On Achieving Maximum Multicast Throughput in Undirected Networks

Zongpeng Li, Baochun Li, Lap Chi Lau

***Abstract*— The transmission of information within a data network is constrained by the network topology and link capacities. In this paper, we study the fundamental upper bound of information dissemination rates with these constraints in undirected networks, given the unique replicable and encodable properties of information flows. Based on recent advances in network coding and classical modelling techniques in flow networks, we provide a natural linear programming formulation of the maximum multicast rate problem. By applying Lagrangian relaxation on the primal and the dual LPs respectively, we derive (a) a necessary and sufficient condition characterizing multicast rate feasibility, and (b) an efficient and distributed subgradient algorithm for computing the maximum multicast rate. We also extend our discussions to multiple communication sessions, as well as to overlay and ad hoc network models. Both our theoretical and simulation results conclude that, network coding may not be instrumental to achieve better maximum multicast rates in most cases; rather, it facilitates the design of significantly more efficient algorithms to achieve such optimality.**

***Index Terms*— Duality, Multicast, Network Coding, Network Flow, Steiner Tree, Subgradient Optimization, Undirected Networks**

I. Introduction

We study in this paper information dissemination in an undirected network, which 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). Our focus is on multicast, which is representative in that the other types of transmissions are special cases of or can be transformed into multicast transmissions.

Packet transmission in data networks may be modelled as the flow of bit streams, referred to as *information flows*. Compared to classical network flows, information flows may not only be buffered and forwarded, but also be replicated and coded. In previous work, it has been shown that by coding information flows at network nodes, referred to as *network coding* [1], [2], the information flow rate in a multicast session may be improved in directed networks.

In this paper, we seek to bring new insights and efficient solutions to the problem of maximizing information flow rates (or *throughput*) in undirected data networks. We first illustrate the power of *network coding* with respect to achieving maximum throughput. Although previous directions of computing the maximum multicast rates involve solving NP-Complete problems, the maximum multicast rates and the corresponding optimal multicast strategy can indeed be computed efficiently in polynomial time, with the unique encodable property of information flows considered. We provide a natural linear programming formulation of the maximum throughput problem, with a polynomial number of variables and constraints. By applying Lagrangian relaxation on the primal LP, we derive a necessary and sufficient condition for multicast rate feasibility in undirected networks, from a distance labelling perspective. We show how it generalizes correspondent results in the unicast and broadcast cases, and how it connects multicast throughput with network capacity and bandwidth consumption. We further apply Lagrangian relaxation on the dual LP, and construct an efficient subgradient algorithm for computing the maximum multicast throughput and the corresponding optimal transmission scheme. We provide intuitive interpretations of the algorithm, and show that it can be implemented in a distributed fashion.

In addition, we extend the solution to multiple concurrent sessions without inter-session coding, 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 and code data), or wireless ad hoc networks (where data is communicated through antennas), similar modelling and solution techniques are still effective.

The availability of efficient solutions makes it possible to study various aspects of properties of the achievable rates, in realistically sized networks. We present empirical studies based on simulation results over thousands of test scenarios using our algorithms. We compare the maximum multicast rates with and without network coding, and show that noticeable gains can only be experienced in contrived network topologies; for random and irregular network topologies, such gain is almost always non-existent. This agrees with our theoretical results on the upper bound of the advantage of network coding in undirected networks: rather than improving the achievable maximum rates, the advantage of network coding is indeed

Zongpeng Li is affiliated with the Department of Computer Science in University of Calgary, his email address is zongpeng@cpsc.ucalgary.ca. Baochun Li is affiliated with the Department of Electrical and Computer Engineering in University of Toronto, his email adress is bli@eecg.toronto.edu. Lap Chi Lau is affiliated with the Department of Computer Science in University of Toronto, his email address is chi@cs.toronto.edu.

Manuscript received February 2005; revised September 2005; accepted November 2005. This work was done when the first author was affiliated with University of Toronto.

to facilitate significantly more efficient computation of the strategies to achieve the maximum rate of information flows. Our empirical studies also show that overlay multicast, which has recently attracted extensive research efforts, may be used to approach maximum rates quite well. To the best of our knowledge, this work is the first that systematically studies the effects of network coding with respect to maximizing information flow rates in *undirected* networks.

The remainder of this paper is organized as follows. We first discuss related work in Sec. II. From Sec. III to Sec. VI, we present the feasibility condition and efficient solutions for the single multicast case. In Sec. VII, 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 and the model of wireless ad hoc networks. We then present empirical studies in Sec. VIII, and conclude the paper in Sec. IX.

II. Related Work

The 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 provide the desired QoS guarantees [3]. 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.,* [4]). The advantage of IP-based multicast is brought by data packet replication on multicast-capable switches, improving bandwidth efficiency and throughput compared to naive all 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 mesh 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 [5], [6]. Though a single multicast tree may not lead to optimal throughput, recent studies (*e.g.,* SplitStream [7], CoopNet [8], Digital Fountain [9] and Bullet [10]) 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 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 [11]. 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.

Recent research in information theory discovers that routing alone is not sufficient to achieve the maximum information transmission rate across a data network [1], [2]. Rather, applying encoding and decoding operations at relay nodes as well as at the sender and receivers, are in general necessary in an optimal transmission strategy. Such coding operations are referred to as *network coding*. The pioneering work by Ahlswede *et al.* [1] and Koetter *et al.* [2] proves that, in a directed network with network coding support, a multicast rate is feasible if and only if it is feasible for a unicast from the sender to each receiver. Li *et al.* [12] then prove that linear coding usually suffices in achieving the maximum rate.

In [13], Jaggi *et al.* study efficient code assignment in directed acyclic networks. They design polynomial time algorithms that determine the coding operations to be applied at each node, in order to achieve the maximum multicast rate. Their result improves the previous algorithm of Li *et al.*, which performs exponentially many linear independence inspections [12]. Code assignment is complementary to our work in this paper. Our subgradient algorithm finds the optimal routing strategy, which specifies how much flow is to be routed through each link. Code assignment then determines the content of these flows, *i.e.,* their linear relation with the original information flows at the sender.

Traditional network flow theory studies the transmission of goods within a capacitied transportation network. The maximum transmission rate between two nodes is characterized by the celebrated max-flow min-cut theorem [14]: *a flow rate χ between nodes u and v is feasible, if and only if every cut between u and v has size at least χ*. Various algorithms may compute the maximum flow efficiently, some of which allow fully distributed implementation, *e.g.,* the push-relabel algorithm [14] and the ϵ-relaxation algorithm [15]. While information flows also need to confine to network topology and respect link capacities, they are different than commodity flows in that they are replicable and encodable. Data replication and coding are essential in throughput optimization for information flows.

Our subgradient solutions in this paper were also inspired by a preliminary version of [16], in which Lun *et al.* successfully design subgradient algorithms for computing the

min-cost multicast topology in directed networks. Both their algorithm and ours target efficiency and the potential for distributed implementation. Their algorithm works on a partial Lagrangian dual of the primal problem, and employs primal recovery techniques to obtain the entire optimal solution. Our algorithm applies Lagrangian relaxation on the dual problem, and compute the entire optimal primal solution from partial primal solution through pure combinatorial computations.

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 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. We use $n = |V|$ to denote the number of nodes and $m = |E|$ to denote the number of undirected 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.

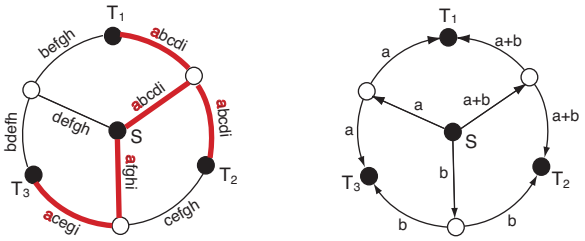
We use $M = \{S, T_1, \ldots, T_k\} \subseteq V$ to specify the set of nodes in the multicast group, with S being the sender, and k being the number of multicast receivers. In graphical illustrations throughout this paper, terminal nodes in M are shown as black, and relay nodes in $V - M$ are shown as white. Links are labelled with their capacities, and unlabelled links each has a capacity of 1.

A. Steiner tree packing and Steiner strength

To compute the optimal throughput of multicast sessions, *Steiner tree packing* [17], [18] 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. 1(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 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.

Unfortunately, Steiner tree packing has been shown to be NP-Complete [17], [19], and the best known polynomial time algorithm has an approximation ratio of $1+\ln 3/2 \approx 1.55$ [19]. With the same example, we can also show that the achievable optimal throughput with network coding is 2 (Fig. 1(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.



(a) Steiner tree packing and multicast without coding.

(b) multicast with network coding.

Fig. 1. The achievable optimal throughput is 1.8 without coding, and 2 with coding.

Steiner strength. In an undirected link-capacitied 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 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* [20] defined for a broadcast network. It appears to be a reasonable direction to compute the optimal throughput by computing the Steiner strength, due to the following two reasons. First, it is known that network strength is equivalent to the achievable optimal throughput in the well-studied case of broadcast networks [21]. Second, these two quantities *are* equal in most networks, especially in real-world networks or randomly generated networks. We are aware of only a few contrived topologies in which they differ from each other. Unfortunately, the Steiner strength problem turns out to be NP-Complete as well.

Theorem 1. 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* [22], 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 [23]. 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 [23] and are omitted in this paper. □

The fact that computing Steiner strength is NP-Complete also rules out the possibility that Steiner strength and optimal multicast throughput are always equal. If the maximum multicast rate always equals to the Steiner strength, then the existence of a partition with $|E_c|/(|p| - 1) = \chi$ can serve as

a short certificate for the claim that no transmission strategy can achieve a higher multicast rate than χ, and vice versa. Consequently both the maximum throughput problem and the Steiner strength problem would be Co-NP. This is very unlikely to be true given the fact the Steiner Strength problem is NP-Complete.

IV. Maximum multicast rate: linear programming formulation

Despite the previous pessimistic views, The introduction of network coding dramatically changes the picture. In this section, We first formulate the maximum multicast throughput problem as a linear network optimization problem, with both the number of variables and the number of constraints bounded by $O(km)$. The resulting LPs are a natural combination of recent advances on multicast with network coding and classical formulation techniques in flow networks. We then show that optimal solutions from the linear program gives exactly the maximum achievable throughput, as well as the corresponding flow routing strategy.

Based on our LP formulation, we derive in Sec. V a necessary and sufficient condition that characterizes multicast rate feasibility in undirected networks. Based on its dual LP, we design in Sec. VI a subgradient algorithm that efficiently solves the LP in a distributed manner. Our primal and dual linear programs have an underlying structure of network flow and cut, respectively. For the ease of understanding and later reference, we present the max-flow and the min-cut linear programs first.

A. The max-flow LP and the min-cut LP

In the max-flow LP, $\vec{TS}$ is a virtual directed link with infinite capacity, going from the destination T to the source S. $N(u) = \{v | uv \in E\}$ denotes the set of neighbors of node u. $f \in \mathcal{Q}_+^A$ is the flow vector, where $A = \{\vec{uv}, \vec{vu} \,|uv \in E\}$ is the set of directed arcs. The scalar χ is the overall end-to-end flow rate. The max-flow LP essentially maximizes the end-to-end flow rate, with link capacity limits and flow conservation requirements (total incoming flow rate at a node equals its total outgoing flow rate). Flow conservation at the source and destination nodes are possible due to the virtual link $\vec{TS}$ we add, on which the flow rate exactly equals the overall flow rate from S to T.

The max-flow linear program

Maximize $\quad \chi = f(\vec{TS})$

Subject to:

$$\begin{cases} f(\vec{uv}) \le C(uv) & \forall\, \vec{uv} \neq \vec{TS} \\ \sum_{v \in N(u)} (f(\vec{uv}) - f(\vec{vu})) = 0 & \forall u \end{cases}$$

$$f(\vec{uv}) \ge 0 \qquad \forall\, \vec{uv}$$

The min-cut linear program

Minimize $\quad \sum_{\vec{uv}} C(uv) y(\vec{uv})$

Subject to:

$$\begin{cases} y(\vec{uv}) + p(v) \ge p(u) & \forall\, \vec{uv} \neq \vec{TS} \\ p(T) - p(S) \ge 1 \end{cases}$$

$$y(\vec{uv}) \ge 0 \qquad \forall\, \vec{uv}$$

In the min-cut LP, vector y indicates which links are "cut". This LP always has an optimal solution that is integral, where each entry in y is valued to either 1 or 0, indicating whether the corresponding link is in the min-cut or not. The constraints imply that, for each path P connecting the source S to the destination T, $\sum_{\vec{uv} \in P} y_i \ge 1$, *i.e.*, at least one link along the path is cut. The objective is to minimize the total link capacity being cut.

B. The primal cFlow linear program

We are now ready to present our linear programming formulation of the maximum multicast rate problem. Intuitively, it orients undirected links into directed ones, and establishes virtual network flows from the sender to each receiver. Vector $c \in \mathcal{Q}_+^A$ stores capacities for directed links, *i.e.*, the allocation of the undirected link capacity in both directions. χ is the overall multicast rate. Vectors $f_i \in \mathcal{Q}_+^A$ denotes the network flow from sender S to receiver T_i. Directed links $\vec{T_iS}$ with infinite capacity are again introduced for a concise presentation of the LP.

Maximize $\quad \chi$

Subject to:

$$\begin{cases} \chi \le f_i(\vec{T_iS}) & \forall i & (4.1) \\ f_i(\vec{uv}) \le c(\vec{uv}) & \forall i, \forall\, \vec{uv} \neq \vec{T_iS} & (4.2) \\ \sum_{v \in N(u)} (f_i(\vec{uv}) - f_i(\vec{vu})) = 0 & \forall i, \forall u & (4.3) \\ c(\vec{uv}) + c(\vec{vu}) \le C(uv) & \forall uv \neq T_iS & (4.4) \end{cases}$$

$$c(\vec{uv}), f_i(\vec{uv}), \chi \ge 0 \qquad \forall i, \forall\, \vec{uv}$$

This linear program is referred to as the *cFlow* LP, reflecting the fact that each network flow f_i is conceptual rather than physical, and two different flows do not contend for the capacity available at a directed link. Constraints in the primal program require capacities allocated to both directions not to exceed the undirected link capacity (4.4), each flow f_i to be a valid network flow (4.2)(4.3), and the multicast rate not to exceed any of these network flow rates (4.1). Essentially, the primal LP tries to establish an orientation of the undirected network, within which it sets up independent network flows from the sender S to each receiver T_i. It does so in an optimal way, in that the minimum of the independent max-flow rates — which by the result of Ahlswede *et al.* [1] is equal to the multicast rate — is maximized. A feasible solution to the primal LP provides an orientation of the original network, $c(\vec{uv})$; a flow routing scheme, $f(\vec{uv}) = \max_i f_i(\vec{uv})$; and a feasible multicast rate, χ.

C. The dual cFlow linear program

The dual linear program for the maximum multicast rate problem is:

$$\begin{array}{ll} \text{Minimize} & \sum_{uv} C(uv)x(uv) \\ \text{Subject to:} & \end{array}$$

$$\begin{cases} x(uv) \geq \sum_i y_i(\vec{uv}) & \forall uv \neq T_iS \quad (4.5) \\ y_i(\vec{uv}) + p_i(v) \geq p_i(u) & \forall i, \forall\, \vec{uv} \neq \vec{T_iS} \quad (4.6) \\ p_i(T_i) - p_i(S) \geq z_i & \forall i \quad (4.7) \\ \sum_i z_i \geq 1 & \quad (4.8) \end{cases}$$

$$x(uv), y_i(\vec{uv}), z_i \geq 0 \qquad \forall i, \forall\, \vec{uv}$$

While the primal LP is in the form of flow maximization, the dual LP is in the form of cut minimization. In an optimal solution, each dual variable in vectors x, y and z is valued between 0 and 1. In the dual constraints, (4.8) distributes weights among the cuts between S and each T_i. (4.6) and (4.7) require each cut y_i to be a valid cut, except that an edge in the cut will now be cut to percentage z_i, rather than 100% as in the minimum cut LP. Then the cut values of a link in the k different cuts are added up in (4.5). If the summations in two directions differ, the larger one is taken to be the cut value for the undirected link.

The variable-constraint correspondence in the primal and dual LPs is given in the table below. It will later assist us to decide which constraints to relax in Sec. V and Sec. VI.

primal	(4.1)	(4.2)	(4.3)	(4.4)	c	$f(\vec{uv})$	$f(\vec{T_iS})$	χ
dual	z	y	p	x	(4.5)	(4.6)	(4.7)	(4.8)

D. Correctness of the cFlow LPs

The primal and the dual *cFlow* LPs are computationally equivalent to each other, with their optimal values being identical. We now prove their correctness by showing that an optimal solution of the primal LP yields the maximum multicast rate and the corresponding flow routing scheme.

Theorem 2. For an undirected data network with a single multicast session, the maximum end-to-end throughput χ 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 $O(km)$. 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 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:

$$\chi = \max_{o \in O}[\min_i(\text{maximum } S \rightarrow T_i \text{ flow rate})],$$

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 [1], [2] 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 the minimum of the maximum source to receiver flow rates. 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 the capacity on the same link, the conflict can always be resolved, provided that network coding is applied appropriately [1], [2].

The number of variables in the primal program and the number of constraints in the dual program are $(2k+1)m+k+1$. The number of constraints in the primal program and the number of variables in the dual program are $(2k+1)m+kn+k$. Both are on the order of $O(km)$.

□

The optimal routing strategy computed by the *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 [13], [24]. Therefore, we have the following corollary of Theorem 2:

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.

V. Multicast rate feasibility: the necessary and sufficient condition

We now apply Lagrangian relaxation on the primal LP to derive the necessary and sufficient condition for multicast rate feasibility in undirected networks. We explain how it generalizes the conditions in unicast and broadcast cases, and provide an interpretation from the perspective of bandwidth efficiency.

A. The condition as a theorem

Theorem 3. A multicast rate χ is feasible in an undirected network G, if and only if for every link distance function $x \in \mathcal{Q}_+^E$,

$$\frac{|G|_x}{Min_{\chi(f)=1}|f|_x} \geq \chi.$$

In the theorem above, $|G|_x$ denotes the size of the network under distance vector x, *i.e.*, $|G|_x = \sum_{uv} C(uv)x(uv)$. $f \in \mathcal{Q}_+^A$ denotes a multicast topology, or a flow routing scheme; and $|f|_x = \sum_{\overrightarrow{uv}} f(\overrightarrow{uv})x(uv)$ is the size of the multicast topology, under distance vector x. $\text{Min}_{\chi(f)=1}|f|_x$ denotes the size of the minimum multicast topology that achieves unit multicast rate. Note that a multicast topology is not necessarily a multicast tree — for instance, the second multicast transmission in Fig. 1 constitutes a counter example.

B. The proof of correctness

Proof of Theorem 3: Consider the primal *cFlow* LP given in Sec. IV-B. We now formulate its Lagrangian dual by relaxing the undirected link capacity constraints (4.4), and introducing the corresponding prices into the objective function, which becomes:

$$\chi - \sum_{uv} x(uv)\Delta(uv).$$

In the modified objective function above, $\Delta(uv) = c(\overrightarrow{uv}) + c(\overrightarrow{vu}) - C(uv)$ denotes the amount of capacity over-use at link uv, and $x(uv)$ is the Lagrangian multiplier acting as the unit price charged for the capacity over-use. At this point, the primal *cFlow* LP is transformed into the Lagrangian subproblem:

$$L(x) = \text{Max}_{\mathcal{P}_1}[\chi - \sum_{uv} x(uv)\Delta(uv)],$$

with P being the following polytope:

$$\mathcal{P}_1 : \begin{cases} \chi \le f_i(\overrightarrow{T_i S}) & \forall i \\ f_i(\overrightarrow{uv}) \le c(\overrightarrow{uv}) & \forall i, \forall\, \overrightarrow{uv} \neq \overrightarrow{T_i S} \\ \sum_{v\in N(u)} f_i(\overrightarrow{uv}) = \sum_{v\in N(u)} f_i(\overrightarrow{vu}) & \forall i, \forall u \\ c(\overrightarrow{uv}), f_i(\overrightarrow{uv}), \chi \ge 0 & \forall i, \forall\, \overrightarrow{uv} \end{cases}$$

The Lagrangian dual problem is then:

$$\begin{array}{ll} \text{Minimize} & L(x) \\ \text{Subject to:} & x \ge 0 \end{array}$$

The Lagrangian duality theorem assures that each feasible value of $L(x)$ is an upper-bound for a feasible multicast rate χ. Furthermore, this bound is tight in the sense that the minimum value of $L(x)$ exactly matches the maximum achievable rate χ, *i.e.*, the optimal objective values of the primal LP and the Lagrangian dual are equal. Consequently, the maximum multicast rate χ^* can be computed as:

$$\chi^* = \text{Min}_{x\ge 0}\{\text{Max}_{\mathcal{P}_1}[\chi - \sum_{uv} x(uv)\Delta(uv)]\}$$

We now perform manipulations on the expression of χ^*, and provide justifications for each step.

$$\begin{aligned}
& \chi^* \\
=_1 \quad & \text{Min}_{x\ge 0}\{\text{Max}_{\mathcal{P}_1}[\chi - \textstyle\sum_{uv} x(uv)\Delta(uv)]\} \\
=_2 \quad & \text{Min}_{x\ge 0}\{\text{Max}_{\mathcal{P}_1}[\chi - \textstyle\sum_{\overrightarrow{uv}} x(uv)c(\overrightarrow{uv}) + \sum_{uv} x(uv)C(uv)]\} \\
=_3 \quad & \text{Min}_{x\ge 0}\{\text{Max}_{\mathcal{P}_1}[\chi - |f|_x + |G|_x]\} \\
=_4 \quad & \text{Min}_{x\ge 0, \text{Min}_{\chi(f)=1}|f|_x \ge 1}\{\text{Max}_{\mathcal{P}_1}[\chi - |f|_x + |G|_x]\} \\
=_5 \quad & \text{Min}_{x\ge 0, \text{Min}_{\chi(f)=1}|f|_x \ge 1}|G|_x \\
=_6 \quad & \text{Min}_{x\ge 0, \text{Min}_{\chi(f)=1}|f|_x = 1}|G|_x \\
=_7 \quad & \text{Min}_{x\ge 0}\frac{|G|_x}{\text{Min}_{\chi(f)=1}|f|_x}
\end{aligned}$$

In the derivations above, $=_1$ holds due to Lagrangian duality, as discussed early. $=_2$ and $=_3$ are due to definitions. $=_4$ is due to dual feasibility. The inner maximization subproblem is unbounded in cases where $\text{Min}_{\chi(f)=1}|f|_x < 1$ — one may scale up flows in f to arbitrarily large, and hence scale up the difference between χ and $|f|_x$ to arbitrarily large. $=_5$ is due to the fact that when $\text{Min}_{\chi(f)=1}|f|_x \ge 1$, we have $\chi - |f|_x \le 0$, and $\text{Max}_P[\chi - |f|_x + |G|_x] = |G|_x$. $=_6$ is due to the observation that for every x where $\text{Min}_{\chi(f)=1}|f|_x > 1$, there exists another vector $x' = x/\text{Min}_{\chi(f)=1}|f|_x$, such that $\text{Min}_{\chi(f)=1}|f|_{x'} = 1$, and $|G|_{x'} < |G|_x$. Finally, $=_7$ is due to the fact that if we scale link distances in x proportionally, the ratio $|G|_x/|f|_x$ remains at the same value.

Now we can claim $\chi^* = \text{Min}_{x\ge 0}\frac{|G|_x}{\text{Min}_{\chi(f)=1}|f|_x}$, and that concludes the proof of Theorem 3. □

C. Interpretation and discussions

Comparison with unicast and broadcast cases

A unicast is an one-to-one data transmission, and a broadcast is an one-to-all data transmission. It is known that for unicast or broadcast, encodability does not make a difference in the maximum achievable transmission rate [21]. Therefore, each atomic unicast topology is a path, and each atomic broadcast topology is a spanning tree. The maximum unicast rate problem is equivalent to the path packing or maximum flow problem, and the maximum broadcast rate problem is equivalent to the spanning tree packing problem. For unicast rate feasibility, the max-flow min-cut theorem constitutes an elegant necessary and sufficient condition. For broadcast rate feasibility, Tutte-Nash-Williams' theorem takes the role [25], [26]: *A capacitied network G contains χ pair-wise capacity-disjoint unit spanning trees, if and only if for every partition that separates the network into k components, the total cross-component link capacity is at least $(k-1)\chi$.*

Unicast and broadcast are special cases of multicast, with the number of receivers being 1 and n, respectively. Consequently, Theorem 3 is a generalization of both the max-flow min-cut theorem and Tutte-Nash-Williams' theorem. For any given cut (vertex partition) of the network, we can assign a distance 1 to each link in the cut (partition), and a distance 0 to all the other links. Then the condition in Theorem 3 implies the cut condition (the partition connectivity condition) in the

max-flow min-cut theorem (Tutte-Nash-Williams' theorem). The reverse implications are also true due to the validity of Theorem 3 and the two special case theorems.

A bandwidth efficiency perspective

Since the total bandwidth capacity of a network is fixed, the achievable multicast rate closely depends on the bandwidth efficiency of the multicast transmission, *i.e.*, how much bandwidth consumption is necessary to achieve a unit end-to-end throughput. Generally speaking, the higher the bandwidth efficiency, the higher the achievable multicast rate. Theorem 3 essentially claims that these two quantities are exactly proportional to each other, once we account for the fact that prolonging or shrinking an internal branch without changing its capacity does not affect the achievable multicast rate. We now reformulate Theorem 3 towards this direction, after giving two definitions. A *link contraction* means replacing a 2-hop internal path u-z-v (*internal* means degree of z is 2) with a link uv, and set $C(uv) = \min\{C(uz), C(zv)\}$. *Link expansion* is the inverse operation for link contraction, where a link uv is replaced with a 2-hop path u-z-v, with $C(uz) = C(zv) = C(uv)$.

Theorem 3.a. For a multicast connection in an undirected network G, a sequence of link contraction and link expansion operations can be applied on G, after which the maximum multicast rate equals the bandwidth capacity of the network divided by the minimum bandwidth consumption required for multicasting one bit information.

The following two facts may help establish the connection between Theorem 3 and Theorem 3.a. First, it is always possible to apply link expansion and contraction operations such that the topological distance between every pair of nodes uv is proportional to its assigned distance $x(uv)$. Second, the "if and only if" relation in Theorem 3 assures that there always exists a distance function that makes the equality hold in Theorem 3. Theorem 3.a shows that a multicast network can be manipulated using link expansion and contraction operations to a stage where (a) it is possible to utilize only the most bandwidth efficient multicast topologies to achieve the max throughput, and (b) the bandwidth capacity of the network is entirely utilized in the transmission scheme achieving the max throughput.

VI. Maximum multicast rate: the subgradient algorithm

Previous experiences show that for network flow type problems with extra side constraints, *e.g.*, the multicommodity flow problem, the performance of general linear programming techniques are often below acceptable levels, when the size of the problem is relatively large. For the multicast rate problem in particular, we have experimented with both the simplex method and the primal-dual interior-point method, as implemented in `glpk 4.4` [27]. We apply both methods to solve the primal LP as a black-box, on networks and multicast groups with various sizes. Our findings show that, on a typical Pentium IV computing platform, the interior-point method may handle networks with a few thousand links within a reasonable amount of time (on the order of seconds), as long as the multicast group is small ($k \leq 5$). For networks that are larger, or for a broadcast network with a few hundred nodes and less than one thousand links, the computation easily takes hours. The performance of the simplex method is constantly worse than that of the interior-point method.

Another critical drawback of applying general linear programming methods is that they are inherently centralized, requiring global information to be collected to one central point of computation. The solution we construct in this section resolves both problems. It decomposes the maximum multicast rate computation into a sequence of max-flow/min-cut computations, for which very efficient algorithms exist and can be applied. It also allows the computation to be distributed onto each node in the network, where only local information is collected.

To construct a subgradient solution for the maximum multicast rate problem, we have the choices of applying Lagrangian relaxation on either constraints in the primal program (dual subgradient), or constraints in the dual program (primal subgradient). We have decided to take the later approach, due to the following facts. First, dual subgradient methods do not always yield optimal primal solutions, which contain the optimal routing information we need. Second, as we will show, our primal subgradient algorithm decomposes the entire problem into a sequence of max-flow/min-cut computations, and allows appealing combinatorial interpretations. We now present the primal subgradient solution in three steps: the dualization strategy, subgradient iterations, and maximum rate computation.

A. The dualization strategy

Consider the dual *cFlow* LP given in Sec.IV-C for the maximum multicast rate problem. We choose to relax constraint group (4.5), which corresponds to primal variables $c(\vec{uv})$. Recall that $c(\vec{uv})$ specifies the capacity of each directed link, and therefore determines an orientation of the original undirected network. The objective function is modified to:

$$\begin{aligned} & \textstyle\sum_{uv} C(uv)x(uv) + \sum_{\vec{uv}} c(\vec{uv})(\sum_i y_i(\vec{uv}) - x(uv)) \\ = \; & \textstyle\sum_{uv} x(uv)(C(uv) - c(\vec{uv}) - c(\vec{vu})) + \sum_{\vec{uv}}(c(\vec{uv}) \sum_i y_i(\vec{uv})) \\ = \; & \textstyle\sum_i \sum_{\vec{uv}} c(\vec{uv}) y_i(\vec{uv}) - \sum_{uv} x(uv)\Delta(uv) \end{aligned}$$

Note when $\Delta(uv) > 0$ for any uv, the modified objective function does not have a lower bound, with $x(uv)$ freely chosen from $[0, \infty)$. Therefore dual feasibility requires $\Delta \leq 0$, *i.e.*, $c(\vec{uv}) + c(\vec{vu}) \leq C(uv)$, $\forall uv$.

The Lagrangian dual we obtain is then:

$$\begin{array}{ll} \text{Maximize} & L(c) \\ \text{Subject to:} & \\ \left\{ \begin{array}{ll} c(\vec{uv}) + c(\vec{vu}) \leq C(uv) & \forall uv \\ c(\vec{uv}) \geq 0 & \forall\, \vec{uv} \end{array} \right. & \end{array}$$

where

$$L(c) = \text{Min}_{\mathcal{P}_2} \sum_i \sum_{\vec{uv}} c(\vec{uv}) y_i(\vec{uv}) \qquad (6.1)$$

with $\mathcal{P}_2$ being the polytope:

$$\mathcal{P}_2 : \begin{cases} y_i(\vec{uv}) + p_i(v) \geq p_i(u) & \forall i, \forall\ \vec{uv} \neq \vec{T_i S} \\ p_i(T_i) - p_i(S) \geq z_i & \forall i \\ \sum_i z_i \geq 1 & \\ y_i(\vec{uv}), z_i \geq 0 & \forall i, \forall\ \vec{uv} \end{cases}$$

Two critical observations justify our choice of the dualization strategy above. First, the price variables introduced through relaxation and optimized through subgradient iterations, c, is exactly the orientation of the network, the optimal values of which are essential to decide the maximum multicast rate and the optimal routing strategy. Second, the minimization subproblem (6.1) is separable, and may be decomposed into k min-cut computations. We shall come back to these two facts in the presentation of the subgradient iterations and the maximum rate computation, respectively.

B. Subgradient iterations

Choosing the initial primal solution

To start the subgradient iterations, we need a valid set of initial values for $c(\vec{uv})$, *i.e.*, an initial orientation of the multicast network. A promising choice is to set $c[0](\vec{uv}) = \frac{1}{2}C(uv),\ \forall\ \vec{uv}$. Note that any transmission that is feasible in the undirected network is feasible in the network obtained by scaling up link capacities in the balanced orientation by a factor of 2. Therefore the balanced orientation is 2-competitive, *i.e.*, if the maximum multicast rate in an optimal orientation is χ^*, then the balanced orientation may support a rate of at least $\frac{1}{2}\chi^*$.

Updating dual variables

During each round k, given current values of $c[k]$, we solve subproblem (6.1) to obtain new dual values in $y[k]$. As previously mentioned, this subproblem has a nice separable structure, in the form of a weighted minimum cut computation. Note that when $\sum_i z_i = 1$,

$$\begin{aligned} L(c) &= \text{Min}_{\mathcal{P}_2} \textstyle\sum_i \sum_{\vec{uv}} c(\vec{uv}) y_i(\vec{uv}) \\ &= \text{Min}_i [\text{Min}_{\mathcal{P}_3} \textstyle\sum_{\vec{uv}} c(\vec{uv}) y_i(\vec{uv})] \end{aligned}$$

where $\mathcal{P}_3$ is the standard cut polytope:

$$\mathcal{P}_3 : \begin{cases} y(\vec{uv}) + p(v) \geq p(u) & \forall\ \vec{uv} \neq \vec{T_i S} \\ p(T_i) - p(S) \geq 1 & \\ y(\vec{uv}) \geq 0 & \forall\ \vec{uv} \end{cases}$$

i.e., the weighted minimum cut is equal to the minimum cut when all weights sum to 1. Further note that $\sum_i z_i = 1$ must be satisfied in any optimal solution, since dual complementary slackness conditions require $\chi(\sum_i z_i - 1) = 0$. Therefore, for our specific problem, we can compute $y[k]$ by first computing k minimum cuts, *i.e.*, one minimum cut between the sender S and each receiver T_i:

$$y_i^* = \text{argmin}_{y \in \mathcal{P}_3} \sum_{\vec{uv}} c[k](\vec{uv}) y(\vec{uv})$$

Then let $j = \text{argmin}_i \sum_{\vec{uv}} c[k](\vec{uv}) y_i^*(\vec{uv})$, we update y as follows:

$$y_j[k] = y_j^*, \textit{and}$$
$$y_i[k] = 0, \forall i \neq j.$$

Updating primal variables

Primal variables in the orientation c are updated in two steps. First, we compute a new orientation vector c' as follows:

$$c' = c[k] + \theta[k] \sum_i y_i[k] \qquad (6.2)$$

where θ is a prescribed sequence of step sizes. The new vector c' is not feasible in general. Therefore we need to project it into the feasible simplex, to obtain a valid new vector for updating c. One possible way of projection is to take a feasible point that is nearest to c':

$$c[k+1] = \text{argmin}_{c \geq 0, \Delta \leq 0} ||c - c'|| \qquad (6.3)$$

Here $||l||$ denotes the geometrical length of a vector l, *i.e.*, for $l = (l_1, \ldots, l_h)$, $||l|| = (\sum_{i=1}^h l_i^2)^{1/2}$. Another simpler way of projection, is to normalize c' according to:

$$c[k+1](\vec{uv}) = \begin{cases} c'(\vec{uv}) & \Delta'(uv) \leq 0 \\ \frac{c'(\vec{uv})}{c'(\vec{uv}) + c'(\vec{vu})} C(uv) & \Delta'(uv) > 0 \end{cases} \qquad (6.4)$$

where $\Delta'(uv) = c'(\vec{uv}) + c'(\vec{vu}) - C(uv)$. After both primal and dual variables are updated, the next iteration starts.

Step size selection and convergence

Step size rules play an important role in subgradient optimization. It governs both the ultimate convergence in theory, and the speed of convergence to the optimal solution in practice. Large step sizes may be unstable, while small step sizes lead to slow convergence speed. Therefore it is common practice to use varying step sizes: take a small number of large steps to reach the proximity of the optimal solution, then switch to small steps to avoid overhitting. In our case, where the original program is linear, designing step sizes that satisfy the following conditions will guarantee convergence:

$$\theta[k] \geq 0, \qquad \lim_{k \to \infty} \theta[k] = 0, \quad \text{and} \quad \sum_{k=1}^{\infty} \theta[k] = \infty$$

A simple sequence that satisfies the conditions above is $\theta[k] = a/(bk + c)$, for some positive constants a, b and c. Below we give an example to illustrate the input, output, and convergence of the proposed algorithm.

In the example shown in Fig. 2, S is the multicast sender, T_1 and T_2 are the multicast receivers. The maximum multicast rate possible is 13.5. The rate computed by the subgradient algorithm converges to range $[13.4, 13.5]$ within 100 iterations. The network in this example is actually among the most adverse to our algorithm, in that network flows towards different

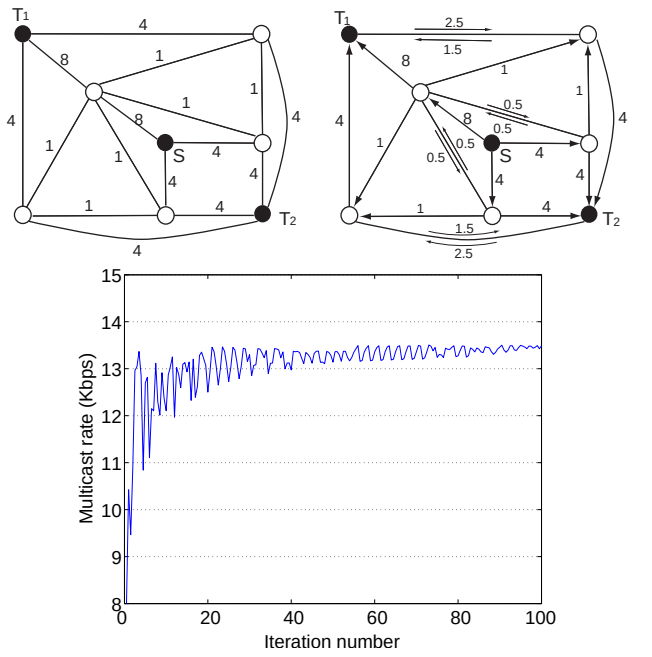


Fig. 2. A test case of the subgradient algorithm: input network, output orientation, and convergence sequence.

receivers constantly compete for link bandwidth in opposite directions. As later shown in Sec. VIII, the convergence speed is usually much faster for randomly generated multicast networks.

Algorithm interpretation

We now take a retrospect of the subgradient algorithm just presented, and show that it has a very appealing combinatorial interpretation. First, the algorithm takes a guessed orientation of the network as a starting point. Then during each iteration, it updates the orientation according to (6.2), (6.3) and (6.4). In (6.2), larger values for $\sum_i y_i[k]$ leads to larger values for c', which in turn leads to larger values for $c[k+1]$ in (6.3)(6.4). Note that non-zero values for $y_i(\overrightarrow{uv})$ means the link uv is in the S-T_i min-cut, and is therefore the "bottleneck" for the $S \rightarrow T_i$ transmission. From the flow perspective, non-zero values of $y(\overrightarrow{uv})$ means $f(\overrightarrow{uv}) = c(\overrightarrow{uv})$, since dual complementary slackness conditions require $y_i(\overrightarrow{uv})(f(\overrightarrow{uv}) - c(\overrightarrow{uv})) = 0$. Therefore links with non-zero y_i values are saturated links in the $S \rightarrow T_i$ max-flow. We conclude that the new capacity allocation in (6.2), (6.3) and (6.4) favors links with larger $\sum_i y_i[k]$ values, which are links that are more saturated.

Therefore, during each iteration of orientation refinement, the algorithm computes the max-flow/min-cut from the sender to each receiver, and increases the capacity share for more saturated links, while decreases the capacity share for under-utilized links. This has been summarized in Table I.

C. Computing the maximum rate

When the subgradient algorithm converges, it yields optimal primal values in c, but not necessarily optimal dual values in y — the dual values upon convergence may not even be feasible. Although there exist convex combination techniques to recover these optimal dual values [16], [28], it is not necessary in our solution. We can directly recover the whole set of optimal primal values from optimal values in c.

TABLE I

MAXIMUM MULTICAST RATE: SOLUTION SUMMARY

(1) *Choose initial orientation (*e.g., *balanced orientation)*

(2) Repeat
Compute $S \rightarrow T_i$ max-flow, $\forall i$
Refine orientation:
increase bandwidth share for saturated links
decrease bandwidth share for under-utilized links
Until convergence
→ optimal orientation obtained

(3) *Compute $S \rightarrow T_i$ max-flow, $\forall i$*
→ optimal multicast rate and routing strategy obtained

(4) *Randomized code assignment*
→ complete transmission strategy obtained

Recall that a feasible vector c specifies an orientation of the undirected network. Therefore optimal values of c give an optimal orientation. Once the orientation is determined, the undirected maximum multicast rate problem boils down to a directed one, *i.e.*, computing the maximum multicast rate in a directed network. By the result on directed multicast rate feasibility proven by Ahlswede *et al.* and Koetter *et al.*, this can be accomplished by invoking a maximum flow computation from sender S to each of the k receivers T_i. Let f_i^* denote the resulting $S \rightarrow T_i$ flow vector, and $|f_i^*|$ denote the corresponding flow rate. Then our final solution to the maximum multicast rate problem is:

- maximum multicast rate: $\chi = \min_i |f_i|$
- optimal routing strategy of information flows: f^*, where $f^*(\overrightarrow{uv}) = \max_i f_i(\overrightarrow{uv})$, $\forall\, \overrightarrow{uv} \in A$

As an illustration, the two network flows computed in the previous example are shown in Fig. 3.

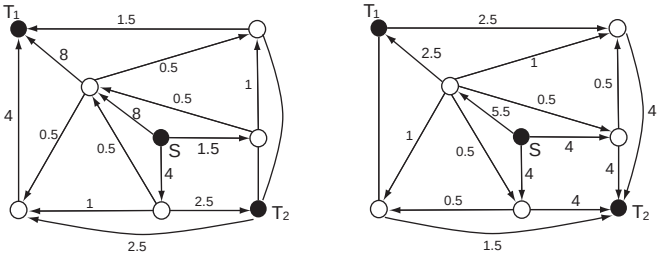


Fig. 3. Output network flow to each multicast receiver.

D. Discussions on distributed implementation

Beside simplicity and efficiency, the potential for distributed implementation has been another objective during our design of the subgradient algorithm. After all, decentralized and scalable implementations are always preferred for real-world protocols. we now take a step-by-step examination of the

entire solution, and discuss how each step can be transferred into distributed, pure local computations, where each node maintains only local information about its incident links and one-hop neighbors.

In the initialization phase of the dual subgradient algorithm, it is sufficient to have each node u compute its local orientation, by setting $c(\vec{uv}) = c(\vec{vu}) = \frac{1}{2}C(uv)$, for each of its incident link uv.

Primal variable update is achieved through pure local computation, since each node can update the capacity of an incident directed link $\vec{uv}$ according to (6.2), based on current values of local variables $c[k](\vec{uv})$, $\theta[k]$ and $y[k](\vec{uv})$.

Most computation in the subgradient algorithm is performed in dual variable updates, and in the final maximum flow rate computation. Each of these steps translates into k max-flow/min-cut computations. As previously mentioned, various efficient algorithms exist for the classical max-flow/min-cut problem, some of which permits natural distributed implementations, such as the push-relabel algorithm [14] and the ϵ-relaxation algorithm [15]. For example, throughout the execution of the distributed version of the push-relabel algorithm, each node exchanges messages with its direct neighbors only, and maintains information about capacities and flow rates on its incident links, plus distance labels of its neighbors and its own.

So far, we have shown that our algorithm for computing the optimal multicast routing strategy can be implemented in a distributed fashion. In order to utilize such optimal routing strategy in data transmission, we need to further decide how each node linearly combines its incoming information flows to form its outgoing information flows. A simple distributed solution to this code assignment problem is randomized coding [29], in which each node just locally generates a random code matrix, without any message-passing required at all. With mild assumptions on the size of the base field for coding operations, the chance of generating a conflict is negligibly small [29].

VII. Achieving Optimal Throughput in Undirected Data Networks: More General Cases

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, (3) to the model of overlay networks, and (4) to the model of wireless ad hoc 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 [21] to be the same as both the spanning tree packing number and the network strength.

Traditionally, the three equal quantities above have been computed from either the perspective of network strength or spanning tree packing. Cunningham [20] first gave a combinatorial algorithm that computes the network strength through $O(n^3)$ max-flow computations. Barahona [30] later improved the time complexity to $O(n^2)$ max-flow computations[1]. 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, our *cFlow* approach provides an effective alternative, which is easy to understand, allows fully-distributed implementation, and still achieves high time efficiency.

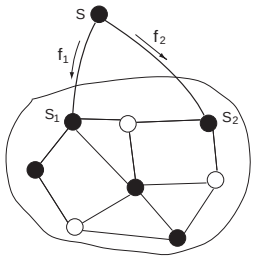


Fig. 4. Transforming group communication into multicast transmission.

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 [2] 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. 4, 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.

[1]A representative max-flow algorithm, such as the FIFO-preflow-push algorithm, runs in $O(n^3)$ time itself.

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* [1], *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. So far we are not aware of any efficient algorithm that can compute max throughput with inter-session coding, or any effective characterization of rate feasibility with inter-session coding. 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 below 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 S_i, a number of receivers $T_{i_1} \dots T_{i_{k_i}}$, 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 is the common weighted throughput for all the multicast sessions, and the target of the *mFlow* LP is to maximize χ.

$$\begin{array}{ll}\text{Maximize} & \chi \\ \text{Subject to:} & \end{array}$$

$$\begin{cases} \chi \le f_{i_j}(\vec{T_{i_j}S_i})/w_i & \forall i, \forall j \\ f_{i_j}(\vec{uv}) \le f_i(\vec{uv}) & \forall i, \forall j, \forall\, \vec{uv} \\ \sum_{v \in N(u)} (f_{i_j}(\vec{uv}) - f_{i_j}(\vec{vu})) = 0 & \forall i, \forall j, \forall u \\ \sum_i f_i(\vec{uv}) \le c(\vec{uv}) & \forall\, \vec{uv} \neq \vec{T_{i_j}S_i} \\ c(\vec{uv}) + c(\vec{vu}) \le C(uv) & \forall uv \neq T_{i_j}S_i \end{cases}$$

$$c(\vec{uv}), f_{i_j}(\vec{uv}), f_i(\vec{uv}), \chi \ge 0 \qquad \forall i, \forall j, \forall\, \vec{uv}$$

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 $f_i(\vec{uv}) = \max_{j \in [1..k_i]} f_{i_j}(\vec{uv})$. The $\max$ function is not linear, so this constraint is relaxed to $f_i(\vec{uv}) \ge f_{i_j}(\vec{uv}), \forall j$.

Theorem 4. 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 2, 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 $O(kms)$. □

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 $N = \{G(V, E), C, M = \{S, T_1, \dots, T_k\} \subseteq H \subseteq V\}$ 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.* [31] 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 [31] 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 both an *underlay* and an *overlay* level. The underlay level corresponds to the physical network topology, and has multicommodity flows g_{uv} connecting each pair of end hosts u and v, via only routers as intermediate nodes. The overlay level is conceptual, and contains end hosts fully connected as a complete graph. The overlay link $\vec{uv}$ has a capacity that is equal to the underlay flow rate $g_{uv}(\vec{vu})$. 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.

$$\begin{array}{ll}\text{Maximize} & \chi \\ \text{Subject to:} & \end{array}$$

$$\begin{cases} \chi \le f_i(\vec{T_iS}) & \forall i \\ \sum_{v \in H-u} (f_i(\vec{uv}) - f_i(\vec{vu})) = 0 & \forall i, \forall u \in H \\ f_i(\vec{uv}) \le g_{uv}(\vec{vu}) & \forall i, \forall\, \vec{uv} \in H \times H - \vec{T_iS} \\ \sum_{v \in N(u)} (g_{pq}(\vec{uv}) - g_{pq}(\vec{vu})) = 0 & \forall p, q \in H, \forall u \\ \sum_{pq} g_{pq}(\vec{uv}) \le c(\vec{uv}), & \forall p, q \in H, \forall\, \vec{uv} \neq \vec{qp} \\ c(\vec{uv}) + c(\vec{vv}) \le C(uv) & \forall uv \end{cases}$$

$$c(\vec{uv}), f_i(\vec{uv}), g_{pq}(\vec{uv}), \chi \ge 0 \qquad \forall i, \forall p, q \in H, \forall\, \vec{uv}$$

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 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 2. Furthermore, inspection on the variables and constraints in the *oFlow* LP reveals that, the number of both are on the order of $O(|H|^2 m)$. □

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

D. The case of wireless ad hoc networks

A wireless ad hoc network is a multi-hop wireless network consisting of nodes communicating via antennas. In contrast to single-hop wireless wireless networks, nodes in an ad hoc network relay data packets for each other to realize multi-hop routing. Here we consider two different transmission technologies in the underlying physical and media access layers: omni-directional antennas and beam-forming antennas.

Omni-directional antennas and the physical model

With omni-directional antennas, signal propagation is broadcast in nature, and nearby transmissions interfere with each other. Traditionally, there have been two major directions in modelling such wireless medium contention, the *logical model* and the *physical model*. Ad hoc routing research uses the logical model and view physical layer interactions as a black-box: a local transmission is successful if the sender and the receiver are within the effective transmission range, and there is no other active transmissions within the interference range. Under such a model, even if the routing and coding schemes have been decided, determining the optimal temporal schedule of per-hop transmissions to achieve the maximum throughput is equivalent to the graph coloring problem, and is therefore NP-hard.

On the other hand, the capacity of a link is determined by its SNR (signal-to-noise ratio) in the physical model:

$$c(\vec{uv}) \le \log(1 + SNR(\vec{uv})), \quad \text{where}$$

$$SNR(\vec{uv}) = \frac{\mu(\vec{uv})}{\sum_{r \neq u,v} \mu(\vec{rv}) + b}, \quad \mu(\vec{uv}) = ||uv||^{-\alpha} P(u) \tag{7.1}$$

In (7.1), $\mu(\vec{uv})$ is the strength of the signal from u perceived at v, $P(u)$ denotes the adjustable power level at node u, α is a constant usually between 2 and 4, and b is background noise.

Under such a physical model, the maximum multicast throughput problem can be formulated as:

$$\begin{array}{ll} \text{Maximize} & \chi \\ \text{Subject to:} & \\ & \begin{cases} \chi \in \mathcal{C}(c) \\ c \in \mathcal{C}(P) \end{cases} \end{array}$$

where $\mathcal{C}(c)$ is equivalent to polytope $\mathcal{P}_1$ given link capacity vector c, as defined in Sec.V-B, and $\mathcal{C}(P)$ is the feasible region for the capacity vector c, as defined in (7.1). A critical observation about this non-linear optimization problem is that, by relaxing $f_i \le c$ in $\mathcal{P}$, we can decompose it into two sub-problems without compromising optimality. Due to space constraints, we only provide a brief sketch of the solution method, more details can be found in our technical report [32]. A similar approach was also taken in [33], where near-optimal throughput is targeted.

After relaxing $f_i \le c$ in $\mathcal{P}_1$ with price x, we can then iteratively solve two sub-problems at the network layer and the physical layer respectively. At the network layer, we have the flow routing sub-problem, maximize $\chi - x^T \sum_i (c - f_i)$ subject to flow conservation constraints only, which can be solved as a network flow problem since x and c are constant vectors here. At the physical layer, we have the power allocation sub-problem, maximize $x^T c$ subject to $c \in \mathcal{C}(P)$. While $\mathcal{C}(P)$ is in general not convex, it can be transformed into a convex one via time sharing techniques such as OFDM. After solving both sub-problems, the price vector x is updated according to prescribed step sizes and the next iteration starts, until convergence.

We point out, however, that the solutions presented above do not take into account the fact that flows going out from the same node may carry the same information, in which case the broadcast nature of omni-directional antennas can be exploited to further increase the throughput. Consequently a lower bound of the optimal throughput is obtained. It is our ongoing research to appropriately incorporate the wireless broadcast advantage into the optimization framework for ad hoc networks. We refer the readers to heuristic for incorporating the broadcast advantage discussed by Lun *et al.* in [34], and to a related study of achieving minimum energy consumption of wireless transmissions by Wu *et al.* in [35].

Beam-forming antennas and the node-centric approach

While omni-directional antenna leads to intense medium contention, beam-forming antennas direct energy radiation towards the intended receiver only, no node in other directions will be affected by the transmission if beam-forming is ideal. Therefore the radio capacity at each node is shared among transmissions to and from it, and flow scheduling becomes node-centric. We show that such a node-centric optimization problem, as formulated below, can also be solved efficiently via the subgradient approach.

$$\text{Maximize} \qquad \chi$$

Subject to:

$$\begin{cases} \chi \le f_i(\vec{T_i}S) & \forall i \quad (1) \\ \sum_{v\in N(u)}(f(\vec{uv}) + f(\vec{vu})) \le C(u) & \forall u \quad (2) \\ \sum_{v\in N(u)}(f_i(\vec{uv}) - f_i(\vec{vu})) = 0 & \forall i, \forall u \quad (3) \\ f_i(\vec{uv}) \le f(\vec{uv}) & \forall i, \forall\, \vec{uv} \ne \vec{T_i}S \quad (4) \end{cases}$$

$$f(\vec{uv}), f_i(\vec{uv}), \chi \ge 0 \qquad \forall i, \forall\, \vec{uv}$$

In the LP above, $C \in \mathcal{Q}_+^V$ denotes total radio capacity available at each node. By relaxing $\sum_{v\in N(u)}(f(\vec{uv}) + f(\vec{vu})) \le C(u)$, we obtain the equivalent Lagrangian dual problem:

Maximize $L(f)$

Subject to:

$$\begin{cases} \sum_{v\in N(u)}(f(\vec{uv}) + f(\vec{vu})) \le C(u) & \forall u \\ f(\vec{uv}) \ge 0 & \forall\, \vec{uv} \end{cases}$$

where

$$L(f) = \text{Min}_{\mathcal{P}_2} \sum_i \sum_{\vec{uv}} f(\vec{uv}) y_i(\vec{uv})$$

and $\mathcal{P}_2$ is the same polytope of y_i as defined in Sec.VI-A. Fixing f, $L(f)$ can be computed in the same way as $L(c)$ in the link-centric case in Sec. VI, by separating the problem into a sequence of min-cut computations. The only difference is in the initialization and update of the primal variables. Here f can be initialized according to an even capacity distribution, by setting $f[0](\vec{uv}) = \min(\frac{C(u)}{2|N(u)|}, \frac{C(v)}{2|N(v)|}), \forall\, \vec{uv}$. Then at the end of each iteration it can be updated based on new values in y_i and prescribed step sizes, and projected such that the total incident flow rate respects the node capacity.

VIII. 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 the LPs and solutions 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 [36], with sizes ranging from 10 to $1,000$ 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?

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 [24], [37]. 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. 1 is $C(3,2)$, and the classic example showing the power of network coding [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
classical	7	3	9	2	1.875	1.067	17
$C(3,2)$	7	4	9	2	1.8	1.111	26
Fig. 2	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

Previous work [24] shows that in directed acyclic networks with integral routing requirement, there exist multicast networks where the coding advantage grows proportionally as $\log(n)$, and is thus not finitely bounded. However, we found the situation is drastically different in undirected networks. In [21], 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 topolo-

gies. 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.

Finally, we point out that the potential for network coding to decrease cost in the min-cost multicast problem is essentially the same as its potential to increase throughput in the max-rate multicast problem [38], and both are equivalent to the integrality gap of a bi-directed relaxation of the minimum Steiner tree problem [39]. The largest value known so far for these three equal quantities is $8/7$ [38], [39]. Our results in Table II have also advanced the previously known largest value in quasi-bipartite networks, which was $10/9$.

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 trade-off 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.

As we may observe from Fig. 5, there exists obvious differences between standard multicast throughput and all unicast throughput, and the differences are more significant in Fig. 5(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

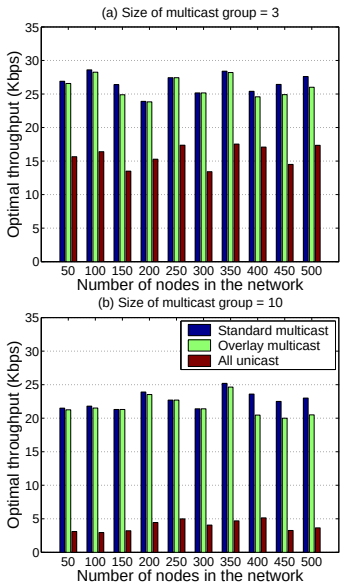


Fig. 5. Achievable optimal throughput using standard multicast, overlay multicast, and all unicast from the sender to all receivers.

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 time-efficient is the subgradient solution?

Fig. 6 shows the convergence speed of the subgradient algorithm on random networks generated by BRITE [36], with network sizes up to 1000 nodes, and with 2, 5, and 10 receivers, respectively. As we can see, the optimal solution is usually approached within 10 iterations, regardless of the network size or the multicast group size. However, the computation time for large networks or for large multicast groups are longer, due to the fact that the time taken by each single max-flow/min-cut computation is roughly proportional to n^3, and that the number of max-flow/min-cut computations in each iteration is proportional to the multicast group size.

We proceed to compare the computation time of the following three approaches for computing maximum multicast rate in networks with various sizes: Steiner tree packing, direct linear program solving, and applying the subgradient algorithm. To make the comparison possible, we further implemented a brute-force algorithm to compute the optimal Steiner tree packing number. The algorithm exhaustively enumerates all distinct Steiner trees connecting the multicast group, then assigns a flow rate to each tree, and maximizes the summation of all tree flow rates with the constraint that, total flow rates of trees using a link uv should not exceed the capacity $C(uv)$.

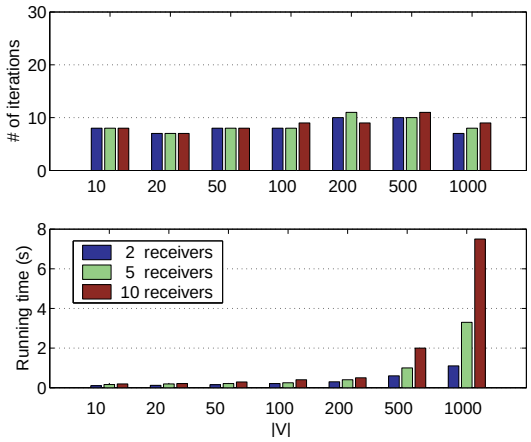


Fig. 6. Convergence speed in random networks.

The comparison result is given in Fig. 7. The multicast group consists of three nodes.

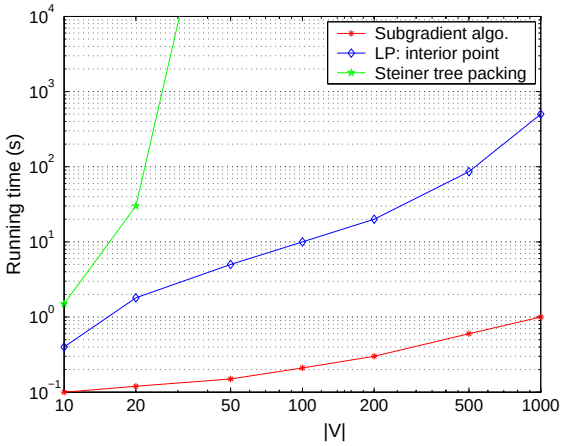


Fig. 7. Computation time comparison: Steiner tree packing, general LP solver, and proposed subgradient solution (axes on log scale).

The scalability of the brute-force Steiner tree packing algorithm is extremely poor. We find that its running time crucially depends on the number of links in the network, in a roughly exponential fashion. It basically fails for networks with more than 50 links, and may take more than a minute for a network with only 11 nodes. Solving the LP with the interior point method may handle small to medium sized networks in a reasonable amount of time (on the order of seconds), but the running time grows rapidly as the network size further grows. On the other hand, the subgradient algorithm may solve networks with one thousand nodes in around 1 second. Another important advantage of the subgradient solution over the direct LP solving method is that, the former is amenable to fully distributed implementations, while the later is inherently centralized. We also point out that the computation time discussed here corresponds to the one-time only session set up delay, and does not apply to successive data packet transmissions.

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. 8(a) shows variations of optimal throughput as the number of nodes in the multicast group increases from three to $\lceil n/2 \rceil$, and then to n (effectively a broadcast session), for various network sizes n. 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 $\lceil n/2 \rceil$, 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.

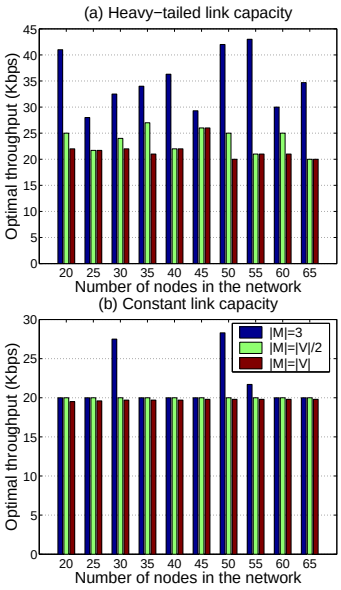


Fig. 8. Variations of optimal throughput due to new nodes joining the multicast session.

We further performed the same tests on power-law network topologies with 10 Kbps constant link bandwidth, and the results are shown in Fig. 8(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. 8(b) correspond to the occasional cases where nodes in the initial multicast group all have relatively high capacities. Both results in Fig. 8(a) and 8(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. VII, makes it feasible to carry out our empirical studies. Fig. 9 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.

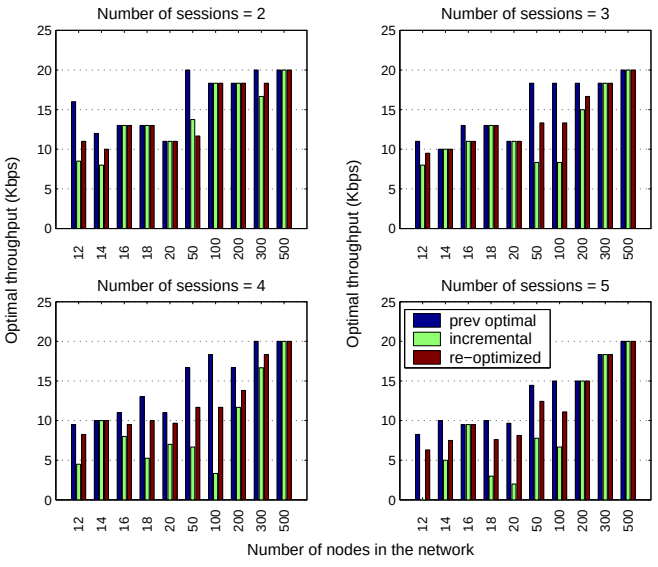


Fig. 9. Throughput variations as a new session is created.

Results in Fig. 9 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. 10 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.

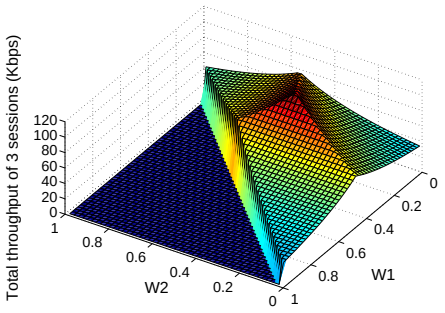


Fig. 10. Total throughput of three multicast sessions, as inter-session fairness requirements change.

Further results in Table III 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 III
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.

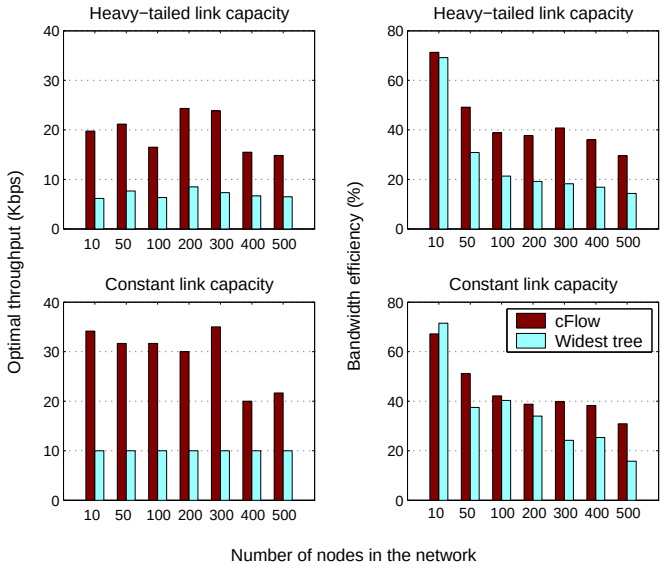


Fig. 11. Achievable throughput and bandwidth efficiency: a comparison between the optimal throughput multicast (*cFlow* LP) and the widest Steiner tree.

In Fig. 11, 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 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.

IX. 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.

ACKNOWLEDGMENT

The authors would like to thank the guest editor, Dr. Ning Cai, and the anonymous referees for valuable insights that helped improve this paper.

REFERENCES

[1] 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.
[2] R. Koetter and M. Médard, "An Algebraic Approach to Network Coding," *IEEE/ACM Transactions on Networking*, vol. 11, no. 5, pp. 782–795, October 2003.
[3] 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.
[4] A. J. Ballardie, P. F. Francis, and J. Crowcroft, "Core Based Trees," in *Proc. of ACM SIGCOMM*, August 1993.
[5] 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.
[6] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable Application Layer Multicast," in *Proc. of ACM SIGCOMM*, August 2002.
[7] 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.
[8] V. Padmanabhan, H. Wang, P. Chou, and K. Sripanidkulchai, "Distributing Streaming Media Content Using Cooperative Networking," in *Proc. of NOSSDAV 2002*, May 2002.
[9] J. Byers and J. Considine, "Informed Content Delivery Across Adaptive Overlay Networks," in *Proc. of ACM SIGCOMM*, August 2002.

[10] 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.

[11] 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.

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

[13] S. Jaggi, P. Sanders, P. A. Chou, M. Effros, S. Egner, K. Jain, and L. Tolhuizen, "Polynomial Time Algorithms for Multicast Network Code Construction," *Submitted to IEEE Transactions on Information Theory*, July 2003.

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

[15] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Prentice Hall, 1989.

[16] D. Lun, N. Ratnakar, R. Koetter, M. Médard, E. Ahmed, and H. Lee, "Achieving Minimum-Cost Multicast: A Decentralized Approach Based on Network Coding," in *Proceedings of IEEE INFOCOM*, 2005.

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

[19] 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.

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

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

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

[23] 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.

[24] 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.

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

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

[27] *GNU Linear Programming Kit*, http://www.gnu.org/ software/glpk/glpk.html.

[28] H. D. Sherali and G. Choi, "Recovery of Primal Solutions When Using Subgradient Optimization Methods to Solve Lagrangian Duals of Linear Programs," *Operations Research Letters*, vol. 19, 1996.

[29] T. Ho, M. Médard, J. Shi, M. Effros, and D. R. Karger, "On Randomized Network Coding," in *41st Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2003.

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

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

[32] J. Yuan, Z. Li, W. Yu, and B. Li, "Achieving Optimal Throughput in Multi-hop Wireless Networks," ECE, University of Toronto, *http://iqua.ece.toronto.edu/∼arcane/ylyl.pdf*, Tech. Rep., 2005.

[33] Y. Wu, P. A. Chou, Q. Zhang, K. Jain, W. Zhu, and S. Kung, "Network Planning in Wireless Ad Hoc Networks: A Cross-Layer Approach," *IEEE Journal on Selected Areas in Communication*, vol. 23, no. 1, January 2005.

[34] D. S. Lun, M. Médard, and R. koetter, "Efficient Operation of Wireless Packet Networks Using Network Coding," in *International Workshop on Convergent Technologies (IWCT)*, 2005.

[35] Y. Wu, P. A. Chou, and S. Kung, "Minimum-Energy Multicast in Mobile Ad Hoc Networks using Network Coding," *to appear in IEEE Trans. on Communications*, 2005.

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

[37] 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.

[38] Z. Li and B. Li, "The Coding Advantage: A Comparison Study," ECE, University of Toronto, *http://iqua.ece.toronto.edu/∼arcane/ll.pdf*, Tech. Rep., 2005.

[39] A. Agarwal and M. Charikar, "On the Advantage of Network Coding for Improving Network Throughput," in *Proceedings of the IEEE Information Theory Workshop*, 2004.

Zongpeng Li is an Assistant Professor at the Department of Computer Science in University of Calgary. He received his B.E. degree in Computer Science and Technology from Tsinghua University in 1999, his M.S. degree in Computer Science from University of Toronto in 2001, and his Ph.D. degree in Electrical and Computer Engineering from University of Toronto in 2005. His research interests are in data networks and distributed algorithms.

Baochun Li received his B.Engr. degree in 1995 from Department of Computer Science and Technology, Tsinghua University, China, and his M.S. and Ph.D. degrees in 1997 and 2000 from the Department of Computer Science, University of Illinois at Urbana-Champaign. Since 2000, he has been with the Department of Electrical and Computer Engineering at the University of Toronto, where he is currently an Associate Professor. He holds the Nortel Networks Junior Chair in Network Architecture and Services since October 2003, and the Bell University Laboratories Endowed Chair in Computer Engineering since August 2005. In 2000, he was the recipient of the IEEE Communications Society Leonard G. Abraham Award in the Field of Communications Systems. His research interests include application-level Quality of Service provisioning, wireless and overlay networks. He is a senior member of IEEE, and a member of ACM.

Lap Chi Lau is a Ph.D. student in the Department of Computer Science at the University of Toronto, under the supervision of Professor Michael Molloy. He obtained an M.Sc. in the same department with Professor Derek Corneil. Previously he received a B.Sc. from the Chinese University of Hong Kong. His current research lies mostly in theoretical computer science, particularly on discrete combinatorial problems and approximation algorithms. For his work on Steiner tree packing, he received the Machtey award for student paper in the IEEE conference FOCS 2004. He is a recipient of the Microsoft Fellowship for 2005-2006.