2009

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

Michael Langberg
Open University of Israel

Aditya Ramamoorthy
Iowa State University, adityar@iastate.edu

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. 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 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. [5] 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 [9] 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. [8] showed that in general separating distributed source coding and network coding is suboptimal except in the case of two sources and two terminals. A practical approach to transmitting correlated sources over a network was considered by Wu et al. [10]. Reference [10] 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 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 an author of this work in [7] 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 [7] 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 [6]. We then turn to obtain encoding schemes for the three source three terminal 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 encoding schemes that allow communication in the $3s/3t$ case. Our main result can be summarized by the following theorem:

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 terminals t_1, t_2, t_3. If there exist two edge disjoint paths between each source/terminal pair, then there exists a linear network

Authors are in alphabetical order.

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 [7] 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.

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 2^m per unit time (we are free to choose m 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

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

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

where the coefficients $a_{j,i}$ and $f_{j,i}$ are from $GF(2^m)$. 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).

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

We now present our counter example to the characterization of [7] 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 (with zero error probability) to all three terminals.

Consider the network shown in Figure 1, with three source nodes and three terminal nodes such that the source nodes observe unit entropy sources X_1, X_2 and X_3 that are also independent. All edges are unit capacity. As showed 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. Therefore such a function $h'(\cdot, \cdot)$ cannot exist[1].

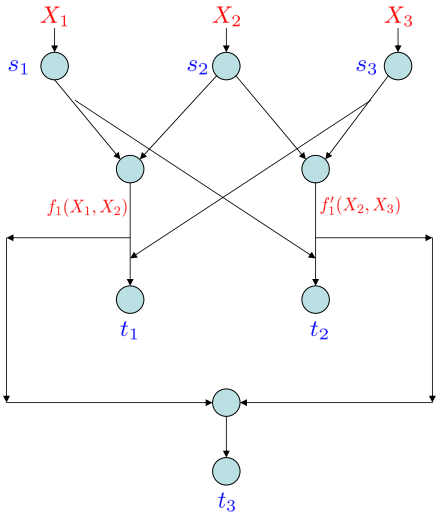

Fig. 1. Example of a network with three sources and three terminals, such that there exists at least one path between each source and each terminal. However all the terminals cannot compute $\sum_{i=1}^{3} X_i$.

[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 [4], and has the additional 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 exists two *vertex* disjoint paths in $\hat{G}$ connecting the corresponding source terminal pair. Here and throughout we say two paths between a source/terminal pair are vertex disjoint even though they share their first and last vertices (i.e., the source and terminal at hand). (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 implied 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 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. We specify the mapping between G and $\hat{G}$ and give proof of properties (a)-(d) in Section IV-A. 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-B.

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

Let $G = (V, E)$ be our input network, and let s_i and t_i be the given sources and terminals. We now efficiently construct a *structured* graph $\hat{G} = (\hat{V}, \hat{E})$ in which each internal node $v \in \hat{V}$ is of total degree three with the additional 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 exists two *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. Our reduction follows that appearing in [4] and is given here for completeness.

The reduction is done iteratively according to the following procedure in which we reduce the total degree of internal vertices to be at most 3. First we note that any source (terminal) in G is also one in $\hat{G}$.

1) Reducing degrees: Let $\hat{G}$ be the graph formed from G by iteratively replacing each node $v \in G$, which is not a source or a terminal node whose degree is more than 3 by a subgraph Γ_v, constructed as follows. Let $\{(x_i, v) \mid i = 1, \ldots, d_{in}(v)\}$ and $\{(v, y_i) \mid i = 1, \ldots, d_{out}(v)\}$ be the incoming and outgoing links of v, respectively, where $d_{in}(v)$ and $d_{out}(v)$ are the in- and out- degrees of v. For each incoming link (x_i, v) of v, we add to Γ_v a node $\hat{x}_i$ and a binary tree X_i with root at $\hat{x}_i$ and $d_{out}(v)$ leaves $\hat{x}_i^1, \ldots, \hat{x}_i^{d_{out}(v)}$. Similarly, for each outgoing link (v, y_i) of v, we add to Γ_v a node $\hat{y}_i$ and an inverted binary tree Y_i with root at $\hat{y}_i$ and $d_{in}(v)$ leaves $\hat{y}_i^1, \ldots, \hat{y}_i^{d_{in}(v)}$. Next, for each $1 \leq i \leq d_{in}(v)$ and $1 \leq j \leq d_{out}(v)$ we add an edge $(\hat{x}_i^j, \hat{y}_j^i)$ to Γ_v. Finally, we connect Γ_v to the rest of the network by adding edges $(x_i, \hat{x}_i)$ for $1 \leq i \leq d_{in}(v)$ and $(y_i, \hat{y}_i)$ for $1 \leq i \leq d_{out}(v)$. Figures 2 and 3 demonstrate the construction of the subgraph Γ_v for a node v with $d_{in}(v) = d_{out}(v) = 3$. Note that for any two links (x_i, v) and (v, y_j) there is a path in Γ_v that connects x_i and y_j.

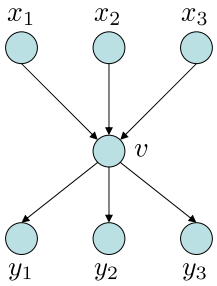

Fig. 2. A node $v \in G$.

We proceed to analyze the properties of $\hat{G}$, namely we show that $\hat{G}$ is structured. The proof of properties (a), (b) and (c) follow directly by our construction. For property (d) consider a feasible network code for the network $\hat{G}$. A feasible network code for G is constructed as follows. Let $e = (u, v)$ be an edge in G. Let e' be the corresponding edge between Γ_u and Γ_v in $\hat{G}$. Here we assume both u and v were replaced by corresponding gadgets. Other cases can be proven analogously. The encoding function f_e for $e = (u, v)$ is determined by the encoding functions $f_{\hat{e}}$ of links $\hat{e}$ that belong to Γ_u. Specifically, let $X = \{(x_1, \hat{x}_1), \ldots, (x_{d_{in}(u)}, \hat{x}_{d_{in}(u)})\}$ be the incoming links of Γ_u where $d_{in}(u)$ is the in-degree of u in G. The

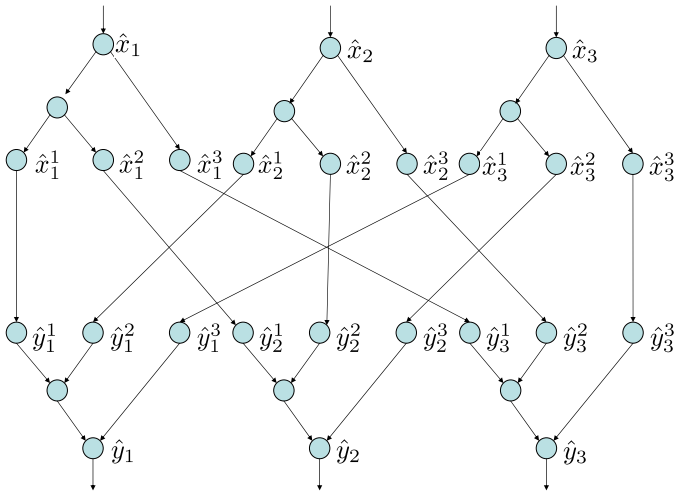

Fig. 3. The gadget Γ_v for v in Figure 2.

construction of $\hat{G}$ implies that the information transmitted on the link e' is a function $f_{e'}$ of the packets transmitted on links X. We use this exact function as the desired encoding function f_e. The fact that the incoming links of u in G correspond to the links in X implies the feasibility of the resulting code for G.

B. 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 with is neither a source or a sink) has total degree at most 3. Moreover, we assume 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 bounded 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 $P = (s_i = v_1, v_2, \ldots, v_\ell = t_j)$ be a path from s_i to t_j. 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. Recall that we assume for each source terminal pair (s_i, t_j) the existence of two (vertex) disjoint paths connecting s_i and t_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. ■

C. 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. they 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 $j = 1, \ldots, 3$. 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 [7] (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. As this case in the main technical contribution of our work, we present it in the upcoming section:

V. Analysis of Case 3

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, using the topological ordering on the edges, we perform *greedy encoding vector assignment* at every r-edge, i.e. the outgoing edge contain the largest sum 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 $X_1 + X_2$. The greedy encoding will be specified in detail shortly (in Section V-A). 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 that at the end of it, each terminal is satisfied i.e. it can recover $\sum_{i=1}^{3} X_i$ from its leaves. Towards this end we first establish some specific properties of the terminal leaves under this case.

Claim 4: Each leaf is of type either $(2,2), (1,2)$ or $(1,3)$.

Proof: A leaf node is of type $(\cdot, 2)$ or $(\cdot, 3)$. Since $(3,3), (2,3)$ and $(3,2)$ nodes are ruled out, the claim follows. ■

Claim 5: There exist at least three leaves for each terminal.

Proof: Recall that there exist two vertex disjoint paths between each source and each terminal. Each such path has a corresponding leaf. For each terminal, different sources may share leaves but there must be two distinct leaves for each source (the corresponding paths are vertex disjoint). By Claim 4 the source label of each leaf can be at most two. This implies that each terminal must have at least three leaves. ■

A. Initial Greedy Encoding

We now specify our initial encoding procedure. We perform our greedy encoding vector assignment on the r-edges ordered in topological order. More formally, we perform the following steps.

i) Suppose the tail of the edge at hand is a node of in-degree 1. The encoding vector on it is a copy of the encoding vector of the incoming edge.

ii) Suppose the tail of the edge at hand is a node of in-degree 2. Let the encoding vectors (of length-3) on the incoming edges be denoted β_1 and β_2 and let supp(β_i) denote the support of β_i. The support of the encoding vector on the outgoing edge is $\cup_{i=1}^{2}$supp(β_i) and every element in the support takes the value 1.

We remark that there are at most two sources connected by a path to the tail of any r-edge. This follows since, otherwise by definition, as the head of an r-edge has terminal label 2 (or 3) this would imply the existence of a $(3,2)$ (or $(3,3)$) node. We conclude that for any r-edge e, the encoding vectors corresponding to its incoming edges span a subspace of dimension at most 2. This will allow (by simple forwarding or addition) to obtain the encoding vector for e specified in ii) above.

We now outline the conditions under which a terminal can recover $\sum_{i=1}^{3} X_i$ from its leaves.

Claim 6: 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 $(1,3)$. Henceforth we refer to such leaves as singleton leaves. 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.

Proof: W.l.og. we assume the terminal to be t_1. We shall demonstrate the existence of encoding vector assignments downstream of the leaves on the t_1-edges that allows the recovery of $\sum_{i=1}^{3} X_i$ at t_1. Suppose that the first condition holds i.e. there exists a t_1-leaf that has source label 1. Note that this can happen in the following ways.

a) There exist three singleton leaves containing each of the X_i's. In this case, each of the leaves has a path consisting of t_1-edges to t_1. We can find a subset of the edges on these paths that form a tree directed into t_1 over which $\sum_{i=1}^{3} X_i$ can be constructed.

b) There exist two singleton leaves containing w.l.o.g. X_1 and X_2 and no singleton leaf for X_3. This implies that there exists at least one other $(2,2)$ leaf connected to X_3. Suppose that the leaf contains $X_1 + X_3$. In this case we form the directed tree into t_1 by considering the paths from this leaf and the leaf containing X_2. The other case can be handled in an identical manner.

c) There exists one singleton leaf containing w.l.o.g X_1 and no singleton leaf for the other sources. It is not hard to verify that t_1 will have leaves connected to X_2 and X_3 as well. This follows directly by our connectivity requirements. If there exists a leaf containing $X_2 + X_3$ then we are done by the approach suggested above. Alternatively there have to exist leaves containing $X_2 + X_1$ and $X_1 + X_3$. In this case once again we form a directed tree rooted into t_1 by finding an appropriate subset of the t_1-edges. Over this tree, we can assign encoding vectors such that t_1 obtains $(-X_1) + (X_2 + X_1) + (X_1 + X_3) = \sum_{i=1}^{3} X_i$.

Likewise, if the leaves of t_1 contain $X_1 + X_2$, $X_2 + X_3$ and $X_1 + X_3$, then we can form a directed tree into t_1 so that it can recover $2\sum_{i=1}^{3} X_i$. Here we need the field to be of characteristic > 2. ■

We say that a source X_i appears at a node if the node contains a sum of the subset of the sources that includes X_i.

Corollary 1: Consider an assignment of coding vectors at every edge of G. Suppose that for a given terminal (w.l.o.g. t_1), (a) each source $X_i, i = 1, \ldots, 3$ appears at least once in the set of t_1-leaves, and (b) at least one source appears as a singleton at one of the t_1-leaves. Then t_1 can compute $\sum_{i=1}^{3} X_i$.

Proof: This follows directly from the proof of Claim 6. ■

We are left to consider the case in which the conditions of Claim 6 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 $X_1 + X_3$. 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 situation it is clear that there is no way that $\sum_{i=1}^{3} X_i$ can be computed using the information available at the leaves (see Section III). 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 6. It may be the case that a single terminal, two terminals or all three terminal are unsatisfied after the initial greedy encoding. In what follows we present two *modification* procedures that modify the initial greedy encoding as

to satisfy unsatisfied terminals. "Modification Procedure 1" is discussed in detail in Section V-B and "Modification Procedure 2" is discussed in detail in Section V-C. The two modification procedures are designed to fit a certain case analysis that will be specified shortly. To present our need for both procedures, for the remainder of this section we assume that all three terminals are unsatisfied after the initial encoding. We will discuss the (easier) cases in which only two terminals are unsatisfied or only a single terminal is unsatisfied at the end of the section. W.l.o.g. we assume that the leaves of t_1 contain only X_1+X_2 and X_2+X_3 (after our greedy encoding process).

B. Modification Procedure 1

As our graph G is structured, note that there exist two node-disjoint paths from s_1 to t_1 that we denote P_1 and P_2. Let ℓ_{11} and ℓ_{12} denote the (2,2) type, t_1-leaves that have $X_1 + X_2$ on these paths (note that the leaves corresponding to these paths cannot hold $X_2 + X_3$). Next perform the following steps.

i) Follow P_1 upwards from ℓ_{11} (i.e. towards s_1) and find the first in-degree 2, out-degree 1 node such that both its incoming edges do not have the encoding vector assignment $[1\ 1\ 0]$. Call this node v_{11}. Repeat the process for P_2 and call the corresponding node v_{12}. We remark that such a v_{11} has to exist since P_1 starts at s_1 and therefore at least one edge on it has a coding vector $[1\ 0\ 0]$.

ii) If v_{11} is downstream of v_{12} or vice versa, call the downstream node the *winner*. More formally, for a fixed topological order on the nodes of G, if the value of v_{11} is greater than that of v_{12} then declare v_{11} as the winner, otherwise v_{12} will be declared as the winner. W.l.o.g. we assume that v_{11} is the winner for our subsequent discussion.

Now, ℓ_{11} is a (2,2) node, which means that it is connected to another terminal distinct from t_1. W.l.o.g. suppose that the other terminal is t_2. Our strategy will be to modify coding vectors downstream of v_{11} such that ℓ_{11} becomes a singleton leaf (containing either X_1 or X_2), and then modifying coding vectors downstream of ℓ_{11} so that at least one leaf of t_2 also becomes a singleton leaf.

We now turn to better understand the leaves of t_2. By our assumption, t_2 is also unsatisfied and since it is connected to ℓ_{11}, this implies that one of its leaves contains $X_1 + X_2$. Therefore its leaves contain one of the following combinations.

i) At least two leaves containing $X_1 + X_2$ and at least two leaves containing $X_2 + X_3$. In this case we say that t_2 *requests* a singleton leaf containing X_1.

ii) At least two leaves containing $X_1 + X_2$ and at least two leaves containing $X_1 + X_3$. In this case t_2 requests a singleton leaf containing X_2.

Loosely speaking, turning a t_2 leaf containing $X_1 + X_2$ into a singleton leaf containing the request of t_2 will allow to satisfy t_2. Next, once we have identified which type of leaf t_2 requests, we perform the following modification to our greedy encoding, that will satisfy both t_1 and t_2. Our modification proceeds according to the cases presented below.

Modification Procedure 1

i) *Case 1.* The path between ℓ_{11} and t_2 intersects P_2. In this case, notice that there could be multiple intersections. However, we can always find a new path between ℓ_{11} and t_2 that intersects a contiguous set of edges on P_2 after which it does not intersect P_2 again. We shall assume that the $path(\ell_{11} - t_2)$ is of this type. Let the first intersection between $path(\ell_{11} - t_2)$ and P_2 occur at node v', and the last at node v''. Observe that v' is necessarily strictly downstream of v_{12}, otherwise this would contradict the fact that v_{11} was the winner in the procedure specified previously. In turn this implies that the coding vectors on the edges between v_{12} and v' are $[1\ 1\ 0]$. It also holds that v'' is equal to or above ℓ_{12}, otherwise ℓ_{12} wouldn't be a t_1-leaf.

   Suppose that t_2 requested an X_1 singleton leaf (the analysis is similar in the other case). Note that the coding vectors at the incoming edges of v_{11} are such that the coding vector on its outgoing edge can be made either $[1\ 0\ 0]$ or $[0\ 1\ 0]$. We change the coding vector on the outgoing edge of v_{11} and on all other edges on the path P_1 between v_{11} and ℓ_{11} to $[0\ 1\ 0]$. In addition, we change the coding vector on $path(\ell_{11} - v')$ to $[0\ 1\ 0]$. We know that at v' the other incoming edge has a coding vector of $[1\ 1\ 0]$. Let ℓ' be the first t_2-leaf on $path(v' - t_2)$. We change the coding vector on $path(v' - \ell')$ to $[1\ 0\ 0]$. Similarly we propagate $[1\ 0\ 0]$ on P_2 between v' and ℓ_{12}.

ii) *Case 2.* The path between ℓ_{11} and t_2 does not have an intersection with P_2. Suppose that t_2 requested an X_1 singleton leaf (the analysis is similar in the other case). Then simply change the coding vectors on path P_1 between v_{11} and ℓ_{11} and between ℓ_{11} and the t_2-leaf on $path(\ell_{11} - t_2)$ to $[1\ 0\ 0]$. We observe that ℓ_{12} continues to receive $[1\ 1\ 0]$ as coding vectors on it are not changed.

At the end of this modification procedure, it may be the case that some coding vector assignments downstream of v_{11} become inconsistent. In this case, we re-perform the greedy encoding step on those edges, while retaining whatever coding vectors we have assigned in the previous step. For example consider an internal node v that initially had two incoming encoding vectors, one of value $[1\ 1\ 0]$ and the other of value $[1\ 0\ 0]$, and an outgoing encoding vector of value $[1\ 1\ 0]$. It might be the case that after our modification procedure it now has incoming encoding vectors, both of value $[1\ 0\ 0]$. The outgoing encoding vector is now inconsistent with the incoming ones and thus must be modified. Our re-encoding (preformed on r-edges only) will follow the greedy procedure outlined previously while we preserve any modification made in "Modification Procedure 1".

Next, we establish that we are in a position where t_1 and t_2 are satisfied. In what follows we refer to Modification Procedure 1 and the re-encoding as "Modification 1".

Claim 7: Any leaf that contained $X_2 + X_3$ or $X_1 + X_3$ before Modification 1, continues to do so at the end of it.

Proof: Note that all modifications are performed downstream of v_{11} which is a $(2,2)$ node since it is connected to ℓ_{11} which is a (2,2) leaf, and since by definition it is connected to both s_1 and s_2. If any leaf containing X_2+X_3 or X_1+X_3 was modified, it must be connected to v_{11}, and thus to s_1 and s_2. However, such a leaf is also connected to s_3. This which would make it a (3,2) node, a contradiction to our assumption that such a node does not exist in G. ■

Claim 8: At the end of Modification 1, both t_1 and t_2 are satisfied.

Proof: Before the procedure was performed, ℓ_{11} and ℓ_{12} were (2,2) nodes containing $X_1 + X_2$. At the end of the procedure, ℓ_{11} becomes a singleton leaf containing either X_1 or X_2. The leaf ℓ_{12} either contains $X_1 + X_2$ or becomes a singleton leaf depending upon whether Case 1 or Case 2 is performed in Modification Procedure 1. If Case 1 occurred then ℓ_{11} and ℓ_{12} obtain distinct sources because of the *flip* that occurs at node v'. In Case 2, ℓ_{12} continues to obtain $X_1 + X_2$. Moreover by Claim 7 above, the other leaves of t_1 containing X_3 are undisturbed. Therefore, we have successfully introduced a singleton t_1-leaf, which implies that it is satisfied (Corollary 1).

The argument for t_2 is a little more subtle. Recall, that in Modification Procedure 1, we considered the "request" of t_2. Thus if it requested X_1, this was because it already had a $X_2 + X_3$ leaf. Now, at the end of the procedure, we have successfully introduced a singleton leaf at t_2 containing X_1 (the "request"). Again, by Claim 7 above, its other leaves containing X_3 are undisturbed. Therefore it can compute $\sum_{i=1}^{3} X_i$ by using $X_2 + X_3$ from the other leaf. ■

Having ensured that t_1 and t_2 are satisfied, we now propose further modifications to ensure that t_3 is satisfied. We start by proving some properties of Modification 1.

Claim 9: Modification 1 does not affect any t_3-leaf.

Proof: Our proof follows the line of proof given in Claim 7. First notice that there cannot be a path between the node v_{11} and any t_3-leaf. This follows from the fact that v_{11} is connected to t_1 and t_2 and to s_1 and s_2. The existence of a path between v_{11} and t_3 would imply that v_{11} is a $(2,3)$ node (connected to t_3 also) in contradiction to our assumption that such nodes are not present in G. As all modifications are performed downstream of v_{11}, and t_3 nodes do not appear downstream of v_{11} we conclude our assertion. ■

In a similar manner we also claim that any coding vector that originally had a 1 in the third component (corresponding to X_3) remains unchanged after Modification 1.

Claim 10: If a coding vector on an edge had a 1 in the third component (corresponding to X_3) originally, it remains unchanged after Modification 1.

Proof: Our proof follows that of Claim 7. All modifications are performed downstream of v_{11} which is connected to X_1 and X_2. Therefore if there is an edge that originally contained a 1 in the third component, it implies that this edge is connected to all the three sources. Next, this edge has to have a terminal label of 2 since we have performed the encoding only on the r-edges. This is a contradiction since it implies the existence of a (3,2) node. ■

C. Modification Procedure 2

We now turn to describe our second modification procedure that will allow to satisfy terminal t_3 while preserving satisfaction of t_1 and t_2. As the information reaching t_3 has not been modified its leaves contain one of the following combinations. i) At least two leaves containing $X_1 + X_2$, at least two leaves containing X_2+X_3 and no other combinations at other leaves. ii) At least two leaves containing $X_1 + X_2$, at least two leaves containing X_1+X_3 and no other combinations at other leaves. iii) At least two leaves containing X_1+X_3, at least two leaves containing X_2+X_3 and no other combinations at other leaves.

We start by considering Case i) above (the remaining cases will be discussed at the end of the section). As before we define the vertices v_{31}, v_{32}, ℓ_{31} and ℓ_{32} as follows. Consider the two node disjoint paths connecting s_3 to t_3, we denote these paths by P_1 and P_2. Let ℓ_{31} and ℓ_{32} denote the (2,2) type t_3-leaves on these paths. It is not hard to verify that they must contain information $X_2 + X_3$. Next, perform the following steps.

i) Follow P_1 upwards from ℓ_{31} (i.e. towards s_3) and find the first in-degree 2, out-degree 1 node such that both its incoming edges do not have the encoding vector assignment $[0\ 1\ 1]$. Call this node v_{31}. Repeat the process for P_2 and call the corresponding node v_{32}.
ii) If v_{31} is downstream of v_{32} or vice versa, call the downstream node the *winner*. W.l.o.g. we assume that v_{31} is the winner for our subsequent discussion.

We now show how to modify the encoding vectors of the network such that *all* terminals will be satisfied. Assume w.l.o.g. that t_2 is the additional terminal connected to v_{31}. We emphasize that this is w.l.o.g since we assume very little on Modification 1 in the analysis to come. Specifically, our proof goes without change if t_1 is the additional terminal connected to v_{31}.

We begin by outlining the basic procedure and providing intuition about it. The purpose of Modification 1 was to ensure that both terminals t_1 and t_2 are satisfied. However, the aim of the current procedure is only to satisfy t_3 while ensuring that t_2 continues to remain satisfied. Towards this end we first examine the structure of the leaves on the two-edge disjoint paths between s_3 and t_2. Depending on the combinations available on the leaves we decide the source symbol that will be propagated on $path(v_{31} - \ell_{31})$ and finally argue that both t_2 and t_3 remain satisfied.

Modification Procedure 2

First consider the two edge-disjoint paths between s_3 and t_2 denoted P_1' and P_2'. Identify the t_2-leaves, ℓ_{21} and ℓ_{22} on these paths and the information contained in them. Note that both leaves necessarily need to contain a component of X_3 (either by itself or as a sum with another source). This is true in the original "greedy" encoding (before Modification 1) and also true after Modification 1 by Claim 7.

i) At least one leaf contains X_1+X_3. We propagate the coding vector $[0\ 1\ 0]$ on $path(v_{31}-\ell_{31})$. Next, we perform the greedy re-encoding step downstream of v_{31} while retaining the coding vectors assigned on $path(v_{31}-\ell_{31})$.

ii) At least one leaf contains just X_3. In this case again, we propagate the coding vector $[0\ 1\ 0]$ on $path(v_{31}-\ell_{31})$ and perform the greedy re-encoding step downstream of v_{31} while retaining the coding vectors assigned on $path(v_{31}-\ell_{31})$.

iii) Both leaves contain X_2+X_3. In this case we need to be careful about how we modify the encoding vectors as there is a possibility that we cause terminal t_2 to stop receiving a particular source. In order to handle this we examine the possible intersections between P_1', P_2' (the edge-disjoint paths between s_3 and t_2) and the $path(v_{31}-\ell_{31})$.

a) If both P_1' and P_2' intersect $path(v_{31}-\ell_{31})$, then perform the following steps.
In general path P_1' (and P_2') can intersect $path(v_{31}-\ell_{31})$ at multiple locations. We say that there is a P_1' type intersection (likewise P_2' type intersection) on a set of edges if these edges are connected and belong to both P_1' and $path(v_{31}-\ell_{31})$.
We claim that there is no loss of generality in assuming that the intersections alternate between the two types when we examine the edges on $path(v_{31}-\ell_{31})$. To see this, note that if this is not the case, we have a situation where two consecutive intersections are of type P_1' (w.l.o.g.). However in this case we can simply modify the path P_1' so these two intersections can be collapsed into one intersection. Thus, we will assume that the intersections alternate.
Next, color the path P_1' red and the path P_2' blue (by convention all other edges are called uncolored). W.l.o.g. we assume that the first intersection (i.e. closest and downstream to v_{31}) is of type P_1'. Then perform the following steps

1) Propagate coding vector $[0\ 1\ 0]$ from v_{31} downwards till the node where the blue path and P_1 first meet. Assign coding vector $[0\ 0\ 1]$ to the outgoing edge. Assign a variable curr-intersection-color = *blue*. Perform greedy re-encoding downstream of all edges that have been modified thus far while retaining the coding vectors assigned on $path(v_{31}-\ell_{31})$.

2) Repeat the following steps (iteratively downstream on the path $path(v_{31}-\ell_{31})$) until all edges on $path(v_{31}-\ell_{31})$ are assigned a new coding vector. For edge $e \in path(v_{31}-\ell_{31})$ that has not been assigned a coding vector do the following.

2a) If e is either uncolored or $color(e)$ = *curr-intersection-color*, then propagate the coding vector of the parent of e that lies on $path(v_{31}-\ell_{31})$.

2b) Otherwise, perform a ***flip*** operation. If the parent of e on $path(v_{31}-\ell_{31})$ had a coding vector $[0\ 0\ 1]$, then e is assigned $[0\ 1\ 0]$ and vice versa. Also ***flip*** the value of *curr-intersection-color* i.e. if it was blue, make it red and vice versa.

2c) Perform greedy re-encoding downstream of all edges that have been modified thus far, without changing the new coding vector assignments on $path(v_{31}-\ell_{31})$.

b) Otherwise, propagate the coding vector $[0\ 1\ 0]$ on $path(v_{31}-\ell_{31})$ and perform the greedy re-encoding step downstream of v_{31} while retaining the coding vectors assigned on $path(v_{31}-\ell_{31})$.

We now prove that at the end of Modification Procedure 2, all terminals are satisfied. Namely, t_3 becomes satisfied and t_1 and t_2 remain satisfied.

For t_1, we follow the line of proof given in Claim 9. Namely, it is not hard to verify that any change in the encoding vector of v_{31} cannot effect the encoding vectors of the leaves of t_1 (otherwise there will be a $(2,3)$ node in G). This implies that t_1 remains satisfied after Modification 2. For t_3, we now show that (after Modification 2) ℓ_{31} receives a singleton (either X_2 or X_3) and ℓ_{32} continues to receive X_2+X_3, this will imply that t_3 is satisfied.

Claim 11: At the end of Modification Procedure 2, t_3 is satisfied.

Proof: Recall that we assume t_3 has leaves with information X_1+X_2 and X_2+X_3. As in the proof of Claim 7 it can be shown that all the leaves that *originally* (in the initial encoding) contain X_1 in any form, will not be modified by Modification 2. This implies that, after Modification 2, t_3 will still receive X_1+X_2. In addition, after Modification 2 leaf ℓ_{31} will receive either the singleton X_2 or X_3. Finally, by our choice of v_{31} there does not exist a path from v_{31} to v_{32}. This implies that the outgoing edge of v_{32} on P_2 continues to carry X_2+X_3 after the procedure. This outgoing edge is connected to ℓ_{32}. Therefore, the greedy re-encoding process ensures that ℓ_{32} receives X_2+X_3. We conclude (using Corollary 1) that t_3 is satisfied after Modification 2. ■

We now address terminal t_2. Our analysis follows the cases outlined in the description of Modification Procedure 2 given above. In each case, we assume that t_2 was satisfied before the modification procedure, and prove that it remains satisfied after the procedure. We first present a general Claim which analyzes the changes in the leaf information of t_2 after Modification Procedure 2.

Claim 12: t_2-leaves receiving X_2+X_3 before Modification Procedure 2 may receive either X_2 or X_3 after the modification. All remaining t_2-leaves receive the exact same information before and after Modification Procedure 2.

Proof: We start by considering the *original* information present at leaves of t_2, namely the information present before Modification Procedure 1. This information satisfies the greedy encoding specified in the beginning of the section. t_2 leaves which before Modification 1 received information including

X_1 in their support, cannot be effected by Modification 2. This follows by arguments similar to those of Claim 7. We are left to consider leaves that originally received the singleton X_2, the singleton X_3 or $X_2 + X_3$. Leaves receiving a singleton X_2 or X_3 cannot be downstream of v_{31} (and thus effected by Modification 2) as in such a case (due to greedy encoding) they would have received X_2+X_3. Leaves receiving X_2+X_3, may indeed be effected by Modification 2. As Modification Procedure 2, starts by either forwarding the encoding vector $[0\ 1\ 0]$ or $[0\ 0\ 1]$ on $path(v_{31} - \ell_{31})$ instead of the original encoding vector $[0\ 1\ 1]$, it is not hard to show by induction that any edge downstream of v_{31} is effected by (at most) *zeroing out* the entry corresponding to either X_2 or X_3 in their encoding vectors. We conclude that the only leaves of t_2 that can be modifies are those that originally received $X_2 + X_3$ in our greedy encoding. By Claim 7 these leaves receive X_2+X_3 after Modification 1 also. ■

Claim 13: If t_2 is unsatisfied after Modification Procedure 2, then after the modification either X_2 or X_3 do not appear in the support of the information appearing in any of the t_2-leaves.

Proof: We assume that t_2 was satisfied before Modification Procedure 2. By Claim 12 the only leaves that may have changed are those that carry $X_2 + X_3$. If after Modification Procedure 2, at least one leaf still carries $X_2 + X_3$ then the total information present at t_2 has not changed and t_2 is still satisfied. If some leaf that previously received $X_2 + X_3$ now receives X_2 and another leaf that was previously receiving $X_2 + X_3$ now receives X_3, then again we claim that t_2 is still satisfied. Indeed, consider the flow of information from the t_2-leaves to terminal t_2 in the satisfying network coding solution (prior to Modification 2), if in this solution t_2 was satisfied by adding information from a leaf containing $X_2 + X_3$, it can now be satisfied by adding the (new) information from the leaves containing X_2 and X_3.

It is left to consider the case in which all leaves that previously received $X_2 + X_3$ now receive (w.l.o.g) X_2 (a single source). Now using Claim 1 we know that if all sources continue to appear at the leaves of t_2, $\sum_{i=1}^{3} X_i$ can be computed at it. Therefore we can conclude that X_3 is not available at the leaves of t_2 after modification procedure 2. ■

Using the Claims above, we now show that during Modification Procedure 2, t_2 remains satisfied. We start with Case i) of Modification Procedure 2:

Claim 14: Assuming Case i) of Modification Procedure 2: at the end of the modification procedure t_2 is satisfied.

Proof: By Claim 13 it suffices to show that X_2 and X_3 appear in the support of the information appearing in any of the t_2-leaves. By Claim 12, the leaf carrying $X_1 + X_3$ is not changed during Modification Procedure 2, and thus has X_3 in its support. If prior to the modification, t_2 had a leaf carrying X_2 or $X_1 + X_2$, then by Claim 12, the leafs would remain unchanged during Modification Procedure 2, and thus t_2 would have X_2 in its support. Finally, as t_2 was satisfied it must have had X_2 somewhere in the support of (the information of) its leaves, and thus we are left to consider the case in which t_2 had a leaf with information X_2+X_3. In this case, by our encoding scheme in Case i) this leaf may either remain unchanged or have information X_2 after Modification Procedure 2 — in both cases X_2 appears in the support. ■

Claim 15: Assuming Case ii) of Modification Procedure 2: at the end of the modification procedure t_2 is satisfied.

Proof: The proof is very similar to Claim 14. By Claim 13 it suffices to show that X_2 and X_3 appear in the support of the information appearing in any of the t_2-leaves. By Claim 12, the leaf carrying X_3 is not changed during Modification Procedure 2, and thus has X_3 in its support. If prior to the modification, t_2 had a leaf carrying X_2 or $X_1 + X_2$, then by Claim 12, the leafs would remain unchanged during Modification Procedure 2, and thus have X_2 in its support. Finally, as t_2 was satisfied it must have has X_2 somewhere in the support of (the information of) its leaves, and thus we are left to consider the case in which t_2 had a leaf with information $X_2 + X_3$. In this case, by our encoding scheme in Case ii) this leaf may either remain unchanged or have information X_2 after Modification Procedure 2 — in both cases X_2 appears in the support. ■

Claim 16: Assuming Case iii) (b) of Modification Procedure 2: at the end of the modification procedure t_2 is satisfied.

Proof: In this case at least one of P_1' or P_2' do not have an intersection with $path(v_{31} - \ell_{31})$. Suppose w.l.o.g. that P_1' is such a path and ℓ_{21} is the corresponding leaf containing $X_2 + X_3$. We will first show that at the end of the procedure, ℓ_{21} continues to receive $X_2 + X_3$ and then use Claim 13. Suppose that none of the edges of P_1' are downstream of v_{31}. In this case it is clear that the modification procedure does not affect ℓ_{21}. In the other case, when there is a path from v_{31} to an edge on P_1' we argue as follows. By Claim 10, we know that the coding vectors on P_1' were not altered at the end of Modification 1. Therefore all coding vectors on edges in P_1' have a 1 in the third component (corresponding to X_3). Next, consider the edge closest to s_3 on P_1' that is downstream of v_{31}. The greedy re-encoding step ensures that the coding vector on this edge is $[0\ 1\ 1]$ since the coding vectors on edges downstream of v_{31} are guaranteed to have a 1 in the second component (corresponding to X_2). Likewise the greedy re-encoding ensures that this coding vector is propagated to ℓ_{21}. ■

It remains to provide the proof of correctness when Case iii)(a) occurs while running Modification Procedure 2. We start by showing that all operations outlined in Steps 1 and 2 in this case are valid.

Claim 17: The coding vector assignment, in Step 1 of Case iii)(a) is valid.

Proof: Denote the edge where the blue path and $path(v_{31} - \ell_{31})$ first meet as b_1. We need to show that the vector $[0\ 0\ 1]$ is in the vector space spanned by the coding vectors of the edges feeding into b_1. According to the algorithm $[0\ 1\ 0]$ is propagated downstream of v_{31} on $path(v_{31} - \ell_{31})$. Therefore we need to show that the coding vector on the blue incoming edge feeding into b_1 is either $[0\ 0\ 1]$ or $[0\ 1\ 1]$. This blue edge is downstream of s_3.

Therefore before the start of the Modification procedure 2, by Claim 10, we know that the coding vector on it will have a 1 in the third component (corresponding to X_3). If this coding vector does not change when $[0\ 1\ 0]$ is propagated downstream of v_{31}, then it is clear that we have the required property. This coding vector will only change if the edge is downstream of v_{31}. However, even in this case the greedy re-encoding property will ensure that the blue edge continues to have a coding vector that has a 1 in the third component. ■

Claim 18: In Case iii)(a) the colored paths are such that between any two contiguous sets of colored edges on $path(v_{31} - \ell_{31})$, we have a contiguous set of uncolored edges.

Proof: Our graphs are such that they have total degree at most three. Recall that, this implies that edge-disjoint paths are also node-disjoint. Since the blue and the red paths under consideration are edge-disjoint, by this property, they are also node-disjoint. Since these paths do not have any node in common, on $path(v_{31} - \ell_{31})$, we cannot have two successive edges of different colors. ■

Claim 19: Step 2b) of Case iii)(a) is valid i.e. the coding vectors on the edges feeding into e are such that the flip operation can take place.

Proof: We shall show this by using induction. Our aim is to prove the following statement. Under the modification procedure, Case iii)(a), in Step 2, for a given edge e, if $color(e) \neq$ curr-intersection-color, then the space spanned by the coding vectors on the incoming edges into e contain the vectors $[0\ 1\ 0]$ and $[0\ 0\ 1]$.

The base case is true by the result of Claim 17. For the induction step, suppose that this is true at the k^{th} intersection at an edge denoted e_k, where the edges feeding into e_k are denoted e_k^c and e_k^{unc} (the superscripts denote colored and uncolored respectively). Note that by Claim 18 we cannot have both edges feeding into e_k to be colored. W.l.o.g. we assume that $e_k^c = blue$. Now consider the closest red edge that is upstream of e_k^{unc} on $path(v_{31} - \ell_{31})$. By Step 2(a) and using the fact that the intersection types alternate, we claim that the coding vector on this red edge is the same as e_k^{unc}, equal to $[0\ 1\ 0]$ (the other case can be handled similarly). Denote the head and tail of this red edge by v_1' and v_2' respectively. Note that v_1' has to be an in-degree 1 and out-degree 2 node since the red path branches off from $path(v_{31} - \ell_{31})$ at this point. Further we note that the outgoing red edge from v_1' is such that its coding vector is $[0\ 1\ 0]$. This implies that the coding vector on the red edges downstream of v_1' before the next intersection with $path(v_{31} - \ell_{31})$ are either $[0\ 1\ 0]$ or $[0\ 1\ 1]$, which means the incoming colored edge at the $k+1$ intersection, e_{k+1}^c has a coding vector either $[0\ 1\ 0]$ or $[0\ 1\ 1]$. Now, by the induction hypothesis, the coding vector assignment $[0\ 0\ 1]$ on e_k is valid. This further means that $e_{k+1}^{unc} = [0\ 0\ 1]$. Thus we have shown that the space spanned by the incoming edges of e_{k+1} contain the vectors $[0\ 1\ 0]$ and $[0\ 0\ 1]$. Therefore we are done. ■

Claim 20: Assuming Case iii) (a) of Modification Procedure 2: at the end of the modification procedure t_2 is satisfied.

Proof: The previous Claims 17 and 19 show that each of the operations are valid. We only need to demonstrate that at the end of the procedure X_2 and X_3 continue to appear in the support of the information available at t_2's leaves. For this we argue in a manner similar to the proof of Claim 19. Let the last intersection (P_1' or P_2' type) with $path(v_{31} - \ell_{31})$ be at edge e_n. W.l.o.g. suppose that e_n is colored blue. We let e_n^c and e_n^{unc} denote the colored and the uncolored edges feeding into e_n. As in the proof of Claim 19, suppose that the coding vector on e_n^{unc} be $[0\ 1\ 0]$. Then we know that the closest red edge upstream of e_n^{unc} also has a coding vector $[0\ 1\ 0]$. This further implies that the t_2-leaf over the red path contains either X_2 or $X_2 + X_3$. Now, after the flip operation, e_n will have the coding vector $[0\ 0\ 1]$ which in turn means that the t_2-leaf over the blue path will contain either X_3 or $X_2 + X_3$. Therefore we have shown that in all cases both X_2 and X_3 continue to appear in the support of the information in t_2's leaves. By Claim 13 we are done. ■

D. Remaining Cases

1) Information of t_3: Throughout Section V-C we considered the case in which t_3 has at least two leaves containing $X_1 + X_2$, at least two leaves containing $X_2 + X_3$ and no other combinations at other leaves. The second case in which at least two leaves contain $X_1 + X_2$ and at least two leaves contain $X_1 + X_3$, is symmetric. The proof for the third case in which at least two leaves contain $X_1 + X_3$ and at least two leaves contain $X_2 + X_3$ slightly differs from that proven in Section V-C. Details follow.

As before we define ℓ_{31} and ℓ_{32}. If the information present at ℓ_{31} and ℓ_{32} equals $X_2 + X_3$ then the proof is identical to that appearing in Section V-C. The same holds if the information present at ℓ_{31} and ℓ_{32} equals $X_1 + X_3$ (as this is a symmetric case). If the information present at ℓ_{31} equals $X_1 + X_3$ and that at ℓ_{32} equals $X_2 + X_3$ (or visa versa) we slightly change Modification Procedure 2 and exclude Case iii a). It is possible to exclude Case iii a) for the following reasons: (a) Using the notation of Section V-C, it is not hard to verify that there will not be any path connecting v_{31} to ℓ_{32} (or visa versa) as such a path would imply a $(3, 2)$ node. Thus we may define either v_{31} or v_{32} to be the "winner". (b) It now holds that either at most one path P_i' intersects $path(v_{31} - \ell_{31})$ or at most one path P_i' intersects $path(v_{32} - \ell_{32})$. Again, otherwise there would be a path connecting v_{31} to ℓ_{32} (or visa versa). After the exclusion above, we note that in Claim 11, t_3 is guaranteed to receive a singleton and have the information of all sources at the support of its leaves (and thus will be satisfied).

2) Number of unsatisfied terminals after initial greedy encoding: As mentioned in Section V-A, it may be the case that a single terminal, two terminals or all three terminal are unsatisfied after the initial greedy encoding. In our presentation we assumed all terminals were satisfied, and proved using both Modification Procedure 1 and 2 that we may modify the greedy encoding as to satisfy all terminals. The reader may have noticed that Modification 1 can be used when we would like to satisfy an unsatisfied terminal, say t_1, *together* with an additional unsatisfied terminal, say t_2, under the restriction that t_2 is connected to the leaf corresponding to

the "winner" among v_{11} and v_{12}. In the discussion to follow, we refer to such t_2 as the terminal *corresponding* to t_1. Recall that Modification 1 does not alter the information present at the remaining t_3. In addition, we associate Modification 1 with two sources (say X_1 and X_2) and we do not change information on edges including the remaining source (X_3) in their support.

Modification 2 can be used to satisfy a single unsatisfied terminal, say t_3. We use Modification 2 on an encoding function that may differ from the original initial greedy encoding. Nevertheless, it is not hard to verify that Modification 2 will succeed if for t_3 and some source, say X_3, the greedy encoding is *preserved*. Namely, if any edge that contained X_3 in its support after the greedy encoding will have exactly the same information in the encoding considered prior to Modification 2; and if all leaves of t_3 contain the exact same information they contained in the initial greedy encoding. We refer to such an encoding as a (t_3, X_3)-preserved encoding. As in Modification 1, there is a terminal that corresponds to the terminal t_3 being considered (in Section V-C this terminal was t_2). The information present at the leaves of the third terminal (that is neither t_3 or the terminal that corresponds to t_3) is not changed during Modification 2. In addition, we associate Modification 2 with two sources (say X_2 and X_3) and we do not change information on edges including the remaining source (X_1) in their support.

Now, if only once terminal, say t_1, is unsatisfied after the greedy encoding, then all we need to do is preform Modification 2. Notice that trivially the initial greedy encoding is (t_i, X_j)-preserved for all i and j. If two terminals, say t_1 and t_2 are unsatisfied after the initial greedy encoding, then we consider two cases. If t_2 corresponds to t_1 then we may perform Modification 1 and satisfy both of them without changing the information present at t_3. If t_2 does not correspond to t_3, then we will preform Modification 2 twice. First we perform Modification 2 on t_1. As t_2 does not correspond to t_1 (in both Modification 1 and 2 correspondence is defined in an equivalent manner), it holds that after the modification process there is a source s_j for which the resulting encoding is (t_2, X_j)-preserved. Moreover, after the modification process both t_1 and t_3 are satisfied. Now applying Modification 2 on t_2 is possible, and after its completion all terminals are satisfied.

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

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

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

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

[8] 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.

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

[10] 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.