Initialization Algorithms for Convolutional Network Coding

Maxim Lvov, *Student Member, IEEE*, and Haim H. Permuter, *Senior Member, IEEE*

***Abstract*— We present algorithms for initializing a convolutional network coding (CNC) scheme in networks that may contain cycles. An initialization process for finding global encoding kernels (GEK) is needed if the network is unknown or if local encoding kernels are chosen randomly. During the initialization process every source node transmits basis vectors and every sink node gets the impulse response of the network. The impulse response is then used to find the GEK, which are needed for a decoding algorithm and to find the set of all achievable rates. We present two initialization algorithms that find the GEK and one algorithm that finds achievable rates from the GEK. In the first initialization algorithm it is assumed that we can perform a reset operation on the network at some fixed times, while the second algorithm does not operate under this assumption. Unlike acyclic networks, for which it is sufficient to transmit basis vectors one after another, the initialization of cyclic networks is more involved, as test symbols from different times interfere with each other and the impulse response is of infinite duration. Our algorithms use only a finite number of the initial values of the impulse response to find the full GEK. This is possible because a CNC scheme can be described by a state space representation and, using the Cayley-Hamilton theorem, it is possible to find its full impulse response from its initial values.**

***Index Terms*— Cayley-Hamilton theorem, convolutional network coding, cyclic networks, linear network coding, system identification.**

I. Introduction

Network coding is a technique that is used to increase a network throughput. The idea behind this coding scheme is that the relay nodes transmit functions of the received symbols on their output links, rather than simply routing them. Ahlswede *et al.* [1] showed that for a one source, multicast acyclic network, the maximal network throughput is equal to the minimum cut between the source and any sink node. For cyclic networks, a CNC scheme was presented by Li *et al.* [2], and the existence of an optimal CNC code (one that achieves the min-cut bound given in [1]) was proved by Koetter and Médard [3]. Since then, much work has been devoted to constructing codes for cyclic networks [3]–[7], but all these code-construction algorithms share one major drawback; they all need to know in advance the network topology. In particular, if the network is large, it might be difficult to learn the exact network structure.

A randomized linear network coding approach was presented by Ho *et al.* [8]. They showed that for a cyclic network, all sink nodes will be able, with high probability, to decode the symbols sent by the source nodes, provided that the transmission rates of all sources satisfy the Min-Cut Max-Flow condition and that the LEK are chosen randomly from a large enough field. The Min-Cut Max-Flow condition states that for every subset $\mathcal{A}$ of source nodes, the sum of source rates $\sum_{s\in\mathcal{A}} R_s$ must be less than or equal to the minimum cut capacity between every sink node and $\mathcal{A}$.

This result makes randomized CNC extremely useful when the network is dynamic and no central authority for assigning encoding kernels exists. The LEK can be chosen randomly from some large enough field and, with high probability, this will lead to a network that enables source nodes to transmit symbols at high rates, thereby enabling all sink nodes to decode the sent symbols. This outcome, however, requires that the source nodes know which rates are achievable and that the sink nodes know their GEK, which are needed for a decoding algorithm. If the network structure or the LEK are not known, an initialization process is needed.

In this paper, we present two initialization algorithms that find the GEK of the network, and one algorithm that finds achievable rates from the GEK. The GEK are found by sending test basis vectors and obtaining the impulse response of the network, a method analogous to the one given in [9, Ch. 19.3.2] for acyclic networks. A direct implementation of the algorithm that is designed for acyclic networks is problematic, because of two assumptions that do not hold: first, the response of the network to an input vector is a single output vector (in the acyclic delay-free case), while in cyclic networks with delays the response is an infinite sequence of vectors (the impulse response). Second, in acyclic delay-free networks every output vector depends only on the currently sent input vector, while in cyclic networks with delays, the impulse response for the current input vector interferes with the responses to previous input vectors. Although the impulse response of the network can be of infinite duration, our algorithms find the GEK using only the initial values of the impulse response. In the first algorithm, we transmit basis vectors and obtain the impulse response of the network under the assumption that the initial symbols sent on the network are zeros. In the second algorithm, we require neither that the initial symbols are zeros nor that it is possible to clear all these symbols at once. Our algorithms do not require the transmission of any additional headers. This simplifies the

Manuscript received January 9, 2017; revised March 8, 2018; accepted March 26, 2018. Date of publication April 19, 2018; date of current version June 20, 2018. M. Lvov and H. H. Permuter were supported in part by the European Research Council, European Unions Seventh Framework Program (FP7/2007-2013)/ERC under Grant 337752, in part by ISF, and in part by DIP. This paper was presented in part at the 2014 International Symposium on Network Coding, Aalborg, Denmark.

The authors are with the Department of Electrical and Computer Engineering, Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel (e-mail: maxlvo55@gmail.com; haimp@bgu.ac.il).

Communicated by A. Ramamoorthy, Associate Editor for Coding Techniques.

Digital Object Identifier 10.1109/TIT.2018.2828432

design of the relay nodes, since they do not operate differently during and after the initialization process. The method for finding achievable rates is based on the fact that the connection between the source and the sink nodes is possible if the GEK matrix is of full column rank [3].

There are two main decoding algorithms for CNC: the time-variant decoding algorithm presented by Guo *et al.* [10] and the sequential decoder presented by Erez and Feder [7]. The time-variant decoding algorithm allows the sink nodes to decode the sent message $\mathbf{u}[n]$ using the previously decoded messages $\{\mathbf{u}[k]\}_{k=0}^{n-1}$, the received vectors $\{\mathbf{y}[k]\}_{k=0}^{n+\delta}$ (where δ is the decoding delay) and the first GEK terms $\{\mathbf{F}[k]\}_{k=0}^{n+\delta}$ (the first terms in the power series expansion of the GEK). The decodability of the network (whether the GEK matrix is of full column rank) can be checked using the first $\delta+1$ terms of the GEK, as stated in [11]. In this way, only the first terms of the GEK are needed (these can be transmitted in the headers of the messages) in order to decode the transmitted symbol each time. In [12], Guo *et al.* present a randomized construction of a CNC for an unknown network and use the time-variant decoding algorithm, as the full GEK are not known to the sink nodes. The sequential decoder, on the contrary, requires the knowledge of the full GEK for a decoding algorithm. The full GEK are used to construct a difference equation between the transmitted and the received symbols, from which the transmitted symbols can be decoded. Unlike the time-variant decoding algorithm, the sequential decoder has a decoding complexity which does not grow with time, and it does not need to use headers, as the full GEK are known ahead of time. Our initialization algorithms can be used to find the full GEK, and then the sequential decoder can be used. The time variant decoding algorithm can also benefit from the knowledge of the full GEK, as the GEK terms no longer need to be transmitted in the headers.

The paper is divided into eight sections and two appendices. In Section II we outline notations and define the problem. In Section III we present two algorithms for finding GEK and one algorithm for finding achievable rates of the network. In Sections IV, V and VI we expand on how each algorithm works, one algorithm per section. In Section VII we generalize the algorithms for CNC with rational LEK. In Section VIII we show that decoding using a sequential decoder after applying our algorithms achieves the minimal decoding delay. Section IX concludes the paper. In Appendix A we give the proofs for all the theorems and lemmas. In Appendix B we demonstrate the algorithm implementation with examples.

II. Notations and Problem Definition

We represent a communication network by a directed graph $G = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of edges. Each edge represents a noiseless directed link that can transmit one symbol per unit time, where the symbols are scalars from some field $\mathbb{F}$. We assume that every link has unit time delay between consequent symbol transmissions (note, we will abandon this assumption in Section VII) and that transmissions on all links are synchronized.

As in [8], we assume a multi-source multi-cast scenario in which a set of source nodes transmits information to a set of

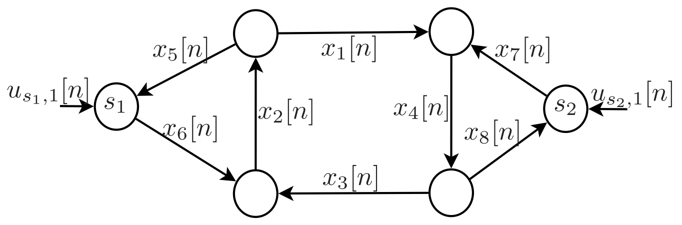

Fig. 1. Shuttle network with two users and four relay nodes.

sink nodes. We denote by $\mathcal{S}$ the set of all source nodes and by $\mathcal{D}$ the set of all sink nodes. Every source node $s \in \mathcal{S}$ transmits R_s symbols per unit time. Every sink node wants to receive all the symbols sent by all the source nodes. For every edge $e \in \mathcal{E}$, we say that $u = head(e)$ and $v = tail(e)$ if $u, v \in \mathcal{V}$ and e is from v to u. We denote by $In(u) = \{e \in \mathcal{E} : u = head(e)\}$ and $Out(u) = \{e \in \mathcal{E} : u = tail(e)\}$. The symbol that is sent on the edge e at time $n \in \mathbb{Z}$ is denoted by $x_e[n]$. We define $\mathbf{x}[n]$ to be the *state vector of the network*, which is a column vector of size $|\mathcal{E}|$ consisting of all symbols $\{x_e[n]\}_{e\in\mathcal{E}}$ organized in some order. We denote vectors or sequences of vectors by lowercase bold letters, while matrices are denoted by bold capital letters. For a sequence of vectors $\{\mathbf{c}[n]\}_{n\in\mathbb{Z}}$ we denote by $(\mathbf{c}[n])_j$ the j'th component of a vector $\mathbf{c}[n]$, and by $\mathbf{c}$ the sequence itself. We define the standard basis for the vector space $\mathbb{F}^\omega$ as $\{\mathbf{e}_k\}_{k=1}^{\omega}$, namely, the components of the vector $\mathbf{e}_k$ are zeros except for the k'th component, which is equal to one. We define the indicator function $1_{\{\cdot\}}$ to be

$$1_\Omega = \begin{cases} 1, & \text{statement } \Omega \text{ is true} \\ 0, & \text{otherwise.} \end{cases}$$

Each source node $s \in \mathcal{S}$ generates R_s messages per unit time $\mathbf{u}_s[n] = \left(u_{s,1}[n], \ldots, u_{s,R_s}[n]\right)^T$, where each message $u_{s,k}[n]$ is a scalar from the field $\mathbb{F}$. We call $\mathbf{u}_s$ the *input sequence of source s*. We define the *input sequence* $\mathbf{u}[n] = \left(\mathbf{u}_{s_1}^T[n], \ldots, \mathbf{u}_{s_{|\mathcal{S}|}}^T[n]\right)^T$. The dimension of the column vector $\mathbf{u}[n]$ is $\omega = \sum_{s\in\mathcal{S}} R_s$.

For every sink node d, we let $\mathbf{y}_d[n]$ be a column vector consisting of all received symbols $\{x_e[n] : e \in In(d)\}$ and the symbols generated by d,$\{u_{d,k}[n]\}_{k=1}^{R_d}$ if d is also a source node, again organized in some order. The sequence $\left\{\mathbf{y}_d[n]\right\}_{n\in\mathbb{Z}}$ will be called the *output sequence* of the sink node d, and the dimension of every vector in that sequence is $l_d = R_d + |In(d)|$. We assume also that $\mathbf{x}[n], \mathbf{u}[n]$ and $\mathbf{y}_d[n]$ are equal to zero for $n < 0$.

Example 1: Consider the shuttle network shown in Fig. 1. The nodes s_1, s_2 are both source and sink nodes, and have the same transmission rates $R_{s_1} = R_{s_2} = 1$. The state vector is $\mathbf{x}[n] = (x_1[n], x_2[n], \ldots, x_8[n])^T$, the input sequence is $\mathbf{u}[n] = \left(u_{s_1,1}[n], u_{s_2,1}[n]\right)^T$ $(\omega = 2)$, and the output sequences are $\mathbf{y}_{s_1}[n] = \left(x_5[n], u_{s_1,1}[n]\right)^T$ and $\mathbf{y}_{s_2}[n] = \left(x_8[n], u_{s_2,1}[n]\right)^T$. Both l_{s_1} and l_{s_2} are equal to 2. The *virtual* links with $u_{s_1,1}$ and $u_{s_2,1}$ do not exist in the real network, and they represent the inputs of the network.

For a sequence of scalars $\{c[n]\}_{n\in\mathbb{Z}}$ (with only a finite number of non zero terms for $n < 0$) we denote by $c(z)$ its formal Laurent series representation $c(z) = \sum_{n=-\infty}^{\infty} c[n]z^n$. If $c[n] = 0$ for $n < 0$, this becomes a formal power series representation. We define the formal Laurent series of a vector sequence in the same way. A power series is called a rational power series if $c(z)$ can be expanded by long division from a rational function $c(z) = q(z)/(1 + zp(z))$, where q and p are polynomials. The n'th term in the Laurent series expansion of $c(z)$ is denoted by $(c(z))[n]$. For instance, the n'th term of $(z^k c(z))$ is $(z^k c(z))[n] = c[n-k]$. For that reason, z can also be thought of as a delay operator acting on a sequence $\{c[n]\}$. For a rational function $H(z) = q(z)/(1 + zp(z))$, where $q(z) = \sum_{k=0}^{N_q} q_k z^k$ and $p(z) = \sum_{k=0}^{N_p} p_k z^k$ are polynomials with coefficients from the field $\mathbb{F}$, and a sequence of scalars $\{c[n]\}_{n\in\mathbb{Z}}$ that satisfies $c[n] = 0$ for $n < 0$, the n'th term in the Laurent series expansion of $w(z) = H(z)c(z)$ is found iteratively from

$$w[n] = -\sum_{k=0}^{N_p} p_k w[n-k-1] + \sum_{k=0}^{N_q} q_k c[n-k], \tag{1}$$

with $w[n] = 0$ for $n < 0$. The coefficients $\{q_k\}$ can also be $m \times k$ matrices over the field $\mathbb{F}$ and, in that case, the sequence $\{\mathbf{c}[n]\}_{n\in\mathbb{Z}}$ should be a sequence of $k \times 1$ vectors and the sequence $\{\mathbf{w}[n]\}_{n\in\mathbb{Z}}$ should be a sequence of $m \times 1$ vectors. Two formal Laurent series $a(z)$ and $b(z)$ are equal if $(a(z))[n] = (b(z))[n]$ for every n. We say that $a(z) = b(z)$ for $n > N_0$ if $(a(z))[n] = (b(z))[n]$ for $n > N_0$. We refer the reader to [9, Ch. 20.2] for further treatment of formal power series.

We assume there is a CNC scheme in the network, so that the symbol sent on a link $i \in Out(j)$ is a linear combination of the symbols received and generated by the node j in the previous time slot. This relationship can be written as:

$$x_i[n+1] = \sum_{e\in In(j)} a_{i,e} x_e[n] + \sum_{k=1}^{R_j} b_{i,k} u_{j,k}[n], \quad \forall i \in \mathcal{E},\ \forall n \geq 0, \tag{2}$$

where $u_{j,k}[n]$ is the k'th symbol generated by node j (if $j \in \mathcal{S}$) at time n, and $\{a_{i,e}, b_{i,k}\}$ are the LEK for node j that were chosen in advance (probably randomly) from the field $\mathbb{F}$. By letting $x_i[n]$ depend only on the previously received symbols, we avoid the problem described in [13] by Cai and Guo, when the CNC may not be well defined in a cyclic network. Note that the symbols transmitted by every source s_i on its output links are not its input sequence $\mathbf{u}_{s_i}$ but rather linear combinations of both $\mathbf{u}_{s_i}$ and of the symbols received by s_i. We define the matrices $\mathbf{A}$ of size $|\mathcal{E}| \times |\mathcal{E}|$ and $\mathbf{B}$ of size $|\mathcal{E}| \times \omega$ as $\mathbf{A} = [a_{i,j}]_{i,j\in\mathcal{E}}$ and $\mathbf{B} = [b_{i,j}]_{i\in\mathcal{E}, 1\leq j\leq\omega}$, respectively, where $a_{i,j}$ is the LEK from link j to link i, and $b_{i,j}$ is the LEK from $(\mathbf{u})_j$ to link i. For every sink node d we define the matrices $\mathbf{C}_d$ of size $l_d \times |\mathcal{E}|$ and $\mathbf{D}_d$ of size $l_d \times \omega$ by $(\mathbf{C}_d)_{i,j} = 1$ if $(\mathbf{y}_d)_i$ corresponds to x_j and zero otherwise, and $(\mathbf{D}_d)_{i,j} = 1$ if $(\mathbf{y}_d)_i$ corresponds to $(\mathbf{u})_j$ and zero otherwise. With these definitions, the network input-output relationship can be written using state equations, as was introduced by Fragouli and Soljanin [14]:

$$\mathbf{x}[n+1] = \mathbf{A}\mathbf{x}[n] + \mathbf{B}\mathbf{u}[n], \quad \mathbf{x}[0] = \mathbf{x}_0,\ \forall n \geq 0, \tag{3}$$

$$\mathbf{y}_d[n] = \mathbf{C}_d\mathbf{x}[n] + \mathbf{D}_d\mathbf{u}[n], \quad \forall n \in \mathbb{Z}. \tag{4}$$

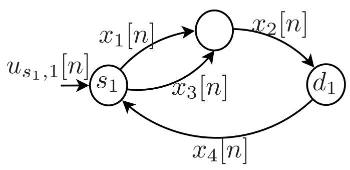

Fig. 2. Network with one source node, one sink node and one relay node.

If the network has a reset option that clears all the symbols, we can assume that $\mathbf{x}_0 = \mathbf{0}$. The GEK matrix for a sink node d is given in [14] by $\mathbf{H}_d(z) = (z/P(z))\,\mathbf{C}_d adj\,(\mathbf{I} - z\mathbf{A})\,\mathbf{B} + \mathbf{D}_d$, where $P(z) = \det(\mathbf{I} - z\mathbf{A})$. The input-output relationship can also be written using formal Laurent series representation:

$$\mathbf{y}_d(z) = \mathbf{H}_d(z)\mathbf{u}(z) + \frac{1}{P(z)}\mathbf{C}_d adj\,(\mathbf{I} - z\mathbf{A})\mathbf{x}_0. \tag{5}$$

Example 2: Consider the network in Fig.2 with one source node s_1 and one sink node d_1. By the Min-Cut Max-Flow Theorem the rate $R_{s_1} = 1$ is achievable and the network state equations are:

$$\mathbf{x}[n+1] = \begin{pmatrix} x_1[n+1] \\ x_2[n+1] \\ x_3[n+1] \\ x_4[n+1] \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & \alpha_{1,4} \\ \alpha_{2,1} & 0 & \alpha_{2,3} & 0 \\ 0 & 0 & 0 & \alpha_{3,4} \\ 0 & \alpha_{4,2} & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1[n] \\ x_2[n] \\ x_3[n] \\ x_4[n] \end{pmatrix} + \begin{pmatrix} b_{1,1} \\ 0 \\ b_{3,1} \\ 0 \end{pmatrix} u_{s_1}[n], \tag{6}$$

$$y_{d_1}[n] = \begin{pmatrix} 0 & 1 & 0 & 0 \end{pmatrix} \mathbf{x}[n] + 0 \cdot u_{s_1}[n]. \tag{7}$$

Note that we have restricted ourselves to the case where LEK are scalars, while in the general case they can be rational power series in the delay operator z [9, Ch. 20.2]. This is not a major restriction, since one can achieve the highest achievable rates without rational LEK if the field one works with is large enough [9, Ch. 20.3]. Nevertheless, we treat network codes with rational power series LEK separately in Section VII.

We are interested in finding the GEK matrix for each sink node, which is needed to decode the sent symbols $\{\mathbf{u}[n]\}$ from the received symbols $\{\mathbf{y}_d[n]\}$. The GEK matrices are obtained in our algorithms by sending test symbols and obtaining the impulse response of the network. Our algorithms neither need to know the network topology nor the local encoding kernels, but just the following parameters of the network:

- The set of source nodes $\mathcal{S}$ and upper bounds for their transmission rates $\{R_s\}_{s\in\mathcal{S}}$.
- An upper bound for the number of edges in the network, which will be called N.

Before the initialization process starts, a transmission rate for every source node should be chosen. If an achievable rate for a specific source node is not known, it is preferable to set its rate to $R_s = |Out(s)|$. After the initialization process ends and achievable rates are found, they should be somehow delivered from the sink to the source nodes. Every source node s can then reduce its rate R_s by sending zeros on some of its input sequences. The delivery of achievable rates from the sink to the source nodes requires a low-capacity communication between the sink and the source nodes, and it is required both in cyclic and in delay-free acyclic networks.

We define the rates $\left(R'_s\right)_{s\in\mathcal{S}}$ to be *achievable for a sink node d with specific LEK* if that sink node can decode the input sequence $\mathbf{u}$ from the output $\mathbf{y}_d$ with the current LEK when every source node s transmits symbols on R'_s out of its input sequences, and zeros on the rest of the $(R_s - R'_s)$ input sequences. We define the rates $\left(R'_s\right)_{s\in\mathcal{S}}$ to be *achievable with specific LEK* if they are achievable for every sink node separately. Note that if the rates $\left(R'_s\right)_{s\in\mathcal{S}}$ are achievable by the Min-Cut Max-Flow condition and the LEK were chosen randomly from a large enough field then for every sink node these rates will also be, with high probability, achievable by our definition and, hence, our definition does not "miss" any achievable rates. Moreover, if the rates $\left(R'_s\right)_{s\in\mathcal{S}}$ are achievable for every sink node separately then by the Min-Cut Max-Flow theorem they are also achievable simultaneously. Therefore, our definition of achievable rates with specific LEK coincides, with high probability, with the definition by the Min-Cut Max-Flow condition.

It may happen, however, that the LEK were not chosen well, and some rates are achievable for different sink nodes separately but not simultaneously (e.g. if a specific rate is achievable for sink nodes d_1 and d_2, but each sink node requires other inputs to be set to zero). The assignment of zero to some inputs of a source node s is equivalent to sending the input sequence $\mathbf{u}_s = \mathbf{B}_s\mathbf{u}_s^{new}$, where $\mathbf{u}_s^{new}$ is the new input sequence of the source s (with the reduced rate), and $\mathbf{B}_s$ is a $R_s \times R'_s$ matrix for which $[\mathbf{B}_s]_{i,j}$ is equal to 1 if $\left(\mathbf{u}_s^{new}\right)_j$ corresponds to $(\mathbf{u}_s)_i$, and zero otherwise. However, instead of deciding which inputs should be set to zero, every source s can generate the matrix coefficients $[\mathbf{B}_s]_{i,j}$ randomly (and independently) from the field $\mathbb{F}$. This will assure that, with high probability, every sink node will be able to decode $\mathbf{u}_s^{new}$ provided that $\mathbb{F}$ is large, a fact that is based on [8, Th. 2]. This theorem states that if there exist LEK for the entire network that allow for k sink nodes to decode $\mathbf{u}$ from their output sequences, then by randomizing some of the LEK those sink nodes will still be able to decode $\mathbf{u}$ with a probability of at least $(1 - k/|\mathbb{F}|)^\eta$, where η is the number of links associated with random coefficients. After updating the rates, an initialization process should be run again so that the sink nodes obtain the new GEK.

III. The Initialization Algorithms

In this section, we present two initialization algorithms that find the GEK for the sink nodes, which are needed for decoding, and one algorithm that finds achievable rates. The goal of the first two is to find a difference equation of the following form:

$$P_d(z)\mathbf{y}_d(z) = \mathbf{G}_d(z)\mathbf{u}(z). \tag{8}$$

This form describes the relationship between the transmitted sequence $\mathbf{u}[n]$ and the received sequence $\mathbf{y}_d[n]$ (for every sink node d). The GEK of a sink node d are $\mathbf{G}_d(z)/P_d(z)$. Using a decoding method similar to the sequential decoder given in [7], we can show that it is possible to decode the input sequence from the output when the polynomial $P_d(z)$ is not the zero polynomial and the *transfer matrix* $\mathbf{G}_d(z)$ is of full column rank over the polynomial ring $\mathbb{F}[z]$.

In the first algorithm it is assumed that we can clear all the symbols on the network at some fixed times and, therefore, this algorithm is a bit faster and easier to analyze than the second algorithm that does not operate under this assumption. The objective of the third algorithm is to find achievable rates for all source nodes. This is done by examining the transfer matrix $\mathbf{G}_d(z)$ for every sink node d. To obtain this matrix, one of the initialization algorithms should be used first.

We now present the first algorithm. Its first part consists of ω loops. Every loop takes $2N + 1$ time units, and after each loop the symbols on all edges are cleared. Algorithm 1 is applied in Example 16 in Appendix B.

Algorithm 1 Initialization Algorithm With Network Resetting

```
1) For i=1 to $\omega$ do:
    • Transmit the input sequence $\mathbf{u}[n] = \begin{cases}\mathbf{e}_i, & \text{if } n = 0\\ \mathbf{0}, & \text{if } 1 \le n \le 2N\end{cases}$. In order to send this sequence,
      the source node that corresponds to the current iteration should send the symbol 1 on one of its
      inputs at time $n = 0$, and zeros at all other times and on all other inputs. All other sources
      should send only zeros during that iteration.
    • Every sink node $d$ should store its received vectors $\left\{\mathbf{y}_d^i[n]\right\}_{n\in\{0,..,2N\},i\in\{1,...,\omega\}}$,
      where the vectors $\{\mathbf{y}_d^i[n]\}_{n=0}^{2N}$ are the output sequence at that iteration.
    • Reset the network after $n = 2N$, by setting $n = 0$ and $\mathbf{x}[0] = \mathbf{0}$.
2) For every sink node $d$ do the following:
    • Combine the received vectors into matrices: $\mathbf{M}_d[n] = \left[\mathbf{y}_d^1[n], \ldots, \mathbf{y}_d^\omega[n]\right]$.
    • Find any non-trivial solution to the system of linear equations
          $$\sum_{k=0}^{N}\alpha_{d,k}\mathbf{M}_d[k+\tau] = \mathbf{O}, \quad \forall\tau = 1, \ldots, N, \tag{9}$$
      where $\mathbf{O}$ is the zero matrix and $\{\alpha_{d,k}\}_{k=0}^N \subseteq \mathbb{F}$ are the unknowns.
      This system has $l_d \times \omega \times N$ equations and, as will be explained in Section IV,
      it always has a non trivial solution.
    • For the found solution $\{\alpha_{d,k}\}_{k=0}^N$, let $N_d = \max\{k : \alpha_{d,k} \neq 0\}$.
3) Construct the polynomial $P_d(z)$ and the matrix $\mathbf{G}_d(z)$ as
          $$P_d(z) = \sum_{k=0}^{N_d}\alpha_{d,k}z^{N_d-k}, \tag{10}$$
          $$\mathbf{G}_d(z) = \sum_{k=0}^{N_d-1}\sum_{j=k+1}^{N_d}\alpha_{d,j}\mathbf{M}_d[j-k]z^{N_d-k} + \mathbf{M}_d[0]P_d(z). \tag{11}$$
```

Algorithm 1 *(Continued.)* Initialization Algorithm With Network Resetting

4) The difference equation that describes the relationship between the input and the output sequences $\mathbf{u}[n]$ and $\mathbf{y}_d[n]$ is given in (8), with the polynomial $P_d(z)$ and the matrix $\mathbf{G}_d(z)$ as defined in (10)-(11). If $\mathbf{G}_d(z)$ is of full column rank over the polynomial ring $\mathbb{F}[z]$, then $\mathbf{u}[n]$ can be decoded from $\mathbf{y}_d[n]$ by solving (8). Otherwise, either the transmission rates $\{R_s\}$ of some source nodes should be reduced, or other LEK should be chosen (or both).

We now present the second algorithm, in which no resetting operation is needed. We consider the case when the network initial state is $\mathbf{x}[0] = \mathbf{x}_0$, where $\mathbf{x}_0$ can be arbitrary and unknown. Algorithm 2 is needed if the network we are using is not under our control (except for the source and the sink nodes) and, hence, we cannot send a reset command to all nodes, or if the relay nodes do not have a reset option at all because of cost/performance considerations. Algorithm 2 is similar to Algorithm 1, except that this algorithm takes an additional $2N+1$ time units and the expression for obtaining $\mathbf{G}_d(z)$ is a bit different. If, nevertheless, we consider the case with $\mathbf{x}_0 = \mathbf{0}$, then we can skip the operations in the first $(2N+1)$ time units since the output vectors will contain only zeros. After applying Algorithm 2, (8) will hold only for $n > N_d$, where N_d will have been defined in Algorithm 2. Nevertheless, (8) enables the decoding of the input sequence. Algorithm 2 is applied in Example 18 in Appendix B.

Algorithm 2 Initialization Algorithm Without Network Resetting

1) Transmit the input sequence $\mathbf{u}[n] = \sum_{k=1}^{\omega} \mathbf{e}_k 1_{\{n=(2N+1)k\}}$, for $0 \le n < (\omega+1)(2N+1)$. Note that in order to send that sequence, every source node $s \in \mathcal{S}$ should send the symbol 1 on each of every one of its inputs in turn $(u_{s,1}, \ldots, u_{s,R_s})$ at the correct time, and zeros at all other times. Every sink node should store its received vectors $\{\mathbf{y}_d[n]\}_{0 \le n < (\omega+1)(2N+1)}$, where each vector $\mathbf{y}_d[n]$ is of dimension l_d.
2) For every sink node d do the following:
   - Find any non trivial solution to the system of linear equations
$$\sum_{j=0}^{N} \alpha_{d,j} \mathbf{y}_d[j+\tau] = \mathbf{0}, \quad \forall \tau \in \bigcup_{p=0}^{\omega} \bigcup_{\tilde{\tau}=1}^{N} \{(2N+1)p + \tilde{\tau}\}, \tag{12}$$
   where $\{\alpha_{d,j}\}_{j=0}^{N} \subseteq \mathbb{F}$ are the unknowns. This system has $l_d \times N \times (\omega+1)$ equations and, as will be explained in Section V, it always has a non trivial solution.
   - For the found solution $\{\alpha_{d,k}\}_{k=0}^{N}$, let $N_d = \max\{k : \alpha_{d,k} \neq 0\}$.

Algorithm 2 *(Continued.)* Initialization Algorithm Without Network Resetting

3) The polynomial $P_d(z)$ and the matrix $\mathbf{G}_d(z)$ are defined as:
$$P_d(z) = \sum_{k=0}^{N_d} \alpha_{d,k} z^{N_d - k}, \tag{13}$$
$$\mathbf{g}_{d,i}(z) = \sum_{k=0}^{N_d} \sum_{j=0}^{N_d} \alpha_{d,j} \mathbf{y}_d[j + (2N+1)i - k] z^{N_d - k}, \tag{14}$$
$$\mathbf{G}_d(z) = \left[\mathbf{g}_{d,1}(z), \mathbf{g}_{d,2}(z), \ldots, \mathbf{g}_{d,\omega}(z)\right]. \tag{15}$$
4) The difference equation that describes the relationship between the input and the output sequences $\mathbf{u}[n]$ and $\mathbf{y}_d[n]$ is given in (8), with the polynomial $P_d(z)$ and the matrix $\mathbf{G}_d(z)$ as defined in (13)-(15). The equation holds for terms with $n > N_d$. If $\mathbf{G}_d(z)$ is of full column rank over the polynomial ring $\mathbb{F}[z]$, then $\mathbf{u}[n]$ can be decoded from $\mathbf{y}_d[n]$ by solving (8). Otherwise, either the transmission rates $\{R_s\}$ of some source nodes should be reduced, or other LEK should be chosen (or both).

The complexity analyses of Algorithms 1 and 2 are as follows: There are $(2N+1)\omega$ transmissions in Algorithm 1 and $(2N+1)(\omega+1)$ transmissions in Algorithm 2. Next, there is the solution of linear equations which have $n_1 = N+1$ unknowns and n_2 equations, where $n_2 = l_d \times \omega \times N$ in Algorithm 1 and $n_2 = l_d \times (\omega+1) \times N$ in Algorithm 2. If the system of equations is solved with Gaussian elimination, it has a complexity of $O(n_1^2 n_2)$.

We now present the third algorithm that enables us to find achievable rates with the chosen LEK. It uses the transfer matrix $\mathbf{G}_d(z)$ from (8) and, hence, Algorithm 1 or 2 should be used first to find the matrix. At the end of this algorithm, every sink node d will be able to tell what rates are achievable for it.

Algorithm 3 Finding Achievable Rates

- For every sink node d, split the matrix $\mathbf{G}_d(z)$ that was obtained in Algorithm 1 or 2 into $|\mathcal{S}|$ matrices, such that each matrix $\mathbf{G}_{d,s}(z)$ has the R_s columns of the matrix $\mathbf{G}_d(z)$ that correspond to $\mathbf{u}_s$ and such that the following will hold:
$$\mathbf{G}_d(z)\mathbf{u}(z) = \left[\mathbf{G}_{d,s_1}(z), \ldots, \mathbf{G}_{d,s_{|\mathcal{S}|}}(z)\right] \begin{bmatrix} \mathbf{u}_{s_1}(z) \\ \vdots \\ \mathbf{u}_{s_{|\mathcal{S}|}}(z) \end{bmatrix} = \sum_{s \in \mathcal{S}} \mathbf{G}_{d,s}(z)\mathbf{u}_s(z). \tag{16}$$
- For every possible $|\mathcal{S}|$-tuple $(R'_s)_{s\in\mathcal{S}}$ with integer entries that satisfy $R'_s \le R_s$, check if for every source node s there exist R'_s column vectors $\left\{\mathbf{v}_{s,1}, \ldots, \mathbf{v}_{s,R'_s}\right\}$ in the columns of the matrix $\mathbf{G}_{d,s}(z)$ such that all the vectors $\cup_{s\in\mathcal{S}} \cup_{k=1}^{R'_s} \{\mathbf{v}_{s,k}\}$ are linearly independent over the polynomial ring $\mathbb{F}[z]$. If there are such vectors, the rates $(R'_s)_{s\in\mathcal{S}}$ are achievable for the sink node d.

Algorithm 3 enables us to find achievable rates with the currently chosen LEK. It is applied in Example 17 in Appendix B. An upper bound on the complexity of Algorithm 3 can be derived as follows: to find out if rates $(R'_s)_{s\in\mathcal{S}}$ are achievable, we need to go over the columns of the matrices $\mathbf{G}_{d,s}(z)$, in every matrix to choose R'_s out of R_s column vectors (there are $\binom{R_s}{R'_s}$ combinations in every matrix), and to check if the vectors are linearly independent. There are $\omega' = \sum_{s\in\mathcal{S}} R'_s$ vectors, each vector of length l_d. If $l_d < \omega'$ then the vectors are linearly dependent. If $l_d = \omega'$, then to check for their independence we need to compute a determinant of a $\omega' \times \omega'$ matrix; each element has a polynomial of degree N_d, where $N_d \le N$. The complexity in computing this determinant depends on the algorithm for determinant computation, but, in the worst case, with Gaussian elimination, it can be $O(\omega'^5 N_d^2)$ (see [15]). If $l_d > \omega'$, then we need to delete $l_d - \omega'$ rows from this matrix (there are $\binom{l_d}{\omega'}$ combinations for this choice), then to compute its determinant, and if it is not zero then the vectors are linearly independent. If it is zero, we can choose different vectors from the matrices $\mathbf{G}_{d,s}(z)$ and/or remove different rows from the final matrix. In the worst case, we will perform $\binom{l_d}{\omega'}\prod_{s\in\mathcal{S}}\binom{R_s}{R'_s}$ determinant computations to verify if the rates are achievable. However, there is a much faster "probabilistic" way which requires only one determinant computation: we can multiply the matrices $\mathbf{G}_{d,s}(z)$ from the right by a random matrix $\mathbf{B}_s$ and from the left by a random matrix $\mathbf{Q}_s$. The components of these matrices should be chosen randomly from the field $\mathbb{F}$. The dimensions of these matrices should be: $(R_s \times R'_s)$ for $\mathbf{B}_s$ and $(\omega' \times l_d)$ for $\mathbf{Q}_s$. Then, there is only one way of choosing the vectors from the matrices $\{\mathbf{Q}_s\mathbf{G}_{d,s}(z)\mathbf{B}_s\}_{s\in\mathcal{S}}$ (which is to take all its vectors), and there is only one determinant that should be computed. Note that if the rates $(R'_s)_{s\in\mathcal{S}}$ are achievable, then choosing specific vectors from the matrix $\mathbf{G}_{d,s}(z)$ is equivalent to multiplying it from the right by a binary matrix $\mathbf{B}_s$, and deleting specific rows from the matrix is equivalent to multiplying it by a binary matrix $\mathbf{Q}_s$ from the left. However, if the matrices $\mathbf{B}_s$ and $\mathbf{Q}_s$ are random, the rates will still be achievable, with high probability, if the field is large enough. This follows from [8, Th. 2] which we stated at the end of Section II.

IV. Derivation of Algorithm 1

Our goal is to find a non zero polynomial $P_d(z)$ and a matrix $\mathbf{G}_d(z)$ such that the difference equation (8), from which it is possible to decode $\mathbf{u}$ from $\mathbf{y}_d$, will hold. We assume, without loss of generality, that N is equal to the number of edges in the network. However, if N is larger, we can assume that there are an additional $2(N - |\mathcal{E}|)$ virtual nodes and $(N - |\mathcal{E}|)$ virtual edges between these nodes. The virtual edges are not connected to the original network and have no influence on it. The number of edges in the new network is N.

The input-output relationship of the network is given by the state equations (3)-(4). Their general solution is:

$$\mathbf{y}_d[n] = 1_{\{n\ge 0\}}\mathbf{C}_d\mathbf{A}^n\mathbf{x}_0 + \sum_{k=-\infty}^{n-1}\mathbf{C}_d\mathbf{A}^{n-1-k}\mathbf{B}\mathbf{u}[k] + \mathbf{D}_d\mathbf{u}[n], \quad \forall n\in\mathbb{Z}, \tag{17}$$

where the initial state is $\mathbf{x}_0 = \mathbf{0}$. Note that $\mathbf{u}[n]$ and $\mathbf{y}_d[n]$ vanish for $n < 0$ and, therefore, (17) also holds for $n < 0$ and the sum in (17) is finite. The first part of the algorithm consists of ω loops and in each loop i the input sequence is

$$\mathbf{u}[n] = \begin{cases}\mathbf{e}_i, & n = 0\\ \mathbf{0}, & 1 \le n \le 2N.\end{cases} \tag{18}$$

The output sequence of every sink node is given in the following lemma.

Lemma 3: Let the input sequence $\boldsymbol{u}[n]$ be as given in (18). The output sequence in that case will be

$$\boldsymbol{y}_d^i[n] = \begin{cases}\boldsymbol{D}_d\boldsymbol{e}_i, & n = 0\\ \boldsymbol{C}_d\boldsymbol{A}^{n-1}\boldsymbol{B}\boldsymbol{e}_i, & 1 \le n \le 2N.\end{cases} \tag{19}$$

Moreover, if one combines the output vectors into matrices, as is done in Algorithm 1, $\boldsymbol{M}_d[n] = \left[\boldsymbol{y}_d^1[n], \ldots, \boldsymbol{y}_d^{\omega}[n]\right]$, then the corresponding matrices will be

$$\begin{aligned}\boldsymbol{M}_d[n] &= \boldsymbol{C}_d\boldsymbol{A}^{n-1}\boldsymbol{B}, \quad \forall 1 \le n \le 2N,\\ \boldsymbol{M}_d[0] &= \boldsymbol{D}_d.\end{aligned} \tag{20}$$

Proof: Follows immediately by substituting the input sequence from (18) into the general solution given in (17) and using the fact that the initial state $\mathbf{x}_0$ is zero. □

We now state the Cayley-Hamilton Theorem [16, p. 284], since it plays an important role in our derivation.

Theorem 4 (Cayley-Hamilton Theorem): For a given $n \times n$ matrix $\boldsymbol{A}$ over the field $\mathbb{F}$, let $P_{\boldsymbol{A}}(t) = \det(t\boldsymbol{I} - \boldsymbol{A})$ be the characteristic polynomial of $\boldsymbol{A}$. Let $\{a_k\}_{k=0}^{n-1}$ be the coefficients of $P_{\boldsymbol{A}}(t)$, namely $P_{\boldsymbol{A}}(t) = t^n + \sum_{k=0}^{n-1} a_k t^k$. Then the following holds:

$$P_{\boldsymbol{A}}(\boldsymbol{A}) = \boldsymbol{A}^n + \sum_{k=0}^{n-1} a_k\boldsymbol{A}^k = \boldsymbol{O}, \tag{21}$$

where $\boldsymbol{O}$ is the zero $n \times n$ matrix.

In the second step of the algorithm we find a non trivial solution $\{\alpha_{d,k}\}_{k=0}^N$ to the system of linear equations (9). If we substitute $\mathbf{M}_d$ from (20) into (9) we get:

$$\begin{aligned}\forall\tau\in\{1,\ldots,N\}: \quad \mathbf{O} &= \sum_{k=0}^{N}\alpha_{d,k}\mathbf{M}_d[k+\tau] &(22)\\ &= \sum_{k=0}^{N}\alpha_{d,k}\mathbf{C}_d\mathbf{A}^{k+\tau-1}\mathbf{B} &(23)\\ &= \mathbf{C}_d\tilde{P}_d(\mathbf{A})\mathbf{A}^{\tau-1}\mathbf{B}, &(24)\end{aligned}$$

where $\tilde{P}_d(t) = \sum_{k=0}^{N}\alpha_{d,k}t^k$. We see, therefore, that solving (9) is equivalent to finding a non zero polynomial $\tilde{P}_d(t)$ of degree N or less that satisfies $\mathbf{C}_d\tilde{P}_d(\mathbf{A})\mathbf{A}^{\tau-1}\mathbf{B} = 0$ for $\tau \in \{1,\ldots,N\}$. Using the Cayley-Hamilton theorem, this system of equations has at least one non trivial solution, where $\tilde{P}_d(t)$ is the characteristic polynomial of $\mathbf{A}$. The next lemma shows that $\mathbf{C}_d\tilde{P}_d(\mathbf{A})\mathbf{A}^{\tau-1}\mathbf{B} = 0$ holds also for $\tau > N$ if it holds for $\tau \in \{1,\ldots,N\}$. Its proof is given in Appendix A and is based on the Cayley-Hamilton Theorem.

Lemma 5: If a polynomial $\tilde{P}_d(t)$ satisfies

$$\boldsymbol{C}_d \tilde{P}_d(\boldsymbol{A})\boldsymbol{A}^\tau \boldsymbol{B} = \boldsymbol{O}, \quad \forall \tau \in \{0, 1, \ldots, N-1\}, \tag{25}$$

where $\boldsymbol{A}$ is a square $N \times N$ matrix, then it also satisfies

$$\boldsymbol{C}_d \tilde{P}_d(\boldsymbol{A})\boldsymbol{A}^\tau \boldsymbol{B} = \boldsymbol{O}, \quad \forall \tau \in \mathbb{N}. \tag{26}$$

We now define a new polynomial $P_d(z) = z^{N_d}\tilde{P}_d(z^{-1})$, where N_d is the degree of $\tilde{P}_d(z)$. Using (17) and Lemma 5 we show in Theorem 6 that $\left(P_d(z)\mathbf{y}_d(z)\right)[n]$ is a linear combination, with matrix coefficients, of only $\{\mathbf{u}[n-k]\}_{k=0}^{N_d}$ and it does not include outcomes from earlier times; hence, a difference equation between $\mathbf{u}$ and $\mathbf{y}_d$ can be constructed.

Theorem 6: For a given sink node d, let $P_d(z) = z^{N_d}\tilde{P}_d(z^{-1})$ and $\mathbf{G}_d(z)$ be the polynomial and the matrix defined in (10)-(11); then (8) holds. Furthermore, it is possible to decode $\boldsymbol{u}$ from $\mathbf{y}_d$ if and only if the transfer matrix $\mathbf{G}_d(z)$ is of full column rank over the polynomial ring $\mathbb{F}[z]$.

Theorem 6, which is proved in Appendix A, gives us a way to decode the transmitted symbols and it also assures us that if the system of linear equations in (8) does not have a unique solution, then there is no way for us to find $\mathbf{u}$ from $\mathbf{y}_d$, even if we know the network topology and the LEK.

V. Derivation of Algorithm 2

We are interested, again, in a difference equation between $\mathbf{u}$ and $\mathbf{y}_d$, as given in (8), that does not depend on $\mathbf{x}_0$. As described in step 1 of Algorithm 2, the input sequence $\mathbf{u}[n]$ is given by

$$\mathbf{u}[n] = \sum_{k=1}^{\omega} \mathbf{e}_k 1_{\{n=(2N+1)k\}}. \tag{27}$$

We substitute this input sequence into the general solution of the network's state equations (17) to get the output sequence for every sink node d:

$$\begin{aligned}
\mathbf{y}_d[n] &= 1_{\{n\geq 0\}}\mathbf{C}_d\mathbf{A}^n\mathbf{x}_0 \\
&\quad + \sum_{k=1}^{\omega}\sum_{i=-\infty}^{n-1} \mathbf{C}_d\mathbf{A}^{n-1-i}\mathbf{B}\mathbf{e}_k 1_{\{i=(2N+1)k\}} \\
&\quad + \mathbf{D}_d \sum_{k=1}^{\omega} \mathbf{e}_k 1_{\{n=(2N+1)k\}} \qquad (28)\\
&= 1_{\{n\geq 0\}}\mathbf{C}_d\mathbf{A}^n\mathbf{x}_0 + \sum_{k=1}^{\min\left\{\omega, \left\lfloor \frac{n-1}{2N+1} \right\rfloor\right\}} \mathbf{C}_d\mathbf{A}^{n-1-(2N+1)k}\mathbf{B}\mathbf{e}_k \\
&\quad + \mathbf{D}_d \sum_{k=1}^{\omega} \mathbf{e}_k 1_{\{n=(2N+1)k\}}. \qquad (29)
\end{aligned}$$

In the second step of the algorithm we find a non trivial solution to the system of linear equations (12). Denote by $\tilde{P}_d(t) = \sum_{k=0}^{N}\alpha_{d,k}t^k$ a polynomial whose coefficients are a solution of (12), its degree by N_d and by $P_d(t) = t^{N_d}\tilde{P}_d(t^{-1})$. The next theorem shows that $\tilde{P}_d(t)$ has important properties that will be used further. Its proof is outlined in Appendix A.

Theorem 7: A polynomial $\tilde{P}_d(t) = \sum_{k=0}^{N}\alpha_{d,k}t^k$ satisfies

$$\boldsymbol{C}_d \tilde{P}_d(\boldsymbol{A})\boldsymbol{A}^\tau \boldsymbol{B} = \boldsymbol{O}, \quad \forall \tau \in \{0, \ldots, N-1\}, \tag{30}$$

$$\boldsymbol{C}_d \tilde{P}_d(\boldsymbol{A})\boldsymbol{A}^{\tau+1} \boldsymbol{x}_0 = \boldsymbol{0}, \quad \forall \tau \in \{0, \ldots, N-1\} \tag{31}$$

if and only if its coefficients are a solution of (12).

In (30)-(31) $\mathbf{O}$ is the $l_d \times \omega$ zero matrix and $\mathbf{0}$ is the zero column vector of dimension l_d. We see that there is at least one non trivial solution to (12) where $\tilde{P}_d(t)$ is the characteristic polynomial of $\mathbf{A}$. The next lemma shows that (30)-(31) are also satisfied for $\tau \geq N$.

Lemma 8: Let $\tilde{P}_d(t)$ be a polynomial and $\boldsymbol{A}$ a square $N \times N$ matrix. If either of the equations (30) or (31) hold for $\tau \in \{0, \ldots, N-1\}$, then it also holds for all $\tau \geq N$.

The proof is similar to the proof for Lemma 5 and is therefore omitted.

Using (17) and Lemma 8 we show in Theorem 9 that $\left(P_d(z)\mathbf{y}_d(z)\right)[n]$ is a linear combination, with matrix coefficients, of $\{\mathbf{u}[n-k]\}_{k=0}^{N_d}$ and not of $\mathbf{u}[k]$ for earlier times and of $\mathbf{x}_0$; thus, a difference equation between $\mathbf{u}$ and $\mathbf{y}_d$ can be constructed. The equation will hold for any time after the initialization process ends, even without resetting the state vector. The proof for Theorem 9 is outlined in Appendix A.

Theorem 9: Let $P_d(z)$ and $\mathbf{G}_d(z)$ be, respectively, the polynomial and the matrix defined in (13)-(15). Then the following difference equation holds:

$$\left(P_d(z)\boldsymbol{y}_d\right)[n] = \left(\mathbf{G}_d(z)\boldsymbol{u}\right)[n], \quad \forall n > N_d. \tag{32}$$

Furthermore, it is possible to decode $\boldsymbol{u}$ from $\mathbf{y}_d$ if and only if the transfer matrix $\mathbf{G}_d(z)$ is of full column rank over the polynomial ring $\mathbb{F}[z]$.

VI. Derivation of Algorithm 3

A direct consequence of Theorems 6 and 9 is the fact that we can find achievable rates for every source node from the matrices $\{\mathbf{G}_d(z)\}_{d\in\mathcal{D}}$. If the transmission rates are not achievable with the given LEK, then at least one sink node d cannot decode the input sequence $\mathbf{u}$ from the output $\mathbf{y}_d$ and, therefore, the transfer matrix $\mathbf{G}_d(z)$ is not of full column rank. In Algorithm 3 we examine the columns of the transfer matrices $\mathbf{G}_d(z)$ to see if some columns can be removed so that $\mathbf{G}_d(z)$ will be of full rank. If it is possible, the resulting rates (after the columns' removal) are achievable. This result is stated in the following theorem, the proof of which is outlined in Appendix A.

Theorem 10: For a given network and a sink node d, let

$$P_d(z)\boldsymbol{y}_d(z) = \boldsymbol{G}_d(z)\boldsymbol{u}(z) = \sum_{s\in\mathcal{S}} \boldsymbol{G}_{d,s}(z)\boldsymbol{u}_s(z) \tag{33}$$

describe the relationship between the input sequence $\boldsymbol{u}$ and the output sequence $\boldsymbol{y}_d$ that was found in Algorithm 1 or 2. For every source node $s \in \mathcal{S}$, let R_s be the transmission rate of s that was set before an initialization algorithm was started. Then, the rates $(R'_s)_{s\in\mathcal{S}}$ are achievable for the sink node d with the current LEK if and only if for every source node $s \in \mathcal{S}$ there exist R'_s linearly independent column vectors $\boldsymbol{v}_{s,1}, \ldots, \boldsymbol{v}_{s,R_s}$ from the columns of the matrix $\boldsymbol{G}_{d,s}(z)$, such that $\cup_{s\in\mathcal{S}} \cup_{k=1}^{R'_s} \{\boldsymbol{v}_{s,k}\}$ is a set of linearly independent vectors over the polynomial ring $\mathbb{F}[z]$.

VII. Generalization to Rational Encoding Kernels

Although we restricted ourselves to the case of scalar LEK, the above algorithms can be extended to networks that use CNC with rational power series as LEK [9, Ch. 20.2]. For each node $j \in \mathcal{V}$ denote by $\mathbf{x}_{out}^{j}[n]$ the vector of symbols sent on the output links of node j at time n, and by $\mathbf{x}_{in}^{j}[n]$ the vector of symbols received by node j at time n. The components of $\mathbf{x}_{out}^{j}[n]$ are some of the components of $\mathbf{x}[n]$, while the components of $\mathbf{x}_{in}^{j}[n]$ can be components of both $\mathbf{x}[n]$ and $\mathbf{u}[n]$ (if $j \in \mathcal{S}$). In case the LEK are rational power series, the relationship between the input and the output of each node j can be written as:

$$(1+q_j(z))\mathbf{x}_{out}^{j}(z) = \mathbf{H}_j(z)\mathbf{x}_{in}^{j}(z), \tag{34}$$

where $q_j(z) = \sum_{k=1}^{M_{j1}} q_j[k]z^k$ is a polynomial and $\mathbf{H}_j(z) = \sum_{k=0}^{M_{j2}} \mathbf{H}_j[k]z^k$ is a matrix of polynomials. We say that the i'th output link of node j has scalar LEK with delay $k \in \{0,1,2,\ldots\}$ if the i'th row of the matrix $\left(\mathbf{H}_j(z)/\left(1+q_j(z)\right)\right)$ is of the form of $\mathbf{a}z^k$, where $\mathbf{a}$ is a row vector of scalars from the field $\mathbb{F}$.

The input-output relationship of every difference equation, and in particular of (34), can also be described using state equations:

$$\tilde{\mathbf{x}}_j[n+1] = \tilde{\mathbf{A}}_j\tilde{\mathbf{x}}_j[n] + \tilde{\mathbf{B}}_j\mathbf{x}_{in}^{j}[n], \quad \tilde{\mathbf{x}}_j[0] = \tilde{\mathbf{x}}_{j,0}, \ \forall n \geq 0, \tag{35}$$

$$\mathbf{x}_{out}^{j} = \tilde{\mathbf{C}}_j\tilde{\mathbf{x}}_j[n] + \tilde{\mathbf{D}}_j\mathbf{x}_{in}^{j}[n], \quad \forall n \in \mathbb{Z}, \tag{36}$$

where the matrices $\tilde{\mathbf{A}}_j, \tilde{\mathbf{B}}_j, \tilde{\mathbf{C}}_j$ and $\tilde{\mathbf{D}}_j$ are determined by the LEK. We refer the reader to [17, Ch. 3.1] to find possible ways of constructing state equations from a difference equation. A construction of state equations from a difference equation is called a *state space realization*. We call $\tilde{\mathbf{x}}_j$ the *inner state vector* of node j because it is used solely to describe the input-output relationship of node j.

Example 11: Consider a node j with two input and two output links with the following LEK:

$$\mathbf{x}_{out}^{j}(z) = \begin{bmatrix} \frac{z}{1-z}, & \frac{1}{1-z} \\ \alpha + z, & 0 \end{bmatrix} \mathbf{x}_{in}^{j}(z). \tag{37}$$

We find a state space realization for these LEK. First, we write a difference equation for the input-output relationship of the node:

$$x_{out,1}^{j}[n] = x_{out,1}^{j}[n-1] + x_{in,1}^{j}[n-1] + x_{in,2}^{j}[n], \tag{38}$$

$$x_{out,2}^{j}[n] = \alpha x_{in,1}^{j}[n] + x_{in,1}^{j}[n-1]. \tag{39}$$

where $x_{out,k}^{j}$ and $x_{in,k}^{j}$ are the k'th components of $\mathbf{x}_{out}^{j}$ and $\mathbf{x}_{in}^{j}$ respectively. We take the state vector $\tilde{\mathbf{x}}_j[n] = \left(x_{out,1}^{j}[n-1], x_{in,1}^{j}[n-1]\right)^T$. With this state vector the following state equations hold:

$$\tilde{\mathbf{x}}_j[n+1] = \begin{pmatrix} x_{out,1}^{j}[n] \\ x_{in,1}^{j}[n] \end{pmatrix} \tag{40}$$

$$= \begin{pmatrix} x_{out,1}^{j}[n-1] + x_{in,1}^{j}[n-1] + x_{in,2}^{j}[n] \\ x_{in,1}^{j}[n] \end{pmatrix} \tag{41}$$

$$= \begin{pmatrix} 1, & 1 \\ 0, & 0 \end{pmatrix} \tilde{\mathbf{x}}_j[n] + \begin{pmatrix} 0, & 1 \\ 1, & 0 \end{pmatrix} \mathbf{x}_{in}^{j}[n], \quad \forall n \geq 0, \tag{42}$$

$$\mathbf{x}_{out}^{j}[n] = \begin{pmatrix} 1, & 1 \\ 0, & 1 \end{pmatrix} \tilde{\mathbf{x}}_j[n] + \begin{pmatrix} 0, & 1 \\ \alpha, & 0 \end{pmatrix} \mathbf{x}_{in}^{j}[n]. \tag{43}$$

For each node $j \in \mathcal{V}$ denote the state vector of node j in its minimal realization (with the smallest dimension state vector) by $\tilde{\mathbf{x}}_j[n]$, and its dimension by $\dim(\tilde{\mathbf{x}}_j[n])$. If we concatenate all state vectors $\{\tilde{\mathbf{x}}_j[n]\}_{j\in\mathcal{V}}$ into one state vector $\tilde{\mathbf{x}}[n]$ of dimension $\sum_{j\in\mathcal{V}} \dim(\tilde{\mathbf{x}}_j[n])$, a global state space representation of the network can be written:

$$\tilde{\mathbf{x}}[n+1] = \hat{\mathbf{A}}\tilde{\mathbf{x}}[n] + \hat{\mathbf{B}}\mathbf{u}[n], \quad \tilde{\mathbf{x}}[0] = \tilde{\mathbf{x}}_0, \ \forall n \geq 0, \tag{44}$$

$$\mathbf{y}_d[n] = \hat{\mathbf{C}}_d\tilde{\mathbf{x}}[n] + \hat{\mathbf{D}}_d\mathbf{u}[n], \quad \forall n \in \mathbb{Z}, \tag{45}$$

where $\hat{\mathbf{A}}$, $\hat{\mathbf{B}}$, $\hat{\mathbf{C}}_d$ and $\hat{\mathbf{D}}_d$ are defined by the network topology and the LEK. The derivation of our algorithms is based only on the fact that the input-output relationship of the network can be described by state equations with a state vector of dimension $|\mathcal{E}|$, which is less than or equal to N. In the case where we use rational power series as LEK, the algorithms will still apply if we take N to be larger that the dimension of the state vector $\tilde{\mathbf{x}}$:

$$N \geq \sum_{j\in\mathcal{V}} \dim(\tilde{\mathbf{x}}_j[n]). \tag{46}$$

In the first algorithm the state vectors of all nodes $\tilde{\mathbf{x}}_j$ should be cleared after every $2N$ transmissions. Note that to calculate the right hand side of (46), one needs additional knowledge of the network, such as an upper bound for the dimension $\dim(\tilde{\mathbf{x}}_j[n])$ for every node. This can be found if we have an upper bound for the degree of the numerator and the denominator of the rational-function-LEK, because a state-space realization can be constructed with a state vector of a dimension which is equal to the maximal degree of the numerator and the denominator of the LEK multiplied by $|Out(j)|$, as shown in [17, Ch. 3.1]. Note also that N can be smaller than the number of edges in the network, if $\dim(\tilde{\mathbf{x}}_j[n]) \leq |Out(j)|$ for some nodes $j \in \mathcal{V}$. For example, if a node j has zero delay on all of its links, it will not have a state vector at all since its input-output relationship $\mathbf{x}_{out}^{j} = \mathbf{H}_j[0]\mathbf{x}_{in}^{j}$ does not require one. In that case, this node will not contribute to N since $\dim(\tilde{\mathbf{x}}_j[n]) = 0$. If a node j has k output links with unit time delay and all other links with zero delay, it will have a state vector of dimension not larger than k. This is stated in the following theorem, the proof for which is outlined in Appendix A.

Theorem 12: If a node v has scalar LEK, k output links with unit time delay and $|Out(v)| - k$ output links with zero delay, then its input-output relationship can be described by state equations with a state vector of dimension not larger than k.

According to Li and Yeung [4], it is sufficient for a network to have unit time delay on only one link of every cycle for a network code to be well defined. If that is the case and all LEK are scalars, then it is enough to take N to be equal to the number of links with unit time delay, which is not more than the number of cycles in the network. We therefore have the following result:

Corollary 13: If a network $(\mathcal{V}, \mathcal{E})$ has a CNC scheme with scalar LEK and it has k links with unit time delay and $|\mathcal{E}| - k$ links without delay, then Algorithms 1 and 2 will give the correct GEK, provided that $N \geq k$.

The proof for Corollary 13 is outlined in Appendix A. We apply Algorithm 2 for the shuttle network with non-scalar LEK in Example 18 in Appendix B.

The upper bound for N that we gave in (46) is not tight, and often a much smaller value of N can be taken, but this requires some additional knowledge of the network. If, for example, we can somehow know that the GEK of every sink node has its maximum numerator degree not larger than N_1 and its denominator degree not larger than N_2, and l_d is not larger than L, then for every sink node a state-space realization can be constructed with a state vector of a dimension not larger than $\max(N_1, N_2)L$. In this case, we can take N equal to that upper bound, and the initialization algorithms will still work with this value of N.

VIII. Decoding Delay

At the end of the proofs of Theorems 6 and 9 we gave a decoding algorithm (which is a small modification of the sequential decoder presented in [7]) that can be used after applying Algorithms 1 and 2. In this section we show that the decoding delay that we achieve is optimal, in the sense that no other decoding algorithm can achieve a shorter decoding delay. As a consequence, the decoding delay of our algorithms does not depend on the specific solution that we choose when we solve the linear equations (9) or (12).

We briefly recap the decoding algorithm; its full details can be found in the proofs of Theorems 6 and 9. First, we compute $\mathbf{q}(z)$ from (71) or (109), and then we use the identity $\mathbf{q}(z) = \mathbf{G}_d(z)\mathbf{u}(z)$ (where $\mathbf{G}_d(z)$ is a square matrix of polynomials, and it is of full column rank), and multiply both sides by $adj\,(\mathbf{G}_d(z))$ to get:

$$adj\,(\mathbf{G}_d(z))\,\mathbf{q}(z) = \det(\mathbf{G}_d(z))\,\mathbf{u}(z) \tag{47}$$

For convenience, we write:

$$adj\,(\mathbf{G}_d(z)) = z^g \sum_{k=0}^{\mathbf{W}_{max}} \mathbf{W}[k]z^k, \quad g \geq 0,\ \mathbf{W}[0] \neq \mathbf{O}, \tag{48}$$

$$\det(\mathbf{G}_d(z)) = z^\tau \beta \left(1 + \sum_{i=1}^{K} \gamma_i z^i\right), \quad \tau \geq 0,\ \beta \neq 0. \tag{49}$$

We substitute these forms of $adj\,(\mathbf{G}_d(z))$ and $\det(\mathbf{G}_d(z)$ into (47), and get:

$$\mathbf{u}[n-\tau] = \left(\beta^{-1} \sum_{k=0}^{\mathbf{W}_{max}} \mathbf{W}[k]\mathbf{q}[n-k-g]\right) - \sum_{i=1}^{K} \gamma_i \mathbf{u}[n-i-\tau] \tag{50}$$

The decoding delay to decode $\mathbf{u}$ from $\mathbf{q}$ in this algorithm is $\delta_c = \min(0, \tau - g)$. Recall that $\mathbf{q}[n]$ is defined (for all $n \geq 0$ in Algorithm 1 and for $n > N_d$ in Algorithm 2) by:

$$\mathbf{q}[n] = \left(P_d(z)\mathbf{y}_d(z)\right)[n] \tag{51}$$

$$= \sum_{k=0}^{N_d} \alpha_{d,k}\mathbf{y}_d[n - N_d + k], \tag{52}$$

and $\alpha_{d,N_d} \neq 0$. This means that we can find $\{\mathbf{q}[k]\}_{k\leq n}$ from $\{\mathbf{y}_d[k]\}_{k\leq n}$ and vice versa. Therefore, δ_c is also the decoding delay of our algorithm to decode $\mathbf{u}$ from $\mathbf{y}_d$.

Denote by $\delta_{\min}$ the minimal decoding delay (in decoding $\mathbf{u}$ from $\mathbf{y}_d$ or from $\mathbf{q}$) achievable by any decoding algorithm. From the definition, we have $\delta_{\min} \leq \delta_c$. We will show that $\delta_{\min} \geq \delta_c$, and this will prove the optimality of our decoding delay.

Without loss of generality and for convenience, we decode $\mathbf{u}$ from $\mathbf{q}$ instead of from $\mathbf{y}_d$. We explained above why this does not change the optimal decoding delay. From the definition of $\delta_{\min}$, it is possible to obtain $\mathbf{u}[0]$ from $\{\mathbf{q}[k]\}_{k=0}^{\delta_{\min}}$. Because the relationship between $\mathbf{u}(z)$ and $\mathbf{q}(z)$ is linear, and because the sequence $\{\mathbf{u}[n]\}_{n\geq 0}$ can be arbitrary, we can assume, without loss of generality, that the expression for $\mathbf{u}[0]$ from $\{\mathbf{q}[k]\}_{k=0}^{\delta_{\min}}$ will also be linear, and it is obtained by simply solving linear equations:

$$\begin{bmatrix} \mathbf{G}_d[0] & \mathbf{O} & \mathbf{O} & \dots & \mathbf{O} \\ \mathbf{G}_d[1] & \mathbf{G}_d[0] & \mathbf{O} & \dots & \mathbf{O} \\ \mathbf{G}_d[2] & \mathbf{G}_d[1] & \mathbf{G}_d[0] & \dots & \mathbf{O} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{G}_d[\delta_{\min}] & \mathbf{G}_d[\delta_{\min}-1] & \mathbf{G}_d[\delta_{\min}-2] & \dots & \mathbf{G}_d[0] \end{bmatrix} \times \begin{bmatrix} \mathbf{u}[0] \\ \mathbf{u}[1] \\ \mathbf{u}[2] \\ \vdots \\ \mathbf{u}[\delta_{\min}] \end{bmatrix} = \begin{bmatrix} \mathbf{q}[0] \\ \mathbf{q}[1] \\ \mathbf{q}[2] \\ \vdots \\ \mathbf{q}[\delta_{\min}] \end{bmatrix} \tag{53}$$

$$\mathbf{u}[0] = \sum_{k=0}^{\delta_{\min}} \mathbf{T}_k \mathbf{q}[k] \tag{54}$$

Now, instead of talking about some "optimal" decoding algorithm, we will limit ourselves to a rational-power-series decoder that decodes $\mathbf{u}$ from $\mathbf{q}$ with a decoding delay $\delta_{\min}$. The following lemma shows that such a decoder exists:

Lemma 14: There exists a rational power series $\boldsymbol{P}(z) = \sum_{k=0}^{\infty} \boldsymbol{P}[k]z^k$ that satisfies:

$$z^{\delta_{\min}}\boldsymbol{u}(z) = \boldsymbol{P}(z)\boldsymbol{q}(z) \tag{55}$$

The proof for this lemma is in Appendix A. In the next theorem, we prove that $\delta_{\min} \geq \delta_c$. For that, we write the equality $\mathbf{G}_d^{-1}(z) = z^{-\delta_{\min}}\mathbf{P}(z)$, and then we show that if $\delta_{\min} < \delta_c$ then $\mathbf{P}(z)$ will no longer be a rational power series, and it will contain negative powers of z in the expansion.

Theorem 15: The following identity holds: $\delta_{\min} \geq \delta_c$

The proof for this theorem is in Appendix A.

IX. Summary

The use of CNC schemes requires the choice of LEK at the relay nodes that would enable the sink nodes to decode the transmitted symbols. The coefficients can be chosen randomly to simplify the network code construction, but this would make it necessary for the sink nodes to know the GEK of the network. The algorithms we presented in this paper enable the sink nodes to find their GEK that facilitates decoding the transmitted from the received symbols without learning the exact topology of the network and the chosen LEK. Achievable rates can also be found from the obtained GEK. The algorithms require the source nodes to transmit basis vectors and the sink nodes to solve a system of linear equations. Although there might be multiple solutions to the linear equations, every solution is suitable for a decoding algorithm, and the decoding delay does not depend on the specific solution that is chosen. The first stage in our algorithms requires the transmission of test vectors in the first $(2N + 1)\omega$ (for Algorithm 1) or $(2N+1)(\omega+1)$ (for Algorithm 2) time units, where N is proportional to the size of the network (number of edges or nodes). This number can be quite large for very large networks, but if the number of transmitted symbols is very large (compared to N), the time of the initialization process will be negligible. Future work can check how N can be decreased for all networks or for some special topologies of networks, or how it can depend on additional knowledge of the network, such as the number of disjoint paths from the source to the sink nodes, the length of the largest cycle in the network, the length of the longest path from every source to every sink node, the maximal accumulated delay along every cycle in the network or along every path from every source to every sink node, etc. Another direction for future work is to check whether it is possible to decrease N or the decoding delay by choosing an appropriate field $\mathbb{F}$ or an appropriate LEK (scalar or rational power series).

APPENDIX A
PROOFS

Proof for Lemma 5: A direct consequence of the Cayley-Hamilton Theorem is that for every $N \times N$ matrix $\mathbf{A}$, its power $\mathbf{A}^\tau$ can be written as a linear combination of $\mathbf{I}, \mathbf{A}, \mathbf{A}^2, \ldots, \mathbf{A}^{N-1}$ for $\tau \geq N$. By substituting this linear combination into (26) we get for every $\tau \geq N$:

$$\mathbf{C}_d \tilde{P}_d(\mathbf{A})\mathbf{A}^\tau \mathbf{B} = \mathbf{C}_d \tilde{P}_d(\mathbf{A}) \left(\sum_{i=0}^{N-1} \gamma_i \mathbf{A}^i \right) \mathbf{B} \tag{56}$$

$$= \sum_{i=0}^{N-1} \gamma_i \left(\mathbf{C}_d \tilde{P}_d(\mathbf{A}) \mathbf{A}^i \mathbf{B} \right) \tag{57}$$

$$= \mathbf{O}, \tag{58}$$

where the last equality holds because $\tilde{P}_d(z)$ satisfies (25). □

To prove Theorem 6 we first use the state equations general solution (17) to show that $\left(P_d(z)\mathbf{y}_d(z)\right)[n]$ is a finite linear combination, with matrix coefficients, of $\{\mathbf{u}[n-k]\}_{k=0}^{\infty}$. We split this linear combination into two sums, the first contains the terms $\{\mathbf{u}[n-k]\}_{k=N_d+1}^{\infty}$ and the second contains $\{\mathbf{u}[n-k]\}_{k=0}^{N_d}$. We use Lemma 5 to show that the first sum vanishes, and we use the definition of $\mathbf{G}_d(z)$ from (11) to show that the second sum is equal to $(\mathbf{G}_d(z)\mathbf{u}(z))\,[n]$.

To prove the second part of the theorem we give a decoding scheme to find $\mathbf{u}$ from $\mathbf{y}_d$ if $\mathbf{G}_d(z)$ is of full rank. If it is not, we show that there are two different input sequences $\mathbf{u}[n]$ that can lead to the same output sequence $\mathbf{y}_d[n]$ and, therefore, decoding in that case is not possible as one cannot be sure which of the input sequences was transmitted.

Proof for Theorem 6: We examine the sequence $\left(P_d(z)\mathbf{y}_d(z) - P_d(z)\mathbf{D}_d\mathbf{u}(z)\right)[n]$:

$$\left(P_d(z)\left(\mathbf{y}_d(z) - \mathbf{D}_d\mathbf{u}(z)\right)\right)[n] \tag{59}$$

$$\overset{(a)}{=} \left(\sum_{j=0}^{N_d} \alpha_{d,j} z^{N_d-j} \left(\mathbf{y}_d(z) - \mathbf{D}_d\mathbf{u}(z)\right) \right)[n] \tag{60}$$

$$= \sum_{j=0}^{N_d} \alpha_{d,j} \left(\mathbf{y}_d[n+j-N_d] - \mathbf{D}_d\mathbf{u}[n+j-N_d]\right) \tag{61}$$

$$\overset{(b)}{=} \sum_{j=0}^{N_d} \sum_{i=-\infty}^{n+j-N_d-1} \alpha_{d,j}\mathbf{C}_d\mathbf{A}^{n+j-i-N_d-1}\mathbf{B}\mathbf{u}[i] \tag{62}$$

$$\overset{(c)}{=} \sum_{i=-\infty}^{n-N_d-1} \sum_{j=0}^{N_d} \alpha_{d,j}\mathbf{C}_d\mathbf{A}^{n+j-i-N_d-1}\mathbf{B}\mathbf{u}[i] + \sum_{i=n-N_d}^{n-1} \sum_{j=i-n+N_d+1}^{N_d} \alpha_{d,j}\mathbf{C}_d\mathbf{A}^{n+j-i-N_d-1}\mathbf{B}\mathbf{u}[i] \tag{63}$$

$$= \sum_{i=-\infty}^{n-N_d-1} \mathbf{C}_d \left(\sum_{j=0}^{N_d} \alpha_{d,j}\mathbf{A}^j \right) \mathbf{A}^{n-i-N_d-1}\mathbf{B}\mathbf{u}[i] + \sum_{i=n-N_d}^{n-1} \left(\sum_{j=i-n+N_d+1}^{N_d} \alpha_{d,j}\mathbf{C}_d\mathbf{A}^{n+j-i-N_d-1}\mathbf{B} \right)\mathbf{u}[i]$$

$$\overset{(d)}{=} \sum_{i=-\infty}^{n-N_d-1} \mathbf{C}_d\tilde{P}_d(\mathbf{A})\mathbf{A}^{n-i-N_d-1}\mathbf{B}\mathbf{u}[i] + \sum_{k=0}^{N_d-1} \left(\sum_{j=k+1}^{N_d} \alpha_{d,j}\mathbf{C}_d\mathbf{A}^{j-k-1}\mathbf{B} \right)\mathbf{u}[n+k-N_d]$$

$$\overset{(e)}{=} 0 + \sum_{k=0}^{N_d-1} \left(\sum_{j=k+1}^{N_d} \alpha_{d,j}\mathbf{C}_d\mathbf{A}^{j-k-1}\mathbf{B} \right)\mathbf{u}[n+k-N_d] \tag{64}$$

$$= \left(\sum_{k=0}^{N_d-1} \left(\sum_{j=k+1}^{N_d} \alpha_{d,j}\mathbf{C}_d\mathbf{A}^{j-k-1}\mathbf{B} \right) z^{N_d-k}\mathbf{u}(z) \right)[n], \tag{65}$$

where

(a) is obtained by substituting $P_d(z)$ from (10),
(b) is obtained by substituting $\mathbf{y}_d$ from (17) and using the fact that $\mathbf{x}_0 = \mathbf{0}$,
(c) is obtained by changing the summation order,
(d) is obtained by changing a summation variable in the second sum to $k = i - n + N_d$,
(e) follows from Lemma 5, as $\tilde{P}_d(z)$ satisfies

$$\mathbf{C}_d\tilde{P}_d(\mathbf{A})\mathbf{A}^\tau\mathbf{B} = \mathbf{O}, \quad \forall \tau \in \mathbb{N}. \tag{66}$$

Therefore, by Lemma 3 and by (11) we get:

$$P_d(z)\mathbf{y}_d = \left(\sum_{k=0}^{N_d-1} \left(\sum_{j=k+1}^{N_d} \alpha_{d,j}\mathbf{C}_d\mathbf{A}^{j-k-1}\mathbf{B} \right) z^{N_d-k} + P_d(z)\mathbf{D}_d \right)\mathbf{u} \tag{67}$$

$$= \mathbf{G}_d(z)\mathbf{u}. \tag{68}$$

We now prove the second part of the theorem that states that it is possible to decode the input sequence from the output if and only if the matrix $\mathbf{G}_d(z)$ is of full column rank over the polynomial ring $\mathbb{F}[z]$. If it is not, there exists a non zero vector of polynomials $\mathbf{v}_d(z)$ such that $\mathbf{G}_d(z)\mathbf{v}_d(z) = \mathbf{0}$. Let $\mathbf{u}(z) = \mathbf{v}_d(z)$, where $\mathbf{u}(z)$ is the formal Laurent series of the input sequence. As $\mathbf{v}_d(z)$ is a vector of polynomials, the input sequence satisfies $\mathbf{u}[n] = 0$ for $n < 0$ and, hence, this is a legal input sequence. Since $P_d(z)$ is not the zero polynomial and can be written as

$$P_d(z) = \alpha_{d,N_d} + zq_d(z), \tag{69}$$

where $q_d(z)$ is a polynomial and $\alpha_{d,N_d} \neq 0$, we can divide (8) by $P_d(z)$ to get $\mathbf{y}_d(z)$:

$$\mathbf{y}_d(z) = \frac{1}{P_d(z)}\mathbf{G}_d(z)\mathbf{u}(z) = \mathbf{0}. \tag{70}$$

In this case, the output sequence $\mathbf{y}_d[n]$ will vanish, and no decoding method will tell if the input sequence was $\mathbf{u}[n]$ or a zero sequence.

On the other hand, if $\mathbf{G}_d(z)$ is of full column rank then we can show that the input sequence can be decoded from the output sequence. We apply a decoding scheme that is slightly different to that of the sequential decoder [7]. We modify the decoding scheme to adapt it to our notations that use the polynomial $P_d(z)$ instead of $\det(\mathbf{I} - z\mathbf{A})$. We multiply both sides of (8) by $adj(\mathbf{G}_d(z))$ (we can assume that $\mathbf{G}_d(z)$ is a square matrix, since, if it is not, we can remove some of its linearly dependent rows to make it square) to get

$$\begin{aligned}
\mathbf{q}(z) &:= P_d(z)\mathbf{y}_d(z) && (71)\\
\mathbf{w}(z) &:= adj(\mathbf{G}_d(z))\mathbf{q}(z) && (72)\\
&\overset{(a)}{=} adj(\mathbf{G}_d(z))\mathbf{G}_d(z)\mathbf{u}(z) && (73)\\
&\overset{(b)}{=} \det(\mathbf{G}_d(z))\mathbf{u}(z) && (74)
\end{aligned}$$

where

(a) follows from (8),

(b) follows from the fact that for any square matrix $\mathbf{G}$ over the polynomial ring $\mathbb{F}[z]$, the following identity holds: $adj(\mathbf{G})\mathbf{G} = \det(\mathbf{G})\mathbf{I}$, where $\mathbf{I}$ denotes the identity matrix.

The polynomial $\det(\mathbf{G}_d(z))$ is non zero, since we assumed that $\mathbf{G}_d(z)$ is of full column rank and, therefore, it is of the form:

$$\det(\mathbf{G}_d(z)) = z^{\tau}\beta\left(1 + \sum_{i=1}^{k}\gamma_i z^i\right), \quad \tau \geq 0,\ \beta \neq 0.$$

If we know the sequence $\mathbf{y}_d[n]$ we can compute $\mathbf{w}[n]$ and from that find $\mathbf{u}[n]$:

$$\mathbf{w}[n+\tau] = \beta\left(\mathbf{u}[n] + \sum_{i=1}^{k}\gamma_i\mathbf{u}[n-i]\right), \tag{75}$$

$$\mathbf{u}[n] = (\beta)^{-1}\mathbf{w}[n+\tau] - \sum_{i=1}^{k}\gamma_i\mathbf{u}[n-i]. \tag{76}$$

□

Proof for Theorem 7: We first prove that if $\tilde{P}_d(t) = \sum_{j=0}^{N}\alpha_{d,j}z^j$ satisfies (30)-(31), then its coefficients are a solution of (12). Afterwards, we prove the same in the other direction. First, we remind that the output sequence in Algorithm 2 is given by:

$$\mathbf{y}_d[n] = 1_{\{n\geq 0\}}\mathbf{C}_d\mathbf{A}^n\mathbf{x}_0 + \sum_{k=1}^{\min\left\{\omega,\left\lfloor\frac{n-1}{2N+1}\right\rfloor\right\}}\mathbf{C}_d\mathbf{A}^{n-1-(2N+1)k}\mathbf{B}\mathbf{e}_k + \mathbf{D}_d\sum_{k=1}^{\omega}\mathbf{e}_k 1_{\{n=(2N+1)k\}}. \tag{77}$$

To prove that (12) holds we substitute this expression for $\mathbf{y}_d[n]$ into (12) and make use of (30)-(31). For every $\tau \in \bigcup_{p=0}^{\omega}\bigcup_{\tilde{\tau}=1}^{N}\{(2N+1)p + \tilde{\tau}\}$ we have:

$$\begin{aligned}
&\sum_{j=0}^{N}\alpha_{d,j}\mathbf{y}_d[j+\tau] && (78)\\
&\overset{(a)}{=} \sum_{j=0}^{N}\alpha_{d,j}\mathbf{C}_d\mathbf{A}^{j+\tau}\mathbf{x}_0 \\
&\quad + \sum_{j=0}^{N}\sum_{k=1}^{\min\left\{\omega,\left\lfloor\frac{j+\tau-1}{2N+1}\right\rfloor\right\}}\alpha_{d,j}\mathbf{C}_d\mathbf{A}^{j+\tau-1-(2N+1)k}\mathbf{B}\mathbf{e}_k \\
&\quad + \sum_{j=0}^{N}\sum_{k=1}^{\omega}\alpha_{d,j}\mathbf{D}_d\mathbf{e}_k 1_{\{j+\tau=(2N+1)k\}} && (79)\\
&\overset{(b)}{=} \mathbf{C}_d\tilde{P}_d(\mathbf{A})\mathbf{A}^{\tau}\mathbf{x}_0 + \sum_{j=0}^{N}\sum_{k=1}^{\lfloor\tau/(2N+1)\rfloor}\alpha_{d,j}\mathbf{C}_d\mathbf{A}^{j+\tau-1-(2N+1)k}\mathbf{B}\mathbf{e}_k \\
&\quad + \sum_{j=0}^{N}\sum_{k=1}^{\omega}\alpha_{d,j}\mathbf{D}_d\mathbf{e}_k 1_{\{j+\tau=(2N+1)k\}} && (80)\\
&\overset{(c)}{=} \mathbf{C}_d\tilde{P}_d(\mathbf{A})\mathbf{A}^{\tau}\mathbf{x}_0 + \sum_{k=1}^{\lfloor\tau/(2N+1)\rfloor}\mathbf{C}_d\tilde{P}_d(\mathbf{A})\mathbf{A}^{\tau-1-(2N+1)k}\mathbf{B}\mathbf{e}_k + \mathbf{0} && (81)\\
&\overset{(d)}{=} \mathbf{0}, && (82)
\end{aligned}$$

where

(a) is obtained by substituting $\mathbf{y}_d$ from (77),

(b) follows from the fact that $0 \leq j \leq N$ and that

$$\tau = (2N+1)p + \tilde{\tau}, \quad \text{where } 0 \leq p \leq \omega,\ 1 \leq \tilde{\tau} \leq N, \tag{83}$$

and, therefore,

$$\begin{aligned}
\left\lfloor\frac{j+\tau-1}{2N+1}\right\rfloor &= \left\lfloor p + \frac{j+\tilde{\tau}-1}{2N+1}\right\rfloor && (84)\\
&= p && (85)\\
&= \lfloor\tau/(2N+1)\rfloor, && (86)
\end{aligned}$$

(c) follows from the fact that $j + \tau$ cannot be a multiple of $(2N+1)$,

(d) follows from (30)-(31) and from Lemma 8.

We now prove the converse. We assume the coefficients $\{\alpha_{d,j}\}_{j=0}^{N}$ of the polynomial $\tilde{P}_d(t) = \sum_{j=0}^{N}\alpha_{d,j}t^j$ satisfy:

$$\sum_{j=0}^{N}\alpha_{d,j}\mathbf{y}_d[j+\tau] = \mathbf{0}, \quad \forall\tau \in \bigcup_{p=0}^{\omega}\bigcup_{\tilde{\tau}=1}^{N}\{(2N+1)p + \tilde{\tau}\}, \tag{87}$$

and show that $\tilde{P}_d(t)$ satisfies:

$$\mathbf{C}_d\tilde{P}_d(\mathbf{A})\mathbf{A}^{\tau}\mathbf{B} = \mathbf{O}, \quad \forall\tau \geq 0, \tag{88}$$

$$\mathbf{C}_d\tilde{P}_d(\mathbf{A})\mathbf{A}^{\tau+1}\mathbf{x}_0 = \mathbf{0}, \quad \forall\tau \geq 0. \tag{89}$$

To prove (89), first note that according to (77) we have

$$\mathbf{y}_d[n] = \mathbf{C}_d\mathbf{A}^n\mathbf{x}_0 \quad 0 \leq n \leq 2N, \tag{90}$$

so if (87) is satisfied for $\tau \in \{1,\dots,N\}$, then (89) is also satisfied for $\tau \in \{0,\dots,N-1\}$ and, hence, for all $\tau \geq 0$ (by Lemma 8).

To prove (88) we prove (by induction on k) that the following holds:

$$\mathbf{C}_d\tilde{P}_d(\mathbf{A})\mathbf{A}^{\tau}\mathbf{B}\mathbf{e}_k = \mathbf{0}, \quad \forall\tau \geq 0,\ k \in \{1,2,\dots,\omega\}. \tag{91}$$

We first prove for $k = 1$. By (77), for $\tau \in \{(2N+1)+1,\dots,(2N+1)+N\}$ we have:

$$\begin{aligned}
\mathbf{0} &= \sum_{j=0}^{N}\alpha_{d,j}\mathbf{y}_d[j+\tau] && (92)\\
&= \sum_{j=0}^{N}\alpha_{d,j}\left(\mathbf{C}_d\mathbf{A}^{j+\tau-1-(2N+1)}\mathbf{B}\mathbf{e}_1 + \mathbf{C}_d\mathbf{A}^{j+\tau}\mathbf{x}_0\right) && (93)\\
&= \mathbf{C}_d\tilde{P}_d(\mathbf{A})\mathbf{A}^{\tau-1-(2N+1)}\mathbf{B}\mathbf{e}_1 + \mathbf{C}_d\tilde{P}_d(\mathbf{A})\mathbf{A}^{\tau}\mathbf{x}_0 && (94)\\
&\overset{(a)}{=} \mathbf{C}_d\tilde{P}_d(\mathbf{A})\mathbf{A}^{\tau-1-(2N+1)}\mathbf{B}\mathbf{e}_1 + \mathbf{0} && (95)
\end{aligned}$$

where (a) follows from (89). In view of Lemma 8, we see that (91) holds for $k = 1$. Assume now that it holds for all $k' \in \{1,\dots,k-1\}$. By (77), for all $\tau \in \{(2N+1)k+1,\dots,(2N+1)k+N\}$ we have:

$$\begin{aligned}
\mathbf{0} &= \sum_{j=0}^{N}\alpha_{d,j}\mathbf{y}_d[j+\tau] && (96)\\
&= \sum_{j=0}^{N}\alpha_{d,j}\left(\mathbf{C}_d\mathbf{A}^{j+\tau}\mathbf{x}_0 + \sum_{k'=1}^{k}\mathbf{C}_d\mathbf{A}^{j+\tau-1-(2N+1)k'}\mathbf{B}\mathbf{e}_{k'}\right) && (97)\\
&= \mathbf{C}_d\tilde{P}_d(\mathbf{A})\mathbf{A}^{\tau-1-(2N+1)k}\mathbf{B}\mathbf{e}_k\\
&\quad + \sum_{k'=1}^{k-1}\mathbf{C}_d\tilde{P}_d(\mathbf{A})\mathbf{A}^{\tau-1-(2N+1)k'}\mathbf{B}\mathbf{e}_{k'} + \mathbf{C}_d\tilde{P}_d(\mathbf{A})\mathbf{A}^{\tau}\mathbf{x}_0\\
&\overset{(a)}{=} \mathbf{C}_d\tilde{P}_d(\mathbf{A})\mathbf{A}^{\tau-1-(2N+1)k}\mathbf{B}\mathbf{e}_k + \mathbf{0}, && (98)
\end{aligned}$$

where (a) follows from the induction assumption of (91) on $k' < k$ and from (89). In view of Lemma 8, we see that (91) holds for all $0 \leq k \leq \omega$ and, therefore, (88) holds as well. □

The proof for Theorem 9 consists of three parts. In the first part we calculate $\mathbf{g}_{d,i}(z)$ from (14) and we substitute $\{\mathbf{y}_d[n]\}$ from

$$\mathbf{y}_d[n] = 1_{\{n\geq 0\}}\mathbf{C}_d\mathbf{A}^n\mathbf{x}_0 + \sum_{k=1}^{\min\left\{\omega,\left\lfloor\frac{n-1}{2N+1}\right\rfloor\right\}}\mathbf{C}_d\mathbf{A}^{n-1-(2N+1)k}\mathbf{B}\mathbf{e}_k + \mathbf{D}_d\sum_{k=1}^{\omega}\mathbf{e}_k 1_{\{n=(2N+1)k\}}. \tag{99}$$

By substituting (99) into (14) and using Theorem 7 we show that $\mathbf{G}_d(z) = \left[\mathbf{g}_{d,1}(z),\dots,\mathbf{g}_{d,\omega}(z)\right]$ is given by

$$\mathbf{G}_d(z) = \sum_{k=0}^{N_d-1}\left(\sum_{j=k+1}^{N_d}\alpha_{d,j}\mathbf{C}_d\mathbf{A}^{j-k-1}\mathbf{B}\right)z^{N_d-k} + P_d(z)\mathbf{D}_d. \tag{100}$$

In the second part of the proof we show that $\left(P_d(z)\mathbf{y}_d(z)\right)[n]$ is equal to two sequences. The first sequence is $(\mathbf{G}_d(z)\mathbf{u}(z))[n]$. The second sequence depends on $\mathbf{x}_0$ and it vanishes for $n > N_d$, and thus the difference equation (32) holds.

In the third part of the proof we give a decoding scheme to find $\mathbf{u}$ from $\mathbf{y}_d$ if $\mathbf{G}_d(z)$ is of full rank. If it is not, we show that there are two different input sequences $\mathbf{u}[n]$ that can lead to the same output sequence $\mathbf{y}_d[n]$ and, therefore, decoding in that case is not possible as one cannot be sure which of the input sequences was transmitted.

Proof for Theorem 9: We begin by calculating $\mathbf{g}_{d,i}(z)$ for all $i \in \{1,2,\dots,\omega\}$:

$$\begin{aligned}
&\mathbf{g}_{d,i}(z) && (101)\\
&= \sum_{k=0}^{N_d}\sum_{j=0}^{N_d}\alpha_{d,j}\mathbf{y}_d[j+(2N+1)i-k]z^{N_d-k}\\
&\overset{(a)}{=} \sum_{k=0}^{N_d}\sum_{j=0}^{N_d}\alpha_{d,j}\mathbf{C}_d\mathbf{A}^j\mathbf{A}^{(2N+1)i-k}\mathbf{x}_0 z^{N_d-k}\\
&\quad + \sum_{k=0}^{N_d}\sum_{j=0}^{N_d}\sum_{k'=0}^{\min\{\omega,\left\lfloor\frac{j-k-1}{2N+1}+i\right\rfloor\}}\alpha_{d,j}\mathbf{C}_d\mathbf{A}^{j-k-1+(2N+1)(i-k')}\\
&\quad \times \mathbf{B}\mathbf{e}_{k'}z^{N_d-k}\\
&\quad + \sum_{k=0}^{N_d}\sum_{j=0}^{N_d}\alpha_{d,j}\sum_{k'=1}^{\omega}\mathbf{D}_d\mathbf{e}_{k'}1_{\{j+(2N+1)i-k=(2N+1)k'\}}z^{N_d-k}, && (102)
\end{aligned}$$

where (a) is obtained by substituting $\{\mathbf{y}_d[n]\}$ from (99). The first part of (102) vanishes because $\tilde{P}_d(t)$ satisfies (31) by Theorem 7 and because of Lemma 8:

$$\begin{aligned}
&\sum_{k=0}^{N_d}\sum_{j=0}^{N_d}\alpha_{d,j}\mathbf{C}_d\mathbf{A}^j\mathbf{A}^{(2N+1)i-k}\mathbf{x}_0 z^{N_d-k}\\
&\quad = \sum_{k=0}^{N_d}\mathbf{C}_d\tilde{P}_d(\mathbf{A})\mathbf{A}^{(2N+1)i-k}\mathbf{x}_0 z^{N_d-k}\\
&\quad = \mathbf{0}.
\end{aligned}$$

The second part of (102) can be written as:

$$\begin{aligned}
&\sum_{k=0}^{N_d}\sum_{j=0}^{N_d}\sum_{k'=0}^{\min\{\omega,\left\lfloor\frac{j-k-1}{2N+1}+i\right\rfloor\}}\alpha_{d,j}\mathbf{C}_d\mathbf{A}^{j-k-1+(2N+1)(i-k')}\mathbf{B}\mathbf{e}_{k'}z^{N_d-k}\\
&\overset{(a)}{=} \sum_{k=0}^{N_d-1}\sum_{j=k+1}^{N_d}\alpha_{d,j}\mathbf{C}_d\mathbf{A}^{j-k-1}\mathbf{B}\mathbf{e}_i z^{N_d-k}\\
&\quad + \sum_{k=0}^{N_d}\sum_{j=0}^{N_d}\sum_{k'=0}^{i-1}\alpha_{d,j}\mathbf{C}_d\mathbf{A}^j\mathbf{A}^{(2N+1)(i-k')-k-1}\mathbf{B}\mathbf{e}_{k'}z^{N_d-k}
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{k=0}^{N_d-1}\sum_{j=k+1}^{N_d} \alpha_{d,j}\mathbf{C}_d\mathbf{A}^{j-k-1}\mathbf{B}\mathbf{e}_i z^{N_d-k} \\
&\quad + \sum_{k=0}^{N_d}\sum_{k'=0}^{i-1}\mathbf{C}_d\tilde{P}_d(\mathbf{A})\mathbf{A}^{(2N+1)(i-k')-k-1}\mathbf{B}\mathbf{e}_{k'}z^{N_d-k} \\
&\overset{(b)}{=} \sum_{k=0}^{N_d-1}\sum_{j=k+1}^{N_d} \alpha_{d,j}\mathbf{C}_d\mathbf{A}^{j-k-1}\mathbf{B}\mathbf{e}_i z^{N_d-k}, \qquad (103)
\end{aligned}
$$

where

(a) is obtained by splitting the summation over k' from 0 to $i-1$, and for $k'=i$ (only when $k<j$).
(b) holds because $\tilde{P}_d(z)$ satisfies (30) by Theorem 7 and because of Lemma 8.

The third part of (102) is:

$$
\begin{aligned}
&\sum_{k=0}^{N_d}\sum_{j=0}^{N_d}\alpha_{d,j}\sum_{k'=1}^{\omega}\mathbf{D}_d\mathbf{e}_{k'}1_{\{j+(2N+1)(i-k')-k=0\}}z^{N_d-k} \\
&\quad \overset{(a)}{=} \sum_{j=0}^{N_d}\alpha_{d,j}z^{N_d-j}\mathbf{D}_d\mathbf{e}_i \\
&\quad = P_d(z)\mathbf{D}_d\mathbf{e}_i, \qquad (104)
\end{aligned}
$$

where (a) holds because $1_{\{j+(2N+1)(i-k')-k=0\}}=1$ only when $k'=i$ and $k=j$. By combining all $\mathbf{g}_{d,i}$ vectors into a matrix we get that $\mathbf{G}_d(z)$ is given by (100). It has already been proved (in Theorem 6) that for such a $\mathbf{G}_d(z)$ matrix the difference equation (8) holds between $\mathbf{u}$ and $\mathbf{y}_d$, provided that $\mathbf{x}_0=\mathbf{0}$ and (in Theorem 7) that $\tilde{P}_d(z)$ satisfies (31). For cases in which the initial state is not zero, the output sequence $\mathbf{y}_d[n]$ can be written as a sum of the zero input response $\mathbf{y}_{\text{ZIR},d}[n]=1_{\{n\ge 0\}}\mathbf{C}_d\mathbf{A}^n\mathbf{x}_0$ and the zero state response $\mathbf{y}_{\text{ZSR},d}[n]=\sum_{k=-\infty}^{n-1}\mathbf{C}_d\mathbf{A}^{n-1-k}\mathbf{B}\mathbf{u}[k]+\mathbf{D}_d\mathbf{u}[n]$, which is the output of the network, as if the initial state was zero. Finally, using Theorem 6 for $\mathbf{y}_{\text{ZSR},d}[n]$ and (31) for $\mathbf{y}_{\text{ZIR},d}[n]$, we get for all $n>N_d$:

$$
\begin{aligned}
\left(P_d(z)\mathbf{y}_d\right)[n] &= \left(P_d(z)\mathbf{y}_{\text{ZIR},d}\right)[n]+\left(P_d(z)\mathbf{y}_{\text{ZSR},d}\right)[n] \qquad (105)\\
&= \sum_{j=0}^{N_d}\alpha_{d,j}\mathbf{C}_d\mathbf{A}^{n-(N_d-j)}\mathbf{x}_0+\left(\mathbf{G}_d(z)\mathbf{u}\right)[n] \qquad (106)\\
&= \mathbf{C}_d\tilde{P}_d(\mathbf{A})\mathbf{A}^{n-N_d}\mathbf{x}_0+\left(\mathbf{G}_d(z)\mathbf{u}\right)[n] \qquad (107)\\
&= \mathbf{0}+\left(\mathbf{G}_d(z)\mathbf{u}\right)[n]. \qquad (108)
\end{aligned}
$$

We now prove that it is possible to decode $\mathbf{u}$ from $\mathbf{y}_d$ if and only if $\mathbf{G}_d(z)$ is of full column rank over the polynomial ring $\mathbb{F}[z]$. If it is not, there exists a non zero vector of polynomials $\mathbf{v}_d(z)$ such that $\mathbf{G}_d(z)\mathbf{v}_d(z)=\mathbf{0}$. Let $\mathbf{u}(z)=\mathbf{v}_d(z)z^{(2N+1)(\omega+1)}$, where $\mathbf{u}(z)$ is the formal Laurent series of the input sequence. As $\mathbf{v}_d(z)$ is a vector of polynomials, the input sequence satisfies $\mathbf{u}[n]=0$ for $n<(2N+1)(\omega+1)$ and, hence, this is a legal input sequence since it starts after Algorithm 2 ends. The sequence $\left(P_d(z)\mathbf{y}_d(z)\right)[n]$ vanishes for $n>N_d$ since $\mathbf{G}_d(z)\mathbf{u}(z)=\mathbf{0}$, and for $n\le N_d$ it is independent of the input sequence. Since $P_d(z)$ is not the zero polynomial we can find $\mathbf{y}_d(z)$ from $\left(P_d(z)\mathbf{y}_d(z)\right)$. In that case, the output sequence $\mathbf{y}_d[n]$ will be independent of the input sequence, and no decoding method will tell if the input sequence was $\mathbf{v}_d(z)z^{(2N+1)(\omega+1)}$ or a zero sequence.

On the other hand, if $\mathbf{G}_d(z)$ is of full column rank then we can show that the input sequence can be decoded from the output sequence. We define:

$$
\mathbf{q}[n]=\begin{cases}\left(\mathbf{G}_d(z)\mathbf{u}(z)\right)[n], & n\le N_d\\ \left(P_d(z)\mathbf{y}_d(z)\right)[n], & n>N_d\end{cases}, \qquad (109)
$$

where

$$
\begin{aligned}
\left(\mathbf{G}_d(z)\mathbf{u}(z)\right)[n] &= \left(\sum_{k=0}^{N_d}\mathbf{G}_d[k]z^k\mathbf{u}(z)\right)[n] \qquad (110)\\
&= \sum_{k=0}^{N_d}\mathbf{G}_d[k]\mathbf{u}[n-k]. \qquad (111)
\end{aligned}
$$

Since $\mathbf{G}_d(z)$ was already found and $\mathbf{u}[0],\dots,\mathbf{u}[N_d]$ are known from (27), $\mathbf{q}[n]$ can be computed for all $n\in\mathbb{Z}$ and it satisfies

$$
\mathbf{q}[n]=\left(\mathbf{G}_d(z)\mathbf{u}(z)\right)[n] \quad \forall n\in\mathbb{Z}. \qquad (112)
$$

Henceforth, the same decoding algorithm (72)-(76) as in the proof for Theorem 6 can be used. □

Proof for Theorem 10: First, we show that if for every $s\in\mathcal{S}$ there are R'_s linearly independent column vectors $\mathbf{v}_{s,1},\dots,\mathbf{v}_{s,R_s}$ from the columns of the matrix $\mathbf{G}_{d,s}(z)$, such that $\cup_{s\in\mathcal{S}}\cup_{k=1}^{R'_s}\{\mathbf{v}_{s,k}\}$ is a set of linearly independent vectors, then the rates $(R'_s)_{s\in\mathcal{S}}$ are achievable for the sink node d with the current LEK. Every source node s can transmit its input symbols only on the inputs $\{u_{s,i}\}$ that correspond to the vectors $\{\mathbf{v}_{s,1},\dots,\mathbf{v}_{s,R'_s}\}$ and zeros on the other inputs:

$$
u_{s,i} = \begin{cases}\text{The zero sequence,} & \text{if } [\mathbf{G}_{d,s}(z)]_i\notin\{\mathbf{v}_{s,1},\dots,\mathbf{v}_{s,R'_s}\}\\ \text{A non zero sequence,} & \text{if } [\mathbf{G}_{d,s}(z)]_i\in\{\mathbf{v}_{s,1},\dots,\mathbf{v}_{s,R'_s}\},\end{cases} \qquad (113)
$$

where $[\mathbf{G}_{d,s}(z)]_i$ is the column i of $\mathbf{G}_{d,s}(z)$. In this case (8) can be simplified to

$$
P_d(z)\mathbf{y}_d=\tilde{\mathbf{G}}_d(z)\tilde{\mathbf{u}}, \qquad (114)
$$

where $\tilde{\mathbf{u}}[n]$ is the input sequence with all zero inputs removed and $\tilde{\mathbf{G}}_d(z)$ is the matrix

$$
\tilde{\mathbf{G}}_d(z)=\left[\mathbf{v}_{s_1,1},\dots,\mathbf{v}_{s_1,R'_{s_1}}\mathbf{v}_{s_2,1},\dots,\mathbf{v}_{s_2,R'_{s_2}},\dots,\mathbf{v}_{s_{|\mathcal{S}|},R'_{s_{|\mathcal{S}|}}}\right]. \qquad (115)
$$

From the second part of the proof of Theorem 6 and the third part of the proof of Theorem 9, we know that it is possible to decode $\tilde{\mathbf{u}}$ from $\mathbf{y}_d$ if and only if the matrix $\tilde{\mathbf{G}}_d(z)$ is of full column rank, i.e. all of its column vectors are linearly independent over $\mathbb{F}[z]$. We assumed that the columns of $\tilde{\mathbf{G}}_d(z)$ are linearly independent and, therefore, the rates $(R'_s)_{s\in\mathcal{S}}$ are achievable for the sink node d.

We now prove the converse of Theorem 10. We assume that the rates $(R'_s)_{s\in\mathcal{S}}$ are achievable for a sink node d and we show that for every $s\in\mathcal{S}$ there are R'_s linearly independent column vectors $\mathbf{v}_{s,1}, \ldots, \mathbf{v}_{s,R'_s}$ from the columns of the matrix $\mathbf{G}_{d,s}(z)$, such that the vectors $\cup_{s\in\mathcal{S}} \cup_{k=1}^{R'_s} \{\mathbf{v}_{s,k}\}$ are linearly independent. By the definition of achievable rates, we know that every source $s \in \mathcal{S}$ can transmit zeros on $(R_s - R'_s)$ out of its input sequences, such that the sink node d is able to decode $\mathbf{u}$ from $\mathbf{y}_d$. If that is the case, the relationship between the input and the output sequences is described by (114), where $\tilde{\mathbf{u}}[n]$ is the input sequence with all zero inputs removed, and $\tilde{\mathbf{G}}_d(z)$ is the matrix $\mathbf{G}_d(z)$ with removed column vectors that correspond to the zero input sequences. Since $\tilde{\mathbf{u}}$ is decodable, we know that the column vectors of $\tilde{\mathbf{G}}_d(z)$ are linearly independent over $\mathbb{F}[z]$. The R'_s column vectors of $\tilde{\mathbf{G}}_d(z)$ that correspond to the non zero inputs of a source node s are also column vectors of $\mathbf{G}_{d,s}(z)$, since $\mathbf{G}_{d,s}(z)$ contains all the column vectors of $\mathbf{G}_d(z)$ that correspond to the input sequence $\mathbf{u}_s$. Therefore, we have shown that for every source s, there are R'_s linearly independent vectors from the columns of $\mathbf{G}_{d,s}(z)$, such that all the vectors together are linearly independent. This completes the proof. □

Proof for Theorem 12: Let a node v have scalar LEK, k output links with unit time delay and $|Out(v)| - k$ output links without delay. Denote its output vector by

$$\mathbf{x}^v_{out} = \begin{bmatrix} \mathbf{x}^v_{out,1} \\ \mathbf{x}^v_{out,2} \end{bmatrix}, \tag{116}$$

where $\mathbf{x}^v_{out,1}$ is of dimension k and consists of the output symbols on the delayed links, and $\mathbf{x}^v_{out,2}$ consists of the rest of the output symbols. Its input-output relationship can be written by

$$\mathbf{x}^v_{out}[n] = \begin{bmatrix} \mathbf{A}_1 \mathbf{x}^v_{in}[n-1] \\ \mathbf{A}_2 \mathbf{x}^v_{in}[n] \end{bmatrix}, \tag{117}$$

where $\mathbf{A}_1, \mathbf{A}_2$ consist of the LEK. Define a state vector to be $\tilde{\mathbf{x}}_v[n] = \mathbf{x}^v_{out,1}[n]$. The input-output relationship can be rewritten using state equations:

$$\tilde{\mathbf{x}}_v[n+1] = \mathbf{A}_1 \mathbf{x}^v_{in}[n], \tag{118}$$

$$\mathbf{x}^v_{out}[n] = \begin{bmatrix} \mathbf{I}_{k\times k} \\ \mathbf{O} \end{bmatrix} \tilde{\mathbf{x}}_v[n] + \begin{bmatrix} \mathbf{O}_{k\times k} \\ \mathbf{A}_2 \end{bmatrix} \mathbf{x}^v_{in}[n]. \tag{119}$$

□

Proof for Corollary 13: The dimension of the state vector of the whole network $\tilde{\mathbf{x}}[n]$ is $\sum_{j\in\mathcal{V}} \dim(\tilde{\mathbf{x}}_j)$, where $\tilde{\mathbf{x}}_j$ is the state vector of node j. If node j has k_j output links with unit time delay, and all its other output links have no delay, then, by Theorem 12, it can have a state vector of dimension not larger than k_j. Since there are only k links in the network with unit time delay, and every link is an output link of only one node, we must have:

$$k = \sum_{j\in\mathcal{V}} k_j \tag{120}$$

$$\geq \sum_{j\in\mathcal{V}} \dim(\tilde{\mathbf{x}}_j) \tag{121}$$

$$= \dim(\tilde{\mathbf{x}}). \tag{122}$$

Therefore, if we take $N \geq k$ then Algorithms 1 and 2 still apply, as it is sufficient to take $N \geq \dim(\tilde{\mathbf{x}})$. □

Proof for Lemma 14: From the definition of $\delta_{\min}$, there is a linear solution for $\mathbf{u}[0]$ from $\{\mathbf{q}[k]\}_{k=0}^{\delta_{\min}}$, and it is given in (54). This is a solution to the linear equations given in (53). Because the sequence $\{\mathbf{u}[k]\}_{k\geq 0}$ is arbitrary, we get the following identity:

$$\begin{bmatrix} \mathbf{T}_0, & \mathbf{T}_1, & \ldots & \mathbf{T}_{\delta_{\min}} \end{bmatrix} \begin{bmatrix} \mathbf{G}_d[0] & \mathbf{O} & \ldots & \mathbf{O} \\ \mathbf{G}_d[1] & \mathbf{G}_d[0] & \ldots & \mathbf{O} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{G}_d[\delta_{\min}] & \mathbf{G}_d[\delta_{\min}-1] & \ldots & \mathbf{G}_d[0] \end{bmatrix} = \begin{bmatrix} I, & \mathbf{O}, & \ldots & \mathbf{O} \end{bmatrix}. \tag{123}$$

We denote $\mathbf{T} = \begin{bmatrix} \mathbf{T}_0, \mathbf{T}_1, \ldots, \mathbf{T}_{\delta_{\min}} \end{bmatrix}$. We use the identity $\mathbf{q}(z) = \mathbf{G}_d(z)\mathbf{u}(z)$ to expand the following expression, from which we will get the rational-power-series decoder:

$$\mathbf{T} \begin{bmatrix} z^{\delta_{\min}} \\ z^{\delta_{\min}-1} \\ \vdots \\ z^0 \end{bmatrix} \mathbf{q}(z)$$

$$= \mathbf{T} \begin{bmatrix} z^{\delta_{\min}} \\ z^{\delta_{\min}-1} \\ \vdots \\ z^0 \end{bmatrix} \mathbf{G}_d(z)\mathbf{u}(z) \tag{124}$$

$$= \mathbf{T} \begin{bmatrix} z^{\delta_{\min}} \\ z^{\delta_{\min}-1} \\ \vdots \\ z^0 \end{bmatrix} (\mathbf{G}_d[0] + \mathbf{G}_d[1]z + \ldots + \mathbf{G}_d[\delta_{\min}]z^{\delta_{\min}}$$

$$+ \sum_{k=\delta_{\min}+1}^{N_d} \mathbf{G}_d[k]z^k)\mathbf{u}(z) \tag{125}$$

$$= \mathbf{T} \begin{bmatrix} z^{\delta_{\min}}\mathbf{G}_d[0] + z^{\delta_{\min}+1}\sum_{k=1}^{N_d} \mathbf{G}_d[k]z^{k-1} \\ z^{\delta_{\min}-1}\mathbf{G}_d[0] + z^{\delta_{\min}}\mathbf{G}_d[1] + z^{\delta_{\min}+1}\sum_{k=2}^{N_d} \mathbf{G}_d[k]z^{k-2} \\ \vdots \\ \sum_{k=0}^{\delta_{\min}} \mathbf{G}_d[k]z^k + z^{\delta_{\min}+1}\sum_{k=\delta_{\min}+1}^{N_d} \mathbf{G}_d[k]z^{k-\delta_{\min}-1} \end{bmatrix}$$
$$\times \mathbf{u}(z) \tag{126}$$

$$= \mathbf{T} \left(\begin{bmatrix} \mathbf{G}_d[0] & \mathbf{O} & \ldots & \mathbf{O} \\ \mathbf{G}_d[1] & \mathbf{G}_d[0] & \ldots & \mathbf{O} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{G}_d[\delta_{\min}] & \mathbf{G}_d[\delta_{\min}-1] & \ldots & \mathbf{G}_d[0] \end{bmatrix} \begin{bmatrix} z^{\delta_{\min}} \\ z^{\delta_{\min}-1} \\ \vdots \\ z^0 \end{bmatrix} \right.$$
$$\left. + z^{\delta_{\min}+1} \begin{bmatrix} \sum_{k=1}^{N_d} \mathbf{G}_d[k]z^{k-1} \\ \sum_{k=2}^{N_d} \mathbf{G}_d[k]z^{k-2} \\ \vdots \\ \sum_{k=\delta_{\min}+1}^{N_d} \mathbf{G}_d[k]z^{k-\delta_{\min}-1} \end{bmatrix} \right) \mathbf{u}(z). \tag{127}$$

Denote the following matrix of polynomials:

$$\mathbf{R}(z) = \begin{bmatrix} \sum_{k=1}^{N_d} \mathbf{G}_d[k]z^{k-1} \\ \sum_{k=2}^{N_d} \mathbf{G}_d[k]z^{k-2} \\ \vdots \\ \sum_{k=\delta_{\min}+1}^{N_d} \mathbf{G}_d[k]z^{k-\delta_{\min}-1} \end{bmatrix}. \tag{128}$$

Then we have the following identity:

$$\mathbf{T}\begin{bmatrix} z^{\delta_{\min}} \\ z^{\delta_{\min}-1} \\ \vdots \\ z^0 \end{bmatrix}\mathbf{q}(z) = [\mathbf{I},\mathbf{O},\ldots,\mathbf{O}]\begin{bmatrix} z^{\delta_{\min}} \\ z^{\delta_{\min}-1} \\ \vdots \\ z^0 \end{bmatrix}\mathbf{u}(z) + z^{\delta_{\min}+1}\mathbf{TR}(z)\mathbf{u}(z) \tag{129}$$

$$= (\mathbf{I}+z\mathbf{TR}(z))\, z^{\delta_{\min}}\mathbf{u}(z). \tag{130}$$

The matrix $(\mathbf{I}+z\mathbf{TR}(z))$ is invertible, and its inverse is given by:

$$(I+z\mathbf{TR}(z))^{-1} = \sum_{k=0}^{\infty}(-z\mathbf{TR}(z))^k. \tag{131}$$

Therefore, we get the following relationship:

$$z^{\delta_{\min}}\mathbf{u}(z) = \left(\sum_{k=0}^{\infty}(-z\mathbf{TR}(z))^k\right)\mathbf{T}\begin{bmatrix} z^{\delta_{\min}} \\ z^{\delta_{\min}-1} \\ \vdots \\ z^0 \end{bmatrix}\mathbf{q}(z) \tag{132}$$

$$= \mathbf{P}(z)\mathbf{q}(z), \tag{133}$$

where $\mathbf{P}(z)$ is a rational power series matrix. □

Proof for Theorem 15: If $\delta_c = 0$ then the result is obvious. Assume, then, that $\delta_c > 0$. The relationship between $\mathbf{u}(z)$ and $\mathbf{q}(z)$ can be written both as:

$$\mathbf{u}(z) = \frac{adj(\mathbf{G}_d(z))}{\det(\mathbf{G}_d(z))}\mathbf{q}(z) \tag{134}$$

and as

$$\mathbf{u}(z) = z^{-\delta_{\min}}\mathbf{P}(z)\mathbf{q}(z), \tag{135}$$

where $\mathbf{P}$(z) is a rational power series, as follows from Lemma 14. The expressions for $adj(\mathbf{G}_d(z))$ and for $\det(\mathbf{G}_d(z))$ are given in (48-49). The definition of δ_c is $\delta_c = \tau - g$, where τ and g are given in in (48-49). From (134-135) we can write:

$$\mathbf{P}(z) = z^{\delta_{\min}}\frac{adj(\mathbf{G}_d(z))}{\det(\mathbf{G}_d(z))} \tag{136}$$

$$= z^{\delta_{\min}}\frac{z^g\sum_{k=0}^{\mathbf{W}_{max}}\mathbf{W}[k]z^k}{z^\tau\beta\left(1+\sum_{i=1}^{K}\gamma_i z^i\right)} \tag{137}$$

$$= \frac{z^{\delta_{\min}}}{z^{\delta_c}}\frac{\sum_{k=0}^{\mathbf{W}_{max}}\mathbf{W}[k]z^k}{\beta\left(1+\sum_{i=1}^{K}\gamma_i z^i\right)} \tag{138}$$

$$= z^{\delta_{\min}-\delta_c}\beta^{-1}\left(\sum_{k=0}^{\mathbf{W}_{max}}\mathbf{W}[k]z^k\right)\left(\sum_{j=0}^{\infty}\left(-\sum_{i=1}^{K}\gamma_i z^i\right)^j\right). \tag{139}$$

The first term in this power series expansion is $z^{\delta_{\min}-\delta_c}\beta^{-1}\mathbf{W}[0]$. Since $\mathbf{W}[0]$ is not zero, and $\mathbf{P}(z)$ is a rational power series, the power of $z^{\delta_{\min}-\delta_c}$ must be non-negative and, therefore, $\delta_{\min} \geq \delta_c$. □

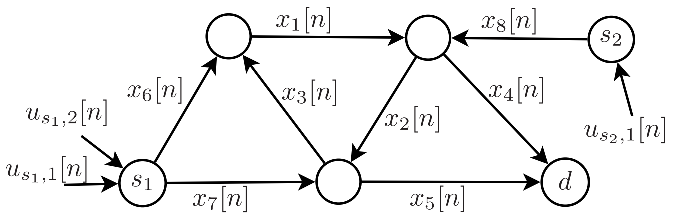

Fig. 3. Network with two source nodes, one sink node and three relay nodes.

Appendix B Examples

Example 16 (Application of Algorithm 1): In this example we apply the first initialization algorithm, find the polynomial $P_d(z)$ and the transfer matrix $\mathbf{G}_d(z)$ (the GEK is found by dividing $\mathbf{G}_d(z)/P_d(z)$) and give a decoding scheme. Consider the network shown in Fig.3, with two source nodes s_1, s_2, one sink node d and three relay nodes. The field over which the network operates is $\mathbb{F}_{2^8}$ with the primitive polynomial $t^8+t^4+t^3+t^2+1$ used to define the field. The elements of the field $\mathbb{F}_{2^8}$ are polynomials of the form:

$$\sum_{k=0}^{7}a_k t^k, \quad \forall i : a_i \in \{0,1\}. \tag{140}$$

For simplicity, we use an integer representation for every scalar from the field, such that every scalar is represented by a number between 0 and 255 whose binary representation $(a_7, a_6, \ldots, a_0)$ is given by the components a_i from (140). The reasons for selecting this field were i) this is a commonly used field since every byte represents a symbol and ii) we wanted to show that the algorithm also works with large fields. In Example 18 we use a small prime field $\mathbb{F}_2$ to simplify the computations.

All LEK were generated randomly and are given by

$$x_1[n+1] = 37x_6[n] + 108x_3[n], \tag{141}$$
$$x_2[n+1] = 234x_1[n] + 203x_8[n], \tag{142}$$
$$x_3[n+1] = 245x_7[n] + 168x_2[n], \tag{143}$$
$$x_4[n+1] = 10x_1[n] + 217x_8[n], \tag{144}$$
$$x_5[n+1] = 239x_7[n] + 174x_2[n], \tag{145}$$
$$x_6[n+1] = 194u_{s_1,1}[n] + 190u_{s_1,2}[n], \tag{146}$$
$$x_7[n+1] = 101u_{s_1,1}[n] + 168u_{s_1,2}[n], \tag{147}$$
$$x_8[n+1] = 44u_{s_2,1}[n]. \tag{148}$$

All nodes know only the following facts:

- The source nodes list is $\mathcal{S} = \{s_1, s_2\}$ and the sink node is d.
- The network has not more than 8 edges ($N = 8$).
- The number of output links for every source node: $|Out(s_1)| = 2$ and $|Out(s_2)| = 1$.

Note that even though the rates $(R_{s_1}, R_{s_2}) = (2,1)$ are not achievable (they do not satisfy the Min-Cut Max-Flow condition), we assume for now that this information is not known a priori. If it was known a priori, we could set the rates to $(R_{s_1}, R_{s_2}) = (1,1)$ (by setting $u_{s_1,2}[n] = 0$) or to $(R_{s_1}, R_{s_2}) = (2,0)$ (by setting $u_{s_2,1}[n] = 0$), since these rates are achievable. In that case the whole initialization process would take 34 time units. The initialization process begins when s_1 and s_2 send the following input sequence $\mathbf{u}[n] = \left(u_{s_1,1}[n], u_{s_1,2}[n], u_{s_2,1}[n]\right)^T$:

$$\mathbf{u}[n] = \mathbf{e}_1, \mathbf{0}, \mathbf{0}, \ldots, \mathbf{0} \quad 0 \le n \le 16, \tag{149}$$

where $\mathbf{e}_1 = (1,0,0)^T$ and $\mathbf{0}$ is the zero column vector of length 3. After $n = 16$, all the symbols in the network are cleared, n is set to zero, and the following input sequence is sent:

$$\mathbf{u}[n] = \mathbf{e}_2, \mathbf{0}, \mathbf{0}, \ldots, \mathbf{0} \quad 0 \le n \le 16, \tag{150}$$

where $\mathbf{e}_2 = (0,1,0)$. Again, after $n = 16$ the network is cleared, and the sent input sequence is:

$$\mathbf{u}[n] = \mathbf{e}_3, \mathbf{0}, \mathbf{0}, \ldots, \mathbf{0} \quad 0 \le n \le 16, \tag{151}$$

where $\mathbf{e}_3 = (0,0,1)$. Meanwhile, the output sequence $\mathbf{y}_d = (x_4, x_5)^T$ received by d is given by

$$\begin{aligned}\mathbf{M}_d[n] &= \left[\mathbf{y}_d^1[n], \mathbf{y}_d^2[n], \mathbf{y}_d^3[n]\right] \quad \forall n \in \{1,\ldots,16\}\\ &= \begin{bmatrix}0&0&0\\0&0&0\end{bmatrix}, \begin{bmatrix}0&0&231\\157&13&0\end{bmatrix}, \begin{bmatrix}57&73&0\\0&0&228\end{bmatrix},\\ &\quad \begin{bmatrix}113&63&0\\185&105&0\end{bmatrix}, \begin{bmatrix}0&0&228\\1&101&0\end{bmatrix}, \ldots,\end{aligned} \tag{152}$$

where $\mathbf{y}_d^1, \mathbf{y}_d^2$ and $\mathbf{y}_d^3$ are the received output sequences in the first, second and third loop of the algorithm respectively. For $n = 0$ and $3 \le n \le 13$ we have

$$\mathbf{M}_d[0] = \mathbf{O}, \tag{153}$$

$$\mathbf{M}_d[n+3] = 209\mathbf{M}_d[n] \quad \forall 3 \le n \le 13. \tag{154}$$

We now solve the system of linear equations given in (9). We look for some non trivial solution. We can take, for example,

$$\alpha_{d,2} = 209, \quad \alpha_{d,5} = 1, \;\; \alpha_{d,k} = 0 \;\; k \in \{0,1,3,4,6,7,8\}. \tag{155}$$

This is, indeed, a solution of (9), as can be seen by:

$$\sum_{k=0}^{N} \alpha_{d,k}\mathbf{M}_d[k+\tau] = 209\mathbf{M}_d[2+\tau] + \mathbf{M}_d[5+\tau] \tag{156}$$

$$= 209\mathbf{M}_d[2+\tau] + 209\mathbf{M}_d[2+\tau] \tag{157}$$

$$= \mathbf{O}, \quad \forall \tau \ge 1. \tag{158}$$

The number N_d, the polynomial $P_d(z)$ and the matrix $\mathbf{G}_d(z)$ given by this solution are:

$$N_d = 5 \tag{159}$$

$$P_d(z) = 1 + 209z^3 \tag{160}$$

$$\begin{aligned}&\mathbf{G}_d(z)\\ &= \left(209\mathbf{M}_d[2] + \mathbf{M}_d[5]\right) z^5 + \left(209\mathbf{M}_d[1] + \mathbf{M}_d[4]\right) z^4\\ &\quad + \left(\mathbf{M}_d[3]\right) z^3 + \left(\mathbf{M}_d[2]\right) z^2 + \left(\mathbf{M}_d[1]\right) z\\ &= \begin{bmatrix}113z^4+57z^3, & 63z^4+73z^3, & 84z^5+231z^2\\ 24z^5+185z^4+157z^2, & 17z^5+105z^4+13z^2, & 228z^3\end{bmatrix}.\end{aligned} \tag{161}$$

If we are interested in finding achievable rates from the matrix $\mathbf{G}_d(z)$, we should apply Algorithm 3, as described in Example 17. For now, we set the rates to be achievable: $R_{s_1} = R_{s_2} = 1$ (by sending $u_{s_1,2}[n] = 0$), and we get the following relationship between the input and the output sequences:

$$(1+209z^3)\mathbf{y}_d = \begin{bmatrix}113z^4+57z^3 & 84z^5+231z^2\\ 24z^5+185z^4+157z^2 & 228z^3\end{bmatrix}\begin{bmatrix}u_{s_1,1}\\ u_{s_2,1}\end{bmatrix}. \tag{162}$$

Note that if we had started the initialization process with the rates $R_{s_1} = R_{s_2} = 1$, we would have obtained the reduced transfer matrix $\tilde{\mathbf{G}}_d(z)$ given on the right side of (162). We can solve (162) for $\mathbf{u}$ by multiplying both sides of the equation by $\left(42 \cdot adj(\tilde{\mathbf{G}}_d(z))\right)$, see (163), as shown at the bottom of this page, where we used the identity

$$42 \cdot adj\left(\tilde{\mathbf{G}}_d(z)\right)\tilde{\mathbf{G}}_d(z) = 42\det(\tilde{\mathbf{G}}_d(z))\mathbf{I}. \tag{165}$$

We used the factor 42 to make the coefficient of z^4 from the left side of (163) equal to one. The difference equation for

$$(105z^{10}+223z^9+152z^7+149z^6+z^4)\begin{bmatrix}u_{s_1,1}\\ u_{s_2,1}\end{bmatrix} = \begin{bmatrix}221z^6+119z^3, & 65z^8+119z^5+9z^2\\ 30z^8+208z^7+42z^5+112z^4+241z^2, & 42z^7+112\,z^6+203z^4+212\,z^3\end{bmatrix}\mathbf{y}_d, \tag{163}$$

$$\begin{aligned}\begin{bmatrix}u_{s_1,1}[n-4]\\ u_{s_2,1}[n-4]\end{bmatrix} &= 149\begin{bmatrix}u_{s_1,1}[n-6]\\ u_{s_2,1}[n-6]\end{bmatrix} + 152\begin{bmatrix}u_{s_1,1}[n-7]\\ u_{s_2,1}[n-7]\end{bmatrix}\\ &\quad + 223\begin{bmatrix}u_{s_1,1}[n-9]\\ u_{s_2,1}[n-9]\end{bmatrix} + 105\begin{bmatrix}u_{s_1,1}[n-10]\\ u_{s_2,1}[n-10]\end{bmatrix}\\ &\quad + \begin{bmatrix}221y_{d,1}[n-6] + 119y_{d,1}[n-3]\\ 30y_{d,1}[n-8] + 208y_{d,1}[n-7] + 42y_{d,1}[n-5]\end{bmatrix}\\ &\quad + \begin{bmatrix}0\\ 112y_{d,1}[n-4] + 241y_{d,1}[n-2]\end{bmatrix}\\ &\quad + \begin{bmatrix}65y_{d,2}[n-8] + 119y_{d,2}[n-5] + 9y_{d,2}[n-2]\\ 42y_{d,2}[n-7] + 112y_{d,2}[n-6] + 203y_{d,2}[n-4] + 212y_{d,2}[n-3]\end{bmatrix}\end{aligned} \tag{165}$$

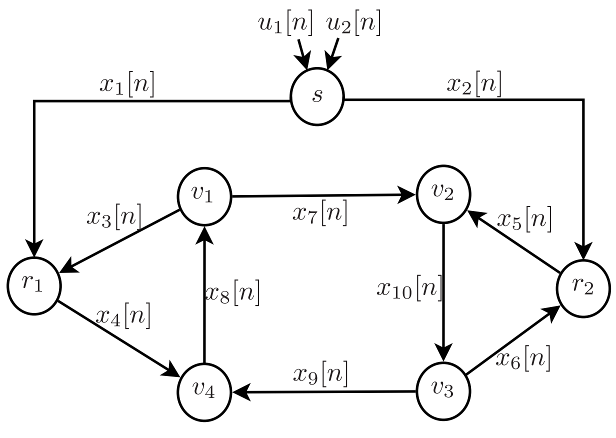

Fig. 4. Shuttle network with one source node s and two sink nodes r_1 and r_2.

$\mathbf{u}[n]$ is (165), as shown at the bottom of the previous page, with the initial conditions

$$\mathbf{u}[n] = \mathbf{0}, \quad \forall n < 0, \tag{166}$$

$$\mathbf{y}_d[n] = \mathbf{0}, \quad \forall n < 0. \tag{167}$$

The decoding delay in this example is 2 since we can decode $\mathbf{u}[n-4]$ once $\mathbf{y}_d[n-2]$ is received.

Example 17 (Application of Algorithm 3): In this example we apply the third algorithm to find achievable rates for the network from Example 16. We return to the network in Fig. 3, with the same field and coefficients as in Example 16. After applying Algorithm 1 or 2, we get the polynomial $P_d(z)$ and the transfer matrix $\mathbf{G}_d(z)$, as given in (160) and (161). We are interested in achievable rates for the sources s_1, s_2, so we follow the instructions given in Algorithm 3. We split $\mathbf{G}_d(z)$ into two matrices:

$$\mathbf{G}_{d,s_1}(z) = \begin{bmatrix} 113z^4+57z^3 & 63z^4+73z^3 \\ 24z^5+185z^4+157z^2 & 17z^5+105z^4+13z^2 \end{bmatrix}, \tag{168}$$

$$\mathbf{G}_{d,s_2}(z) = \begin{bmatrix} 84z^5+231z^2 \\ 228z^3 \end{bmatrix}. \tag{169}$$

The rates $(R_{s_1}, R_{s_2}) = (1,1)$ are achievable, since the vectors

$$\mathbf{v}_1 = (113z^4+57z^3, 24z^5+185z^4+157z^2)^T, \tag{170}$$

$$\mathbf{v}_2 = (84z^5+231z^2, 228z^3)^T, \tag{171}$$

$$\mathbf{v}_1 \in \text{Columns of } (\mathbf{G}_{d,s_1}),$$

$$\mathbf{v}_2 \in \text{Columns of } (\mathbf{G}_{d,s_2}),$$

are linearly independent over the polynomial ring $\mathbb{F}[z]$. The rates $(R_{s_1}, R_{s_2}) = (2,0)$ are also achievable, since $\mathbf{G}_{d,s_1}$ is of full rank over the polynomial ring $\mathbb{F}[z]$.

Example 18 (Shuttle Network): In this example we apply the second initialization algorithm to a network with non-scalar LEK, we obtain the GEK of every sink node, we find achievable rates from the transfer matrices and we give a decoding algorithm for each sink node based on the found GEK. Consider the shuttle network in Fig.4, with one source node s and two sink nodes r_1 and r_2. This network and its LEK were taken from [12], where Guo *et al* presented the adaptive randomized algorithm (ARCNC) to choose LEK for unknown networks. The field over which the network operates is $\mathbb{F}_2$. All LEK were generated using the ARCNC algorithm in [12], and are given by

$$x_1[n] = u_{s,1}[n], \tag{172}$$

$$x_2[n] = u_{s,2}[n], \tag{173}$$

$$x_3[n] = x_8[n], \tag{174}$$

$$x_4[n] = x_1[n] + x_1[n-1], \tag{175}$$

$$x_5[n] = x_2[n] + x_6[n-1], \tag{176}$$

$$x_6[n] = x_{10}[n], \tag{177}$$

$$x_7[n] = x_8[n], \tag{178}$$

$$x_8[n] = x_4[n] + x_4[n-1] + x_9[n-1], \tag{179}$$

$$x_9[n] = x_{10}[n], \tag{180}$$

$$x_{10}[n] = x_7[n-1] + x_5[n]. \tag{181}$$

Note that with these LEK, some links have zero time delay. This fact helps us, since the state vector of each node j can now be smaller. For instance, nodes s, v_1 and v_3 do not have a state vector at all, since the input-output relationship for them does not require a state vector:

$$\mathbf{x}^j_{out}[n] = \begin{pmatrix} 1 \\ 1 \end{pmatrix} x^j_{in}[n], \quad j \in \{v_1, v_3\}, \tag{182}$$

$$\mathbf{x}^s_{out}[n] = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} u_{s,1}[n] \\ u_{s,2}[n] \end{pmatrix}. \tag{183}$$

Nodes r_1, r_2 and v_2 need a state vector with dimension 1, since their input-output relationship can be written as:

$$\tilde{x}_{r_1}[n+1] = x_1[n], \quad n \geq 0 \tag{184}$$

$$x_4[n] = \tilde{x}_{r_1}[n] + (1,0)\begin{pmatrix} x_1[n] \\ x_3[n] \end{pmatrix}, \tag{185}$$

$$\tilde{x}_{r_2}[n+1] = x_6[n], \quad n \geq 0 \tag{186}$$

$$x_5[n] = \tilde{x}_{r_2}[n] + (1,0)\begin{pmatrix} x_2[n] \\ x_6[n] \end{pmatrix}, \tag{187}$$

$$\tilde{x}_{v_2}[n+1] = x_7[n], \quad n \geq 0 \tag{188}$$

$$x_{10}[n] = \tilde{x}_{v_2}[n] + (1,0)\begin{pmatrix} x_5[n] \\ x_7[n] \end{pmatrix}. \tag{189}$$

Node v_4 needs a state vector with dimension 2, since its input-output relationship can be written as:

$$\tilde{\mathbf{x}}_{v_4}[n+1] = \begin{pmatrix} x_4[n] \\ x_9[n] \end{pmatrix}, \quad n \geq 0 \tag{190}$$

$$x_8[n] = (1,1) \cdot \tilde{\mathbf{x}}_{v_4}[n] + (1,0)\begin{pmatrix} x_4[n] \\ x_9[n] \end{pmatrix}. \tag{191}$$

The state vector $\tilde{\mathbf{x}}[n]$ of the entire network is the concatenation of the state vectors $\tilde{x}_{r_1}, \tilde{x}_{r_2}, \tilde{x}_{v_2}, \tilde{\mathbf{x}}_{v_4}$ and, hence, its dimension is $\dim(\tilde{\mathbf{x}}) = 5$. In our algorithm, every $N \geq 5$ will be satisfactory. Note that N is less than $|\mathcal{E}|$, and this is possible because some links no longer have unit time delay. We will take $N = 7$ to show that the algorithm works even when $N > \dim(\tilde{\mathbf{x}})$. To take into account the possibility that $\tilde{\mathbf{x}}[0] \neq \mathbf{0}$, we will take the following initial state vector for the network:

$$\tilde{\mathbf{x}}[0] = \begin{pmatrix} \tilde{x}_{r_1}[0] \\ \tilde{x}_{r_2}[0] \\ \tilde{x}_{v_2}[0] \\ \tilde{\mathbf{x}}_{v_4}[0] \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}. \tag{192}$$

The source and the sink nodes know only the following facts:

- There is one source node s and two sink nodes r_1 and r_2.
- The dimension of the state vector of the network is not more than 7 ($N = 7$).
- The number of output links for the source node $|Out(s)| = 2$.

We follow the instructions of Algorithm 2 to get difference equations for $\mathbf{u}$ and $\mathbf{y}_{r_1} = (x_1, x_3)^T$, $\mathbf{y}_{r_2} = (x_2, x_6)^T$. Initially, the source node s transmits the following sequences:

$$u_{s,1}[n] = \begin{cases} 1, & n = 15 \\ 0, & \text{otherwise} \end{cases} \quad 0 \le n < 45,$$
$$u_{s,2}[n] = \begin{cases} 1, & n = 30 \\ 0, & \text{otherwise} \end{cases} \quad 0 \le n < 45. \tag{193}$$

The output sequences $\{\mathbf{y}_{r_1}[n]\}_{1\le n\le 44}$, $\{\mathbf{y}_{r_2}[n]\}_{1\le n\le 44}$ are stored at the sink nodes r_1 and r_2 respectively. Here are some of the initial and final values of $\{\mathbf{y}_{r_1}[n]\}_{n=0}^{44}$, $\{\mathbf{y}_{r_2}[n]\}_{n=0}^{44}$:

$$\mathbf{y}_{r_1}[n] = \begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix}, \dots \begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}0\\0\end{pmatrix}, \quad 0 \le n \le 14, \tag{194}$$
$$\begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}0\\0\end{pmatrix}, \dots \begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix}, \quad 15 \le n \le 44, \tag{195}$$
$$\mathbf{y}_{r_2}[n] = \begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}0\\0\end{pmatrix}, \dots \begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}0\\0\end{pmatrix}, \quad 0 \le n \le 29, \tag{196}$$
$$\begin{pmatrix}1\\1\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix}, \dots \begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}0\\0\end{pmatrix}, \quad 30 \le n \le 44. \tag{197}$$

We now solve the system of linear equations

$$\sum_{j=0}^{7} \alpha_{r_i,j} \mathbf{y}_{r_i}[j+\tau] = \mathbf{0}, \quad \forall \tau \in \bigcup_{p=0}^{2} \bigcup_{\tilde{\tau}=1}^{7} \{15p + \tilde{\tau}\}. \tag{198}$$

For both r_1 and r_2 these equations can be reduced to :

$$\alpha_{r_i,0} = 0, \tag{199}$$
$$\alpha_{r_i,1} + \alpha_{r_i,3} + \alpha_{r_i,4} + \alpha_{r_i,6} + \alpha_{r_i,7} = 0, \tag{200}$$
$$\alpha_{r_i,2} + \alpha_{r_i,3} + \alpha_{r_i,5} + \alpha_{r_i,6} = 0. \tag{201}$$

We seek some non trivial solution. We can take, for example,

$$\alpha_{r_i,1} = \alpha_{r_i,2} = \alpha_{r_i,3} = 1, \tag{202}$$
$$\alpha_{r_i,0} = \alpha_{r_i,4} = \alpha_{r_i,5} = \alpha_{r_i,6} = \alpha_{r_i,7} = 0. \tag{203}$$

In this case we get the same N_d and $P_d(z)$ for both sink nodes, but different transfer matrices $\mathbf{G}_d(z)$ given by (15-14)

$$N_{r_1} = N_{r_2} = 3 \tag{204}$$
$$P_{r_1}(z) = P_{r_2}(z) = z^2 + z + 1 \tag{205}$$
$$\mathbf{G}_{r_1}(z) = \begin{bmatrix} z^2+z+1, & 0 \\ z^3+z^2+z+1, & z \end{bmatrix}, \tag{206}$$
$$\mathbf{G}_{r_2}(z) = \begin{bmatrix} 0, & z^2+z+1 \\ z^3+z, & 1 \end{bmatrix}. \tag{207}$$

Both transfer matrices are of full rank and, hence, the rate $R_s = 2$ is achievable for both sink nodes. The GEK matrices can be found by the division $\mathbf{G}_{r_i}(z)/P_{r_i}(z)$.

For decoding, each sink node first finds

$$\mathbf{q}_{r_i}[n] = \begin{cases} \left(\mathbf{G}_{r_i}(z)\mathbf{u}(z)\right)[n], & n \le N_{r_i} \\ \left(P_{r_i}(z)\mathbf{y}_{r_i}(z)\right)[n], & n > N_{r_i} \end{cases} \tag{208}$$
$$= \begin{cases} \mathbf{0}, & n \le 3 \\ \mathbf{y}_{r_i}[n] + \mathbf{y}_{r_i}[n-1] + \mathbf{y}_{r_i}[n-2], & n > 3. \end{cases} \tag{209}$$

By Theorem 9 we have

$$\mathbf{q}_{r_i}[n] = \left(\mathbf{G}_{r_i}(z)\mathbf{u}(z)\right)[n], \quad n \in \mathbb{Z}, i \in \{1,2\}. \tag{210}$$

Henceforth, we continue as explained in the last part of the proof of Theorem 6. We calculate $\mathbf{w}_{r_i}(z) = adj(\mathbf{G}_{r_i})\mathbf{q}_{r_i}(z)$:

$$\mathbf{w}_{r_1}(z) = \begin{bmatrix} z, & 0 \\ z^3+z^2+z+1, & z^2+z+1 \end{bmatrix} \mathbf{q}_{r_1}(z), \tag{211}$$
$$\mathbf{w}_{r_2}(z) = \begin{bmatrix} 1, & z^2+z+1 \\ z^3+z, & 0 \end{bmatrix} \mathbf{q}_{r_2}(z). \tag{212}$$

The determinants of $\mathbf{G}_{r_1}(z), \mathbf{G}_{r_2}(z)$ are:

$$\det(\mathbf{G}_{r_1}(z)) = z\left(1 + z + z^2\right), \tag{213}$$
$$\det(\mathbf{G}_{r_2}(z)) = z\left(1 + z + z^3 + z^4\right). \tag{214}$$

The decoding is done as in (76):

$$\mathbf{u}[n] = \mathbf{w}_{r_1}[n+1] - (\mathbf{u}[n-1] + \mathbf{u}[n-2]) \tag{215}$$
$$= \mathbf{w}_{r_2}[n+1] - (\mathbf{u}[n-1] + \mathbf{u}[n-3] + \mathbf{u}[n-4]). \tag{216}$$

The decoding delay for both sinks is 1 since once $\mathbf{y}_{r_i}[n]$ is known, $\mathbf{q}_{r_i}[n]$ and $\mathbf{w}_{r_i}[n]$ can be calculated, and, consequently, $\mathbf{u}[n-1]$ can be found.

Acknowledgments

The authors would like to thank Dr. Izchak Lewkowicz for his lectures in Linear Control theory, which were of great help.

References

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

[2] S.-Y. R. Li, R. W. Yeung, and N. Cai, "Linear network coding," *IEEE Trans. Inf. Theory*, vol. 49, no. 2, pp. 371–381, Feb. 2003.

[3] R. Koetter and M. Médard, "An algebraic approach to network coding," *IEEE/ACM Trans. Netw.*, vol. 11, no. 5, pp. 782–795, Oct. 2003.

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

[5] S.-Y. R. Li and Q. T. Sun, "Network coding theory via commutative algebra," *IEEE Trans. Inf. Theory*, vol. 57, no. 1, pp. 403–415, Jan. 2011.

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

[7] E. Erez and M. Feder, "Efficient network code design for cyclic networks," *IEEE Trans. Inf. Theory*, vol. 56, no. 8, pp. 3862–3878, Aug. 2010.

[8] T. Ho *et al.*, "A random linear network coding approach to multicast," *IEEE Trans. Inf. Theory*, vol. 52, no. 10, pp. 4413–4430, Oct. 2006.

[9] R. W. Yeung, *Information Theory and Network Coding*. New York, NY, USA: Springer, 2008.

[10] W. Guo, N. Cai, and Q. T. Sun, "Time-variant decoding of convolutional network codes," *IEEE Commun. Lett.*, vol. 16, no. 10, pp. 1656–1659, Oct. 2012.

[11] Q. T. Sun and S.-Y. R. Li, "On decoding of DVR-based linear network codes," *Appl. Algebra Eng., Commun. Comput.*, vol. 26, no. 6, pp. 527–542, 2015.

[12] W. Guo, X. Shi, N. Cai, and M. Médard, "Localized dimension growth: A convolutional random network coding approach to managing memory and decoding delay," *IEEE Trans. Commun.*, vol. 61, no. 9, pp. 3894–3905, Sep. 2013.

[13] N. Cai and W. Guo, "The conditions to determine convolutional network coding on matrix representation," in *Proc. IEEE NetCod*, Jun. 2009, pp. 24–29.

[14] C. Fragouli and E. Soljanin, "A connection between network coding and convolutional codes," in *Proc. IEEE Int. Conf. Commun.*, vol. 2. Jun. 2004, pp. 661–666.

[15] W. M. Gentleman and S. C. Johnson, "Analysis of algorithms, a case study: Determinants of matrices with polynomial entries," *ACM Trans. Math. Softw.*, vol. 2, no. 3, pp. 232–241, 1976.

[16] S. H. Friedberg, A. J. Insel, and L. E. Spence, *Linear Algebra*, 2nd ed. Englewood Cliffs, NJ, USA: Prentice-Hall, 1989.

[17] G. Gu, *Discrete-Time Linear Systems: Theory and Design With Applications*. Secaucus, NJ, USA: Springer, 2012.

Maxim Lvov received his B.Sc. *(summa cum laude)* degree in Electrical and Computer Engineering from the Ben-Gurion University, Israel, in 2013, where he is currently pursuing the M.Sc. degree.

He is currently an algorithm engineer and researcher at AudioCodes Ltd., Israel. His current research interests include information theory, machine learning, signal and image processing, computer vision, speech recognition and probability theory.

Haim H. Permuter (M'08–SM'13) received his B.Sc. *(summa cum laude)* and M.Sc *(summa cum laude)* degrees in Electrical and Computer Engineering from the Ben-Gurion University, Israel, in 1997 and 2003, respectively, and the Ph.D. degree in Electrical Engineering from Stanford University, California in 2008.

Between 1997 and 2004, he was an officer at a research and development unit of the Israeli Defense Forces. Since 2009 he is with the department of Electrical and Computer Engineering at Ben-Gurion University where he is currently a professor, Luck-Hille Chair in Electrical Engineering. Haim also serves as head of the communication track in his department.

Prof. Permuter is a recipient of several awards, among them the Fullbright Fellowship, the Stanford Graduate Fellowship (SGF), Allon Fellowship, and and the U.S.-Israel Binational Science Foundation Bergmann Memorial Award. Haim served on the editorial boards of the IEEE TRANSACTIONS ON INFORMATION THEORY in 2013–2016.