An Asymptotically Optimal Push-Pull Method for Multicasting over a Random Network

Vasuki Narasimha Swamy, Srikrishna Bhashyam, Rajesh Sundaresan, and Pramod Viswanath

***Abstract*—We consider allcast and multicast flow problems where either all of the nodes or only a subset of the nodes may be in session. Traffic from each node in the session has to be sent to every other node in the session. If the session does not consist of all the nodes, the remaining nodes act as relays. The nodes are connected by undirected links whose capacities are independent and identically distributed random variables. We study the asymptotics of the capacity region (with network coding) in the limit of a large number of nodes, and show that the normalized sum rate converges to a constant almost surely. We then provide a decentralized push-pull algorithm that asymptotically achieves this normalized sum rate without network coding.**

***Index Terms*—allcast, broadcast, Erdős-Rényi random graph, flows, matching, multicast, network coding, random graph, Steiner tree, tree packing**

I. Introduction

In this paper, we investigate the capacity of allcast and multicast sessions over random link-capacitated graphs. Two questions motivated us to study these problems in the context of random graphs.

(1) While it is known that network coding in general provides a large coding advantage over multicast flows in directed graphs, Li et al. [1] showed that the coding advantage in undirected graphs is upper bounded by 2. In some specific topologies a tighter upper bound is known [2]. However several simulation experiments showed nearly no coding advantage for some class of random undirected graphs [3]. Is there a provable statement that there is negligible multicast coding advantage for a rich class of random undirected networks?

(2) If we stick to the domain of flows (with duplication), as we will soon see, optimal allcasting and multicasting lead to tree and Steiner tree packing problems respectively. While packing of trees is known to be easy (see [4], [5], [6]), Steiner tree packing is known to be hard [7]. Due to its application in multicasting over wired networks and in VLSI layout optimization, practitioners and theorists have over many years provided hardness results, heuristics, and approximation algorithms (see [8], [9], [7], [10], [11], etc.) Are there "quick-but-dirty" (terminology from [12]), decentralized, scalable, yet near-optimal algorithms for allcasting and multicasting over a rich class of random undirected networks?

In this paper, we provide affirmative answers to both these questions. We begin by making precise what we mean by allcast and multicast.

Allcast: Consider a setting where there are n nodes, all of which are engaged in a conference over a wired network. Each node has data that needs to be made entirely available over the network to each of the other $n-1$ nodes in a simultaneous fashion. (To be more precise, this is a *multiple* allcast problem). The data can be split, or routed, or coded, or transmitted in any combination thereof, so long as all nodes eventually get the information. The underlying complete undirected graph on n vertices is capacitated: each undirected link e has capacity C_e sampled independently and identically from a distribution F. An allcast information flow assignment is said to be feasible if for every link, the net (possibly coded) flow over the link (summed over both directions) respects the link's capacity constraint. For each feasible flow assignment, let r_i be the bit-rate of traffic sent by node i to each of the other nodes. We address the question of the set of all achievable rate tuples $r_1, \cdots, r_n$ in the asymptotics of a large number of nodes n. As we shall soon see, this problem is closely related to packing of disjoint spanning trees in a link-capacitated network with integer capacities. Minor extensions of some previous results readily yield that the achievable rate region is almost surely (a.s.)

$$\left\{(r_1, r_2, \ldots) \ : \ \limsup_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} r_i \leq \frac{1}{2}\mathbb{E}[C]\right\} \qquad (1)$$

where the expectation is of a random variable C having distribution F. The linear programming formulation of this problem is given in Section II, and the proof of (1) is given in Sections III (converse) and IV (achievability). Our proof of achievability is via a combination of "push" and "pull" that suggests a decentralized implementation. Section V contains some estimates needed to establish the correctness (with high probability) of the push-pull algorithm. Section VIII deals with the case when the link probabilities vanish, but not too quickly.

This paper was presented in part at the 2012 IEEE International Symposium on Information Theory (ISIT 2012) held at Cambridge, MA, USA.

Vasuki Narasimha Swamy is with the Department of EECS, University of California at Berkeley, CA, USA. Srikrishna Bhashyam is with the Department of Electrical Engineering at the Indian Institute of Technology Madras, Chennai, India. Rajesh Sundaresan is currently visiting the Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, IL, USA, and on leave of absence from the ECE Department of the Indian Institute of Science, Bangalore, India. Pramod Viswanath is with the Department of ECE and the Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, IL, USA.

This work was supported by the Department of Science and Technology, Government of India, by the University Grants Commission, India, by a fellowship awarded by the Indo-US Science and Technology Forum, and by the US National Science Foundation under grant CCF-1017430.

Parts of this work were carried out when (1) the first author was a student intern and the last author was on sabbatical leave at the Indian Institute of Science, (2) when the first author returned to complete her Bachelors project at the Indian Institute of Technology Madras, and (3) when the third author was on sabbatical leave at the University of Illinois at Urbana-Champaign. Supports from all these host institutions are gratefully acknowledged.

It is known that network coding does not yield any gain in allcast settings [1], and thus we have an asymptotic characterization of allcast capacity region.

Multicast: We next address a more general setting with only a subset of k_n nodes in the multicast session, where $\lim_{n\to\infty} k_n/n = \alpha$ and $0 \le \alpha \le 1$. Data from each of the k_n nodes has to reach every one of the other $k_n - 1$ nodes. The remaining $n - k_n$ nodes serve as relays. This is therefore a problem of *multiple* multicast among common *session nodes*. Again, in a link-capacitated framework where each link is independent and identically distributed (iid) with distribution F, we are interested in the set of all achievable rate tuples $r_1, \cdots, r_{k_n}$ in the asymptotics of a large number of nodes n. We demonstrate that the capacity region is almost surely

$$\left\{ (r_1, r_2, \ldots) \ : \ \limsup_{n\to\infty} \frac{1}{n} \sum_{i=1}^{k_n} r_i \le \left(1 - \frac{\alpha}{2}\right) \mathbb{E}[C]. \right\} \quad (2)$$

The LP formulation of this problem is in Section II, proof of the converse is in Section III, and proof of achievability is in Section VII. Here too, our proof of achievability is via a decentralized push-pull algorithm. Section VI is a digression to study single commodity flows over random networks and develops the ingredients necessary to establish the correctness (with high probability) of the push-pull algorithm.

Our achievability proofs are based on flows (allowing for duplications) and thus do not employ network coding. In particular, they establish that any gain from network coding in multicast settings is at best sublinear in the number of nodes.

II. A Linear Programming Formulation

A. Random graph models

We are given a countable collection of iid random variables $\{C_{i,j}, 1 \le i < j < \infty\}$ where each element has distribution F on $\mathbb{R}_+$. We then obtain a sequence of graphs, denoted $\{K_n, n \ge 1\}$, where for each n, the graph K_n is the complete graph on the vertex set $\{1, 2, \ldots, n\}$ along with the collection of all $\binom{n}{2}$ links. Each link (i, j) with $1 \le i < j \le n$ has link capacity $C_{i,j}$.

Later on, we will have a need to study Erdős-Rényi random graphs where the link capacity distribution is Bernoulli(p), which is $\Pr\{C = 1\} = p$ and $\Pr\{C = 0\} = 1 - p$. If $C_{i,j} = 0$, then the undirected link (i, j) has zero capacity and is effectively absent. We then use the notation $G(n, p)$ to denote the obtained graph for a fixed n.

We will also study Erdős-Rényi random graphs where p depends on n and vanishes with n. We shall denote these $G(n, p_n)$. These may be constructed as follows. We assume that we are now given a collection of iid random variables $\{Z_{i,j}, 1 \le i < j < \infty\}$ where each $Z_{i,j}$ has the uniform distribution on $[0, 1]$. The graph $G(n, p_n)$ is the graph on n vertices $\{1, 2, \ldots, n\}$ where each link $\{i, j\}$ with $1 \le i < j \le n$ has binary capacity $C_{i,j} = \mathbf{1}\{Z_{i,j} \le p_n\}$. The notation $\mathbf{1}\{\cdots\}$ stands for the indicator of an event. This construction is of course consistent with the construction of $G(n, p)$ when $p_n \equiv p$ is a constant.

Finally, we will also study random bipartite graph sequences $\{G(n, n, p), n \ge 1\}$ and $\{G(n, n, p_n), n \ge 1\}$. These are constructed from the collection of iid random variables $\{Z_{i,j}, i \ge 1, j \ge 1\}$ where once again each entry has the uniform distribution on $[0, 1]$. In the graph $G(n, n, p_n)$, for example, there are $2n$ vertices with vertex set $V_1 \cup V_2$ where $V_1 = \{v_1, v_2, \ldots v_n\}$ and $V_2 = \{\omega_1, \omega_2, \ldots \omega_n\}$, and the capacity on the link between node v_i and node ω_j is $C_{i,j} = \mathbf{1}\{Z_{i,j} \le p_n\}$.

B. Allcast

Consider the allcast problem described in Section I. Li et al. prove in [1, Cor. 4.a] that a multiple allcast rate vector $(r_1, r_2, \ldots, r_n)$ is achievable in an undirected capacitated network if and only if the rate vector $(\sum_{i=1}^n r_i, 0, \ldots, 0)$ is achievable, i.e., the sum rate is achievable for a *single* allcast with node 1 as sender and with the other $n - 1$ nodes as receivers. This is intuitively clear since network coding does not help for allcast, and one can make do with multicommodity flows in multiple allcast.

We may therefore assume that there is only one sender (say node 1), and all other $n - 1$ nodes are recipients that must receive all information sent by node 1. The rates in such a setting are given by $(r_1, 0, 0, \ldots)$, and we characterize r_1.

This maximum rate is obtained by solving the following linear programming (LP) problem. Consider the graph K_n on n vertices with associated link capacities. Let $\mathcal{T}_n$ be the set of all spanning trees on the complete graph (ignoring capacities). The vertices are labeled, and so Cayley's formula tells that the number of such trees is n^{n-2}. Solve the LP (Tutte [4], Nash-Williams [5], Barahona [6], Li et al. [1]):

$$\begin{aligned} \text{Maximize} \quad & \sum_{T \in \mathcal{T}_n} \lambda_T & (3) \\ \text{subject to} \quad (a) \quad & \sum_{T \in \mathcal{T}_n : T \ni e} \lambda_T \le C_e \quad \text{for all } e \\ (b) \quad & \lambda_T \ge 0 \quad \text{for all } T \in \mathcal{T}_n. \end{aligned}$$

Denote the maximum value of (3) as π_n. Then π_n is the maximum rate at which node 1 can allcast its information to all the other nodes. The LP has a simple and intuitive explanation.

- If one tags an infinitesimal information element originating at node 1 and follows the path of its spread to each of the $n - 1$ recipients, one gets a directed graph rooted at the source node 1 and spanning all the n nodes.
- If the undirected version of this directed graph is not a tree, i.e., there is some cycle, then some node in the cycle is receiving this information element from two other nodes. One of these two incoming links can be removed without affecting the allcast property. We can thus reduce the directed graph to a *spanning arborescence*, which is a directed graph with no incoming links at the root node, exactly one incoming link at every other node, and all vertices are covered.
- This spanning arborescence is in one-one correspondence with a tree, because the root is specified as node 1. So we may simply focus on the spanning tree associated with the arborescence. Call this tree T (which is in $\mathcal{T}_n$).
- Collect all information elements that are spread via this tree. Call its volume λ_T.

It is clear that each $\lambda_T \geq 0$ and constraint (a) in (3) is the capacity constraint associated with each of the links. Consequently, the value of the optimization problem in (3) is an upper bound on the optimal net flow from node 1. But it is immediate that any set of λ_T satisfying the two constraints provides a means to achieve a rate $\sum_T \lambda_T$, since λ_T units of information may be directed through the spanning arborescence associated with the tree T and root vertex 1. Thus the maximum rate of allcast flow from a single sender is π_n, the solution to the LP in (3).

When link capacities are random, π_n is a random variable whose asymptotics we shall soon characterize.

C. Multicast

For the multicast problem, without loss of generality, let us index the session nodes as $\{1, 2, \ldots, k_n\}$. As for allcast, by [1, Cor. 4.a], a multiple multicast rate vector $(r_1, r_2, \ldots, r_{k_n})$ with identical session nodes is achievable in an undirected capacitated network if and only if the rate vector $(\sum_{i=1}^{k_n} r_i, 0, \ldots, 0)$ is achievable, i.e., the sum rate is achievable for a *single* multicast with node 1 as sender and with the other $k_n - 1$ nodes of the session as receivers[1]. We may therefore assume that there is but one sender, he is node 1, and all other $k_n - 1$ nodes are recipients that must receive all information sent by node 1. Denote by $\mathcal{T}_n(k_n)$ the set of all Steiner trees that span the vertices $1, 2, \ldots, k_n$. Obviously $\mathcal{T}_n(n) \equiv \mathcal{T}_n$. For multicast, again as for allcast, the maximum simultaneously transmissible rate from one sender (node 1) to the $k_n - 1$ other recipients is the maximum value of the modified LP ([3], [13], [1]):

$$\begin{aligned} \text{Maximize} \quad & \sum_{T \in \mathcal{T}_n(k_n)} \lambda_T & (4) \\ \text{subject to} \quad (a) \quad & \sum_{T \in \mathcal{T}_n(k_n): T \ni e} \lambda_T \leq C_e \quad \text{for all } e \\ (b) \quad & \lambda_T \geq 0 \qquad\qquad \text{for all } T \in \mathcal{T}_n(k_n). \end{aligned}$$

Set $\alpha_n = k_n/n$, and denote the maximum value of (4) as $\pi_n(\alpha_n)$. The above LP is the same as that of (3) with $\mathcal{T}_n$ replaced by the less restrictive $\mathcal{T}_n(k_n)$.

Again, when link capacities are random, $\pi_n(\alpha_n)$ is a random variable whose asymptotics we shall soon characterize.

III. An Upper Bound

Consider the following definitions.

- Let χ_n and $\chi_n(k_n)$ denote the *maximum throughput achievable* in the allcast and multicast settings with the added possibility of network coding at each node. (The dependence of these quantities on the link capacities is understood and suppressed).
- Let η_n denote the *strength* of the allcast network defined as follows. Let $\mathcal{P}$ denote the set of all partitions of the vertex set $\{1, 2, \ldots, n\}$. Consider a partition $\wp \in \mathcal{P}$. Let $\partial\wp$ denote the set of intercomponent links. Define
$$\eta_n := \min_{\wp \in \mathcal{P}} \frac{\sum_{e \in \partial\wp} C_e}{|\wp| - 1} \tag{5}$$
where $|\wp|$ denotes the number of subsets in the partition.
- Let $\eta_n(k_n)$ denote the strength of the multicast network with k_n nodes in the session. This is defined as follows. Let $\mathcal{P}(k_n)$ denote the set of all partitions of the vertex set $\{1, 2, \ldots, n\}$ such that each component of a partition contains at least one of the session nodes $\{1, 2, \ldots, k_n\}$. Define
$$\eta_n(k_n) := \min_{\wp \in \mathcal{P}(k_n)} \frac{\sum_{e \in \partial\wp} C_e}{|\wp| - 1}. \tag{6}$$

Li et al. [1] showed the following result.

Theorem 1: (Li et al. [1, Th. 2 and Th. 3])
(a) For any allcast session, $\pi_n = \chi_n = \eta_n$.
(b) For any multicast session, $\pi_n(k_n) \leq \chi_n(k_n) \leq \eta_n(k_n)$. □

We can easily find good upper bounds on η_n and $\eta_n(k_n)$ in random settings as shown in the following theorem.

Theorem 2: Let $\{C_{i,j}\}_{1 \leq i < j \leq n}$ denote the undirected link capacities. We then have the following upper bounds:

$$\eta_n \leq \frac{1}{n-1} \sum_{1 \leq i < j \leq n} C_{i,j} \tag{7}$$

$$\eta_n(k_n) \leq \frac{1}{k_n - 1} \left(\sum_{i < k_n} \sum_{j \geq k_n} C_{i,j} + \sum_{1 \leq i < j < k_n} C_{i,j} \right) \tag{8}$$

As a consequence, with $\lim_{n\to\infty} k_n/n = \alpha$, the inequalities

$$\limsup_{n\to\infty} \frac{\eta_n}{n} \leq \frac{1}{2}\mathbb{E}[C] \tag{9}$$

$$\limsup_{n\to\infty} \frac{\eta_n(k_n)}{n} \leq \left(1 - \frac{\alpha}{2}\right)\mathbb{E}[C] \tag{10}$$

hold almost surely. □

Proof: Consider the partition $\wp = \{\{1\}, \{2\}, \cdots, \{n\}\}$. There are n subsets in the partition, and $\partial\wp$ is the set of all links. Apply now the definition (5) of η_n and we immediately get (7) as the upper bound for the allcast case.

For the multicast case, consider the partition

$$\wp = \{\{1\}, \{2\}, \cdots, \{k_n - 1\}, \{k_n, \ldots, n\}\}.$$

There are k_n subsets in the partition. The set of links in $\partial\wp$ are

$$\{(i,j) : 1 \leq i < k_n, j \geq k_n\} \cup \{(i,j) : 1 \leq i < j < k_n\}.$$

Apply now the definition (6) of $\eta_n(k_n)$ and we immediately get (8) as the upper bound for the multicast case.

Note that $|\partial\wp| = n(n-1)/2$ for allcast, and

$$\begin{aligned} |\partial\wp| &= (k_n - 1)(n - k_n + 1) + \frac{(k_n - 1)(k_n - 2)}{2} \\ &= (k_n - 1)\left(n - \frac{k_n}{2}\right) \end{aligned} \tag{11}$$

for multicast.

[1]There is some subtlety involved here since, in general, network coding provides a coding advantage for multicasting in undirected networks; see [1, Th. 4] for a proof of source independence in the single multicast case which is then generalized to get [1, Cor. 4.a]

Using $|\partial\wp| = n(n-1)/2$ for allcast in (7), we obtain

$$\frac{\eta_n}{n} \leq \frac{1}{2}\frac{1}{|\partial\wp|}\sum_{e\in\partial\wp} C_e.$$

The sum on the right-hand side is composed of independent and identically distributed random variables. Consequently, the right-hand side converges almost surely to $\frac{1}{2}\mathbb{E}[C]$ by the strong law of large numbers, and we obtain (9).

For the multicast case, use (11) in (8) to obtain

$$\frac{\eta_n(k_n)}{n} \leq \left(1-\frac{k_n}{2n}\right)\frac{1}{|\partial\wp|}\sum_{e\in\partial\wp} C_e.$$

Again by an application of the strong law of large numbers, the conclusion (10) follows. ∎

Observe that, by Theorem 1, the upper bounds in Theorem 2 apply for capacity with the possibility of network coding. Let us now turn to achievability of these rates in their respective settings.

IV. Allcast: Achievability

In this section we consider the allcast setting and argue that the upper bound in (9) is tight, and moreover, the upper bound is achievable via flows. After first establishing the existence of a scheme, we then provide a practical decentralized asymptotically optimal push-pull algorithm.

Theorem 3: For the allcast problem, we have

$$\lim_{n\to\infty}\frac{\pi_n}{n} = \frac{1}{2}\mathbb{E}[C] \quad \text{a.s.}$$

□

Proof: The fact that we cannot do better than $\mathbb{E}[C]/2$ was already established in (9). So the proof of the above theorem would be complete if we can establish that $\mathbb{E}[C]/2$ is achievable. We first argue achievability on the simpler Erdős-Rényi graphs. We then lift this result to the general case.

Take the random graph $G(n,p)$ where each link capacity is iid with Bernoulli(p) distribution. Catlin et al. [14, Sec. 3] proved the stronger result that, even if p vanishes with n, so long as it is larger than $(28\log n/n)^{1/3}$, we have for all sufficiently large n the equality

$$\pi_n = \left\lfloor \frac{\sum_{1\leq i<j\leq n} C_{i,j}}{n-1} \right\rfloor \quad \text{a.s.} \tag{12}$$

For any $\varepsilon > 0$, using $p > 0$, the result in (12), and the strong law of large numbers, we have

$$\liminf_{n\to\infty}\frac{\pi_n}{n} \geq \frac{p}{2}(1-\varepsilon) \quad \text{a.s.} \tag{13}$$

By excluding all null sets associated with rational $\varepsilon \in (0,1)$, it follows that

$$\liminf_{n\to\infty}\frac{\pi_n}{n} \geq \frac{p}{2} \quad \text{a.s.}$$

There now remains the step of lifting this result to any generic distribution F, for the iid capacities $C_{i,j}$, satisfying

$$0 < \mathbb{E}[C] = \int_0^\infty \Pr\{C > x\}\, dx = \int_0^\infty [1-F(x)]\, dx < \infty. \tag{14}$$

This is readily done. Fix an arbitrary $\varepsilon > 0$. By (14) and the fact that the function $1-F(x)$ is Riemann integrable (for it is Lebesgue integrable, bounded, and has at most a countable number of discontinuities), we can choose a natural number $M < \infty$ and $\delta > 0$ such that

$$\sum_{k=1}^{M} \delta\cdot[1-F(k\delta)] \quad \geq \quad \mathbb{E}[C]\cdot(1-\varepsilon). \tag{15}$$

We now build a family of M coupled graphs, each with n vertices. For a realization of the iid link capacities, let G_k be a new graph on the n vertices with link between i and j if and only if $C_{i,j} > k\delta$, for $k = 1, 2, \ldots, M$. Clearly, G_k is an Erdős-Rényi graph on n vertices with parameter

$$p(k) := \Pr\{C > k\delta\} = 1 - F(k\delta).$$

On G_k, we interpret each link, if present, as having capacity δ. While the graphs are coupled across the parameter k, for a fixed k, the links on the graph G_k are iid Bernoulli($p(k)$) random variables. Let $\pi_n(G_k)$ be the maximum number of disjoint trees that can be packed in G_k. By the result (13) applied to each fixed k, we have

$$\begin{aligned}
\liminf_{n\to\infty}\frac{\pi_n}{n} &\geq \liminf_{n\to\infty}\frac{1}{n}\sum_{k=1}^{M}\delta\cdot\pi_n(G_k)\\
&\geq \delta\cdot\sum_{k=1}^{M}\frac{p(k)}{2}(1-\varepsilon), \quad \text{a.s.}\\
&= \frac{1}{2}\sum_{k=1}^{M}\delta\cdot[1-F(k\delta)]\cdot(1-\varepsilon)\\
&\geq \frac{1}{2}\cdot\mathbb{E}[C]\cdot(1-\varepsilon)\cdot(1-\varepsilon)\\
&\geq \frac{\mathbb{E}[C]}{2}(1-2\varepsilon),
\end{aligned}$$

where the penultimate inequality follows from (15). It follows as before that $\lim_n \frac{\pi_n}{n} \geq \frac{\mathbb{E}[C]}{2}$ almost surely. This completes the proof. (See [15] or [16] for a similar truncation, quantization, and scaling argument). ∎

The key to proving Theorem 3 is the result (13) on Erdős-Rényi graphs. In order to show this, we utilized the result (12) of Catlin et al. [14]. The main point of the rest of this section is to demonstrate that (13) can be proved constructively using a rather simple and decentralized algorithm.

A. ALLCAST: A decentralized algorithm for allcast in a random graph

In this section we describe a decentralized *push-pull* algorithm for allcast that achieves (13) for an arbitrary $\varepsilon > 0$. For ease of exposition, we shall assume a total of $n+1$ nodes with node 0 as the source node. The source node 0 has to push a total of $\frac{1}{2}np(1-\varepsilon)$ bits to all nodes. We have ignored integer rounding and a factor $(n+1)/n$ both of which are easily absorbed into ε. The algorithm broadly has two push steps and two pull steps, as described next. See Figure 1. The analysis that comes later will argue that with overwhelming probability none of the steps fail.

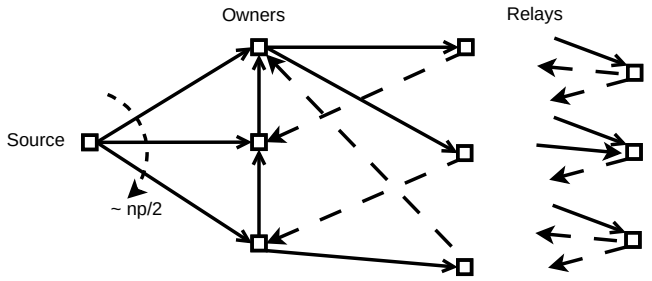

Figure 1. Graph showing the three sets of nodes: source, owners, and relays. Source pushes bits to owners who then push to relays. All nodes then pull from owners and any remaining bits from relays.

Algorithm ALLCAST:

- *Setting up of directions*: All links that do not involve the source node 0 are assigned one of the two directions with equal probability, independently of the choices of directions at other links. All links that involve the source node 0 have a direction pointing away from the source.
- *Push step 1*: Source node 0 *pushes* $\frac{1}{2}np(1-\varepsilon)$ different bits to that many of its neighbors. We number the bits $b_1, b_2, \ldots, b_{np(1-\varepsilon)/2}$, call the respective recipient nodes as *owners* of these bits, and denote the owners (sometimes) as $O_1, O_2, \ldots, O_{np(1-\varepsilon)/2}$ instead of saying node 1, node 2, ..., node $np(1-\varepsilon)/2$. There may be several other neighbors of node 0, but the corresponding links are left unused. These and other nodes who are not owners are called *relays*, and are denoted $R_{np(1-\varepsilon)/2+1}, \ldots, R_n$ (instead of saying node $np(1-\varepsilon)/2+1$, ..., node n).
- *Push step 2*: Each owner O_i pushes his bit b_i one more level along links that point outward from i, regardless of the status of the recipient as an owner of another bit or a relay. The receiving node will then have b_i (and similarly many other bits) for other nodes to pull in the next couple of steps of the algorithm.
- *Pull step 1*: Each node, say node j, collects all incoming bits b_i coming directly from owners O_i via links $i \to j$. (This is the bit pushed by O_i in push step 2).
- *Pull step 2*: Having collected some bits directly from owners, node j identifies the remaining bits, the relays to which it is connected with direction pointing towards j, and the bits that these relays have available having received the bits directly from owners. A representation of this information is the *bit-map* matrix of nodes and bits they have available for pulling (see Table I and its description). Node j then identifies a *complete matching* of these desired bits to the helper relays: each desired yet-to-be-pulled bit is pulled from a suitable relay that has the bit, with each relay accounting for one bit, and this constitutes a matching. □

Before we dive into an analysis of this algorithm, we describe the bit-map of Table I in more detail. The rows and columns are indexed as

$$O_1, O_2, \ldots, O_{np(1-\varepsilon)/2}, R_{np(1-\varepsilon)/2}+1, \ldots, R_n.$$

In addition, the first $np(1-\varepsilon)/2$ columns will also refer to the corresponding bits.

- For $1 \le i \le np(1-\varepsilon)/2$, we write $X_{i,i} = 1$ to signify that node O_i has bit b_i.
- For $i \ne j$, since the link $\{i, j\}$ itself occurs with probability p, and further, may have either direction with equal probability, we have

$$\begin{aligned} X_{i,j} = 1, X_{j,i} = 0 \qquad & \text{if } j \to i; \\ X_{j,i} = 0, X_{i,j} = 1 \qquad & \text{if } i \to j; \\ X_{j,i} = 0, X_{i,j} = 0 \qquad & \text{if no link between } i \text{ and } j. \end{aligned}$$

  These are mutually exclusive, with the first setting occurring with probability $p/2$, the second setting with probability $p/2$, and the third setting with probability $1-p$.
- If $X_{i,j} = 1$, then node i (owner or relay) can obtain bit b_j from owner O_j (if $1 \le j \le np(1-\varepsilon)/2$) or some bit that relay R_j has (if $j > np(1-\varepsilon)/2$).
- The set of bits node i receives directly from owners corresponds to the set of 1s in the first $np(1-\varepsilon)/2$ columns of the ith row, for if $X_{i,j} = 1$, then owner O_j pushes his bit b_j to node i. (For example, in Table I, owner O_t has bits $b_1, b_2, b_t, b_{np(1-\varepsilon)/2}$, but does not have b_a, b_b, b_c).
- The 1s in the ith row beyond column $np(1-\varepsilon)/2$ point to relays that can be used by node i to pull any remaining bits in pull step 2. (For example, owner O_t is connected to relays R_u, R_v, R_w with directions pointing towards O_t. These relays will help node O_t get the yet-to-be-pulled bits b_a, b_b, b_c).
- Clearly, while the random variables $X_{i,j}$ and $X_{j,i}$ are coupled, the nondiagonal entries of the ith row

$$\{X_{i,j}, 1 \le j \le n, j \ne i\}$$

  are iid Bernoulli($p/2$) random variables, for $1 \le i \le n$. The same holds for nondiagonal entries of any column.

Our main assertion is that the algorithm ALLCAST succeeds with high probability in distributing the $np(1-\varepsilon)/2$ bits to all nodes.

Theorem 4: For any $\varepsilon > 0$, the following event occurs almost surely: for all but finitely many n, the algorithm ALLCAST succeeds in distributing all $np(1-\varepsilon)/2$ bits to each of the n nodes. □

Remarks: 1) It follows immediately that, for any $\varepsilon > 0$, the inequality (13) holds.

2) The above theorem also implies that, for all sufficiently large n, we can pack $np(1-\varepsilon)/2$ disjoint (spanning) trees in $G(n,p)$, with each tree having the property that it has depth at most 3.

3) ALLCAST is decentralized in the following sense. The direction of each link, when present and if the source node is not involved, is picked at random by the toss of a fair coin, and this information is needed only at these two incident nodes. The two levels of pushes, and thus the first pull stage, are easily seen to be decentralized. At each node, the actions depend only on the links incident on it and the agreed upon link directions. Each node then keeps a list of bits it receives from owners. For the final pull stage, each node has to get this list associated with each of its potential helper relays. This is

Table I
ALLCAST BIT-MAP

	O_1 b_1	O_2 b_2	$\cdots$	O_t b_t	$\cdots$	O_a b_a	O_b b_b	O_c b_c	$\cdots$	$O_{\frac{np(1-\varepsilon)}{2}}$ $b_{\frac{np(1-\varepsilon)}{2}}$	$R_{\frac{np(1-\varepsilon)}{2}+1}$	$\cdots$	R_u	R_v	R_w	$\cdots$	R_n
O_1	1		$\cdots$			X_{1a}			$\cdots$								
O_2		1															
$\vdots$			$\ddots$														
O_t	1	1	$\cdots$	1	$\cdots$	0	0	0	$\cdots$	1	0	$\cdots$	1	1	1	$\cdots$	0
$\vdots$					$\ddots$												
O_a	X_{a1}		$\cdots$			1			$\cdots$		X_{ai}						
O_b							1										
O_c								1									
$\vdots$									$\ddots$								
$O_{\frac{np(1-\varepsilon)}{2}}$										1							
$R_{\frac{np(1-\varepsilon)}{2}+1}$						X_{ia}											
$\vdots$				$\vdots$													
R_u				0	$\cdots$		1		$\cdots$								
R_v				0	$\cdots$			1	$\cdots$								
R_w				0	$\cdots$	1			$\cdots$								
$\vdots$				$\vdots$													
R_n										1							

the step that may involve significant exchange of information, but the cost involved is a one-time set-up cost that can be amortized over multiple rounds of data communication. Note that all information exchanges (link directions, pushing of owned bits, lists of bits available at neighboring helper relays) are of information which are of local relevance that are, in addition, locally available. The matching can be identified in $O(n^2)$ steps [12].

4) We need three elementary tools to establish the result. The first is the following well known concentration result for the binomial distribution, which we state without proof.

Lemma 5: ([17, Th. 1.7(i)]) Suppose $0 < q < \frac{1}{2}$, $0 < \varepsilon < 1/12$, and $\varepsilon nq(1-q) \geq 12$. Let $S_{n,q}$ be the sum of n Bernoulli(q) random variables. Then

$$\Pr\left\{\left|\frac{1}{nq}S_{n,q} - 1\right| > \varepsilon\right\} \leq \frac{1}{\sqrt{\varepsilon^2 nq}} e^{-nq\varepsilon^2/3}. \qquad (16)$$

□

This result holds for every n and q satisfying $\varepsilon nq(1-q) \geq 12$, and as such, q can vary with n. The second tool is the Borel-Cantelli lemma that gives us a sufficient condition for almost sure convergence. The third tool is one of existence of matchings on random bipartite graphs, which will be the subject of Section V.

Proof of Theorem 4: By the Borel-Cantelli lemma, it suffices to show that the probability that the algorithm fails for a particular n is summable over n. If the algorithm fails, then at least one of the following is true.

1) The event $A_1^{(n)}$ occurs, which is defined to be the event that there are fewer than $\frac{1}{2}np(1-\varepsilon)$ vertices connected to node 0. By Lemma 5, there is some $c_1 > 0$ such that for all sufficiently large n, we have $\Pr\{A_1^{(n)}\} \leq e^{-c_1 n}$.

2) For some node t, the event $A_2^{(n)}(t)$ occurs, which is defined to be the event that the node t is connected to a certain number of owners outside the range $\frac{1}{2}np(1-\varepsilon) \cdot \frac{1}{2}p(1 \pm \varepsilon)$ with links pointing towards t. (If node t is an owner, there are $\frac{1}{2}np(1-\varepsilon))-1$ other owners, but the 1 can be absorbed into the $(1-\varepsilon)$ factor). Again by Lemma 5, there is some $c_2 > 0$ such that for all sufficiently large n, we have $\Pr\{A_2^{(n)}(t)\} \leq e^{-c_2 n}$.

3) For some node t, the event $A_3^{(n)}(t)$ occurs, which is the event that the node t is connected to fewer than

$$\begin{aligned} \beta_n &:= \left(n - \frac{1}{2}np(1-\varepsilon)\right) \cdot \frac{1}{2}p(1-\varepsilon) \\ &= \frac{1}{2}np(1-\varepsilon) \cdot \left(1 - \frac{1}{2}p(1-\varepsilon)\right) \end{aligned}$$

relays with links pointing towards t. (Again, the case of 1 less relay when node t is a relay is easily handled). Once again by Lemma 5, there is a $c_3 > 0$ such that for all sufficiently large n, we have $\Pr\{A_3^{(n)}(t)\} \leq e^{-c_3 n}$.

4) For some node t, if $A_1^{(n)} \cup A_2^{(n)}(t) \cup A_3^{(n)}(t)$ does not occur, then the event $M^{(n)}(t)$ occurs, which is the event that

node t is unable to pull the desired bits. We claim that

$$\Pr\left\{M^{(n)}(t) \mid \left(A_1^{(n)} \cup A_2^{(n)}(t) \cup A_3^{(n)}(t)\right)^c\right\} \leq \gamma(\beta_n) \tag{17}$$

for some sequence $\gamma : \mathbb{N} \to [0,1]$ satisfying

$$\sum_{n=1}^{\infty} n\gamma(\beta_n) < \infty. \tag{18}$$

The event that the algorithm fails is then a subset of

$$A_1^{(n)} \bigcup_{t=1}^{n} \left(A_2^{(n)}(t) \cup A_3^{(n)}(t) \cup M^{(n)}(t)\right)$$

whose probability is upper bounded via the union bound and (17) by

$$n \cdot \left(e^{-nc_1} + e^{-nc_2} + e^{-nc_3} + \gamma(\beta_n)\right)$$

which, by the summability claim in (18) and the exponentially decaying nature of the other terms, is summable.

Let us now prove (17) and (18).

Fix a node t, where $1 \leq t \leq n$. The event $A_1^{(n)}$ has not occurred, and so the source has sent out exactly $\frac{1}{2}np(1-\varepsilon)$ bits to that many owners. The event $A_2^{(n)}(t)$ has not occurred, and so node t is connected to between $\frac{1}{2}np(1-\varepsilon) \cdot \frac{1}{2}p(1 \pm \varepsilon)$ owners with links towards node t. The connected owners directly furnish their bits to node t. But node t needs at least $\frac{1}{2}np(1-\varepsilon) - \frac{1}{2}np(1-\varepsilon) \cdot \frac{1}{2}p(1+\varepsilon)$ additional bits to be pulled in pull step 2. This set of yet-to-be-pulled bits points to some random selection of columns from amongst the first $\frac{1}{2}np(1-\varepsilon)$ columns and does not include column t.

The event $A_3^{(n)}(t)$ has not occurred, and so node t is connected to at least β_n relays that could potentially furnish these missing bits (that is, with links towards node t). Consider the rows corresponding to these relays. This set of rows is a random selection of at least β_n rows from amongst the indices $\frac{1}{2}np(1-\varepsilon)+1$ through n and does not include t.

Observe that conditioned on these selections, the entries of the submatrix continue to be iid Bernoulli($p/2$) random variables. If $M^{(n)}(t)$ occurs, there is no coverage of these the yet-to-be-pulled bits (columns) using the helper relays (rows), with each helper relay furnishing at most one missing bit. But this in particular implies that there is no coverage of the yet-to-be-pulled bits (columns) by some subset of exactly β_n helper relays (rows) with each helper relay furnishing at most one bit. But this further implies that any superset of β_n columns that includes the yet-to-be-pulled bits (columns), and continues to exclude column t, cannot be *matched* to the selected β_n helper relays (rows). Now, Lemma 9 of Section V shows that this probability is upper bounded by $\gamma(\beta_n)$, which is (17), and that $n\gamma(\beta_n)$ is summable, which is (18). This concludes the proof. ∎

The matching step above is the key to complete the deliveries. It ensures that all required bits are available at some helper relay, and that each link has at most 1 bit load so that capacity constraints are not violated. We now devote a section to demonstrating this key step.

V. The existence of a bipartite matching

In this section, we establish the crucial step of existence of bipartite matchings. The following lemma, taken from Bollobás [17], is key to showing that matchings exist almost surely and one can pull the β_n bits from relays. We first present the result for a random bipartite graph with n vertices on each side. The results of this section are well-known and are provided only for completeness and ease of reference.

Lemma 6: ([17, Lem. 7.12, p. 174]). Let G be a bipartite graph with vertex sets V_1, V_2 such that $|V_1| = |V_2| = n$. Suppose G does not have any isolated vertices and it does not have a complete matching. Then there is a set $A \subset V_i$ for either $i = 1$ or 2 such that the following three conditions hold:

(i) $\Gamma(A)$ has $|A| - 1$ elements,
(ii) the subgraph spanned by $A \cup \Gamma(A)$ is connected,
(iii) $2 \leq |A| \leq (n+1)/2$. □

The above conditions are simple consequences of Hall's marriage theorem and some elementary observations. The proof can be found in [17, Lem. 7.12, p. 174]. We now bound the probability of these events on a random bipartite graph $G(n,n,p)$ (see Section II-A).

Lemma 7: Let F_a be the event that there is a set A of size a with $A \subset V_i$ for $i = 1$ or 2 satisfying (i)-(iii) of Lemma 6. Let $n_1 = (n+1)/2$. Consider $G(n,n,p)$. Then $\Pr\{\cup_{a=2}^{n_1} F_a\} \leq \varepsilon_n$ where ε_n summable, and hence $\varepsilon_n \to 0$. Furthermore, we also have $\sum_{n\geq 1} n\varepsilon_n < \infty$. □

Proof: Fix a. There are two choices for i in the condition $A \subset V_i$, there are $\binom{n}{a}$ ways to choose the subset A, and there are $\binom{n}{a-1}$ ways to choose the subset $\Gamma(A)$. Once chosen, there must be no links between the a vertices of A and the $n-a+1$ vertices of $V_2 - \Gamma(A)$. By the union bound (for the possibilities for A and $\Gamma(A)$), we get

$$\Pr\{F_a\} \leq 2\binom{n}{a}\binom{n}{a-1}(1-p)^{a(n-a+1)}. \tag{19}$$

Using $\binom{n}{a} \leq n^a$, by a second application of the union bound, and by dropping some factors that are smaller than 1, we get

$$\Pr\{\cup_{a=2}^{n_1} F_a\} \leq 2\sum_{a=2}^{n_1} n^{2a-1}(1-p)^{an}(1-p)^{-a^2} =: \varepsilon_n. \tag{20}$$

For an a_0, set $n_0 = 2a_0 - 1$. It suffices to show that for n_0 large, $\sum_{n\geq n_0} \varepsilon_n < \infty$. Interchanging the indices of summation, and changing limits appropriately, we get

$$\begin{aligned}\sum_{n\geq n_0} \varepsilon_n &= 2\sum_{a=2}^{a_0}(1-p)^{-a^2} \sum_{n\geq n_0} n^{2a-1}(1-p)^{an} \\ &\quad + 2\sum_{a>a_0}(1-p)^{-a^2} \sum_{n\geq 2a-1} n^{2a-1}(1-p)^{an}.\end{aligned} \tag{21}$$

The first term is easily seen to be summable for any finite a_0. For the second one, observe that for any $\delta > 0$ and any $C > 0$, there is an a_0 large enough so that for all $a > a_0$ and all $n \geq 2a-1$, we have $n^{2a-1} \leq n^{2a} \leq C(1+\delta)^{an}$. By

taking $C = (1-p)(1-\delta)(1-(1-p)(1-\delta))$ it follows that

$$\sum_{n \geq 2a-1} n^{2a-1}(1-p)^{an} \leq (1-p)^{2a^2}(1+\delta)^{2a^2}.$$

Choose δ small enough so that $(1-p)(1+\delta)^2 < 1$. Substitute this in the second term in (21), and we see that it is summable.

Finally, to show that $\sum_{n \geq 1} n\varepsilon_n < \infty$, we modify (21) as

$$\begin{aligned}\sum_{n \geq n_0} n\varepsilon_n &= 2\sum_{a=2}^{a_0}(1-p)^{-a^2} \sum_{n \geq n_0} n^{2a}(1-p)^{an} \\ &\quad + 2\sum_{a > a_0}(1-p)^{-a^2} \sum_{n \geq 2a-1} n^{2a}(1-p)^{an}.\end{aligned}$$

By our choice of a_0 and δ, we also have $n^{2a} \leq C(1+\delta)^{an}$, and so all the steps that followed (21) apply, which establishes summability of $n\varepsilon_n$. ∎

We now put these together to argue that a bipartite matching exists in $G(n,n,p)$ with high probability.

Theorem 8: The probability that $G(n,n,p)$ does not have a complete matching is upper bounded by $\gamma(n) := 2n(1-p)^n + \varepsilon_n$, where ε_n, defined in (20), has all the properties indicated in Lemma 7. □

Proof: If $G(n,n,p)$ does not have a complete matching, then either (1) there is an isolated vertex, or (2) there is no isolated vertex and by virtue of Lemma 6, $\cup_{a=1}^{n_1} F_a$ must occur, where $n_1 = (n+1)/2$ as before. By Lemma 7, the probability of the second case event is at most ε_n. The probability that there is no isolated vertex is, by the union bound, at most $2n(1-p)^n$. ∎

In the previous section, we had a need to study existence of bipartite matchings over left and right sets of size $\beta_n := \lfloor cn \rfloor$ where $0 < c < 1$.

Lemma 9: For a fixed $0 < c < 1$, let $\beta_n := \lfloor cn \rfloor$. The probability that $G(\beta_n, \beta_n, p)$ does not have a complete matching is upper bounded by $\gamma(\beta_n)$ where γ is the upper bounding function defined in Theorem 8. Furthermore, $\sum_{n \geq 1} n\gamma(\beta_n) < \infty$. □

Proof: The upper bound on the probability that a matching does not exist is immediate. We now show that $\sum_n n\gamma(\beta_n)$ converges. Note that any particular integer repeats at most $1/c+1$ times in the sequence $\{\beta_n, n \geq 1\}$. As a consequence

$$\begin{aligned}\sum_{n \geq 1} n\gamma(\beta_n) &\leq \frac{1}{c}\sum_{n \geq 1}(cn)\cdot\gamma(\beta_n) \\ &\leq \frac{1}{c}\sum_{n \geq 1}(\beta_n + 1)\cdot\gamma(\beta_n) \\ &\leq \frac{1}{c}\left(\frac{1}{c}+1\right)\sum_{k \geq 1}(k+1)\cdot\gamma(k) \;<\; \infty.\end{aligned}$$

∎

VI. A Digression of Not Just Interpretive Value: Maximum Single Commodity Flow

Let us now take a step back to see how matching arises naturally in the simpler case of a single commodity flow between a source node s and a sink node t. We shall assume that additional nodes $1, 2, \ldots, n$ are merely relays. The random graph of interest is now $G(n+2, p)$, where the number $n+2$ comes from n relay nodes and the two source and sink nodes. Our interest is in the maximum rate of information flow between source and sink $\pi_n(2)$. (To be strictly conforming to our earlier notation, we must use $\pi_{n+2}(2)$ for there are $n+2$ nodes in the network and with the first two nodes being in session. The asymptotics does not change of course).

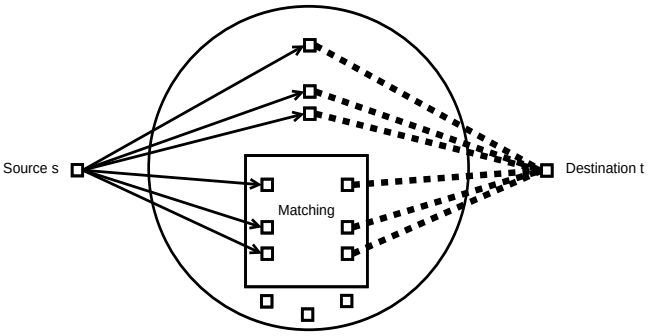

Figure 2. Single source single sink setting indicating how matching arises.

Grimmett and Suen [18] showed that $\pi_n(2)$ grows linearly in n and that $\lim_n \frac{\pi_n(2)}{n} = p$, almost surely. It is then clear that the cut that isolates the source is a tight cut. So is the cut that isolates the sink. Motivated by this, Karp et al. [12] provided an algorithm that achieves the minimum cut capacity. We will show that, for a fixed $\varepsilon > 0$, the following algorithm transports $np(1-\varepsilon)$ bits from the source to the sink with vanishing probability of failure. See Figure 2.

Algorithm MaxFlow:

- The source floods exactly $np(1-\varepsilon)$ links with one bit per link.
- The sink pulls all these bits from $np(1-\varepsilon)$ links connected to it in the following two steps.
  (a) If any node connected to the sink is directly connected to the source, the sink draws the corresponding bit. With overwhelming probability, there are at least $np(1-\varepsilon)\cdot p(1-\varepsilon)$ such connections.
  (b) Here is how the sink draws the remaining bits. There are at most $\beta_n = np(1-\varepsilon)(1-p(1-\varepsilon))$ such yet-to-be-pulled bits, and these reside with let us say *source side relays* not in direct contact with the sink. Among those relays that did not get a bit directly from the source (and these are $n - np(1-\varepsilon) = n(1-p(1-\varepsilon))$ in number) the sink is connected to at least $n(1-p(1-\varepsilon))\cdot p(1-\varepsilon) = \beta_n$, again with overwhelming probability. Let us call these the *sink side relays*. There is a matching, again with overwhelming probability, between the source side relays and the sink side relays. This matching is then used in the obvious way to draw the yet-to-be-pulled bits. □

Obviously, the direct link between s and t is inconsequential for the asymptotics. It is further obvious from the analysis of the previous section that the probability of failure is overwhelmingly small, and moreover, it is summable over n (Lemma 9). This is essentially the argument of Karp et al. [12] to show the achievability direction of the result of Grimmett and Suen [18].

What if we have not one sink t, but two sinks t_1 and t_2? There is one matching needed for t_1 and another needed for t_2. These matchings depend on the connections at the respective sinks, but can be found with overwhelmingly small probability of failure via the union bound for probabilities. Once these are found, while the relays may be overworked, the links are utilized within their capacity limits. Indeed, if a common sink-side relay is required to deliver the same bit (from a particular source side relay) to both sinks, then the relay simply copies the obtained bit on both links to the sinks. If the relay is required to supply two different bits to the two sinks, the matchings are to different bits, the relay fetches the two bits from the respective source side relays on two different links (as per matching), and supplies them to the two sinks via two different links. This matching on an as-needed basis minimizes link usage. But every time a new sink is added, new flows should be initiated to make all bits available to the new sink. Can we *prepare* the network to be in a state of readiness so that upon addition of a new sink, it is merely the new sink that does the necessary work to obtain all bits?

Our next goal is to modify Algorithm MaxFlow into one that pushes two steps and then pulls, as in Algorithm ALLCAST, yielding a decentralized algorithm that easily extends to the case of multiple sinks.

Consider the single source single sink case again, and the following algorithm.

Algorithm MaxFlowPUSHPULL:

- *Push step 1*: The source node s floods $np(1-\varepsilon)$ links with one bit per link. We shall call the bits $b_1, b_2, \ldots, b_{np(1-\varepsilon)}$ and the recipient nodes of these bits as the owners $O_1, O_2, \ldots, O_{np(1-\varepsilon)}$ of the respective bits. All other nodes are termed relays and indexed $R_{np(1-\varepsilon)+1}, \ldots, R_n$.
- *Push step 2*: Each owner O_i pushes his bit b_i one more level, but only to neighbors who are not owners, and to the sink t if there is a link to the sink. Owner-owner links are unutilized.
- *Pull step 1*: The sink t collects all bits sent directly by owners.
- *Pull step 2*: The sink t identifies the list of additional bits needed, the list of relays it is connected to, the list of bits they have in their possession, and does an appropriate matching of relays with the required bits. It then pulls the desired bits from these relays via the by now all-too-familiar matching. □

The bit-map for this setting is much simpler (see Table II). The columns are indexed by the bits. The rows are indexed by the nodes, with the first $np(1-\varepsilon)$ representing the owners and the rest representing the relays. Row i, when it corresponds to owner O_i (which is when $1 \le i \le np(1-\varepsilon)$) has a 1 only on the ith column. But when row i corresponds to a relay (which is when $i > np(1-\varepsilon)$), it has entry $X_{ij} = 1$ if O_j is connected to R_i. Clearly, the presence or absence of this link is independent of the status of all other links, and $X_{i,j}$ is a Bernoulli(p) random variable, when $i > np(1-\varepsilon) \ge j$.

We then have the following result.

Table II
BIT-MAP FOR ONE SOURCE ONE SINK FLOW

	b_1	b_2	$\cdots$	$b_{np(1-\varepsilon)}$
O_1	1	0	$\cdots$	0
O_2	0	1	$\cdots$	0
$\vdots$			$\ddots$	
$O_{np(1-\varepsilon)}$	0	0	$\cdots$	1
$R_{np(1-\varepsilon)+1}$ $\vdots$ R_n	$((X_{i,j}))$			

Theorem 10: For any $\varepsilon > 0$, the following event occurs almost surely: for all but finitely many n, the algorithm MaxFlowPUSHPULL succeeds in transporting all $np(1-\varepsilon)$ bits from the source s to the sink t. □

Proof: This is almost immediate. If the algorithm fails, one of the following must happen.

(1) The event $A_1^{(n)}$ occurs, which is the event that node s is connected to less than $np(1-\varepsilon)$ relays. By Lemma 5, there is a $c_1 > 0$ such that for all sufficiently large n, we have $\Pr\{A_1^{(n)}\} \le e^{-nc_1}$.

(2) The event $A_2^{(n)}$ occurs, which is the event that the sink t is connected to a number of owners outside the range $np(1-\varepsilon) \cdot p(1 \pm \varepsilon)$. Again by Lemma 5, there is a $c_2 > 0$ such that for all sufficiently large n, we have $\Pr\{A_2^{(n)}\} \le e^{-nc_2}$ for some $c_2 > 0$.

(3) The event $A_3^{(n)}$ occurs, which is the event that the sink t is connected to fewer than $\beta_n := n(1-p(1-\varepsilon)) \cdot p(1-\varepsilon)$ relays. Again by Lemma 5, there is a $c_3 > 0$ such that for all sufficiently large n, we have $\Pr\{A_3^{(n)}\} \le e^{-nc_3}$.

(4) If $A_1^{(n)} \cup A_2^{(n)} \cup A_3^{(n)}$ does not occur, the number of bits that remain to be pulled is at least $np(1-\varepsilon) - np(1-\varepsilon) \cdot p(1+\varepsilon)$ which is at most β_n. The number relays that can help the sink pull these bits is at least β_n. For the algorithm to fail, the event $M^{(n)}$, that there is no coverage of the yet-to-be-pulled bits by the available relays with each relay accounting for at most one bit (capacity constraint), must then occur. This implies that if a particular set of β_n relays are chosen, there is no coverage of the required bits. This further implies that any superset of β_n bits that includes the yet-to-be-pulled bits cannot be covered by the β_n chosen and available relays.

The matrix rows corresponding to the β_n chosen relays (rows) and the β_n chosen bits (columns) is a $\beta_n \times \beta_n$ square submatrix whose entries are conditionally iid Bernoulli(p) random variables. Again, we may view this as a bipartite graph with the chosen relays on the one side and chosen bit indices on the other side. Thus, if $A_1^{(n)} \cup A_2^{(n)} \cup A_3^{(n)}$ does not occur, but $M^{(n)}$ does, then there is no matching on the random bipartite graph. Using Theorem 8, the probability that such a matching does not exist, conditioned on $(A_1^{(n)} \cup A_2^{(n)} \cup A_3^{(n)})^c$, is upper bounded by $\gamma(\beta_n)$.

Thus, the event that the sink is unable to pull all the bits implies the event

$$A_1^{(n)} \cup A_2^{(n)} \cup A_3^{(n)} \cup M^{(n)},$$

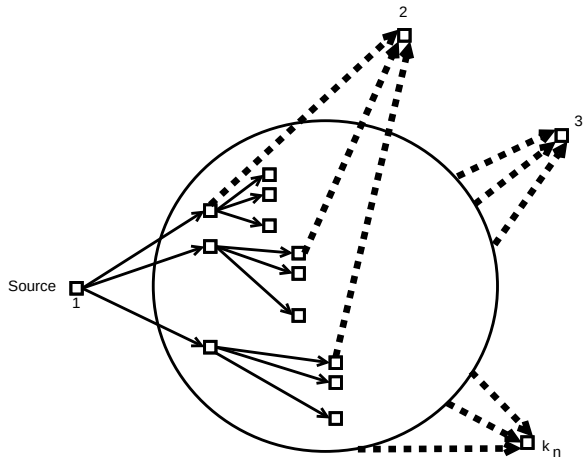

Figure 3. The relay(k_n, n) network. Source pushes bits to owners who then push to relays (solid lines). The sinks pull the bits from either owners or relays (dashed lines).

and its probability is upper bounded by

$$e^{-nc_1} + e^{-nc_2} + e^{-nc_3} + \gamma(\beta_n). \tag{22}$$

This is summable by Lemma 9, and the rest follows. ■

Instead of one sink, suppose we have two sinks t_1 and t_2 that are not connected directly to each other or directly to the source. The source has to transport all its $np(1-\varepsilon)$ bits to each of the two sinks using only the n relay nodes. We may continue to use MaxFlowPUSHPULL with the following extension. The two push steps are common. But each sink simply executes its own pull steps based on the connections it sees at its end and the information from its helper nodes. Using the union bound, it immediately follows that Theorem 10 holds for one source and two sinks when there are no direct connections between the set of nodes constituted by the source and the sinks.

Indeed, we can say something much stronger. One version that suffices to address the multicast setting of the next section is the following. Consider a scenario where there is one source s and a total of $k_n - 1$ sinks $t_1, t_2, \ldots, t_{k_n-1}$ where $\sup_{n\geq 1} \frac{k_n}{n} \leq C$ for some $C < \infty$. The source and the sinks have no links among themselves, but are connected through a network of n relays. See Figure 3. The internal links between the relays and the links between the source/sinks and the relays are iid Bernoulli(p) random variables. The source wishes to transfer all its bits of information to each of the sinks. Let us denote this random network as relay(k_n, n).

Theorem 11: For any $\varepsilon > 0$, the following event occurs almost surely: for all but finitely many n, the algorithm MaxFlowPUSHPULL, with the pull stages implemented by each sink, succeeds in transporting all $np(1-\varepsilon)$ bits from the source s to each of the $k_n - 1$ sinks on the relay(k_n, n) network. □

Proof: Observe that the first three terms in the upper bound for the probability of failure in (22) decay exponentially fast in n. The last term $\gamma(\beta_n)$ satisfies $\sum_{n\geq 1} n\gamma(\beta_n) < \infty$. Since there are $k_n - 1 = O(n)$ sinks, by the union bound, the probability that the algorithm fails for some sinks is at most $Cn\left(e^{-nc_1} + e^{-nc_2} + e^{-nc_3} + \gamma(\beta_n)\right)$. This upper bound is summable, and the rest follows. ■

VII. Multicast: Achievability

We now return to the setting of n nodes of which k_n are in a multicast session. Node 1 is the source node and nodes $2, 3, \ldots, k_n$ are the sinks. Our goal in this section is to show that the upper bound (10) is achievable. While one could in principle proceed as in Catlin et al. [14] to prove achievability, we shall directly jump to a constructive proof.

Theorem 12: For the multicast problem with k_n nodes in the session, let $\lim_{n\to\infty} k_n/n = \alpha \in [0, 1]$. We then have

$$\lim_{n\to\infty} \frac{\pi_n(k_n)}{n} = \left(1 - \frac{\alpha}{2}\right) \mathbb{E}[C] \quad \text{a.s.}$$

□

Proof: As in the proof of Theorem 3, converse was already shown in (10). So showing achievability suffices, and further showing it on Erdős-Rényi random graphs with parameter p suffices. Moreover, as before, it is enough to show that: *For any $\varepsilon > 0$, the following event occurs almost surely: for all but finitely many n, there is an algorithm that succeeds in transporting $\pi_n(k_n) \geq n\,(1-\alpha/2)\,p(1-2\varepsilon)$ bits from the source to each of the $k_n - 1$ sinks.* We claim that this holds.

We first dispose two easy cases.

When $\alpha = 0$, this follows from Theorem 11, by simply ignoring the links between the session nodes and by using MaxFlowPUSHPULL and $n(1-\varepsilon)$ relays, and with pulls implemented at each of the sink nodes.

When $\alpha = 1$, pretend that all nodes are in session and implement ALLCAST. The result follows from Theorem 4.

Only the case when $0 < \alpha < 1$ remains, for which we will use a combination of the above.

Observe that the subset of session nodes alone form a complete graph with k_n vertices for which Theorem 4 is applicable. Using ALLCAST and without using any of the relay nodes, we have that the source can distribute

$$\pi_n^{(1)} \geq \frac{k_n}{2} p(1-\varepsilon) \tag{23}$$

bits to the other $k_n - 1$ nodes in the session, for all but finitely many n, almost surely. (Summability of the probability upper bound sequence holds since $k_n = \Omega(n)$).

Removing these direct links between the session nodes, we end up with the graph in Figure 3, where the session nodes are now only connected to the $m_n = n - k_n$ relay nodes. The link to each relay node from each session node has Bernoulli(p) capacity. Further the relay nodes have interrelay link capacities that are independent Bernoulli(p) random variables. By Theorem 11, using MaxFlowPUSHPULL, the source can distribute

$$\pi_n^{(2)} \geq m_n p(1-\varepsilon) \tag{24}$$

bits to the $k_n - 1$ sinks (solely with the help of the relay nodes), for all but finitely many n, almost surely. (Summability of the probability upper bound sequence holds since $m_n = \Omega(n)$).

The result immediately follows from (23) and (24) since $\pi(k_n) \geq \pi_n^{(1)} + \pi_n^{(2)}$ and $k_n/2 + m_n = n - k_n/2 \geq n(1 - \alpha/2)(1-\varepsilon)$ for all sufficiently large n. ■

VIII. Vanishing Link Probabilities

Our results extend to the case when p is a function of n, denoted p_n, and vanishes but sufficiently slowly. We shall focus only on the allcast problem. The results for multicast can be obtained in an analogous fashion.

Theorem 13: Let $p_n = \sqrt{\frac{\tau_n \log n}{n}}$ where $\tau_n \to \infty$ but $p_n \to 0$. For any $\varepsilon > 0$, the following event occurs almost surely: for all but finitely many n, the algorithm ALLCAST succeeds in distributing $\frac{1}{2}np_n(1-\varepsilon)$ bits to each of the n nodes. Furthermore, $\lim_{n\to\infty} \frac{\pi_n}{np_n} = \frac{1}{2}$ almost surely. □

Proof: The proof of the first part is similar to the proof of Theorem 4, with some additional effort to get better probability upper bound estimates. Again, we argue that the probability that algorithm ALLCAST fails is summable over n. If the algorithm fails for a particular n, at least one of the following events must have occurred.

1) The event $A_1^{(n)}$ occurs, which is defined to be the event that there are fewer than $\frac{1}{2}np_n(1-\varepsilon)$ vertices connected to node 0. By Lemma 5, applied with $q = p_n/2$, there is some $c_1 > 0$ such that for all sufficiently large n, we have

$$\Pr\{A_1^{(n)}\} \le e^{-n\cdot\frac{1}{2}p_n\cdot\varepsilon^2/3} = e^{-c_1\sqrt{n\tau_n\log n}}.$$

2) For some node t, the event $A_2^{(n)}(t)$ occurs, which is defined to be the event that node t is connected to a certain number of owners outside the range $\frac{1}{2}np_n(1-\varepsilon)\cdot\frac{1}{2}p_n(1\pm\varepsilon)$ with links pointing towards t. (The case when node t is an owner leads to one fewer number of owners which as before is absorbed into $(1\pm\varepsilon)$ factor). Again by Lemma 5, there is some $c_2 > 0$ such that for all sufficiently large n, we have

$$\begin{aligned}\Pr\{A_2^{(n)}(t)\} &\le e^{-\frac{1}{2}np_n(1-\varepsilon)\cdot\frac{1}{2}p_n\cdot\varepsilon^2/3}\\ &\le e^{-c_2np_n^2}\\ &= e^{-c_2\tau_n\log n} = \frac{1}{n^{c_2\tau_n}}. \end{aligned} \quad (25)$$

Note that c_2 can be arbitrarily small because of the ε^2 factor.

3) Let $A_1^{(n)}$ not occur. Then there are exactly $\frac{1}{2}np_n(1-\varepsilon)$ owners. For some node t, the event $A_3^{(n)}(t)$ occurs, which is the event that the node t is connected to fewer than

$$\begin{aligned}\beta_n &:= \left(n - \frac{1}{2}np_n(1-\varepsilon)\right)\cdot\frac{1}{2}p_n(1-\varepsilon)\\ &= \frac{1}{2}np_n(1-\varepsilon)\cdot\left(1-\frac{1}{2}p_n(1-\varepsilon)\right)\end{aligned} \quad (26)$$

relays with links pointing towards t. (As before, the case of 1 less relay when node t is a relay is easily handled). Once again by Lemma 5, there is a $c_3 > 0$ such that for all sufficiently large n, we have

$$\begin{aligned}\Pr\{A_3^{(n)}(t) \mid (A_1^{(n)})^c\} &\le e^{-\left(n-\frac{1}{2}np_n(1-\varepsilon)\right)\cdot\frac{1}{2}p_n\cdot\varepsilon^2/3}\\ &\le e^{-c_3\sqrt{n\tau_n\log n}}.\end{aligned}$$

4) For some node t, if $A_1^{(n)} \cup A_2^{(n)}(t) \cup A_3^{(n)}(t)$ does not occur, then the event $M^{(n)}(t)$ occurs, which is the event that node t is unable to pull the desired bits. We claim that

$$\Pr\left\{M^{(n)}(t) \mid \left(A_1^{(n)} \cup A_2^{(n)}(t) \cup A_3^{(n)}(t)\right)^c\right\} \le \delta_n \quad (27)$$

where

$$\sum_{n=1}^{\infty} n\delta_n < \infty. \quad (28)$$

The event that the algorithm fails is thus a subset of

$$A_1^{(n)} \bigcup_{t=1}^{n}\left(A_2^{(n)}(t) \cup A_3^{(n)}(t) \cup M^{(n)}(t)\right)$$

whose probability is upper bounded via the union bound and (27) by

$$n\cdot\left(e^{-c_1\sqrt{n\tau_n\log n}} + \frac{1}{n^{c_2\tau_n}} + e^{-c_3\sqrt{n\tau_n\log n}} + \delta_n\right).$$

By (28) and the assumption that $\tau_n \to \infty$, we see that this bound is summable.

What remains is to prove (27) and (28).

As before, the probability on the left-hand side of (27) is upper bounded by the probability that there is no matching in a bipartite graph with β_n vertices and link probability p_n.

We first sharpen Lemma 7. The bound in (19), after noting that we now have β_n vertices on one side, can be sharpened (see [17, p.174]) to

$$\begin{aligned}\Pr\{F_a\} \le 2\binom{\beta_n}{a}\binom{\beta_n}{a-1}(1-p_n)^{a(\beta_n-a+1)}\\ \cdot\left(\binom{a(a-1)}{2a-2}\cdot p_n^{2a-2}\right)\end{aligned}$$

where the extra term within parentheses in the second line can be included because it is an upper bound (via the union bound) on the probability that some $2a-2$ links, among the possible $a(a-1)$ links from A to $\Gamma(A)$, are active. Recall that a is an integer satisfying $2 \le a \le (\beta_n+1)/2$. Using the bounds $\binom{m}{a} \le \left(\frac{em}{a}\right)^a$ and $(1-x) \le e^{-x}$, we get

$$\begin{aligned}&\Pr\{F_a\}\\ &\le 2\left(\frac{e\beta_n}{a}\right)^a\left(\frac{e\beta_n}{a-1}\right)^{a-1}\left(\frac{ea}{2}\right)^{2a-2}p_n^{2a-2}\\ &\qquad\cdot\left(e^{-\beta_np_na\left(1-\frac{a}{\beta_n}+\frac{1}{\beta_n}\right)}\right)\\ &\le 2\left(\frac{e\beta_n}{a}\right)\left(\frac{e^2\beta_np_n}{2}\right)^{2a-2}\left(1+\frac{1}{a-1}\right)^{a-1}\\ &\qquad\cdot\left(e^{-\beta_np_na\left(1-\frac{a}{\beta_n}+\frac{1}{\beta_n}\right)}\right)\\ &\le Cnp_n\left(\frac{e^2np_n^2}{4}\right)^{2a-2}e^{-anp_n^2(1-2\varepsilon)/4}\end{aligned}$$

for some finite constant C, where in the last inequality we have used $(1+1/k)^k \le e$, the bound $1-(a-1)/\beta_n \ge 1/2$ when $2 \le a \le (\beta_n+1)/2$, and the obvious upper and lower bounds on β_n from (26). Now, using $np_n^2 = \tau_n\log n$, we get

$$\begin{aligned}\Pr\{F_a\} &\le C\sqrt{n\tau_n\log n}\left(\frac{e^2\tau_n\log n}{4}\right)^{2a-2}n^{-a\tau_n(1-2\varepsilon)/4}\\ &\le C\left(16\frac{\sqrt{n}}{e^4(\tau_n\log n)^{1.5}}\right)\left(\frac{e^4(\tau_n\log n)^2}{16n^{\tau_n(1-2\varepsilon)/4}}\right)^a.\end{aligned}$$

Since the term inside the second parentheses converges to zero as $n \to \infty$, it follows that for all sufficiently large n and some

finite constants C_1 and C_2, we have

$$\begin{aligned}\sum_{a=2}^{(\beta_n+1)/2} \Pr\{F_a\} &\le C_1 \left(\frac{\sqrt{n}}{(\tau_n \log n)^{1.5}}\right)\left(\frac{e^4(\tau_n \log n)^2}{16 n^{\tau_n(1-2\varepsilon)/4}}\right)^2 \\ &= C_2 \left(\frac{\sqrt{n}(\tau_n \log n)^{2.5}}{n^{\tau_n(1-2\varepsilon)/2}}\right) \\ &=: \kappa_n.\end{aligned}$$

The probability that there is no matching is then upper bounded by $\delta_n := \kappa_n + 2\beta_n(1-p_n)^{\beta_n}$. The second term is upper bounded, using the bounds on β_n, as

$$2\beta_n(1-p_n)^{\beta_n} \le np_n e^{-np_n^2(1-2\varepsilon)/2} = \frac{\sqrt{n\tau_n \log n}}{n^{\tau_n(1-2\varepsilon)/2}}.$$

From these two bounds, using $\tau_n \to \infty$, it is clear that not only $\delta_n \to 0$, but in addition, $\sum_{n\ge 1} n\delta_n < \infty$. This establishes (27) and (28) and proves validity of algorithm ALLCAST.

The above achievability result also establishes that

$$\liminf_{n\to\infty} \frac{\pi_n}{np_n} \ge \frac{1}{2}.$$

The upper bound

$$\limsup_{n\to\infty} \frac{\eta_n}{np_n} \le \frac{1}{2}$$

follows from (7) and Lemma 5. This concludes the proof of the second statement. ∎

The extension to multicasting can be done similarly.

IX. Discussion

We began the problem of allcast and multicast capacity region for multiple allcast and multiple multicast. Yet, we largely focused on single allcast or single multicast with just one sender and with remaining nodes of the session as receivers. But study of single multicast suffices, thanks to the result [1, Cor. 4.a] of Li et al. on transferability of rates across sources (even with network coding). It is therefore clear how the established results imply the validity of (1) and (2). The requirement that the session nodes be identical for each of the multiple multicasts is crucial for this transferability.

Moreover, we largely studied multicasting techniques that do not use network coding. One message coming out of this work is that though network coding provides a coding advantage in specific undirected scenarios, and one such example can be found in Li et al. [13], in large dense random undirected networks of the variety studied in our paper the coding advantage is at most $1 + o(1)$ in the number of nodes. While our results applied to graphs $G(n, p_n)$ with $p_n \to 0$, we did require that p_n vanishes sufficiently slowly. In particular, $p_n = \sqrt{(\tau_n \log n)/n}$ so that a typical node has degree $np_n = \sqrt{n\tau_n \log n}$. These are well connected, but by no means sparse graphs. This naturally raises two questions. One, can one extend these results to some useful classes of sparse random graphs? Two, can one show rates of convergence for the expected rates to the asymptotic limits and concentration around the means?

The message of asymptotically negligible network coding advantage in single or multiple multicast settings (with identical session nodes) may evoke the question of a possible connection with a conjecture of Li and Li [19] for multiple unicasts. Li and Li [19] conjectured that for multiple unicast, network coding provides no coding advantage in undirected graphs. While their conjecture holds true for some specific classes of undirected graphs ([20], [21]), the general conjecture remains unresolved. The negligible gain for multicasting in random graphs studied here arises from the dense interconnectivity between relays. The bottlenecks are primarily at the periphery. So there does not seem to be much insight that one can glean from our study to prove or disprove the Li and Li conjecture for multiple unicasts in undirected networks.

While we studied multiple multicasts, our communication application naturally restricted us to a single set of session nodes. We thus had to study Steiner tree packings for a single subset of nodes. VLSI applications require efficient packing of Steiner trees across a multiplicity of such subsets (or nets; see [8]). One could apply our random network framework to such problems and attempt to devise similar quick-but-dirty algorithms. This is an interesting topic that is beyond the scope of this paper.

Acknowledgements

The authors would like to thank Prof. Navin Kashyap for bringing references [6] and [14] to their attention.

References

[1] Z. Li, B. Li, and L. C. Lau, "A constant bound on throughput improvement of multicast network coding in undirected networks," *IEEE Trans. Inf. Theory*, vol. 55, no. 3, pp. 1016–1026, Mar. 2009.

[2] S. Maheshwar, Z. Li, and B. Li, "Bounding the coding advantage of combination network coding in undirected networks," *IEEE Trans. Inf. Theory*, vol. 58, no. 2, pp. 570–584, Feb. 2012.

[3] Z. Li, B. Li, D. Jiang, and L. C. Lau, "On achieving optimal throughput with network coding," in *Proc. IEEE INFOCOM*, 2005.

[4] W. T. Tutte, "On the problem of decomposing a graph into n connected factors," *J. London Math. Soc.*, vol. 36, pp. 221–230, 1961.

[5] C. S. J. A. Nash-Williams, "Edge-disjoint spanning trees of finite graphs," *J. London Math. Soc.*, vol. 36, pp. 445–450, 1961.

[6] F. Barahona, "Packing spanning trees," *Mathematics of Operations Research*, vol. 20, no. 1, pp. 104–115, Feb. 21995.

[7] K. Jain, M. Mahdian, and M. R. Salavatipour, "Packing Steiner trees," in *Proc. 10th Annu. ACM-SIAM Symp. Discr. Algor. (SODA)*, 2003.

[8] M. Grötschel, A. Martin, and R. Weismantel, "The Steiner tree packing problem in VLSI design," *Mathematical Programming*, vol. 78, pp. 265–281, 1997.

[9] S. Chen, O. Günlük, and B. Yener, "The multicast packing problem," *IEEE/ACM Trans. Netw.*, vol. 8, no. 3, pp. 311–318, Jun. 2000.

[10] L. C. Lau, "An approximate max-Steiner-tree-packing min-Steiner-cut theorem," in *Proc. 45th IEEE Symp. Found. Comput. Sci. (FOCS)*, 2004.

[11] M. Saad, T. Terlaky, A. Vannelli, and H. Zhang, "Packing trees in communication networks," *J. Comb. Optim.*, vol. 16, pp. 402–423, 2008.

[12] R. M. Karp, R. Motwani, and N. Nisan, "Probabilistic analysis of network flow algorithms," *Mathematics of Operations Research*, vol. 18, no. 1, pp. 71–97, 1993.

[13] Z. Li, B. Li, and L. C. Lau, "On achieving optimal multicast throughput in undirected networks," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2467–2485, Jun. 2006.

[14] P. A. Catlin, Z. Hong Chen, and E. M. Palmer, "On the edge arboricity of a random graph," *ARS Combinatorica*, vol. 35-A, pp. 129–134, 1995.

[15] D. J. Aldous, C. McDiarmid, and A. Scott, "Uniform multicommodity flow through the complete graph with random edge-capacities," *Operations Research Letters*, vol. 37, no. 5, pp. 299–302, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/B6V8M-4WH2KW3-2/2/2ff93740f0388a324da9b8a6c589bbfe

[16] M. Khandwawala and R. Sundaresan, "Optimal multicommodity flow through the complete graph with random edge-capacities," *Journal of Applied Probability*, vol. 47, no. 1, pp. 201–215, Mar. 2010.

[17] B. Bollobás, *Random Graphs*, 2nd ed. Cambridge, UK: Cambidge University Press, 2001, vol. Cambridge studies in advanced mathematics, no. 73.
[18] G. R. Grimmett and W.-C. S. Suen, "The maximal flow through a directed graph with random capacities," *Stochastics*, vol. 8, pp. 153–159, 1982.
[19] Z. Li and B. Li, "Network coding: The case of multiple unicast sessions," in *Proc. of the 42nd Annual Allerton Conference on Communication, Control, and Computing*, 2004.
[20] N. J. A. Harvey, R. Kleinberg, and A. R. Lehman, "On the capacity of information networks," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2345–2364, 2006.
[21] K. Jain, V. Vazirani, and G. Yuval, "On the capacity of multiple unicast sessions in undirected graphs," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2805–2809, Jun. 2006.