Code Construction for Two-Source Interference Networks

Elona Erez and Meir Feder
Dept. of Electrical Engineering-Systems, Tel Aviv University, Tel Aviv, 69978, Israel, E-mail:{elona, meir}@eng.tau.ac.il

I. Introduction

Most literature on network coding focuses on the multicast case where performance bounds and codes that achieve these bounds were found. The general case seems to be much more complicated to analyze. Koetter and Médard [1] gave an algebraic formulation of linear network codes in the general communication problem with multiple sources. However, the complexity of finding the coding coefficients of the code is NP-complete. Furthermore, their procedure finds the solvability of a given set of rates required at certain sinks, but it does not provide, in general, the optimal rate region. Yeung [2, Chapter 15] gave information-theoretic inner and outer bounds to the rate region in the general case of acyclic networks with multiple sources. The inner and outer bounds in [2] are not explicit as in the multicast case, and are exponential in the graph size. This result is extended in [3] for the case of zero-error network codes. A different framework for formulating the general network coding problem uses a graph theoretic model [4]. In this model, the conflict hypergraph of the network is defined and it was shown that a network coding is solvable over a given alphabet and a class of codes if the conflict hypergraph has a stable set. Unfortunately, the size of the conflict graph might grow exponentially with the field size, since the number of possible codes grows exponentially, and also, in general, the problem of finding the stable set might be difficult. A constructive mulit-source network coding is presented in [5]. The construction is based on the observation that random mixing is asymptotically optimal for a pollution-free multi-source network coding problem, where no receiver can be reached by a source it does not need. Here too, the computational cost might be prohibitively high.

In the problem of multiple unicast for d users transmitter s_i has to transmit information to terminal t_i at rate h_i. The d sources are simultaneous and independent. When no codes are employed in the network, then the data can be treated as flow. The problem of multiple sources and multiple sinks in the network is termed in network optimization as the multicommodity flow problem. The multicommodity flow can be found using linear programming. In [6] it was shown how the multicommodity problem can be implemented distributively.

In [7], which considers the multiple unicast problem, the operations of the network codes are restricted to binary XOR. The problem is formulated as a linear program or an integer program. This constitutes a suboptimal, yet practical scheme to construct network codes, that improves the multicommodity solution for some particular settings. The computational complexity, however, is high in comparison to multicommodity. This approach was solved in a decentralized way in [8],[9].

In this paper we present a different approach based on modifying the multicommodity solution and thus improving, in certain settings, the achievable rate region. Specifically, we focus on the interference network for the unicast problem with two users (s_1, t_1) and (s_2, t_2). We start by a special case, where one of the sources, say s_2, transmits information at its maximal min-cut max-flow rate and find a possible rate for s_1. We show that this rate of s_1 is better than the best multicommodity flow rate. Building on this special case, we generalize it as follows. Suppose we are given a certain point on the border of the rate region of the multicommodity flow (R_1, R_2). That is, when s_2 transmits at rate R_2, source s_1 cannot transmit at a rate higher than R_1 using multicommodity. We show how our method, that uses network codes, can improve R_1. We formulate our method as a linear programming problem that is closely related to the flow problem. It is often desirable for network algorithms to be localized, such that the topology need not be globally known to the designer. The similarity to the flow problem allows our method to be implemented distributively, analogously to the distributive multicommodity algorithm in [6]. For both the non-distributive case and the distributive case, the computational complexity of our algorithms for network coding are comparable to those of the parallel multicommodity problems.

II. Code Construction for Special Case

In this section we present a code construction algorithm for the special case where one of the sources transmits at its maximal rate, and the other source tries to simultaneously transmit data at a certain, as high as possible, rate.

Consider a directed acyclic, unit capacity network $G = (V, E)$ where parallel edges are allowed. There are two sources s_1 and s_2 and two sinks t_1 and t_2. Source s_1 is required to deliver data to t_1 and s_2 is required to deliver data to t_2. For edge e, denote as $\Gamma_I(e)$ the set of edges entering e and $\Gamma_O(e)$ the set of edges outgoing from e. The coding coefficient between edge e and edge e' is denoted as $m(e, e')$. That is, the symbol $y(e)$ on edge e is given by:

$$y(e) = \sum_{e' \in \Gamma_I(e)} m(e, e') y(e') \tag{1}$$

In the first stage of our code construction we perform an algorithm similar to the polynomial time algorithm in the multicast case [10]. The algorithm starts by finding the maximal flow G_1 of rate h_1 from s_1 to t_1 and the maximal flow G_2 of rate h_2 from s_2 to t_2. In the sequel we will be interested only in the subgraph $G_1 \cup G_2$, so we can assume that our original network is $G = G_1 \cup G_2$. The algorithm steps through the edges in G in topological order. For each edge e a coding vector of dimension $h_1 + h_2$ is assigned, where the first h_1 coordinates are associated with s_1 and the last h_2 coordinates are associated with s_2. We restrict ourselves to edges that participate both in G_1 and G_2. For edge e that participates in either G_1 or G_2, but not both, we assign $m(e', e) = 1$ for the edge e' that preceded e in the flow. For edge e that participates in both G_1 and G_2, the algorithm determines the coefficients $m(e_1, e), m(e_2, e)$ where e_1 precedes e in the flow G_1 and e_2 in G_2.

The invariant maintained for the network is that for sink t_1 there is a set of h_1 edges $\mathcal{C}_1$ such that the global coding vectors $V_1 = \{\mathbf{v}(e) : e \in \mathcal{C}_1\}$ have the property that their first h_1 coordinates span $F[D]^{h_1}$. Likewise, for sink t_2 there is a set of h_2 edges $\mathcal{C}_2$ such that the global coding vectors $V_2 = \{\mathbf{v}(e) : e \in \mathcal{C}_2\}$ have the property that their last h_2 coordinates span $F[D]^{h_2}$. The set $\mathcal{C}_l, l = \{1, 2\}$ contains one edge from each path in $G_l, l = \{1, 2\}$, the edge whose global coding vector was defined most recently. The coding coefficient $m(e', e)$ is drawn from a certain algebraic field (or a ring). Similarly to the proof in [10], for field size two or larger, we are ensured that at least a single $m(e', e)$ maintains the invariant. At the end of the construction the edges in $\mathcal{C}_1$ are incoming into t_1 and the edges in $\mathcal{C}_2$ are incoming into t_2.

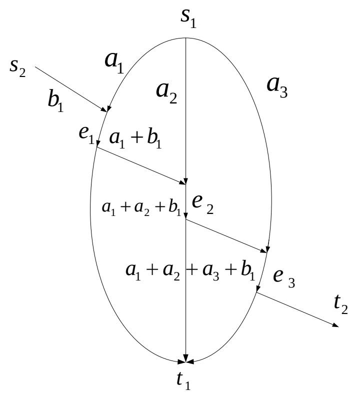

Fig. 1. Example Code

An example of the resulting code is given in Figure 1, where in the graph G only the coding edges e_1, e_2, e_3 are shown. The code is over the binary field. As seen from the example, each of the sources tries to transmit at its maximal rate. If source s_2 is silent, then sink t_1 is able to reconstruct its intended information, since when $b_1 = 0$ the edges incoming into t_1 carry the symbols $a_1, a_1+a_2, a_1+a_2+a_3$. Likewise, if source s_1 is silent, then sink t_2 is able to reconstruct its intended information, since when $a_1 = a_2 = a_3 = 0$ the edge incoming into t_1 carries the symbol b_1. However, when both sources transmit information, then for each sink there is interference noise from the source intended to the other sink.

Note that similarly to the multicast case [11], by using a sufficiently large field size, the coding coefficients can be chosen in a randomized, distributive manner.

At this point it is natural to examine how the interference noise can be canceled. We are targeting at finding a possible rate R_1 of s_1, conditioned that the rate of s_2 is $R_2 = h_2$. For the example in Figure 1 we notice that when $R_2 = h_2 = 1$, the largest possible R_1 using multicommodity flow without coding is zero. So the natural question is whether this "bottleneck effect" of the pair $s_2 - t_2$ can be relieved by coding.

At the set of the h_2 coding vectors V_2, currently on the edges entering into t_2, there is interference noise from s_1. The sink t_2 is required to receive the maximal rate $R_2 = h_2$, and therefore it is not allowed to reduce its rate. However, it is allowed to reduce the rate of s_1. The sink t_2 will be able to decode the information intended to it if all the interference noise from s_1 is canceled. Consider the subset of the first h_1 coordinates of each of the vectors in V_2. Denote as $V_2' = \{\mathbf{v}'(e) : e \in \mathcal{C}_2\}$ the corresponding set of vectors of dimension h_1. The dimension R_{12} of the vector space spanned by V_2'

$$R_{12} \leq \min\{h_1, C_{12}, h_2\} \tag{2}$$

where C_{12} is the capacity from source s_1 to sink t_2. We can choose a subset of V_2' of size R_{12} that is the basis of V_2'. Denote the basis of V_2' as:

$$B_2' = \{\mathbf{v}'(e_1), \ldots, \mathbf{v}'(e_{R_{12}})\} \tag{3}$$

Denote the edges whose vectors are in B_2' as $\mathcal{C}_2'$.

In the example network in Figure 1, we have $R_{12} \leq h_2 = 1$. Since the symbol received by t_2 is $a_1+a_2+a_3+b_1$, the vector in V_2 is:

$$V_2 = \{\mathbf{v}(e_3)\} = \{(1, 1, 1, 1)^T\} \tag{4}$$

The vector in V_2' is in fact the first three coordinates of the vector in V_2

$$V_2' = \{\mathbf{v}'(e_3)\} = \{(1, 1, 1)^T\} \tag{5}$$

and so the basis of V_2' is,

$$B_2' = \{\mathbf{v}'(e_3)\} = \{(1, 1, 1)^T\} \tag{6}$$

and $\mathcal{C}_2' = \{e_3\}$.

In order for t_2 to achieve rate $h_2 = 1$ and reconstruct b_1 we force the symbol at e_3 to depend only on b_1. This is possible by setting the constraint $a_1 + a_2 + a_3 = 0$.

In general, the cancelation of the interference is possible by forcing the symbols transmitted on the edges in $\mathcal{C}_2'$ to be functions of the source s_2 only. By canceling the interference noise from s_1 on $\mathcal{C}_2'$, the noise on the other edges at $\mathcal{C}_2$ will also be canceled. This is because the noise on the other edges are linear combinations of the noise on $\mathcal{C}_2'$. Each edge in $\mathcal{C}_2'$ would add at most a constraint on the symbols transmitted by s_1, in order to ensure that the resulting symbol on the edge is "clean" of source s_1. Each such constraint reduces the rate of s_1 by a single bit. Therefore rate $h_1' \geq h_1 - \min\{h_2, C_{12}, h_1\}$

would be transmitted from s_1. For the case $h_1 > h_2$, rate h'_1 would be strictly positive. The sink t_2 has to inform s_1 by feedback the required constraints on the source of s_1, such that t_2 can reconstruct the data of s_2.

Similarly, define for the sink t_1 the set of the coding vectors V_1 of the edges in $\mathcal{C}_1$, which at this stage are entering t_1. Specifically, consider the subset of the last h_2 coordinates of each of these vectors. Denote as $V'_1 = \{\mathbf{v}'(e) : e \in \mathcal{C}_1\}$ the corresponding set of vectors of dimension h_2. The dimension R_{21} of the vector space spanned by V'_1

$$R_{21} \leq \min\{h_2, C_{21}, h_1\} \tag{7}$$

where C_{21} is the capacity from source s_2 to sink t_1. Therefore we can choose a subset of V'_1 of size R_{21} that is the basis of V'_1. Denote the basis of V'_1 as:

$$B'_1 = \{\mathbf{v}'(e_1), \ldots, \mathbf{v}'(e_{R_{21}})\} \tag{8}$$

Denote the nodes whose vectors are in B'_1 as $\mathcal{C}'_1$.

In the example network in Figure 1, we have $R_{21} \leq h_2 = 1$. The dimension of V'_1 is 1 and the vectors in V'_1 are in fact the last coordinate of the vectors in V_1:

$$V_1 = \{\mathbf{v}(e_1), \mathbf{v}(e_2), \mathbf{v}(e_3)\} = \{ \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \} \tag{9}$$

$$V'_1 = \{\mathbf{v}'(e_1), \mathbf{v}'(e_2), \mathbf{v}'(e_3)\} = \{(1), (1), (1)\} \tag{10}$$

and so we can choose $\mathcal{C}'_1 = \{e_3\}$ and

$$B'_1 = \{\mathbf{v}'(e_3)\} = \{(1)\}. \tag{11}$$

In the general case, sink t_1 finds which additional constraints to set on the source s_1 such that the symbols on the edges in $\mathcal{C}'_1$ are functions of the source s_2 only. Sink t_1 informs source s_1 of these additional constraints. Since the symbols in $\mathcal{C}'_1$ are forced to be functions of s_2 only, sink t_1 knows the projection of source s_2 on the symbols on $\mathcal{C}'_1$. The noise on the other edges entering t_1 are linear combinations of the symbols on $\mathcal{C}'_1$. Thus sink t_1 could use the symbols on $\mathcal{C}'_1$ in order to cancel the interference noise, and would be able to reconstruct the information intended to it from s_1. Each node in $\mathcal{C}'_1$ adds at most a single constraint on the symbols transmitted by s_1, in order to ensure that the resulting symbol on the node is "clean" of source s_1. Each such constraint reduces the rate of s_1 by a single bit. Since the rate was already reduced to at least $h'_1 \geq h_1 - \min\{h_2, C_{12}, h_1\}$, the final rate of t_1 is at least $h'_1 \geq h_1 - \min\{h_2, C_{12}, h_1\} - \min\{h_2, C_{21}, h_1\} \geq h_1 - 2h_2$. For the case $h_1 > 2h_2$, rate h'_1 would be strictly positive.

Returning to the example in Figure 1, we note that since we already have the constraint $a_1 + a_2 + a_3 = 0$, t_1 is already able to decode b_1, which is the interference noise of sink t_1. If this is not the case, it would be necessary to set an additional constraint on the source at s_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 $h'_1 \geq h_1 - \min\{h_2, C_{12}, h_1\} - \min\{h_2, C_{21}, h_1\} \geq h_1 - 2h_2 = 1$. The final code is shown in Figure 2.

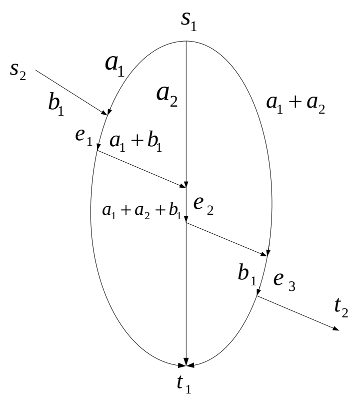

Fig. 2. Final Code

The rate region for the network in Figure 1 is shown in Figure 3 for both the multicommodity case, and for the coding scheme described above. Note that for this example this scheme is optimal since it achieves the min-cut bound, which is 3. Also 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 for networks using this coding scheme.

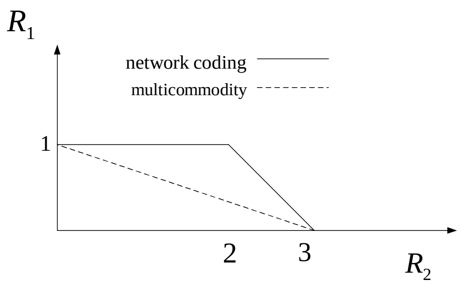

Fig. 3. Rate Region for Example Network

The coding scheme we have suggested guarantees that if s_1 is willing to reduce its rate h_1 by a certain amount of 2Δ bits, then s_2 will be able to transmit information of at least Δ bits, as long as the min-cut condition is not violated. This is because each increase in the bit rate of t_2 forces at most two additional constraints on s_1: the first to cancel the noise at t_1 and the second at t_2. This indicates that the angle of the slope of the boundary of the rate region will be between $22.5^0 < \Theta < 67.5^0$ (2:1 ratio to 1:2 ratio). In the example network in Figure 1, the slope is 45^o, which is 1:1 ratio.

Unfortunately, the 1:1 tradeoff is not always possible, even with network codes, as the following example shows. Consider Figure 4. A possible code is given, prior to the constraints setting. In this network $h_1 = 3$. If a ratio 1:1 is possible, then we expect the rate pair $h_1 = 1, h_2 = 2$ to be achievable. It is necessary to set 2 constraints on s_1, in order for t_2 to be able to receive both b_1 and b_2. In order to cancel the interference noise at t_2, the constraints sets are $a_1 + a_2 = 0$ and $a_1 + a_2 + a_3 = 0$. That is $a_1 + a_2 = 0$ and $a_3 = 0$. With this constraints the rate of s_1 is already not larger than 1. The sink t_1 receives on its incoming edges $a_1 + b_1$,$b_1 + b_2$,$b_1 + b_2$. It cannot recover b_1, and therefore achieves zero bit rate. It can be shown that the rate pair $h_1 = 1, h_2 = 2$ is in fact not achievable by any code.

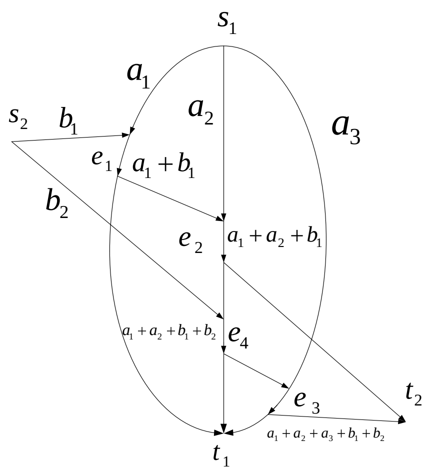

Fig. 4. Final Code

III. Improving the Multicommodity Flow

In this section we show how to improve a general point in the rate region of the multicommodity flow. Suppose we are given a multicommodity solution (h_1, h_2). Denote the flow of source s_1 at edge e as x_e^1 and the flow of source s_2 at edge e as x_e^2. At each edge e from the law of conversion of flow for each commodity it follows that:

$$\sum_{e'\in\Gamma_I(e)} x_e^1 = \sum_{e'\in\Gamma_O(e)} x_e^1, \quad \sum_{e'\in\Gamma_I(e)} x_e^2 = \sum_{e'\in\Gamma_O(e)} x_e^2 \tag{12}$$

The capacity constraints are:

$$x_e^1 + x_e^2 \le c(e)\ \forall e, \quad x_e^1 \ge 0, \quad x_e^2 \ge 0 \tag{13}$$

where for edge $e = (u, v)$, the capacity $c(e)$ is the multiplicity of the unit capacity edges between u and v. If rate (h_1, h_2) is achieved by the multicommodity flow, then:

$$x_s^1 = x_t^1 = h_1, \quad x_s^2 = x_t^2 = h_2 \tag{14}$$

where x_s^1 is the flow leaving s_1 and x_t^1 is the flow of source s_1 reaching t_1. Any multicommodity flow solution has to maintain conditions (12)-(14). The solution to the multicommodity problem defines flow G_1' from s_1 to t_1 and flow G_2' from s_2 to t_2. If we consider G_1' it is not necessarily the maximal flow from s_1 to t_1. It might be a subgraph of a larger flow G_1'' from s_1 to t_1, which again is not necessarily the maximal flow. Therefore, given G_1' we add additional paths from s_1 to t_1 in order to compose G_1''. We denote the additional paths in $G_1'' \setminus G_1'$ as D_1. Likewise, we can construct D_2 a set of paths added to G_2', which together compose G_2''.

Due to construction, in the union network of $U = D_1 \cup G_2'$, if s_2 transmits flow G_2' at rate h_2, then s_1 cannot receive data without coding. Using the coding scheme in the previous section, however, s_1 can transmit data at a certain rate. Thus in the total network G, s_1 transmits data at a rate higher than h_1 and s_2 transmits data at rate h_2, which improves the rates of the multicommodity point (h_1, h_2).

In case we are not given an initial multicommodity flow solution, we can formulate the conditions on G_1', D_1, G_2' and D_2 that have to be maintained. For that we define four commodities. The two commodities x_e^1 and x_e^3 are transmitted by s_1 and received by t_1. The two commodities x_e^2 and x_e^4 are transmitted by s_2 and received by t_2. The flows x_e^1 and x_e^3 define G_1' and D_1, respectively. Thus we can define them by the following conditions:

$$\sum_{e'\in\Gamma_I(e)} x_e^1 = \sum_{e'\in\Gamma_O(e)} x_e^1, \quad \sum_{e'\in\Gamma_I(e)} x_e^3 = \sum_{e'\in\Gamma_O(e)} x_e^3 \tag{15}$$

$$x_e^1 + x_e^3 \le c(e)\ \forall e, \quad x_e^1 \ge 0, \quad x_e^3 \ge 0 \tag{16}$$

Likewise the flows x_e^2 and x_e^4 define G_2' and D_2, respectively and satisfy the following conditions:

$$\sum_{e'\in\Gamma_I(e)} x_e^2 = \sum_{e'\in\Gamma_O(e)} x_e^2, \quad \sum_{e'\in\Gamma_I(e)} x_e^4 = \sum_{e'\in\Gamma_O(e)} x_e^4 \tag{17}$$

$$x_e^2 + x_e^4 \le c(e)\ \forall e, \quad x_e^2 \ge 0, \quad x_e^4 \ge 0 \tag{18}$$

The flows x_e^1 and x_e^2 constitute together the multicommodity flow. Therefore they have to maintain the usual conditions for multicommodity flows the conditions:

$$x_e^1 + x_e^2 \quad \le \quad c(e)\ \forall e \tag{19}$$

Note that flow x_e^1 is allowed to overlap with flow x_e^4. Likewise for flows x_e^2 and x_e^3.

Definition 3.1: Denote the set $x_e^1, x_e^2, x_e^3, x_e^4, \forall e \in E$ that maintain conditions (15)-(19) as quasiflow Z.

The computational complexity of finding the quasiflow Z is similar to the complexity of finding the multicommodity flow, since both involve linear programming with a similar number of variables and inequalities. Given Z, we can take the following procedure to find the code. At the first stage consider the flows x_e^1, x_e^2, x_e^3. We know from the conditions that x_e^1 does not intersect x_e^2, x_e^3. The data transmitted on edges in x_e^1 is uncoded and behaves as flow. On the other hand, the data on x_e^2, x_e^3 is coded according to the method we have introduced in the previous section. In the second stage we likewise consider x_e^1, x_e^2, x_e^4.

IV. Localized Code Construction

A. A Distributed Quasiflow Construction

The quasiflow Z can be found by linear programming, but the algorithm is not localized and the topology has to be fully known to the designer. For the multicommodity problem, the algorithm presented in [6] can be operated in a localized, distributed manner using bounded queues and local control at each node. The algorithm is an efficient practical approximation and has lower complexity than linear programming. In this section we show how to modify the algorithm in order to find the quasiflow Z. Once Z is found, network coding can be constructed for Z, as will be shown in Section IV-B.

We briefly summarize the algorithm in [6], with minor modifications for quasiflow. The algorithm injects into the network rate $(1+\epsilon)d_i$ per commodity $i = 1, 2, 3, 4$, per unit time, provided that there exists feasible quasiflow with demand

$(1+2\epsilon)d_i$ per commodity i. The algorithm finds a feasible solution with demands d_i for commodity i. It is assumed that for each source there is a single outgoing edge and for each sink a single incoming edge. If this is not the case, we can add a dummy source(sink) with a single outgoing(incoming) edge with infinite capacity. There is a regular queue for each commodity at the head and tail of each directed edge. When a unit of flow traverses an edge, it is removed from the tail queue and added to the head queue. A potential function is associated with each queue. The potential of a regular queue of size q for commodity i is defined as:

$$\phi_i(q) = e^{\alpha_i q} \tag{20}$$

where $\alpha_i = \epsilon/8ld_i$ and l is the length of the longest flow path in G. The size of the source queue (regular queue at the tail of the edge leading from the source s_i) 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 a special overflow queue at each source. The potential function associated with the overflow queue of size b for commodity i is defined as:

$$\sigma_i(b) = b\phi_i'(Q_i) = b\alpha_i e^{\alpha_i Q_i} \tag{21}$$

The algorithm proceeds in rounds, where each unit-time round consists of the following four phases:

- For each source s_i add $(1+\epsilon)d_i$ units of flow to the overflow queue of commodity i and move as much flow as possible from the overflow queue to the source queue.
- For each edge push flow 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 (16), (18) and (19). This optimization problem, can be solved by standard calculus methods.
- For each commodity i empty the sink queue.
- For each commodity i and for each node v rebalance commodity i within node 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.

It can be shown that all but a bounded amount of flow reaches its destination. The algorithm does not guarantee that each unit of flow will get eventually to its destination. However, the amount of undelivered flow stays bounded over the time as can shown by upper bounding the size of the regular queues and overflow queues. The analysis for the quasiflow is similar and can be directly derived from the analysis in [6] for multicommodity flow. 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 will ensure that Rd_i units of commodity i arrive at sink t_i. It can be shown that a sufficient number of rounds is:

$$R = O\left(\frac{El(1+ln(1/\epsilon))}{\epsilon^2}\right) \tag{22}$$

The rate d_i is obtained by averaging the flows over R rounds.

B. Incorporating Network Codes

Since the quasiflow is obtained in R rounds, we can interpret the network as a time slotted network [12], [13].

Definition 4.1: Given a network G, and positive integer R, the associated time slotted network denoted as G^R includes nodes s_1, s_2 and all nodes of 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 $r \le R$ the capacity of the edge from $s_i, i = 1,2$ to x^r, is the same as that of the edge from s_i to x in the network 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 the quasiflow Z, consider the flows $x_e^i, i = 1,2,3,4$. If we consider a single flow $x_e^1, \forall e \in E$, the corresponding (uncoded) transmission of symbols is performed as the following:

- A symbol sent from s_i to x^r corresponds to the symbol sent on edge (s_i, 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 $i = 1, \dots, 4$ the edges in G^R that participate in the flow $x_e^i, \forall e \in E$ form a subgraph with capacity Rd_i. Therefore, after the algorithm finds the quasiflow Z, we can take the pairs x_e^1, x_e^4 or x_e^2, x_e^3 and construct the network code for network G^R according to the algorithm in Section II. That is, the information is coded whenever the flows x_e^1, x_e^4 in the quasiflow Z overlap on the same portion of the capacity. The data on x_e^2 will remain uncoded and will add to the rate of s_2 achieved by the network code. In order for the algorithm to be distributed we will use random coding, where each node chooses locally the coding coefficients from some field. The total number of coding edges in the network is ER. For field size $\mathcal{F}$, the probability of success is $P_{succ} = (1 - \frac{2}{\mathcal{F}})^{ER}$, as can be seen from an analysis similar to the multicast case given in [11]. Therefore the field size will be chosen as $O(ER)$ and the block size as $O(\log(ER))$. Since the delay of the scheme is at least R rounds, the addition of the delay due to the coding is of logarithmic order and is relatively negligible.

The algorithm does not guarantee that all the flow will arrive to the sinks, and therefore some packet will be lost. The fact that the network is assumed to be constant helps to deal with this problem, but we omit here the details. The choice of the coding coefficients is performed randomly. There are a number of ways to set up the coding scheme, and we briefly present here a possible set up. We denote a set of R rounds as a stage. We assume that at the first stage no information is conveyed to the sinks. In the first stage the quasiflow Z is determined.

After the first stage, the flows in all future stages will behave the same as in the first stage, since we assume that the network does not change over time and the algorithm for determining the flow is deterministic. After the first stage, all the nodes draw the randomized coding coefficients required for the coding and store them in their memory for future use. The block size is cER. It can be larger than the capacity of each edge. We therefore change the time scale, so that each edge can carry at least a single symbol in each round. Since a block is treated as an inseparable unit that has to be completely transmitted in a single round, some capacity might be lost since the capacity of each edge has to be divisible by the block length after the time scaling. If it is not divisible, the remainder is disregarded and its capacity is not used. A solution to this problem appears in Section V. At this point, we assume that the capacity are given as integers, where a unit capacity can transmit a single symbol in a single time unit.

In the second stage s_1 transmits a unit matrix while all the nodes perform network coding. As explained in [14], Section 1.5.1., this enables s_1 to inform each sink of the coding coefficients from s_1 to the sink. Then this procedure is repeated for s_2. The coding coefficients are made known to the sinks in order to set the coding constraints, as explained in Section II, and in order to decode the information in future stages. Since the source has to set the coding constraints, the coding coefficients are sent from the sinks to the sources by means of feedback. The sources determine the coding constraints and set them on the symbols they transmit.

If at some point the requirements of the rates change, then as long as the change can be achieved by other coding constraints, no new setup is required for the entire network. The sources need only to find again the new coding constraints and make them known to the sinks. This is in contrary to regular multicommodity, where each change in the data rates requires a totally new setup.

V. Packetized Code Construction

We return to the quasiflow construction in Section IV-A. At each round each edge needs to optimize the flow that passes through it. As in [6], the computation requirements can be reduced by partitioning the flow into packets. We briefly repeat the basic idea of [6]. The flow for commodity i is partitioned into packets of size $(1+\epsilon)d_i$. The approximate packetized queue size at the tail of each edge is defined as the integer number of packets p such that

$$q - 2(1+\epsilon)d_i \leq p(1+\epsilon) \leq q \tag{23}$$

where q is the true queue size. The approximate packetized queue size at the head of each edge is defined as the integer number of packets p such that

$$q \leq p(1+\epsilon) \leq q + 2(1+\epsilon)d_i \tag{24}$$

During each round, one packet will be added to the source for each commodity. Packets will be routed across edges according to the following protocol. Each edge finds the commodity i for which

$$\phi_i'(p_{i,tail}(1+\epsilon)d_i) - \phi_i'(p_{i,head}(1+\epsilon)d_i) \tag{25}$$

is maximized and then a packet for this commodity is routed across the edge, assuming that $p_{i,tail} \geq p_{i,head} + 2$. If there is still excess capacity remaining, the queue sizes are updated and another packet is selected by the same measure. In the case when the capacity of the edge is smaller than the size of the packet, then as much of the packet as the capacity allows is routed during this round. In the next round, the routing of this packet either continues or it is interrupted because some other commodity has become more desirable in the meantime.

In order to minimize the computing costs, the queue sizes are updated at either end of an edge when the final piece of each packet has been delivered or when the delivery of the packet has been interrupted. In the case that the delivery of a packet is interrupted the true queue sizes are updated at either end of the edge. The approximate packetized queue size and the new measure in (25) are updated only if the true queue size has changed by more than one packet worth since the last time the approximate packetized queue size was updated. In [6] the computation complexity of the algorithm is computed.

The problem that arises is how to incorporate network coding into the packetized quasiflow construction. As we have noted earlier in Section IV-B, in block network coding each symbol is treated as an inseparable unit. However, in this algorithm the transmission of a packet can be interrupted, and will not necessary continue in the next round. Moreover, the packet size might be smaller than the block length $\log(cER)$. It seems that convolutional network codes are more natural in the case of packetized quasiflow, since the symbols are treated sequentially and do not require inseparable packets. As in [15], we restrict ourselves to binary convolutional network codes. The memory in each node in the network, for each coding coefficient is $\log(cER)$. The polynomial coding coefficients are randomly drawn, just as in the case of block network codes. The decoding delay is also analyzed in [15]. The decoding delay is bounded by $ER\log(cER)$, whereas the decoding delay for the uncoded case is R.

VI. Further Research

We have showed that our new construction improves the rates of the multicommodity solution. A more precise analysis of the performance seems to be complicated for deterministic coding. As usual in these cases, random codes facilitate the analysis. For the case of random codes, we achieved upper and lower bounds on the performance of our construction [16]. The values of the bounds depend on the topology of the network. As expected, the lower bound improves the multicommodity. The upper bound of the rates is below the trivial min-cut bound.

References

[1] R. Koetter and M. Médard, “Beyond routing: An algebraic approach to network coding,,” *IEEE/ACM Trans. on Networking*, vol. 11, no. 5, pp. 782–796, Oct. 2003.

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

[3] L. Song, R.W. Yeung, and N. Cai, “Zero-error network coding for acyclic networks,” *IEEE Trans. Inform. Theory*, vol. 49, no. 12, pp. 3129 – 3139, Dec. 2003.

[4] 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*, February 2006.

[5] Y. Wu, “On constructive multi-source network coding,” *Proc. Int. Symp. Inform. Theory*, Seattle, July 2006.

[6] B. Awerbuch and T. Leighton, “Improved approximation algorithms for the multicommodity flow problem and local competitive routing in dynamic networks,” *Proc. of the 26th ACM Symp. on Theory of Computing*, pp. 487–496, 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,” *Proc. Int. Symp. Inform. Theory*, Seattle, July 2006.

[8] T. C. Ho, Y-H Chang, and K. J. Han, “On constructive network coding for multiple unicasts,” *44th Allerton Conf. on Commun., Control, and Comupting*, Sept. 2006.

[9] A. Eryilmaz and D.S. Lun, “Control for inter-session network coding,” Tech. Rep. 2722, MIT LIDS, August 2006.

[10] 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 Trans. on Inform. Theory*, vol. 51, no. 6, pp. 1973–1982, June 2005.

[11] T. Ho, M. Médard, M. Effros, and D. Karger, “On randomized network coding,” *41th Allerton Conf. on Commun., Control, and Comupting*, Oct. 2003.

[12] R. Ahlswede, N. Cai, S.-Y. R. Li, and R.W. Yeung, “Network information flow,” *IEEE Trans. on Inform. Theory*, vol. 46, no. 4, pp. 1204–1216, July 2000.

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

[14] E. Erez and M. Feder, “On codes for network multicast,” *42th Allerton Conf. on Commun., Control and Computing*, Oct. 2003.

[15] E. Erez and M. Feder, “Convolutional network codes,” *IEEE Int. Symp. on Inform. Theory*, Chicago, 2004.

[16] E. Erez, *Topics in Network Coding*, Ph.D. thesis, Tel Aviv University, in preparation.