Front. Electr. Electron. Eng. China 2010, 5(3): 363–390
DOI 10.1007/s11460-010-0103-1

RESEARCH ARTICLE

Raymond W. YEUNG

Network coding theory: An introduction

Abstract For a long time, *store-and-forward* had been the transport mode in network communications. In other words, information had been regarded as a commodity that only needs to be routed through the network, possibly with replication at the intermediate nodes. In the late 1990's, a new concept called *network coding* fundamentally changed the way a network can be operated. Under the paradigm of network coding, information can be processed within the network for the purpose of transmission. It was demonstrated that compared with store-and-forward, the network throughput can generally be increased by employing network coding. Since then, network coding has made significant impact on different branches of information science. The impact of network coding has gone as far as mathematics, physics, and biology. This expository work aims to be an introduction to this fast-growing subject with a detailed discussion of the basic theoretical results.

Keywords network coding, network communications, wireless communications, cryptography

1 Introduction

The concept of *network coding* was first introduced for satellite communication networks in Yeung and Zhang [1] and then fully developed in Ahlswede et al. [2], where in the latter the term "network coding" was coined. In Li et al. [3], it was established that optimality can be achieved by linear network coding.

Originally studied in information theory, network coding has made significant impact on various fields in information science (channel coding [4–6], wireless communications [7,8], computer networks [9], computer science [10], cryptography [11,12]). The impact of network coding has gone as far as mathematics (matroid theory [13], graph theory [14], game theory [15], optimization theory [16]), physics (quantum information theory [17]), and biology (cellular communications [18]). We note that the papers referenced above is very far from being an exhaustive list.

Since 2005, network coding has been under very heavy research. To date, seven special journal issues on network coding related topics [19–25] have been or will be published. There are now two annual conferences dedicated to network coding, the Workshop on Network Coding, Theory, and Applications[1)] (NetCod) since 2005 and the IEEE International Workshop on Wireless Network Coding (WiNC) since 2008. There have also been special reports on network coding by *Scientific American* [26] and *New Scientist* [27].

We first start with a historical perspective. For a point-to-point communication system, classical information theory [28] asserts that asymptotic optimality can be achieved by separating source coding and channel coding. The goal of source coding is to represent the information source in (almost) fair bits[2)] (see Ref. [29], Section 5.3). The goal of channel coding is to enable the transmission of fair bits through the channel essentially free of error with no reference to the meaning of these fair bits. Thus a theme in classical information theory for point-to-point communication is that fair bits can be drawn equivalence to a *commodity*.

It is intuitively appealing that this theme in classical information theory would continue to hold in network communication where the network consists of *noiseless* point-to-point communication channels. If so, in order to multicast[3)] information from a source node to possibly more than one sink node, we only need to compress the information at the source node into fair bits,

Received February 10, 2010; accepted March 5, 2010

Raymond W. YEUNG (✉)
Institute of Network Coding and Department of Information Engineering, The Chinese University of Hong Kong, Hong Kong, China
E-mail: whyeung@ie.cuhk.edu.hk

1) To be changed to International Symposium on Network Coding in 2010.
2) Fair bits refer to i.i.d. bits, each distributed uniformly on $\{0, 1\}$.
3) Multicast means to transmit information from a source node to a specified set of sink nodes.

organize them into data packets, and route the packets to the sink node through the intermediate nodes in the network. In the case when there are more than one sink node, the information needs to be replicated at certain intermediate nodes so that every sink node can receive a copy of the information. This method of transmitting information in a network is generally referred to as *store-and-forward* or *routing*. As a matter of fact, almost all computer networks built in the last few decades are based on this principle, where *routers* are deployed at the intermediate nodes to switch a data packet from an input channel to an output channel without processing the data content. The delivery of data packets in a computer network resembles mail delivery in a postal system. We refer the readers to textbooks on *data communication* [30,31] and *switching theory* [32,33].

However, we will see very shortly that in network communication, it does not suffice to simply route and/or replicate information within the network. Specifically, coding generally needs to be employed at the intermediate nodes in order to achieve bandwidth optimality. This notion is called *network coding*.

This paper serves as an introduction to network coding with emphasis on single-source network coding, i.e., there is only one information source in the network. The rest of the paper is organized as follows. Section 2 gives a few simple examples that explain the concept of network coding, with applications to wireless/satellite communications. Section 3 discusses linear network coding for acyclic networks. Section 4 discusses convolutional network coding for cyclic networks. Section 5 concludes the paper.

2 Some examples

In this section, the advantage of network coding over routing is explained by means of a few simple examples. The application of network coding in wireless and satellite communication is also discussed.

2.1 Butterfly network

We will use a finite directed graph to represent a point-to-point communication network. A node in the network corresponds to a vertex in the graph, while a communication channel in the network corresponds to an edge in the graph. We will not distinguish a node from a vertex, nor will we distinguish a channel from an edge. In the graph, a node is represented by a circle, with the exception that the unique source node, denoted by s (if exists), is represented by a square. Each edge is labeled by a positive integer called the *capacity*[4] or the *rate constraint*, which gives the maximum number of information symbols taken from some finite alphabet that can be transmitted over the channel per unit time. In this section, we assume that the information symbol is binary. When there is only one edge from node a to node b, we denote the edge by (a, b).

Example 1 (Butterfly network I) Consider the network in Fig. 1(a). In this network, two bits b_1 and b_2 are generated at source node s, and they are to be multicast to two sink nodes t_1 and t_2. In Fig. 1(b), we try to devise a routing scheme for this purpose. By symmetry, we send the two bits on different output channels at node s. Without loss of generality, b_1 is sent on channel $(s, 1)$ and b_2 is sent on channel $(s, 2)$. At nodes 1 and 2, the received bit is replicated and the copies are sent on the two output channels. At node 3, since both b_1 and b_2 are received but there is only one output channel, we have to choose one of the two bits to be sent on the output channel $(3, 4)$. Suppose we send b_1 as in Fig. 1(b). Then the bit is replicated at node 4 and the two copies are sent to nodes t_1 and t_2, respectively. At node t_2, both b_1 and b_2 are received. However, at node t_1, two copies of b_1 are received and b_2 cannot be recovered. Thus this routing scheme does not work. Similarly, if b_2 instead of b_1 is sent on channel $(3, 4)$, then b_1 cannot be recovered at node t_2.

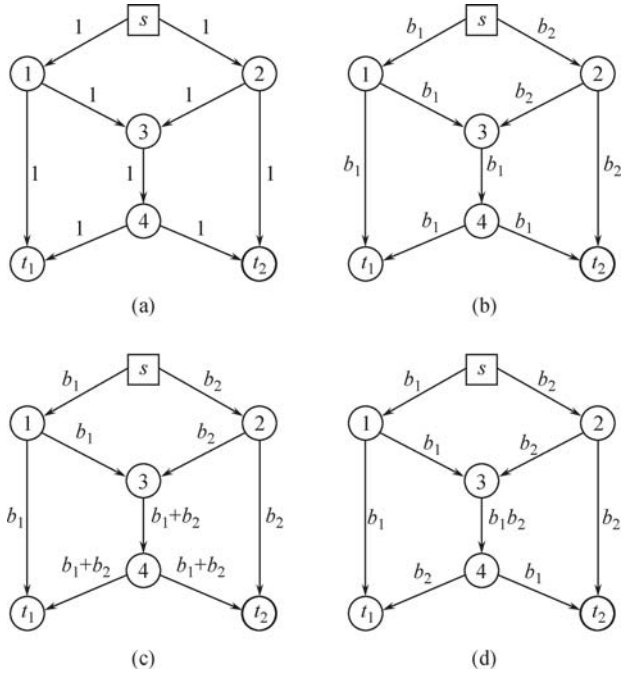


Fig. 1 Butterfly network I

However, if network coding is allowed, it is actually possible to achieve our goal. Figure 1(c) shows a scheme which multicasts both b_1 and b_2 to nodes t_1 and t_2, where '+' denotes *modulo* 2 addition. At node t_1, b_1 is received,

4) Here the term "capacity" is used in the sense of graph theory.

and b_2 can be recovered by adding b_1 and b_1+b_2, because

$$b_1 + (b_1 + b_2) = (b_1 + b_1) + b_2 = 0 + b_2 = b_2. \quad (1)$$

Similarly, b_2 is received at node t_2, and b_1 can be recovered by adding b_2 and $b_1 + b_2$.

In this scheme, b_1 and b_2 are encoded into the bit $b_1 + b_2$ which is then sent on channel $(3, 4)$. If network coding is not allowed, in order to multicast both b_1 and b_2 to nodes t_1 and t_2, at least one more bit has to be sent. Figure 1(d) shows such a scheme. In this scheme, however, the capacity of channel $(3, 4)$ is exceeded by 1 bit. If the capacity of channel $(3, 4)$ cannot be exceeded and network coding is not allowed, it is not difficult to show that at most 1.5 bits can be multicast per unit time on the average.

The above example shows the advantage of network coding over routing for a single multicast in a network. The next example shows the advantage of network coding over routing for multiple unicasts[5] in a network. These examples refute the folklore that information transmission in a point-to-point network is equivalent to a commodity flow.

Example 2 (Butterfly network II) In Fig. 1, instead of both being generated at node s, suppose bit b_1 is generated at node 1 and bit b_2 is generated at node 2. Then we can remove node s and obtain the network in Fig. 2(a). We again want to multicast b_1 and b_2 to both nodes t_1 and t_2. Since this network is essentially the same as the previous one, Fig. 2(b) shows the obvious network coding solution.

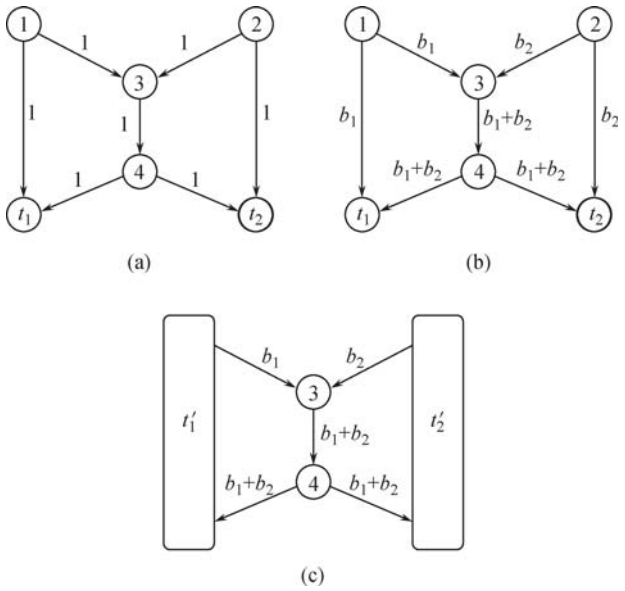


Fig. 2 Butterfly network II

There are two multicasts in this network. However, if we merge node 1 and node t_1 into a new node t_1' and merge node 2 and node t_2 into a new node t_2', then we obtain the network and the corresponding network coding solution in Fig. 2(c). In this new network, bits b_1 and b_2 are generated at nodes t_1' and t_2', respectively, and the communication goal is to exchange the two bits through the network. In other words, the two multicasts in Fig. 2(a) become two unicasts in Fig. 2(c).

If network coding is not allowed, we need to route b_1 from node t_1' to node t_2' and to route b_2 from node t_2' to node t_1'. Since each of these routes has to go through node 3 and node 4, if b_1 and b_2 are routed simultaneously, the capacity of channel $(3, 4)$ is exceeded. Therefore, we see the advantage of network coding over routing when there are multiple unicasts in the network.

2.2 Wireless and satellite communications

In wireless communication, when a node broadcasts, different noisy versions of the signal is received by the neighboring nodes. Under certain conditions, with suitable channel coding, we can assume the existence of an error-free channel between the broadcast node and the neighboring nodes such that each of the latter receives exactly the same information. Such an abstraction, though generally suboptimal, provides very useful tools for communication systems design.

Our model for network communication can be used for modeling the above broadcast scenario by imposing the following constraints on the broadcast node:

1) all the output channels have the same capacity;
2) the same symbol is sent on each of the output channels.

We will refer to these constraints as the *broadcast constraint*. Figure 3(a) is an illustration of a broadcast node b with two neighboring nodes n_1 and n_2, where the two output channels of node b have the same capacity.

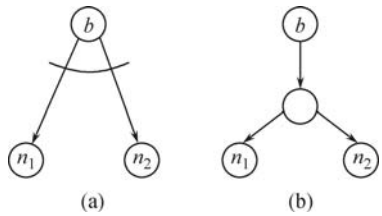


Fig. 3 A broadcast node b with two neighboring nodes n_1 and n_2

In order to express the broadcast constraint in the usual graph-theoretic terminology, we need to establish the following simple fact about network coding.

Proposition 1 Network coding is not necessary at a node if the node has only one input channel and the capacity of each output channel is the same as that of the input channel.

Proof Consider a node in the network as prescribed

5) Unicast is the special case of multicast with one sink node.

and denote the symbol(s) received on the input channel by x. (There is more than one symbol in x if the input channel has capacity larger than 1.) Let a coding scheme be given, and denote the symbol sent on the ith output channel by $g_i(x)$.

We now show that one may assume without loss of generality that x is sent on all the output channels. If x instead of $g_i(x)$ is sent on the ith output channel, then the receiving node can mimic the effect of receiving $g_i(x)$ by applying the function g_i on x upon receiving it. In other words, any coding scheme that does not send x on all the output channels can readily be converted into one which does. This proves the proposition.

We now show that the broadcast constraint depicted in Fig. 3(a) is logically equivalent to the usual graph representation in Fig. 3(b). In this figure, the unlabeled node is a dummy node associated with the broadcast node which is inserted for the purpose of modeling the broadcast constraint, where the input channel and all the output channels of the dummy node have the same capacity as an output channel of the broadcast node b in Fig. 3(a). Although no broadcast constraint is imposed on the dummy node in Fig. 3(b), by Proposition 1, we may assume without loss of generality that the dummy node simply sends the symbol received on the input channel on each of the output channels. Then Figs. 3(a) and 3(b) are logically equivalent to each other because a coding scheme for the former corresponds to a coding scheme for the latter, and vice versa.

Example 3 (A wireless/satellite system) Consider a communication system with two wireless nodes t'_1 and t'_2 that generate two bits b_1 and b_2, respectively, and the two bits are to be exchanged through a relay node. Such a system can also be the model of a satellite communication system, where the relay node corresponds to a satellite, and the two nodes t'_1 and t'_2 correspond to ground stations that communicate with each other through the satellite.

We make the usual assumption that a wireless node cannot simultaneously

1) transmit and receive;
2) receive the transmission from more than one neighboring node.

A straightforward routing scheme which takes a total of 4 time units to complete is shown in Fig. 4(a), with k being the discrete time index.

By taking into account the broadcast nature of the relay node, the system can be modeled by the network in Fig. 2(c), where node 3 corresponds to the relay node and node 4 corresponds to the associated dummy node. Then the network coding solution is shown in Fig. 4(b), which takes a total of 3 time units to complete. In other words, a very simple coding scheme at the relay node can save 50% of the downlink bandwidth. This application of network coding has recently been proposed for space communications [34].

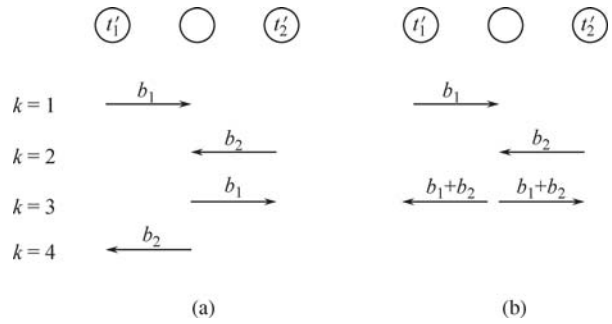


Fig. 4 A network coding application in wireless communication

Recently, Zhang et al. [35] proposed *physical-layer network coding* (PNC) that exploits the multiple access nature in wireless communication. With PNC, nodes t'_1 and t'_2 can transmit simultaneously, and therefore it takes only 2 time units to complete the communication process.

2.3 Source separation

In an error-free point-to-point communication system, suppose we want to transmit two information sources X and Y. If we compress the two sources separately, we need to transmit approximately $H(X) + H(Y)$ bits. If we compress the two sources jointly, we need to transmit approximately $H(X, Y)$ bits, where $H(\cdot)$ denotes the entropy function of a random variable. If X and Y are independent, we have

$$H(X, Y) = H(X) + H(Y). \tag{2}$$

In other words, if the information sources are independent, asymptotically there is no difference between coding them separately or jointly.

We will refer to coding independent information sources separately as *source separation*. Example 2 reveals the important fact that source separation is not necessarily optimal in network communication[6)], which is explained as follows. Let B_1 and B_2 be random bits generated at nodes t'_1 and t'_2, respectively, where B_1 and B_2 are independent and each of them are distributed uniformly on $\{0, 1\}$. With B_2 as side-information which is independent of B_1, node t'_2 has to receive at least 1 bit in order to decode B_1. Since node t'_2 can receive information only from node 4 which in turn can receive information only from node 3, any coding scheme that transmits B_1 from node t'_1 to node t'_2 must send at least 1 bit on channel $(3, 4)$. Similarly, any coding scheme that transmits B_2 from node t'_2 to node t'_1 must send

6) The suboptimality of source separation was first demonstrated by Yeung [36].

at least 1 bit on channel $(3, 4)$. Therefore, any source separation solution must send at least 2 bits on channel $(3, 4)$. Since the network coding solution in Fig. 2(c) sends only 1 bit on channel $(3, 4)$, we see that source separation is not optimal.

For a network coding problem with multiple information sources, since source separation does not guarantee optimality, the problem cannot always be decomposed into a number single-source problems. We will see in the next section that single-source network coding has a relatively simple characterization. However, the characterization of multi-source network coding is much more involved [29].

3 Linear network coding for acyclic networks

In this section, we will establish the max-flow bound as the fundamental bound for multicasting a single information source in a point-to-point communication network. In particular, we will construct *linear network codes* that achieve the max-flow bound at various levels of generality for acyclic networks.

A finite field is a system of symbols on which one can perform operations corresponding to the four operations in arithmetic for real numbers, namely addition, subtraction, multiplication, and division. The set of real numbers together with the associated operations are referred to as the field of real numbers, or simply the real field. Unlike the real field that has an infinite number of elements, a finite field has only a finite number of elements. For finite field theory, we refer the reader to Ref. [37]. For our discussions here, since we will not make use of the detailed structural properties of a finite field, the reader may by and large regard the algebra on a finite field and the algebra on the real field as the same.

In a linear network code, all the information symbols are regarded as elements of a finite field F called the *base field*. These include the symbols that comprise the information source as well as the symbols transmitted on the channels. For example, F is taken to be the binary field $GF(2)$ when the information unit is the bit. Furthermore, encoding and decoding are based on linear algebra defined on the based field, so that efficient algorithms for encoding and decoding as well as for code construction can be obtained.

In this section, we consider acyclic networks, i.e., networks with no directed cycle. We study the network coding problem in which a message consisting of a finite block of symbols is multicast. We make the ideal assumption that the propagation delay in the network, which includes the processing delay at the nodes and the transmission delay over the channels, is zero. In a general setting, a pipeline of messages may be multicast, and the propagation delay may be non-negligible. If the network is acyclic, then the operations in the network can be so synchronized that sequential messages are processed independent of each other. In this way, the network coding problem is independent of the propagation delay. Therefore, it suffices to study the network coding problem as described.

On the other hand, when a network contains directed cycles, the processing and transmission of sequential messages can convolve with together. Then the amount of delay incurred becomes part of the consideration in network coding. This will be discussed in the next section.

3.1 Acyclic networks

Denote a directed network by $G = (V, E)$, where V and E are the sets of nodes and channels, respectively. A pair of channels $(d, e) \in E \times E$ is called an *adjacent pair* if there exists a node $t \in V$ such that $d \in \mathrm{In}(t)$ and $e \in \mathrm{Out}(t)$. A *directed path* in G is a sequence of channels

$$e_1, e_2, \ldots, e_m \tag{3}$$

such that (e_i, e_{i+1}) is an adjacent pair for all $1 \leqslant i < m$. Let $e_1 \in \mathrm{Out}(t)$ and $e_m \in \mathrm{In}(t')$. The sequence in (3) is called a directed path from e_1 to e_m, or equivalently, a directed path from node t to node t'. If $t = t'$, then the directed path is called a *directed cycle*. A directed network G is *cyclic* if it contains a directed cycle, otherwise G is *acyclic*.

Acyclic networks are easier to handle because the nodes in the network can be ordered in a way which allows encoding at the nodes to be carried out in a sequential and consistent manner. The following proposition describes such an order.

Proposition 2 If G is a finite directed acyclic graph, then it is possible to order the nodes of G in a sequence such that if there is an edge from node i to node j, then node i appears before node j in the sequence.

Following the direction of the edges, we will refer to an order prescribed by Proposition 2 as an *upstream-to-downstream order*[7]. For a given acyclic network, such an order (not unique) is implicitly assumed. The nodes in the network encodes according to this order, referred to as the *encoding order*. Then whenever a node encodes, all the information needed would have already been received on the input channels of that node.

3.2 Linear network codes

In this section, we formulate a linear network code on

7) Also called an ancestral order in graph theory.

an acyclic network G. By allowing parallel channels between a pair of nodes, we assume without loss of generality that all the channels in the network have unit capacity, i.e., one symbol in the base field F can be transmitted on each channel. There exists a unique node s in G, called the source node, where a message consisting of ω symbols taken from the base field F is generated. To avoid trivially, we assume that every non-source node has at least one input channel.

We assume that there is no loop in G, and there is no input channel at node s. To facilitate our discussion, however, we let $\mathrm{In}(s)$ be a set of ω *imaginary channels* that terminate at node s but have no originating nodes. The reader may think of the ω symbols forming the message as being received by source node s on these ω imaginary channels. We emphasize that these imaginary channels are not part of the network, and the number of these channels is context dependent. Figure 5(a) illustrates the butterfly network with $\omega = 2$ imaginary channels appended at source node s.

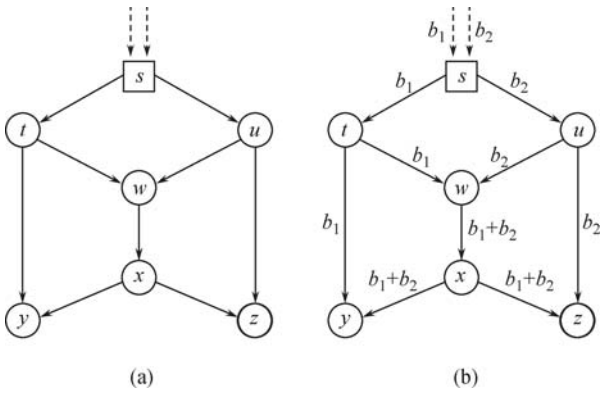


Fig. 5 (a) Two imaginary channels are appended to the source node of butterfly network; (b) a 2-dimensional network code for butterfly network

Two directed paths P_1 and P_2 in G are *edge-disjoint* if the two paths do not share a common channel. It is not difficult to see that for a non-source node t, the maximum number of edge-disjoint paths from node s to node t is equal to $\mathrm{maxflow}(t)$.

The message generated at source node s, consisting of ω symbols in the base field F, is represented by a row ω-vector $\mathbf{x} \in F^\omega$. Based on the value of $\mathbf{x}$, source node s transmits a symbol over each output channel. Encoding at the nodes in the network is carried out according to a certain upstream-to-downstream order. At a node in the network, the ensemble of received symbols is mapped to a symbol in F specific to each output channel, and the symbol is sent on that channel. The following definition of a network code formally describes this mechanism. Since the code so defined is not necessarily linear, the base field F can be regarded in this context as any finite alphabet.

Definition 1 (Local description of a network code) An ω-dimensional network code on an acyclic network over a base field F consists of a *local encoding mapping*

$$\tilde{k}_e : F^{|\mathrm{In}(t)|} \to F \tag{4}$$

for every channel e in the network, where $e \in \mathrm{Out}(t)$.

With the encoding mechanism as described, the local encoding mappings derive recursively the symbols transmitted over all channels e, denoted by $\tilde{f}_e(\mathbf{x})$. The above definition of a network code does not explicitly give the values of $\tilde{f}_e(\mathbf{x})$, whose mathematical properties are at the focus of the present discussion. Therefore, we also present an equivalent definition below, which describes a network code in terms of both the local encoding mechanisms as well as the recursively derived values $\tilde{f}_e(\mathbf{x})$.

Definition 2 (Global description of a network code) An ω-dimensional network code on an acyclic network over a base field F consists of a local encoding mapping

$$\tilde{k}_e : F^{|\mathrm{In}(t)|} \to F \tag{5}$$

and a global encoding mapping

$$\tilde{f}_e : F^\omega \to F \tag{6}$$

for each channel e in the network, where $e \in \mathrm{Out}(t)$, such that:

For every node t and every channel $e \in \mathrm{Out}(t)$, $\tilde{f}_e(\mathbf{x})$ is uniquely determined by $(\tilde{f}_d(\mathbf{x}) : d \in \mathrm{In}(t))$ via the local encoding mapping $\tilde{k}_e$. (7)

The mappings $\tilde{f}_e$ for the ω imaginary channels $e \in \mathrm{In}(s)$ project F^ω onto the distinct dimensions of F^ω.(8)

Example 4 Let $\mathbf{x} = [\,b_1 \quad b_2\,]$ denote a generic row vector in $GF(2)^2$. Figure 5(b) shows a 2-dimensional binary network code for the butterfly network with the following global encoding mappings:

$$\tilde{f}_e(x) = b_1 \text{ for } e = (o,s),(s,t),(t,w),(t,y), \tag{9}$$

$$\tilde{f}_e(x) = b_2 \text{ for } e = (o,s)',(s,u),(u,w),(u,z), \tag{10}$$

$$\tilde{f}_e(x) = b_1 + b_2 \text{ for } e = (w,x),(x,y),(x,z), \tag{11}$$

where (o,s) and $(o,s)'$ denote the two imaginary channels at node s. The corresponding local encoding mappings are

$$\tilde{k}_{(s,t)}(b_1,b_2) = b_1,\ \tilde{k}_{(s,u)}(b_1,b_2) = b_2, \tag{12}$$

$$\tilde{k}_{(t,w)}(b_1) = \tilde{k}_{(t,y)}(b_1) = b_1, \tag{13}$$

$$\tilde{k}_{(u,w)}(b_2) = \tilde{k}_{(u,z)}(b_2) = b_2, \quad \tilde{k}_{(w,x)}(b_1,b_2) = b_1 + b_2, \tag{14}$$

etc.

When a global encoding mapping $\tilde{f}_e$ is linear, it corresponds to a column ω-vector $\mathbf{f}_e$ such that $\tilde{f}_e(x)$ is the product $\mathbf{x} \cdot \mathbf{f}_e$, where the row ω-vector $\mathbf{x}$ is the message generated at node s. Similarly, when a local encoding mapping $\tilde{k}_e$, where $e \in \mathrm{Out}(t)$, is linear, it corresponds to a column $|\mathrm{In}(t)|$-vector $\mathbf{k}_e$ such that $\tilde{k}_e(\mathbf{y}) = \mathbf{y} \cdot \mathbf{k}_e$,

where $\mathbf{y} \in F^{|\mathrm{In}(t)|}$ is the row vector representing the symbols received at node t. In an ω-dimensional network code on an acyclic network, if all the local encoding mappings are linear, then so are the global encoding mappings since they are functional compositions of the local encoding mappings. The converse is also true: If the global encoding mappings are all linear, then so are the local encoding mappings. We leave the proof as an exercise.

In the following, we formulate a linear network code as a network code whose local and global encoding mappings are all linear. Again, both the local and global descriptions are presented even though they are equivalent. The global description of a linear network code will be very useful when we construct such codes in Section 3.4.

Definition 3 (Local description of a linear network code) An ω-dimensional linear network code on an acyclic network over a base field F consists of a scalar $k_{d,e}$, called the local encoding kernel, for every adjacent pair of channels (d, e) in the network. The $|\mathrm{In}(t)| \times |\mathrm{Out}(t)|$ matrix

$$K_t = [k_{d,e}]_{d \in \mathrm{In}(t), e \in \mathrm{Out}(t)} \tag{15}$$

is called the local encoding kernel at node t.

Note that the matrix structure of K_t implicitly assumes an ordering among the channels.

Definition 4 (Global description of a linear network code) An ω-dimensional linear network code on an acyclic network over a base field F consists of a scalar $k_{d,e}$ for every adjacent pair of channels (d, e) in the network as well as a column ω-vector $\mathbf{f}_e$ for every channel e such that:

$$\mathbf{f}_e = \sum\nolimits_{d \in \mathrm{In}(t)} k_{d,e}\, \mathbf{f}_d \text{ for } e \in \mathrm{Out}(t). \tag{16}$$

The vectors $\mathbf{f}_e$ for the ω imaginary channels $e \in \mathrm{In}(s)$ form the standard basis of the vector space F^ω. (17)

The vector $\mathbf{f}_e$ is called the global encoding kernel for channel e.

We now explain how the global description above specifies the linear network code. Initially, source node s generates a message $\mathbf{x}$ as a row ω-vector. In view of (17), the symbols in $\mathbf{x}$ are regarded as being received by source node s on the imaginary channels as $\mathbf{x} \cdot \mathbf{f}_d, d \in \mathrm{In}(s)$. Starting at source node s, any node t in the network receives the symbols $\mathbf{x} \cdot \mathbf{f}_d$, $d \in \mathrm{In}(t)$, from which it calculates the symbol $\mathbf{x} \cdot \mathbf{f}_e$ for sending on each channel $e \in \mathrm{Out}(t)$ via the linear formula

$$\mathbf{x} \cdot \mathbf{f}_e = \mathbf{x} \sum_{d \in \mathrm{In}(t)} k_{d,e}\, \mathbf{f}_d = \sum_{d \in \mathrm{In}(t)} k_{d,e} (\mathbf{x} \cdot \mathbf{f}_d), \tag{18}$$

where the first equality follows from (16). In this way, the symbol $\mathbf{x} \cdot \mathbf{f}_e$ is transmitted on any channel e (which may be an imaginary channel) in the network.

Given the local encoding kernels for all the channels in an acyclic network, the global encoding kernels can be calculated recursively in any upstream-to-downstream order by (16), while (17) provides the boundary conditions.

Remark A partial analogy can be drawn between the global encoding kernels for the channels in a linear network code and the columns of a generator matrix of a linear block code in algebraic coding theory [38–40]. The former are indexed by the channels in the network, while the latter are indexed by "time". However, the global encoding kernels in a linear network code are constrained by the network topology via (16), while the columns in the generator matrix of a linear block code in general are not subject to any such constraint.

The following two examples illustrate the relation between the local encoding kernels and the global encoding kernels of a linear network code. The reader should understand these two examples thoroughly before proceeding to the next section.

Example 5 The network code in Fig. 5(b) is in fact linear. Assume the alphabetical order among the channels (o, s), $(o, s)'$, (s, t), …, (x, z). Then the local encoding kernels at the nodes are:

$$K_s = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},\ K_w = \begin{bmatrix} 1 \\ 1 \end{bmatrix},$$
$$K_t = K_u = K_x = [1 \quad 1]. \tag{19}$$

The corresponding global encoding kernels are:

$$\mathbf{f}_e = \begin{cases} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \text{ for } e = (o,s), (s,t), (t,w), \text{ and } (t,y), \\ \begin{bmatrix} 0 \\ 1 \end{bmatrix} \text{ for } e = (o,s)', (s,u), (u,w), \text{ and } (u,z), \\ \begin{bmatrix} 1 \\ 1 \end{bmatrix} \text{ for } e = (w,x), (x,y), \text{ and } (x,z). \end{cases} \tag{20}$$

The local/global encoding kernels are summarized in Fig. 6. In fact, they describe a 2-dimensional linear network code regardless of the choice of the base field.

Example 6 For a general 2-dimensional linear network code on the network in Fig. 6, the local encoding kernels at the nodes can be expressed as

$$K_s = \begin{bmatrix} a & c \\ b & d \end{bmatrix},\ K_t = [e \quad f],\ K_u = [g \quad h], \tag{21}$$

$$K_w = \begin{bmatrix} i \\ j \end{bmatrix},\ K_x = [k \quad l], \tag{22}$$

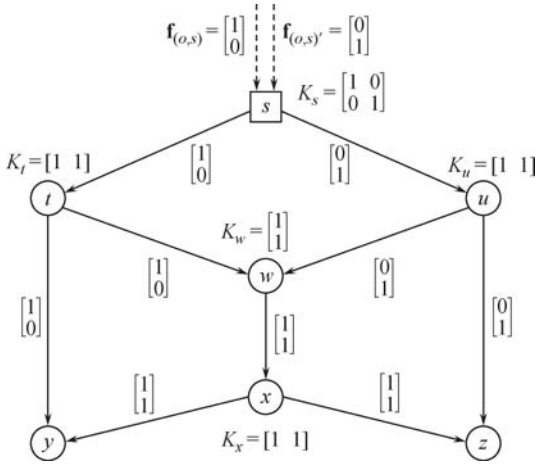


Fig. 6 Global and local encoding kernels for 2-dimensional linear network code in Example 5

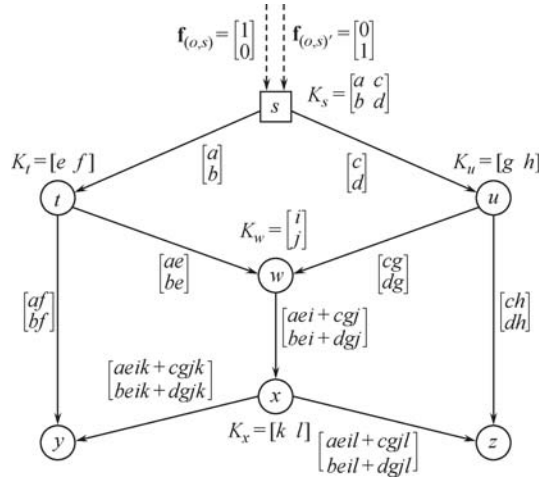


Fig. 7 Local/global encoding kernels of a general 2-dimensional linear network code

where $a, b, c, \ldots, l$, the entries of the matrices, are indeterminates in the base field F. Starting with

$$\mathbf{f}_{(o,s)}=\begin{bmatrix}1\\0\end{bmatrix} \text{ and } \mathbf{f}_{(o,s)'}=\begin{bmatrix}0\\1\end{bmatrix}, \tag{23}$$

we can obtain all the global encoding kernels below by applying (16) recursively:

$$\mathbf{f}_{(s,t)}=\begin{bmatrix}a\\b\end{bmatrix},\ \mathbf{f}_{(s,u)}=\begin{bmatrix}c\\d\end{bmatrix}, \tag{24}$$

$$\mathbf{f}_{(t,w)}=\begin{bmatrix}ae\\be\end{bmatrix},\ \mathbf{f}_{(t,y)}=\begin{bmatrix}af\\bf\end{bmatrix},$$

$$\mathbf{f}_{(u,w)}=\begin{bmatrix}cg\\dg\end{bmatrix},\ \mathbf{f}_{(u,z)}=\begin{bmatrix}ch\\dh\end{bmatrix}, \tag{25}$$

$$\mathbf{f}_{(w,x)}=\begin{bmatrix}aei+cgj\\bei+dgj\end{bmatrix}, \tag{26}$$

$$\mathbf{f}_{(x,y)}=\begin{bmatrix}aeik+cgjk\\beik+dgjk\end{bmatrix},\ \mathbf{f}_{(x,z)}=\begin{bmatrix}aeil+cgjl\\beil+dgjl\end{bmatrix}. \tag{27}$$

For example, $\mathbf{f}_{(w,x)}$ is obtained from $\mathbf{f}_{(t,w)}$ and $\mathbf{f}_{(u,w)}$ by

$$\mathbf{f}_{(w,x)}=k_{(t,w),(w,x)}\mathbf{f}_{(t,w)}+k_{(u,w),(w,x)}\mathbf{f}_{(u,w)} \tag{28}$$

$$=i\begin{bmatrix}ae\\be\end{bmatrix}+j\begin{bmatrix}cg\\dg\end{bmatrix} \tag{29}$$

$$=\begin{bmatrix}aei+cgj\\bei+dgj\end{bmatrix}. \tag{30}$$

The local/global encoding kernels of the general linear network code are summarized in Fig. 7.

3.3 Desirable properties of a linear network code

For a non-source node t, denote by $\mathrm{maxflow}(t)$ the value of a maximum flow from node s to node t. For a collection of non-source nodes T, denote by $\mathrm{maxflow}(T)$ the value of a maximum flow from node s to T.

In the sequel, we adopt the conventional notation $\langle\cdot\rangle$ for the linear span of a set of vectors. For a node t, let

$$V_t=\langle\{\mathbf{f}_e : e\in \mathrm{In}(t)\}\rangle \tag{31}$$

and for a collection T of nodes, let

$$V_T=\langle\cup_{t\in T}V_t\rangle. \tag{32}$$

For a collection ξ of channels, let

$$V_\xi=\langle\{\mathbf{f}_e : e\in\xi\}\rangle, \tag{33}$$

with the convention $V_\varnothing=\{0\}$, where 0 denotes the zero column ω-vector.

In the next theorem, we establish the *max-flow bound* for linear network coding.

Theorem 1 (Max-flow bound) For an ω-dimensional linear network code on an acyclic network, for any collection T of non-source nodes,

$$\dim(V_T)\leqslant\min\{\omega,\mathrm{maxflow}(T)\}. \tag{34}$$

Proof Let the acyclic network be $G=(V,E)$. Consider a cut U between source node s and a collection T of non-source nodes, and let E_U be the set of edges across the cut U. Then V_T is a linear transformation of V_{E_U}, where

$$\dim(V_T)\leqslant\dim(V_{E_U})\leqslant|E_U|. \tag{35}$$

Minimizing over all the cuts between s and T and invoking the max-flow min-cut theorem, we have

$$\dim(V_T)\leqslant\mathrm{maxflow}(T). \tag{36}$$

On the other hand, V_T is a linear transformation of $V_s=F^\omega$. Therefore,

$$\dim(V_T)\leqslant\dim(V_s)=\omega. \tag{37}$$

Then the proof is completed by combining (36) and (37).

For a collection of channels $\xi \subset E$ (i.e., not including the imaginary channels), we denote by maxflow(ξ) the value of a max-flow from source node s to ξ. Theorem 1 has the following straightforward corollary.

Corollary 1 For an ω-dimensional linear network code on an acyclic network, for any collection of channels $\xi \subset E$,

$$\dim(V_\xi) \leqslant \min\{\omega, \text{maxflow}(\xi)\}. \tag{38}$$

Whether the max-flow bound in Theorem 1 or Corollary 1 is achievable depends on the network topology, the dimension ω, and the coding scheme. Three special classes of linear network codes are defined below by the achievement of this bound to three different extents.

Definition 5 An ω-dimensional linear network code on an acyclic network qualifies as a *linear multicast*, a *linear broadcast*, or a *linear dispersion*, respectively, if the following hold:

$$\dim(V_t) = \omega \text{ for every non-source node } t \text{ with } \text{maxflow}(t) \geqslant \omega; \tag{39}$$

$$\dim(V_t) = \min\{\omega, \text{maxflow}(t)\} \text{ for every non-source node } t; \tag{40}$$

$$\dim(V_T) = \min\{\omega, \text{maxflow}(T)\} \text{ for every collection } T \text{ of non-source nodes.} \tag{41}$$

For a set ξ of channels, including possibly the imaginary channels, let

$$F_\xi = [\,\mathbf{f}_e\,]_{e\in\xi} \tag{42}$$

be the $\omega \times |\xi|$ matrix obtained by putting $\mathbf{f}_e, e \in \xi$ in juxtaposition. For a node t, the symbols $\mathbf{x}\cdot\mathbf{f}_e$, $e \in \text{In}(t)$ are received on the input channels. Equivalently, the row $|\text{In}(t)|$-vector

$$\mathbf{x}\cdot F_{\text{In}(t)} \tag{43}$$

is received. Obviously, the message $\mathbf{x}$, consisting of ω information units, can be uniquely determined at the node if and only if the rank of $F_{\text{In}(t)}$ is equal to ω, i.e.,

$$\dim(V_t) = \omega. \tag{44}$$

By the max-flow bound (Theorem 1), the rate at which information is transmitted from source node s to a non-source node t (a collection T of non-source nodes) cannot exceed maxflow(t) (maxflow(T)).

For a linear multicast, a node t can decode the message $\mathbf{x}$ if and only if maxflow(t) $\geqslant \omega$. For a node t with maxflow(t) $< \omega$, nothing is guaranteed. An application of an ω-dimensional linear multicast is for multicasting information at rate ω to all (or some of) those non-source nodes with max-flow at least equal to ω.

For a linear broadcast, like a linear multicast, a node t can decode the message $\mathbf{x}$ if and only if maxflow(t) $\geqslant \omega$. For a node t with maxflow(t) $< \omega$, the set of all received vectors, namely

$$\{\mathbf{x}\cdot F_{\text{In}(t)} : \mathbf{x} \in F^\omega\}, \tag{45}$$

form a vector subspace of F^ω with dimension equal to maxflow(t), but there is no guarantee on which such subspace is actually received[8]. An application of linear broadcast is for multicasting information on a network at a variable rate [41]. A random version of linear broadcast is also useful for identifying the max-flow of a non-source in an unknown network topology [42].

For a linear dispersion, a collection T of non-source nodes can decode the message $\mathbf{x}$ if and only if maxflow(T) $\geqslant \omega$. If maxflow(T) $< \omega$, the collection T receives a vector subspace with dimension equal to maxflow(T). Again, there is no guarantee on which such subspace is actually received. An application of linear dispersion is in a two-tier network system consisting of the backbone network and a number of local area networks (LANs), where each LAN is connected to one or more nodes on the backbone network. An information source with rate ω, generated at a node s in the backbone network, is to be transmitted to every user on the LANs. With a linear dispersion on the backbone network, every user on a LAN can receive the information source as long as the LAN acquires through the backbone network an aggregated max-flow from node s at least equal to ω. Moreover, new LANs can be established under the same criterion without modifying the linear dispersion on the backbone network.

Note that for all the three classes of linear network codes in Definition 5, a sink node is not explicitly identified. Also, it is immediate from the definition that every linear dispersion is a linear broadcast, and every linear broadcast is a linear multicast. The example below shows that a linear broadcast is not necessarily a linear dispersion, a linear multicast is not necessarily a linear broadcast, and a linear network code is not necessarily a linear multicast.

Example 7 Figure 8(a) shows a 2-dimensional linear dispersion on an acyclic network with the global encoding kernels as prescribed. Figure 8(b) shows a 2-dimensional linear broadcast on the same network that is not a linear dispersion because

$$\text{maxflow}(\{t, u\}) = 2 = \omega, \tag{46}$$

while the global encoding kernels of the channels in $\text{In}(t)\cup\text{In}(u)$ span only a 1-dimensional subspace. Figure 8(c) shows a 2-dimensional linear multicast that is not a linear broadcast since node u receives no information

8) Here F^ω refers to the row vector space.

at all. Finally, the 2-dimensional linear network code in Fig. 8(d) is not a linear multicast.

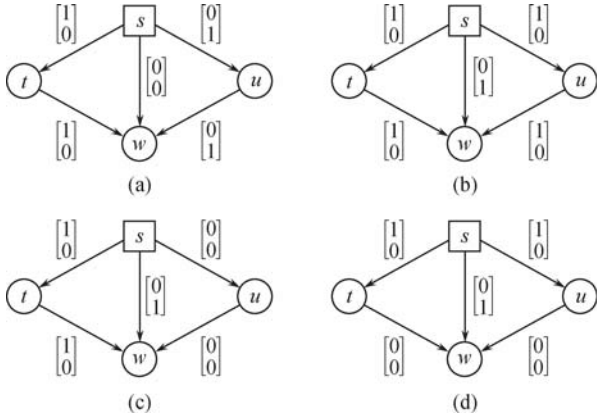


Fig. 8 (a) A 2-dimensional linear dispersion over an acyclic network; (b) a 2-dimensional linear broadcast that is not a linear dispersion; (c) a 2-dimensional linear multicast that is not a linear broadcast; (d) a 2-dimensional linear network code that is not a linear multicast

Example 8 The linear network code in Example 5 meets all the criteria (39) through (41) in Definition 5. Thus it is a 2-dimensional linear dispersion, and hence also a linear broadcast and linear multicast, regardless of the choice of the base field. The same applies to the linear network code in Fig. 8(a).

Example 9 The general linear network code in Example 6 meets the criterion (39) for a linear multicast when

- $\mathbf{f}_{(t,w)}$ and $\mathbf{f}_{(u,w)}$ are linearly independent;
- $\mathbf{f}_{(t,y)}$ and $\mathbf{f}_{(x,y)}$ are linearly independent;
- $\mathbf{f}_{(u,z)}$ and $\mathbf{f}_{(x,z)}$ are linearly independent.

Equivalently, the criterion says that $e, f, g, h, k, l, ad-bc$, $abei+adgj-baei-bcgj$, and $daei+dcgj-cbei-cdgj$ are all nonzero. Example 5 has been the special case with

$$a = d = e = f = g = h = i = j = k = l = 1 \tag{47}$$

and

$$b = c = 0. \tag{48}$$

3.3.1 Implementation of a linear network code

In implementation of a linear network code, be it a linear multicast, a linear broadcast, a linear dispersion, or any linear network code, in order that the code can be used as intended, the global encoding kernels $\mathbf{f}_e, e \in \mathrm{In}(t)$ must be known by each node t if node t is to recover any useful information from the symbols received on the input channels. These global encoding kernels can be made available ahead of time if the code is already decided. Alternatively, they can be delivered through the input channels if multiple usage of the network is allowed.

One possible way to deliver the global encoding kernels to node t in a coding session of length n, where $n > \omega$, is as follows. At time $k = 1, 2, \ldots, \omega$, the source node transmits the dummy message $\mathbf{m}_k$, a row ω-vector with all the components equal to 0 except that the kth component is equal to 1. Note that

$$\begin{bmatrix} \mathbf{m}_1 \\ \mathbf{m}_2 \\ \vdots \\ \mathbf{m}_\omega \end{bmatrix} = I_\omega, \tag{49}$$

the $\omega \times \omega$ identity matrix. At time $k = \omega + i$, where $i = 1, 2, \ldots, n - \omega$, the source node transmits the message $\mathbf{x}_i$. Then throughout the coding session, node t receives

$$\begin{bmatrix} \mathbf{m}_1 \\ \mathbf{m}_2 \\ \vdots \\ \mathbf{m}_\omega \\ \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_{n-\omega} \end{bmatrix} F_{\mathrm{In}(t)} = \begin{bmatrix} I_\omega \\ \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_{n-\omega} \end{bmatrix} F_{\mathrm{In}(t)} = \begin{bmatrix} F_{\mathrm{In}(t)} \\ \mathbf{x}_1 \cdot F_{\mathrm{In}(t)} \\ \mathbf{x}_2 \cdot F_{\mathrm{In}(t)} \\ \vdots \\ \mathbf{x}_{n-\omega} \cdot F_{\mathrm{In}(t)} \end{bmatrix} \tag{50}$$

on the input channels. In other words, the global encoding kernels of the input channels at node t are received at the beginning of the coding session. This applies to all the sink nodes in the network simultaneously because the ω dummy messages do not depend on the particular node t. If $F_{\mathrm{In}(t)}$ has full rank, then node t can start to decode $\mathbf{x}_1$ upon receiving $\mathbf{x}_1 \cdot F_{\mathrm{In}(t)}$.

Since $n - \omega$ messages are transmitted in a coding session of length n, the utilization of the network is equal to $(n - \omega)/n$, which tends to 1 as $n \to \infty$. That is, the overhead for delivering the global encoding kernels through the network is asymptotically negligible.

3.4 Existence and construction

For a given acyclic network, the following three factors dictate the existence of an ω-dimensional linear network code with a prescribed set of desirable properties:

- the value of ω,
- the network topology,
- the choice of the base field F.

We begin with an example illustrating the third factor.

Example 10 On the network in Fig. 9, a 2-dimensional ternary linear multicast can be constructed by the following local encoding kernels at the nodes:

$$K_s = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 2 \end{bmatrix} \text{ and } K_{u_i} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \tag{51}$$

for $1 \leqslant i \leqslant 4$. On the other hand, we can prove the nonexistence of a 2-dimensional binary linear multicast on this network as follows. Assuming the contrary that a 2-dimensional binary linear multicast exists, we will derive a contradiction. Let the global encoding kernel $\mathbf{f}_{(s,u_i)} = [\,y_i\ z_i\,]^\top$ for $1 \leqslant i \leqslant 4$. Since $\text{maxflow}(t_k) = 2$ for all $1 \leqslant k \leqslant 6$, the global encoding kernels for the two input channels to each node t_k must be linearly independent. Thus, if node t_k is at the downstream of both nodes u_i and u_j, then the two vectors $[\,y_i\ z_i\,]^\top$ and $[\,y_j\ z_j\,]^\top$ must be linearly independent. As each node t_k is at the downstream of a different pair of nodes among u_1, u_2, u_3, and u_4, the four vectors $[\,y_i\ z_i\,]^\top$, $1 \leqslant i \leqslant 4$, are pairwise linearly independent, and consequently, must be four distinct vectors in $GF(2)^2$. Then one of them must be $[\,0\ 0\,]^\top$ since there are only four vectors in $GF(2)^2$. This contradicts the pairwise linear independence among the four vectors.

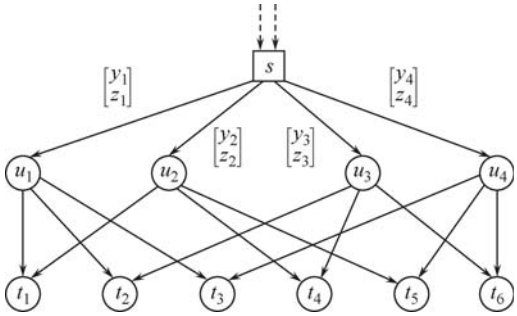


Fig. 9 A network with a 2-dimensional ternary linear multicast but without a 2-dimensional binary linear multicast

In order for the linear network code to qualify as a linear multicast, a linear broadcast, or a linear dispersion, it is required that certain collections of global encoding kernels span the maximum possible dimensions. This is equivalent to certain polynomial functions taking nonzero values, where the indeterminates of these polynomials are the local encoding kernels. To fix ideas, take $\omega = 3$, consider a node t with two input channels, and put the global encoding kernels of these two channels in juxtaposition to form a 3×2 matrix. Then, this matrix attains the maximum possible rank of 2 if and only if there exists a 2×2 submatrix with nonzero determinant.

According to the local description, a linear network code is specified by the local encoding kernels, and the global encoding kernels can be derived recursively in an upstream-to-downstream order. From Example 5, it is not hard to see that every component in a global encoding kernel is a polynomial function whose indeterminates are the local encoding kernels.

When a nonzero value of a polynomial function is required, it does not merely mean that at least one coefficient in the polynomial is nonzero. Rather, it means a way to choose scalar values for the indeterminates so that the polynomial function is evaluated to a nonzero scalar value.

When the base field is small, certain polynomial equations may be unavoidable. For instance, for any prime number p, the polynomial equation $z^p - z = 0$ is satisfied for any $z \in GF(p)$. The nonexistence of a binary linear multicast in Example 10 can also trace its root to a set of polynomial equations that cannot be avoided simultaneously over $GF(2)$.

However, when the base field is sufficiently large, every nonzero polynomial function can indeed be evaluated to a nonzero value with a proper choice of the values taken by the set of indeterminates involved. This is formally stated in the following elementary lemma, which will be instrumental in the proof of Theorem 2 asserting the existence of a linear multicast on an acyclic network when the base field is sufficiently large.

Lemma 1 Let $g(z_1, z_2, \ldots, z_n)$ be a nonzero polynomial with coefficients in a field F. If $|F|$ is greater than the degree of g in every z_j for $1 \leqslant j \leqslant n$, then there exist $a_1, a_2, \ldots, a_n \in F$ such that

$$g(a_1, a_2, \ldots, a_n) \neq 0. \tag{52}$$

Proof The lemma is proved by induction on n. For $n = 0$, g is a nonzero constant in F, and the lemma is obviously true. Assume that the lemma is true for $n - 1$ for some $n \geqslant 1$. Express $g(z_1, z_2, \ldots, z_n)$ as a polynomial in z_n with coefficients in the polynomial ring $F[z_1, z_2, \ldots, z_{n-1}]$, i.e.,

$$g(z_1, z_2, \ldots, z_n) = h(z_1, z_2, \ldots, z_{n-1}){z_n}^k + \cdots, \tag{53}$$

where k is the degree of g in z_n and the leading coefficient $h(z_1, z_2, \ldots, z_{n-1})$ is a nonzero polynomial in $F[z_1, z_2, \ldots, z_{n-1}]$. By the induction hypothesis, there exist $a_1, a_2, \ldots, a_{n-1} \in F$ such that $h(a_1, a_2, \ldots, a_{n-1}) \neq 0$. Thus $g(a_1, a_2, \ldots, a_{n-1}, z)$ is a nonzero polynomial in z with degree $k < |F|$. Since this polynomial cannot have more than k roots in F and $|F| > k$, there exists $a_n \in F$ such that

$$g(a_1, a_2, \ldots, a_{n-1}, a_n) \neq 0. \tag{54}$$

Corollary 2 Let $g(z_1, z_2, \ldots, z_n)$ be a nonzero polynomial with coefficients in a field F with $|F| > m$, where m is the highest degree of g in z_j for $1 \leqslant j \leqslant n$. Let $a_1, a_2, \ldots, a_n$ be chosen independently according to the uniform distribution on F. Then

$$\Pr\{g(a_1, a_2, \ldots, a_n) \neq 0\} \geqslant \left(1 - \frac{m}{|F|}\right)^n. \tag{55}$$

In particular,

$$\Pr\{g(a_1, a_2, \ldots, a_n) \neq 0\} \to 1 \tag{56}$$

as $|F| \to \infty$.

Proof The first part of the corollary is proved by induction on n. For $n = 0$, g is a nonzero constant in F,

and the proposition is obviously true. Assume that the proposition is true for $n-1$ for some $n \geqslant 1$. From (53) and the induction hypothesis, we see that

$$\Pr\{g(z_1, z_2, \ldots, z_n) \neq 0\} \tag{57}$$
$$= \Pr\{h(z_1, z_2, \ldots, z_{n-1}) \neq 0\} \tag{58}$$
$$\cdot \Pr\{g(z_1, z_2, \ldots, z_n) \neq 0 | h(z_1, z_2, \ldots, z_{n-1}) \neq 0\} \tag{59}$$
$$\geqslant \left(1 - \frac{m}{|F|}\right)^{n-1} \Pr\{g(z_1, z_2, \ldots, z_n) \neq 0 | h(z_1, z_2, \ldots, z_{n-1}) \neq 0\} \tag{60}$$
$$\geqslant \left(1 - \frac{m}{|F|}\right)^{n-1} \left(1 - \frac{m}{|F|}\right) \tag{61}$$
$$= \left(1 - \frac{m}{|F|}\right)^{n}. \tag{62}$$

This proves the first part of the corollary. As n is fixed, the lower bound above tends to 1 as $|F| \to \infty$. This completes the proof.

Example 11 Recall the 2-dimensional linear network code in Example 6 that is expressed in the 12 indeterminates $a, b, c, \ldots, l$. Place the vectors $\mathbf{f}_{(t,w)}$ and $\mathbf{f}_{(u,w)}$ in juxtaposition into the 2×2 matrix

$$L_w = \begin{bmatrix} ae & cg \\ be & dg \end{bmatrix}, \tag{63}$$

the vectors $\mathbf{f}_{(t,y)}$ and $\mathbf{f}_{(x,y)}$ into the 2×2 matrix

$$L_y = \begin{bmatrix} af & aeik + cgjk \\ bf & beik + dgjk \end{bmatrix}, \tag{64}$$

and the vectors $\mathbf{f}_{(u,z)}$ and $\mathbf{f}_{(x,z)}$ into the 2×2 matrix

$$L_z = \begin{bmatrix} aeil + cgjl & ch \\ beil + dgjl & dh \end{bmatrix}. \tag{65}$$

Clearly,

$$\det(L_w) \cdot \det(L_y) \cdot \det(L_z) \neq 0 \in F[a, b, c, \ldots, l]. \tag{66}$$

Applying Lemma 1 to the polynomial on the left-hand side above, we can set scalar values for the 12 indeterminates so that it is evaluated to a nonzero value in F when F is sufficiently large. This implies that the determinants on the left-hand side of (66) are evaluated to nonzero values in F simultaneously. Thus these scalar values yield a 2-dimensional linear multicast. In fact,

$$\det(L_w) \cdot \det(L_y) \cdot \det(L_z) = 1 \tag{67}$$

when

$$b = c = 0 \tag{68}$$

and

$$a = d = e = f = \cdots = l = 1. \tag{69}$$

Therefore, the 2-dimensional linear network code depicted in Fig. 6 is a linear multicast, and this fact is regardless of the choice of the base field F.

Theorem 2 There exists an ω-dimensional linear multicast on an acyclic network for sufficiently large base field F.

A stronger version of this theorem was first proved by Li et al. [3] by means of a vector space approach. The proof given below, based on a matrix approach, is due to Koetter and Médard [43].

Proof For a directed path $P = e_1, e_2, \ldots, e_m$, define

$$K_P = \prod_{1 \leqslant j < m} k_{e_j, e_{j+1}}. \tag{70}$$

Calculating by (16) recursively from the upstream channels to the downstream channels, it is not hard to find that

$$\mathbf{f}_e = \sum_{d \in \mathrm{In}(s)} \left(\Sigma_{P:\text{ a path from } d \text{ to } e} K_P\right) \mathbf{f}_d \tag{71}$$

for every channel e (see Example 12 below). Denote by $F[*]$ the polynomial ring over the field F with all the $k_{d,e}$ as indeterminates, where the total number of such indeterminates is equal to $\sum_t |\mathrm{In}(t)| \cdot |\mathrm{Out}(t)|$. Thus, every component of every global encoding kernel belongs to $F[*]$. The subsequent arguments in this proof actually depend on this fact alone but not on the exact form of (71).

Let t be a non-source node with $\mathrm{maxflow}(t) \geqslant \omega$. Then there exist ω edge-disjoint paths from the ω imaginary channels to ω distinct channels in $\mathrm{In}(t)$. Put the global encoding kernels of these ω channels in juxtaposition to form an $\omega \times \omega$ matrix L_t. We will prove that

$$\det(L_t) = 1 \tag{72}$$

for properly set scalar values of the indeterminates.

Toward proving this claim, we set

$$k_{d,e} = 1 \tag{73}$$

for all adjacent pairs of channels (d, e) along any one of the ω edge-disjoint paths, and set

$$k_{d,e} = 0 \tag{74}$$

otherwise. With such local encoding kernels, the symbols sent on the ω imaginary channels at source node s are routed to node t via the ω edge-disjoint paths. Thus the columns in L_t are simply the global encoding kernels of the imaginary channels, which form the standard basis of the space F^ω. Then (72) follows, and the claim is proved.

Consequently,

$$\det(L_t) \neq 0 \in F[*], \tag{75}$$

i.e., $\det(L_t)$ is a nonzero polynomial in the indeterminates $k_{d,e}$. Since this conclusion applies to every non-source node t with $\mathrm{maxflow}(t) \geqslant \omega$,

$$\prod_{t:\mathrm{maxflow}(t) \geqslant \omega} \det(L_t) \neq 0 \in F[*]. \tag{76}$$

Applying Lemma 1 to the above polynomial when the field F is sufficiently large, we can set scalar values in F for the indeterminates so that

$$\prod_{t:\text{maxflow}(t)\geqslant\omega} \det(L_t) \neq 0 \in F, \tag{77}$$

which in turns implies that

$$\det(L_t) \neq 0 \in F \tag{78}$$

for all t such that $\text{maxflow}(t) \geqslant \omega$. These scalar values then yield a linear network code that meets the requirement (39) for a linear multicast.

Corollary 3 There exists an ω-dimensional linear broadcast on an acyclic network for sufficiently large base field F.

Proof For every non-source node t in the given acyclic network, install a new node t' and ω input channels to this new node, with $\min\{\omega, \text{maxflow}(t)\}$ of them from node t and the remaining $\omega - \min\{\omega, \text{maxflow}(t)\}$ from source node s. This constructs a new acyclic network. Now consider an ω-dimensional linear multicast on the new network whose existence follows from Theorem 2. For every node t' as described above, $\dim(V_{t'}) = \omega$ because $\text{maxflow}(t') = \omega$. Moreover, since $|\text{In}(t')| = \omega$, the global encoding kernels $\mathbf{f}_e, e \in \text{In}(t')$ are linearly independent. Therefore,

$$\begin{aligned}&\dim(\langle\{\mathbf{f}_e : e \in \text{In}(t') \cap \text{Out}(t)\}\rangle)\\ &\quad= |\text{In}(t') \cap \text{Out}(t)| &(79)\\ &\quad= \min\{\omega, \text{maxflow}(t)\}. &(80)\end{aligned}$$

By (16),

$$\langle\{\mathbf{f}_e : e \in \text{In}(t') \cap \text{Out}(t)\}\rangle \subset V_t. \tag{81}$$

Therefore,

$$\begin{aligned}\dim(V_t) &\geqslant \dim(\langle\{\mathbf{f}_e : e \in \text{In}(t') \cap \text{Out}(t)\}\rangle) &(82)\\ &= \min\{\omega, \text{maxflow}(t)\}. &(83)\end{aligned}$$

Then by invoking Theorem 1, we conclude that

$$\dim(V_t) = \min\{\omega, \text{maxflow}(t)\}. \tag{84}$$

In other words, an ω-dimensional linear multicast on the new network incorporates an ω-dimensional linear broadcast on the original network.

Corollary 4 There exists an ω-dimensional linear dispersion on an acyclic network for sufficiently large base field F.

Proof For every nonempty collection T of non-source nodes in the given acyclic network, install a new node u_T and $\text{maxflow}(t)$ channels from every node $t \in T$ to this new node. This constructs a new acyclic network with

$$\text{maxflow}(u_T) = \text{maxflow}(T) \tag{85}$$

for every T. Now consider an ω-dimensional linear broadcast on the new network whose existence follows from Corollary 3. By (16),

$$V_{u_T} \subset V_T. \tag{86}$$

Then

$$\begin{aligned}\dim(V_T) &\geqslant \dim(V_{u_T}) &(87)\\ &= \min\{\omega, \text{maxflow}(u_T)\} &(88)\\ &= \min\{\omega, \text{maxflow}(T)\}. &(89)\end{aligned}$$

By invoking Theorem 1, we conclude that

$$\dim(V_T) = \min\{\omega, \text{maxflow}(T)\}. \tag{90}$$

In other words, an ω-dimensional linear broadcast on the new network incorporates an ω-dimensional linear dispersion on the original network.

Example 12 We now illustrate the formula (71) in the proof of Theorem 2 with the 2-dimensional linear network code in Example 6 which is expressed in the 12 indeterminates $a, b, c, \ldots, l$. The local encoding kernels at the nodes are

$$K_s = \begin{bmatrix} a & c \\ b & d \end{bmatrix},\ K_t = [\,e \quad f\,],\ K_u = [\,g \quad h\,], \tag{91}$$

$$K_w = \begin{bmatrix} i \\ j \end{bmatrix},\ K_x = \begin{bmatrix} k & l \end{bmatrix}. \tag{92}$$

Starting with $\mathbf{f}_{(o,s)} = [\,1 \quad 0\,]^\top$ and $\mathbf{f}_{(o,s)'} = [\,0 \quad 1\,]^\top$, we can calculate the global encoding kernels by the formula (71). Take $\mathbf{f}_{(x,y)}$ as the example. There are two paths from (o,s) to (x,y) and two from $(o,s)'$ to (x,y). For these paths,

$$K_P = \begin{cases} aeik & \text{for } P = (o,s),(s,t),(t,w),(w,x),(x,y),\\ beik & \text{for } P = (o,s)',(s,t),(t,w),(w,x),(x,y),\\ cgjk & \text{for } P = (o,s),(s,u),(u,w),(w,x),(x,y),\\ dgjk & \text{for } P = (o,s)',(s,u),(u,w),(w,x),(x,y). \end{cases} \tag{93}$$

Thus

$$\begin{aligned}\mathbf{f}_{(x,y)} &= (aeik)\mathbf{f}_{(o,s)} + (beik)\mathbf{f}_{(o,s)'}\\ &\quad + (cgjk)\mathbf{f}_{(o,s)} + (dgjk)\mathbf{f}_{(o,s)'} &(94)\\ &= \begin{bmatrix} aeik + cgjk \\ beik + dgjk \end{bmatrix}, &(95)\end{aligned}$$

which is consistent with Example 6.

The proof of Theorem 2 provides an algorithm for constructing a linear multicast that uses Lemma 1 as a subroutine to search for scalars $a_1, a_2, \ldots, a_n \in F$ such that $g(a_1, a_2, \ldots, a_n) \neq 0$ whenever $g(z_1, z_2, \ldots, z_n)$ is a nonzero polynomial over a sufficiently large field F. The straightforward implementation of this subroutine is exhaustive search, which is generally computationally inefficient. Nevertheless, the proof of Theorem 2 renders a simple method to construct a linear multicast randomly.

Corollary 5 Consider an ω-dimensional linear network code on an acyclic network. By choosing the local encoding kernels $k_{d,e}$ for all adjacent pairs of channels (d, e) independently according to the uniform distribution on the base field F, a linear multicast can be constructed with probability tends to 1 as $|F| \to \infty$.

Proof This follows directly from Corollary 2 and the proof of Theorem 2.

The technique described in the above theorem for constructing a linear network code is called *random network coding*. Random network coding has the advantage that the code construction can be done independent of the network topology, making it very useful when the network topology is unknown. A case study for an application of random network coding will be presented in Section 3.5.

While random network coding offers a simple construction and more flexibility, a much larger base field is usually required. In some applications, it is necessary to verify that the code randomly constructed indeed possesses the desired properties. Such a task can be computationally non-trivial.

The next algorithm, due to Jaggi and Sanders et al. [44], constructs a linear multicast deterministically in polynomial time. Unlike the algorithm given in the proof of Theorem 2 that assigns values to the local encoding kernels, this algorithm assigns values to the global encoding kernels.

Algorithm 1 (Jaggi-Sanders algorithm) This algorithm constructs an ω-dimensional linear multicast over a finite field F on an acyclic network when $|F| > \eta$, the number of non-source nodes t in the network with $\text{maxflow}(t) \geqslant \omega$. Denote these η non-source nodes by $t_1, t_2, \ldots, t_\eta$.

A sequence of channels $e_1, e_2, \ldots, e_l$ is called a *path* leading to a node t_q if $e_1 \in \text{In}(s)$, $e_l \in \text{In}(t_q)$, and (e_j, e_{j+1}) is an adjacent pair for all $1 \leqslant j \leqslant l-1$. For each q, $1 \leqslant q \leqslant \eta$, there exist ω edge-disjoint paths $P_{q,1}, P_{q,2}, \ldots, P_{q,\omega}$ leading to t_q. All together there are $\eta\omega$ such paths. The following procedure assigns a global encoding kernel $\mathbf{f}_e$ for every channel e in the network in an upstream-to-downstream order so that $\dim(V_{t_q}) = \omega$ for $1 \leqslant q \leqslant \eta$.

```
{
        // By definition, the global encoding kernels of
        // the ω imaginary channels form the standard
        // basis of F^ω.
  for (q = 1; q ⩽ η; q++)
    for (i = 1; i ⩽ ω; i++)
      e_{q,i} = the imaginary channel initiating path P_{q,i};
        // This initializes e_{q,i}. Subsequently, e_{q,i} will
        // be dynamically updated by moving down
        // path P_{q,i} until it finally becomes a channel
        // in In(t_q).
  for (every node t, in any upstream-to-downstream
       order)
  {
    for (every channel e ∈ Out(t))
    {
        // With respect to this channel e, define a
        // "pair" as a pair (q, i) of indices such that
        // channel e is on the path P_{q,i}. Note that for
        // each q, there exists at most one pair (q, i).
        // Thus the number of pairs is at least 0 and
        // at most η. Since the nodes t are chosen in
        // an upstream-to-downstream order, if (q, i)
        // is a pair, then e_{q,i} ∈ In(t) by induction, so
        // that f_{e_{q,i}} ∈ V_t. For reasons to be explained
        // in the algorithm verification below,
        // f_{e_{q,i}} ∉ ⟨{f_{e_{q,j}} : j ≠ i}⟩, and therefore
        // f_{e_{q,i}} ∈ V_t \ ⟨{f_{e_{q,j}} : j ≠ i}⟩.
      Choose a vector w in V_t such that w ∉ ⟨{f_{e_{q,j}} :
j ≠ i}⟩
      for every pair (q, i);
        // To see the existence of such a vector w, let
        // dim(V_t) = ν. Then, dim(V_t ∩ ⟨{f_{e_{q,j}} : j ≠
        // i}⟩) ⩽ ν − 1 for every pair (q, i) since
        // f_{e_{q,i}} ∈ V_t \ ⟨{f_{e_{q,j}} : j ≠ i}⟩. Thus
        // |V_t ∩ (∪_{(q,i): a pair} ⟨{f_{e_{q,j}} : j ≠ i}⟩)|
        // ⩽ η|F|^{ν−1} < |F|^ν = |V_t|.
      f_e = w;
        // This is equivalent to choosing scalar values
        // for local encoding kernels k_{d,e} for all d ∈
        // In(t) such that Σ_{d∈In(t)} k_{d,e} f_d ∉ ⟨{f_{e_{q,j}} :
        // j ≠ i}⟩ for every pair (q, i).
      for (every pair (q, i))
        e_{q,i} = e;
    }
  }
}
```

Algorithm verification For $1 \leqslant q \leqslant \eta$ and $1 \leqslant i \leqslant \omega$, the channel $e_{q,i}$ is on the path $P_{q,i}$. Initially $e_{q,i}$ is an imaginary channel at source node s. Through dynamic updating, it moves downstream along the path until finally reaching a channel in $\text{In}(t_q)$.

Fix an index q, where $1 \leqslant q \leqslant \eta$. Initially, the vectors $\mathbf{f}_{e_{q,1}}, \mathbf{f}_{e_{q,2}}, \ldots, \mathbf{f}_{e_{q,\omega}}$ are linearly independent because

they form the standard basis of F^ω. At the end, they need to span the vector space F^ω. Therefore, in order for the eventually constructed linear network code to qualify as a linear multicast, it suffices to show the preservation of the linear independence among $\mathbf{f}_{e_{q,1}}, \mathbf{f}_{e_{q,2}}, \ldots, \mathbf{f}_{e_{q,\omega}}$ throughout the algorithm.

We need to show the preservation in the generic step inside the "for loop" for each channel e in the algorithm. The algorithm defines a "pair" as a pair (q, i) of indices such that channel e is on path $P_{q,i}$. When no (q, i) is a pair for $1 \leqslant i \leqslant \omega$, the channels $e_{q,1}, e_{q,2}, \ldots, e_{q,\omega}$ are not changed in the generic step; neither are the vectors $\mathbf{f}_{e_{q,1}}, \mathbf{f}_{e_{q,2}}, \ldots, \mathbf{f}_{e_{q,\omega}}$. So we only need to consider the scenario that a pair (q, i) exists for some i. The only change among the channels $e_{q,1}, e_{q,2}, \ldots, e_{q,\omega}$ is that $e_{q,i}$ becomes e. Meanwhile, the only change among the vectors $\mathbf{f}_{e_{q,1}}, \mathbf{f}_{e_{q,2}}, \ldots, \mathbf{f}_{e_{q,\omega}}$ is that $\mathbf{f}_{e_{q,i}}$ becomes a vector

$$w \notin \langle \{\mathbf{f}_{e_{q,j}} : j \neq i\} \rangle. \tag{96}$$

This preserves the linear independence among $\mathbf{f}_{e_{q,1}}, \mathbf{f}_{e_{q,2}}, \ldots, \mathbf{f}_{e_{q,\omega}}$ as desired.

Complexity analysis There are a total of $|E|$ channels in the network. In the algorithm, the generic step in the "for loop" for each channel e processes at most η pairs. Throughout the algorithm, at most $|E|\eta$ such collections of channels are processed. From this, it is not hard to implement the algorithm within a polynomial time in $|E|$ for a fixed ω. The computational details can be found in Ref. [44].

Remark 1 In the Jaggi-Sanders algorithm, all nodes t in the network with $\text{maxflow}(t) \geqslant \omega$ serve as a sink node that receives the message $\mathbf{x}$. The algorithm can easily be modified accordingly if only a subset of such nodes need to serve as a sink node. In that case, the field size requirement is $|F| > \eta'$, where η' is the total number of sink nodes.

Remark 2 It is not difficult to see from the lower bound on the required field size in the Jaggi-Sanders algorithm that if a field much larger than sufficient is used, then a linear multicast can be constructed with high probability by randomly choosing the global encoding kernels.

3.5 Random network coding: A case study

We have seen in Corollary 5 that if the local encoding kernels of a linear network code are randomly chosen, a linear multicast can be obtained with high probability provided that the base field is sufficiently large. Since the code construction is independent of the network topology, the network code so constructed can be used when the network topology is unknown. This technique, called *random network coding*, was proposed by Ho et al. [45].

In this section, we study an application of random network coding in *peer-to-peer* (P2P) networks. The model to be presented is based on a system called *Avalanche* for file distribution proposed by Gkantsidis and Rodriguez [9]. Avalanche can be regarded as a network-coded version of BitTorrent [46], and it has been further developed into Microsoft Secure Content Distribution (MSCD) that has been trialed on the Internet for software distribution[9]. A similar system for video streaming has been deployed by UUSee.com [47]. The analysis of such systems presented here is due to Yeung [48].

3.5.1 How the system works

A file originally residing on a single server is to be distributed to a large number of users through a network. The server divides the file into k data blocks, $B_1, B_2, \ldots, B_k$, and uploads coded versions of these blocks to different users according to some protocol. These users again help distributing the file by uploading blocks to other users in the network. By means of such repeated operations, a logical network called an *overlay network* is formed by the users as the process evolves. On this logical network, henceforth referred to as the network, information can be dispersed very rapidly, and the file is eventually delivered to every user in the network. Note that the topology of the network is not known ahead of time.

In the system, new users can join the network as a node at any time as long as the distribution process is active. Upon arrival, a new user will contact a designated node called the *tracker* that provides a subset of the other users already in the system forming the set of neighboring nodes of the new user. Subsequent information flow in the network is possible only between neighboring nodes.

For the purpose of coding, the data blocks $B_1, B_2, \ldots, B_k$ are represented as symbols in a large finite field F referred to as the base field[10]. At the beginning of the distribution process, a client A contacts the server and receives a number of *encoded blocks*. For example, the server uploads two encoded blocks E_1 and E_2 to client A, where for $i = 1, 2$,

$$E_i = c_1^i B_1 + c_2^i B_2 + \cdots + c_k^i B_k, \tag{97}$$

with c_j^i, $1 \leqslant j \leqslant k$, being chosen randomly from the base field F. Note that each E_1 and E_2 is some random linear combination of $B_1, B_2, \ldots, B_k$.

In general, whenever a node needs to upload an encoded block to a neighboring node, the block is formed

9) http://en.wikipedia.org/wiki/Avalanche_(P2P)

10) In the system proposed in Ref. [9], the size of the base field is of the order 2^{16}.

by taking a random linear combination of all the blocks possessed by that node. Continuing with the above example, when client A needs to upload an encoded block E_3 to a neighboring client B, we have

$$E_3 = c_1^3 E_1 + c_2^3 E_2, \tag{98}$$

where c_1^3 and c_2^3 are randomly chosen from F. Substituting (97) into (98), we obtain

$$E_3 = \sum_{j=1}^{k} \left(c_1^3 c_j^1 + c_2^3 c_j^2\right) B_j. \tag{99}$$

Thus E_3 and in general every encoded block subsequently uploaded by a node in the network is some random linear combination of the data blocks $B_1, B_2, \ldots, B_k$.

The exact strategy for downloading encoded blocks from the neighboring nodes so as to avoid receiving redundant information depends on the implementation. The main idea is that downloading from a neighboring node is necessary only if the neighboring node has at least one block not in the linear span of all the blocks possessed by that particular node. Upon receiving enough linearly independent encoded blocks, a node is able to decode the whole file.

Compared with store-and-forward, the application of network coding as described in the above system can reduce the file download time because an encoded block uploaded by a node contains information about every block possessed by that node. Moreover, in case some nodes leave the system before the end of the distribution process, it is more likely that the remaining nodes have the necessary information to recover the whole file if network coding is used. In the following, we will give a quantitative analysis to substantiate these claimed advantages of network coding.

3.5.2 Model and analysis

Let V be the set of all the nodes in the system. In implementation, blocks of data are transmitted between neighboring nodes in an asynchronous manner, and possibly at different speeds. To simplify the analysis, we assume that every transmission from one node to a neighboring node is completed in an integral number of time units. Then we can unfold the network of nodes in discrete time into a graph $G^* = (V^*, E^*)$ with the node set

$$V^* = \{i_t : i \in V \text{ and } t \geqslant 0\}, \tag{100}$$

where node $i_t \in V^*$ corresponds to node $i \in V$ at time t. The edge set E^* specified below is determined by the strategy adopted for the server as well as for all the other nodes in V to request uploading of data blocks from the neighboring nodes. Specifically, there are two types of edges in E^*:

1) There is an edge with capacity m from node i_t to node $j_{t'}$, where $t < t'$, if m blocks are transmitted from node i to node j, starting at time t and ending at time t'.
2) For each $i \in V$ and $t \geqslant 0$, there is an edge with infinite capacity from node i_t to node i_{t+1}.

An edge of the second type models the assumption that the blocks, once possessed by a node, are retained in that node indefinitely over time. Without loss of generality, we may assume that all the blocks possessed by nodes $i_l, l \leqslant t$ are transmitted uncoded on the edge from node i_t to node i_{t+1}.

An illustration of the graph G^* up to $t = 3$ with V consisting of the server S and three clients A, B, and C is given in Fig. 10, where the edges with infinite capacities are lightened for clarity. Note that the graph G^* is acyclic because each edge is pointed in the positive time direction and hence a cycle cannot be formed.

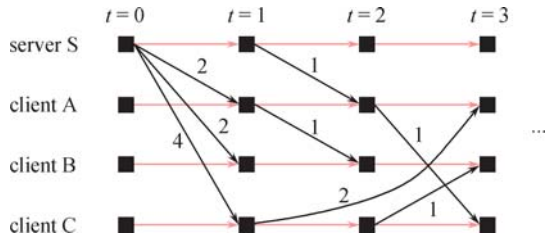


Fig. 10 An illustration of graph G^*

Denote the server S by node $s \in V$ and regard node s_0 in G^* as the source node generating the whole file consisting of k data blocks and multicasting it to all the other nodes in G^* via random linear network coding, with the coefficients in the random linear combinations forming the encoded blocks being the local encoding kernels of the network code. Note that random network coding is applied on G^*, not the logical network formed by the user nodes.

Also note that in order to simplify our description of the system, we have omitted the necessity of delivering the global encoding kernels to the nodes for the purpose of decoding. We refer the reader to the discussion toward the end of Section 3.3 for this implementation detail.

We are now ready to determine the time it takes for a particular node $i \in V$ to receive the whole file. Denote the value of a max-flow from node s_0 to a node $v \in G^*$ other than s_0 by maxflow(v). When the base field is sufficiently large, by Corollary 5, with probability close to 1, the network code generated randomly during the process is a linear multicast, so that those nodes i_t with

$$\text{maxflow}(i_t) \geqslant k \tag{101}$$

can receive the whole file. In other words, with high probability, the time it takes a node $i \in V$ to receive the whole file is equal to t^*, the minimum t that satisfies (101). Obviously, this is a lower bound on the time it takes a node $i \in V$ to receive the whole file,

and it is achievable with high probability by the system under investigation. In the rare event that node i cannot decode at time t^*, it can eventually decode upon downloading some additional encoded blocks from the neighboring nodes.

When some nodes leave the system before the end of the distribution process, an important question is whether the remaining nodes have the necessary information to recover the whole file. To be specific, assume that a subset of users $U^c \subset V$ leave the system after time t, and we want to know whether the users in $U = V \backslash U^c$ have sufficient information to recover the whole file. If they do, by further exchanging information among themselves, every user in U can eventually receive the whole file (provided that no more nodes leave the system). Toward this end, again consider the graph G^*. Let

$$U_t = \{u_t : u \in U\} \tag{102}$$

and denote the value of a max-flow from node s_0 to the set of nodes U_t by maxflow(U_t). If

$$\text{maxflow}(U_t) \geqslant k, \tag{103}$$

then the users in U with high probability would have the necessary information to recover the whole file. This is almost the best possible performance one can expect from such a system, because if

$$\text{maxflow}(U_t) < k, \tag{104}$$

it is simply impossible for the users in U to recover the whole file even if they are allowed to exchange information among themselves.

Thus we see that random network coding provides the system with both maximum bandwidth efficiency and maximum robustness. However, additional computational resource is required compared with store-and-forward. These are engineering tradeoffs in the design of such systems.

We conclude this section by an example further demonstrating the advantage of random network coding when it is applied to packet networks with packet loss.

Example 13 The random network coding scheme discussed in this section can be applied to packet networks with packet loss. Consider the network depicted in Fig. 11 consisting of three nodes, s, t, and u. Data packets are sent from node s to node u via node t. Let the packet loss rates of channels (s,t) and (t,u) be γ, i.e., a fraction γ of packets are lost during their transmission through the channel. Then the fraction of packets sent by node s that are eventually received at node u is $(1-\gamma)^2$.

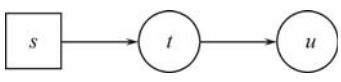


Fig. 11 A simple packet network

To fix idea, assume the packet size is sufficiently large and one packet is sent on each channel per unit time. To remedy the problem of packet loss, a fountain code [49] can be employed at node s. This would allow data packets to be sent from node s to node u reliably at an effective rate equal to $(1-\gamma)^2$. In such a scheme, node t simply forwards to node u the packets it receives from node s. On the other hand, by using the random network coding scheme we have discussed, data packets can be sent from node s to node u reliably at an effective rate equal to $1-\gamma$, which is strictly higher than $(1-\gamma)^2$ whenever $\gamma > 0$. This can be proved by means of the analysis presented in this section. The details are left as an exercise.

While a fountain code can remedy the problem of packet loss between the source node and the sink node, it cannot prevent the packet loss rate from accumulating when packets are routed through the network. On the other hand, the use of random network coding allows information to be transmitted from the source node to the sink node at the maximum possible rate, namely the min-cut between the source node and the sink node after the packet loss in the channels has been taken into account.

4 Linear network coding for cyclic networks

A directed network is *cyclic* if it contains at least one directed cycle. In Section 3, we have discussed network coding over an acyclic network, for which there exists an upstream-to-downstream order on the nodes. Following such an order, whenever a node encodes, all the information needed would have already been received on the input channels of that node. For a cyclic network, such an order of the nodes does not exists. This makes network coding over a cyclic network substantially different from network coding over an acyclic network.

4.1 Delay-free cyclic networks

When we discussed network coding over an acyclic network in Section 3, we assumed that there is no propagation delay in the network. Based on this assumption, a linear network code can be specified by either the local description in Definition 3 or the global description in Definition 4. The local and global descriptions of a linear network code are equivalent over an acyclic network because given the local encoding kernels, the global encoding kernels can be calculated recursively in any upstream-to-downstream order. In other words, (16) has a unique solution for the global encoding kernels in terms of the local encoding kernels, while (17) serves as the boundary conditions.

If these descriptions are applied to a cyclic network,

it is not clear whether for any given set of local encoding kernels, there exists a unique solution for the global encoding kernels. In the following, we give one example with a unique solution, one with no solution, and one with multiple solutions.

Example 14 Consider the cyclic network in Fig. 12. Let (s,t) precede (v,t) in the ordering among the channels. Similarly, let (s,t') precede (v,t'). Given the local encoding kernels

$$K_s = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, K_t = K_{t'} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \tag{105}$$

$$K_u = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, K_v = [1 \quad 1], \tag{106}$$

(16) yields the following unique solution for the global encoding kernels:

$$\mathbf{f}_{(s,t)} = \mathbf{f}_{(t,u)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \ \mathbf{f}_{(s,t')} = \mathbf{f}_{(t',u)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \tag{107}$$

$$\mathbf{f}_{(u,v)} = \mathbf{f}_{(v,t)} = \mathbf{f}_{(v,t')} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}. \tag{108}$$

These global encoding kernels are shown in Fig. 12, and they in fact define a 2-dimensional linear broadcast regardless of the choice of the base field. Since

$$k_{(v,t),(t,u)} = 0 \tag{109}$$

and

$$k_{(v,t'),(t',u)} = 0, \tag{110}$$

information looping in the directed cycles (t,u), (u,v), (v,t) and (t',u), (u,v), (v,t') is prevented.

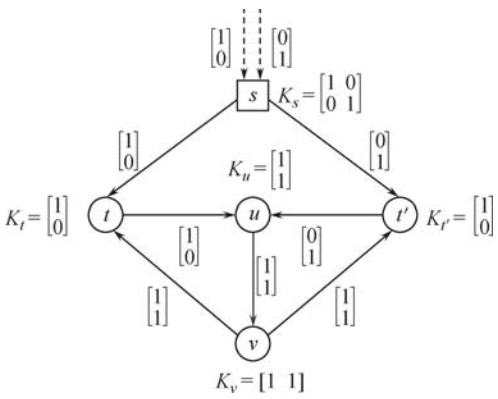


Fig. 12 A 2-dimensional linear broadcast on a cyclic network

Example 15 An arbitrarily prescribed set of local encoding kernels on a cyclic network is unlikely to be compatible with any global encoding kernels. In Fig. 13(a), a local encoding kernel is prescribed at each node in a cyclic network. Had a global encoding kernel $\mathbf{f}_e$ existed for each channel e, the requirement (16) would imply the equations

$$\mathbf{f}_{(x,y)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \mathbf{f}_{(w,x)}, \tag{111}$$

$$\mathbf{f}_{(y,w)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} + \mathbf{f}_{(x,y)}, \tag{112}$$

$$\mathbf{f}_{(w,x)} = \mathbf{f}_{(y,w)}, \tag{113}$$

which sum up to

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \tag{114}$$

a contradiction.

The nonexistence of compatible global encoding kernels can also be interpreted in terms of the message transmission. Let the message $\mathbf{x} = [a \quad b]$ be a generic vector in F^2, where F denotes the base field. The symbol transmitted on channel e, given by $\mathbf{x} \cdot \mathbf{f}_e$, are shown in Fig. 13(b). In particular, the symbols transmitted on channels (x,y), (y,w), and (w,x), namely p, q, and r, are related through

$$p = a + r, \tag{115}$$

$$q = b + p, \tag{116}$$

$$r = q. \tag{117}$$

These equalities imply that

$$a + b = 0, \tag{118}$$

a contradiction to the independence between the two components a and b of a generic message.

Example 16 Let F be an extension field of $GF(2)$[11]. Consider the same prescription of the local encoding kernels at the nodes as in Example 15 except that

$$K_s = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}. \tag{119}$$

The following three sets of global encoding kernels meet the requirement (16) in the definition of a linear network code:

$$\mathbf{f}_{(s,x)} = \mathbf{f}_{(s,y)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \ \mathbf{f}_{(x,y)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$
$$\mathbf{f}_{(y,w)} = \mathbf{f}_{(w,x)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}; \tag{120}$$

$$\mathbf{f}_{(s,x)} = \mathbf{f}_{(s,y)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \ \mathbf{f}_{(x,y)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix},$$
$$\mathbf{f}_{(y,w)} = \mathbf{f}_{(w,x)} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}; \tag{121}$$

11) In an extension field of $GF(2)$, the arithmetic on the symbols 0 and 1 are modulo 2 arithmetic.

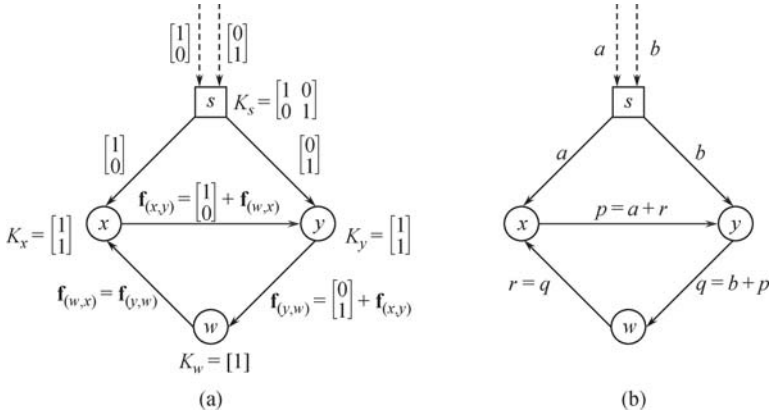


Fig. 13 An example of a cyclic network and local encoding kernels that do not render a solution for global encoding kernels

$$\mathbf{f}_{(s,x)}=\mathbf{f}_{(s,y)}=\begin{bmatrix}1\\0\end{bmatrix},\ \mathbf{f}_{(x,y)}=\begin{bmatrix}0\\1\end{bmatrix},$$
$$\mathbf{f}_{(y,w)}=\mathbf{f}_{(w,x)}=\begin{bmatrix}1\\1\end{bmatrix}. \tag{122}$$

4.2 Convolutional network codes

In a real network, the propagation delay, which includes the processing delay at the nodes and the transmission delay over the channels, cannot be zero. For a cyclic network, this renders the implementation non-physical because the transmission on an output channel of a node can only depend on the information received on the input channels of that node. Besides, technical difficulties as described in the last section arise even with the ideal assumption that there is no propagation delay.

In this section, we introduce the *unit-delay network* as a model for network coding on a cyclic network $G=(V,E)$, where V and E are the sets of nodes and channels of the network, respectively. In this model, a symbol is transmitted on every channel in the network at every discrete time index, with the transmission delay equal to exactly one time unit. Intuitively, this assumption on the transmission delay over a channel ensures no information looping in the network even in the presence of a directed cycle. The results to be developed in this section, although discussed in the context of cyclic networks, apply equally well to acyclic networks.

As a time-multiplexed network in the combined space-time domain, a unit-delay network can be unfolded with respect to the time dimension into an indefinitely long network called a *trellis network*. Corresponding to a physical node t is a sequence of nodes $t_0,t_1,t_2,\ldots$ in the trellis network, with the subscripts being the time indices. A channel e_j in the trellis network represents the transmission on the physical channel e between times j and $j+1$. When the physical channel e is from node t to node u, the channel e_j in the trellis network is from node t_j to node u_{j+1}. Note that the trellis network is acyclic regardless of the topology of the physical network, because all the channels are pointing in the forward time direction so that a directed cycle cannot be formed.

Example 17 Regard the network in Fig. 13 as a unit-delay network. For each channel e in the network, the scalar values in the base field F transmitted on the channels e_j, $j\geqslant 0$ in the corresponding trellis network are determined by the local encoding kernels. This is illustrated in Fig. 14. For instance, the channels $(x,y)_j$, $j\geqslant 0$ carry the scalar values

$$0,\ 0,\ a_0,\ a_1,\ a_2+b_0,\ a_0+a_3+b_1,\ a_1+a_4+b_2,\ \ldots, \tag{123}$$

respectively. This constitutes an example of a convolutional network code to be formally defined in Definition 6.

Let c_j be the scalar value in F transmitted on a particular channel in the network at time j. A succinct mathematical expression for the sequence of scalars $c_0,c_1,c_2,\ldots$ is the z-transform

$$\sum_{j=0}^{\infty}c_jz^j=c_0+c_1z+c_2z^2+\cdots, \tag{124}$$

where the power j of the *dummy variable* z represents discrete time. The pipelining of scalars transmitted over a time-multiplexed channel can thus be regarded as the transmission of a power series over the channel. For example, the transmission of a scalar value on the channel $(x,y)_j$ for each $j\geqslant 0$ in the trellis network in Fig. 14 translates into the transmission of the power series

$$a_0z^2+a_1z^3+(a_2+b_0)z^4+(a_0+a_3+b_1)z^5 + (a_1+a_4+b_2)z^6+\cdots$$

over the channel (x,y) in the network in Fig. 13.

The z-transform in (124) is a power series in the dummy variable z, which would be regarded as either a real number or a complex number in the context of signal analysis. However, in the context of convolutional

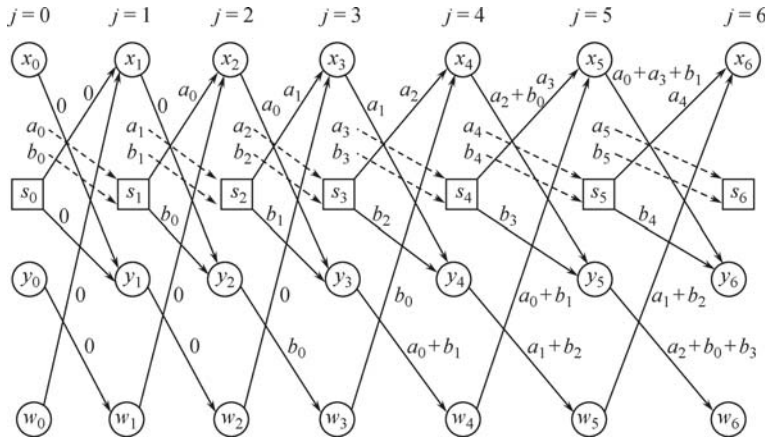


Fig. 14 Trellis network depicting a convolutional network code defined on physical network in Fig. 13

coding, such a power series should not be regarded as anything more than a representation of the sequence of scalars $c_0, c_1, c_2, \ldots$. Specifically, the dummy variable z is not associated with any value, and there is no notion of convergence. Such power series are called *formal power series*.

Given a field F, consider rational functions of a dummy variable z of the form

$$\frac{p(z)}{1+zq(z)}, \tag{125}$$

where $p(z)$ and $q(z)$ are polynomials. The following properties of such a function are relevant to our subsequent discussion:

1) The denominator has a constant term, so the function can be expanded into a power series by long division (see Example 18).
2) If $p(z)$ is not the zero polynomial, the inverse function, namely,
$$\frac{1+zq(z)}{p(z)}, \tag{126}$$
exists.

Note that the rational function in (126) does not represent a power series if $p(z)$ contains the factor z, or equivalently, does not contain a constant term.

The ring of power series over F is conventionally denoted by $F[[z]]$. Rational functions of the form (125) will be called *rational power series* which constitute a ring denoted by $F\langle z\rangle$ [50]. It follows directly from the definitions that $F\langle z\rangle$ is a subring of $F[[z]]$. We refer the reader to Ref. [51] for a comprehensive treatment of abstract algebra.

In the following, we illustrate the concepts of rational power series through a few simple examples.

Example 18 If z is a complex number, then we can write

$$\frac{1}{1-z} = 1 + z + z^2 + z^3 + \cdots \tag{127}$$

provided that $|z| < 1$, where we have interpreted the coefficients in the power series on the right-hand side as real (or complex) numbers. If $|z| > 1$, the above expression is not meaningful because the power series diverges.

However, if we do not associate z with a value but regard the coefficients in the power series as elements in a commutative ring, we can always write

$$\begin{aligned}(1-z)&(1+z+z^2+z^3+\cdots)\\ &= (1+z+z^2+z^3+\cdots) - (z+z^2+z^3+\cdots) \quad (128)\\ &= 1. \quad (129)\end{aligned}$$

In this sense, we say that $1-z$ is the reciprocal of the power series $1+z+z^2+z^3+\cdots$ and write

$$\frac{1}{1-z} = 1 + z + z^2 + z^3 + \cdots. \tag{130}$$

We also say that $1+z+z^2+z^3+\cdots$ is the power series expansion of $1/(1-z)$. In fact, the power series on the right-hand side can be readily obtained by dividing 1 by $1-z$ using long division.

Alternatively, we can seek the inverse of $1-z$ by considering the identity

$$(1-z)(a_0 + a_1 z + a_2 z^2 + \cdots) = 1. \tag{131}$$

By equating the powers of z on both sides, we have

$$\begin{aligned} a_0 &= 1, && (132)\\ -a_0 + a_1 &= 0, && (133)\\ -a_1 + a_2 &= 0, && (134)\\ &\vdots && (135)\end{aligned}$$

Then by forward substitution, we immediately obtain

$$1 = a_0 = a_1 = a_2 = \cdots, \tag{136}$$

which gives exactly the power series obtained by long division. The reader can easily verify that long division indeed mimics the process of forward substitution.

For polynomials $p(z)$ and $q(z)$ where $q(z)$ is not the zero polynomial, we can always expand the rational function $p(z)/q(z)$ into a series. However, such a series is not

always a power series. For example,

$$\frac{1}{z-z^2} = \frac{1}{z}\left(\frac{1}{1-z}\right) \tag{137}$$

$$= \frac{1}{z}(1+z+z^2+\cdots) \tag{138}$$

$$= z^{-1}+1+z+z^2+\cdots. \tag{139}$$

The above is not a power series because of the term involving a negative power of z. In fact, the identity

$$(z-z^2)(a_0+a_1z+a_2z^2+\cdots)=1 \tag{140}$$

has no solution for $a_0, a_1, a_2, \ldots$ since there is no constant term on the left-hand side. Therefore, $1/(z-z^2)$ indeed does not have a power series expansion.

From the above example, we see that $p(z)/q(z)$ represents a rational power series if and only if $q(z)$ has a nonzero constant term, or equivalently, does not contain the factor z.

Definition 6 (Convolutional network code) An ω-dimensional convolutional network code on a unit-delay network over a base field F consists of an element $k_{d,e}(z) \in F\langle z\rangle$ for every adjacent pair of channels (d,e) in the network as well as a column ω-vector $\mathbf{f}_e(z)$ over $F\langle z\rangle$ for every channel e such that:

$$\mathbf{f}_e(z) = z\sum_{d\in \mathrm{In}(t)} k_{d,e}(z)\mathbf{f}_d(z) \text{ for } e \in \mathrm{Out}(t). \tag{141}$$

The vectors $\mathbf{f}_e(z)$ for the imaginary channels $e \in \mathrm{In}(s)$ consist of scalar components that form the standard basis of the vector space F^ω. (142)

The vector $\mathbf{f}_e(z)$ is called the *global encoding kernel* for channel e, and $k_{d,e}(z)$ is called the local encoding kernel for the adjacent pair of channels (d,e). The $|\mathrm{In}(t)| \times |\mathrm{Out}(t)|$ matrix

$$K_t(z) = [k_{d,e}(z)]_{d\in\mathrm{In}(t),e\in\mathrm{Out}(t)} \tag{143}$$

is called the local encoding kernel at node t.

The constraint (141) is the time-multiplexed version of (16), with the factor z in the equation indicating a unit-time delay that represents the transmission delay over a channel. In the language of electronic circuit theory, for an adjacent pair of channels (d,e), the "gain" from channel d to channel e is given by $zk_{d,e}(z)$.

A convolutional network code over a unit-delay network can be viewed as a discrete-time *linear time-invariant* (LTI) system defined by the local encoding kernels, where the local encoding kernel $k_{d,e}(z)$ specifies the impulse response of an LTI filter from channel d to channel e. The requirement that $k_{d,e}(z)$ is a power series corresponds to the causality of the filter. The additional requirement that $k_{d,e}(z)$ is rational ensures that the filter is implementable by a finite circuitry of shift registers. Intuitively, once the local encoding kernels are given, the global encoding kernels are uniquely determined. This is explained as follows. Write

$$\mathbf{f}_e(z) = \sum_{j=0}^{\infty} \mathbf{f}_{e,j} z^j = \mathbf{f}_{e,0} + \mathbf{f}_{e,1}z + \mathbf{f}_{e,2}z^2 + \cdots \tag{144}$$

and

$$k_{d,e}(z) = \sum_{j=0}^{\infty} k_{d,e,j} z^j = k_{d,e,0} + k_{d,e,1}z + k_{d,e,2}z^2 + \cdots, \tag{145}$$

where $\mathbf{f}_{e,j}$ is a column ω-vector in F^ω and $k_{d,e,j}$ is a scalar in F. Then the equation in (141) can be written in time domain as the convolutional equation

$$\mathbf{f}_{e,j} = \sum_{d\in\mathrm{In}(t)} \left(\sum_{u=0}^{j-1} k_{d,e,u}\,\mathbf{f}_{d,j-1-u}\right) \tag{146}$$

for $j \geqslant 0$, with the boundary conditions provided by (142):

- The vectors $\mathbf{f}_{e,0}, e \in \mathrm{In}(t)$ form the standard basis of the vector space F^ω.
- The vectors $\mathbf{f}_{e,j}, e \in \mathrm{In}(t)$ are the zero vector for all $j \geqslant 1$.

For $j = 0$, the summation in (146) is empty, so that $\mathbf{f}_{e,0}$ vanishes. For $j \geqslant 0$, the right-hand side of (146) involves the vectors $\mathbf{f}_{d,i}$ only for $0 \leqslant i \leqslant j-1$. Thus the vectors $\mathbf{f}_{e,j}$, $j \geqslant 1$ can be calculated recursively via (146) with the boundary condition

$$\mathbf{f}_{d,0} = 0 \quad \text{for all } d \in E. \tag{147}$$

Together with $\mathbf{f}_{e,0} = 0$, the global encoding kernel $\mathbf{f}_e(z)$ is determined (cf. (144)). In other words, in a convolutional network code over a unit-delay network, the global encoding kernels are determined once the local encoding kernels are given. From (144), we see that the components of $\mathbf{f}_e(z)$ are power series in z, so $\mathbf{f}_e(z)$ is a column ω-vector over $F[[z]]$. In Theorem 3, we will further establish that the components of the global encoding kernels are in fact rational functions in z, proving that $\mathbf{f}_e(z)$ is indeed a column ω-vector over $f\langle z\rangle$ as required in Definition 6 for a convolutional network code.

Example 19 In Fig. 13, denote the two imaginary channels by (o,s) and $(o,s)'$. A convolutional network code is specified by the prescription of a local encoding kernel at every node as shown in the figure:

$$K_s(z) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},\ K_x(z) = K_y(z) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \tag{148}$$

$$K_w(z) = [\,1\,], \tag{149}$$

and a global encoding kernel for every channel:

$$\mathbf{f}_{(o,s)}(z) = \begin{bmatrix} 1 \\ 0 \end{bmatrix},\ \mathbf{f}_{(o,s)'}(z) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \tag{150}$$

$$\mathbf{f}_{(s,x)}(z) = z \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} z \\ 0 \end{bmatrix}, \tag{151}$$

$$\mathbf{f}_{(s,y)}(z) = z \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ z \end{bmatrix}, \tag{152}$$

$$\mathbf{f}_{(x,y)}(z) = \begin{bmatrix} z^2/(1-z^3) \\ z^4/(1-z^3) \end{bmatrix}, \tag{153}$$

$$\mathbf{f}_{(y,w)}(z) = \begin{bmatrix} z^3/(1-z^3) \\ z^2/(1-z^3) \end{bmatrix}, \tag{154}$$

$$\mathbf{f}_{(w,x)}(z) = \begin{bmatrix} z^4/(1-z^3) \\ z^3/(1-z^3) \end{bmatrix}, \tag{155}$$

where the last three global encoding kernels have been solved from the following equations:

$$\mathbf{f}_{(x,y)}(z) = z \begin{bmatrix} \mathbf{f}_{(s,x)}(z) & \mathbf{f}_{(w,x)}(z) \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{156}$$

$$= z^2 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + z\,\mathbf{f}_{(w,x)}(z), \tag{157}$$

$$\mathbf{f}_{(y,w)}(z) = z \begin{bmatrix} \mathbf{f}_{(s,y)}(z) & \mathbf{f}_{(x,y)}(z) \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{158}$$

$$= z^2 \begin{bmatrix} 0 \\ 1 \end{bmatrix} + z\,\mathbf{f}_{(x,y)}(z), \tag{159}$$

$$\mathbf{f}_{(w,x)}(z) = z \left(\mathbf{f}_{(y,w)}(z)\right) \begin{bmatrix} 1 \end{bmatrix} = z\,\mathbf{f}_{(y,w)}(z). \tag{160}$$

These local and global encoding kernels of a 2-dimensional convolutional network code are summarized in Fig. 15.

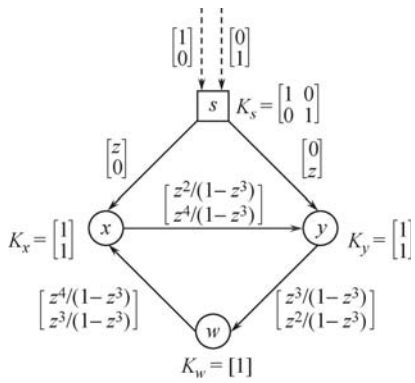


Fig. 15 Local and global encoding kernels of convolutional network code in Example 19

Represent the message generated at source node s at time j, where $j \geqslant 0$, by a row ω-vector $\mathbf{x}_j \in F^\omega$. Equivalently, source node s generates the message pipeline represented by the z-transform

$$\mathbf{x}(z) = \sum_{j=0}^{\infty} \mathbf{x}_j z^j, \tag{161}$$

which is a row ω-vector over $F[[z]]$, the ring of power series over F. Here, $\mathbf{x}(z)$ is not necessarily rational.

Through a convolutional network code, each channel e carries the power series $\mathbf{x}(z)\,\mathbf{f}_e(z)$. Write

$$\mathbf{x}(z)\,\mathbf{f}_e(z) = \sum_{j=0}^{\infty} m_{e,j} z^j, \tag{162}$$

where

$$m_{e,j} = \sum_{u=0}^{j} \mathbf{x}_u\,\mathbf{f}_{e,j-u}. \tag{163}$$

For $e \in \mathrm{Out}(t)$, from the equation in (141), we obtain

$$\mathbf{x}(z)\,\mathbf{f}_e(z) = \mathbf{x}(z) \left[z \sum_{d \in \mathrm{In}(t)} k_{d,e}(z)\,\mathbf{f}_d(z) \right] \tag{164}$$

$$= z \sum_{d \in \mathrm{In}(t)} k_{d,e}(z) \left[\, \mathbf{x}(z)\,\mathbf{f}_d(z) \,\right], \tag{165}$$

or equivalently in time domain,

$$m_{e,j} = \sum_{d \in \mathrm{In}(t)} \left(\sum_{u=0}^{j-1} k_{d,e,u}\, m_{d,j-1-u} \right). \tag{166}$$

The reader should compare (166) with (146). Note that the scalar values $m_{e,j}$, $j \geqslant 1$ can be calculated recursively via (166) with the boundary condition

$$m_{d,0} = 0 \quad \text{for all } d \in E. \tag{167}$$

Thus a node t calculates the scalar value $m_{e,j}$ for transmitting on each output channel e at time j from the cumulative information it has received on all the input channels up to time $j-1$. The convolutional equation (166) can be implemented by a finite circuit of shift-registers in a causal manner because the local encoding kernels belong to $F\langle z \rangle$, the ring of rational power series over F (cf. Definition 6).

Example 20 Consider the convolutional network code in Example 19. Let source node s pipelines the message

$$\mathbf{x}(z) = \begin{bmatrix} \sum_{j=0}^{\infty} a_j z^j & \sum_{j=0}^{\infty} b_j z^j \end{bmatrix}. \tag{168}$$

Then the five channels $(s,x), (s,y), (x,y), (y,w)$, and (w,x) carry the following power series, respectively:

$$\mathbf{x}(z)\,\mathbf{f}_{(s,x)}(z) = \sum_{j=0}^{\infty} a_j z^{j+1}, \tag{169}$$

$$\mathbf{x}(z)\,\mathbf{f}_{(s,y)}(z) = \sum_{j=0}^{\infty} b_j z^{j+1}, \tag{170}$$

$$\mathbf{x}(z)\,\mathbf{f}_{(x,y)}(z) = \left(\sum_{j=0}^{\infty} a_j z^{j+2} + \sum_{j=0}^{\infty} b_j z^{j+4}\right)/(1-z^3) \tag{171}$$

$$= \left(\sum_{j=0}^{\infty} a_j z^{j+2} + \sum_{j=0}^{\infty} b_j z^{j+4}\right)\sum_{j=0}^{\infty} z^{3j} \tag{172}$$

$$= a_0 z^2 + a_1 z^3 + (a_2+b_0)z^4 + (a_0+a_3+b_1)z^5 + \cdots, \tag{173}$$

$$\mathbf{x}(z)\,\mathbf{f}_{(y,w)}(z) = \left(\sum_{j=0}^{\infty} a_j z^{j+3} + \sum_{j=0}^{\infty} b_j z^{j+2}\right)/(1-z^3), \tag{174}$$

$$\mathbf{x}(z)\,\mathbf{f}_{(w,x)}(z) = \left(\sum_{j=0}^{\infty} a_j z^{j+4} + \sum_{j=0}^{\infty} b_j z^{j+3}\right)/(1-z^3). \tag{175}$$

At each time $j \geqslant 0$, the source generates a message $\mathbf{x}_j = [\,a_j \quad b_j\,]$. Thus channel (s,x) carries the scalar 0 at time 0 and the scalar a_{j-1} at time $j \geqslant 1$. Similarly, channel (s,y) carries the scalar 0 at time 0 and the scalar b_{j-1} at time $j \geqslant 1$. For every channel e, write

$$\mathbf{x}(z)\,\mathbf{f}_e(z) = \sum_{j=0}^{\infty} m_{e,j} z^j \tag{176}$$

as in (162). The actual encoding process at node x is as follows. At time j, node x has received the sequence $m_{d,0}, m_{d,1}, \ldots, m_{d,j-1}$ for $d = (s,x)$ and (w,x). Accordingly, at time $j \geqslant 1$, channel (x,y) transmits the scalar value

$$m_{(x,y),j} = \sum_{u=0}^{j-1} k_{(s,x),(x,y),u} m_{(s,x),j-1-u} + \sum_{u=0}^{j-1} k_{(w,x),(x,y),u} m_{(w,x),j-1-u} \tag{177}$$

$$= m_{(s,x),j-1} + m_{(w,x),j-1}. \tag{178}$$

Similarly, channels (y,w) and (w,x) transmit the scalar values

$$m_{(y,w),j} = m_{(s,y),j-1} + m_{(x,y),j-1} \tag{179}$$

and

$$m_{(w,x),j} = m_{(y,w),j-1}, \tag{180}$$

respectively. The values $m_{(x,y),j}$, $m_{(y,w),j}$, and $m_{(w,x),j}$ for $j \geqslant 1$ can be calculated recursively by the above formulas with the boundary condition

$$m_{e,0} = 0 \quad \text{for all } e \in E, \tag{181}$$

and they are shown in the trellis network in Fig. 14 for small values of j. For instance, the channel (x,y) carries the scalar values

$$m_{(x,y),0} = 0,\ m_{(x,y),1} = 0,\ m_{(x,y),2} = a_0,\ m_{(x,y),3} = a_1,\ m_{(x,y),4} = a_2 + b_0,\ m_{(x,y),5} = a_0 + a_3 + b_1,\ \ldots. \tag{182}$$

The z-transform of this sequence is

$$x(z)\,\mathbf{f}_{(x,y)}(z) = \left(\sum_{j=0}^{\infty} a_j z^{j+2} + \sum_{j=0}^{\infty} b_j z^{j+4}\right)/(1-z^3), \tag{183}$$

as calculated in (173).

In the discussion following Definition 6, we have shown that once the local encoding kernels of a convolutional network code over a unit-delay network are given, the global encoding kernels are determined. The proof of the next theorem further provides a simple closed-form expression for the global encoding kernels $\mathbf{f}_e(z)$, from which it follows that the entries in $\mathbf{f}_e(z)$ indeed belong to $F\langle z\rangle$ as required in Definition 6.

Theorem 3 Let F be the base field and $k_{d,e}(z) \in F\langle z\rangle$ be given for every adjacent pair of channels (d,e) on a unit-delay network. Then there exists a unique ω-dimensional convolutional network code over F with $k_{d,e}(z)$ as the local encoding kernel for every (d,e).

Proof Let the unit-delay network be represented by a directed graph $G = (V,E)$. Let $[k_{d,e}(z)]$ be the $|E|\times|E|$ matrix in which both the rows and columns are indexed by E, with the (d,e)th entry equal to the given $k_{d,e}(z)$ if (d,e) is an adjacent pair of channels, and equal to zero otherwise. Denote the global encoding kernel of channel e by $\mathbf{f}_e(z)$ if exists. Let $[\mathbf{f}_e(z)]$ be the $\omega \times |E|$ matrix obtained by putting the global encoding kernels $\mathbf{f}_e(z), e \in E$ in juxtaposition. Let $H_s(z)$ be the $\omega \times |E|$ matrix obtained by appending $|E| - |\mathrm{Out}(s)|$ columns of zeroes to the local encoding kernel $K_s(z)$. The requirements (141) and (142) in Definition 6 can be written as

$$[\mathbf{f}_e(z)] = z[\mathbf{f}_e(z)]\,[k_{d,e}(z)] + zIH_s(z), \tag{184}$$

where I in the above denotes the $\omega \times \omega$ identity matrix representing the global encoding kernels $\mathbf{f}_e(z)$, $e \in \mathrm{In}(s)$ in juxtaposition. Rearranging the terms in (184), we obtain

$$[\mathbf{f}_e(z)](I - z[k_{d,e}(z)]) = zH_s(z). \tag{185}$$

In the matrix $z[k_{d,e}(z)]$, the diagonal elements are equal to zero because (e,e) does not form an adjacent pair of channels for all $e \in E$, while the non-zero off-diagonal elements all contain the factor z. Therefore, $\det(I - z[k_{d,e}(z)])$ has the form

$$1 + zq(z), \tag{186}$$

where $q(z) \in F\langle z\rangle$, so that it is invertible inside $F\langle z\rangle$ because

$$[\det(I - z[k_{d,e}(z)])]^{-1} = \frac{1}{1 + zq(z)} \tag{187}$$

is a rational power series. It follows that

$$(I - z[k_{d,e}(z)])^{-1} \tag{188}$$

exists and is a matrix over $F\langle z\rangle$. Then the unique solution for $[\mathbf{f}_e(z)]$ in (185) is given by

$$[\mathbf{f}_e(z)] = zH_s(z)(I - z[k_{d,e}(z)])^{-1}. \tag{189}$$

With the two matrices $[k_{d,e}(z)]$ and $H_s(z)$ representing the given local encoding kernels and the matrix $[\mathbf{f}_e(z)]$ representing the global encoding kernels, (189) is a closed-form expression for the global encoding kernels in terms of the local encoding kernels. In particular, $[\mathbf{f}_e(z)]$ is a matrix over $F\langle z\rangle$ because all the matrices on the right-hand side of (189) are over $F\langle z\rangle$. Thus we conclude that all the components of the global encoding kernels are in $F\langle z\rangle$. Hence, the given local encoding kernels $k_{d,e}(z)$ for all adjacent pairs (d, e) together with the associated global encoding kernels $\mathbf{f}_e(z), e \in \text{In}(s) \cup E$ constitute a unique convolutional network code over the unit-delay network G.

In view of Definition 4 for the global description of a linear network code over an acyclic network, Definition 6 can be regarded as the global description of a convolutional network code over a unit-delay network, while Theorem 3 renders a local description by specifying the local encoding kernels only.

4.3 Decoding of convolutional network codes

For a node t, let

$$F_t(z) = [\mathbf{f}_e(z)]_{e\in\text{In}(t)} \tag{190}$$

be the $\omega \times |\text{In}(t)|$ matrix obtained by putting the global encoding kernels $\mathbf{f}_e(z)$, $e \in \text{In}(t)$ in juxtaposition. In the following, we define a *convolutional multicast*, the counterpart of a linear multicast defined in Section 3, for a unit-delay cyclic network. The existence of a convolutional multicast will also be established.

Definition 7 (Convolutional multicast) An ω-dimensional convolutional network code on a unit-delay network qualifies as an ω-dimensional convolutional multicast if for every non-source node t with $\text{maxflow}(t) \geqslant \omega$, there exists an $|\text{In}(t)| \times \omega$ matrix $D_t(z)$ over $F\langle z\rangle$ and a positive integer τ such that

$$F_t(z)\, D_t(z) = z^\tau I, \tag{191}$$

where $\tau > 0$ depends on node t and I is the $\omega\times\omega$ identity matrix. The matrix $D_t(z)$ and the integer τ are called the decoding kernel and the decoding delay at node t, respectively.

Source node s generates the message pipeline

$$\mathbf{x}(z) = \sum_{j=0}^{\infty} \mathbf{x}_j z^j, \tag{192}$$

where $\mathbf{x}_j$ is a row ω-vector in F^ω and $\mathbf{x}(z)$ is a row ω-vector over $F[[z]]$. Through the convolutional network code, a channel e carries the power series $\mathbf{x}(z)\,\mathbf{f}_e(z)$. The power series $\mathbf{x}(z)\,\mathbf{f}_e(z)$ received by a node t from the input channels $e \in \text{In}(t)$ form the row $|\text{In}(t)|$-vector $\mathbf{x}(z)\,F_t(z)$ over $F[[z]]$. If the convolutional network code is a convolutional multicast, node t can use the decoding kernel $D_t(z)$ to calculate

$$\begin{aligned}(\mathbf{x}(z)\,F_t(z))D_t(z) &= \mathbf{x}(z)(F_t(z)D_t(z)) &(193)\\ &= \mathbf{x}(z)(z^\tau I) &(194)\\ &= z^\tau \mathbf{x}(z). &(195)\end{aligned}$$

The row ω-vector $z^\tau\mathbf{x}(z)$ of power series represents the message pipeline generated by source node s delayed by τ time units. Note that $\tau > 0$ because the message pipeline $\mathbf{x}(z)$ is delayed by one time unit at node s.

Example 21 Consider the network in Fig. 15. Again let source node s pipelines the message

$$\mathbf{x}(z) = \begin{bmatrix} \sum_{j=0}^{\infty} a_j z^j & \sum_{j=0}^{\infty} b_j z^j \end{bmatrix}. \tag{196}$$

For node x, we have

$$F_x(z) = \begin{bmatrix} z & z^4/(1-z^3) \\ 0 & z^3/(1-z^3) \end{bmatrix}. \tag{197}$$

Let

$$D_x(z) = \begin{bmatrix} z^2 & -z^3 \\ 0 & 1-z^3 \end{bmatrix}. \tag{198}$$

Then

$$F_x(z)D_t(z) = z^3 I_2 \tag{199}$$

(I_2 is the 2×2 identity matrix). From channels (s, x) and (w, x), node x receives the row vector

$$\mathbf{x}(z)F_x(z) = \begin{bmatrix} \sum_{j=0}^{\infty} a_j z^{j+1} & \sum_{j=0}^{\infty} \dfrac{a_j z^{j+4} + b_j z^{j+3}}{1-z^3} \end{bmatrix} \tag{200}$$

and decodes the message pipeline as

$$z^3\,\mathbf{x}(z) = \begin{bmatrix} \sum_{j=0}^{\infty} a_j z^{j+1} & \sum_{j=0}^{\infty} \dfrac{a_j z^{j+4} + b_j z^{j+3}}{1-z^3} \end{bmatrix} \cdot \begin{bmatrix} z^2 & -z^3 \\ 0 & 1-z^3 \end{bmatrix}. \tag{201}$$

Decoding at node y is similar. Thus the 2-dimensional convolutional network code is a convolutional multicast.

Toward proving the existence of a convolutional multicast, we first observe that Lemma 1 can be strengthened as follows with essentially no change in the proof.

Lemma 2 Let $g(y_1, y_2, \ldots, y_m)$ be a nonzero polynomial with coefficients in a field $\tilde{F}$. For any subset $\tilde{E}$ of $\tilde{F}$, if $|\tilde{E}|$ is greater than the degree of g in every y_j, then there exist $a_1, a_2, \ldots, a_m \in \tilde{E}$ such that

$$g(a_1, a_2, \ldots, a_m) \neq 0. \tag{202}$$

In the above lemma, the values $a_1, a_2, \ldots, a_m$ can be found by exhaustive search in $\tilde{E}$ provided that $\tilde{E}$ is finite. If $\tilde{E}$ is infinite, simply replace $\tilde{E}$ by a sufficiently large finite subset of $\tilde{E}$.

Theorem 4 There exists an ω-dimensional convolutional multicast over any base field F. Furthermore, the local encoding kernels of the convolutional multicast can be chosen in any sufficiently large subset Φ of $F\langle z\rangle$.

Proof Recall (189) in the proof of Theorem 3:

$$[\mathbf{f}_e(z)] = zH_s(z)(I - z[k_{d,e}(z)])^{-1}. \tag{203}$$

In this equation, the $\omega \times |E|$ matrix $[\mathbf{f}_e(z)]$ on the left-hand side represents the global encoding kernels, while the $\omega \times |E|$ matrix $H_s(z)$ and the $|E| \times |E|$ matrix $[k_{d,e}(z)]$ on the right-hand side represent the local encoding kernels. Analogous to the proof of Theorem 2, denote by $(F\langle z\rangle)[*]$ the polynomial ring over $F\langle z\rangle$ with all the $k_{d,e}(z)$ as indeterminates.

Let t be a non-source node with $\text{maxflow}(t) \geqslant \omega$. Then there exist ω edge-disjoint paths from the ω imaginary channels to ω distinct channels in $\text{In}(t)$. Put the global encoding kernels of these ω channels in juxtaposition to form the $\omega \times \omega$ matrix $L_t(z)$ over $(F\langle z\rangle)[*]$. We will show that

$$\det(L_t(z)) \neq 0 \in (F\langle z\rangle)[*]. \tag{204}$$

Toward proving (204), it suffices to show that

$$\det(L_t(z)) \neq 0 \in F\langle z\rangle \tag{205}$$

when the determinant is evaluated at some particular values for the indeterminates $k_{d,e}(z)$. Analogous to the proof of Theorem 2, we set

$$k_{d,e}(z) = 1 \tag{206}$$

for all adjacent pairs of channels (d, e) along any one of the ω edge-disjoint paths, and set

$$k_{d,e}(z) = 0 \tag{207}$$

otherwise. Then with a suitable indexing of the columns, the matrix $L_t(z)$ becomes diagonal with all the diagonal entries being powers of z. Hence, $\det(L_t(z))$ is equal to some positive power of z, proving (205) for this particular choice of the indeterminates $k_{d,e}(x)$ and hence proving (204). As the conclusion (204) applies to every non-source node t with $\text{maxflow}(t) \geqslant \omega$, it follows that

$$\prod_{t:\text{maxflow}(t)\geqslant\omega} \det(L_t(z)) \neq 0 \in (F\langle z\rangle)[*]. \tag{208}$$

Let $F(z)$ be the conventional notation for the field of rational functions in z over the given base field F. The ring $F\langle z\rangle$ of rational power series is a subset of $F(z)$. Then any subset Φ of $F\langle z\rangle$ is also a subset of $F(z)$. Note that the ring $F\langle z\rangle$ is infinite. Then for any sufficiently large subset Φ of $F\langle z\rangle$, we can apply Lemma 2 to the polynomial in (208) with $\tilde{F} = F(z)$ and $\tilde{E} = \Phi$ to see that we can choose a value $a_{d,e}(z) \in F\langle z\rangle$ for each of the indeterminates $k_{d,e}(z)$ so that

$$\prod_{t:\text{maxflow}(t)\geqslant\omega} \det(L_t(z)) \neq 0 \in F\langle z\rangle \tag{209}$$

when evaluated at $k_{d,e}(z) = a_{d,e}(z)$ for all (d, e), which in turn implies that

$$\det(L_t(z)) \neq 0 \in F\langle z\rangle \tag{210}$$

for all nodes t such that $\text{maxflow}(t) \geqslant \omega$.

Henceforth, the local encoding kernel $k_{d,e}(z)$ will be fixed at the appropriately chosen value $a_{d,e}(z)$ for all (d, e) as prescribed above. Without loss of generality, we assume that $L_t(z)$ consists of the first ω columns of $F_t(z)$. From (210), we can write

$$\det(L_t(z)) = z^\tau \left[\frac{1 + zq(z)}{p(z)}\right], \tag{211}$$

where $p(z)$ and $q(z)$ are polynomials over F and $p(z)$ is not the zero polynomial. Note that the right-hand side of (211) is the general form for a nonzero rational function in z. In this particular context, since the columns of $L_t(z)$ are global encoding kernels as prescribed by (189), each containing the factor z in the numerator, we see that $\tau > 0$.

Denote by $J_t(z)$ the adjoint matrix[12] of $L_t(z)$. Take the $\omega \times \omega$ matrix

$$\left[\frac{p(z)}{1 + zq(z)}\right] J_t(z) \tag{212}$$

and append to it $|\text{In}(t)| - \omega$ rows of zeroes to form an $|\text{In}(t)| \times \omega$ matrix $D_t(z)$. Then

$$F_t(z)D_t(z) = [\,L_t(z) \quad 0\,] \begin{bmatrix} \left[\frac{p(z)}{1+zq(z)}\right] J_t(z) \\ 0 \end{bmatrix} \tag{213}$$

12) For a matrix B whose entries are elements in a ring, denote by $\text{Adj}(B)$ the adjoint matrix of B. Then $\text{Adj}(B)B = B\text{Adj}(B) = \det(A)I$.

$$= \left[\frac{p(z)}{1+zq(z)}\right] L_t(z)J_t(z) \quad (214)$$

$$= \left[\frac{p(z)}{1+zq(z)}\right] \det(L_t(z))I \quad (215)$$

$$= z^{\tau} I, \quad (216)$$

where the last equality follows from (211). Hence, the matrix $D_t(z)$ qualifies as a decoding kernel at node t in Definition 7. This proves the existence of the convolutional multicast as required.

The asymptotic achievability of the max-flow bound for unit-delay cyclic networks was proved by Ahlswede et al. [2], where an example of a convolutional network code achieving this bound was given. Li et al. [3] conjectured the existence of convolutional multicasts on such networks. This conjecture was subsequently proved by Koetter and Médard [43]. Construction and decoding of convolutional multicast have been studied by Erez and Feder [52,53], Fragouli and Soljanin [54], and Barbero and Ytrehus [55]. The unifying treatment of convolutional codes here is based on Li and Yeung [56] (see also Yeung et al. [50]). Li and Ho [57] recently obtained a general abstract formulation of convolutional network codes based on ring theory.

The proof of Theorem 4 constitutes an algorithm for constructing a convolutional multicast. By noting the lower bound on the size of $\tilde{E}$ in Lemma 2, a convolutional multicast can be constructed with high probability by randomly choosing the local encoding kernels in the subset Φ of $F\langle z\rangle$ provided that Φ is much larger than sufficient.

Example 22 When the base field F is sufficiently large, Theorem 4 can be applied with $\Phi = F$ so that the local encoding kernels of the convolutional multicast can be chosen to be scalars. This special case is the convolutional counterpart of Theorem 2 for the existence of a linear multicast over an acyclic network. In this case, the local encoding kernels can be found by exhaustive search over F.

More generally, by virtue of Lemma 2, the same exhaustive search applies to any large enough subset Φ of $F\langle z\rangle$. For example, F can be $GF(2)$ and Φ can be the set of all binary polynomials up to a sufficiently large degree.

5 Conclusion

In this tutorial work, we have given an introduction to network coding, with emphasis on single-source network coding. Multi-source network coding is considerably more complicated and it draws heavily on information theory concepts, in particular, constraints on the entropy function [58]. For a discussion, we refer the reader to Refs. [29] and [50]. For further reading, we also recommend the expository works of Fragouli and Soljanin [59] and Ho and Lun [60].

Network coding has developed into a field that straddles information science, mathematics, physics, and even biology. During the past few years, research papers on network coding are published at an increasing rate[13)], and new applications are identified from time to time. In the next decade, we expect the integration of a number of fields in information science, including channel coding, computer networks, wireless communications, and cryptography, into a modern theory of communications under the framework of network coding.

Acknowledgements The work of the author was partially supported by a grant from the University Grants Committee of the Hong Kong Special Administrative Region, China (Project No. AoE/E-02/08).

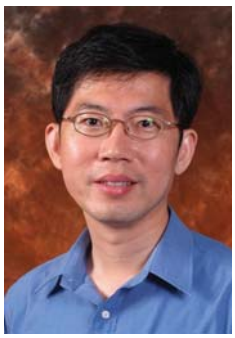
References

1. Yeung R W, Zhang Z. Distributed source coding for satellite communications. IEEE Transactions on Information Theory, 1999, 45(4): 1111–1120
2. Ahlswede R, Cai N, Li S-Y R, Yeung R W. Network information flow. IEEE Transactions on Information Theory, 2000, 46(4): 1204–1216
3. Li S-Y R, Yeung R W, Cai N. Linear network coding. IEEE Transactions on Information Theory, 2003, 49(2): 371–381
4. Yeung R W, Cai N. Network error correction, Part I: Basic concepts and upper bounds. Communications in Information and Systems, 2006, 6(1): 19–36
5. Cai N, Yeung R W. Network error correction, Part II: Lower bounds. Communications in Information and Systems, 2006, 6(1): 37–54
6. Koetter R, Kschischang F R. Coding for errors and erasures in random network coding. IEEE Transactions on Information Theory, 2008, 54(8): 3579–3591
7. Wu Y, Chou P A, Kung S-Y. Information exchange in wireless networks with network coding and physical-layer broadcast. In: Proceedings of 2005 Conference on Information Science and Systems. The Johns Hopkins University, 2005
8. Katti S, Rahul H, Hu W, Katabi D, Médard M, Crowcroft J. XORs in the air: Practical wireless network coding. IEEE/ACM Transactions on Networking, 2008, 16(3): 497–510
9. Gkantsidis C, Rodriguez P R. Network coding for large scale content distribution. In: Proceedings of IEEE INFOCOM 2005. Miami, FL, 2005
10. Rasala Lehman A. Network coding. Dissertation for the Doctoral Degree. Cambridge, MA: Massachusetts Institute of Technology, 2005
11. Cai N, Yeung R W. Secure network coding. In: Proceedings

13) As of October 2009 when this paper was finalized, the Google Scholar citation of Ref. [2] has reached 1500, with over 600 of these papers written in the preceding 12 months.

of 2002 IEEE International Symposium on Information Theory. Lausanne, 2002
12. Cai N, Yeung R W. Secure network coding. submitted to IEEE Transactions on Information Theory
13. Dougherty R, Freiling C, Zeger K. Networks, matroids, and non-Shannon information inequalities. IEEE Transactions on Information Theory, 2007, 53(6): 1949–1969
14. Wu Y, Jain K, Kung S-Y. A unification of network coding and tree-packing (routing) theorems. IEEE Transactions on Information Theory (Joint Special Issue of *IEEE Transactions on Information Theory* and *IEEE/ACM Transactions on Networking* on Networking and Information Theory), 2006, 52(6): 2398–2409
15. Mohsenian-Rad A H, Huang J, Wong V W S, Jaggi S, Schober R. A game-theoretic analysis of inter-session network coding. In: Proceedings of IEEE International Conference on Communications 2009. Dresden, 2009
16. Lun D S, Ratnakar N, Médard M, Koetter R, Karger D R, Ho T, Ahmed E, Zhao F. Minimum-cost multicast over coded packet networks. IEEE Transactions on Information Theory (Joint Special Issue of *IEEE Transactions on Information Theory* and *IEEE/ACM Transactions on Networking* on Networking and Information Theory), 2006, 52(6): 2608–2623
17. Hayashi M, Iwama K, Nishimura H, Raymond R, Yamashita S. Quantum network coding. Lecture Notes in Computer Science, 2007, 4393: 610–621
18. Liu J-Q. On information-theoretical formalization of intracellular communications based on linear network coding. In: Proceedings of SICE Annual Conference 2007. Kagawa University, 2007
19. Joint Special Issue on Networking and Information Theory. IEEE Transactions on Information Theory and IEEE/ACM Transactions on Networking, 2006, 52(6)
20. Special Issue on Cooperation in Wireless Networks. Springer — Wireless Personal Communications, 2007, 43(1)
21. Special Issue on Multiuser Cooperative Diversity for Wireless Networks. EURASIP Journal on Wireless Communications and Networking, 2006
22. Special Issue on Network Coding. Journal of Communications and Networks, 2008, 10(4)
23. Special Issue on Network Coding for Wireless Communication Networks. IEEE Journal on Selected Areas in Communications, 2009, 27(5)
24. Special Issue on Network Coding for Wireless Networks. EURASIP Journal on Wireless Communications and Networking, 2010, Apr
25. Special Issue on Physical Layer Network Coding for Wireless Cooperative Networks. EURASIP Journal on Wireless Communications and Networking, 2010, Jul
26. Effors M, Koetter R, Médard M. Breaking network logjams. Scientific American, 2007, 296(6): 78–85
27. Graham-Rowe D. Repackaging data could 'double internet speed'. New Scientist, 2008, 2 Oct, 24–25
28. Shannon C E. A Mathematical theory of communication. The Bell System Technical Journal, 1948, 27: 379–423, 623–656
29. Yeung R W. Information Theory and Network Coding. Springer, 2008
30. Bertsekas D, Gallager R. Data Networks. 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1992
31. Kurose J F, Ross K W. Computer Networking: A Top-Down Approach Featuring the Internet. 3rd ed. Addison Wesley, 2004
32. Hui J Y. Switching and Traffic Theory for Integrated Broadband Networks. Springer, 1990
33. Li S-Y R. Algebraic Switching Theory and Broadband Applications. Academic Press, 2000
34. Cheung K-M, Woo S, Stoenescu T, Chang C S, Klimesh M, Dolinar S, Cheng M K. Improved *In Situ* Communications Using Network Coding. Annual Report, JPL Task #R.07.023.014
35. Zhang S L, Liew S C, Lam P P. Physical-layer network coding. In: Proceedings of the Twelfth Annual International Conference on Mobile Computing and Networking (MobiCom 2006). Los Angeles, CA, 2006
36. Yeung R W. Multilevel diversity coding with distortion. IEEE Transactions on Information Theory, 1995, 41(2): 412–422
37. McEliece R J. Finite Fields for Computer Scientists and Engineers. Kluwer Academic Publishers, 1987
38. Lin S, Costello D J Jr. Error Control Coding: Fundamentals and Applications. Prentice-Hall, 1983 (2nd ed. 2004)
39. Blahut R E. Theory and Practice of Error Control Codes. Reading, MA: Addison-Wesley, 1983
40. Wicker S B. Error Control Systems for Digital Communication and Storage. Englewood Cliffs, NJ: Prentice-Hall, 1995
41. Fong S L, Yeung R W. Variable-rate linear network coding. In: Proceedings of 2006 IEEE Information Theory Workshop. Chengdu, 2006, 409–412
42. Toledo A L, Wang X. Efficient multipath in sensor networks using diffusion and network coding. In: Proceedings of the 40th Annual Conference on Information Sciences and Systems. Princeton, NJ: Princeton University, 2006, 87–92
43. Koetter R, Médard M. An algebraic approach to network coding. IEEE/ACM Transactions on Networking, 2003, 11(5): 782–795
44. Jaggi S, Sanders P, Chou P A, Effros M, Egner S, Jain K, Tolhuizen L. Polynomial time algorithms for multicast network code construction. IEEE Transactions on Information Theory, 2005, 51(6): 1973–1982
45. Ho T, Koetter R, Médard M, Karger D R, Effros M. The benefits of coding over routing in a randomized setting. In: Proceedings of 2003 IEEE International Symposium on Information Theory. Yokohama, 2003, 442
46. Cohen B. Incentive build robustness in BitTorrent. In: Proceedings of the 1st Workshop on Economics of Peer-to-Peer Systems. Berkeley, CA, 2003
47. Liu Z, Wu C, Li B, Zhao S. UUSee: Large-scale operational on-demand streaming with random network coding. In: Proceedings of IEEE INFOCOM 2010. San Diego, CA, 2010
48. Yeung R W. Avalanche: A network coding analysis. Communications in Information and Systems, 2007, 7(4): 353–358
49. Byers J, Luby M, Mitzenmacher M. A digital fountain approach to asynchronous reliable multicast. IEEE Journal

on Selected Areas Communications, 2002, 20(8): 1528–1540
50. Yeung R W, Li S-Y R, Cai N, Zhang Z. Network coding theory. Foundations and Trends in Communications and Information Theory, 2005, 2(4 and 5): 241–381
51. Fraleigh J B. A First Course in Abstract Algebra. 7th ed. Addison Wesley, 2003
52. Erez E, Feder M. Convolutional network codes. In: Proceedings of 2004 IEEE International Symposium on Information Theory. Chicago, IL, 2004, 146
53. Erez E, Feder M. Convolutional network codes for cyclic networks. In: Proceedings of the First Workshop on Network Coding, Theory, and Applications (NetCod 2005). Riva del Garda, 2005
54. Fragouli C, Soljanin E. Information flow decomposition for network coding. IEEE Transactions on Information Theory, 2006, 52(3): 829–848
55. Barbero Á I, Ytrehus Ø. Cycle-logical treatment for "cyclopathic" networks. IEEE Transactions on Information Theory (Joint Special Issue of *IEEE Transactions on Information Theory* and *IEEE/ACM Transactions on Networking* on Networking and Information Theory), 2006, 52(6): 2795–2804
56. Li S-Y R, Yeung R W. On convolutional network coding. In: Proceedings of 2006 IEEE International Symposium on Information Theory. Seattle, WA, 2006, 1743–1747
57. Li S-Y R, Ho S T. Ring-theoretic foundation of convolutional network coding. In: Proceedings of the Fourth Workshop on Network Coding, Theory and Applications. Hong Kong, 2008, 1–6
58. Zhang Z, Yeung R W. On characterization of entropy function via information inequalities. IEEE Transactions on Information Theory, 1998, 44(4): 1440–1452
59. Fragouli C, Soljanin E. Network coding fundamentals. Foundations and Trends in Networking, 2007, 2(1): 1–133
60. Ho T, Lun D S. Network Coding: An Introduction. Cambridge University Press, 2008



Raymond W. Yeung was born in Hong Kong on June 3, 1962. He received the B.S., M.Eng., and Ph.D. degrees in electrical engineering from Cornell University, Ithaca, NY, in 1984, 1985, and 1988, respectively.

He was a Member of Technical Staff of AT&T Bell Laboratories from 1988 to 1991. Since 1991, he has been with the Department of Information Engineering, The Chinese University of Hong Kong, where he is now a chair professor. He is also a Changjiang Chair Professor at Xidian University (2009–2012) and an Advisory Professor at Beijing University of Post and Telecommunications (2008–2011). He has held visiting positions at Cornell University, Nankai University, the University of Bielefeld, the University of Copenhagen, Tokyo Institute of Technology, and Munich University of Technology. He was a Consultant in a project of Jet Propulsion Laboratory, Pasadena, CA, for salvaging the malfunctioning Galileo Spacecraft and a Consultant for NEC, USA.

He is the author of the textbooks *A First Course in Information Theory* (Kluwer Academic/Plenum 2002) and its revision *Information Theory and Network Coding* (Springer 2008). His research interests include information theory and network coding.

Dr. Yeung was a member of the Board of Governors of the IEEE Information Theory Society from 1999 to 2001. He has served on the committees of a number of information theory symposiums and workshops. He was General Chair of the First and the Fourth Workshop on Network, Coding, and Applications (NetCod 2005 and 2008), a Technical Co-Chair for the 2006 IEEE International Symposium on Information Theory, and a Technical Co-Chair for the 2006 IEEE Information Theory Workshop. He currently serves as an Editor-at-Large of *Communications in Information and Systems*, an Editor of *Foundation and Trends in Communications and Information Theory* and of *Foundation and Trends in Networking*, and was an Associate Editor for Shannon Theory of this Transactions from 2003 to 2005. He was a recipient of the Croucher Foundation Senior Research Fellowship for 2000/2001, the Best Paper Award (Communication Theory) of the 2004 International Conference on Communications, Circuits and System (with C. K. Ngai), the 2005 IEEE Information Theory Society Paper Award (for his paper "Linear network coding" coauthored with S.-Y. R. Li and N. Cai), and the Friedrich Wilhelm Bessel Research Award of the Alexander von Humboldt Foundation in 2007. He is a Fellow of the IEEE and the Hong Kong Institution of Engineers.

Since January 2010, Dr. Yeung has been serving as a Co-Director of the Institute of Network Coding at The Chinese University of Hong Kong.