# On Achieving Optimal End-to-End Throughput in Data Networks: Theoretical and Empirical Studies

Zongpeng Li, Baochun Li, Dan Jiang, Lap Chi Lau

***Abstract*— With the constraints of network topologies and link capacities, achieving the optimal end-to-end throughput in data networks has been known as a fundamental but computationally hard problem. In this paper, we seek efficient solutions to the problem of achieving optimal throughput in data networks, with single or multiple unicast, multicast and broadcast sessions. Towards this objective, we first investigate upper and lower bounds of such optimal throughput in the case of a single multicast session, and show that it is computationally hard to compute these bounds. We then show the surprising result that, facilitated by the recent advances of network coding, computing the strategies to achieve the optimal end-to-end throughput can be performed in polynomial time, using the algorithm we propose. In addition, we extend our results to the cases of multiple sessions of unicast, multicast, broadcast and group communication, as well as the model of overlay networks. In all these cases, we also show that the optimal achievable throughput is independent from the selection of data sources within each session. Finally, supported by empirical studies, we present the surprising observation that in most topologies, applying network coding may not improve the achievable optimal throughput; rather, it facilitates the design of significantly more efficient algorithms to achieve such optimality.**

## I. Introduction

In the most general case, a data network consists of a set of end hosts and switches interconnected via undirected (or duplex) communication links. In data networks with known topologies and bandwidth capacity bounds for each undirected link, a fundamental problem is to compute and achieve the maximum end-to-end throughput for one or multiple active communication sessions. Depending on the objectives of applications, a communication session may be in the form of unicast (one to one), multicast (one to many), broadcast (one to all), or group communication (many to many). The solutions to this problem may lead to fundamental and new insights with respect to optimal routing and traffic engineering. For example, the recent paradigm of selfish routing [1] allows end hosts to choose routes themselves using source routing strategies. Finding the optimal strategy to disseminate data to multiple destinations with maximum throughput is of natural interests in such a paradigm, especially when we wish to optimally exploit existing network capacities to disseminate large volumes of data.

Nevertheless, computing the strategies to achieve optimal throughput in data networks has been, unfortunately, very hard with respect to its computational complexity. As an example to support this claim, consider the case of optimizing one multicast session from a source to a set of destinations. It can be shown that, traditionally, such a problem is equivalent to the problem of *steiner tree packing*, which seeks to find the maximum number of pairwise edge-disjoint steiner trees, in each of which the multicast group remains connected. An intuitive explanation to such equivalence is that, each unit throughput corresponds to a unit information flow being transmitted along a tree that connects every node in the group. The maximum number of trees we can find corresponds to the optimal throughput for the session. The steiner tree packing problem has been shown to be NP-complete and APX-hard [2], [3], [4]. Brute-force solutions to the steiner tree packing problem are obviously not realistically feasible, not to mention more general cases where multiple sessions co-exist in the network.

In this paper, we seek to bring fundamentally new insights and efficient solutions to the problem of optimizing end-to-end throughput in data networks. We first illustrate the power of *network coding* [5], [6] with respect to achieving optimal throughput. In the paradigm of network coding, information flows in data networks may not only be stored and forwarded, but also be encoded and decoded in any nodes in the network. We prove that, on one end, the achievable optimal throughput with network coding is lower bounded by that achieved without network coding, which corresponds to the problem of steiner tree packing. On the other end, it is also upper bounded by the *steiner strength* of the network. We show that, unfortunately, the computation of both bounds are NP-complete problems, which seems to suggest that the problem of achieving optimal throughput with network coding is also computationally hard to solve.

Surprisingly, as one of our most important contributions, we are able to prove the following: Given a single multicast session and undirected data networks with per-link capacity bounds, a complete solution to the problem of achieving optimal throughput can be computed in *polynomial time*, using a new algorithm we propose. We then show that this conclusion can be readily extended to multiple concurrent sessions, as well as to other types of communication, including unicast, broadcast and group communication. Even when the general form of data networks is modified to reflect realistic characteristics of overlay networks (where only end hosts at the edge may be able to replicate, encode and decode data), the same conclusion still holds. The solutions to the problems include not only optimal routing strategies to transmit data in the network, but also how data may be encoded and decoded as they are relayed towards the destinations. Though there exist previous results on network coded throughput in *directed* networks, to the best of our knowledge, this paper is the first work that systematically studies the effects of network coding with respect to optimizing throughput in *undirected* data

Zongpeng Li, Baochun Li and Dan Jiang are with the Department of Electrical and Computer Engineering, University of Toronto. Their email addresses are {*arcane,bli,djiang*}*@eecg.toronto.edu*}. Lap Chi Lau is with the Department of Computer Science, University of Toronto. His email address is {*chi@cs.toronto.edu*}.

networks.

Beyond these efficient solutions that compute achievable optimal throughput, additional empirical observations and theoretical contributions that we present in this paper are as follows. First, from a graph theoretic perspective, we prove that the advantage of network coding — the ratio of optimal throughput with network coding over that without coding — is bounded by a constant factor of *two*. This is in contrast to previous results [7] in directed networks, which show that such a ratio may be arbitrarily high. Second, we prove that, the achievable optimal throughput is not affected by the selection of sources in each communication session. Third, even more surprisingly, supported by empirical studies using over $1000$ topologies, we show that the achieved optimal throughput with coding is mostly identical to that achieved without coding. This implies that the fundamental benefit of network coding is *not* higher optimal throughput, but to facilitate *significantly more efficient computation* of strategies to achieve such optimal throughput. Finally, our algorithms to efficiently compute optimal throughput are also instrumental towards finding efficient approximation algorithms for graph theoretic problems such as *steiner tree packing* and *minimum steiner tree*.

The remainder of this paper is organized as follows. We first discuss related work in Sec. II. In Sec. III, we present our main theorems and algorithm with respect to achieving optimal end-to-end throughput in data networks with a single multicast session. In Sec. IV, we extend our results to the cases of multiple sessions of unicast, multicast, broadcast, and group communication. We also consider the model of overlay networks, where only a subset of nodes are capable of replication and coding. Based on the algorithms we propose, we present both empirical and theoretical insights with respect to achievable optimal throughput in Sec. V. Finally, we conclude the paper in Sec. VI.

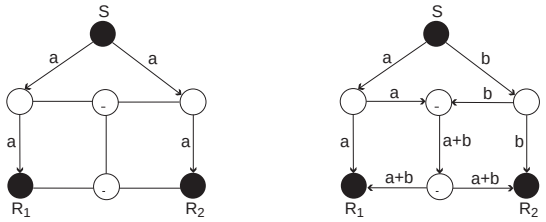
## II. Related Work

The open problem of achieving optimal end-to-end throughput with efficient algorithms has not been discussed in depth in existing literature. There exist, however, similar problems that have been extensively studied. Towards the direction of Quality of Service (QoS) routing, the objective is to find end-to-end paths or multicast trees that satisfy specific bandwidth or delay constraints, and therefore providing the desired QoS guarantees [8]. With respect to end-to-end throughput, finding good topologies that satisfy bandwidth requirements is obviously different from — and arguably easier than — finding *optimal* ones.

There exists an extensive body of research in the area of multicast routing in wide-area IP networks (*e.g.*, [9]). The advantage of IP-based multicast is brought by data packet replication on multicast-capable switches, improving bandwidth efficiency and throughput compared to all (naive) unicast between the source and the multicast receivers. However, since it is based on the construction of a single tree, the end-to-end throughput is not optimal compared to what is achievable by a topology beyond a tree.

As IP multicast is not readily deployed, algorithms promoting application-layer overlay multicast have recently been proposed as remedial solutions, focusing on the issue of constructing and maintaining a multicast tree using only end hosts [10], [11]. Though a single multicast tree may not lead to optimized throughput, recent studies (*e.g.*, SplitStream [12], CoopNet [13], Digital Fountain [14] and Bullet [15]) have proposed to utilize either multiple multicast trees (*forest*) or a topological *mesh* to deliver striped data from the source, using either multiple description coding or source erasure codes to split content to be multicast. These proposals have indeed improved end-to-end throughput beyond that of a single tree, but there have been no discussions on whether the optimal throughput may be achieved, or how close the proposed algorithms approach optimality. In this paper, we study such achievable optimality, while considering the most general case where the data source transmits a stream of bytes, and is not assumed to perform any source or error correction coding.

There have been studies on achieving optimality with respect to computing *oblivious routing* strategies in data networks. The objectives are to maximize throughput for a source-destination pair, and to minimize congestion on the network. Most notably, using linear programming techniques, *polynomial time* algorithms (with a polynomial number of variables and constraints in the LP formulation) can be constructed to compute strategies for *optimal* oblivious routing for any network, directed or undirected [16], [17]. Though we also employ linear optimization tools and study undirected networks, our problem domain is more general: while optimal oblivious routing focuses on origin-destination pairs of *unicast* sessions (possibly exploiting path diversity), we focus on a variety of communication sessions, including unicast, multicast, broadcast and group communication. We seek fundamental insights on how optimal a routing strategy may become, and what is the maximum achievable throughput in a communication session.




(a) Maximum throughput without network coding is 1.875. If one multicast tree is used, the achieved throughput is 1.

(b) Maximum throughput with network coding is 2.

Fig. 1. The advantage of network coding with respect to improving the end-to-end multicast throughput from $S$ to $R_1$ and $R_2$.

The theory of *network flows* [18] studies the transmission of commodities of the same type (unicommodity flows) through a network with known capacities. The maximum flow rate between the source and the destination is known to be equal to the minimum cut between them, which may be computed with efficient algorithms. When commodities to be transmitted are of different types (multicommodity flows), computing the maximum flow rate can be formulated as a linear optimization problem, and then solved using general linear program solvers [18], [19]. In both unicommodity and multicommodity flows, commodities may only be *forwarded* at intermediate nodes, comparable to all unicast in data networks. The concept of *network coding* extends the capabilities of network nodes in a communication session: from basic data forwarding (as in all unicast) and data replication (as in IP or overlay multicast), to *coding in Galois fields*. Fig. 1

illustrates a classic example of how network coding assists to improve end-to-end throughput. As $R_1$ receives both $a$ and $a+b$ (encoded over GF(2)), it is able to decode and retrieve both $a$ and $b$. If the link capacities are 1, it is apparent that the maximum achievable throughput with network coding is 2. Without coding, it can be computed that the optimal throughput is 1.875. If only one multicast tree is used (as in IP multicast), the achieved throughput is 1.

The recent breakthrough theorem in network coding shows that, for a multicast session in directed networks, if a rate $x$ can be achieved from the sender to each of the multicast receivers independently, it can also be achieved for the entire multicast session (refer to independent proofs of Ahlswede *et al.* [5] and Koetter *et al.* [6]). In addition, Li *et al.* [20] show that *linear codes* suffice to achieve such a property. All linear coding operations are defined as linear combinations over Galois fields with fixed element lengths, thus the size of the data does not increase after being encoded. The power of network coding comes from the difference between *information flows* and traditional *commodity flows*: information may not only be replicated, but also be *coded and forwarded without increases in size*, which are impossible with commodities. As a consequence, the theory of network flows may only be used to study unicast communication sessions in data networks. This paper seeks to design efficient solutions for the problem of achieving optimal throughput in general, and network coding is one of the avenues leading to such an objective.

## III. Achieving Optimal Throughput in Undirected Data Networks: the Single Multicast Case

We begin our theoretical study from the case of a single multicast session. We consider the most general form of data networks, represented by a simple graph $G = (V, E)$ with *undirected* edges between network nodes. Each edge represents a communication link, and the edge capacities are specified by a function $C : E \to \mathcal{Q}^+$ (where $\mathcal{Q}^+$ denotes the set of positive rational numbers), representing the available bandwidth capacities of communication links. Throughout this paper, we focus on the *fractional* model of data routing, where the capacity of each link may be shared fractionally in both directions, and information flows may be split and merged at arbitrarily fine scales.

Now consider a single multicast communication session. We use $M = \{m_0, m_1, \ldots, m_k\} \subseteq V$ to specify the set of nodes involved in the multicast communication, with $m_0$ being the data source. In graphical illustrations throughout this paper (*e.g.*, Fig. 1), nodes in $M$ are shown as black, and nodes in $V - M$ are shown as white. Links are labeled with their capacities, and *all unlabeled links have a capacity of* 1.

### A. Optimal throughput without coding: steiner tree packing

To achieve optimal throughput in undirected data networks in general, a natural direction is to first consider the problem of achieving optimal throughput without coding, but with data replication on network nodes (as in IP multicast). This problem is equivalent to the graph theoretic problem of steiner tree packing [2], [21].

A steiner tree is a sub-tree of the network that connects every multicast node. In the steiner tree packing problem, we seek to decompose the network into weighted steiner trees, such that the total of tree weights is maximized, referred to as the *steiner tree packing number*. A link $e$ may appear in a set of different steiner trees, $T$; but it is required that $\sum_{t \in T} c(t) \leq c(e)$ (where $c(t)$ is the weight of tree $t$), *i.e.*, the total weights of trees using a common edge $e$ should not exceed the capacity of $e$. Without network coding, a solution to steiner tree packing leads to a specific strategy to achieve optimal multicast throughput, since we can transmit a data stream of throughput $c(t)$ along each steiner tree $t$, and the resulting throughput is precisely the total weights of all trees.

To illustrate how steiner tree packing corresponds to achievable optimal throughput without coding, we show an example in Fig. 2(a). In the illustration, each letter corresponds to a distinct steiner tree that connects the multicast group, consisting of $m_0, m_1, m_2$ and $m_3$. Nine such steiner trees exist in the shown packing scheme (from $a$ to $i$), where the tree corresponding to $a$ is highlighted. Since each link with unit capacity needs to accommodate 5 steiner trees, the achievable throughput on each tree is, therefore, 0.2. This leads to a multicast throughput of 1.8, which is optimal without coding.

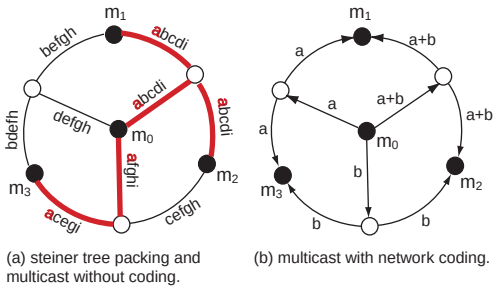
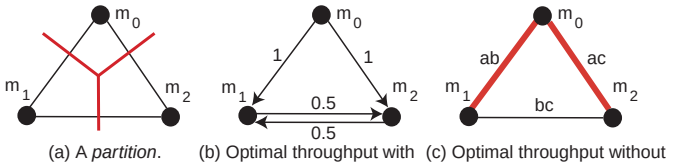



Fig. 2. The achievable optimal throughput is 1.8 without coding, and 2 with coding.

Unfortunately, steiner tree packing has been shown to be NP-complete and APX-hard [2], [4], and the best known polynomial time algorithm has an approximation ratio of around 1.55 [3]. It is not sufficiently accurate to be used in practice. On the brighter side, with the same example, we also show that the achievable optimal throughput with network coding is 2 (Fig. 2(b)), which is higher than that achieved without coding.

### B. Optimal throughput with coding: bounds and complexities

Since network coding may lead to higher optimal throughput, we turn our focus on the problem of achieving optimal throughput with coding, and consider the steiner tree packing number as a natural lower bound. With coding, our first attempt towards a solution is to apply the breakthrough theorem of network coding in *directed* networks, which reveals that if a rate $x$ can be achieved for each receiver in the group independently, it can also be achieved for the entire multicast session. It implies that we may compute the optimal throughput achievable in the session by finding the minimum of the maximum flow rates from the source to each of the receivers, where the maximum flow rates may be efficiently computed by any of the polynomial time maximum flow algorithms.




(a) A *partition*. (b) Optimal throughput with network coding is 1.5. (c) Optimal throughput without network coding is also 1.5.

Fig. 3. A counter-example on the effects of network coding in undirected networks. The independently achievable throughput is 2 for both $m_1$ and $m_2$, but the multicast throughput is only 1.5 with or without network coding.

Unfortunately, this theorem is not valid in undirected networks. Fig. 3 shows a counter-example. In the example, if the source $m_0$ sends to either $m_1$ or $m_2$ independently, the achievable throughput is 2. However, it is impossible to achieve a throughput of 2 from $m_0$ to both $m_1$ and $m_2$ simultaneously in a multicast session. To understand this infeasibility, consider a *partition*, shown in Fig. 3(a), that divides the network into three components. In order to achieve a multicast throughput of $x$, it is necessary to have a capacity of $x$ to concurrently flow into the $m_1$ and the $m_2$ component, respectively. Since the inter-component capacity is 3 (three unit-capacity links being cut), we have $2x \leq 3$. The optimal throughput is therefore upper bounded by 1.5, even with network coding.

In this example, our informal intuition corresponds to the concept of *steiner strength*. Formally, in an undirected network $N$, we consider partitions of the network where there exists at least one source or receiver node in each component of the partition. Let $P$ be the set of all such partitions. The steiner strength of $N$ is defined as $\min_{p\in P} |E_c|/(|p|-1)$, where $|E_c|$ is the total inter-component capacity on the set of links $E_c$ being cut, and $|p|$ is the number of components in the partition $p$. In our example, the steiner strength is $3/2 = 1.5$.

Our discussions so far have effectively led to Theorem 1 — in the case of a single multicast session, the achievable optimal throughput is in between the steiner tree packing number and the steiner strength in undirected data networks.

**Theorem 1.** For an undirected data network with a single multicast session, $N = \{G(V,E), C : E \to \mathcal{Q}^+, M = \{m_0, m_1, \ldots, m_k\} \subseteq V\}$, we have

$$\pi(N) \leq \chi(N) \leq \eta(N)$$

where $\pi(N)$ is the steiner tree packing number, $\chi(N)$ is the achievable optimal throughput, and $\eta(N)$ is the steiner strength.

*Proof:* First, optimal throughput obtained without coding is always feasible when coding is supported — one just ignores network coding and applies the same routing strategy as in the case without coding. Thus, throughput with coding is lower bounded by throughput without coding, and $\pi(N) \leq \chi(N)$. Second, as argued in the previous example, the partition condition is necessary for a certain multicast throughput to be feasible. Therefore multicast throughput is upper bounded by steiner strength, and $\chi(N) \leq \eta(N)$. □

Though the inequality in $\pi(N) \leq \chi(N)$ can not be replaced by equality in general (an obvious counter-example is shown in Fig. 2), one naturally wonders if $\chi(N) = \eta(N)$ is true, *i.e.*, if the optimal throughput is equivalent to its upper bound — the steiner strength of the network. If this conjecture holds in general, it would be a nice max-min characterization that essentially renders both the optimal throughput problem and the steiner strength problem Co-NP[1]. Given that the optimal throughput problem is apparently NP, if it is also Co-NP, it usually implies the existence of an efficient solution.

If we further study our example, we can show that steiner strength may be precisely achieved by using the directed network shown in Fig. 3(b), obtained by appropriately orienting the undirected links. Using the main theorem of network coding in directed networks, the optimal multicast throughput is 1.5 to both $m_1$ and $m_2$. In addition, if we compute the achievable optimal throughput without coding using steiner tree packing, since we can optimally pack 3 trees (annotated with letters, with the tree $a$ highlighted) with a weight of 0.5 in each, the result is also 1.5.

The favorable properties of this example are, in fact, due to its *broadcast nature*: all nodes in the network are in the multicast group. It is known that in the special case of a single *broadcast* session where the source transmits to all nodes, steiner tree packing degrades to *spanning tree packing*, and steiner strength is renamed to *network strength*. The good news is that, the spanning tree packing number is always equal to the network strength in the case of broadcast, independently proved by Tutte [22] and Nash-Williams [23]. It can be derived from Theorem 1 that, $\pi(N) = \chi(N) = \eta(N)$ in the broadcast case.

However, we have investigated the complexity of the steiner strength problem, and find it to be — unfortunately — NP-complete, as proved in Theorem 2.

**Theorem 2.** The steiner strength problem is NP-complete.

*Proof:* We present a brief outline of the proof. We can reduce another well-known NP-complete problem, *max cut* [24], to the steiner strength problem in polynomial time. The reduction works in essentially the same way as the one given by Dahlhaus *et al.*, in their NP-completeness proof for the *multiterminal cut* problem [25]. We observe that in the instance graph constructed in their proof, the optimal multiterminal cut always leads to the minimum $|E_c|/(|p|-1)$ ratio, and is therefore always the optimal partition that corresponds to the steiner strength value of the network. Since the max cut in the original graph corresponds to the optimal multiterminal cut, it also corresponds to the optimal partition for the steiner strength. The remaining steps of the proof can be found in [25] and are omitted in this paper. □

It immediately becomes unlikely for optimal throughput to be equal to steiner strength in general, since this implies that the steiner strength problem is both Co-NP and NP-complete, and such problems are not believed to exist. Still, it is exceedingly hard to find a specific counter-example where the optimal throughput is not equal to the steiner strength. Fig. 4 shows one of the counter-examples that we have found, in which case the steiner strength is 14. We will later show that the achievable optimal throughput is 13.5. With this counter-example, we can conclude that neither of the inequalities in Theorem 1 can be replaced by equality in general.

Though both steiner tree packing and steiner strength are natural research directions towards optimal throughput and offer tight

[1]The reason is that, the existence of a partition with $|E_c|/|p| - 1 = x$ can serve as a short certificate for the claim that no transmission strategy may achieve a higher multicast throughput than $x$, and vice versa.

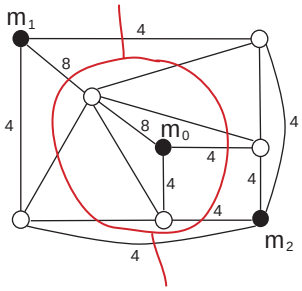



Fig. 4. Optimal throughput and steiner strength: a counter-example. The source is $m_0$. The optimal partition is shown, cutting the network to three components. The steiner strength of the network is 14.

bounds to the problem, they are computationally intractable. It seems to suggest that computing the optimal throughput and its corresponding transmission strategies may also be computationally intractable in general. Our search for efficient solutions to achieve optimal throughput continues.

### *C. Efficient solutions for throughput optimization: The cFlow Linear Program*

Contrary to the previous pessimistic views, we present the surprising result that efficient solutions do exist for computing optimal throughput in undirected networks, including the corresponding routing and coding strategies that achieve such throughput. To design such efficient solutions, we first formulate the problem as a linear optimization problem, in which both the number of variables and the number of constraints are bounded by $O(|M| \cdot |E|)$. We then show that the result of such optimization exactly gives the maximum achievable throughput, as well as the corresponding routing strategy.

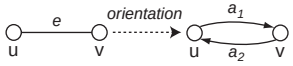



Fig. 5. Orienting each undirected link in the network into two directed ones, such that $c(e) = c(a_1) + c(a_2)$.

We begin by presenting the *orientation constraints* of the linear program that computes optimal throughput. As illustrated in Fig. 5, an *orientation* of a network $N$ is a strategy to replace each undirected link $e = uv$ with two directed links $a_1 = \overrightarrow{uv}$ and $a_2 = \overrightarrow{vu}$, such that $C(e) = C(a_1) + C(a_2), \forall e \in E$. After the orientation, the undirected link set $E$ becomes a directed link set $D$, with the number of links in the set doubled. Since the capacity of a directed link must be non-negative, we also have $C(a) \geq 0, \forall a \in D$. For example, Fig. 3(b) is an orientation of the undirected network in Fig. 3(a).

We proceed to consider flows from the source to the multicast receivers. To take advantage of the power of network coding to resolve competition for link capacities, we introduce the concept of *conceptual flows* (*cFlow*). We define conceptual flows as network flows that co-exist in the network *without* contending for link capacities. In comparison, if we view information flows as traditional *commodity flows* (where data can only be forwarded, and can not be coded or replicated), they do compete for the capacity of a shared link, similar to the case of all unicast between the source and each of the receivers. Fig. 6 illustrates the fundamental difference between conceptual and commodity flows with respect to achievable optimal throughput, in a random network of 20 nodes generated by BRITE [26], with the size of the multicast group increasing. The rapid decrease of optimal session throughput is due to competition for shared link capacities among different commodity flows, especially at the source.

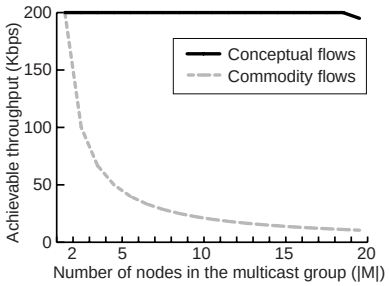



Fig. 6. Achievable optimal throughput: a comparison between *conceptual* and *commodity* flows, as the number of nodes in the multicast session increases.

Our linear program to compute the optimal throughput with coding, shown in Table I, is referred to as the *cFlow* LP since it is based on conceptual flows. In the LP, $m_0$ is the source and $m_1 \ldots m_k$ are the multicast receivers. $f^1 \ldots f^k$ are the conceptual flows to each of the receivers. Each $f^i$ specifies a flow rate $f^i(a)$ for each directed link $a \in D$. $f^i_{in}(v)$ denotes the total incoming $f^i$ flow rate at a node $v$, similar for $f^i_{out}(v)$. Finally, the scalar $f^*$ is the target flow rate of optimization.

In addition to the orientation constraints, the *cFlow* LP also includes the *network flow* constraints for each conceptual flow, and the *equal rate* constraints. The network flow constraints are specified in a compact form for all conceptual flows, which requires the following: (1) Flow rates must be non-negative and upper bounded by link capacities; (2) *flow conservation*, which requires that the incoming flow rate in the conceptual flow $f^i$ equals to outgoing flow rate in $f^i$ at a relay node for $f^i$; and (3) the incoming flow rate at the source and the outgoing flow rates at the receiver are all zero, for each $f^i$. The equal rate constraints require that the flow rates of conceptual flows are identical, with $f^*$ being the uniform flow rate. With these linear constraints, the target flow rate $f^*$ is then maximized.

TABLE I
THE *cFlow* LP

**Maximize:** $f^*$
**Subject to:**
Orientation constraints:

$$\begin{cases} 0 & \leq & C(a) & \forall a \in D \\ C(a_1) + C(a_2) & = & C(e) & \forall e \in E \end{cases}$$

Independent network flow constraints for each conceptual flow:

$$\begin{cases} 0 & \leq & f^i(a) & \forall i \in [1..k], \forall a \in D \\ f^i(a) & \leq & C(a) & \forall i \in [1..k], \forall a \in D \\ f^i_{in}(v) & = & f^i_{out}(v) & \forall i \in [1..k], \forall v \in V - \{m_0, m_i\} \\ f^i_{in}(m_0) & = & 0 & \forall i \in [1..k] \\ f^i_{out}(m_i) & = & 0 & \forall i \in [1..k] \end{cases}$$

Equal rate constraints:

$$f^* \quad = \quad f^i_{in}(m_i) \quad \forall i \in [1..k]$$

We are now ready to formally present one of our main contributions of this paper. We show that the *cFlow* LP provides an efficient algorithm to compute the achievable optimal throughput, as well as the routing strategy.

**Theorem 3.** For an undirected data network with a single multicast session, $N = \{G(V,E), C : E \rightarrow \mathcal{Q}^+, M = \{m_0, m_1, \ldots, m_k\} \subseteq V\}$, the maximum end-to-end throughput $\chi(N)$ *and* its corresponding optimal routing strategy can be computed in *polynomial time* using the *cFlow* LP, in which both the number of variables and the number of constraints are polynomial, and on the order of $O(|M| \cdot |E|)$. The conceptual flows $f^1 \ldots f^k$ constitute the optimal routing strategy.

*Proof:* The orientation constraints reflect complete flexibility in orienting the undirected network $N$, without being too restrictive or too relaxed. For each fixed orientation, conceptual flows are being maximized with independent and standard network flow constraints, as well as the extra constraint that conceptual flow rates are equal to each other. Therefore, the result of the maximization is the maximum possible flow rate that can be independently achieved from the source to all receivers, over all possible orientations of the network:

$$f^* = \max_{o \in O}[\min_{m_i \in M - \{m_0\}}(\text{maximum } m_0 \rightarrow m_i \text{ flow rate})],$$

where $O$ denotes all possible orientations of the network, and $M - \{m_0\}$ is the set of multicast receivers. The recent breakthrough in network coding [5], [6] shows that, for a fixed orientation of the network, a rate $x$ can be achieved for the entire multicast session if and only if it can be achieved for each multicast receiver independently. This implies that, the maximum throughput in each orientation equals to the minimum of the maximum source $\rightarrow$ receiver flow rate. The *cFlow* LP essentially maximizes this min-max flow over all possible network orientations, and obtains the max-min-max flow that is precisely the maximum multicast throughput in the original undirected network. Further, the source may transmit information to each receiver $m_i$ according to the conceptual flow $f^i$. Should more than one conceptual flows utilize capacity on the same link, the conflict can always be resolved, provided that network coding is applied appropriately [5], [6].

The *cFlow* LP contains $2|E|$ orientation variables $C(a)$, $2|M| \cdot |E|$ virtual flow variables $f^i(a)$, and one target flow rate variable $f^*$. Therefore, the total number of variables is $2(|M|+1)|E|+1$, which is on the order of $O(|M| \cdot |E|)$. In addition, the *cFlow* LP contains $3|E|$ orientation constraints, $(4|E|+|V|)(|M|-1)$ network flow constraints, as well as $|M|-1$ equal rate constraints. The total number of constraints is, therefore, $(4|E| + |V| + 1)(|M| - 1) + 3|E|$, which is also on the order of $O(|M| \cdot |E|)$. □

The optimal routing strategy computed by *cFlow* LP specifies the rate of data streams being transmitted along each link. Based on the routing strategy, we need to perform the additional step of *code assignment* to compute the *coding* strategy, before data streams may be transmitted. The coding strategy includes one transformation matrix for each node, which specifies how incoming data streams are linearly coded into outgoing streams. Given the routing strategy from the *cFlow* LP, there exist polynomial time algorithms to perform such code assignments. For completeness of this paper, we include a more detailed discussion in the Appendix. The existence of polynomial time code assignment algorithms leads to the following corollary of Theorem 3:

**Corollary.** The complete solution that achieves optimal throughput in undirected data networks with a single multicast session can be computed in polynomial time, including both the routing and coding strategies.

As an example of applying the *cFlow* LP on actual networks, Fig. 7 shows the optimal orientation to achieve the optimal throughput of $13.5$ in the previous network in Fig. 4.

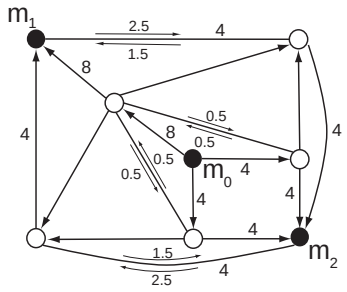



Fig. 7. The optimal network orientation computed by the *cFlow* LP, in which the maximum $m_0 \rightarrow m_1$ flow and the maximum $m_0 \rightarrow m_2$ flow are both $13.5$.

In order to evaluate the advantage of network coding with respect to improving achievable optimal throughput, we have implemented both the *cFlow* LP and a brute-force algorithm to compute the steiner tree packing number. The steiner tree packing algorithm enumerates all steiner trees in the network, assigns a flow variable to each tree, and then maximizes the summation of all tree flows, subject to the constraints that the total weight (throughput) of trees using each link should not exceed its capacity.

We have evaluated both the *cFlow* LP and the steiner tree packing algorithm using our previous examples, as well as a set of *uniform bipartite* networks, which are believed to be good candidates to show the power of coding on improving throughput [7], [27]. A uniform bipartite network $C(n,k)$ consists of the data source and two layers: one with $n$ relay nodes and the other with $\binom{n}{k}$ receivers. Each relay node is connected to the sender, and each receiver is connected to a different group of $k$ relay nodes, and all links have a capacity of 1. For instance, $C(4,3)$ is shown in Fig. 8, the example in Fig. 2 is $C(3,2)$, and the classic example of network coding in Fig. 1 is isomorphic to $C(3,2)$.

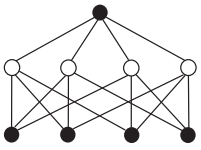


Fig. 8. The uniform bipartite network $C(4,3)$.

Table II summarizes the results of our empirical studies, from which we have derived the following observations. First, the *cFlow* LP is much more scalable and efficient than steiner tree packing, which fails to compute a solution for a network as small

as $C(5,3)$, with only 16 nodes and 35 links, but almost 50 million different steiner trees. In separate experiments, the *cFlow* LP is able to compute the optimal throughput for networks having thousands of nodes. Second, the results support Theorem 1, and show that the optimal throughput with coding is always lower bounded by that without coding; however, network coding only introduces a slight advantage, with the $\chi(N)/\pi(N)$ ratio no higher than 1.125, even in uniform bipartite networks where coding is particularly beneficial.

TABLE II

COMPUTING OPTIMAL THROUGHPUT: *cFlow* LP VS. STEINER TREE PACKING

| Network | $\|V\|$ | $\|M\|$ | $\|E\|$ | $\chi(N)$ | $\pi(N)$ | $\frac{\chi(N)}{\pi(N)}$ | # of trees |
|---|---|---|---|---|---|---|---|
| Fig. 1 | 7 | 3 | 9 | 2 | 1.875 | 1.067 | 17 |
| $C(3,2)$ | 7 | 4 | 9 | 2 | 1.8 | 1.111 | 26 |
| Fig. 7 | 8 | 3 | 16 | 13.5 | 13.5 | 1.0 | 298 |
| $C(4,3)$ | 9 | 5 | 16 | 3 | 2.667 | 1.125 | 1,113 |
| $C(4,2)$ | 11 | 7 | 16 | 2 | 1.778 | 1.125 | 1,128 |
| $C(5,4)$ | 11 | 6 | 25 | 4 | 3.571 | 1.12 | 75,524 |
| $C(5,2)$ | 16 | 11 | 25 | 2 | 1.786 | 1.12 | 119,104 |
| $C(5,3)$ | 16 | 11 | 35 | 3 | – | – | 49,956,624 |

## IV. ACHIEVING OPTIMAL THROUGHPUT IN UNDIRECTED DATA NETWORKS: MORE GENERAL CASES

Our efficient solution, the *cFlow* LP, can be extended to solve the optimal throughput problem in cases beyond a single multicast session. We present our contributions to extend the *cFlow* LP to unicast, broadcast and group communication sessions, as well as to the case of multiple communication sessions and to the model of overlay networks.

### *A. The cases of unicast, broadcast and group communication sessions*

Since unicast and broadcast can be viewed as special cases of multicast where two nodes and all nodes are in the multicast group, respectively, our solution in the single multicast case can be readily applied to a single unicast or broadcast session without modifications. In the case of a unicast session, the *cFlow* LP essentially solves a linear program for a single network flow. In the case of a broadcast session, the *cFlow* LP computes the optimal broadcast throughput, which has been shown to be the same as both the spanning tree packing number and the network strength.

Traditionally, these three equal quantities have been computed from either the perspective of network strength or spanning tree packing. Cunningham [28] first gave a combinatorial algorithm that computes the network strength, which was later improved by Barahona [29]. Both algorithms are based on matroid theory, and are highly sophisticated. Though the spanning tree packing problem has an LP formulation, the number of variables is exponential. It is therefore necessary to work on its dual program, where the minimum spanning tree algorithms can serve as the separation oracle. In comparison, the *cFlow* LP is much simpler to implement yet efficient to run, with a polynomial number of constraints and variables, building upon mature theories and algorithms of linear optimization.

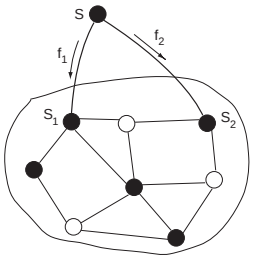



Fig. 9. Transforming group communication into multicast transmission.

Group communication refers to many-to-many communication sessions where multiple sources multicast independent data to the same group of receivers, the set of senders and the set of receivers may or may not overlap. Previous work [6] has shown that a many-to-many session can be easily transformed into a multicast session, by adding a *super* source, which is a traditional technique in network flows. As illustrated in Fig. 9, we can add an additional source $S$ to the network, and connect it to each of the sources in the group communication session, with links of unbounded capacity. We may then apply the *cFlow* LP to maximize the multicast throughput from $S$ to all the receivers. Additional constraints can be applied to flow rates on the newly added links between the super source and the original sources in the session, governing fairness among the original sources. The outcome from the *cFlow* LP is the optimal throughput and its corresponding routing strategy for the original group communication session.

### *B. The case of multiple sessions*

In its most general form, the optimal throughput problem allows multiple communication sessions of different types to coexist in the same network. Since multicast is representative — in that unicast, broadcast and group communication can all be transformed into multicast — it is sufficient to consider the optimal throughput problem in the case of multiple multicast sessions.

To achieve optimal throughput with multiple sessions, we need to consider the problem of inter-session fairness. The definition of fairness is usually application dependent; however, as long as it can be expressed using linear constraints, we can easily include them in the LP formulation. With respect to network coding in multiple sessions, it is theoretically possible to apply network coding on multiple incoming streams of different sessions. However, we argue against this possibility, and use *coding by superposition* [5], *i.e.*, network coding is only applied to incoming streams of the same session. This argument is mainly supported by the computational intractability of the optimal throughput problem if inter-session coding is allowed[2]. In addition, our empirical experiences show that allowing inter-session coding can hardly improve optimal throughput, and it is not practical to code data streams from different applications.

The *mFlow* LP given in Table III is designed to solve the optimal throughput problem with multiple multicast sessions, where we use weighted proportional fairness as the fairness model. It

[2]It is known that finding sufficient and necessary conditions for the feasibility of multiple sessions in this case is equivalent to finding a point in an algebraic variety, which is NP-hard [6].

is the result of extending the *cFlow* LP to its multicommodity variant. We assume there exist a total of $s$ multicast sessions, numbered as $1 \dots s$. Each session $i$ has a source $m_{i_0}$, a number of receivers $m_{i_1} \dots m_{i_{k_i}}$, a set of conceptual flows $f^{i_1} \dots f^{i_{k_i}}$, as well as a weight $w_i$ indicating the importance of the session. The scalar $f^{i*}$ is the common rate for conceptual flows within session $i$, the scalar $f^*$ is the common weighted throughput for all the multicast sessions, and the target of the *mFlow* LP is to maximize $f^*$.

The *mFlow* LP replaces the standard network flow constraints in the *cFlow* LP with a set of multicommodity *cFlow* constraints. Since flows of different sessions contend for link capacity, the summation of the per-session flow rates should not exceed link capacities. Since flows within the same session do not compete for link capacity, the effective flow rate within a session $i$ on link $a$ is $f^i(a) = \max_{j \in [1..k_i]} f^{i_j}(a)$. The max function is not linear, so this constraint is relaxed to $f^i(a) \geq f^{i_j}(a), \forall j \in [1 \dots k_i]$. The validity of this relaxation will be shown in the proof of Theorem 4.

It is easy to check that the total number of variables and the total number of constraints in the *mFlow* LP are both on the order of $O(s \cdot |M| \cdot |E|)$, where $s$ is the number of sessions and $|M|$ is the size of the largest multicast group.

TABLE III
THE *mFlow* LP

**Maximize:** $f^*$

**Subject to:**

Orientation constraints:

$$\begin{cases} 0 & \leq & C(a) & \forall a \in D \\ C(a_1) + C(a_2) & = & C(e) & \forall e \in E \end{cases}$$

Multicommodity *cFlow* constraints:

$$\begin{cases} 0 & \leq & f^{i_j}(a) & \forall i \in [1..s], \forall j \in [1..k_i], \\ & & & \forall a \in D \\ f^{i_j}(a) & \leq & f^i(a) & \forall i \in [1..s], \forall j \in [1..k_i], \\ & & & \forall a \in D \\ \sum_{i=1}^{s} f^i(a) & \leq & C(a) & \forall a \in D \\ f_{in}^{i_j}(v) & = & f_{out}^{i_j}(v) & \forall i \in [1..s], \forall j \in [1..k_i] \\ & & & \forall v \in V - \{m_{i_0}, m_{i_j}\} \\ f_{in}^{i_j}(m_{i_0}) & = & 0 & \forall i \in [1..s], \forall j \in [1..k_i] \\ f_{out}^{i_j}(m_{i_j}) & = & 0 & \forall i \in [1..s], \forall j \in [1..k_i] \end{cases}$$

Equal rate constraints:

$$f^{i*} = f_{in}^{i_j}(m_{i_j}) \quad \forall i \in [1..s], \forall j \in [1..k_i]$$

Fairness constraints:

$$f^* = f^{i*}/w_i \quad \forall i \in [1..s]$$

**Theorem 4.** In the case of multiple multicast sessions, the optimal end-to-end throughput and its corresponding optimal routing strategy in undirected data networks can be correctly computed by the *mFlow* LP, with coding by superposition.

*Proof:* The correctness of the *mFlow* LP builds upon the correctness of the *cFlow* LP, which we have proved. We only need to verify that the relaxation of $f^i(a) = \max_{j \in [1 \dots k_i]} f^{i_j}(a)$ to $f^i(a) \geq f^{i_j}(a), \forall j \in [1 \dots k_i]$ does not affect the overall result of optimization. Note that variables $f^i(a)$ appear only once later in the linear program, where the total effective flow rates are required to be upper bounded by link capacities. Therefore, if there is a feasible solution point for the linear program in which $f^i(a) \neq \max_{j \in [1 \dots k_i]} f^{i_j}(a)$, changing the values of $f^i(a)$ so that $f^i(a) = \max_{j \in [1..k_i]} f^{i_j}(a)$ must result in another feasible solution point with the same optimization result, since the new $f^i(a)$ values still satisfy all the constraints. Therefore, the relaxation will not lead to a higher overall optimal value. □

### C. The case of overlay networks

Since neither network coding nor data replication (for IP multicast) are widely supported in the current-generation network elements in the core, we consider the case of *overlay networks* where only the end hosts have the full capabilities to forward, replicate and code data streams, and the core network elements (henceforth referred to as *routers*) may only forward data packets as is. We note that the case of overlay networks is actually more general than the classical model of undirected data networks we have used so far, which hints that the optimal throughput problem may become harder to solve.

Let $N = \{G(V, E), C :\rightarrow \mathcal{Q}^+, M = \{m_0, \dots, m_k\}, H = M \cup \{m_{k+1}, \dots m_h\} \subseteq V\}$ be an overlay network with a multicast session. The multicast group $M$ is a subset of the end hosts $H$. If $M = H$, *i.e.*, all end hosts are in the multicast group, Garg *et al.* [30] has shown that the optimal multicast throughput can be efficiently computed in this case, by working on the dual program of a natural LP formulation. It has also been shown in [30] that, in the general case the optimal throughput problem without network coding is the overlay steiner tree packing problem, and is still APX-hard.

With the support of network coding, however, we are able to extend the *cFlow* LP to its overlay variant, referred to as the *oFlow* LP, to solve the optimal throughput problem in the model of overlay networks. The *oFlow* LP takes a hierarchical view of the multicast transmission, with an *underlay* and an *overlay* level. The underlay level corresponds to the physical network topology, and has multicommodity flows $g^{ij}$ connecting each pair of end hosts $m_i$ and $m_j$, only via routers as intermediate nodes. The overlay level is conceptual, and contains end hosts fully connected as a complete graph. The link $a'_{ij}$ from $m_i$ to $m_j$ has a capacity equal to the underlay flow rate $g^{ij}$. We then apply the *cFlow* LP in the overlay level to maximize the end-to-end throughput, where each node is capable of replication and coding.

In the *oFlow* LP shown in Table IV, we include three groups of constraints. First, the orientation constraints are identical to those included in the *cFlow* LP. Second, the standard multicommodity flow constraints are specified for the underlay flows between end hosts and only via routers. Third, we introduce the mapping constraints that map the underlay $g^{ij}$ flow rate to the overlay link capacity (referred to as $C'(a'_{ij})$), and then apply the original constraints in the *cFlow* LP at the overlay level. The target of the *oFlow* LP is to maximize throughput in the overlay level. It is easy to derive that the number of variables and the number of constraints in the *oFlow* LP are both bounded by $O(|H|^2 \cdot |E|)$.

TABLE IV
THE *oFlow* LP

**Maximize:** $f^*$
**Subject to:**
Orientation constraints:

$$\begin{cases} 0 & \leq & C(a) & \forall a \in D \\ C(a_1)+C(a_2) & = & C(e) & \forall e \in E \end{cases}$$

Underlay multicommodity flow constraints:

$$\begin{cases} 0 & \leq & g^{ij}(a) & \forall i,j \in [1..h], \forall a \in D \\ \sum g^{ij}(a) & \leq & C(a) & \forall i,j \in [1..h], \forall a \in D \\ g^{ij}_{in}(v) & = & g^{ij}_{out}(v) & \forall i,j \in [1..h], \forall v \in V-H \\ g^{ij}_{in}(v) & = & 0 & \forall i,j \in [1..h], \forall v \in H-\{m_j\} \\ g^{ij}_{out}(v) & = & 0 & \forall i,j \in [1..h], \forall v \in H-\{m_i\} \end{cases}$$

Overlay *cFlow* constraints:

$$\begin{cases} C'(a'_{ij}) & = & g^{ij}_{out}(m_i) & \forall i,j \in [1..h] \\ 0 & \leq & f^i(a') & \forall i \in [1..k], \\ & & & \forall a' \in D' = \{a'_{ij} | 1 \leq i,j \leq h\} \\ f^i(a') & \leq & C'(a') & \forall a' \in D', \forall i \in [1..k] \\ f^i_{in}(v) & = & f^i_{out}(v) & \forall i \in [1..k], \forall v \in H-M \\ f^i_{in}(m_0) & = & 0 & \forall i \in [1..k] \\ f^i_{out}(m_i) & = & 0 & \forall i \in [1..k] \\ f^* & = & f^i_{in}(m_i) & \forall i \in [1..k] \end{cases}$$

**Theorem 5.** In the case of a single multicast session in the model of overlay networks, the optimal end-to-end throughput and its corresponding optimal routing strategy can be correctly computed by the *oFlow* LP.

*Proof:* Since relay nodes in the overlay network can not replicate or encode data, a data stream that is transmitted between two end hosts without passing a third end host remains unchanged throughout the transmission and upon arrival. Therefore, it is valid to model these direct transmissions between end hosts as multicommodity flows. The validity of the *cFlow* constraints in the overlay layer may be derived from the correctness of the *cFlow* LP, which we have proved in Theorem 3. □

Similar to the extension from *cFlow* to *mFlow*, one may extend the *oFlow* LP into its multicommodity variant to accommodate multiple sessions in overlay networks. More specifically, one needs to replace the overlay *cFlow* constraints with the overlay *mFlow* constraints in the third group of constraints of the *oFlow* LP. The resulting linear program has both its number of variables and number of constraints bounded by $O((|H|^2 + s|M|) \cdot |E|)$. This is usually not worse than those of the single-session *oFlow* LP, since $|H|^2$ dominates $s|M|$ in most cases.

## V. ACHIEVING OPTIMAL THROUGHPUT: EMPIRICAL AND THEORETICAL INSIGHTS

Due to the lack of efficient algorithms, previous studies on the problem of improving session throughput are largely based on experimental or intuitive insights. We argue that the availability of the *cFlow*, *mFlow* and *oFlow* LPs has significantly changed the landscape, and has made it computationally feasible to study the exact benefits of various proposals to achieve higher throughput, including a single multicast tree with data replication, multiple multicast trees, and network coding. Based on both empirical and theoretical studies, we seek to answer a number of important questions and to present fundamental insights on achieving optimal throughput.

While our theoretical insights are supported by proofs, our empirical studies are based on the implementation of all three LPs that we have proposed[3]. In comparison studies, we have also implemented algorithms to compute the optimal throughput with multiple multicast trees but without coding (*i.e.,* the steiner tree packing number), the optimal throughput with a single multicast tree, as well as the optimal throughput with all unicast from the source to all receivers. Topologies used in our simulations are generated by the BRITE topology generator [26], with sizes ranging from 10 to 500 nodes, both with and without power-law properties, with heavy-tailed or constant link capacities. We use `glpk` as our LP solver of choice.

### *How advantageous is network coding with respect to improving optimal throughput?*

The ratio of achievable optimal throughput with coding over that without coding (*i.e.,* $\chi(N)/\pi(N)$) is referred to as the *coding advantage*. Recall that we have investigated the coding advantage in Table I, where we test several uniform bipartite networks, and are unable to experimentally find cases where network coding may improve optimal throughput by a factor higher than 1.125. We are naturally led to the question: *What is the upper bound of the coding advantage?*

Theoretically, previous work [7] argues that there exist multicast networks where the coding advantage grows proportionally as $\log(|V|)$, and is thus not finitely bounded in general. This result is obtained in directed acyclic networks with the integral routing model. We show that, in undirected networks with the fractional model, the coding advantage is bounded by a constant factor of 2.

**Theorem 6.** For multicast transmissions in undirected data networks, the coding advantage is bounded by a constant factor of 2.

*Proof:* We use $\lambda(N)$ to represent the minimum of the pair-wise connectivity among nodes in the multicast group. We proceed to prove that $\chi(N)/\pi(N) \leq 2$ always holds by showing $\chi(N) \leq \lambda(N)$ and $\frac{1}{2}\lambda(N) \leq \pi(N)$.

Since multicast throughput is always bounded by edge connectivity between the source and a multicast receiver, $\chi(N) \leq \lambda(N)$ obviously holds. In the remainder of this proof, we show $\frac{1}{2}\lambda(N) \leq \pi(N)$ in two steps: (1) from $N$, we obtain a network $N'$ containing a broadcast transmission without relay nodes, such that $\frac{1}{2}\lambda(N') \leq \pi(N')$ holds only if $\frac{1}{2}\lambda(N) \leq \pi(N)$ holds; (2) we prove $\frac{1}{2}\lambda(N') \leq \pi(N')$ holds for $N'$.

In step (1), we repeatedly apply one of the following two operations on the network: (a) apply a complete splitting at a non-cut relay node, preserving pairwise edge connectivities among multicast nodes in $M$; or (b) add a relay node that is a cut node into the multicast group $M$.

[3]All the algorithms we have implemented, as well as all network topologies used in this paper, are publicly available. The location is omitted to accommodate double-blinded reviews.

A split off operation at a node $z$ refers to the replacement of some 2-hop unit path $u$-$z$-$v$ by a direct unit edge between $u$ and $v$, as illustrated in Fig. 10. A complete splitting at $z$ means to repeatedly apply split off operations at $z$ until $z$ is isolated.

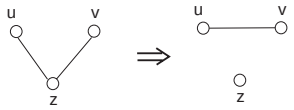



Fig. 10. A split off at node $z$.

Mader's Undirected Splitting Theorem [31] guarantees that operation (a) is always possible, given that non-cut relay nodes exist: *Let $G(V+z, E)$ be an undirected graph so that there is no node-cut incident to $z$ and the degree $d(z)$ is even. Then there exists a complete splitting at $z$ preserving the local edge-connectivities of all pairs of nodes in $V$*. Note that the even node degree requirement is irrelevant in the fractional model.

Therefore, as long as there are relay nodes in the network, we can successfully apply either (a) or (b) to decrease the number of relay nodes by one. Eventually we obtain a network $N'$ containing nodes in the communication group only. Furthermore, operation (a) does not increase $\pi(N)$, and operation (b) does not affect either $\pi(N)$ or $\lambda(N)$. Therefore, if $\frac{1}{2}\lambda(N) \leq \pi(N)$ holds after applying either operation, it holds before applying the operation as well.

We now prove step (2), $\frac{1}{2}\lambda(N') \leq \pi(N')$. By Nash-Williams' Weak Graph Orientation Theorem [31], *a graph has an $x$-edge connected orientation if and only if it is $2x$-edge connected*. Therefore there exists an orientation of $N'$ where the directed connectivity is at least $\frac{1}{2}\lambda(N')$. Within such an orientation, the max-flow from the source to each receiver is at least $\frac{1}{2}\lambda(N')$, and therefore the broadcast throughput achieved is at least $\frac{1}{2}\lambda(N')$. Recall that for broadcast transmissions, optimal throughput with coding equals to that without coding. We thus have $\frac{1}{2}\lambda(N') \leq \pi(N')$. □

Having proved a theoretical bound of 2, We have further studied the coding advantage using over $1000$ *randomly* generated networks, rather than uniform bipartite networks used in Table I. The networks we test are of small sizes (with up to $35$ links), due to the poor scalability of the steiner tree packing algorithm to compute optimal throughput without coding. We have observed that *the coding advantage remains* $1.0$ in all $1000$ random topologies, *i.e.*, network coding is not beneficial to improve optimal throughput. We believe the observation holds for larger networks as well, where the topologies and link capacities remain random. This implies that the fundamental benefit of network coding is *not* higher optimal throughput, but to facilitate *significantly more efficient computation and implementation* of strategies to achieve such optimal throughput.

Our theoretical study has shown a bound of 2, while the highest $\chi(N)/\pi(N)$ ratio we have observed in our empirical studies is only $1.125$. If the highest ratio in general is indeed near the $1.125$ end, it would lead to better approximation algorithms for two theoretical problems: *steiner tree packing* and *minimum steiner tree*, which can be approximated to exactly the same ratio [2], [30]. To date, the best known approximation ratio for these problems is around $1.55$ [3].

***How advantageous is standard multicast compared to unicast and overlay multicast?***

The *cFlow* LP is instrumental to precisely compute the achievable optimal throughput with one multicast communication session, either with network coding or with multiple multicast trees, since the outcomes from the two are hardly different. In either case, data replication need to be supported on all network nodes, including core network elements. It has been common knowledge that, when compared to unicast from the source to all receivers, standard multicast brings better bandwidth efficiency and higher end-to-end session throughput. However, even in the case of unicast, path diversity needs to be exploited to achieve optimal throughput, equivalent to the maximum unicommodity flow problem. It is not immediately clear how advantageous standard multicast is.

Overlay multicast balances the tradeoff between the practicality of standard multicast and unicast. It refers to the case where only the members of the multicast group may replicate or code data, whereas all other nodes may only forward data. The optimal throughput achieved by overlay multicast is efficiently computed by the *oFlow* LP.

We perform a quantitative study that compares the optimal throughput achieved with standard multicast, overlay multicast and unicast. The study is performed in random networks of different sizes, the largest of which has $500$ nodes and over $1000$ links. There are $3$ and $10$ members in the multicast group respectively, in two different sets of tests. Multicast nodes are randomly selected from all network nodes, with different multicast groups being as disjoint as possible. For each network size, multiple tests are performed with different network topologies and different choices of the multicast group, the results are then averaged.

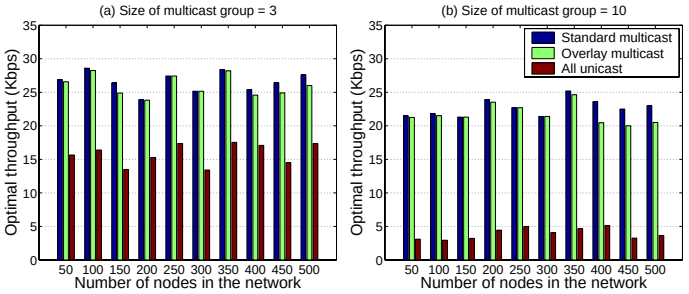



Fig. 11. Achievable optimal throughput using standard multicast, overlay multicast, and all unicast from the sender to all receivers.

As we may observe from Fig. 11, there exists obvious differences between standard multicast throughput and all unicast throughput, and the differences are more significant in Fig. 11(b), where the scale of the multicast transmission is larger. Surprisingly, the figure also suggests that, the optimal throughput achieved by overlay multicast is almost identical to that achieved by standard multicast, where all network nodes are able to replicate or code data. On average, the optimal throughput of overlay multicast is over $95\%$ of standard multicast. This observation shows that, from the perspective of maximum achievable throughput, while there may exist contrived network topologies that show more significant advantages of standard multicast over overlay multicast, little difference remains once large scale practical network topologies are considered.

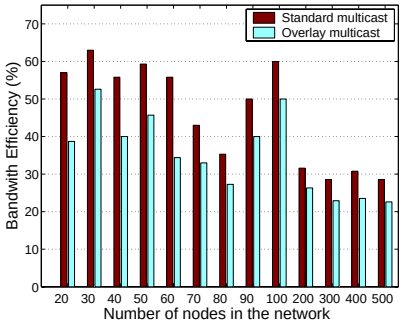



Fig. 12. Bandwidth efficiency of standard multicast and overlay multicast.

Fig. 12 shows that the near-optimal throughput of overlay multicast comes with a price, in the form of a slightly lower *bandwidth efficiency* than that of standard multicast. In this comparison, *bandwidth efficiency* is computed as the achievable session throughput divided by the total bandwidth consumption on all links, and then scaled by the number of receivers in the multicast group. This definition implies that an end-to-end transmission strategy that consumes bandwidth on a large set of links is not favorable for the same set of receivers. We may observe from both Fig. 11 and Fig. 12 that, for random network topologies, there naturally exist residual capacities after applying the optimal transmission strategy of standard multicast. Such residual capacities may be exploited by overlay multicast to improve its optimal throughput, and contribute to the insignificant difference between the optimal throughput achieved by standard and overlay multicast.

In cases where little residual capacity exists in the standard multicast strategy, throughput of overlay multicast should be noticeably lower. This is confirmed by the network in Fig. 7, previously presented in Sec. III. In this example, the transmission strategy of standard multicast utilizes all link capacities to $100\%$, in order to achieve the optimal throughput of $13.5$. The maximum throughput for overlay multicast in this topology is at a lower value of $12$, as expected.

***How sensitive is optimal throughput to node joins?***

When new nodes join the multicast session, how may achievable optimal throughput be affected? Intuitively, if a relay node joins the multicast group and becomes a new receiver, the achievable session throughput should decrease, due to the following two causes: (1) a larger number of receivers may lead to more intense competition for bandwidth; and (2) a new node with low capacity may become a bottleneck and limit the throughput for the entire session. Our simulation results show that, the second cause has a much more significant impact than the first one.

Fig. 13(a) shows variations of optimal throughput as the number of nodes in the multicast group increases from three to $\lceil |V|/2 \rceil$, and then to $|V|$ (effectively a broadcast session), for various network sizes $|V|$. In this experiment, network topologies are generated with two edges per node without power-law relationships, with heavy-tailed bandwidth distribution between 10 and 50 Kbps on the links. As we can observe, when the size of the multicast group increases from three to $\lceil |V|/2 \rceil$, the effects on achievable throughput is rather significant. However, further expanding the multicast group to the entire network leads to a much smaller decrease. Both causes that we have discussed contribute to the initial decrease of throughput, while the second cause (*i.e.*, the effects of a bottleneck node) plays a less important role in the subsequent decrease — when the multicast group contains half of the nodes in the network, it is very likely for the group to have already contained a node with low capacity.

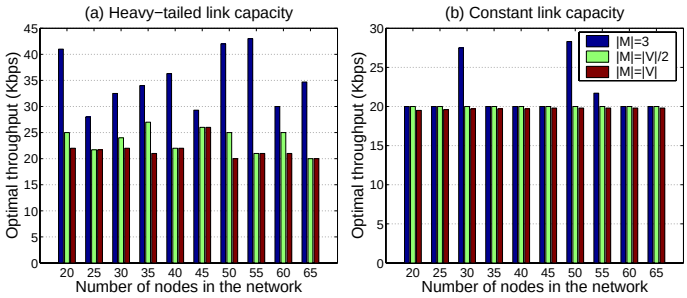



Fig. 13. Variations of optimal throughput due to new nodes joining the multicast session.

We further performed the same tests on power-law network topologies with $10$ Kbps constant link bandwidth, and the results are shown in Fig. 13(b). In the power-law topologies, most nodes have small degrees of two or three, while a small number of nodes have high degrees. Therefore, the initial multicast group usually contains a node with a small degree already, which also has a low capacity, since the link bandwidth is constant. In this case, only inter-receiver bandwidth competition remains as a major concern. However, as we can observe in the figure, in most cases the optimal multicast throughput remains roughly constant, even after all the nodes have joined the multicast session. This counter-intuitive observation shows that, new receivers may share bandwidth with existing receivers well, and do not significantly affect the achievable throughput, as long as their capacities are not too low. Spikes in Fig. 13(b) correspond to the occasional cases where nodes in the initial multicast group all have relatively high capacities. Both results in Fig. 13(a) and 13(b) have led to the same observation that, when new nodes join a multicast session, the decreased optimal throughput is mainly due to bottleneck receivers with lower capacities.

***How sensitive is optimal throughput to the addition of new sessions?***

When new sessions are added to the network, how do they affect achievable optimal throughput? The *mFlow* LP, presented in Sec. IV, makes it feasible to carry out our empirical studies. Fig. 14 shows the variation of optimal throughput as new communication sessions are created. Three types of throughput are shown: (1) *previous optimal*, which represents the optimal weighted session throughput before the new session is added; (2) *incremental*, which is the weighted throughput for the new session using residual link capacities only, or just the previous optimal throughput if the achievable throughput of the new session is higher; and (3) *re-optimized*, which is the re-computed optimal session throughput after the new session is added. Four groups of simulations are performed, with two, three, four, and five existing sessions, respectively, before the new session is established. Each multicast group has a size five, and nodes in different multicast groups are chosen to be as disjoint as possible. Each session

is assigned an equal weight.

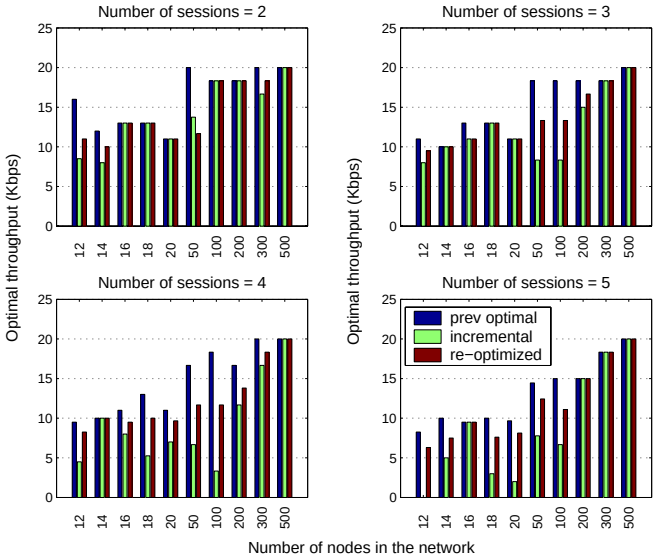



Fig. 14. Throughput variations as a new session is created.

Results in Fig. 14 show that, the addition of an extra session does not dramatically affect the achievable optimal throughput, especially when the network size is large in comparison to the number of nodes involved in the transmissions. However, if the existing sessions remain transmitting according to the optimal transmission strategy computed before the new session joins, and only residual capacities can be utilized to serve the new session (the *incremental throughput* case), then the resulting throughput is not satisfactory unless the number of sessions is very small ($s = 2$). In general, this may lead to very low, even zero, throughput for the new session. Therefore it is necessary to perform re-optimization before a new session starts to transmit.

***How sensitive is optimal throughput to fairness constraints?***

In order to investigate how inter-session fairness requirements affect the optimal throughput, we establish three one-to-two multicast sessions in networks of various sizes between 10 and 350, and computed their total optimal throughput with the following fairness constraints, respectively: (a) no fairness requirement, which leads to the maximum value possible for the total throughput; (b) absolute fairness, in which each session is required to have exactly the same throughput; (c) weighted proportional fairness, where the throughput of each session is proportional to the associated weight of that session; and (d) max-min fairness, in which no session throughput can be increased without decreasing another already smaller session throughput.

As a first small-scale experiment to gain some insights, Fig. 15 shows the total throughput of three sessions in a network with twenty nodes, using the *mFlow* LP. Multicast groups are chosen to be as disjoint as possible. The total weight of three sessions $w_1 + w_2 + w_3 = 1$. As we can see, the weight distribution has a significant impact on the achievable total throughput. When the three weights are heavily unbalanced, the session with the smallest weight can not realize its throughput potential, and consequently leads to a small value of total throughput. The achievable throughput with absolute fairness at $w_1 = w_2 = w_3 = 0.333$ is $91.8$ Kbps. The global optimal throughput $107.0$ Kbps is achieved at $(w_1, w_2, w_3) = (0.287, 0.407, 0.306)$, which turns out to be identical to the throughput with max-min fairness in this case.

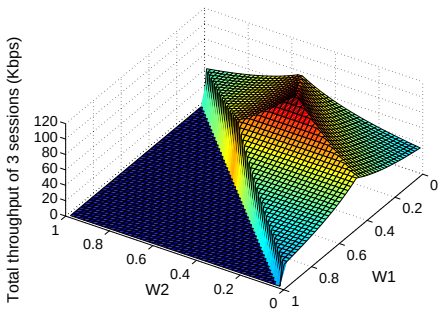



Fig. 15. Total throughput of three multicast sessions, as inter-session fairness requirements change.

Further results in Table V show that the excellent performance of max-min fairness in the above example is not a coincidence. As we may observe, when the network size is relatively large ($50$ and above in the table), max-min fairness always leads to optimal throughput. When the network size is small ($10$ and $20$ in the table), the inter-session competition for bandwidth becomes more intense. The throughput with max-min fairness may be inferior to the optimal throughput in this case, but the difference is usually small.

TABLE V

TOTAL ACHIEVABLE THROUGHPUT WITH MAX-MIN FAIRNESS VS. GLOBAL OPTIMAL THROUGHPUT

| network size | 10 | 20 | 50 | 100 | 150 | 250 | 350 |
|---|---|---|---|---|---|---|---|
| max-min (Kbps) | 120.0 | 136.7 | 173.3 | 160.0 | 146.7 | 146.7 | 183.3 |
| optimal (Kbps) | 126.1 | 140.0 | 173.3 | 160.0 | 146.7 | 146.7 | 183.3 |

***Does optimal throughput lead to low bandwidth efficiency?***

In order to find out whether achieving optimal throughput sacrifices bandwidth efficiency, we have conducted performance comparisons between optimal throughput multicast and single tree multicast. In the latter case, we compute the *widest steiner tree*, which has the highest throughput from all possible multicast trees. The throughput of a tree is defined as the lowest capacity of its links. We choose the tree with the highest throughput rather than the one that is most bandwidth efficient, since the latter is equivalent to the minimum steiner tree problem, which is hard to compute or to approximate. Even when we can find such a bandwidth efficient tree, it may have an exceedingly low throughput, which is not practical for data transmissions.

In Fig. 16, we compare both achievable throughput and bandwidth efficiency between the two approaches. We tested two groups of networks, one with variable link capacity conforming to the heavy-tailed distribution, the other with constant link capacity. For the variable link capacity case, optimal throughput is higher than the widest steiner tree throughput by a factor of over 2 on average, showing the advantage of using the optimal transmission strategy computed with the *cFlow* LP, beyond a single

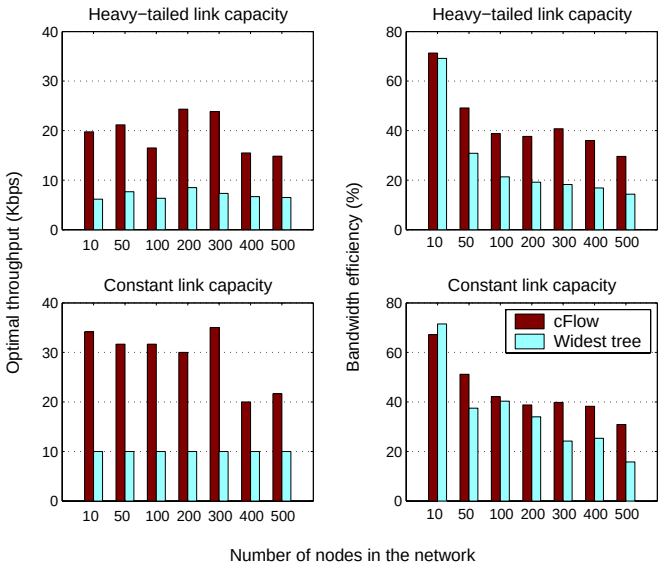



Fig. 16. Achievable throughput and bandwidth efficiency: a comparison between the optimal throughput multicast (*cFlow* LP) and the widest steiner tree.

multicast tree. Interestingly, the bandwidth efficiency of optimal throughput multicast also outperforms that of the widest steiner tree multicast. The widest steiner tree insists to use links with the highest bandwidth possible, and therefore may result in rather long tree branches, especially when the network size is large. For the constant link capacity case, the difference between the optimal and widest steiner tree throughput becomes even larger. Every tree in this case has the same throughput, therefore the "widest" selection criterion becomes irrelevant. However, the difference in bandwidth efficiency decreases, since it is no longer necessary to include long tree branches to achieve the maximum tree throughput.

***Is the achievable optimal throughput dependent on the selection of data sources in a communication session?***

It is rather straightforward to see that optimal throughput without coding is source independent. For example, in the case of multicast without coding, each unit throughput is achieved along a unit-weight steiner tree. The same throughput can be achieved regardless of which node in the multicast group is selected as the root of the trees. It is much less obvious, though, whether the source independent property still holds when network coding is considered. Our simulation results show that, when the data source within a multicast group switches from one node to another, the same optimal throughput is always achieved using our algorithms. We further provide a theoretical proof for this claim.

**Theorem 7.** For an undirected data network with one or more communication sessions, the achievable optimal throughput is completely determined by the topology of the network, the link capacities, the set of nodes in each communication session, as well as the definition of inter-session fairness. It is *irrelevant* to the selection of the data source within each session.

*Proof:* Assume nodes $A$ and $B$ are the source and one of the receivers in session $i$, respectively. Consider moving the data source of session $i$ from node $A$ to node $B$, and exchanging the source/receiver roles between $A$ and $B$. Let **S** be an optimal transmission strategy before the movement. Let $f$ be the flow from $A$ to $B$ in session $i$ before the movement. We show that the transmission strategy **S'** obtained by reversing the directions of information flows in $f$ only, achieves the same throughput for each session as in **S**.

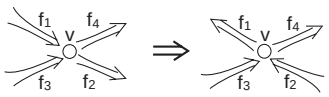



Fig. 17. Flows reversing at an intermediate node $v$.

In **S'**, each receiver is receiving independent information flows at the same rate as it is in **S**. We only need to verify that **S'** is a *valid* transmission scheme, *i.e.*, the information flows on each link can be generated from its preceding flows. Consider a node $v$ in the network. If $f$ does not pass $v$, then the validity at $v$ in **S'** is guaranteed by the validity at $v$ in **S**. Otherwise, as shown in Fig. 17, let $f_1$ and $f_2$ be the set of incoming and outgoing flows at $v$ that belong to $f$, respectively, and let $f_3$ and $f_4$ be the set of incoming and outgoing flows at $v$ that do not belong to $f$, respectively. The validity of **S** implies that $f_1 \cup f_3$ linearly spans $f_2$ and $f_4$. Let $|f_1| = |f_2| = n_1$, $|f_3| = n_2$, and let $f_2 = T_1 f_1 + T_2 f_2$, where $T_1$ and $T_2$ are code matrices of scale $n_1 \times n_1$ and $n_1 \times n_2$ respectively. $|T_1| \neq 0$, since otherwise $f_1$ contains redundant flows that are not necessary to be included in $f$. Therefore $f_1 = T_1^{-1}(f_2 - T_2 f_3)$, and $f_2 \cup f_3$ linearly spans $f_1$. Finally, $f_2 \cup f_3$ linearly spans $f_1 \cup f_3$ and therefore $f_4$. Hence, we have verified the validity of **S'**. □

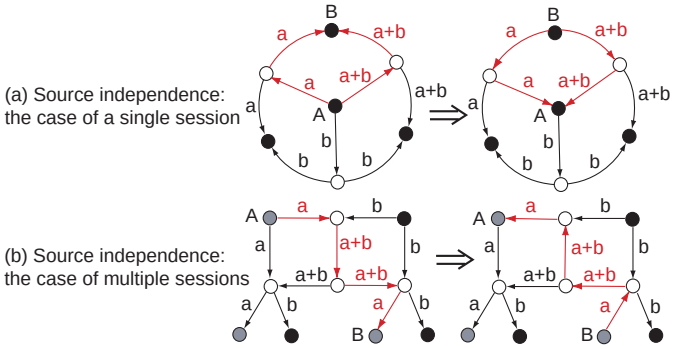



Fig. 18. The achievable optimal throughput is independent from the selection of data sources in each session: two examples.

Fig. 18 shows two examples of the flow reversing procedure discussed in the proof of Theorem 7. The figures on the left show the optimal transmission strategies when the source is at $A$. The ones on the right show the new transmission strategies with the $A \rightarrow B$ flow reversed. It may be verified that the new transmission strategies are valid and optimal, given that the source is moved from $A$ to $B$.

## VI. Concluding Remarks

The main problem we have studied in this paper is to compute and achieve optimal transmission throughput for data communications of various types. We have been pleasantly surprised at how results from network coding are able to facilitate the design of efficient solutions to this fundamental problem that was previously viewed as very hard. We also show the counter-intuitive

conclusion that, the most significant benefit of network coding is not to achieve higher optimal throughput, but to make it feasible to achieve such optimality in polynomial time. We show that such efficient algorithms may be designed for multiple communication sessions of a variety of types, and for the more realistic model of overlay networks.

There exist a number of avenues for further research exploiting the full potential of network coding in data networks. For example, it may be feasible to seamlessly integrate network coding with source erasure codes (*e.g.*, Digital Fountain [14]) to improve fault resilience of data dissemination. We may also extend the theoretical foundations in this paper to wireless networks, addressing their unique characteristics such as spatial contention of capacities. Nevertheless, we firmly believe that it is necessary to start with *theoretical* studies based on the most general form of undirected networks with bounded link capacities and arbitrary source data streams, clearing the paths leading to research in more specific situations.

## APPENDIX: POLYNOMIAL TIME CODE ASSIGNMENT

Li *et al.* [20] proposed the first code assignment algorithm, which performs an exponential number of vector independence tests. Sanders *et al.* [7] observed that, exploiting flow information in the routing strategy dramatically simplifies the task, and gave a polynomial time algorithm, referred to as NIF. The NIF algorithm proceeds from the source to the receivers along links involved in the transmission, in topological order. At each step, codes are assigned for outgoing links of a node, guaranteeing that after the assignment, flows arriving at each receiver are linearly independent. The finite field required is $GF(2^k)$, for $2^k > 2|M|$.

The NIF algorithm works in harmony with *cFlow*, since the *cFlow* LP generates the necessary flow information for NIF as input. However, there are a few additional noteworthy issues. First, the fractional output from *cFlow* needs to be transformed into an integral one, due to the discrete nature of code assignment. An atomic flow rate can be selected, such that each flow in the routing strategy has an integral rate. Round-off operations may be applied if necessary, to avoid generating a large number of small flows. Second, the NIF algorithm assumes a directed acyclic network as input, so that nodes can be naturally processed in topological order. Generally speaking, an optimal routing strategy for an undirected network may contain directed cycles, as in the case of the *cFlow* output. In this case, nodes should be processed along conceptual flows such that flows are synchronized at where they are encoded. Third, the routing strategy from the *cFlow* LP may contain *noise*, in the form of redundant flows circulating around cycles with residual capacities. This problem is actually inherited from regular network flow linear programs. To remove the redundant flows, one may consider solving the *cFlow* LP with multiple target functions, with the primary goal of maximizing the effective flow rate $f^*$, and the secondary goal of minimizing the total bandwidth consumption. A simpler solution that works well in practice is to just maximize the weighted difference between these two target functions, with a much smaller weight associated with the second one.

## REFERENCES


[1] L. Qiu, Y. R. Yang, Y. Zhang, and S. Shenker, "On Selfish Routing in Internet-Like Environments," in *Proc. of ACM SIGCOMM*, 2003.

[2] K. Jain, M. Mahdian, and M. R. Salavatipour, "Packing Steiner Trees," in *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2003.

[3] G. Robins and A. Zelikovsky, "Improved Steiner Tree Approximation in Graphs," in *Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2000.

[4] M. Thimm, "On The Approximability Of The Steiner Tree Problem," in *Mathematical Foundations of Computer Science 2001, Springer LNCS 2136, 678-689*, 2001.

[5] R. Ahlswede, N. Cai, S. R. Li, and R. W. Yeung, "Network Information Flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, July 2000.

[6] R. Koetter and M. Medard, "An Algebraic Approach to Network Coding," *IEEE/ACM Transactions on Networking*, vol. 11, no. 5, pp. 782–795, October 2003.

[7] P. Sanders, S. Egner, and L. Tolhuizen, "Polynomial Time Algorithm for Network Information Flow," in *Proceedings of the 15th ACM Symposium on Parallelism in Algorithms and Architectures*, 2003.

[8] Z. Wang and J. Crowcroft, "Quality of Service Routing for Supporting Multimedia Applications," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 7, pp. 1228–1234, September 1996.

[9] A. J. Ballardie, P. F. Francis, and J. Crowcroft, "Core Based Trees," August 1993.

[10] Y. Chu, S. G. Rao, S. Seshan, and H. Zhang, "A Case for End System Multicast," *IEEE Journal on Selected Areas in Communications*, pp. 1456–1471, October 2002.

[11] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable Application Layer Multicast," in *Proc. of ACM SIGCOMM*, August 2002.

[12] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "SplitStream: High-Bandwidth Multicast in Cooperative Environments," in *Proc. of the 19th ACM Symposium on Operating Systems Principles (SOSP)*, October 2003.

[13] V. Padmanabhan, H. Wang, P. Chou, and K. Sripanidkulchai, "Distributing Streaming Media Content Using Cooperative Networking," in *Proc. of NOSSDAV 2002*, May 2002.

[14] J. Byers and J. Considine, "Informed Content Delivery Across Adaptive Overlay Networks," in *Proc. of ACM SIGCOMM*, August 2002.

[15] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat, "Bullet: High Bandwidth Data Dissemination Using an Overlay Mesh," in *Proc. of the 19th ACM Symposium on Operating Systems Principles (SOSP 2003)*, 2003.

[16] Y. Azar, E. Cohen, A. Fiat, H. Kaplan, and H. Racke, "Optimal Oblivious Routing in Polynomial Time," in *Proc. of the 35th ACM Symposium on the Theory of Computing (STOC)*, 2003.

[17] D. Applegate and E. Cohen, "Making Intra-Domain Routing Robust to Changing and Uncertain Traffic Demands: Understanding Fundamental Tradeoffs," in *Proc. of ACM SIGCOMM*, August 2003, pp. 313–324.

[18] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Prentice Hall, Upper Saddle River, New Jersey, 1993.

[19] T. C. Hu, "Multi-commodity network flows," *Operations Research*, vol. 11, pp. 344–360, 1963.

[20] S. Y. R. Li, R. W. Yeung, and N. Cai, "Linear Network Coding," *IEEE Transactions on Information Theory*, vol. 49, pp. 371, 2003.

[21] S. Chen, O. Günlük, and B. Yener, "The Multicast Packing Problem," *IEEE/ACM Transactions on Networking*, vol. 8, no. 3, pp. 311–318, 2000.

[22] W. T. Tutte, "On the Problem of Decomposing a Graph Into $n$ Connected Factors," *J. London Math. Soc.*, vol. 36, pp. 221–230, 1961.

[23] C. St. J. A. Nash-Williams, "Edge-disjoint Spanning Trees of Finite Graphs," *J. London Math. Soc.*, vol. 36, pp. 445–450, 1961.

[24] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, New York, 1979.

[25] E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis, "The Complexity of Multiterminal Cuts," *SIAM Journal on Computing*, vol. 23, no. 1, pp. 864–894, 1994.

[26] A. Medina, A. Lakhina, I. Matta, and J. Byers, *BRITE: Boston University Representative Internet Topology Generator*, http://www.cs.bu.edu/brite.

[27] C. Gröpl, S. Hougardy, T. Nierhoff, and H. J. Prömel, "Steiner Trees in Uniformly Quasi-bipartite Graphs," *Information Processing Letters*, vol. 83, no. 4, pp. 195–200, 2002.

[28] W. H. Cunningham, "Optimal Attack and Reinforcement of a Network," *Journal of the ACM*, vol. 32, pp. 549–561, 1985.

[29] F. Barahona, "Packing Spanning Trees," *Mathematics of Operations Research*, vol. 20, no. 1, pp. 104–115, 1995.

[30] N. Garg, R. Khandekar, K. Kunal, and V. Pandit, "Bandwidth Maximization in Multicasting," in *Proceedings of the 11th European Symposium on Algorithms (ESA)*, 2003.

[31] A. Frank, "Edge-connection of Graphs, Digraphs, and Hypergraphs," Tech. Rep., Enervary Research Group on Combinatorial Optimizations, Budapest, Hungary, www.cs.elte.hu/egres, September 2000.