then they should have a common parent. Therefore, by clustering such nodes, one can infer the topology for a binary tree network. The correctness of this idea was rigorously established in [9] and this framework was extended to more general trees and to other measurements such as delay variance etc. The ideas were then extended to unicast networks by [10], [11], [16]. All these techniques relied on either packet loss (and success) patterns or other measurements with "monotonic" properties which grew with number of traversed links, combined with the correlation structure imposed by a multicast tree. Finally, tomographic approaches for inferring the link characteristics [14] can be combined with topology inference [16].

The basic idea of network coding was proposed in [1], [2]. In the context of network tomography, network coding ideas have been explored to infer link loss rate for known topologies. Active techniques have been proposed in [12] and passive techniques have been explored in [13]. These ideas demonstrated that one can decrease the bandwidth used by probes, improve the accuracy of estimation, and decrease the complexity of selecting paths or trees to send probes.

A. Cost-benefit analysis

Since intermediate nodes need to be equipped with additional functionality (more than packet forwarding), one natural question that needs to be answered is the requirements and the benefits of using such an approach.

We envisage our approach to be primarily used in overlay networks where nodes could already have network coding capabilities. The main argument is that since the linear combining functionality could be widely deployed for data delivery using network coding [4], [5], we can further utilize this functionality for tomographic applications. These techniques can be incrementally deployed in overlay networks, where both topology inference is important and where operations can be done in the application layer. For such overlay networks, in terms of requirements, the complexity of the local XOR-operations required is not much more than packet forwarding or multicasting. Since the operations are local, the intermediate nodes do not need to forward statistics or connectivity information. Moreover, the intermediate nodes do not have to reveal *any* identity information since they just forward linear packet combinations, and hence there are no issues of security which might make methods that use node identity tags less attractive. Also, as it will be seen in Section IV, the network coding approach allows for rapid topology discovery, i.e. using very few probes, as compared to methods based on measuring monotonic properties. We therefore believe that in infrastructures where the network coding functionalities are already deployed, the cost-benefit trade-off of our proposed approach for topology inference is quite attractive.

III. Algorithms for topology inference

Consider a tree graph $G = (V, E)$ with $n = |V|$ nodes. A tree with n nodes has exactly $n - 1$ edges, and there exists exactly one path that connects any two vertices. For simplicity, we are going to restrict our attention to algorithms that infer the topology of binary trees. These are trees with two type of vertices: leaf-vertices, that have degree one, and intermediate vertices that have degree three. Our algorithms directly extend to the case of trees where intermediate nodes have an arbitrary degree, by using for example operations over finite fields; we defer the detailed description of such algorithms to a full version of the paper.

We assume that the network can be represented as an undirected tree, in the sense that each edge can be used in either direction, and the connection between any two vertices is reciprocal. We will also denote by $\mathcal{L} = \{1, 2, ..., L\}$ the leaf-vertices (leaves) of the tree which correspond to end-hosts, that can act as sources or receivers of probe packets.

Our algorithms proceed in iterations, where in each iteration a different set of leaves act as sources and as

receivers of probe packets. The basic idea is that each iteration successively divides the leaves in the network into groups, and reveals how the groups are connected to each other. So in a sense, our approach is a "center to leaves" approach for revealing the tree structure.

We will discuss three type of algorithms: in Section III-A we assume that there are no link losses in the network and give a *deterministic* topology inference technique. In Section III-B and III-C, we give two algorithms when there are link losses.

A. Lossless Binary Tree

The following example illustrates the basic idea of our algorithm.

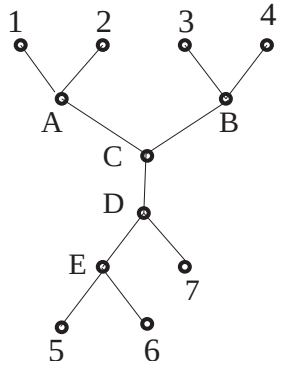


Fig. 1. A network topology that is an undirected binary tree with seven leaves and five intermediate nodes.

Example 1. Consider the network in Figure 1. Assume that nodes 1 and 7 act as sources S_1 and S_2 of probe packets, while the rest of the nodes act as receivers of probe packets. Thus, nodes 1 and 7 send probe packets $x_1 = [1\ 0]$ and $x_2 = [0\ 1]$ respectively. Node A receives packet x_1, duplicates it and forwards it to leaf 2 and to node C. Similarly, node D receives packet x_2, duplicates it and forwards it to node E which in turn forwards it to leaves $5, 6$. Probe packets x_1 and x_2 arrive (within a predetermined time window) to node C. Node C creates the packet $x_3 = x_1 \oplus x_2 = [1\ 1]$ and forwards x_3 to node B which in turn forwards it to leaves $3, 4$.[1]

As a result, leaf 2 will receive packet x_1, leaves $5, 6$ will receive packet x_2 and leaves $3, 4$ will receive packet $x_3 = x_1 \oplus x_2$. Thus our tree will be divided into three areas, $\mathcal{L}_1$ containing S_1 and the leaves that received probe packet x_1 (in total, $\mathcal{L}_1 = \{1, 2\}$). Similarly $\mathcal{L}_2 = \{5, 6, 7\}$ are nodes containing S_2 and the leaves that received probe packet x_2 and $\mathcal{L}_3 = \{3, 4\}$ containing the leaves that received probe packets $x_1 \oplus x_2$. From this information, observed at the edge of the network, we can deduce that the tree will have the structure depicted in Figure 2.

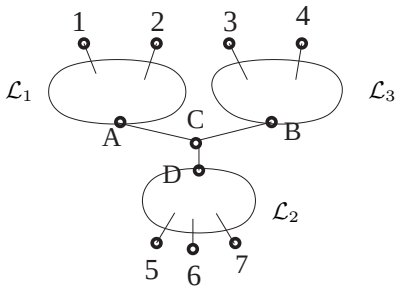


Fig. 2. Structure revealed after one iteration.

To infer the structure that connects leaves $\{5, 6, 7\}$ to node C, we need to perform a second experiment, where we now randomly choose two of these three leaves to act as sources of probe packets. For example, assume that nodes 5 and 6 act as sources S_1 and S_2 of probe packets. Note that any probe packet leaving node D will be multicast to all the remaining leaves of the network, *i.e.*, nodes $\{1, 2, 3, 4\}$ will observe the same packet. Thus in this sense we can think of node D as a single "aggregate–receiver" for this second experiment, that will observe the common packet received at nodes $\{1, 2, 3, 4\}$. Following the same procedure as before, assuming that packets x_1 and x_2 meet at node E, receivers 7 and $\{1, 2, 3, 4\}$ receive packet $x_3 = x_1 \oplus x_2$. Using this additional information we refine the inferred network structure as depicted in Figure 3. Since the

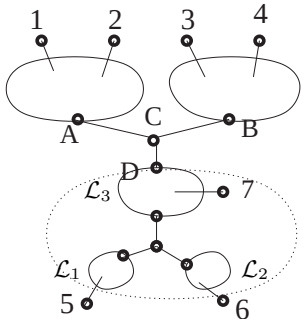
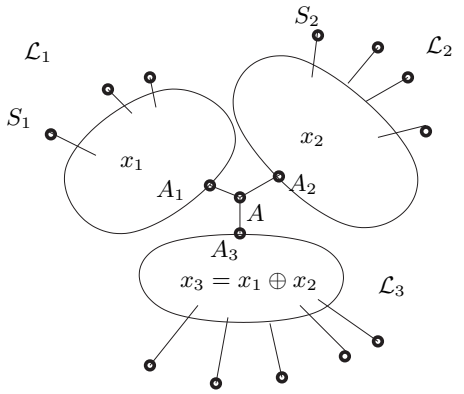


Fig. 3. Structure revealed after two iterations.

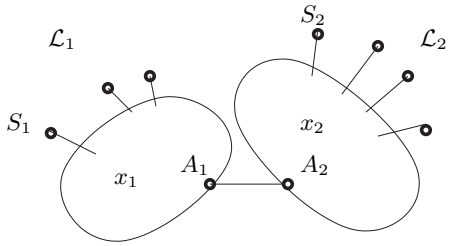
tree is binary, we can deduce from Figure 3 the overall topology of Figure 1. ■

[1]Note that we have chosen the directionality of the edges depending on which source reaches the vertex first. If there is variable delay, then the vertex where the packets x_1, x_2 meet could be different, but this does not affect the algorithm as we will discuss in Theorem 1.

The basic ingredients of our algorithm are already described in the previous example. Let us now describe the general algorithm for arbitrary binary trees, summarized in Algorithm 1 and shown in Figure 4.



(a) Dividing into three components.



(b) Dividing into two components.

Fig. 4. Edges and vertices of the graph, as revealed by a single iteration of Algorithm 1.

The algorithm proceeds in iterations. Each iteration is concerned with a part of the binary tree, of which we want to infer the topology. Let $\mathcal{L}$ be the set of leaves in (that part of) the tree. In each iteration, exactly two leaves, out of the set $\mathcal{L}$, are randomly chosen and act as sources (S_1 and S_2) sending probe packets (x_1 and x_2 respectively). All remaining nodes in $\mathcal{L}$ act as receivers. Intermediate nodes that receive one probe packet (either x_1 or x_2) simply forward it to all outgoing links. Intermediate nodes that receive both probe packets XOR (linearly combine) them and forward $x_1 \oplus x_2$ to the outgoing link. Probe packets go through the network, either forwarded and/or being linearly combined, and eventually reach the receivers. Each receiver observes one probe packet, either x_1, or x_2 or $x_1 \oplus x_2$. The leaves in $\mathcal{L}$ are therefore divided into three sets $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{L}_3$, depending on whether they observed x_1, x_2 or $x_1 \oplus x_2$ respectively.

The algorithm starts by considering $\mathcal{L}$ to be the leaves of the entire tree. At each iteration, it partitions the leaves of the tree into the three areas $\mathcal{L}_1$, $\mathcal{L}_2$, $\mathcal{L}_3$. The algorithm proceeds iteratively within each area until all edges are revealed.

Algorithm 1 Topology Inference for Lossless Tree

- *Iteration* 1: Consider the set $\mathcal{L}$ of all leaves.
  - Randomly choose two leaves to act as the sources S_1, S_2, sending probes x_1, x_2 respectively.
  - All other leaves $\mathcal{L} - \{S_1, S_2\}$ act as receivers. Observe the first packet each one receives and partition $\mathcal{L}$ into $\mathcal{L}_1 \cup \mathcal{L}_2 \cup \mathcal{L}_3$ as follows. Set $\mathcal{L}_1$ contains the source S_1 and all receivers that observe x_1. Set $\mathcal{L}_2$ contains the source S_2 and all receivers that observe x_2. Set $\mathcal{L}_3$ contains all receivers that observe $x_3 = x_1 \oplus x_2$.
  - If $\mathcal{L}_3$ is not empty, replace the original graph with the three components $\mathcal{L}_1$, $\mathcal{L}_2$, $\mathcal{L}_3$, connected through three edges and four vertices (where component $\mathcal{L}_i$ is connected through node A_i) as depicted in Figure 4(a). If the set $\mathcal{L}_3$ is empty, replace the original graph with two components $\mathcal{L}_1$ and $\mathcal{L}_2$, connected through a single edge as depicted in Figure 4(b).
  - If component $\mathcal{L}_i$ contains one or two leaves, replace the component with either one or two edges, connecting the leaves through node A_i to the rest of the network. If component $\mathcal{L}_i$ contains three or more leaves, iteratively reveal the structure inside the component at an iteration i.
- *Iteration* i: Consider one of the previously identified components $\mathcal{L}_i$ and repeat.
  As before, two (randomly chosen) leaves in $\mathcal{L}_i$ act as sources S_1 and S_2 and all remaining nodes in $\mathcal{L}_i$ act as receivers. Node A_i that connects $\mathcal{L}_i$ to the network will also act as an aggregate receiver: whatever packet is received by A_i will be multicasted and received by all leaves in $\mathcal{L}$ that are not in $\mathcal{L}_i$. Repeat the exact same procedure as in iteration 0 to reveal the structure of component $\mathcal{L}_i$. Connect the component to the network depending on what packet is received by A_i.
- Continue until all edges and vertices are identified.
- Remove vertices of degree two.

Algorithm 1 requires the intermediate node of the network to operate as follows. These are standard functionalities in networks that support network coding.

Intermediate Node Operation: If, within a predetermined time window W, an intermediate node receives a single probe packet from one of its adjacent neighbors, it replicates it, and forwards it to its other two neighbors. If it receives two packets from two different neighbors within W it XORs them, and forward the resulting packet to the remaining neighbor.

Theorem 1: Algorithm 1 terminates in less than $|\mathcal{L}|$ iterations, and exactly infers the binary tree topology.

Proof Outline

Consider at a particular iteration the sources S_1 and S_2. During this iteration, exactly one probe packet will be forwarded from each source towards all other leaves in the network. Each probe packet will traverse the undirected links in a source to receivers direction.

Consider now the intermediate nodes in the path $\mathcal{P}$ that connects the two sources. Depending on the delay associated with the links of the network, there exist two possibilities:

• The probe packets x_1 and x_2 meet (arrive within the same time-window W) at any of the internal nodes on path $\mathcal{P}$, say node A. Node A then forwards their XOR to its third link, and the iteration "reveals" the neighboring edges and vertices to A as depicted in the configuration in Figure 4(a). Note that, for the purpose of the algorithm, it plays no role at which vertex A the probe packets meet.

• An alternative possibility is that packets x_1 and x_2 "cross each other" while traversing the same link of $\mathcal{P}$ in opposite directions, *i.e.,* they do not meet at a node. As a result, given the prescribed operation of intermediate nodes, leaves in the network may receive more than one probe packets, of which they only keep the first received. In this case we infer the configuration in Figure 4(b) that reveals one edge.

In any case at each iteration the leaves of the network will be divided into two or three more components. Once a component has two or less leaves, and since we have a binary tree, we know its structure. □

Note that inferring the binary tree topology without any error requires to send at most $|\mathcal{L}|$ times two probe packets through the tree. Also note that since each link will be traversed exactly once at each iteration by a useful probe packet, delay variations along links of the network do not affect our algorithm.

B. Lossy Binary Tree

In this section, we consider trees with packet loss, i.e. a probe packet might be lost while traversing a link with a certain probability. This may have a detrimental effect on our algorithm. Recall that in the lossless case, at a given iteration, since there exist only one probe packet generated by each source, the probe packets can at most meet at one intermediate node as described in the previous section, and delay variability along network links plays no role. However, when the links are lossy, we need to send more than one probes during each iteration as we discuss next. Given packet losses and delay variability, this might result in probes meeting at different nodes during the same iteration, causing confusion when dividing the receivers into components.

This problem is effectively created by the fact that we deal with undirected graphs, where a link may be traversed in opposite directions by probe packets during the same iteration. Thus, an easy method to avoid this problem, is to fix the directionality of the tree edges during each iteration. This can be achieved in a completely distributed manner by the first packet arriving at each intermediate node as described in the following.

Intermediate Node Operation: Each intermediate node keeps a table of its neighbors. In each iteration, it will mark these neighbors as "source" neighbors or "sink" neighbors.[2] The first time during an iteration that an intermediate node receives a probe packet[3], the node waits for a window W to receive probe packets from another of its neighbors. After this window W passes, the node marks all neighbors from which it received packets as sources and all other neighbors as sinks. For the remaining duration of the iteration, the source accepts packets only if they originate from its source neighbors. If an intermediate node within a time-window W receives a packet from one of its adjacent source neighbors, it replicates it, and forwards the same packet to all its sink neighbors. If it receives more than one packets from two different source neighbors, it linearly combines them, and forwards it to all its sink neighbors. The node rejects probe packets coming from sinks, and does not forward packets towards sources.

We can now extend Algorithm 1 to Algorithm 2, so as to operate over lossy networks. The only difference is that, in each iteration, we send M instead of one probe packets from each of the sources.

[2]Once this marking is done, it does not change for the duration of the iteration. It may however change for the next iteration.

[3]We might use a special type of probe packet, noting the beginning of an iteration.

Algorithm 2 Topology Inference for Lossy Binary Tree

- If a receiver receives only x_1, then assign it to the set $\mathcal{L}_1$.
- If a receiver receives only x_2, then assign it to the set $\mathcal{L}_2$.
- If a receiver receives both x_1 and x_2, or it receives an $x_1 \oplus x_2$ packet, then assign it to set $\mathcal{L}_3$.
- If a node does not receive anything, then randomly assign it to one of the components.
- For aggregate receiver nodes (A_i), apply the same rule using the union of the aggregate receiver observations.

Clearly Algorithm 2 has an associated probability of error, due to the fact that a leaf might not receive the "correct" probe packet. For example, in a given iteration we might make an error either because a node does not receive any probe packet (which can be made arbitrarily small by increasing the number of probe packets M) or, because it belongs in $\mathcal{L}_3$ but happens to receive only x_1 or only x_2 packets. This probability again decreases very fast as M increases, as we will see in Section IV. We note that, the number of probe packets M we need to send to infer the topology within a given probability of error, is much smaller than the number of probe packets required by the inference methods in the literature. Our algorithm will operate correctly if each node receives exactly *one* probe packet from each of the sources it is connected to.[4] Nodes in $\mathcal{L}_1$ and $\mathcal{L}_2$ are connected to one source while nodes in $\mathcal{L}_3$ to two sources. The inference methods on the contrary require the reception of sufficient packets to accurately estimate probability distributions.

C. Inverse Multicast for Lossy Trees

In Sections III-A and III-B we argued that allowing intermediate nodes in the network to XOR incoming probe packets can significantly reduce the number of probes require to infer the network topology, *i.e.,* improve the bandwidth efficiency. In this section, we argue that similar ideas can be used to offer benefits towards a different goal, locality of operations.

In particular, all tree topology inference methods proposed in the literature that employ multicast trees require that the set of observations is collected from all the leaves of the trees and processed by a centralized processor that will then disseminate the topology information back to the tree leaves. However, in applications such as peer-to-peer networks, where the tree might have an arbitrarily large number of leaves, and where not all leaves might be interested in acquiring topological information, it is clearly desirable to have algorithms where an end terminal can accurately infer the overall network topology by processing exclusively its own observations.

The following algorithm falls in this category. Our basic observation is that, we can use all the algorithms in the literature for tree topology inference by sending probe packets over an inverse multicast tree with a single receiver, and mapping the information collected by the receiver to measurements over a multicast tree, without any loss of estimation accuracy.

Algorithm 3 Inverse Multicast Algorithm

Let $\mathcal{L}$ denote the set of leaves of the binary tree.

- Select one leaf to act as receiver of probe packets. The remaining $\mathcal{L}-1$ nodes act as sources of probe packets.
- Source S_i sends M times the probe packet x_i, that consists of $\mathcal{L}-2$ zeroes and one 1 at position i.
- Intermediate nodes in the tree XOR their incoming probe packets and forward it to the outgoing link that leads to the receiver node.

Theorem 2: Consider an undirected tree topology G. Algorithm 3 allows to infer the tree topology, using an inverse multicast tree over G, with exactly the same accuracy, as achieved using a multicast tree over G.

In the inverse multicast tree the single receiver node infers information not only from the number but also from the content of its received probe packets. The proof of this theorem relies on the fact that, the error events experienced on an inverse multicast tree with a single receiver are in a one-to-one correspondence with the error events experiences on a multicast tree with a single source (coinciding with the single receiver of the previous case). The proof of this result has partly appeared in [12].

IV. Preliminary Simulation Results

In this section we simulate our algorithms for the Example 1, shown in Figure 1 and discussed in Section 3. As discussed in Example 1, two iterations are sufficient to infer the topology of this network: In the first iteration, leaves $S_1 = 1$ and $S_2 = 7$ act as sources, we assume that packets meet at node C, and the network gets partitioned into three components $\mathcal{L}_1 = \{1, 2\}$,

[4]This equals the success probability of a geometric distribution.

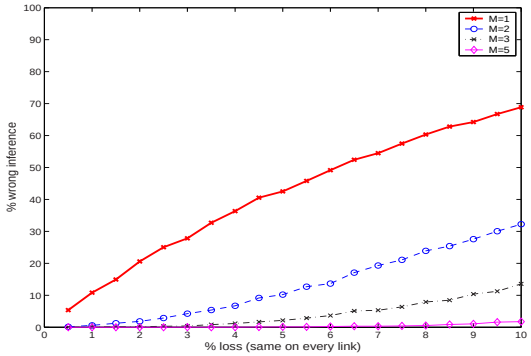
$\mathcal{L}_2 = \{5, 6, 7\}$, $\mathcal{L}_3 = \{3, 4\}$, as shown in Figure 2. Components $\mathcal{L}_1$ and $\mathcal{L}_3$ need no further investigation. In the second iteration, leaves $S_1 = 5$ and $S_2 = 6$ act as sources and we assume that the probe packets meet at node E which reveals the structure shown in Figure 3 and completes the topology inference.

In the case that there is no packet loss, we infer the topology deterministically in two iterations, and with only one set of probes per iteration (Algorithm 1). In the case that the links are lossy, we still perform the same two iterations, but we send M sets of probes per iteration (Algorithm 2). In the lossy case, it is possible that we make an error in inferring the topology. The probability of error is increasing with the loss rates of the links p, and decreasing with the number of probes M per iteration.

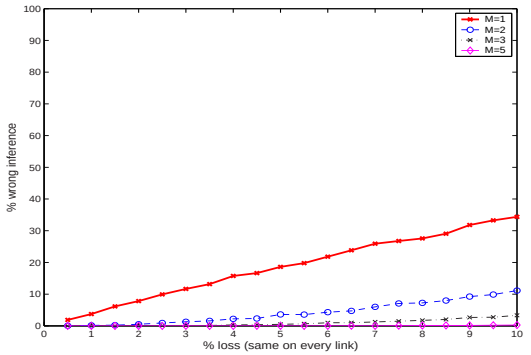
Figure 5 plots the percentage of inference errors in each of the two iterations as a function of p and M. For this set of simulations, we assume that all links have the same loss probability p. We considered values of $p \in [0, 10\%]$ and $M = 1, ...10$. We consider an error to be any divergence from the true topology; in a future stage we plan to consider metrics that capture the distance between the real and the inferred topology. The results shown in Figure 5 are averaged over $10,000$ instantiations of the loss process.

The following observations can be made from these graphs. First, as expected, the probability of incorrect inference is indeed increasing with p, since packet losses may lead to the misclassification of a leaf to the incorrect component. Note also that, for a fixed number of probe packets and loss rate p, the error probability varies with the iterations, and diminishes as the size of the inferred network also decreases. Second, the probability of incorrect inference is decreasing very rapidly with M: having 2 to 3 probes per iteration significantly decreases the probability of error, even for large p. The probability of error was practically zero for more than 5 probes per iteration in our simulation.

It is important to note that this second property is due to the fact that any *one* correctly received packet is sufficient for the correct operation of Algorithm 2. For example, if a node receives a mixture of x_1 and x_2, it will be correctly assigned to component $\mathcal{L}_3$ even if several probes are lost. In contrast, the methods in the tomography literature require each receiver to receive enough probe packets to infer the probability of link loss rate associated with the network links with a certain accuracy, which requires a much larger number of probe packets.



(a) Iteration 1: Receivers 1 and 7 act as sources. The iteration infers the topology shown in Figure 2.



(b) Iteration 2: Receivers 5 and 6 act as sources. The iteration infers the topology shown in Figure 3.

Fig. 5. Probability of incorrect inference as a function of the link loss probability p (same for all links) and the number of probes M in the iteration.

V. Discussion

Although this is the first paper that makes the connection between network coding and topology inference, this observation is in fact not surprising: by combining the incoming information flows, the intermediate nodes inherently reveal information about the network structure. Actively utilizing this property using probe packets is a natural step, given this realization.

The preliminary ideas presented are currently being extended in several directions. The presented algorithms do not fully exploit the information from lossy measurements: to do so, we need algorithms that exploit both the correlation introduced by link losses and network coding. Another direction we are exploring is extending the algorithms proposed in this paper to arbitrary network topologies. Finally, we are exploring the use of passive measurements to infer the topology in situations where network coding is already deployed.

REFERENCES

[1] R. Ahlswede, N. Cai, S-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, Vol. 46, pp. 1204-1216, July 2000.

[2] S-Y. R. Li, R.W. Yeung, and N. Cai, "Linear network coding," *IEEE Trans. on Information Theory*, Vol. 49, 2003.

[3] C. Fragouli, J. Widmer and J. Y. LeBoudec, "Network coding: an instant primer", *ACM SIGCOM Computer Communication Review*, January 2006.

[4] C. Gkantsidis and P. Rodriguez, "Network coding for large scale content distribution", Infocom March 2005.

[5] "Avalanche: File Swarming with Network Coding", *http://research.microsoft.com/ pablo/avalanche.aspx*

[6] D. Andersen, H. Balakrishnan, F. Kaashoek and R. Morris, "Resilient overlay networks," *in Proc. of* 18^{th} *ACM SOSP*, Canada, Oct. 2001.

[7] Y. Chen, D. Bindel, H.Song and R.Katz, "An algebraic approach to practical and scalable overlay network monitoring," *in Proc. ACM SIGCOMM 2004*.

[8] S. Ratnasamy and S. McCanne, "Inference of multicast routing trees and bottleneck bandwidths using end-to-end measurements", Infocom New York 1999.

[9] N.G. Duffield, J. Horowitz, F. Lo Presti, and D. Towsley, "Multicast topology inference from measured end-to-end loss", *IEEE/ACM Transactions on Networking*, 2002.

[10] K. Harfoush, A. Bestavros and J. Byers, "Robust Identification of Shared Losses Using End-to-End Unicast Probes," *in Proc. of ICNP 2000*,

[11] M. Coates, R. Castro, R. Nowak and Y. Tsang, "Maximum Likelihood Network Topology Identification from Edge-Based Unicast Measurements", *in Proc. ACM Sigmetrics 2002*.

[12] C. Fragouli and A. Markopoulou, "Network coding for network tomography", *Allerton*, 2005.

[13] T .Ho, B. Leong, Y. Chang, Y. Wen and R. Koetter, "Network monitoring in multicast networks using network coding", International Symposium on Information Theory (ISIT) 2005.

[14] R. Caceres, N. G. Duffield, J. Horowitz and D. Towsley, "Multicast-based inference of network-internal loss characteristics", IEEE Trans. in Inf. Theory, vol. 45, pp. 2462–2480, 1999.

[15] M. Adler, T. Bu, R. K. Sitaraman and D. Towsley, "Tree layout for internal network characterizations in multicast networks," *in Proc. of ACM NGC 2001*, Nov. 2001.

[16] M. Rabbat, R. Nowak and M. Coates, "Multiple source, multiple destination network tomography", *in Proc. of IEEE Infocom 2004*.

[17] T.S.E. Ng and H.Zhang, "Predicting internet network distance with cordinated-based approaches," *in Proc. of IEEE INFOCOM 2002*.

[18] Y.Zhu, B. Li, J. Guo, "Multicast with network coding in application-layer overlay networks," *in IEEE JSAC, Special Issue on Service Overlay Networks*, 4^{th} Quarter, 2003.

[19] "The network coding webpage," *http://www.netcod.org*.