Knotwork Coding

Ángela I. Barbero
Dept. of Applied Mathematics
University of Valladolid
47011 Valladolid
Spain
E-mail: angbar@wmatem.eis.uva.es
Under grant PR-2005-0011 MEC Spain

Øyvind Ytrehus
Dept. of Informatics
University of Bergen
N-5020 Bergen
Norway
E-mail:oyvind@ii.uib.no

***Abstract*— In [3], we described an algorithm for producing network codes for networks with simple cycles. In this paper we discuss strategies for designing codes in networks with *knots* [3].**

I. Introduction

The Linear Information Flow (LIF) algorithm, presented by Jaggi *et al.* in [1], is a polynomial time centralized algorithm for producing the network encoding equations in an error- and erasure free multicast network. The algorithm requires that the *flow path graph* (see Section II) associated with the network is acyclic.

While playing with our own flow path algorithms [6] we have observed that many "real" situations require a cyclic flow path graph. We are not aware of many papers that deal directly with the problem of encoding for cyclic networks. Fragouli and Soljanin [2] provide an interesting notation for describing the encoding process, but do not relate the cyclic networks directly to the efficient approach of the LIF algorithm. Two LIF-related algorithms, the LIFE and LIFE-CYCLE algorithms, were developed by us in [3] to deal with networks that contain simple cycles.

In this paper we discuss the encoding of more complex cycles. Section II gives the necessary notation and background. We describe the algorithms from [3] in Section III, and discuss *knotwork* coding in Section IV.

II. Notation and previous results

Consider the network G, where $G = (N, E)$ is a directed multigraph. N represents the set of nodes (or vertices) and E the set of edges or links, each link of unit capacity. Note that any network with links of integer capacity can be represented by this model, using multiple parallel edges when necessary. Let $S = \{s_1, s_2, \ldots, s_h\} \subset N$ be the set of unit rate information sources and $T = \{t_1, t_2, \ldots, t_r\} \subset N$ the set of sinks. We assume that no edge enters any source and that no edge exits from any sink. We will denote by σ_i the symbol produced by source s_i. In case the given network has one unique source s that sends information at rate h we will simply create virtual sources $s_1, \ldots, s_h$ and link them to the actual source s. Any similar scenario concerning the number of sources and the rate of information delivered by them can be analogously treated. For each edge $e \in E$ we will denote by $start(e)$ and $end(e)$ the nodes at which e starts and ends, respectively.

A. Flow paths

We will assume that for each $s \in S$ and each $t \in T$ there exists a *flow path* $f^{s,t}$, such that for each sink t all the paths $f^{s_i,t}, i = 1, \ldots h$, are edge disjoint. Following the notation of [1] we will denote by f^t the union of all the flow paths $f^{s_i,t}, i = 1, \ldots, h$ and we will call it the *flow* for sink t. In the following, we will assume that the set of flows $f = \bigcup f^t$ has been found by some deterministic algorithm, and thus is an implicit part of the network description. We will call f the *flow path graph* corresponding to G. The edges that do not lie on any flow path f^{s_i,t_j} can be discarded, since there is no need to send any symbol on them. Hence we can consider only networks where each edge is on some flow path.

The results in [4] guarantee that in such a network, all the sinks can receive all the h input symbols produced by the h sources, and the results in [5] state that it can be done using linear coding on the network. In this paper we deal with linear coding and each edge will be encoded using a linear encoding equation.

We emphasize that the subsequent discussion on cyclicity of networks applies to a specific flow path graph for the network. In general a multicast network may have several valid flow path graphs, each of which may have different properties. In [6] we address the issue of constructing flow path graphs.

B. A taxonomy of cycles in flow path graphs

We may assume now that G is a flow path graph, i. e. any edge in E is in some flow path. The flow path graph may contain different types of cycles. We start by considering the simplest one.

Definition 1 (Link cyclic and link acyclic networks): A *link cyclic* network is a network where there exists a cyclic subset of edges, i. e., a set $\{e_0, e_1, \ldots, e_{p-1}, e_p = e_0\} \subset E$ for some positive integer p such that $end(e_i) = start(e_{i+1})$ for $0 \leq i < p$. If no such cycle exists, the network is *link acyclic*.

Suppose e is an edge that lies on the flow path $f^{s,t}$. We will denote by $f^t_{\leftarrow}(e)$ the predecessor of edge e in that path. There is no ambiguity in the notation: e can lie on several flow paths

$f^{s,t}$ for different sinks t, but not for different sources s and the same sink t, since all the flow paths arriving in t are edge disjoint.

Let $T(e) \subseteq T$ denote the set of sinks t that use e in some flow path in f^t, and let $P(e) = \{f^t_{\leftarrow}(e) \mid t \in T(e)\}$ denote the set of all predecessors of edge e. Note that all edges will have some predecessor ($P(e) \neq \emptyset$), except those with $start(e) = s_i$ for $i \in \{1, \ldots, h\}$.

We will introduce some extra notation to denote the temporal order induced in the edges by the flow paths in f. When two edges e_1 and e_2 lie on the same flow path and e_1 is the predecessor of e_2 in that path, we will write $e_1 \prec e_2$. We will use transitivity to define relationships among other pairs of edges that lie on the same path but are not consecutive. We observe that in each path the relation $\prec$ defines a total order in the edges that form that path, since a path that contains cycles, that is, edges that satisfy $e_1 \prec e_2 \prec \cdots \prec e_n \prec e_1$ can be simplified by avoiding taking the trip around that cycle, as shown in Figure 1.

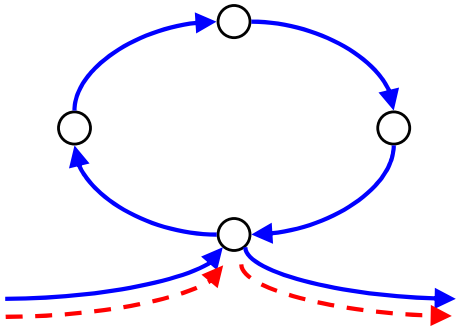

Fig. 1. Each flow path can be chosen to have no cycles.

Definition 2 (Flow cyclic and flow acyclic networks): A *flow acyclic* network is a network where the relation $\prec$ defines a partial ordering in E. If the relation does not define a partial ordering, the network is flow cyclic.

Note that a flow cyclic network is always link cyclic, but the converse is not true.

Lemma 1: Let $C = \{e_0, e_1, \ldots, e_{p-1}, e_p = e_0\} \subset E$, for some positive integer p, such that $end(e_i) = start(e_{i+1})$ for $0 \leq i < p$, be a link cycle. This link cycle is a flow cycle only if for each cycle edge e_i and its predecessor e_{i-1} there exists a flow path $f^{s,t}$ for some s, t that contains e_{i-1} and e_i.

Definition 3 (Simple flow cycles): A flow cycle is *simple* if it does not share an edge with another flow cycle.

Definition 4 (Knots): A collection of flow cycles $K = \{C_1, \ldots, C_q\}$ is a *knot* if each flow cycle C_i shares at least one edge with another flow cycle C_j and the set K is *connected*, i. e. it cannot be partitioned into disjoint subknots $K_1 = \{C_1, \ldots, C_p\}$ and $K_2 = \{C_{p+1}, \ldots, C_q\}$ such that no $C_i \in K_1$ shares any edge with a flow cycle $C_j \in K_2$.

Figure 2 shows examples of the three types of cycles. Only the edges and nodes that are part of the cycles are shown.

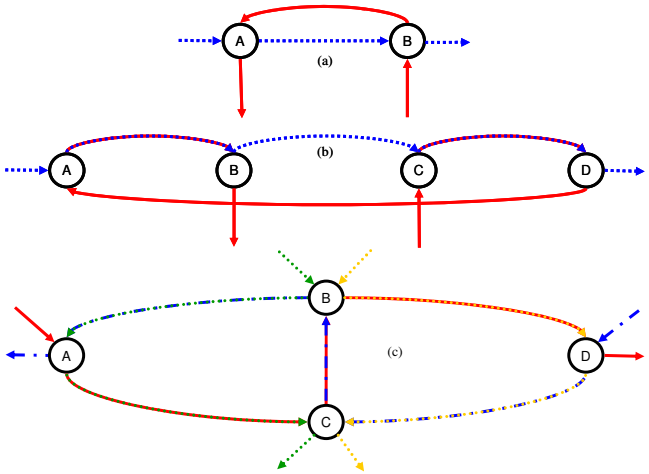

Fig. 2. Examples of (a) link cycle, (b) flow cycle, (c) knot.

III. The Science of Linear Network Coding

Let $\mathbb{F}_q$ be the finite field used to represent the symbols sent along the edges of the network. Jaggi et al. [1] presented a deterministic algorithm for coding in G, provided the size of the field used is sufficiently large[1] and *provided the network G is not link cyclic.* The LIF algorithm and its relatives [3] are efficient because they work along a flow path graph for the network.

A. The LIF and the LIFE algorithms

In what follows we will consider two ways of expressing the symbol $v(e) \in \mathbb{F}_q$ to be transmitted on edge e. The *local encoding equation* specifies the action of each router. The local encoding equation is

$$v(e) = \sum_{p \in P(e)} m_e(p) v(p), \tag{1}$$

where $m_e(p) \in \mathbb{F}_q$ are the coefficients of the linear combination of the symbols received on the predecessor edges for e. We remark that employing a time invariant encoding (i. e., the coefficients $m_e(p)$ do not depend on the time instant), as in this case, ensures that streams of symbols from each source can be pipelined through the network.

The *global encoding equation* expresses the symbol $v(e)$ to be transmitted on edge e in terms of the source symbols:

$$v(e) = \sum_{i=1}^{h} \alpha(e, i) \sigma_i, \tag{2}$$

where $\alpha(e, i)$ is the coefficient that specifies the influence of source symbol σ_i on $v(e)$.

The goal of the LIFE algorithm is to determine the local encoding equations of each edge e in each flow path, and hence the global encoding equation of the last edge in each

[1] In [7] we present a heuristic opportunistic algorithm that seek to produce an encoding in a "small" field.

flow path. In fact, during the course of the algorithm, all the global encoding equations will be determined since each global encoding equation is determined by the corresponding local encoding equation together with the global encoding equations of the predecessors on the flow paths.

In particular, the global encoding should be such that each sink can extract all the h information symbols from the h sources. Each sink t receives h symbols, namely $\{v(e_i) \,|e_i \in f^{s_i,t}, t = end(e_i), i = 1, \ldots, h\}$. If the corresponding h global encoding equations are linearly independent, the sink t can reconstruct the h input symbols by solving the set of equations.

In the case of a flow acyclic network, all the edges in E can be visited according to the partial order $\prec$ induced by the flow paths, starting by the edges with no predecessor ($e \in E \mid start(e) = s_i$ for some i), and proceeding in such a way that an edge e will not be visited until all its predecessors (all $p \in P(e)$) have already been visited and their global encoding equations computed. If the global encoding equations of all predecessors of e are substituted into (1), we get the global encoding equation for e.

Similar to the LIF algorithm, the LIFE algorithm [3] presented below proceeds by maintaining, through the iterations of the main loop, the following sets for each $t \in T$:

- A set $E_t \subset E$, $|E_t| = h$, such that E_t contains *the most recently visited edge* on each flow path in f^t.
- A set of global encoding equations, $V_t = \{(v(e) = \sum_{i=1}^{h} \alpha(e,i)\sigma_i) \mid e \in E_t\}$,

and the *full rank invariant: The set of equations V_t has full rank.*

ALGORITHM LIFE (Linear Information Flow on Edges)

1) **Input:** A directed multigraph G; a set of flow paths f.
2) **Initialization:**

   $\forall t$:
   - $E_t = \{e | e \in f^{s_i,t}, start(e) = s_i, i = 1, \ldots, h\}$,
   - $V_t = \{v(e) = \sigma_i | \; e \in E_t\}$,

   that is, if $e \in f^{s_i,t}$ with $start(e) = s_i$, then $v(e) = \sigma_i$.
3) **Main Loop:**

   - Select an edge e for which the encoding equations have not yet been determined, but for which the global encoding equations of all the predecessor edges in $P(e)$ have been determined. Then update the set of current edges,

     $$E_t := (E_t \setminus \{f^t_{\leftarrow}(e)\}) \cup \{e\}, \text{ for each } t \in T(e)$$

   - Further, for each $t \in T(e)$, replace in V_t the global encoding equation for $f^t_{\leftarrow}(e)$ with that of the new edge e. Provided the field $\mathbb{F}_q$ is sufficiently large, it is always possible to determine a local encoding as given by (1) in such a way that the full rank invariant is satisfied. See [1] for details.
4) **Output:**

   - For each edge, the local encoding as given by (1) is produced.
   - At the end of the algorithm, $E_t = \{e \mid end(e) = t\}$ and V_t is still a set of h linearly independent equations from which t can recover the input.

The algorithm will execute the main loop $|E|$ times. The exact complexity is polynomial and depends on details of the algorithm not discussed here.

The only difference between the LIFE algorithm and the LIF algorithm is that the LIF algorithm proceeds by visiting the nodes in N, while the LIFE algorithm proceeds by visiting the edges in E. This apparently subtle difference allows the LIFE algorithm to deal with many network structures with which the LIF algorithm cannot work, namely, those which are flow acyclic but link cyclic. This is due to the fact that the LIF algorithm cannot visit the same node several times, and a node n cannot be visited until all the encoding equations of the edges e such that $end(e) = n$ have been already computed.

B. Network coding of flow cycles

As shown in the previous subsection, the encoding can be straightforward as long as the flow graph does not contain flow cycles. But in the presence of flow cycles we need to expand the notation and to introduce new concepts.

We will first state some more assumptions about the communication model. At time x, each source s_i will *synchronously* generate a symbol $\sigma_i(x)$. We will refer to the collection of source symbols generated at time x as *generation* x. Further, we will for simplicity assume that all edges in the network *synchronously* will transmit one symbol per time instant. The local encoding equation for each edge will be time invariant also in the case of flow cyclic networks. However, in this case there will be edges where the corresponding global encoding equation is a linear combination of symbols possibly from different sources and from different generations.

Hence the actual symbol transmitted on edge e at time x is, expressed as a global encoding of the source symbols,

$$v(e;x) = \sum_{i=1}^{h} \sum_{y=0}^{\Delta_{start(e)}} \alpha(e,i,y)\sigma_i(x-y), \qquad (3)$$

where Δ_n is the maximum delay from any source to node n (in general Δ_n does not even have to be finite (but it would be an advantage if it is (and in practice the sum in (3) at a given time x is always finite since it assumed that $\sigma_i(x) = 0, \forall x < 0)$)), and $\alpha(e,i,y)$ is a coefficient denoting the influence of the symbol $\sigma_i(x-y)$ in the linear combination. The *value* $v(e;x)$ is a *snapshot* of the information transmitted on edge e at time x. This global encoding is still achieved by a time invariant local encoding of each edge, like in (1), which now takes the

form

$$v(e;x) = \sum_{p \in P(e)} m_e(p) v(p; x-1). \quad (4)$$

When the flow path graph is flow cyclic, symbols may circulate in the graph, and routers and sinks may receive symbols that may be "contaminated" by old symbols. As long as the nature of this contamination is determined, the sinks may still be able to extract the desired information. However, we point out two disadvantages of allowing old symbols to circulate in the network:

- When a network node (here, a sink or a router) receives a symbol on a link that contains new information mixed with old information symbols previously known to the node, the new information can of course be recovered, but this requires that the node keeps such old information symbols in memory for when they are needed. In the worst case it may require an infinite memory.
- We assume an error-and-erasure free network. If we relax this condition, however, the existence of old symbols circulating in the network may lead to error propagation.

For these reasons we suggest that an encoding algorithm for flow cyclic graphs should seek to minimize the effects of old symbols circulating in the graph.

Hence, an algorithm in the LIF family, i. e. an algorithm that works on the flow path graph by extending one or a few edges at the time, needs to consider the following objectives at each extension step:

- The full rank invariant must be maintained.
- The propagation of symbols in the cycle should be minimized.

In [3] we developed the LIFE-CYCLE algorithm which deals with networks that contain *simple* flow cycles. The algorithm proceeds like the LIFE algorithm. If there is no edge e ready for encoding at the beginning of the main loop, it means the algorithm has encountered a flow cycle. If this flow cycle is simple we can use the principle of old symbol removal:

Definition 5 (Principle of old symbol removal): If the symbol v enters a node on the cycle from "outside" through edge p at time $x-1$, it must be removed again one cycle-length later. To be more formal; if e is on a simple flow cycle of length λ and q is the predecessor of e on the flow cycle, then the local encoding for edge e will be

$$\begin{aligned} v(e;x) &= m_e(q) v(q; x-1) \\ &\quad + \textstyle\sum_{p \in P(e) \setminus \{q\}} m_e(p)(v(p; x-1) - v(p; x-\lambda)). \end{aligned} \quad (5)$$

The old symbol removal guarantees that on the simple flow cycles, there will be no symbol propagation. The LIFE-CYCLE algorithm works by extending all edges on the flow cycle at once. It is shown in [3] that the encoding coefficients $m_e(\cdot)$ can be chosen in such a way that the full rank invariant is maintained. Moreover, the delay $\Delta_{start(e)}$ is always finite and hence the memory needed by routers and sinks is also always finite.

IV. The art of linear network coding for knots

I just carve away anything that isn't art. - Michelangelo

This section concerns coding for flow path graphs that contain knots. We will illustrate the issues by presenting three examples at the end of the section.

The ideas presented in the previous section are still useful, but the way to use them can vary depending on the particular structure of the knot and/or on the encoding features (memory requirements, less error propagation, etc.) to which we want to give priority. In general the encoding for each network is not unique.

We have failed to find a general procedure that will work in every possible case. However, we can offer some observations:

1) The principle of old symbol removal, suitably adjusted for each case, works for all the examples we have studied, but in some of these examples alternative encodings can be designed without using this principle.
2) Encoding by passing information along the flow paths, rather than passing on information from all edges entering a node to all edges emerging from the node, seems to be a good idea.
3) The sink and router memory requirements are finite in all the examples we have studied, even when the delay $\Delta_{start(e)}$ is sometimes infinite. Furthermore, the delay with which each symbol can be recovered at each sink is exactly equal to the length of the flow path that carries it.
4) If the network allows an encoding where an input symbol (to the knot) is passed, with the same local encoding coefficient, onto each edge in the knot, then Mason's formula [8],[9] can be used to calculate the propagation of that symbol through the knot. This can be used for the old symbol removal and for calculating what is passed on out of the knot. We apologize that we, due to lack of space, will not explain this method in detail. The notation we use in Example 2 is similar to the one used in Chapter 11 of [9].

Example 1: [3] Figure 3 shows a knotwork with four sources (A,B,C,D), three sinks (t_1, t_2, t_3), and four intermediate routers that together with the colored flow paths make up the knot from Figure 2 (c). Many encodings are possible; we show one. A symbol transmitted from source A at time z is denoted by $a(z)$, and similarly for the other sources. The figure shows a snapshot, i. e., what is being transmitted on each edge, at time x. The example uses all four suggestions above.

Example 2: Figure 4 shows another knotwork with four sources (A,B,C,D), three sinks (t_1, t_2, t_3), and four intermediate routers. The drawing is simplified; imagine direct edges between sources and sinks where flows are not already shown. Mason's formula can be used to calculate how old symbols can be contained. A symbol from source A that enters the knot at router E will propagate through the knot and return to E (mixed with other symbols) through the edge GE. The

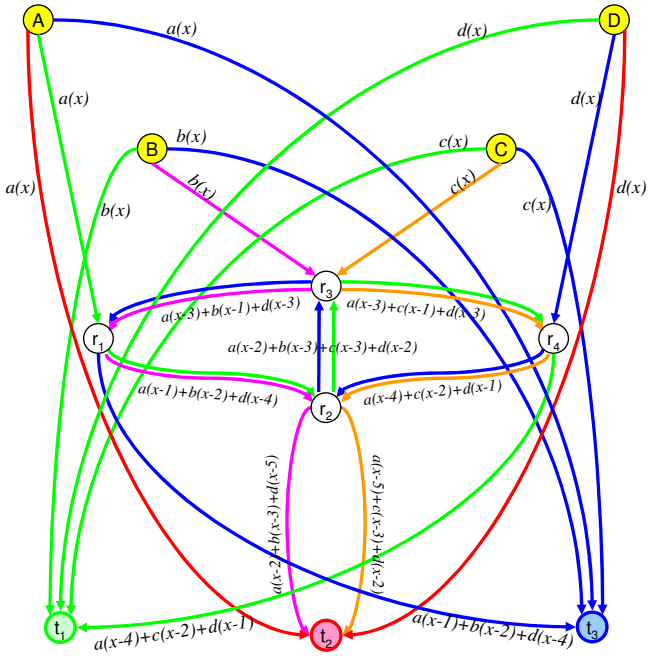

Fig. 3. A simple knotwork.

transfer function given by Mason's formula for this symbol on this edge is $x^3(1+x^3)/(1+x^3+x^4)$. This can be used by router E for old symbol removal. Similarly, the transfer function for this symbol arriving on the edge IJ is $x^3/(1+x^3+x^4)$, to be passed on to the blue sink. Node J can do the same old symbol removal for symbols arriving from D. Nodes G and I can apply a straightforward symbol removal of the symbols from B and C, respectively. We leave the encoding to the reader as an exercise (we have an encoding but there was no space in the margin to present it.)

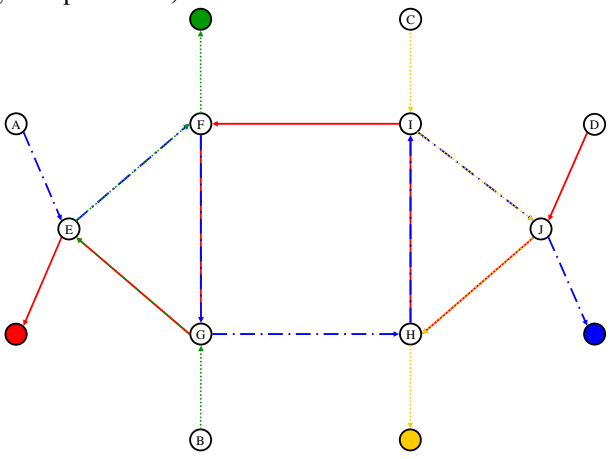

Fig. 4. A case where Mason's formula works. (Hint: zoom in.)

Example 3: Figure 5 shows yet another knotwork with six sources (A, B, C, D, E, F) and three sinks. As the figure shows, all "internal" edges need to be shared by two flow paths. Mason's formula turns out to be a disappointment in this case (why?). Still we have found several encodings; one is shown in the figure. Here $A(z) = \sum_{i=0}^{k(z)} a(3i+l(z))$, where the functions $k(y)$ and $l(y)$ are determined by $z = 3k(z) + l(z))$ and $0 \le l(z) \le 2$. The encoding uses suggestions 1, 2, and 3 above.

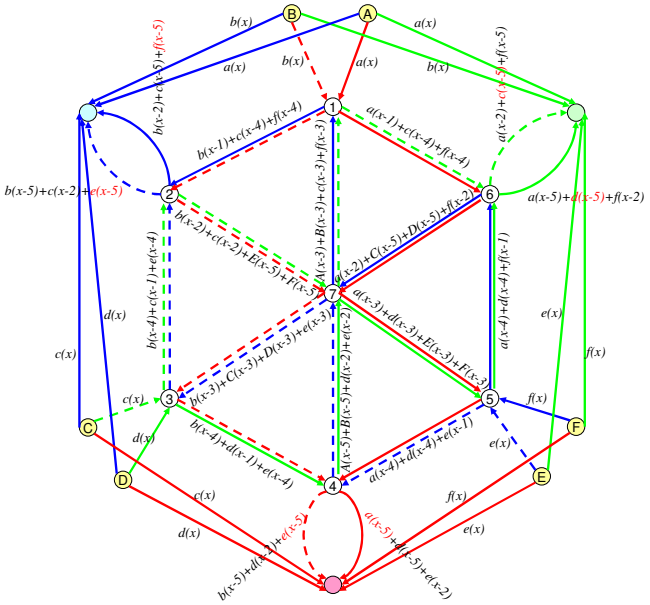

Fig. 5. The big knot.

A final word of comfort: In a practical system with many sources and sinks we anticipate that link cycles will occur very often, flow cycles are rather unlikely, and random knots will be very rare outside of the laboratory.

References

[1] S. Jaggi, P. Sanders, P. A. Chou, M. Effros, S. Egner, K. Jain, and L. Tolhuizen, "Polynomial Time Algorithms for Multicast Network Code Construction", *IEEE Transactions on Information Theory,* Vol. 51, June 2005, pp. 1973-1983.

[2] C. Fragouli and E. Soljanin, "A Connection Between Network Coding and Convolutional Codes", *Proc. ICC 2004,* Paris, June 2004, pp. 661-666.

[3] Á. Barbero and Ø. Ytrehus, "Cycle-logical treatment of cyclopathic networks", accepted for publication in *IEEE Transactions on Information Theory*, 2006.

[4] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network Information Flow", *IEEE Transactions on Information Theory,* Vol. 46, April 2000, pp. 1204-1216.

[5] S.-Y. R. Li, R. W. Yeung and N. Cai, "Linear Network Coding", *IEEE Transactions on Information Theory,* Vol. 49, February 2003, pp. 371-381.

[6] Á. Barbero and Ø. Ytrehus, "On flow paths for network coding", submitted to *ISIT 2006*.

[7] Á. Barbero and Ø. Ytrehus, "On the required field size of the Linear Information Flow family of algorithms", submitted to *ISIT 2006*.

[8] S. Mason and H. Zimmermann, *Electronic Circuits, Signals, and Systems*, John Wiley, New York, N. Y. 1960.

[9] S. Lin and D. Costello, *Error Control Coding, 2nd Ed.*, Pearson Prentice Hall, 2004.