

INPoD: In-Network Processing over Sensor Networks based on Code Design

Kiran Misra, Shirish Karande, and Hayder Radha, *Senior Member, IEEE*

Abstract—In this paper, we develop a joint Network Coding (NC)-channel coding error-resilient sensor-network approach that performs In-Network Processing based on channel code Design (INPoD). INPoD represents a major development of an underlying framework for designing Code-on-Network-Graph (CNG). CNG (and hence INPoD) maps variable nodes of Low Density Parity Check (LDPC) codes onto sensor nodes, and consequently translates check equations (used in linear algebraic LDPC codes) into in-network processing. INPoD/CNG codes not only improve capacity, but are also resilient to errors in noisy environments. In absence of INPoD, basic CNG employs standard LDPC codes while assuming the underlying sensor network is capable of supporting the degree distribution dictated by these codes. In practice, however, we usually have the network topology pre-determined by the placement of the sensors, and hence we are constrained to map a code onto a given topology. In this paper, we specifically address this problem, and propose the INPoD framework, which enables the use of LDPC design tools in the design of CNG for a given sensor network topology. We formulate the design of CNG, as a convex optimization problem, which determines the best codes to be used, given the underlying network connectivity and channel conditions. Specifically, we use *density evolution* to design degree distributions which controls the performance of the joint NC-channel code CNG. We also give a code construction algorithm, which achieves a designed degree distribution. We show that well-designed INPoD provide gains of 1.5 to 2.5 dB, when compared with the best known erasure codes – LT codes.

Index Terms—Low Density Parity Check Codes, LT codes, Wireless Sensor Networks, Network Coding

I. INTRODUCTION

RELIABLE transmission of data collected by a large-scale sensor networks poses many unique challenges. Network-Coding (NC) [3] achieves capacities higher than traditional routing algorithms [1][2], making them a prime candidate for large-scale data transmissions. Moreover, the broadcast-nature of wireless channels in sensor networks makes a strong case for use of NC in Wireless Sensor Networks (WSN). The basic philosophy of NC is to enable transmission of

the information via multiple paths, while making efficient use of the available bandwidth with the intermediate coding nodes. Initial efforts in NC required knowledge of complete network topology, however, in [4][5] Ho et. al. proposed a decentralized approach, and derived performance bounds on randomized NC. These randomized network codes, exceeded the performance of randomized routing algorithms. However, these codes are not robust to errors, as seen in wireless environments. In [6] Bao et. al. formulated the NC problem as a joint network-channel coding problem which overcomes this drawback, and allows the construction of joint NC-channel codes which can operate even in noisy environments. It was shown that the algebraic operations (at the intermediate nodes) of NC correspond to the check equations of a linear algebraic channel code. This unique perspective allows us to map the sensor network nodes (more specifically the data they transmit) to the variable nodes of channel code (Figure 1). This in turn permits the information received at the sink to be processed using well-known channel decoding algorithms.

In this paper, we provide key contributions to the notion of mapping channel Code-on-Network-Graph (CNG). Under basic CNG, the mapping of variable nodes, of Low Density Parity Check (LDPC) codes onto sensor nodes takes place without considering crucial constraints imposed by the underlying sensor network topology. Consequently, we develop a joint NC-channel coding CNG approach that performs In-Network Processing based on channel code Design (INPoD) that addresses key constraints including topology constraints as explained further below. (For the remainder of the paper, a joint CNG code design that addresses these constraints is synonymous with an INPoD code.)

It is known that, channel codes (when properly designed) are robust to erasures as well as errors, making CNG resilient to noise and applicable for wide-range of sensor networks. Current literature [6][7] use standard channel codes in their work, which assume that the underlying sensor network is capable of supporting the topology dictated by these channel codes. However, in real-world scenarios, the underlying topology is often

The authors are with the Electrical and Computer Engineering Department, Michigan State University, East Lansing, MI 48824-1226 USA. (e-mail: {misrakir, karandes, radha}@egr.msu.edu).

This work was supported in part by NSF Award CCF-0515253, NSF Award CNS-0430436, MEDC Grant GR-296, and unrestricted gift from Microsoft Research.

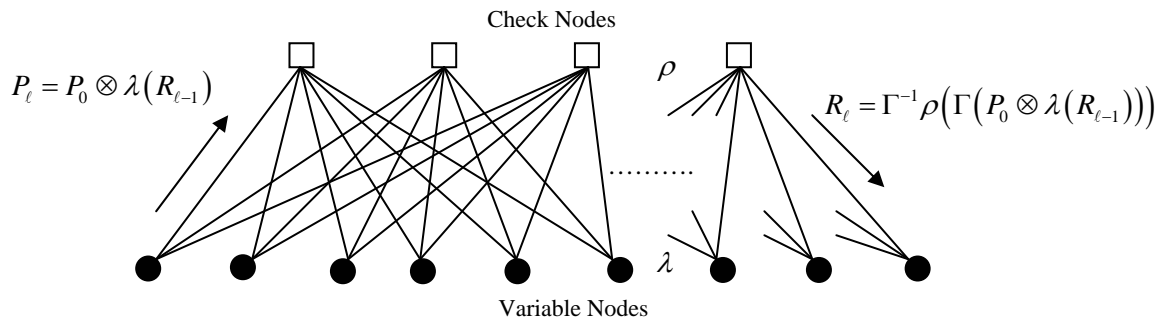


Figure 1: Belief propagation algorithm for LDPC codes

pre-determined by the placement of the sensors, and we are required to determine a code which could be mapped to the given topology. In this paper, we do precisely that, and formulate the design of CNG network codes as a channel-coding design problem, constrained by the underlying sensor network topology and relay channel conditions. In our problem formulation we consider finite neighborhood sizes, while designing the joint NC-channel coding CNG codes. To the best of our knowledge this is the first attempt to include topological constraints while performing code design. Furthermore, we present an algorithm to construct CNG given a degree distribution, on the underlying sensor network. We show that dense sensor networks in general perform better than sparse networks for the same degree distribution. We also demonstrate that the joint CNG designed by our proposed framework (which considers underlying network connectivity) gives degree distributions, which have superior performance when compared to existing channel codes – LT codes. We exploit the analogous behavior of network codes and low-density parity-checks (LDPC) codes in achieving lower *outage rates*, i.e. the rate at which data-bits are lost in the WSN. Although, there are many methods for designing LDPC codes [10]-[17], we concentrate our efforts on using *density evolution* [10][14] for network code design, due to their higher accuracy in predicting the average behavior of an LDPC ensemble, and also because of their utility in designing extremely good channel codes.

Section II of this paper describes how to convert an arbitrary NC problem into a channel coding problem. More specifically we describe how the XOR operations in NC can be represented as check equations of the bipartite graph of LDPC codes. XORing is a common technique employed in NC, when the network nodes are devices with low-complexity [23]. In section III, we look at how *density evolution* is employed to analyze the performance of the degree distributions used in Belief

Propagation Algorithm (BPA). The choice of variable (λ) and check (ρ) degree distribution is restricted by the connectivity of the underlying WSN, whereas the channel code rate (R) is determined by the behavior of the relay network. Section IV analyzes the channel as seen by the sink, and determines bounds on the channel code rate R . We next define a homogeneous sensor network in section V, and give an algorithm to construct the parity generator matrix G constrained by the graph connectivity. Section VI gives lower bounds for successful code construction. These bounds must be satisfied to get close to the designed parity degree distribution, and to guarantee *coverage* (i.e. all data-nodes participate in forming at least one parity-bit). In section VII, we give the mathematical formulation of the optimization problem for a general sensor network. We discuss the optimization issues involved, and suggest heuristics for good initial guesses. Finally we compare the *outage rates* for channel codes with LT degree distribution [18], to the designed degree distribution in VIII and present our conclusions in IX.

II. CONVERTING NETWORK CODING TO CHANNEL CODING

In NC the intermediate nodes of a network perform linear algebraic operations on the received data, before forwarding them to their neighbors. These operations can be represented by a global matrix at the sink. As long as, the linear algebraic operations result in a global matrix which is invertible, the transmitted data can be recovered successfully at the sink. Typically, linear algebraic operations can result in data in higher Galois fields, however, since we restrict our attention to XORs, we can say without loss of generality the data in the sensor networks is binary. This XORing of bits can be represented as check equations in a belief propagation graph, transforming network codes into a belief network. Figure 2 illustrates such an example. Part a) of the figure, represents the following NC operations:

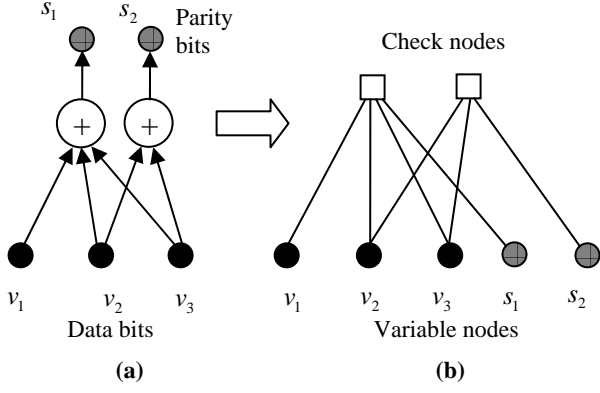


Figure 2: Converting network coding operations into a Belief Network for LDPC decoding

$$\begin{aligned} v_1 \oplus v_2 \oplus v_3 &= s_1 \\ v_2 \oplus v_3 &= s_2 \end{aligned} \quad (\text{II.1})$$

where, \oplus represents componentwise modulo-2 addition or XORing. The left side of the equation represents data received by a node, whereas the right side of the equation represents data obtained after XORing, which is forwarded to the downstream nodes. Equation (II.1) can be rewritten as:

$$\begin{aligned} v_1 \oplus v_2 \oplus v_3 \oplus s_1 &= 0 \\ v_2 \oplus v_3 \oplus s_2 &= 0 \end{aligned} \quad (\text{II.2})$$

This corresponds to the bipartite graph shown in Figure 2b) with zero-syndrome. As can be seen, the incoming data-bits and the parity-bit generated by NC get mapped to the variables nodes of the belief network. There exists a relationship between the node-degree distributions of network-code to the degree distributions of the belief network. Specifically, if n represents the maximum degree a data-node can take, and m represents the maximum degree for a parity node, and if the data and parity node degree distributions are $\bar{\theta} = \{\bar{\theta}_1, \bar{\theta}_2, \dots, \bar{\theta}_n\}$ and $\bar{\phi} = \{\bar{\phi}_1, \bar{\phi}_2, \dots, \bar{\phi}_m\}$ respectively, then:

$$\begin{aligned} \bar{\lambda} &= \{\bar{\lambda}_1, \bar{\lambda}_2, \dots, \bar{\lambda}_n\} = R * \{\bar{\theta}_1 - 1 + 1/R, \bar{\theta}_2, \bar{\theta}_3, \dots, \bar{\theta}_n\} \\ \bar{\rho} &= \{\bar{\rho}_2, \bar{\rho}_3, \dots, \bar{\rho}_{m+1}\} = \{\bar{\phi}_1, \bar{\phi}_2, \dots, \bar{\phi}_m\} \end{aligned} \quad \dots(\text{II.3})$$

where,

- $\bar{\theta}_i$ is the proportion of data nodes with degree i
- $\bar{\phi}_i$ is the proportion of parity nodes with degree i
- $\bar{\lambda}_i$ is the proportion of variable nodes with degree i
- $\bar{\rho}_i$ is the proportion of check nodes with degree i
- R is the code rate

Since the above vectors essentially represent the probability distributions, we have the following additional constraints on the degree distributions:

$$\sum_{i=1}^n \bar{\lambda}_i = 1 \quad \sum_{i=2}^{m+1} \bar{\rho}_i = 1 \quad \sum_{i=1}^n \bar{\theta}_i = 1 \quad \sum_{i=1}^m \bar{\phi}_i = 1 \quad (\text{II.4})$$

The sink then uses BPA on the resulting graph, to perform decoding. The BPA iteratively passes messages between the variable-nodes, corresponding to the codeword side, and the check-nodes, corresponding to the check-equations, as shown in Figure 1. The algorithm stops when all the checks are satisfied, i.e. when a valid codeword is reached. The mapping translates the inversion operation (of the global matrix) in NC to decoding using BPA. The design of network codes can now be posed as the design of the bipartite graph shown in Figure 2b). Invertibility of the global matrix in NC, translates to determination of a graph with near full-rank incidence matrix. This incidence matrix is also called the parity check matrix H and has the general form (where K is the number of data-bits):

$$H = M \text{ checks} \left\{ \begin{array}{c} \left[\begin{array}{ccc|c} 1 & 0 & . & 1 \\ 0 & 1 & . & 0 \\ . & . & . & . \\ 1 & 0 & . & 1 \end{array} \right] I_{M \times M} \end{array} \right. \quad (\text{II.5})$$

K columns

The degree distribution of the bipartite graph represented by H , and shown in Figure 1 plays a pivotal role in determining the performance of the proposed system. A more detailed look at its analysis is given in section III.

III. ANALYSIS OF BELIEF PROPAGATION ALGORITHM USING DENSITY EVOLUTION

The performance of the BPA can be determined by calculating the fraction of incorrect messages passed from the variable node to the check nodes at the ℓ -th iteration, assuming that the graph does not contain cycles of length 2ℓ or less. The cycle-free assumption, for ℓ iterations of the decoder, is necessary to assume independence between incoming messages. Independence in turn, implies that the summation of message at variable and check-nodes, translates to convolution of their probability distributions, making it easier to keep track of the evolution of the message densities. This technique of evaluating code performance by tracking evolution of message densities is labeled *density evolution* [13]. The evolution of message densities from check-nodes to variable-nodes can be succinctly represented by the following equation:

$$R_\ell = \Gamma^{-1} \rho \left(\Gamma \left(P_0 \otimes \lambda(R_{\ell-1}) \right) \right) \quad (\text{III.1})$$

where,

R_ℓ – represents the probability distribution of messages from the check nodes to variable nodes after ℓ iterations.

P_0 – represents the probability distribution of messages received from the symmetric relay channel.

$\lambda = \{\lambda_1, \lambda_2, \dots, \lambda_n\}$, where λ_i is the proportion of *edges* connected to degree i variable nodes, and

$$\sum_{i=1}^n \lambda_i = 1$$

$\rho = \{\rho_2, \rho_3, \dots, \rho_{m+1}\}$, where ρ_i is the proportion of *edges* connected to degree i check nodes, and

$$\sum_{i=2}^{m+1} \rho_i = 1$$

\otimes – represents convolution

$\Gamma = \Gamma^{-1}$ – is change of variable operation associated with the function $\Psi(x) = \log((e^x + 1)/(e^x - 1))$

$$\lambda(x) = \sum_{\forall i} \lambda_i x^{\otimes i-1} \quad \rho(x) = \sum_{\forall i} \rho_i x^{\otimes i-1}$$

Note, here the degree distributions used have an *edge perspective*, while equation (II.3) uses the *node perspective*. The following equations, allow conversions between the two perspectives [14]:

$$\begin{aligned} \bar{\lambda}_i &= \frac{\lambda_i}{i \sum_{\forall j} \lambda_j / j} & \bar{\rho}_i &= \frac{\rho_i}{i \sum_{\forall j} \rho_j / j} \\ \lambda_i &= \frac{i \bar{\lambda}_i}{\sum_{\forall j} j \bar{\lambda}_j} & \rho_i &= \frac{i \bar{\rho}_i}{\sum_{\forall j} j \bar{\rho}_j} \end{aligned}$$

Similar to equation (III.1), the message densities from variable-nodes to check-nodes can be calculated using:

$$P_\ell = P_0 \otimes \lambda(R_{\ell-1}) \quad (III.2)$$

where,

P_ℓ – represents the probability distribution of messages from variable to check nodes after ℓ iterations.

Density evolution is usually carried out for the all-zero codeword, since the performance of the decoder is assumed to be invariant under codeword translation [13]. Now, the fraction of incorrect messages from the

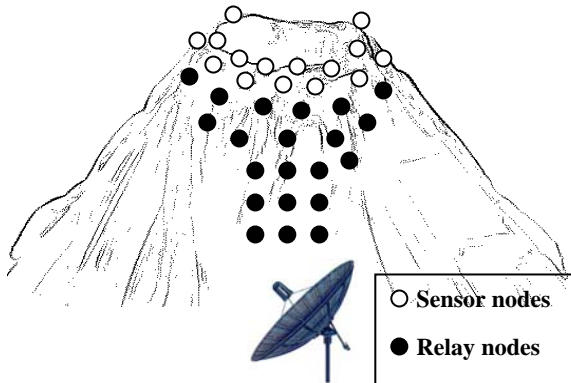


Figure 3: Network Model

variable nodes to the check nodes, after ℓ iterations, can be determined using:

$$P_e^\ell = \int_{-\infty}^0 P_\ell(x) dx \quad (III.3)$$

The above quantity forms the objective function to be minimized in the code design problem. Choosing degree distributions which minimize P_e^ℓ , result in channel codes, which minimize the *outage rate* in the WSN. A more detailed discussion on *density evolution* for LDPC codes is contained in [12][13].

IV. NETWORK MODEL

Figure 3 shows the network model, where the white-nodes represent the sensor nodes, collecting data, and the black-node represents a network of relay nodes forwarding information to the sink. In the very first iteration of data-transmission, all the sensor nodes transmit their data to their neighboring nodes. These data-bits are then XORed by the sensors using the rules defined in the generator matrix G (constructed by the algorithm described in section V). The XORed bits are then transmitted to the sink (by the first node f selected for parity formation, in the code construction algorithm) via the relay nodes. Since only the XORed parity-bits are transmitted to the sink, the data-bits are effectively erased. Also, for our analysis we assume that the relay channel can introduce errors while forwarding the transmitted bits. More specifically, the relay channel is chosen to be a channel with Additive White Gaussian Noise (AWGN). The overall channel as seen by the sink is a mixture of the erasure channel (for data-bits) and the Gaussian channel.

A. Determining the code rate to be used for the mixed channel

Shannon's Noisy Channel Coding Theorem states that given a noisy channel with capacity C , data can be transmitted over the channel with an arbitrarily small probability of error, so long as the information rate (channel code rate R) is less than channel capacity C [19]. The capacity of the mixed channel is determined by the following equation:

$$\begin{aligned} C &= C_{AWGN} (M/N) + C_{ERASURE} (K/N) \\ &= C_{AWGN} (1-R) + C_{ERASURE} R \quad \text{bits/symbol} \end{aligned} \quad (IV.1)$$

where,

C_{AWGN} is the Capacity of the AWGN channel.

$C_{ERASURE}$ is the Capacity of the erasure channel.

N is the number of bits in the codeword.

M is the number of parity bits.

K is the number of data bits.

We assume that the parity bits are transmitted through the relay channel using BPSK modulation and the deviation of the AWGN is σ . Then we have the following capacity expressions [19]:

$$C_{AWGN} = C_{BSC} \left(0.5 \left(1 + \operatorname{erf} \left(-1/\sigma\sqrt{2} \right) \right) \right) \quad (\text{IV.2})$$

$$C_{ERASURE} = 1 - P_{ERASURE}$$

where,

$$C_{BSC}(x) = 1 - (-x \log_2 x - (1-x) \log_2 (1-x))$$

$$P_{ERASURE} = 1, \text{ since only parity bits are transmitted}$$

When we substitute the above expressions into equation (IV.1) the second term disappears. After the substitution, we get an expression for capacity of the mixed channel, which still depends on the channel code rate R . Substituting the obtained expression of capacity in the inequality $R < C$, the condition for asymptotic reliable communication (in noisy channel) can be rewritten as:

$$R < C_{AWGN} / (1 + C_{AWGN}) \quad (\text{IV.3})$$

While designing degree distribution for the mixed channel with Gaussian deviation σ , we pick a code rate which satisfies the inequality given in (IV.3). Note, the channel code rate for a given degree distribution is:

$$R = 1 - \left(\sum_{i=2}^{m+1} \rho_i / i \right) / \left(\sum_{i=1}^n \lambda_i / i \right) \quad (\text{IV.4})$$

This equation is used as a constraint while determining the optimal degree distribution.

V. HOMOGENEOUS SENSOR NETWORKS AND THEIR CODE CONSTRUCTION

We can convert the sensor network in Figure 3 into a directed graph based on the range of transmission of each node. The out-degree of a sensor node is determined by the number of sensor nodes lying in its transmission range, whereas its in-degree is determined by the number of nodes, able to transmit data to it. The out-degree distribution plays a crucial role in the selection of the data-side degree distribution for NC, whereas the in-degree distribution constrains the choices for parity-side degree distribution. For our analysis, we will consider a homogeneous sensor network, i.e. a network where any sensor node v , will have the same number of nodes $|N_v|$ within its transmission range. Also every sensor node has the same number of nodes transmitting to it. This implies that the in-degree and the out-degree of the graph under consideration is the same. As we shall see in section VI, the neighborhood size $|N_v|$ plays a critical role in determining the feasibility of the degree distributions

during the optimization process. $|N_v|$ also influences the *coverage* of the sensor-nodes.

The system illustrated in Figure 3, can be represented as a line of sensor nodes, with neighboring nodes lying next to each other as shown in Figure 4. We choose this representation for ease in explaining the construction algorithm for the generator matrix G . The choice of a line representation does not result in any loss of generality for the homogeneous sensor network. The range of sensor transmissions determines the size of the neighborhood. The following algorithm lists how we can construct a parity generator matrix G , for the line sensor network given data and parity degree distributions, $\bar{\theta}$ and $\bar{\phi}$ respectively.

A. Algorithm to construct Parity Generator Matrix G

1. Create a list L_D of data nodes based on $\bar{\theta}$:
 - a. Set data-node start index $s \leftarrow 1$, and degree-index $i \leftarrow 1$.
 - b. Set data-node end index $e \leftarrow s + \operatorname{round}(K\bar{\theta}_i)$
 - c. Add i copies of each data-node s to e to the list L_D .
 - d. Update $s \leftarrow e + 1$.
 - e. Increment $i \leftarrow i + 1$. Go to step 1b if all the non-zero elements of data degree distribution $\bar{\theta}$ are not exhausted
2. Create a list L_M representing the degree of each parity node as follows:
 - a. Set parity-node start index $s \leftarrow 1$, and iterator $i \leftarrow 1$
 - b. Set parity-node end index $e \leftarrow s + \operatorname{round}(M\bar{\phi}_i) - 1$
 - c. Set elements s to e of list L_M to i ; where i represents the number of data-nodes XORed to generate the parity-bit.
 - d. Update $s \leftarrow e + 1$.
 - e. Increment $i \leftarrow i + 1$. Go to step 2b if all the non-zero elements of parity degrees distribution $\bar{\phi}$ are not exhausted
3. Now for each element in L_M create a list of associated data-nodes as follows:
 - a. Set parity node index $i \leftarrow 1$
 - b. Create an empty set of data-nodes $d \leftarrow \{\}$ associated with current parity-node i
 - c. Create a list L_{DQ} of unique data-nodes remaining in L_D
 - d. Select one data-node f at random from the list

- L_{DQ} . Add f to set d : $d \leftarrow d + \{f\}$
- e. Remove f from unique data-node list $L_{DQ} \leftarrow L_{DQ} - \{f\}$
 - f. Determine the neighborhood f^N , of incoming nodes associated with f and present in list L_{DQ} . Select the remaining $L_M(i) - 1$ data-nodes randomly from f^N and add them to set d .
 - g. Remove one copy of data-nodes in d from L_D
 - h. Update generator matrix $G(i) \leftarrow d$
 - i. Increment $i \leftarrow i + 1$. Go to step 3b if all the parity-bits have not been created, i.e. all the elements of L_M have not been processed.

4. Remove 4-cycles from generator matrix G

In step 1, of the algorithm we create a list containing copies of the data-nodes so that the fraction of nodes, having i copies, is equal to $\bar{\theta}_i$. This step ensures that we get close to $\bar{\theta}$, in the constructed G . In step 2, we determine the degree to be associated with each parity-equation such that, the fraction of parity-equations with degree i is $\bar{\phi}_i$. Step 3 randomly selects data-nodes to be associated with each parity-equation, the procedure is illustrated in Figure 4. The solid-line represents the randomly selected data-node f , whereas the dashed-lines, represent the remaining nodes selected randomly from the neighborhood f^N . Steps 1-3 result in a parity generator matrix G which has a distribution $(\bar{\theta}, \bar{\phi})$. However, this random-construction can result in four-cycles in the generator graph. Four-cycles have a detrimental effect on the decoding process at the sink, and therefore are removed in step 4 of the algorithm. This is achieved by deleting edges. If $x_{K \times 1}$ represents the data-vector then the parity-vector $p_{M \times 1}$ can now be generated as follows:

$$p_{M \times 1} = G_{M \times K} x_{K \times 1} \quad (\text{V.1})$$

The parity generator matrix G can be transformed into the parity check matrix H using the procedure outlined in section II, and used for belief-propagation decoding at the sink.

We must be careful while designing the degree distribution for channels which involve erasures, for the following reason. The formula to calculate the message to be transmitted from the check node c to the variable node v for a log-likelihood belief propagation decoder is given by:

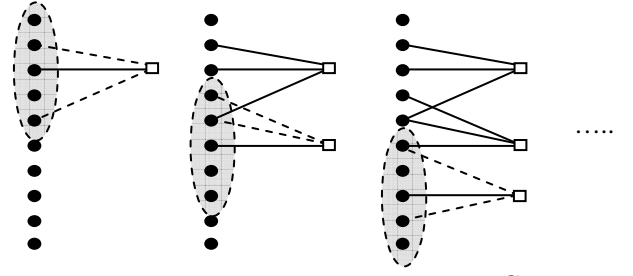


Figure 4: Step-3 in construction of G

$$L(c \rightarrow v) = \left(\prod_{a \in N_c \setminus v} \alpha_a \right) \left(\Psi \left(\sum_{a \in N_c \setminus v} \Psi(\beta_a) \right) \right) \quad (\text{V.2})$$

where,

$N_c \setminus v$ is neighborhood of c in parity check graph H , except the variable node v

α_a is the sign of the message coming from variable node a to c .

β_a is the magnitude of the message coming from variable node a to c .

$$\Psi(x) = \log \left(\frac{e^x + 1}{e^x - 1} \right)$$

An erased variable node will transmit zero to the check node, at the very first iteration. As can be seen from equation (V.2), if more than one variable node associated with the check node c sends a zero to it, then the check node c will transmit back a zero to all its variable nodes (due to the function Ψ). In our code formulation all the check nodes of the parity check matrix H are connected to data-bits, which are erased (as per problem definition). If the parity degree distribution $\bar{\phi}$ does not contain a non-zero fraction for degree-one nodes, then all the check-nodes of H , will have at-least two variable nodes with erasures connected to it. Therefore, to jump-start the BPA it is critical that there are some degree-one nodes in $\bar{\phi}$. These degree-one nodes create a ripple effect [18], i.e. if the data-nodes connected to degree one parity nodes are also connected to another parity node, then the belief of the degree-one parity node ripples down to the other parity-node via the

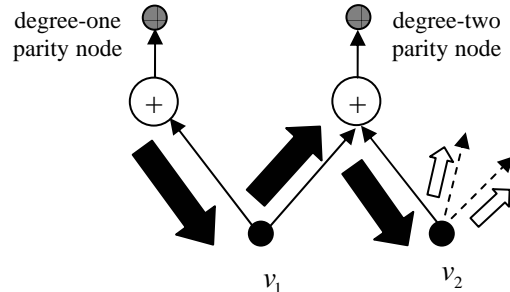


Figure 5: Belief ripple from degree-one parity node to degree-two parity node and beyond.

common data-node, as shown in Figure 5. If this parity node, in turn has degree-two, then the ripple propagates further to the next data-node. This ripple propagation via degree-one and degree-two parity nodes is essential for successful belief-propagation decoding for channels containing erasures. In fact, we need to make sure that there is at-least a fraction of the parity nodes which have degree-one and degree-two:

$$\begin{aligned} \overline{\phi}_1 &> \tau_1 \\ \overline{\phi}_2 &> \tau_2 \end{aligned} \quad (\text{V.3})$$

For our analysis, we choose this lower bound to be $\tau_1 = 0.005$, and $\tau_2 = 0.49$.

VI. BOUNDS ON ACHIEVABILITY/COVERAGE

While mapping a channel code to the underlying sensor network, the connectivity/topology of the actual network constraints the degree of freedom in selecting data/parity nodes. This constraint is reflected in step 3 of the code construction algorithm given in section V. Based on the graph connectivity we give the following bounds corresponding to the parity and data sides.

A. Parity side feasibility bound

Let the expected size of the neighborhood in the underlying sensor network be given by μ_N . For step 3 in the code construction algorithm to be successful, we need to select a node f , for which:

$$|f^n| \geq \text{number of data-nodes required} - 1$$

We relax this constraint to:

$$|N_f| \geq \text{number of data-nodes required} - 1 \quad (\text{V.4})$$

where, $|N_f|$ is the size of incoming-neighborhood of f . If the number of data nodes required is k , the probability of success is given by:

$$\begin{aligned} P(\text{success} | k) &= 1 - P(|N_f| < k - 1) \\ &= 1 - P(|N_f| < (1 - \delta)\mu_N) \end{aligned} \quad (\text{V.5})$$

$$\text{where, } \delta = 1 - (k - 1)/\mu_N$$

Assuming $(k - 1) < \mu_N$, the second term in eq. (V.5), can be bounded below, by using Chernoffs lower tail-inequality:

$$P(|N_f| < (1 - \delta)\mu_N) < \left(\frac{e^{-\delta}}{(1 - \delta)^{(1 - \delta)}} \right)^{\mu_N} \quad (\text{V.6})$$

Substituting (V.6) in (V.5) gives the lower bound for probability of success for a particular degree k . Therefore the lower bound on the probability of achieving the parity degree distribution is given by:

$$\begin{aligned} P(\text{success}) &= \sum_{\substack{\forall k \\ \text{non-zero } \overline{\phi}_k}} P(k)P(\text{success} | k) \\ P(\text{success}) &> \sum_{\substack{\forall k \\ \overline{\phi}_k \neq 0}} \overline{\phi}_k \left(1 - \left(\frac{e^{-\delta}}{(1 - \delta)^{(1 - \delta)}} \right)^{\mu_N} \right) \end{aligned} \quad (\text{V.7})$$

This inequality gives the lower bound on the probability of achieving a given parity distribution. To ensure any degree distribution selected by the optimization algorithm is achievable, we would like the probability of success to be at-least greater than ω_1 . We therefore introduce the following inequality as a constraint in the eventual optimization problem:

$$\sum_{\substack{\forall k \\ \overline{\phi}_k \neq 0}} \overline{\phi}_k \left(1 - \left(\frac{e^{-\delta}}{(1 - \delta)^{(1 - \delta)}} \right)^{\mu_N} \right) \geq \omega_1 \quad (\text{V.8})$$

We choose $\omega_1 = 0.9$ for our analysis. Interestingly, this bound deals only with expectations, and therefore can be applied even to non-homogeneous sensor networks with asymmetric transmission ranges. Due to the relaxation used in (V.4), this is an optimistic bound.

B. Data side coverage bound

The random construction of code given in section V, can result in parity generator graph G , in which some of the data nodes are left hanging. The BPA cannot recover these unconnected nodes, since the sink does not receive any information about them. The probability that a data node is left hanging depends on the size of the neighborhood (ngbd.) in the homogeneous sensor network. We can determine the probability that the data-node is connected, i.e. selected in step 3 of the construction algorithm as follows:

$$\begin{aligned} P(\text{data node } v \text{ selected}) &= \\ &P(\text{ngbd. selected in which } v \text{ lies}) \\ &* P(v \text{ selected} | \text{ngbd. selected in which } v \text{ lies}) \end{aligned} \quad (\text{V.9})$$

Now,

$$\begin{aligned} P(\text{ngbd. selected in which } v \text{ lies}) \\ = 1 - (1 - (|N_v| + 1)/K)^M \end{aligned} \quad (\text{V.10})$$

If the degree of the parity node being formed is k :

$$\begin{aligned} P(v \text{ selected} | \text{ngbd. selected in which } v \text{ lies}, k) \\ = k / (|N_v| + 1) \end{aligned} \quad (\text{V.11})$$

Again we have made use of the following relaxation: $|f^N| \approx |N_f| = |N_v|$. Using (V.11), we obtain:

$$\begin{aligned} P(v \text{ selected} | \text{ngbd. selected in which } v \text{ lies}) \\ = \sum_{\substack{\forall k \\ \overline{\phi}_k \neq 0}} \overline{\phi}_k P(v \text{ selected} | \text{ngbd. selected in which } v \text{ lies}, k) \\ \dots (\text{V.12}) \end{aligned}$$

Substituting (V.12) and (V.10), in (V.9) gives the probability of that an arbitrary node v is connected in the parity generator graph G . This probability is also the probability of successful coverage of sensor nodes, and we therefore introduce the following inequality constraint in the optimization problem formulated in section VII:

$$P(\text{data node } v \text{ selected}) = P(\text{coverage}) \geq \omega_2 \quad (\text{V.13})$$

In our analysis we choose $\omega_2 = 0.9$.

VII. MATHEMATICAL FORMULATION

We now setup the optimization problem for a generic non-homogeneous sensor network. We will restrict *density evolution* analysis to 25 iterations of the BPA. If we let the in-degree distribution of the sensor network be $\alpha = \{\alpha_1, \alpha_2, \dots, \alpha_n\}$ where α_i is the proportion of sensor nodes with in-degree i , and $\sum_{i=1}^n \alpha_i = 1$. This defines an upper bounds on the fraction of data-nodes with degree i as $\alpha_i^{\text{lim}} = \sum_{j=i}^n \alpha_j$. Similarly, if the out-degree distribution of the sensor network is $\beta = \{\beta_1, \beta_2, \dots, \beta_m\}$ where β_i is the proportion of sensor nodes with in-degree i , and $\sum_{i=1}^m \beta_i = 1$. This defines an upper bounds on the fraction of parity-nodes with degree i as $\beta_i^{\text{lim}} = \sum_{j=i}^m \beta_j$. The design problem can then be written as:

$$f(\lambda, \rho) = \min_{\lambda, \rho} P_e^{25} \quad (\text{V.14})$$

$$\text{where, } \quad 0 \leq \bar{\theta}_i \leq \alpha_i^{\text{lim}}, \forall 1 \leq i \leq n$$

$$0 \leq \bar{\rho}_i \leq \beta_{i-1}^{\text{lim}}, \forall 2 \leq i \leq m+1$$

Equations (IV.4), (V.3), (V.8) and (V.13), are additional constraints on the optimization problem. The standard deviation of AWGN for which the optimization is carried out is $\sigma = 0.82$, and the neighborhood size for the homogeneous sensor network is assumed to be $|N_v| = 18$. While performing the optimization we observed that limiting the parity and data degrees to 12, results in the constraints (V.8) and (V.13) being satisfied for the channel code rate $R = 1/3$.

As a rule of thumb, the initial guess value for the optimization problem is setup to be degree distributions, which are close to the desired rate. Moreover, we would like these initial guesses to have low average parity-degree, so that decisions in BPA can be made quickly on the check side. Also average data-degree is chosen to be high, so that the data-nodes receive correct information fairly quickly. We then use *sequential quadratic programming* method, along with line search and active

set strategy [20], to determine locally optimal solutions. The search-space for degree distributions is highly multimodal and therefore different initial points can result in significantly different degree distributions. One way to overcome getting stuck in local minima is by using techniques such as simulated annealing [21] or evolutionary computation [22]. We plan to explore these advanced techniques in future work.

VIII. EXPERIMENTAL SETUP AND RESULTS

Substituting, the AWGN deviation $\sigma = 0.82$, in equations (IV.2) and (IV.3) results in a channel capacity $C \approx 0.33$ as observed by the sink. We therefore design a channel code of rate $R = 1/3$. In a rate-1/3 code the number of parity-bits generated is twice the number of data-bits. The degree distribution obtained by solving the above optimization problem results, in codes which approach channel capacity, as the number of data-nodes goes to infinity; however there is a caveat to this statement which we shall discuss below. Since we select a finite number of data-nodes in our analysis, there is a loss in coding performance, when compared to asymptotic bounds given by *density evolution*.

The best known codes for erasure channels are LT codes. We select the LT distribution given in [18] as one

Table I: Parity and Data Degree Distributions

	Maximum degree 12	LT derived, max. 12	Maximum degree 6	LT
$\bar{\phi}_1$	0.0048	0.0048	0.0051	0.007969
$\bar{\phi}_2$	0.4900	0.4900	0.4900	0.493570
$\bar{\phi}_3$			0.0412	0.166220
$\bar{\phi}_4$	0.0183	0.0068	0.4276	0.072646
$\bar{\phi}_5$	0.0383	0.0239	0.0362	0.082558
$\bar{\phi}_6$		0.0989		
$\bar{\phi}_7$	0.0759			
$\bar{\phi}_8$	0.0089			0.056058
$\bar{\phi}_9$		0.0021		0.037229
$\bar{\phi}_{10}$	0.0496	0.0139		
$\bar{\phi}_{11}$	0.1164	0.1968		
$\bar{\phi}_{12}$	0.1978	0.1628		
$\bar{\phi}_{19}$				0.055590
$\bar{\phi}_{65}$				0.025023
$\bar{\phi}_{66}$				0.003135
$\bar{\theta}_6$			1.0000	
$\bar{\theta}_{12}$	1.0000	1.0000		1.0000

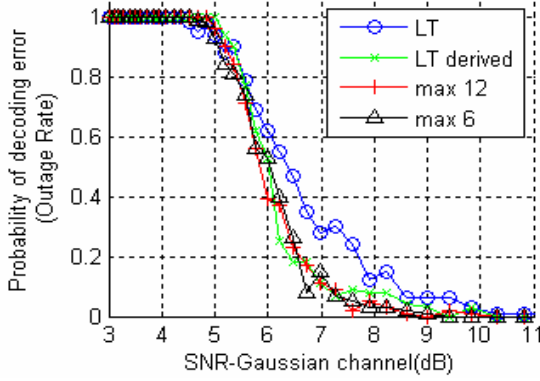


Figure 6: Outage Rate for designed degree distributions and LT distribution for 1000 data-nodes, $|N_v|=18$

of the distributions for our code construction. The distribution is listed in the fourth column of Table I. For LT-codes, the fraction of parity nodes which correspond to degrees greater than $|N_v|+1$, are included in degree $|N_v|+1$, during code construction. The first three columns of Table I represent the designed degree distributions obtained by solving the optimization algorithm given above. The first column represents the case when the data and parity node degrees are constrained to be less than equal to 12. The constraints (V.8) and (V.13), in the optimization problem, result in this upper bound on degree. For the second column, we artificially constraint the max. degree of the data/parity distributions to 6. This corresponds to a code, where the parity is formed relatively quickly as the receiving node has to wait for a fewer number of incoming information-bits. Again, the third column is a degree distribution whose maximum degree is constrained to 12, however this distribution was arrived at by choosing the truncated LT distribution as the starting point for the optimization algorithm.

Figure 6 depicts the performance of these codes over a sensor network containing 1000 data-nodes, and a neighborhood size of $|N_v|=18$. As can be seen for an *outage rate* of less than 0.1 the designed degree distributions have an advantage of about 1.5 dB when compared to the LT degree distribution. Moreover, all the designed codes have nearly identical performance. This suggests that there is considerable flexibility in choice of a good degree distribution for the sensor network. As seen in Figure 6, LT codes which are optimal for erasure channels suffer significantly due to the noise in the relay channel.

In general, the *outage rate* decrease as the SNR of the relay channel goes up. However, we see that the *outage rate* is not a monotonically decreasing function, and has

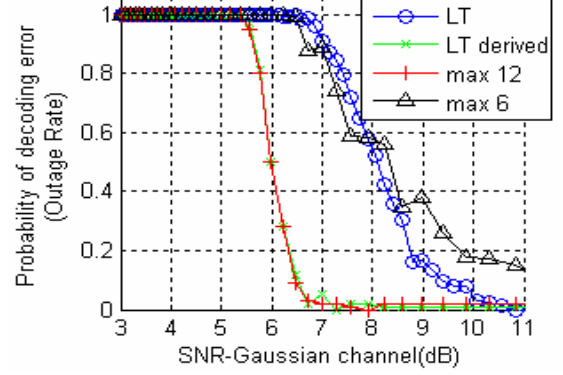


Figure 7: Outage Rate for designed degree distributions and LT distribution for 10,000 data-nodes, $|N_v|=18$

certain artifacts associated with it. This is because the constraints placed on neighborhood size, result in short cycles while constructing the code. Smaller is the ratio of the neighborhood size $|N_v|$ to data-bits K , more likely is the probability of getting a short cycles in parity generator graph G . This can be ascertained by examining the expression for probability of seeing a 4-cycle in the graph (assuming step-4 in code construction is not carried out):

$$P(4 \text{ cycle exists} | \bar{\phi}, |N_v|) \geq \left[1 - (1-z)^{\binom{k}{2}} \right]$$

$$\text{where, } z = e^{-2\alpha|N_v|}, \alpha = \sum_{i \geq 2} \bar{\phi}_i \left(\frac{i}{|N_v|+1} \right)^2, 0 < \alpha < 1$$

Note, the above expression can be easily obtained by first determining the probability (z) of a 4-cycle existing between any two arbitrary data-nodes of the parity generator graph (for large K). These short-cycles cause the artifacts seen in the performance curves shown in Figure 6. Also, increasing the number of data nodes, while keeping the neighborhood size the same, actually increases the probability of getting short cycles, and results in deterioration of the code performance. This is totally contrary to conventional wisdom in channel coding where an increase in codeword length, usually translates to improved performance. Figure 7, shows the performance curves for a sensor network with 10,000 data-nodes and $|N_v|=18$. This network has a smaller $|N_v|/K$ ratio, and shows worse performance for LT codes and codes with maximum degree 6. This indicates that sparser sensor networks perform worse when compared with denser sensor networks for the same degree distribution. Codes with maximum degree 12 and the *LT derived* codes show very small degradation in performance, and are better by 2.5 dB when compared to

LT distributed codes. Restricting neighborhood size to 18, results in a performance loss of about 2dB, when compared to the case when there is no constraint on neighborhood size i.e. $|N_v| = K - 1$. We do not reproduce the results for the unconstrained case here, for brevity. When $|N_v| = K - 1$, increasing the number of data-nodes improves the performance of the code slightly, as is observed in conventional channel coding.

IX. CONCLUSION

In this paper, we take advantage of the ability to map NC to channel coding, and use *density evolution* in designing good CNG network codes. We design codes for homogeneous sensor networks, with a finite neighborhood size. The connectivity of the WSN determines the feasibility of a given degree distribution. We determine lower-bounds on feasibility and include them as constraints in the code design problem. We design codes for relay channels which can be aggregated as AWGN channels; however the design procedure is generic enough to be used for any other symmetric channel, so long as the channel message densities are *consistent*. Interested readers can find a more thorough discussion on the *consistency condition* in [12]. We see that codes designed using the proposed approach, have lower *outage rates*, which translates to a SNR gain of 1.5 to 2.5 dB, when compared to LT distribution. All the designed codes have almost identical performance indicating there are several choices of good CNG network codes. The ratio $|N_v| / K$ plays a critical role in determining the probability of short cycles, for a given degree distribution. Presence of short cycles leads to degradation of code performance. Note, although the results presented here made the assumption that the sensor network is homogeneous, minor modifications of the bounds and the code construction algorithm, can allow design of CNG network codes for non-homogeneous networks.

REFERENCES

- [1] C. Perkins, "Ad Hoc On-Demand Distance Vector (AODV) Routing," IETF MANET, Internet-Draft, December 1997.
- [2] S. Singh, M. Woo, C.S. Raghavendra, "Power-aware routing in mobile ad-hoc networks," ACM/IEEE International Conf. on Mobile Computing and Networks (MOBICOM'98), pp. 76-84.
- [3] R. Ahlswede, N. Cai, S.-Y. R. Li and R. W. Yeung, "Network information flow," IEEE Trans. on Information Theory, vol. 46, pp. 1204-1216, 2000.
- [4] T. Ho, R. Koetter, M. Medard, D. Karger and M. Effros, "The Benefits of Coding over Routing in a Randomized Setting", IEEE ISIT, pp.442, 2003.
- [5] M. Medard, M. Effros, T. Ho, D. Karger, "On coding for non-multicast networks", In Proc. 41st Allerton Conf. on Communication, Control, and Computing, Oct. 2003.
- [6] X. Bao and J.(T). Li . " Matching Code-on-Graph with Network-on-Graph: Adaptive Network Coding for Wireless Relay Networks", In Proc. 43rd Allerton Conf. on Communication, Control, and Computing, Sep. 2005.
- [7] C. Hausl, F. Schreckenbach, I. Oikonomidis, and G. Bauch, "Iterative network and channel decoding on a tanner graph," In Proc. 43rd Allerton Conf. on Communication, Control, and Computing, Sept. 2005.
- [8] Judea Pearl. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, 1988.
- [9] R. G. Gallager, Low Density Parity-Check Codes. MIT Press, Cambridge, MA, 1963.
- [10] S.-Y. Chung, D. Forney, T. Richardson, and R. Urbanke, "On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit," IEEE Comm. Letters, v.5, pp.58-60, 2001.
- [11] M. Luby, M. Mitzenmacher, A. Shokrollahi, and D. Spielman, "Analysis of low density codes and improved designs using irregular graphs," IEEE Trans. Inform. Theory, v. 47, pp. 585-598, 2001.
- [12] T. Richardson, A. Shokrollahi, and R. Urbanke, "Design of provably good low-density parity check codes," Proc. IEEE ISIT, pp. 199, June 2000.
- [13] T. Richardson and R. Urbanke, "The capacity of low-density parity check codes under message-passing decoding," IEEE Trans. Inform. Theory, vol. 47, pp. 599-618, 2001.
- [14] T. Richardson, A. Shokrollahi, and R. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," IEEE Trans. Inform. Theory, vol. 47, pp. 619-637, 2001.
- [15] A. Orlitsky and J. Zhang, "Finite-length analysis of LDPC codes with large left degrees," in Proceedings of the International Symposium on Information Theory, 2002.
- [16] T. Richardson, A. Shokrollahi, and R. Urbanke, "Finite-length analysis of various low-density parity-check ensembles for the binary erasure channel," in Proc of ISIT, 2002.
- [17] S. ten Brink, "Convergence behaviour of iteratively decoded parallel concatenated codes," IEEE Trans. on Comm., vol. 49, Oct 2001.
- [18] Amin Shokrollahi "Raptor codes," IEEE Tran. on Information Theory, Volume-52, Issue-6, pp. 2551- 2567, June 2006.
- [19] T. M. Cover and J. A. Thomas, "Elements of Information Theory," Wiley Series in Telecommunications, 1991.
- [20] Gill, P.E., W. Murray, and M.H. Wright, Practical Optimization, London, Academic Press, 1981.
- [21] S. Kirkpatrick and C. D. Gelatt and M. P. Vecchi, "Optimization by Simulated Annealing," Science, Vol. 220, No. 4598, pp. 671-680, 1983.
- [22] Goldberg, David E, "Genetic Algorithms in Search, Optimization and Machine Learning," Kluwer Academic Publishers, Boston, MA, 1989.
- [23] S. Katti, R. Hariharan, W. Hu, D. Katabi, M. Médard, Jon Crowcroft, "XORs in the air: practical wireless network coding" ACM SIGCOMM Computer Communication Review, Vol. 36, No. 4, pp. 243-254, October 2006.