Improving the Multicommodity Flow Rates with Network Codes for Two Sources

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

***Abstract*—In this work we introduce a construction and analysis of network codes for two sources. The region of achievable rates for this problem is still unknown. The scheme we suggest is based on modifying the multicommodity flow solution and thus improving the achievable rate region, w.r.t the uncoded case. The similarity to the flow problem allows our method to be implemented distributively. We show how the construction algorithm can be combined with distributed backpressure routing algorithms for wireless ad-hoc networks. For both the non-distributed case and the distributed case, the computational complexity of our algorithm for network coding is comparable to that of the parallel multicommodity flow problem. We provide non trivial upper and lower bounds on the performance of our scheme, using random coding techniques.**

***Index Terms*—Multicommodity flow, Backpressure, Distributed network coding, Multiple unicast, Ad-hoc networks**

I. Introduction

MOST literature on network coding focuses on multicast where bounds on achievable rates and codes that achieve these bounds were found. The general case of multiple sources seems to be much more complicated. A classification complexity of network coding problems was carried out in [1], where it was shown that some of the non-multicast cases are NP-hard. For this category linear codes cannot achieve in general the optimal rates. It was further shown in [2] that vector linear codes of any finite dimension cannot achieve optimal rates for some networks. Koetter and Médard [3] gave an algebraic formulation of linear network codes with multiple sources. Yeung [4, Chapter 15] gave information-theoretic inner and outer bounds to the rate region in the general case of acyclic networks with multiple sources. This result is extended in [5] to zero-error network codes. While these schemes are elegant and insightful, the computational cost is high for practical implementation.

In multiple unicast for d users, source s_i transmits to sink t_i at rate R_i. When no codes are employed, the problem is termed the multicommodity flow, which can be solved using linear programming. In [6] it was shown that the problem can be implemented distributively. In [7], which considers network coding for multiple unicast, the operations are restricted to binary XOR. The scheme is systematic but the computational complexity is high in comparison to the multicommodity flow. This approach was solved distributively in [8],[9]. A different approach in the wireless setting is given in [10] and is again limited to XOR operations, which may lead to throughput loss. In [11] the problem is formulated as finding the stable set of the conflict hypergraph of the network. However, the size of the conflict graph grows exponentially and finding the stable set may be difficult.

We provide a code construction that improves the rate region of multicommodity flows, with computational complexity comparable to that of the multicommodity flow construction. We focus on multiple unicast with two users. We formulate our method as a linear programming that is closely related to the flow problem. The similarity to the flow problem allows our method to be implemented distributively, analogously to the backpressure algorithm in [6]. There is only one exception for the distributivity, in the sense that the sources do require a small amount of feedback from the sinks at the setup stage of the code construction. Other than that, there is no global mechanism responsible for the communication scheme. The coding coefficients are taken from a general field and unlike many previous schemes operations are not restricted to XOR, giving rise to richer families of codes.

The distributed implementation is especially important for ad-hoc wireless networks, where nodes may have knowledge only on their outgoing links. We show how to combine our construction with the backpressure algorithm for wireless ad-hoc networks [12]. Unlike previous schemes, for both the non-distributed and the distributed cases, the complexity of our network coding construction is comparable to that of multicommodity. For the case of random codes, we find non-trivial upper and lower bounds on the performance of our construction. These results partially appear in [13], [14].

II. Definitions, Notations and Preliminaries

Consider an acyclic, unit capacity directed network $G = (V, E)$ where parallel edges are allowed. There are two sources $s_1, s_2 \in V$ and two sinks $t_1, t_2 \in V$. Source s_i transmits to sink t_i at rate R_i. As in [15], when convenient we add a dummy source s_i' which is connected to source s_i with R_i edges $\{e_1^i, \ldots, e_{R_i}^i\}$. The dummy source s_i' is mentioned only when necessary for ease of description. The size of the minimal cut between s_i and sink t_i is h_i.

Define for each source-sink pair $s_i - t_i$, the subgraph G_i of G containing only the nodes and edges that participate in a certain flow of magnitude h_i from s_i to the sink t_i. Denote by $\Gamma_I(v)$ and $\Gamma_O(v)$ the set of incoming and outgoing edges of node v, respectively. Similarly, denote by $\Gamma_I(e)$ and $\Gamma_O(e)$ the set of incoming edges of the tail of e and outgoing edges of the head of e, respectively.

Unless otherwise specified, we assume that the topology of the network is completely known to the code designer. In this case of a known network, we assume that G is given by:

$$G = \cup_{i=1,2} G_i \tag{1}$$

Manuscript received 31 July 2008; revised 20 February 2009.

Elona Erez is with Department of of Electrical Engineering, Yale University, New Haven, CT, 06511, USA (e-mail: elona.erez@yale.edu).

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

Digital Object Identifier 10.1109/JSAC.2009.090620.

For a general network, we can always find a subgraph of the network that has the form of (1) and implement our algorithm for that subgraph.

For linear network codes, any edge e has a global coding vector $\mathbf{v}(e)$ of dimension $R_1 + R_2$ associated with it. For algebraic block network codes, the dummy source node s_i' gets R_i input symbols denoted as $X^i = (X_1^i, \cdots, X_{R_i}^i)$ from the field $\mathcal{F}$. For an outgoing edge $e_j^i, 1 \leq j \leq R_i$ of s_i' all the coordinates are zero except for the coordinate $\sum_{k<i} R_k + j$, which is equal to X_j^i. For the rest of the edges $\mathbf{v}(e)$ is given recursively by:

$$\mathbf{v}(e) = \sum_{e' \in \Gamma_I(e)} m(e', e)\mathbf{v}(e') \tag{2}$$

where e' is an incoming edge of e, and $m(e', e)$ is the coding coefficient. The symbol on e is

$$y(e) = \sum_{e' \in \Gamma_I(v)} m(e', e)y(e') = \mathbf{v}(e)^T(X^1, X^2) \tag{3}$$

where (X^1, X^2) is a vector of dimension $R_1 + R_2$ containing the input symbols of s_1 and s_2.

III. Code Construction

We start our construction with global coding vectors of dimension $h_1 + h_2$. The first h_1 coordinates are associated with s_1 and the last h_2 with s_2. The vector of dimension h_1 containing the first h_1 coordinates of $\mathbf{v}(e)$ is denoted by $\mathbf{v^1}(e)$, and analogously $\mathbf{v^2}(e)$ is defined for the last h_2 coordinates. We find an achievable rate region which improves that of the multicommodity flows. We start by a special case, where one of the sources, say s_2, transmits at its maximal flow rate h_2 and find a possible rate for s_1. We show that this rate of s_1 is better than the best multicommodity flow rate. We generalize this special case by considering a certain point on the boundary of the multicommodity flow rate region (R_1, R_2). We show how R_1 can be improved for a given R_2 using network codes.

A. Code Construction for a Special Case

Assume that s_2 transmits at its maximal rate $R_2 = h_2$ and that s_1 tries to transmit at a certain, as high as possible, rate. The construction includes three stages. The first stage is the internal coding which determines the coding coefficients of the intermediate nodes. The second and the third stages are the constraints setting stages, Stage A and Stage B. These stages determine the additional required coding at the s_1. Stage A ensures that t_2 would be able to reconstruct s_2 and Stage B ensures that t_1 would be able to reconstruct s_1. It will be shown that with this scheme for each 2Δ bits that s_1 reduces its rate below h_1, s_2 would be able to increase its rate by at least Δ bits, as long as the capacity constraints are not violated. This tradeoff is not always possible without coding.

1) Internal Coding: This stage is similar to the polynomial time algorithm for multicast [15]. A path p is a sequence of consecutive edges. A flow is a set of edge disjoint paths between two nodes. The algorithm starts by finding the maximal flows G_1 of rate h_1 from s_1 to t_1 and G_2 of rate h_2 from s_2 to t_2. Each of the flows $G_l, l = 1, 2$ consists of h_l edge disjoint paths, denoted by $\{p_1^l, \cdots, p_{h_l}^l\}$. There can be multiple choices for G_1 and G_2, since there may be several different maximal flows associated with each source-destination pair. For our purposes, we choose arbitrary maximal flows $G_l, l = 1, 2$. We use only edges in the subgraph $G_1 \cup G_2$, so we assume that our original network is $G = G_1 \cup G_2$. The algorithm steps through the edges in G in topological order. For edge e a coding vector of dimension $h_1 + h_2$ is assigned. For e that participates in either G_1 or G_2, but not both, we assign $m(e', e) = 1$ for e' that preceded e in the flow. For e that participates in both $G_l, l = 1, 2$, the coefficients $m(e_1, e)$, $m(e_2, e)$ are determined, where e_1 precedes e in G_1 and e_2 precedes e in G_2.

The code coefficients are drawn from an algebraic field $\mathcal{F}$. The set $\mathcal{C}_l, l = \{1, 2\}$ contains one edge from each of the paths $\{p_1^l, \cdots, p_h^l\}$ in the flow $G_l, l = \{1, 2\}$, the edge whose global coding vector was defined most recently. Denote the global coding vectors of the edges in $\mathcal{C}_l, l = \{1, 2\}$ by $V_l = \{\mathbf{v}(e) : e \in \mathcal{C}_l\}$. For $l = 1, 2$, denote by $V_l^1 = \{\mathbf{v^1}(e) : e \in \mathcal{C}_l\}$ the corresponding set of vectors of dimension h_1 and by $V_l^2 = \{\mathbf{v^2}(e) : e \in \mathcal{C}_l\}$ the corresponding set of vectors of dimension h_2. The invariant maintained by the algorithm is that for t_1 the set V_1^1 spans $\mathcal{F}^{h_1}$, while for t_2 the set V_2^2 spans $\mathcal{F}^{h_2}$. Similarly to the proof in [15] for multicast, it can be seen that for field size two or larger, we are ensured that at least a single $m(e', e)$ in $\mathcal{F}$ maintains the invariant. At the end of the construction the edges in $\mathcal{C}_l, l = 1, 2$ are $\Gamma_I(t_l)$. The property of a code constructed by this scheme is that a certain sink can decode the data intended for it, provided that the other source is silent.

An example of a code over the binary field is given in Figure 1, where $h_1 = 3, h_2 = 1$. In Figure 1(a) G_1 is in thick lines, while in 1(b) G_2 is in thick lines. Define the symbols a_1, a_2, a_3 as the symbols transmitted by t_1 on its outgoing links from left to right, respectively. The sets $\mathcal{C}_1, \mathcal{C}_2$ at the end of the internal coding are also specified. At each stage of the internal coding $\mathcal{C}_1$ contains one edge from each of the three paths from s_1 to t_1. In Figure 1(a), $\mathcal{C}_1$ contains e_1, e_2, e_3 which are the edges in the three paths from s_1 to t_1, from left to right, respectively. At the end of the internal coding $V_1 = \{(1, 0, 0, 1)^T, (1, 1, 0, 1)^T, (1, 1, 1, 1)^T\}$. It follows that $V_1^1 = \{(1, 0, 0)^T, (1, 1, 0)^T, (1, 1, 1)^T\}$. Likewise $V_2 = \{(1, 1, 1, 1)^T\}$ and $V_2^2 = \{(1)\}$. Note that when $R_2 = h_2 = 1$, we cannot achieve a nonzero rate R_1 using multicommodity flow.

2) Constraints Setting- Stage A: Consider $\mathcal{C}_2$, which at the end of the internal coding is $\Gamma_I(t_2)$. The sink t_2 will be able to decode at rate h_2 if all interference noise from s_1 is canceled. The dimension R_{12} of the vector space spanned by V_2^1 is $R_{12} \leq \min\{h_1, C_{12}, h_2\}$, where C_{12} is the capacity from s_1 to t_2. We choose a subset of V_2^1 of size R_{12} that is the basis of V_2^1. Denoted the basis by $B_2^1 = \{\mathbf{v^1}(e_1), \ldots, \mathbf{v^1}(e_{R_{12}})\}$ and the edges whose vectors are in B_2^1 by $\mathcal{C}_2^1$. In Figure 1, we have $R_{12} \leq h_2 = 1$ and $V_2 = \{\mathbf{v}(e_3)\} = \{(1, 1, 1, 1)^T\}$. The vector in V_2^1 is given by $V_2^1 = \{\mathbf{v^1}(e_3)\} = \{(1, 1, 1)^T\}$, and so $B_2^1 = \{\mathbf{v^1}(e_3)\} = \{(1, 1, 1)^T\}$ and $\mathcal{C}_2^1 = \{e_3\}$. In order for t_2 to achieve rate $h_2 = 1$ we set the constraint $a_1 + a_2 + a_3 = 0$. In general, the interference noise can be canceled by forcing the symbols on $\mathcal{C}_2^1$ to be functions of s_2 only. The interference components on the other edges of $\mathcal{C}_2$ will also be

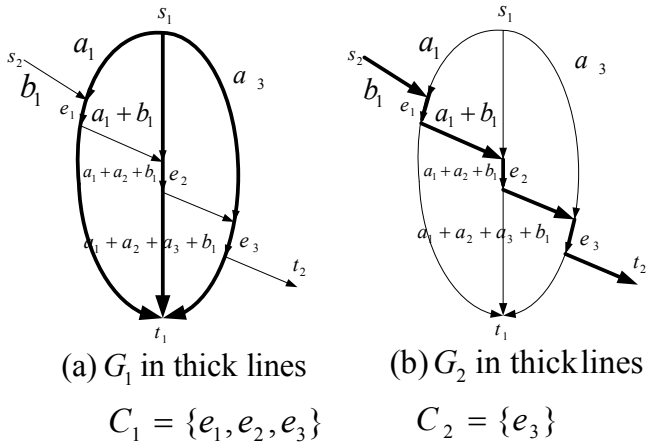

Fig. 1. Example Code for Internal Coding

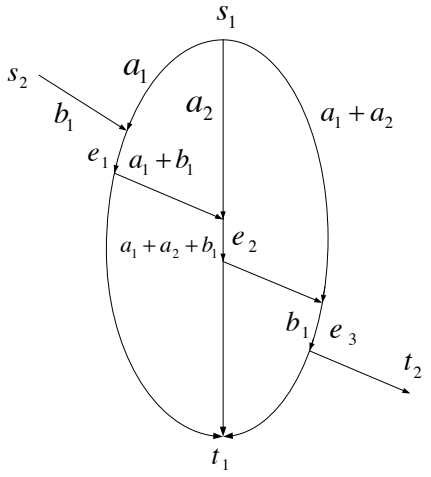

Fig. 2. Final Code for Example Network

canceled, since they are linear combinations of those on $\mathcal{C}_2^1$. Each edge in $\mathcal{C}_2^1$ would add at most a single constraint on s_1. Each such constraint reduces the rate of s_1 by at most a single bit. Therefore, rate $h_1' \geq h_1 - \min\{h_2, C_{12}, h_1\}$ is transmitted from s_1. If $h_1 > h_2$, h_1' is strictly positive.

3) Constraints Setting - Stage B: Consider t_1 and the edges in $\mathcal{C}_1$, which are now $\Gamma_I(t_1)$. The dimension R_{21} of the vector space spanned by V_1^2 is $R_{21} \leq \min\{h_2, C_{21}, h_1\}$, where C_{21} is the capacity from s_2 to t_1. We choose a subset of V_1^2 of size R_{21} that is the basis of V_1^2, denoted by $B_1^2 = \{\mathbf{v^2}(e_1), \ldots, \mathbf{v^2}(e_{R_{21}})\}$. Denote the nodes whose vectors are in B_1^2 as $\mathcal{C}_1^2$. In Figure 1, $R_{21} \leq h_2 = 1$ and $V_1^2 = \{\mathbf{v^2}(e_1), \mathbf{v^2}(e_2), \mathbf{v^2}(e_3)\} = \{(1), (1), (1)\}$. We choose $\mathcal{C}_1^2 = \{e_3\}$ and $B_1^2 = \{\mathbf{v^2}(e_3)\} = \{(1)\}$. In general, we find additional constraints on s_1, such that the symbols on $\mathcal{C}_1^2$ depend on s_2 only. Sink t_1 can cancel the interference components on the other edges in $\Gamma_I(t_1)$ using the the symbols on $\mathcal{C}_1^2$, since they are linearly dependent. Each edge in $\mathcal{C}_1^2$ adds at most a single constraint on s_1, which reduces the rate of s_1 by at most a single bit. Since the rate was already reduced to $h_1' \geq h_1 - \min\{h_2, C_{12}, h_1\}$, the final rate of t_1 is $h_1' \geq h_1 - \min\{h_2, C_{12}, h_1\} - \min\{h_2, C_{21}, h_1\} \geq h_1 - 2h_2$. If $h_1 > 2h_2$, h_1' is strictly positive.

In Figure 1, observe that since we already set the constraint $a_1 + a_2 + a_3 = 0$, t_1 is already able to decode b_1, which is the interference noise at t_1. The final rate at t_1 is $h_1' = h_1 - \min\{h_2, C_{12}, h_1\} = 3 - 1 = 2$, which is larger than the bound $h_1' \geq h_1 - 2h_2 = 1$. The final code is shown in Figure 2. The rate region is shown in Figure 3 for multicommodity flow, and for our coding scheme. For this example, our scheme is optimal since it achieves the cut set bound, which is 3. Note that the rate region has an angle 45^o with the negative x axis. This, however, turns out not to be the general case.

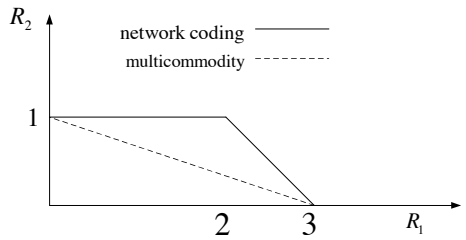

Fig. 3. Rate Region for Example Network

Note that in the original network there may be several choices for the maximal flows G_1 and G_2. Different choices of G_1 and G_2 might result in multiple code constructions and possibly different rates can be achieved. Nevertheless, enumerating all the possible flows is complicated. From practical viewpoint, we propose to choose certain flows G_1 and G_2 and operate on them. An interesting problem would be to find an efficient way to choose the optimal flows G_1 and G_2 for the purpose of coding.

B. Tradeoff Property of the Code

Theorem 1: If s_1 reduces its maximal rate h_1 by 2Δ bits, then there is a code such that s_2 will be able to transmit at a rate of at least Δ bits, as long as the minimal cut condition is not violated.

Proof: The proof is by constructing the code. We have shown in Section III-A that for the special case when s_2 transmits at its maximal rate $R_2 = h_2$ then s_1 can achieve rate which is at least $h_1 - 2h_2$. For the general case, we define a subgraph of G by $\tilde{G} = G_1 \cup G_2^{\Delta}$ where G_1 is a flow of size h_1 from s_1 to t_1 and G_2^{Δ} is a flow of size Δ from s_2 to t_2. For the subgraph $\tilde{G}$ we construct a code, according to the special case in Section III-A. The size of the maximal flow from s_1 to t_1 in $\tilde{G}$ is denoted by $\tilde{h}_1$ and according to construction $\tilde{h}_1 \geq h_1$. In fact, $\tilde{h}_1 = h_1$ since $\tilde{G}$ is a subgraph of G, and thus the maximal flow $\tilde{h}_1$ in $\tilde{G}$ cannot be larger than the maximal flow h_1 in G. The size of the maximal flow from s_2 to t_2 in $\tilde{G}$ is denoted by $\tilde{h}_2$ and according to construction $\tilde{h}_2 \geq \Delta$. If $\tilde{h}_2 > \Delta$, then s_2 can ignore the symbols that are not in G_2^{Δ} and not use them for decoding. Thus we can always assume that $\tilde{h}_2 = \Delta$. Therefore, the rate of s_1 in $\tilde{G}$ achieved by the code constructed according to Section III would be at least $\tilde{h}_1 - 2\tilde{h}_2 \geq h_1 - 2\Delta$. It follows that h_1 lost at most 2Δ bits relative to its maximal flow h_1 and that the rate of s_2 is at least Δ. $\diamond$

It follows that if we take the point $(h_1, 0)$ in the rate region and draw a line of slope $1/2$ (which is equivalent to a 22.5^o angle) with the negative x axis (that represents h_1), it will be within the capacity region of our scheme (as long as $R_2 \leq h_2$). This observation can be helpful in determining the situations for which our scheme is most useful. If for the multicommodity rate region the slope with the negative x of

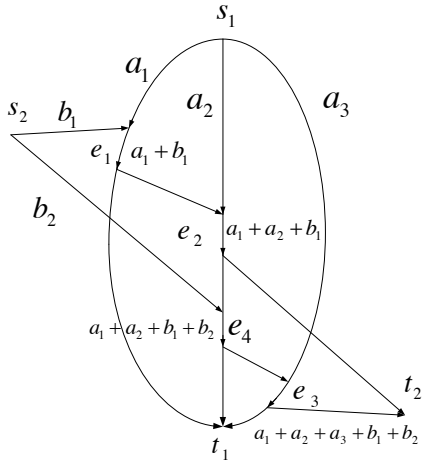

Fig. 4. Example Code for Counterexample Network

the line from $(h_1, 0)$ is smaller than $1/2$, then our coding scheme is guaranteed to improve the multicommodity rate region. This can be performed by the construction in the proof of Theorem 1, which achieves the rate pair $(h_1 - 2\Delta, \Delta)$, for Δ s.t. $R_1 = h_1 - 2\Delta \geq 0$ and $R_2 = \Delta \leq h_2$. Likewise, for $(0, h_2)$ if we draw a line of slope $1/2$ with the negative y axis, it will be contained in the capacity region of our scheme (as long as $R_2 \leq h_2$). If for the multicommodity rate region the slope with the negative y of the line from $(0, h_2)$ is smaller than $1/2$, then our coding scheme is guaranteed to improve the multicommodity region. Given a general point in the rate region (R_1, R_2) our coding scheme is guaranteed to improve the multicommodity rate region if the points $(R_1 - 2\Delta, R_2 + \Delta)$ or $(R_1 + \Delta, R_2 - 2\Delta)$ are not in the multicommodity rate region, for some $\Delta > 0$, if the individual capacity constraints are not violated. In Figure 1 the slope from the point $(h_1, 0) = (3, 0)$ with the negative x axis is $1/3$, which is less than $1/2$. Our coding scheme is guaranteed to improve the multicommodity rate region. We take the point $(3, 0)$ and apply the construction in the proof of Theorem 1 with $\Delta \leq h_2 = 1$ we are guaranteed to achieve the point $(3 - 2 \cdot 1, 1) = (1, 1)$, which is not achievable without coding. On the other hand, if we look at the point $(0, 1)$ in the multicommodity rate region, the slope with the negative y axis is 3 which is larger than $1/2$. It follows that we are not guaranteed to improve this point with our scheme.

For the network in Figure 1 we are able to improve the rate region beyond what Theorem 1 guarantees. In the coded case, the slope from $(h_1, 0) = (3, 0)$ is 1. For each Δ bits that s_1 reduces its rate below h_1, s_2 can increase its rate by Δ. The one-to-one tradeoff is not always possible. In Figure 4, where $h_1 = 3$, if a one-to-one ratio is possible, then we expect the rate pair $R_1 = 1, R_2 = 2$ to be achievable. A possible code is given, prior to the constraints setting. For $R_2 = 2$, the constraints setting reduces R_1 to zero. In fact, for this network $R_1 = 1, R_2 = 2$ is not achievable by any network code [14]. Thus for this example, the min-cut bound is not always achievable.

C. Improving the Multicommodity Flow

Suppose we are given a rate pair (R_1, R_2), which is on the boundary of the rate region of the multicommodity flow. We show how the solution (R_1, R_2) can be improved. Denote the flow of source $s_i, i = 1, 2$ at edge e as x_e^i. At edge e, from the flow conversion and the capacity constraints:

$$\sum_{e' \in \Gamma_I(e)} x_e^1 = \sum_{e' \in \Gamma_O(e)} x_e^1, \quad \sum_{e' \in \Gamma_I(e)} x_e^2 = \sum_{e' \in \Gamma_O(e)} x_e^2,$$
$$x_e^1 + x_e^2 \leq c(e) \ \forall e, \ x_e^1 \geq 0, \ x_e^2 \geq 0 \quad (4)$$

where for edge $e = (u, v)$, $c(e)$ is the multiplicity of the unit capacity edges between u and v. If rate (R_1, R_2) is achieved by the multicommodity flow,

$$x_s^1 = x_t^1 = R_1, \quad x_s^2 = x_t^2 = R_2 \quad (5)$$

where x_s^1 is the flow leaving s_1 and x_t^1 is the flow of s_1 reaching t_1. The solution to the multicommodity defines flow G_1' from s_1 to t_1 and flow G_2' from s_2 to t_2. Flow G_1' might be a subgraph of a larger flow G_1'' from s_1 to t_1. Given G_1' we add additional paths from s_1 to t_1 to compose G_1''. Denote the additional paths in $G_1'' \backslash G_1'$ by D_1. Likewise, we construct D_2, a set of paths added to G_2', which compose flow G_2''. Since the pair (R_1, R_2) is on the boundary of the multicommodity region, in the network $U = D_1 \cup G_2'$, if s_2 transmits G_2' at rate R_2, then s_1 cannot transmit without coding. Using the scheme in Section III-A for U, s_1 can transmit at a certain rate. Thus in G, s_1 transmits at a rate higher than R_1 and s_2 transmits at rate R_2, improving the point (R_1, R_2).

If we are not given a multicommodity flow solution, we can formulate the required conditions on G_1', D_1, G_2' and D_2. Define four commodities. The two commodities x_e^a and x_e^c (x_e^b and x_e^d) are transmitted by s_1 (s_2) and received by t_1 (t_2). The flows x_e^a and x_e^c define G_1' and D_1, respectively:

$$\sum_{e' \in \Gamma_I(e)} x_e^a = \sum_{e' \in \Gamma_O(e)} x_e^a, \quad \sum_{e' \in \Gamma_I(e)} x_e^c = \sum_{e' \in \Gamma_O(e)} x_e^c,$$
$$x_e^a + x_e^c \leq c(e) \ \forall e, \quad x_e^a \geq 0, \quad x_e^c \geq 0 \quad (6)$$

Likewise the flows x_e^b and x_e^d define G_2' and D_2, respectively:

$$\sum_{e' \in \Gamma_I(e)} x_e^b = \sum_{e' \in \Gamma_O(e)} x_e^b, \quad \sum_{e' \in \Gamma_I(e)} x_e^d = \sum_{e' \in \Gamma_O(e)} x_e^d,$$
$$x_e^b + x_e^d \leq c(e) \ \forall e, \quad x_e^b \geq 0, \quad x_e^d \geq 0 \quad (7)$$

The flows x_e^a and x_e^b constitute together the multicommodity flow, with rate pair (R_a, R_b). Therefore:

$$x_e^a + x_e^b \quad \leq \quad c(e) \ \forall e \quad (8)$$

Note that flow x_e^a is allowed to overlap with flow x_e^d. Likewise for flows x_e^b and x_e^c.

Definition 1: Denote the set $x_e^a, x_e^b, x_e^c, x_e^d, \forall e \in E$ that maintain conditions (6)-(8) as quasiflow Z.

The computational complexity of finding the quasiflow Z is similar to that of the multicommodity since the linear programming has a similar number of variables and inequalities. Given Z, the code that improves the multicommodity solution x_e^a, x_e^b can be constructed as follows. At the first stage, to improve R_1, consider x_e^a, x_e^b, x_e^c. The data transmitted on x_e^a is left uncoded. The data on x_e^b, x_e^c is coded according to Section III-A. To improve R_2, in the second stage likewise consider x_e^a, x_e^b, x_e^d.

D. Distributed Code Construction

1) Distributed Quasiflow Construction: In our distributed setting all nodes, except the sources, code and make routing decisions according to the information they receive from their neighboring nodes. There is one exception to the distributivity since the sources require a small amount of feedback from the sinks. Other than that, there is no global mechanism responsible for the communication scheme. Generally, distributed schemes incur some rate loss, but the rate loss becomes smaller if a large field size is employed and larger routing delays are allowed.

The quasiflow Z can be found by linear programming, which is not distributed. For the multicommodity problem, the backpressure algorithm in [6] is distributed. The algorithm is an approximation and has lower complexity than linear programming. We show how to modify the algorithm in [6] in order to find the quasiflow. Once Z is found, network coding can be constructed, as will be shown in Section III-D2. The algorithm finds a solution with demands d_i for commodity $i = a, b, c, d$ provided that there exists feasible quasiflow with demand $(1+2\epsilon)d_i$ per commodity i. For each source(sink), we connect a dummy source(sink) with several unit capacity edges to the source(sink), where the number of connecting edges is $R(1+2\epsilon)d_i$ and R is the number of rounds of the algorithm, defined below and upper bounded by (9). This quantity an upper bound on the flow that goes through the source, for a single operation of the algorithm. There is a regular queue for each commodity at the head and tail of each edge. The potential of a regular queue is defined as $\phi_i(q) = e^{\alpha_i q}$, where $\alpha_i = \epsilon/8ld_i$ and l is the length of the longest flow path in G. The size of the source queue for each commodity i is bounded by Q_i , where $Q_i = \Theta(\frac{ld_i ln(1/\epsilon)}{\epsilon})$. The excess of commodity i is placed in an overflow queue at each source. The potential of the overflow queue is defined as $\sigma_i(b) = b\phi_i'(Q_i) = b\alpha_i e^{\alpha_i Q_i}$. Details on these definitions can be found in [6]. The algorithm proceeds in rounds, where each unit-time round consists of the following four phases:

1) For each source s_i add $(1+\epsilon)d_i$ units of quasiflow to the overflow queue of commodity i and move as much quasiflow as possible from the overflow queue to the source queue.
2) For each edge push quasiflow across it (from the tail queue to the head queue) so as to minimize the sum of potentials of the queues in it subject to constraints (6)-(8).
3) For each commodity i empty the sink queue.
4) For each i and each node v, rebalance commodity i within v so that the head queues of the incoming edges and the tail queues of the outgoing edges for commodity i are of equal size.

The algorithm does not guarantee that each unit of quasiflow will reach its destination. However, it can be shown that the amount of undelivered quasiflow stays bounded over time, by upper bounding the size of the regular queues and overflow queues. The analysis is similar to the analysis in [6], except a quasiflow is treated instead of a flow. The full modification to quasiflow appears in [14].

Over R rounds we inject $R(1+\epsilon)d_i$ units of commodity i. If we require the undelivered flow to be at most $R\epsilon d_i$, it can be shown that it is sufficient to take

$$R = O\left(\frac{El(1+ln(1/\epsilon))}{\epsilon^2}\right) \tag{9}$$

rounds. The complexity is R times the work performed at a single round. The number of rounds R required for the convergence of the algorithm is finite and decreases when we allow a larger fraction of the quasiflow to remain undelivered.

2) Incorporating Network Codes: We interpret the network as a time slotted network [16], [17].

Definition 2: Given a network G, and a positive integer R, the time slotted network denoted by G^R, is defined as follows. It includes the nodes s_1, s_2 and all nodes of the type x^r where x is a non-source node in G and r ranges through integers 1 and R. The edges in the network belong to one of the three types listed below. For any non-source nodes x and y in G:

- For $r \leq R$ the capacity of the edge from $s_l, l = 1, 2$ to x^r, is that of the edge from s_i to x in G.
- For $r < R$ the capacity of the edge from x^r to y^{r+1} is the capacity of the edge from x to y in G.
- For $r < R$ the capacity of the edge from x^r to x^{r+1} is infinity.

In Z, for each of the flows $x_e^i, i = a, b, c, d$, a symbol sent from $s_l, l = 1, 2$ to x^r corresponds to the symbol sent on edge (s_l, x) during round r. A symbol sent from x^r to y^{r+1} corresponds to the symbol sent on edge (x, y) during round r. A symbol sent from x^r to x^{r+1} corresponds to the accumulation of a symbol in the queue of x from round r to round $r+1$. For each flow $x_e^i, \forall e \in E$ the edges in G^R that participate in the flow form a subgraph with capacity Rd_i. After the algorithm finds the quasiflow Z, we construct the network code for network G^R according to Section III-C.

For the distributivity of the algorithm, we use random coding for the internal coding stage in Section III-A1, where each node chooses the coding coefficients from a field. The total number of coding edges in the network is ER. The success probability of the internal coding is at least $P_{succ} = \left(1 - \frac{2}{|\mathcal{F}|}\right)^{ER}$ [18]. The block size is cER, where c is chosen according to the desired success probability and the block size as $O(\log(ER))$. The block size can be larger than the capacity of each edge. The time scale is changed, so that each edge can carry at least a single symbol at each round. Since the delay of the quasiflow is at least R rounds, the coding delay is of logarithmic order.

The distributed quasiflow construction in Section III-D1 does not guarantee that all of the flow will reach the sinks, and some packets may be lost. There are a number of ways to set up the coding scheme, and we present a possible way. We denote a set of R rounds as a step. In the first step quasiflow Z is determined using the algorithm in Section III-D1. The quasiflow in future steps will behave the same, since the quasiflow is constant and deterministic. The symbols that have not reached the sinks after an entire step are emptied from the queues in the network, so that they are not mixed with symbols in future steps. Since the quasiflow in G^R is constant and deterministic, after the first step it is possible to learn exactly

which packets will be lost for each R rounds. In the second step, only the flows $\{x_e^a\},\{x_e^c\}$ carry symbols. Since these flows do not overlap, no coding is necessary. The symbols are transmitted, according to the quasiflow determined in the first step. The sink t_1 receives the symbols intended for it, except the lost packets, and informs s_1 of the missing packets by feedback. The source s_1 now knows not to encode data on these packets in future steps. This procedure is repeated in the third step for $\{x_e^b\},\{x_e^d\}$ and the pair s_2-t_2. Next all the nodes draw the randomized coding coefficients and store them for future use.

In the forth step s_1 transmits a unit matrix while all the nodes perform network coding, enabling s_1 to inform each sink of the coding coefficients from s_1 to the sink [19], Section 1.5.1. In the fifth step s_2 transmits the unit matrix. Since the source has to set the coding constraints, the coding coefficients are sent from the sinks to the sources by feedback. The sources determine by a common algorithm the constraints and set them on the transmitted symbols. Once the constraints are set, each sink can decode the data intended for it. Note that if the rate requirements change, then as long as the change can be achieved by other coding constraints, no new setup is required. The sources need only to set again the new coding constraints and make them known to the sinks. In contrast, in regular multicommodity each change in the data rates requires in general a totally new setup.

E. Distributed Construction of Quasiflow for Ad-hoc Wireless Networks

In [12] a backpressure algorithm was considered for ad-hoc wireless network. We modify our network model according to [12], while we construct a quasiflow instead of a flow. Packet arrivals are i.i.d. over timeslots and $A_v^{(i)}(n)$ is the number of packets of commodity i that arrive to v during slot n. At most one packet can be transmitted from a node during a timeslot, where $\mu_v(n)\in\{0,1\}$ is the number of packets transmitted by v during slot n. Transmission opportunities are determined by a (deterministic) time division multiple access structure and $\xi_v(n)$ is 1 if node v is allowed to transmit during slot n and 0 otherwise. If the node transmits a packet at a certain time slot it can choose a commodity a,b,c,d or a pair of commodities (a,d) or (b,c) (since these pairs of commodities are allowed to occupy the same capacity unit in the quasiflow). Let $U_v^{(i)}(n)$ be the number of packets of commodity i at node v at the beginning of time slot n. Define $U_t^{(i)}(n)=0$ for any n if t is the destination of commodity i. Each packet transmission consumes power P_{tran}, and is successfully received by node u with probabilities $q_{vu}(n)$. Let $K_v(n)$ represent the set of all potential receivers for node v during slot n. The success probability of packet transmission can be correlated over various links, and the probability $q_{v,\Omega_v}(n)$, where Ω_v is a subset of nodes within $K_v(n)$, represents the probability that the set of nodes that successfully receive the packet is exactly Ω_v. The control mechanism includes ACK/NACK messages from each receiving node through a feedback channel to v [12]. For a power vector $\mathbf{p}=\{p_1,\cdots,p_{|V|}\}$ define the cost function $h(\mathbf{p})=\sum_{i=1}^{|V|}h(p_i)$ where $h(p_i)$ is a nonnegative, continuous function and $h(0)=0$. The power consumed at time slot n is $\mathbf{p}(n)=P_{tran}\cdot(\mu_1(n),\cdots,\mu_{|V|}(n))$. The nonnegative parameter V determines the degree to which power cost is emphasized. Every timslot n each node v observes the queue backlogs in each of its potential receiver nodes and the current channel probabilities associated with its receivers. If $\xi_v(n)=1$ then v performs the following:

1) Compute for commodities $i=a,b,c,d$ and receivers $u\in K_v(n)$ the differential backlog weights:
$$W_{vu}^{(i)}(n)=\max[U_v^{(i)}(n)-U_u^{(i)}n),0] \tag{10}$$
For the pairs of commodities $(i,j)=(a,d)$ and $(i,j)=(c,d)$ define the joint weights:
$$W_{vu}^{(i,j)}(n)=W_{vu}^{(i)}(n)+W_{vu}^{(j)}(n) \tag{11}$$
2) The receivers $u\in K_v(n)$ are priority ranked according to the $W_{vu}^{(i,j)}(n)$ weights. Define $u(v,i,j,n,b)$ as the node $u\in K_v(n)$ with the bth largest weight $W_{vu}^{(i,j)}(n)$ for commodity pair (i,j):
$$W_{vu(v,i,j,n,1)}^{(i)}(n)\geq W_{vu(v,i,j,n,2)}^{(i)}(n)\geq\cdots \tag{12}$$
3) Define $\phi_{v,u}^{(i,j)}(n)$ as the probability that a packet transmission from node v is correctly received by node u, but is not received by any other nodes $u\in K_v(n)$ that are ranked with higher priority than node u according to the commodity pair (i,j) rank ordering of the previous step.
4) Define the *optimal commodity pair* $(i,j)_v^*(n)$ as the commodity pair (i,j) that maximizes:
$$\sum_{b=1}^{|K_v(n)|}W_{vu}^{(i,j)}(n)\phi_{v,u}^{(i,j)}(n) \tag{13}$$
Define $W_{vu}^*(n)=\sum_{b=1}^{|K_v(n)|}W_{vu}^{(i,j)_v^*(n)}(n)\phi_{v,u}^{(i,j)_v^*(n)}(n)$ as the resulting maximum value.
5) If $W_{vu}^*(n)-Vh_v(P_{tran})>0$, choose pair (i,j). If $W_{vu}^{(i)}(n)>0$ and $W_{vu}^{(j)}(n)>0$ then in the quasiflow, both commodities i and j are transmitted by node v at slot n. If $W_{vu}^{(i)}(n)=0$ then the node transmits only commodity j. Likewise, if $W_{vu}^{(j)}(n)=0$ then the node transmits only commodity i. If $W_{vu}^*(n)-Vh_v(P_{tran})\leq 0$ node v remains idle for slot n
5) After receiving ACK/NACK feedback from the recipients nodes, v shifts responsibility of the packet to the successful receiver u with the largest positive $W_{(v,u)}^{(i,j)_v^*(n)}(n)$. If no successful receiver has positive differential backlog, node v keeps the responsibility for the packet.

In [12] optimality and stability properties of the routing algorithm are proved. Similar properties hold also for the quasiflow construction under our model and the proofs can be modified in a straightforward way. Once the quasiflow is constructed, network codes can be incorporated similarly to Section III-D2, provided that the network changes slowly, so that we can assume that if a transmission was successful at some slot and at some recipient node, then future transmissions to that receiver node will continue to be successful during several rounds. The setup scheme in Section III-D2

could be implemented. After the network changes and other recipient nodes are successful, a new setup stage will be required.

IV. Random Coding Analysis

We find non-trivial lower and upper bounds on the rates that our scheme achieves. We focus on the special case in Section III-A, in which t_2 receives its maximal rate $R_2 = h_2$. As in III-A, the construction includes three stages: internal coding, Stage A and Stage B. We assume that in the internal coding the coefficients are drawn randomly from a field $\mathcal{F}$.

A. Lower Bound

Definition 3: Rank r_{ij} is the average rate that t_j receives from s_i after the internal coding (prior to the constraints setting) when the other source is silent, where the average is over the random codes. In other words, r_{ij} is the average rank of the set $V_j^i = \{\mathbf{v}^i(e), e \in \Gamma_I(t_j)\}$ (defined in Section III-A). In short, we say that "sink t_j receives rank r_{ij} from source s_i".

We find the rank that t_j receives from $s_i, i = 1, 2$. The capacity $h_i, i = 1, 2$ from s_i to t_i can be achieved using a flow, which is denoted by f_{ii} and contains E_{ii} edges. The maximal flow of size $C_{ij}, i = 1, 2, j = 1, 2, i \neq j$ from s_i to t_j is denoted by f_{ij} and contains E_{ij} edges. Consider random coding with a single source and a single sink. The maximal flow of size h from s to t is denoted by f_{st} and contains E_{st} edges. At each edge the failure probability is at most $\frac{1}{|\mathcal{F}|}$ [15]. The rank of the set of the global coding vectors of $\Gamma_I(t)$ is reduced by at most 1 per code failure at an edge. On average sink t loses rank $\frac{1}{|\mathcal{F}|}$ per edge in f_{st}. The total average rank loss at sink t is at most $\frac{E_{st}}{|\mathcal{F}|}$. Sink t will be able on the average to reconstruct at least $h - \frac{E_{st}}{|\mathcal{F}|}$ symbols. We will choose $\mathcal{F}$ such that $|\mathcal{F}| > E_{st}$. Since r_{ij} is defined when one of the sources is silent, we can similarly find:

$$r_{11} = h_1 - \frac{E_{11}}{|\mathcal{F}|},\ r_{12} = C_{12} - \frac{E_{12}}{|\mathcal{F}|},$$
$$r_{21} = C_{21} - \frac{E_{21}}{|\mathcal{F}|},\ r_{22} = h_2 - \frac{E_{22}}{|\mathcal{F}|} \tag{14}$$

where $\epsilon_{ij} = \frac{E_{ij}}{|\mathcal{F}|}$. In Section III-A we assumed t_2 receives the maximal rate h_2. When the code is random, the rate that t_2 receives from s_2 is on average $r_{22} = h_2 - \epsilon_{22}$. The sink t_2 receives rank $r_{12} = C_{12} - \epsilon_{12}$ from source s_1. The interference noise at t_2 can be canceled in Stage A by at most r_{12} constraints on s_1, and so the rate of s_1 is reduced by at most r_{12}. We define the NOR region, in order to find how many additional constraints are set on s_1 in Stage B. For the definition we assume that the field size is infinite. We describe this region and summarize the algorithm that computes it. In Section IV-A1, we give the full analysis. We also show how a finite field size is treated.

The noise only region (NOR)- the set of edges in $\Gamma_I(t_1)$ that are forced to carry after stage A symbols that depend only on the source s_2.

- Go over the h_1 edges in $\Gamma_I(t_1)$, one after another.

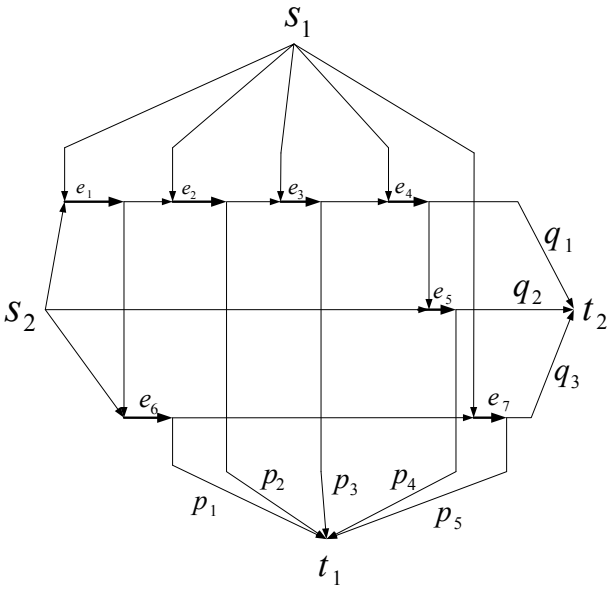

Fig. 5. Example 1

- Consider the network $G' = (V \bigcup T, E \bigcup \{e_{t_1}\} \bigcup \{e_{t_2}\})$, where $\{e_{t_i}\}, i = 1, 2$ is a set of $h_1 + h_2$ edges from sink t_i to supersink T.
- For edge $e \in \Gamma_I(t_1)$ capacity C_e is the capacity from s_1 to T in G' when all of the edges in $\Gamma_I(t_1)$, except e itself, are disconnected.
- Edge e belongs to the NOR if and only if $C_e = C_{12}$.
- Find a maximal flow f_{NOR} from s_2 to t_1 when all edges in $\Gamma_I(t_1) \backslash$NOR are disconnected.
- Define the edges in NOR$\bigcap f_{NOR}$ as the Effective Noise Only Region (ENOR).

Example 1: In Figure 5 edges with in-degree one forward the symbols they receive. The sink t_1 receives p_1, p_2, p_3, p_4, p_5 and t_2 receives q_1, q_2, q_3. The coding edges are enumerated $e_1, \ldots, e_7$. The code after the internal coding in Table I is binary. To avoid interference at t_2, the symbols q_1, q_2, q_3 are forced to be functions of the s_2 only. The constraints set on s_1 are $b_1 + b_2 + b_3 + b_4 = 0$ and $b_1 + b_5 = 0$. The resulting code is shown in Table II, where p_4, p_5 depend only on s_2, which suggests that the edges that carry them might be in the NOR. To find the NOR, we disconnect in Figure 6 all edges in $\Gamma_I(t_1)$ except the edge that carries p_5. In G' defined above, the capacity from s_1 to T is 2, the same as the capacity from s_1 to t_2 in G. Thus the edge that carries p_5 is in the NOR. Similarly, the edge that carries p_4 is in the NOR. The procedure for the edge that carries p_3 is shown in Figure 7. The capacity from s_1 to T in G' is 3 which is greater than the capacity from s_1 to t_2 in G. Thus the edge that carries p_3 is not in the NOR. Similarly, the edges that carry p_1 and p_2 are not in the NOR. If follows that the NOR consists of the edges that carry p_4 and p_5. In order to find the ENOR we disconnect in Figure 8 all edges in $\Gamma_I(t_1)$ except the edges that carry p_4 and p_5. In Figure 8, the capacity from s_2 to the NOR is 2 and therefore both edges that carry p_4 and p_5 are in the ENOR.

1) The Noise Only Region (NOR): According to Section III-A, $G = G_1 \cup G_2$. The set $\Gamma_I(t_1)$ is restricted to the h_1 incoming edges of t_1 in G_1. That is, t_1 will not use for decoding the symbols it might receive from edges that are

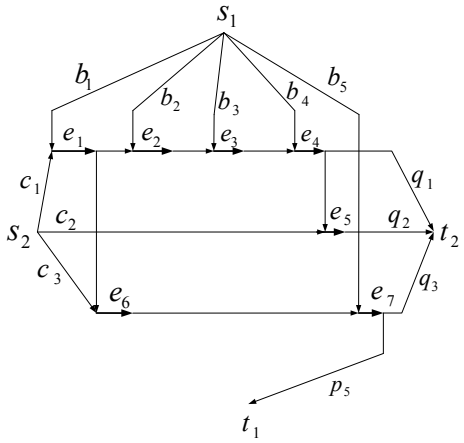

Fig. 6. Example 1 - Finding the NOR

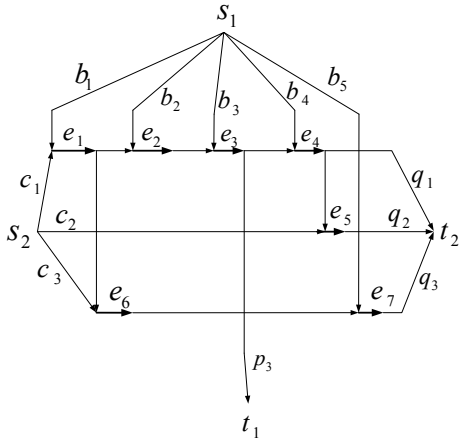

Fig. 7. Example 1 - Finding the NOR

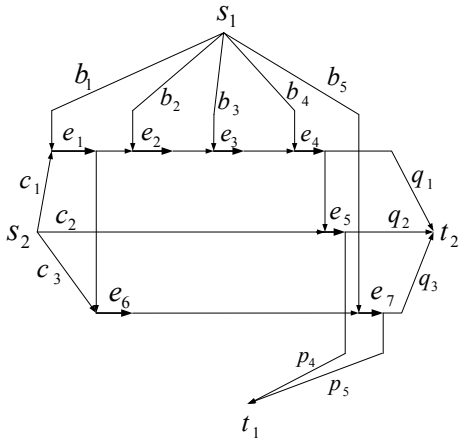

Fig. 8. Example 1 - Finding the ENOR

TABLE I
AN EXAMPLE CODE AFTER THE INTERNAL CODING

edge	symbol	code
e_1		$b_1 + c_1$
e_2	p_2	$b_1 + b_2 + c_1$
e_3	p_3	$b_1 + b_2 + b_3 + c_1$
e_4	q_1	$b_1 + b_2 + b_3 + b_4 + c_1$
e_5	$p_4 = q_2$	$b_1 + b_2 + b_3 + b_4 + c_1 + c_2$
e_6	p_1	$b_1 + c_1 + c_3$
e_7	$p_5 = q_3$	$b_1 + b_5 + c_1 + c_3$

not in G_1. Since this section deals with achievable rates this assumption is valid. Similarly, the set $\Gamma_I(t_1)$ is restricted to the h_2 incoming edges of t_2 in G_2.

Since for an infinite field size t_2 is required to receive its maximal rate, the h_2 symbols at $\Gamma_I(t_2)$ are required to be after Stage A functions of s_2 only. For an infinite field size, the Noise Only Region (NOR) is the set of edges in $\Gamma_I(t_1)$ that are forced to carry after stage A symbols that depend only on s_2. For a finite field size, the symbols at the edges in the NOR might depend also on s_1. In order to find the NOR, we go over the h_1 edges in $\Gamma_I(t_1)$, one after another. Define the capacity from s_1 to $t_1 + t_2$ as the capacity from s_1 to a supersink T in G' defined above. For the definition of G', we connect both t_1 and t_2 to T with $h_1 + h_2$ edges, since $h_1 + h_2$ is an upper bound on any rate that can be transmitted in the network. For e in $\Gamma_I(t_1)$, we find the capacity C_e from s_1 to $t_1 + t_2$ when all the edges in $\Gamma_I(t_1)$, except e, are disconnected. For the network in Figure 5 we compute C_{p_4} (where in our notation p_4 denotes both the edge and the symbol on the edge) by finding the capacity from s_1 to sink T in Figure 9, which is 2. The sinks t_1 and t_2 are each connected to T with $h_1 + h_2 = 8$ edges.

Theorem 2: Edge e belongs to the NOR if and only if $C_e = C_{12}$.

Proof: Assume s_2 is silent. For an infinite field size $\mathcal{F}$, prior to stage A, t_2 receives rank C_{12} from s_1. If $C_e = C_{12}$ the symbol on e is linearly dependent on the symbols received by t_2. Stage A forces the symbols on $\Gamma_I(t_2)$ to be canceled. It follows that the symbol at e will also be canceled. If s_2 is not silent, due to the linearity of the code, the symbol on e will depend only on s_2. According to definition e belongs to the NOR. Similarly, if $C_e = C_{12} + 1$, then e is not in the NOR. ■

In Figure 9 since $C_{p_4} = C_{12} = 2$ edge e is in the NOR. In the average case, because of the finite field, the rank received by t_2 from s_1 is $C_{12} - \epsilon_{12}$. Consider the capacity from s_1 to $t_1 + t_2$, when all of the edges in $\Gamma_I(t_1)$ that are not in the NOR (defined for an infinite field size) are disconnected. That capacity is C_{12}, otherwise the condition on the NOR will be violated. Therefore, when all of the edges in $\Gamma_I(t_1)$ that are not in the NOR are disconnected, after setting the $C_{12} - \epsilon_{12}$ constraints on s_1 in Stage A, t_1 receives in the average case rank at most ϵ_{12} from s_1. Setting additional ϵ_{12} constraints on s_1 in Stage B ensures that the symbols on the edges in NOR do not depend on s_1.

Denote the number of edges in the NOR as C_{nor}. The algorithm finds a maximal flow f_{NOR} from s_2 to t_1 when all of the edges in $\Gamma_I(t_1)$ that are not in the NOR are disconnected. The size of f_{NOR} is denoted by C_{enor}. The set of edges in the NOR that also participate in f_{NOR} is defined as the Effective Noise Only Region (ENOR). For an infinite field size, the symbols that are in the NOR but not in the ENOR are disregarded since they are linearly dependent on the ENOR. The symbols in the ENOR will be used in order to cancel the interference noise on the other edges in $\Gamma_I(t_1)$.

For a finite field size, when all the edges in $\Gamma_I(t_1)$ that are not in the NOR are disconnected, t_1 receives from s_2 on average rank $C_{enor} - \epsilon_{enor}$, where $\epsilon_{enor} = \frac{E_{enor}}{|\mathcal{F}|}$ and E_{enor} is the number of edges in f_{NOR}. The sink t_1 will be able to use the $C_{enor} - \epsilon_{enor}$ independent symbols received at the

TABLE II
AN EXAMPLE CODE AFTER CONSTRAINTS SETTING

edge	symbol	code
e_1		$b_1 + c_1$
e_2	p_2	$b_1 + b_2 + c_1$
e_3	p_3	$b_1 + b_2 + b_3 + c_1$
e_4	q_1	c_1
e_5	$p_4 = q_2$	$c_1 + c_2$
e_6	p_1	$b_1 + c_1 + c_3$
e_7	$p_5 = q_3$	$c_1 + c_3$

ENOR from s_2 in order to cancel the interference at the other edges in $\Gamma(t_1)$. The edges that are in the NOR but are not in the ENOR might contribute at most ϵ_{enor} additional linearly independent symbols of source s_2. Otherwise, the capacity condition on C_{enor} will be violated. For a large field $\mathcal{F}$ this contribution is negligible and therefore the sink t_1 disregards these edges for decoding.

As for computational complexity, finding the NOR requires h_1 capacity computations and the ENOR requires a single capacity computation. Therefore, h_1+1 capacity computations are required.

2) A Lower Bound on the Achievable Rate:

Theorem 3: When s_2 transmits rate $R_2 = h_2$, a lower bound on average achievable rate R_1 of s_1:

$$R_1 \geq h_1 - C_{12} - C_{21} + C_{enor} - \epsilon_{11} - \epsilon_{enor} + \epsilon_{12} + \epsilon_{21} \quad (15)$$

Proof: Before the constraints setting, the average rank r_{12} that t_2 receives from s_1 is given by $C_{12} - \epsilon_{12}$. Therefore, in the average case after setting at most $C_{12} - \epsilon_{12}$ constraints on s_1, sink t_2 will be able to decode s_2 at rate $h_2 - \epsilon_{22}$. Sink t_1 receives from s_2 before the constraint settings rank $C_{21} - \epsilon_{21}$. After setting the $C_{12} - \epsilon_{12}$ constraints, sink t_1 can decode $C_{enor} - \epsilon_{enor}$ symbols of source s_2, according to the construction of the ENOR. Therefore, setting at most $(C_{21} - \epsilon_{21}) - (C_{enor} - \epsilon_{enor}) = C_{21} - C_{enor} - \epsilon_{21} + \epsilon_{enor}$ additional constraints on s_1 would enable t_1 to cancel the interference. Thus the total number of constraints set on s_1 is $C_{12} + C_{21} - C_{enor} - \epsilon_{12} - \epsilon_{21} + \epsilon_{enor}$, which is the rate loss of s_1 relative to the achievable rate $h_1 - \epsilon_{11}$ for random coding when s_2 is silent. ■

B. An Upper Bound on the Achievable Rate

We derive an upper bound on the achievable rate region of our scheme in the average case.

Theorem 4: With our code construction, sink t_1 can decode s_1 at rate at most:

$$R_1 \leq h_1 - C_{21} + \epsilon_{21} \quad (16)$$

Proof: We can model the total system seen by t_1, of the network combined with the random coding, as the vector channel of dimension h, $\mathbf{y} = A\mathbf{x}+\mathbf{z}$. The vector $\mathbf{y}$ is the vector received by t_1, A is the transfer matrix between s_1 to t_1, $\mathbf{x}$ is the vector of s_1, and $\mathbf{z}$ is the interference noise. In the average case t_1 receives from s_2 after the internal coding, before the constraints setting, rank $C_{21} - \epsilon_{21}$. Therefore, the entropy of $\mathbf{z}$ is on average $H(Z) = C_{21} - \epsilon_{21}$. The entropy of $\mathbf{y}$ is at most $H(Y) \leq h_1$, because of the capacity constraints. Thus, the mutual information between the s_1 and the received symbols at t_1 is $I(X;Y) = H(Y) - H(Y|X) = H(Y) - H(Z) \leq h_1 - C_{21} + \epsilon_{21}$. ■

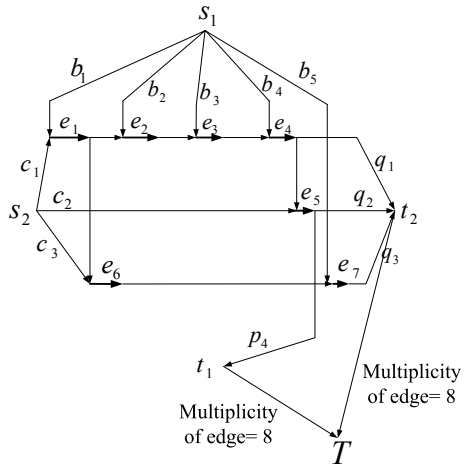

Fig. 9. Definition of C_{p_4}

The gap between the lower bound in (15) and the upper bound in (16) is $C_{12} - C_{enor} + \epsilon'$, where ϵ' can be made arbitrarily small by enlarging the field size. The gap is nonnegative since according to construction $C_{12} \geq C_{enor}$ (in fact, $C_{12} \geq C_{nor} \geq C_{enor}$). From definition, it follows that $C_{21} \geq C_{enor}$. For an infinite field size, the gap vanishes when $C_{12} = C_{enor}$, and it follows that $C_{21} \geq C_{12}$. After Stage A, t_1 can reconstruct the interference from s_2 at rate $C_{enor} = C_{12}$. The number of constraints setting in Stage A is at most C_{12}. In Stage B, additional $C_{21} - C_{12}$ will suffice in order for t_1 to reconstruct all the interference noise. Therefore the rate of s_1 after Stage A and Stage B is equal to $R_1 = h_1 - C_{12} - (C_{21} - C_{12}) = h_1 - C_{21}$, which is the upper bound in (16) for an infinite field.

Since t_2 receives from s_2 at most rate $R_2 = h_2$, the upper bound on the sum rate is:

$$R_1 + R_2 \leq h_1 + h_2 - C_{21} + \epsilon_{21} \quad (17)$$

for $\epsilon_{21} \geq 0$, which can be made arbitrarily small by enlarging the field size. Denote the upper bound for infinite field by C_{ub}, where $C_{ub} = h_1 + h_2 - C_{21}$. We show that the bound (17) is not trivial.

Theorem 5: The capacity of the minimal cut separating the sources s_1, s_2 from the sinks t_1, t_2, is denoted as C_{mc}. The rate C_{ub} is not larger than C_{mc} and in general is strictly smaller than C_{mc}.

Proof: Suppose that the opposite is true. That is, for some $\delta > C_{21}$:

$$C_{mc} = h_1 + h_2 - \delta \quad (18)$$

Consider a certain minimal cut that separates the sources (s_1, s_2) from the sinks (t_1, t_2). In Figure 10 thicker arrows depict the minimal cut that contains seven edges: $e_2, e_3, e_4, e_5, e_6, e_7, e_8$. For each of the h_1 paths in any maximal flow from s_1 to t_1 there is at least a single edge in the minimal cut and similarly for the h_2 paths from s_2 to t_2. For each of the h_2 paths from s_2 to t_2 we choose the first edge that is in the minimal cut, to form the set E_{22}. Thicker arrows in Figure 11 depict the edges in E_{22}: e_2, e_5, e_6, e_8. The set E_{22} contains edges in Δ different paths in the flow from s_1

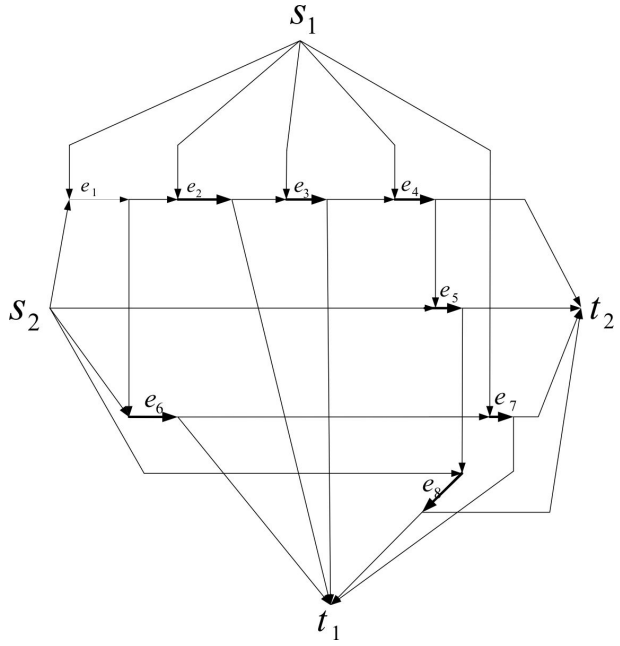

Fig. 10. An Upper Bound on the Achievable Rate (a)

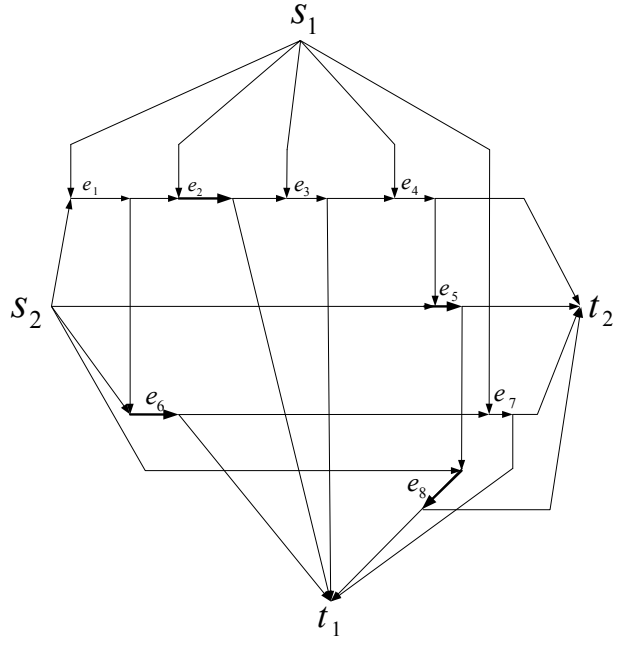

Fig. 11. An Upper Bound on the Achievable Rate (b)

to t_1, where $\Delta \geq \delta$. This is because for each of the other $h_1 - \Delta$ paths from s_1 to t_1 there is at least a single edge in the minimal cut. So together with the h_2 edges in E_{22}, the number of edges in the minimal cut is at least $h_1 + h_2 - \Delta$. So if $\Delta < \delta$ then (18) is violated, since we have found a minimal cut which is smaller than C_{mc}. Consider the Δ paths from s_1 to t_1 that contain edges in E_{22}. Denote them as $p_1, \cdots, p_\Delta$. If a certain path p_i has several edges in E_{22}, we choose the last one in the path p_i and denote it as edge E_i. The path from s_2 to E_i (not including E_i) is denoted as l_i. The part of the path p_i, which is from E_i to t_1 (not including E_i), is denoted as q_i. See Figure 12, where $\Delta = 3$.

Suppose that a certain path q_i intersects a certain path $l_j, j \neq i$. Denote the intersection edge as $e_{i,j}$. According to

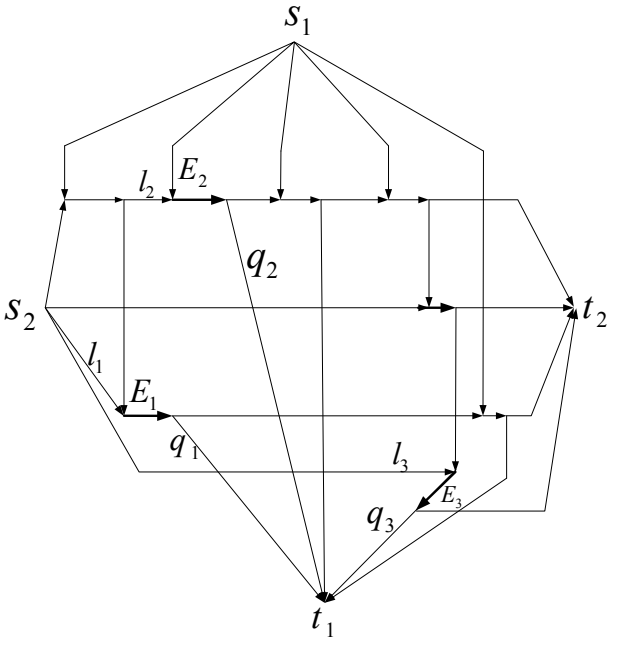

Fig. 12. An Upper Bound on the Achievable Rate (c)

construction, path q_i does not contain an edge in E_{22}, since E_i was chosen such that it is the last edge in p_i which is in the set E_{22}. Since we have initially chosen the first edge in the minimal cut for each path from s_2 to t_2 to be in the set E_{22}, it follows that l_j does not contain an edge in the minimal cut. Therefore, if q_i does not have an edge in the minimal cut, there is a path from s_2 to $e_{i,j}$ (path l_j) and from $e_{i,j}$ to t_1 (path q_i) which does not contain any edge in the minimal cut. This contradicts the fact that the minimal cut we have initially chosen is indeed a cut. It follows that q_i has an edge that is in the minimal cut, but according to the construction not in E_{22}.

Consider all the paths $\{q_i\}$ that do not intersect any path $l_j, j \neq i$. Suppose that there are such Λ paths out of Δ possible paths. Then it follows that there is a flow of size Λ from s_2 to t_1, where the kth path in the flow contains path l_k followed by edge E_k, followed by path q_k. It is required that the size of this flow will be smaller than the capacity, that is $\Lambda \leq C_{21}$. As follows from the argument above, each of the remaining $\Delta - \Lambda$ paths $\{q_i\}$ contains an edge that is in the minimal cut, but not in E_{22}. In addition to these $\Delta - \Lambda$ edges in the minimal cut (that are not in E_{22}) there are h_2 edges in the minimal cut (in E_{22}). In the remaining $h_1 - \Delta$ paths from s_1 to t_1 there is at least a single edge in the minimal cut for each path. It follows that the size of the minimal cut is at least $h_2 + \Delta - \Lambda + h_1 - \Delta = h_1 + h_2 - \Lambda$. Since $\Lambda \leq C_{21} < \delta$, this contradicts (18).

We have shown so far that C_{ub} is not larger than C_{mc}. For the network in Figure 10 the minimal cut is of size $C_{mc} = 7$. Since $h_1 = 5$,$h_2 = 4$ and $C_{21} = 3$ it follows that $C_{ub} = 6$. Therefore, the bound C_{ub} is strictly smaller than C_{mc} for some networks. ■

Note that the proof of Theorem 4 applies to any network code with random coding coefficients, not just to our code construction. Also note that Theorem 5 applies to any network, not necessarily of structure (1). To see this, denote the original network as G. Construct for G the subgraph G', which consists

of the union of flows, as given by the RHS of (1). Since G' is of structure (1), Theorem 5 applies to G', where the values h'_1,h'_2, C'_{21}, C'_{ub}, C'_{mc} are defined for G'. Returning to the original network G with values h_1,h_2, C_{21}, C_{ub},C_{mc}, according to construction $h_1 = h'_1$,$h_2 = h'_2$. Since G' is a subgraph of G, it follows that $C'_{21} \leq C_{21}$ and $C'_{mc} \leq C_{mc}$. According to Theorem 5, we have $C'_{ub} \leq C'_{mc}$, where $C'_{ub} = h'_1 + h'_2 - C'_{21}$. For the original network, we have $C_{ub} = h_1 + h_2 - C_{21} = h'_1 + h'_2 - C_{21} \leq h'_1 + h'_2 - C'_{21} \leq C'_{mc} \leq C_{mc}$, where the second inequality follows from Theorem 5. It follows that $C_{ub} \leq C_{mc}$ and Theorem 5 applies to G as well. For G, a sufficient condition for the upper bound C_{ub} to be strictly smaller than C_{mc} is $C'_{21} < C_{21}$ or $C'_{mc} < C_{mc}$.

V. Summary and Future Research

The question of constructing efficient network codes for multiple sources and finding the capacity regions is still largely open. This is one of the most significant and challenging areas of network coding. Our paper focused on interference networks, the case of two sources and two sinks, where each source is required to be reconstructed at a single sink. For this problem, the computational complexity of most previous schemes is high in comparison to routing without coding, which is the multicommodity flow. We found a network code which improves the rate region of multicommodity flows. The code construction has non-distributed and distributed versions. We have shown how the distributed algorithm can be implemented for ad-hoc wireless networks. For both the distributed and the non-distributed cases, the computational complexity of our algorithm is similar to that of multicommodity. We analyzed the performance of these codes, using techniques of random coding. We found non-trivial lower and upper bounds on the rates of our scheme, where the upper bound is generally below the min-cut bound. It would be interesting to generalize our results to a general number of users. Another challenging problem is to find a scheme to other network models, for example fast time-variant networks. There are practical issues to consider, such as synchronization.

References

[1] A. Rasala-Lehman and E. Lehman. Complexity classification of network information flow problems. *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 142–150, New Orleans, Louisiana, January 2004.

[2] R. Dougherty, C. Freiling, and K. Zeger. Insufficiency of linear coding in network information flow. *IEEE Tran. Inform. Theory*, 51(8):2745–2759, August 2005.

[3] R. Koetter and M. Médard. An algebraic approach to network coding,. *IEEE/ACM Transactions on Networking*, 11(5):782–795, October 2003.

[4] R. W. Yeung. *A First Course in Information Theory*. Kluwer Academic/Plenum Publishers, March 2002.

[5] L. Song, R.W. Yeung, and N. Cai. Zero-error network coding for acyclic networks. *IEEE Tran. Inform. Theory*, 49(12):3129 – 3139, December 2003.

[6] B. Awerbuch and T. Leighton. Improved approximation algorithms for the multicommodity flow problem and local competitive routing in dynamic networks. *Proceedings of the 26th ACM Symposium on Theory of Computing*, pages 487–496, May 1994.

[7] D. Traskov, N. Ratnakar, D. S. Lun, R. Koetter, and M. Médard. Network coding for multiple unicasts: An approach based on linear optimization. *IEEE International Symposium on Information Theory*, pages 1758–1762, Seattle, WA, July 2006.

[8] T. C. Ho, Y-H Chang, and K. J. Han. On constructive network coding for multiple unicasts. *44th Allerton Conference on Communication, Control, and Comupting*, Monticello, IL, September 2006.

[9] A. Eryilmaz and D.S. Lun. Control for inter-session network coding. Technical report, Proceedings of the Workshop on Network Coding, Theory and Applications (NetCod), January 2007.

[10] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft. Xors in the air: practical wireless network coding. *SIGCOMM Comput. Commun. Rev.*, 36(4):243–254, 2006.

[11] J. K. Sundararajan, M. Médard, R. Koetter, and E. Erez. A systematic approach to network coding problems using conflict graphs. *UCSD Information Theory and Applications Inaugural Workshop*, San Diego, CA, February 2006.

[12] M. J. Neely. Optimal backpressure routing for wireless networks with multi-receiver diversity. *Conference on Information Sciences and Systems (CISS)*, March 2006.

[13] E. Erez and M. Feder. Code construction for two source interference networks. *Third Workshop on Network Coding, Theory, and Applications*, San Diego, CA, January 2007.

[14] E. Erez. *Topics in Network Coding*. PhD thesis, Tel Aviv University, 2007.

[15] 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 Tran. Inform. Theory*, 51(6):1973– 1982, June 2005.

[16] R. Ahlswede, N. Cai, S.-Y. R. Li, and R.W. Yeung. Network information flow. *IEEE Transactions on Informmation Theory*, 46(4):1204–1216, July 2000.

[17] S-Y. R. Li, R. W. Yeung, and N. Cai. Linear network coding. *IEEE Tran. Inform. Theory*, 49(2):371–381, Febuary 2003.

[18] T. Ho, M. Médard, J. Shi, M. Effros, and D. Karger. On randomized network coding. *41st Allerton Conference on Communication, Control, and Comupting*, Monticello, IL, October 2003.

[19] E. Erez and M. Feder. On codes for network multicast. *41st Allerton Conference on Communication, Control and Computing*, Monticello, IL, October 2003.

Elona Erez Elona Erez received the B.Sc. (summa cum laude), the M.Sc. (summa cum laude) and the Ph.D. degrees from Tel Aviv University, all in Electrical Engineering in 1999, 2002 and 2007, respectively. She is currently a postdoctoral associate at Yale University, at the Department of Electrical Engineering. During 2007-2008 she was a postdoctoral scholar at California Institute of Technology (Caltech), at the Department of Electrical Engineering. She received Weinstein Prize for an outstanding student in signal processing in 2003 and 2005 and Weinstein Prize for an outstanding publication in signal processing in 2002 and 2003. She received Colton Prize for an outstanding student in 2003-2006. During the summer of 2005 she was a visiting researcher at the Laboratory for Information and Decision Systems (LIDS), Massachusetts Institute of Technology (MIT). Her research interests are in the fields of information theory and data networks, with special interest in network coding.

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

After being a research associate and lecturer in MIT he joined in 1989 the Department of Electrical Engineering - Systems, Tel-Aviv University, where he is now a Professor. He had visiting appointments at the Woods Hole Oceanographic Institution, Scripps Institute, Bell laboratories and in 1995/1996 he has been a visiting professor at MIT. He is also extensively involved in the high-tech industry and co-founded several companies including Peach Networks, a developer of a unique server-based interactive TV solution which was acquired on March 2000 by Microsoft, and Amimon a leading provider of ASICs for wireless high-definition A/V connectivity at the home.

Prof. Feder is a co-recipient of the 1993 IEEE Information Theory Best Paper Award. He also received the 1978 "creative thinking" award of the Israeli Defense Forces, the 1994 Tel-Aviv University prize for Excellent Young Scientists, the 1995 Research Prize of the Israeli Electronic Industry, and the research prize in applied electronics of the Ex-Serviceman Association, London, awarded by Ben-Gurion University. Between June 1993-June 1996 he served as an Associate Editor for Source Coding of the IEEE Transactions on Information Theory.