On network coding for acyclic networks with delays

K. Prasad and B. Sundar Rajan

Dept. of ECE, IISc, Bangalore 560012, India

Email: {prasadk5,bsrajan}@ece.iisc.ernet.in

Abstract—Problems related to network coding for acyclic, instantaneous networks (where the edges of the acyclic graph representing the network are assumed to have zero-delay) have been extensively dealt with in the recent past. The most prominent of these problems include (a) the existence of network codes that achieve maximum rate of transmission, (b) efficient network code constructions, and (c) field size issues. In practice, however, networks have transmission delays. In network coding theory, such networks with transmission delays are generally abstracted by assuming that their edges have integer delays. Note that using enough memory at the nodes of an acyclic network with integer delays can effectively simulate instantaneous behavior, which is probably why only acyclic instantaneous networks have been primarily focused on thus far. In this work, we elaborate on issues $((a), (b)$ and (c) above) related to network coding for acyclic networks with integer delays, which have till now mostly been overlooked. We show that the delays associated with the edges of the network cannot be ignored, and in fact turn out to be advantageous, disadvantageous or immaterial, depending on the topology of the network and the problem considered i.e., $(a), (b)$ or (c). In the process, we also show that for a single source multicast problem in acyclic networks (instantaneous and with delays), the network coding operations at each node can simply be limited to storing old symbols and coding them over a binary field. Therefore, operations over elements of larger fields are unnecessary in the network, the trade-off being that enough memory exists at the nodes and at the sinks, and that the sinks have more processing power.

I. INTRODUCTION

Network coding was introduced in [1] as a means to improve the rate of transmission in networks. Linear network coding was introduced in [2] and it was found to be sufficient to achieve the maxflow-mincut capacity in certain scenarios such as multicast. The linear network coding problem on a network with given sink demands can be considered to have three major subproblems.

- Existence of a network code that satisfies the demands.
- Efficient construction of such a network code.
- Minimum field size for the existence of such a network code.

An algebraic theory of network coding was developed in [3], which converted the existence problem of network coding into an algebraic geometry problem. As for the latter two, most of the literature in network coding has focused on the multicast problem, i.e., where all sinks demand the information generated by all the sources in the network. A polynomial-time algorithm for designing a multicast network code on a single-source acyclic instantaneous (zero-delay) network was presented in [4]. This algorithm was further generalized in [5]

and [6] for the case of multicast on networks with cycles. Note that the notion of delays in the network is inherent to any algorithm on cyclic networks. Delays are therefore assumed either on the edges of the networks that contribute to the cycles alone, or throughout the network. An information flow decomposition based approach to the problem of network code construction was discussed in [7].

Computing the minimum field size required to solve a network coding problem is known [8] to be NP-hard. However, for the multicast case on acyclic networks and certain kinds of cyclic networks, it is known [3] [4] that a field size larger than the number of sinks in the network is sufficient. Further results on the field size issue can be found in [6] [7] [9]–[11]. In certain networks, linear network coding itself is found to be insufficient to achieve the given demands [12].

The case of acyclic networks with delays was abstracted in [3] as acyclic networks where each edge in the network has an integer delay associated with it. With this setting, the authors of [3] were able to naturally generalize the notion of linear network coding and thereby the framework for the problem of the existence of a linear network code on such networks was presented. According to the framework of [3], a network code on an acyclic network with general demands has to satisfy two conditions at every sink to be a solution for the network, which we refer to as (a) *invertibility* conditions, which have to be satisfied to recover the information sequences demanded at each sink, and (b) *zero-interference* conditions, which have to be satisfied so that information sequences not needed at a sink do not interference with those that are demanded (a formal description of these conditions are given in Section III). If at least one such network coding solution exists for the network, then the network is said to be *solvable*.

A *delay profile* for a network consists of a set of non-negative integers, one for each edge in the network indicating the integer delay experienced by the symbols on that edge. For a given delay profile for a network, it has been noted (see [13] [14], for example) that an instantaneous behaviour can be simulated in acyclic networks with integer delays using enough memory at the nodes of the network. It is assumed in most of network coding literature that this can always be done, and that this is indeed the source of the instantaneous behaviour in the network. However, several questions remain unexplored, such as how solvability, field size, etc., are affected when there is a change in the delay profile of the network, given that the network has already been configured to be instantaneous under a known delay profile.

In [15], it is shown that for multicast networks which are equipped with memory at the nodes, there always exists a network code (using memory at the nodes) which is a valid

solution for the network for any delay profile under any field size. The authors of [15] further show that such a *delay-invariant* code can be found with high probability using a random choice of the network coding coefficients from a large field. They also give a deterministic algorithm to construct such a delay-invariant multicast network code for acyclic and cyclic networks, as long as the field size is larger than the number of sinks. In [16], it is shown that under the conditions where the nodes in the network are always equipped with enough memory to counteract any amount of delays in the network, a network (with arbitrary demands) is solvable for any delay profile if and only if it is solvable with an all-zero delay profile. In other words, networks codes which are solutions for the network under a certain delay profile can always be converted into solutions for the network under a different delay profile by utilizing the appropriate number of memory elements at the nodes of the network.

In this work, we wish to study the effect of the relationship between some of the network coding problems for instantaneous networks and their counterparts with non-trivial delay profiles. The instantaneous network of a network $\mathcal{G}$, which is referred to as $\mathcal{G}_{inst}$ throughout the paper, corresponds to the network $\mathcal{G}$ with a known delay profile where memory elements have already been used to simulate instantaneous (zero-delay) behaviour. We assume that any delay profile in $\mathcal{G}$ can only have greater delays (on one or many edges) compared to those in the basic delay profile that gives rise to the instantaneous network $\mathcal{G}_{inst}$. Therefore, throughout the paper, we view the instantaneous networks as networks with the all-zero delay profile.

Many of the results in this paper compare the network coding problems on $\mathcal{G}_{inst}$ with an *unit-delay network*, $\mathcal{G}_{ud}$. The unit-delay network is the network where the delays in the edges are exactly one unit above the delays in $\mathcal{G}_{inst}$. To derive these comparison results, we assume that the intermediate (non-source non-sink) nodes of $\mathcal{G}$ are equipped with memory sufficient only to simulate the instantaneous behaviour of $\mathcal{G}_{inst}$. This forms the major difference between our results and that of [16], where there is no bound on the amount of memory used by the intermediate nodes. In contrast with [15], [16], we concern ourselves with analysing whether delays over those in $\mathcal{G}_{inst}$ can be advantageous or disadvantageous or neutral, given that the intermediate nodes do not have memory beyond what is used by them to simulate instantaneous behaviour in $\mathcal{G}_{inst}$. Following our framework of viewing $\mathcal{G}_{inst}$ as $\mathcal{G}$ with the all-zero delay profile, and because of the fact that we limit the amount of memory elements at the intermediate nodes of $\mathcal{G}$, we refer to the intermediate nodes of $\mathcal{G}$ as being *memory-free*, i.e, utilizing no memory elements. The unit-delay network $\mathcal{G}_{ud}$ is then appropriately viewed as the network $\mathcal{G}$ with an all-one delay profile (i.e, one where all edges have a delay of unity associated with them). Although our results show the comparisons between network coding on $\mathcal{G}_{inst}$ and $\mathcal{G}_{ud}$, they can be generalized without much difficulty to general delay profiles. The all-one delay profile is chosen only because it is sufficient to illustrate the differences obtained through our results between instantaneous networks and those with delays, and less cumbersome to handle in terms of notation.

Table I summarises the relationships obtained in this paper between some of the network coding problems for instantaneous and unit-delay networks. The contributions of our work are as follows.

- We prove that the solvability of $\mathcal{G}_{inst}$ preserves the invertibility conditions (Proposition 1) in $\mathcal{G}_{ud}$, but not necessarily zero-interference conditions (Example 1). On the other hand, we prove that if $\mathcal{G}_{ud}$ is solvable, then $\mathcal{G}_{inst}$ is always solvable (Proposition 2), and thereby proving that if $\mathcal{G}_{inst}$ is not solvable, then so is $\mathcal{G}_{ud}$ (Corollary 2). These results on the relationship between the solvability and non-solvability of $\mathcal{G}_{inst}$ and $\mathcal{G}_{ud}$ are tabulated in the first two rows of Table I.
- We show that whenever there is a polynomial-time algorithm for constructing a network code for $\mathcal{G}_{ud}$ that satisfies all sink demands, then there is a polynomial-time algorithm for constructing a network code for $\mathcal{G}_{inst}$ which satisfies all sink demands (Corollary 3). The third row of Table I captures these results.
- We prove that under certain conditions on the topology of the network there exists an equivalence between a network code over any particular field constructed on $\mathcal{G}_{inst}$ and $\mathcal{G}_{ud}$ (Proposition 3). Thus, for networks obeying the constraints given in Proposition 3, the minimum field size for constructing a network code satisfying all demands for $\mathcal{G}_{inst}$ and $\mathcal{G}_{ud}$ is the same. We also prove that under such constraints on topology, the non-solvability of $\mathcal{G}_{ud}$ implies the non-solvability of $\mathcal{G}_{inst}$ (Corollary 4). The last two rows of Table I lists these results.
- We prove that there exist networks for which the delays prove useful for the field size problem, i.e., network codes can be constructed over a smaller field size for $\mathcal{G}_{ud}$ compared to $\mathcal{G}_{inst}$, and also show a construction of such networks (Corollary 5). These results are also tabulated in the last two rows of Table I. Towards that end, we prove the feasibility of two multicast algorithms, one of which works for acyclic networks (instantaneous and with delays) and was conjectured in [18] based on the multicast algorithm of [4], and the other works for certain special acyclic networks (Proposition 4 and Proposition 5). These modified algorithms employ low-complexity encoding at the intermediate nodes over $\mathbb{F}_2$ using memory elements, while possibly demanding a larger complexity of decoding at the sinks compared to traditional network coding schemes.

The rest of this work is organized as follows. In Section II, we set up the model and the terminology for acyclic networks with delays. In Section III, we explore the relationship between the network code existence problem in $\mathcal{G}_{ud}$ and $\mathcal{G}_{inst}$ for an acyclic network $\mathcal{G}$ with given set of demands, and also present examples where having delays prevent the existence of any solution for $\mathcal{G}_{ud}$ while solutions exist for $\mathcal{G}_{inst}$. In Subsection IV-A, we analyze the conditions on topology which result in an equivalence of network coding solutions between $\mathcal{G}_{ud}$ and $\mathcal{G}_{inst}$. After briefly reviewing the Linear Information Flow (LIF) algorithm of [4] in Subsection IV-B, in Subsection IV-C we use a modified version of this algorithm to obtain a class

TABLE I
RELATIONSHIP BETWEEN NETWORK CODING PROBLEMS BETWEEN $\mathcal{G}_{inst}$ AND $\mathcal{G}_{ud}$ FOR AN ACYCLIC GRAPH $\mathcal{G}$.

Property of interest	If the property holds for $\mathcal{G}_{inst}$, does it continue to hold for $\mathcal{G}_{ud}$?	If the property holds for $\mathcal{G}_{ud}$, does it continue to hold for $\mathcal{G}_{inst}$?
Solvability	*Yes*, if zero-interference conditions are satisfied (Proposition 1), or for multicast (Corollary 1). *No*, if they are not satisfied (Example 1).	*Yes* (Proposition 2).
Non-solvability	*Yes* (Corollary 2).	*No* (Example 1).
Polynomial-time algorithms for code construction	*No*, in general, as $\mathcal{G}_{ud}$ might not even be solvable (Example 1).	*Yes* (Corollary 3).
Solvability over a particular field	*No*, in general, as $\mathcal{G}_{ud}$ might not even be solvable (Example 1). *Yes*, for certain conditions on topology given by Proposition 3, or for multicast (Corollary 1).	*No*, in general (Corollary 5), as illustrated in Example 2 and Example 3. *Yes*, under certain conditions given by Proposition 3.
Non-solvability over a particular field	*No*, in general (Corollary 5), as illustrated in Example 2 and Example 3.	*No*, in general (Example 1). *Yes*, under conditions on topology given by Corollary 4.

of networks in which delays prove beneficial in the minimum field size problem, i.e., where feasible binary network codes always exist for $\mathcal{G}_{ud}$ irrespective of the field size required for $\mathcal{G}_{inst}$. We conclude the paper in Section V with remarks and directions for further research.

II. NETWORK CODES FOR ACYCLIC NETWORKS WITH DELAYS

Following the terminology of [4], an acyclic network is modeled as an acyclic graph $\mathcal{G}$ with $\mathcal{V}$ being the set of nodes and $\mathcal{E}$ the set of edges in the network. The set $\mathcal{V}$ contains a set of source nodes $\mathcal{S}$ and a set of sink nodes $\mathcal{T}$. We assume that the sources have no incoming edges in the network, while the sinks have no outgoing edges. The time unit under consideration shall imply one use of the channels in the network. Each source $s \in \mathcal{S}$ generates h_s information sequences at the rate of h_s $\mathbb{F}_q$ symbols per every time unit, $\mathbb{F}_q$ being the finite field with q symbols. For each source $s \in \mathcal{S}$, we introduce h_s parallel edges (denoted by $\mathcal{E}_s$) incoming at s, which carry the h_s information sequences to the source s. Let $h = \sum_{s \in \mathcal{S}} h_s$.

Assuming an ordering on the set of information sequences available at the sources, let $\mathcal{I}_t$ denote an indicator function for a sink $t \in \mathcal{T}$, defined as

$$\mathcal{I}_t : \{1, 2, ..., h\} \to \{0, 1\},$$

such that, $\mathcal{I}_t(i) = 1$, if sink t demands the ith information sequence, and 0 otherwise. Let $\mathcal{C}$ denote the collection of the functions $\mathcal{I}_t, \forall t \in \mathcal{T}$.

Each sink node $t \in \mathcal{T}$ demands some subset of size h_t of the h information sequences generated at the sources. Let $h_{\mathcal{T}} = \sum_{t \in \mathcal{T}} h_t$. For each sink t, we assume h_t imaginary outgoing edges from t, denoted by $\mathcal{E}_t$. We represent a network $\mathcal{G}(\mathcal{V}, \mathcal{E})$ with a set of sources $\mathcal{S}$ and a set of sinks $\mathcal{T}$ with a set of demands given by $\mathcal{C}$ as $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{S}, \mathcal{T}, \mathcal{C})$.

Every edge in the directed graph representing the network has a capacity of one $\mathbb{F}_q$ symbol. We abstract the case of networks with delay by assuming a unit-delay associated with edges of the graph $\mathcal{G}$, represented by the parameter z. We denote the graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ along with the delays as $\mathcal{G}_{ud}$, the unit-delay version of $\mathcal{G}$ or simply the *unit-delay network* $\mathcal{G}_{ud}$. Note that network links with integer delays greater than unit are modeled as serially concatenated edges in the directed multi-graph. Because of this reason, we view networks with integer delays and those with unit-delays equivalently.

The set of symbols generated at the sources at any particular instant of time is called a *generation* of symbols. Any node in a unit-delay network may receive information of different generations on its incoming edges at any particular time instant. Except for the discussion in Subsection IV-C, we assume that the intermediate (non-sink, non-source) nodes are memory-free and merely transmit a $\mathbb{F}_q$ linear combination of the incoming sequences on their outgoing edges. Also, the zero-delay version of $\mathcal{G}$, referred to as the *instantaneous network*, is denoted by $\mathcal{G}_{inst}$. The following notations will be used throughout the paper.

$\Gamma_I(v)$	: Set of incoming (including imaginary) edges at node v
$\Gamma_O(v)$	: Set of outgoing (including imaginary) edges at node v
$\delta_I(v)$	: $|\Gamma_I(v)|$.
$\delta_O(v)$	: $|\Gamma_O(v)|$.
$v = head(e)$	: if $e \in \Gamma_I(v)$.
$v = tail(e)$	: if $e \in \Gamma_O(v)$.

For an edge $e \in \mathcal{E} \cup \mathcal{E}_t$, we define the *local encoding vector* as a $\delta_I(tail(e))$-length vector, $(m_{e,p}(z) : p \in \Gamma_I(tail(e)))$, where $m_{e,p}(z) \in \mathbb{F}_q(z)$, the field of rational functions over $\mathbb{F}_q$. The local encoding vector determines the sequence $y_e(z) = \sum_i y_{e,i} z^i$ ($y_{e,i} \in \mathbb{F}_q$ being the symbol at i^{th} time index) flowing on edge e based on the sequences incoming at

$tail(e)$, i.e.,

$$y_e(z) = \sum_{p \in \Gamma_I(tail(e))} m_{e,p}(z) y_p(z). \tag{1}$$

Note that as the intermediate nodes are allowed to take only $\mathbb{F}_q$ linear combinations of the incoming sequences, we have for an edge $e \notin \Gamma_O(s)$ (for any $s \in \mathcal{S}$), $m_{e,p}(z) = z m_{e,p}$, for some $m_{e,p} \in \mathbb{F}_q$ and the parameter z denotes the delay incurred during the transmission through edge e. For an edge $e \in \Gamma_O(s)$ of some source $s \in \mathcal{S}$, we have $m_{e,p}(z) = z\tilde{m}_{e,p}(z)$, for some $\tilde{m}_{e,p}(z) \in \mathbb{F}_q(z)$, as we let the sources take arbitrary combinations over $\mathbb{F}_q(z)$. The additional z again denotes the delay incurred on the edge e. For $\mathcal{G}_{inst}$, note that

$$m_{e,p}(z) = m_{e,p} \in \mathbb{F}_q, \tag{2}$$

for any pair of edges e and p, and therefore the corresponding input-output relationship for any edge e is given independent of the time index as

$$y_e = \sum_{p \in \Gamma_I(tail(e))} m_{e,p} y_p,$$

where $y_e, y_p \in \mathbb{F}_q$.

Let $\boldsymbol{m}$ denote the set of all *local encoding coefficients* (all taking values from $\mathbb{F}_q$). For $\mathcal{G}_{ud}$, $\boldsymbol{m}$ is the set of all $\mathbb{F}_q$ coefficients of the numerators and denominators of all $m_{e,p}(z)$. For $\mathcal{G}_{inst}$, $\boldsymbol{m}$ denotes the set of all $m_{e,p}$. The difference between the two will be clear from the context.

The network coding problem implies a choice of the local encoding coefficients $m_{e,p}$ such that each sink can recover the information it demands. Because of the linearity of (1), we can associate with every edge e a h-length *global encoding vector* over $\mathbb{F}_q(z)$. The global encoding vector $\boldsymbol{b}(e)$ of edge e, indicates the particular $\mathbb{F}_q(z)$ linear combination of the h information sequences, flowing in e. The global encoding vectors of the h incoming edges at the sources correspond to the basis vectors of $\mathbb{F}_q^h$. By (1), the vector $\boldsymbol{b}(e)$ can be recursively calculated from the global encoding vectors of the edges incoming at $tail(e)$. The global encoding vectors are well defined because of the acyclicity of the network.

Having ordered the h input sequences and the $h_{\mathcal{T}}$ output sequences, the input-output relationship of $\mathcal{G}_{ud}$ can be represented as a $h \times h_{\mathcal{T}}$ matrix over $\mathbb{F}_q(z)$ called the *overall transfer matrix* [3], $M(z)$, of the network, the columns of which are the global encoding vectors of the imaginary outgoing edges from the sinks. The transfer matrix corresponding to a particular sink t, is the $h \times h_t$ matrix $M_t(z)$, the columns of which are the global encoding vectors of the imaginary outgoing edges from the sink t. Therefore, for $\boldsymbol{x}(z)$ being the h-length input vector and $\boldsymbol{y_t}(z)$ being the h_t-length output vector at sink t, we have $\boldsymbol{y_t}(z) = \boldsymbol{x}(z)M_t(z)$. For $\mathcal{G}_{inst}$, the components of the global encoding vectors and network transfer matrices are all elements from $\mathbb{F}_q$. For more details on the structure of these matrices, we refer the reader to [3].

III. EXISTENCE OF NETWORK CODES FOR ACYCLIC NETWORK WITH DELAYS

The problem of network code existence was presented from an algebraic geometry point of view in [3]. The local encoding coefficients $\boldsymbol{m}$ are assumed to be variables which can take values from a large enough finite field. A network code, i.e., a particular choice of the set of all local encoding coefficients $\boldsymbol{m}$, is defined to be *feasible*, i.e., it achieves the given set of demands at the sinks, if the following two conditions are satisfied.

- *Invertibility conditions:* For each sink t, the $h_t \times h_t$ sub-matrix $M'_t(z)$ of $M_t(z)$, the rows of which corresponding to the inputs demanded at sink t, is invertible over $\mathbb{F}_q(z)$.
- *Zero-Interference conditions:* For each sink t, the elements of the matrix $M_t(z)$ which are not part of $M'_t(z)$ are zero.

Note that if the mincut between any source s and any sink t is less than the number of information sequences demanded by t from s, then the network coding problem is clearly not solvable. Besides the mincut conditions, the topology of the network also affects the ability to satisfy the demands in the network.

For each sink t, some elements of $M_t(z)$ are not a part of the $M'_t(z)$ matrix. Let $f_1, f_2, ..., f_K$ be all such elements, for all possible sinks $t \in \mathcal{T}$. Note that each $f_i \in \mathbb{F}_q(z)$ for any particular choice of $\boldsymbol{m}$, hence we represent each f_i as $f_i(\boldsymbol{m}, z)$. Similarly, let $g_1(\boldsymbol{m}, z), g_2(\boldsymbol{m}, z), ..., g_L(\boldsymbol{m}, z)$ be the determinants of the $M'_t(z)$ matrices. Let $g(\boldsymbol{m}, z) = \prod_{i=1}^{L} g_i(\boldsymbol{m}, z)$. The invertibility and zero-interference conditions then imply that the assignment of $\boldsymbol{m}$ should satisfy $g(\boldsymbol{m}, z) \neq 0$ and $f_1(\boldsymbol{m}, z) = f_2(\boldsymbol{m}, z) = ... = f_K(\boldsymbol{m}, z) = 0$ respectively. Similar conditions (except for the delay parameter z) for feasibility hold good for the $\mathcal{G}_{inst}$ also. Note that for certain network topologies or sink demands, the invertibility conditions alone will suffice for feasibility, while the zero-interference conditions might not arise at all [3]. The multicast case, where all sinks demand all the information sequences, is one such example.

We now provide some results regarding the question of whether the solvability of $\mathcal{G}_{ud}$ implies the solvability of $\mathcal{G}_{inst}$ also, and vice versa. The following proposition is a generalized version of Proposition 1 of [17], where the statement was proved only for a multicast case. A simpler proof for this proposition can also be derived easily from the results of [15].

Proposition 1: Let $\mathcal{G}_{ud}(\mathcal{V}, \mathcal{E}, \mathcal{S}, \mathcal{T}, \mathcal{C})$ be an acyclic, unit-delay network with a given set of sink demands and $\mathcal{G}_{inst}(\mathcal{V}, \mathcal{E}, \mathcal{S}, \mathcal{T}, \mathcal{C})$ be the corresponding instantaneous network. Let $\boldsymbol{m'}$ be a set of local encoding kernels which result in a network code for $\mathcal{G}_{inst}$, satisfying the invertibility conditions. Then $\boldsymbol{m'}$ continues to satisfy the invertibility conditions for $\mathcal{G}_{ud}$.

Proof: See Appendix A.

For the multicast case, which has no zero-interference conditions, we then have the following corollary, which was proved in [17].

Corollary 1: Let $\mathcal{G}_{ud}(\mathcal{V}, \mathcal{E}, \mathcal{S}, \mathcal{T}, \mathcal{C})$ be an acyclic, unit-delay network with multicast demands, i.e., all sinks require all the information sequences, and $\mathcal{G}_{inst}(\mathcal{V}, \mathcal{E}, \mathcal{S}, \mathcal{T}, \mathcal{C})$ be the corresponding instantaneous network. Then a feasible network code for $\mathcal{G}_{inst}$ continues to be feasible for $\mathcal{G}_{ud}$.

In a general non-multicast network coding problem, it might not be possible to satisfy the zero-interference conditions in the network $\mathcal{G}_{ud}$, though they can be satisfied in the network $\mathcal{G}_{inst}$. This is because of the fact that different flows which cancelled out the interference in $\mathcal{G}_{inst}$ can take paths of different delays in the corresponding acyclic network with delays, thereby preventing the cancelling effect. Example 1 illustrates one such network, for which there exists solutions in $\mathcal{G}_{inst}$, but none for $\mathcal{G}_{ud}$.

Example 1: Consider the network $\mathcal{G}$ shown in Fig. 1. Let the field under consideration be $\mathbb{F}_q$. Source s_1 has a sequence $x_1(z)$, which has to be conveyed to sink t_1, while the sequence $x_2(z)$ at source s_2 has to be conveyed to sink t_2. In both $\mathcal{G}_{inst}$ and $\mathcal{G}_{ud}$, the topology of the network demands that the linear combination of the two incoming sequences at node v_1 should be such that both the local encoding coefficients are non-zero.

In $\mathcal{G}_{inst}$, the information sequence $x_1(z)$ is cancelled out at node v_2 to enable sink t_2 to receive $x_2(z)$, and similarly cancellation of $x_2(z)$ happens at node v_3 for sink t_1. In $\mathcal{G}_{ud}$, this cancellation, while being necessary for the network code to be feasible, cannot happen at the nodes v_2 and v_3 because of the disparity in the delays of the flows at their incoming edges. Since the choice of our finite field was arbitrary, it is therefore clear that unless memory is used at some of the intermediate nodes, there exists no feasible network code for this network considered with delays over any finite field.
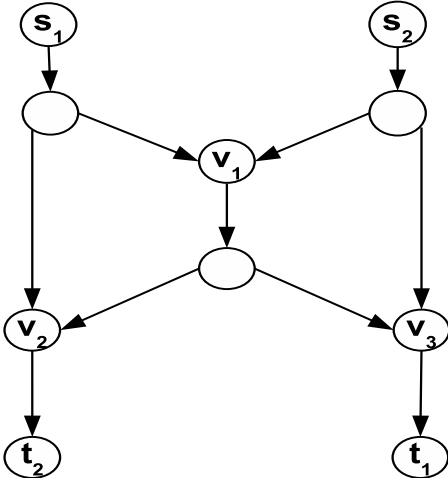

Fig. 1. A network $\mathcal{G}$ where zero-interference conditions fail to hold in $\mathcal{G}_{ud}$

In light of Proposition 1 and Example 1, it can be observed that the solvability of a network coding problem for $\mathcal{G}_{inst}$ need not imply solvability for $\mathcal{G}_{ud}$. The following proposition answers the reverse problem, i.e., that solvability of a given $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{S}, \mathcal{T}, \mathcal{C})$ for $\mathcal{G}_{ud}$ always implies its solvability for $\mathcal{G}_{inst}$.

Proposition 2: Let $\mathcal{G}_{ud}(\mathcal{V}, \mathcal{E}, \mathcal{S}, \mathcal{T}, \mathcal{C})$ be an acyclic, unit-delay network with a given set of sink demands and $\mathcal{G}_{inst}$ be the corresponding instantaneous network. If there exists a feasible network code for $\mathcal{G}_{ud}(\mathcal{V}, \mathcal{E}, \mathcal{S}, \mathcal{T}, \mathcal{C})$, then there exists a feasible network code for $\mathcal{G}_{inst}(\mathcal{V}, \mathcal{E}, \mathcal{S}, \mathcal{T}, \mathcal{C})$.

Proof: See Appendix B.

Proposition 2 leads to the following corollary.

Corollary 2: Let $\mathcal{G}_{ud}(\mathcal{V}, \mathcal{E}, \mathcal{S}, \mathcal{T}, \mathcal{C})$ be an acyclic, unit-delay network with a given set of sink demands and $\mathcal{G}_{inst}$ be the corresponding instantaneous network. If there exists no feasible network code for $\mathcal{G}_{inst}(\mathcal{V}, \mathcal{E}, \mathcal{S}, \mathcal{T}, \mathcal{C})$, then there exists no feasible network code for $\mathcal{G}_{ud}(\mathcal{V}, \mathcal{E}, \mathcal{S}, \mathcal{T}, \mathcal{C})$.

Note that the proof of Proposition 2 involved an actual construction of a feasible network code for $\mathcal{G}_{inst}$ starting from a feasible network code for $\mathcal{G}_{ud}$. Such a construction implies the following corollary on a polynomial-time construction for a feasible network coding solution for $\mathcal{G}_{inst}$.

Corollary 3: Let $\mathcal{G}_{ud}(\mathcal{V}, \mathcal{E}, \mathcal{S}, \mathcal{T}, \mathcal{C})$ be an acyclic, unit-delay network with a given set of sink demands and $\mathcal{G}_{inst}$ be the corresponding instantaneous network. If there exists a polynomial-time construction algorithm for a feasible network coding solution on $\mathcal{G}_{ud}$, then there exists a polynomial-time construction algorithm for a feasible network coding solution on $\mathcal{G}_{inst}$.

Proof: See Appendix C.

IV. RELATIONSHIP BETWEEN THE MINIMUM FIELD SIZE PROBLEM FOR $\mathcal{G}_{ud}$ AND $\mathcal{G}_{inst}$

In this section, we discuss the effect of considering delays in the network on the field size over which a valid network code can be designed for an acyclic network $\mathcal{G}$. We assume that $\mathcal{G}_{ud}$ is solvable, which mean that $\mathcal{G}_{inst}$ is also solvable, according to Proposition 2. Note that Proposition 1 already gives a small insight into the field size issue, showing that for a multicast network, the minimum field size that satisfies the invertibility conditions in $\mathcal{G}_{ud}$ is at most as large as the minimum field size for $\mathcal{G}_{inst}$.

It is not difficult to observe that in some of the usual examples in network coding literature such as the butterfly network and combination networks, the feasibility of a given network code is preserved between the unit-delay network and the corresponding instantaneous network, because the topology of these networks prevents the mixing of information symbols from different generations at the intermediate nodes. In the forthcoming subsection, we formalize such a topological constraint for networks with general demands and thereby obtain sufficient conditions on the equivalence of network coding solutions between $\mathcal{G}_{ud}$ and $\mathcal{G}_{inst}$ for an acyclic network $\mathcal{G}$ with given demands.

A. Equivalence of minimum field size problem between $\mathcal{G}_{inst}$ and $\mathcal{G}_{ud}$

The following proposition gives a class of networks for which the minimum field size is equal for both $\mathcal{G}_{inst}$ and $\mathcal{G}_{ud}$, by demonstrating a sufficient condition under which certain network coding solutions remain feasible for both $\mathcal{G}_{inst}$ and $\mathcal{G}_{ud}$. We define for a node $v \in \mathcal{V} \backslash \mathcal{S}$, a set $Q(v)$ which consists of all possible paths (a path being a sequence of edges following an ancestral order) from the source nodes to v such that any two paths differ by at least one edge. We also define for a node $v \in \mathcal{V} \backslash \mathcal{S}$, a $|Q(v)|$-length *depth vector* $\boldsymbol{d}(v)$, each component (in $\mathbb{Z}^+$) of which indicates the total delay incurred in the corresponding path of $Q(v)$ from some source s to node v.

Proposition 3: Let $\mathcal{G}_{ud}(\mathcal{V}, \mathcal{E}, \mathcal{S}, \mathcal{T}, \mathcal{C})$ be an acyclic, unit-delay network with a given set of sink demands and $\mathcal{G}_{inst}(\mathcal{V}, \mathcal{E}, \mathcal{S}, \mathcal{T}, \mathcal{C})$ be the corresponding instantaneous network. Suppose the topology of $\mathcal{G}_{ud}$ is such that for any $v \in \mathcal{V} \backslash \mathcal{S}$, the components of the depth vector $\boldsymbol{d}(v)$ are all equal. Let $\mathcal{U}$ be the set of all feasible solutions for $\mathcal{G}_{ud}(\mathcal{V}, \mathcal{E}, \mathcal{S}, \mathcal{T}, \mathcal{C})$ such that the sources combine information symbols without using memory, i.e. the symbols only from the current generation, and $\mathcal{U}_q$ be the subset of $\mathcal{U}$ with solutions from the field $\mathbb{F}_q$. Then the following statements are true.

(A) Any solution from $\mathcal{U}$ for $\mathcal{G}_{ud}$ is also a feasible solution for $\mathcal{G}_{inst}$.

(B) Any feasible solution for $\mathcal{G}_{inst}$ is a feasible solution for $\mathcal{G}_{ud}$.

(C) If q_{min} is the minimum field size for which a feasible network code exists for $\mathcal{G}_{ud}$ and the subset $\mathcal{U}_{q_{min}}$ of $\mathcal{U}$ is non-empty, then q_{min} is the minimum field size required for a feasible solution for $\mathcal{G}_{inst}$ too.

Proof: See Appendix D.

Proposition 3 formalizes the easily observed sufficient conditions for the same network codes to be solutions for both $\mathcal{G}_{inst}$ and $\mathcal{G}_{ud}$. However, deriving necessary conditions for the same seems difficult. Proposition 3 also leads to the following obvious corollary.

Corollary 4: Let $\mathcal{G}_{ud}(\mathcal{V}, \mathcal{E}, \mathcal{S}, \mathcal{T}, \mathcal{C})$ be an acyclic, unit-delay network with a given set of sink demands and $\mathcal{G}_{inst}(\mathcal{V}, \mathcal{E}, \mathcal{S}, \mathcal{T}, \mathcal{C})$ be the corresponding instantaneous network. Suppose the topology of $\mathcal{G}_{ud}$ is such that for any $v \in \mathcal{V} \backslash \mathcal{S}$, the components of the depth vector $\boldsymbol{d}(v)$ are all equal. If $\mathcal{G}_{ud}$ has no feasible solutions over some particular field $\mathbb{F}_q$, then neither does $\mathcal{G}_{inst}$.

B. Reduction of minimum field size in $\mathcal{G}_{ud}$ - Review of Linear Information Flow Algorithm

Proposition 3 illustrates some network conditions which lead to the equivalence between finding the minimum required field size for a given set of demands on $\mathcal{G}_{ud}$ and $\mathcal{G}_{inst}$ of a given acyclic network $\mathcal{G}$. Example 1 illustrated a situation where the disparity in the delays of the symbols arriving at a node prevented the possibility of obtaining a feasible network code. However, such delay disparity can also be useful. In particular, because of this delay disparity, there exist networks in which the feasible network codes exist over a smaller field for $\mathcal{G}_{ud}$ compared to $\mathcal{G}_{inst}$. Towards understanding how such situations can arise, we discuss a couple of examples.

Example 2: Consider the network $\mathcal{G}$ shown in Fig. 2. The source s has two sequences $x_1(z)$ and $x_2(z)$ to be transmitted to the six sinks $t_i : 1 \leq i \leq 6$. This network is clearly a cascaded version of the usual butterfly network and the $\binom{4}{2}$ network. As in the case of the $\binom{4}{2}$ network, a feasible network coding solution for this network (either in $\mathcal{G}_{inst}$ or in $\mathcal{G}_{ud}$) implies that any two of the four global encoding vectors on the edges $e_i : 1 \leq i \leq 4$ should be linearly independent. Therefore, for $\mathcal{G}_{inst}$, a minimum field size of 3 is required to construct a feasible network code.

However, for $\mathcal{G}_{ud}$, a binary field is sufficient. Consider the usual network code over $\mathbb{F}_2$ in the butterfly subnetwork of the given network, where the global encoding vectors at the node v_2 are $\begin{pmatrix} z^2 \\ 0 \end{pmatrix}$ and $\begin{pmatrix} z^4 \\ z^4 \end{pmatrix}$, while those at node v_3 are $\begin{pmatrix} 0 \\ z^2 \end{pmatrix}$ and $\begin{pmatrix} z^4 \\ z^4 \end{pmatrix}$. Then the vectors $\begin{pmatrix} z^3 \\ 0 \end{pmatrix}$, $\begin{pmatrix} z^5 \\ z^5 \end{pmatrix}$, $\begin{pmatrix} 0 \\ z^3 \end{pmatrix}$ and $\begin{pmatrix} z^5 \\ z^5 + z^3 \end{pmatrix}$ can be chosen as global encoding vectors for the edges $e_i : 1 \leq i \leq 4$ respectively, which render the network code feasible for $\mathcal{G}_{ud}$. Therefore, even in a multicast situation, the minimum field size requirement of the unit-delay network can be smaller than that of the corresponding instantaneous network. Note that such a situation is made possible because of the difference in the delays between the incoming symbols at the two edges of v_3.
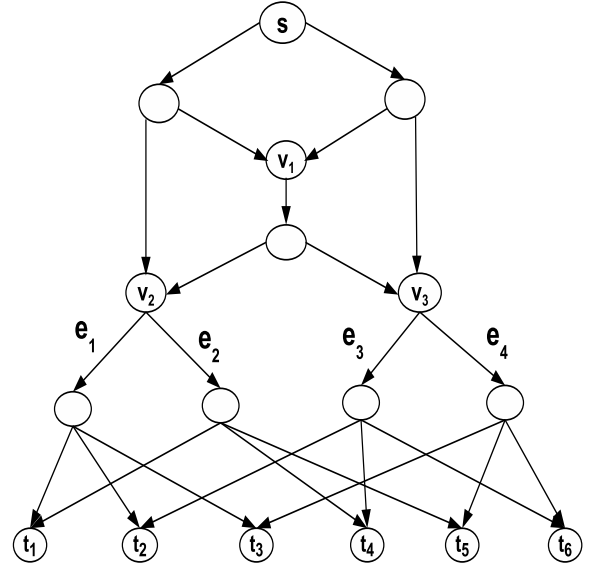

Fig. 2. A multicast network in which $\mathcal{G}_{ud}$ requires smaller field size than $\mathcal{G}$

Example 3: Consider the network $\mathcal{G}$ shown in Fig. 3. For $1 \leq j \leq 3$, each source s_j has an information sequence $x_j(z)$. This network has non-multicast demands, with sinks $t_i : 1 \leq i \leq 3$ requiring all three information sequences, while sink t_4 requires $\{x_1(z), x_3(z)\}$ and t_5 demands $\{x_2(z), x_3(z)\}$. We show that no feasible network code exists for $\mathcal{G}_{inst}$ over $\mathbb{F}_2$, while such a code exists for $\mathcal{G}_{ud}$.

We now argue that we cannot obtain a feasible network code for $\mathcal{G}_{inst}$ over $\mathbb{F}_2$. The sinks $t_i : 1 \leq i \leq 5$ have direct paths from the source(s) $\{s_1, s_2\}, \{s_1, s_3\}, \{s_2, s_3\}, \{s_3\}$ and $\{s_3\}$ respectively. As the sinks $t_i : 1 \leq i \leq 3$ require information sequences from all three sources, the edge e_1 should carry a coded version of all three information sequences for the network code to be feasible at sinks $t_i : 1 \leq i \leq 3$. Thus, over $\mathbb{F}_2$, the global encoding vector for edge e_1 should be $\boldsymbol{a} = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}^T$. As sink t_5 has a direct path from source s_3, the edge e_2 should carry a linear combination of both $x_2(z)$ and $x_3(z)$. Thus the global encoding vector of edge e_2 over $\mathbb{F}_2$ must be $\boldsymbol{b} = \begin{pmatrix} 0 & 1 & 1 \end{pmatrix}^T$. Now, as sink t_4 has a direct path from source s_3, the edge e_3 should carry a linear combination

of both $x_1(z)$ and $x_3(z)$, i.e., $(1 \ \ 0 \ \ 1)^T$. However, the global encoding vectors of the incoming edges at node v_1 are $\boldsymbol{a}$ and $\boldsymbol{b}$, using which the vector $(1 \ \ 0 \ \ 1)^T$ cannot be obtained. Thus no feasible network code can be found for $\mathcal{G}_{inst}$ over $\mathbb{F}_2$. A feasible network code can be found for this network over any field with size $q \geq 3$.

Now we prove by argument that there exists a code for $\mathcal{G}_{ud}$. Because e_1 should carry a linear combination of all three information sequences, let its global encoding vector be $\boldsymbol{a}(z) = \left(z^2 \ \ z^2 \ \ z^2\right)^T$, after accounting for the delays incurred in the transmission. Accounting for the disparity in the delays at the node v_2, let the global encoding vector of edge e_2 be $\boldsymbol{b}(z) = \left(0 \ \ z^2 \ \ z^3\right)^T$. Thus the global encoding vectors of the incoming edges at node v_1 are $z\boldsymbol{a}(z)$ and $z\boldsymbol{b}(z)$. Node v_1 can then simply send a sum of two incoming symbols on edge e_3, in which case the global encoding vector of edge e_3 is $\left(z^4 \ \ 0 \ \ z^4 + z^5\right)^T$. Accounting for all the direct paths in the network, it can be seen that all sink demands are satisfied, i.e., the invertibility conditions hold over $\mathbb{F}_2(z)$ and so do the zero-interference conditions. Thus there is a feasible network code for $\mathcal{G}_{ud}$ over $\mathbb{F}_2$. As in the previous example, the delay disparity at node v_2 is what makes this possible.
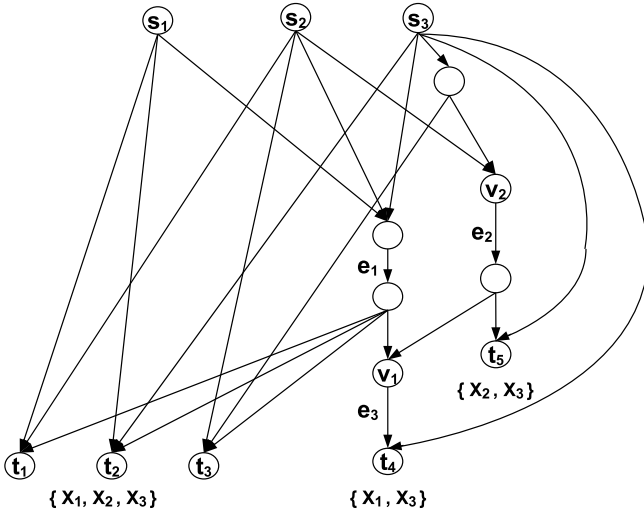

Fig. 3. A non-multicast network in which $\mathcal{G}_{ud}$ requires smaller field size than $\mathcal{G}$

In the remainder of this section, we show that there exist several occasions in which a binary field is sufficient for constructing feasible network codes for unit delay networks, irrespective of the field size required for their instantaneous counterparts. We concentrate only on the single-source multicast case. As the results in this section are all centered on the deterministic version of the *Linear Information Flow* (LIF) algorithm in [4], we briefly discuss the terminology and lemmas related to the LIF algorithm before proving our results.

The steps of the LIF algorithm for constructing a multicast network code in a single source acyclic instantaneous network $\mathcal{G}_{inst}$ with $|\mathcal{T}|$ sinks are as follows.

1) Identify the h_s edge-disjoint paths from the source to the sinks, where h_s is the number of information symbols

at the source. Note that h_s can be at most equal to the minimum of the mincuts between the source and each sink and not more, otherwise the multicast problem is infeasible. Let $\mathcal{G}^*(\mathcal{V}^*, \mathcal{E}^*)$ be the subnetwork of $\mathcal{G}$ consisting of the nodes and edges on these edge-disjoint paths alone. The rest of the algorithm works only with $\mathcal{G}^*$, as a feasible network code for $\mathcal{G}^*$ can be converted to a feasible network code for $\mathcal{G}$ by simply assigning zeros for any other local encoding coefficients.

2) For a sink $t \in \mathcal{T}$, let f^t denote the set of h_s edge-disjoint paths from the source to t. For each sink $t \in \mathcal{T}$, the algorithm maintains for each sink a set C_t, which consists of the h_s most recently processed edges (one from every path in f^t) and a $h_s \times h_s$ matrix B_t, which has the global encoding vectors of the edges in C_t. The set C_t is initialized with the set $\{e_i : 1 \leq i \leq h_s\}$ (the h_s imaginary edges at the source), while the columns of the matrix B_t is initialized with the h_s-length global encoding vectors $\{\boldsymbol{b}(e_i) : 1 \leq i \leq h_s\}$ respectively. For every sink t, the algorithm maintains the full-rank property of B_t by an appropriate choice of the local encoding coefficients at $tail(e)$, $e \in C_t$ being the most recently processed edge as C_t is incremented in some ancestral ordering. The algorithm also maintains another $h_s \times h_s$ matrix A_t which has the inverse vectors of B_t at every step of the algorithm.

3) For an edge $e \in \mathcal{E}^*$, let $f^t_\leftarrow(e)$ denote the predecessor edge of e on some flow path from the source s to sink t. Note that there can be at most one flowpath from s to any sink through e. After processing any edge e, the LIF algorithm updates the set C_t and the matrices B_t and A_t. The updated values for C_t, B_t and A_t are denoted by $\tilde{C}_t, \tilde{B}_t$ and $\tilde{A}_t$ and are obtained as follows.

   a) $\tilde{C}_t = C_t \setminus \{f^t_\leftarrow(e)\} \cup \{e\}$.
   b) $\tilde{B}_t = \tilde{B}_t \setminus \{\boldsymbol{b}(f^t_\leftarrow(e))\} \cup \{\boldsymbol{b}(e)\}$.
   c) $\tilde{\boldsymbol{a}}_t(e) = (\boldsymbol{b}(e).\boldsymbol{a}_t(f^t_\leftarrow(e)))^{-1} \boldsymbol{a}_t(f^t_\leftarrow(e))$.
   d) For all $c \in C_t \setminus \{f^t_\leftarrow(e)\}$,

$$\tilde{\boldsymbol{a}}_t(c) = \boldsymbol{a}_t(c) - (\boldsymbol{b}(e).\boldsymbol{a}_t(c)) \, \tilde{\boldsymbol{a}}_t(c). \tag{3}$$

4) Ultimately the algorithm ends with the matrix B_t equal to the network transfer matrix M_t of the sink t, whose full-rank property is guaranteed by the algorithm, therefore rendering a feasible network code.

Let $T(e)$ be the number of sinks which have a flow path through e, and let $P(e) = \{f^t_\leftarrow(e) : t \in T(e)\}$ be the set of all predecessor edges of e. The only non-zero local encoding coefficients to be chosen for the edge e by the algorithm are $m_{e,p}, \ p \in P(e)$. At every step of the algorithm where the global encoding vector of the next edge (according to a chosen ancestral order) e is selected, the set B_t has to be kept linearly independent, i.e., the choice of $m_{e,p}$ should be such that $B_t \setminus \{\boldsymbol{b}(f^t_\leftarrow(e))\} \cup \{\boldsymbol{b}(e)\}$ is linearly independent. The following lemma proved in [4] gives a fast way to test this linear independence based on the dot product in $\mathbb{F}_q$. In the following lemma, we have $\delta_{a,b} = 1$, if $a = b$.

Lemma 1: Consider a basis B of $\mathbb{F}_q^{h_s}$ and vectors $\boldsymbol{b} \in B$, $\boldsymbol{a} \in \mathbb{F}_q^{h_s}$ such that $\forall \boldsymbol{b}' \in B$, we have $\boldsymbol{b}'.\boldsymbol{a} = \delta_{\boldsymbol{b},\boldsymbol{b}'}$. Then, any

vector $\boldsymbol{x} \in \mathbb{F}_q^{h_s}$ is linearly dependent on $B \backslash \{\boldsymbol{b}\}$ if and only if $\boldsymbol{x}.\boldsymbol{a} = 0$.

Given the full-rank matrix B_t of global encoding vectors $\{\boldsymbol{b}(e) \in \mathbb{F}_q^{h_s} : e \in C_t\}$, we will denote the corresponding columns of the inverse matrix A_t of B_t, as $\{\boldsymbol{a_t}(e) \in \mathbb{F}_q^{h_s} : e \in C_t\}$. Then the linear independence condition to be checked in the LIF algorithm takes the following form due to Lemma 1:

$$\forall t \in T : \forall e, e' \in C_t : \boldsymbol{b}(e).\boldsymbol{a_t}(e') = \delta_{e,e'}. \tag{4}$$

The following lemma gives the sufficient field size for the construction of a feasible network code for multicast on a single source acyclic network.

Lemma 2: [4] Let $q \geq n$. Consider pairs $(\boldsymbol{x_i}, \boldsymbol{y_i}) \in \mathbb{F}_q^{h_s} \times \mathbb{F}_q^{h_s}$ with $\boldsymbol{x_i}.\boldsymbol{y_i} \neq 0$ for $i \leq i \leq n$. There exists a linear combination $\boldsymbol{u}$ of $\boldsymbol{x_1}, \boldsymbol{x_2}, ..., \boldsymbol{x_n}$ such that $\boldsymbol{u}.\boldsymbol{y_i} \neq 0$ for $1 \leq i \leq n$.

Outline of proof: We provide only an outline here in order that we might use similar proof ideas later in Subsection IV-C. For the complete proof, the reader is referred to [4]. The proof involves the iterative construction of vectors $\boldsymbol{u_1}, \boldsymbol{u_2}, ..., \boldsymbol{u_n}$ (each $\boldsymbol{u_i} \in \mathbb{F}_q^{h_s}$) such that for any i, $\boldsymbol{u_i}$ is some linear combination of the vectors $\{\boldsymbol{x_k} : 1 \leq k \leq i\}$ and for any j such that $1 \leq j \leq i$, $\boldsymbol{u_i}.\boldsymbol{y_j} \neq 0$. As long as the field size is more than n, the vectors $\boldsymbol{u_i}$ can always be found, with the final vector $\boldsymbol{u_n}$ being the desired $\boldsymbol{u}$. ∎

With e being the edge under consideration, let

$$\{(\boldsymbol{x_i}, \boldsymbol{y_i}) : i \leq i \leq n\} = \{(\boldsymbol{b}(f_{\leftarrow}^t(e)), \boldsymbol{a_t}(f_{\leftarrow}^t(e))) : t \in T(e)\}$$

in Lemma 2. Then the vector $\boldsymbol{u}$ found using Lemma 2 satisfies (by invoking Lemma 1) the requirements for the global encoding vector $\boldsymbol{b}(e)$ of edge e, i.e., (4). The particular linear combination of the vectors $\{\boldsymbol{b}(f_{\leftarrow}^t(e)) : e \in P(e)\}$ used to obtain $\boldsymbol{u}$ gives the local encoding coefficients at $tail(e)$, i.e., $\{m_{e,p} : p \in P(e)\}$. By Lemma 2, a field size q such that

$$q > \max_{e \in \mathcal{E}} T(e), \tag{5}$$

$T(e)$ being the number of sinks which have flow paths through e, is always sufficient for constructing a multicast network code for $\mathcal{G}_{inst}^*$ according to the LIF algorithm. Therefore, for constructing a multicast network code in any single source acyclic network with $|\mathcal{T}|$ sinks, $q > |\mathcal{T}|$ is sufficient.

Note that although the LIF designs a feasible network code for $\mathcal{G}_{inst}$, the extension of the LIF algorithm (and the associated lemmas) to $\mathcal{G}_{ud}$ is straightforward. While the local encoding coefficients (picked according to Lemma 2) continue to be over $\mathbb{F}_q$, the matrices A_t and B_t are over $\mathbb{F}_q(z)$ for $\mathcal{G}_{ud}$ according to (1). Therefore, the dot product involved in the Lemma 1 and Lemma 2 are the standard dot product in $\mathbb{F}_q(z)$, and full-rank property of B_t is checked over $\mathbb{F}_q(z)$. By Corollary 1, (5) holds for multicast network code construction for $\mathcal{G}_{ud}^*$ too.

C. Delay-and-code: A technique for single-source multicast on acyclic networks

As discussed in Subsection IV-B, the LIF algorithm uses Lemma 2 together with Lemma 1 for constructing a multicast network code in a given instantaneous network $\mathcal{G}_{inst}$, or the corresponding unit-delay network $\mathcal{G}_{ud}$. Based on the LIF algorithm, we now present another network coding scheme called a *delay-and-code* scheme, which reduces the complexity of encoding at the intermediate nodes at the cost of potentially increased complexity of decoding at the sink nodes. The finite field under consideration is always $\mathbb{F}_2$. As a theoretical by-product of this scheme, we show that there exist networks for which the binary field is sufficient for constructing a multicast network code in $\mathcal{G}_{ud}$, irrespective of the field size requirement in $\mathcal{G}_{inst}$.

We assume that each node is equipped with memory elements and a linear combination of the stored symbols is then transmitted on the outgoing edges. Abusing the definition of the delay parameter z, we also denote a memory element by z. We however do not allow all possible $\mathbb{F}_2(z)$-linear combinations of the incoming symbols that is possible using the memory elements available at the node under consideration. In other words, the input-output relationship of the edge e given by (1) is restricted to be of the form

$$y_e(z) = \sum_{p \in \Gamma_I(tail(e))} m_{e,p} z^{a_{e,p}} y_p(z), \tag{6}$$

where $m_{e,p} \in \mathbb{F}_2$, and $a_{e,p} \in \mathbb{Z}^{\geq 0}$, in general (for both $\mathcal{G}_{inst}$ and $\mathcal{G}_{ud}$). For $\mathcal{G}_{ud}$, $a_{e,p} \in \mathbb{Z}^+$, to account for mandatory delay incurred in the transmission through edge e.

If a delay-and-code scheme on $\mathcal{G}$ is such that for any non-source node $v \in \mathcal{V} \cup \mathcal{T}$, and for any $p \in \Gamma_I(v)$,

$$a_{e,p} = a_p, \ \forall e \in \Gamma_O(v) \text{ such that } m_{e,p} \neq 0, \tag{7}$$

then we refer to the delay-and-code scheme as a *uniform delay-and-code* scheme. Otherwise, we refer to it as a *non-uniform delay-and-code* scheme. In other words, in the uniform delay-and-code scheme, an intermediate node is not allowed to code differently delayed versions of the symbols arriving from any particular edge. In the non-uniform case, this is permitted. Note that we consider only intermediate nodes in the network, i.e., the non-source non-sink nodes. The non-uniform delay-and-code technique was already mentioned in [18] for acyclic and cyclic networks. It was however only conjectured that a feasible multicast network code can be designed using the non-uniform delay-and-code scheme. In this work, we prove this conjecture for the case of acyclic networks.

Similar to the usual network coding formulation, such linear combinations also result in a network transfer matrix at each sink, which should be full-rank over $\mathbb{F}_2(z)$ for the network code to be feasible for that particular sink. The algorithm for constructing a delay-and-code scheme for any given acyclic single-source network with multicast demands follows that of the LIF algorithm, with the change that the local encoding coefficients are based on the formulation of (6). The following proposition shows that any acyclic network with multicast demands can be solved using the non-uniform delay-and-code scheme.

Proposition 4: Let $\mathcal{G}(\mathcal{V}, \mathcal{E}, s, \mathcal{T}, \mathcal{C})$ be an acyclic single-source network with multicast demands with the mincut between s and any $t \in \mathcal{T}$ being at least h_s, the number of information sequences generated at s. Then a feasible network

code can be designed for $\mathcal{G}_{inst}$ (or $\mathcal{G}_{ud}$) using the non-uniform delay-and-code scheme provided the total number of memory elements present at each node for each incoming edge is at least $(|\mathcal{T}|-1)$.

Proof: See Appendix E.

We now deal with the uniform delay-and-code scheme. We consider only the special case of those networks in which the paths from the source to each sink are not only edge-disjoint but also node-disjoint, i.e., the h_s paths from the source to any sink do not have any common node except the source and that particular sink itself. The general case, where paths are not necessarily node-disjoint, is more difficult and might not be solvable because of the following reason.

Consider a network with paths that are not node-disjoint. Because of the formulation specified by (7), for any given intermediate node v, all the priorly processed outgoing edges of v should be considered when processing any particular $e' \in \Gamma_O(v)$. Let $e, e' \in \Gamma_O(v)$ and suppose $T(e) \cap T(e') \neq \Phi$. Suppose the global encoding vector of e has been decided before e'. Once edge e' has been processed ($\boldsymbol{b}(e')$ has been decided), note that the elements of the set $S(e)$, which was used to determine $\boldsymbol{b}(e)$, would have been updated according to (3). Thereby $\boldsymbol{b}(e)$ might no longer satisfy the required properties of maintaining the ranks of the B_t matrices of some $t \in T(e) \cap T(e')$. Now if a new global encoding vector $\boldsymbol{b}(e)$ was chosen for edge e, then the set $S(e')$ might change, and $\boldsymbol{b}(e')$ might no longer be a valid global encoding vector for e'. Because of such a see-saw effect, it might not be possible to design a feasible network code using the uniform delay-and-code scheme.

Now suppose the h_s edge-disjoint paths to each sink from s are also node-disjoint, i.e., at any intermediate node in the network, there exists at most one incoming-outgoing edge pair which lies on any path from the source to any particular sink. Therefore, for any $e \in \Gamma_O(v)$ being the currently processed edge, any $c \in C_t \setminus \{f_{\leftarrow}^t(e)\}$ for any $t \in T(e)$ is such that $c \notin \Gamma_I(v)$. In other words, there are no $e, e' \in \Gamma_O(v)$ such that $T(e) \cap T(e') \neq \Phi$. Thus, fixing the global encoding vector for any $e' \in \Gamma_O(v)$ does not affect the sets $S(e)$ for any other edge $e \in \Gamma_O(v)$. For this reason, we focus on networks with node-disjoint paths for the uniform delay-and-code case.

Proposition 5: Let $\mathcal{G}(\mathcal{V}, \mathcal{E}, s, \mathcal{T}, \mathcal{C})$ be an acyclic single-source network with multicast demands with at least h_s node-disjoint paths between s and any $t \in \mathcal{T}$, h_s being the number of information sequences generated at s. Let $\delta := \max_{v \in \mathcal{V}} \delta_O(v)$. Then a feasible network code can be designed for $\mathcal{G}_{inst}$ (or $\mathcal{G}_{ud}$) using the uniform delay-and-code scheme provided the total number of memory elements present at each node for each incoming edge is at least $\delta(|\mathcal{T}|-1)$.

Proof: See Appendix F.

The following corollary to Proposition 5 shows that there exist several unit-delay networks for which a binary field is sufficient for constructing a feasible network code, irrespective of the field size required for their instantaneous counterparts.

Corollary 5: There exist acyclic networks for which a feasible binary network code exists for the unit-delay networks as a result of differently delayed information available at the coding nodes (where paths to different sinks intersect) in the network,

irrespective of the minimum field size required to design a feasible network code for the corresponding instantaneous networks. In particular, given a single-source acyclic network $\mathcal{G}$ with multicast demands and with at least h_s node-disjoint paths (h_s being the number of information symbols at source) from the source to each sink, it is always possible to construct a modified network $\tilde{\mathcal{G}}$ such that $\tilde{\mathcal{G}}_{inst}$ has the same minimum field size requirement as $\mathcal{G}_{inst}$, but a binary field size would suffice to obtain a feasible network code for $\tilde{\mathcal{G}}_{ud}$.

Proof: See Appendix G.

Although the delay-and-code scheme can be used to construct feasible network codes in the multicast situation, for a network with more general demands, it might not prove to be useful. We now present an example where it is not possible to design a network code using the delay-and-code scheme.

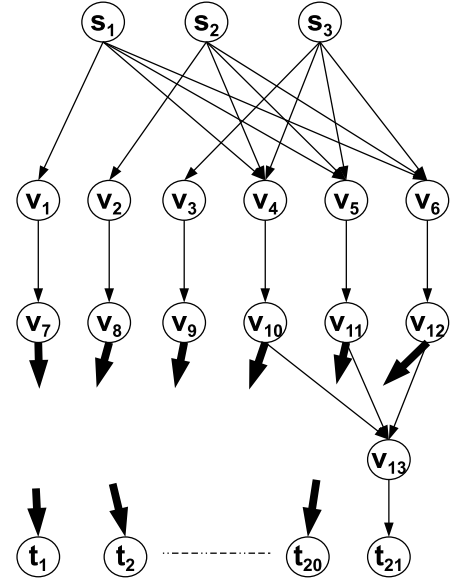*Example 4:* Consider the network $\mathcal{G}$ shown in Fig. 4,

Fig. 4. A network for which the delay-and-code scheme cannot be used to construct a feasible network code

with sources $\{s_i : 1 \leq i \leq 3\}$ and sinks $\{t_i : 1 \leq i \leq 21\}$. The source s_i generates the information sequence $x_i(z)$. The subnetwork of $\mathcal{G}$ consisting of all nodes and edges of $\mathcal{G}$ except the nodes v_{13} and sink t_{21} is derived from the $\binom{6}{3}$ combination network, i.e., for every possible three-combination of the nodes $\{v_i : 7 \leq i \leq 12\}$, there exists a sink to which there is precisely one edge incoming from the three nodes, each of which demands all three of the information sequences. There are therefore $\binom{6}{3} = 20$ such sinks, and the bolded arrows indicate the ten outgoing edges from each node $v_i : 7 \leq i \leq 12$, and the three incoming edges from $\{v_i : 7 \leq i \leq 12\}$ to each sink $t_i : 1 \leq i \leq 20$. Each of these 20 sinks demand all three information sequences. The additional sink t_{21} demands the information sequence $x_1(z)$. Note that there exists a solution to this network (for both $\mathcal{G}_{inst}$ and $\mathcal{G}_{ud}$) if the field size $q \geq 4$.

We now attempt to obtain a delay-and-code scheme on this

network (in either $\mathcal{G}_{inst}$ or $\mathcal{G}_{ud}$). Because each node v_i is only connected to the source s_i for $1 \leq i \leq 3$, each of the global encoding vectors of the outgoing edges from the nodes $\{v_i : 1 \leq i \leq 3\}$ has one component of the form z^a, a being some non-negative integer, with the other two components being zero. Also, the three global encoding vectors of the outgoing edges from the nodes $\{v_i : 4 \leq i \leq 6\}$ have to be linearly independent and of the form $\left(z^a \quad z^b \quad z^c \right)^T$, because each of these three vectors have to be linearly independent with any two of the three global encoding vectors of the outgoing edges from $\{v_i : 1 \leq i \leq 3\}$. For $4 \leq i \leq 6$, let the global encoding vector of node v_i be $\boldsymbol{f}_i = \left(z^{a_i} \quad z^{b_i} \quad z^{c_i} \right)^T$, for some non-negative integers a_i, b_i and c_i.

To satisfy the requirements for sink t_{21}, a delay-and-code based linear combination of the vectors $\boldsymbol{f}_i : 4 \leq i \leq 6$ should generate a vector of the form $\left(z^d \quad 0 \quad 0 \right)^T$, for some non-negative integer d. In other words, for some non-negative integers $\tilde{a}, \tilde{b}$ and $\tilde{c}$, and for some $m_1, m_2, m_3 \in \mathbb{F}_2$, we want

$$
\begin{pmatrix} z^{a_4} & z^{a_5} & z^{a_6} \\ z^{b_4} & z^{b_5} & z^{b_6} \\ z^{c_4} & z^{c_5} & z^{c_6} \end{pmatrix} \begin{pmatrix} m_1 z^{\tilde{a}} \\ m_2 z^{\tilde{b}} \\ m_3 z^{\tilde{c}} \end{pmatrix} = \begin{pmatrix} z^d \\ 0 \\ 0 \end{pmatrix}.
$$

Note that m_1, m_2 and m_3 cannot all be 0. They cannot all be 1, as it is not possible to find non-negative integers $\tilde{a}, \tilde{b}$ and $\tilde{c}$ such that $z^{\tilde{a}+b_4} + z^{\tilde{b}+b_5} + z^{\tilde{c}+b_6} = 0$, or such that $z^{\tilde{a}+c_4} + z^{\tilde{b}+c_5} + z^{\tilde{c}+c_6} = 0$.

Now, suppose two of m_1, m_2 and m_3 are non-zero, then this means that the global encoding vector of the outgoing edge from node v_1 lies in the space spanned by two of the global encoding vectors of the outgoing edges from $\{v_i : 4 \leq i \leq 6\}$. But this contradicts our original choice of these global encoding vectors, according to which any two are linearly independent with the global encoding vector of the outgoing edge from node v_1. Therefore, a delay-and-code based scheme cannot satisfy the requirements of all sinks in this network, in either $\mathcal{G}_{inst}$ or $\mathcal{G}_{ud}$.

V. Concluding remarks

We have discussed the effects of using the delay inherent in the network in problems related to network code existence and designs. The delay-and-code algorithms presented in this paper enable low-complexity encoding at the intermediate nodes at the cost of using large memories for decoding at the sinks. A simple upper bound for the maximum number of memory elements required at any sink to decode the information sequences which are encoded using a delay-and-code scheme can be obtained without much difficulty. Similar algorithms can be found in [18]–[20]. Also, while the equivalence between memory elements and delays might not in practice make sense as the actual value of the delay incurred in the two might not be equal, the parameter z used can be equivalently used to express both and therefore Corollary 5 still holds.

The results obtained in this work indicate that using delays in the network might be beneficial in certain situations, while being not useful in others. In any case, the delays in the network cannot be ignored for analyzing any network coding

problem. Subsequent work might include the analysis of random network coding in unit-delay networks and the study of cyclic networks in a similar manner.

References

[1] R. Ahlswede, N. Cai, R. Li and R. Yeung, "Network Information Flow", IEEE Transactions on Information Theory, vol.46, no.4, July 2000, pp. 1204-1216.

[2] N. Cai, R. Li and R. Yeung, "Linear Network Coding", IEEE Transactions on Information Theory, vol. 49, no. 2, Feb. 2003, pp. 371-381.

[3] R. Koetter and M. Medard, "An Algebraic Approach to Network Coding", IEEE/ACM Transactions on Networking, vol. 11, no. 5, Oct. 2003, pp. 782-795.

[4] S. Jaggi, P. Sanders, P.A. Chou, M. Effros, S. Egner, K. Jain, and L.M.G.M. Tolhuizen, "Polynomial-time algorithms for multicast network code construction", IEEE Transactions on Information Theory, vol. 51, no. 6, June 2005, pp.1973-1982.

[5] E. Erez and M. Feder, "Efficient Network Codes for Cyclic Networks", ISIT, 4-9 Sept. 2005, Adelaide, Australia, pp. 1982 - 1986.

[6] A. I. Barbero and O. Ytrehus, "Cycle-Logical Treatment for Cyclopathic Networks", IEEE Transactions on Information Theory, Vol. 52, No. 6, June 2006.

[7] C. Fragouli and E. Soljanin "Information Flow Decomposition for Network Coding", IEEE Transactions on Information Theory, Vol. 52, No. 3, March 2006.

[8] A. Lehman and E. Lehman, "Complexity classification of network information flow problems", SODA, 2004, New Orleans, USA, pp. 142-150.

[9] A. Tavory, M. Feder, and D. Ron, "Bounds on Linear Codes for Network Multicast", Technical Report TR03-033, Electronic Colloquium on Computational Complexity, 2003.

[10] C. Chekuri, C. Fragouli, and E. Soljanin, "On Average Throughput and Alphabet Size in Network Coding", IEEE Transactions on Information Theory, Vol. 52, No. 6, June 2006.

[11] A. Lehman and E. Lehman, "Network Coding: Does the Model Need Tuning?", SODA, 2005, Vancouver, Canada, pp. 499-504.

[12] R. Dougherty, C. Freiling, and K. Zeger, "Insufficiency of Linear Coding in Network Information Flow", IEEE Transactions on Information Theory, Vol. 51, No. 8, August 2005.

[13] X. Wu, C. Zhao and X. You, "Generation-Based Network Coding over Networks with Delay", IFIP International Conference on Network and Parallel Computing, Shangai, China, Oct. 18-21 2008, pp. 365-368.

[14] K. Prasad and B. Sundar Rajan, "Single-generation Network Coding for Networks with Delay", arXiv:0909.1638v1[cs.IT], September 2009, Available at: http://arxiv.org/abs/0909.1638.

[15] Q. T. Sun, S. Jaggi, S.-Y. R. Li, "Delay Invariant Convolutional Network Codes", Proceedings of ISIT 2011, St. Petersburg, Russia, July 31 - Aug. 5, 2011.

[16] Michelle Effros, "On Dependence and Delay: Capacity bounds for Wireless Networks", Proceedings of WCNC 2012, Paris, France, April 1-4, 2012.

[17] K. Prasad and B. Sundar Rajan, "Network error correction for unit-delay, memory-free networks using convolutional codes", arXiv:0903.1967v3[cs.IT], September 2009, Available at: http://arxiv.org/abs/0903.1967.

[18] A. I. Barbero and O. Ytrehus, "An efficient centralized binary multicast network coding algorithm for any cyclic network", arXiv:0705.0085v1[cs.IT], May 2007, Available at http://arxiv.org/abs/0705.0085v1.

[19] H. Lu, "Binary Linear Network Codes", IEEE ITW on Information Theory for Wireless Networks, 1-6 July 2007, Solstrand, Norway.

[20] A. Keshavarz-Haddad and M.A. Khojastepour, "Rotate-and-add coding: A novel algebraic network coding scheme", IEEE ITW 2010, Aug. 30 - Sept. 3, Dublin, Ireland.

[21] T. Mulders and A. Storjohann, "On lattice reduction for polynomial matrices", Journal of Symbolic Computation, Vol. 35, Issue 4, April 2003, pp. 377-401.

[22] A. Borodin and I. Munro, "The computational complexity of algebraic and numeric problems", American Elsevier Pub. Co., 1975.

Appendix A
Proof of Proposition 1

Proof: Let M'_t be the $h_t \times h_t$ submatrix of the network transfer matrix of any particular sink node $t \in \mathcal{T}$ in $\mathcal{G}_{inst}$,

involving the h_t information symbols to be inverted. Let $M_t'(z)$ be the corresponding matrix of the same sink t in $\mathcal{G}_{ud}$. We first note that the matrix M_t' can be obtained from $M_t'(z)$ by substituting $z = z^0 = 1$, i.e.,

$$M_t' = M_t'(z)|_{z=1}.$$

Given that M_t' is of full rank over $\mathbb{F}_q$, we prove that $M_t'(z)$ is of full rank over $\mathbb{F}_q(z)$ by contradiction.

Suppose that $M_t'(z)$ was not of full rank over $\mathbb{F}_q(z)$, then we have

$$\sum_{i=1}^{h_t-1} \frac{a_i(z)}{b_i(z)} \boldsymbol{M_i'}(z) = \boldsymbol{M_{h_t}'}(z), \tag{8}$$

where $\boldsymbol{M_i'}(z)$ is the ith column of $M_t'(z)$ and $a_i(z), b_i(z) \in \mathbb{F}_q[z] \ \forall \ i = 1, 2, .., h_t - 1$ are such that $b_i(z) \neq 0 \ \forall i$, and $a_i(z) \neq 0$ for at least one i, and $gcd(a_i(z), b_i(z)) = 1, \ \forall i$. We have the following two cases

Case 1: $b_i(z)|_{z=1} \neq 0 \ \forall i$.

Substituting $z = 1$ in (8), we have

$$\sum_{i=1}^{h_t-1} \frac{a_i}{b_i} \boldsymbol{M_i'} = \boldsymbol{M_{h_t}'}, \tag{9}$$

where $a_i = a_i(z)|_{z=1}, b_i = b_i(z)|_{z=1}$ and $\boldsymbol{M_i'} = \boldsymbol{M_i'}(z)|_{z=1}$ is the ith column of M_t'.

Clearly $\boldsymbol{M_{h_t}'} \neq \boldsymbol{0}$ since M_t' is of full rank, and hence the left hand side of (9) cannot be zero. Therefore, some non-zero linear combination of the first $h_t - 1$ columns of M_t' is equal to its h_t^{th} column, which contradicts the given statement that M_t' is of full rank over $\mathbb{F}_q$. Therefore, $M_t'(z)$ must be of full rank over $\mathbb{F}_q(z)$.

Case 2: $b_i(z)|_{z=1} = 0$ for at least one i.

Let $\mathcal{I}' \subseteq \{1, 2, ..., h_t\}$ be such that $(z-1)^{p'}|b_i(z)$ for some positive integer p'. Let p be an integer such that

$$p = \max_{i \in \mathcal{I}'} p'.$$

Now, from (8) we have

$$\sum_{i=1}^{h_t-1} (z-1)^p \frac{a_i(z)}{b_i(z)} \boldsymbol{M_i'}(z) = (z-1)^p \boldsymbol{M_{h_t}'}(z). \tag{10}$$

Let $\mathcal{I} \subseteq \{1, 2, .., h_t\}$ be such that $(z-1)^p|b_i(z) \ \forall \ i \in \mathcal{I}$. Then, we must have that $(z-1) \nmid a_i(z) \ \forall \ i \in \mathcal{I}$, since $gcd(a_i(z), b_i(z)) = 1$. Also, let $b_i'(z) = b_i(z)/(z-1)^p \in \mathbb{F}_q[z] \ \forall \ i \in \mathcal{I}$. Then we have

$$\left((z-1)^p \frac{a_i(z)}{b_i(z)} \right)|_{z=1} = \left(\frac{a_i(z)}{b_i'(z)} \right)|_{z=1} = \frac{a_i}{b_i'} \in \mathbb{F}_q \backslash \{0\},$$

where $b_i' = b_i'(z)|_{z=1} \in \mathbb{F}_q \backslash \{0\}$, since $(z-1) \nmid b_i'(z)$. Substituting $z = 1$ in (10), we have

$$\sum_{i \in \mathcal{I}} \frac{a_i}{b_i'} \boldsymbol{M_i'} = \boldsymbol{0},$$

i.e., a non-zero linear combination of the columns of M_t' is equal to zero, which contradicts the full-rankness of M_t', thus proving that $M_t'(z)$ has to be of full rank over $\mathbb{F}_q(z)$. As the choice of the sink t was arbitrary, this completes the proof. ∎

APPENDIX B
PROOF OF PROPOSITION 2

Proof: Let $\boldsymbol{m'}$ be a set of local encoding coefficients taking values from some field $\mathbb{F}_q$ which result in a feasible network code for $\mathcal{G}_{ud}$, satisfying the invertibility and zero-interference conditions, i.e., $f_1(\boldsymbol{m'}, z) = f_2(\boldsymbol{m'}, z) = ... = f_K(\boldsymbol{m'}, z) = 0$ and the product of the determinants $g(\boldsymbol{m'}, z) = \frac{g_n(\boldsymbol{m'}, z)}{g_d(\boldsymbol{m'}, z)} \neq 0$, where $g_n(\boldsymbol{m'}, z), g_n(\boldsymbol{m'}, z) \in \mathbb{F}_q[z]$, the ring of polynomials in variable z over $\mathbb{F}_q[z]$, are the numerator and denominator polynomials corresponding to g.

Note that, if we assign some appropriate value in $\mathbb{F}_q$ for the parameter z in (1), we get a well-defined network code for $\mathcal{G}_{inst}$. In other words, if throughout the network, we let $z = z_q \in \mathbb{F}_q$ such that $(z - z_q)$ does not divide any numerator polynomial of $m_{e,p}(z)$ corresponding to any pair of edges e and p, then the unit-delay equation

$$y_e(z) = \sum_{p \in \Gamma_I(tail(e))} m_{e,p}'(z)|_{z=z_q} y_p(z)$$

reduces to the instantaneous form, without the time index, as

$$y_e = \sum_{p \in \Gamma_I(tail(e))} m_{e,p}'' y_p,$$

where $m_{e,p}'' = m_{e,p}'(z)|_{z=z_q}$, for all pairs of edges e and p in the network. The new set of local encoding coefficients, denoted by $\boldsymbol{m''}$ is a well-defined network code for $\mathcal{G}_{inst}$ (although this might not be feasible). Using this technique, we now show that given a feasible network code ($\boldsymbol{m'}$) for $\mathcal{G}_{ud}$ over some field $\mathbb{F}_q$, we can obtain a feasible network code for $\mathcal{G}_{inst}$, over a possibly larger field $\mathbb{F}_Q$.

Given that $g(\boldsymbol{m'}, z) = \frac{g_n(\boldsymbol{m'}, z)}{g_d(\boldsymbol{m'}, z)}$, let $\mathbb{F}_Q$ be an extension of $\mathbb{F}_q$, such that $Q > degree(g_n) + degree(g_d)$. As $\mathbb{F}_q \subset \mathbb{F}_Q$, we can view the coefficients $\boldsymbol{m'}$ to be elements of $\mathbb{F}_Q$, which we shall now refer to as $\boldsymbol{m_Q'}$. We now choose some $z_Q \in \mathbb{F}_Q$ such that $g_n(\boldsymbol{m_Q'}, z)|_{z=z_Q} \neq 0$, and $g_d(\boldsymbol{m_Q'}, z)|_{z=z_Q} \neq 0$. Such a choice is possible because the polynomial $g_n(\boldsymbol{m_Q'}, z) g_d(\boldsymbol{m_Q'}, z)$ can have at most $degree(g_n) + degree(g_d)$ zeros in $\mathbb{F}_Q$. Therefore, with $z = z_Q$, we have a well-defined network code for $\mathcal{G}_{inst}$ with $g \neq 0$, satisfying the invertibility condition in $\mathcal{G}_{inst}$. Let the set of local encoding coefficients obtained for $\mathcal{G}_{inst}$ by assuming $z = z_Q$ be $\boldsymbol{m_Q''}$.

As for the zero-interference conditions, $f_1(\boldsymbol{m_Q'}, z) = f_2(\boldsymbol{m_Q'}, z) = ... = f_K(\boldsymbol{m_Q'}, z)$ all being zero polynomials implies that any choice of z does not alter their value. Therefore the network code defined by $\boldsymbol{m_Q''}$ is a feasible network code for $\mathcal{G}_{inst}(\mathcal{V}, \mathcal{E}, \mathcal{S}, \mathcal{T}, \mathcal{C})$. This completes the proof. ∎

APPENDIX C
PROOF OF COROLLARY 3

Proof: Let $\boldsymbol{m'}$ be a set of local encoding coefficients taking values from some field $\mathbb{F}_q$ which result in a feasible network code for $\mathcal{G}_{ud}$, satisfying the invertibility and zero-interference conditions. Let the resulting network transfer matrix of any particular sink $t \in \mathcal{T}$ in $\mathcal{G}_{ud}$ be $M_t(z)$ and let $M_t'(z)$ be the $h_t \times h_t$ submatrix of $M_t(z)$ which involves

those rows of $M_t(z)$ that correspond to the information sequences to be obtained by sink t. Although $M_t'(z)$ involves rational functions in z, the denominators can be factored out to obtain a matrix of the form $M_t'(z) = (a(z))^{-1}M_t''(z)$ where $a(z) \in \mathbb{F}_q[z]$ and $M_t''(z)$ is a matrix consisting of bounded-degree polynomials in z. It is known (see [21], for example) that the determinant of a polynomial matrix of size $k \times k$ and degree at d can be calculated with complexity $O(k^3 d^2)$. The determinant of $M_t'(z)$ can thus be calculated with polynomial complexity. Thus, following the notations from the proof of Proposition 2, it is therefore clear that the product $g(\boldsymbol{m'}, z) = \frac{g_n(\boldsymbol{m'}, z)}{g_d(\boldsymbol{m'}, z)}$ of the determinants of all the sinks can also be calculated in polynomial-time.

The next step in the construction of a feasible network code for $\mathcal{G}_{inst}$ according to the proof of Proposition 2 is to pick a value z_Q from a large enough field $\mathbb{F}_Q$ so that $g(\boldsymbol{m_Q'}, z_Q) \neq 0$. Finding such a z_Q involves at most $degree(g_n) + degree(g_d)$ evaluations of the polynomial $g_n g_d$. Such polynomial evaluations can be performed with complexity linear in the degree of the polynomial concerned (see [22], for example). Therefore, identifying an appropriate z_Q takes $O\left((degree(g_n) + degree(g_d))^2\right)$ operations over the field concerned.

Once such a z_Q has been identified, the local encoding coefficients $\boldsymbol{m_Q''}$ for $\mathcal{G}_{inst}$ can be obtained can be obtained by evaluating the local encoding coefficients $\boldsymbol{m_Q'}$ of $\mathcal{G}_{ud}$ at $z = z_Q$. All the elements of $\boldsymbol{m_Q'}$ are rational functions and there are at most $|\mathcal{E}|^2$ of them, thus the total complexity involved in these evaluations is also polynomial. Once these evaluations have been obtained, we have a feasible network code for $\mathcal{G}_{inst}$. The complexity of obtaining such a feasible network code for $\mathcal{G}_{inst}$ at each step has been polynomial including that of the obtaining a feasible network code for $\mathcal{G}_{ud}$, as assumed. This proves the corollary. ∎

APPENDIX D
PROOF OF PROPOSITION 3

Proof: Consider a feasible network coding solution from $\mathcal{U}$ for $\mathcal{G}_{ud}$. Because of the conditions on the topology of the network, the columns of the $h \times h_t$ (following the notations in Section II) network transfer matrix $M_t(z)$ of a sink $t \in \mathcal{T}$ are of the form

$$\boldsymbol{M_{i,t}}(z) = z^{a_i}\boldsymbol{M_{i,t}}, 1 \leq i \leq h_t \tag{11}$$

where $\boldsymbol{M_{i,t}}(z)$ is the ith column of $M_t(z)$, $\boldsymbol{M_{i,t}} \in \mathbb{F}_q^h$ and $a_i \in \mathbb{Z}^+$. We also have $M_t = M_t(z)|_{z=1}$, the network transfer matrix of the sink t in $\mathcal{G}_{inst}$, the ith column of which is $\boldsymbol{M_{i,t}}$. Let $M_t'(z)$ be the $h_t \times h_t$ submatrix of $M_t(z)$, involving those rows (say, those indexed by $\{j_i \in \{1, 2, 3, ..., h\} : 1 \leq i \leq h_t\}$) of $M_t(z)$ which correspond to the information sequences that need to be inverted at t, and let M_t' be the corresponding matrix for $\mathcal{G}_{inst}$. Because of (11), the determinant of $M_t'(z)$ is of the form

$$det(M_t'(z)) = det(M_t')\prod_{i=1}^{h_t} z^{a_{j_i}}. \tag{12}$$

Thus, if $det(M_t'(z)) \neq 0$, then $det(M_t') \neq 0$, which means M_t' is invertible. Also, any zero element of $M_t(z)$ is also zero in M_t. As the choice of the sink t was arbitrary, both the invertibility and the zero-interference conditions are satisfied for all sinks in $\mathcal{G}_{inst}$, thus proving (A).

We now prove (B). Suppose there is a feasible solution in place for $\mathcal{G}_{inst}$. Because of the condition on the network topology, the network transfer matrices in $\mathcal{G}_{ud}$ is of the form (11). Then, by (12) the corresponding invertible $h_t \times h_t$ submatrix $M_t'(z)$ of the network transfer matrix $M_t(z)$ of sink t in $\mathcal{G}_{ud}$ with the same local encoding coefficients as $\mathcal{G}_{inst}$ has a non-zero determinant and is thus full-rank. Thus the invertibility conditions for sink t are carried over to $\mathcal{G}_{ud}$. To prove the zero-interference conditions, suppose $M_{i,j,t}$ is the element (i,j) of M_t which is zero. Then the corresponding element of $M_t(z)$, $M_{i,j,t}(z)$, is such that $M_{i,j,t}(z) = 0$, or $M_{i,j,t}(z) \neq 0$ with $(z-1)|M_{i,j,t}(z)$ (as $M_t(z)|_{z=1} = M_t$). However, because of (11), $M_{i,j,t}(z) = z^{a_i}M_{i,j,t}$ which means that $(z-1) \nmid M_{i,j,t}(z)$. Thus $M_{i,j,t}(z) = 0$. The zero-interference conditions are also satisfied for t in $\mathcal{G}_{ud}$. Again, as the choice of sink t was arbitrary, any solution for $\mathcal{G}_{inst}$ is also a solution for $\mathcal{G}_{ud}$.

To prove (C), we first note that, by (B), the minimum field size requirement for a feasible solution for $\mathcal{G}_{ud}$ is not larger than that of $\mathcal{G}_{inst}$. Also, by (A), any solution for $\mathcal{G}_{ud}$ from any non-empty $\mathcal{U}_q$ is feasible for $\mathcal{G}_{inst}$, which holds for $q = q_{min}$ too. This fact along with (B) proves (C). ∎

APPENDIX E
PROOF OF PROPOSITION 4

Proof: Throughout this proof, we assume that the network we are working with is $\mathcal{G}^*(\mathcal{V}^*, \mathcal{E}^*)$, the subnetwork of $\mathcal{G}$ consisting only of the nodes and edges on the h_s edge-disjoint paths from the source s to each sink $t \in \mathcal{T}$. Just as Lemma 2 together with Lemma 1 justifies the maintenance of the rank of the matrices $B_t : t \in \mathcal{T}$ in every step of the LIF algorithm, we prove this proposition by showing a variant of Lemma 2 which will maintain the rank of the matrices $B_t : t \in \mathcal{T}$ according to the delay-and-code schemes.

Let e be the edge whose global encoding vector is to be decided in the current step of the non-uniform delay-and-code LIF algorithm, and let $v = tail(e)$. We have sets of ordered pairs (elements from $\mathbb{F}_q(z)^{h_s} \times \mathbb{F}_q(z)^{h_s}$) as in Lemma 2,

$$S(e) := \{(\boldsymbol{x_i}, \boldsymbol{y_i}) : i \leq i \leq n\}$$
$$= \left\{(\boldsymbol{b}(f_\leftarrow^t(e)), \boldsymbol{a_t}(f_\leftarrow^t(e))) : t \in T(e)\right\}, \tag{13}$$

such that $\boldsymbol{x_i}.\boldsymbol{y_i} \neq 0, 1 \leq i \leq n$, n being the cardinality of the set in the RHS. We seek to iteratively construct the vectors $\boldsymbol{u_1}, \boldsymbol{u_2}, ..., \boldsymbol{u_n}$ (each $\boldsymbol{u_i} \in \mathbb{F}_q(z)^{h_s}$) such that the following conditions hold for each $i, 1 \leq i \leq n$.

1) $\boldsymbol{u_i}$ is some delay-and-code based linear combination of the vectors $\{\boldsymbol{x_k} : 1 \leq k \leq i\}$.
2) For any j such that $1 \leq j \leq i, \boldsymbol{u_i}.\boldsymbol{y_j} \neq 0$.

If such vectors can be found, then we fix $\boldsymbol{b}(e) = \boldsymbol{u_n}$ as the global encoding vector of e as it can be seen using Lemma 1 that such a choice preserves the necessary requirements for the

current step of the non-uniform delay-and-code LIF algorithm. The vectors $\boldsymbol{u_1}, \boldsymbol{u_2}, ..., \boldsymbol{u_n}$ are constructed as follows.

Let $\boldsymbol{u_1} = \alpha_1 \boldsymbol{x_1}$, where $\alpha_1 \in \mathbb{F}_2$ and $\alpha_1 \neq 0$ as we need $\boldsymbol{u_1}.\boldsymbol{y_1} \neq 0$. Now suppose for some $i, 1 \leq i \leq n-1$, we have $\boldsymbol{u_i}$ such that $\boldsymbol{u_i}.\boldsymbol{y_j} \neq 0 \; \forall 1 \leq j \leq i$. Then we will show that we can get $\boldsymbol{u_{i+1}}$ such that $\boldsymbol{u_{i+1}}.\boldsymbol{y_j} \neq 0 \; \forall 1 \leq j \leq i+1$, as long as the total number of memory elements present at each node for each incoming edge is at least $(|\mathcal{T}| - 1)$.

Suppose $\boldsymbol{u_i}.\boldsymbol{y_{i+1}} \neq 0$. Then we choose $\boldsymbol{u_{i+1}} = \boldsymbol{u_i}$, which then satisfies our requirements. Else, we choose

$$\boldsymbol{u_{i+1}} = \boldsymbol{u_i} + \alpha_{i+1} z^{\beta_{i+1}} \boldsymbol{x_{i+1}}. \tag{14}$$

Again, we have $\alpha_{i+1} \in \mathbb{F}_2$ and $\alpha_{i+1} \neq 0$ as we want $\boldsymbol{u_{i+1}}.\boldsymbol{y_{i+1}} \neq 0$. $\beta_{i+1} \in \mathbb{Z}^{\geq 0}$ is the number of memory elements used to delay the symbols on that particular incoming edge. Thus $\boldsymbol{u_{i+1}} = \boldsymbol{u_i} + z^{\beta_{i+1}} \boldsymbol{x_{i+1}}$.

Now suppose for some choice of $\beta_{i+1} = \beta$ and for some $j, 1 \leq j \leq i$, we have $\boldsymbol{u_{i+1}}.\boldsymbol{y_j} = 0$, i.e.,

$$\boldsymbol{u_i}.\boldsymbol{y_j} = -z^\beta \left(\boldsymbol{x_{i+1}}.\boldsymbol{y_j} \right), \tag{15}$$

then β is not a valid choice for β_{i+1}, as (15) should not hold for any $j, 1 \leq j \leq i$. Note that there are at most i choices for j at which (15) will hold. There are therefore at most i choices for β_{i+1} that cannot be used. As $1 \leq i \leq n-1$, if we have at least n choices for β_{i+1}, then we can always choose one value such that (15) does not hold for any $1 \leq j \leq i$ for any given $i, 1 \leq i \leq n-1$. With $|\mathcal{T}| - 1$ memory elements for each incoming edge at node v, we have $|\mathcal{T}|$ choices for any particular β_{i+1}. This, coupled with Lemma 1 and the fact that $n \leq |\mathcal{T}|$ ensures that a non-uniform delay-and-code scheme can be constructed for the given multicast problem. ∎

APPENDIX F
PROOF OF PROPOSITION 5

Proof: As in Proposition 4, we prove the proposition using a variant of the proof for Lemma 2. We again assume that the network we are working with is $\mathcal{G}^*(\mathcal{V}^*, \mathcal{E}^*)$, the subnetwork of $\mathcal{G}$ consisting only of the nodes and edges on the h_s edge-disjoint paths from the source s to each sink $t \in \mathcal{T}$. We follow an ancestral ordering which enables us to process all outgoing edges of a particular intermediate node before moving to the next. As the uniform delay-and-code technique is defined only for the intermediate nodes, we use the non-uniform delay-and-code technique at the source node to preserve the ranks of the matrices B_t for each sink t, until all the edges in $\Gamma_O(s)$ have been processed.

Let v be the intermediate node whose outgoing edges are to be processed together. As in the proof of Proposition 4, for each $e \in \Gamma_O(v)$, we have the set $S(e)$ as defined in (13), and we seek to iteratively construct $\boldsymbol{u_1}, \boldsymbol{u_2}, ..., \boldsymbol{u_n}$, such that

conditions 1) and 2) (as in the proof of Proposition 4) are satisfied. Also, (7) needs to be satisfied because we seek to design a uniform delay-and-code.

In the process of choosing each $\boldsymbol{u_i}$ for any particular $e \in \Gamma_O(v)$, the local encoding coefficient $m_{e,p}$ and the delay $a_{e,p}$ ($m_{e,p}$ and $a_{e,p}$ are as in (6)) corresponding to e and some $p \in \Gamma_I(v)$ has to be chosen. Based on the arguments developed in the proof of Proposition 4, the choices for $m_{e,p}$ and $a_{e,p}$ are restricted. For any particular $p \in \Gamma_I(v)$ and for any edge $e \in \Gamma_O(v)$ for which $m_{e,p} \neq 0$, it was shown that there are at most $|\mathcal{T}| - 1$ choices that are not allowed for $a_{e,p}$ in the non-uniform delay-and-code case.

For the uniform delay-and-code case, we need $a_{e,p} = a_p, \forall e \in \Gamma_O(v)$ such that $m_{e,p} \neq 0$. Thus the number of choices of a_p that cannot be allowed for any edge $p \in \Gamma_I(v)$ is at most $\delta (|\mathcal{T}| - 1)$, in the case of $m_{e,p} \neq 0 \; \forall e \in \Gamma_O(v)$. If $m_{e,p} = 0$ for any $e \in \Gamma_O(v)$, this number of disallowed choices for $a_{e,p}$ can only reduce. Therefore if every node in the given network has $\delta (|\mathcal{T}| - 1)$ memory elements for each incoming edge, then there exists at least one choice for a_p such that the conditions on the invertibility of the B_t matrices (again by invoking Lemma 1) and the uniformity of the delay-and-code scheme given by (7) are satisfied.

Note that the proof hinges on the fact that the network has node-disjoint paths, as we have seen in Subsection IV-C, that a feasible uniform delay-and-code scheme might not be possible to design in a general network with only edge-disjoint paths. This concludes the proof. ∎

APPENDIX G
PROOF OF COROLLARY 5

Proof: Proposition 5 shows that a feasible network code based on the uniform delay-and-code scheme can be constructed for a multicast situation on networks with node-disjoint paths. Note that given such a network, all that the uniform delay-and-code scheme effectively does is to introduce different delays on the incoming edges. Because of the uniformity of the delay-and-code scheme, i.e., the formulation given by (7), the a_p memory elements used for each edge $p \in \Gamma_I(v)$ at some intermediate node v can be viewed as a_p additional delays on edge p, or equivalently as additional forwarding nodes with a_p forwarding edges. In other words, given an acyclic network $\mathcal{G}$ with multicast demands and node-disjoint paths, a feasible uniform delay-and-code scheme was obtained for the unit delay network $\mathcal{G}_{ud}$. Then the unit-delay network $\mathcal{G}_{ud}$ along with the uniform delay-and-code network code naturally invokes the unit-delay network $\tilde{\mathcal{G}}_{ud}$ on which there exists a feasible network code over $\mathbb{F}_2$, by using the equivalence between the memory elements and delays. ∎