Information Flow Decomposition for Network Coding

Christina Fragouli, Member, IEEE, and Emina Soljanin, Senior Member, IEEE

Abstract

The famous min-cut, max-flow theorem states that a source node can send a commodity through a network to a sink node at the rate determined by the flow of the min-cut separating the source and the sink. Recently it has been shown that by linear re-encoding at nodes in communications networks, the min-cut rate can be also achieved in multicasting to several sinks. Constructing such coding schemes efficiently is the subject of current research. The main idea in this paper is the identification of structural properties of multicast configurations, by decomposing the information flows into a minimal number of subtrees. This decomposition allows us to show that very different networks are equivalent from the coding point of view, and offers a method to identify such equivalence classes. It also allows us to divide the network coding problem into two almost independent problems: one of graph theory and the other of classical channel coding theory. This approach to network coding enables us to derive tight bounds on the network code alphabet size and calculate the throughput improvement network coding can offer for different configurations. But perhaps the most significant strength of our approach concerns future network coding practice. Namely, we propose algorithms to specify the coding operations at network nodes without the knowledge of the overall network topology. Such decentralized designs facilitate the construction of codes which can easily accommodate future changes in the network, e.g., addition of receivers and loss of links.

Index Terms

Network coding, network multicast, arcs, information flow, decentralized codes, max-flow min-cut theorem.

This work was supported by DIMACS, NSF under award No. CCR-0325673, and FNS under award No. 200021-103836/1. C. Fragouli is with the EPFL, E. Soljanin is with Bell Labs.
I. INTRODUCTION

Communication networks are, like their transportation or fluid counterparts, mathematically represented as directed graphs $G = (V, E)$. We are concerned with multicast communications networks in which $h$ unit rate information sources $S_1, \ldots, S_h$ simultaneously transmit information to $N$ receivers $R_1, \ldots, R_N$ located at $N$ distinct nodes. We assume that all edges have unit capacity, and that there are $h$ edge-disjoint paths connecting each receiver with the $h$ sources. Consequently, the unit-rate sources can send the information to each receiver when that receiver is the only one using the network.

Traditionally, information flows in communication networks were treated like fluid flows in networks of pipes, in which a unit-capacity edge cannot be used simultaneously by more than one unit-rate source. Information flows are sequences of bits, or if we look at $q$ bits at a time, sequences of elements of some finite field $F$. Thus, in communication networks, a unit-capacity edge can be used simultaneously by more than one unit-rate source to carry, for example, a linear combination over $F_q$ of the symbols the sources emit. Which symbol an edge carries is decided by its parent node, which can not only re-route but also re-encode (e.g., linearly combine) the information it receives. These features of communication networks make multicasting at the min-cut rate possible in the network scenario described above, as shown in the seminal work of Ahlswede, Cai, Li, and Yeung [1], and of Li, Yueng, and Cai [2].

Network codes for multicast are schemes which specify what each node in the network has to perform in order to make the multicast possible. Constructing such coding schemes efficiently for various network scenarios is the subject of current research. An algebraic framework for network coding was developed by Koetter and Medard in [3], who translated the network code design to an algebraic problem which depends on the structure of the underlying graph. Li, Yueng, and Cai showed constructively in [2] that multicast at rate $h$ can be achieved by linear coding. A fast implementation of their approach was developed by Sanders, Egner, and Tolhuizen in [4] and independently by Jaggi, Chou and Jain in [5] (see also [6]), resulting in fast polynomial-time algorithms for constructing linear codes for multicast.

The basic idea of our approach to network coding is partitioning the network graph into subgraphs through which the same information flows. For the network code design problem, the structure of the network inside these subgraphs does not play any role; we only need to know...
how the subgraphs are connected and which receivers nodes are in each subgraph. Thus, we can contract each subgraph to a node and retain only the edges that connect them. We call this process and the resulting object the information flow decomposition of the network. To illustrate this idea, let us look at the familiar example of network with two sources and two receivers shown in Fig. 1a. Note that, because of the topology of the graph, there are three different information flows in this network: one that carries unaltered symbols of the first source, one that carries unaltered symbols of the second source, and one that carries a linear combination (which will be specified by the code) of the symbols from the first and the second source. The three flows will be referred to as the source flows and the coding flow. They are connected as shown in Fig. 1b. The figure also shows which receivers have access to which flows. Network coding should ensure that the two flows a receiver has access to are independent.

One immediate advantage of the information flow decomposition method is that it significantly reduces the dimensionality of the network code design problem, making all algorithms that depend on the graph size faster. Moreover, although a network with $h$ sources and $N$ receivers satisfying the multicast min-cut condition may otherwise have arbitrary structure, its (in a certain sense minimal) information flow decomposition will have a very tightly described structure, and
very different networks with the same number of sources and receivers may have identical information fbw decomposition. For example, we will show later that all networks with two sources and two receivers have one of the two possible information fbw decompositions shown in Fig. 2. Fig. 2a shows the network scenario in which each receiver have access to both source flows, and thus no network coding is required. Fig. 2b shows the network scenario in which each receiver has access to a different source fbw and a common coding fbw.

Recognizing numerous structural properties that a fbw decomposition must satisfy enabled us to derive the size of the network code alphabet $\mathbb{F}_q$ which is sufficient for all networks with $h = 2$ sources and $N$ receivers and necessary for some of such networks. We also state regularity conditions under which the derived alphabet size is sufficient for all networks with $h$ sources and $N$ receivers. In addition to its theoretical merits, bounding the alphabet size has important practical implications, since network coding and decoding requires multiplications and inversions over finite fields whose complexity quickly increases with the field size. The original work of Ahlswede, Cai, Li, and Yeung shows that network multicast is asymptotically possible if coding is performed over infinitely large fields [1]. The subsequent work of Li, Yueng, and Cai [2] shows that the network multicast is possible if linear coding is performed over a sufficiently large finite field. Koetter and Medard upper bounded the required alphabet size by $hN$ in [3]. Sanders, Egner, and Tolhuizen reduced the upper bound of the required alphabet size to $N$ in [4].
and Jain made a claim in [5] that $\sqrt{2N + 1/4} - 1/2$ is a lower bound. Feder, Ron and Tavory [9] using information theory arguments also derived a lower bound of size $\sqrt{2N(1 - o(1))}$, and in addition provided upper bounds for some specific network configurations. The result in this paper is that an alphabet of size $\sqrt{2N - 7/4} + 1/2$ is always sufficient and sometimes necessary, for all configurations with $h = 2$ sources, and under some regularity conditions for configurations with $h > 2$ sources. Note that all of the alphabet bounds depend on the number of receivers. Consequently, the maximum alphabet size a network can support determines the maximum number of users that can be accommodated.

For the $h = 2$ case we derive the code alphabet size bounds by bounding the chromatic number of a class of graphs defined based on the information $fbw$ decompositions of networks with $h = 2$ sources and $N$ receivers. The code design itself amounts to the vertex coloring problem of these class of graphs. This connection allows to directly apply results from coloring [7]. The authors in [14] independently used similar arguments that reduce the problem of network code design to coloring to show that the problem of identifying the minimum alphabet size required for a specific configuration is NP-complete.

The required alphabet size is not always the most important criterion for network code design. Codes which can be defined without the knowledge of the overall network topology and easily extended to accommodate future changes in the network, such as addition of receivers and loss of links are particularly desirable in practice, but have not yet received adequate attention in the literature. The deterministic network code design methods proposed so far result in codes that may need to be completely redesigned to accommodate addition of a single user. Randomized codes recently proposed in [15] seem to alleviate this problem, at the cost of an error probability. We propose a deterministic method to design decentralized codes, which is also based on the information $fbw$ decomposition. One of the main advantages of these decentralized codes is that they do not have to be changed with the addition of new receivers as long as the information decomposition of the network remains the same. Furthermore, addition of new users does not change the proposed network code for the existing users as long as the new subtree graph contains the original subtree graph.

For networks with delay, information $fbw$ decomposition makes connections between network codes and convolutional codes transparent. This leads us to an alternative simpler derivation of the transfer function result in [3]. The convolutional code framework naturally takes delay
into account, but at the cost of increased complexity for both encoding and decoding. We investigate different methods to reduce the complexity requirements taking advantage of the subtree decomposition. Moreover, we discuss implementation using binary encoders, and propose a simplified version of the method in [2] to deal with cycles in the network. Independently, the authors in [9] propose to add node-memory to increase the alphabet size a network can support, which has a similar flavor to our proposed binary encoders implementation.

The paper is organized as follows. We first in Section II, state the problem and present the notation that we follow in the paper, and then in Section III introduce the information flow decomposition. In Section IV we calculate throughput benefits that network coding can offer for some families of configurations. We describe decentralized network coding in Section V and derive alphabet size bounds in Section VI. In Section VII, we investigate connections with vertex coloring, and in Section VIII, connections with convolutional codes. Section IX concludes the paper.

II. THE NETWORK CODING MODEL

We consider a communications network represented by a directed acyclic graph $G = (V, E)$ with unit capacity edges. Our results, however, also hold for a class of multicast configurations over graphs with cycles. There are $h$ unit rate information sources $S_1, \ldots, S_h$ and $N$ receivers $R_1, \ldots, R_N$. The number of edges of the min-cut between the source and each receiver node is $h$. The $h$ sources multicast information simultaneously to all $N$ receivers at rate $h$. As in the previous work [2], we assume zero delay meaning that all nodes simultaneously receive all their inputs and produce their outputs.

We denote by $(S_i, R_j), 1 \leq i \leq h$, a set of $h$ edge disjoint paths from the sources to the receiver node $j$. Under the min-cut assumption, the existence of such paths is guaranteed by the Menger theorems (see for example [12, p. 203]). The choice of the paths is not unique, and will, as we discuss later, affect the complexity of the network code. Our object of interest is the subgraph $G'$ of $G$ consisting of the $hN$ paths $(S_i, R_j), 1 \leq i \leq h, 1 \leq j \leq N$. A way to specify a network code is to describe which operations each node in $G'$ has to perform on its inputs for each of its outgoing edges.

We assume that source $S_i$ emits $\sigma_i$ which is an element of some finite field $\mathbb{F}_q$. In linear network coding, each node of $G'$ receives an element of $\mathbb{F}_q$ from each input edge, and then
forwards a linear combination of its inputs to each output edge. Consequently, through each edge of $G'$ flows a linear combination of source symbols, e.g., the symbol flowing through some edge $e$ of $G'$ is given by

$$\alpha_1(e)\sigma_1 + \alpha_2(e)\sigma_2 + \cdots + \alpha_h(e)\sigma_h = \begin{bmatrix} \alpha_1(e) & \alpha_2(e) & \cdots & \alpha_h(e) \\ c(e) \end{bmatrix} \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \vdots \\ \sigma_h \end{bmatrix},$$

where vector $c(e) = [\alpha_1(e) \alpha_2(e) \cdots \alpha_h(e)]$ belongs to an $h$ dimensional vector space over $\mathbb{F}_q$. We shall refer to the vector $c(e)$ as the coding vector of edge $e$. Note that the coding vector of an output edge of a node has to lie in the linear span of the coding vectors of the node’s input edges. To describe a network code, we need to specify which linear coefficients a node should use to multiply its inputs for each of its outgoing edges. Equivalently, we may specify the coding vector for each edge of the network.

The coding vectors associated with input edges of a receiver node define the system of linear equations that the receiver needs to solve to determine the source symbols. More specifically, consider receiver $R_j$. Let $\rho^j_i$ be the symbol on the last edge of the path $(S_i, R_j)$, and $\mathbf{A}_j$ the matrix whose $i$-th row is the coding vector of the last edge on the path $(S_i, R_j)$. Then the receiver $j$ has to solve the system of linear equations

$$\begin{bmatrix} \rho^j_1 \\ \rho^j_2 \\ \vdots \\ \rho^j_h \end{bmatrix} = \mathbf{A}_j \cdot \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \vdots \\ \sigma_h \end{bmatrix}$$

(1)

to retrieve the information symbols $\sigma_i$, $1 \leq i \leq h$, transmitted from the $h$ sources. Thus, the network code design problem is to select a coding vector for each edge of the network so that the matrix $\mathbf{A}_j$ is full rank for each receiver $j$, subject to the constraint that the coding vector of an output edge of a node lies in the linear span of the coding vectors of the node’s input edges. We refer to an assignment of coding vectors that achieves this goal as a valid network code.
III. DECOMPOSITION INTO SUBTREES

A. Definitions

Throughout this discussion, we will use the example network with two sources multicasting to the same set of three receivers shown in Fig. 3.

![Network Diagram](image)

Fig. 3. Network with two sources \{S_1, S_2\} and three receivers \{F, E, K\}.

Since for a network code we eventually have to describe which operations each node in \(G'\) has to perform on its inputs for each of its outgoing edges, we find it more transparent to work with the graph

\[
\gamma = \bigcup_{1 \leq i \leq h, 1 \leq j \leq N} L(S_i, R_j),
\]

where \(L(S_i, R_j)\) denotes the line graph of the path \((S_i, R_j)\). That is, \(L(S_i, R_j)\) is the graph with vertex set \(E(S_i, R_j)\) in which two vertices are joined if and only if they are adjacent as edges in \((S_i, R_j)\). In other words, \(\gamma\) is the line graph of \(G'\). The graph \(\gamma\) for our example network of Fig. 3 is depicted in Fig. 4a.

Without loss of generality, by possibly introducing auxiliary nodes, we can assume that the line graph contains a node corresponding to each of the \(h\) sources. We refer to these nodes as
source nodes. Each node with a single input edge merely forwards its input symbol to its output edges. Each node with two or more input edges performs a coding operation (linear combining) on its input symbols, and forwards the result to all of its output edges. We refer to these nodes as coding points:

Definition 1: Coding points are the nodes of $\gamma$ with two or more inputs.

We refer to the node corresponding to the last edge of the path $(S_i, R_j)$ as the receiver node for receiver $R_j$ and source $S_i$. For a configuration with $h$ sources and $N$ receivers there exist $hN$ receiver nodes. For example, in Fig. 4a, $S_1A$ and $S_2C$ are source nodes, BD and GH are coding points, and AF, HF, HK, DK, DE and CE are receiver nodes.

We partition the line graph into subsets $T_i$ so that the following properties hold:

1) each $T_i$ contains exactly one source node or a coding point, and
2) every other node belongs to the $T_i$ containing its first ancestral coding point or source node.

We shall call the subset $T_i$ a source subtree if it starts with a source node or a coding subtree.
if it starts with a coding point. The following properties of the subtree graph $\Gamma$ follow directly from its definition and the fact that the min-cut condition is satisfied in the original network $G$:

**Theorem 1:** The subtree graph satisfies the following:

1) each $T_i$ is a tree because the only nodes with two or more input edges in the line graph are the coding points,

2) the same linear combination of source symbols flows through all the nodes (edges in the original graph) that belong to the same $T_i$.

3) the number of edges going out of a subtree plus the number of receivers in the subtree is greater than or equal to the number of its inputs.

4) for each receiver $R_j$, the $h$ paths from the $h$ source nodes to the $h$ receiver $R_j$ nodes are both edge and vertex disjoint,

5) for each receiver $R_j$, the $h$ receiver nodes corresponding to the last edges on the paths $(S_i, R_j)$, $1 \leq i \leq h$, belong to distinct subtrees, and thus

6) each subtree contains at most $N$ receiver nodes.

For the network code design problem, we only need to know how the subtrees are connected and which receiver nodes are in each $T_i$, whereas the structure of the network inside a subtree does not play any role. Thus we can contract each subtree to a node and retain only the edges that connect the subtrees, to get the *subtree graph* $\Gamma$. Fig. 4b shows the subtree graph for the the network in Fig. 3; there are four subtrees: $T_1$ and $T_2$ are source subtrees, $T_3$ and $T_4$ are coding subtrees.

Network coding assigns an $h$-dimensional coding vector $c(T_i) = [c_1(T_i) \ldots c_h(T_i)]$ to each subtree $T_i$. The fbw through $T_i$ is given by

$$c_1(T_i)\sigma_1 + \cdots + c_h(T_i)\sigma_h.$$ 

Receiver $j$ takes $h$ coding vectors from $h$ distinct subtrees to form the rows of the matrix $A_j$ and solves the system of linear equations (1).

**Definition 2:** A valid network code is any assignment of coding vectors to subtrees such that the matrix $A_j$ is full rank for each receiver $j$ and the coding vector of a subtree lies in the linear span of the coding vectors of the subtree’s parents.

**Example 1:** A valid code for the network in Fig. 3 can be obtained by assigning the following
coding vectors to the subtrees in Fig. 4b:
\[ c(T_1) = [1 \ 0], \quad c(T_2) = [0 \ 1], \quad c(T_3) = [1 \ 1], \quad c(T_4) = [0 \ 1]. \]

For this code, the field with two elements is sufficient. Nodes B and G in the network (corresponding to coding points BD and GH) perform binary addition of their inputs and forward the result of the operation. The rest of the nodes in the network merely forward the information they receive. The matrices for receivers \( R_1 \), \( R_2 \), and \( R_3 \) are
\[
A_1 = \begin{bmatrix} c(T_1) \\ c(T_4) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad A_2 = \begin{bmatrix} c(T_2) \\ c(T_3) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}, \quad A_3 = \begin{bmatrix} c(T_3) \\ c(T_4) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}.
\]

B. Minimal Subtree Graphs and Their Properties

For a given communications network graph \( G = (V, E) \), the choice of the \( h \) edge disjoint paths \( (S_i, R_j) \), \( 1 \leq i \leq h \), from the source to the receiver node \( j \) is not unique. We are in particular interested in the choice of paths which gives raise to the subtree decomposition which is in a certain sense minimal. If a subtree decomposition is not minimal, it may require a less efficient network code (in terms of network resources) than the minimal one.

While discussing the properties of a subtree graph \( \Gamma \) we observed that the min-cut condition in the network \( G \) implies that in \( \Gamma \), for each receiver \( R_j \), the \( h \) paths from the \( h \) source nodes to the \( h \) receiver \( R_j \) nodes are vertex disjoint. We will call this property the multicast property of the subtree graph.

**Definition 3:** A subtree graph is called **minimal** with the multicast property if removing any edge would violate the multicast property.

To illustrate the above issues, we consider the following example of a network with two sources and two receivers shown in Fig. 5a. Notice that node \( x \) and its incident edges can be removed without affecting the multicast conditions in the network. The resulting graph is then identical to the one shown in Fig. 2a in Sec. I, which has the subtree decomposition shown in Fig. 2b. Consider now the choice of two sets of edge-disjoint paths (corresponding to the two receivers) shown in Fig. 5b and Fig. 5c. The resulting subtree graph shown in Fig. 5d has more edges and nodes than the one shown in Fig. 2b. We see, however, that the edge between \( T_1 \) and
Fig. 5. A network with two sources and two receivers: (a) the original graph, (b) two edge-disjoint paths from the sources to the receiver $R_1$, (c) two edge-disjoint paths from the sources to the receiver $R_2$, and (d) the resulting non-minimal subtree graph.

Removing this edge incorporates $T_3$ into $T_1$ and $T_4$ into $T_3$, giving the subtree decomposition shown in Fig. 2b.

For the rest of the paper, we will deal only with minimal subtree graphs. This will in particular allow us to derive the largest field size that a code for a network with $N$ receivers may require. However, the principles of coding do not depend on the decomposition.

Note that the multicast property is satisfied in $\Gamma$ if and only if the min cut condition is satisfied for every receiver in $G$. Since the min cut condition is necessary and sufficient for multicast, the following holds:

**Lemma 1:** There is no valid codeword assignment (in the sense of Definition 2) for a graph of subtrees which does not satisfy the multicast property.

We use the lemma to show some properties a minimal subtree graph has.

**Theorem 2:** For a minimal subtree graph, the following holds:

1) There does not exist a valid network code where a subtree is assigned the same coding vector as one of its parents.
2) There does not exist a valid network code where the vectors assigned to the parents of any given subtree are not linearly independent.

3) There does not exist a valid network code where the code vector assigned to a child belongs to a proper subspace of the space spanned by vectors assigned to its parents.

4) Each coding subtree has at most $h$ parents.

5) If a coding subtree has $2 \leq P \leq h$ parents, then the min-cut from the source nodes to the subtree is $P$.

 Proof:

1) Suppose a subtree is assigned the same coding vector as one of its parents. Then removing the edge between the subtree and the other parent results in a subtree graph with a valid coding vector assignment which does not satisfy the multicast condition. This scenario contradicts Lemma 1.

2) Suppose there is a subtree whose parents are assigned linearly dependent vectors. Then removing the edge between the subtree and any parent results in a subtree graph with a valid coding vector assignment which does not satisfy the multicast condition. This scenario contradicts Lemma 1.

3) This is shown in the same manner as the first two claims, and is actually a generalization of claim 1).

4) This is a direct consequence of claim 2).

5) If this claim was not true, claim 2) would not be satisfied.

The first three claims of the theorem describe properties of valid codes for minimal subtree graphs, while the last two claims describe structural properties of minimal subtree graphs. The additional structural properties listed below follow from Theorem 2 and the general properties of subtree graphs described in Sec. III-A.

**Theorem 3:** In a minimal subtree decomposition of a network with $h = 2$ sources and $N$ receivers,

1) a parent and a child subtree have either a child or a receiver in common,

2) each coding subtree contains at least two receiver nodes,

3) each source subtree contains at least one receiver node.
Proof:

1) In the $h = 2$ case, if we assign to a parent and a child subtree the same coding vector, this will not affect the multicast property of any receiver unless the parent and a child subtrees have a child or a receiver in common. This scenario contradicts Theorem 2.

2) If a coding subtree is a terminal node of the subtree graph, it has two receivers and no children. Consider a coding subtree $T$ that is an inner node of the subtree graph. Let $P_1$ and $P_2$ be its parents. By claim 1), a parent and a child have either a receiver or a child in common. If $T$ has a receiver in common with each of its parents, then $T$ has two receivers, since each receiver is shared by exactly two subtrees. If $T$ and say $P_1$ do not have receiver in common, then they have a child in common, say $C_1$. Now, if $T$ and $C_1$ do not have receiver in common, then they have a child in common. And so forth, following the posterity brunch of $P_1$, one eventually reaches a child of $T$ that is a terminal node of the subtree graph, and thus have no children. Consequently, $T$ has to have a receiver in common with this subtree. Similarly, if $T$ and $P_2$ do not have child in common, there will be a descendent of $P_2$ and child of $T$ which must have a receiver in common with $T$.

3) If network coding is not required, each source subtree contains $N$ receivers. If network coding is required, each source subtree will have at least one child. The proof that a source subtree has at least one receiver is based on the same reasoning as above for coding subtrees, except that there is only one brunch of posterity.

Theorem 4: A minimal subtree decomposition of a network with two sources and $N$ receivers has at most $N - 1$ coding subtrees, and there exist minimal subtree configurations with $N - 1$ subtrees and $N$ receivers.

Proof: Recall that there are exactly $2N$ receiver nodes. The first part of claim then follows directly from Theorem 3.

Following we give a contructive proof that minimal subtree configurations with $N - 1$ coding subtrees and $N$ receivers exist. For $N = 2$ the minimal subtree graph in Figure 1b has $N = 2$ receiver and $N - 1 = 1$ coding subtrees. We are going to construct a configuration $\Gamma_k$ that has $N = k$ receiver and $N - 1 = k$ coding subtrees from a configuration $\Gamma_{k-1}$ that has $N = k - 1$ receiver and $N - 1 = k - 1$ coding subtrees as follows.

- Add a new subtree $T$ to $\Gamma_{k-1}$ with parents subtrees $P_1$ and $P_2$ in $\Gamma_k$, such that $P_1$ and
$P_2$ have a receiver (say $R$) in common and no other child in common. This implies that $P_1$ and $P_2$ have a parent-child relationship themselves in $\Gamma_{k-1}$. Note that such $P_1$ and $P_2$ always exist.

- Add a new receiver $R'$ that observes $P_1$ and $T$, i.e. has a receiver node in $P_1$ and a receiver node $T$. Take the receiver node of $R$ in $P_1$ and move it in $T$.

In other words, receiver $R$ observes $T$ and $P_1$ and receiver $R'$ observes $T$ and $P_2$. Thus $T$ cannot be assigned the same coding vector as any of its two parents. Moreover, $P_1$ and $P_2$ have now a child in common, and thus cannot be assigned the same coding vector.

For example, for $N = 3$ the minimal subtree graph in Fig. 4b that has $N = 3$ receiver and $N - 1 = 2$ coding subtrees can be constructed from the $N = 2$ the minimal subtree graph in Figure 1b.

**Corollary 1:** For a network with two sources and two receivers, there exist exactly two minimal subtree graphs shown in Figure 2 in Sec. I.

**Proof:** The scenario shown in Fig. 2a is the case when no network coding is required i.e., there are no coding subtrees. If network coding is required, then by Theorems 3 and 4, there is exactly one coding subtree containing two receiver nodes, and at least one source node contains a receiver. Because of the last claim of Theorem 2, the configuration shown in Fig. 2b is the only possible.

Continuing along these lines, it is easy to show that there exist exactly three minimal configurations with two sources and three receivers, seven minimal configurations with two sources and four receivers, etc.

**IV. Throughput Benefits of Network Coding**

In this section we investigate throughput benefits that network coding can offer as compared to routing, for some classes of configurations.

To measure throughput, we examine the rate sum of the receivers. More specifically, let $T_{nc}^i$ denote the rate that receiver $i$ experiences when network coding is used, and $T_u^i$ the rate when only uncoded transmission is allowed. We are interested in comparing the total aggregate rate when network coding is used $T_{nc} = \sum_{i=1}^{N} T_{nc}^i$ with the total aggregate rate when only uncoded transmission is allowed $T_u = \sum_{i=1}^{N} T_u^i$. 
For a subtree configuration with \( h \) parents and \( N \) receivers, where the min-cut condition is satisfied for every receiver, it holds that

\[ T_{nc} = Nh \quad \text{and} \quad N \leq T_u \leq hN. \tag{2} \]

**Theorem 5:** For all networks with \( h = 2 \) sources and \( N \) receivers, if the min-cut condition is satisfied for every receiver, it holds that

\[ \frac{T_u}{T_{nc}} \geq \frac{1}{2} + \frac{1}{2N} \tag{3} \]

**Proof:** Since the mincut condition is satisfied for every receiver, \( T_{nc} = 2N \). Moreover, the mincut to each node of the graph is at least one, thus there exists a tree that spans the source and the receivers. Finally, since each source subtree contains at least one receiver node (Theorem 3), at least one of the receivers will be able to receive both sources. Thus a lower bound on the achievable \( T_u \) throughput is \( N + 1 \).

Note that for every \( N \), there do exist minimal configurations where without network coding we cannot achieve throughput better than \( N + 1 \), i.e., the bound is tight. Such configurations are the minimal subtree graphs with \( N - 1 \) receivers and \( N - 1 \) coding subtrees, which are described in Theorem 4.

Intuitively one would expect that network coding would offer most benefit, when the receiver nodes are “equally distributed”, that is, in each subtree, there exist an equal fraction of receiver nodes that wants to receive each of the \( h \) sources. The following theorem shows that this intuition may be inaccurate.

**Theorem 6:** Consider a bipartite subtree configuration with \( h \) sources and \( N \) receivers. Assume that each coding subtree has \( h \) parents, and that there exists a receiver that observes every subset of \( h \) subtrees. Then

\[ \frac{T_u}{T_{nc}} \geq a, \quad \text{with} \quad 0 < 1 - \frac{1}{e} \leq a. \tag{4} \]

**Proof:** Note that the min-cut condition is satisfied for every receiver, thus \( T_{nc} = Nh \). Moreover the configuration is minimal, since removing any edge would violate the multicast condition for at least one receiver. Assume that the total number of subtrees is \( kh \), that is, there exist \( h \) source subtrees and \((k-1)h\) coding subtrees\(^1\). Since there exist one receiver that observes

---

\( ^1 \)If the total number of subtrees, say \( M \), is not a multiple of \( h \), we can use \( k = \lfloor \frac{M}{h} \rfloor \) for some sources and \( k = \lfloor \frac{M}{h} \rfloor + 1 \) for some others which does not affect the result.
each $h$ subtrees, the aggregate throughput when using network coding is equal to

$$T_{nc} = Nh = h \binom{kh}{h}.$$  \hfill (5)

We are going to calculate the throughput achieved when transmitting each of the $h$ sources to exactly $k$ subtrees, which will give us a lower bound on the uncoded throughput $T_u$. Under this transmission scheme, let $M_i$ denote the number of receivers that do not receive source $S_i$.

The total loss of throughput, as compared to $T_{nc}$, will be equal to $\sum_{i=1}^{h} M_i$. Since source $S_i$ is transmitted to $k$ subtrees, there exist $M_i = \binom{kh-k}{h}$ receivers that do not receive source $S_i$. Using symmetry, the total loss in throughput is $h \binom{kh-k}{h}$ and

$$T_u = h \binom{kh}{h} - h \binom{kh-k}{h}.$$  \hfill (6)

The fraction of the throughput loss can be calculated as

$$a = \frac{T_u}{T_{nc}} = \frac{h \binom{kh}{h} - h \binom{kh-k}{h}}{h \binom{kh}{h}} = 1 - \frac{\binom{kh-k}{h}}{\binom{kh}{h}}.$$  \hfill (7)

So it is sufficient to show that the term $\frac{\binom{kh-k}{h}}{\binom{kh}{h}}$ does not become equal to one. But its easy to see that

$$\frac{\binom{kh-k}{h}}{\binom{kh}{h}} = \prod_{i=0}^{k-1} \left(1 - \frac{h}{kh-i}\right) \leq (1 - \frac{1}{k})^k \leq e^{-1}$$

$$\frac{\binom{kh-k}{h}}{\binom{kh}{h}} = \prod_{i=0}^{h-1} \left(1 - \frac{k}{kh-i}\right) \leq (1 - \frac{1}{h})^h \leq e^{-1},$$

and thus

$$a = \frac{T_u}{T_{nc}} \geq 1 - \frac{1}{e} > 0.$$  \hfill (8)

V. DECENTRALIZED CODES

As discussed in Section III-A, network coding assigns an $h$-dimensional coding vector $e(T_i) = [c_1(T_i) \ldots c_h(T_i)]$ to each subtree $T_i$. The fbw through $T_i$ is given by

$$c_1(T_i)\sigma_1 + \cdots + c_h(T_i)\sigma_h.$$  \hfill (9)

$^2$It is easy to show that this choice maximizes $T_u$, and that the lower bound is in fact tight.
Receiver $j$ takes $h$ coding vectors from $h$ distinct subtrees to form the rows of the matrix $A_j$ and solves the system of linear equations (1). Since the flow through the subtree corresponding to the source $S_i$ is $\sigma_i$, we assign coding vectors

$$e_1 = [1 \ 0 \ 0 \ldots 0], \ e_2 = [0 \ 1 \ 0 \ldots 0], \ldots, \text{ and } e_h = [0 \ 0 \ldots 1]$$


to the $h$ source subtrees. For each coding subtree, the associated coding vector has to lie in the span of the coding vectors associated with its parent subtrees.

All deterministic network code design algorithms reported so far in the literature rely on information about the entire network structure, i.e., global information. Our goal here is to show how network codes can be designed in a decentralized manner using only local information in the following sense:

**Definition 4:** By decentralized network coding we mean assigning coding vectors to subtrees taking into account only the local information available at the subtree, namely, which receiver nodes it contains and which coding vectors have been assigned to its parent subtrees.

To make decentralized coding possible, we need special sets of coding vectors (alphabets) to use as labels for subtrees as well as special rules (algorithms) for assigning the labels to the subtrees. In the remainder of the section, we first describe alphabets and then algorithms for decentralized network coding.

### A. Coding Vectors and Arcs

Coding vectors for networks with $h$ sources live in the $h$ dimensional space over the field $\mathbb{F}_q$. Since in network coding, we only need to ensure that the coding vectors assigned to the subtrees having receivers in common be linearly independent, it is enough to consider only the vectors in the projective space $\mathbb{P}G(h - 1, q)$ defined as follows:

**Definition 5:** Projective $(h - 1)$-space over $\mathbb{F}_q$ is the set of $h$-tuples of elements of $\mathbb{F}_q$, not all zero, under the equivalence relation given by

$$[a_1 \ldots a_h] \sim [\lambda a_1 \ldots \lambda a_h], \ \lambda \neq 0, \ \lambda \in \mathbb{F}_q.$$  

For networks with two sources, we will use the points on the projective space of dimension 1, i.e., the projective line $\mathbb{P}G(1, q)$:

$$[0 \ 1], \ [1 \ 0], \text{ and } [1 \ \alpha^i] \text{ for } 0 \leq i \leq q - 2,$$  

(10)
where $\alpha$ is a primitive element of $\mathbb{F}_q$. Any two different points on the projective line $\mathbb{P}G(1, q)$ form a basis for $\mathbb{F}_q^2$. Thus any code which assigns different points of $\mathbb{P}G(1, q)$ to different coding trees is a valid network multicast code. Note that this type of coding is decentralized since we only need to use information available locally at a subtree, without taking into account how the subtrees are connected or how the receiver nodes that are not inside our subtree are distributed over the graph.

Example 2: Two codes for the network in Fig. 3

1) A Decentralized Code: Since there are two source and two coding trees, we take the first four points from the list (10):

$$c(T_1) = [1 0], c(T_2) = [0 1], c(T_3) = [1 1], c(T_4) = [1 \alpha].$$

For this code, we need the field with three elements.

2) The Smallest Alphabet Code: If we take into account the information on the receivers in trees, we see that $T_2$ and $T_4$ can be assigned the same coding vector since they do not share any receivers:

$$c(T_1) = [1 0], c(T_2) = [0 1], c(T_3) = [1 1], c(T_4) = [0 1].$$

For this code, the field with two elements is sufficient.

It does not hold in general that any $h$ different points on the projective space $\mathbb{P}G(h - 1, q)$ form a basis for $\mathbb{F}_q^h$. Geometric objects known that have that property are known as arcs.

Definition 6: In a projective plane, a $k$-arc is a set of $k$ points no three of which are collinear (hence the name arc). In general, in $\mathbb{P}G(h - 1, q)$, a $k$-arc is a set of $k$ points any $h$ of which form a basis for $\mathbb{F}_q^h$.

In combinatorics, arcs correspond to sets of vectors in general position:

Definition 7: Set $A$ of vectors in $\mathbb{F}_q^h$ are said to be general position if any $h$ vectors in $A$ are linearly independent.

Example 3: The following set of $h + 1$ points are in general position in $\mathbb{F}_q^h$, and form an arc
in $\mathbb{P}G(h-1,q)$:

\[
\begin{pmatrix}
1 & 0 & \ldots & 0 \\
0 & 1 & \ldots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \ldots & 1 \\
1 & 1 & \ldots & 1
\end{pmatrix}
\]

**Example 4:** The following set of $q+1$ points are in general position in $\mathbb{F}_q^h$, and form an arc in $\mathbb{P}G(h-1,q)$:

\[
\begin{pmatrix}
1 & x_1 & x_1^2 & \ldots & x_1^{h-1} \\
1 & x_2 & x_2^2 & \ldots & x_2^{h-1} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & x_q & x_q^2 & \ldots & x_q^{h-1} \\
0 & 0 & 0 & \ldots & 1
\end{pmatrix}
\]

where $x_i \in \mathbb{F}_q$ and $x_i \neq x_j$ for $i \neq j$.

This arc is known as a **normal rational curve**:

**Definition 8:** A normal rational curve is any set of points projectively equivalent to the set

\[
\{(1, t, t^2, \ldots, t^{h-1}) | t \in \mathbb{F}\} \cup \{(0, 0, 0, \ldots, 0, 1)\}.
\]

Arcs are of special interest for us because they enable decentralized network coding. Namely, as long as we take a point from an arc to be a coding vector of a subtree, we do not need to know other subtrees’ coding vectors (as long as they are different points of the same arc) or the structure of the network. We do, however, have to know which sources are available at the subtree being coded to choose a coding vector with zeros at coordinates corresponding to the sources which are not available.

Given a subtree decomposition $\Gamma$ of a network with $h$ sources, we are interested in finding an arc of the appropriate structure (i.e., containing points with zeros at prescribed places) in $\mathbb{P}G(h-1,q)$ with the smallest possible field size $q$. Thus given $h$ and $q$, we are interested in the length $g(h,q)$ and the structure of the maximal arcs in $\mathbb{P}G(h-1,q)$.

The question on the size of maximal arcs has been known to coding theorists in the context of maximum distance separable (MDS) codes: Consider an MDS code over $\mathbb{F}_q$ of dimension $h$ and generator matrix $G$. The columns of $G$ are vectors in general position in $\mathbb{F}_q^h$. The maximum length $g(h,q)$ such code can have is not known in general. Although the problem looks combinatorial
in nature, most of the harder results on the size of maximal arcs have been obtained by using algebraic geometry (see [11] and references therein), which is also a natural tool to use for understanding the structure (i.e., geometry) of arcs.

A good survey on the size of arcs in projective spaces can be found in [11]. Some specific results presented there include the following:

- $g(h, q) = q + 1$ if $h = 2$ or if $h = 3$ and $q$ is odd,
- $g(h, q) = q + 1$ if $q = h + 1$ and $q$ is odd,
- $g(h, q) = q + 2$ if $h = 3$ and $q$ is even,
- $g(4, q) = g(5, q) = q + 1$,
- $g(6, q) = g(7, q) = q + 1$ if $q$ is even,
- $g(6, q) = q + 1$ if $q = 11$ or $13$.

In general, for $h \geq q$, we know that $g(h, q) = h + 1$, whereas, for $h \leq q$, it holds that $g(h, q) \geq h + 1$, and it is widely believed that

$$g(h, q) = \begin{cases} q + 2 & \text{if } q \text{ is even and either } h = 3 \text{ or } h = q - 1, \\ q + 1 & \text{otherwise.} \end{cases}$$

To handle possible constraints on coding vectors, we can either develop algorithms to generate appropriate arcs, or use theorems about geometry of arcs to derive $\mathcal{A}$’s, as discussed below.

**B. Codes for Networks with $h = 2$ Sources and $N$ Receivers**

As discussed above, to label the nodes of a subtree graph of a network with two sources, we can use the points on the projective line $\mathbb{P}G(1, q)$:

$$[0 \ 1], [1 \ 0], \text{ and } [1 \ \alpha^i] \text{ for } 0 \leq i \leq q - 2. \quad (11)$$

Recall that for a valid network code, it is sufficient and necessary that the coding vector associated with a subtree lie in the linear span of the coding vectors associated with its parent subtrees, and the coding vectors of any two subtrees having a receiver in common be linearly independent. Since any two different points on the line are linearly independent and each point on the line is in the span of any two different points on the line, both coding conditions are satisfied if each node in a subtree graph of a network with two sources and $N$ receivers is assigned a unique point of the projective line $\mathbb{P}G(1, q)$ . This is the basic idea in the following algorithm.
Algorithm 1:

1) **Find Subtree Graph:** Identify in \( G \) a set of \( h \) edge disjoint paths \( (S_i, R_j), 1 \leq i \leq h \), from the source to the receiver \( j, j = 1 \ldots N \). Let \( G' \) denote the subgraph of \( G \) consisting of the \( hN \) paths \( (S_i, R_j), 1 \leq i \leq h, 1 \leq j \leq N \). Denote by \( C \) the set of edges in \( G' \) that are coding points (see def. 1), that is, edges where paths meet for the first time.

2) If \( |C| \leq N - 1 \) go to step 3, else go to step 4.

3) **Find Minimal Subtree Graph:** Associate a weight \( w(e) \) with each edge \( e \) in \( G' \), \( w(e) = c > 0 \) if \( e \) is an incoming edge to any of the edges in \( C \) and zero otherwise. Let \( G'_w \) denote the resulting weighted graph. Identify in \( G'_w \) minimum-weight max-flow paths for each receiver.

4) **Label Subtrees:** Create a number of tokens, each token associated with a different point in \( \mathbb{P}G(1, q) \), and forward the tokens from the sources towards the destinations. Each coding point (corresponding terminal in the network) gets hold a token and uses the respective coding vector as long as required, then releases it back in the network for possible reuse.

To summarize, this algorithm assigns to each different subtree a different vector in the set \( \mathbb{P}G(1, q) \). Each receiver is going to observe two distinct such vectors, and have a full rank system of equations to solve to retrieve the source information.

In the first step of the algorithm we identify the paths to use and the edges where we need to perform linear combinations. If the number of such edges happens to be smaller than \( N \), we can directly proceed to labeling. If not, or if we wish to optimize for employed resources, we proceed to identify the minimal subtree configuration. From definition 3, no edge in a minimal subtree graph can be removed without violating the multicast condition for at least one receiver. Thus the minimal subtree graph has only the required number of such edges. Associating a weight with the corresponding edges and performing a weighted max-fbw algorithm over \( G' \) allows to use the minimum number of such edges.

In the last step of the algorithm, we propose a method to ensure that a distinct coding vector is mapped to each subtree. An alternative simple way to organize this mapping is described below.

Recall that for each subtree, we locally know which receivers it contains and which sources are associated with each receiver (at the terminal before the coding point). In networks with two sources, each subtree contains at least one receiver node associated with \( S_1 \) and at least one
receiver node associated with $S_2$. Let $\{R_{i_1}, R_{i_2}, \ldots, R_{i_u}\}$, where $i_1 < i_2 < \cdots < i_u$ be the set of receivers associated with $S_1$ in a given subtree. We choose $[1 \ a^{i_1-1}]$ to be the label of that subtree. This way no other subtree can be assigned the same label since the receiver $R_{i_1}$ can be associated with the source $S_1$ in at most one subtree. Note that this is not the most efficient mapping as it may require alphabet size of $q - 2 = N - 1$, as opposed to $q - 1 = N - 1$.

C. Codes for Networks with $h$ sources and 2 Receivers and Binary Multicast Codes

From the alphabet bounds derived in [3] on codes using global information, we know that there are valid binary codes for networks with $h$ sources and 2 receivers. We here show that there is only one valid binary code assignment for the minimal subtree graph of a network with 2 receivers, and that this assignment does not need global information.

Since $N = 2$ each coding subtree has at most two receiver nodes, and thus, because of the multicast (min-cut) condition, each coding subtree has exactly two inputs.

Theorem 7: The binary code that assigns to each source subtree, a different basis vector, and to a coding subtree, the binary sum of the vectors assigned to its two parents is the only valid binary code for the minimal subtree graph of a network with 2 receivers.

Proof: From Claim 1) of Th. 2, we know that there does not exist a valid network code where a subtree is assigned the same coding vector as one of its parents. Therefore, since the code is binary and there are exactly two parents, a code subtree must be assigned the binary sum of the vectors assigned to its two parents. Therefore, this is the only code that satisfies a necessary condition for validity. Since there exist binary codes for networks with 2 receivers, the code must be valid.

D. Codes for Networks with Three Sources and $N$ Receivers

We consider the special case of networks with three sources and $N$ receivers where no coding subtree has a child. Thus the subtree graph is bipartite with one set of nodes consisting of the three source subtrees and another set of nodes consisting of coding subtrees. An example is shown in Figure 6.

To the source subtrees, we assign the basis vectors $e_1 = [1 \ 0 \ 0]$, $e_2 = [0 \ 1 \ 0]$ and $e_3 = [0 \ 0 \ 1]$. Depending on which source subtrees a coding subtree has as its parents, we assign to it a vector
belonging to one of the following sets:

$$\mathbb{P}(\langle e_1, e_2 \rangle), \mathbb{P}(\langle e_1, e_3 \rangle), \mathbb{P}(\langle e_2, e_3 \rangle), \text{ or } \mathbb{P}(\langle e_1, e_2, e_3 \rangle)$$

where $\langle x, y \rangle$ denotes the span of vectors $x$ and $y$, and $\mathbb{P}(X)$ denotes the projective space of the vector space $X$. Since each coding subtree contains at least two receivers, the maximum number of subtrees is $\left\lceil \frac{3N}{2} \right\rceil - 1$. Given a set of constraints that coding vectors have to satisfy, we can look for an appropriate arc as illustrated by the following example.

**Example 5:** Suppose we need 6 3-dimensional vectors in general position such that: 2 are in $\Pi_{12}$, 2 in $\Pi_{23}$, 1 in $\Pi_{13}$, 1 in $\Pi_{123}$. Is there an arc of length 6 in $\mathbb{P}G(2, 4)$? In other words, can we start with a known arc of length 6 in $\mathbb{P}G(2, 4)$ (such as the one on the left-hand side in the equation (12) below), and obtain the arc we are interested in by applying a projective transformation. In this particular case, the answer is positive, and the desired arc is obtained as follows:

$$\begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1 \\
1 & 1 & 1 \\
1 & \alpha & \alpha^2 \\
1 & \alpha^2 & \alpha
\end{bmatrix} \cdot \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & \alpha \\
0 & 1 & 1 \\
1 & 0 & \alpha^2 \\
0 & 1 & 1 \\
1 & 0 & \alpha^2
\end{bmatrix} = \begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & \alpha \\
0 & 1 & 1 \\
1 & 0 & \alpha^2 \\
1 & 1 & 0 \\
1 & 1 & \alpha^2
\end{bmatrix}
$$

(12)
E. Codes for Networks with $h$ Sources and $N$ Receivers

The idea in the following algorithms is that, to simplify the code design problem, apart from requiring that the mincut to each receiver is $h$, we can also require some additional structure. For example, if we require that the mincut to each node of the graph is $h$, then we can find $h$ edge-disjoint trees that span the source and the destination nodes, and thus no network coding is required. The following algorithms impose less severe requirements.

Algorithm 2: A straightforward algorithm in the case where $h$ is an even number, is to decompose the problem in $h/2$ two-source configurations and then apply Algorithm 1 (Section V-B) to each one separately. For such a decomposition to exist, the min-cut from every pair of sources to each receiver has to be two, and paths corresponding to sources outside the pairs cannot overlap. Thus, this is obviously a suboptimal algorithm that will require increased resources (additional edges). However, even this very simple suboptimal algorithm can offer throughput benefits as compared to not using network coding for certain configurations. Indeed, for the configurations that can be decomposed in $h/2$ configurations that achieve the lower bound in Theorem 5, if network coding is not employed the total aggregate throughput is $\frac{1}{2}(N + 1)$, while employing the described algorithm allows to achieve throughput $hN$.

Algorithm 3: The basic idea in this algorithm is to use as coding vectors vectors in arcs. We propose to artificially create subtree graphs where each subtree has exactly $h$ parents, so that the coding vectors of the parent subtrees form a basis of the $h$-dimensional space. That is, we require that the mincut towards each coding point is $h$, and have at the input of each subtree incoming coding vectors that form a basis of the $h$-dimensional space. There are many possible variations in implementing this idea. For example, we can first identify a minimal subtree configuration, and then join the coding points that have less than $h$ inputs to the corresponding sources.

Once we create our configuration, we need to use an alphabet large enough for the number of coding points in our network.

Corollary 2: In a minimal subtree decomposition of a network with $h$ sources and $M$ coding subtrees the alphabet of size $M + h - 1$ is sufficiently large for decentralized coding.

Proof: We can use the $M + h$ points in the normal rational curve (see definition 8) in $\mathbb{P}G(h - 1, M + h - 1)$ to assign $h$ vectors to the source subtrees and $M$ vectors to the coding subtrees.
F. Scalability

One of the main advantages of decentralized codes is that they do not have to be changed with the growth of the network as long as the subtree decomposition remains the same, or the new subtree graph contains the original subtree graph. To achieve that, for example in the $h = 2$ case, since the coding vectors associated with any two subtrees provide a basis of the 2-dimensional space, we can think of subtrees as “secondary sources” and allow new receivers to connect to any two different subtrees. Thus we can extend our network, without perturbing the already existing users.

Note that, the projective line $\mathbb{P}G(1,q)$ forms a subset of the projective line $\mathbb{P}G(1,q^f)$, where $\mathbb{F}_q^f$ is an extension field of $\mathbb{F}_q$. Thus, if we need to create additional coding vectors to allocate to new subtrees, we can employ unused points from the projective line $\mathbb{P}G(1,q^f)$.

In some cases even the codes which are not decentralized can remain the same, and the subtree decomposition method shows us how to ensure that. These points are illustrated here by considering the network in Fig. 3 and the codes in the Example 2.

Suppose that a new node $J$ with the receiver $R_4$ is introduced in our example network in Fig. 3, as shown in Fig. 7. We are interested to find out if the two codes described in the Example 2 are still valid. If the network changes, its code will not have to be changed if the same subtree decomposition is still possible. If the code is decentralized, this condition is also sufficient. In our example, the condition is fulfilled. Thus the first code in the Example 2 is also valid for the network in Fig. 7. To see if the other code in the Example 2 is also valid, we have to see which two subtrees in Fig. 4 will contain the receiver $R_4$. We see that $R_4$ will have to be in the subtree $T_4$ and in either subtree $T_2$ or $T_3$. If $R_4$ is placed in $T_2$ the code is not valid since $T_2$ and $T_4$ have identical labels and thus the matrix $A_4$ is singular. If $R_4$ is placed in $T_3$, the code remains valid.

VI. Bounds on Code Alphabets

We are here interested in the maximum alphabet size that a code for a network with $h$ sources and $N$ receivers may require. We characterize a class of networks which require this maximum alphabet by describing their minimal subtree decomposition. For all other networks with $h$ sources and $N$ receivers, this alphabet size is sufficient but may not be necessary. Recall that the binary alphabet is sufficient for networks which require only routing.
A. Networks with Two Sources and \( N \) Receivers

We can equivalently prove that \( k = q + 1 \) vectors are sufficient to construct a valid code since, as we have seen, the projective line \( \mathbb{P}G(1,q) \) supplies \( q + 1 \) different coding vectors for networks with two sources.

Let \( \Gamma \) be a minimal subtree graph with the number of vertices (subtrees) \( n > 2 \); \( n - 2 \) is the number of coding subtrees. (Note that when \( n = 2 \), \( \Gamma \) has only source subtrees and no network coding is required.) We relate the problem of assigning vectors to the vertices of \( \Gamma \) to the problem of vertex coloring of a suitably defined graph \( \Omega \). Let \( \Omega \) be a graph with \( n \) vertices, each vertex corresponding to a different subtree in \( \Gamma \). We connect two vertices in \( \Omega \) with an edge when the corresponding subtrees cannot be allocated the same coding vector.

If two subtrees have a common receiver node, they cannot be assigned the same coding vector. Thus, we connect the corresponding vertices in \( \Omega \) with an edge which we call receiver edge. Similarly, if two subtrees have a common child, by Theorem 2, they cannot be assigned the same coding vector. We connect the corresponding vertices in \( \Omega \) with an edge which we call a flow edge. By Theorem 2, a parent and a child subtrees cannot be assigned the same coding vector.
vector. However, we need not worry about this case separately since by the same theorem, a parent and a child subtrees have either a child or the receiver in common. Fig. 8 plots $\Omega$ for our example subtree graph.

![Figure 8](image-url)

**Fig. 8.** A subtree graph $\Gamma$ and its associated graph $\Omega$. The receiver edges in $\Omega$ are labelled by the corresponding receivers.

**Lemma 2:** For a minimal configuration with $n > 2$, every vertex $i$ in $\Omega$ has degree at least two, that is, $d_i = 2 + d$, for some $d \geq 0$.

**Proof:**

1) **Source subtrees:** If $n = 3$, the two source subtrees have exactly one child which shares a receiver with each parent. If $n > 3$, the two source subtrees have at least one child which shares a receiver or a child with each parent.

2) **Coding subtrees:** Each coding subtree has two parents. Since the configuration is minimal, it cannot be allocated the same coding vector as either of its parents. This implies that in $\Omega$ there should exist edges between a subtree and its parents, that may be either fbw edges, or receiver edges, and the corresponding vertex has degree at least two.

**Lemma 3:** ([12], Ch. 8) Every $k$-chromatic graph has at least $k$ vertices of degree at least $k - 1$. 

December 18, 2004
Theorem 8: For any minimal configuration with \( N \) receivers, the code alphabet \( \mathbb{F}_q \) of size

\[
\lfloor \sqrt{2N - 7/4} + 1/2 \rfloor
\]

is sufficient. There exist configurations for which it is necessary.

Proof: Assume that our graph \( \Omega \) has \( n \) nodes and chromatic number \( \chi(\Omega) = k \leq n \). Let \( m = n - k \), where \( m \) is a nonnegative integer. We are going to count the number of edges in \( \Omega \) in two different ways:

1) From Lemmas 2 and 3, we know that each vertex has degree at least 2 and at least \( k \) vertices have degree at least \( k - 1 \). Consequently, we can lower bound the number of edges of \( \Omega \) as

\[
E(\Omega) \geq \left[ k(k-1) + (2+d)m \right]/2, \tag{13}
\]

where \( d \geq 0 \).

2) Since there are \( N \) receivers and \( n - 2 \) coding subtrees, we have at most \( N \) receiver edges and at most \( n - 2 \) fbw edges. Thus

\[
E(\Omega) \leq N + n - 2 = N + k + m - 2. \tag{14}
\]

From Equations (13) and (14), we get obtain

\[
N \geq \frac{k(k-1)}{2} - k + 2 + dm \leq \frac{k(k-1)}{2} - k + 2. \tag{15}
\]

The equation provides a lower bound on the number of receivers we need in order to have chromatic number \( k \). Solving for \( q = k - 1 \) we get the bound

\[
q \leq \lfloor \sqrt{2N - 7/4} + 1/2 \rfloor.
\]

This proves the first claim of the theorem that, for any minimal configuration with \( N \) receivers, the code alphabet \( \mathbb{F}_q \) of size \( \lfloor \sqrt{2N - 7/4} + 1/2 \rfloor \) is sufficient.

To show that there exist configurations for which this size alphabet is necessary, we take \( m = 0 \) which implies \( n = k \). Consequently, \( \Omega \) is a complete graph with \( n = k = q + 1 \) vertices and \( E(\Omega) = \frac{k(k-1)}{2} \) edges. From, equation (15), we see that the minimum number of receivers required for such a configuration is \( N = \frac{k(k-1)}{2} - k + 2 \). Thus, the remaining \( k - 2 \) edges are fbw edges.

It was previously shown that the of alphabet size \( N - 1 \) is sufficient for networks with 2 sources and \( N \) receivers, and that there are networks with \( (q + 1)q/2 \) receivers for which the
alphabet of size \( q \) is necessary [4]–[9]. Both bounds can be derived by bounding the chromatic number of the \( \Omega \) defined above. Since the sum of number of receivers a subtree contains and the number of its children can not exceed \( N \) (see Sec. III-A), the maximum degree of \( \Omega \) is \( N \), and thus its chromatic number is smaller then or equal to \( N \). Therefore, the of alphabet size \( N - 1 \) is sufficient for all networks with two sources. Now, consider the special case when the subtree graph \( \Gamma \) is bipartite with one set of nodes corresponding to the source subtrees and the other set of nodes corresponding to the coding subtrees, and the corresponding \( \Omega \) is a complete graph in which any two nodes share a receiver. If such a network has \( q + 1 \) subtrees and thus at least \( (q + 1)q/2 \) receivers, then the alphabet of size \( q \) is necessary.

We found that there are networks with even fewer receivers for which the alphabet of size \( q \) is necessary, and found that the minimum number of receivers such a network must have is \( (q+1)q/2 - q + 1 \). Thus, for any network with fewer receivers, the alphabet of size \( q \) is sufficient. The bound is plotted in Fig. 9.

Fig. 9. Alphabet size bound as a function of the number of receivers.

**Definition 9:** A *k-critical two-source topology* is the minimal subtree configuration of a network with two sources which requires an alphabet size of \( k = \lfloor \sqrt{2N - 7/4} + 1/2 \rfloor \).

**B. Networks with \( h \) Sources and \( N \) Receivers**

We know that there exist networks with \( h \geq 2 \) sources and \( N \) receivers for which the alphabet size of \( k = \lfloor \sqrt{2N - 7/4} + 1/2 \rfloor \) is necessary. These are all the subtree configurations with \( h \).
sources which contain a $k$-critical two-source topology. The interesting question is whether the same alphabet size is also sufficient.

In Theorems 9 and 10 we derive regularity conditions on the subtree graphs structure under which the alphabet size of Theorem 8 is sufficient. We conjecture that all subtree configurations actually satisfy one of these conditions, and thus the alphabet size in Theorem 8 is always sufficient. Before stating the theorems, we will need the following definitions, that are basically an extension of the definitions for $h$ sources.

We say that a minimal subtree configuration with $h$ sources requires alphabet size $k + 1$ if there exists a valid network code with alphabet size $k + 1$ and there does not exist a valid network code with alphabet size $k$. We say that a minimal configuration critically requires alphabet size $k + 1$ if removing any one receiver leads to a configuration that has a valid labeling with an alphabet of size $k$.

Every subtree configuration that requires an alphabet of size $k + 1$ can be reduced, by removing receivers, to a configuration that critically requires an alphabet of size $k + 1$.

**Definition 10:** A minimal subtree configuration that critically requires alphabet size $k + 1$ is called a $k$-critical $h$-source topology if it has $N$ receivers and there does not exist another minimal subtree configuration that critically requires the same alphabet size and has a smaller number of receivers.

We are following interested in isolating the structure that leads to the alphabet size requirement. From the min-cut condition in a subtree configuration with $h$ sources, there exist $h$ non-overlapping paths from the sources to the receivers nodes of each receiver, thus we can associate each receiver node with a distinct source. If we move a receiver node from a subtree to the source node it is associated with, we will still have a configuration where the min-cut condition to each receiver is satisfied. The resulting configuration may however, have a different number of subtrees.

**Definition 11:** A $k$-structural subtree configuration is a $k$-critical $h$-source topology such that moving a receiver node from a subtree to the corresponding source subtree would result to a configuration that does not require alphabet $k + 1$.

We can always reduce a $k$-critical $h$-source topology to a $k$-structural topology by moving all the receiver nodes that we can to the corresponding source subtrees.

**Lemma 4:** In a $(k + 1)$-structural configuration each non-empty coding subtree takes part in
at least $k$ different bases.

Proof: Take any coding subtree $T$ and move all its receiver nodes to their corresponding source nodes. Label the resulting configuration using an alphabet of size $k$.

Implicitly in the following argument we use the fact that if $h$ vectors form a basis of an $h$-dimensional space, their ($h$ in number, not necessarily distinct) projections on any subspace will generate the subspace.

Assume that the subtree $T$ has $l + 1 \leq h$ parents. An alphabet of size $k$ implies that the coding vector $c(T)$ can be any point in the projective space $PG(l, k)$. An alphabet of size $k$ is not sufficient if none of these points iseligible. A point is not eligible if it already belongs in the span of $l$ vectors that together with $c(T)$ need to form a basis of the $l + 1$-dimensional space. By definition, a subspace of dimension $l$ is an hyperplane. So we can restate our question as, what is the minimum number of hyperplanes that contain every point in the $PG(l, k)$. This is greater or equal to $k$.

Theorem 9: For a configuration with $h \geq 2$ sources and $N$ receivers an alphabet of size $k = \lceil \sqrt{2N - 7/4} + 1/2 \rceil$ is sufficient if the following condition is satisfied:

- The embedded $k$-critical topology can be constructing by a $(k - 1)$-critical topology.

Proof: Let $N^h_k$ denote the number of receivers required in a $k$-critical $h$-source topology.

For the $h = 2$ case we know that

$$N^2_{k+1} - N^2_k = k, \quad N^2_3 = 4. \quad (16)$$

We want to show that $N^h_k \geq N^2_k$.

Denote by $G_k$ a $k$-critical $h$-source topology, and $S_k$ its associated $k$-structural topology. From Lemma 4 to create $S_{k+1}$ (that will be incorporated in $G_{k+1}$) from $S_k$ we need to add at least $k$ receivers. Thus

$$N^h_{k+1} - N^h_k \geq k, \quad N^h_3 = 4, \quad (17)$$

which implies that $N^2_k \geq N^2_k$.

The following theorem gives an alternative characterization.

Theorem 10: A $h \geq 2$ sources and $N$ receivers an alphabet of size $k = \lceil \sqrt{2N - 7/4} + 1/2 \rceil$ is sufficient if the following condition is satisfied:
Starting from a network with $h \geq 2$ sources and $N$ receivers that requires alphabet of size $q$, we can construct a subnetwork with two sources and $N$ receivers that itself requires the alphabet of size $q$.

Following we describe a method to get such projections. Define the “projection” into the plane $\pi_{ij} = \langle e_i, e_j \rangle$ as follows.

- Project all coding vectors on the $\pi_{ij}$ plane. Identify the projection of each coding vector with a point in the projective line $PG(1, k + 1)$.
- For each receiver $R$ we can find three non-overlapping paths, one from each source. Keep only the paths that correspond to the $i$ and $j$ sources. Remove the edges that correspond to the third source.
- If a subtree is left with one parent, incorporate it with the parent.

This “projection” will give us a subtree graph with $h = 2$ sources and $N$ receivers.

We conjecture that if an alphabet of size $k$ is necessary, there exists a plane on which the projection also requires an alphabet of size $k$, and thus the bound is applicable to all configurations.

As an example, if we have a bipartite configuration with $h$ sources, where each coding subtree has exactly two parents, it is easy to see that an alphabet of size $k$ is required if and only if the condition in Theorem 10 is satisfied.

VII. CONNECTIONS WITH COLORING

In Section VI-A we reduced the problem of designing a network code for a multicast configuration with $h = 2$ sources and $N$ receivers over an arbitrary underlying graph, to the problem of coloring an appropriately defined graph. Once the connection with coloring is realized, a number of combinatorial results can be readily applied [7]. We present here some of the most exciting ones, and refer the interested reader to (chapter 7, [16]) and [17] and the references therein.

A. Min-cut alphabet-size trade-off

The bound in Theorem 8 expresses the connection between required alphabet size and maximum possible number of users to accommodate. An underlying assumption of this bound is that the min-cut towards each user is exactly equal to the number of sources. We would expect that,
if the min-cut towards some or all of the users is greater than the number of sources, a smaller alphabet size would be possible.

For the special case where the subtree graph is a bipartite graph, we can readily apply the following result. Consider a set of points $X$ and a family $F$ of subsets of $X$. A coloring of the points $X$ is legal if no subset of $F$ is monochromatic. If a family admits a legal coloring with $q$ colors then it is called $q$-colorable.

**Theorem 11:** (Erdős 1963) Let $F$ be a family of sets each of size at least $m$. If $|F| < q^{m-1}$ then $F$ is $q$-colorable.

In our case, $X$ is the set of coding subtrees, $m$ is the min-cut from the sources to each receiver, and each subset of $F$ corresponds to the subtrees that a receiver observes. We want to find a coloring such that each receiver observes at least 2 different colors, i.e. has a basis of the 2-dimensional space. Theorem 11 tells us that by increasing the min-cut $m$ we can accommodate the same number of users $N = |F|$ with a smaller alphabet size (alphabet size=$q-1$).

An algorithm for identifying a legal $q$-coloring can be found for example in [18].

**B. Almost good codes**

Again we consider the case where the subtree graph is bipartite. Assume that a legal coloring does not exist. The question here is, what is the maximum number of legally colored subsets that we can have.

**Theorem 12:** (chapter 19, [17]) For every $m$-uniform family $F$ there exists a $q$-coloring of its points which colors at most $|F|q^{1-m}$ of the sets of $F$ monochromatically.

A family of sets is $m$-uniform if all its members have size $m$. Thus if we have $|F| = N$ receivers, the min-cut to each receiver is $m$, and we use an alphabet of size $q-1$, at most $Nq^{1-m}$ receivers will not be able to decode.

**C. Structural Information**

Having some information about the structure of the underlying graph can help reduce the number of colors employed and design new algorithms. The authors in [9] have derived alphabet size bounds in this direction. For example, if we know the number of vertices of the graph $\Omega$ introduced in Sec. VI-A, we can use this number to upper bound the number of colors we need in the previous algorithm.
Similarly, we may know what is the maximum number of receiver nodes inside a subtree, that is, what is the maximum number of receivers that observe the same coding vector. For the graph \( \Omega \), this quantity corresponds to \( \Delta(\Omega) \), where \( \Delta(\Omega) \) is defined as the maximum degree of its vertices, and the degree \( d_i \) of vertex \( i \) is the number of edges adjacent to it.

The greedy coloring algorithm ([10], pg.98) sequentially visits the vertices of the graph and colors each vertex with a color not already used to color any of its neighbors. This algorithm uses a maximum of \( \Delta(\Omega) + 1 \) colors. Thus, the maximum alphabet size required would be \( q = \Delta(\Omega) \).

Thus if we have some information about the structure of the underlying graph we can derive bounds that apply to specific configurations. Again there is also a number of results in extremal combinatorics, such as the following theorem.

**Theorem 13:** (Erdős-Lovasz 1975) If every member of a \( m \)-uniform family intersects at most \( q^{m-3} \) other members, then the family is \( q \)-colorable.

Thus if the min-cut to each receiver is \( m \), and every coding subtree is observed by at most \( q^{m-3} + 1 \) receivers, then it is sufficient to use an alphabet of size \( q + 1 \), irrespective of the number of receivers.

**VIII. CONVOLUTIONAL CODES**

In the development of the previous sections we assumed zero delay, meaning that all nodes simultaneously receive all their inputs and produce their outputs. We now relax this assumption and discuss a connection with convolutional codes over finite fields. The convolutional code framework naturally takes delay into account, but at the cost of increased complexity for both encoding and decoding. We investigate different methods to reduce the complexity requirements.

Additionally, we discuss implementation using binary encoders, and propose a simplified version of the method in [2] to deal with cycles in the network.

**A. Connection**

To relax the zero delay assumption we can associate a unit delay \( D \) with each node of the line graph. Associating a unit delay with each edge of the network was also proposed in [3]. Our contribution is the observation that then the line graph can be thought of as a convolutional
code over a finite field, with number of memory elements $m$ equal to the number of edges in $G$.

A general description of a convolutional encoder over a finite field with $h$ inputs, $N$ outputs, and $m$ memory elements is given by the well known state-space equations:

$$s_{j+1} = As_j + Bu_j$$
$$y_j = Cs_j + Du_j$$

where $s_j$ is the $m \times 1$ state vector, $y_j$ is the $Nh \times 1$ output vector, $u_j$ is the $h \times 1$ input vector, and $A$, $B$, $C$, and $D$ are matrices with appropriate dimensions. The corresponding generator matrix $G(D)$ is given by

$$G(D) = D + C(D^{-1}I - A)^{-1}B,$$  \hspace{1cm} (18)

where $D$ is the indeterminate delay operator. The expression in Eq. (18) coincides with the transfer matrix $M$ derived in [3], giving a different and simpler derivation of the same result.

Matrix $A$ reflects the way the memory elements are connected. An element in matrix $A$ can be nonzero, only if a corresponding edge exists at the given network configuration. Network code design amounts to selecting the nonzero-element values for matrix $A$. Matrices $B$, $C$, and $D$ are completely determined by the network configuration.

We observe that the size of matrices $A$, $B$, $C$, and $D$ depends upon the number of memory elements of the convolutional code, which in turns is equal to the number of edges in the original graph $G$. This number can get quite large, resulting in large size of matrices to handle. Using the subtree graph as a convolutional code instead, as discussed in detail [8], allows to significantly decrease the number of memory elements and thus accelerate all algorithms that depend on the involved dimensionality. An example subtree configuration is shown in Fig. 12 and its corresponding convolutional code in Fig. 13. Unless otherwise stated, we will be considering the convolutional code associated with the subtree graph.

**B. Structural Properties**

We next examine the structure of matrices $A$, $B$, $C$, and $D$. Determining the structure of these matrices can be used for example to perform exhaustive searches over all possible configurations to satisfy a given criterion. We distinguish two cases, depending on whether a partial order constraint, which we will following describe, is satisfied.
We observe that each path \((S_i, R_j)\) from source \(S_i\) to receiver \(R_j\) induces a partial order on the set of the line graph nodes: if edge \(a\) is a child of edge \(b\) then we say that \(a < b\). The source node is the maximal element. Each different path imposes a different partial order on the same set of edges. We distinguish the graphs depending on whether the partial order imposed by the different paths is consistent. Consistency implies that for all pairs of edges \(a\) and \(b\), if \(a < b\) in some path, there does not exist a path where \(b < a\). A sufficient, but not necessary, condition is that the underlying graph \(G\) is acyclic. For example, consider Fig. 10, that depicts a subgraph of a network graph. Sources \(S_1\) and \(S_2\) use the cycle \(ABCD\) to transmit information to receivers \(R_1\) and \(R_2\), respectively. In the case \(A\), the paths from sources \(S_1\) and \(S_2\) impose consistent partial orders on the edges of the cycle. In the case \(B\), for the paths from source \(S_1\), we have \(AB > CD\), whereas for the paths from source \(S_2\), we have \(CD > AB\).

![Case A and Case B](image)

Fig. 10. Case A: partial order preserved. Case B: partial order not preserved.

1) Case 1: Consistent Partial Order: Each subtree corresponds to one element in the state vector \(s\). Let \(m\) be the total number of subtrees. It is easy to see that we can arrange the state vector so that matrix \(A\) is lower diagonal, and matrix \(B\) has the form

\[
B = \begin{bmatrix}
I \\
0
\end{bmatrix},
\]

(19)

where \(I\) is the \(h \times h\) identity matrix. The \(Nh \times m\) matrix \(C\) is a zero-one matrix of the form

\[
C = \begin{bmatrix}
C_1 \\
\vdots \\
C_h
\end{bmatrix},
\]

(20)
where the $h \times m$ matrix $C_i$ corresponds to the receiver $i$. Each row of $C_i$ corresponds to one of the subtrees whose state is observed by the receiver $i$. Thus matrix $C_i$ has exactly one 1 in each row and at most one 1 in each column. Matrix $D$ is identically zero since we associate with source subtrees. Matrices $C_i$ are completely determined by the subtree configuration.

The min-cut, max-flow requirement is equivalent to the condition that the $h \times h$ transfer matrix that corresponds to each receiver

$$G_i(D) = C_i(D^{-1}I - A)^{-1}B, \quad i = 1 \ldots R,$$

has full rank, as described in [3].

2) Case 2: Non-Consistent Partial Order: In some networks with cycles, the partial order imposed by the different paths is not consistent. The corresponding subtree graph form recursive convolutional encoders, and can be analyzed by taking into account the feedback as proposed in [3].

Alternatively, we may follow a simplified version of the approach in [2]. Observe that an information source may need to be transmitted through the edges of a cycle at most once, and then can be removed from the circulation by the node that introduced it. For example consider the cycle in Fig. 10-B, and assume that each edge corresponds to one memory element. Then the fbws through the edges of the cycle are

$$AB : \quad \sigma_1(k) + \sigma_2(k - 2)$$
$$BC : \quad \sigma_1(k - 1) + \sigma_2(k - 3)$$
$$CD : \quad \sigma_1(k - 2) + \sigma_2(k)$$
$$DA : \quad \sigma_1(k - 3) + \sigma_2(k - 1).$$

where $\sigma_i(k)$ is the symbol transmitted from source $S_i$ at time $k$. Equations (22) can be easily implemented by employing a block of memory elements as shown in Fig. 11. Thus, we can still have a feedforward encoder, by representing the cycle with a block of memory elements in the subtree graph, and accordingly altering the structure of matrices $A$ and $C$.

C. Decoding Complexity

Taking delay into account implies that each receiver no longer has a linear system of equations to solve, but needs to perform trellis decoding (Eq. 23). However, the task of decoding is very much simplified by the fact that there is no noise.
Receiver $i$ experiences a rate-1 non-catastrophic encoder with generator matrix

$$G_i(D) = DC_i(I - DA)^{-1}B$$  \hspace{1cm} (23)

Assume that at time $j$ the receiver sees output $y_j$. To decode, the receiver only needs to additionally know the previous state $s_{j-1}$ at time $j-1$, and the trellis diagram of the convolutional code. In other words, the traceback depth of decoding is one. Thus the complexity of decoding is proportional to the complexity of the trellis diagram.

Two methods to reduce the complexity that a receiver experiences are the following.

- **Method 1**: Minimize the trellis diagram used for encoding.
- **Method 2**: Identify the minimal strictly equivalent encoder to $G_i(D)$ to decode, to minimize the trellis diagram used for decoding.

**Method 1**

We are interested in identifying, among all encoders that are subject to the constraints of a given topology, and that satisfy the min-cut max-fbw conditions for each receiver, the encoder that has the smallest number of memory elements. The minimization does not need to preserve the same set of outputs, as we are not interested in error-correcting properties, but only the min-cut property for each receiver. Equivalently, we need to identify a minimal subtree configuration.

As the number of states of the convolutional code depends upon the number of subtrees in the minimal configuration, it is interesting to observe that a given subtree graph can be reduced to minimal configurations that have a different number of coding subtrees.
Method 2

The information to be received by receiver $R_i$ will be encoded by the encoder $G_i(D)$. This encoder cannot be chosen arbitrarily, but subject to configuration restrictions.

However, to decode, we may use a different trellis, the one associated with the minimal strictly equivalent encoder to $G_i(D)$. Two codes are called strictly equivalent if they have the same mapping of input sequences to output sequences. Among all strictly equivalent encoders, that produce the same mapping of input to output sequences, the encoder that uses the smallest number of memory elements is called minimal [13].

Although we can not select the minimal $G_i(D)$ to encode because we were restricted by the configuration, at the decoder we may still use the minimal strictly equivalent trellis to reduce decoding complexity.

D. Binary Alphabet

The convolutional codes corresponding to network codes are over a finite field $\mathbb{F}$ that (but for the simplest cases) is not binary. If a network supports only binary transmission, we consider $f = \lceil \log_2 |\mathbb{F}| \rceil$ uses of the network to comprise a symbol of a higher alphabet. This implies that each node that performs network coding has to store and process $f$ binary bits before retransmitting, thus effectively needs to use $f$ binary memories.

An alternative approach would be to restrict the network coding alphabet to be binary, and allow each node to use up to $f$ binary memory elements in an arbitrary fashion. In our subtree configuration, we may replace each subtree with any convolutional code with $f$ binary memory elements. Using an alphabet of size $2^f$ becomes a special case, so it is guaranteed that there exists a topology that employs possibly less and at most an equal number of binary memory elements. An example is provided following.

Numerical Example

Consider the configuration with $h = 2$ source subtrees and $m - h = 2$ leaf subtrees depicted in Fig. 12. The corresponding convolutional encoder is depicted in Fig. 13. Matrix $A$ will have
Fig. 12. Subtree configuration with two sources and five receivers.

\[
\begin{array}{c}
T_1 \\
T_2 \\
T_3 \\
T_4 \\
\end{array}
\Rightarrow
\begin{array}{c}
\text{Receiver 0} \\
\vdots \\
\text{Receiver 4} \\
\end{array}
\]

Fig. 13. Convolutional encoder corresponding to the subtree configuration with two sources and five receivers.

the form

\[
A = \begin{bmatrix}
0 & 0 \\
A_{1,1} & 0
\end{bmatrix} = \begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
* & * & 0 & 0 \\
* & * & 0 & 0
\end{bmatrix},
\]

where \(A_{1,1}\) is of dimension \(2 \times 2\), and "*" denotes a nonzero element. The generator matrix of dimension \(2 \times 2\) corresponding to every receiver will have the form

\[
G_i(D) = DC_i(I + DA)B = DC_i \begin{bmatrix}
I \\
DA_{1,1}
\end{bmatrix}.
\]

The identity matrices \(I\) are of different dimension, as determined by the context. Matrix \(G\) is common for all receivers. A possible choice for matrix \(A\) over a finite field of size greater or
equal to three would be

\[
A = \begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 \\
1 & 2 & 0 & 0
\end{bmatrix}.
\]

Alternatively, we may use a binary network code. Since we consider two uses of the network, the input/output to each subtree \( T_i \) in Fig. 13 would be two bits. Then each subtree can perform at time \( k \) the following binary operation

\[
M_i \cdot [\sigma_1(k) \ \sigma_1(k-1) \ \sigma_2(k) \ \sigma_2(k-1)]^T,
\]

where

\[
M_1 = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0
\end{bmatrix}, \quad M_2 = \begin{bmatrix}
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix},
\]

\[
M_3 = \begin{bmatrix}
1 & 0 & 1 & 0 \\
1 & 1 & 0 & 1
\end{bmatrix}, \quad M_4 = \begin{bmatrix}
1 & 1 & 1 & 0 \\
1 & 0 & 1 & 1
\end{bmatrix}.
\]

Receiver \( k \) will observe a matrix of the form

\[
A_k = \begin{bmatrix} M_i \\ M_j \end{bmatrix}, \quad i \neq j,
\]

which has full rank.

**IX. Conclusions**

In this paper, we introduced the information flow decomposition which offers a method to study the common underlying structural properties of different multicast configurations. We showed that this method can be used to group together different configurations that are equivalent from a coding point of view, to derive alphabet size bounds, to calculate throughput benefits that network coding may offer, and to develop decentralized scalable algorithms. We believe that this is a promising tool that may find many more applications both in developing the theory and enhancing the practice of network coding.
ACKNOWLEDGEMENTS

The authors would like to thank Ellen Esichen for useful discussions on arcs, Dana Ron for her comments on an earlier version of this manuscript and Amin Shokrollahi for stimulating discussions.

REFERENCES