In IEEE INFOCOM 2005

On Achieving Optimal Throughput with Network Coding

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. Although previous approaches lead to solving NP-complete problems, we 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. This result holds for one or more communication sessions, as well as in the overlay network model. 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.**

Index terms: Graph theory, Information theory, Mathematical programming/optimization, Simulations.

I. Introduction

In its most general form, 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.

The focus on the undirected network model is supported by the following justifications. First, as past research in network flow theory [2] and information theory [3] suggests, the undirected network model has its own rhythm, and results obtained there may be drastically different from those obtained in the directed network model. In fact, the undirected model is more general and fundamental in that, a solution constructed for undirected networks can usually be applied to solve the same problem in directed networks, but not vice versa. This is particularly true for our problem and solution in this paper. Second, undirected communication links provide the complete flexibility in capacity allocation, and consequently leads to higher transmission rates that better represent the optimal information flow rate. Finally, in special network scenarios such as wireless ad hoc networks, communication links are naturally undirected, in the sense that data transmission along both directions of the wireless link share the available spectrum.

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*}.

In this paper, we seek to bring fundamentally new insights and efficient solutions to the problem of optimizing end-to-end throughput in undirected data networks. We first illustrate the power of *network coding* [4], [5] 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 show that, although previous directions of computing optimal multicast throughput involve solving NP-complete problems, the maximum multicast throughput and the corresponding optimal multicast strategy can indeed be computed efficiently *in polynomial time*, with the unique encodable property of information flows considered. We then show that this conclusion can be 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 networks.

The availability of efficient solutions makes it finally possible to study various aspects of properties of the achievable throughput, in realistically sized networks. We present empirical studies based on simulation results over thousands of test scenarios using our algorithms. We compare the optimal multicast throughput with and without network coding, and show that noticeable throughput gains can only be experienced in contrived network topologies; for random and irregular network topologies it is almost always zero. This agrees with out previous theoretical results on the upper bound of the advantage of network coding in undirected networks [3]: rather than increasing throughput, the advantage of network coding is indeed to facilitate significantly more efficient computation of the strategies to achieve *optimal* throughput of information flows. Our empirical studies also show that overlay multicast, which has recently attracted extensive research efforts, may approach optimal throughput quite well.

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 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. We then present empirical studies in Sec. V, and 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 [6]. 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.*, [7]). 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 [8], [9]. Though a single multicast tree may not lead to optimized throughput, recent studies (*e.g.*, SplitStream [10], CoopNet [11], Digital Fountain [12] and Bullet [13]) 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 [14]. 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.

The theory of *network flows* studies the transmission of commodities of the same type (unicommodity flows) through a capacitied network. The maximum flow rate between the source and the destination which may be computed with various efficient combinatorial algorithms [2]. When commodities to be transmitted are of different types (multicommodity flows), computing the maximum flow rate can be solved as a linear optimization problem. 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, the maximum achievable throughput with network coding is 2. Without coding, it can be computed that the optimal throughput is 1.875 [3]. If only one multicast tree is used (as in IP multicast), the achieved throughput is 1.

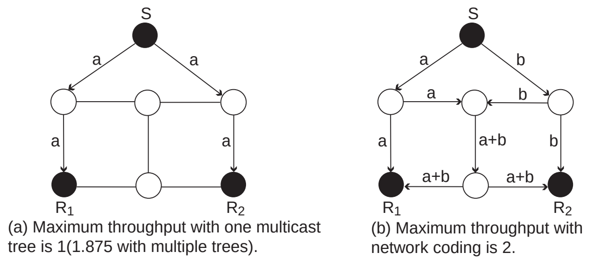
S
a
a
a
a
R1
R2
(a) Maximum throughput with one multicast tree is 1(1.875 with multiple trees).
S
a
b
a
b
a
a+b
b
a+b
a+b
R1
R2
(b) Maximum throughput with network coding is 2.

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 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.* [4] and Koetter *et al.* [5]). In addition, Li *et al.* [15] 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.

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

We begin our 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 $C : E \rightarrow \mathcal{Q}^+$ (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.

We use $M = \{m_0, m_1, \ldots, m_k\} \subseteq V$ to specify the set of nodes in the multicast group, with m_0 being the sender. In graphical illustrations throughout this paper, 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. Steiner tree packing and Steiner strength

To compute the optimal throughput of multicast sessions, *Steiner tree packing* [16], [17] and *Steiner strength* have been the state-of-the-art. Unfortunately, both are NP-hard solutions.

Steiner tree packing. Consider the case of information flows in one multicast session from a source to a set of destinations. It can be theoretically shown that, if coding is not considered, achieving optimal throughput via multiple multicast trees 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. Fig. 2(a) shows such an example. In the figure, each letter corresponds to a distinct Steiner tree, and nine such Steiner trees (a to i) exist in the shown packing scheme, where the tree corresponding to a is highlighted. Since each link with unit capacity needs to accommodates 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.

Unfortunately, Steiner tree packing has been shown to be NP-complete [17], [18], and the best known polynomial time algorithm has an approximation ratio of around 1.55 [18]. With the same example, we can also show that the achievable optimal throughput with network coding is 2 (Fig. 2(b)), which is higher than that achieved without coding. Consequently, even if Steiner tree packing is computationally feasible, it may not always yield the actual optimal multicast throughput.

Steiner strength. In an undirected capacitied network N, we consider partitions of the network where there exists at

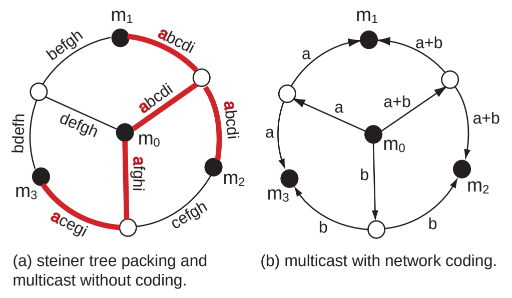
m1
befgh
abcdi
abcdi
abcdi
bdefh
defgh
m0
afghi
m2
m3
acegi
cefgh
(a) steiner tree packing and multicast without coding.
m1
a
a+b
a
a+b
a
a+b
m0
b
m3
m2
b
b
(b) multicast with network coding.

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

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 link capacity on the set of links E_c being cut, and $|p|$ is the number of components in the partition p. It is a natural extension of *network strength* [19] defined for a broadcast network. It is known from our previous work that network strength is equivalent to the achievable optimal throughput in broadcast sessions [3]. Therefore, it is a natural direction to compute optimal multicast throughput by computing the Steiner strength.

Unfortunately, the Steiner strength problem turns out to be NP-complete as well. The fact that computing Steiner strength is NP-complete also rules out the possibility that Steiner strength and optimal multicast throughput are always equal. In fact, we find that Steiner strength is either equal to or higher than the achievable optimal throughput[1].

B. 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. We first formulate the problem as a linear network optimization problem, in which both the number of variables and the number of constraints are bounded by $O(|M||E|)$. We then show that the result of such optimization exactly gives the maximum achievable throughput, as well as the corresponding routing strategy. We also discuss possible solutions to the linear program.

We begin by presenting the *orientation constraints* of the linear program that computes optimal throughput. 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 $C(e) = C(a_1) + C(a_2)$. After the orientation, the set of undirected links E becomes a set of directed links A, with the number of links in the set doubled.

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

[1]Observing space constraints, we exclude the proofs of this result and the NP-completeness of Steiner strength. Interested readers are referred to our technical report [20], which also includes more detailed explanations and an example in which the Steiner strength is higher than the optimal throughput.

flows as network flows that co-exist in the network *without* contending for link capacities.

Our linear program to compute the optimal throughput, shown in Table I, is referred to as the *cFlow* LP since it is based on conceptual flows. In the LP, $f^1 \dots f^k$ are the conceptual flows from sender m_0 to each of the receivers. Each flow vector f^i specifies a flow rate $f^i(a)$ for each directed link $a \in A$. $f^i_{in}(v)$ denotes the total incoming f^i flow rate at a node v, similar for $f^i_{out}(v)$. Finally, the scalar χ 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 (1) flow rates must be upper bounded by link capacities; (2) *flow conservation*, *i.e.*, 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 χ being the uniform flow rate. With these linear constraints, the target flow rate χ is then maximized.

TABLE I
THE *cFlow* LP

Maximize: χ
Subject to:
Orientation constraints:
$$\begin{cases} 0 & \leq & C(a) & \forall a \in A \\ C(a_1) + C(a_2) & = & C(e) & \forall e \in E \end{cases}$$
Independent network flow constraints for each conceptual flow:
$$\begin{cases} 0 & \leq & f^i(a) & \forall i \in [1..k], \forall a \in A \\ f^i(a) & \leq & C(a) & \forall i \in [1..k], \forall a \in A \\ f^i_{in}(v) & = & f^i_{out}(v) & \forall i \in [1..k], \forall v \in V - \{m_0, m_i\} \\ f^i_{in}(m_0) & = & 0 & \forall i \in [1..k] \\ f^i_{out}(m_i) & = & 0 & \forall i \in [1..k] \end{cases}$$
Equal rate constraints:
$$\chi \quad = \quad f^i_{in}(m_i) \quad \forall i \in [1..k]$$

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

Theorem 1. For an undirected data network with a single multicast session, $N = \{G(V,E), C : E \rightarrow \mathcal{Q}^+, M = \{m_0, m_1, \dots, m_k\} \subseteq V\}$, 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 $O(|M||E|)$. The conceptual flows $f^1 \dots f^k$ 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:

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

where O denotes all possible orientations of the network, and $M - \{m_0\}$ is the set of multicast receivers. Recall the recent breakthrough in network coding [4], [5] 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 to 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 [4], [5].

The *cFlow* LP contains $2|E|$ orientation variables $C(a)$, $2|M||E|$ virtual flow variables $f^i(a)$, and one target flow rate variable χ. Therefore, the total number of variables is $2(|M|+1)|E|+1$, which is on the order of $O(|M||E|)$. 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 $O(|M||E|)$. $\square$

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 [21]. Therefore, we have the following corollary of Theorem 1:

Corollary 1. 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.

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 Steiner tree packing (denoted as $\pi(N)$) using our previous example in

Fig. 1, 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 [21], [22]. 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, the network in Fig. 2 is $C(3,2)$, and the classic example of network coding in Fig. 1 is isomorphic to $C(3,2)$.

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, 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. Third, coded transmission may lead to more integral flow rates and throughput than uncoded transmission.

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
$C(4,3)$	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

As a final note, we point out that beyond applying general linear programming solutions — such as the simplex method — it is also possible to design custom-tailored algorithms for the *cFlow* LP, to take advantage of its underlying network flow structure. In an accompanying paper [23], we apply Lagrangian relaxation on the dual program of the cFlow LP, and design a distributed subgradient solution. The algorithm iteratively refines an existing orientation of the original network, until an optimal one is reached. At this point, $|M|$ max-flow computations are invoked to find the optimal multicast throughput.

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 now present its extensions (1) to unicast, broadcast and group communication sessions, (2) to the case of multiple communication sessions, and (3) 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 by our previous work to be the same as both the spanning tree packing number and the network strength [3].

Traditionally, these three equal quantities have been computed from either the perspective of network strength or spanning tree packing. Cunningham [19] first gave a combinatorial algorithm that computes the network strength, which was later improved by Barahona [24]. 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 provides an efficient alternative, with a polynomial number of constraints and variables, and with both general LP solvers and custom-tailored distributed subgradient solutions [23] available.

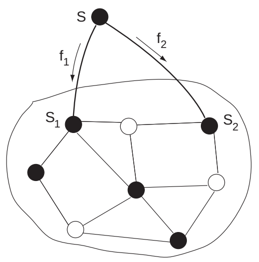

Fig. 3. Transforming group communication into multicast transmission.

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 [5] 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. 3, 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 co-exist 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* [4], *i.e.,* network coding is applied only 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 either.

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 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 χ^i is the common rate for conceptual flows within session i, the scalar χ is the common weighted throughput for all the multicast sessions, and the target of the *mFlow* LP is to maximize χ.

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 $f^i(a) \geq f^{i_j}(a), \forall j \in [1 \dots k_i]$.

Theorem 2. In the case of multiple multicast sessions with coding by superposition, the optimal end-to-end throughput and its corresponding optimal routing strategy in undirected data networks can be computed in polynomial time, by the *mFlow* LP.

Proof: The correctness of the *mFlow* LP builds upon the correctness of the *cFlow* LP, which is proved in Theorem 1, plus the fact that for coding by superposition, data transmission from different sessions constitute totally different commodities when competing for link capacity. Furthermore, it is easy to check that both the number of variables and the number of constraints in the *mFlow* LP are on the order of $O(s|M||E|)$, where s is the number of sessions. $\square$

[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 [5].

TABLE III

THE *mFlow* LP

Maximize: χ

Subject to:

Orientation constraints:

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

Multicommodity *cFlow* constraints:

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

Equal rate constraints:

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

Fairness constraints:

$$\chi = \chi^i / w_i \quad \forall i \in [1..s]$$

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.* [25] 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 [25] that, in the general case the optimal throughput problem without network coding is the overlay Steiner tree packing problem, and is still NP-complete.

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, via only 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 via routers only. 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.

TABLE IV
THE *oFlow* LP

Maximize: χ
Subject to:
Orientation constraints:

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

Underlay multicommodity flow constraints:

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

Overlay *cFlow* constraints:

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

Theorem 3. 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 computed in polynomial time, using 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 1. Furthermore, inspection on the variables and constraints in the *oFlow* LP reveals that, the number of both are on the order of $O(|H|^2|E|)$. □

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 $O((|H|^2 + s|M|)|E|)$. This is usually not worse than those of the single-session *oFlow* LP, since $|H|^2$ dominates $s|M|$ in most cases.

V. EMPIRICAL STUDIES

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. Our empirical studies are based on the implementation of all three LPs that we have proposed. In comparison studies, we have also implemented algorithms to compute the optimal throughput with multiple multicast trees but without coding, the optimal throughput with a widest 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.

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

The ratio of achievable optimal throughput with coding over that without coding is referred to as the *coding advantage*. Recall that we have investigated the coding advantage in Table I, 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?*

Previous work [21] shows that in directed acyclic networks with integral routing requirement, there exist multicast networks where the coding advantage grows proportionally as $\log(|V|)$, and is thus not finitely bounded. However, we found the situation is drastically different in undirected networks. In [3], we use undirected splitting and graph orientation to prove that, for multicast transmissions in undirected networks, *the coding advantage is bounded by a constant factor of* 2.

Given the bound 1.125 obtained for contrived networks, and the bound 2 proven in theory, we further studied the coding advantage in over one thousand *randomly* generated topologies. Our observation is that, for *all* the random topologies we tested, the coding advantage always remains 1.0, *i.e.,* network coding does not introduce any improvement in achievable throughput. 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.

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 with up to 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, 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.

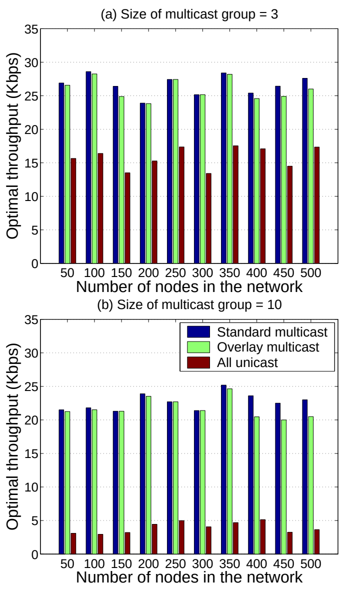

Fig. 4. Achievable optimal throughput using standard multicast, overlay multicast, and all unicast from the sender to all receivers.

As we may observe from Fig. 4, there exists obvious differences between standard multicast throughput and all unicast throughput, and the differences are more significant in Fig. 4(b), where the scale of the multicast transmission is larger. This is due to the fact that with a large number of receivers, the number of unicast flows increases in the all unicast approach, and links incident to the sender become bottlenecks for the transmission. 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. In summary, the all unicast approach does not scale, while overlay multicast may closely approach optimal throughput without requiring core routers to be modified.

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. 5(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.

We further performed the same tests on power-law network topologies with 10 Kbps constant link bandwidth, and the results are shown in Fig. 5(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

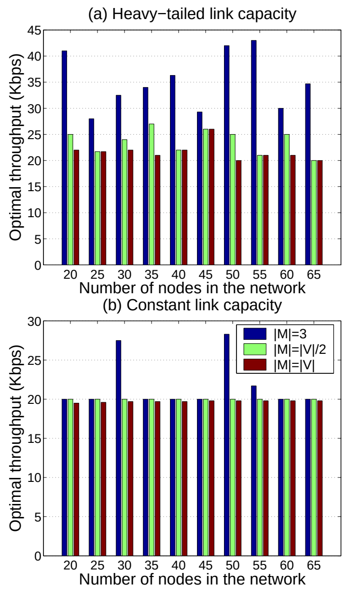

Fig. 5. Variations of optimal throughput due to new nodes joining the multicast session.

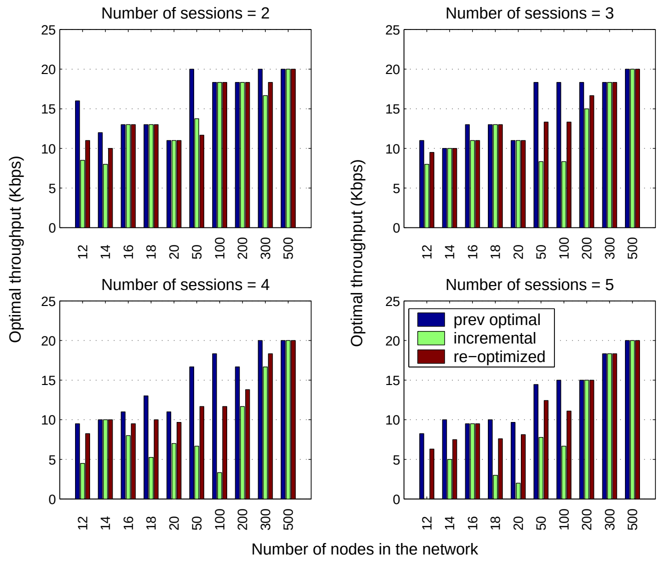

Fig. 6. Throughput variations as a new session is created.

affect the achievable throughput, as long as their capacities are not too low. Spikes in Fig. 5(b) correspond to the occasional cases where nodes in the initial multicast group all have relatively high capacities. Both results in Fig. 5(a) and 5(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. 6 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.

Results in Fig. 6 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. 7 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.

Further results in Table V show that the excellent performance of max-min fairness in the above example is not a

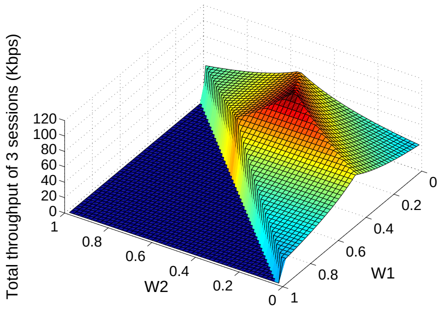

Fig. 7. Total throughput of three multicast sessions, as inter-session fairness requirements change.

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	50	100	150	250	350
max-min (Kbps)	120.0	173.3	160.0	146.7	146.7	183.3
optimal (Kbps)	126.1	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 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. 8, we compare both achievable throughput and bandwidth efficiency between the two approaches. Bandwidth efficiency is computed as the total receiving rate at all receivers divided by the total bandwidth consumption. 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 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

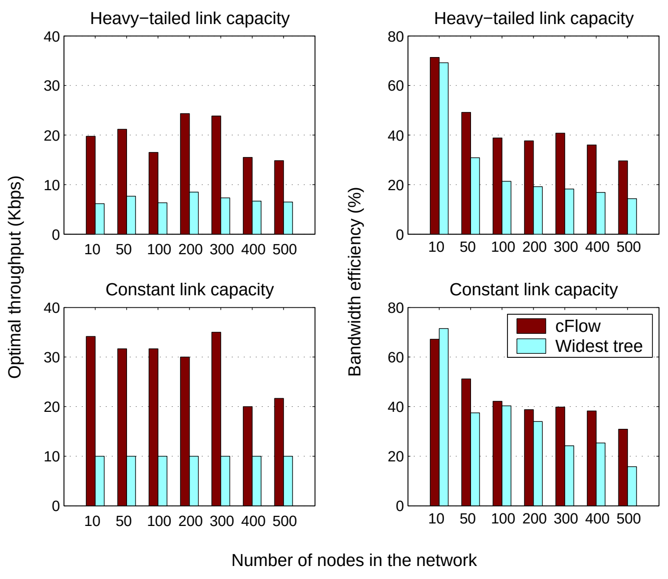

Fig. 8. Achievable throughput and bandwidth efficiency: a comparison between the optimal throughput multicast (*cFlow* LP) and the widest Steiner tree.

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.

VI. Concluding Remarks

The main problem we have studied in this paper is to compute and achieve optimal throughput in data networks, in the general case of undirected communication links. 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. Simulation studies also suggest that, overlay multicast techniques may approach optimal multicast throughput quite well.

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

[3] Z. Li and B. Li, “Network Coding in Undirected Networks,” in *Proc. of the 38th Annual Conference on Information Sciences and Systems (CISS)*, 2004.

[4] 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.

[5] 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.

[6] 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.

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

[8] 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.

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

[10] 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.

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

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

[13] 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.

[14] 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.

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

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

[17] 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.

[18] 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.

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

[20] Z. Li, B. Li, D. Jiang, and L. C. Lau, "On Achieving Optimal End-to-end Throughput in Data Networks: Theoretical and Empirical Studies," Tech. Rep., ECE, University of Toronto, 2004.

[21] 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.

[22] 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.

[23] Z. Li and B. Li, "Efficient Computation of Maximum Multicast Rates," in *Proc. of IEEE INFOCOM*, 2005.

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

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

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