Efficient Network Code Design for Cyclic Networks

Elona Erez, *Member, IEEE*, and Meir Feder, *Fellow, IEEE*

***Abstract*—This paper introduces an efficient polynomial-time code construction algorithm for *cyclic* networks, which achieves the optimal multicast rate. Until this work, no explicit capacity-achieving polynomial-time code construction for *cyclic* networks has been known. This new construction algorithm has the additional advantage that as sinks are added or removed from the network, it can modify the existing code in an efficient localized manner, which is beneficial also for acyclic networks. For decoding this code, a polynomial-time sequential decoder for convolutional network codes is also proposed.**

***Index Terms*—Convolutional codes, cyclic networks, multicast, network coding, sequential decoding.**

I. Introduction

MOST attention in the literature of network codes has been directed towards acyclic networks, which are represented by directed acyclic graphs. For these networks, it was proved that optimal multicast block network codes can achieve simultaneously the optimal min-cut bound [1]. These block codes are defined over a non-binary alphabet and are represented, for example, as elements of an algebraic field (actually the field size must be at least a square root of the number of terminals, see [2]–[4]). Furthermore, in [5] an efficient polynomial time algorithm has been proposed to construct such optimal block network codes (with field size which is equal to the number of terminals). However, most practical networks are cyclic, i.e., contain at least a single directed cycle in the network. For cyclic networks, in the original work [1] it was suggested to transform the cyclic network into an acyclic network using the idea of unrolling the network into a layered network. This approach has many drawbacks: it leads to time-variant schemes, it has high encoding and decoding complexities and it induces a large delay. It seems that, in general, using block codes for cyclic networks will result in large delay and complexity.

Analogously to standard coding theory, convolutional codes are an attractive alternative to block codes. Convolutional codes and related schemes have indeed been proposed, e.g., [1], [6]–[9], and [10]. While convolutional codes can be used in acyclic networks, it appears that one of their main advantages is their natural implementation in cyclic networks. A simple example of a convolutional network code for a cyclic graph was given in [1]. Actually, it was noted in [1] that convolutional codes can be beneficial for both acyclic and cyclic graphs as they may be simpler and require smaller delay and memory.

Cyclic networks not only call for convolutional codes, but may require that nodes will insert some delay. These issues were considered in [8], where it was shown that if each edge in the network has a delay, then there exists a time-invariant linear network code that achieves the optimal rate. However, no efficient construction algorithm was given there. A heuristic code construction for a linear time-invariant code was given in [7], but this construction is not specified explicitly. Similar drawbacks appear in the proposed approach of [9]. Convolutional network codes for cyclic networks were also considered in [11] and [12], where the coding coefficients are rational power series. It was shown in [11] that for networks with delay at each edge, if the coding coefficients are rational power series, then the code is causal and implementable, and furthermore, the existence of an optimal multicast code with rational power series coding coefficients was proved. It was also shown that these codes are decodable, and how the decoder can be found. However, the construction algorithm given in [11] is not guaranteed to be polynomial, since it involves an exhaustive search, and the maximal degree of the rational power series is not given.

In this paper, for the first time in network coding literature, we provide an explicit, efficient polynomial-time code construction of a multicast linear network code for *cyclic* networks that achieves the exact optimal rate. The maximal required degree of a polynomial coefficient of the code is $O(\log d)$, similarly to the degree in the acyclic case. Interestingly, the code construction algorithm does not require a delay in each node, but only that each cycle in the graph will contain a single delay.

As aforementioned, for acyclic networks there is an efficient linear time code construction, e.g., the construction in [5]. Our algorithm for cyclic networks assumes that the code designer has full knowledge of the network. This is similar to the setup in [5] for acyclic networks. This assumption is valid for moderately sized static networks. Moreover, developing a good centralized construction may lead also to better understanding of the decentralized and dynamic network code construction problem. We note that in the case of single source multicast for acyclic networks, simple decentralized random linear network coding approaches capacity very closely and with very low complexity (see, for example, [13]).

While our algorithm assumes full knowledge of the network, in a sense our algorithm is more localized than that in [5] for acyclic networks. Our construction algorithm can work for both acyclic and cyclic networks with one important advantage: our

Manuscript received November 26, 2008; revised November 07, 2009. Date of current version July 14, 2010. The material in this paper was presented in part at the IEEE International Symposium on Information Theory, Adelaide, Australia, September 2005.

E. Erez was with the Department of Electrical Engineering Systems, Fleischman Faculty of Engineering, Tel-Aviv University, Tel-Aviv 69978 Israel. She is now with the Department of Electrical Engineering, Yale University, New Haven, CT 06511 USA (e-mail: elona.erez@yale.edu).

M. Feder is with the Department of Electrical Engineering Systems, Fleischman Faculty of Engineering, Tel-Aviv University, Tel-Aviv 69978 Israel (e-mail: meir@eng.tau.ac.il).

Communicated by A. Nosratinia, Associate Editor for Communication Networks.

Digital Object Identifier 10.1109/TIT.2010.2050934

algorithm can modify the existing code in an efficient localized manner when sinks are added or removed, without the necessity to construct again the code for the entire network. We note that some information will still be required to be exchanged between nodes. However, the nodes that will be required to exchange information are limited only to the nodes that are in a certain flow between the source and the added (or removed) sink. Thus, the number of nodes involved in the process is generally significantly smaller for practical networks. In contrast, using the algorithm introduced in [5], there is no direct simple way to add and remove sinks—if we remove or add new sinks, the entire code, and generally even the field size, have to be changed.

Since with convolutional codes the sinks get a mix of the information sent during different time steps, it is impossible to decode convolutional codes in a "one shot" fashion as block codes. To overcome this, we also propose in this paper an efficient *sequential* decoder that is natural for convolutional codes and can be applied for cyclic networks. We analyze the decoding delay of the decoder. That algorithm further emphasizes the advantage of using convolutional codes for cyclic networks, since block decoders (for block codes) are cumbersome in cyclic networks due to the delays that are inserted to ensure stability, resulting in at the sink (decoder) a mix of the data sent at different time steps. An additional benefit of convolutional coding relative to random constructions is the delay-tuned characteristics of the sequential decoder. We note that we chose to call the decoder sequential due to its operation, but it is different than the classical sequential decoders (e.g., Fano decoder) of convolutional codes.

The convolutional codes we consider in this work are such that the coding coefficients are polynomial convolutional coefficients—in other words, the codes are feed-forward convolutional codes. This is a special case of the rational power series defined in [11]. We show that polynomial coefficients are sufficient in order to achieve the capacity for cyclic networks. Since the entire network is cyclic and so contains "feedback," the elements of the coding vectors become rational functions, or more specifically rational power series, as defined in [11]. The polynomials in this work are assumed to have binary coefficients. By generalizing [7], [14], [15], it was shown in [10] that for acyclic networks, binary convolutional codes suffice in order to achieve the capacity. Using a different approach, we will show that this is the case also for cyclic networks.

Finally we note that parts of the results presented in this paper appear in [16]–[19].

The paper is organized as follows. In Section II we introduce the network model and the notation. In Section III we introduce an efficient code construction for cyclic networks. In Section IV we propose an efficient sequential decoder. A discussion of the results obtained and possible directions for future research in Section V conclude the paper.

II. Definitions and Notation

A. General

Consider a cyclic, unit capacity directed network $G=(V,E)$ where parallel edges are allowed. In multicast there is a single source s and a set of d sinks $T=\{t_1,\ldots,t_d\}\subset V$. The size of the minimal individual min-cut between s and any of the sinks $t\in T$ is denoted by h. As in [5], when convenient we add a dummy source s' which is connected to source s with h edges $\{e_1,\ldots,e_h\}$. The dummy source s' is mentioned only when necessary for ease of description, but we can generally ignore it, since it can be thought of as an internal part of the source s itself. For our convenience, we assume that the network does not contain self-loops, which are edges from node v into the same node v. If there are self-loops in the network, we can always eliminate them and not use them for coding, without reducing the achievable rates of the network. For the directed edge $e=(u,v)$ from node u to node v, node u is defined as the tail of edge e and v as the head.

Similarly to [8], we define the directed *line graph* of $G=(V,E)$ as $L(\mathcal{V},\mathcal{E})$ with vertex set $\mathcal{V}=E\cup s\cup T$ and edge set $\mathcal{E}=\{(e',e)\in E^2:\text{head}(e')=\text{tail}(e)\}\cup\{(s,e):e \text{ leaving } s\}\cup\{(e,t_i):e \text{ entering } t_i, 1\le i\le d\}$. In other words, the nodes in L are the edges in G, the source s and the nodes in T. There is an edge (e',e) in L if in G the head of edge e' is the tail of edge e. There is an edge (s,e) in L if in G, e is an outgoing edge of s. There is an edge (e,t_i) in L if in G, e is an incoming edge of t_i. We denote nodes of $L(\mathcal{V},\mathcal{E})$ as $e\in\mathcal{V}$, and the edges as $(e',e)\in\mathcal{E}$. Clearly, if there are h edge-disjoint paths between source s and sink t_i in G, there are corresponding h node-disjoint paths in L. In the line graph L, for the directed edge $e_l=(e,e')$ from node e to node e', node e is defined as the tail of the edge e_l and e' as the head.

We define for each sink t_l the graph G_l which is the subgraph of G containing only the nodes and edges that participate in the flow of magnitude h from s to the sink t_l. We note that such a flow is general not unique. We therefore choose an arbitrary flow of size h, since our algorithm works regardless of the choice of a particular flow. The line graph of G_l is L_l. Denote by $\Gamma_I(v)$ and $\Gamma_O(v)$ the set of incoming and outgoing edges of node v, respectively. Similarly, denote by $\Gamma_I(e)$ and $\Gamma_O(e)$ the set of incoming edges of the tail of e and the set of outgoing edges of the head of e, respectively.

Unless otherwise specified, we assume that the topology of the network is completely known to the code designer and that the code is constructed only once, prior to the transmission. In this case of a known network, we further assume that G is given by

$$G=\bigcup_{l=1}^{d}G_l \tag{1}$$

and likewise L is given by

$$L=\bigcup_{l=1}^{d}L_l. \tag{2}$$

For the multicast case without edge failures this assumption is valid, since it follows from [1] that any edge in the original graph G that does not participate in any of the flows can be removed from the network without affecting the achievable optimal rate.

For an acyclic network, a topological order can be defined. In a topological order each node comes before all nodes to which it has outgoing edges.

Unless otherwise specified, the base of logarithms throughout the paper is 2.

B. Linear Network Codes

For linear network codes, any edge e has a global coding vector $\mathbf{v}(e)$ of dimension h associated with it. The symbol transmitted on an edge e, denoted by $y(e)$, is given by

$$y(e) = \sum_{e' \in \Gamma_I(v)} m(e', e) y(e') = \mathbf{v}(e)^T X. \tag{3}$$

For algebraic block network codes, the dummy source node s' gets h input symbols denoted by $X = (X_1, \ldots, X_h)$ from the field of the code $\mathcal{F}$. For an outgoing edge e_i, $1 \leq i \leq h$ of s' all the coordinates are zero except for the coordinate i, which is equal to X_i. For the rest of the edges $\mathbf{v}(e)$ satisfies the recursive relation

$$\mathbf{v}(e) = \sum_{e' \in \Gamma_I(e)} m(e', e) \mathbf{v}(e') \tag{4}$$

where $\mathbf{v}(e')$ is the *global vector* associated with the edge e', an incoming edge of e, and $m(e', e)$ is the *coding coefficient*.

C. Convolutional Network Codes

For convolutional network codes, let $F(D)$ denote the ring of polynomials over the binary field with variable D. The variable D represents a unit time shift. Since in this work most of the discussion on convolutional network codes is in terms of the line graph L, the following definitions for convolutional codes are also in terms of L. In the following, we use definitions and notations given in [11]. As shown in [11], in order for the code to be causal and implementable the local coding coefficients $m(e', e)$ are in the form $p(D)/(1 + Dq(D))$ where $p(D)$ and $q(D)$ are polynomials over the field F (the binary field in our formulation). These functions are termed in [11] as "rational power series". The set of these functions is an integral domain (a commutative ring with unity $1 \neq 0$, which does not contain any divisors of 0) denoted by $F\langle D \rangle$. The set $F\langle D \rangle$ is a sub-domain of $F[[D]]$, which was defined in [11] as the integral domain of all power series over F.

In our scheme, we take the local coding coefficients $m(e', e)$ to be polynomials from $F(D)$, and therefore they are also in $F\langle D \rangle$. Node $e \in L$ is associated with an h-dimensional global vector $\mathbf{v}(e)$, where each element of the vector is an element of $F\langle D \rangle$.

The source node s gets h input binary streams $x_1(n), \ldots, x_h(n)$. The source starts transmitting symbols at $n = 0$. The power series in variable D of the input stream $x_j(n)$ is

$$X_j(D) = \sum_{n=0}^{\infty} x_j(n) D^n. \tag{5}$$

The global coding vector associated with node e is given by

$$\mathbf{v}(e) = \sum_{e' \in \Gamma_I(e)} m(e', e) \mathbf{v}(e') \tag{6}$$

where $\mathbf{v}(e')$ is the global coding vector associated with node e', an incoming node of e, and $m(e', e) \in F(D)$ is the coding coefficient of edge (e', e). In (6), additions and multiplications are regular for rational power series in $F\langle D \rangle$. In order for the recursion in (6) to be defined, the h global coding vectors between the dummy source s' and the source s form the natural basis of the vector space $GF^h(2)$.

In the original network G, assume $y_e(n)$ is the symbol transmitted on node e at time instant n and consider the power series in variable D

$$Y_e(D) = \sum_{n=0}^{\infty} y_e(n) D^n \tag{7}$$

which is given by

$$Y_e(D) = \sum_{e' \in \Gamma_I(v)} m(e', e) Y_{e'}(D) \tag{8}$$

and so

$$Y_e(D) = \mathbf{v}(e)^T \mathbf{x}(D) \tag{9}$$

where $\mathbf{x}(D) = (X_1(D), \ldots, X_h(D))^T$. In the line graph L, the corresponding node e is associated with the symbol $y_e(n)$ and the power series $Y_e(D)$. We note that these definitions correspond to binary convolutional codes, since all the coding coefficients and the h input streams are binary.

In multicast, the code construction has to determine the polynomial coefficients $m(e', e)$ such that all sinks can reconstruct the original information from the symbols reaching them. It is assumed that after the code design, this set of global coding vectors at $\Gamma_I(t)$ is made known to the decoder at sink t_l. It will be shown in Section IV how reconstruction of the original information from the received symbols can be achieved in polynomial time. As we show in Section IV, for each sink there are h nodes incoming into the sink, denoted by $\{e_1, \ldots, e_h\}$, whose symbols are sufficient for decoding. Denote the vector of the symbols carried by these nodes as

$$\mathbf{y}(D) = (Y_{e_1}(D), \ldots, Y_{e_h}(D))^T. \tag{10}$$

For sink t_l, $l = 1 \ldots, d$, denote by $A_l(D)$ the $h \times h$ matrix whose rows are the vectors $\mathbf{v}^T(e)$, $e \in \Gamma_l(t_l)$. Then, the vector $\mathbf{y}(D)$ is given by the matrix relation

$$\mathbf{y}(D) = A_l(D) \mathbf{x}(D). \tag{11}$$

Denote by $F[D]$ the field of rational functions over the binary field with variable D. The vector space of dimension h over the field $F[D]$ is denoted by $F[D]^h$.

III. Efficient Code Construction for Cyclic Networks

Unless otherwise specified, we consider in the rest of this section the line graph $L(\mathcal{V}, \mathcal{E})$. In this section, we assume that there is a dummy source s'.

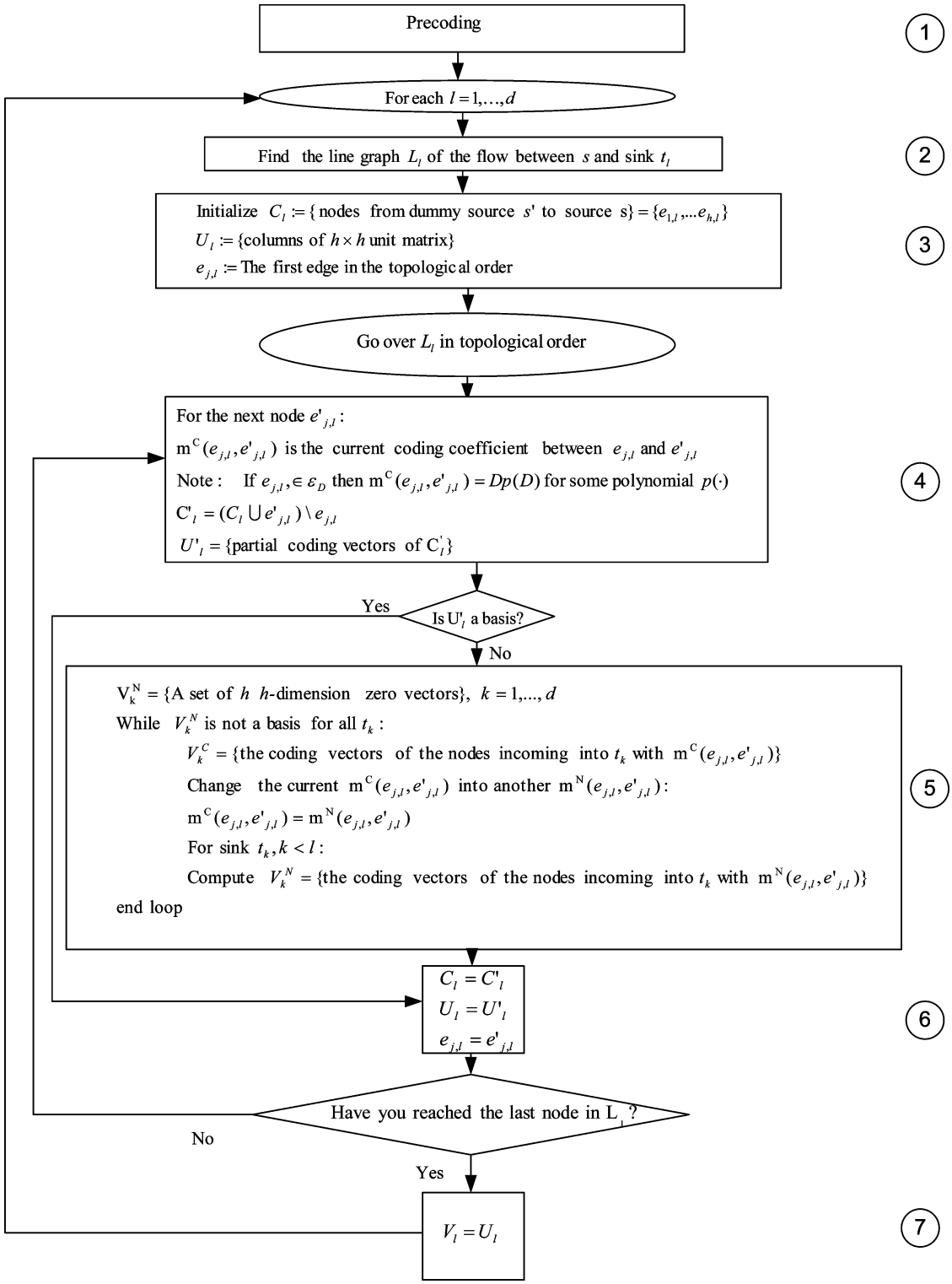

Fig. 1. Flow chart of the construction algorithm.

In the following we give an explicit polynomial time code construction algorithm of a multicast convolutional network code for cyclic networks that achieves the min-cut optimal rate. We have introduced the code construction in [17]. A flow chart of the entire algorithm is given in Fig. 1. We note that it is straightforward to modify this algorithm so that it can be applied to construct, in polynomial time, block network codes for cyclic networks.

A. Realizability of the Code

Network codes for networks with cycles have to introduce delay, since otherwise the code will not be stable and causal. This delay element has to be inserted in order to avoid the problem of "racing condition," which is the situation of a node receiving new information before stabilizing.

In [8] and [11], a delay was introduced at each node of L. This kind of network is known as "unit-delay network." In our for-

mulation, in unit delay networks, we assume in this section that this delay is incorporated into the coding coefficient $m(e,e')$. That is, we assume that each local coding coefficient has D as a multiplicative factor (i.e., it is divisible by D) and do not explicitly write this delay in our formulations. Therefore, the relations in Section II-C are still valid, only it is assumed that $m(e,e')$ contains the factor D. Note, however, that the constructed code in Section III-C may not necessarily generate a unit delay on each edge in the network. As we will see in Section III-C, it is a sufficient condition for the code construction that each directed cycle will introduce a delay.

To ensure that the code is realizable, it was shown in [11] that the global coding vectors are in $F\langle D\rangle$ by obtaining a closed form solution. For completeness, we briefly repeat here this derivation from [11]. Define for the line graph L the $|E|\times|E|$ matrix $C(D)$ given by

$$C_{i,j}(D)=\begin{cases} m(e_i,e_j) & (e_i,e_j)\in L \\ 0 & \text{otherwise}\end{cases} \tag{12}$$

where, as we recall, (e_i,e_j) are the edges of the line graph L. Denote by $H_S(D)$ an $h\times|E|$ matrix. For the first h columns element (i,j) is the coefficient $m(e_i',e_j)$, where e_i' is the i'th node of L between the dummy source s' and s and e_j is the j'th node outgoing from source s. All the rest of the elements of $H_S(D)$ are zero.

Denote by $F_M(D)$ the matrix which contains in the i'th column the global coding vector $\mathbf{v}(e_i)$ of e_i assuming that all the nodes in L are enumerated according to some order (not including the dummy source and the nodes between the dummy source s' and source s). Equation(6), as well as the initial conditions for the recursion (the global coding vectors of the nodes between s and s' form the natural basis), can be written now in the matrix form

$$F_M(D)=F_M(D)C(D)+H_S(D). \tag{13}$$

Recall that all of elements of $C(D)$ have D as a multiplicative factor. The determinant of the matrix $I-C(D)$ is of the form $1+Dq(D)$ where $q(D)\in F\langle D\rangle$ and where I is the $|E|\times|E|$ identity matrix. Therefore $\det(I-C(D))$ is invertible inside $F\langle D\rangle$ and the unique solution for $F_M(D)$ is

$$F_M(D)=\frac{1}{\det(I-C(D))}H_S(D)Adj(D) \tag{14}$$

where $Adj(D)$ is the adjoint matrix of $I-C(D)$. Therefore $F_M(D)$ is a matrix over $F\langle D\rangle$ and (14) is its closed form expression.

The existence of a multicast code that achieves the optimal rate with rational power series coding coefficients was also proved in [11] and [12]. For our purposes, we will show the existence of a multicast code that achieves the optimal rate with polynomial coding coefficients by providing the code construction in the following. The existence proof in [11] and [12] is more general than our proof in the sense that it applies to any large enough collection of power series coefficients, which includes a collection of polynomials, as in our case. However, the construction algorithm given in [11] and [12] is not guaranteed to be polynomial, since it involves an exhaustive search, and the maximal degree of the rational power series is not given.

B. Precoding

In the following, we show that it is not necessary for each node to introduce a delay, but there should be only a single delay for each directed cycle in the network. In this section we no longer assume as in Section III-A that all the elements of the matrix $C(D)$ in (12) are divisible by D. We denote the set of nodes in L that introduce delay by $\mathcal{E}_D$. The corresponding edges in G are denoted by E_D. In this case the coefficient $m(e_i,e_j)$ in matrix $C(D)$ contains common factor D if node e_i introduces a delay, i.e., $e_i\in\mathcal{E}_D$. The set $\mathcal{E}_D$ is chosen such that if we remove from the network all the nodes with delay, then the resulting network would be acyclic. Since the coding coefficients of the nodes in $\mathcal{E}_D$ are divisible by D, setting $D=0$ sets the coding coefficients of these nodes to zero and these nodes are effectively removed. That is, setting $D=0$ results in an effectively acyclic network. Since for $D=0$ the network is effectively acyclic, it follows that we can always arrange the remaining nodes of L in topological order such that $C(0)$ would be upper triangular when setting $D=0$. Note that the elements in the diagonal of $C(D)$ are zero because we assume there are no self-loops in L. It follows that when $D=0$ the matrix $I-C(D)$ is upper triangular and on its diagonal all elements are equal to 1. Therefore the determinant of $I-C(D)$ when $D=0$ is 1. Since the elements of the matrix $C(D)$ are in $F\langle D\rangle$, and the determinant is calculated using only multiplications and additions, $\det(I-C(D))$ is also in $F\langle D\rangle$. Since $\det(I-C(D))$ is equal to 1 for $D=0$ it follows that $\det(I-C(D))$ can be written in the form $1+Dq(D)$ where $q(D)\in F\langle D\rangle$. Therefore $\det(I-C(D))$ is invertible inside $F\langle D\rangle$ and the unique solution for $F_M(D)$ is in $F\langle D\rangle$ and is given again by (14). We note that this observation is not actually required in order to show the correctness of the construction in Section III-C. We show in Section III-C independently and by construction that a single delay in each directed cycle is sufficient in order to ensure the feasibility of the code.

The situation of a single delay in each cycle is defined in [20] as asynchronous transmission. It is shown that the min-cut is an upper bound on the possible rate also for asynchronous transmission, as for synchronous transmission. Since in this work we show that this bound is always achievable, it is also tight. It the following we introduce the precoding, which is designed to find the set $\mathcal{E}_D$.

We have to choose a set of edges E_D in G, such that if we eliminate them from the network G there will be no directed cycles. The nodes corresponding to E_D in $L(\mathcal{V},\mathcal{E})$ are denoted by $\mathcal{E}_D$. For edges outgoing from nodes not in $\mathcal{E}_D$ we will draw in later stages of the algorithm the coding coefficients among polynomials with degree at most M, for some M we will later determine. For edges outgoing from nodes in $\mathcal{E}_D$ we will draw in later stages the coding coefficients among polynomials with degree at most M, and multiply the result by D. Therefore edges outgoing from nodes in $\mathcal{E}_D$ always introduce at least a single delay. In order to minimize the number of delay elements, it is desired to minimize $|\mathcal{E}_D|$, or equivalently $|E_D|$ in the original network G. Finding the minimal set E_D is essentially the known, long

standing problem of finding the minimal arc feedback set. This problem is NP-hard [21]. The best known approximation algorithm with polynomial complexity achieves performance ratio $O(\log|V|\log\log|V|)$ [22]. For our purposes, we can use approximate solutions and insert enough delays in the cycles.

We note that after the precoding stage, which inserted the delays into the network, the coding coefficients themselves $m(e,e')$ are not required to be necessarily convolutional. They can be taken from any algebraic field or a ring. However, because of the delays inserted into the network the resulting code seen by the decoder at the sink can always be interpreted as a convolutional code.

C. Code Construction

We begin by presenting the flow chart of the algorithm shown in Fig. 1. In this section we will explain step by step the parts of the flow chart and the justification for them. The different blocks in the figure are enumerated from 1 to 7, and in the following we will address these blocks according to this enumeration. An example network that illustrates the algorithm will follow the description in Section III-D.

The first step of the construction in block 1 is the precoding stage, which selects a node subset of the line graph denoted by $\mathcal{E}_D$, such that any cycle in the line graph contains at least one node in $\mathcal{E}_D$. For the following steps of the code construction, we assign the coding coefficients $m(e,e')$. If $e \in \mathcal{E}_D$, we choose $m(e,e')$ to be of the form $Dp(D)$ where $p(D)$ is a binary polynomial in variable D.

It is assumed that prior to the code construction, all the coding coefficients in the network are set to zero. The code construction algorithm goes in steps over the sinks. In block 2, in the l'th step of the algorithm we consider the subgraph L_l, which is some maximal flow from source s to sink t_l, which maintains the min-cut condition.

In block 3 the algorithm initializes the sets $\mathcal{C}_l$ and U_l. The algorithm maintains a list of h nodes $\mathcal{C}_l$, each belongs to a different path in L_l and this list includes the most recently updated nodes. Initially, the set $\mathcal{C}_l = \{e_1^0,\ldots,e_h^0\}$ is the set of nodes with edges incoming from the dummy source s'. Each node in $\mathcal{C}_l$ has a vector associated with it in the set U_l. We will specify this set of vectors in the following. Initially, the vectors in U_l are the natural basis. In the following, we show that at each stage of the code construction, the data can be decoded from the set of vectors U_l. This property is maintained also when the algorithm terminates.

The algorithm goes through the nodes in L_l in topological order. It is possible to define a topological order for L_l since L_l is acyclic as it can be decomposed into h paths and cycles can be eliminated from each path. In block 4 the algorithm proceeds to the next node $e'_{j,l}$ in the topological order in L_l. The j'th path in flow L_l from s to t_l is denoted by $P_{j,l}$, $j=1,\ldots,h$. We note that the index j simply means that the next node in the topological order, which is denoted by $e'_{j,l}$, belongs to path $P_{j,l}$. The topological order does not necessarily follow one path $P_{j,l}$ after the other, but can be any topological order. We use the index j only in order to facilitate our notation in the remainder of the description of the construction algorithm and in the proofs of the theorems that follow.

For each node $e \in L_l$ the algorithm will determine a polynomial coding coefficient $m(e,e') \in F(D)$ for the edge (e,e') that connects this node to the node e' that follows it in the flow L_l. At the beginning of the l'th step some of the edges may already have nonzero coding coefficients assigned to them at the previous steps $\{k : 1 \le k < l\}$. The coefficients in L_l are updated during the l'th step, in a manner described next.

In block 4, node $e_{j,l}$ is the current node in $\mathcal{C}_l$ in the jth path $P_{j,l}$. Denote by $p_{j,l}$ the subset of path $P_{j,l}$ consisting of all nodes following the node $e_{j,l} \in \mathcal{C}_l$ in $P_{j,l}$ (not including $e_{j,l}$). We denote by $c_{j,l}$ the set of coding coefficients of edges with tail in $p_{j,l}$ and head in $L \setminus p_{j,l}$ (i.e., all the nodes in L that are not in $p_{j,l}$). We define r_l as the union of these sets of coefficients

$$r_l = \cup_{1\le j\le h} c_{j,l}. \tag{15}$$

In Fig. 2(a) a schematic illustration of $\mathcal{C}_l$ is given, in which the coefficients in r_l are specified and set to zero (the original network G is shown). The thick arrows are of edges in $\mathcal{C}_l$. Edges in the path $p_{j,l}$ are marked by unfilled arrows. The dashed arrows are of edges outgoing from edges in $p_{j,l}$. Fig. 2(b) depicts the same scenario for the corresponding line graph L.

Observe that all the coefficients in r_l correspond to edges exiting from nodes that come after the nodes in $\mathcal{C}_l$ in the topological order. We define for each node $e \in \mathcal{C}_l$ a "partial" coding vector $\mathbf{u}(e)$ in addition to the global coding vector $\mathbf{v}(e)$:

Definition 1: The global coding vector $\mathbf{v}(e)$ maintains the relation (6) when all the coding coefficients in the network are set to their current value. The partial coding vector $\mathbf{u}(e)$ maintains the same relation (6) (with $\mathbf{v}(e)$, $\mathbf{v}(e')$ replaced by $\mathbf{u}(e)$, $\mathbf{u}(e')$, respectively) when all the coding coefficients in r_l are set to zero.

In Fig. 2 the coefficients in r_l are shown to be set to zero for the definition of $\mathbf{u}(e)$. The distinction between $\mathbf{v}(e)$ and $\mathbf{u}(e)$ stems from the fact that the coding coefficients in r_l might not currently be zero, since they were already updated in previous steps, performed for other sinks $t_k : 1 \le k < l$. Let $V_l = \{\mathbf{v}(e) : e \in \mathcal{C}_l\}$ and $U_l = \{\mathbf{u}(e) : e \in \mathcal{C}_l\}$. Initially, the vectors in U_l are set to be the columns of the $h \times h$ unit matrix.

The code construction algorithm for acyclic networks in [5] ensures that the set V_l is a basis throughout its stages. In the algorithm in [5] for acyclic networks, each coding coefficient was determined once, jointly for all sinks. However, in the case of this algorithm which proceeds in steps, one sink after the other, the condition on V_l is not sufficient to ensure that the network code is decodable at each sink. The reason is that there are coefficients that affect elements of V_l, whose value may change later in the algorithm. When the topological order is not well defined (as for cyclic networks) or when the construction scheme does not follow a topological order (when designing the acyclic network code in a flow-by-flow manner) there may not be a valid choice of a coding coefficient for a particular edge. This problem occurs if we focus on the global coding vectors for the code construction. This is the major challenge for designing a code for cyclic networks. However, as we will show in the following, when we focus for the construction on the partial coding vectors instead of the global coding vectors, it is guaranteed that we can still design the network code in an edge-by-edge fashion even

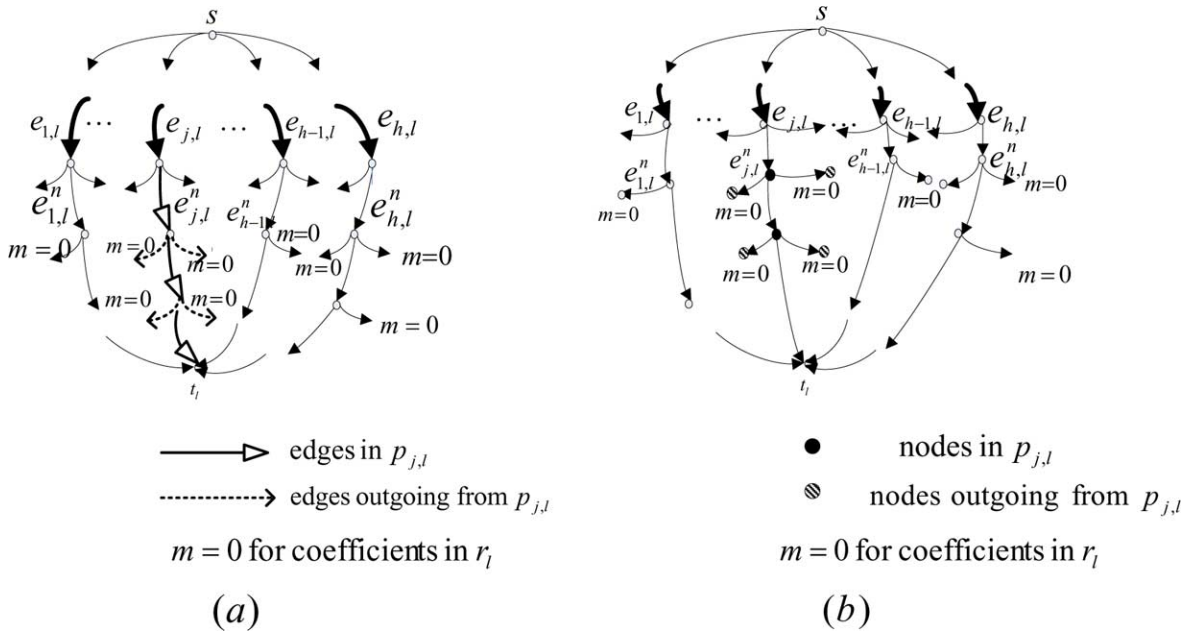

Fig. 2. A schematic illustration of C_l and $p_{j,l}$. (a) OriginalNetwork G. (b) Line Graph L.

even for cyclic networks. This fact enables us to design an efficient code construction method for cyclic networks. An example that clarifies this point appears in Section III-E.We note that we chose to call the decoder sequential due to its operation, but it is different than the classical sequential decoders (e.g., Fano decoder) of convolutional codes

We will show in this section that a sufficient condition to ensure that the network code is decodable at the sink is to require that the set U_l will span $F[D]^h$. As will be seen below, the algorithm can progress and maintain U_l as a basis up to the end of the l'th step, when $\mathcal{C}_l$ becomes the set of the incoming nodes of sink t_l. In this event the sets $p_{j,l}$, $1 \leq j \leq h$ are empty and accordingly r_l is empty, and so $V_l = U_l$. Thus, at the end of the l'th step the condition on U_l is equivalent to the condition in [5] for acyclic networks and the original input can be reconstructed from the information carried by the incoming edges of t_l using the sequential decoder (as will be shown in Section IV).

Going back to the flow chart in block 4, when the algorithm reaches node $e_{i,l}$, it replaces $e_{i,l}$ with a new node $e'_{i,l}$ that follows it in $P_{i,l}$, and generates a new list $\mathcal{C}'_l = \mathcal{C}_l \cup e'_{i,l} \setminus e_{i,l}$. The subset of path i consisting of all nodes following nodes $e'_{i,l}$ in path $P_{i,l}$ is updated

$$p'_{i,l} = p_{i,l} \setminus e'_{i,l}. \tag{16}$$

Accordingly, $c'_{i,l}$ is the set of coding coefficients of edges with the tail in $p'_{i,l}$ and the head in $L \setminus p'_{i,l}$ and r'_l is also changed to

$$r'_l = \cup_{1 \leq j \leq h, j \neq i} c_{j,l} \cup c'_{i,l}. \tag{17}$$

There is a new set of partial coding vectors

$$U'_l = \left\{\mathbf{u}'(e_{1,l}), \ldots, \mathbf{u}'\left(e'_{i,l}\right), \ldots, \mathbf{u}'(e_{h,l})\right\}$$

, defined when the coefficients in r'_l are set to zero.

We note that in any step of the algorithm when the partial coding vectors or the global coding vectors are required to be computed, we use (14) for computing the vectors. We substitute in it the corresponding values, either for the partial vectors or the global vectors.

Before block 5, the algorithm has to determine if the set U'_l is a basis. Let $m^C\left(e_{i,l}, e'_{i,l}\right)$ be the current coding coefficient between $e_{i,l}$ and the new node $e'_{i,l}$. If U'_l is a basis with coefficient $m^C\left(e_{i,l}, e'_{i,l}\right)$ then the coefficient need not be updated and the algorithm goes directly to block 6. If U'_l is not a basis then the coding coefficient needs to be updated and the algorithm performs block 5.

Let $m^N\left(e_{i,l}, e'_{i,l}\right)$ be the new coefficient determined at the current step. If U'_l is not a basis, we have to change the current coefficient $m^C\left(e_{i,l}, e'_{i,l}\right)$ into another coefficient $m^N\left(e_{i,l}, e'_{i,l}\right)$. Consider the following theorem we prove in Appendix A:

Theorem 1: Suppose that with the coefficient $m^C\left(e_{i,l}, e'_{i,l}\right)$ the set U'_l is not a basis. Then by changing it to any other value $m^N\left(e_{i,l}, e'_{i,l}\right)$, the set U'_l will become a basis.

Changing $m^C\left(e_{i,l}, e'_{i,l}\right)$ to an arbitrary value may affect the sinks treated in the previous steps of the algorithm. Thus, if the coding coefficient has to be replaced (i.e., when U'_l is not a basis) before replacing it into a new value we need to analyze its effect on the other sinks. Specifically, let $\mathcal{C}_k$ be the set of incoming nodes of the sink t_k, $k < l$. We have the following theorem.

Theorem 2: Denote by $V_k^C = \left\{\mathbf{v}^C(e_{1,k}), \ldots, \mathbf{v}^C(e_{h,k})\right\}$, $e_{j,k} \in \mathcal{C}_k$, the set of global coding vectors of the incoming nodes of t_k before changing $m^C\left(e_{i,l}, e'_{i,l}\right)$ to $m^N\left(e_{i,l}, e'_{i,l}\right)$. If V_k^C is a basis, then after the replacement at most a single value of

$m^N\left(e_{i,l}, e'_{i,l}\right)$ will cause the new set of global coding vectors $V_k^N = \{\mathbf{v}^N(e_{1,k}), \ldots, \mathbf{v}^N(e_{h,k})\}$ not to be a basis.

The proofs of the theorem appear in the Appendix. Based on these theorems, the procedure for replacing $m^C\left(e_{i,l}, e'_{i,l}\right)$ is as follows. In block 5, if $m^C\left(e_{i,l}, e'_{i,l}\right)$ must be replaced i.e., U'_l is not a basis, we pick a new value $m^N\left(e_{i,l}, e'_{i,l}\right)$ according to some enumeration. According to Theorem 1 for any other $m^N\left(e_{i,l}, e'_{i,l}\right)$, the set U'_l will be a basis. The old value of the global coding vectors of the incoming nodes of t_k with coefficient $m^C\left(e_{i,l}, e'_{i,l}\right)$ was the set V_k^C. We can assume by induction that the set V_k^C was a basis, since our assumption is that the condition was satisfied prior to the current l'th step (to initialize the induction, the assumption is trivially valid since for $l = 1$ the set $k < l$, $k = 1, \ldots, d$ is empty).

In block 5, we compute the new set of global coding vectors V_k^N with the new coding coefficient $m^N\left(e_{i,l}, e'_{i,l}\right)$. We check if the independence condition is satisfied for all sinks. That is, if V_k^N is a basis for all sinks, $k < l$. If the condition is not satisfied for all sinks, we repeat the loop and pick the next coefficient $m^N\left(e_{i,l}, e'_{i,l}\right)$ in the enumeration. Since by Theorem 2 for each sink at most a single choice of $m^N\left(e_{i,l}, e'_{i,l}\right)$ is bad, if we have more than d coefficients to choose from, we are guaranteed to have at least a single choice which is good for all previous sinks simultaneously. If the condition is satisfied for all sinks, then block 5 is terminated. For the d coefficients we can take, for example, all the polynomials with minimal degree 0 and maximal degree $\lceil \log(d) \rceil$. For the edges in the set $\mathcal{E}_D$, determined in the precoding stage, we multiply the chosen coefficient $m\left(e_{i,l}, e'_{i,l}\right)$ by D.

In block 6, the C_l, U_l and $e_{j,l}$ are updated to their new value. After block 6, the algorithm checks whether we have already reached the last edge in the topological order of L_l. If the last node of the topological order has already been reached, then the algorithm proceeds to the block 7. The l'th step of the algorithm continues until it reaches the sink t_l, when $V_l = U_l$ as shown in block 7. If the last node in L_l has not been reached yet, the algorithm continues to block 4. The algorithm terminates when it goes over all d sinks.

At the end of the algorithm we are guaranteed that for each sink t_l the set V_l is a basis, according to our construction. According to the analysis in [11], which we have repeated in Section III-A, the elements of V_l will be defined in $F\langle D\rangle$.

The authors of [11] define the decodability of a code, which we will repeat in Definition 2, in Section IV. In [11] it is shown that if the elements of V_l are in $F\langle D\rangle$, then the code is decodable. Since the elements of V_l in our construction are defined in $F\langle D\rangle$, it follows that our code is decodable. We will further show in Section IV how the sequential decoder can be used in order to efficiently decode the source data.

D. Code Construction for Example Network

In this section we illustrate the code construction for a simple example network. The example network is depicted in Fig. 3(a), with its line graph L in Fig. 3(b). For simplicity, in the line graph

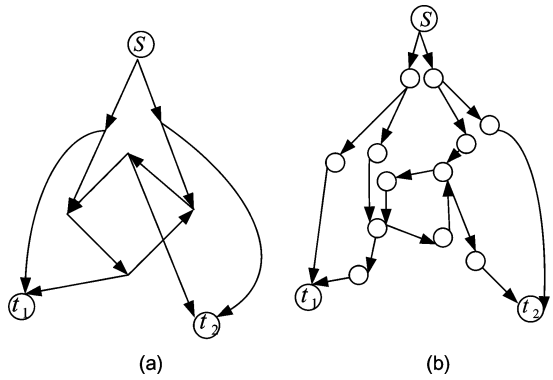

Fig. 3. Example network.

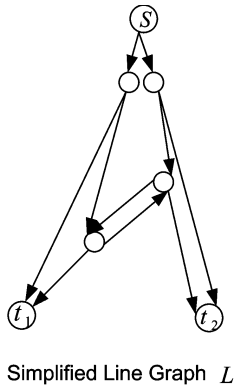

Fig. 4. Example network.

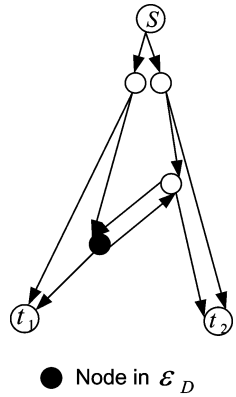

Fig. 5. Example network.

L in Fig. 4, nodes that have a single input and a single output were omitted, since they simply forward the symbol they receive to the next node. In Fig. 5 the node chosen for $\mathcal{E}_D$ is shown as a filled circle. Fig. 6 shows L_1 and L_2 in dashed arrows. The topological order of the nodes in each flow is also shown. Note that the topological order is different for each flow.

In Fig. 7(a) the coding stage for the first sink t_1 in L_1 is shown upon its termination, which is straightforward since it is the first sink to be considered. The nodes are enumerated according to the topological order in L_1. Since node 4 is in $\mathcal{E}_D$ its coding coefficients are multiplied by D and the coding vector at the edge incoming into sink t_1 is $\binom{0}{D}$.

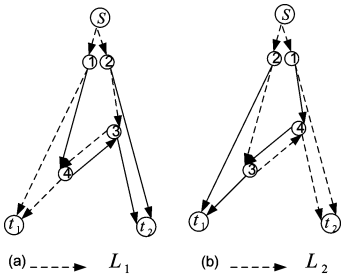

Fig. 6. Example network.

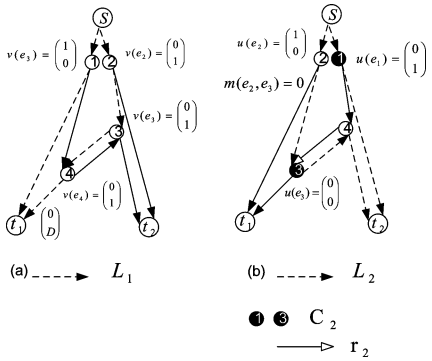

Fig. 7. Example network.

In Fig. 7(b), the stage of the second sink t_2 on L_2 is shown, where at the current stage $\mathcal{C}_2$ contains node e_1 and e_3 (shown as filled circles), where now the nodes are enumerated according to the topological order in L_2. The first path from the source to sink t_2 is $P_{1,2}=\{e_2,e_3,e_4\}$. The second path is $P_{2,2}=\{e_1\}$. Since e_1 and e_3 are in $\mathcal{C}_2$ it follows from definition that $p_{1,2}=\{e_4\}$ and $p_{2,2}$ is an empty set. Thus the edge that is outgoing from e_4 and incoming into e_3 is outgoing from a node in $p_{1,2}$ and incoming into a node in $L\setminus p_{1,2}$. It follows that the coefficient of that edge is in r_2 and is set to zero for the definition of U_2. The unfilled arrow of the edge in Fig. 7(b) indicates that the coefficient is in r_2.

At the current step the coding coefficient $m(e_2,e_3)=0$. The partial coding coefficients for this scenario are shown. Since $\mathbf{u}(e_3)$ is the zero vector, clearly the vectors in U_2 are not a basis and a new coefficient $m(e_2,e_3)$ has to be found.

In Fig. 8(a) we set $m(e_2,e_3)=1$. The partial coding vectors of e_1 and e_3 are now independent, as anticipated by Theorem 1. To find $v(e_3)$ and $v(e_4)$, the following equations have to be solved:

$$v(e_3)=v(e_2)+v(e_4)=\begin{pmatrix}1\\0\end{pmatrix}+v(e_4)$$
$$v(e_4)=Dv(e_3)+v(e_1)=Dv(e_3)+\begin{pmatrix}0\\1\end{pmatrix}. \qquad (18)$$

The resulting global coding vectors are shown in Fig. 8(b).

The global coding vectors incoming into t_1 are $v(e_2)$ and $Dv(e_3)$ (recall that e_3 is in $\mathcal{E}_D$). Since they are independent, the condition of Theorem 2 is maintained. Likewise, the global coding vectors incoming into t_2 are $v(e_1)$ and $v(e_4)$, which are also independent. Therefore, the construction terminated successfully.

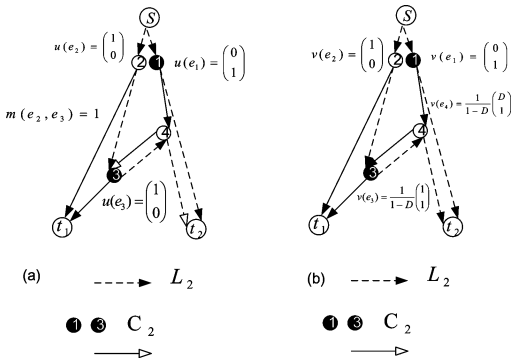

Fig. 8. Example Network.

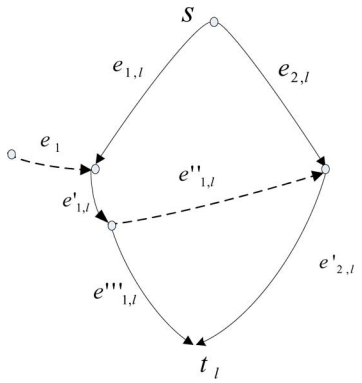

Fig. 9. G for example of a bad code.

E. Example of a Bad Code Designed Using V_l

The set of partial coding vectors U_l is critical for designing codes for cyclic networks. In the following, we consider an example network to show that in general using the set of global coding vectors V_l alone cannot lead to a feasible convolutional network code.

Consider the example network G illustrated in Fig. 9. Note that since edge $e'_{1,l}$ is the only incoming edge into $e''_{1,l}$, we can assume that the head node of $e'_{1,l}$ simply forwards the symbol from $e'_{1,l}$ into $e''_{1,l}$. The same assumption can be made also for $e'''_{1,l}$. Therefore for simplicity, in the line graph in Fig. 10 we omit the nodes that represent $e''_{1,l}$ and $e'''_{1,l}$ (these nodes would simply receive and forward the same symbol).

This particular example is an acyclic network, but the algorithm we examine for this example is our new algorithm. The dashed edges are in L_k for some $k<l$, but not in L_l. The other edges are in L_l. In this example $h=2$. The coefficients $m(\cdot,\cdot)$ were determined in previous steps, for other sinks. The current nodes in $\mathcal{C}_l$ are $e_{1,l}$ and $e_{2,l}$. The current values of the global coding vectors and coding coefficients are shown in the network. Suppose we have reached $e'_{2,l}$ in the topological order and we examine now whether to change the value of $m\left(e_{2,l},e'_{2,l}\right)$. Since $\mathbf{v}\left(e'_{2,l}\right)=(1,1)^T$ and $\mathbf{v}(e_{1,l})=(1,0)^T$ are already a basis, we leave the previous value $m\left(e_{2,l},e'_{2,l}\right)=0$ unchanged. Next we reach $e'_{1,l}$ in the topological order and we need to determine $m\left(e_{1,l},e'_{1,l}\right)$. But now for any value of

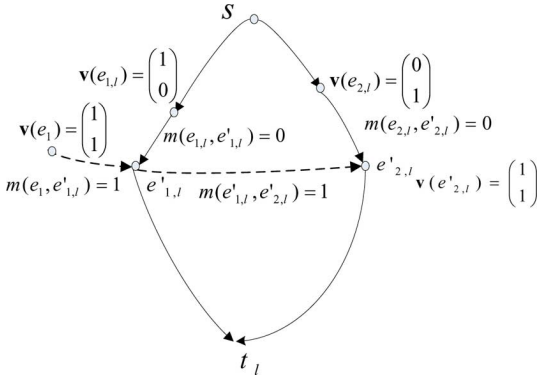

Fig. 10. L of example of a bad code.

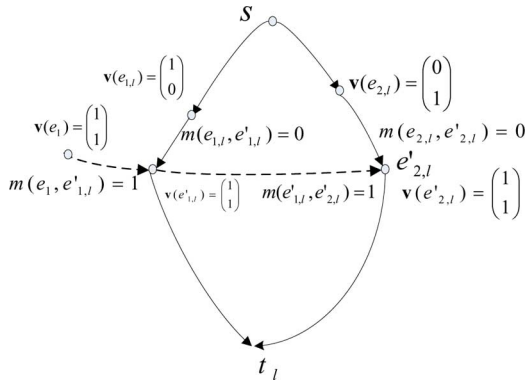

Fig. 11. L of example of a bad code (continued).

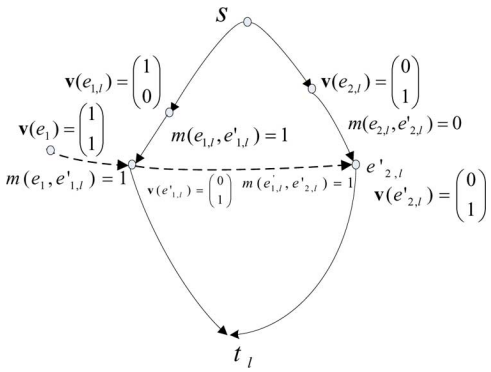

Fig. 12. L of example of a bad code (continued).

$m\left(e_{1,l},e'_{1,l}\right)$, we have $\mathbf{v}\left(e'_{1,l}\right)=\mathbf{v}\left(e'_{2,l}\right)$ and the new set of vectors cannot be a basis! If we keep $m\left(e_{1,l},e'_{1,l}\right)=0$ in its old value as in Fig. 10, the situation will be as shown in Fig. 11. As we see, $\mathbf{v}\left(e'_{1,l}\right)=\mathbf{v}\left(e'_{2,l}\right)=(1,1)^T$. Therefore, the two vectors are not independent. If we set $m\left(e_{1,l},e'_{1,l}\right)=1$ the situation will be as shown in Fig. 12. As we see, $\mathbf{v}\left(e'_{1,l}\right)=\mathbf{v}\left(e'_{2,l}\right)=(0,1)^T$. Therefore, the two vectors are not independent.

F. Adding and Removing Sinks

In the code construction for acyclic networks in [5], there is no direct simple way to add and remove sinks. If we remove or add new sinks, the entire code and generally even the field size have to be changed. In our algorithm for convolutional codes, however, adding a new sink simply corresponds to a new step in the algorithm. As we have shown in the construction algorithm, in the l'th step performed for sink t_l only coefficients of edges in the flow L_l between the source and the new sink might be required to be changed. The modifications of the code in the network are therefore more local. We note that some information will still be required to be exchanged between nodes. However, the coefficients that will be required to be modified are limited only to the nodes that are in a certain flow between the source and the added (or removed) sink. Removing sinks is analogous to adding sinks. By removing a sink we mean that the sink will function as a regular node in the new network. Upon removing a sink we will go over the edges in the flow between the source and the removed sink in a way similar to a step of the original construction algorithm. In this step we check for each edge whether it is possible to decrease the memory size for the coding coefficient while still maintaining the independence for the other remaining sinks, not including of course the removed sink. We observe that the efficient algorithm for removing and adding sinks can be performed for any block or convolutional linear network code, even if it was not originally constructed according to our algorithm.

G. Complexity of the Code Construction

The complexity of the precoding stage depends on the specific algorithm we choose for selecting the nodes in $\mathcal{E}_D$. This stage is very general, and is required also for other construction algorithms of stable codes for cyclic networks. Therefore we do not include it in the complexity computation.

For the construction algorithm it is required to compute the transfer function from a certain node to another node. Therefore, a data structure representing the matrix $C(D)$ defined in (12) is maintained. The relevant values of the coding coefficients can be substituted in $C(D)$ with complexity $O(|E|^2\log d)$. It follows from [23] and from the fact that for unit capacity networks $h\leq|E|$ that $F_M(D)$ given in (14) can be computed from $C(D)$ with complexity $O(|E|^{w+1}\log d\log(|E|\log d))$ where w is called the exponent of multiplying two matrices and known to be in the range $2\leq w<2.37$. The coding vector $\mathbf{v}(e)$ (or $\mathbf{u}(e)$) is given by the ith column of matrix $F_M(D)$ where $\mathbf{v}(e)$ and $\mathbf{u}(e)$ are distinguished by the relevant values of the coding coefficients substituted in matrix $C(D)$. Therefore the coding vectors of all nodes can be computed in complexity $O(|E|^{w+1}\log d\log(|E|\log d))$.

After the precoding, the algorithm finds the d flows from the source s to the sinks. Since the complexity of the Ford and Fulkerson algorithm [24] for maximal flow is $O(|E|h)$, the total complexity of this stage, as in [5], is $O(d|E|h)$.

Now we turn to the d steps of the algorithm. In the lth step, we initialize V_l, U_l and $\mathcal{C}_l$. The complexity of the initialization is $O(h^2)$. For each node e, the new set U_l^n is computed in complexity $O(|E|^{w+1}\log d\log(|E|\log d))$. Since the independence test can be carried out by computing the determinant of the matrix whose columns are the vectors in U_l^n it follows from [23] that the complexity is $O(h^w\log(|E|\log d)\log(h\log(|E|\log d)))$. In the worst case the new coding coefficient $m(e',e)$ is drawn d times, according

to some enumeration. The independence test for U_l^n is performed only once, since according to Theorem 1 if it fails the first time, it will succeed with any other coefficient $m(e',e)$.

We also test the independence of the new sets V_k, $k < l$. The computation of the vectors in V_k, $k < l$ for the d different coefficients $m(e',e)$ has complexity $O(d|E|^{w+1}\log d\log(|E|\log d))$. The independence test of the new sets for at most d different coefficients $m(e,e')$ and at most d different sinks has complexity $O(d^2h^w\log(|E|\log d)\log(h\log(|E|\log d)))$. For all edges in d steps the complexity is $O(d^2|E|^{w+2}\log d\log(|E|\log d))+ d^3h^w|E|\log(|E|\log d)\log(h\log(|E|\log d)))$. For simplicity, if we neglect logarithmic factors (which were also neglected in [5]) the complexity is $O(d^2|E|^{w+2}+d^3h^w|E|) = O(d^3|E|^{w+2})$, since in a unit capacity network $|E| \geq h$. In comparison, the algorithm in [5] for acyclic networks has complexity $O(|E|dh^2+|E|hd^2)$ in the deterministic case.

IV. The Sequential Decoder

We show how the code can be decoded in a polynomial time. In this section we define $\Gamma_I(t_l)$ as the set of h nodes in the flow L_l. This is because in multicast t_l needs only the symbols of these h incoming nodes for decoding. We slightly modify for our notation the definition of the decodability of an h dimensional convolutional multicast code from [11].

Definition 2: In a decodable convolutional network code, for every sink t_l there exists an $h \times h$ matrix denoted by $B_l(D)$ over $F\langle D\rangle$ and a positive integer N such that $B_l(D)\cdot A_l(D) = D^N I_h$, where $A_l(D)$ is defined in (11) and N is defined as the *decoding delay* that depends on the sink t_l and I_h is the $h \times h$ identity matrix.

We briefly repeat here the derivation from [11] of the matrix $B_l(D)$ for decoding. Since the matrix $A_l(D)$ has a nonzero determinant over $F\langle D\rangle$, it can be written as

$$\det(A_l(D)) = D^N\frac{1+Dq(D)}{p(D)} \tag{19}$$

where N is some positive integer, and $p(D)$ and $q(D)$ are polynomials over the binary field. We choose

$$B_l(D) = \frac{p(D)}{1+Dq(D)}J_l(D) \tag{20}$$

where $J_l(D)$ the adjoint matrix of $A_l(D)$. It can be verified that matrix $B_l(D)$ maintains the requirements of Definition 2 and N is the decoding delay.

Since the coding coefficients $m(e,e')$ of our code are polynomials, the elements of the matrices $H_s(D)$ and $Adj(D)$ in (14) are also polynomials. Therefore, the elements of $F_M(D)$ can be written as rational power series, whose denominator is the polynomial $\det(I-C(D))$.

The first step of the sequential decoder is to multiply the sequence of symbols it receives at $e \in \Gamma_I(t_l)$ by $\det(I-C(D))$. The resulting symbols are given by

$$\hat{\mathbf{y}}(D) = \hat{A}_l(D)\mathbf{x}(D) = \det(I-C(D))\mathbf{y}(D) \tag{21}$$

where $\mathbf{y}(D)$ was defined in (10) and given by the matrix relation in (11) and $\hat{A}_l(D) = \det(I-C(D))A_l(D)$. If $A_l(D)$ is invertible, as is the case for our code construction, so is $\hat{A}_l(D)$. Since the determinant of $\hat{A}_l(D)$ is a polynomial, it can be written as

$$\det(\hat{A}_l(D)) = D^N(1+D\hat{q}(D)) \tag{22}$$

where $\hat{q}(D)$ is a polynomial. In (22) we can identify N, which is also the decoding delay, as the smallest degree additive term of $\det(\hat{A}_l(D))$. The decoder at t_l receives the vector $\hat{\mathbf{y}}(D)$ given in (21). We assume that the elements of $\mathbf{x}(D)$ are polynomials, which is the case for finite length transmission. Since the elements of matrix $\hat{A}_l(D)$ are also in $F(D)$, so are the elements of $\hat{\mathbf{y}}(D)$. We choose

$$\hat{B}_l(D) = \frac{1}{1+D\hat{q}(D)}\hat{J}_l(D) \tag{23}$$

where $\hat{J}_l(D)$ is the adjoint matrix of $\hat{A}_l(D)$. If the decoder left multiplies the vector $\hat{\mathbf{y}}(D)$ by $\hat{B}_l(D)$ the result would be

$$\hat{B}_l(D)\cdot\hat{\mathbf{y}}(D) = \frac{1}{1+D\hat{q}(D)}\hat{J}_l(D)\hat{\mathbf{y}}(D) = D^N\mathbf{x}(D). \tag{24}$$

Multiplying by $1+D\hat{q}(D)$ results in

$$D^N\mathbf{x}(D)(1+D\hat{q}(D)) = \hat{J}_l(D)\hat{\mathbf{y}}(D). \tag{25}$$

We can write for any integer $Q > 0$

$$D^N\mathbf{x}(D)(1+D\hat{q}(D)) = \hat{J}_l(D)\hat{\mathbf{y}}(D) \bmod (D^Q) \tag{26}$$

where in our notation $\bmod(D^Q)$, for each polynomial element of the matrices and vectors in both sides of the equation we keep only the additive terms with degree lower than Q.

From (26) for $Q = N+1$ we have

$$\begin{aligned} &D^N\mathbf{x}(D)(1+D\hat{q}(D)) \bmod (D^{N+1}) \\ &\quad = D^N\mathbf{x}(D) \\ &\quad = \hat{J}_l(D)\hat{\mathbf{y}}(D) \bmod (D^{N+1}). \end{aligned} \tag{27}$$

The decoder finds the smallest degree additive term of $\det\hat{A}_l(D)$, which is D^N according to (22). Writing the i'th row of (27) gives

$$D^N X_i(D) = \hat{J}_{l,i}(D)\hat{\mathbf{y}}(D) \bmod (D^{N+1}) \tag{28}$$

where $\hat{J}_{l,i}(D)$ is the i'th row of the matrix $\hat{J}_l(D)$. The left-hand side (LHS) of (28) is the first bit of $X_i(D)$, which can be computed from the right-hand side (RHS) term. Since we are interested in $\hat{J}_l(D) \bmod (D^{N+1})$ we look only at the suffix of length $N+1$ of each element of $\hat{J}_l(D)$. From the received signal $\hat{Y}_{e_i}(D)$ we also need only the symbols that have arrived from instant 0 to instant N, which are altogether $N+1$ symbols. Therefore, the delay for decoding the first symbol of each of the h source streams is indeed at most N.

From (28) and the definition of matrix multiplication, we get the "decoding relation"

$$\begin{aligned} D^N X_i(D) = (&\hat{Y}_{e_1}(D)\hat{J}_{li,1}(D) + \cdots \\ &+\hat{Y}_{e_h}(D)\hat{J}_{li,h}(D)) \bmod (D^{N+1}) \end{aligned} \tag{29}$$

where $\hat{J}_{li,j}(D)$ is element (i,j) of matrix $\hat{J}_l(D)$. After we decode the first bit, there is a feed-forward step that compensates for the effect of the first bit on all the future received symbols. Suppose that the vector of the decoded first bit of each stream is $\hat{\mathbf{x}} = (x_1(0),\ldots,x_h(0))^T$. The decoder has to multiply this vector by the matrix $\hat{A}_l(D)$

$$\tilde{\mathbf{y}}(D) = \hat{A}_l(D)\hat{\mathbf{x}}. \tag{30}$$

We subtract this vector from the vector $\hat{\mathbf{y}}(D)$

$$\hat{\mathbf{y}}_{\text{eff}}(D) = \hat{\mathbf{y}}(D) - \tilde{\mathbf{y}}(D). \tag{31}$$

Since only the first transmitted bit of each stream affects the first received bit of each of the h incoming edges, after the compensation described above, all the elements of $\hat{\mathbf{y}}_{\text{eff}}(D)$ will have a zero constant term. Therefore, we can divide $\hat{\mathbf{y}}_{\text{eff}}(D)$ by D

$$\hat{\mathbf{y}}'_{\text{eff}}(D) = \hat{\mathbf{y}}_{\text{eff}}(D)/D. \tag{32}$$

From this point we use $\hat{\mathbf{y}}'_{\text{eff}}(D)$ for decoding as the vector $\hat{\mathbf{y}}(D)$ and continue to decode the second bit of each stream. Thus, the binary symbols are decoded sequentially. The decoding process is therefore given by (29)–(32).

We note that for some (or all) of the streams X_i, the delay may be smaller than N. In (29), if all the terms $\hat{J}_{li,j}(D)$ have D^{N_1} as a common factor, then both sides of (29) can be divided by D^{N_1}, and a decoding delay of $N - N_1$ is achieved.

A. Decoding Delay

We denote the length of the longest trail of L by r_M. The numerator of element (i,j) of matrix $F_M(D)$, denoted by $f_{i,j}$, corresponds to all the trails from node e_i to node e_j. Since the degree of each polynomial coding coefficient $m(e,e')$ is at most $\lceil\log(d)\rceil + 1$, the degree of the numerator of the element $f_{i,j}$ is at most the sum of the degrees of the coding coefficients over the trail, bounded by $r_M(\lceil\log(d)\rceil + 1)$. It follows that the elements of the matrix $\hat{A}_l(D) = \det(I - C(D))A_l(D)$ are polynomials with degree at most $r_M(\lceil\log(d)\rceil + 1)$. We conclude that the degree of the determinant of $\hat{A}_l(D)$ is at most $hr_M(\lceil\log(d)\rceil + 1)$. Since the determinant is not the zero polynomial, it follows from Section IV that the delay of the sequential decoder for matrix $\hat{A}_l(D)$ is upper bounded by $hr_M(\lceil\log(d)\rceil + 1)$.

V. Discussion and Future Research

The main result of the work is a polynomial time code construction of a multicast linear network code for cyclic networks that achieves the optimal rate. This is the first time such an algorithm is given explicitly. We introduced an efficient sequential decoding scheme for the code. We have analyzed the decoding delay of this code. We have shown that the new code construction is useful also for acyclic networks, since when sinks are added or removed, our algorithm can modify the existing code in an efficient localized manner.

Quoting from [25], in which the polynomial time construction for acyclic networks was introduced: "There are a number of open problems for graphs with cycles. Is there a polynomial time algorithm that finds a coding scheme with rate exactly h? From a practical point of view even an approximate algorithm would be interesting if it allows coding schemes that are faster to decode." Indeed, our construction is polynomial-time, achieves the exact optimal rate h and, using our sequential decoder, can also be efficiently decoded.

An interesting direction for future research would be a code construction for cyclic networks with multiple sources. Another question for future research is how to design network codes for cyclic networks such that the decoding delay is minimized. Future research could also focus on the design and analysis of codes for cyclic networks in the presence of noise.

Appendix

The following lemma will be needed for the proof of Theorems 1 and 2. The lemma considers the relation between the set of coding vectors (either global or partial) and the same set of vectors, when a node associated with one of the vectors in the set is in open loop, that is when all of the coefficients out of that node are set to zero.

1) Lemma 1:

Lemma 1: Let L be a line graph of the network G. Let $\{e_1,\ldots,e_h\}$ be a set of nodes of L and let $W = \{\mathbf{w}(e_1),\ldots,\mathbf{w}(e_i),\ldots,\mathbf{w}(e_h)\}$, be their coding vectors, which may be partial or global coding vectors. Pick index i and consider the coding vectors of the same set of nodes $\tilde{W} = \{\tilde{\mathbf{w}}(e_1),\ldots,\tilde{\mathbf{w}}(e_i),\ldots,\tilde{\mathbf{w}}(e_h)\}$, where we set $m(e_i,e) = 0$ for $\forall\, e \in L$. The set W is a basis if and only if the set $\tilde{W}$ is a basis.

Proof: We find the relation between $\tilde{W}$ and W. The difference is that in the definition of $\tilde{W}$ $m(e_i,e) = 0$ for $\forall\, e \in L$. Suppose that $m(e_i,e), \forall\, e \in L$ are now set to their true values. Since the code is linear, the effect of the network on the coding vector of e_i is that of a linear system. We split node e_i into 3 nodes: e_{tail}, e_{mid} and e_{head}, which are connected by edges $(e_{\text{tail}}, e_{\text{mid}})$ and $(e_{\text{mid}}, e_{\text{head}})$. For the split node, the coefficient $m(e_{\text{head}}, e)$, $e \in L$ is equal to $m(e_i,e)$, $e \in L$ in the original network. Likewise, the coefficient $m(e, e_{\text{tail}})$, $e \in L$ is equal to $m(e,e_i)$, $e \in L$ in the original network. Since, as shown in Section III-A, the code is defined over the ring of rational power series, $F\langle D\rangle$ we can find a rational power series G_{ee} with a variable D which represents the transfer function of the linear system from node e_{head} to node e_{tail} in the network $L \setminus e_{\text{mid}}$. According to the definition of a rational power series, G_{ee} can be written in the form $p(D)/(1 + Dq(D))$, where $p(D)$ and $q(D)$ are polynomials in D. Since each directed cycle in the network contains at least a single delay D, $p(D)$ can be written as $p(D) = Ds(D)$, where $s(D)$ is a polynomial in D. We conclude that G_{ee} can be written in the form $Ds(D)/(1+Dq(D))$. The relation between $\tilde{\mathbf{w}}(e_i)$ and $\mathbf{w}(e_i)$ (when the coefficients $m(e_i,e), \forall\, e \in L$ are set to their true values) is shown in Fig. 13.

According to linear systems theory the relation between $\tilde{\mathbf{w}}(e_i)$ and $\mathbf{w}(e_i)$ is given by

$$\mathbf{w}(e_i) = \frac{1}{1 - G_{ee}}\tilde{\mathbf{w}}(e_i). \tag{33}$$

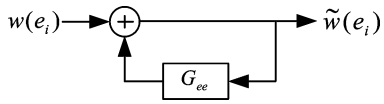

Fig. 13. Relation between $\tilde{\mathbf{w}}(e_i)$ and $\mathbf{w}(e_i)$.

An alternative way to show the relation between $\tilde{\mathbf{w}}(e_i)$ and $\mathbf{w}(e_i)$ is by observing that $\mathbf{w}(e_i)$ can be also interpreted as a sum of a geometric series

$$\begin{aligned}\mathbf{w}(e_i) &= \tilde{\mathbf{w}}(e_i) + G_{ee}\tilde{\mathbf{w}}(e_i) + G_{ee}^2\tilde{\mathbf{w}}(e_i)\cdots \\ &= \frac{1}{1-G_{ee}}\tilde{\mathbf{w}}(e_i). \end{aligned} \tag{34}$$

The factor $1-G_{ee}$ never vanishes identically to zero since G_{ee} is in the form $Ds(D)/(1+Dq(D))$. The other vectors are given by

$$\mathbf{w}(e_j) = \tilde{\mathbf{w}}(e_j) + F_{ij}\frac{1}{1-G_{ee}}\tilde{\mathbf{w}}(e_i), \quad j \neq i \tag{35}$$

where F_{ij} is the transfer function from e_i to e_j. The vectors in W can represent rows of some matrix A. Likewise, the vectors in $\tilde{W}$ can represent rows of some matrix $\tilde{A}$. The matrices A and $\tilde{A}$ have the same rank, since A can be reached from $\tilde{A}$ by multiplication of a row by a nonzero factor and adding a multiplication of this row to the other rows. Therefore A is full rank if and only if $\tilde{A}$ is full rank and the lemma follows. ∎

2) Proof of Theorem 1:

Proof: Recall that the set of partial coding vectors $U_l = \{\mathbf{u}(e_{1,l}), \ldots, \mathbf{u}(e_{i,l}), \ldots, \mathbf{u}(e_{h,l})\}$ is defined when all the coefficients in r_l are zero. Also recall that in the definition of the set of partial coding vectors $U'_l = \left\{\mathbf{u}'(e_{1,l}), \ldots, \mathbf{u}'\left(e'_{i,l}\right), \ldots, \mathbf{u}'(e_{h,l})\right\}$ the coefficients in r'_l are set to zero.

Let $\tilde{U}'_l = \left\{\tilde{\mathbf{u}}'(e_{1,l}), \ldots, \tilde{\mathbf{u}}'\left(e'_{i,l}\right), \ldots, \tilde{\mathbf{u}}'(e_{h,l})\right\}$ denote the coding vectors of $\mathcal{C}'_l$ when all the coefficients in r_l are set to zero and with the new coefficient $m^N(e_{i,l}, e_{i,l})$. Denote by $\tilde{U'}_l^C = \left\{\tilde{\mathbf{u}}'^C(e_{1,l}), \ldots, \tilde{\mathbf{u}}'^C\left(e'_{i,l}\right), \ldots, \tilde{\mathbf{u}}'^C(e_{h,l})\right\}$ the corresponding set with the current coefficient $m^C\left(e_{i,l}, e^n_{i,l}\right)$. From these definitions we have

$$\begin{aligned}\tilde{\mathbf{u}}'\left(e'_{i,l}\right) &= \tilde{\mathbf{u}}'^C\left(e'_{i,l}\right) \\ &\quad + \left(m^N\left(e_{i,l}, e'_{i,l}\right) - m^C\left(e_{i,l}, e'_{i,l}\right)\right)\mathbf{u}(e_{i,l}). \end{aligned} \tag{36}$$

Note that all the vectors in (36) are defined when the coefficients in r_l set to zero. Since in the definition of $\mathbf{u}(e_{i,l})$ the coefficients in r_l are set to zero, it follows that there is no feedback from $e'_{i,l}$ back to $e_{i,l}$ and therefore the vector $\mathbf{u}(e_{i,l})$ is independent of $m^C\left(e_{i,l}, e^n_{i,l}\right)$ or $m^N\left(e_{i,l}, e^n_{i,l}\right)$.

We assumed that the previous set of partial coding vectors $U_l = \{\mathbf{u}(e_{1,l}), \ldots, \mathbf{u}(e_{i,l}), \ldots, \mathbf{u}(e_{h,l})\}$ is a basis. We want to show that the set $\tilde{U}'_l = \left\{\tilde{\mathbf{u}}'(e_{1,l}), \ldots, \tilde{\mathbf{u}}'\left(e'_{i,l}\right), \ldots, \tilde{\mathbf{u}}'(e_{h,l})\right\}$ is also a basis. We have $U_l \setminus \mathbf{u}(e_{i,l}) = \tilde{U}'_l \setminus \tilde{\mathbf{u}}'\left(e'_{i,l}\right)$ since both sets are defined with the coefficients in r_l set to zero. It follows that the vectors in the set $\tilde{U}'_l \setminus \tilde{\mathbf{u}}'\left(e'_{i,l}\right) = \{\tilde{\mathbf{u}}'(e_{1,l}), \ldots, \tilde{\mathbf{u}}'(e_{i-1,l}), \tilde{\mathbf{u}}'(e_{i+1,l}), \ldots, \tilde{\mathbf{u}}'(e_{h,l})\}$ are independent. It remains to show that $\tilde{\mathbf{u}}'\left(e'_{i,l}\right)$ is independent of $\tilde{U}'_l \setminus \tilde{\mathbf{u}}'\left(e'_{i,l}\right)$. The vector $\tilde{\mathbf{u}}'^C\left(e'_{i,l}\right)$ in (36) is dependent on $\tilde{U}'_l \setminus \tilde{\mathbf{u}}'\left(e'_{i,l}\right)$ because otherwise according to Lemma 1 the set U'_l with $m^C\left(e_{i,l}, e'_{i,l}\right)$ would be a basis and we would not have to change $m^C\left(e_{i,l}, e'_{i,l}\right)$ to another $m^N\left(e_{i,l}, e'_{i,l}\right)$. Trivially, it follows that the vector $\tilde{\mathbf{u}}'^C\left(e'_{i,l}\right)$ is dependent on $U_l \setminus \mathbf{u}(e_{i,l})$. If the new $\tilde{\mathbf{u}}'\left(e'_{i,l}\right)$ is dependent on $\tilde{U}'_l \setminus \tilde{\mathbf{u}}'\left(e'_{i,l}\right) = U_l \setminus \mathbf{u}(e_{i,l})$ then following from (36) that it can be written in the following form:

$$\begin{aligned}&\tilde{\mathbf{u}}'\left(e'_{i,l}\right) \\ &\quad = \tilde{\mathbf{u}}'^C\left(e'_{i,l}\right) + \left(m^N\left(e_{i,l}, e'_{i,l}\right) - m^C\left(e_{i,l}, e'_{i,l}\right)\right)\mathbf{u}(e_{i,l}) \\ &\quad = \alpha_1\mathbf{u}(e_{1,l}) + \cdots + \alpha_{i-1}\mathbf{u}(e_{i-1,l}) \\ &\qquad + \alpha_{i+1}\mathbf{u}(e_{i+1,l}) + \cdots + \alpha_h\mathbf{u}(e_{h,l}) \end{aligned} \tag{37}$$

and it follows that:

$$\begin{aligned}\tilde{\mathbf{u}}'^C\left(e'_{i,l}\right) &= \alpha_1\mathbf{u}(e_{1,l}) + \cdots + \alpha_{i-1}\mathbf{u}(e_{i-1,l}) + \cdots \\ &\quad + \left(m^C\left(e_{i,l}, e'_{i,l}\right) - m^N\left(e_{i,l}, e'_{i,l}\right)\right)\mathbf{u}(e_{i,l}) \\ &\quad + \alpha_{i+1}\mathbf{u}(e_{i+1,l}) + \cdots + \alpha_h\mathbf{u}(e_{h,l}). \end{aligned} \tag{38}$$

Since U_l is a basis and since $\tilde{\mathbf{u}}'^C\left(e'_{i,l}\right)$ is dependent on $U_l \setminus \mathbf{u}(e_{i,l})$, (38) can be maintained only if $m^N\left(e_{i,l}, e'_{i,l}\right) = m^C\left(e_{i,l}, e'_{i,l}\right)$. Therefore, for any other choice of $m^N\left(e_{i,l}, e'_{i,l}\right)$ the vector $\tilde{\mathbf{u}}'\left(e'_{i,l}\right)$ is independent of $\tilde{U}'_l \setminus \tilde{\mathbf{u}}'\left(e'_{i,l}\right)$. Thus $\tilde{U}'_l$ is a basis. It follows from Lemma 1 that since $\tilde{U}'_l$ is a basis, the set U'_l is a basis for any $m^N\left(e_{i,l}, e'_{i,l}\right) \neq m^C\left(e_{i,l}, e'_{i,l}\right)$. ∎

3) Proof of Theorem 2:

Proof: Assume that $V_k^C = \left\{\mathbf{v}^C(e_{1,k}), \ldots, \mathbf{v}^C(e_{h,k})\right\}$, defined in the theorem is a basis. We analyze under which conditions the set of global coding vectors $V_k^N = \left\{\mathbf{v}^N(e_{1,k}), \ldots, \mathbf{v}^N(e_{h,k})\right\}$, obtained after replacing $m^C\left(e_{i,l}, e'_{i,l}\right)$ to $m^N\left(e_{i,l}, e'_{i,l}\right)$, is also a basis.

Assume that the outgoing edges of $e_{i,l}$, except $\left(e_{i,l}, e'_{i,l}\right)$, are $\Gamma_O = \{(e_{i,l}, e_1), \ldots, (e_{i,l}, e_q)\}$. The system G_{ee} defined in the proof of Lemma 1 can be expressed as $G_{ee} = G_1 + m\left(e_{i,l}, e'_{i,l}\right)G_2$, where G_1 is a transfer function, defined as G_{ee} but in the network $L \setminus \left(e_{i,l}, e'_{i,l}\right)$ and $m\left(e_{i,l}, e'_{i,l}\right)G_2$ is the transfer function defined in $G \setminus \Gamma_O$. The vector $\tilde{\mathbf{v}}(e_{i,l})$ is the coding vector of $e_{i,l}$ when the coefficients of the outgoing edges

of $e_{i,l}$ are all zero. Then for an arbitrary coefficient $m\left(e_{i,l},e'_{i,l}\right)$ the global coding vector of $e_{i,l}$ is $\mathbf{v}(e_{i,l})$ and is given by

$$\begin{aligned}\mathbf{v}(e_{i,l}) &= \tilde{\mathbf{v}}(e_{i,l}) + G_{ee}\tilde{\mathbf{v}}(e_{i,l}) + \cdots = \frac{1}{1-G_{ee}}\tilde{\mathbf{v}}(e_{i,l}) \\ &= \frac{1}{1-G_1-m\left(e_{i,l},e'_{i,l}\right)G_2}\tilde{\mathbf{v}}(e_{i,l}) \\ &= \frac{1}{(1-G_1)\left(1-\frac{m\left(e_{i,l},e'_{i,l}\right)G_2}{1-G_1}\right)}\tilde{\mathbf{v}}(e_{i,l}). \end{aligned} \tag{39}$$

We can divide by $1-G_1$ since the rational power series G_1 can be written as $Ds(D)/(1+Dq(D))$, where $s(D)$ and $q(D)$ are polynomials in D, since each directed cycle in the network contains at least a single delay. Note that G_2 can also be written in the same form. Define as $\mathbf{y}(e_{i,l})$ the coding vector of $e_{i,l}$ when $m\left(e_{i,l},e'_{i,l}\right) = 0$. From (39) we observe that when $m\left(e_{i,l},e'_{i,l}\right)=0$, $\mathbf{v}(e_{i,l})=\mathbf{y}(e_{i,l})=\tilde{\mathbf{v}}(e_{i,l})/(1-G_1)$. Thus

$$\mathbf{v}(e_{i,l}) = \frac{1}{1-m\left(e_{i,l},e'_{i,l}\right)Q}\mathbf{y}(e_{i,l}) \tag{40}$$

where $Q = G_2/(1-G_1)$. Note the rational power series Q can be written in the form $Ds(D)/(1+Dq(D))$, since G_1 and G_2 can be written in this form. Therefore, for the new coding coefficient $m^N\left(e_{i,l},e'_{i,l}\right)$ we have

$$\mathbf{v}^N(e_{i,l}) = \frac{1}{1-m^N\left(e_{i,l},e'_{i,l}\right)Q}\mathbf{y}(e_{i,l}). \tag{41}$$

Similarly, for coefficient $m^C\left(e_{i,l},e'_{i,l}\right)$ we have

$$\mathbf{v}^C(e_{i,l}) = \frac{1}{1-m^C\left(e_{i,l},e'_{i,l}\right)Q}\mathbf{y}(e_{i,l}). \tag{42}$$

The difference between the two vectors is

$$\begin{aligned}&\mathbf{v}^N(e_{i,l}) - \mathbf{v}^C(e_{i,l}) \\ &\quad= \left(\frac{1}{1-m^N\left(e_{i,l},e'_{i,l}\right)Q} - \frac{1}{1-m^C\left(e_{i,l},e'_{i,l}\right)Q}\right) \\ &\qquad\times \mathbf{y}(e_{i,l}) \\ &\quad= \frac{\left(m^N\left(e_{i,l},e'_{i,l}\right)-m^C\left(e_{i,l},e'_{i,l}\right)\right)Q}{\left(1-m^N\left(e_{i,l},e'_{i,l}\right)Q\right)\left(1-m^C\left(e_{i,l},e'_{i,l}\right)Q\right)}\mathbf{y}(e_{i,l}) \\ &\quad= f\left(m^N\left(e_{i,l},e'_{i,l}\right)\right)\cdot Q\cdot\mathbf{y}(e_{i,l}) \end{aligned} \tag{43}$$

where the function $f\left(m^N\left(e_{i,l},e'_{i,l}\right)\right)$ is defined for a fixed $m^C\left(e_{i,l},e'_{i,l}\right)$

$$\begin{aligned}&f\left(m^N\left(e_{i,l},e'_{i,l}\right)\right) \\ &\quad= \frac{\left(m^N\left(e_{i,l},e'_{i,l}\right)-m^C\left(e_{i,l},e'_{i,l}\right)\right)}{\left(1-m^N\left(e_{i,l},e'_{i,l}\right)Q\right)\left(1-m^C\left(e_{i,l},e'_{i,l}\right)Q\right)}. \end{aligned} \tag{44}$$

Consider the effect of the replacement from $\mathbf{v}^C(e_{i,l})$ to $\mathbf{v}^N(e_{i,l})$ on the set of vectors at the input of another sink t_k. The linear network code is equivalent to a linear system operating on the coding vectors. Define the transfer function from $e_{i,l}$ to $e_{j,k}$, when $m(e_{i,l},e)=0, \forall\, e\in\Gamma_O\setminus\left(e_{i,l},e'_{i,l}\right)$ as the rational power series $F_{1,j}$. Similarly, define the transfer function from $e_{i,l}$ to $e_{j,k}$, when only the coefficient $m\left(e_{i,l},e'_{i,l}\right)=0$ as the rational power series $F_{2,j}$. Due to superposition in linear systems, the total transfer function from $e_{i,l}$ to $e_{j,k}$ before replacing $m^C\left(e_{i,l},e'_{i,l}\right)$ is $F_j^C = m^C\left(e_{i,l},e'_{i,l}\right)F_{1,j}+F_{2,j}$ and after the replacement $F_j^N = m^N\left(e_{i,l},e'_{i,l}\right)F_{1,j}+F_{2,j}$. Therefore, the relation between $\mathbf{v}^N(e_{j,k})$ and $\mathbf{v}^C(e_{j,k})$ is given by

$$\begin{aligned}&\mathbf{v}^N(e_{j,k}) - \mathbf{v}^C(e_{j,k}) \\ &\quad= \left(m^N\left(e_{i,l},e'_{i,l}\right)F_{1,j}+F_{2,j}\right)\mathbf{v}^N(e_{i,l}) \\ &\qquad- \left(m^C\left(e_{i,l},e'_{i,l}\right)F_{1,j}+F_{2,j}\right)\mathbf{v}^C(e_{i,l}) \\ &\quad= \left(\frac{m^N\left(e_{i,l},e'_{i,l}\right)F_{1,j}+F_{2,j}}{1-m^N\left(e_{i,l},e'_{i,l}\right)Q}\right. \\ &\qquad\left.- \frac{m^C\left(e_{i,l},e'_{i,l}\right)F_{1,j}+F_{2,j}}{1-m^C\left(e_{i,l},e'_{i,l}\right)Q}\right)\mathbf{y}(e_{i,l}), \\ &\quad 1\le j\le h. \end{aligned} \tag{45}$$

Rearranging terms in (45)

$$\begin{aligned}&\mathbf{v}^N(e_{j,k}) - \mathbf{v}^C(e_{j,k}) \\ &\quad= \left(\frac{m^N\left(e_{i,l},e'_{i,l}\right)F_{1,j}}{1-m^N\left(e_{i,l},e'_{i,l}\right)Q} - \frac{m^C\left(e_{i,l},e'_{i,l}\right)F_{1,j}}{1-m^C\left(e_{i,l},e'_{i,l}\right)Q}\right. \\ &\qquad\left.+ \frac{F_{2,j}}{1-m^N\left(e_{i,l},e'_{i,l}\right)Q} - \frac{F_{2,j}}{1-m^C\left(e_{i,l},e'_{i,l}\right)Q}\right) \\ &\qquad\times \mathbf{y}(e_{i,l}) \\ &\quad= \left(\frac{\left(m^N\left(e_{i,l},e'_{i,l}\right)-m^C\left(e_{i,l},e'_{i,l}\right)\right)F_{1,j}}{\left(1-m^N\left(e_{i,l},e'_{i,l}\right)Q\right)\left(1-m^C\left(e_{i,l},e'_{i,l}\right)Q\right)}\right. \\ &\qquad\left.+ \frac{\left(m^N\left(e_{i,l},e'_{i,l}\right)-m^C\left(e_{i,l},e'_{i,l}\right)\right)QF_{2,j}}{\left(1-m^N\left(e_{i,l},e'_{i,l}\right)Q\right)\left(1-m^C\left(e_{i,l},e'_{i,l}\right)Q\right)}\right) \\ &\qquad\times \mathbf{y}(e_{i,l}) \\ &\quad= \left(\frac{\left(m^N\left(e_{i,l},e'_{i,l}\right)-m^C\left(e_{i,l},e'_{i,l}\right)\right)}{\left(1-m^N\left(e_{i,l},e'_{i,l}\right)Q\right)\left(1-m^C\left(e_{i,l},e'_{i,l}\right)Q\right)}\right) \\ &\qquad\times (F_{1,j}+QF_{2,j})\mathbf{y}(e_{i,l}) \\ &\quad= f\left(m^N\left(e_{i,l},e'_{i,l}\right)\right)\cdot H_j\cdot\mathbf{y}(e_{i,l}), \quad 1\le j\le h \end{aligned} \tag{46}$$

where $H_j \triangleq F_{1,j}+QF_{2,j}$. We know that before changing $m^C\left(e_{i,l},e'_{i,l}\right)$ to $m^N\left(e_{i,l},e'_{i,l}\right)$ the set of vectors $V_k^C = \left\{\mathbf{v}^C(e_{1,k}),\ldots,\mathbf{v}^C(e_{h,k})\right\}$ was a basis. We analyze under

which conditions on $m^N\left(e_{i,l},e'_{i,l}\right)$ the new set of coding vectors, which is $V_k^N=\left\{\mathbf{v}^N(e_{1,k}),\ldots,\mathbf{v}^N(e_{h,k})\right\}$ is also a basis. Suppose the representation of $\mathbf{y}(e_{i,l})$ in basis V_k^C is

$$\mathbf{y}(e_{i,l})=\beta_1\mathbf{v}^C(e_{1,k})+\beta_2\mathbf{v}^C(e_{2,k})+\cdots+\beta_h\mathbf{v}^C(e_{h,k}). \quad (47)$$

We examine independence of the vectors in V_k^N according to definition

$$\alpha_1\mathbf{v}^N(e_{1,k})+\cdots+\alpha_h\mathbf{v}^N(e_{h,k})=0. \quad (48)$$

If this equation has a solution other than $\alpha_1=\cdots=\alpha_h=0$, then V_k^N is not a basis. Using (46), (48) translates into

$$\begin{aligned}&\alpha_1\Big\{\mathbf{v}^C(e_{1,k})+f\left(m^N\left(e_{i,l},e'_{i,l}\right)\right)\\&\qquad\cdot H_1\mathbf{y}(e_{i,l})\Big\}+\cdots\\&+\alpha_h\Big\{\mathbf{v}^C(e_{h,k})+f\left(m^N\left(e_{i,l},e'_{i,l}\right)\right)\\&\qquad\cdot\cdot H_h\mathbf{y}(e_{i,l})\Big\}=0.\end{aligned} \quad (49)$$

Rearranging terms and using (47)

$$\begin{aligned}&\mathbf{v}^C(e_{1,k})\Big\{\alpha_1+\alpha_1 f\left(m^N\left(e_{i,l},e'_{i,l}\right)\right)\cdot H_1\beta_1\\&\qquad+\alpha_2 f\left(m^N\left(e_{i,l},e'_{i,l}\right)\right)\cdot H_2\beta_1+\cdots\\&\qquad+\alpha_h f\left(m^N\left(e_{i,l},e'_{i,l}\right)\right)\cdot H_h\beta_1\Big\}+\cdots\\&+\mathbf{v}^C(e_{h,k})\Big\{\alpha_h+\alpha_1 f\left(m^N\left(e_{i,l},e'_{i,l}\right)\right)\cdot H_1\beta_h\\&\qquad+\alpha_2 f\left(m^N\left(e_{i,l},e'_{i,l}\right)\right)\cdot H_2\beta_h+\cdots\\&\qquad+\alpha_h f\left(m^N\left(e_{i,l},e'_{i,l}\right)\right)\cdot H_h\beta_h\Big\}=0.\end{aligned} \quad (50)$$

Since V_k^C is a basis it is required that

$$\begin{aligned}&\alpha_1+\alpha_1 f\left(m\left(e_{i,l},e'_{i,l}\right)\right)\cdot H_1\beta_1\\&\quad+\alpha_2 f\left(m\left(e_{i,l},e'_{i,l}\right)\right)\cdot H_2\beta_1+\cdots\\&\quad+\alpha_h f\left(m\left(e_{i,l},e'_{i,l}\right)\right)\cdot H_h\beta_1=0\\&\quad\vdots\\&\alpha_h+\alpha_1 f\left(m\left(e_{i,l},e'_{i,l}\right)\right)\cdot H_1\beta_h\\&\quad+\alpha_2 f\left(m\left(e_{i,l},e'_{i,l}\right)\right)\cdot H_2\beta_h+\cdots\\&\quad+\alpha_h f\left(m\left(e_{i,l},e'_{i,l}\right)\right)\cdot H_h\beta_h=0\end{aligned}$$

or in matrix notation

$$-\frac{1}{f\left(m^N\left(e_{i,l},e'_{i,l}\right)\right)}\begin{pmatrix}\alpha_1\\\alpha_2\\\vdots\\\alpha_h\end{pmatrix} \quad (51)$$

$$\begin{aligned}&=\begin{pmatrix}H_1\beta_1&H_2\beta_1&\ldots&H_h\beta_1\\H_1\beta_2&H_2\beta_2&\ldots&H_h\beta_2\\\vdots&\ldots&\ddots&\vdots\\H_1\beta_h&H_2\beta_h&\ldots&H_h\beta_h\end{pmatrix}\begin{pmatrix}\alpha_1\\\alpha_2\\\vdots\\\alpha_h\end{pmatrix}\\&\triangleq A\begin{pmatrix}\alpha_1\\\alpha_2\\\vdots\\\alpha_h\end{pmatrix}.\end{aligned} \quad (52)$$

We could divide by $f\left(m^N\left(e_{i,l},e'_{i,l}\right)\right)$ since $f\left(m^N\left(e_{i,l},e'_{i,l}\right)\right)=0$ only for $m^N\left(e_{i,l},e'_{i,l}\right)=m^C\left(e_{i,l},e'_{i,l}\right)$, but we examine the case where $m^N\left(e_{i,l},e'_{i,l}\right)\neq m^C\left(e_{i,l},e'_{i,l}\right)$. We see that (51) can be maintained only if $\alpha_1=\cdots=\alpha_h=0$ or if A has an eigenvalue

$$\lambda=-\frac{1}{f\left(m^N\left(e_{i,l},e'_{i,l}\right)\right)}. \quad (53)$$

In the following we show that matrix A has an eigenvalue 0 with multitude $h-1$ and an eigenvalue

$$\lambda=\operatorname{trace}(A)=H_1\beta_1+H_2\beta_2+\cdots+H_h\beta_h \quad (54)$$

with multitude 1. The rank of the $h\times h$ matrix A is 1. Therefore, the solution of the equation $A\mathbf{v}=0$ has dimension $h-1$. Thus the geometric multiplicity of the eigenvalue $\lambda=0$ is $h-1$. The algebraic multiplicity of an eigenvalue is not smaller than the geometric multiplicity [26, Th. 8.5]. Therefore A has an eigenvalue 0 of algebraic multiplicity of at least $h-1$. Since the sum of the eigenvalues of a matrix equals its trace, A also has an eigenvalue $\lambda=\operatorname{trace}(A)$ of algebraic multiplicity 1.

According to (53)$\lambda\neq 0$. Therefore, we have only to consider $\lambda=\operatorname{trace}(A)$. Assume for now that

$$\lambda=-\frac{1}{f\left(m^N\left(e_{i,l},e'_{i,l}\right)\right)}=\operatorname{trace}(A). \quad (55)$$

Then according to the definition of $f(m^N\left(e_{i,l},e'_{i,l}\right)$ in (44)

$$\begin{aligned}&f\left(m^N\left(e_{i,l},e'_{i,l}\right)\right)\\&\quad=\frac{m^N\left(e_{i,l},e'_{i,l}\right)-m^C\left(e_{i,l},e'_{i,l}\right)}{\left(1-m^N\left(e_{i,l},e'_{i,l}\right)Q\right)\left(1-m^C\left(e_{i,l},e'_{i,l}\right)Q\right)}\\&\quad=-\frac{1}{\operatorname{trace}(A)}\end{aligned} \quad (56)$$

where we could divide (55) by $\operatorname{trace}(A)$ in order to derive (56), since we already showed that if $\lambda=\operatorname{trace}(A)=0$ then (53) cannot be maintained. We could multiply (55)

by $f\left(m^N\left(e_{i,l},e'_{i,l}\right)\right)$, since $f\left(m^N\left(e_{i,l},e'_{i,l}\right)\right) \neq 0$ if $m^N\left(e_{i,l},e^n_{i,l}\right) \neq m^C\left(e_{i,l},e'_{i,l}\right)$. Thus

$$\frac{m^N\left(e_{i,l},e'_{i,l}\right)}{\left(1-m^N\left(e_{i,l},e'_{i,l}\right)Q\right)\left(1-m^C\left(e_{i,l},e'_{i,l}\right)Q\right)} = \frac{m^C\left(e_{i,l},e'_{i,l}\right)}{\left(1-m^N\left(e_{i,l},e'_{i,l}\right)Q\right)\left(1-m^C\left(e_{i,l},e'_{i,l}\right)Q\right)} - \frac{1}{\text{trace}(A)}. \quad (57)$$

Or

$$m^N\left(e_{i,l},e'_{i,l}\right) = m^C\left(e_{i,l},e'_{i,l}\right) - \frac{\left(1-m^N\left(e_{i,l},e'_{i,l}\right)Q\right)\left(1-m^C\left(e_{i,l},e'_{i,l}\right)Q\right)}{\text{trace}(A)} \quad (58)$$

where $(1-m^N\left(e_{i,l},e'_{i,l}\right)Q)(1-m^C\left(e_{i,l},e'_{i,l}\right)Q))$ does not vanish identically to zero since we recall that $m^N\left(e_{i,l},e'_{i,l}\right)$ and $m^C\left(e_{i,l},e'_{i,l}\right)$ are polynomials in D whereas Q can be written in the form $Ds(D)/(1+Dq(D))$, where $s(D)$ and $q(D)$ are polynomials in D. Rearranging terms in (58) we have

$$m^N\left(e_{i,l},e'_{i,l}\right)\left(1-Q\frac{1-m^C\left(e_{i,l},e'_{i,l}\right)Q}{\text{trace}(A)}\right) = m^C\left(e_{i,l},e'_{i,l}\right) - \frac{1-m^C\left(e_{i,l},e'_{i,l}\right)Q}{\text{trace}(A)}. \quad (59)$$

To show that we can divide (59) by

$$1-Q\frac{1-m^C\left(e_{i,l},e'_{i,l}\right)Q}{\text{trace}(A)} \quad (60)$$

we have to prove that it does not vanish identically to zero. Suppose that it does vanish identically to zero

$$1-Q\frac{1-m^C\left(e_{i,l},e'_{i,l}\right)Q}{\text{trace}(A)} = 0. \quad (61)$$

Then according to (59) it follows that:

$$m^C\left(e_{i,l},e'_{i,l}\right) - \frac{1-m^C\left(e_{i,l},e'_{i,l}\right)Q}{\text{trace}(A)} = 0. \quad (62)$$

Therefore (61) becomes

$$1-Q\frac{1-m^C\left(e_{i,l},e'_{i,l}\right)Q}{\text{trace}(A)} = 1-Qm^C\left(e_{i,l},e'_{i,l}\right) = 0. \quad (63)$$

But if $1-Qm^C\left(e_{i,l},e'_{i,l}\right)=0$, then (59) cannot be maintained since $m^N\left(e_{i,l},e'_{i,l}\right) \neq m^C\left(e_{i,l},e'_{i,l}\right)$ [also, (61) cannot be maintained since $1 \neq 0$]. We conclude that the term in (60) does not vanish identically to zero and we can therefore divide (59) by this expression, which yields

$$m^N\left(e_{i,l},e'_{i,l}\right) = \frac{m^C\left(e_{i,l},e'_{i,l}\right) - \frac{1-m^C\left(e_{i,l},e'_{i,l}\right)Q}{\text{trace}(A)}}{1-Q\frac{1-m^C\left(e_{i,l},e'_{i,l}\right)Q}{\text{trace}(A)}}. \quad (64)$$

Therefore, for at most a single choice of $m^N\left(e_{i,l},e'_{i,l}\right)$, the one given in (64), (55) can be maintained. It follows that for at most a single choice of $m^N\left(e_{i,l},e'_{i,l}\right)$ the set V_k^N will not be a basis. ■

REFERENCES

[1] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. Yeung, "Network information flow," *IEEE Trans. Inf. Theory*, vol. 46, no. 4, pp. 1204–1216, Jul. 2000.

[2] M. Feder, D. Ron, and A. Tavory, "Bounds on linear codes for network multicast," *Electron. Colloq. Computat. Complex. (ECCC)*, vol. 10, no. 33, 2003.

[3] A. Rasala-Lehman and E. Lehman, "Complexity classification of network information flow problems," in *Proc. 15th Annu. ACM-SIAM Symp. Discrete Algorithms*, New Orleans, LA, Jan. 2004, pp. 142–150.

[4] C. Fragouli, E. Soljanin, and A. Shokrollahi, "Network coding as a coloring problem," in *Proc. 2004 Conf. Inf. Sci. Syst.*, Princeton, NJ, Mar. 2004.

[5] S. Jaggi, P. Sanders, P. A. Chou, M. Effros, S. Egner, K. Jain, and L. Tolhuizen, "Polynomial time algorithms for multicast network code construction," *IEEE Trans. Inf. Theory*, vol. 51, pp. 1973–1982, Jun. 2005.

[6] S. Jaggi, M. Effros, T. Ho, and M. Médard, "On linear network coding," in *Proc. 42nd Allerton Conf. Commun., Contr. Comput.*, Monticello, IL, Sep. 2004.

[7] S.-Y. R. Li, R. W. Yeung, and N. Cai, "Linear network coding," *IEEE Trans. Inf. Theory*, vol. 49, pp. 371–381, Feb. 2003.

[8] R. Koetter and M. Médard, "An algebraic approach to network coding," *IEEE/ACM Trans. Netw.*, vol. 11, pp. 782–795, Oct. 2003.

[9] C. Fragouli and E. Soljanin, "A connection between network coding and convolutional codes," in *Proc. Int. Conf. Commun. (ICC)*, Paris, France, Jun. 2004, vol. 2, pp. 661–666.

[10] H. F. Lu, "Binary linear network codes," in *Proc. 2007 IEEE Inf. Theory Workshop in Inf. Theory for Wireless Netw.*, 2007.

[11] R. W. Yeung, R. Li S.-Y, N. Cai, and Z. Zhang, "Network coding theory," *Found. Trends Commun. Inf. Theory*, vol. 2, no. 4 and 5, pp. 241–381, 2005.

[12] S.-Y. R. Li and R. W. Yeung, "On convolutional network coding," in *Proc. IEEE Int. Symp. Inf. Theory*, Seattle, WA, Jul. 2006, pp. 1743–1747.

[13] T. Ho, R. K. M. Médard, M. Effros, J. Shi, and D. Karger, "A random linear network coding approach to multicast," *IEEE Trans. Inf. Theory*, vol. 52, pp. 4413–4430, Oct. 2006.

[14] T. C. Ho, Y.-H. Chang, and K. J. Han, "On constructive network coding for multiple unicasts," in *Proc. 44th Allerton Conf. Commun., Contr. Comupt.*, Monticello, IL, Sep. 2006.

[15] N. Harvey, "Deterministic Network Coding by Matrix Completion," Master's, MIT, Cambridge, MA, 2005.

[16] E. Erez and M. Feder, "Convolutional network codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Chicago, IL, Jun. 2004, pp. 146–146.

[17] E. Erez and M. Feder, "Convolutional network codes for cyclic networks," in *Proc. First Workshop on Network Coding, Theory, Appl.*, Riva del Garda, Italy, Apr. 2005.

[18] E. Erez and M. Feder, "Efficient network codes for cyclic networks," in *Proc. IEEE Int. Symp. Inf. Theory*, Adelaide, Australia, Sep. 2005, pp. 1982–1986.

[19] E. Erez, "Topics in network coding," Ph.D., Tel Aviv Univ., Tel Aviv, 2007.
[20] L. Song, R. Yeung, and N. Cai, "A separation theorem for single-source network coding," *IEEE Trans. Inf. Theory*, vol. 52, pp. 1861–1871, May 2006.
[21] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations*, R. E. Miller and T. W. Thatcher, Eds. New York: Plenum, 1972, pp. 85–109.
[22] G. Even, J. Naor, B. Schieber, and M. Sudan, "Approximating minimum feedback sets and multicuts in directed graphs," *Algorithmica*, vol. 20, no. 2, pp. 151–174, 1998.
[23] C. P. Jeannerod and G. Villard, "Essentially optimal computation of the inverse of generic polynomial matrices," *J. Complex.*, vol. 21, no. 1, pp. 72–86, Feb. 2005.
[24] L. R. Ford and D. R. Fulkerson, *Flows in Networks*. Princeton, NJ: Princeton Univ. Press, 1962.
[25] P. Sanders, S. Egner, and L. Tolhuizen, "Polynomial time algorithms for network information flow," in *Proc. 15th Ann. ACM Symp. Parallel Algorithms Architect.*, San Diego, CA, Jun. 2003, pp. 286–294.
[26] S. Roman, *Advanced Linear Algebra*, 3rd ed. New York: Springer, 2008.

Elona Erez (S'01–M'08) received the B.Sc. (*summa cum laude*), M.Sc. (*summa cum laude*), and Ph.D. degrees from Tel Aviv University, Israel, all in electrical engineering in 1999, 2002, and 2007, respectively.

She is currently a Postdoctoral Associate with Yale University, New Haven, CT, with the Department of Electrical Engineering. During 2007–2008, she was a Postdoctoral scholar with the Department of Electrical Engineering, California Institute of Technology (Caltech). During summer 2005, she was a visiting researcher with the Laboratory for Information and Decision Systems (LIDS), Massachusetts Institute of Technology (MIT), Cambridge. Her research interests are in the fields of information theory and data networks, with special interest in network coding.

Dr. Erez received the Weinstein Prize for an outstanding student in signal processing in 2003 and 2005 and the Weinstein Prize for an outstanding publication in signal processing in 2002 and 2003. She received Colton Prize for an outstanding student in 2003–2006.

Meir Feder (S'81–M'87–SM'93–F'99) received the B.Sc. and M.Sc. degrees from Tel-Aviv University, Israel, and the D.Sc. degree from the Massachusetts Institute of Technology (MIT) Cambridge, and the Woods Hole Oceanographic Institution, Woods Hole, MA, all in electrical engineering in 1980, 1984, and 1987, respectively.

After being a Research Associate and Lecturer with the Massachusetts Institute of Technology (MIT), Cambridge, he joined the Department of Electrical Engineering—Systems, Tel-Aviv University, in 1989, where he is now a Professor. He had visiting appointments with the Woods Hole Oceanographic Institution, Scripps Institute, Bell Laboratories, and in 1995–1996, he was a Visiting Professor at MIT. He is also extensively involved in the high-tech industry and cofounded several companies including Peach Networks, a developer of a unique server-based interactive TV solution which was acquired on March 2000 by Microsoft, and Amimon, a leading provider of ASIC's for wireless high-definition A/V connectivity for the home.

Prof. Feder is a corecipient of the 1993 IEEE Information Theory Best Paper Award. He also received the 1978 "Creative Thinking" award of the Israeli Defense Forces, the 1994 Tel-Aviv University Prize for Excellent Young Scientists, the 1995 Research Prize of the Israeli Electronic Industry, and the Research Prize in Applied Electronics of the Ex-Serviceman Association, London, awarded by Ben-Gurion University. Between June 1993–June 1996, he served as an Associate Editor for Source Coding of the IEEE TRANSACTIONS ON INFORMATION THEORY.