Maximizing Transported Data Before Partition in a Wireless Sensor Network

Samta Shukla, Siddhartha Sarma and Joy Kuri
Department of Electronic Systems Engineering
Indian Institute of Science, Bangalore, India
{samta,siddharth,kuri}@dese.iisc.ernet.in

Abstract—We consider a structured sensor network in which sensors and relays are arranged in a regular grid. Sources \(S_1, S_2, \ldots, S_K\) are located sequentially at the top row, and destinations \(D_1, D_2, \ldots, D_K\) are arranged in an arbitrary order at the bottom row; data from \(S_i\) is required at \(D_i\). Such networks serve as models for planned sensor deployments (for example, sensors deployed in a precision agriculture project). Nodes have limited energy, and our objective is to maximize the aggregate bits transported from source(s) to destination(s) before network partition. In contrast to formulations that seek to optimize either the network lifetime or the throughput, our criterion considers both aspects; the criterion is natural and novel. We show that our problem can be converted to a Linear Program. Next, we move on to consider relay nodes that not only forward data but are also capable of XOR-ing received packets. We study how this “coding capability” can be leveraged to improve the performance objective. We provide an algorithm that chooses the optimal number of coding points. Finally, we evaluate the additional benefit that XOR-ing can yield. For this evaluation, we formulate an optimization problem using “functional flow conservation.” Two XOR-ed packets get “fused” to one, because of which, at the coding points, the usual flow conservation notion does not apply; hence the need for functional flow conservation. We propose functional flow equations for directed acyclic graph networks; this is novel, as the current literature considers only tree graphs. Our results indicate that the gain in bits transported can be as much as 83%.

Keywords—Network Partition Time (NPT), Multiple Unicast, Network Coding, Maximal Independent Set (MIS), Functional Flow Conservation

I. INTRODUCTION

We study the question of maximizing the aggregate bits transported from sources to respective destinations across a sensor network before network partition due to node deaths. Several authors [1], [2] have considered the objective of network lifetime maximization, while others [3], [4] have focused on maximizing network throughput. However, maximizing each metric individually may not be enough. What is really of interest is the useful data transferred before at least one source loses connectivity to its destination. This motivates our criterion which considers the product of the two factors. We introduce a new metric, weighted aggregate bit transported before network partition which denotes the product of throughput and network lifetime. [5] reports an initial study with this performance criterion. However, in that work, the schedule of link activations was given and fixed; further, nodes were not capable of network coding. In this paper, we obtain the schedule of link activations as a solution to our problem, and also study how network coding capability can be exploited to improve our performance objective.

We consider a planned wireless sensor network deployment for multiple source-destination pairs. We assume that the nodes are arranged in a regular grid; this is for analytical tractability. More general networks are of interest and we are currently working on them. In this paper, we compare the maximum bits transported without coding to that obtained with network coding. The network coding scheme considered is limited to XOR codes only. We formulate optimization problems for both cases and convert them to linear programs; our results show that with coding the performance can improve by up to 83% of what was obtained without coding.

Apart from optimization formulations for maximizing transported bits, for the problem with XOR-capable nodes, we propose an algorithm to choose the minimum number of coding points and positions of coding nodes in the grid. To the best of our knowledge, there is no prior work related to efficiently using nodes in coding-capable networks.

Bitwise-XOR coding combines two different data types of the same length and generates a new data type of that length. Fragouli et al. [6], Traskov et al. [7], Ding et al. [8] referred to these newly generated flows as “conceptual flows,” “poisoned flows” and “virtual flows,” respectively, and asserted that the usual flow conservation does not hold good in this case. We use the notion of functional flow conservation on our tessellation which is a directed acyclic graph (DAG). Our key contribution is to propose functional flow conservation equations for directed acyclic graphs (DAG) with XOR operation where the earlier work was limited to tree-like graphs only [9].

The flavour of the work here is theoretical. We are interested in finding the maximum gain in data transported by using network coding. We consider a fluid data model with centralized mode of operation. The practical issue of scheduling the wireless links is analyzed using a model that uses the notion of maximal independent sets (MIS-es) on the network. Certainly, finding all possible MIS-es is a well known hard problem in graph theory; we assume that MIS-es are given to us.

Our model can find use in several application scenarios. In the agricultural domain, suppose that \(K\) plots of land need to be monitored remotely. These plots become \(K\) data sources, while the corresponding monitoring stations (destinations) that require data from all the \(K\) sources are placed some distance away. The intervening network of energy-limited sensors must be planned so as to convey data from source \(S_k\) to destination...
network partitions. Interested readers can refer to the papers on soil moisture monitoring in a semi-arid region in the state of Karnataka in South India [10]–[12].

Similarly, in a border surveillance or intrusion detection application [13], $S_1, S_2, \ldots, S_K$ source data to corresponding monitoring stations some distance away. One plausible approach to design such networks is by randomly deploying a large number of nodes between sources and destinations to ensure connectivity. But this is not an economical solution. We leverage network coding to perform the same task with fewer nodes in an energy-efficient way. In the rest of the paper, we use the term without network coding and routing interchangeably.

In Section II, we introduce the model and analyze the grid we study in this paper. Section III studies the problem of aggregate bit transport maximization on the grid with routing. In Section IV we explain the basic unit of the grid and propose algorithms to choose minimum coding points. Further, we formulate functional conservation flow equations and address the problem of maximizing transported bits with network coding in this section. Section V shows that XOR coding boosts the aggregate bits transported before partition. We conclude and discuss future scope of the problem in Section VI.

II. SYSTEM MODEL

We model the sensor network as a regular structured network arranged in the manner shown in Fig. 1, and refer to it as NCoNET in the rest of the paper. In this abstraction, sources $S_1, S_2, \ldots, S_K$ are present at the top edge, and $K$ monitoring terminals are placed at the bottom edge. At a given instant, a monitoring terminal needs to receive data from exactly one of the sources; however, the identity of the source node can change over time. For example, over some interval $(t_1, t_2)$, terminal $j$ needs to receive data from source $S_k$ (say); later, over some interval $(t_3, t_4)$, terminal $j$ needs to receive data from source $S_m$ (say). Thus, at a given instant, for $1 \leq i \leq K$, source $S_i$’s data must reach the terminal at position $D_i$, where $\{D_1, D_2, \ldots, D_K\}$ is a permutation of $\{1, 2, \ldots, K\}$.

Formally, NCoNET is a graph $G = (N, L)$. The total number of nodes is $N = |N|$, and the total number of links is $L = |L|$. Let $w_1, w_2, \ldots, w_K$ be “weights” associated with $S_1, S_2, \ldots, S_K$, where the weights sum to unity. The weight assigned to a source reflects its priority. Considering the intermediate nodes, i.e., the nodes apart from the $K$ sources $S_1, S_2, \ldots, S_K$ and the corresponding destinations $D_1, D_2, \ldots, D_K$, let $E_i$ denote the energy available with node $i$ (in Joules). We denote by $\gamma_i$ the energy consumed for transmitting a data bit (Joules/bit) and by $\gamma_r$ the energy consumed for receiving a data bit (Joules/bit). The values are the same for all nodes.

The network topology can be represented by a $N \times L$ node-link incidence matrix $A$, with the entries $A_{nl}$, $n \in N, l \in L$ given by:

$$A_{nl} = \begin{cases} 1, & \text{if } l \text{ originates at } n \\ -1, & \text{if } l \text{ terminates at } n \\ 0, & \text{otherwise.} \end{cases}$$

Fig. 1: Regular grid consisting of the basic units aligned together for $K$ S-D pairs with an arbitrary destination permutation. Rows and columns are as indicated in the figure.

Network partition time (NPT), denoted by $\hat{T}$, is defined as the instant when the first source-destination pair disconnects (it is possible that several sources are disconnected from their respective destinations at this time). We consider the network to be partitioned because there is at least one source that is unable to reach its destination. The basic model assumptions are as follows:

- Sources continuously generate fluid data.
- There is a centralized scheduler with a global view of the network which decides the set of links to be activated at any instant.
- Each node has a single radio interface, i.e., a node cannot transmit and receive at the same time.
- Static network topology; negligible propagation delay.

III. BIT MAXIMIZATION WITHOUT NETWORK CODING

In this section we study the problem of transported bits maximization on NCoNET with routing.

Scheduling of Wireless Links

In a wireless network, links that interfere with one another cannot be activated simultaneously. To address link interference, the notions of Link Contention Graph (LCG) and Maximal Independent Set (MIS) have been introduced [14]. In the LCG, each vertex represents a link, and there is an edge between two vertices if the corresponding links interfere. A maximal subset of non-adjacent nodes in the LCG defines a MIS. Evidently, the links in an MIS can be activated simultaneously because they do not interfere.

A schedule of wireless link activations can be viewed as a sequence of MIS-s that are activated in sequence.

Let the bit rate supported by link $l$ be denoted by $C_l$, $l = 1, 2, \ldots, L$. Let $M_1, M_2, \ldots, M_W$ denote the various MIS-s for the given graph, let $\alpha_l$ denote the fraction (of the period of the schedule) for which the $i$th MIS is on. A link $l \in L$ can
be present in any of the MIS-s (not necessarily disjoint). The fraction of time for which link \( l \) is active is obtained by

\[
u_l = \sum_{i \in M_l} a_{i_l}.
\]

The equivalent link capacity for link \( l \) is given by: \( C^l = \nu_l \times C_l \). As the equivalent link capacity \( C^l \) is obtained by taking into account the fraction of time for which the link is active, for purposes of evaluating bits transported, we can substitute this link by an equivalent \textit{wire}d link of capacity \( C^l \) that is active all the time. Evidently, the bits transported by the link over the schedule period remains the same, no matter which view is adopted. This observation forms the basis underlying our approach of constructing an \textit{equivalent wired network}.

\textbf{Note:} In the rest of the paper, we assume \textit{lossless} transmissions (for simplicity). However, our analysis can be easily extended to the lossy case where corruption losses will result in \textit{derated link capacities}.

\section{A. Multi-Commodity Flow Problem}

\textbf{The Linear Programming Problem Formulation:} Sets of end-to-end multi-hop flows constitute the network traffic, where each flow has a source and a destination. Traffic is routed by the network along one or multiple paths from source to destination.

For each flow, we denote by \( x^i \) the amount of aggregate flow (measured in bits/s) injected into the network per unit time by the \( i^{\text{th}} \) source, \( S_i \), \( y^l_i \) denotes the amount of flow traversing link \( l \) per unit time. We represent by \( y^l_i \), the flow through link \( l \) due to the \( S_i - D_j \) pair; \( \hat{y}^l_i \) denotes the vector of link flows due to the \( S_i - D_j \) pair; and \( y \) represents the \( L \times K \) link flow matrix, with each element being \( y^l_i \), \( l \in \mathcal{L}, i = 1,2,\ldots,K \).

Unless otherwise stated, we will assume that vectors are always column vectors. For a vector \( z \), \( z^T \) denotes transpose of \( z \).

\textbf{Flow Constraint:} Assuming lossless transmission, the flow conservation law leads to:

\[
\begin{bmatrix}
    y^1_i \\
    \vdots \\
    y^K_i
\end{bmatrix}_{N \times L} \times \begin{bmatrix}
    y^1_L \\
    \vdots \\
    y^K_L
\end{bmatrix}_{L \times K} = \begin{bmatrix}
    x^1 & 0 & 0 & 0 \\
    0 & x^2 & \cdots & \vdots \\
    \vdots & \ddots & \vdots & \vdots \\
    0 & \cdots & 0 & x^K \\
    \vdots & \ddots & \vdots & \vdots \\
    0 & \cdots & 0 & -x^K
\end{bmatrix}_{N \times K}
\]

In the \( N \times K \) matrix on the right, we have nonzero entries corresponding to the source and destination nodes only. We define \( u^i_n = (u^i_n), n \in \mathcal{N} \), where \( u^i_n \) denotes the amount of flow injected (removed) to (from) the network at node \( n \), \( i \).

\[
u^i_n = \begin{cases} 
x^i, & \text{if } n = S_i \\
-x^i, & \text{if } n = D_i \\
0, & \text{otherwise}
\end{cases}
\]

Hence, the flow equations can be compactly written as

\[
A_i^I y_i = u^i, \forall i = 1,2,\ldots,K
\]

\textbf{Energy Expenditure Constraint:} The amount of data transmitted and received by any node is limited by the maximum energy available. Let \( \hat{A} \) denote the node-link incidence matrix with elements \( \hat{A}_{nl}, n \in \mathcal{N}, l \in \mathcal{L} \), constructed as follows: \( \hat{A}_{nl} = \begin{cases} 
\gamma_{rl}, & \text{if } A_{nl} = 1 \\
0, & \text{otherwise}
\end{cases} \)

Recall that the Network Partition Time \( \hat{T} \) is the time when the first source-destination pair disconnects. We have:

\[
\begin{bmatrix}
\hat{A} \times y \times 1 \\
\hat{y}^1_i \\
\vdots \\
\hat{y}^K_i
\end{bmatrix}_{N \times L} \times \begin{bmatrix}
1 \\
\vdots \\
1
\end{bmatrix}_{K \times 1} \leq \begin{bmatrix}
E^1 \\
\vdots \\
E^K
\end{bmatrix}_{N \times 1}
\]

or

\[
\hat{A} \times y \times 1 \times \hat{T} \leq E
\]

where \( E \) is the vector representing the energy available with each node in the network.

\textbf{Remark:} Our formulation can accommodate nodes that have access to unlimited energy — we set \( E_n = \infty \) for any such node.

\textbf{Capacity Constraints:} The link capacity corresponds to the reduced equivalent capacities obtained in Section II. We have:

\[
\sum_{i=1}^{K} y^l_i \leq C^l, \forall l \in \mathcal{L}
\]

\textbf{Problem Formulation}

Our aim is to maximize the (weighted) aggregate bits carried till NPT. Let \( w_i, i = 1,2,\ldots,K \), represent the weights associated with the \( K \) source-destination pairs; the weights reflect the “importance” of each source-destination pair. Then the problem can be stated as:

\textbf{Problem P*:} \[
\max \sum_{i=1}^{K} w_i x^i \hat{T}
\]

such that:

\[
A_i^I y_i = u^i, \quad i = 1,2,\ldots,K \quad (1b)
\]

\[
\hat{A} \times y \times 1 \times \hat{T} \leq E \quad (1c)
\]

\[
\sum_{i=1}^{K} y^l_i \leq \sum_{j|l \in M_j} a_j \times C_l, \quad \forall l \in \mathcal{L} \quad (1d)
\]

\[
y^l_i \geq 0, \quad l \in \mathcal{L}, i = 1,2,\ldots,K \quad (1e)
\]

\[
x^i \geq 0, \quad i = 1,2,\ldots,K \quad (1f)
\]

\[
0 < \sum_{j=1}^{W} a_j \leq 1 \quad (1g)
\]

\[
\hat{T} > 0, \quad a_j > 0, \quad \forall j = 1,2,\ldots,W \quad (1h)
\]

The formulation above is not a linear program owing to the presence of the product of unknowns \( x^i \cdot \hat{T} \) in the objective function. However, we can convert this problem to a Linear Program in terms of the new variables \( (x^{(i)}_l \hat{T}) \)
From these two inequalities, the claim follows. In this section, nodes are coding capable, i.e., if a node is selected as a coding point, and receives data from two distinct links, it XORs the data before forwarding.

\[ \begin{align*}
\text{Problem MCF:} & \quad \max \sum_{i=1}^{K} w_i(x^i T) \\
\text{such that:} & \quad A((x^i)^T) = (u^i)^T, \quad i = 1, 2, \ldots, K \\
& \quad A((y^i)^T) \times 1 \leq E \\
& \quad (\sum_{i=1}^{K} y_i^j T) \leq \sum_{j \in M} (a_i^j) C_i, \forall l \in L \\
& \quad y_i^j T \geq 0, \forall l \in L, \quad i = 1, 2, \ldots, K \\
& \quad (x^i T) \geq 0, \quad i = 1, 2, \ldots, K \\
& \quad \hat{T} \geq 0 \\
& \quad a_j^j > 0, \quad j = 1, 2, \ldots, W \\
& \quad 0 < \sum_{j=1}^{W} (a_j T) \leq \hat{T}
\end{align*} \]

Lemma 1: The optimal objective function values in Problem P* and Problem MCF are equal.

Proof: Let \( \hat{T}^*, x^*, i = 1, 2, \ldots, K, \underline{y}_i^*, i = 1, 2, \ldots, K, \underline{a}_j, j = 1, 2, \ldots, W \), represent the optimal solution to Problem P*. We generate the following point which is feasible for Problem MCF: \( \hat{T} = \hat{T}^*, (\underline{x}^i \hat{T}) = x^i \hat{T}^*, i = 1, 2, \ldots, K, (\underline{y}_i \hat{T}) = \underline{y}_i \hat{T}^*, i = 1, 2, \ldots, K, \underline{a}_j \hat{T} = a_j \hat{T}^* \). Clearly, the point \((\hat{T}, (\underline{x}^i \hat{T}), (\underline{y}_i \hat{T}), (\underline{a}_j \hat{T}))\) is feasible for Problem MCF as the point \((\hat{T}^*, x^*, \underline{y}_i^*, \underline{a}_j^*)\) is feasible for Problem P*, and the objective function for Problem MCF at \((\hat{T}, (\underline{x}^i \hat{T}), (\underline{y}_i \hat{T}), (\underline{a}_j \hat{T}))\) is equal to the optimal objective function for Problem P*. Therefore,

\[ \sum_{i=1}^{K} w_i (\underline{x}^i \hat{T}^*) \geq \sum_{i=1}^{K} w_i x^i \hat{T}^* \]

Similarly, starting from the optimal solution \((\hat{T}^*, (\underline{x}^i \hat{T})^*, (\underline{y}_i \hat{T})^*, (\underline{a}_j \hat{T})^*))\) to the Problem MCF, we can generate a feasible point for the Problem P*, and the objective function value of Problem P* at this point will equal the optimal objective function value of Problem MCF. This implies that

\[ \sum_{i=1}^{K} w_i (\underline{x}^i \hat{T}^*) \leq \sum_{i=1}^{K} w_i x^i \hat{T}^* \]

From these two inequalities, the claim follows.

We will consider example networks and discuss results in Section V.

IV. Bit Maximization with Coding

In this section, nodes are coding capable, i.e., if a node is selected as a coding point, and receives data from two distinct links, it XORs the data before forwarding.

Ideally, one would like to solve the joint problem of locating coding points and adjusting flows so as to maximize the objective. However, this joint problem is difficult. So, our approach is to break up the joint problem into two pieces. First, we focus on locating the coding points so that the fewest number of coding points are required. Then, with the coding points located, we study the problem of maximizing the aggregate bits transported.

A. Understanding Network Coding in NCoNET

1) The Basic Unit: The basic unit of the repetitive pattern is shown in Figure 2. Pairs of nodes within hearing range are shown as being connected by links. Transmissions from \( S_1 \) are received at \( R \) as well as \( D_2 \). Similarly, transmissions from \( S_2 \) are received at \( R \) as well as \( D_1 \). With traditional routing, four transmissions are required to provide data to destinations from corresponding sources: \( S_1 \rightarrow R \rightarrow D_1 \) and \( S_2 \rightarrow R \rightarrow D_2 \). On the other hand, if \( R \) has the option of employing some form of network coding — in our study, the bitwise XOR coding — only three transmissions were enough: the source nodes broadcast their data as \( S_1 \rightarrow \{ R, D_2 \} \), and \( S_2 \rightarrow \{ R, D_1 \} \) (two transmissions) and the third transmission is the broadcast by the relay node \( R \), which XORs the received data and sends it as \( R \rightarrow \{ D_2, D_1 \} \). Both the destinations obtain the intended data after XOR-ing the operands received. Thus, with nodes capable of XOR-ing, we save one transmission per basic unit.

Remark: Observe that, when \( D_1 \) is below \( S_1 \) and \( D_2 \) is below \( S_2 \), or in other words, when destinations are sequentially aligned in the basic unit, network coding is not required.

We model broadcast links as distinct links that carry equal data flows through them, and have equal capacities at each instant. Note that the same link may appear as an in-link for several nodes. For example, in Figure 2, the links \( \{ a, b, c \} \) are broadcast links (the links shown in red, blue and green, respectively). Two distinct links represent the broadcast link \( a \). Both these links carry the same data at the same rates. The links in \( a \) are in-links for both \( D_2 \) and \( R \). Likewise for links \( b \) and \( c \).

Fig. 2: The basic unit of the regular grid NCoNET. It consists of 2 sources, 1 relay node and 2 destinations.

2) Data flow in NCoNET: A new data operand in NCoNET is generated at every source, and by every node that XORs two different operands. Every operand thus generated has its own identity, we refer to it as the type of the data operand. We denote the type of an operand by \( \theta \) and the set of all such operands as \( \Gamma \). For a link \( l \), we define \( tail(l) \) and \( head(l) \) as the nodes \( v \) and \( u \) such that \( l \) is the directed link from \( v \) to \( u \). Let \( \phi_\Gamma(\theta) \) denote the successor of operand \( \theta \), i.e. \( \phi_\Gamma(\theta) \equiv \{ \eta \in \Gamma : tail(\eta) = head(\theta) \} \). We define \( In(v) \equiv \{ l \in L :
head(l) = v} and Out(v) \triangleq \{l \in L : \text{tail}(l) = v\}. The flow through link l of operand \( \theta \) is denoted by \( f_\theta^l \).

We introduce some definitions that are going to be used throughout. The rows and columns for NCoNET are defined as follows:

**Definition 2:** **Row:** A row in a grid is a horizontal line in which all the sources or the nodes carrying operands from the sources are aligned. In Figure 1, the first row is highlighted in green.

**Definition 3:** **Column:** A column in a grid is a vertical line connecting a source and destination where the destination may or may not be the intended destination for that source. Figure 1 indicates the fourth column in brown.

Figure 1 shows \( P \) rows in all; the row count starts from 0 which corresponds to the line where all sources are aligned. Given \( K S - D \) pairs, the total number of columns becomes \( K \). The indexing for columns starts from one.

3) **Coding Points on the Grid:** NCoNET is a two-dimensional grid, where each coding node is referred to using its \( (r, c) \) co-ordinates: \( r \) indicates the row number (increasing downwards) and \( c \) denotes the column number (increasing from left to right). The sources are placed on the top row (\( 0^{th} \) row) with their coordinates being \((0, 1)\) for \( S_1 \), \((0, 2)\) for \( S_2 \), and so on, with \((0, K)\) for \( S_K \). The coding points in the grid are located at the co-ordinates of the form \((0.5 + R, 0.5 + C)\), where \( C \) denotes the \( C^{th} \) column, and \( R \) is the \( R^{th} \) row.

Bitwise-XOR imposes restrictions on where coding can occur: Coding points can be located in non-adjacent columns only, i.e., the columns which are two units apart from each other. For example, between any two rows, the coding points in columns 1.5 and 2.5 cannot be chosen simultaneously; however, those in non-adjacent columns 1.5 and 3.5 can be chosen. If coding is performed in two adjacent columns, then the destination node at the center will receive data from three nodes — the two adjacent coding nodes and the source in center — making decoding difficult.

If the destination permutation is such that each \( D_i \) appears directly beneath \( S_i \), \( i = 1, 2, \ldots, K \), then we refer to such a grid as the direct NCoNET. For the direct NCoNET, coding renders no additional benefit; shortest path routing itself is the key. Whenever a basic unit occurs in the network, it is important to understand the coding operation over a basic unit. The key is: whenever a basic unit occurs in the network, the sources’ data is flipped.

**B. Choosing Coding Points for \( K S-D \) pairs**

1) **Gradient Algorithm:** This algorithm takes \( D \) as input, where \( D \) is the required destination permutation. We define the vector \( d := (d(1), d(2), \ldots, d(K)) \), where \( d(i) \) denotes the column wise separation between \( D_i \) and \( S_i \). Note that the elements in \( d \) are either positive (if the source is located at column \( c_1 \) and the destination at column \( c_2 \), with \( c_2 > c_1 \)), or negative (if \( c_2 < c_1 \)) or zero (if both lie on the same column, i.e., \( c_1 = c_2 \)). A coding point at location \( p \) is selected between two source operands \( a \) and \( b \) if their “gradient” \( g(p) := d(b) - d(a) \) is negative. Several coding points may be selected in a row, if they do not violate the “consecutive columns” constraint. If the consecutive columns constraint is such that only one coding point can be selected, the point corresponding to the smallest (most negative) gradient is chosen. Ties are resolved arbitrarily. New rows are added to the grid till the \( d \) vector reduces to a zero-vector.

**Example to illustrate the gradient algorithm:** Refer to Figure 3. To retain clarity, we show the working on the grid with destination permutation \( D_1, D_2, D_1, D_2 \). The values of the distance parameters and the gradients are as follows:

**Row 0:** \( d(1) = 2, d(2) = 2, d(3) = -1, d(4) = -3 \)

**Row 1:** \( g(1) = 0, g(2) = -3, g(3) = -2 \)

**Row 2:** \( g(2) \) is the smallest and even though both \( g(2) \) and \( g(3) \) are negative, the two coding points cannot be operational simultaneously (consecutive columns). So, the algorithm selects the 2\(^{nd} \) coding point. After this, the operands from \( S_2 \) and \( S_3 \) are flipped.

**Row 3:** \( d(3) = 1, d(4) = -1, d(1) = 0, d(2) = 0 \)

The algorithm selects the 2\(^{nd} \) coding point.

Finally, we get the arrangement of coding points as shown in Figure 3.

2) **Minimum Number of Coding Points:** To find the minimum number of coding points required to achieve a given permutation, we consider the following problem: Given an arbitrary permutation of the integers \( 1, 2, \ldots, K \), arrange them in sequence using the minimum number of swaps. Figure 4 shows the 10 swaps involved in arranging the given permutation 54321 in sequence, viz., in ascending order. Evidently, a swap corresponds to a coding point because the coding point results in an interchange of the two operands involved. The approach is to get the number 1 in its correct position by successive swaps. With 1 in place, we have a smaller subproblem to solve; The remaining numbers must be arranged in the sequence 2,3,4,5. Iterating in this way, we find that the total number of swap operations required is \( K(K - 1)/2 \). Clearly, the algorithm is essentially Bubble Sort, and its complexity is \( \mathcal{O}(K^2) \).

**Remark:** It can be shown analytically that the Gradient...
C. Maximizing Bits Transported with Network Coding

We formulate the problem of maximizing the aggregate bits carried before network partition. When network coding is employed, new operands are generated at a node as a result of fusing two operands. For example, a “1” and a “0” are XOR-ed to yield a single “1;”

\[ \text{C. Maximizing Bits Transported with Network Coding} \]

The amount of input operand \( x \) is viewed as a new type of operand, denoted by \( \hat{x}_1 \), that is generated at \( D_2 \) and expelled from the network at \( D_2 \). Also, \( D_2 \) forwards the uncoded \( x_2 \) that comes via the broadcast link \( l_3 \) along with the decoded \( x_2 \) obtained. A similar process occurs at \( D_1 \).

For each operand type \( \theta \) flowing into a node \( v \), all the operand types that may be generated because of processing at node \( v \) are the “successor operands” of \( \theta \), and are collected together in the set \( \Phi_v(\theta) \).

A new operand is generated in NCoNET either by a source (1 to K) or an intermediate coding point. Hence, the number of types of operands in the grid is \( K + \text{no of coding points} \). We denote the data generated at source \( S_\theta \) of any arbitrary basic unit of NCoNET as \( \lambda_\theta \). The optimization problem with bit maximization as the objective is formulated as follows:

**Problem P1:**

\[ \max_{\theta=1}^{K} \sum_{\eta=1}^{\Phi_v(\theta)} w_{\theta}\lambda_{\theta}\hat{T} \]

such that:

1. At every node of the basic unit:

\[ \sum_{l \in \text{Out}(v)} f_{l}^{\theta} + \sum_{l \in \text{Out}(v)} f_{\phi}^{\theta} - \sum_{l \in \text{In}(v)} f_{l}^{\theta} = 0, \]

\[ \forall \theta \in \Gamma \text{ such that } \eta \in \Phi_v(\theta) \]

2. Conservation of operands at the destination of the basic unit (shown as blue diamonds in Figure 5 (a)):

\[ \sum_{l \in \text{Out}(v)} f_{l}^{\theta} = \lambda_{\theta}, \quad \text{where } v = D_\theta \]

3. Generation of operands in sources of basic unit:

\[ \sum_{l \in \text{In}(v)} f_{l}^{\theta} = \lambda_{\theta}, \quad \text{where } v = S_\theta \]

4. Energy Constraint:

\[ \left( \sum_{l \in \text{In}(v)} f_{l}^{\theta} + \sum_{l \in \text{Out}(v)} f_{l}^{\theta} \hat{\gamma}_l \right) \hat{T} \leq E_v, \forall v \in N, \forall \theta \in \Gamma \]

5. Capacity constraints:

\[ \sum_{l \in \text{Out}(v)} f_{l}^{\theta} \leq \sum_{l \in \text{Out}(v)} a_{l}C_{l}, \forall l \in L \]

6. Non-negativity constraints:

\[ \hat{T} \geq 0, \lambda_{\theta} \geq 0, f_{l}^{\theta} \geq 0, \forall l \in L \text{ and } \forall \theta \in \Gamma \]

Functionality of flow in the first constraint holds for every operand of type \( \theta \). One or more types of \( \eta \) operands are produced from \( \theta \). The amount of input operand \( \theta \) at a node equals the sum of \( \{i\} \) the amounts of operands of types

**Swapping puzzle:** To obtain 12345 from 54321 in minimum number of swaps

4 swaps

3 swaps

2 swaps

1 swap

Total = 10 swaps

**Fig. 4:** Arranging five numbers in a line using minimum number of swaps

Algorithm chooses the optimal number of coding points. We omit the proof owing to lack of space.

R generates a new operand type \( x_3 := x_1 \oplus x_2 \). In general, we have three types of operands transmitted by \( R \) on link \( l_3 \): \( x_1, x_2 \) and \( x_3 \). This is because \( R \) chooses equal parts of \( f_{l_3}^{x_1} \) and \( f_{l_3}^{x_2} \), combines them to generate \( f_{l_3}^{x_3} \), and also forwards the remaining “un-coded” data with the coded data. We have,

\[ -f_{l_3}^{x_1} + f_{l_3}^{x_2} + f_{l_3}^{x_3} = 0 \]

\[ -f_{l_3}^{x_2} + f_{l_3}^{x_1} + f_{l_3}^{x_3} = 0 \]

Node \( D_2 \) receives \( x_1 \) directly over \( l_1 \), and \( x_3 \) over \( l_3 \); further, \( D_2 \) may receive \( x_1 \) (or \( x_2 \)) over \( l_3 \) as well, in case \( R \) has forwarded part of \( x_1 \) (or \( x_2 \)) un-coded over \( l_3 \). The processing at \( D_2 \) is as follows: \( D_2 \) takes equal amounts of \( x_3 \) and \( x_1 \) and uses them to generate the same amount of \( x_2 \) — this is the decoding operation. Any excess \( x_1 \) flowing into \( D_2 \) is “consumed” at \( D_2 \), and plays no further part in the network. The “excess flow” of type \( x_1 \) is viewed as a new type of operand, denoted by \( \hat{x}_1 \), that is generated at \( D_2 \) and expelled from the network at \( D_2 \). Also, \( D_2 \) forwards the un-coded \( x_2 \) that comes via the broadcast link \( l_3 \) along with the decoded \( x_2 \) obtained. A similar process occurs at \( D_1 \).

In Figure 5 (a), let \( x_1 \) and \( x_2 \) denote the operands sent into the network from \( S_1 \) and \( S_2 \), respectively. The rates of operands flowing into \( R \) are denoted as \( f_{l_3}^{x_1} \) and \( f_{l_3}^{x_2} \); these need not be equal in general.

**Fig. 5:** Graph showing link flows for the basic unit: (a) With Network Coding: the gray inverted triangles are sinks for unused flows, broadcast links are shown in the same colour. (b) Without Network Coding: 4 unicast transmissions are shown.
\( \eta \) which are generated at the node (by coding) and (ii) the unused uncoded operand forwarded by that node. Figure 5 (a) shows a sink attached to the destination of the basic unit to destroy excess amount of an operand, if any. The second constraint captures the consumption of flows at destinations. Every destination forwards the decoded operand denoted as \( \lambda_0 \), and sinks the unused operands. The third constraint captures the generation of flows at sources. The fourth, fifth and sixth constraints are self explanatory.

Solving the Optimization Formulations: As in 2a, we can formulate a linear program for this scenario also. For the sake of completeness, it is presented below:

\[
\text{Problem NCF:} \quad \max \sum_{\theta \in \Gamma} \sum_{l \in \text{Out}(v)} w_\theta (f_l^\theta \bar{T}) + \sum_{l \in \text{In}(v)} \sum_{l \in \text{Out}(v)} (f_l^\theta \bar{T}) - \sum_{l \in \text{In}(v)} (f_1^\theta \bar{T}) = 0, \quad \forall \theta \in \Gamma \text{ such that } \eta \in \Phi_\eta(\theta)
\]

\[
\sum_{l \in \text{Out}(v)} (f_l^\theta \bar{T}) = (\lambda_0 \bar{T}), \quad \text{where } v = D_0
\]

\[
\sum_{l \in \text{In}(v)} (f_l^\theta \bar{T}) = (\lambda_0 \bar{T}), \quad \text{where } v = S_0
\]

\[
\sum_{l \in \text{In}(v)} (f_l^\theta \bar{T}) \leq \sum_{l \in \text{Out}(v)} (a_i \bar{T}) C_i, \quad \forall l \in \mathcal{L}
\]

\[
0 \leq \sum_{j=1}^{K} (a_j \bar{T}) \leq \bar{T}, \quad \bar{T} \geq 0
\]

(\( \lambda_0 \bar{T} \geq 0, (f_l^\theta \bar{T}) \geq 0, \forall l \in \mathcal{L} \) and \( \forall \theta \in \Gamma \))

Lemma 4: The optimal objective function values in Problem P1 and Problem NCF are equal.

Proof: The proof is similar to that of Lemma 1. \( \blacksquare \)

V. RESULTS: PERFORMANCE COMPARISON

We obtain routes and the aggregate bits transported on NCoNET with routing using the formulations proposed in 2a. Then, we execute the Gradient Algorithm to find coding points on the grid, and solve the optimization formulation in Problem NCF to find the total bits transported with coding.

We construct grids of varying sizes, with 2, 3, 4 and 5 \( S-D \) pairs to study the problem of bit maximization. Given a grid, the algorithm proposed in Section IV-B selects a set of coding points depending on the required destination permutation. This gives a specific graph structure on which the functional flow conservation equations (with coding) are written. We introduce a metric, coding gain, defined as the ratio of the number of bits transported with coding to that of bits transported without coding.

Numerical solutions to the problem on NCoNET are found using MATLAB and tabulated in Table I. Results show that, with certain weight combinations, the percentage improvement in bits transported can be as high as 83 %, which corresponds to a coding gain of 1.83 (values are shown in bold). We observe that, when a single source is prioritized over others (with weight 1, while the rest have 0), the bits obtained with and without coding are the same.

The coding gain of 1.83 is the maximum gain possible on any S-D pair. This can be explained from the very basic analysis of routing versus coding. Consider a node receiving two different flows which have to be forwarded. With routing, there will be two receptions and two transmissions; the node will spend a total of \( 2(\gamma_r + \gamma_t) \) units of energy per bit. With coding, the node receives two individual flows, fuses them to transmit a single flow; hence, in this case, the node spends \( (2 \gamma_r + \gamma_t) \) units of energy per bit. For the numerical study, we have chosen \( \gamma_t = 10 \gamma_r \). The ratio of the amount of bits transported by coding when compared to that by routing is inversely proportional to the per bit energies spent, i.e., the factor comes out to be \( \frac{2(10 + \gamma_r + \gamma_t)}{10 \gamma_r + 2 \gamma_t} = 1.83 \).

The problem of maximizing the weighted bits transferred as a function of the sources’ weights is analyzed for the basic unit. Figure 6 shows that for weights around 0.5, coding renders maximum benefit. The weights are close to 0 and 1, the performance without coding approaches that of coding. At \( w_1 = 1 \), the bits transported are equal for both the cases. The solution to the optimization problem illustrates that when a single source is prioritized, no coding can be performed and hence the bits transported with and without coding turn out to be the same.

VI. CONCLUSION

In this paper, we studied the problem of designing a regular sensor network intended to carry data from sources \( S_1, S_2, \ldots, S_K \) to corresponding destinations \( D_1, D_2, \ldots, D_K \), where the destinations are arranged in an arbitrary order. The first question was about flow control, routing and scheduling of links, so as to maximize the aggregate bits transported before network partition. We showed that this problem can be converted to a Linear Program. Next, we considered sensor nodes that were coding-capable; specifically, each was capable of XOR-ing received packets. We were interested in understanding the extent to which XOR-ing can improve the aggregate bit transfer capability of the network.
To this end, we proposed an algorithm to select coding points for supplying data to arbitrarily arranged destinations. Then, to assess the benefit that XOR-ing can yield, we formulated an optimization problem that maximizes the aggregate bits transferred before network partition. The conclusion is that for a regular structured network like NCoNET, simple XOR-ing at appropriately selected coding points can yield substantial gains in aggregate bits transported.

Our approach has raised several issues that we would like to study further. While our approach does use the minimum number of coding points, it does not say anything about the minimum number of sensors required to achieve a given permutation. We would like to understand how to locate the coding points optimally, so that a given permutation can be achieved using as few rows as possible. A solution to this problem would be important in understanding how to design the most economical network (using as few sensors as possible) that is able to support any of the $K!$ destination permutations possible.

### REFERENCES


<table>
<thead>
<tr>
<th>S-D Pair (Permutation)</th>
<th>Weights</th>
<th>With coding</th>
<th>Without Coding</th>
<th>% Improvement</th>
<th>Coding Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 (2-1)</td>
<td>(0.1)</td>
<td>454545.45</td>
<td>454545.45</td>
<td>0.00</td>
<td>1.00</td>
</tr>
<tr>
<td></td>
<td>(0.25,0.75)</td>
<td>416666.67</td>
<td>340909.09</td>
<td>22.22</td>
<td>1.22</td>
</tr>
<tr>
<td></td>
<td>(0.5,0.5)</td>
<td>416666.67</td>
<td>227272.73</td>
<td>83.33</td>
<td>1.83</td>
</tr>
<tr>
<td>3 (2-3-1)</td>
<td>(0.1,0)</td>
<td>416670.00</td>
<td>303181.82</td>
<td>30.95</td>
<td>1.31</td>
</tr>
<tr>
<td></td>
<td>(0.33,0.33,0.33)</td>
<td>431818.18</td>
<td>431818.18</td>
<td>0.00</td>
<td>1.00</td>
</tr>
<tr>
<td>4 (2-3-4-1)</td>
<td>(0.2,0.2,0.3,0.3)</td>
<td>416670.00</td>
<td>363640.00</td>
<td>14.58</td>
<td>1.15</td>
</tr>
<tr>
<td></td>
<td>(0.25,0.25,0.25,0.25)</td>
<td>416670.00</td>
<td>303181.82</td>
<td>30.95</td>
<td>1.31</td>
</tr>
<tr>
<td></td>
<td>(0.0,1,0)</td>
<td>416670.00</td>
<td>318180.00</td>
<td>30.95</td>
<td>1.31</td>
</tr>
<tr>
<td>5 (2-5-1-3-4)</td>
<td>(1,0,0,0,0)</td>
<td>416670.00</td>
<td>340910.00</td>
<td>22.22</td>
<td>1.22</td>
</tr>
<tr>
<td></td>
<td>(0.1,0,2,0,3,0,25,0)</td>
<td>416670.00</td>
<td>340910.00</td>
<td>22.22</td>
<td>1.22</td>
</tr>
<tr>
<td></td>
<td>(0.5,0,5,0,5,0)</td>
<td>416670.00</td>
<td>340910.00</td>
<td>22.22</td>
<td>1.22</td>
</tr>
<tr>
<td></td>
<td>(0.2,0,0,3,0,0,0)</td>
<td>416670.00</td>
<td>227270.00</td>
<td>83.38</td>
<td>1.83</td>
</tr>
</tbody>
</table>