Information Flow Decomposition for Network Coding

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

***Abstract*—We propose a method to identify structural properties of multicast network configurations, by decomposing networks into regions through which the same information flows. This decomposition allows us to show that very different networks are equivalent from a coding point of view, and offers a means to identify such equivalence classes. It also allows us to divide the network coding problem into two almost independent tasks: one of graph theory and the other of classical channel coding theory. This approach to network coding enables us to derive the smallest code alphabet size sufficient to code any network configuration with two sources as a function of the number of receivers in the network. But perhaps the most significant strength of our approach concerns future network coding practice. Namely, we propose deterministic 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 that can easily accommodate future changes in the network, e.g., addition of receivers and loss of links.**

***Index Terms*—Arcs, convolutional codes, decentralized codes, information flow, max-flow min-cut theorem, maximum distance separable (MDS) codes, network coding, network multicast.**

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, for each receiver, there are h edge-disjoint paths connecting the 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 simultaneously used by more than one unit-rate source. Information flows are sequences of bits, or if we look at n bits at a time, sequences of elements of some finite field $\mathbb{F}_q$ where $q = 2^n$. 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 $\mathbb{F}_q$ of the symbols the sources emit. Which symbol an edge carries is decided by its parent node, which cannot only forward 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, Yeung, 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 Médard in [3], who translated the network code design to an algebraic problem which depends on the structure of the underlying graph. Li, Yeung, and Cai showed constructively in [2] that multicast at rate h can be achieved by linear coding. The first deterministic polynomial-time algorithms for constructing linear codes for multicast were proposed in [4]–[6] and the first randomized in [20] and [21].

The basic idea of our approach to network coding is partitioning the network graph into subgraphs through which the same information flows. Processing (combining of different flows) happens only at the "border" of these subgraphs. 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 observe the information flow 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 a network with two sources and two receivers shown in Fig. 1(a). 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 first two flows will be referred to as *source flows* and the third as *coding flow*. They are connected as shown in Fig. 1(b). 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 linearly 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

Manuscript received June 15, 2004; revised June 2, 2005. This work was supported by DIMACS, NSF under Grant CCR-0325673, and FNS under Grant 200021-103836/1.

C. Fragouli is with the School of Computer and Communication Sciences, EPFL, Lausanne, CH-1015 Switzerland (e-mail: christina.fragouli@epfl.ch).

E. Soljanin is with Lucent Technologies, Bell Labs, Murray Hill, NJ 07974 USA (e-mail: emina@lucent.com).

Communicated by M. Médard, Associate Editor for Communications.

Digital Object Identifier 10.1109/TIT.2005.864435

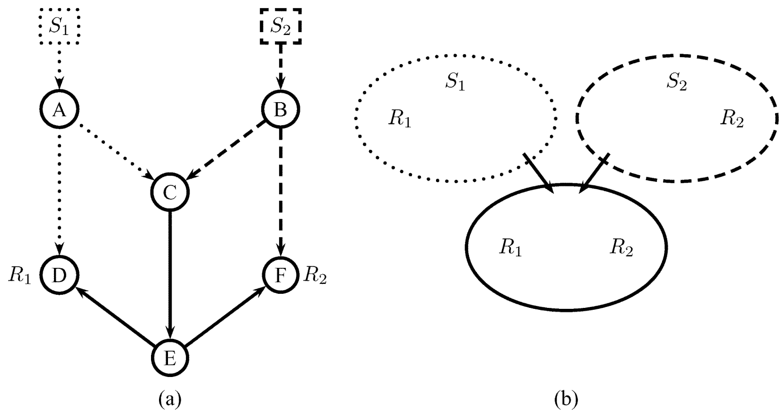

Fig. 1. The Butterfly network and its information flow decomposition. At each time slot, the sources S_1 and S_2 produce bits x_1 and x_2. Node C combines its incoming bits x_1 and x_2 to create, for example, the bit $x_1 + x_2$ that it then forwards toward receivers R_1 and R_2. Each receiver solves a system of linear equations to retrieve the sources.

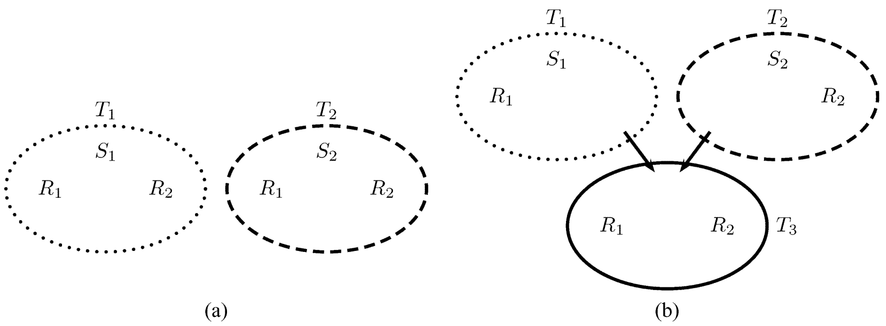

Fig. 2. Two possible information flow decompositions for networks with two sources and two receivers: (a) no coding required and (b) network coding necessary.

with h sources and N receivers satisfying the 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 flow decomposition. For example, we will show later that all networks with two sources and two receivers have one of the two possible information flow decompositions shown in Fig. 2. Fig. 2(a) shows the network scenario in which each receiver has access to both source flows, and thus no network coding is required. Fig. 2(b) shows the network scenario in which each receiver has access to a different source flow and a common coding flow.

Recognizing numerous structural properties that a flow 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 implementation 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, Yeung, and Cai [2] shows that the network multicast is possible if linear coding is performed over a sufficiently large finite field. Koetter and Médard upper-bounded the required alphabet size by hN in [3]. Sanders, Egner, and Tolhuizen in [4], and Ho *et al.* in [20] reduced the upper bound of the required alphabet size to N. Jaggi, Chou, and Jain made a claim in [5] that $\sqrt{2N + 1/4} - 1/2$ is a lower bound. Feder, Ron, and Tavory [12], using information theory arguments, also derived a lower bound of order $\sqrt{2N}(1 - o(1))$, and in addition provided upper bounds for some specific network configurations. The lower bound of $O(\sqrt{N})$ was also found by Rasala-Lehman and Lehman in [14]. 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 affects the maximum number of users that can be accommodated.

We derive the code alphabet size bounds by bounding the chromatic number of a class of graphs defined based on the information flow decompositions of networks with $h = 2$ sources and N receivers. The code design itself amounts to the vertex coloring problem of this class of graphs. This connection allowed us to directly apply results from coloring in [8] to derive the code alphabet size bounds. The authors in [14] independently used similar arguments that reduce the problem of network code design for some special graphs 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 in a distributed manner without the knowledge of the overall network topology and easily extended to accommodate future changes in the network, such as addition of receivers or loss of links, are particularly desirable in practice, but have not yet received adequate attention in 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 [21], [22] alleviate this problem, at the cost of an error probability and increased decoding complexity. We propose a *deterministic* method to design decentralized codes, which is also based on the information flow 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 flow decomposition of the network remains the same. The authors in [25] have recently used the information flow decomposition approach for decentralized coding over wireless networks.

For networks with delay, information flow 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 information flow 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 [12] proposed to add node-memory to increase the alphabet size a network can support, which has a similar flavor to our proposed binary encoder 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. We describe decentralized network coding in Section IV and derive alphabet size bounds in Section V. In Section VI, we investigate connections with vertex coloring, and in Section VII, connections with convolutional codes. Section VIII concludes the paper.

II. The Network Coding Model

We consider a communication 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, as we will discuss in Section VII-A. There are h unit rate information sources $S_1,\dots,S_h$ and N receivers $R_1,\dots,R_N$. The number of edges of the min-cut between the sources and each receiver node is h. The h sources multicast information simultaneously to all N receivers at rate h. As in the previous work (e.g., [2]), we assume zero delay meaning that all nodes simultaneously receive all their inputs and produce their outputs.

We denote by $\{(S_i,R_j), 1\le i\le h\}$ a set of h edge-disjoint paths from the sources to the receiver R_j. Under the min-cut assumption, the existence of such paths is guaranteed by the Menger theorems (see, for example, [17, 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 $(S_i,R_j), 1\le i\le h, 1\le j\le N$. 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 σ_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 (possibly different) linear combinations of its inputs to its output edges. Consequently, through each edge of G' flows a linear combination of source symbols; namely, the symbol flowing through some edge e of G' is given by

$$c_1(e)\sigma_1+\cdots+c_h(e)\sigma_h=\underbrace{[c_1(e)\quad\cdots\quad c_h(e)]}_{c(e)}\begin{bmatrix}\sigma_1\\ \sigma_2\\ \vdots\\ \sigma_h\end{bmatrix}$$

where the vector $c(e)=[c_1(e)\;\; c_2(e)\;\;\cdots\;\; c_h(e)]$ 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 need to 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 ρ_i^j be the symbol on the last edge of the path (S_i,R_j), and $\boldsymbol{A}_j$ the matrix whose ith 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

$$\begin{bmatrix}\rho_1^j\\ \rho_2^j\\ \vdots\\ \rho_h^j\end{bmatrix}=\boldsymbol{A}_j\begin{bmatrix}\sigma_1\\ \sigma_2\\ \vdots\\ \sigma_h\end{bmatrix} \qquad (1)$$

to retrieve the information symbols $\sigma_i, 1\le i\le h$, transmitted from the h sources. The network code design problem is to select a coding vector for each edge of the network so that the matrix $\boldsymbol{A}_j$ 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 section, we will use the example network with two sources multicasting to the same set of three receivers shown in Fig. 3. Since for a network code we eventually have to describe which operations each node in G' has to perform on its

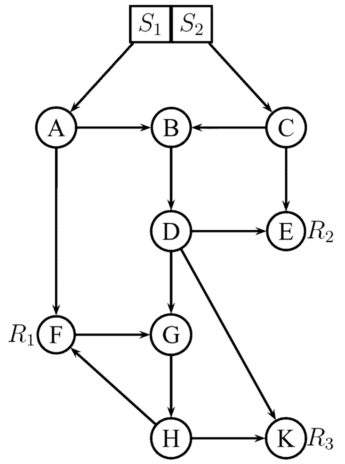

Fig. 3. Network with two unit rate sources $\{S_1, S_2\}$ and three receivers $\{F, E, K\}$. Each edge has unit capacity.

inputs for each of its outgoing edges, we find it more transparent to work with the graph

$$\gamma = \bigcup_{\substack{1 \le i \le h \\ 1 \le j \le N}} L(S_i, R_j),$$

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). The graph γ for our example network of Fig. 3 is depicted in Fig. 4(a).

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 γ 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. 4(a), 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 vertices of the line graph into subsets V_i^l so that the following holds:

1) each V_i^l contains exactly one source node or a coding point, and
2) each node that is neither a source node nor a coding point belongs to the V_i^l which contains its closest ancestral coding point or source node.

Let T_i be the subgraph of γ induced by V_i^l (namely, V_i^l together with the edges whose both endpoints are in V_i^l).

Theorem 1: The line graph γ and its subgraphs T_i satisfy the following properties.

1) Each subgraph T_i of γ is a tree with root vertex either a coding point or a source node.
2) The same linear combination of source symbols flows through all the nodes in γ (edges in the original graph) that belong to the same T_i.
3) For each receiver R_j, there exist in γ, h vertex-disjoint paths from the source subtrees to the receiver nodes of R_j.
4) For each receiver R_j, the h receiver nodes corresponding to the last edges on the paths $(S_i, R_j), 1 \le i \le h$, belong to distinct subtrees.
5) Each subtree contains at most N receiver nodes.

Proof:

1) By construction, T_i contains no cycles, the source node (or coding point) has no incoming edges, and there exists a path from the source node (or coding point) to each other vertex of T_i. Thus, T_i is a tree rooted at the source node (coding point).
2) Since T_i is a tree, the linear combination that flows through the root source node or coding point will also have to flow through the rest of the nodes.
3) Because the min-cut condition is satisfied in the original network G', there exist h edge-disjoint paths to each receiver. Edge-disjoint paths in the original graph correspond to vertex-disjoint paths in the line graph γ.
4) Assume that the receiver nodes corresponding to the last edges of paths (S_k, R_j) and (S_l, R_j) belong to the same subtree T_i. Then both paths go through the root vertex (source node or coding point) of subtree T_i. But by the previous claim, the paths for receiver R_j are vertex-disjoint.
5) This claim holds because there are N receivers and, by the previous claim, all receiver nodes contained in the same subtree are distinct. □

Since each subgraph T_i of γ is a tree, we shall call the process described above *subtree decomposition*, and T_i a *source subtree* if it starts with a source node or a *coding subtree* if it starts with a coding point. Note that subtree decomposition partitions the network into subgraphs through which the same information flows; hence the name information flow decomposition.

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. The resulting combinatorial object, which we will refer to as the *subtree graph* Γ, is defined by its underlying topology (V_Γ, E_Γ) and the association of receivers with nodes V_Γ. Fig. 4(b) shows the subtree graph for 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. We shall use the notation T_i to describe both the subgraph of the line graph γ, and the corresponding node in the subtree graph Γ, depending on the context. Note that claims 3)–5) in Theorem 1 translate directly to properties of the subtree graph Γ.

Network coding assigns an h-dimensional coding vector $c(T_i) = [c_1(T_i) \ \cdots \ c_h(T_i)]$ to each subtree T_i. The flow 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 $\boldsymbol{A}_j$ and solves the system of linear equations (1).

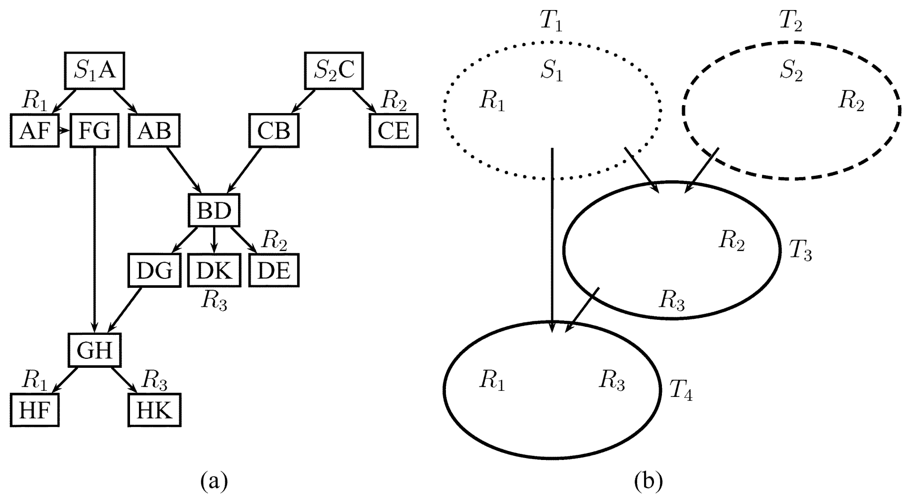

Fig. 4. Line graph with coding points BD and GH for the network in Fig. 3 and its subtree graph.

Definition 2: An assignment of coding vectors to subtrees is *feasible* if the coding vector of a subtree lies in the linear span of the coding vectors of the subtree's parents.

Definition 3: A *valid* network code is any feasible assignment of coding vectors to subtrees such that the matrix $\boldsymbol{A}_j$ is full rank for each receiver j.

Note that whether a code is valid depends on both the underlying topology of the subtree graph and the distribution of the receivers over the subtrees whereas feasibility depends only on the topology.

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. 4(b):

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

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

$$A_1 = \begin{bmatrix} c(T_1) \\ c(T_4) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$
$$A_2 = \begin{bmatrix} c(T_2) \\ c(T_3) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$$
$$A_3 = \begin{bmatrix} c(T_3) \\ c(T_4) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}. \qquad \square$$

B. Minimal Subtree Graphs and Their Properties

For a given communication network graph $G = (V, E)$, the choice of the h edge-disjoint paths $(S_i, R_j), 1 \leq i \leq h$, is not unique, and thus the network subtree decomposition is not unique either. We are interested in a subtree decomposition which is in a certain sense minimal. As we will discuss later, if a subtree decomposition is not minimal, it may require a less efficient network code (in terms of network resources) than the minimal one.

In Theorem 1, we observed that the min-cut condition in the network G implies that in the graph γ, and as a consequence in the subtree graph Γ, for each receiver R_j, the paths (S_i, R_j) 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 4: 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. 5(a). Notice that node x in Fig. 5(a) and its incident edges can be removed without affecting the min-cut conditions in the network. The resulting graph is then identical to the one shown in Fig. 1(a) in Section I, whose subtree graph is shown in Fig. 1(b). Consider now the choice of two sets of edge-disjoint paths (corresponding to the two receivers) shown in Fig. 5(b) and (c). The resulting subtree graph shown in Fig. 5(d) has more edges and nodes than the one shown in Fig. 1(b). We see, however, that the edge between T_1 and T_4 and the edge between T_2 and T_3 can be removed without violating the multicast property. Removing these edges allows us to incorporate T_3 into T_1 and T_4 into T_2, and consequently obtain the subtree graph shown in Fig. 1(b).

In the previous example, we implicitly described a method to identify a minimal subtree graph. We first find paths toward every receiver, and then construct a subtree graph. Each subtree graph $\Gamma = (V_\Gamma, E_\Gamma)$ can be reduced to a minimal subtree graph, by sequentially examining each edge in E_Γ. If removing an edgedoes not violate the multicast property, we remove it, and proceed to examine the next one. Note that this is a polynomial time algorithm in the number of nodes $|V_\Gamma|$ because checking if the multicast property holds after removing an edge and the number of edges $|E_\Gamma|$ are both polynomial in $|V_\Gamma|$.

Identifying a minimal subtree graph before multicasting may allow us to reduce the number of coding points and thus to use less network resources. Another reason we are interested in minimal subtree graphs is that, as we will see next, such graphs have a number of structural properties, which can be exploited to derive certain theoretical results. For example, in Section V-A, we will use properties of minimal subtree graphs to derive the

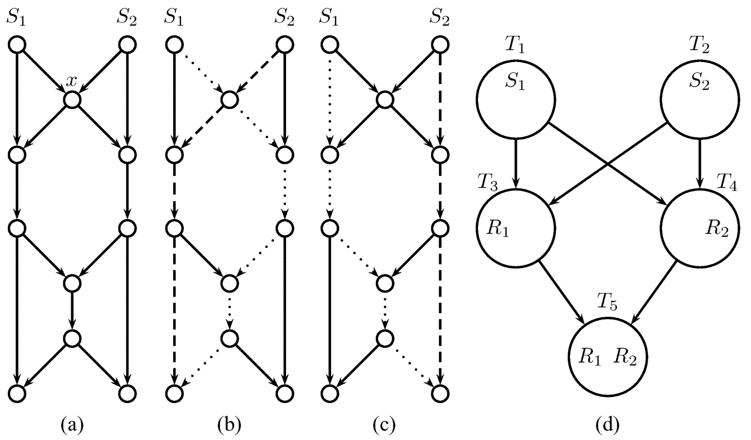

Fig. 5. A network with two sources and two receivers: (a) the original graph, (b) two edge-disjoint paths from the sources to the receiver R_1, (c) two edge-disjoint paths from the sources to the receiver R_2, and (d) the resulting nonminimal subtree graph.

largest field size that a code for a network with two sources and N receivers may require.

Note that the multicast property is satisfied in Γ 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 3) for a subtree graph which does not satisfy the multicast property.

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

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

1) A valid network code where a subtree is assigned the same coding vector as one of its parents does not exist.
2) A valid network code where the vectors assigned to the parents of any given subtree are linearly dependent does not exist.
3) A valid network code where the coding vector assigned to a child belongs to a subspace spanned by a proper subset of the vectors assigned to its parents does not exist.
4) Each coding subtree has at most h parents.
5) If a coding subtree has $2 \leq P \leq h$ parents, then there exist P vertex-disjoint paths from the source nodes to the subtree.

Proof:

1) Suppose a subtree is assigned the same coding vector as one of its parents. Then removing the edge(s) between the subtree and the other parent(s) results in a subtree graph with a valid coding vector assignment. By Lemma 1, the multicast property is also satisfied. But we started with a minimal subtree graph and removed edges without violating the multicast property, which contradicts Definition 4.
2) Suppose there is a subtree T whose parents $T_1 \ldots T_P$ are assigned linearly dependent vectors $c_1 \ldots c_P$. Without loss of generality, assume that c_1 can be expressed as a linear combination of $c_2 \ldots c_P$. Then removing the edge between T and T_1 results in a subtree graph with a valid coding vector assignment. From Lemma 1, the multicast property is also satisfied. But this contradicts Definition 4.
3) Suppose there is a subtree T whose parents $T_1 \ldots T_P$ are assigned vectors $c_1 \ldots c_P$. Without loss of generality assume that T is assigned a vector c that is a linear combination of $c_2 \ldots c_P$. Then removing the edge between T and T_1 results in a subtree graph with a valid coding vector assignment. From Lemma 1 the multicast property is also satisfied. But this contradicts Definition 4.
4) Since coding vectors are h-dimensional, this claim is a direct consequence of claim 2).
5) By claim 2), the coding vectors assigned to the P parent subtrees must linearly independent, which requires the existence of P vertex disjoint paths. □

The first three claims of Theorem 2 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 for networks with two sources follow from Theorems 2 and 1.

Theorem 3: In a minimal subtree decomposition of a network with $h = 2$ sources and N receivers we have the following.

1) A parent and a child subtree have a child or a receiver in common or both.
2) Each coding subtree contains at least two receiver nodes.
3) Each source subtree contains at least one receiver node.

Proof:

1) Suppose that the minimal subtree graph has two source and m coding subtrees. Consider a network code which assigns $[0\ 1]$ and $[1\ 0]$ to the source subtrees, and $[1\ \alpha^k], 0 \leq k \leq m$, where α is a primitive element of $\mathbb{F}_q, q > m$ to coding subtrees. This code is valid (see Definition 3) since any two different coding vectors form a basis for $\mathbb{F}_q^2$, and thus any other coding vector on the line is in their span. Let T_i and T_j denote a parent and a child subtree with no child or receiver in common. Now, alter the code by assigning the coding vector of T_i to T_j, and keep the rest of coding vectors unchanged. This assignment is feasible (Definition 2) because T_i and T_j do not share a child, which would have to be assigned a scalar multiple of the coding vector of T_i and T_j. Since T_i and T_j do not share a receiver, the code is still valid.

Therefore, by claim 2) of Theorem 2, the configuration is not minimal, which contradict the assumption.

2) Consider a coding subtree T. 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 two parents, then T has two receiver nodes. If T and one of the parents, say P_1, do not have a receiver in common, then they have a child in common, say C_1. Similarly, if T and C_1 do not have receiver in common, then they have a child in common. And so forth, following the descendants of P_1, one eventually reaches a child of T that is a terminal node of the subtree graph, and thus has no children. Consequently, T has to have a receiver in common with this terminal subtree. Similarly, if T and P_2 do not have a child in common, there exists a descendant of P_2 and child of T which must have a receiver in common with T.

3) If the two source subtrees have no children, then each must contain N receiver nodes. If the source subtree has a child, say T_1, then by claim 1) it will have a receiver or a child in common with T_1. Following the same reasoning as in the previous claim, we conclude that each source subtree contains at least one receiver node. □

Theorem 4: In a minimal subtree decomposition of a network with two sources and N receivers, the number of coding subtrees is upper-bounded by $N-1$. There exist networks which have a minimal subtree decomposition that achieves this upper bound.

Proof: Recall that there are exactly $2N$ receiver nodes. The first part of the claim then follows directly from Theorem 3. Fig. 6 demonstrates a minimal subtree graph for a network with two sources and N receivers that achieves the upper bound on the maximum number of subtrees. □

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

Proof: The scenario shown in Fig. 2(a) 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 can only exist one coding subtree containing two receiver nodes. □

Continuing along these lines, it is easy to show for example that there exist exactly three minimal configurations with two sources and three receivers, and seven minimal configurations with two sources and four receivers. In fact, one can enumerate all the minimal configurations with a given number of sources and receivers.

IV. Decentralized Codes

As discussed in Section III-A, network coding assigns an h-dimensional coding vector $c(T_i) = [c_1(T_i) \ \cdots \ c_h(T_i)]$ to each subtree T_i. The flow 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 $\boldsymbol{A}_j$ and solves the system of linear equations (1). Since the flow through the subtree corresponding to the source S_i is σ_i, we assign coding vectors

$$e_1 = [1\,0\,\cdots\,0],\ e_2 = [0\,1\,\cdots\,0],\ \ldots,\ e_h = [0\,0\,\cdots\,1]$$

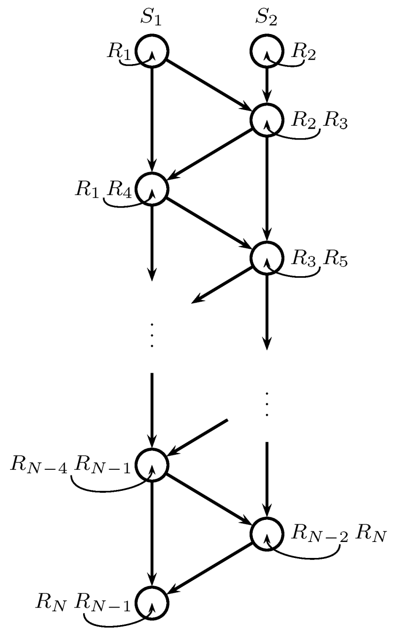

Fig. 6. A minimal subtree graph for a network with two sources, N receivers and $N-1$ coding subtrees.

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 [10] using only *local* information in the following sense:

Definition 5: Decentralized network coding assigns coding vectors to subtrees by 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.

Note that the algorithms proposed in [4]–[6] are not decentralized according to this definition.

To make decentralized coding possible, we need special sets of coding vectors 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 sets of coding vectors and then algorithms for decentralized network coding. It is interesting to note that the set of coding vectors we use will, for a network with h sources, correspond to the columns of the generator matrix of an h-dimensional maximum distance separable (MDS) code.

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 $\mathbb{PG}(h-1,q)$ defined as follows.

Definition 6: 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

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

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

$$[0\,1], \quad [1\,0], \quad \text{and} \quad [1\,\alpha^i], \qquad \text{for } 0 \le i \le q-2 \tag{2}$$

where α is a primitive element of $\mathbb{F}_q$. Any two different points on the projective line $\mathbb{PG}(1,q)$ form a basis for $\mathbb{F}_q^2$, and thus, any point on the line is in their span. Consequently, any code which assigns different points of $\mathbb{PG}(1,q)$ to different subtrees 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 are distributed over the graph.

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

1) *A Decentralized Code*: Since there are two source and two coding subtrees, we take the first four points from 2)

$$\begin{aligned} c(T_1) &= [1\,0], \quad c(T_2) = [0\,1],\\ c(T_3) &= [1\,1], \quad c(T_4) = [1\,\alpha]. \end{aligned}$$

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 subtrees, we see that T_2 and T_4 can be assigned the same coding vector since they do not share any receivers:

$$\begin{aligned} c(T_1) &= [1\,0], \quad c(T_2) = [0\,1],\\ c(T_3) &= [1\,1], \quad c(T_4) = [0\,1]. \end{aligned}$$

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 $\mathbb{PG}(h-1,q)$ form a basis for $\mathbb{F}_q^h$. A set of points that has this property is called an *arc*.

Definition 7: 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 $\mathbb{PG}(h-1,q)$, 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 8: Set $\mathcal{A}$ of vectors in $\mathbb{F}_q^h$ are said to be in general position if any h vectors in $\mathcal{A}$ 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 $\mathbb{PG}(h-1,q)$

$$\begin{matrix} 1 & 0 & \cdots & 0\\ 0 & 1 & \cdots & 0\\ \vdots & \vdots & \cdots & \vdots\\ 0 & 0 & \cdots & 1\\ 1 & 1 & \cdots & 1. \end{matrix}$$

□

Example 4: For $q+1 \ge h$, the following set of $q+1$ points are in general position in $\mathbb{F}_q^h$, and form an arc in $\mathbb{PG}(h-1,q)$:

$$\begin{matrix} 1 & x_1 & x_1^2 & \cdots & x_1^{h-1}\\ 1 & x_2 & x_2^2 & \cdots & x_2^{h-1}\\ \vdots & \vdots & \vdots & \cdots & \vdots\\ 1 & x_q & x_q^2 & \cdots & x_q^{h-1}\\ 0 & 0 & 0 & \cdots & 1 \end{matrix}$$

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 9: A normal rational curve is any set of $q+1$ points projectively equivalent to the set

$$\{(1\,t\,t^2\,\cdots\,t^{h-1})\,|\,t \in \mathbb{F}_q\} \cup \{(0\,0\,\cdots\,0\,1)\}.$$

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, and we make sure that no other subtree uses the same vector (we can achieve that in a decentralized manner as we will see in the following when we discuss algorithms), we do not need to know other subtree's coding vectors or the structure of the network. We do, however, have to know which sources are available at the subtree being coded as we have to choose a coding vector with zeros at the coordinates corresponding to the sources which are not available.

Given a subtree decomposition Γ 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 $\mathbb{PG}(h-1,q)$ with the smallest possible field size q.

Arcs have been used by coding theorists in the context of MDS codes. Consider a matrix G with columns the vectors in general position in $\mathbb{F}_q^h$. Matrix G is a generator matrix for an MDS code of dimension h over $\mathbb{F}_q$. The maximum length $g(h,q)$ such a code can have, or equivalently, the maximum size of an arc, is not known in general. Although the problem looks combinatorial in nature, most of the harder results on the maximum size of arcs have been obtained by using algebraic geometry (see [16] 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 [16]. 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 $h \ge q$, we know that $g(h,q) = h+1$, whereas, for $h \le q$, it holds that $g(h,q) \ge h+1$, 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 use theorems about geometry of arcs or develop algorithms to generate appropriate arcs, as discussed in the following sections.

B. Algorithms for Subtree Graph Design

All our algorithms for code design start with the information flow decomposition (described in Section III), whose output is a subtree graph. The proposed decentralized code design algorithms are founded on two key ideas: 1) reduce the number of coding points in the network, and 2) use as coding vectors points in arcs to perform coding in a decentralized manner. Recall that coding points correspond to terminals in the network that need to perform coding operations, and thus have enhanced functionalities. Reducing the number of coding points, which can be achieved by identifying minimal configurations described in Section III-B, lowers the number of such special terminals and decreases the operational complexity of the network. We first outline Algorithm IV.1 that finds the minimal subtree graph for a given communication network graph $G = (V, E)$, and a choice of the h edge-disjoint paths $(S_i, R_j), 1 \leq i \leq h, 1 \leq j \leq N$. The details were discussed in Section III-B. Note that this procedure is not decentralized.

```
Algorithm IV.1: MINIMAL SUBTREE GRAPH ($(S_i, R_j),\ 1 \leq i \leq h,\ 1 \leq j \leq N$)
$\gamma = (V_\gamma, E_\gamma) \leftarrow \cup_{\substack{1\leq i\leq h\\ 1\leq j\leq N}} L(S_i, R_j)$
$\mathcal{C} \leftarrow$ coding points of $\gamma$
$\mathcal{E} \leftarrow$ edges of $\gamma$ terminating in $\mathcal{C}$
for each $e \in \mathcal{E}$
    if $(V_\gamma, E_\gamma \setminus \{e\})$ satisfies the multicast property
    then
        $E_\gamma \leftarrow E_\gamma \setminus \{e\}$
        $\gamma \leftarrow (V_\gamma, E_\gamma)$
        $\mathcal{C} \leftarrow$ coding points of $\gamma$
        $\mathcal{E} \leftarrow$ edges of $\gamma$ terminating in $\mathcal{C}$
Find the subtree graph $\Gamma$ corresponding to $\gamma = (V_\gamma, E_\gamma)$
return ($\Gamma$)
```

We next outline Algorithm IV.2 that finds a reduced-state subtree graph for networks with two sources, which is not necessarily minimal but has at most $N-1$ coding subtrees. We know that such a configuration always exist by Theorem 3, which asserts that a minimal configuration with two sources and N receivers can have at most $N-1$ coding subtrees.

```
Algorithm IV.2: REDUCED STATE SUBTREE GRAPH ($(S_1, R_j), (S_2, R_j)$ $1 \leq j \leq N$)
$\gamma = (V_\gamma, E_\gamma) \leftarrow \cup_{\substack{1\leq i\leq h\\ 1\leq j\leq N}} L(S_i, R_j)$
$\mathcal{C} \leftarrow$ coding points of $\gamma$
$\mathcal{E} \leftarrow$ edges of $\gamma$ terminating in $\mathcal{C}$
while $|\mathcal{C}| \geq N$
    for each $e \in \mathcal{E}$
        if $(V_\gamma, E_\gamma \setminus \{e\})$ satisfyies the multicast property
        then
            $E_\gamma \leftarrow E_\gamma \setminus \{e\}$
            $\gamma \leftarrow (V_\gamma, E_\gamma)$
            $\mathcal{C} \leftarrow$ coding points of $\gamma$
            $\mathcal{E} \leftarrow$ edges of $\gamma$ terminating in $\mathcal{C}$
Find the subtree graph $\Gamma$ corresponding to $\gamma = (V_\gamma, E_\gamma)$
return $\Gamma$
```

In the special case of networks with two sources, we can find a minimal subtree graph in a distributed manner based on the following observation. From Theorem 3, we know that in a minimal subtree graph, a coding subtree has either a receiver or a child subtree in common with each of its parent subtrees. Therefore, if in a subtree graph, a parent and a child do not share a receiver or a child, the edge between them can be removed. The distributed algorithm for finding a minimal subtree graph is based on the assumption that subtrees can exchange information with their parents and children, and the assumption that each coding subtree T knows (i.e., locally stores the information) 1) the set of the receiver nodes $\mathfrak{R}(T)$ it contains and 2) its two parents $P_1(T)$ and $P_2(T)$, and 3) its set of children $\mathfrak{C}(T)$. Recall that each coding point is incident to a terminal of our network, which can store and process information about the subtree starting with that coding point. When we say that a "subtree stores/sends/receives information," we mean that, in the physical network, the terminal associated with the coding point (root) of the subtree performs these actions. Note that the same terminal may serve as the root for more than one subtrees.

The algorithm follows a bottom-up approach: initially, the subtrees that have no children (i.e., the leafs of the subtree graph) will first examine whether they can be incorporated with their parents, then, depending on their findings, initiate the local update of the subtree graph, and finally, send an update information to their parents. Subsequently, the subtrees will repeat this procedure as soon as they receive an update from all their children. Algorithm IV.3 outlines the described procedure. Here, $\mathcal{V}'$ denotes the set of subtrees that have completed their local processing, and $\mathcal{V}$ the set of subtrees that can start processing (leafs or those that have just received an update from all their children).

```
Algorithm IV.3: MINIMAL SUBTREE GRAPH ($(S_1, R_j), (S_2, R_j)$ $1 \leq j \leq N$)
$\gamma = (V_\gamma, E_\gamma) \leftarrow \cup_{\substack{1\leq i\leq 2\\ 1\leq j\leq N}} L(S_i, R_j)$
$\mathcal{C} \leftarrow$ coding points of $\gamma$
Find the subtree graph $\Gamma = (V_\Gamma, E_\Gamma)$ corresponding to $\gamma$
$\mathcal{V} \leftarrow$ leafs of $\Gamma$
$\mathcal{V}' \leftarrow \emptyset$
while $\mathcal{V}' \neq V_\Gamma$
do
    for each $T \in \mathcal{V}$
    do
        if $[\mathfrak{R}(T) \bigcap \mathfrak{R}(P_1(T)) = \emptyset$ & $\mathfrak{C}(T) \bigcap \mathfrak{C}(P_1(T)) = \emptyset]$
            then incorporate $T$ into $P_1(T)$
        else if $[\mathfrak{R}(T) \bigcap \mathfrak{R}(P_2(T)) = \emptyset$ & $\mathfrak{C}(T) \bigcap \mathfrak{C}(P_2(T)) = \emptyset]$
            then incorporate $T$ into $P_2(T)$
    $\mathcal{V}' \leftarrow \mathcal{V} \bigcup \mathcal{V}'$
    $\mathcal{V} \leftarrow \emptyset$
    for each $T \in \mathcal{C} \setminus \mathcal{V}'$
    do
        if $\mathfrak{C}(T) \subseteq \mathcal{V}$
            then $\mathcal{V} \leftarrow \mathcal{V} \bigcup \{T\}$
```

C. Codes for Networks With $h = 2$ Sources and N Receivers

Once the subtree graph has been obtained and the number of its nodes minimized or reduced, it remains to label the nodes by coding vectors. As discussed in a Section IV-A, to label the nodes of a subtree graph of a network with two sources, we can use the points on the projective line $\mathbb{PG}(1, q)$

$$[0\ 1], \quad [1\ 0], \quad \text{and} \quad [1\ \alpha^i], \qquad \text{for } 0 \leq i \leq q-2. \tag{3}$$

Recall that for a valid network code, it is sufficient and necessary that the coding vector associated with a subtree lies 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 are 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 is assigned a unique point of the projective line $\mathbb{PG}(1, q)$. We here present two algorithms which assign distinct coding vectors to the nodes in the subtree graph.

The first method is inspired by the carrier sense multiple access (CSMA) systems, where access to the network is governed by the possession of a *token*. The token is circulating around the network, and when a device needs to transmit data, it seizes the token when it arrives at the device. The token remains at the possession of the transmission device until the data transfer is finished. In our algorithm, a token will be generated for each coding vector. Each coding point (associated terminal) seizes a token and uses the associated coding vector. If a change in the network occurs, for example, receivers leave the multicast session, tokens that are no longer in use may be released back in the network to be possibly used by other coding points. Code design based on this method is outlined as Algorithm IV.4.

```
**Algorithm IV.4:** DECENTRALIZED CODE $((S_1, R_j), (S_2, R_j),\ 1 \le j \le N)$
Obtain $\Gamma$ by using any of the algorithms IV.1, IV.2, or IV.3
$\mathcal{C} \leftarrow$ coding points of $\Gamma$
Create $|\mathcal{C}|$ tokens, each associated with a different point in $\mathbb{PG}(1, |\mathcal{C}| - 1)$.
Circulate the tokens through the network until each gets seized by coding point.
```

An alternative simple way to organize a mapping from coding vectors to coding points is described below. Recall that at 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 coding subtree contains at least one receiver node associated with S_1 and at least one receiver node associated with S_2. Let $\mathfrak{R}(T; S_1) = \{R_{i_1}, R_{i_2}, \ldots, R_{i_u}\}$, where $i_1 < i_2 < \cdots < i_u$ be the set of receivers associated with S_1 in a given subtree T. We choose $[1\ \alpha^{i_1 - 1}]$ to be the label of that subtree. In 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 $q = N + 1$, as opposed to $q = N$. This is because for N receivers, we will use coding vectors from the set $[1\ \alpha^i]$ for $0 \le i \le q - 2$ with $q - 2 = N - 1$. Code design based on the described method is outlined as Algorithm IV.5.

```
**Algorithm IV.5:** DECENTRALIZED CODE $((S_1, R_j), (S_2, R_j), 1 \le j \le N)$
Obtain $\Gamma$ by using any of the algorithms IV.1, IV.2, or IV.3
$\mathcal{C} \leftarrow$ coding points of $\Gamma$
for each $T \in \mathcal{C}$
    do $p \leftarrow \arg\min\{j : R_j \in \mathfrak{R}(T; S_1), 1 \le j \le N\}$
       $c(T) \leftarrow [1\ \alpha^{p-1}]$
```

D. Codes for Networks With h Sources and Two 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 two receivers. We here show that there is only one valid binary code assignment for the minimal subtree graph of a network with two receivers, and that this assignment does not need global information. Since $N = 2$, from Theorem 1 claim 5), each coding subtree has at most two receiver nodes. Moreover, from Theorem 1 claim 3), since the paths to each receiver need to be vertex disjoint, and there exist $N = 2$ different receivers, each subtree can be shared by at most two paths, and thus it has exactly two inputs.

Theorem 5: The binary code that assigns to each source subtree, a different basis vector, and to each 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 two receivers.

Proof: From claim 1) of Theorem 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 coding subtree must be assigned the binary sum of the vectors assigned to its two parents. This is the only code that satisfies a necessary condition for validity. Since there exist binary codes for networks with two receivers, the code must be valid. This assignment indeed does not need global information. □

```
**Algorithm IV.6:** DECENTRALIZED CODE $((S_i, R_1), (S_i, R_2), 1 \le i \le N)$
Obtain $\Gamma$ by using Algorithm IV.1
Each coding point of $\Gamma$ performs binary addition of its inputs.
```

E. Codes for Bipartite 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 Fig. 7.

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:

$$\Pi_{12} \subseteq \mathbb{P}(\langle e_1, e_2 \rangle), \quad \Pi_{13} \subseteq \mathbb{P}(\langle e_1, e_3 \rangle),$$
$$\Pi_{23} \subseteq \mathbb{P}(\langle e_2, e_3 \rangle), \quad \text{and} \quad \Pi_{123} \subseteq \mathbb{P}(\langle e_1, e_2, e_3 \rangle)$$

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.

To design deterministic decentralized codes it is sufficient to select sets of coding vectors Π that can be used for all possible bipartite configurations with N receivers. For example, one configuration would be when all coding subtrees are connected to S_1 and S_2 only, and thus all coding vectors belong in Π_{12}. Another configuration would be when all coding subtrees have three parents, and thus all coding vectors belong in Π_{123}.

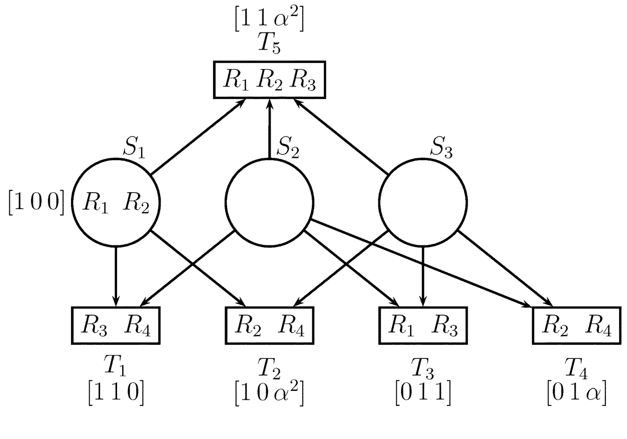

Fig. 7. A subtree graph for a network with three sources and four receivers labeled by the points of an arc in $\mathbb{PG}(2,4)$.

Sets that accommodate all these different cases have to satisfy the following conditions.

1) Each set contains $N+1$ elements.
2) Any three vectors in Π_{123} are linearly independent.
3) Any three different vectors such that not all of them belong to the same set are linearly independent.

The first condition holds because we are interested in bipartite configurations: if a subtree has l parents, then l paths share the coding point, and thus the subtree contains at least l receiver nodes.

Such sets of coding vectors for different numbers of receivers can be found by computer search, or constructed for example as follows. We start with a finite field $\mathbb{F}_q$ with a characteristic different then 2, and its quadratic extension $\mathbb{F}_{q^2}$. For set Π_{123}, we take elements of the form $[1\ \sqrt{\alpha^l}\ \alpha^l]$, where a is a primitive element, such that $\alpha^l \in \mathbb{F}_q$ but $\sqrt{\alpha^l} \notin \mathbb{F}_q$; thus, $\sqrt{\alpha^l} \in \mathbb{F}_{q^2}$ (see, for example, [18, Ch. 2]). For sets Π_{12}, Π_{23}, and Π_{13}, we take elements of the form $[1\ \alpha^i\ 0]$, $[0\ 1\ \alpha^i]$, and $[1\ 0\ \alpha^i]$, respectively, such that $\alpha^i \in \mathbb{F}_q$. For the set Π_{13}, use α^l such that $\sqrt{\alpha^i}\sqrt{\alpha^j} \neq \alpha^l$, for α^i, α^j employed in set Π_{123}. Additionally, we use a mapping such that not both the vectors $[1\ 0\ \alpha^k]$ and $[0\ 1\ \alpha^k]$ are used for the same configuration. Similarly for the vectors $[1\ 0\ \alpha^k]$ and $[1\ \sqrt{\alpha^k}\ \alpha^k]$.

Example 5: Let $\mathbb{F}_{3^2}$ and $\mathbb{F}_{3^4}$. Then

$$\{1, \alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6, \alpha^7\} \in \mathbb{F}_{3^2}$$

where α and is a primitive element for $\mathbb{F}_{3^2}$, and

$$\{\beta = \sqrt{\alpha}, \beta^3 = \sqrt{\alpha^3}, \beta^5 = \sqrt{\alpha^5}, \beta^7 = \sqrt{\alpha^7}\} \notin \mathbb{F}_{3^2}.$$

For example, in Fig. 7, depending on the receiver nodes inside each subtree and the employed mapping we can associate with each coding subtree the following coding vectors: $T_1 : [1\ \alpha\ 0]$, $T_2 : [1\ 0\ 1]$, $T_3 : [0\ 1\ \alpha^2]$, $T_4 : [0\ 1\ \alpha^3]$, $T_5 : [1\ \sqrt{\alpha}\ \alpha]$.

The requirement that the same set of coding vectors apply for all possible configurations leads to an increase of the required alphabet size. Alternatively, we can optimize the coding vectors for particular classes of configurations and achieve a smaller alphabet size. 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 6: Suppose we need six three-dimensional vectors in general position such that: two are in $\mathbb{P}(\Pi_{12})$, two in $\mathbb{P}(\Pi_{23})$, one in $\mathbb{P}(\Pi_{13})$, one in $\mathbb{P}(\Pi_{123})$. Is there such an arc in $\mathbb{PG}(2,4)$? In other words, can we start with a known arc of length 6 in $\mathbb{PG}(2,4)$ (such as the one on the left-hand side in (4) below), and obtain the arc we are interested in by applying a change of basis in the projective space. In this particular case, the answer is positive, and the desired arc is obtained as follows:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & \alpha & \alpha^2 \\ 1 & \alpha^2 & \alpha \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & \alpha \\ 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & \alpha \\ 0 & 1 & 1 \\ 1 & 0 & \alpha^2 \\ 1 & 1 & 0 \\ 1 & 1 & \alpha^2 \end{bmatrix} \tag{4}$$

□

F. Codes for Networks With h Sources and N Receivers That Use Additional Resources

We here demonstrate how the code design problem can be simplified at the cost of using some additional network resources. We are motivated by applications, such as overlay and *ad hoc* networks, where we have at our disposal large graphs rather then predetermined sets of paths from sources to receivers.

In a network with $|\mathcal{C}|$ coding subtrees, a simple decentralized code design is possible if we are willing to use an alphabet of size $|\mathcal{C}| + h - 1$ as well as additional network resources (edges and terminals) to ensure that the min-cut to each coding subtree be h, because of the following fact.

Theorem 6: If a minimal subtree decomposition of a network with h sources has $|\mathcal{C}|$ coding subtrees, where the min-cut to each coding subtree is h, the alphabet of size $|\mathcal{C}| + h - 1$ is sufficiently large for decentralized coding.

Proof: We can use the $|\mathcal{C}| + h$ points in a normal rational curve (see Definition 9) in $\mathbb{PG}(h-1, |\mathcal{C}| + h - 1)$ to assign h vectors to the source subtrees and $|\mathcal{C}|$ vectors to the coding subtrees. □

Algorithm IV.7 outlines the procedure for code design in the described scenario.

```
Algorithm IV.7: DECENTRALIZED CODE $((S_i, R_j), 1 \le i \le h, 1 \le j \le N)$
Obtain $\Gamma$ by using Algorithm IV.1
$\mathcal{C} \leftarrow$ coding points of $\Gamma$

for each $T \in \mathcal{C}$
    for $1 \le i \le h$
        do if $T$ does not have a connection
              with source $S_i$
           then create a connection with source $S_i$
Coding vectors are the points of the normal rational curve in
    $\mathbb{PG}(h-1, |\mathcal{C}| + h - 1)$.
Label coding points by using the approach of Algorithm IV.4 or
    Algorithm IV.5.
```

Here, we can think of coding points as edges incident to special network terminals, that have not only enhanced computational power, but also enhanced connectivity. Note that, even in this case, multicast with network coding requires fewer resources then multicast without coding, because multicast without coding is possible iff the min-cut toward *every* node of the graph is h, not just the coding points.

Another way to take advantage of the increased connectivity is to (if possible) partition the network G with h sources (h even) into $h/2$ independent two-source configurations, and then code each subnetwork separately in a possibly decentralized manner. Algorithm IV.8 outlines this procedure.

```
Algorithm IV.8: PAIRED-SOURCES CODE ($G$)
Group sources into pairs.
for each pair of sources $(S_{i_1}, S_{i_2})$
    Find paths $(S_{i_1}, R_j)$, $(S_{i_2}, R_j)$, $1 \le j \le N$ in $G$
    Design a network code based on these paths.
    Update $G$ by removing the used paths.
```

G. Scalability

Recall that whether a code is valid depends on both the underlying topology of the subtree graph and the distribution of the receivers over the subtrees. 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 their subtree decomposition retains the same topology, regardless of the distribution of the receivers, or the new subtree graph contains the original subtree graph.

In a network using decentralized coding, the coding vectors associated with any h subtrees provide a basis of the h-dimensional space. We can think of subtrees as "secondary sources" and allow the new receivers to connect to any h different subtrees. Thus, we can extend the multicast network, without any coding/decoding changes for existing users. Subtree decomposition enables scalability in the described manner even when the code is not decentralized, since we can have new receivers contact subtrees, much like today terminals contact databases, until they connect to h subtrees that allow them to retrieve the source information.

In the above scenario, the receivers are allowed to join the network only in a way that will not change the topology of the subtree graph. If, for example, in a network with two sources, addition of new receivers results in new subtrees without disturbing the existing ones, then the existing code can be simply extended without any coding/decoding changes for existing users. Note that the projective line $\mathbb{PG}(1,q)$ can be thought of as a subset of the projective line $\mathbb{PG}(1,q')$, where $\mathbb{F}_{q'}$ 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 $\mathbb{PG}(1,q')$.

In some cases, even the codes which are not decentralized can remain the same, and the subtree decomposition shows us how to ensure that. This point is 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. 8. We are interested in finding out if the two codes described in the Example 2 for the network in Fig. 3 are also valid for the network in Fig. 8. The topology of the subtree graphs for the networks are the same. Since the

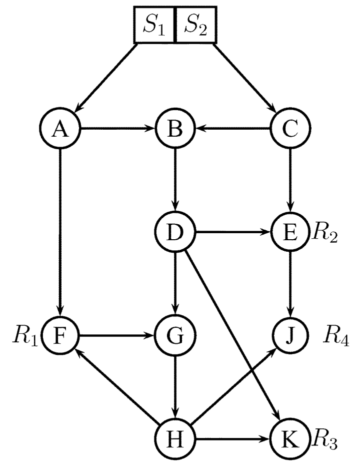

Fig. 8. Network with two sources $\{S_1, S_2\}$ and four receivers F, E, K, and J.

first code in the Example 2 is decentralized, it is also valid for the network in Fig. 8. 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 (i.e., observe one of the two corresponding coding vectors). 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.

Having a decentralized code which does not have to be redesigned as long as the topology of the subtree graph remains the same (regardless of the distribution of the receivers) does not always come with the price of having to use a larger alphabet size. Consider, for example, the network in Fig. 6. It is easy to see that a code over the binary alphabet where each coding point performs the binary addition of its two inputs is a valid code for this network. But if we introduce a receiver for each pair of subtrees that does not already share one, then all the subtrees will have to be assigned a different coding vector. Therefore, the only valid code for the latter scenario is effectively decentralized.

V. 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. This alphabet size is sufficient but not necessary for all networks with h sources and N receivers. Recall that the binary alphabet is sufficient for networks which require only routing. We will also characterize a class of networks that require the maximum alphabet size in the case with two sources.

Definition 10: We say that a network *requires* an alphabet of size q if there exists a valid network over an alphabet of size q but not over over an alphabet of size $q-1$.

We restrict our attention to minimal subtree graphs because a valid network code for a minimal subtree graph Γ directly translates to a valid network code for any subtree graph Γ' that can be reduced to Γ (the code for Γ simply does not use some edges of Γ'). Thus, an alphabet size sufficient for all possible minimal subtree graphs, will be sufficient for all possible nonminimal subtree graphs as well.

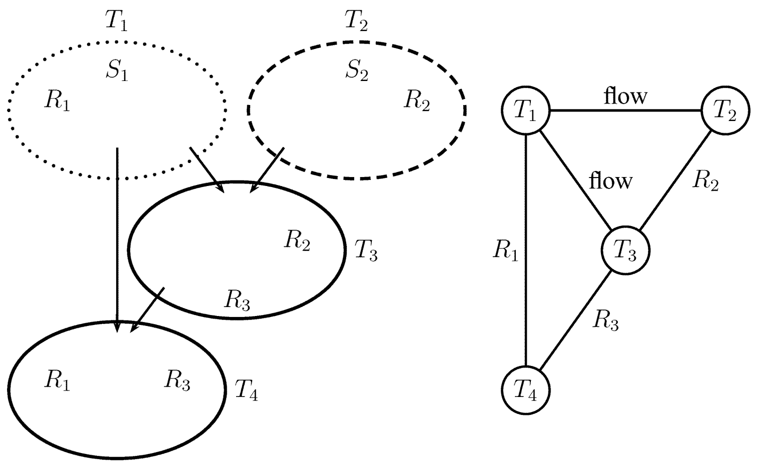

Fig. 9. A subtree graph Γ and its associated graph Ω. The receiver edges in Ω are labeled by the corresponding receivers.

A. Networks With Two Sources and N Receivers

To show that an alphabet of size q is sufficient, we can equivalently prove that we can construct a valid code by using as coding vectors the $k = q + 1$ different points in the projective line $\mathbb{PG}(1, q)$. Remember that any two such coding vectors form a basis of the two-dimensional space.

Let Γ be a minimal subtree graph with $n > 2$ vertices (subtrees); $n - 2$ is the number of coding subtrees. (Note that when $n = 2$, Γ has only source subtrees and no network coding is required.) We relate the problem of assigning vectors to the vertices of Γ to the problem of vertex coloring a suitably defined graph Ω. Let Ω be a graph with n vertices, each vertex corresponding to a different subtree in Γ. We connect two vertices in Ω 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 Ω with an edge which we call a *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 Ω with an edge which we call a *flow edge*. By Theorem 2, a parent and a child subtree cannot be assigned the same coding vector. However, we need not worry about this case separately since by Theorem 3, a parent and a child subtrees have either a child or a receiver in common. Fig. 9 plots Ω for our example subtree graph.

Lemma 2: For a minimal configuration with $n > 2$, every vertex in Ω has degree at least 2.

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 Ω, there should exist edges between a subtree and its parents, that may be either flow edges, or receiver edges, and the corresponding vertex has degree at least two. □

Lemma 3: [17, Ch. 9] Every k-chromatic graph has at least k vertices of degree at least $k - 1$.

Theorem 7: For any minimal configuration with two sources and 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 Ω has n vertices 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 Ω in two different ways.

1) By 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 Ω as
$$E(\Omega) \geq [k(k-1) + 2m]/2. \tag{5}$$
2) Since there are N receivers and $n - 2$ coding subtrees, we have at most N receiver edges and at most $n - 2$ flow edges. Thus,
$$E(\Omega) \leq N + n - 2 = N + k + m - 2. \tag{6}$$

From (5) and (6), we obtain

$$N \geq \frac{k(k-1)}{2} - k + 2. \tag{7}$$

Equation (7) 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, an alphabet of size $\lfloor \sqrt{2N - 7/4} + 1/2 \rfloor$ is sufficient.

To show that there exist configurations for which an alphabet of this size is necessary, we consider a network which has a minimal subtree graph with the same topology as the one shown in Fig. 6, but different number of receivers. There are k subtrees in the graph and $k-2$ of them share a child with another subtree.

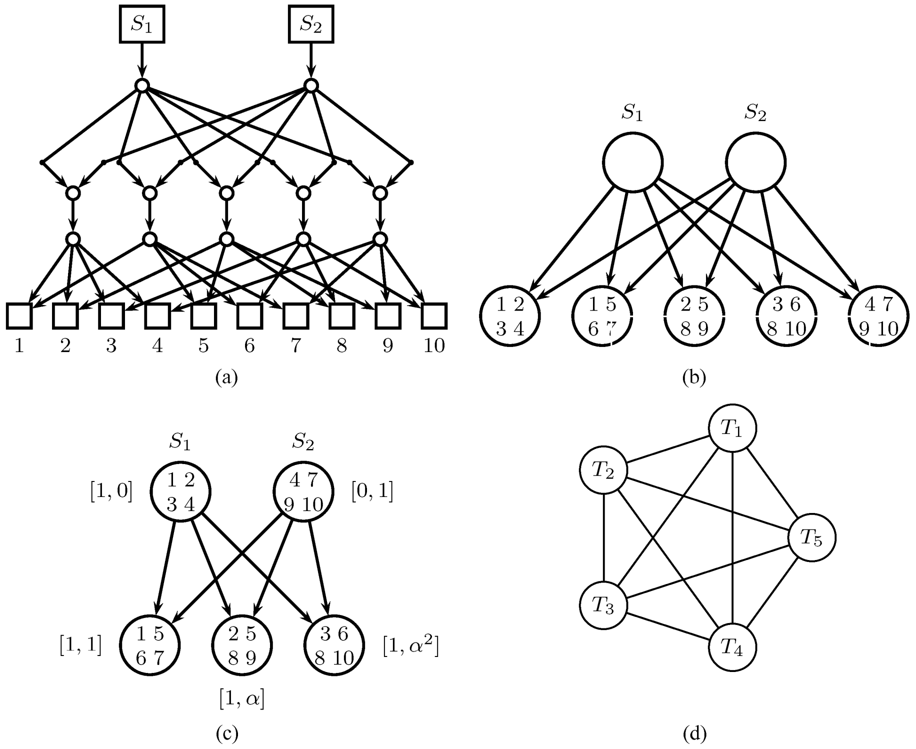

Fig. 10. (a) A network with two sources and 10 receivers; (b) a subtree decomposition of the network; (c) a minimal subtree decomposition of the network; (d) graph for coloring.

For each pair of subtrees that do not share a child, we introduce a common receiver. Therefore, we will have $N=k(k–1)/2–k+2$ receivers, and the corresponding graph Ω will be a complete graph with k nodes and $E(\Omega) = k(k-1)/2$ edges ($k(k-1)/2-k+2$ receiver edges and $k-2$ flow edges). The chromatic number of Ω is k and thus the required alphabet size is $q = k-1$. The bound is plotted in Fig. 11. □

It was previously shown that an alphabet of size $q = N$ is sufficient for networks with two sources and N receivers, and that there are networks with $N = k(k-1)/2$ receivers for which the alphabet of size $q = k-1$ is necessary [4], [14]. Both bounds can be derived by bounding the chromatic number of the graph Ω, as we explain next.

Since (by Theorem 4 in Section III-B) a minimal subtree graph with two sources and N receivers has at most $N+1$ vertices, the chromatic number of the corresponding graph Ω is smaller than or equal to $N+1$. Therefore, an alphabet of size N is sufficient for all networks with two sources. Moreover, we have shown that an alphabet of size N is sufficient for *decentralized* coding of all networks with two sources.

Now, consider the special case when the subtree graph Γ is bipartite with one set of vertices corresponding to the source subtrees and the other set of vertices corresponding to the coding subtrees, and for every set of two vertices there exists a receiver that observes them. That is, if the network has k subtrees, then there exist $k(k-1)/2$ receivers. The corresponding Ω is a complete graph with k vertices and thus cannot be colored with fewer than $q = k-1$ colors. Γ has $k(k-1)/2$ receivers, and requires an alphabet of size $q = k-1$. Fig. 10(a) depicts such a configuration for $k = 5$. We found that there are networks with

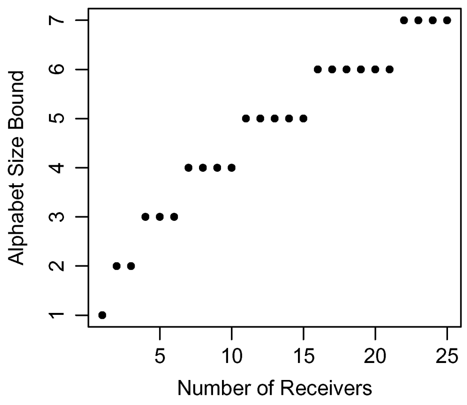

Fig. 11. Alphabet size upper bound $\lfloor\sqrt{2N-7/4}+1/2\rfloor$ as a function of the number of receivers N. Since alphabets are finite fields, their actual size is the prime or the power of prime closest but larger than the bound. In practice, one would most likely use the closest larger power of 2.

even fewer receivers for which the alphabet of size $q = k-1$ is necessary, and found that the minimum number of receivers such a network must have is $k(k-1)/2 - k + 2$.

B. Networks With h Sources and N Receivers

The required alphabet size for networks with h sources and N receivers whose subtree graph consists of $h-2$ source nodes with no children and a two-source configuration is clearly determined by the two-source subgraph. Therefore, based on the results of the previous section, a lower bound on the alphabet size a network with h sources and N receivers may require is $\lfloor\sqrt{2N-7/4}+1/2\rfloor$. The interesting question is whether this is also an upper bound. We next demonstrate two scenarios in which this is the case.

Definition 11: We say that a subtree graph is *q-critical* if

1) it requires an alphabet of size q (see Definition 10), and
2) the same subtree configuration with a different nonequivalent receiver-to-subtree assignment requires a smaller alphabet.

Definition 12: We say that a feasible assignment (Definition 2) of coding vectors to the (subset of) nodes of a subtree graph is *valid for receiver R* if under that assignment, receiver R observes h independent flows (and thus is able to decode all sources).

Theorem 8: Any nonempty subtree in a q-critical configuration with $h \geq 2$ sources and N receivers, contains at least q receivers.

Proof: Consider a nonempty coding subtree T. Since Γ_q is q-critical, there is at least one assignment of coding vectors over an alphabet of size $q-1$ to V_{Γ_q} that is valid for all receivers in the network but one of those observing T. Such an assignment can be obtained by moving one of the receivers in T to the source node corresponding to the source it observes in T and designing a valid network code for this subgraph. Any assignment of coding vectors over an alphabet of size $q-1$ to $V_{\Gamma_q} \setminus \{T\}$ that is valid for all receivers other then those observing T makes all feasible choices for $c(T)$ over the same alphabet invalid for at least one of the receivers observing T. Recall that $c(T)$ is feasible if it belongs to the linear span $W(T)$ of the vectors assigned to the parents of T. If T has k parents, there are $(q-1)^k - 1$ feasible (nonzero) choices for $c(T)$. If a feasible vector $c(T)$ is not valid for a receiver observing T, say R_1, then it must belong to the $(h-1)$-dimensional subspace spanned by the coding vectors assigned to the $h-1$ remaining subtrees observed by R_1, and therefore to the $(k-1)$-dimensional intersection between this subspace and $W(T)$. (That the dimension of this intersection is $(k-1)$ follows from the elementary linear algebra and the min-cut condition.) Therefore, $(q-1)^{k-1} - 1$ choices for $c(T)$ out $(q-1)^k - 1$ feasible will make the assignment not valid for a single receiver. The remaining $(q-1)^k - (q-1)^{k-1}$ feasible choices are therefore not valid for some of the remaining receivers. Thus, the smallest number of receivers $N(T)$ necessary to eliminate all the feasible points for $c(T)$ satisfies

$$(q-1)^k - N(T)[(q-1)^{k-1} - 1] \leq 0$$

giving $N(T) \geq q$. In the special case when the $(h-1)$-dimensional subspaces corresponding to different receivers contain the same $(k-2)$-dimensional subspace of $W(T)$, we have

$$(q-1)^k - N(T)[(q-1)^{k-1} - 1] + (N(T)-1)[(q-1)^{k-2} - 1] = 0$$

giving $N(T) = q$. □

By observing that each coding subtree can contain at most N receiver nodes, we immediately have the following corollary.

Corollary 2: For any network with h sources and N receivers, there exists a valid network code over an alphabet of size N.

This result was derived using different approaches in [3] and [4].

Theorem 9: If a removal of a nonempty coding subtree from a q-critical subtree graph with $h \geq 2$ sources and N receivers followed by the removal from the network of all the receivers observing that subtree results in a $(q-1)$-critical subtree graph, then $q \leq \lfloor \sqrt{2N - 7/4} + 1/2 \rfloor$.

Proof: Let N_q^h denote the number of receivers required in a q-critical h-source configuration. For the $h = 2$ case, we know that

$$N_q^2 - N_{q-1}^2 = q, \quad N_3^2 = 4.$$

For $h \geq q$, we have

$$N_q^h - N_{q-1}^h \geq q, \quad N_3^h = 4$$

where the inequality follows from Theorem 8, and $N_3^h = 4$ is found through exhaustive search. This implies that $N_q^h \geq N_q^2$, that is, the number of receives in a q-critical h-source configuration is greater than or equal to the number of receives in a q-critical two-source configuration. The claim then follows from Theorem 7, which relates N_q^2 and q.

To understand another scenario in which an alphabet of size $\lfloor \sqrt{2N - 7/4} + 1/2 \rfloor$ is sufficient, we consider a subtree graph Γ^h with h sources and N receivers together with its valid code C^h which labels the source nodes by the basis vectors $e_1, \ldots e_h$. Starting from Γ^h and C^h, the following procedure derives a two-source configuration Γ^2 with N receivers together with a valid code C^2 over the alphabet of C^h.

1) Consider a pair of linearly independent vectors vectors x_i and x_j in the h-dimensional vector space $\langle e_1, \ldots e_h \rangle$, and the projection operator $\mathcal{P}_{ij}$ that maps the h-dimensional space onto the plane $\langle x_i, x_j \rangle$ leaving x_i and x_j unchanged.
2) Consider one of the $\binom{h}{2}$ possible pairs of sources in Γ^h, say S_i and S_j. Find a code isomorphic to C_q^h by a change of basis that maps e_i to x_i and e_j to x_j, and then project the resulting coding vectors in the plane $\langle x_i, x_j \rangle$ by applying $\mathcal{P}_{ij}$.
3) Disconnect from the graph all the sources but S_i and S_j. For each receiver R, find two vertex-disjoint paths from the S_i and S_j source subtrees to two receiver nodes of R. Remove the remaining $h-2$ receiver nodes of R. Identify the associated minimal subtree graph Γ^2. The projected coding vectors obtained in step 2) that are associated with the nodes of Γ^2 constitute C^2 that is a valid code for Γ^2 over the alphabet of C.

Now Γ^2 may have a valid network code over a smaller alphabet in the plane $\langle x_i, x_j \rangle$, but we conjecture that if Γ^h requires an alphabet of size q, then there exists a pair of vectors x_i and x_j in step 1), a size-two set of source subtrees in step 2), and a choice of vertex-disjoint paths from the sources S_i and S_j to the receiver nodes in step 3), such that the resulting Γ^2 also requires an alphabet of size q in the plane $\langle x_i, x_j \rangle$. Clearly, if that is the case, based on our results for the networks with two sources, we have $q \leq \lfloor \sqrt{2N - 7/4} + 1/2 \rfloor$.

VI. Coding for Networks With Two Sources With Limited Alphabet Size

In Section V-A, we used the subtree decomposition to reduce the problem of designing a network code for a multicast configuration with two sources to the problem of vertex coloring. We now make another connection with coloring problems in combinatorics to study coding in networks with two sources where the processing complexity is a stronger constraint than the bandwidth, meaning that the system cannot support an alphabet size large enough to accommodate all users, but on the other hand, the min-cut toward each receiver is larger than the information rate that we would like to multicast. Below, we mention only a few relevant combinatorial results, but there are many more in the literature that are applicable in this and other practical network scenarios (see for example [23, Ch. 6]).

A. Min-Cut Alphabet-Size Tradeoff

When the min-cut toward each user is exactly equal to the number of sources, the bound in Theorem 7 gives the maximum alphabet size a network with N users may require. One would expect that, if the min-cut toward some or all of the users is greater than the number of sources, a smaller alphabet size would be sufficient. For the special case when the subtree graph is bipartite, we can show that this is indeed true by applying the following result. Consider a set of points X and a family $\mathcal{F}$ of subsets of X. A coloring of the points in X is legal if no element of $\mathcal{F}$ is monochromatic. If a family admits a legal coloring with q colors, then it is called k-colorable.

Theorem 10: (Erdös 1963) Let $\mathcal{F}$ be a family of sets each of size at least m. If $|\mathcal{F}| < k^{m-1}$, then $\mathcal{F}$ is k-colorable.

An algorithm for identifying a legal k-coloring can be found for example in [24].

In our case, X is the set of subtrees, m is the min-cut from the sources to each receiver, each element of $\mathcal{F}$ corresponds to the set of m subtrees observed by a receiver, and the set of colors are the points on the projective line. Therefore, $\mathcal{F}$ is a family of sets each of size m, and $|\mathcal{F}| = N$. Suppose that we can use an alphabet of size $k-1$ (which gives k colors). Note that each receiver can observe both sources if $\mathcal{F}$ is k-colorable, since that means that no set of subtrees observed by a receiver is monochromatic. By Theorem 10, this holds as long as

$$N < k^{m-1}.$$

The above inequality shows the tradeoff between the min-cut m to each user and the alphabet size $k-1$ required to accommodate N receivers. We expect a similar tradeoff in the case where the graph is not bipartite as well. However, Theorem 10 cannot be directly applied, because in this case there are additional constraints on coloring of the elements of X coming from the constraints that each child subtree has to be assigned a vector lying in the linear span of its parents' coding vectors.

B. Almost Good Codes

Consider again the case where the subtree graph is bipartite and the min cut to each receiver is m. We are interested in the number of receivers which will get only a single source when a code over an alphabet of size $k-1$ is used. We obtain a bound to this number by making use of the following result.

Theorem 11: [23, Ch. 9] For every family $\mathcal{F}$ whose all members have size exactly m, there exists a k-coloring of its points that colors at most $|\mathcal{F}|k^{1-m}$ of the sets of $\mathcal{F}$ monochromatically.

Thus, if we have $|\mathcal{F}| = N$ receivers, the min-cut to each receiver is m, and we use an alphabet of size $k-1$, then at most Nk^{1-m} receivers will get only one source. In other words, if the alphabet size is not large enough to accommodate all users, but on the other hand, the min-cut toward each receiver is larger than the information rate that we would like to multicast, at least $|F|(1-k^{1-m})$ receivers will still be able to successfully decode both sources.

C. Structural Information

Having some information about the structure of the underlying graph can be helpful both in bounding the alphabet size and in designing codes. For example, the required alphabet size, which is equal to the chromatic number of the graph Ω introduced in Section V-A, can be bounded in terms of the maximum degree of Ω. Although the chromatic number of Ω is smaller or equal to its maximum degree $\Delta(\Omega)$, we can find a legal coloring using at most $\Delta(\Omega)+1$ colors by simply applying the greedy algorithm (see, for example, [15, p. 98]).

Recall that the maximum degree of Ω is equal to the maximum number of receiver nodes inside any subtree of a bipartite configuration. Again, if the min-cut toward each receiver in the network is greater than the required, the code alphabet size may be reduced, as shown by the following result.

Theorem 12: (Erdös–Lovasz 1975) If every member of an m-uniform family intersects at most k^{m-3} other members, then the family is k-colorable.

Thus, if the min-cut to each receiver is m and every coding subtree is observed by at most $k^{m-3}+1$ receivers, then it is sufficient to use an alphabet of size $k-1$, irrespective of the number of receivers. The authors in [12] have derived alphabet size bounds in this direction.

VII. Connection With Convolutional Codes

In the development of the previous sections, we assumed that all nodes simultaneously receive all their inputs and produce their outputs. We now relax this zero-delay assumption and discuss a connection between network codes and convolutional codes over finite fields. To relax the zero delay assumption, we can associate a unit delay $\mathcal{D}$ either with each node of the line graph or only with coding points. Associating a unit delay with each edge of the network was 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 the number of memory elements m equal to the number of edges in G.

The convolutional code framework naturally takes delay into account, but at the cost of increased complexity for both encoding and decoding. We investigate methods to reduce the complexity requirements that differ from the standard in the theory of convolutional codes, because our design is subject to network topology constraints. We also discuss implementation using binary encoders, and propose a simplified version of the method in [2] to deal with cycles in the network.

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

$$\begin{aligned} \boldsymbol{s}_{j+1} &= \boldsymbol{A}\boldsymbol{s}_j + \boldsymbol{B}\boldsymbol{u}_j \\ \boldsymbol{y}_j &= \boldsymbol{C}\boldsymbol{s}_j + \boldsymbol{D}\boldsymbol{u}_j \end{aligned}$$

where $\boldsymbol{s}_j$ is the $m\times 1$ state vector, $\boldsymbol{y}_j$ is the $Nh\times 1$ output vector, $\boldsymbol{u}_j$ is the $h\times 1$ input vector, and $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$, and $\boldsymbol{D}$ are matrices with appropriate dimensions. The corresponding generator matrix $\boldsymbol{G}(\mathcal{D})$ is given by

$$\boldsymbol{G}(\mathcal{D}) = \boldsymbol{D} + \boldsymbol{C}(\mathcal{D}^{-1}\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{B} \tag{8}$$

where $\mathcal{D}$ is the indeterminate delay operator. The expression in (8) coincides with the transfer matrix $\boldsymbol{M}$ derived in [3], giving a different and simpler derivation of the same result.

Matrix $\boldsymbol{A}$ reflects the way the memory elements are connected. An element in matrix $\boldsymbol{A}$ 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 $\boldsymbol{A}$. Matrices $\boldsymbol{B}, \boldsymbol{C}$, and $\boldsymbol{D}$ are completely determined by the network configuration.

We observe that the dimensions of matrices $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$, and $\boldsymbol{D}$ depend upon the number of memory elements of the convolutional code, which in turn 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 in [9], 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 later in Fig. 14(a) and its corresponding convolutional code in Fig. 14(b). Unless otherwise stated, we will be considering the convolutional code associated with the subtree graph.

A. Structural Properties of Codes

We next examine the structure of matrices $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$, and $\boldsymbol{D}$. 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 describe below, 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 orders imposed by the different paths are 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 for consistency is that the underlying graph G is acyclic. Consider the two example networks shown in Fig. 12. Sources S_1 and S_2 use the cycle $ABCD$ to transmit information to receivers R_1 and R_2, respectively. In the network shown in Fig. 12(a), the paths from sources S_1 and S_2 impose consistent partial orders on the edges of the cycle. In the network shown in Fig. 12(b), for the path from source S_1, we have $AB > CD$, whereas for the path from source S_2, we have $CD > AB$.

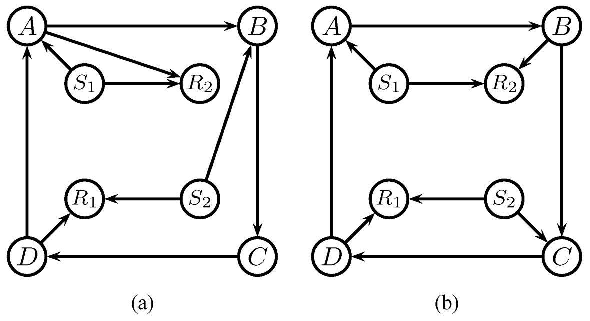

Fig. 12. Two networks with a cycle $ABCD$: (a) paths $(S_1, R_1) = S_1A \to AB \to BC \to CD \to DR_1$ and $(S_2, R_2) = S_2B \to BC \to CD \to DA \to AR_2$ impose consistent partial orders to the edges of the cycle; (b) paths $(S_1, R_1) = S_1A \to AB \to BC \to CD \to DR_1$ and $(S_2, R_2) = S_2C \to CD \to DA \to AB \to BR_2$ impose inconsistent partial orders to the edges of the cycle.

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

$$\boldsymbol{B} = \begin{bmatrix} \boldsymbol{I} \\ \boldsymbol{0} \end{bmatrix} \tag{9}$$

where $\boldsymbol{I}$ is the $h \times h$ identity matrix. The $N\ h \times m$ matrix $\boldsymbol{C}$ is a zero–one matrix of the form

$$\boldsymbol{C} = \begin{bmatrix} \boldsymbol{C}_1 \\ \vdots \\ \boldsymbol{C}_h \end{bmatrix} \tag{10}$$

where the $h \times m$ matrix $\boldsymbol{C}_i$ corresponds to receiver i. Each row of $\boldsymbol{C}_i$ corresponds to one of the subtrees whose state is observed by the receiver i. Thus, matrix $\boldsymbol{C}_i$ has exactly one 1 in each row and at most one 1 in each column. Matrix $\boldsymbol{D}$ is identically zero since we associate a memory element with each source subtree. Matrices $\boldsymbol{C}_i$ are completely determined by the subtree configuration. To design matrix $\boldsymbol{A}$, we can either use an exhaustive search, as is typically done to identify good convolutional codes, or we can use an adaptation of any of the algorithms for network code design, as for example proposed in [13].

The min-cut, max-flow requirement is equivalent to the condition that the $N\ h \times h$ transfer matrices

$$\boldsymbol{G}_i(\mathcal{D}) = \boldsymbol{C}_i(\mathcal{D}^{-1}\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{B}, \qquad i = 1\ldots, N \tag{11}$$

that corresponds to the N receivers have full rank, as described in [3].

2) Nonconsistent Partial Order: When the partial orders imposed by the different paths are not consistent, the corresponding subtree graph forms a recursive convolutional encoder, which can be analyzed by taking into account the feedback as proposed in [3]. Observe that an information source needs 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

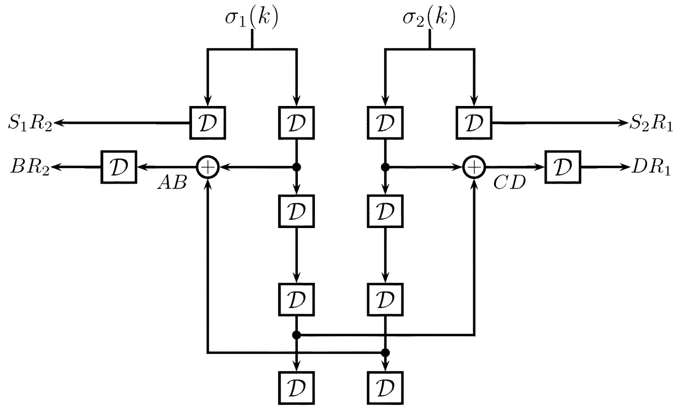

Fig. 13. Block representation of the cycle in Fig. 12(b).

in Fig. 12(b), and for simplicity assume that each edge corresponds to one memory element. Then the flows through the edges of the cycle are

$$\begin{aligned} AB&: \quad \sigma_1(k-1)+\sigma_2(k-3)\\ BC&: \quad \sigma_1(k-2)+\sigma_2(k-4)\\ CD&: \quad \sigma_1(k-3)+\sigma_2(k-1)\\ DA&: \quad \sigma_1(k-4)+\sigma_2(k-2) \end{aligned} \tag{12}$$

where $\sigma_i(k)$ is the symbol transmitted from source S_i at time k. Equation (12) can be easily implemented by employing a block of memory elements as shown in Fig. 13. Thus, we can still have a feedforward encoder by representing the cycle with a block of memory elements, and accordingly altering the structure of matrices $\boldsymbol{A}$ and $\boldsymbol{C}$.

This is a simplified version of the approach in [2]. Instead of starting with the original graph G, and "expand it in time" to create G^T as is described in [2], we use the graph $\gamma = \cup(S_i, R_j)$ (or the subtree graph) where vertices correspond to edges in the original graph G, and treat each cycle separately. Every vertex k in γ (edge in the original graph) corresponds to a state variable $\boldsymbol{s}_k$ in our notation. It is sufficient to express how the information that goes through each $\boldsymbol{s}_k$ evolves with time. To do that, we look at the I inputs (incoming edges) of the cycle (for example in Fig. 13, we have $I = 2$ inputs). Each input follows a path of length L_i (in Fig. 13, $L_1 = L_2 = 3$). For each input, we create L_i memory elements. Through each edge of the cycle flows a linear combination of a subset of these $\sum L_i$ memory elements. This subset is defined by the structure of the paths and the structure of the cycle, i.e., it is not selected by the code designer. The code designer can only select the coefficients for the linear combinations. Using this approach, we create an expanded matrix $\boldsymbol{A}$ that contains for every cycle an additional number of $\sum L_i$ memory elements. For example, in Fig. 13, we use six additional memory elements resulting a convolutional code with a total of 12 memory elements. For the same configuration, the approach in [2] for $T = 4$ (where T is a parameter described in [2]) would create a graph with $6T+2 = 26$ nodes, $4T + 6(T-1) + 6(T-1) = 52$ edges, and the corresponding convolutional code would have 52 memory elements. We can think of our approach as "expanding in time" like in [2], but only when necessary, and along specific paths.

B. 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 of the code whose generator matrix is given by (11) in the noiseless scenario. Thus, the complexity of decoding is proportional to the complexity of the trellis diagram.

One way to reduce the decoder complexity is to identify among all encoders that are subject to the constraints of a given topology, and that satisfy the min-cut max-flow 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 condition for each receiver. Equivalently, we need to identify a minimal subtree configuration that has the smallest possible number of subtrees. 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.

Another way to reduce the decoder complexity is to use for decoding the trellis associated with the minimal strictly equivalent encoder to $\boldsymbol{G}_i(\mathcal{D})$. Two codes are *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* [19].

Typically for convolutional encoders, we are interested in equivalent encoders that produce the same set of output sequences. Only recently, with the emergence of turbo codes where the mapping from input to output sequence affects the code's performance, the notion of strictly equivalent encoders has become important. Here we provide another example where this notion is useful. Note that in the conventional use of convolutional codes, there is no need to use a different encoder to encode and decode. In our case, we are restricted by the network configuration for the choice of the encoder $\boldsymbol{G}_i(\mathcal{D})$. Our observation is that, given these constraints, we still have some freedom at the receiver to optimize for decoding complexity.

C. Codes Over the 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 constitute 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, and thus, effectively it 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.

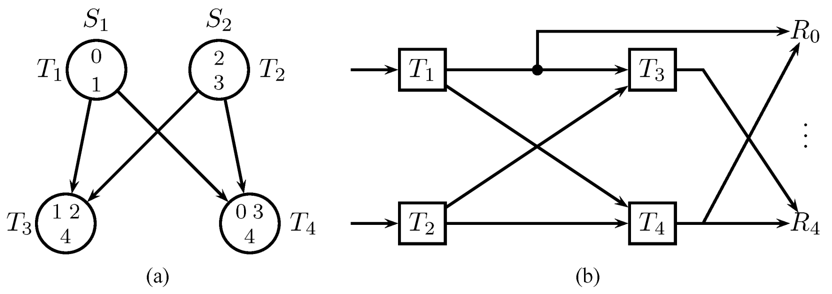

Fig. 14. Configuration with two sources and five receivers: (a) the subtree graph; (b) the corresponding convolutional encoder.

Example 7: Consider the configuration with $h = 2$ source subtrees and $m - h = 2$ coding subtrees depicted in Fig. 14(a). The corresponding convolutional encoder is depicted in Fig. 14(b). In this case, matrix $\boldsymbol{A}$ has the following form:

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{A}_{1,1} & \boldsymbol{0} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ * & * & 0 & 0 \\ * & * & 0 & 0 \end{bmatrix}$$

where $\boldsymbol{A}_{1,1}$ is of dimension 2×2, and "$*$" denotes a nonzero[1] element. Each receiver has its corresponding 2×2 generator matrix of the form

$$\boldsymbol{G}_i(\mathcal{D}) = \mathcal{D}\boldsymbol{C}_i(\boldsymbol{I}_{4\times 4} + \mathcal{D}\boldsymbol{A})\boldsymbol{B} = \mathcal{D}\boldsymbol{C}_i \underbrace{\begin{bmatrix} \boldsymbol{I}_{2\times 2} \\ \mathcal{D}\boldsymbol{A}_{1,1} \end{bmatrix}}_{\boldsymbol{G}}.$$

Matrix $\boldsymbol{G}$ is common for all receivers. A possible choice for matrix $\boldsymbol{A}$ over a finite field of size greater than or equal to three would be

$$\boldsymbol{A} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 2 & 0 & 0 \end{bmatrix}.$$

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. 14 would be 2 bits. Then each subtree can perform at time k the following binary operation:

$$\boldsymbol{M}_i \cdot [\sigma_1(k)\ \sigma_1(k-1)\ \sigma_2(k)\ \sigma_2(k-1)]^T$$

where

$$\boldsymbol{M}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \quad \boldsymbol{M}_2 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
$$\boldsymbol{M}_3 = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}, \quad \boldsymbol{M}_4 = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix}.$$

Receiver k will observe a matrix of the form

$$\boldsymbol{A}_k = \begin{bmatrix} \boldsymbol{M}_i \\ \boldsymbol{M}_j \end{bmatrix}, \qquad i \neq j$$

which has full rank. □

VIII. Conclusion

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 classify network configurations based on their equivalence from a coding point of view, to derive alphabet size bounds, to develop decentralized scalable algorithms, and draw connections with convolutional codes. 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.

[1]A zero coefficient would correspond to effectively removing an edge, which in a minimal configuration we cannot do without violating the min-cut condition.

Acknowledgment

The authors would like to thank Andrew Brown, Ellen Esichen, Dana Ron, and Amin Shokrollahi for stimulating discussions and comments on earlier versions of this manuscript. The authors would also like to thank the anonymous reviewers for detailed reviews that helped to greatly improve the clarity and presentation of this paper.

References

[1] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. Inf. Theory*, vol. 46, no. 4, pp. 1204–1216, Jul. 2000.
[2] S.-Y. R. Li, R. W. Yeung, and N. Cai, "Linear network coding," *IEEE Trans. Inf. Theory*, vol. 49, no. 2, pp. 371–381, Feb. 2003.
[3] R. Koetter and M. Médard, "Beyond routing: An algebraic approach to network coding," in *Proc. IEEE INFOCOM 2002*, vol. 1, New York, Jun. 2002, pp. 122–130.
[4] P. Sanders, S. Egner, and L. Tolhuizen, "Polynomial time algorithms for network information flow," in *Proc. 15th ACM Symp. Parallel Algorithms and Architectures*, San Diego, CA, Jun. 2003, pp. 286–294.
[5] S. Jaggi, P. Chou, and K. Jain, "Low complexity algebraic multicast network codes," in *Proc. IEEE Int. Symp. Information Theory*, Yokohama, Japan, Jun./Jul. 2003, p. 368.
[6] P. A. Chou, M. Effros, S. Egner, S. Jaggi, K. Jain, P. Sanders, and L. Tolhuizen, "Linear multicast network coding algorithms," *IEEE Trans. Inf. Theory*, submitted for publication.
[7] C. Chekuri, C. Fragouli, and E. Soljanin, "On average throughput benefits and alphabet size for network coding," *IEEE Trans. Inf. Theory*. Special Issue on Networking and Information Theory (Joint Special Issue of the *IEEE Trans. Inf. Theory* and the *IEEE/ACM Trans. Netw.*), submitted for publication.
[8] C. Fragouli, E. Soljanin, and A. Shokrollahi, "Network coding as a coloring problem," in *Proc. Conf. Information Sciences and Systems*. Princeton, NJ, Mar. 2004.
[9] C. Fragouli and E. Soljanin, "A connection between network coding and convolutional codes," in *Proc. IEEE Int. Conf. Communications*, vol. 2, Paris, France, Jun. 2004, pp. 661–666.
[10] ——, "Decentralized network coding," in *Proc. Information Theory Workshop*, San Antonio, TX, Oct. 2004.
[11] ——, "On average throughput benefits for network coding," *Proc. Allerton Conf. Communications, Control and Computing*, Sep./Oct. 2004.
[12] M. Feder, D. Ron, and A. Tavory, "Bounds on linear codes for network multicast," in *Electronic Colloquium on Computational Complexity*, 2003, Rep. 33.

[13] E. Erez and M. Feder, “Convolutional network codes,” in *Proc. IEEE Int. Symp. Information Theory* , Chicago, IL, Jun./Jul. 2004, p. 146.

[14] A. Rasala-Lehman and E. Lehman, “Complexity classification of network information flow problems,” in *Proc. Symp. Discrete Algorithms (SODA)*, New Orleans, LA, Jan. 2004, pp. 142–150.

[15] R. Diestel, *Graph Theory*, 2nd ed. Berlin, Germany: Springer-Verlag, 2000.

[16] A. H. Ali, J. W. P. Hirschfeld, and H. Kaneta, “On the size of arcs in projective spaces,” *IEEE Trans. Inf. Theory*, vol. 41, no. 5, pp. 1649–1656, Sep. 1995.

[17] J. A. Bondy and U. S. R. Murty, *Graph Theory with Applications*. Amsterdam, The Netherlands: North-Holland, 1979.

[18] R. Lidl and H. Niederreiter, *Finite Fields*. New York: Cambridge Univ. Press, 1997.

[19] H.-A. Loeliger, G. D. Forney, T. Mittelholzer, and M. D. Trott, “Minimality and observability of group systems,” *Linear Algebra and Its Applications*, vol. 205–206, pp. 937–963, Jul. 1994.

[20] T. Ho, R. Koetter, M. Médard, D. R. Karger, and M. Effros, “The benefits of coding over routing in a randomized setting,” in *Proc. IEEE Int. Symp. Information Theory*, Yokohama, Japan, Jun./Jul. 2003.

[21] T. Ho, D. R. Karger, M. Médard, and R. Koetter, “Network coding from a network flow perspective,” in *Proc. IEEE Int. Symp. Information Theory*, Yokohama, Japan, Jun./Jul. 2003.

[22] T. Ho, M. Mèdard, J. Shi, M. Effros, and D. R. Karger, “On randomized network coding,” in *Proc. Allerton Conf. Communications, Control and Computing*, Monticello, IL, Sep./Oct. 2004.

[23] S. Jukna, *Extremal Combinatorics*. Berlin, Germany: Springer-Verlag, 2001.

[24] U. Manber, *Introduction to Algorithms: A Creative Approach*. Reading, MA: Adison-Wesley, 1989.

[25] Y. Sagduyu and A. Ephremides, “Crosslayer design for distributed MAC and network coding in wireless ad hoc networks,” in *Proc. IEEE Int. Symp. Information Theory*, Adelaide, S.A., Australia, Sep. 2005, pp. 1863–1867.