

Bounds on Linear Codes for Network Multicast

Ami Tavory[†], Meir Feder, and Dana Ron

Dept. of EE - Systems

Tel-Aviv University

Tel Aviv, Israel

E-mail: {atavory, meir, danar}@eng.tau.ac.il

Abstract

Traditionally, communication networks are composed of *routing* nodes, which relay and duplicate data. Work in recent years has shown that for the case of multicast, an improvement in both rate and code-construction complexity can be gained by replacing these routing nodes by *linear coding* nodes. These nodes transmit linear combinations of the inputs transmitted to them.

In this work, we deal with bounds on the alphabet size and amount of coding necessary for linear codes for multicast. We show that known bounds on maximum distance separable codes can be applied to bound the required alphabet-size. We show upper bounds on the number of “clashes” between flows from source to terminals. Using this, we show upper bounds on the number of nodes in which coding must be performed, and graph-specific upper bounds on the alphabet-size. We show how the addition of a small amount of memory to internal nodes can be used to increase the effective alphabet-size available for coding, and show bounds on the throughput and latency of this technique. Finally, we show that the above bounds also pertain to a case less considered in previous network-multicast work, *static broadcast*, wherein the source transmits the same set of data to terminals at different rates.

Index Terms

Network Information-Flow, Multicast, Alphabet Size, Bounds, Static Broadcast.

I. INTRODUCTION

Work in recent years (*e.g.*, [1], [2], [3], and [4], among others) has shown that for the case of network multicast, linear coding can increase transfer rate and decrease code construction complexity. In this paper, we consider various bounds on the alphabet size and amount of coding necessary for such coding.

We show that some graphs have the property that a rate-achieving multicast code for the graph, reduces to the requirement that processes at terminal nodes can decode, from any subset of a set of values carried by some edges, the data multicast by the source. Effectively, a set of values carried by some edges is thus a *maximum distance separable* (MDS) code. We then use known bounds on MDS to bound the required alphabet-size.

Linear multicast codes are affected by the number of “clashes” between flows from the source to the terminals. We find upper bounds on the number of these clashes. Using these bounds, we find graph-specific upper bounds on the required alphabet size, and the number of nodes in which coding must be performed.

It is possible that the alphabet size supported by the network is not large enough for network coding. We show how the addition of a small amount of memory to internal nodes can be used to increase the effective alphabet-size available for coding, and find bounds on the throughput and latency of this technique.

Finally, we show that the above bounds also pertain to a case less considered in previous network-multicast work, *static broadcast*, wherein the source transmits the same set of data to terminals at different rates. Intuitively, the larger the min-cut from a source to a terminal, the lesser the time it should have to wait to receive all of the data. We show a simple scheme for this case, which is asymptotically optimal.

[†] Also at IBM’s Haifa Research Labs.

A. Paper Layout

We continue the introduction with definitions and notations in Subsection I-B, a brief review of the transmission scheme in Subsection I-C, and a short review of related work in Subsection I-D. In Section II we show an alphabet-size lower-bound. In Section III we consider “clashes” between the flows from the source to the terminals. In Section IV we show upper bounds on the number of such clashes. In Section V we discuss the technique of simulating a large alphabet size by adding node memory, and show bounds on the throughput and latency. In Section VI we discuss graph-specific upper bounds on the alphabet size. In Section VII we discuss *static broadcast*. We conclude in Section VIII.

B. Definitions and Notations.

We consider a network over an acyclic, directed, graph $G = (V, E)$, where parallel edges are allowed. We denote the number of nodes in G by, $n = |V|$. When there is no ambiguity, we denote an edge between nodes $u, v \in V$ by (u, v) . For any node $v \in V$, we use $\mathbf{E}^-(v)$ and $\mathbf{E}^+(v)$ to denote the set of edges reaching and leaving node v , respectively. For any edge $e \in E$, we use $\mathbf{v}^-(e)$ and $\mathbf{v}^+(e)$ to denote the tail and head of e , respectively.

Node $s \in V$ is the *source* node, $T \subseteq V$ is the set of *terminal* nodes, $d = |T|$ is the number of terminal nodes, and h is the minimum, taken over all $t \in T$, of the size of the minimum cut separating s from t . The capacity of a link (*i.e.*, edge) $e \in E$, is a single symbol from a field \mathcal{F} , with size $q = |\mathcal{F}|$, and the value carried by the link is denoted by $y(e)$. A process at node s observes a random process X with entropy¹ $H(X) = h \cdot \log(q)$ per time unit, and wishes to transmit the information of this process to all $t \in T$.

In some of the work, we explicitly consider a *sequential* setting, in which, for some $N \gg \max\{n, q, h\}$, s transmits the values which X attains in N consecutive time units. In this setting, we sometimes explicitly consider a different entropy rate for X .

C. The Transmission Scheme

The transmission scheme in linear multicast was described in [2], [3], and [4], and we repeat the description here in brief.

We introduce a virtual vertex $s' \notin V$, and h parallel virtual edges from s' to s , $e_1^{s'}, \dots, e_h^{s'}$ (this is done for notational convenience throughout the paper). The values carried by these h edges, $y(e_1^{s'}), \dots, y(e_h^{s'})$, are any h symbols from \mathcal{F} which describe the process X . The code is determined by means of a set of auxiliary functions

$$m_e : \mathbf{E}^-(\mathbf{v}^-(e)) \rightarrow \mathcal{F}, \quad (e \in E). \quad (1)$$

The determination of m_e will be described in Section III. For any edge $e \in E$ s.t. $y(e')$ has been determined for all $e' \in \mathbf{E}^-(\mathbf{v}^-(e))$, the value carried by e is

$$y(e) = \sum_{e' \in \mathbf{E}^-(\mathbf{v}^-(e))} m_e(e') \cdot y(e'). \quad (2)$$

We call a code *good*, if for any $t \in T$, there is a subset $E_t \subseteq \mathbf{E}^-(t)$ s.t. $y(e_1^{s'}), \dots, y(e_h^{s'})$ can be reconstructed from $\{y(e)\}_{e \in E_t}$

D. Related Work

Ahlswede *et. al.* [1] have shown that a source can multicast information at a rate approaching h to all terminals, as the symbol size approaches infinity. Li *et. al.* [4] constructively showed that linear coding can be used for multicast with rate h and finite symbol-size. Koeter and Medard [2] showed, through an algebraic framework for network-coding which they developed, that a finite field of size² $O(d \cdot h)$ is sufficient for rate h multicast, and showed how to verify the validity of a given code. Sanders *et al.* [3] showed the first polynomial-time algorithms for constructing linear codes for multicast, and showed that a field of size $O(d)$ is sufficient for this. Our work is an extension of [3], and utilizes many of its ideas and notations.

¹We use $H(\cdot)$ and $\log(\cdot)$ to denote the binary entropy function and logarithm to base 2, respectively.

²We use the order of growth notations O and o as described in [5].

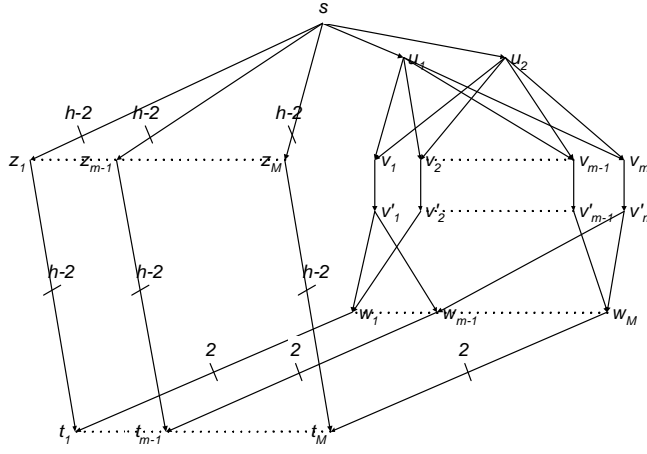


Fig. 1. Graph construction for the alphabet-size lower-bound.

II. ALPHABET-SIZE LOWER-BOUND

In this section we prove a lower bound on the alphabet size. This proof holds for both linear and nonlinear codes.

Theorem 1: For some graphs, the alphabet size must obey

$$q \geq \sqrt{2d} \cdot (1 - o(1)). \quad (3)$$

We prove the lower bound by constructing the graph G , which is shown in Figure 1. In this graph, let m be a number which we will later specify, and let $M = \binom{m}{2}$.

The graph is constructed as follows. Let³ $i \in [m], j \in [M]$, and $k \in [2]$, be arbitrary indices. The source s is connected to nodes u_1 and u_2 via a single link to each. Both u_1 and u_2 are connected to all nodes v_i . Each node v_i is connected to a corresponding node v'_i via a single link. For each pair of nodes in $\{v'_1, \dots, v'_m\}$, there is a node w_j , with links from the nodes of the pair to w_j . Each node w_j is connected to a terminal t_j via 2 links. The source is connected to each of z_j via $h - 2$ links. Each node z_j is connected to t_j via $h - 2$ links.

Clearly, this graph is cycle free, and there is a min-cut of capacity h between s and each of t_j . By the Max-Flow Min-Cut theorem, h symbols can be sent from s to any terminal t_j individually. It follows from [1], [2], [3], [4] that s can multicast h symbols of information to all t_j simultaneously.

We define some random processes observed at some of the graph's nodes.

$$\begin{aligned} Z_j &= \{y(e) \mid e \in \mathbf{E}^-(z_j)\}, & j \in [M], \\ U_k &= \{y((s, u_k))\}, & k \in [2], \\ V'_i &= \{y(e) \mid e \in \mathbf{E}^-(v'_i)\}, & i \in [m], \\ W_j &= \{y(e) \mid e \in \mathbf{E}^-(w_j)\}, & j \in [M], \\ T_j &= \{y(e) \mid e \in \mathbf{E}^-(t_j)\}, & j \in [M]. \end{aligned} \quad (4)$$

Lemma 1: For $j \in [M]$, the entropies of the processes defined in (4), satisfy

$$H(U_1, U_2 | Z_j, W_j) = 0 \quad (5)$$

$$H(U_1, U_2 | Z_j) = 2 \cdot \log(q). \quad (6)$$

Proof: From the Max-Flow Min-Cut theorem, we clearly have that

$$H(U_k) \leq \log(q), \quad (7)$$

$$H(Z_i) \leq (h - 2) \cdot \log(q). \quad (8)$$

³For any ℓ , we use $[\ell]$ to denote $\{1, \dots, \ell\}$.

Also,

$$\begin{aligned}
h \cdot \log(q) &\stackrel{(a)}{=} H(T_j) & (9) \\
&\stackrel{(b)}{\leq} H(Z_j, W_j) \\
&\stackrel{(c)}{\leq} H(Z_j, U_1, U_2) \\
&= H(Z_j) + H(U_1, U_2 | Z_j) \\
&\stackrel{(d)}{\leq} H(Z_j) + H(U_1) + H(U_2) \\
&\stackrel{(e)}{\leq} h \cdot \log(q),
\end{aligned}$$

where (a) follows from the fact that the process at t_j can reconstruct the information of X , (b) and (c) follow from the data-processing inequality [6], (d) follows from the independence bound on entropy [6], and (e) follows from (7) and (8).

Combining (7), (8), and (9), we obtain

$$H(Z_j, W_j) = h \cdot \log(q), \quad (10)$$

$$H(Z_j, U_1, U_2) = h \cdot \log(q), \quad (11)$$

$$H(Z_j) + H(U_1, U_2 | Z_j) = h \cdot \log(q). \quad (12)$$

We obtain (6) from (8) and (12). To obtain (5), note that

$$\begin{aligned}
&H(U_1, U_2 | W_j, Z_j) + H(Z_j, W_j) & (13) \\
&= H(W_j, Z_j, U_1, U_2) \\
&= H(Z_j, U_1, U_2) + H(W_j | Z_j, U_1, U_2) \\
&= H(Z_j, U_1, U_2)
\end{aligned}$$

$$\begin{aligned}
&\stackrel{(a)}{\Rightarrow} \\
&H(U_1, U_2 | W_j, Z_j) = 0 & (14)
\end{aligned}$$

where (a) follows from (10), and (11). ■

We now prove Theorem 1:

Proof: Let

$$\begin{aligned}
E^z &= \{e \mid \mathbf{v}^-(e) = s \wedge \mathbf{v}^+(e) = z_j\} \\
E^{v'} &= \{e \mid \mathbf{v}^-(e) = v_i \wedge \mathbf{v}^+(e) = v'_i\}.
\end{aligned} \quad (15)$$

Consider any fixed values of $y(e)$, $e \in E^z$. It follows from Lemma 1 that $y(s, u_1), y(s, u_2)$ obtain all values from $\mathcal{F} \times \mathcal{F}$ (i.e., (6)), and that given any $\{e_1, e_2\} \subseteq E^{v'}$, $(y(e_1), y(e_2))$ is sufficient for determining $(y(s, u_1), y(s, u_2))$ (i.e., 5). Thus, $y(e')$, $e' \in E^{v'}$, is effectively a *maximal distance separable* (MDS) code. By known bounds on MDS codes [7],

$$m = |E^{v'}| \leq q \cdot (1 + o(1)), \quad (16)$$

and so

$$d = M \leq \binom{q \cdot (1 + o(1))}{2}. \quad (17)$$

■

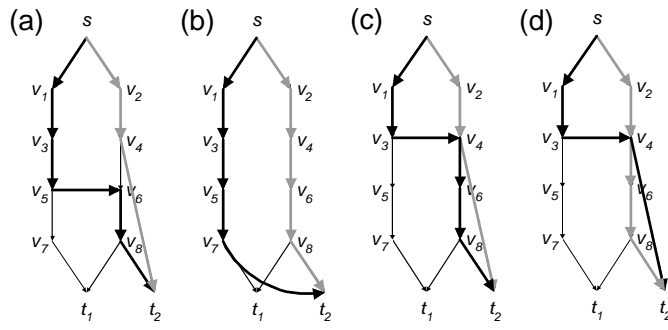


Fig. 2. Graph constructions illustrating flow clashes.

III. CLASH SETS

Consider the four diagrams in Figure 2. They illustrate four cases in which multicast is possible, since there is a rate-2 flow from s to t_1 and t_2 . In each of the diagrams, the rate-2 flow from s to t_1 runs along two paths: $s \rightarrow v_1 \rightarrow v_3 \rightarrow v_5 \rightarrow v_7 \rightarrow t_1$, and $s \rightarrow v_2 \rightarrow v_4 \rightarrow v_6 \rightarrow v_8 \rightarrow t_1$. The paths through which there is a rate-2 flow from s to t_2 are shown in thick black and gray arrows.

In terms of coding, network (a) requires coding, whereas networks (b), (c), and (d) do not. In terms of flows from s to t_1 and t_2 , the flow paths in (a) “clash” in the sense that flows from the source to different terminals “merge” into a single edge (namely the edge (v_6, v_8)), whereas those in (b), (c), and (d) do not. This hints that the amount of network coding, and through it the alphabet size, depends on the number of “clashes”.

In Subsection III-A we repeat, in brief, the determination of the auxiliary functions m_e using *global coding vectors* [3]. In Subsection III-B we define the *clash set* and *minimal clash set*, two sets which are relevant for the determination of m_e . We will use these sets in Sections IV, V, VI, and VII.

A. Global Coding Vectors

In this subsection we repeat, in brief, the determination of the auxiliary functions m_e using *global coding vectors* [3].

For any terminal $t \in T$, let $G_t^f = (V_t^f, E_t^f)$ be a subgraph of G composed of h edge-disjoint paths from s' to t . A subgraph G_t^f can be constructed from G by running any appropriate flow algorithm (see [5]). For any $e \in E_t^f \setminus \{e_1^{s'}, \dots, e_h^{s'}\}$, let $f_t^-(e)$ denote the immediate-predecessor edge of e on the path in G_t^f containing e . As an extension, for any $e \in E_t^f \setminus \{e_1^{s'}, \dots, e_h^{s'}\}$ and any $T' \subseteq T$, let $f_{T'}^-(e)$ denote $\bigcup_{t' \in T'} f_{t'}^-(e)$.

We associate with each edge $e \in (\bigcup_{t \in T} E_t^f)$ a *global coding vector*⁴ $\underline{b}(e) \in \mathcal{F}^h$. We first set the global coding vectors for $e_k^{s'}$, ($k \in [h]$), as

$$\underline{b}(e_k^{s'}) = [\underbrace{0, \dots, 0}_{k-1}, 1, \underbrace{0, \dots, 0}_{h-k}]. \quad (18)$$

The global coding vector of any other edge $e \in \bigcup_{i=1, \dots, d} G_{t_i}^f$ is set so that it satisfies the invariant

$$\underline{b}(e) = \sum_{e' \in E^-(v^-(e))} m_e(e') \cdot \underline{b}(e'). \quad (19)$$

Assume the algorithm terminates after ℓ steps. We generate $\ell \cdot d$ edge-sets, $C_{t_1}^1, \dots, C_{t_d}^1, \dots, C_{t_1}^\ell, \dots, C_{t_d}^\ell$. Let $j \in [\ell]$ be an arbitrary step of the algorithm, and let $i \in [d]$ be the index of an arbitrary terminal node. Each $C_{t_i}^j$ is composed of h edges, one from each path between s and t_i in $G_{t_i}^f$. Initially, we set $C_{t_1}^1 = \dots = C_{t_d}^1 = \{e_1^{s'}, \dots, e_h^{s'}\}$.

⁴We use \underline{x} to denote a row vector, and \underline{x}^+ to denote its transpose. For a vector $\underline{x} = [x_1, \dots, x_\ell]$, we use $\text{len}(\underline{x})$ to denote ℓ . For any $i \in [\text{len}(\underline{x})]$, we use $x[i]$ to denote the element at the i th coordinate of \underline{x} .

At the termination of the algorithm, we have $C_{t_i}^m \subseteq E^-(t_i)$. At step j an edge e_j is chosen so that, for some i , $e_j \in G_{t_i}^f$ and $f_{t_i}^-(e_j) \in C_{t_i}^{j-1}$. Then, m_e is set according to an invariant shown below (see Lemma 2), and through it, $\underline{b}(e)$. Step j is completed by assigning

$$C_t^j = \begin{cases} C_t^{j-1} & , f_t^-(e_j) \notin C_t^{j-1} \\ (C_t^{j-1} \setminus f_t^-(e_j)) \cup \{e_j\} & , f_t^-(e_j) \in C_t^{j-1} \end{cases} . \quad (20)$$

The following lemma was proved in [3].

Lemma 2: Let m_e be set with the invariant that for $j \in [\ell]$, and $i \in [d]$,

$$\text{Rank} \left(\left\{ \underline{b}(e) \mid e \in C_{t_i}^j \right\} \right) = h. \quad (21)$$

Then the code is good.

B. Clash Sets and Minimum Clash Sets

In this subsection, we define two sets which are relevant for the determination of m_{e_j} , where e_j is the edge chosen in step j .

Definition 1: (Clash Set and Nontrivial Clash Set) Let $e_j \in E$ be an arbitrary edge.

The *clash set* (CS) of e_j is the set of terminals whose flows pass through e_j , i.e.,

$$T'(e_j) = \{t \mid e_j \in G_t^f\} \subseteq T. \quad (22)$$

We call the clash set a *nontrivial* CS (NCS) iff at least two distinct flows merge into e_j , i.e., $\left| f_{T'(e_j)}^-(e_j) \right| \geq 2$.

Since we assume that G is an acyclic graph, there is a natural order that can be imposed on any edge $e \in E$. It follows that e_j , can be chosen so that it satisfies

$$\forall t' \in T'(e_j) f_{t'}^-(e_j) \in C_{t'}^{j-1}, \quad (23)$$

i.e., e_j is incorporated in a single step into to all edge sets which will eventually incorporate it. In the remainder of the section, we assume, without loss of generality, that (23) holds.

The next definition defines a subset of a CS, which is minimum in the sense that it suffices to check condition (21) in Lemma 2 in relation to it. Example 1 illustrates this definition.

Definition 2: (Minimum Clash-Set) Let $e_j \in E$ and $T'(e_j) \subseteq T$ be an edge and its CS, respectively. The *minimum clash-set* (MCS) of e_j , $T''(e_j)$, is a minimum subset $T''(e_j) \subseteq T'(e_j)$ satisfying

$$\begin{aligned} & \forall t' \in T'(e_j) \setminus T''(e_j) \exists t'' \in T''(e_j) \forall e' \in C_{t'}^j \setminus f_{t'}^-(e_j) \\ & \text{Rank} \left(\left\{ \underline{b}(e') \right\} \cup \left\{ \underline{b}(e'') \mid e'' \in C_{t''}^j \setminus f_{t''}^-(e_j) \right\} \right) = \\ & h - 1. \end{aligned} \quad (24)$$

Example 1: Figure 3 shows an example of a CS and an MCS. Diagram (a) shows the graph, and diagram (b) shows the global coding vectors associated with e'_1 , e'_2 , and e'_3 . As shown in part (c) of the figure, the CS of e_j is $T'(e_j) = \{t_1, t_2, t_3\}$. The MCS of e_j is $T''(e_j) = \{t_1, t_2\}$, since $\underline{b}(e'_1)$ and $\underline{b}(e'_2)$ are not linearly dependent, but $\underline{b}(e'_3)$ is linearly dependent on $\underline{b}(e'_1)$. □

Clearly, the size of any MCS is not larger than the possible number of ways to group the distinct global coding vectors into groups of size $h - 1$ each. We will make use of the following Lemma in the determination of an upper bound on the alphabet size.

Lemma 3: At step j , the size of any MCS is at most

$$\binom{\left| \left\{ \underline{b}(e) \mid \exists t \in T e \in C_t^{j-1} \right\} \right|}{h - 1}. \quad (25)$$

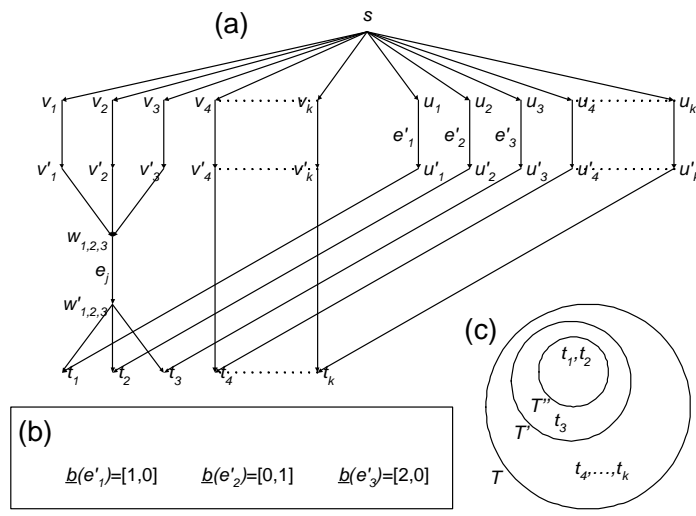


Fig. 3. Example of a CS and a MCS.

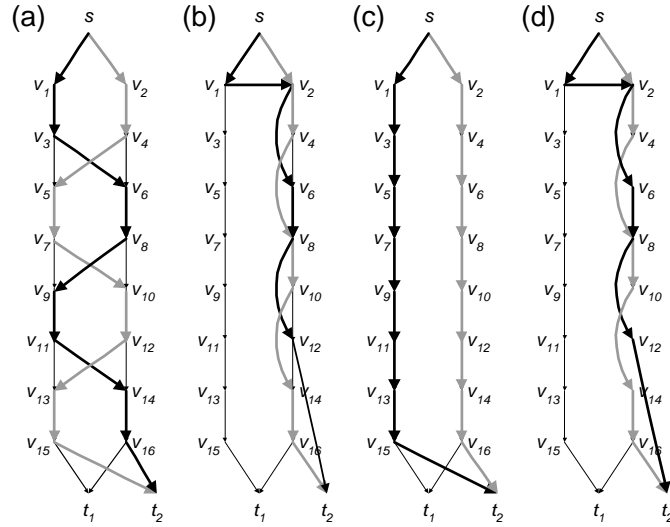


Fig. 4. Examples of disentangling for the case $d = h = 2$.

IV. MINIMUM NUMBER OF NONTRIVIAL CLASH-SETS

Consider again the four networks depicted in Figure 2. Out of all the nodes in all the networks, node v_6 in network (a) is the only one requiring coding, since it is the only one with an NCS.

The question can be asked whether graphs can be built, whose minimal number of NCSs can be made arbitrarily large. In Figure 4 we see two other graphs with flows for the case $d = h = 2$. Recall, from Subsection III-B, that the algorithm from [3] begins by finding two flows, one from s to t_1 , and one from s to t_2 . The flows in this case are more complicated, in the sense that the number of edges which have NCSs is larger than those in Figure 2. However, the paths can be “disentangled”, resulting in the corresponding flows in diagrams (c) and (d) containing flows which are isomorphic to those in diagrams (b) and (a) in Figure 2.

In this section we generalize this observation. As we will show in the following sections, the number of NCSs determines the number of nodes in which coding must be performed, the alphabet size, and the transmission latency for some settings. Ideally, we would like to show that the required number of NCSs is bounded by a function which is determined by the number of terminals and the min-cut of each terminal, and does not grow indefinitely with the graph’s size. We show that this is so for the case of an arbitrary number of terminals, all of which except one has a min-cut of 2. We also show that if this is so for the case of 2 terminals, each with arbitrary min-cuts, then

this is so for the general case as well.

The remainder of the section is organized as follows. In Subsection IV-A we define a “disentangling” transformation, and state precisely the main results of this section. In Subsection IV-B we discuss its application to the case $d = 2$, with one of the flows having an arbitrary rate, and the other having rate 1. In Subsection IV-C we discuss its application to the case $d = 2$, with one of the flows having an arbitrary rate, and the other having rate 2. For both these cases, we show that any two such flows can be disentangled s.t. the number of NCSs is bounded by a function which is dependent on the rates alone, and *not* on the graph size and topology. While we conjecture that this is true for the case $d = 2$, with each flow having an arbitrary rate, we have not proved this. In Subsection IV-D we discuss, for arbitrary rates, the relationship between the case $d = 2$ and the case of an arbitrary d .

It should be noted that minimizing the number of NCSs is not the same as minimizing the number of edges shared between different flows. As noted in [3], the latter problem is equivalent to some problems in Steiner trees [8], [9]. Approximation schemes for these problems can be found in [8], [10], [11].

A. Disentangling

Before presenting our main result in Theorem 2, we define a *disentangling* operation, which takes as input flows, and outputs corresponding flows, whose number of nontrivial clashes is not larger than that of the original flows.

We define precisely the operands of disentangling. Example 2 illustrates the next definition.

Definition 3: (Path set and flow vector) Let $P = \{e_1, \dots, e_k\} \subseteq E$ be an *ordered* set of edges. We say that P is a *path set* (PS) between $v^-(e_1)$ and $v^+(e_k)$, if for $i \in [k-1]$, $e_i \in \mathbf{E}^-(v^-(e_{i+1}))$. We will also explicitly consider an *empty* PS, which is a path set containing no edges, *i.e.*, $P = \emptyset$.

We say that \underline{R} is a *flow vector* (FV), if it is a vector of edge-disjoint path sets.

Let $\underline{R}^1, \dots, \underline{R}^{d'}$ be FVs. We denote by a length- d' vector,

$$\underline{c} = \underline{c}(\underline{R}^1, \dots, \underline{R}^{d'}) = \left[\text{len}(\underline{R}^1), \dots, \text{len}(\underline{R}^{d'}) \right], \quad (26)$$

their *rate vector*. We denote by $\alpha = \alpha(\underline{R}^1, \dots, \underline{R}^{d'})$ the number of NCSs between all the PSs of the FVs.

Example 2: In diagram (b) of Figure 6, $\{(v_1, v_2), (v_2, v_3), \dots, (v_8, v_9)\}$ forms a PS P^1 between v_1 and v_9 . In diagrams (c) and (d) Figure 6, let

$$\begin{aligned} P_1^1 &= \{(v_1, v_4), (v_4, v_7), \dots, (v_{22}, v_{25})\} \\ P_2^1 &= \{(v_2, v_5), (v_5, v_8), \dots, (v_{23}, v_{26})\} \\ P_3^1 &= \{(v_3, v_6), (v_6, v_9), \dots, (v_{24}, v_{27})\}, \end{aligned} \quad (27)$$

be path sets, and let P_2 be the path identified by thick black arrows. The vector $\underline{R}^1 = [P_1^1, P_2^1, P_3^1]$, is FV of three path sets. The vector $\underline{R}^2 = [P_2]$, is a FV of one PS. The rate vector of $[\underline{R}^1, \underline{R}^2]$, is $\underline{c} = [3, 1]$, whereas the rate vector of $[\underline{R}^2, \underline{R}^1]$, is $\underline{c} = [1, 3]$. In diagram (c), $\alpha(\underline{R}^2, \underline{R}^1) = 7$, while in diagram (d), $\alpha(\underline{R}^2, \underline{R}^1) = 2$. □

We sometimes partition an FV, and deal with each partitioning element separately. We use only partitions whose elements can be ordered non-ambiguously by any path in the graph. The following definition defines this precisely. Example 3 illustrates this definition.

Definition 4: (FV partitioning) Let h' be an arbitrary rate, $M \leq |E|$ be a number, $i \in [h']$, $j \in [M]$ be arbitrary indices, and $\underline{R} = \{P_i\}$ be an FV. A *partitioning* of \underline{R} is an ordered set of M FVs $\{\underline{R}(j) = \{P_i(j)\}\}$, which satisfy the following. For any i , the $P_i(j)$ form a partitioning of P_i into consecutive sub-FVs (some possibly empty PSs), and there is no “back” path P'' from a partitioning element to any preceding it. *I.e.*, $\bigcup_j P_i(j) = P_i$, and

$$\begin{aligned} \forall_{j' \leq j} \neg \exists_{e'_j \in \underline{R}(j), e'_{j'} \in \underline{R}(j'), k \in [|E|], P'' = \{e'_1, \dots, e'_k\}} \\ \left(\bigwedge_{\ell \in [k-1]} v^+(e''_\ell) = v^-(e''_{\ell+1}) \right) \wedge \\ v^-(e'_1) = v^+(e'_j) \wedge v^+(e'_k) = v^-(e'_{j'}). \end{aligned} \quad (28)$$

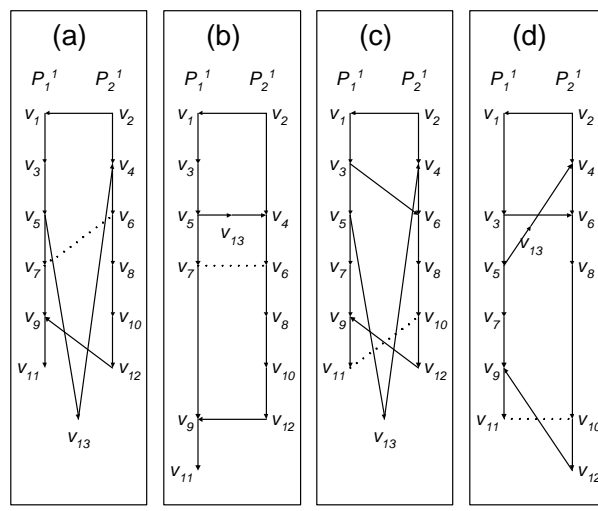


Fig. 5. Examples of partitioning.

Example 3: Consider the four diagrams in Figure 5. In each of these let P_1 and P_2 be the PSs $(v_1, v_3), (v_3, v_5), \dots, (v_9, v_{11})$ and $(v_2, v_4), (v_4, v_6), \dots, (v_{10}, v_{12})$, respectively. Diagram (a) shows a partitioning into 2 elements, indicated by the dotted line between v_6 and v_7 . Note that as the graph is cycle free, it can always be redrawn s.t. the edges do not point upward, and partition lines are drawn horizontally, as in diagram (b). Were the diagram *not* cycle-free, as in diagram (c) then some edges might need to be drawn point upward, as in diagram (d). Diagrams (c) and (d) also show a partitioning in which there is a back path (via (v_{12}, v_9)). As we do not consider these cases, in the following, we will only draw edges which do not point upward, and partitioning lines which are horizontal. \square

In the remainder of the subsection, let d' be an arbitrary number of terminals, and let $\underline{R}^1, \dots, \underline{R}^{d'}$ be flow vectors, such that for $i \in [d']$, $h_i = \text{len}(\underline{R}^i)$ is the rate of the i th FV.

Given flow vectors, it might be possible to “reflow” them so that they represent flows from the same origins to effectively the same terminations, but through different paths. Some “reflows” can decrease the number of NCSs, in which we call them “disentanglements”. We next define these terms. Example 5 in Subsection IV-C illustrates the following two definitions.

Definition 5: (Reflow) For $j \in [h_i]$, let the PS in $\underline{R}^i[j]$ describe a path whose first and last edge are e_j^i and $e_{\pi^i(j)}^i$, respectively. Let π^i be some permutation of the elements of $[h_i]$.

A reflow $\Gamma([\underline{R}^1, \dots, \underline{R}^{d'}])$ is a transformation

$$\Gamma : [\underline{R}^1, \dots, \underline{R}^{d'}] \rightarrow [\underline{R}'^1, \dots, \underline{R}'^{d'}], \quad (29)$$

s.t. $\underline{R}'^1, \dots, \underline{R}'^{d'}$ are flow vectors, the path sets in \underline{R}'^i are from edges e_j^i to $e_{\pi^i(j)}^i$

Definition 6: (Disentangling) A *disentangling* $\Gamma([\underline{R}^1, \dots, \underline{R}^{d'}])$ is a reflow s.t.

$$\alpha(\underline{R}'^1, \dots, \underline{R}'^{d'}) \leq \alpha(\underline{R}^1, \dots, \underline{R}^{d'}), \quad (30)$$

i.e., the number of NCSs in $\underline{R}'^1, \dots, \underline{R}'^{d'}$ is not larger than the number of NCSs in $\underline{R}^1, \dots, \underline{R}^{d'}$.

Let $\underline{c} = [h_1, \dots, h_{d'}]$ denote the rate vector of the flow vectors. We use $\gamma(\underline{c})$ to denote the minimum guaranteed number of NCSs obtainable by disentangling. *I.e.*, letting G , s , and T range over all possible graphs, source node and terminal set, respectively, letting $\underline{R}^1, \dots, \underline{R}^{d'}$ range over all flow vectors for $\{G, s, T\}$ s.t. their rate vector is \underline{c} , and letting $\underline{R}'^1, \dots, \underline{R}'^{d'}$ range over all possible reflows of $\underline{R}^1, \dots, \underline{R}^{d'}$, then

$$\gamma(\underline{c}) = \max_{G, s, T} \max_{\underline{R}^1, \dots, \underline{R}^{d'}} \min_{\underline{R}'^1, \dots, \underline{R}'^{d'}} \alpha(\underline{R}'^1, \dots, \underline{R}'^{d'}). \quad (31)$$

The main result we prove in this section concerns the dependency of the minimum number of NCSs on the rate vector alone.

Theorem 2: For any number of terminals d'

1) $\gamma\left(\underbrace{[h_1, 2, \dots, 2]}_{d'-1}\right)$ is finite.

2) If for any h' and h'' , $\gamma([h', h''])$ is finite, then $\gamma([h_1, \dots, h'_d])$ is finite.

While we conjecture that for any h' and h'' , $\gamma([h', h''])$ is finite, and so for any h_1, \dots, h_d , $\gamma([h_1, \dots, h_d])$ is finite, we have not proved this, and leave this as a conjecture.

The section continues as follows. In Subsection IV-B we discuss the case $\underline{c} = [h_1, 1]$. In Subsection IV-C we discuss the case $\underline{c} = [h_1, 2]$. In Subsection IV-D we discuss the relationship between the case $\underline{c} = [h_1, \dots, h_d, 2^{d-1}]$ and the case $\underline{c} = [h', h'']$, where h' and h'' are arbitrary.

B. Arbitrary Flow Vector and Rate 1 Flow Vector

In this subsection we discuss disentangling for the case $\underline{c} = [h_1, 1]$, for any h_1 . We will use this in Subsection IV-C.

We first define edge relationships in PSs. Example 4 illustrates the next definition.

Definition 7: (PS relationships.) Let P be a PS from $\mathbf{v}^-(e'_1)$ and $\mathbf{v}^+(e'_k)$, e.g., $P = \{e'_1, \dots, e'_k\}$. We denote by $e \stackrel{P}{\preceq} e'$ the fact that edge e is the predecessor of edge e' along P , i.e., the existence of a sub-PS $P' \subseteq P$ from $\mathbf{v}^-(e)$ to $\mathbf{v}^+(e')$.

For any group of edges $E' \in E$, s.t. $E' \cap P \neq \emptyset$, we denote by $\min_P(E')$ and $\max_P(E')$,

$$e' \in E \cap P' \bigwedge \forall_{e'' \in E'} e'' \stackrel{P}{\preceq} e', \quad (32)$$

and

$$e' \in E \cap P' \bigwedge \forall_{e'' \in E'} e' \stackrel{P}{\preceq} e'', \quad (33)$$

respectively.

Example 4: In Figure 6 (b), let P^1 be defined as in Example 2. Then $(v_3, v_4) \stackrel{P^1}{\preceq} (v_8, v_9)$, and $\min_{P^1}(\{(v_3, v_4), (v_5, v_6), (v_8, v_9)\}) = (v_3, v_4)$. □

Throughout the section we use the following transformation. Let P^a and P^b be arbitrary paths sets, and let e', e be two edges s.t. $\{e', e\} \subseteq P^a \cap P^b$ and $e' \stackrel{P^a}{\preceq} e$. We define the PS $D(P^a, P^b, e', e)$, as the PS which runs along P^b until e' , then runs along P^a until e , and then resumes running along P^b . I.e.,

$$D(P^a, P^b, e', e) = \left\{ e'' \mid e'' \in P^b \bigwedge e'' \stackrel{P^b}{\preceq} e' \right\} \dot{\cup} \left\{ e'' \mid e'' \neq e \bigwedge e'' \neq e' \bigwedge e' \stackrel{P^a}{\preceq} e'' \stackrel{P^a}{\preceq} e \right\} \dot{\cup} \left\{ e'' \mid e'' \in P^b \bigwedge e \stackrel{P^b}{\preceq} e'' \right\}. \quad (34)$$

Using the above, we now prove the case $\underline{c} = [h_1, 1]$. Example 5 illustrates the proof of the next lemma.

Lemma 4: For any h_1 , $\gamma(\underline{c}) = h_1$.

Proof: The proof is by induction on h_1 . If $P^1 \cap P^2 = \emptyset$, there is nothing to prove. Otherwise, assume the lemma holds for $h_1 - 1$. Let $P_1^1, \dots, P_{h_1}^1$ be h_1 edge-disjoint paths, and let P^2 be a path. If $P^2 \cap \left(\dot{\cup}_{i \in [h_1]} P_i^1 \right) = \emptyset$,

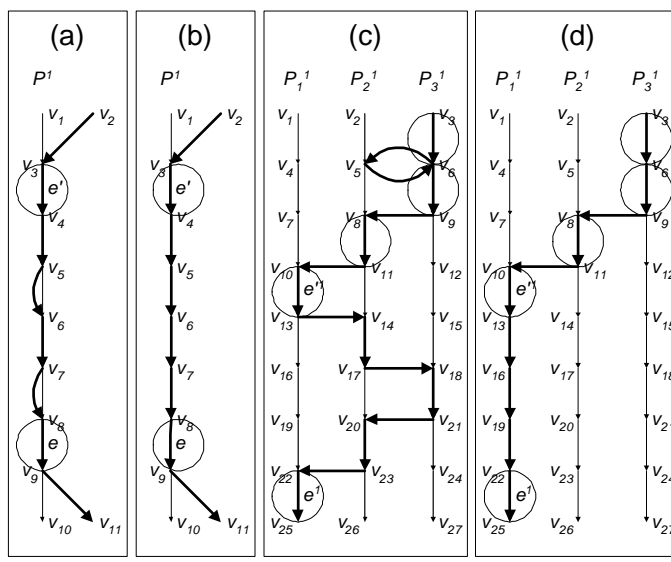


Fig. 6. Path disentangling examples for $\underline{c} = [1, 1]$ and $\underline{c} = [3, 1]$.

there is nothing to prove. Otherwise, let $e^1 = \max_{P_1^1}(\bigcup_{i \in [h_1]} P_i^1)$, and let $j^1 = P_{j^1}^1$ s.t. $e \in P_{j^1}^1$, and let $e'^1 = \min_{P_1^1}(P_1^2)$. Defining $P'^2 = D(P^1, P^2, e^1, e'^1)$ as in (34), and

$$P''^2 = \left\{ e'' \mid e'' \in P'^2 \wedge e'' \neq e^1 \wedge e'^1 \preceq e'' \preceq e^1 \right\}. \quad (35)$$

Since, by definition of e'^1 , $P''^2 \cap P_{j^1}^1 = \emptyset$, then by the induction assumption, there is a disentanglement which can be carried on P''^2 (i.e., defining j^2 as the next PS with which P^1 has an edge in common, and so on), with $\alpha([P_1^1, \dots, P_{h_1}^1], [P''^2]) \leq h_1 - 1$ which implies that $\alpha([P_1^1, \dots, P_{h_1}^1], [P'^2]) \leq h_1$.

The induction base runs along similar lines. ■

Example 5: An example of the induction base can be seen in cases (a) and (b) of Figure 6, where P^1 is a PS from v_1 to v_{10} , and P^2 is the PS indicated by thick arrows from v_2 to v_{11} . Following disentanglement, the only nontrivial clash is in $E^-(v_3)$.

An example of the more general case is shown in cases (c) and (d) of Figure 6. Let P_1^1, P_2^1 and P_3^1 be defined as in Example 2. Let P^2 be the PS, in thick edges, from (v_3, v_6) to (v_{22}, v_{25}) . Let $\underline{R}^1 = [P_1^1, P_2^1, P_3^1]$, $\underline{R}^2 = [P^2]$. In this case, $j^1 = 1$. From the example it is clear that the disentanglement can be continued on, with $j^2 = 2$, and $j^3 = 3$. □

C. Arbitrary Flow Vector and Rate 2 Flow Vector

In this subsection we discuss disentanglement for the case $\underline{c} = [h_1, 2]$, for any h_1 . We will consider throughout the subsection the flow vectors $\underline{R}^1 = [P_1^1, \dots, P_{h_1}^1]$ and $\underline{R}^2 = [P_1^2, P_2^2]$. Let $i, j \in [h_1]$ be arbitrary indices. Clearly, a large number of NCSs can exist only if P_1^2 clashes nontrivially many times with some P_i^1 and/or P_2^2 clashes nontrivially many with some P_j^1 . We will consider the various ways in which this can occur, finding a disentanglement for each case. Specifically, we prove the following.

Lemma 5: For any h_1 ,

$$\gamma([h_1, 2]) \leq 6 \binom{h_1}{2}. \quad (36)$$

The first case we consider is when P_1^2 and P_2^2 each clash nontrivially with at most a single path from \underline{R}^1 . Example 6 illustrates the proof of the next lemma.

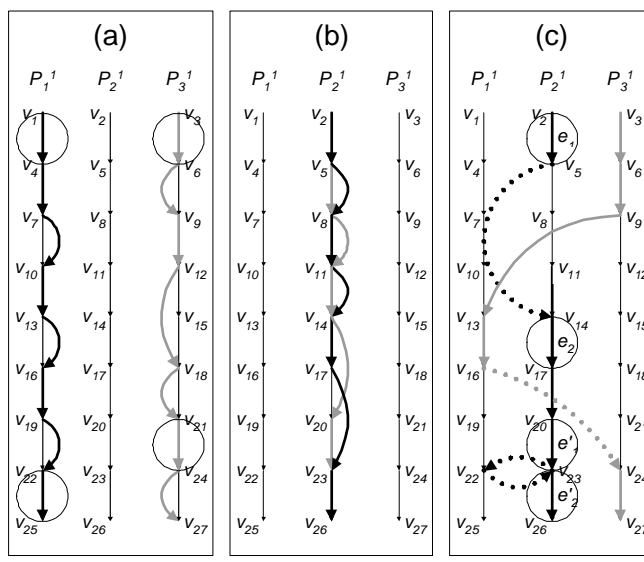


Fig. 7. Single-path clashes for the case $\underline{c} = [3, 2]$.

Lemma 6: Assume that P_1^2 clashes with P_i^1 , P_2^2 clashes with P_j^1 , and furthermore, they each do not coincide with any other path from \underline{R}^1 , i.e.,

$$\begin{aligned} \forall \ell \in [h_1] \setminus \{i\} P_1^2 \cap P_\ell^1 &= \emptyset, \\ \forall \ell \in [h_1] \setminus \{j\} P_2^2 \cap P_\ell^1 &= \emptyset. \end{aligned} \quad (37)$$

Then $\alpha(\Gamma(\underline{R}^1, \underline{R}^2)) \leq 2$.

Proof: If $i \neq j$, then $[P_1^2]$ and $[P_i^1]$ can be viewed as members of flow vectors each of rate 1, and by Lemma 4 they can be disentangled. The same is true of P_2^2 and P_j^1 . If $i = j$, then $[P_1^2, P_2^2]$ and $[P_i^1]$ can be viewed as members of flow vectors with rates 2 and 1, respectively, and by Lemma 4 they can be disentangled. ■

Example 6: In diagrams (a) and (b) of Figure 7, let P_1^1 , P_2^1 , and P_3^1 be defined as in Example 2, and let $\underline{R}^1 = [P_1^1, P_2^1, P_3^1]$. Let the paths P_1^2 and P_2^2 be the ones shown in thick black and thick grey, respectively. Diagrams (a) and (b) illustrates the cases $i \neq$ and $i = j$, respectively. □

Consider one of the paths of \underline{R}^2 , say P_1^2 , and one of the paths of \underline{R}^1 , say P_i^1 . Assume that P_1^2 leaves P_i^1 at e_1 , and returns to it at e_2 . The next lemma shows that we need only consider such cases where there is an edge on P_i^1 between e_1 and e_2 , s.t. P_2^2 runs on that edge. Example 7 illustrates the proof of the next lemma.

Lemma 7: Assume that there are two edges, e_1 and e_2 , s.t. $\{e_1, e_2\} \subseteq P_i^1 \cap P_1^2$, $e_1 \preceq^{P_1^2} e_2$, and that

$$\neg \exists e' e' \in P_i^1 \cap P_2^2 \bigwedge e_1 \preceq^{P_i^1} e' \preceq^{P_i^1} e_2 \quad (38)$$

(this is trivially the case if $e_1 \in \mathbf{E}^-(\mathbf{v}^-(e_2))$). Let $P_1'^2 = D(P_i^1, P_1^2, e_1, e_2)$. Then

$$\alpha(\underline{R}^1, [P_1'^2, P_2^2]) \preceq \alpha(\underline{R}^1, [P_1^2, P_2^2]) . \quad (39)$$

Example 7: In diagram (c) of Figure 7, let \underline{R}^1 , and \underline{R}^2 be defined as in Example 6. Let the dotted arrow (v_8, v_{20}) represent the fact that P_1^2 leaves P_2^1 at v_5 and returns to P_2^1 for the first time in v_{14} , but is unconstrained about what it does in the interim. Clearly, since there is no edge $e' \in P_2^1$ on the sub-path $P' = \{(v_5, v_8), (v_8, v_{11}), (v_{11}, v_{14})\}$, then P_1^2 can be altered to run along P' , saving (at least one) NCS.

This is true also when considering e_1' and e_2' . Let the dotted arrows (v_{23}, v_{22}) and (v_{22}, v_{23}) represent the fact that P_1^2 leaves from and returns to P_2^1 at v_{23} , but is unconstrained about what it does in the interim. Then P_1^2 can be made to run from e_1' directly to e_2' , saving (at least one) NCS. □

Justified by Lemma 7, in the remainder of the subsection we will consider w.l.o.g. only cases in which between every two occurrences of P_1^2 on what of the paths of \underline{R}^1 there is an occurrence of P_2^2 (and vice versa).

Assume one of the paths of \underline{R}^2 , say P_1^2 clashes nontrivially many times with (at least) two paths of \underline{R}^1 , say P_i^1 and P_j^1 . Note that by Lemma 7, we can assume that P_2^2 clashes nontrivially with (at least) P_i^1 and P_j^1 many times, specifically between any two nontrivial clashes of P_2^2 with them. Intuitively, a large amount of “crisscrossing” between P_1^2 and P_2^2 is excessive, as for the most part, P_1^2 and P_2^2 could run along P_i^1 and P_j^1 .

We will show that this is indeed the case. To do so, we define the PS $D'(P^a, P^b, P^c, e', e)$, as the PS which runs along P^b until e' , then runs along P^a until e , and then runs along P^c . I.e.,

$$D'(P^a, P^b, P^c, e', e) = \tag{40}$$

$$\left\{ e'' \mid e'' \in P^b \wedge e'' \stackrel{P^b}{\preceq} e' \right\} \dot{\cup}$$

$$\left\{ e'' \mid e'' \neq e \wedge e'' \neq e' \wedge e' \stackrel{P^a}{\preceq} e'' \stackrel{P^a}{\preceq} e \right\} \dot{\cup}$$

$$\left\{ e'' \mid e'' \in P^c \wedge e \stackrel{P^c}{\preceq} e'' \right\}.$$

Note that D' is a general case of D , since $D(P^a, P^b, e', e) = D'(P^a, P^b, P^b, e', e)$.

The following lemma shows that in the case of “excessive” crisscrossing, a disentangling can be found which runs along the paths of \underline{R}^1 . Example 8 illustrates the statement of the lemma.

Lemma 8: Let P_i^1 and P_j^1 be two distinct paths from \underline{R}^2 . Consider one of the paths of \underline{R}^2 , say P_1^2 . Assume that there are 5 edges, e^1, \dots, e^5 , s.t. $\{e^1, e^3, e^5\} \subseteq P_i^1 \cap P_1^2$, $\{e^2, e^4\} \subseteq P_j^1 \cap P_1^2$, and $e^1 \stackrel{P_1^2}{\preceq} \dots \stackrel{P_1^2}{\preceq} e^5$.

Then there must exist two pairs of edges, (e'_1, e'_2) and (e_1, e_2) , which satisfy at least one of the following:

- 1) $\{e'_1, e_1\} \subseteq P_1^2 \cap P_i^1$, $\{e'_2, e_2\} \subseteq P_2^2 \cap P_j^1$, and

$$\alpha(\underline{R}^1, [P_1'^2, P_2'^2]) \preceq \alpha(\underline{R}^1, \underline{R}^2), \tag{41}$$

where

$$P_1'^2 = D(P_i^1, P_1^2, e'_1, e_1), \tag{42}$$

$$P_2'^2 = D(P_j^1, P_2^2, e'_2, e_2).$$

- 2) $\{e'_1, e_2\} \subseteq P_1^2 \cap P_i^1$, $\{e'_2, e_1\} \subseteq P_2^2 \cap P_j^1$, and

$$\alpha(\underline{R}^1, [P_1'^2, P_2'^2]) \preceq \alpha(\underline{R}^1, \underline{R}^2), \tag{43}$$

where

$$P_1'^2 = D'(P_i^1, P_1^2, P_2^2, e'_1, e_1), \tag{44}$$

$$P_2'^2 = D'(P_j^1, P_2^2, P_1^2, e'_2, e_2).$$

Example 8: Consider diagrams (a) and (b) in Figure 8. In these diagrams let \underline{R}^1 , and \underline{R}^2 be defined as in Example 6, with the dotted lines explained shortly. PS P_1^2 clashes nontrivially with both P_1^1 and P_2^1 . Since we are only interested in clashes between P_1^2 and either P_1^1 or P_2^1 , we let dotted arrows indicate only the origin and target of a transition of P_1^1 , without regard to what P_1^1 does in the interim. W.l.o.g. PS P_2^2 also clashes nontrivially with both P_1^1 and P_2^1 , once between any two nontrivial clashes of P_1^1 with them. It is also possible that P_1^2 and/or P_2^2 clash nontrivially with, say, P_3^1 (as in (v_{18}, v_{21})), although this is immaterial to the disentangling.

Diagram (a) illustrates case 1 in the above. The corresponding disentangling is shown in diagram (a) of Figure 9.

Diagram (b) illustrates case 2 in the above. The corresponding disen[tanglement is shown in diagram (b) of Figure 9.

□

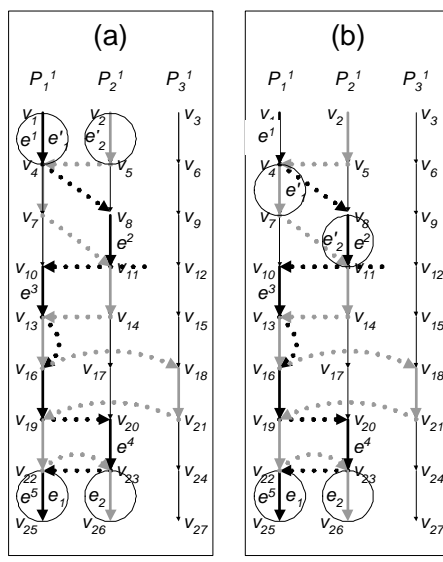


Fig. 8. “Crisscrossing” examples for the case $\underline{c} = [3, 2]$.

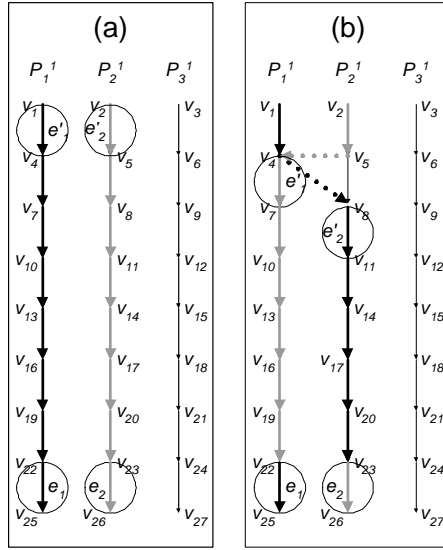


Fig. 9. “Crisscrossing”-disentangling examples for the case $\underline{c} = [3, 2]$.

The following shows how to choose the edges e'_1 and e'_2 in Lemma 8. Choosing the edges e_1 and e_2 is similar. Example 9 illustrates the next procedure.

Let $f_1'^1$ and f_1^1 be two non-consecutive edges s.t. P_1^2 leaves P_i^1 at $f_1'^1$, clashes nontrivially with P_j^1 , and the first edge where it returns to P_i^1 is f_1^1 . I.e.,

$$\begin{aligned}
 \{f_1'^1, f_1^1\} &\subseteq P_1^2 \cap P_i^1, \\
 f_1^1 &\notin \mathbf{E}^-(v^-(f_1'^1)), \\
 \forall e'' \in P_1^2 f_1'^1 \preceq^{P_1^2} e'' \preceq^{P_1^2} f_1^1 &\Rightarrow e'' \notin P_i^1, \\
 \exists e'' \in P_1^2 f_1'^1 \preceq^{P_1^2} e'' \preceq^{P_1^2} f_1^1.
 \end{aligned} \tag{45}$$

Let $f_2'^1$ be the first edge of P_1^2 on P_j^1 after $f_1'^1$, and let f_2^1 be the last edge of P_1^2 on P_j^1 before f_1^1 . I.e., letting

$$P_j'^1 = \left\{ e'' \mid e'' \in P_j^1 \wedge f_1'^1 \preceq^{P_1^2} e'' \preceq^{P_1^2} f_1^1 \right\}, \tag{46}$$

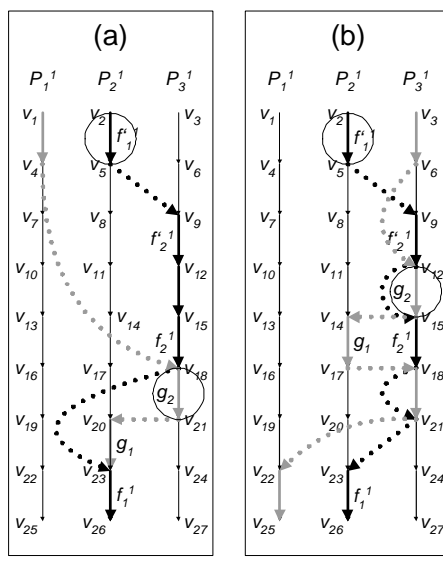


Fig. 10. Examples of case 1 when finding e'_1 and e'_2 .

then $f_2^{\prime 1} = \min_{P_1^1}(P_j^{\prime 1})$ and $f_2^1 = \max_{P_1^1}(P_j^{\prime 1})$.

As mentioned above, we assume w.l.o.g that there is an edge g_1 which is on P_i^1 between $f_1^{\prime 1}$ and f_1^1 , s.t. P_2^1 runs through g_1 , and that w.l.o.g. we assume that g_1 is the first such edge between $f_1^{\prime 1}$ and f_1^1 , i.e.,

$$\begin{aligned} g_1 &\in P_1^2 \cap P_i^1, & (47) \\ f_1^{\prime 1} &\preceq^{P_i^1} g_1 \preceq^{P_i^1} f_1^1, \\ \forall_{g' \in P_2^2} f_1^{\prime 1} &\preceq^{P_i^1} g' \preceq^{P_i^1} f_1^1 \Rightarrow g_1 \preceq^{P_i^1} g'. \end{aligned}$$

It is possible that there is an edge on P_j^1 which is the predecessor of g_1 along P_2^2 . If such an edge exists, we denote it by g_2 , i.e., $g_2 \in P_1^2 \cap P_j^1$, and $g_2 \preceq^{P_2^2} g_1$. Choosing e'_1 and e'_2 depends on whether g_2 exists, and if so, on its relationship, along the path P_j^1 , to $f_2^{\prime 1}$.

- 1) Assume g_2 exists, and it is the successor of $f_2^{\prime 1}$ along P_j^1 , i.e.,

$$\exists_{g_2 \in P_2^2 \cap P_j^1} f_2^{\prime 1} \preceq^{P_j^1} g_2 \wedge g_2 \preceq^{P_2^2} g_1. \quad (48)$$

Then we choose $e'_1 = f_1^{\prime 1}$ and $e'_2 = g_2$.

- 2) Assume that g_2 does not exist, or that it does exist but is *not* the successor of $f_2^{\prime 1}$ along P_j^1 , i.e.,

$$\neg \exists_{g_2 \in P_2^2 \cap P_j^1} f_2^{\prime 1} \preceq^{P_j^1} g_2 \wedge g_2 \preceq^{P_2^2} g_1. \quad (49)$$

Then we choose $e'_1 = g_1$ and $e'_2 = f_1^{\prime 2}$.

Example 9: In Figures 10 and 11, let \underline{R}^1 , and \underline{R}^2 be defined as in Example 2. Diagrams (a) and (b) in Figure 10 show examples of case 1 in the above. Diagrams (a), (b), and (c) in Figure 11 show examples of case 2 in the above. □

The following lemma shows the properties of e'_1 and e'_2 which we will use for the proof of Lemma 8.

Lemma 9: Let e'_1 and e'_2 be chosen as described above. Assume that $e'_1 \in P_i^1$ and $e'_2 \in P_j^1$. Let e''_1 and e''_2 be edges on P_i^1 and P_j^1 , respectively, s.t. they are successors of e'_1 and e'_2 , respectively. I.e., $e'_1 \preceq^{P_i^1} e''_1$ and $e'_2 \preceq^{P_j^1} e''_2$

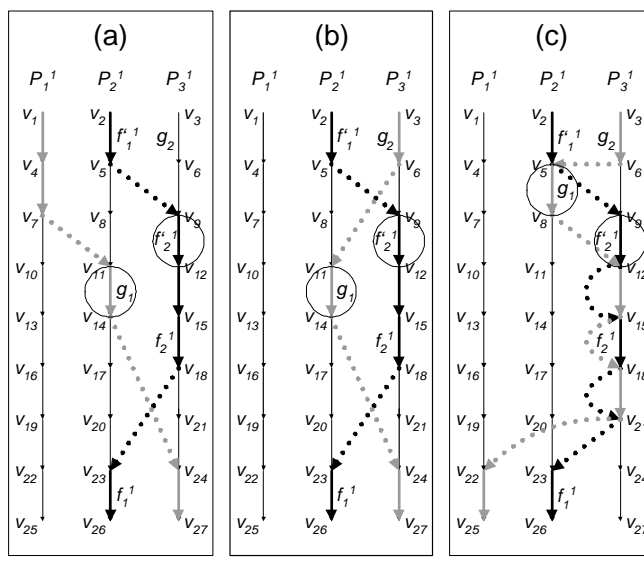


Fig. 11. Examples of case 2 when finding e'_1 and e'_2 .

Let $P_1'^2$ and $P_2'^2$ be PSs defined by

$$\begin{aligned} P_1'^2 &= D'(P_1^2, P_i^1, \emptyset, e'_1, e''_1), \\ P_2'^2 &= D'(P_2^2, P_j^1, \emptyset, e'_2, e''_2). \end{aligned} \quad (50)$$

Then $P_1'^2 \cap P_2'^2 = \emptyset$.

Using Lemma 9, we prove Lemma 8.

Proof: Assume we find e_1 and e_2 from Lemma 8 by using the “mirrored” version of the technique used for finding e'_1 and e'_2 . That is, starting from t_2 we traverse *backward* on the 4 paths involved, and the backward paths. By Lemma 9 there is a min-cut of 2 from s to the e''_1 and e''_2 described above. In particular, these two edges can be e_1 and e_2 , respectively. By symmetry, there is a min-cut of 2 from t_2 to e_1 and e_2 . It follows that the disentangling maintains a min-cut of 2 from s to t_2 . The min-cut from s to t_1 clearly is unchanged. ■

Combining all the above, we prove Lemma 5. Example 10 illustrates some points of the proof.

Proof: Let $k \in [|E|]$ be arbitrary indices. Starting off with \underline{R}^1 and \underline{R}^2 , we create a series of $|E|$ FVs \underline{R}_k^1 and \underline{R}_k^2 , defined by

$$[\underline{R}_k^1, \underline{R}_k^2] = \begin{cases} [\underline{R}^1, \underline{R}^2] & , k = 1 \\ \Gamma([\underline{R}_{k-1}^1, \underline{R}_{k-1}^2]) & , 1 \leq k \leq |E| \end{cases} , \quad (51)$$

and set in the following manner.

We first note that at stage k , we may apply Lemma 7 when applicable, obtaining the next stage. If Lemma 7 is inapplicable, then consider all possible partitioning of \underline{R}_k^1 . If a partitioning element can be found to which Lemmas 6 or 8 is applicable, the appropriate lemma can be applied, obtaining the next stage.

Assume that at some stage none of the above can be applied. Choose two arbitrary PSs of \underline{R}_k^1 , say $P_i^1 = \underline{R}_k^1[i]$, and $P_j^1 = \underline{R}_k^1[j]$. Consider a partitioning of \underline{R}_k^1 s.t. each partitioning element is the maximum one in which P_i^1 does not transition between P_i^1 and P_j^1 . By the fact that a disentangling is not possible, and by Lemma 8, there are at most 4 partitioning elements, implying at most 3 transitions between P_i^1 and P_j^1 . This shows that the number of NCSs caused by transitions of PSs from \underline{R}_k^2 between PSs from \underline{R}_k^1 is at most $6 \binom{h_1}{2}$. Note that from the proof of Lemma 6 it follows that these are the only NCSs.

Thus

$$\alpha(\underline{R}_{|E|}^1, \underline{R}_{|E|}^2) = \Gamma(\underline{R}^1, \underline{R}^2) \leq 6 \binom{h_1}{2}. \quad (52)$$

■

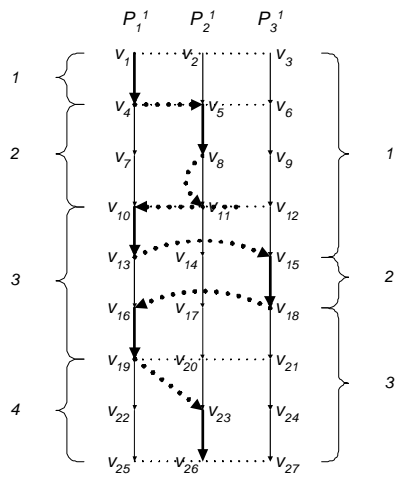


Fig. 12. Possible partitioning when no disentangling is possible.

Example 10: In Figures 12 let \underline{R}^1 and \underline{R}^2 be defined as in Example 2, with only one of the paths of \underline{R}^2 shown. As viewed by P_1^1 and P_2^1 , the FV can be partitioned into the 4 elements marked by 1–4. As viewed by P_1^1 and P_3^1 , however, the FV can be partitioned into the 3 elements marked by 1’–4’.

□

D. The General Case

In this subsection we discuss the relationship between disentangling for the case $\underline{c} = [h_1, \dots, h_{d'}]$, for any d' , and $h_1, \dots, h_{d'}$, and disentangling for the case $\underline{c} = [h', h'']$, for arbitrary h' and h'' .

The following lemma proves, in conjunction with Subsections IV-B and IV-C, Theorem 2. Example 11 illustrates the general proof idea.

Lemma 10:

$$\begin{aligned} \gamma([h_1, \dots, h_i, h_{i+1}]) &\leq \\ &\gamma([h_1, \dots, h_i]) + \\ &\left(\sum_i h_i\right) \cdot \gamma([h_1, \dots, h_i]) \cdot \\ &\left(\gamma\left(\left[\sum_i h_i, h_{i+1}\right]\right) + h_{i+1}\right). \end{aligned} \quad (53)$$

Example 11: In diagram (a) of Figure 13, we see the case of $\underline{c} = [3, 3, 2]$.

Assume we first perform a disentangling on the flows from s to t_1 with s to t_2 . The result is shown in diagram (b), with the flows from s to t_1 shown in thick-edged paths. The graph can then be redrawn to appear as in (c). In this diagram, all edges are drawn in parallel. Dotted edges represent that one edge feeds directly into another. In diagram (c) it can be observed that the disentangling result can be partitioned, so that in each partitioning element, the paths are disjoint, and their number is fixed. A partitioning element is caused by either the merging or the splitting of two PSs from s to t_1 and to t_2 .

When next performing a disentangling of the two first flows and the flow from s to t_3 , as in diagram (a) of Figure 14, the partitioning form of the graph becomes useful. In diagram (b), we can see that the flow from s to t_3 can be disentangled with each partitioning element separately. The flow from s to t_3 views each partitioning element as a flow with rate that can range between 3 and 6.

□

In the remainder of the subsection we formalize and generalize the argument in Example 11.

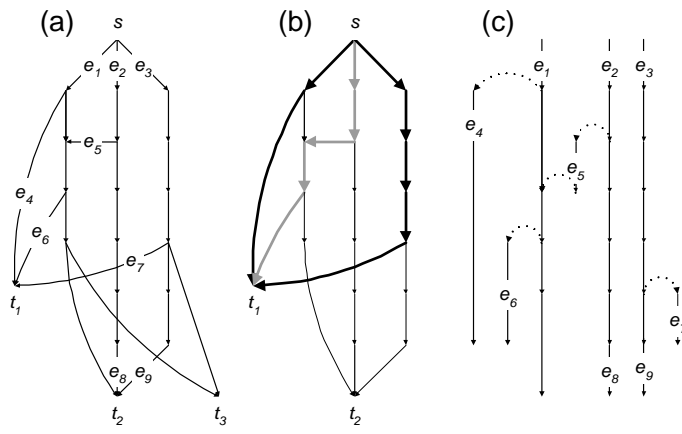


Fig. 13. Disentangling of flows of s to t_1 with s to t_2 .

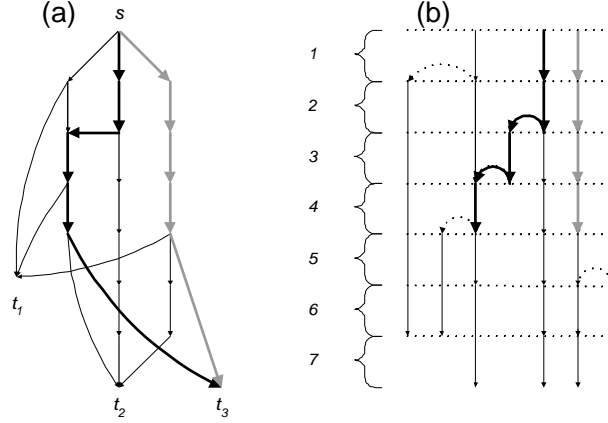


Fig. 14. Disentangling of flows of s to t_1 and to t_2 with s to t_3 .

We next define a *compound* FV, a partitioning of it, and discuss one partitioning in particular. Example 12 illustrates the next definitions.

Definition 8: (Compound FV) Let d' FVs be given, and let $i \in [d']$ be an arbitrary index. The *compound* FV (CFV) of these FVs is a d' -dimensional vector of ordered sets, with a total of $\prod_i (h_i + 1)$ entries, one corresponding to each choice from $\left(\{0\} \dot{\cup} [h_1] \times \cdots \times \{0\} \dot{\cup} [h_{d'}] \right)$. An arbitrary element of \underline{W} , say $\underline{W}[j_1, \dots, j_{d'}]$ is set s.t.

$$\underline{W}[j_1, \dots, j_{d'}] = \left\{ e'' \mid \bigwedge_i \left\{ \begin{array}{ll} e'' \in \underline{R}^i[j_i] & , \quad j_i \neq 0 \\ \forall_{j'_i \neq j_i} e'' \notin \underline{R}^i[j'_i] & , \quad j_i = 0 \end{array} \right. \right\}. \quad (54)$$

It should be noted, that while $\underline{W}[j_1, \dots, j_{d'}]$ is an ordered union of PSs, it is not, in general, a PS itself.

We define a partition of \underline{W} in the same way it is defined for a FV. That is, the partitioning of \underline{W} induces a partitioning on each $Q_{j_1, \dots, j_{d'}}$, and there are no “back” paths from a partitioning element to any of its predecessors.

We consider in particular the following partitioning of \underline{W} . We traverse along the graph using breadth-first search [5], breaking edge ties by the order of the edges in E . A new partitioning element begins only by an edge e which satisfies one of the following. Either (at least) one of the PSs leaves a different PS at e , or (at least) one of the PSs merges into a different PS at e . It can be seen that the partition elements each contain only edge-disjoint PSs, and the number of PSs in each partition element is at most $\sum_i h_i$.

Example 12: Consider diagram (a) in Figure 13. Following a disentangling of the flows of s to t_1 with s to t_2 , \underline{W} is a 2-dimensional vector. Since e_2 is used for both the second path of the flow from s to t_1 and the second

path of the flow from s to t_2 , then $e_2 \in \underline{W}[2, 2]$. Since e_5 is used for the second path of the flow from s to t_1 , but is not used by any path of the flow from s to t_2 , then $e_2 \in \underline{W}[2, 0]$ and $e_2 \notin \underline{W}[2, 1] \cup \underline{W}[2, 2] \cup \underline{W}[2, 3]$.

The result of the above-mentioned partitioning on \underline{W} is shown in diagram (c) of Figure 14. □

We now prove Lemma 10.

Proof: The proof is by induction on i , the length of the rate vector \underline{c} . For $i = 1$ there is nothing to prove. Assume the lemma holds for i . We perform a disentangling on $\underline{R}^1, \dots, \underline{R}^i$, find the CFV of the result, and find the partitioning mentioned above. By the induction assumption, at this point there are at most

$$\gamma([h_1, \dots, h_i]) \quad (55)$$

NCSs. As mentioned above, each partition element consists of at most $\sum_i h_i$ edge-disjoint paths, and each partition element can be attributed to either a merging or splitting of PSs. Since a PS merge is equivalent to the existence of an NCSs, the number of merges is at most $\gamma([h_1, \dots, h_i])$. Between any two PS merges, any PS split increase the number of PSs by 1. As the number of paths is at most $\sum_i h_i$, the number of path splits between any path merge is at most $\sum_i h_i$. It follows that the number of partitioning elements is at most

$$(\gamma([h_1, \dots, h_i])) \left(\sum_i h_i \right). \quad (56)$$

Assume we disentangle \underline{R}^{i+1} with each of the partitioning elements. This can be viewed as a disentangling of two flow vectors, the first of which has a rate in the range $\{\min_i h_i, \dots\}$, and the second has a rate of h_{i+1} . By the induction assumption, then, a disentangling with any partitioning element would result in at most

$$\gamma\left(\left[\sum_i h_i, h_{i+1}\right]\right) \quad (57)$$

NCSs.

When the PSs of \underline{R}^{i+1} leave a partitioning element they possibly merge with the PSs of its successor. This results in at most h_{i+1} NCSs. Combining this with (55), (56), and (57), proves the theorem. ■

V. BOUNDS ON LARGE-FIELD SIMULATION BY NODE MEMORY

We saw in Section II that there is a lower bound on the coding alphabet size. If the alphabet size supported by the links, q , is not large enough for the given multicast requirements, the addition of memory in the nodes can be used to simulate a large alphabet size. In this subsection we describe this technique, and show bounds on its latency and bandwidth.

Consider a network W' over a graph G , specified source node s , and terminal group T , with the following attributes. The links of W' transmit symbols over a field \mathcal{F}' of size $q' = |\mathcal{F}'|$. At its source node, s , a process X' is observed, which attains every k th time unit an independent value of entropy $h \cdot \log(q')$. Assume that q' is large enough for linear multicast of h symbols from \mathcal{F}' . Let

$$k = k(q, q') = \lceil \log_q(q') \rceil. \quad (58)$$

In the graph G , if the minimum of min-cuts is h , then the multicast rate is clearly $\frac{h \cdot \log(q')}{k}$.

Consider now a network W over the same G , s and T , but with the following attributes. The links of W transmit symbols over the field \mathcal{F} of size $q' = |\mathcal{F}|$. At its source node, s , a process X is observed, which attains every time unit an independent value of entropy $h \cdot \log(q)$. Since the value that X attains in every k time units has entropy $h \cdot \log(q')$, and since the minimum of min-cuts is h , it might be expected that in network W , information can be multicast at a rate $\frac{h \cdot \log(q')}{k}$ as well. It might be the case, though, that q might not be sufficiently large for multicasting to h terminals.

It is clear that, in network W , if each node accumulates every k consecutive symbols (using node memory), a code over \mathcal{F}' can be simulated. We show in Subsection V-A an upper bound on the throughput and total latency for this scheme. We show in Subsection V-A that, in general, each such node induces some latency on any path on which it resides.

A. An Upper Bound on Bandwidth and Latency

Let $\underline{c} = \underbrace{[h, \dots, h]}_d$ and $\gamma(\underline{c})$ be defined as in Section IV. The following shows that $k \cdot \gamma(\underline{c})$ is an upper bound on the latency

Theorem 3: By adding in each node an additional

$$k \cdot \left(\max_{v \in V} \left| \mathbf{E}^-(v) \cup \mathbf{E}^+(v) \right| + 1 \right) \quad (59)$$

units of memory over \mathcal{F} , a good multicast code can be built, s.t. at every time unit each link carries a symbol from \mathcal{F} , and whose latency and throughput for sending $N \frac{H(X)}{k}$ information are

$$k \cdot \min(n, \gamma(\underline{c})), \quad (60)$$

and

$$\frac{N \frac{H(X)}{k}}{N + k \cdot \min(n, \gamma(\underline{c}))} \xrightarrow{N \rightarrow \infty} \frac{H(X)}{k}, \quad (61)$$

respectively.

Proof: By the assumption that there is a multicast code over a field \mathcal{F}' , there is a set of elements (each from \mathcal{F}') m_e which constitute a valid code. Let $v \in V$ be any node. Assume that v has enough memory to accommodate the $|\mathbf{E}^-(v) + \mathbf{E}^+(v)|$ length- k vectors

$$\underline{y}^v(e_1^-), \dots, \underline{y}^v(e_{|\mathbf{E}^-(v)|}^-), \underline{y}^v(e_1^+), \dots, \underline{y}^v(e_{|\mathbf{E}^+(v)|}^+), \quad (62)$$

each of whose elements is from \mathcal{F} .

Let $i \in [|\mathbf{E}^-(v)|]$, and $j \in [|\mathbf{E}^+(v)|]$ be arbitrary indices. At time unit d , the process at node v does the following. For and $d_k = d \pmod k$:

- If $d < k$ then $y(v(e_j^+))$ is set to 0.
- If $d \geq k \wedge d_k \neq 0$, then the d_k entry of $\underline{y}^v(e_i^-)$ is set to $y(e_i^-)$, and $y(e_j^+)$ is set to the d_k entry of $\underline{y}^v(e_j^+)$.
- If $d \geq k \wedge d_k = 0$, v interprets each vector $\underline{y}^v(e_i^-)$ as an element from \mathcal{F}' . It performs over \mathcal{F}' the appropriate operations specified by m_e , and writes the appropriate results, in vector form, in $\underline{y}^v(e_j^+)$.

It follows that for any d for which $d_k = 0$, the field elements from \mathcal{F}' described by $\underline{y}^v(e_1^-), \dots, \underline{y}^v(e_{|\mathbf{E}^-(v)|}^-)$, are the same as $y(e_1), \dots, y(e_{|\mathbf{E}^-(v)|})$ in the code over \mathcal{F}' at time d . Since this is true, in particular, for all $t \in T \subseteq V$, the code over \mathcal{F} is valid as well.

Clearly, in the above scheme, each node has an initial latency of k time units from the time it receives the first symbol, until the time it outputs its first symbol. On each path from s to any $t \in T$, there is an initial latency which is at most the sum of the latencies along the paths. This proves (60). Following the initial latency, whenever $d_k = 0$, $h \cdot \log(q')$ are reconstructed at each terminal. Combining, this proves (61). ■

B. A Lower Bound on Latency

For any node $v \in V$ which requires coding, consider an edge $e^+ \in \mathbf{E}^+(v)$, and the input edges $e_1^-, \dots, e_{|\mathbf{E}^-(v)|}^- \in \mathbf{E}^-(v)$. As seen from (2) in Subsection I-C, the linear transmission scheme requires the calculation of the dot product

$$y(e^+) = [y(e_1^-), \dots, y(e_{|\mathbf{E}^-(v)|}^-)] \cdot [m_{e^+}(e_1^-), \dots, m_{e^+}(e_{|\mathbf{E}^-(v)|}^-)]. \quad (63)$$

Let the fields \mathcal{F}' and \mathcal{F} be defined as in the previous subsection, and let all the elements in (63) be taken from \mathcal{F}' . In Subsection V-A a latency-incurring scheme for calculating (63) was shown. In this subsection we show that any scheme, linear or non-linear, implementing the general form of (63) incurs latency.

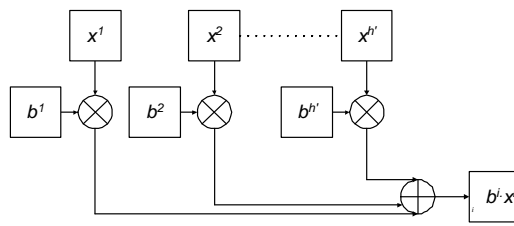


Fig. 15. Circuit for dot-product calculation over \mathcal{F}' .

Let $i \in [d]$ be an arbitrary index, m_i be d arbitrary elements from \mathcal{F}' , x_i be d variables from \mathcal{F}' , and y be a variable from \mathcal{F}' . Consider the circuit in Figure 15, which calculates $y = \sum_i m_i \cdot x_i$ using elements from \mathcal{F}' . We consider a circuit calculating the same function, but using elements from \mathcal{F} . We do *not* restrict the circuit to calculate functions which are linear in \mathcal{F} .

Let $f(D)$ be an irreducible polynomial of degree k over a finite field $GF(q)$. Since any field \mathcal{F}' must be [12] isomorphic to the field formed by considering the set of all polynomials of degree $k - 1$ or less over $GF(q)$, and defining the addition and multiplication operations, $+^{\mathcal{F}'}$ and $*^{\mathcal{F}'}$, as

$$\begin{aligned} g_1(D) +^{\mathcal{F}'} g_2(D) &= g_1(D) + g_2(D) \pmod{f(D)}, \\ g_1(D) *^{\mathcal{F}'} g_2(D) &= g_1(D) \cdot g_2(D) \pmod{f(D)}, \end{aligned} \quad (64)$$

it suffices to consider this field only. Let $j \in [k]$ be an arbitrary index. As in this field, each element c is a polynomial,

$$c(D) = \sum_j c_j \cdot D^{j-1}, \quad (65)$$

we can represent any element by its coefficients $c \xleftrightarrow{\mathcal{F}'} \underline{c} = [c_1, \dots, c_k]$, e.g., we represent y by $[y_1, \dots, y_k]$, and m_i by $[m_{i,1}, \dots, m_{i,k}]$.

Let $\sum_i m_i *^{\mathcal{F}'} x_i$ denote $m_1 *^{\mathcal{F}'} x_1 + \dots + m_d *^{\mathcal{F}'} x_d$. Viewed over elements from \mathcal{F} , $\sum_i m_i *^{\mathcal{F}'} x_i$ is a function with $d \cdot k$ inputs, and k outputs. Let $x'_{i,j}$ be $d \cdot k$ be variables from \mathcal{F} , y'_j be k variables from \mathcal{F} , and y' be a variable from \mathcal{F} . Let the following functions (linear or otherwise)

$$\begin{aligned} \mathcal{X}'_{i,j} &: \mathcal{F}^j \rightarrow \mathcal{F}, \\ \mathcal{Y}'_j &: \mathcal{F}^{d \cdot j} \rightarrow \mathcal{F}, \\ \mathcal{Y}_j &: \mathcal{F}^j \rightarrow \mathcal{F}, \end{aligned} \quad (66)$$

be defined s.t.

$$\begin{aligned} \mathcal{X}'_{i,j}(x_{i,1}, \dots, x_{i,j}) &= x'_{i,j}, \\ \mathcal{Y}'_j(x'_{1,1}, \dots, x'_{d,1}, \dots, x'_{1,j}, \dots, x'_{d,j}) &= y'_j, \\ \mathcal{Y}_j(y'_1, \dots, y'_j) &= y_j. \end{aligned} \quad (67)$$

A corresponding circuit is shown in Figure 16. In this circuit, at time unit j the circuit has $x_{1,1}, \dots, x_{d,1}, \dots, x_{1,j+1}, \dots, x_{d,j+1}$ as inputs, and outputs y'_1, \dots, y'_{j+1} , s.t. there is a bijection

$$\sum_i m_i *^{\mathcal{F}'} x_i \leftrightarrow [y'_1, \dots, y'_k]. \quad (68)$$

We next show that this is, in general, not possible.

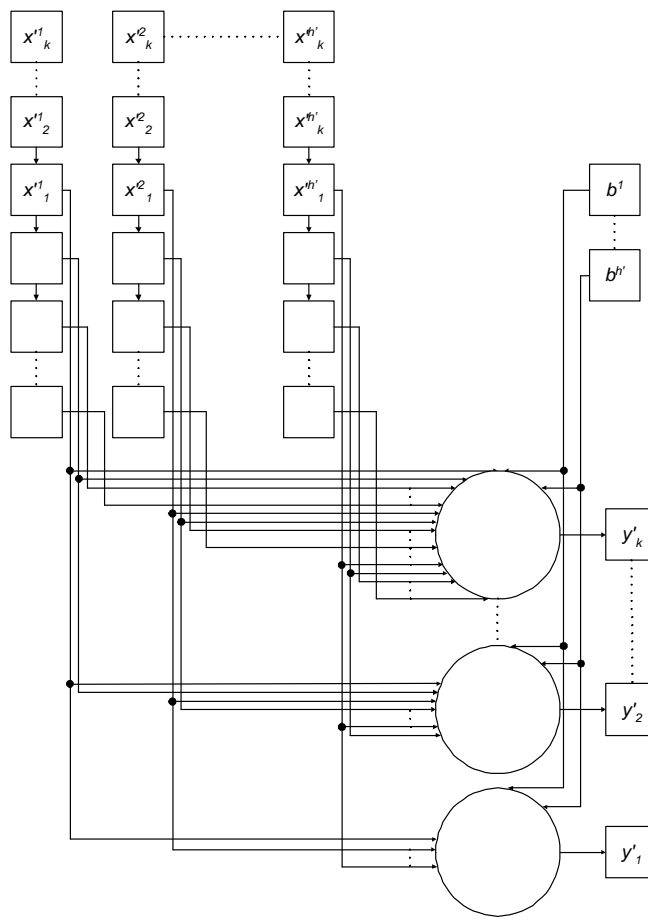


Fig. 16. A latency-free circuit over \mathcal{F} for dot-product calculation over \mathcal{F}' .

Theorem 4: For some values of m_i, x_i , any functions corresponding to (66), (67), and (68), do not calculate the dot-product properly. That is,

$$\exists m_1, \dots, m_d \exists x_1, \dots, x_d \quad (69)$$

$$\sum_i^{\mathcal{F}'} m_i \overset{\mathcal{F}'}{*} x_i \not\overset{\mathcal{F}'}{\leftrightarrow} [\mathcal{Y}_1(y'_1), \dots, \mathcal{Y}_k(y'_1, \dots, y'_d)].$$

Proof: Let $M = \{\underline{m}_1, \dots, \underline{m}_d\}$. Assume the lemma does *not* hold. Then we show that

$$0 \stackrel{(a)}{=} \quad (70)$$

$$H(y_1 | y'_1, M) \stackrel{(m)}{\geq}$$

$$H\left(y_1 \mid \sum_i x_{i,1} \cdot m_{i,1}, M\right) \stackrel{(c)}{=} \quad (70)$$

$$H(y_1 | \underline{x}_1, \dots, \underline{x}_d, M) +$$

$$H\left(\underline{x}_1, \dots, \underline{x}_d \mid \sum_i x_{i,1} \cdot m_{i,1}, M\right) -$$

$$H\left(\underline{x}_1, \dots, \underline{x}_d | y_1, \sum_i x_{i,1} \cdot m_{i,1}, M\right) \stackrel{(d)}{\geq}$$

$$0,$$

which is clearly impossible.

In the above, (a) follows from the assumption that y_1 can be determined by y'_1 (via \mathcal{Y}_1).

We show that inequality (m) follows from the data processing inequality [6]. It is easy to show that for an arbitrary general M , y_1 attains all q possible values. Conversely, y'_1 can attain at most q values. From equality (a), it follows that there is a bijection $y_1 \leftrightarrow y'_1$. Let $x_{1,1}, \dots, x_{d,1}$ be set. Note that y_1 is uniquely determined (from (a)) by y'_1 , y'_1 is completely determined, via \mathcal{Y}'_1 , by $x'_{1,1}, \dots, x'_{d,1}$, and $x'_{1,1}, \dots, x'_{d,1}$ are determined, via $\mathcal{X}'_{1,1}, \dots, \mathcal{X}'_{d,1}$ by $x_{1,1}, \dots, x_{d,1}$. It follows that y_1 is independent from any $x_{i,j}$ where $j \neq 1$. Hence, since $x_{1,1}, \dots, x_{d,1}$ are set, we may as well assume that $x_i \leftrightarrow [x_{i,1}, \underbrace{0, \dots, 0}_{k-1}]$, in which case

$$y_1 = \sum_i x_{i,1} \cdot m_{i,1}. \quad (71)$$

Since (as shown before), y_1 can attain q values, $\sum_i x_{i,1} \cdot m_{i,1}$ can attain at most q values, and by (71), y_1 is determined by $\sum_i x_{i,1} \cdot m_{i,1}$, there is a bijection $y_1 \leftrightarrow \sum_i x_{i,1} \cdot m_{i,1}$. It follows that there is a bijection $y'_1 \leftrightarrow \sum_i x_{i,1} \cdot m_{i,1}$, and so $y'_1 \rightarrow \sum_i x_{i,1} \cdot m_{i,1} \rightarrow y_1$ form a Markov chain.

Equality (c) follows from the definition of conditional entropy.

To show inequality (d), it suffices to show that

$$H\left(\underline{x}_1, \dots, \underline{x}_d \mid \sum_i x_{i,1} \cdot m_{i,1}, M\right) \geq H\left(\underline{x}_1, \dots, \underline{x}_d \mid y_1, \sum_i x_{i,1} \cdot m_{i,1}, M\right). \quad (72)$$

Fix $x_{i,1}$ and M . Let X' be the set of all values that $\underline{x}_1, \dots, \underline{x}_d$ can attain in this case. Let X'' be the set of all values that $(\underline{x}_1, \dots, \underline{x}_d)$ can attain if y_1 is given as well. We will show that $X'' \subsetneq X'$. Let $\{x'_1, \dots, x'_d\} \subseteq \mathcal{F}^d$ be a set of elements s.t.

$$\left(\sum_j y'_j \cdot D^{j-1}\right) = \sum_i \left((x'_i \cdot D) \overset{\mathcal{F}}{*} \left(\sum_j m_{i,j} \cdot D^{j-1}\right) \right), \quad (73)$$

for some y'_1, \dots, y'_k s.t. $y'_1 \neq 0$ (which is certainly possible to find given a free choice of m_i). Letting $\underline{x}_i = [x_{i,1}, x'_i, \underbrace{0, \dots, 0}_{k-2}]$, then $(\underline{x}_1, \dots, \underline{x}_d) \in X' \setminus X''$. ■

VI. GRAPH-SPECIFIC UPPER BOUNDS ON ALPHABET-SIZE

An upper bound on the alphabet size as a function of terminal-set size was shown in [3]. Application of the disentangling transformation from Section IV possibly reduces the NCSs to the point where a smaller alphabet size can suffice.

In this subsection, we show how the MCS determines the alphabet size needed for setting m_{e_j} , where e_j is the edge chosen in step j .

Lemma 11: Let $T''(e_j)$ be the MCS of e_j . If m_{e_j} is set so that

$$\forall t'' \in T''(e_j) \quad \text{Rank} \left(\{\underline{b}(e_j)\} \cup \{\underline{b}(e'') \mid e'' \in C_{t''}^j \setminus f_{t''}^-(e_j)\} \right) = h, \quad (74)$$

then the code is good.

Proof: We prove, by induction on $j \in [\ell]$, that for $i \in [i]$, (21) holds. By Lemma 2, it will follow that the code is good.

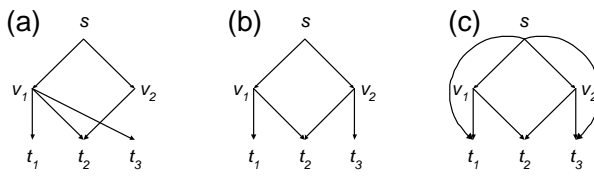


Fig. 17. Static broadcast possibilities.

The induction base, *i.e.*, $j = 1$, holds trivially by the determination of the global coding vectors between s' and s . Assume that (21) holds for $j - 1$, and that $\underline{b}(e_j)$ is set so that (74) is satisfied. Clearly then, $\underline{b}(e_j) \neq \underline{0}$. For any $t'' \in T''(e_j)$, (21) holds by definition. For any $t' \in T'(e_j)$, there is a $t'' \in T''(e_j)$, and a matrix $B_{e''}^{e'}$, s.t.

$$[\underline{b}(e'_1), \dots, \underline{b}(e'_{h-1})] = [\underline{b}(e''_1), \dots, \underline{b}(e''_{h-1})] \cdot B_{e''}^{e'}, \quad (75)$$

for $e'_1, \dots, e'_{h-1} = C_{t'}^j \setminus f_{t'}^-(e_j)$, and $e''_1, \dots, e''_{h-1} = C_{t''}^j \setminus f_{t''}^-(e_j)$. It follows that $\underline{b}(e_j) \neq \underline{0}$ cannot be a linear combination of $\underline{b}(e'_1), \dots, \underline{b}(e'_{h-1})$, since, by (75) it would be then also be a linear combination of $\underline{b}(e''_1), \dots, \underline{b}(e''_{h-1})$, which would contradict (74). ■

The following is a slight variation of a lemma in [3]:

Lemma 12: If at step j of the algorithm, $|T''(e_j)| \leq q$, then m_{e_j} can be set so that (74) is satisfied.

Using Lemma 12, we find an upper bound on the alphabet size as a function of the number of “flow-path clashes”:

Theorem 5: Let m be the number of “clashes”, once the flows are found, *i.e.*,

$$m = \left| [\ell] \cap \{j\} \mid |T'(e_j)| \geq 2 \right|. \quad (76)$$

Then the sufficient alphabet-size is bounded by

$$q_{\max} = \begin{cases} 2 & , \quad h = 1 \\ \min \left\{ d, \binom{h+m}{h-1} \right\} & , \quad h \geq 2 \end{cases} \quad (77)$$

Proof: The case of $h = 1$ is trivial, and so we consider only $h \geq 2$. the upper bound of d was proven in [3]. Prior to the first clash, $|\{\underline{b}(e)\} \exists_{t \in T} e \in C_t^1| = h$, and each clash increments the size of this set by at most 1. The theorem now follows by Lemmas 3 and 12. ■

The upper bound in Theorem 5 can probably be lowered. Our main point is that a reduction in the number of NCSs can reduce the required alphabet size.

VII. STATIC BROADCAST

Let $i \in [d]$ be a terminal index. Assume that the size of the min-cut from s to t_i is h_i . In the previous sections we have focused on the sender transmitting at rate equal to the size of the minimum of min-cuts separating s from all $t \in T$. In this section we discuss *static broadcast* [13], [14], wherein $h_1 \geq \dots \geq h_d = h$, but regardless of the difference in min-cut sizes, s wishes to broadcast a static (*i.e.*, fixed, identical) set of data to all t_i .

Intuitively, if h_j and h_k satisfy $h_j \geq h_k$, then the terminal t_j should be able to receive and decode the information before the terminal t_k . This might necessitate encoding the source, in some cases. Consider cases (a) and (b) in Figure 17. In case (a), each terminal can indeed receive the same number of symbols as the min-cut from s to it. In case (b), if the same symbols are sent to t_1 and t_3 , then t_2 does not receive information at the rate indicated by the min-cut from s to it. For the case of a sequential transmission, we show that there is an asymptotically optimal transmission scheme, based on retransmission of random linear combinations of past values.

Let $\alpha(N, c, h_1)$ be defined as

$$\begin{aligned} \alpha(N, c, h_1) = & \\ & N \cdot h_1 + \\ & c \cdot \log_q(N \cdot h_1) \left(\ln(c \cdot \log_q(N \cdot h_1)) + 1 \right). \end{aligned} \quad (78)$$

In this section X is a process with entropy $H(X) = h_1 \cdot \log(q)$ per time unit. Assume that the values that X attains between times 1 and $N \gg h_1$ are to be transmitted to all $t \in T$. The main result in this section is the following theorem.

Theorem 6: There is a transmission scheme in which the source transmits between times 1 and $\frac{\alpha(N,c,h_1)}{h}$, s.t. with probability at least $1 - \frac{d}{N \cdot h_1^{c-1}}$, each t_i can reconstruct the N values of X at rate

$$\frac{N \cdot h_1}{\frac{\alpha(N,c,h_1)}{h_i}} \xrightarrow{N \rightarrow \infty} h_i. \quad (79)$$

The fact that, in general, information can be multicast to any terminal at rate equal to its min-cut, was observed in [4]. The result here is presented in the general context of static broadcast, and offers a conceptually-simple constructive coding-scheme which has a far lower computational complexity for both G and N .

For the proof of Theorem 6, we first compose a graph $G' = (V', E')$ from $G = (V, E)$, by adding edges from s to $t \in T$, so that the minimum of the set of min-cuts separating s from any $t \in T$ is h_1 . That is,

$$V' = V, \quad (80)$$

$$E' = E \cup \left(\bigcup_i \{e_1^{t_i}, \dots, e_{h_1-h_i}^{t_i}\} \right), \quad (81)$$

where for any $j \in [h_1 - h_i]$, $v^-(e_j^{t_i}) = s$ and $v^+(e_j^{t_i}) = t_i$ (e.g., G and G' are the graphs in cases (b) and (c), respectively, in Figure 17). We then find [3] a global coding assignment (as described in Subsection III-A) for G' , and then discard all edges in $E' \setminus E$. It should be noted that the addition of edges from s to t_i cannot introduce NCSs, and so the results from Sections V and VI can be directly applied.

Let

$$\underline{x} = x_{1,1}, \dots, x_{1,h_1}, \dots, x_{N,1}, \dots, x_{N,h_1} \quad (82)$$

be a length- $(N \cdot h_1)$ vector, s.t. for any $\ell \in [N]$, $\{x_{\ell,1}, \dots, x_{\ell,h_1}\}$ describes the value of X observed at time unit ℓ . For $j' \in \left[\frac{\alpha(N,c,h_1)}{h} - N \right]$, we generate h_1 vectors,

$$\underline{a}_{j',1}, \dots, \underline{a}_{j',h_1}, \quad (83)$$

each of length $N \cdot h_1$, s.t. each vector is distributed i.i.d. from⁵ $\mathcal{U}_{\mathcal{F}}^{N \cdot h_1}$. At time j , s transmits the h_1 symbols

$$\begin{cases} x_{j,1}, \dots, x_{j,h_1} & , \quad j \leq N \\ \underline{a}_{j-N,1} \cdot \underline{x}^+, \dots, \underline{a}_{j-N,h_1} \cdot \underline{x}^+ & , \quad N < j \leq \frac{\alpha(N,c,h_1)}{h} \end{cases} .$$

Let t_i be a terminal node, and let $G_{t_i}^f$ be the flow graph from s to t_i described in Subsection III-A. Let $\{\underline{b}_1^i, \dots, \underline{b}_{h_i}^i\}$ denote the set of global coding vectors on edges in $G_{t_i}^f \cap E$ terminating in t_i , i.e.,

$$\begin{aligned} \{\underline{b}_1^i, \dots, \underline{b}_{h_i}^i\} = & \\ \left\{ \underline{b}(e) \mid e \in \mathbf{E}^-(t_i) \bigwedge e \in E_{t_i}^f \setminus \{e_1^{t_i}, \dots, e_{h_1-h_i}^{t_i}\} \right\}. & \end{aligned} \quad (84)$$

In the remainder, let j be an index s.t. $N < j \leq \frac{\alpha(N,c,h_1)}{h}$, and let $j' = j - N$. The h_i values observed by t_i are then

$$[\underline{b}_k^i] \cdot [\underline{a}_{j',1}^+, \dots, \underline{a}_{j',h_1}^+]^+ \cdot \underline{x}^+, \quad (85)$$

for $k \in [h_i]$.

From (85) we see that the $[\underline{b}_k^i] \cdot [\underline{a}_{j',1}^+, \dots, \underline{a}_{j',h_1}^+]^+$ serve as “random global coding vectors” for \underline{x} . We next find their joint distribution.

⁵For any m , we signify that the distribution of an m -rate vector with elements from \mathcal{F} is uniform over all q^m possibilities, by $\mathcal{U}_{\mathcal{F}}^m$.

VIII. CONCLUSIONS AND FUTURE WORK

In this work we have shown bounds on the alphabet size required for linear codes for network multicast. We have also considered the number of nodes where coding is required. For this, we have defined the notion of nontrivial clash set that determine the number of coding nodes, and led to upper bounds on the alphabet size.

As seen from this analysis, the required alphabet size for a given graph can be reduced by disentangling. It is also evident that there are classes of graphs for which the alphabet size can be smaller than that suggested at previous work. Characterizing these classes and determining for a given graph (network) its class can be helpful in designing an efficient multicast linear code for that network.

Another issue we plan to consider is the question of rate in cyclic networks with small alphabet-size. Previous work ([1], [2], [3], [4]) has dealt with cyclic graphs by means very similar to a *TTL* (time to live) ([15]). By reserving some of the information in each transmitted symbol for storing the number of nodes it had encountered, nodes can discard transmitted symbols they have already encountered. It has been noted, both in the theoretical work mentioned above, and in practice, that this is negligible for symbols taken from large alphabets (*e.g.*, IP packets). We will examine the effect of cycles on small alphabet sizes as well.

IX. ACKNOWLEDGEMENTS

Thanks to V. Dreizin, E. Erez, and Prof. S. Litsyn of the department of EE - Systems in Tel-Aviv University, for useful discussions.

APPENDIX

In this section we prove Lemma 14 from Section VII.

Proof: Let $\hat{N} = c \cdot \log_q(N \cdot h_1)$, and let $S_{N'_i+1}, \dots, S_N$, be a partition of the rows of A_i 's rectangular part, s.t.

$$|S_j| = \begin{cases} 1 & , N'_i + 1 \leq j \leq N \cdot h_1 - \hat{N} + 1 \\ \frac{\hat{N}}{N \cdot h_1 - j + 1} & , N \cdot h_1 - \hat{N} + 2 \leq j \leq N \cdot h_1 \end{cases} \quad (90)$$

Let $\underline{s}_1, \dots, \underline{s}_{N'_i}$ be the vectors in the first rows of A_i . We choose $N \cdot h_1 - N'_i$ vectors $\underline{s}_j \in S_j$, so that \underline{s}_j maximizes $\text{Rank}(\{\underline{s}_1, \dots, \underline{s}_{j-1}, \underline{s}_j\})$. Let V_j be the event

$$\forall_{s \in S_j} \text{Rank}(\{\underline{s}_1, \dots, \underline{s}_{j-1}, \underline{s}_j\}) = \text{Rank}(\{\underline{s}_1, \dots, \underline{s}_{j-1}\}). \quad (91)$$

Then

$$\begin{aligned} \mathbf{P} \left(\bigcup_{j=N'_i+1, \dots, N \cdot h_1} V_j \right) &\leq \sum_{j=N'_i+1}^{N \cdot h_1} \mathbf{P}(V_j) \\ &\leq \sum_{j=N'_i+1}^{N \cdot h_1} \left(\frac{q^{j-1}}{q^{N \cdot h_1}} \right)^{|S_j|} \leq \sum_{j=N'_i+1}^{N \cdot h_1} q^{-\hat{N}} \\ &\leq \frac{1}{(N \cdot h_1)^{c-1}}. \end{aligned} \quad (92)$$

We now verify the total number of vectors by summing the number of vectors generated within each group.

$$\begin{aligned} \alpha(N, c, h_1) &= \sum_{j=N'_i+1, \dots, N \cdot h_1} N_j = \\ &N \cdot h_1 - \hat{N} + 1 + \hat{N} \sum_{k=1}^{\hat{N}-1} \frac{1}{k} \leq \\ &N \cdot h_1 + \\ &c \cdot \log_q(N \cdot h_1) (\ln(c \cdot \log_q(N \cdot h_1)) + 1). \end{aligned} \quad (93)$$

■

REFERENCES

- [1] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. Inform. Theory*, vol. 46, no. 4, pp. 1204–1216, 2000.
- [2] M. Médard and R. Koetter, "An algebraic approach to network coding," *IEEE/ACM Trans. Networking*, to be published.
- [3] P. Sanders, S. Egner, and L. Tolhuizen, "Polynomial time algorithms for network information flow," in *Proc. 15th ACM Symposium on Parallel Algorithms and Architectures (SPAA'03)*, 2003.
- [4] S.-Y. R. Li, R. W. Yeung, and N. Cai, "Linear network coding," *IEEE Trans. Inform. Theory*, 2002.
- [5] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. Cambridge, MA: MIT Press, 2001.
- [6] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, ser. Wiley Series in Telecommunications. New York, NY, USA: John Wiley & Sons, 1991.
- [7] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. New York, NY: North Holland Mathematical Library, 1977.
- [8] B. Carr and S. Vempala, "Randomized meta-rounding," *Random Structures and Algorithms*, vol. 20, no. 3, pp. 343–352, 2002.
- [9] K. Jain, M. Mahdian, and M. R. Salavatipour, "Packing steiner trees," in *Proc. 14th ACM SIAM Symposium on Discrete Algorithms (SODA'03)*, 2003.
- [10] K. Jansen and H. Zhang, "An approximation algorithm for the multicast congestion problem via minimum steiner trees," in *Proc. 3rd International Workshop on Approximation and Randomized Algorithms in Communication Networks (ARANCE'02)*, Rome, Italy, Sept. 2002.
- [11] S. Vempala and B. Vöcking, "Approximating multicast congestion," in *Proc. 10th International Symposium on Algorithms and Computation (ISAAC'99)*, Chennai, India, 1999, pp. 367–372.
- [12] R. Gallager, *Discrete Stochastic Processes*. Boston: Kluwer Academic Publishers, 1995.
- [13] M. Feder and N. Shulman, "The static broadcast channel," in *Proceedings of the International Symposium on Information Theory (ISIT'00)*, Italy, 2000.
- [14] N. Shulman and M. Feder, "Communication over an unknown channel via common broadcasting," Ph.D. dissertation, Faculty of EE - Systems, Tel-Aviv University, Tel Aviv, Israel, June 2003.
- [15] "Rfc 791 internet protocol," IETF, 1981. [Online]. Available: <http://www.ietf.org/rfc/rfc0791.txt>