Passive Loss Inference in Wireless Sensor Networks Based on Network Coding

Yunfeng Lin, Ben Liang, Baochun Li
Department of Electrical and Computer Engineering
University of Toronto
{*ylin, bli*}*@eecg.toronto.edu, liang@comm.toronto.edu*

***Abstract*—The highly stochastic nature of wireless environments makes it desirable to monitor link loss rates in wireless sensor networks. In this paper, we study the loss inference problem in sensor networks with network coding. Unlike traditional transmission protocols, network coding offers reliable communication without using control messages for individual packets. We show, however, that network coding changes the fundamental connection between path and link loss probabilities such that new inference algorithms need to be developed. As end-to-end data are not sufficient to compute link loss rates precisely, we propose inference algorithms based on Bayesian principles to discover the set of highly lossy links in sensor networks. We show that our algorithms achieve high detection and low false-positive rates through extensive simulations.**

I. Introduction

Recent technological advances have made it feasible to deploy large-scale sensor networks using low-cost sensor nodes. However, as the scale of sensor networks becomes larger, two key challenges potentially arise. First, *node failures*. Due to their inherent instability and energy constraints, sensor nodes are prone to failures. It would thus be useful to determine which set of nodes or which geographical areas within the network are experiencing high loss rates. Such information is potentially valuable to the design of fault-tolerant protocols or monitoring mechanisms, so that the problem areas may be re-deployed, and critical data may be rerouted to avoid these failure-prone areas suffering high loss rates. Second, *bandwidth constraints*. One cannot rely on the use of active acknowledgments, which are neither scalable nor bandwidth-efficient, in the design of sensor network protocols. This renders the direct collection of loss rate data impossible in sensor networks. Furthermore, it would also be infeasible, due to limited bandwidth, for individual sensor nodes to collect and transmit loss rate data to a centralized location for processing in large-scale sensor networks.

The research on network coding has received tremendous amount of attention in recent years. By allowing intermediate nodes to perform coding operations besides simple replication and forwarding, it has been shown that network coding [1] achieves the network multicast capacity. Furthermore, network coding is found to be effective and helpful in lossy wireless networks, as it naturally offers error recovery and reliable communication due to its root from erasure coding [2]. In addition, network coding is also able to utilize the wireless broadcast advantage while significantly simplifying protocol design [3]. Finally, network coding can compress spatially correlated sensing data in a distributed fashion [4].

On one hand, there exists the need of identifying specific geographical areas where high loss rates are experienced in large-scale sensor networks. On the other hand, though bandwidth constraints prevent centralized loss data collection and processing, network coding offers inherent resilience to losses in sensor networks in a decentralized fashion [2], [3]. Would it be feasible to design new practical algorithms when network coding is used for data flows in wireless sensor networks, such that areas with high loss rates are *inferred* without centralized collection? In this paper, we study the problem of efficiently determining highly lossy links in wireless sensor networks by *passively* monitoring network coding traffic at the sink.

We believe that existing loss inference solutions in the literature are not effective in the context of network coding. In traditional loss inference problems considered in IP networks and sensor networks, either end-to-end retransmission (TCP) is employed to ensure reliable communication [5], or no reliability [6]–[8] is provided in order to avoid the overhead of acknowledgment messages for individual packets. All these inference problems use the following fundamental model between end-to-end observation and network characteristics: the path successful-transmission probability is the product of all link successful-transmission probabilities, assuming loss events occur independently among links. With network coding, however, this fundamental model is changed. In particular, we show that the path successful-transmission probability is the minimum of all link successful-transmission probabilities on this path.

Such a simple difference between path observation and link characteristics has two important consequences on loss inference. *First*, the most lossy link on a path essentially blocks the information of all other links on the same path. Hence, it is infeasible to determine the exact loss rates of all links. Therefore, we seek to achieve a less ambitious goal, to discover the set of highly lossy links in this paper. *Second*, in the traditional model, even if all links are good, a long path may still have high loss rates due to the accumulated effect of link losses. In fact, it can be shown that it is infeasible to find a path threshold — a threshold value to separate good and bad paths — without both false positives and negatives. However, with data flows using network coding, a high-loss path always implies that there is at least one poor link on the path. This

fact suggests that a natural path threshold is the link threshold itself.

With these new insights in mind, in this paper, we propose Bayesian inference algorithms based on factor graphs [9] and Gibbs sampling [10]. Both algorithms are able to derive the posterior distributions of link rates, given the data observed at the sink. In general, the factor graph based solution offers the complete numerical posterior distributions, whereas Gibbs sampling is only able to generate random samples from these distributions. However, in our inference problem, the factor graph based algorithm suffers from the "the curse of dimensionality" [10], and is hence not suitable for large-scale problems. We also adapt the algorithm in [11], originally proposed for traditional routing protocols, to our new problem setting and show that its performance improves under network coding. Furthermore, this combinatorial algorithm is much more efficient than the two numerical Bayesian inference algorithms. However, it is accurate only if there is a sufficiently large amount of data in the network. In contrast, the Bayesian inference algorithms demonstrate excellent performance even with limited data. We evaluate these algorithms through extensive simulations.

The remainder of the paper is organized as follows. Sec. II discusses related work in network tomography and wireless sensor networks. Sec. III presents the network model that will be used throughout the paper. Sec. IV presents the data collection protocol utilizing network coding in sensor networks and its associated inference problem. Sec. V describes the three inference algorithms to detect the highly lossy links in wireless sensor networks. Sec. VI evaluates the three algorithms under different network settings. Finally, Sec. VII concludes the paper.

II. Related Work

Tomography on the wireline networks has been extensively studied in recent years [12]. Most tomography algorithms infer the network characteristics by utilizing inherent correlation among multicast probing packets [13]. However, since IP multicast is not widely supported by the network, there have been alternative proposals on loss inferences based on sending unicast probing packets, where clever protocols (for example, using back-to-back packets) are incorporated, which essentially turns the inference problem to a multicast problem (see, for example, [14]).

Recently, several studies propose to passively monitor wireless sensor networks through application data in order to avoid active monitoring traffic. [6] and [7] perform loss inference by assuming aggregation of application data packets at each sensor. On the other hand, Nguyen *et al.* [8] use uncorrelated end-to-end data to identify lossy links. These works are not concerned with the benefit and consequence of network coding in tomography.

The literature of network coding tomography starts from [15], where it is shown that the coding coefficients used in randomized network coding can be used to infer different failure patterns in a wireline network with the assumption that all links are lossless. In contrast, we study passive tomography in a wireless sensor network with lossy links.

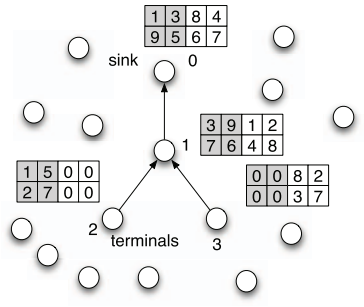


Fig. 1. A simple of example of a sensor network, where node 2 and 3 are terminal nodes, and node 0 is the sink. The squares are the coding coefficients of coded packets. The shaded and empty squares represent the coding coefficients corresponding to the source packets from node 2 and 3, respectively. The segment size K is 2, and the number of terminal nodes n is 2.

Subsequently, the advantages of *active* tomography with network coding have been shown for loss inference in [16]–[18] and topology inference [19] in wireline networks. Furthermore, Jafarisiavoshani, *et al.* utilize the subspace property of randomized network coding to infer topology in wireline networks [20] and to discover bottleneck in a peer-to-peer file sharing application [21]. Most of these tomography techniques actively inject probing traffic [16]–[19] to infer network characteristics such that these solutions are unsuitable for wireless sensor networks with limited radio bandwidth and energy budgets. In addition, the work in [20] assumes global knowledge and lossless environments, which is not the case in our study. The passive inference techniques in [21] are integrated closely with the peer-to-peer protocol used in their system and hence are not applicable to our work.

Shama *et al.* [22] discuss passive *topology* tomography with network coding in both wireline and wireless networks. In contrast, our work proposes algorithms for passive link *loss* tomography, which is fundamentally different from topology inference. To the best of our knowledge, our paper represents the first attempt to address loss inference in wireless networks under network coding traffic.

III. Network Model

We start with a simple example of passive tomography on a sensor network shown in Fig. 1. The terminal nodes, nodes 2 and 3, continuously obtain sensed data from the environment and transmit them through node 1 to the sink, node 0. All wireless links are lossy due to the inherent instability of wireless ratio. By inspecting the received data packets, node 0 would like to infer the loss characteristics of these wireless links. More generally, we consider a sensor network as a directed graph, where each node represents either a terminal (sensor), a relay node, or the sink (data collecting) node, each directed edge represents the link between these nodes, and the direction of an edge indicates the direction of the data flow on the link.

We will not consider, except for the sink, degree-two nodes in the graph; that is, if a degree-two node is not the sink, it is suppressed in the graph representation of the network. In this way, what we refer to as a *link* is not necessarily in its physical sense, since it can be a path consisting of several connected physical links as long as no other paths are branched from an intermediate node in the path. It can be verified that such a notion of links is defined without loss of generality, as far as loss rates are concerned. In this paper, the term *path* refers to a sequence of links starting from a terminal node and ending at the sink.

We assume that packets are transmitted over a link independently of other links. Such an assumption is supported by the measurement studies from [23]. Let α_e denote the successful-transmission probability of link e. $1-\alpha_e$ then refers to the loss probability of this link. We use E and W to denote the set of all links and all paths in a sensor network, respectively. In addition, we assume that all sensors have the same transmission speed. Hence, we can model the packet transmission synchronously, where a time slot represents the time to transmit a data packet. Moreover, as the normalized link rate is equivalent to the successful-transmission probability under this synchronous model, we will use "successful-transmission probability" and "rate" interchangeably throughout the paper.

IV. Network Coding Based Data Collecting Protocol and Inference Problems

In this section, we describe a data collecting protocol utilizing network coding and define its inference problem.

A. Network Coding Based Data Collecting Protocol

In order to show the difference of the inference problems between a protocol with and without network coding, it is useful to briefly review a typical traditional routing protocol, MintRoute [23], used in sensor networks without network coding. MintRoute constructs a reverse multicast rooted at the sink with all terminal nodes as the leaf nodes. Data are transmitted through the paths on the tree from terminal nodes to the sink. With MintRoute, the inference algorithms proposed for sensor networks in [6]–[8] essentially assume no reliability in delivering packets such that a packet is lost if it is lost on one of the links on a path. Assuming independence of loss events among different links, the connection between the path successful-transmission probability β_i and link successful-transmission probabilities α_e is characterized by $\beta_i = \Pi_{e\in P_i}\alpha_e$, where P_i denotes the set of all links on the ith path.

Network coding, however, demonstrates a different connection between path and link transmission probabilities due to its inherent capability of reliable communication. Such an observation will become clear after we present the details of the protocol based on network coding. Note that our objective is not to design the optimal protocol, but a simple one suitable for sensor networks. In particular, similarly to MintRoute, we restrict our attention to a unipath protocol. To utilize the result of previous research efforts, we rely on MintRoute to select the path from a terminal node to the sink. However, we equip the data transmission protocol along a path with network coding.

In the protocol design, we assume that each terminal node produces unlimited data, and partitions its data packets to *segments*, where each segment comprises the same number of data packets[1]. All terminal nodes attach a sequence number to each segment. The protocol then transmits the segments with the same sequence number, say i, from all terminal nodes to the sink until all data in these segments are decoded before the transmission for the next group of segments. Without loss of generality, we only consider the transmission of the segments with sequence number i because the transmission processes are identical for other segments.

We next describe the encoding and decoding algorithms of the random linear codes [24] used to transmit the ith group of segments. Assume that there are n terminal nodes in the network. There are thus Kn source packets in the ith group of segments. We refer to these Kn source packets as a *super segment*. In general, a coded packet x is a linear combination of the Kn source packets $E_1,\ldots,E_{Kn}$ with the form $x=\sum_{i=1}^{Kn}\gamma_i E_i$, where γ_i are coding coefficients chosen randomly from a Galois field. At any time slot, if a node a between a terminal node to the sink wishes to transmit a coded packet, a produces a coded packet x_a by encoding all coded packets in its buffer belonging to the same super segment, namely $x_1,\ldots,x_m$, where m is the total number of coded packets in the buffer that belong to the super segment:

$$x_a=\sum_{i=1}^{m}\eta_i x_i, \tag{1}$$

where all multiplication and addition operations are defined on a Galois field (such as GF(2^8) when the operations are performed on each byte), and η_i is *randomly* chosen from the field. It is easy to see that x_a is also a linear combination of the Kn original packets from terminal nodes, and the coefficients can be derived. Node a then transmits x_a along with its coding coefficients over the original packets to the next hop.

Suppose node b, the next hop of node a, successfully receives the coded packet x_a. It first checks whether x_a is linearly independent with its buffered coded packets within the same super segment. If so, node b inserts x_a into its buffer. Otherwise, x_a is discarded. If node b is the sink, it recovers all Kn source packets in the ith super segment by the following algorithm. Since the coding coefficients and the coded packets are known, each coded packet represents a linear equation with the Kn source packets as unknown variables. Hence, decoding the Kn source packets is equivalent to solving a linear system composed of all coded packets received so far. The *decoding matrix* represents the coefficient matrix of such a linear system. When the rank of the decoding matrix is Kn, the linear system can be solved and the Kn source packets are decoded. Otherwise, there exists linear dependence among

[1]This assumption is made for presentation clarity. It is trivial to extend to the general case where different terminal nodes have segments with different sizes.

the coded packets, so the sink will continue to receive coded packets from its neighbors until all Kn packets are decoded. Gaussian elimination is usually used to solve the linear system.

We remark that when multiple nodes transmit coded packets to a common next hop, all these coded packets are encoded together. For instance, node 1 encodes the data from node 2 and 3 together in Fig. 1. Furthermore, as the result of a reverse routing tree, many coded packets are encoded from only a subset of the source packets. The coding coefficients corresponding to the absent source packets are zero. As an example, the coded packets transmitting from node 2 to node 1 have zero coefficients corresponding to the source packets from node 3.

In this paper, we utilize the *rateless* property of randomized linear codes for a node to decide when to transmit a coded packet. A node always produces a coded packet and transmits it to the next hop until it receives the notification that all data from its upstream nodes in the current super segment has been received at the next hop. Such a protocol enjoys the advantage of being insensitive to the variation of link qualities, and requiring control messages only per segment. This control overhead can be ignored if the segment size is sufficiently large.

B. Inference Problems with Network Coding

In this section, we describe the inference problems associated with network coding traffic. We first show the connection between path and link successful-transmission probabilities, assuming only one data flow is transmitted from a terminal node to the sink in the network.

Proposition 1: Let β and α_e denote the path and link successful-transmission probabilities, respectively. Assuming one flow exists, we have

$$\beta = \min_{e \in P}(\alpha_e), \tag{2}$$

where P is the set comprising all links on this path.

Proof: A wireless link e can be modeled as a binary erasure channel [25], with a capacity α_e. With network coding, the transmission on each hop is equipped with an erasure code, which achieves the link capacity when the code length K is sufficiently large. Furthermore, the capacity of any graph is the mincut [26]. Hence, the capacity of this path P is the minimum capacity of any link: $\min_{e \in P}(\alpha_e)$. Finally, it is easy to see that the path from the terminal node to the sink can also be modeled as a binary erasure channel. Hence, the path successful-transmission probability β is the capacity $\min_{e \in P}(\alpha_e)$ [25]. ■

We remark that although Proposition 1 holds for any erasure codes, network coding enjoys the advantage of significantly shorter transmission delay. This is because with traditional erasure codes, a node needs to receive all data to be encoded together, and decode them before it generates the coded packets for the next hop. In contrast, the recoding ability of network coding enables a node to produce new coded packets for the next hop as soon as the node receives a new coded packet from one of its upstream nodes.

Despite the strong assumption, that only one flow is allowed to transmit, is used in Proposition 1, we will show next that our selected end-to-end observation demonstrates a similar property even when multiple flows share links in the network. Specifically, we divide the decoding matrix to n submatrices such that each individual submatrix, composed of K columns, corresponds to the source packets from one terminal node. For example, in Fig. 1, the shaded and blank columns of the decoding matrix at the sink represent the two submatrices corresponding to the source packets from node 2 and 3, respectively. As coded packets are continuously transmitted from the terminal node to the sink before successful decoding, the ranks of all submatrices reach the segment size K at some time. Apparently, if the path successful-transmission probability of path i is higher than that of path j, the rank of submatrix i reaches K earlier than the time that the rank of submatrix j reaches K. Hence, such relative timing contains the information of the path successful-transmission probabilities, which we use to infer link rates.

More formally, let $\bar{t}_i$ denote the time that the rank of submatrix i reaches K. $\bar{t}_i$ is composed of the time used to traverse path i and the wasted time due to link losses. Hence, if we use d_i to represent the length of the ith path, $t_i = \bar{t}_i - d_i$ is the wasted time due to link losses. We thus use $D = \{t_1, \ldots, t_n\}$, referred to as *full-rank times* hereafter, as inputs into our inference algorithms. We then state the following observation, which is essential to our decision to choose D as the end-to-end input, and will significantly simply the design of Bayesian inference algorithms in Sec. V-A and V-B.

Proposition 2: The ranks of the submatrices increase independently.

Proof: This observation can be easily justified from the matrix form of the encoding algorithm (1). Let γ'_j be the jth coding coefficient in a coded packet generated from node a, and $\gamma_{i,j}$ denote the jth coding coefficient in the ith coded packet in the node buffer. Then, the relation of coding coefficients in (1) can be expanded as follows:

$$\begin{bmatrix} \gamma'_1 \\ \vdots \\ \gamma'_{Kn} \end{bmatrix} = \sum_{i}^{m} \eta_i \begin{bmatrix} \gamma_{i,1} \\ \vdots \\ \gamma_{i,Kn} \end{bmatrix} \tag{3}$$

where η_i is a coefficient randomly chosen from the Galois field on which network coding is based. (3) essentially implies that the jth coding coefficient in a coded packet only depends on the jth coding coefficient of the coded packets that it is encoded from. In other words, the jth column in the decoding matrix are independent from the other columns. Hence, all columns are independent from one another. Therefore, all submatrices, composed by columns, are independent from one another, and their ranks increase independently. ■

When a terminal node sends out a packet, the path successful-transmission probability β_i is essentially the probability of a rank increase in submatrix i. Hence, Proposition 2 implies that Proposition 1 holds even when multiple flows

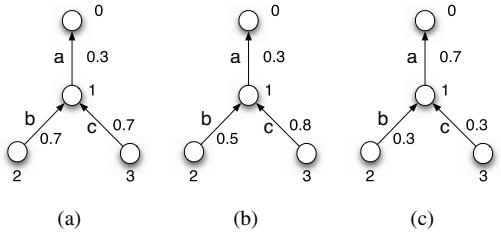


Fig. 2. The path rates from node 2 and 3 to the sink are both 0.3 in (a), (b), and (c).

share links. We thus revise Proposition 1 into the following corollary:

Corollary 1: In the sense of rank-increase probability of a submatrix, (2) holds even when multiple flows exist in the network.

All remaining parts of this paper consider β_i in this sense. In addition, we note that β_i can be estimated by $\beta_i = \frac{K}{t_i}$. It is easy to see that the accuracy of this equation increases as K goes to infinity.

We further remark that Proposition 2 does not violate the intuition that the transmissions of multiple flows interfere each other on a shared link. The reason is that even when the ranks of all submatrices reach K, the sink may still not be able to decode all Kn packets. For instance, in Fig. 1, despite the two submatrices in the decoding matrix both have rank $K = 2$, the sink is unable to decode all $Kn = 4$ packets. In fact, with network coding, the source packets from different terminal nodes are decoded at the same time.

Link loss inference essentially attempts to assign link rates based on a system of constraints induced from all path rates observed at the sink, where the constraints are given by the connection between the path rate and the corresponding link rates. Because the number of paths to the sink are smaller than the number of links in the network, the number of constraints are smaller than the number of unknown variables. Therefore, we are generally unable to determine a set of unique link rates. Furthermore, in our inference problem, a path rate provides only the information of the link with the lowest rate because of (2). For example, the path rates are not sufficient to detect the difference between Fig. 2(a) and (b). Finally, the same set of path rates may be due to a different set of bottleneck links as shown by the difference between Fig. 2(a) and (c). Therefore, in this paper, we do not seek to obtain the precise link loss probabilities, but rather attempt to identify the subset of highly lossy links.

We are now ready to describe the inference problem in a more formal way. The inputs of the problem are the full-rank times D observed at the sink, and the link rate threshold T_l, where T_l is a threshold value such that a link is defined as good if its rate is higher than T_l, and bad otherwise. If we define an indicator variable s_e assuming the value 1 if link e is classified as a good link, and 0 otherwise, the output of the problem is the states of all links $S = \{s_e\}$, where $e \in E$.

V. Passive Network Tomography

In this section, we describe three different inference algorithms to discover highly lossy links. We discuss their different computational complexities and relative performance.

A. Bayesian Inference Using Factor Graphs

The recent notion of factor graphs [9] has attracted intense research interest, since it was recognized that factor graphs, and the *sum-product* algorithm operating on them, unify a variety of previously discovered well-known algorithms, such as the Viterbi algorithm, belief propagation, FFT, and so on. The factor-graph framework, as one example of probabilistic graphic models [10], essentially uses a graph to represent the dependency among random variables. With factor graphs, a "global" function, representing the joint distribution of all random variables, can be factorized to multiple "local" functions. Furthermore, by utilizing intermediate computation results, the sum-product algorithm is able to compute the marginal distribution of all random variables simultaneously, hence dramatically reducing the computation complexity in inference [9]. In this section, we demonstrate how to apply factor graphs for link loss inference with network coding.

In Bayesian inference, all parameters are considered as random variables to quantify the belief on their values. It is easy to see that we can capture the dependency among link rates α_e and path rates β using the following conditional distribution[2] due to (2):

$$P(\beta_i|\{\alpha_{e_j}\}) = \begin{cases} 1 & \text{if } \beta_i = \min(\alpha_{e_j}), \\ 0 & \text{otherwise,} \end{cases} \tag{4}$$

where $e_j \in P_i$, and P_i denotes the set of links on path i. Such a conditional probability is usually called a factor function.

Next, we consider the dependency between path rates β_i and the full-rank time t_i (described in Sec. IV-B). Any K coded packets are equivalently useful for decoding with network coding. Hence, the rank of a submatrix reaches K if there are at least K linearly independent coded packets arriving at the sink. Hence, the full-rank time t_i follows a negative binomial distribution:

$$P(t_i|\beta_i) = \binom{t_i - 1}{K - 1} \beta_i^K (1 - \beta_i)^{t_i - K} \ . \tag{5}$$

With (4) and (5), the joint distribution of β_i, α_e, and t_i can be factorized as follows:

$$P(\{\beta_i\}, \{\alpha_{e_j}\}, \{t_i\}) = \Pi_{i \in W, e_j \in P_i} P(\beta_i|\{\alpha_{e_j}\}) P(t_i|\beta_i) P(\alpha_{e_j}) \tag{6}$$

where W denotes the set of all paths. Fig. 3 shows the factor graph of the network with the topology shown in Fig. 2, where each square is a function vertex representing the factors defined in (4) and (5), and each circle is a variable vertex representing variables α_e, β_i, and t_i. Note that we omit the dummy factor vertices representing $P(\alpha_{e_j})$ in Fig. 3.

[2]For notation simplicity, in this work we use the short hand $P(x)$ to generally represent either the probability mass, $Pr[X = x]$, or the probability density, $f_X(x)$, depending on the continuity of random variable X.

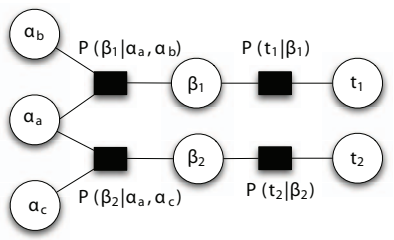


Fig. 3. The factor graph formulation of the examples in Fig. 2.

After constructing the factor graph, we set the prior distributions of the vertices representing link rates and path rates to the uniform distribution, assuming no *a prior* knowledge on them. We further set the evidence on the vertex t_i representing the full-rank times with the observed data D. Afterwards, the sum-product algorithm is operated on this graph to compute the marginal probabilities, *i.e.,* the posterior probabilities, of link rates.

With all link posterior probabilities $P(\alpha_e|D)$ given the observed data, we classify a link to be a bad link if the probability that the link rate is smaller than T_l is greater than 1/2, where T_l is the given link rate threshold defined in Sec. IV-B. We then have

$$s_e = \begin{cases} 1 & \text{if } \int_0^{T_l} P(\alpha_e|D)\,\mathrm{d}\alpha_e < 1/2, \\ 0 & \text{if } \int_0^{T_l} P(\alpha_e|D)\,\mathrm{d}\alpha_e \geq 1/2, \end{cases} \tag{7}$$

where s_e is the link state indicator variable as defined in Sec. IV-B.

For this algorithm, rather than using the path rate calculated from K/t_i, which is accurate only if K is sufficiently large, we utilize the observed full-rank time t_i directly. Hence, we expect this algorithm to perform well even with a small K.

Note that since the factor graph shown in Fig. 3 is a tree, the sum-product algorithm is able to produce exact inference, and furthermore, the sum-product algorithm only need to traverse the graph twice in order to compute all marginal probabilities. Hence, the complexity of this algorithm is dramatically reduced by factorization. However, in reality, the conditional probabilities (4) still pose challenges. To illustrate this, we briefly discuss the implementation details. To facilitate computer-based numerical calculation, we use discrete distribution vectors with, say, M elements, to approximate the continuous distributions of random variables β_i and α_{e_j}. Then, both the space and time complexities to store (4) and compute (4) grow with M^{m+1}, where m is the length of path i. To obtain reliable classification, M is usually chosen to be greater than 100, and a path length can easily be larger than 5, rendering this algorithm impractical for large-scale networks. This type of computational difficulty is usually referred to as "the curse of dimensionality" [10].

B. Bayesian Inference Using Gibbs Sampling

In this section, we describe a Bayesian formulation based on Gibbs sampling [10]. Gibbs sampling is a well-known technique that has also been similarly applied in [5] and [8] under different system models. Let D represent the observed data, and α denote the parameters to infer, we have the well-known Bayesian inference formulation:

$$P(\alpha|D) = \frac{P(\alpha)P(D|\alpha)}{\int P(\alpha)P(D|\alpha)\,\mathrm{d}\alpha}. \tag{8}$$

For our problem, the parameters $\alpha = \{\alpha_e\}$ are the set of link rates, for any $e \in E$, and the observed data D is the full-rank times $\{t_i\}$ for all submatrices as described in Sec. IV-B.

We define the likelihood function $P(t_i) = \alpha_i^K(1-\alpha_i)^{t_i-K}$ on a path assuming independence among the transmissions of different coded packets. Furthermore, because the ranks of different submatrices increase independently according to Corollary 1, we have the likelihood function

$$P(D|\alpha) = \Pi_{i\in W}\beta_i^K(1-\beta_i)^{t_i-K}, \tag{9}$$

where $\beta_i = \min_{e\in P_i}(\alpha_e)$ as justified in Corollary 1. The prior distribution $P(\alpha)$ indicates the prior knowledge about the link rates. Similar to Sec. V-A, we set the prior to be uniform assuming no *a prior* knowledge about link rates.

The Gibbs sampling algorithm belongs to the family of Markov Chain Monte Carlo (MCMC) algorithms [10]. It is particularly useful if marginal distributions are very difficult to compute directly. Rather than doing so, MCMC algorithms seek to generate samples from these distributions. We describe the Gibbs sampling algorithm in the context of our problem as follows. We start with an arbitrary initial assignments of link rates α. Afterwards, one of the links is picked either randomly or according to a particular order. Assuming that link e is chosen, we then compute the posterior distribution of α_e conditioned on the observed data D and all currently assigned other link rates, $\{\bar{\alpha}_e\} = \cup_{f\neq e}\{\alpha_f\}$. That is, we compute the following conditional probability:

$$P(\alpha_e|D,\{\bar{\alpha}_e\}) = \frac{P(D|\{\alpha_e\},\{\bar{\alpha}_e\})P(\alpha_e)}{\int_{\alpha_e} P(D|\{\alpha_e\},\{\bar{\alpha}_e\})P(\alpha_e)\,\mathrm{d}\alpha_e} \tag{10}$$

Since $\{\alpha_e\} \cup \{\bar{\alpha}_e\} = \alpha$, $P(D|\{\alpha_e\},\{\bar{\alpha}_e\})$ is $P(D|\alpha)$. Furthermore, $P(\alpha_e)$ is a constant because of the assumption of uniform prior probability. Thus, (10) is simplified to

$$P(\alpha_e|D,\{\bar{\alpha}_e\}) = \frac{P(D|\alpha)}{\int_{\alpha_e} P(D|\alpha)\,\mathrm{d}\alpha_e}, \tag{11}$$

which can then be used to generate a new sample α_e' for the rate of link e. In this way, we repeat this procedure for all e in E such that each link is assigned a new rate. We then iterate this algorithm until it converges, at which time, by the theory of Gibbs sampling [10], the samples are produced from the true posterior distribution of link rates $P(\alpha_e|D)$.

In this work we iterate the above algorithm for $M = 2\kappa$ times, where κ is a sufficiently large number of iterations for the algorithm to converge. Using the link threshold T_l defined in Sec. IV-B, we classify link e to be a bad link if more than half of the samples from the later $M/2$ samples are less than T_l. Hence, this classification rule essentially labels a link as

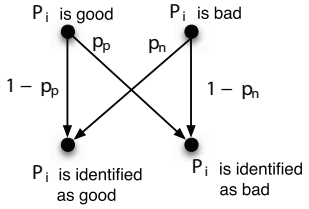


Fig. 4. The definition of false positive rate p_p and false negative rate p_n for a path.

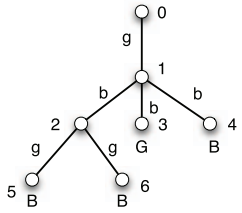


Fig. 5. The SCFS algorithm. g or b indicates the link is good and bad, respectively. G or B specifies the path is good or bad, respectively.

bad if the posterior probability, that the link rate is lower than T_l, is greater than 1/2.

With Gibbs sampling, the infeasible computation task to compute (8) directly for all link rates is reduced to a numerical computation of the conditional probability (11) on one link rate. Hence, this formulation does not suffer from the high dimensional problem in Sec. V-A.

C. Smallest Consistent Failure Set (SCFS)

In this section, we modify the SCFS approach proposed in [11], originally to infer lossy links without network coding, and show its performance under network coding is better than in the case of traditional routing. As shown in Sec. IV-B, the constraints from end-to-end observations are not sufficient to uniquely identify a set of link rate assignments. In fact, the constraints cannot uniquely determine a set of bad links either. The basic idea of the SCFS approach is that, if there are multiple choices to assign whether links are good or bad in order to satisfy the end-to-end observations, one should select the smallest set of bad links, under the assumption that a large fraction of the network is well behaved. In the sensor networks under consideration, SCFS implies that the links closer to the sink are more likely to be chosen as bad links, since they influence more paths than the links closer to the sources.

The SCFS algorithm consists of two steps. First, it obtains the *state*, good or bad, of each path from all terminal nodes to the sink. Second, based on these path states, the algorithm assigns link states with the principle to assign as few bad links as possible. For the first step, a proper path threshold is required to separate good paths from bad paths. We will show that our setting with network coding make the choice of an ideal path threshold more natural and straightforward, compared with the case of tradition routing.

More precisely, due to (2), a good path consists of only good links, whereas a bad path is one consisting of at least one bad link. We use a path threshold T_p such that a path is good if the path rate is higher than T_p; otherwise, it is bad. As illustrated in Fig. 4, we define the false-detection events *false positive* if the path is good but we classify it as bad, and *false negative* if a path is in fact bad but we classify it as good[3]. In our setting, it is clear that we should set the path threshold T_p as the link threshold T_l, as we can easily draw the following observation:

Proposition 3: If we set $T_p = T_l$, both false positive rate and false negative rate are zero for the network coding based protocol.

In contrast, we can show that an ideal path threshold does not exist for traditional routing protocols:

Proposition 4: It is infeasible to select a path threshold such that both false positive rate p_p and false negative rate p_n are zero for traditional routing protocols.

Proof: Recall the path rate is the product of all link rates in traditional routing protocols. Suppose an ideal path threshold T_p exists such that both p_p and p_n are zero. We further assume that the length of the path is m and $m > 1$. Suppose that all links on this path has rate $T_l + \epsilon$, where ϵ is a small positive number. This path is a good path because all links have rate greater than T_l. The path rate is $(T_l + \epsilon)^m \approx T_l^m$. Hence, we have $T_p < T_l^m$ to avoid classifying this path as bad, *i.e.*, to avoid false positive. Furthermore, suppose that there is one bad link with rate $T_l - \epsilon$, whereas all other links have rate 1. This path is a bad path because it consists of a bad link. The path rate is $T_l - \epsilon$. Therefore, we have $T_p > T_l - \epsilon \approx T_l$, to avoid classifying this link as good, *i.e.* to avoid false negative. Collectively, we then have both $T_p < T_l^m$ and $T_p > T_l$, which is a contradiction, since $0 < T_l < 1$ and $m > 1$. Hence, no such an ideal path threshold exists. ■

Proposition 3 and 4 suggest that we should expect SCFS to perform better under our problem setting.

The second step of the SCFS algorithm in our setting is identical to that in [11]. For completeness, we briefly review it using the example shown in Fig. 5. Each link in the tree belongs to a pair of parent and child, and the child can uniquely identify this link in a tree. Hence, for brevity, we use the node ID of the child to name the link (*e.g.*, the link between node 1 and 2 is referred to as link 2). The input of the algorithm is the result of the first step of SCFS: the states of all paths from terminal nodes to the sink. The algorithm first assigns each link the states of all paths that it belongs to. For example, link 1 has path states $\{B, B, G, B\}$, and link 2 has path state $\{B, B\}$, where G and B represent "good" and "bad", respectively. Then, the algorithm traverses the tree from the root recursively to classify each link according to these states. If one of the paths consists of the current link is good, this link is certainly good and classified as good. For example, link 1 is good. On the other hand, if all paths comprising this link is bad, there are multiple possibilities. For an illustration, link 2 has two bad paths across it. Hence, either link 2 is bad, or both link 5 and 6 are bad. According to the assumption that the fraction of bad links in the network is small, the algorithm classifies link 2 as bad, and all links on the subtree rooted at node 2 as good in order to minimize the total number of bad links in the network.

This combinatorial algorithm executes much faster than the two numerical Bayesian algorithms presented in Sec. V-A and V-B. However, it relies on the estimation of path rates, which

[3]Note that these false detections are only used to to measure whether T_p is the right choice, and not the final metrics for link rate assignments, which are defined in Sec. VI.

may not be accurate if K is small. Furthermore, it is clear that the algorithm produces wrong answers when the smallest consistent rule is violated. For instance, if $\{B, B\}$ at link 2 is due to the fact that link 5 and 6 are both bad, and link 2 is actually good, then link 2 will be detected as false positive, where as link 5 and 6 will be detected as false negative.

VI. Evaluation

In this section, we compare the three proposed inference algorithms under different network settings. For this purpose, we have developed a customized packet-level network simulator in C++, with the implementation of randomized network coding, using GF(2^8) as the code space. The inference algorithms are implemented in Matlab.

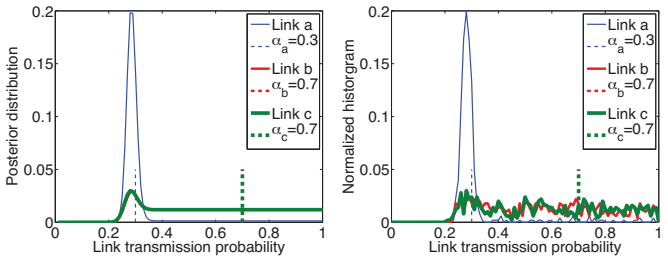
We use two different loss rate models introduced in our previous work [6], [7]. In the first model (LRM1) [6], all good and bad links are assigned rates of 0.7 and 0.3, respectively. We use a link threshold $T_l = 0.5$ to partition them. This simple model is a good approximation of the real sensor link qualities. With sensor testbeds, [23] and [8] have shown that a wireless link has either low or high loss probability, but rarely has intermediate loss rates. Furthermore, the MintRoute routing algorithm adopted in our protocol rarely uses a link with intermediate quality [23]. This loss rate model is used for most of our experiments. The second model (LRM2) [7] assigns a loss rate to a link from a distribution with density function $f(\alpha) = \lambda\alpha^{(\lambda-1)}$, where $0 < \alpha \leq 1$. We use $\lambda = 10$ and $T_l = 0.8$ in our simulations, such that there are 12% bad links in the network on average.

We use two metrics to measure algorithm performance. First, Detection Rate (DR) is defined as the fraction of links correctly identified as bad among all bad links. Second, False Positive Rate (FPR) is defined as the percentage of good links among all links identified as bad.

A. Bayesian Inference on a Simple Topology

In this section, we present the results of the Bayesian inference algorithms in a simple topology shown in Fig. 2(a) and (c) with a segment size $K = 100$. Fig. 6 shows that the algorithms based on factor graphs or Gibbs sampling produce similar results. This is because, in our factor graph formulation, the tree structure ensures exact inference by the sum-product algorithm, whereas Gibbs sampling, as a randomized approximate inference algorithm, always generates random samples from the correct distribution, even if exact inference is infeasible [10]. However, factor graph formulation suffers from intractable computation in large-scale networks as explained in Sec. V-A. Hence, for the larger topology in the next subsection, we focus on Gibbs sampling as the representation for Bayesian inference algorithms.

Furthermore, as shown in Fig. 6(a) and (b), for the network settings in Fig. 2(a), we observe that the Bayesian algorithms produce a posterior distribution of link a concentrating on its true rate 0.3. Furthermore, link a obscures the information on links b and c, since Fig. 6(a) and (b) show that the posterior distributions of the rates of link b and c range approximately



(a) (b)

Fig. 6. The posterior distributions of the link rates from Bayesian inference algorithms. α_i denotes the true rate of link i. (a) and (b) are the results of factor graphs and Gibbs sampling under the settings of Fig. 2(a).

uniformly from 0.3 to 1, implying that the link rates are equally likely from 0.3 to 1. Such observation is consistent with our intuition that the rates of the two links are unidentifiable due to the blocking effect of link a. Hence, it is clear, we should not use the principle of maximum a posterior to assign link rates, since this method would have chosen a link rate lower than 0.4 to both links b and c, which is far from the truth. Instead, we follow the classification rule described in Sec. V-A and V-B. If we use 0.5 as the threshold to partition good and bad links, it is highly likely for the algorithm to classify link b, c to be good links from the posterior distributions of their rates. In this way, both algorithms produce the correct results.

B. Effect of Different Network Settings

In this section, we study the effect of different network settings on a randomly generated network with 100 nodes. Excluding the sink and terminal nodes, the average number of children of a node on the tree is 3.0625, and the average length of a path from a terminal node to the sink is 4.8209. We use LRM1 unless explicitly pointed out. Furthermore, we repeat the same experiments for 30 times, with different random seeds, and show the mean DR and FPR. Since the factor graphs based algorithm is not scalable, we show only the results of Gibbs sampling along with SCFS.

We first study the effect of segment sizes. We set the fraction of bad links to 5%. Fig. 7(a) shows that DR increases with segment sizes. This is because with a larger segment size, we inject more data into the network and, hence, obtain more reliable observations. More importantly, we notice that the Bayesian algorithm significantly outperforms SCFS, in FPR, when the segment size is 20 and 30. This is due to the facts that the Bayesian algorithm uses observed data directly, whereas SCFS uses estimated path rates, which are not accurate when given a small amount of data. In the following experiments, we set segment size to 50.

We next investigate the algorithm performance under different fractions of lossy links. From Fig. 7(b), we notice that DR decreases when the fraction of lossy links increases, whereas FPR remains low. The lower detection rates occur because although more and more bad links appear, their information are blocked by the bad links closer to the root. Furthermore, the assumption of SCFS is gradually violated with an increasing

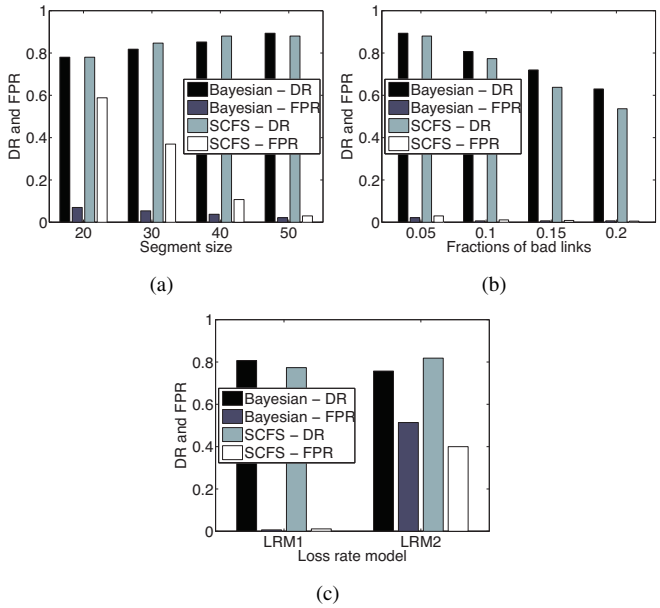


Fig. 7. DR and FPR under different (a) segment sizes, (b) fractions of lossy links, and (c) loss rate models.

number of lossy links. In the following experiments, we set the fractions of bad links to 10%.

It is obvious that any inference algorithm would perform better when the quantities to be inferred are more clearly distinguishable. Next, we compare the inference results between LRM1 and LRM2. As shown in Fig. 7(c), we observe that the performance of both algorithms degrades in LRM2. With LRM1, the good and bad links are sufficiently separated. On the other hand, the link rates in LRM2 are generated from a continuous random variable, and are thus difficult to separate. Fortunately, we are more interested in the results under LRM1 because it is closer to real-world scenarios [23], [8].

VII. Conclusions

This paper represents the first attempt to address passive link loss inference under network coding traffic in wireless sensor networks. We show that network coding changes the fundamental connection between path and link successful-transmission probabilities. To address this new challenge, we develop Bayesian inference algorithms based on factor graphs and Gibbs sampling, and we further demonstrate that network coding traffic improves the performance of the SCFS heuristic algorithm. Finally, through analysis and extensive simulations, we show that these inference algorithms achieve different performance tradeoffs. In particular, factor graphs offer most complete and accurate information for all link rates. However, it is not as scalable as Gibb sampling. Furthermore, Bayesian algorithms perform well even with limited data, whereas the heuristic algorithm requires a sufficiently large amount of data. On the other hand, the heuristic algorithm can achieve nearly optimal results with more efficient operations.

References

[1] R. Ahlswede, N. Cai, S. R. Li, and R. W. Yeung, "Network Information Flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, July 2000.

[2] D. S. Lun, M. Medard, and M. Effros, "On Coding for Reliable Communication over Packet Networks," in *Proc. of 42rd Allerton Conference on Communication, Control, and Computing*, 2004.

[3] S. Chachulski, M. Jennings, S. Katti, and D. Katabi, "Trading Structure for Randomness in Wireless Opportunistic Routing," in *Proc. of ACM SIGCOMM*, 2007.

[4] T. Ho, M. Medard, M. Effros, and R. Koetter, "Network Coding for Correlated Sources," in *CISS*, 2004.

[5] V. N. Padmanabhan, L. Qiu, and H. J. Wang, "Server-based Inference of Internet Link Lossiness," in *Proc. of IEEE INFOCOM*, 2003.

[6] G. Hartl and B. Li, "Loss Inference in Wireless Sensor Networks Based on Data Aggregation," in *Proc. of IPSN*, 2004.

[7] Y. Mao, F. R. Kschischang, B. Li, and S. Pasupathy, "A Factor Graph Approach to Link Loss Monitoring in Wireless Sensor Networks," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 820–829, April 2005.

[8] H. X. Nguyen and P. Thiran, "Using End-to-End Data to Infer Lossy Links in Sensor Networks," in *Proc. of IEEE INFOCOM*, 2006.

[9] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor Graphs and the Sum-Product Algorithm," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, February 2001.

[10] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[11] N. Duffield, "Network Tomography of Binary Network Performance Characteristics," *IEEE Transactions on Information Theory*, vol. 52, no. 12, pp. 5373–5388, December 2006.

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

[13] R. Caceres, N. Duffield, J. Horowitz, and D. Towsley, "Multicast-Based Inference of Network-Internal Loss Characteristics," *IEEE Transactions on Information Theory*, vol. 45, no. 7, pp. 2462–2480, Nov. 1999.

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

[15] T. Ho, B. Leong, Y.-H. Chang, Y. Wen, and R. Koetter, "Network Monitoring in Multicast Networks Using Network Coding," in *Proc. of IEEE International Symposium on Information Theory (ISIT)*, 2005.

[16] C. Fragouli and A. Markopoulou, "A Network Coding Approach to Overlay Network Monitoring," in *Proc. of 43rd Allerton Conference on Communication, Control, and Computing*, 2005.

[17] C. Fragouli, A. Markopoulou, R. Srinivasan, and S. Diggavi, "Network Monitoring: It Depends on your Points of View," in *Proc. of Information Theory and Applications Workshop (ITA)*, 2007.

[18] M. Gjoka, C. Fragouli, P. Sattari, and A. Markopoulou, "Loss Tomography in General Topologies with Network Coding," in *Proc. of IEEE Globecom*, 2007.

[19] C. Fragouli, A. Markopoulou, and S. Diggavi, "Topology Inference using Network Coding," in *Proc. of 44th Annual Allerton Conference on Communication, Control and Computing*, 2006.

[20] M. Jafarisiavoshani, C. Fragouli, and S. Diggavi, "Subspace Properties of Randomized Network Coding," in *Proc. of IEEE Information Theory Workshop (ITW)*, 2007.

[21] M. Jafarisiavoshani, C. Fragouli, S. Diggavi, and C. Gkantsidis, "Bottleneck Discovery and Overlay Management in Network Coded Peer-to-Peer Systems," in *Proc. of SIGCOMM Workshop on Internet Network Management (INM)*, 2007.

[22] G. Sharma, S. Jaggi, and B. Dey, "Network Tomography via Network Coding," in *Proc. of Information Theory and Applications (ITA)*, 2008.

[23] A. Woo, T. Tong, and D. Culler, "Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks," in *Proc. of ACM Sensys*, 2003.

[24] T. Ho, R. Koetter, M. Medard, D. R. Karger, and M. Effros, "The Benefits of Coding over Routing in a Randomized Setting," in *Proc. of IEEE International Symposium on Information Theory*, 2003.

[25] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. Wiley, 1991.

[26] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. The MIT Press, 2001.