Network Coding for Multiple Unicasts: An Approach based on Linear Optimization

Danail Traskov, Niranjan Ratnakar, Desmond S. Lun, Ralf Koetter, and Muriel Médard

***Abstract*—In this paper we consider the application of network coding to a multiple unicast setup. We present two suboptimal, yet practical code construction techniques. One consists of a linear program and the other of an integer program with fewer variables and constraints. We discuss the performance of the proposed techniques as well as their complexity.**

I. Introduction

We consider the problem of setting up several simultaneous and independent point-to-point connections that share a common network and refer to this as a *multiple unicast* setup. The notion of *network coding* [1] permits intermediate network nodes to encode observed data as opposed to traditional operation, where the network nodes are restricted to routing and replicating. An example of such an encoding is illustrated in Fig. 1(a), where u observes b_1 and b_2 and transmits $b_1 \oplus b_2$.

We are interested in applying network coding to the multiple unicast setup. Most results on network coding so far assume a multicast setting, i.e. a single source transmits the same data simultaneously to a set of receivers [1, 2]. This scenario is considerably simpler than the multiple unicast problem. One simplifying aspect is the fact that the problem of finding a minimum-cost multicast connection decomposes into two independent subproblems [3]. First, one has to identify a subgraph whose link capacities can support the multicast connection. Second, the code can be constructed *independent* of the subgraph by e.g. a randomized code construction [2]. On the other hand, in the multiple unicast scenario, the problems of selecting a subgraph and the code construction have to be solved jointly. The selection of a subgraph is typically modeled as an optimization problem on flows in a network, whereas the code construction is an algebraic and combinatorial problem. Since we have to solve them jointly, we restrict ourselves to simple network codes, whose construction can be combined with network flow problems of tractable complexity.

We assume the underlying network to be a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ are the sets of vertices and directed edges respectively. We desire communication of independent information from nodes s_i to d_i at a rate R_i for $i = 1, \ldots, n$. A connection is defined as a triple (s_i, d_i, R_i). The n triples (s_i, d_i, R_i) are *feasible with network coding* if there exists a network code that allows for simultaneous transmission of information at a rate R_i from s_i to d_i. The triples are said to be *feasible with routing* if there exists a network code involving only routing and in this case the problem of determining flows that satisfy these connections is referred to as a multicommodity flow problem [4].

The application of network coding in a multiple unicast setup is discussed in [5]. In particular, we know of two approaches for constructing network codes for the non-multicast scenario. In [6], the authors present algebraic code construction techniques for all connection triples that are feasible with linear network coding. However, the computational complexity of solving the resulting system of polynomial equation is exponential in the size of the underlying network and therefore hardly tractable in practice. On the other hand, the authors in [7] present an opportunistic heuristic code construction for wireless networks. The approach presented in this paper fits in between these two schemes. We present code construction techniques for *certain* connection triples that are feasible with network coding, but are not necessarily feasible with routing. However, we emphasize that the techniques described here fail to provide network codes for all connection triples that are feasible with linear network coding. We restrict network coding to operations using only binary XOR, derive flow formulations of the problem, and state it first as a linear program and then as an integer program. This constitutes a suboptimal, yet practical and systematic scheme to construct network codes. The linear and integer program model the same problem and, although not equivalent, are quite similar. Their main difference lies in computational issues.

The paper is organized as follows: In Section II, we present a code construction scheme based on a linear program. In Section III we present an alternative formulation with fewer variables and constraints, albeit in the form of an integer program. A comparison of both formulations in terms of their complexity and simulation results is given in Section IV. We summarize our results in Section V.

II. Code construction as a linear program

For an edge $e = (u, v)$ we define tail(e) = u and head(e) = v. We use $\Gamma_I(v)$ to denote the set of edges whose head is $v \in \mathcal{V}$ and $\Gamma_O(v)$ to denote the set of edges with tail v. The n triples (s_i, d_i, R_i) are feasible with routing if the following conditions are satisfied [4]. For $i = 1, 2, \ldots, n$ and $e \in \mathcal{E}$, $v \in \mathcal{V}$,

$$\sum_{i=1}^{n} x_e(i) \le z_e \le C_e, \quad x_e(i) \ge 0 \tag{1}$$

$$\sum_{e \in \Gamma_O(v)} x_e(i) - \sum_{e \in \Gamma_I(v)} x_e(i) = \begin{cases} R_i & \text{if } v = s_i \\ -R_i & \text{if } v = d_i \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

D. Traskov, N. Ratnakar, and R. Koetter are with the Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, 1308 Main St., Urbana IL 61801 (email: {traskov2, ratnakar, koetter}@uiuc.edu). This material is based upon work supported by the NSF under Award No. NSF CCR-0325673.

D. S. Lun, M. Médard are with the Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA 02139 (email: {dslun, medard}@mit.edu).

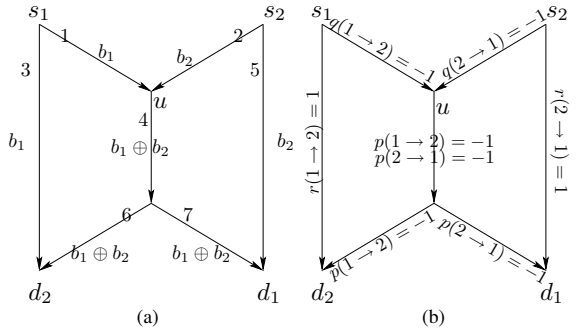

Fig. 1. (a) shows the butterfly network. The edges 1 through 7 are marked and the network coding solution with the corresponding transmission on each edge is shown. b_i represents data for the i-th connection for $i = 1, 2$. In (b), the butterfly network with the poison, antidote request, and antidote variables that are non-zero on each edge. All variables correspond to the poison generated at node u.

In the above equations, $x_e(i)$ is a flow from s_i to d_i of value R_i and z_e is the aggregate load on e.

However, when network coding is permitted, the notion of a flow is not clear. For example, consider the so-called *butterfly* network shown in Fig. 1(a) where all the edges have unit capacity. It can be verified that $R_1 = R_2 = 1$ is not feasible with routing, yet feasible with network coding as shown in Fig. 1(a). It is no longer clear how to interpret this code as concurrent flows. Note that the data transmitted on edge 4 is $b_1 \oplus b_2$ and knowledge of b_2 is needed at d_1 in order to recover b_1. We interpret the transmission of $b_1 \oplus b_2$ as a "poisoning" of the two flows and the transmission of b_2 (b_1) on edge 5 (3) as an "antidote". We also introduce the notion of an "antidote request" which notifies node s_1 (s_2) of the poisoning downstream and requests an antidote to d_2 (d_1).

Given an arbitrary network and multiple unicast connections, we wish to identify network codes using this poison-antidote approach. In order to do so, we enrich the variable-space by introducing variables $p_e(i \to j, u), q_e(i \to j, u), r_e(i \to j, u)$ for every vertex u. These variables respectively keep track of the poison, antidote request, and antidote associated with the poisoning of user j's data by user i's data at vertex u. This is done to ensure that the poison originating at a particular vertex u is remedied by the appropriate antidote. In this section, we derive sufficient conditions for the existence of a network coding solution involving routing, duplicating, and bit-level XOR.

Theorem 1: For a network $G = (\mathcal{V}, \mathcal{E})$, with the capacity of edge e given by C_e and the ranges of the variables given by $v, u \in \mathcal{V}$, $e \in \mathcal{E}$, $1 \le i \le n$, $1 \le j \le n$, and $i \ne j$, consider the following set of equations.

For all v, u, i, and j:

$$\sum_{e \in \Gamma_I(v)} p_e(i \to j, u) + q_e(i \to j, u) + r_e(i \to j, u) = \sum_{e \in \Gamma_O(v)} p_e(i \to j, u) + q_e(i \to j, u) + r_e(i \to j, u) \quad (3)$$

$$\sum_{e \in \Gamma_I(v)} q_e(i \to j, u) - \sum_{e \in \Gamma_O(v)} q_e(i \to j, u) \begin{cases} \le 0 & \text{if } v = u \\ \ge 0 & \text{otherwise} \end{cases} \quad (4)$$

$$\sum_{e \in \Gamma_I(v)} p_e(i \to j, u) - \sum_{e \in \Gamma_O(v)} p_e(i \to j, u) \begin{cases} \ge 0 & \text{if } v = u \\ \le 0 & \text{otherwise} \end{cases} \quad (5)$$

For all u, i, and j:

$$p_e(j \to i, u) = p_e(i \to j, u) \quad \text{if} \quad e \in \Gamma_O(u) \quad (6)$$

At every edge e:

$$\sum_u \sum_{\substack{i,j \\ i<j}} \max(p_e(i \to j, u), p_e(j \to i, u)) + \sum_{i=1}^{n} x_e(i) + \sum_u \sum_{\substack{i,j \\ i \ne j}} r_e(i \to j, u) \le z_e \le C_e \quad (7)$$

At every edge e and j:

$$x_e(j) + \sum_u \sum_i (p_e(i \to j, u) + q_e(j \to i, u)) \ge 0 \quad (8)$$

$$x_e(\cdot) \ge 0, \quad r_e(\cdot) \ge 0, \quad p_e(\cdot) \le 0, \quad q_e(\cdot) \le 0$$

If the conditions specified in (2) - (8) are satisfied, then the rate triples (s_i, d_i, R_i) for $i = 1, 2, \ldots, n$ are feasible with network coding. Further, the only operations involved are duplication, routing, and bit-level XOR.

Proof: We assume that the variables satisfying (2) - (8) are integers, which is justified by scaling the network over time [6]. We begin by making the following observations.

Observation 1: From (4) we see that antidote request $q_e(i \to j, u)$ (a negative quantity) can "terminate" only at node u.

Observation 2: From (5) we see that poison $p_e(i \to j, u)$ (a negative quantity) can "originate" only at node u.

Observation 3: For edge e, let $f_e(i \to j, u) = p_e(i \to j, u) + q_e(i \to j, u) + r_e(i \to j, u)$. Fix i, j, and u. From (3) we see that $f_e(i \to j, u)$ is conserved at each node. In the example shown in Fig. 1(b), $f_e(1 \to 2, u) = 1$ for $e = 3$ and $f_e(1 \to 2, u) = -1$ for $e = 1, 4, 6$. Notice that the edges 1,3,4, and 6 form a cycle if we reverse the direction of edges with $f_e(1 \to 2, u) = -1$, i.e., 1,4, and 6. In general, for any network, $f_e(i \to j, u)$ can be decomposed as a superposition of cycles where $f'_e(i \to j, u) = \pm 1$ for all edges e with the understanding that if $f'_e(i \to j, u) = -1$, then the direction of e is reversed. Due to paucity of space, we skip the proof here. We refer to each of these cycles as an $f'(i \to j, u)$ cycle for i, j, and u. Further, since the quantities $p(\cdot)$, $q(\cdot)$, and $r(\cdot)$ are integers, it follows that along any edge of an $f'(i \to j, u)$ cycle, only one of these quantities is nonzero.

From Observations 1 and 2 we see that $q_e(i \to j, u)$ "terminates" only at u and that $p_e(i \to j, u)$ "originates" only at u. (To simplify notation we represent a path from u_1 to v_m as (u_1, v_m). The distinction between an edge (u, v) and a path (u_1, v_m) is clear from the context.) Thus, any $f'(i \to j, u)$ cycle comprises three paths (v, u), (u, w) and (v, w) such that (v, u) satisfies $q_e(i \to j, u) = -1$, (u, w) satisfies $p_e(i \to j, u) = -1$, and (v, w) satisfies $r_e(i \to j, u) = 1$. Some $f'(i \to j, u)$ cycles might have $v = u$, which would mean that it comprises of only the poison and antidote paths.

From (6), we see that if tail$(e) = u$ then we can associate $p_e(j \to i, u)$ with $p_e(i \to j, u)$. This can be extended and we can associate an $f'(j \to i, u)$ cycle with an $f'(i \to j, u)$ cycle. Thus, all the poison, antidote, and antidote-request variables can be written as a superposition of pairs of $f'(i \to j, u)$ and $f'(j \to i, u)$ cycles.

We show that we can perform network coding on each of these pairs of cycles. Consider one such pair of cycles. Let the cycles be $f'(i \to j, u_3)$ and $f'(j \to i, u_3)$. From the discussion above, the $f'(i \to j, u_3)$ cycle is composed of paths (u_1, u_3) with $q_e(i \to j, u_3) = -1$, (u_3, u_4) with $p_e(i \to j, u_3) = -1$, and (u_1, u_4) with $r_e(i \to j, u_3) = 1$. Similarly, the $f'(j \to i, u_3)$ cycle is composed of paths (u_2, u_3) with $q_e(j \to i, u_3) = -1$, (u_3, u_5) with $p_e(j \to i, u_3) = -1$, and (u_2, u_5) with $r_e(j \to i, u_3) = 1$.

Using (8), we note that we can associate a unique unit-rate path, say, $x'_e(i) = 1$ to the paths (u_1, u_3) and (u_3, u_5). Similarly we can associate a unique unit-rate path say $x'_e(j) = 1$ to the paths (u_2, u_3) and (u_3, u_4).

In the example shown in Fig. 1(b), we have $u_1 = s_1, u_2 = s_2, u_3 = u, u_4 = d_1$, and $u_5 = d_2$. Note that $x_e(1) = 1$ ($x_e(2) = 1$)) for $e = 1, 4, 7$. ($e = 2, 4, 6$.) Let b_1 represent a unit rate bit stream corresponding to $x(i)$ available at u_1 and b_2 represent a unit rate bit stream corresponding to $x(j)$ available at u_2. We perform the following network coding in order to reliably transmit one unit data from u_1 to u_5 and one unit of data from u_2 to u_4.

- b_1 is transmitted along the paths (u_1, u_3) and (u_1, u_4).
- b_2 is transmitted along the paths (u_2, u_3) and (u_2, u_5).
- $b_1 \oplus b_2$ is transmitted along the paths (u_3, u_4) and (u_3, u_5).

This encoding, in effect, is equivalent to replacing $x'_e(i), x'_e(j)$ and the $f'(i \to j, u_3), f'(j \to i, u_3)$ cycles by two unit-capacity virtual edges $e_1 = (u_1, u_5)$ and $e_2 = (u_2, u_4)$ such that $x_{e_1}(i) = 1$ and $x_{e_2}(j) = 1$. This replacement ensures that the number of $f'(i \to j, u)$and $f'(j \to i, u)$ cycles is reduced while still ensuring that $x_e(i)$ and $x_e(j)$ (the affected flows) satisfy the flow-conservation equations.

For this encoding, it can be verified that the usage of all edges along the paths $(u_1, u_3), (u_2, u_3), (u_3, u_4), (u_3, u_5), (u_1, u_4)$, and (u_2, u_5) is given by the expression

$$\max(p_e(i \to j, u_3), p_e(j \to i, u_3)) + x_e(i) + x_e(j) + r_e(i \to j, u_3) + r_e(j \to i, u_3) \quad (9)$$

which evaluates to one along all the edges. Thus, the usage of the edges in this pair of $f'(i \to j, u_3)$ and $f'(j \to i, u_3)$ cycles is given by (9).

Note that the terms $r_e(i \to j, u_3) + r_e(j \to i, u_3)$ correspond to physical transmissions of bits. The term $x_e(i) + x_e(j)$ can be thought of as the data that would be transmitted if data is sent by routing. However, by sending $b_1 \oplus b_2$, we transmit at a lower rate, lower by $\max(p_e(i \to j, u_3), p_e(j \to i, u_3))$. These can be thought of as *savings* obtained due to network coding.

Using Observation 3, we see that $f_e(i \to j, u)$ is a superposition of pairs of cycles, each of which can be replaced by a pair of virtual edges over which routing suffices. By performing this operation till there are no more such cycles left, we end up with an equivalent network over which the connection triples (s_i, d_i, R_i) for $i = 1, 2, \ldots, n$ are feasible with routing.

If all the data is transmitted by routing, then the load on any edge is $\sum_{i=1}^{n} x_e(i)$. However, recall that due to an $f'(i \to j, u)$ cycle, the load on the edges in the cycle is adjusted by the quantity in (9). Since $f_e(i \to j, u)$ is a superposition of many $f'(i \to j, u)$ cycles, the net adjustment of the load on the edges is obtained by summing (9) over all the $f'(i \to j, u)$ cycles. This expression is given by the left hand side of (7). The interpretation of the various terms appearing in the summation is as follows: By network coding, we obtain a savings of $\sum_u \sum_{\substack{i,j \\ i \neq j}} \max(p_e(i \to j, u), p_e(j \to i, u))$ (which is negative) which is offset by excess load of $\sum_u \sum_{\substack{i,j \\ i \neq j}} r_e(i \to j, u)$ (which is positive) due to the remedies. Usually, the savings are non-zero on congested edges and the remedies are non-zero on edges with spare capacity, thus allowing us to trade off bottleneck links with excess capacities. ■

Every solution that satisfies the constraints (2) - (8) corresponds to a network code that transmits data at a rate z_e (see (7)) on edge e. These constraints can be combined with several possible objective functions, e.g., one can maximize the total throughput (the sum of rates $\sum_i R_i$) or minimize the cost of communicating certain fixed rates subject to the capacity constraints of the graph.

III. Integer programming formulation

The previous linear program offers some flexibility in the choice of objective functions, permits routing (and network coding) along several paths from source to destination and makes little assumption about the underlying network topology. However, it results in large optimization problems, both in terms of the number of constraints and the number of variables. We will discuss the complexity in Section IV. In particular, keeping track of the origin of the poison increases the complexity drastically. (The term u in $p_e(i \to j, u)$ can be thought of as the variable keeping track of the origin of the poison.) This motivates us to consider a simpler optimization problem, where we restrict the connections to be of unit rate.[1] This makes the resulting optimization problem an integer program, but with fewer constraints and variables than the original formulation.

Consider the following linear program, where throughout $1 \leq i, j \leq n$, and $i \neq j$:

$$\min \sum_{e \in E} a_e \cdot z_e$$

at every node $v \in \mathcal{V}$:

$$\sum_{e \in \Gamma_O(v)} x_e(i) - \sum_{e \in \Gamma_I(v)} x_e(i) = \begin{cases} +1 & \text{if } v = s_i \\ -1 & \text{if } v = d_i \\ 0 & \text{otherwise} \end{cases} \quad (10)$$

$$\sum_{e \in \Gamma_O(v)} r_e(i) - \sum_{e \in \Gamma_I(v)} r_e(i) \leq \sum_{e \in \Gamma_I(v)} (x_e(i) + \sum_{j=1}^{n} p_e(i, j)) \quad (11)$$

$$\sum_{e \in \Gamma_O(v)} p_e(i, j) - \sum_{e \in \Gamma_I(v)} p_e(i, j) \leq \sum_{e \in \Gamma_I(v)} r_e(j) \quad (12)$$

[1] Higher rates can be modeled by placing several connections between the same source-destination pair.

at every destination node d_i:

$$\sum_{e\in\Gamma_O(d_i)}\sum_{j=1}^{n} p_e(i,j) = 0 \tag{13}$$

at every edge $e \in \mathcal{E}$:

$$\sum_{\substack{i,j\\ i<j}} \max(p_e(i,j), p_e(j,i)) + \sum_{i=1}^{n} x_e(i) + \sum_{i=1}^{n} r_e(i) \le z_e \le C_e \tag{14}$$

$$x_e(i) + \sum_{j=1}^{n} p_e(i,j) \ge 0 \tag{15}$$

$$x_e(\cdot) \ge 0, \quad r_e(\cdot) \ge 0, \quad p_e(\cdot) \le 0$$

Note that we require flows of unit rate (10), thus losing some flexibility in the problems that can be modeled. The cost function takes the form of a minimum cost flow problem, with cost coefficients a_e multiplying the usage z_e of every edge. Furthermore, we require the underlying graph to be acyclic, which becomes necessary to exclude invalid network coding solutions. In the following we prove that solutions of this integer program are in fact valid network codes, which can be constructed by using routing and the binary XOR operation.

Lemma 1: The cost of the optimal solution of the linear program (10)-(15) is smaller or equal to the optimal cost of the associated multi-commodity flow problem (10), (16), and (17).

Proof: Set all variables $r_e(\cdot)$ and $p_e(\cdot)$ to zero. Then, the linear program reduces to the following form:

$$\min \sum_{e\in E} a_e \cdot z_e$$

subject to (10) at every node $v \in \mathcal{V}$ and

$$\sum_{i=1}^{n} x_e(i) \le z_e \le C_e \tag{16}$$

$$x_e(i) \ge 0 \tag{17}$$

at every edge $e \in \mathcal{E}$. This is identified to be the associated multi-commodity flow problem consisting of the flow conservation constraints at the nodes, the constraint that the edge usages z_e do not exceed the edge capacities C_e and the nonnegativity constraints on the flows. The optimal solution to this multi-commodity flow problem is a basic feasible solution to the linear program (10)-(15) and therefore is also an upper bound on its optimal solution. Continuing along similar lines, it is possible to prove that the optimal cost of the linear program (2)-(8) is also lower than the cost of the optimal solution of the associated multi-commodity flow problem ■

Theorem 2: If a solution to the linear program (10)-(15) is integer, then it corresponds to a feasible network coding solution involving only the binary XOR as a coding operation.

Proof: Assume, we have an integer solution. Since the flows are of unit rate we have $x_e(i) \in \{0, +1\}$. Furthermore, from (15) we conclude that $p_e(i,j) \in \{0, -1\}$. If all $p_e(\cdot,\cdot)$ variables are zero, then according to the previous lemma the solution corresponds to a multi-commodity flow problem and therefore the connections can be established by routing only.

Assume now some $p(\cdot,\cdot)$ variables equal to -1 and pick an edge e (with tail$(e) = u$ and head$(e) = v$), where $p_e(i,j) = -1$. (15) implies that $x_e(i) = 1$ on this edge also. Since the flows are of rate one and we insist on integer solutions, flow i takes precisely one path from its source s_i to its destination d_i and clearly edge e lies on this path. The poison $p_e(i,j)$ satisfies a conservation law of the form (12), and therefore forms a path which can use only edges on the path of flow i (15). From (13) the $p_e(i,j)$-path has to terminate at a node $w \le d_i$[2]. At the termination point w of the $p_e(i,j)$-path an antidote $r(j)$ has to enter the node (12). This antidote also forms a path (11) and let its source be node t. The RHS of (11) has to be strictly positive at node t, which implies that the flow $x(j)$ entering node t must be one, and the poison $p(j,\cdot)$ entering t must be zero.

Given a solution to the proposed integer linear program, we construct the network code by transmitting the following bit streams on edge e:

- $x(i)$ if $x_e(i) = 1$ and $p_e(i,\cdot) = 0$,
- $x(i) \oplus x(j)$ if $x_e(i) = 1$ and $p_e(i,j) = -1$ for some j,
- $x(i)$ if $r_e(i) = 1$.

We have proved, that every poison is canceled by the appropriate antidote, and that every antidote has its origin at a node, where unpoisoned flow of the proper type enters. The assumption that the graph is acyclic is needed to ensure that the antidote comes from a node upstream of the poisoning node. This shows the validity of the proposed code construction scheme and completes the proof. ■

	Variables	Constraints
Linear Program	$3\|\mathcal{E}\|\|\mathcal{V}\|n(n-1)$	$\|\mathcal{V}\|(n^2-n)(3\|\mathcal{V}\|+\|\mathcal{E}\|)$
Integer Program	$\|\mathcal{E}\|n(n-1)$	$(n^2+n)\|\mathcal{V}\|+(n+1)\|\mathcal{E}\|$

TABLE I

THE COMPLEXITY OF BOTH OPTIMIZATION PROBLEMS. WE APPROXIMATE THE NUMBERS UP TO LEADING TERMS.

IV. PERFORMANCE EVALUATION

Counting the number of variables and constraints (see Table I) of both formulations, we see that the integer program is of considerably smaller size than the linear program. On the other hand, solving an integer program is in general more complex than solving a linear program without integer constraints. Integer programming is NP-complete [8] and networks exist where the integer program is very difficult to solve. To find integer solutions we have used a linear programming-based branch-and-bound algorithm. In general one has to consider the trade-off between the reduced size of the optimization problem (10)-(15) and the increase in complexity due to branch-and-bound.

The first setup that we considered for performance evaluation of our network coding scheme is the frequently used geometric random graph model. Here, a number of nodes are scattered uniformly over a unit square and nodes whose distance is below a certain threshold are considered connected with a point-to-point link. We have conducted extensive simulations

[2]Since we require a directed acyclic graph, the vertices can be topologically sorted, thereby defining an order on the vertex set $\mathcal{V}$.

on random geometric networks, using different objective functions such as total throughput or minimum cost, but could not observe gains. This observation motivates us to consider more regular grid networks, such as the one depicted in Fig. 2. We illustrate the performance of the integer program[3] on the grid network as documented in Table II. We consider no capacity constraints and a minimum-cost objective function. The random parameters in the simulations are the cost coefficients of the links. We consider two different distributions according to which they are drawn. The first distribution models a scenario where most links have low cost, and a few links have large cost, thus representing bottlenecks. In this setup we observe gains due to network coding. When the link costs are uniformly distributed, however, the gains disappear.

A comparison with the simulation results given by the authors in [9] and more recently [7] is in order. In particular, in [7] the authors consider a wireless network, which uses geographic (min-hop) routing. They report large gains in the total throughput, when implementing an opportunistic heuristic code construction technique. These gains can be explained with several factors, the most important being the lack of congestion control when using solely geographic routing. If the routes are chosen based on geography, without taking congestion into account, a drastic breakdown of the total throughput is experienced as the number of flows is increased. Network coding acts here as a congestion resolution technique and gives large gains. Another fundamental difference between the setup in [7] and our model is that the former works on a wireless network, where nodes broadcast their messages to a set of neighbors, a phenomenon referred to as the *wireless multicast advantage* [10]; we on the other hand consider a wired network with point-to-point links.

The gains that we report are fairly modest, compared to [7,9]. However, these results hold only for the narrow setup that we consider, namely wireline networks, optimal routing[4] as the benchmark and no consideration of practical implementation issues such as protocol design. Furthermore, we assume the flows in the network static and do not consider the dynamic behavior of the network. It is worthwhile to investigate the utility of network coding, when any of these restrictions is relaxed. Based on the previous discussion, larger gains might be observed in wireless networks (due to the wireless multicast advantage), when the cost coefficients or capacities of the links are distributed highly non-uniform, or when considering network coding as a tool to enhance routing, e.g. as a congestion resolution technique. Also, further research is indicated on the dynamic aspects of networks using network coding and the interplay between practical network design and network coding.

V. Conclusions

Few explicit code construction techniques are known for the multiple unicast setup. In this paper we proposed a systematic approach to introduce suboptimal, yet practical network coding for multiple unicasts. Our scheme comes in two variations, a linear program and an integer program of smaller size. We discussed the computational complexity and based on simulations evaluated the network coding gains. We also compared them with recently published results by other authors.

[3]Since both the linear and the integer program model the same problem, the linear program in this case would have given equivalent results.

[4]By optimal routing we mean the optimal solution of the associated multicommodity flow problem.

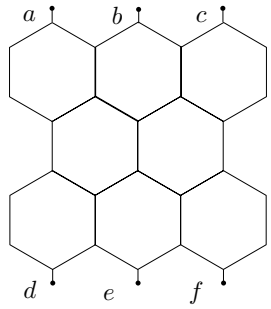

Fig. 2. The grid network used for simulations. Note, that all edges are directed from the sources a, b, c down to the sinks d, e, f.

Approach	Connections	Networks with gains	Average gain
$a_i \sim$ 1 wp 0.9, 100 wp 0.1	2	1.1%	28.0%
	3	5.8%	24.9%
$a_i \sim \text{unif}(1, 100)$	2	0	0
	3	0	0

TABLE II

Simulation results on the grid network in Fig. 2: A number of connections is randomly selected from the designated source and destination pairs. The link costs are randomly fixed according to the specified distribution. We report the fraction of networks, where network coding resulted in a lower transmission cost than routing. The cost gain is defined as $\frac{cost_{routing}-cost_{coding}}{cost_{routing}}$ and is calculated over the networks, where a gain was observed.

References

[1] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow", *IEEE Trans. on Inf. Theory*, vol. IT-46, pp. 1204–1216, 2000.

[2] T. Ho, M. Médard, R. Koetter, D.R. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast", submitted to *IEEE Trans. on Inf. Theory*.

[3] D. S. Lun, M. Medard, T. Ho, and R. Koetter, "Network coding with a cost criterion", in *Proc. Intl. Symp. on Inf. Theory and its Appl.*, 2004.

[4] D. P. Berstekas, *Network optimization: Continuous and discrete models*, Belmont, MA: Athena Scientific, 1998.

[5] N. Ratnakar, D. Traskov, and R. Koetter, "Approaches to network coding for multiple unicast", in *Intern. Zurich Seminar on Comm.*, Febr. 2006.

[6] R. Koetter and M. Médard, "An algebraic approach to network coding," *IEEE/ACM Trans. on Networking*, vol. 11, no. 5, pp. 782-795, Oct. 2003.

[7] S. Katti, D. Katabi, Wenjun Hu, and Rahul Hariharan, "The importance of being opportunistic: Practical network coding for wireless environments", in *Allerton Conference*, 2005.

[8] I. Barland, P.G. Kolaitis, and M.N. Thakur, "Integer programming as a framework for optimization and approximability", In *Proc. IEEE Conference on Computational Complexity*, pages 249–259, 1996.

[9] R. Khalili and K. Salamatian, "A new relaying scheme for cheap wireless relay nodes" in *Wiopt 2005*, Trentino, Italy.

[10] D.S. Lun, N. Ratnakar, M. Medard, R. Koetter, D.R. Karger, T. Ho, E. Ahmed, "Minimum-cost multicast over coded packet networks", submitted to *IEEE Trans. on Inf. Theory*.