Communicating the sum of sources in a 3-sources/3-terminals network

Michael Langberg
Computer Science Division
Open University of Israel
Raanana 43107, Israel
Email: mikel@openu.ac.il

Aditya Ramamoorthy
Department of Electrical and Computer Engineering
Iowa State University
Ames, Iowa 50011
Email: adityar@iastate.edu

***Abstract*—We consider the network communication scenario in which a number of sources s_i each holding independent information X_i wish to communicate the sum $\sum X_i$ to a set of terminals t_j. In this work we consider directed acyclic graphs with unit capacity edges and independent sources of unit-entropy. The case in which there are only two sources or only two terminals was considered by the work of Ramamoorthy [ISIT 2008] where it was shown that communication is possible if and only if each source terminal pair s_i/t_j is connected by at least a single path.**

In this work we study the communication problem in general, and show that even for the case of three sources and three terminals, a single path connecting source/terminal pairs does not suffice to communicate $\sum X_i$. We then present an efficient encoding scheme which enables the communication of $\sum X_i$ for the three sources, three terminals case, given that each source terminal pair is connected by *two* edge disjoint paths. Our encoding scheme includes a structural decomposition of the network at hand which may be found useful for other network coding problems as well.

I. Introduction

Network coding is a new paradigm in networking where nodes in a network have the ability to process information before forwarding it. This is unlike routing where nodes in a network primarily operate in a replicate and forward manner. The problem of multicast has been studied intensively under the paradigm of network coding. The seminal work of Ahlswede et al. [1] showed that under network coding the multicast capacity is the minimum of the maximum flows from the source to each individual terminal node. The work of Li et al. [6] showed that linear network codes were sufficient to achieve the multicast capacity. The algebraic approach to network coding proposed by Koetter and Médard [3] provided simpler proofs of these results.

In recent years there has also been a lot of interest in the development and usage of distributed source coding schemes due to their applications in emerging areas such as sensor networks. Classical distributed source coding results such as the famous Slepian-Wolf theorem [10] usually assume a direct link between the sources and the terminals. However in applications such as sensor networks, typically the sources would communicate with the terminal over a network. Thus, considering the distributed compression jointly with the network information transfer is important. Network coding for correlated sources was first examined by Ho et al. [2]. The work of Ramamoorthy et al. [9] showed that in general separating distributed source coding and network coding is suboptimal.A practical approach to transmitting correlated sources over a network was considered by Wu et al. [11]. Reference [11] also introduced the problem of *Network Arithmetic* that comes up in the design of practical systems that combine distributed source coding and network coding.

In the network arithmetic problem, there are source nodes each of which is observing independent sources. In addition there is a set of terminal nodes that are only interested in the sum of these sources over a finite field, i.e., unlike the multicast scenario where the terminals are actually interested in recovering all the sources, in this case the terminals are only interested in the sum of the sources.

The rate region of the network arithmetic problem was characterized recently by Ramamoorthy in [8] for the case of directed acyclic networks (DAGs) with unit capacity edges and independent, unit entropy sources in which the network has at most two sources or two terminals. Basically, it was shown that as long as there exists at least one path from each source to each terminal, there exists an assignment of coding vectors to each edge in the network such that the terminals can recover the sum of the sources.

In this work we continue the study of the network arithmetic problem for networks with more than two sources and two terminals. Primarily, we show that the characterization of [8] no longer holds when the number of sources and terminals is greater than two. We note that a similar result was obtained recently in an independent manner by [7]. We then turn to obtain encoding schemes for the three sources - three terminals case ($3s/3t$). We show that as long as each source is connected by *two* edge disjoint paths to each terminal, the network arithmetic problem is solvable. Namely, we present efficient linear encoding schemes that allow communication in this case. Our main result can be summarized as follows.

Theorem 1: Let $G = (V, E)$ be a directed acyclic network with unit capacity edges and three sources s_1, s_2, s_3 containing independent unit-entropy sources X_1, X_2, X_3 and three termi-

Authors are in alphabetical order. The work of M. Langberg was supported in part by ISF grant 480/08. The work of A. Ramamoorthy was supported in part by NSF grant CNS-0721453.

nals t_1, t_2, t_3. If there exist two edge disjoint paths between each source/terminal pair, then there exists a linear network coding scheme in which the sum $X_1 + X_2 + X_3$ is obtained at each terminal t_j. Moreover, such a network code can be found efficiently.

This paper is organized as follows. Section II presents the network coding model that we shall be assuming. In Section III we present our counter example to the characterization of [8] containing 3 sources and 3 terminals. In Sections IV and V we present our main result: the proof of Theorem 1. In Section VI we outline our conclusions. Due to space limitations, many of our claims and all of Section V (our main technical section) appear without proof. The interested reader may find a full version of the paper in [4].

II. Network coding model

In our model, we represent the network as a directed graph $G = (V, E)$. The network contains a set of source nodes $S \subset V$ that are observing independent, discrete unit-entropy sources and a set of terminals $T \subset V$. Our network coding model is basically the one presented in [3]. We assume that each edge in the network has unit capacity and can transmit one symbol from a finite field of size q per unit time (we are free to choose q large enough). If a given edge has a higher capacity, it can be treated as multiple unit capacity edges. A directed edge e between nodes v_i and v_j is represented as $(v_i \rightarrow v_j)$. Thus $head(e) = v_j$ and $tail(e) = v_i$. A path between two nodes v_i and v_j is a sequence of edges $\{e_1, e_2, \ldots, e_k\}$ such that $tail(e_1) = v_i, head(e_k) = v_j$ and $head(e_i) = tail(e_{i+1}), i = 1, \ldots, k-1$.

Our counter-example in Section III considers arbitrary network codes. However, our constructive algorithm for the proof of Theorem 1 shall use linear network codes. In linear network coding, the signal on an edge $(v_i \rightarrow v_j)$, is a linear combination of the signals on the incoming edges on v_i and the source signal at v_i (if $v_i \in S$). In this paper we assume that the source (terminal) nodes do not have any incoming (outgoing) edges from (to) other nodes. If this is not the case one can always introduce an artificial source (terminal) connected to the original source (terminal) node by an edge of sufficiently large capacity that has no incoming (outgoing) edges. We shall only be concerned with networks that are directed acyclic and can therefore be treated as delay-free networks [3]. Let Y_{e_i} (such that $tail(e_i) = v_k$ and $head(e_i) = v_l$) denote the signal on the i^{th} edge in E and let X_j denote the j^{th} source. Then, we have

$$Y_{e_i} = \sum_{\{e_j | head(e_j) = v_k\}} f_{j,i} Y_{e_j} \text{ if } v_k \in V \backslash S, \text{ and}$$

$$Y_{e_i} = \sum_{\{j | X_j \text{ observed at } v_k\}} a_{j,i} X_j \quad \text{if } v_k \in S,$$

where the coefficients $a_{j,i}$ and $f_{j,i}$ are from the operational field. Note that since the graph is directed acyclic, it is possible to express Y_{e_i} for an edge e_i in terms of the sources X_j's. Suppose that there are n sources $X_1, \ldots, X_n$. If $Y_{e_i} = \sum_{k=1}^{n} \beta_{e_i,k} X_k$ then we say that the global coding vector of edge e_i is $\boldsymbol{\beta}_{e_i} = [\beta_{e_i,1} \ \cdots \ \beta_{e_i,n}]$. We shall also occasionally use the term coding vector instead of global coding vector in this paper. We say that a node v_i (or edge e_i) is downstream of another node v_j (or edge e_j) if there exists a path from v_j (or e_j) to v_i (or e_i).

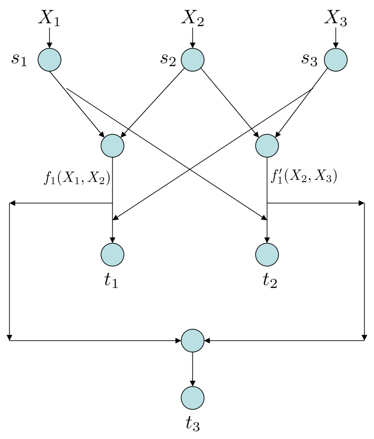

Fig. 1. Example of a network with three sources and three terminals, such that there exists at least one path between each source and each terminal. However all the terminals cannot compute $\sum_{i=1}^{3} X_i$.

III. Example of three sources and three terminals with independent unit-entropy sources

We now present our counter example to the characterization of [8] containing 3 sources and 3 terminals. Namely, we present a $3s/3t$ network with at least one path connecting each source terminal pair, in which the sum of sources cannot (under any network code) be transmitted to all three terminals.

Consider the network in Figure 1, with three source nodes and three terminal nodes such that the source nodes observe independent unit entropy sources X_1, X_2 and X_3. All edges are unit capacity. As shown in Figure 1, the incoming edges into terminal t_3 contain the values $f_1(X_1, X_2)$ and $f_1'(X_2, X_3)$ where f_1 and f_1' are some functions of the sources.

Suppose that $X_3 = 0$. This implies that t_1 should be able to recover $X_1 + X_2$ (that has entropy 1) from just $f_1(X_1, X_2)$. Moreover note that each edge is unit capacity. Therefore the entropy of $f_1(X_1, X_2)$ also has to be 1. i.e. there exists a one-to-one mapping between the set of values that $f_1(X_1, X_2)$ takes and the values of $X_1 + X_2$. In a similar manner we can conclude that there exists a one-to-one mapping between the set of values that $f_1'(X_2, X_3)$ takes and the values of $X_2 + X_3$. At terminal t_3, there needs to exist some function $h(f_1(X_1, X_2), f_1'(X_2, X_3)) = \sum_{i=1}^{3} X_i$. By the previous observations, this also implies the existence of a function $h'(X_1 + X_2, X_2 + X_3)$ that equals $\sum_{i=1}^{3} X_i$. We now demonstrate that this is a contradiction. Let $X_1 = a, X_2 = 0, X_3 = c$ and $X_1' = a - b, X_2' = b, X_3' = c - b$. In both cases the inputs to the function $h'(\cdot, \cdot)$ are the same. However $\sum_{i=1}^{3} X_i = a + c$, while $\sum_{i=1}^{3} X_i' = a - b + c$, that are in general different (over any field). Therefore such a function $h'(\cdot, \cdot)$ cannot exist [1]. ■

[1]These arguments extend naturally even if we consider encoding over time.

IV. Proof of Theorem 1

We start by giving an overview of our proof. Roughly speaking, our proof for determining the desired network code has three steps. In the first step, we turn our graph G into a graph $\hat{G} = (\hat{V}, \hat{E})$ in which each internal node $v \in \hat{V}$ is of total degree at most three. We refer to such graphs as *structured* graphs. Our efficient reduction follows that appearing in [5], and has the following properties: (a) $\hat{G}$ is acyclic. (b) For every source (terminal) in G there is a corresponding source (terminal) in $\hat{G}$. (c) For any two edge disjoint paths P_1 and P_2 connecting a source terminal pair in G, there exist two (internally) *vertex* disjoint paths in $\hat{G}$ connecting the corresponding source terminal pair. (d) Any feasible network coding solution in $\hat{G}$ can be efficiently turned into a feasible network coding solution in G.

It is not hard to verify that proving Theorem 1 on structured graphs implies a proof for general graphs G as well. Indeed, given a network G satisfying the requirements of Theorem 1, construct the corresponding network $\hat{G}$. By the properties above, $\hat{G}$ also satisfies the requirements of Theorem 1. Assuming that Theorem 1 is proven for structured graphs $\hat{G}$, we conclude the existence of a feasible network code in $\hat{G}$. Finally, this network code can be converted (by property (d) above) into a feasible network code for G as desired.

Due to space limitations, we omit the details of the mapping between G and $\hat{G}$ and the proof of properties (a)-(d) (details can be found in [4]). We note that our reduction and proof are strongly based on that appearing in [5]. For notational reasons, from this point on in the discussion we will assume that our input graph G is structured — which is now clear to be w.l.o.g.

In the second step of our proof, we give edges and vertices in the graph G certain labels depending on the combinatorial structure of G. This step can be viewed as a decomposition of the graph G (both the vertex set and the edge set) into certain *class* sets which may be of interest beyond the context of this work. These classes will later play a major role in our analysis. The decomposition of G is given in detail in Section IV-A.

In the third and final step of our proof, using the labeling above we evoke on a lengthy case analysis for the proof of Theorem 1. Namely, based on the terminology set in Section IV-A, we identify several scenarios, and prove Theorem 1 assuming that they hold. As the different scenarios we consider will cover all possible ones, we conclude our proof. A detailed case analysis is given in Sections IV-B and V.

All in all, as will be evident from the sections yet to come, our proof is constructive, and each of its steps can be done efficiently. This will result in the efficient construction of the desired network code for G. We now proceed to formalize the steps of our proof.

A. The decomposition

In this section we present our structural decomposition of $G = (V, E)$. We assume throughout that G is directed and acyclic, that it has three sources s_1, s_2, s_3, three terminals t_1, t_2, t_3 and that any internal vertex in V (namely, any vertex which is neither a source or a sink) has total degree at most three. Moreover, we assume that G satisfies the connectivity requirements specified in Theorem 1.

We start by labeling the vertices of G. A vertex $v \in V$ is labeled by a pair (c_s, c_t) specifying how many sources (terminals) it is *connected* to. Specifically, $c_s(v)$ equals the number of sources s_i for which there exists a path connecting s_i and v in G. Similarly, $c_t(v)$ equals the number of terminals t_j for which there exists a path connecting v and t_j in G. For example, any source is labeled by the pair $(1, 3)$, and any terminal by the pair $(3, 1)$. An internal vertex v labeled $(\cdot, 1)$ is connected to a single terminal only. This implies that any information leaving v will reach at most a single terminal. Such vertices v play an important role in the definitions to come. This concludes the labeling of V.

An edge $e = (u, v)$ for which v is labeled $(\cdot, 1)$ will be referred to as a *terminal* edge. Namely, any information flowing on e is constrained to reach at most a single terminal. If this terminal is t_j then we will say that e is a t_j-edge. Clearly, the set of t_1-edges is disjoint from the set of t_2-edges (and similarly for any pair of terminals). An edge which is not a terminal edge will be referred to as a *remaining* edge or an r-edge for short.

We now prove some structural properties of the edge sets we have defined. First of all, there exists an ordering of edges in E in which any r-edge comes before any terminal edge, and in addition there is no path from a terminal edge to an r-edge. This is obtained by an appropriate topological order in G. Moreover, for any terminal t_j, the set of t_j-edges form a connected subgraph of G rooted at t_j. To see this note that by definition each t_j-edge e is connected to t_j and all the edges on a path between e and t_j are t_j-edges. Finally, the head of an r-edge is either of type $(\cdot, 2)$ or $(\cdot, 3)$ (as otherwise it would be a terminal edge).

For each terminal t_j we now define a set of vertices referred to as the leaf set L_j of t_j. This definition shall play an important role in our discussions.

Definition 1: Leaf set of a terminal. Let $P = (v_1, v_2, \ldots, v_\ell)$ be a path from s_i to t_j (here $s_i = v_1$ and $t_j = v_\ell$). Consider the intersection of P with the set of t_j-edges, This intersection consists of a subpath P', $(v_P, \ldots, v_\ell = t_j)$ of P for which the label of v_P is either $(\cdot, 2)$ or $(\cdot, 3)$, and the label of any other vertex in P' is $(\cdot, 1)$. We refer to v_P as the leaf of t_j corresponding to path P, and the set of all leaves of t_j as the leaf set L_j.

We remark that (a) the leaf set of t_j is the set of nodes of in-degree 0 in the subgraph consisting of t_j-edges and (b) a source node can be a leaf node for a given terminal. Furthermore, we have the following claim about leaf nodes.

Claim 1: A leaf node which is not a source node has in-degree = 1 and out-degree = 2.

Proof: Assume otherwise, i.e. that the leaf ℓ has out-degree = 1 and suppose that the outgoing edge is denoted (ℓ, v). Note that this implies that $c_t(v) = c_t(\ell) \geq 2$, since ℓ has only one outgoing edge. This is a contradiction since ℓ is a leaf node and has to be connected to at least one node of type $(\cdot, 1)$. Therefore out-degree(ℓ) = 2 and since it is an internal node, it has in-degree = 1. ■

B. Case analysis

We now present a classification of networks based on the node labeling procedure presented above. For each class of networks we shall argue that each terminal can compute the sum of the sources $(X_1 + X_2 + X_3)$. Our proof shall be constructive, i.e. it can be interpreted as an algorithm for finding the network code that allows each terminal to recover $(X_1 + X_2 + X_3)$.

1) Case 0: There exists a node of type $(3, 3)$ in G. Suppose node v is of type $(3, 3)$. This implies that there exist $path(s_i - v)$, for $i = 1, \ldots, 3$ and $path(v - t_j)$, for $j = 1, \ldots, 3$. Here, and in what follows, we denote by $path(u - v)$ a path from u to v. Consider the subgraph induced by these paths and color each edge on $\cup_{i=1}^{3} path(s_i - v)$ red and each edge on $\cup_{j=1}^{3} path(v - t_j)$ blue. We claim that as G is acyclic, at the end of this procedure each edge gets only one color. To see this suppose that a red edge is also colored blue. This implies that it lies on a path from a source to v and a path from v to a terminal, i.e. its existence implies a directed cycle in the graph. Now, we can find an inverted tree that is a subset of the red edges directed into v and similarly a tree rooted at v with t_1, t_2 and t_3 as leaves using the blue edges. Finally, we can compute $(X_1 + X_2 + X_3)$ at v over the red tree and multicast it to t_1, t_2 and t_3 over the blue subgraph. More specifically, one may use an encoding scheme in which internal nodes of the red tree receiving Y_1 and Y_2 send on their outgoing edge the sum $Y_1 + Y_2$.

2) Case 1: There exists a node of type $(2, 3)$ in G. Note that it is sufficient to consider the case when there does not exist a node of type $(3, 3)$ in G. We shall show that this case is equivalent to a two sources, three terminals problem.

Without loss of generality we suppose that there exists a $(2, 3)$ node v that is connected to s_2 and s_3. We color the edges on $path(s_2 - v)$ and $path(s_3 - v)$ blue. Next, consider the set of paths $\cup_{i=1}^{3} path(s_1 - t_i)$. We claim that these paths do not have any intersection with the blue subgraph. This is because the existence of such an intersection would imply that there exists a path between s_1 and v which in turn implies that v would be a $(3, 3)$ node. We can now compute $(X_2 + X_3)$ at v by finding a tree consisting of blue edges that are directed into v. Suppose that the blue edges are removed from G to obtain a graph G'. Since G is directed acyclic, we have that there still exists a path from v to each terminal after the removal. Now, note that (a) G' is a graph such that there exists at least one path from s_1 to each terminal and at least one path from v to each terminal, and (b) v can be considered as a source that contains $(X_2 + X_3)$. Now, G' satisfies the condition given in [8] (which addresses the two sources version of the problem at hand), therefore we are done.

3) Case 2: There exists a node of type $(3, 2)$ in G. As before it suffices to consider the case when there do not exist any $(3, 3)$ or $(2, 3)$ nodes in the graph. Suppose that there exists a $(3, 2)$ node v and without loss of generality assume that it is connected to t_1 and t_2. We consider the subgraph G' induced by the union of the following sets of paths

1) $\cup_{i=1}^{3} path(s_i - v)$,
2) $\cup_{i=1}^{2} path(v - t_i)$, and
3) $\cup_{i=1}^{3} path(s_i - t_3)$.

Note that as argued previously, a subset of edges of $\cup_{i=1}^{3} path(s_i - v)$ can be found so that they form a tree directed into v. For the purposes of this proof, we will assume that this has already been done i.e. the graph $\cup_{i=1}^{3} path(s_i - v)$ is a tree directed into v.

The basic idea of the proof is to show that the paths from the sources to terminal t_3 i.e. $\cup_{i=1}^{3} path(s_i - t_3)$ are such that their overlap with the other paths is very limited. Thus, the entire graph can be decomposed into two parts, one over which the sum is transmitted to t_1 and t_2 and another over which the sum is transmitted to t_3. Towards this end, we have the following two claims.

Claim 2: The path, $path(s_1 - t_3)$ cannot have an intersection with either $path(s_2 - v)$ or $path(s_3 - v)$.

Proof: Suppose that such an intersection occurred at a node v'. Then, it is easy to see that v' is connected to at least two sources and to all three terminals and therefore is a node of type $(2, 3)$, which is a contradiction. ■

In an analogous manner we can see that (a) $path(s_2 - t_3)$ cannot have an intersection with either $path(s_1 - v)$ or $path(s_3 - v)$, and (b) $path(s_3 - t_3)$ cannot have an intersection with either $path(s_1 - v)$ or $path(s_2 - v)$.

Claim 3: The paths, $path(s_1 - t_3), path(s_2 - t_3)$ and $path(s_3 - t_3)$ cannot have an intersection with either $path(v - t_1)$ or $path(v - t_2)$.

Proof: To see this we note that if such an intersection happened, then v would also be connected to t_3 which would imply that v is a $(3, 3)$ node. This is a contradiction. ■

Let v_i be the node closest to v that belongs to both $path(s_i - v)$ and $path(s_i - t_3)$ (notice that v_i may equal s_i but it cannot equal v). Consider the following coding solution on G'. On the paths $path(s_i - v_i)$ send X_i. On the paths $path(v_i - v)$ send information that will allow v to obtain $X_1 + X_2 + X_3$. This can be easily done, as these (latter) paths form a tree into v. Namely, one may use an encoding scheme in which internal nodes receiving Y_1 and Y_2 send on their outgoing edge the sum $Y_1 + Y_2$. By the claims above (and the fact that G' is acyclic) it holds that the information flowing on edges e in the paths $path(v_i - t_3)$ has not been specified by the encoding defined above. Thus, one may send information on the paths $path(v_i - t_3)$ that will allow t_3 to obtain $X_1 + X_2 + X_3$. Here we assume the paths $path(v_i - t_3)$ form a tree into t_3, if this is not the case we may find a subset of edges in these paths with this property. Once more, by the claims above (and the fact that G' is acyclic) it holds that the information flowing on edges e in the paths $path(v - t_1)$ and $path(v - t_2)$ has not been specified (by the encodings above). On these edges we may transmit the sum $X_1 + X_2 + X_3$ present at v.

4) Case 3: There do not exist $(3, 3), (2, 3)$ and $(3, 2)$ nodes in G. Note that thus far we have not utilized the fact that there exist two edge-disjoint paths from each source to each terminal in G. In previous cases, the problem structure that has emerged due to the node labeling, allowed us to communicate

$(X_1 + X_2 + X_3)$ by using just one path between each $s_i - t_j$ pair. However, for the case at hand we will indeed need to use the fact that there exist two paths between each $s_i - t_j$ pair. As we will see, this significantly complicates the analysis. This case is the main technical contribution of our work. However, due to space limitations we are only able to give an outline of our proof. The detailed proof can be found in the full version of the paper [4].

V. Analysis of Case 3: outline

The basic idea of the proof is as follows. We first label each edge in the graph as a t_j-edge or an r-edge. Next we perform a *greedy encoding* vector assignment at every r-edge, i.e. the outgoing edge contain the sum of "largest support" that one can possibly obtain from the input edges. For example, in our greedy encoding, if the input edges contain X_1 and X_2, then the outgoing edge will carry $X_1 + X_2$, and if they carry X_1 and $X_1 + X_2$, the outgoing edge will still carry $X_1 + X_2$. We then examine the state of the leaves of each terminal to see whether each terminal can recover $\sum_{i=1}^{3} X_i$ using the information available on its leaves. If this is the case then we are done, otherwise, we perform a procedure that consists of a sequence of careful modifications to the current encoding vector assignments on the edges so each terminal becomes satisfied, i.e., it can recover $\sum_{i=1}^{3} X_i$ from its leaves.

We first characterize the information available at the leaves of satisfied terminals at the end of our greedy encoding.

Claim 4: A terminal can recover $\sum_{i=1}^{3} X_i$ under the following conditions. i) At least one of the leaves of the terminal is of type $(1,2)$ or $(1,3)$. ii) There exist three leaves of type $(2,2)$ such that one is connected to s_1 and s_2, one to s_2 and s_3 and one to s_1 and s_3.

We are left to consider the case in which the conditions of Claim 4 do not hold. Namely, the case in which a given terminal has only leaves of type $(2,2)$, and does not have leaves containing all three combinations X_1+X_2, X_2+X_3 and X_3+X_1. This may only hold if a given terminal has four $(2,2)$ leaves such that (w.l.o.g.) two of them contain $X_1 + X_2$ and two contain $X_2 + X_3$. In this case it is clear (see discussion in Section III) that there is no way that $\sum_{i=1}^{3} X_i$ can be computed using the information available at the leaves. We shall now outline a sequence of modifications that will eventually result in the terminal being able to compute $\sum_{i=1}^{3} X_i$.

We say that a terminal is unsatisfied if it does not satisfy the conditions of Claim 4. It may be the case that a single terminal, two terminals or all three terminal are unsatisfied after the initial greedy encoding. We suggest two *modification* procedures that modify the initial greedy encoding in order to satisfy the unsatisfied terminals. The two modification procedures are designed to fit an involved case analysis.Due to space limitations, we will not be able to present the modification procedures. As before details can be found in [4].

In a nutshell, Modification 1 is designed to be used when we want to satisfy an unsatisfied terminal, say t_1, *together* with an additional unsatisfied terminal, say t_2, without changing information reaching t_3. For Modification 1 to succeed we need certain connectivity requirements between t_1 and t_2. Modification 2 is designed to be used in the case when we want to satisfy an unsatisfied terminal, say t_1, while preserving the satisfiability of both t_2 and t_3. Hence, if after our initial greedy encoding all three terminals are unsatisfied we may apply Modification 1 to satisfy t_1 and t_2 and then Modification 2 to satisfy t_3. If only two terminals are unsatisfied we may use either Modification 1 alone or use Modification 2 twice depending upon the connectivity of the unsatisfied terminals. Finally, if only a single terminal is unsatisfied we may use Modification 2 alone.

Roughly speaking, Modification 1 and 2 follow a subtle and elaborate case analysis in which, based on the case at hand, certain local changes in the initial greedy encoding are performed. These changes and their effect on the remaining network are then analysed.

VI. Conclusion

In this work we have addressed the network arithmetic problem in the scenario in which the network has three sources and three terminals. We have shown that the characterization obtained in [8] no longer holds for the case in which there are more than two sources and two terminals. For the $3s/3t$ case we show that the network arithmetic problem is efficiently solvable if each source terminal pair is connected by at least two edge disjoint paths.

Several questions remain open. Primarily, is the 2-connectivity condition (between s_i/t_j pairs) necessary or can other combinatorial connectivity requirements characterize the capacity of the network arithmetic problem for the $3s/3t$ case. Secondly, as our proof involves a tedious case analysis it would be very interesting to see a simpler more accessible proof for the 2-connectivity case. Finally, the case of more sources and terminals is completely left open in this work.

References

[1] R. Ahlswede, N. Cai, S.-Y.R. Li, and R. W. Yeung. Network Information Flow. *IEEE Trans. on Info. Th.*, 46, no. 4:1204–1216, 2000.

[2] T. Ho, M. Médard, M. Effros, and R. Koetter. Network Coding for Correlated Sources. In *Conf. on Information Sciences and Systems*, 2004.

[3] R. Koetter and M. Médard. An Algebraic approach to network coding. *IEEE/ACM Trans. on Netw.*, 11, no. 5:782–795, 2003.

[4] M. Langberg and A. Ramamoorthy. Communicating the sum of sources in a 3-sources/3-terminals network. *Manuscript, 2009, available at http://www.openu.ac.il/home/mikel/ISIT09/ISIT09.pdf.*

[5] M. Langberg, A. Sprintson, and J. Bruck. The encoding complexity of network coding. *IEEE Trans. on Info. Th.*, 52, no. 6:2386–2397, 2006.

[6] S.-Y.R. Li, R. W. Yeung, and N. Cai. Linear Network Coding. *IEEE Trans. on Info. Th.*, 49, no. 2:371–381, 2003.

[7] B. K. Rai, B. K. Dey, and A. Karandikar. Some results on communicating the sum of sources over a network. *Manuscript*, 2008.

[8] A. Ramamoorthy. Communicating the sum of sources over a network. In *IEEE Intl. Symposium on Info. Th.*, pages 1646–1650, 2008.

[9] A. Ramamoorthy, K. Jain, P. A. Chou, and M. Effros. Separating Distributed Source Coding from Network Coding. *IEEE Trans. on Info. Th.*, 52:2785–2795, June 2006.

[10] D. Slepian and J. Wolf. Noiseless coding of correlated information sources. *IEEE Trans. on Info. Th.*, 19:471–480, Jul. 1973.

[11] Y. Wu, V. Stanković, Z. Xiong, and S. Y. Kung. On practical design for joint distributed source coding and network coding. In *Proceedings of the First Workshop on Network Coding, Theory and Applications, Riva del Garda, Italy*, 2005.