

Optimal Layered Multicast

AJAY GOPINATHAN and ZONGPENG LI

University of Calgary

Recent advances in network coding research dramatically changed the underlying structure of optimal multicast routing algorithms and made them efficiently computable. While most such algorithm design assume a single file/layer being multicast, layered coding introduces new challenges into the paradigm due to its cumulative decoding nature. Layered coding is designed to handle heterogeneity in receiver capacities, and a node may decode layer k only if it successfully receives all layers in $1..k$. We show that recently proposed optimization models for layered multicast do not correctly address this challenge. We argue that in order to achieve the absolute maximum throughput (or minimum cost), it is necessary to decouple application layer throughput from network layer throughput. In particular, a node should be able to receive a non-consecutive layer or a partial layer even if it cannot decode and utilize it (*e.g.*, for playback in media streaming applications). The rationale is that nodes at critical network locations need to receive data just for helping other peers. We present a mathematical programming model that addresses the above challenges and achieves the absolute optimal performance. Simulation results show considerable throughput gain (cost reduction) compared with previous models, in a broad range of network scenarios. We then provide a formal proof that the layered multicast problem is NP-complete. We design a randomized rounding algorithm to approximate the optimal layered multicast, and show the efficacy of our technique using simulations. We then proceed to further generalize our model by studying the optimal progression of layer sizes. We show that such optimization is non-convex, and apply a Simulated Annealing algorithm to solve it, with flexible trade-off between solution quality and running time. We verify the effectiveness of the new model and the Simulated Annealing algorithm through extensive simulations, and point out insights on the connection between optimal layer size progression and node capacity distribution.

Categories and Subject Descriptors: C.2.0 [**Computer-Communication Networks**]: General; C.2.2 [**Computer-Communication Networks**]: Network Protocols; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems; G.1.6 [**Numerical Analysis**]: Optimization—*Integer programming; Linear Programming; Simulated Annealing*

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Multicast Routing, Network Coding, Optimization

1. INTRODUCTION

Multicast is an efficient mechanism in communication networks for data delivery from a source to multiple receivers. As a canonical example, media streaming applications that recently proliferated over the Internet transmit live or on-demand media flows to large groups of users across the world. As opposed to separate one-

Authors' address: Department of Computer Science, 2500 University Drive NW, T2N 1N4 AB, Canada; email: {ajay.gopinathan, zongpeng}@ucalgary.ca

Permission to make digital/hard copy of all or part of this material without fee for personal or classroom use provided that the copies are not made or distributed for profit or commercial advantage, the ACM copyright/server notice, the title of the publication, and its date appear, and notice is given that copying is by permission of the ACM, Inc. To copy otherwise, to republish, to post on servers, or to redistribute to lists requires prior specific permission and/or a fee.

© 20XX ACM 0000-0000/20XX/0000-0001 \$5.00

to-one (unicast) communication sessions, multicast exploits the replicable property of information flows, reduces redundancy in the transmission and improves the efficiency of network bandwidth utilization [Feigenbaum et al. 2001; Chen et al. 2000; Wang et al. 2005; Garg et al. 2003; Banerjee et al. 2002]. Computing optimal multicast flows, however, was shown to be equivalent to packing Steiner trees [Chen et al. 2000; Li et al. 2006], which is NP-hard [Jain et al. 2003; Thimm 2001]. However, recent breakthroughs in network coding research [Ahlsweide et al. 2000; Koetter and Médard 2003; Li et al. 2005; Lun et al. 2005] demonstrate that, by further exploiting the unique encodable property of information flows, optimal multicast routing can be modelled as a union of *conceptual* network flows and therefore efficiently computed.

The presence of heterogeneity in networks such as the Internet presents a new dimension of challenge for designing optimal multicast algorithms. Single rate multicast suffers from either starving receivers with low capacity or under-utilizing available bandwidth for high capacity receivers. The solution is to encode source data into layers for transmission [Sacham 1992; McCanne et al. 1996; Kar et al. 2001; Li et al. 1997]. Layering provides flexibility in that each receiver may subscribe to as many layers as possible, subject to its receiving capacity. When transmitting layered media streams, layers can be incrementally used to provide progressive refinement in terms of playback quality. Layering schemes broadly fall into two categories - *non-cumulative* and *cumulative* layering. In non-cumulative schemes such as Multiple Description Coding [Gamal and Cover 1982], media is encoded into independent layers, and any subset of them can be decoded by a receiver for playback at a corresponding quality level. Examples of cumulative layering codes include MPEG-2 and H.263 [ISO/IEC 1995; ITU 1998]. In this model, media is encoded into a base layer first, which is the minimum requirement for playback, and then subsequent layers that may be decoded to enhance the playback quality – but only if they are received in a consecutive sequence. In order to decode layer k , layers 1 through $k - 1$ must be present as well.

Such cumulative decoding nature constitutes a unique challenge towards optimal multicast algorithm design. Clearly, there is a distinction between data received at the network layer, and data playable at the application layer. Not all data received can be played, unless they constitute entire layers that form a consecutive sequence. Based on this observation, previous models [Zhao et al. 2006; Xi et al. 2007] for high throughput layered multicast chose to model cumulative decoding by enforcing users to receive exactly what is decodable. The rationale was that data that cannot be decoded does not contribute towards playback quality and is a waste of resource. Surprisingly, such restricted reception mechanisms, although intuitively sound, will not lead to accurate algorithm models that provide the absolute optimal performance. It turns out that having a node “waste” its bandwidth in receiving data that it cannot decode and play is in general necessary. The reason is that nodes at critical network locations may need to assist in the coding and forwarding of data at the network layer just for the benefit of downstream peers. We illustrate the social benefit of such altruistic behavior using two examples.

Fig. 1 (a) shows an example where sender S multicasts to receivers T_1 and T_2 . Source data consists of three layers with constant size progression: (2,2,2). If

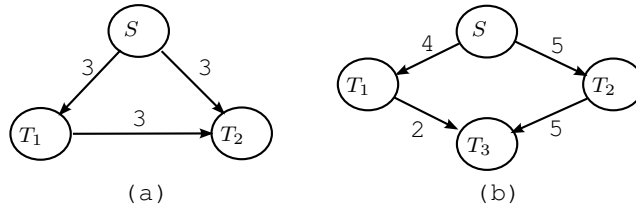


Fig. 1. In (a), layer sizes are (2,2,2). Receiver T_2 is unable to achieve maximum throughput unless T_1 receives and forwards partial layers. In (b), layer sizes are (2,3,2). T_3 can only obtain all three layers if T_1 receives and forwards non-continuous layers.

receivers are allowed to receive layers only in entirety, the maximum throughput is 6: T_1 receives layer 1 and forwards it to T_2 as well, and T_2 receives layer 2 directly from S . The residual network now has unit capacity in each link, and cannot accommodate the last layer. However, with partial layer reception enabled, T_1 should receive a part of layer 3 so that T_2 can receive and play all three layers, thereby leading to the optimal throughput of 8. Similarly, obtaining non-continuous layers also helps to improve throughput. In Fig. 1 (b), S multicasts three layers of size (2,3,2). With in-sequence layer reception only, the total throughput is at most 12, which occurs when T_1 receives layer 1, and T_2 receives layers 1 and 2 while forwarding it to T_3 . The residual network now has 2 units of capacity on links $S - T_1$ and $T_1 - T_3$ and zero capacity on the other two links. By additionally obtaining the out-of-order third layer, T_1 can assist T_3 in receiving this last layer, leading to a total throughput of 14.

From the discussions above, we can conclude that an optimal multicast algorithm should both (a) take into account the cumulative decoding nature of layered data, by screening out non-decodable data when computing throughput, and (b) be flexible enough so that reception of non-decodable data is still allowed whenever necessary, including in particular receiving partial layers or non-consecutive layers. We address the above challenges by decoupling application layer throughput from network layer throughput, and keep track of them separately in the algorithm model. Such separate treatment is naturally based on recent conceptual flow models for single-layer multicast with network coding [Li et al. 2005; Lun et al. 2005; Li et al. 2006]. The result is a more sophisticated model that is both safe, *i.e.*, generates legal multicast flows only, yet flexible, *i.e.*, does not preclude any legal multicast flow and provides the absolute optimal flow in the solution. Simulation results confirm that considerably higher throughput as well as lower routing costs can be achieved, as compared to previous layered multicast models [Zhao et al. 2006; Xi et al. 2007] or one-layer only optimal multicast [Li et al. 2005; Lun et al. 2005].

We additionally study the complexity of computing the optimal layered multicast flow. We provide a polynomial time transformation from the NP-complete integer knapsack problem [Karp 1972; Garey and Johnson 1979; Papadimitriou and Steiglitz 1998], to the layered multicast problem, thereby proving that the latter problem is NP-complete as well. Subsequently, we provide an approximation technique for computing the optimal layered multicast based on the randomized rounding. Our simulations show that the randomized rounding algorithm computes a layered

multicast routing scheme that is close to the optimal in most cases.

We further study the related problem: how achievable multicast throughput depends upon the layer size progression. The progression of layer sizes defines how data is distributed across layers, and profoundly affects the achievable optimal multicast throughput. Although layer sizes are not always freely tunable in existing layered coding schemes, such study will at least shed light onto the opportunity cost of current coding schemes (the loss in throughput due to the fixed layer size progression), which in turn may serve as a valuable reference for future design of layered codes. We show that computing the optimal size for each layer is not a straightforward task, since no prefixed layer size progression performs well for all multicast configurations. When layer sizes become variables, our mathematical program takes the form of a non-linear integer program whose solutions form a non-convex space. We propose to apply simulated annealing [Kirkpatrick et al. 1983] to solve it and compute the optimal layer size progression. Our algorithm allows flexible trade-off between optimality and solution time. We perform extensive simulations to verify the correctness of the algorithm, and identify the underlying connection between optimal layer size progression and user capacity distribution.

The rest of this paper is organized as follows. In section 2, we discuss previous research related to this work. We discuss our solution and mathematical program for optimal layered multicast in section 3. At the same time, we prove the intractability of the layered multicast problem, and describe an approximation technique based on randomized rounding. In section 4, we seek to optimize layer sizes for layered multicast using simulated annealing. We then conclude our findings in section 5.

2. RELATED WORK

Traditional multicast algorithm design are mostly based on multicast trees or meshes that can be decomposed into a set of trees [Feigenbaum et al. 2001; Chen et al. 2000; Wang et al. 2005; Garg et al. 2003; Banerjee et al. 2002]. However, under such tree models, optimizing multicast throughput and cost are equivalent to the combinatorial problems of Steiner tree packing and minimum Steiner tree respectively [Li et al. 2006], and are NP-hard [Jain et al. 2003; Thimm 2001]. A recent breakthrough in information theory dramatically changed the picture. A new multicast feasibility characterization is developed by exploiting the encodable as well as replicable properties of information flows: *a multicast rate d is feasible in a directed network if and only if it is feasible as a unicast to each receiver separately* [Ahlsvede et al. 2000; Koetter and Médard 2003]. Such coding operations applied potentially at any node in the network is referred to as *network coding*. Efficient multicast algorithm design were soon spawned from this new network flow based multicast routing structure, in directed, undirected, wireless, and non-cooperative network models [Li et al. 2005; Li and Li 2005; Lun et al. 2005; Wu et al. 2005; Yuan et al. 2006; Li 2007]. These algorithms are essentially transmitting one layer only. We instead study in this paper optimal multicast algorithms for multiple layers encoded with layered coding.

Layering is a well known mechanism that aims to better exploit heterogeneity in a network. Prior to layering, rate adaptation was performed by the sender to best match available network capacity. Sacham [1992] studied optimal routing and error

control mechanisms for layered coding with layered transmission rates. McCanne et al. [1996] introduced a receiver driven approach to rate adaptation in which receivers adapt to both static bandwidth limitations as well as congestion conditions in the network, by dynamically adding and dropping available data layers. Kar et al. [2001] propose two distributed and scalable algorithms for achieving optimal rates that maximize receiver utility in multirate multicast. Li et al. [1997] proposed a system for distribution of layer coded video over the Internet with a focus on improving reception quality, using efficient retransmission mechanisms. Li and Liu [2003] give an overview of layered multicast for video applications on the Internet.

Sundaram et al. [2007] propose an algorithm that computes multicast routing with network coding for heterogeneous receivers. Basically, as many layers are created as there are distinct receiving capacities. However, the number of receivers may differ dramatically in different scenarios and is usually unknown. In contrast, a prescribed number of layers is considered in other recent studies of layered multicast [Zhao et al. 2006; Xi et al. 2007] and in this work.

A model for layered multicast with network coding was first proposed by Zhao et al. [2006]. In the proposed integer linear program, cumulative decoding was handled by enforcing nodes to receive only what they can play. Consequently, there is no difference between network layer throughput and application layer throughput. This simplifies the modelling but results in sub-optimal solutions. A similar model was proposed by Xi et al. [2007], which also suffers from the same problem of restricting receivers to receive complete and in sequence layers only. In contrast to previous work, we make explicit the distinction between network and application layer data, and design a more sophisticated mathematical program that provides absolute maximum playback throughput. We also generalize our solution to min-cost multicast and optimal layer size computation.

Both the models of Zhao et al. [2006] and Xi et al. [2007] focused on the case when coding is only performed across flows carrying data from the same layer. In contrast, the recent work of Dumitrescu et al. [2009] performs network coding across layers as well. The result is that different users may receive different flow rates within a single layer. This latter property requires an algorithm to compute the requisite codes in order to achieve the computed flow rates, which adds to extra overhead in the overall scheme. While this can sometimes achieve better overall throughput, it is not entirely clear if the extra overhead is justifiable. In our work, we focus only on the case of intra-layer coding. Such a scheme is guaranteed to have a coding scheme, and further, random coding schemes will succeed with high probability [Ho et al. 2006]. Consequently, our scheme is more amenable to simple distributed implementations.

Non-linear integer programming is NP-hard in general. Previous solution methods include branch and bound [Quesada and Grossmann 1992] [Gupta and Ravindran 1985], outer approximation [Fletcher and Leyffer 1994] and generalized bender's decomposition [Geoffrion 1972]. These methods tend to be problem specific, assume generalized convexity or rely on solutions of relaxed non-linear sub-problems. More general heuristic-based methods include local search, genetic algorithms, and simulated annealing [Kirkpatrick et al. 1983]. Simulated annealing has proven successful in that it does not require convexity and can be applied to non-

convex and non-linear problems. Unlike gradient based algorithms such as local search, it can avoid being trapped at local minima (maxima), and has been shown to provably converge to the global optimum if the cooling schedule is suitably long enough [Hajek 1988].

3. OPTIMAL LAYERED MULTICAST

We present in this section our model for optimal layered multicast, in the form of a mixed integer linear program (MILP). We explain our network model and notations, derive the MILP in detail, and use empirical results to demonstrate its optimality.

3.1 Network Model and Notations

We model the network as a directed, capacitated graph. We assume that the topology of the network is fully known, and that the nodes are under some centralized control. The graph topology is $\mathcal{G} = (V, E)$, and each link \vec{uv} has an associated capacity $C(\vec{uv})$. $S \in V$ is the multicast sender, and $T = \{T_1, \dots, T_{|T|}\} \subseteq V$ is the set of receivers. The set of multicast receivers is assumed to be static. In the dynamic case when receivers leave and join the multicast session, the optimal multicast flow has to be recomputed after each join/leave operation. The data to be multicast is represented by a vector $\mathbf{l} = (l_1, \dots, l_L)$ where l_k is the size of layer k and L is the number of layers in total.

We use two separate vectors x and y to help keep track of application layer and network layer throughput, respectively. The fractional variable $0 \leq y_k^i \leq 1$ denotes the fraction of layer k data received by receiver i . The binary variable x_k^i denotes whether receiver i may decode and use layer k in its playback:

$$x_k^i = \begin{cases} 1, & T_i \text{ is able to play layer } k \\ 0, & \text{otherwise} \end{cases}$$

The celebrated multicast feasibility condition with network coding states that a multicast rate d is feasible if and only if it is feasible as an independent unicast to each receiver [Ahlsvede et al. 2000; Koetter and Médard 2003]. In fact, we can take the union of such conceptual unicast flows f^i to construct a multicast flow f , such that $f(\vec{uv}) = \max_i f^i(\vec{uv}), \forall \vec{uv}$. Here $f^i(\vec{uv})$ is the flow rate from S to T_i on link \vec{uv} . Flows toward different receivers can always share the capacity on $f^i(\vec{uv})$, due to information replication and encoding [Li et al. 2006]; therefore the aggregated flow rate on the link is just the maximum conceptual flow rate. However, since we study the multicast of layered data, we essentially have a multicast session within each layer, where f_k^i represents a conceptual flow to T_i in layer k , and f_k is the overall multicast flow in layer k .

In practice, each of the layers may consist of both original data and data for error protection, such as Cyclic Redundancy Check (CRC). We view original data and redundancy data equally in the multicast routing and do not distinguish between them. However, we allow a node to receive partial layers, and CRC validity check for a certain layer is performed only on nodes who can successfully receive and decode that layer.

We are now ready to introduce our MILP model for optimal layered multicast.

3.2 Mathematical Program Formulation

We first describe our model for throughput maximization; cost minimization will be discussed later. Naturally, the goal is to maximize the total data rate for playback at all users, *i.e.*, the total application layer throughput:

$$\sum_i \sum_k l_k \cdot x_k^i \quad (1)$$

We next establish linear constraints that validate vector x , such that once these constraints are satisfied, then $x_k^i = 1$ if and only if T_i can play layer k . Such constraints can be separated into two groups: the first group sets up conceptual flows f_k^i and the multicast flows f_k at the network layer, and the second group establishes connections between network layer throughput and application layer throughput. We start with the first group. At network layer, flow conservation must be satisfied by each conceptual unicast flow f_k^i :

$$\sum_{v \in N(u)} [f_k^i(\vec{uv}) - f_k^i(\vec{vu})] = 0 \quad \forall k, \forall i, \forall u \quad (2)$$

The constraint in (2) ensures that the conceptual flow in each layer k from S to T_i is conserved at all nodes u . Here, $N(u)$ represent nodes that are either upstream or downstream neighbors of u . We assume that there is a conceptual feedback link $\overrightarrow{T_i S}$ from every receiver T_i back to S , therefore flow conservation is feasible at the source and the receivers too. Furthermore, the flow rate on $\overrightarrow{T_i S}$ exactly equals the network flow rate from S to T_i due to global flow conservation.

The next constraint expresses the fact that the multicast flow f_k in each layer k should be the union of the conceptual flows in that layer, *i.e.*, $f_k = \max_i f_k^i$. Since max is not a linear function, it is relaxed to the inequalities below. We will argue later that such relaxation does not affect the optimal solution.

$$f_k^i(\vec{uv}) \leq f_k(\vec{uv}) \quad \forall k, \forall i, \forall \vec{uv} \quad (3)$$

Aggregated flow rates from all layers should respect link capacity limits:

$$\sum_k f_k(\vec{uv}) \leq C(\vec{uv}) \quad \forall \vec{uv} \quad (4)$$

We have now established all the flow constraints at the network layer, and proceed to define application layer variables based on them. Recall that $f_k^i(\overrightarrow{T_i S})$ is the network flow rate from S to T_i at layer k , and y_k^i denotes the fraction of total layer size received by T_i , therefore:

$$f_k^i(\overrightarrow{T_i S}) = l_k \cdot y_k^i \quad \forall i, \forall k \quad (5)$$

A receiver T_i may play layer k only if it receives it in its entirety, and that can be modelled by:

$$x_k^i \leq y_k^i \quad \forall i, \forall k \quad (6)$$

Finally, a receiver T_i may play layer k only if it is already able to play all the lower layers:

$$x_L^i \leq x_{L-1}^i \dots \leq x_2^i \leq x_1^i \quad \forall i \quad (7)$$

As an aside, we note that if non-cumulative layered coding is to be modelled instead, then we need only to remove the last constraint group in (7).

Having presented our objective function as well as the necessary constraints, we are almost ready to present our mathematical program for optimal layered multicast. Before we do so however, we note that constraints (5) - (7) can be manipulated and expressed in a more compact fashion, by making the variables in y_k^i implicit within the formulation. This can be achieved by expressing y_k^i as the ratio of conceptual flow received in a given layer to the size of that layer, and substituting this new term into (6). This results in a new expression for x_k^i , which we can now substitute into (7), yielding the following constraint that is equivalent to (5), (6) and (7) combined:

$$x_{k+1}^i \leq x_k^i \leq \frac{f_k^i(\vec{T}_i S)}{l_k} \quad k = 1, \dots, L-1, \forall i \quad (8)$$

The reformulated constraint in (8) eliminates variables in y_k^i , and reduces the complexity of the final model. We now present our mixed integer linear program for optimal layered multicast:

$$\begin{array}{ll} \text{Maximize} & \sum_i \sum_k l_k \cdot x_k^i \\ \text{Subject to:} & \end{array} \quad (9)$$

$$\left\{ \begin{array}{ll} \sum_{v \in N(u)} [f_k^i(\vec{uv}) - f_k^i(\vec{vu})] = 0 & \forall k, \forall i, \forall u \\ f_k^i(\vec{uv}) \leq f_k(\vec{uv}) & \forall k, \forall i, \forall \vec{uv} \\ \sum_k f_k(\vec{uv}) \leq C(\vec{uv}) & \forall \vec{uv} \\ x_{k+1}^i \leq x_k^i \leq \frac{f_k^i(\vec{T}_i S)}{l_k} & \forall k = 1..L-1, \forall i \end{array} \right.$$

$$f_k(\vec{uv}), f_k^i(\vec{uv}) \geq 0, \quad x_k^i \in \{0, 1\} \quad \forall k, \forall i, \forall \vec{uv}$$

Our mathematical program accurately models and optimizes layered multicast with network coding, in that it includes all the necessary constraints without being overly restrictive. In particular, the relaxation of the max function in (3) will not affect the optimal solution given the context of the overall mathematical program. In contrast to previous work [Zhao et al. 2006; Xi et al. 2007], no valid multicast flow at the network layer is precluded by the set of constraints. The cumulative decoding nature of layered coding is also appropriately handled by the constraint

$$x_{k+1}^i \leq x_k^i \leq \frac{f_k^i(\vec{T}_i S)}{l_k}, \quad \forall i, \forall k \neq L.$$

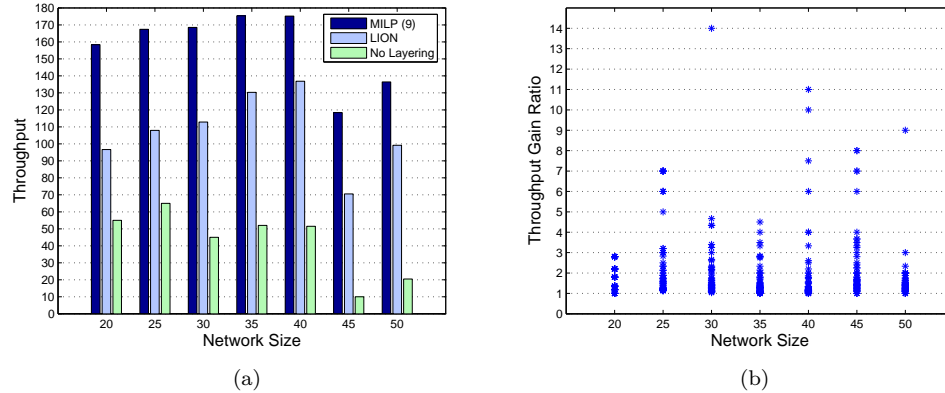


Fig. 2. The bar chart in (a) shows the average throughput obtained by our model as compared to LION and multicast without layering for varying network sizes and a fixed number of receivers. In (b), we plot the throughput gain ratio observed throughout our simulations.

We next study the performance of the MILP model and verify its optimality through simulations. Since we are interested in studying and comparing the absolute maximum throughput that can be achieved, we resort to an exact solution algorithm, namely, branch-and-bound, while keeping network sizes moderate. For faster approximation algorithms, the simulated annealing method described in the next section can be applied, to enable flexible trade-offs between solution optimality and running time. We also point out that since our objective is to maximize absolute throughput, the obvious decomposition of our mathematical program by relaxing the last constraint in (7) and solving each subproblem at each layer (layer-by-layer optimization) will not lead to the correct solution; this can be verified by the example in Fig. 1 (b), with layer size progression (2,2,3).

3.3 Simulation Results and Discussion

We solve both our MILP model and previous models for layered multicast to compare the maximum throughput that they achieve in various network settings. Network topologies were generated using the Boston University Representative Internet Topology generator (BRITE) [BRITE], a tool for generating topologies that resemble the structure of the Internet. Link capacities in the network are uniformly randomly distributed from 5 to 20. Unless stated otherwise, the total amount of data to be multicast was 30, which was split into layers. To solve the mixed integer linear program in (9), we employed the branch and bound algorithm as implemented in the GNU Linear Programming Kit (g1pk 4.16) [GLPK].

Fig. 2a shows a comparison of the maximum throughput obtained in networks with various sizes, using our mixed integer linear program in (9), and two other schemes. The multicast group was chosen at random, with a single sender multicasting layered data to 19 receivers. We denote by LION the incomplete models of Zhao et al. [2006] and Xi et al. [2007], while the “No-layering” model represents optimal single-rate multicast with network coding [Li et al. 2005; Li et al. 2006].

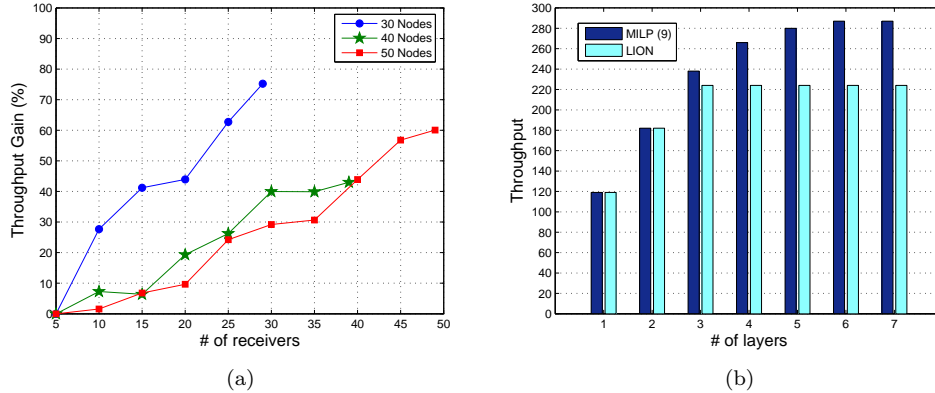


Fig. 3. In (a), the throughput gain of our model over LION for different network sizes with varying number of receivers is shown. The throughput obtained by our model as compared to LION with varying number of layers is shown in (b). As the number of layers increase, LION is unable to improve after 3 layers, while our model is able to more efficiently utilize bandwidth to route the extra data as the number of layers increases.

Without layering, we are unable to accommodate the heterogeneity in the network, resulting in the lowest throughput. While the use of layers is able to improve on this, our model provides significantly higher throughput than LION does. In fact, as Fig. 2b shows, our model has an average throughput gain of between 1 and 3 times the throughput obtained with LION. In some cases, the recorded maximum gain was as high as 14 times the throughput obtained by LION. The underlying reason is that our MILP has a strictly larger feasibility region than LION does, and embraces all legal multicast flows.

We next investigate the effect of changing the number of receivers in the network. For a given network size, we generated five random graphs, for each of which five random permutations of multicast receivers were chosen. In Fig. 3a, as the number of receivers increases, our model shows an increase in the relative gain in throughput compared to LION. As more receivers join the multicast group, it becomes increasingly crucial that these receivers also receive partial or out of sequence layers in order to efficiently utilize the available bandwidth, *i.e.*, it is even more important for a larger user community to be collaborative.

Fig. 3b shows the effect of increasing the number of layers in a network with 30 nodes and 20 receivers. In this example, the total layer size was not fixed. As the number of layers increases, our model is able to take advantage of the residual bandwidth to route the extra available data and thereby increase the multicast throughput. On the other hand, LION plateaus when the number of layers reaches 3 in this particular example. Further increasing the number of layers does not increase throughput, since in LION, saturated receivers cannot receive partial layers and therefore are unable to take advantage of the residual bandwidth, thus forming a bottleneck.

We also examined the effect of layer sizes on the performance of our model as compared to LION. Fig. 4a shows the throughput gained for different layer sizes,

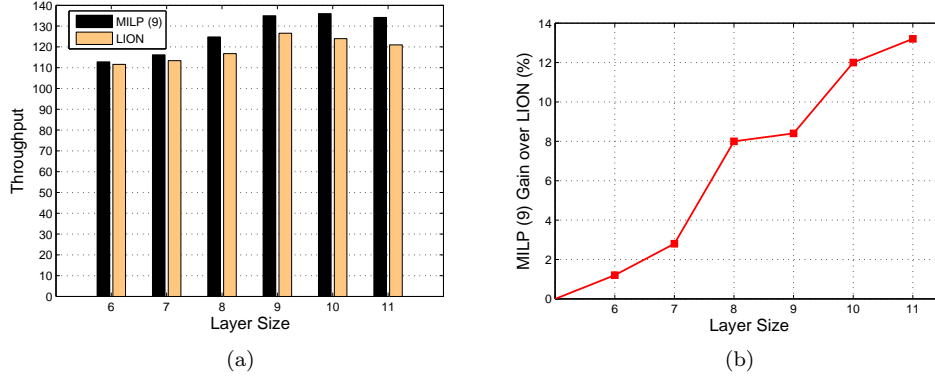


Fig. 4. The throughput for different layer sizes (with the number of layers fixed at 3) is shown in (a), for networks with 50 nodes and 10 multicast receivers. The throughput obtained by our model is higher than LION in all cases, and improves as the layer size increases. The percentage of throughput gain by our model over LION is shown in (b).

while fixing the number of layers. The results were averaged over random topologies with 50 nodes and 10 multicast receivers. As layer sizes increase, our model continues to obtain more throughput than LION. In Fig. 4b, we plot the gain of our model over LION. We note that the gain of our model as compared to LION increases with the layer size. This is because as the layer size grows, our model is able to stay flexible by routing partial and out of sequence layers, thus better exploiting the residual bandwidth in order to achieve the absolute optimal throughput. Nevertheless, we remark that when the layer sizes are small, LION achieves throughput that is comparable to our model.

Our model for optimal layered multicast also readily extends to the problem of minimizing the total cost for layered multicast routing. For the min-cost problem, each edge has an associated unit flow cost $c(\vec{uv})$. Given a fixed per-node layer playback requirement specified in a *constant* vector x , we can formulate the min-cost optimization problem as the following linear program:

$$\begin{aligned} \text{Minimize} \quad & \sum_{\vec{uv}} \sum_k f_k(\vec{uv}) c(\vec{uv}) \quad (10) \\ \text{Subject to:} \quad & \end{aligned}$$

$$\begin{cases} \sum_{v \in N(u)} [f_k^i(\vec{uv}) - f_k^i(\vec{vu})] = 0 & \forall k, \forall i, \forall u \\ f_k^i(\vec{uv}) \leq f_k(\vec{uv}) & \forall k, \forall i, \forall \vec{uv} \\ \sum_k f_k(\vec{uv}) \leq C(\vec{uv}) & \forall \vec{uv} \\ x_k^i \leq \frac{f_k^i(T_i S)}{l_k} & \forall k, \forall i \\ f_k(\vec{uv}), f_k^i(\vec{uv}) \geq 0 & \forall k, \forall i, \forall \vec{uv} \end{cases}$$

We examine the effect of our min-cost multicast formulation for layered multicast in Fig. 5. We generated topologies of various sizes, and assigned costs on each link randomly from the range [20, 40]. Each topology consists of 20 randomly

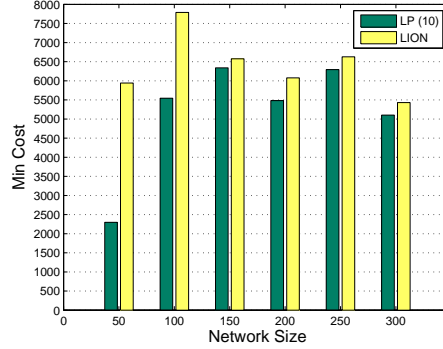


Fig. 5. Average min-cost for different sizes of networks as computed by our linear program in (10) as compared to the model in LION.

chosen multicast receivers. Our simulations demonstrate that our model is more efficient than LION, and consistently computes cheaper min-cost flows even in large networks.

3.4 Intractability of the layered multicast problem

In section 3, we have modeled the layered multicast problem as a mixed integer linear program. Since integer linear programs are NP-complete in general [Garey and Johnson 1979], one is motivated to ask if better modeling may yield a polynomial-time computable solution, or if the problem itself is inherently intractable. In this section, we prove that optimal layered multicast is in fact NP-complete, thereby justifying our mixed integer linear program formulation.

Before we proceed with our proof, let us introduce the integer knapsack problem.

Definition 3.1. The INTEGER KNAPSACK problem is defined as follows – given a set of integers $\{c_1 \dots c_m\}$, together with an integer K , are there integers $x_i \geq 0$ such that $\sum_{i=1}^m c_i x_i = K$?

The INTEGER KNAPSACK problem is known to be NP-complete [Karp 1972; Papadimitriou and Steiglitz 1998; Garey and Johnson 1979]. We will show a polynomial time reduction from any INTEGER KNAPSACK problem to the layered multicast problem, thereby proving the latter to be NP-complete.

Given an instance of the INTEGER KNAPSACK problem, we will reduce it to an equivalent instance of the layered multicast problem, denoted by LAYERED MULTICAST. Assume without loss of generality that the set $\{c_1, \dots, c_m\}$ is sorted in ascending order. We create an instance of LAYERED MULTICAST by first creating m layers, $l_1 \dots l_m$ using $\{c_1, \dots, c_m\}$ in the following way

$$l_i = c_i - c_{i-1} \quad i = 1 \dots m, c_0 = 0$$

To construct the network for LAYERED MULTICAST, we create $|T| = m \cdot K$ receivers $T_1 \dots T_{|T|}$, and connect a source node S to each receiver using a single directed edge with capacity $\max_i c_i$. The construction is illustrated in Fig. 6. We then claim that the following theorem is true:

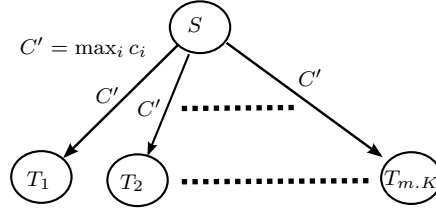


Fig. 6. An instance of the **INTEGER KNAPSACK** problem can be reduced to the **LAYERED MULTICAST** problem by constructing a network as shown in this figure. The source node S is connected to $m.K$ receivers separately, using a single directed edge of capacity $C' = \max_i c_i$ to each receiver.

THEOREM 3.2. *There exists a vector $x \in \mathbb{Z}^+$ such that $\sum_{i=1}^m c_i x_i = K$ for **INTEGER KNAPSACK** if and only if in the corresponding **LAYERED MULTICAST** problem, there exists a multicast flow that achieves a throughput of K .*

PROOF. Assume that there exists a multicast flow in the **LAYERED MULTICAST** problem that achieves a throughput of K . Observe that by construction, each multicast receiver's total throughput can only take on values in $\{c_1, \dots, c_m\}$. For every possible throughput value c_i , let the number of receivers with this throughput be z_i . Then clearly, there exists a solution to the **INTEGER KNAPSACK** problem x with $x_i = z_i$, such that $\sum_i c_i x_i = K$.

To prove the other direction, assume that there is a solution vector x to the **INTEGER KNAPSACK** problem such that $\sum_i c_i x_i = K$. Without loss of generality, we can assume that $\max_i c_i \leq K$. Clearly, $x_i \leq K$ for all i . Since there are $m.K$ receivers, the number of receivers are sufficient such that a multicast flow that achieves a throughput of K exists, where x_i receivers achieve a throughput of c_i . \square

The reduction from an instance of **INTEGER KNAPSACK** to an instance of **LAYERED MULTICAST** can clearly be done in polynomial time. Further, the **LAYERED MULTICAST** problem is in NP, since a valid layer assignment can also be checked in polynomial time, by solving the degraded linear program of (9). This allows us to conclude that the **LAYERED MULTICAST** problem is NP-complete.

3.5 A randomized rounding based approximation algorithm

We have shown that computing the optimal solution to the mixed integer linear program (MILP) in (9) is NP-Hard. In this section, we will describe a randomized rounding algorithm, that runs in polynomial time, and allows us to compute an *approximate* solution to MILP (9). We demonstrate the efficacy of our algorithm using simulations, and show that the approximate solution computed by our randomized rounding algorithm performs close to the optimal in most cases.

The algorithm is show in Algorithm 1, and works as follows – we first solve the linear programming (LP) relaxation of MILP (9). The LP relaxation provides a fractional solution vector (\hat{x}, y) . We then examine the fractional layer assignment x for each receiver $T_i \in T$. Note that since the fractional value \hat{x}_k^i as computed by the LP relaxation of MILP (9) always lies between 0 and 1, it can be viewed as the *probability* that receiver T_i should be assigned layer k . The LP relaxation also provides us with an *estimate* of the available capacity $C_e(T_i)$, for receiver T_i

```

Input: Network graph,  $\mathcal{G} = (V, E)$ , capacity vector  $C$ , source node  $S$ ,
          multicast receivers  $T$ , layer sizes  $l$ 
Output: Feasible assignment of layers to receivers,  $x$ , optimal throughput  $O$ 
 $O := 0$  ;
Solve LP relaxation of MILP (9), let  $(\hat{x}, y)$  denote the fractional solution ;
while trials < MAXTRIALS do
  foreach  $T_i \in T$  do
     $j := 1$  ;
     $x_0^i := 1$ ;
     $C_e(T_i) := \sum_k l_k \cdot y_k^i$  ;
    while  $j \leq L$  do
      if  $l_j \leq C_e(T_i)$  and  $x_{j-1}^i = 1$  then
         $r :=$  random number in  $[0, 1]$  ;
        if  $r \leq \hat{x}_j^i$  then
           $x_j^i := 1$  ;
           $C_e(T_i) := C_e(T_i) - l_j$  ;
        else
           $x_j^i := 0$ 
        else
           $x_j^i := 0$ 
         $j := j + 1$  ;
      end
    end
     $O_t :=$  optimal throughput of degraded LP relaxation of (9), using  $x$  ;
    if  $O_t > O$  and  $x$  is a feasible assignment then
       $O := O_t$  ;
    trials := trials + 1 ;
  end

```

Algorithm 1: A randomized rounding based algorithm for approximating the optimal end-to-end throughput. The LP relaxation of (9) is used to provide an initial starting fractional solution, which is then rounded to an integral solution probabilistically.

where $C_e(T_i) = \sum_k l_k \cdot y_k^i$. Let x_k^i be the layer assignment for receiver T_i and layer k as obtained by our randomized rounding algorithm. Then we set $x_k^i = 1$ with probability \hat{x}_k^i , if the following two conditions hold; (1) the remaining estimated capacity is greater than the size of layer k , and (2) $x_{k-1}^i = 1$, thus ensuring the cumulative layering requirement feasibility is always satisfied. We then plug the layer assignment variables x_k^i back into the *degraded linear programming relaxation* of (9), to compute the optimal multicast throughput and routing scheme based on the values of x_k^i .

The randomized nature of our layer assignment scheme means that there is no guarantee that the vector x computed constitutes a legal and feasible assignment. Hence, we may need to repeat the rounding procedure a number of times to obtain a feasible assignment. In our experiments, we found that an upper bound of 20 trials sufficed to guarantee a feasible assignment in all cases.

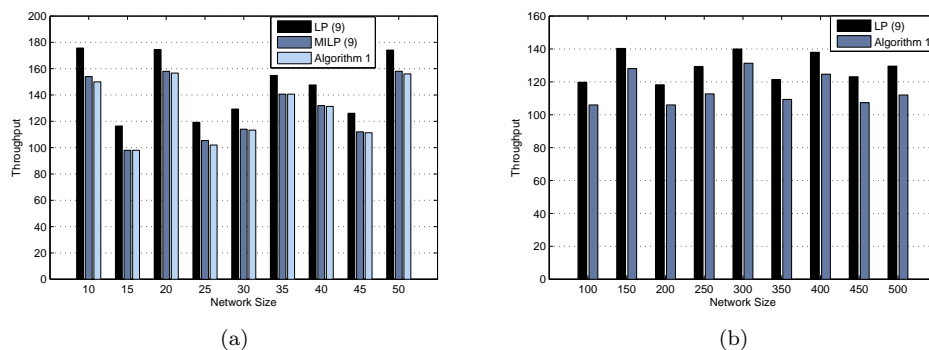


Fig. 7. In (a), the performance of the randomized rounding technique of Algorithm 1 as compared to the optimal throughput computed by MILP (9), as well as and the throughput upper bound of the LP relaxation of (9) is shown for networks of 50 nodes. The throughput obtained by the randomized rounding algorithm is compared to the throughput upper bound for larger networks sizes in (b).

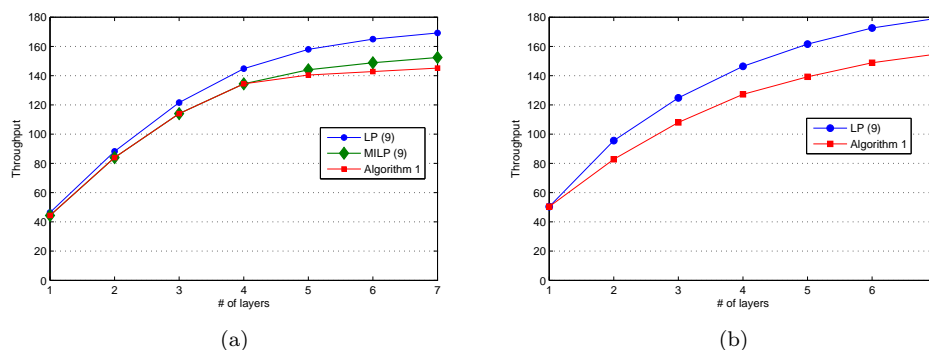


Fig. 8. The effect of the number of layers is shown on the performance of Algorithm 1 in (a) for networks with 50 nodes, while (b) shows a similar experiment for networks with 300 nodes.

To demonstrate the efficacy of the randomized rounding algorithm, we ran simulations on both small as well as large networks. For large networks, MILP (9) becomes intractable, and hence, we compared the solution obtained by Algorithm 1 with the linear programming relaxation of MILP (9), which we denote by LP (9). The linear programming relaxation provides an upper bound on the optimal throughput. Fig. 7a shows the optimal throughput as computed by MILP (9) and its LP relaxation, as well as that obtained by Algorithm 1, for various small network sizes, with 10 multicast receivers and 5 layers. As can be seen, Algorithm 1 performs significantly well, and is never more than 5% away from the optimal solution, and always within 15% of the upper bound. Fig. 7b shows the comparison of throughput obtained by Algorithm 1, against the upper bound of the LP relaxation of MILP (9) for larger networks. Again, the throughput obtained by Algorithm 1

is always within 15% of the upper bound. We also studied the effect of the number of layers on the performance of Algorithm 1. Fig. 8a shows the throughput obtained for different number of layers, for a small network of 50 nodes, with 10 multicast receivers. As the number of available layers increase, MILP (9), as well as its LP relaxation, is able to continuously exploit heterogeneity and increase the overall multicast throughput. Algorithm 1 obtains the *optimal* throughput up to 4 layers. For 5 layers and above, Algorithm 1 still obtains throughput within 5% of the optimal solution. `figref:lplayers` shows a similar experiment, but this time using large networks with 300 nodes. Again, we observe that Algorithm 1 is able to increase the throughput achieved as the number of layers is increased, and is always within 12% of the upper bound.

4. THE EFFECT OF LAYER SIZE PROGRESSION ON MULTICAST THROUGHPUT

In section 3, we presented mathematical programs for the optimal multicast of layered data to receivers in a network with heterogeneous bandwidth, assuming a fixed configuration of layer sizes. However, as our simulation results will show (section 4.2), there is no predefined layer size progression that is guaranteed to be always optimal, even for the same network with different sets of receivers. This naturally motivates us to generalize our MILP model, so that it can help compute the optimal layer size progression, for a given network configuration. As explained earlier, this helps determine the opportunity cost of a given layered coding scheme, even if we are not allowed to freely adjust the size of each layer.

Assuming variable layer sizes renders the mathematical program in (9) a *non-linear* integer program (NLIP). Recall that the objective function, $\sum_k \sum_i l_k x_k^i$, is the sum of the product between the layer size l_k and the integer x_k^i . Previously, the layer sizes are constants, and the objective function is linear. With layer sizes as variables too, the objective function, as well as the constraint in (5), become quadratic instead. This rules out solving the optimization problem using linear optimization tools such as `glpk`.

Non-linear integer programs are in general computationally intractable. Existing solutions mostly employ branch and bound methods [Gupta and Ravindran 1985; Quesada and Grossmann 1992], outer approximation [Fletcher and Leyffer 1994] or generalized bender’s decomposition [Geoffrion 1972]. These methods are able to guarantee optimality only under generalized convexity. Additionally, they are only capable of handling small problem sizes. Fig. 9 shows the change in throughput for a network of 50 nodes and a fixed total layer size of 30. The direction of change in the objective function as layer sizes vary is inconsistent, illustrating the non-convex nature of the problem. This rules out convex optimization algorithms such as the interior-point algorithm. The solution landscape is highly irregular with many locally maximum points, indicating that gradient based search algorithms would not work well, since they would be trapped in these local maxima. We propose a two-level algorithm that employs *simulated annealing* to find the optimal layer sizes while simultaneously maximizing throughput for each candidate solution of layer sizes using the mathematical program we designed in the previous section.

Simulated annealing [Kirkpatrick et al. 1983] is loosely based on the idea of

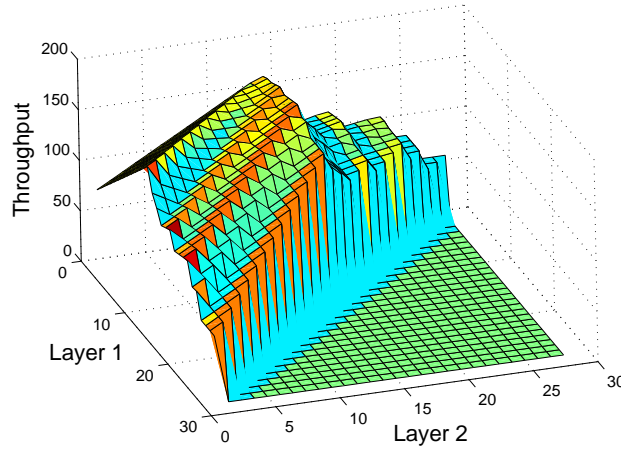


Fig. 9. This figure plots the throughput obtained by exhaustively enumerating all possible layer size progressions for a 3-layer multicast in a network with 50 nodes, 20 receivers. The solutions obtained clearly illustrate the non-convex nature of the optimization of layer size progression.

annealing in metallurgy. In this process, metal is cooled in a slow and controlled fashion to reduce the defects in the final product. The cooling process is adiabatic, which allows molecules to escape from locally minimum energy levels before finally recrystallizing in a state corresponding to the globally minimum amount of energy. The simulated annealing algorithm tries to achieve a similar effect when searching for a globally minimum (or maximum) solution. At each stage, the algorithm moves from the current solution to a neighbour solution with some probability that partially depends on how advanced the current stage of the algorithm is, represented by a *temperature* parameter, in analogy with the physical process.

Simulated annealing has enjoyed successful applications in a variety of optimization problems. Its major advantage roots in its ability to escape from local maxima (minima). Unlike other methods, simulated annealing may accept both “good” and “bad” solutions, the latter with some probability. When searching for the global maximum, the algorithm moves “uphill” while also making occasional and controlled “downhill” moves, thereby avoiding being trapped in local maxima. This makes the algorithm ideal when optimizing non-convex functions. Furthermore, the performance of simulated annealing does not depend on the starting point of the algorithm or on the structure of the objective function, which can be non-linear or even non-continuous. Simulated annealing can also be tailored to provide a compromise between optimality and running time, by controlling the number of iterations in the annealing schedule.

4.1 Methodology

In order to optimize the layer configuration, we anneal the layer sizes while using the mathematical program in (9) to evaluate the objective function, $f(\mathbf{l})$ at each point. The solution space is given by an L -dimension vector $\mathbf{l}, (l_1, \dots, l_L)$, where l_k

represents the size of layer k . The number of layers and total layer size is fixed, so that

$$\sum_k^L l_k = F \quad (11)$$

where F is the total data to be multicast.

At each iteration, a *neighbour* solution is generated by introducing small and random changes to the current solution \mathbf{l} to obtain \mathbf{l}' . Let \mathbf{v}^i be an L -dimensional unit vector with 1 in position i and 0 everywhere else. Our algorithm does the following:

- (1) Randomly choose i, j such that $1 \leq i, j \leq L$ and $i \neq j$
- (2) Generate a neighbour solution $\mathbf{l}' = \mathbf{l} + \mathbf{v}^i - \mathbf{v}^j$

The random neighbour is generated subject to (11). We also impose the additional constraint that each layer has at least unit size, since we are interested in optimizing for a prescribed number of layers.

Unlike in a local search algorithm, not only solutions leading to better objective function values can be accepted. Instead, neighbour solutions are accepted with a probability given by a probability transition function p , which depends on the current solution, the generated neighbour solution, as well as the current *temperature*, τ :

$$p = \begin{cases} e^{(f(\mathbf{l}') - f(\mathbf{l}))/\tau} & f(\mathbf{l}') < f(\mathbf{l}) \\ 1 & f(\mathbf{l}') \geq f(\mathbf{l}) \end{cases}$$

While solutions that result in a better objective function value are accepted with probability one, solutions that result in a worse outcome are accepted with some probability that depends on the current temperature. During the initial stage when the temperature is high, the algorithm is equally likely to accept bad solutions as good ones. This allows the search to escape from locally maximum points. As the algorithm progresses and the value of τ decreases, the probability of accepting bad solutions decreases as well, and the search gradually transits to accepting good solutions only.

Due to the probabilistic nature of accepting solutions based on the value of τ , the *cooling schedule* that determines how τ decreases with each iteration, is crucial to obtaining accurate and optimal solutions. The cooling schedule represents a trade-off between optimality and running time. Decreasing τ slowly results in more accurate solutions, but increases the runtime of the algorithm; decreasing τ too quickly could result in sub-optimal solutions. For our simulations, we determined experimentally that an exponentially decreasing cooling schedule worked best, where

$$\tau_{m+1} = \alpha \tau_m$$

where $0 \leq \alpha \leq 1$ and m is the number of iterations. The value of α was tuned empirically, and we found a value of 0.98 to be sufficient to achieve optimality in most cases.

Progression	Layer Sizes	Progression	Layer Sizes
constant	(6, 6, 6, 6, 6)	expInc	(1, 2, 4, 8, 15)
linearInc	(2, 4, 6, 8, 10)	expDec	(15, 8, 4, 2, 1)
linearDec	(10, 8, 6, 4, 2)	random	(1, 6, 7, 2, 14)

Table I. Different layer size progressions tested

The effectiveness of the cooling schedule also depends on the initial temperature, τ_0 . During the early stages of the algorithm, we would like some probability p_0 of accepting bad solutions. To achieve this, we sample the initial solution space randomly and accept all bad solutions to compute the average initial decrease in the objective function, δf , and then set the initial temperature as follows

$$\tau_0 = \frac{-\delta f}{\ln(p_0)}$$

Setting $p_0 = 0.8$ means bad solutions initially have an 80% chance of being accepted, which we found experimentally to be optimal.

The algorithm terminates after a fixed number of iterations, $m_{MAX} = \log_{\alpha} \tau_0$. Early termination may also happen if the algorithm hits the theoretically absolute upper bound $F|T|$, where F is the total size of all layers, and $|T|$ is the number of multicast receivers.

4.2 Simulation Results and Discussion

Having designed our simulated annealing algorithm, we now investigate how well the throughput of the output layer sizes compares with that of prefixed layer sizes, and throughput achieved by local search.

We performed simulations on networks generated randomly with BRITE [BRITE], with links chosen to have bandwidth in the range of 5-20. We fixed the total layer size at 30, and set the number of layers to 5. We then created 6 different layer configurations, *constant*, *linearly increasing*, *linearly decreasing*, *exponentially increasing*, *exponentially decreasing* as well as *random*. Table I shows the progression of these layer configurations for a fixed total size of 30. We ran simulations using all these different layer progressions on 5 different networks of 50 nodes. For each network, we randomly generated 5 sets of 10 multicast receivers. We then computed the optimal throughput using our mathematical program in (9). Table II shows the progression that gave the highest throughput for each network and permutation of multicast receivers respectively. No layer size progression performed best at all times, even *within the same network*. It is clear that optimal layer sizes depend not only on the bandwidth distribution and network topology, but also on the choice of the multicast group, since each receiver has potentially different receiving capacities. This suggests that throughput can be improved by optimizing layer sizes with respect to each multicast group in the network.

We ran both simulated annealing and a local search algorithm to optimize layer sizes for networks with sizes ranging from 20 to 50 nodes, and compared the throughput obtained with the *constant* and *linearly increasing* fixed layer size configurations. To gain insight into the simulated annealing process, we plot different

Topology	Random Multicast Group				
	a	b	c	d	e
50a	linearDec	linearInc	linearDec	linearDec	linearDec
50b	constant	linearDec	constant	constant	random
50c	linearInc	constant	constant	linearInc	linearInc
50d	linearDec	linearDec	linearDec	constant	linearInc
50e	random	linearDec	constant	linearDec	linearDec

Table II. Best progression from each experiment

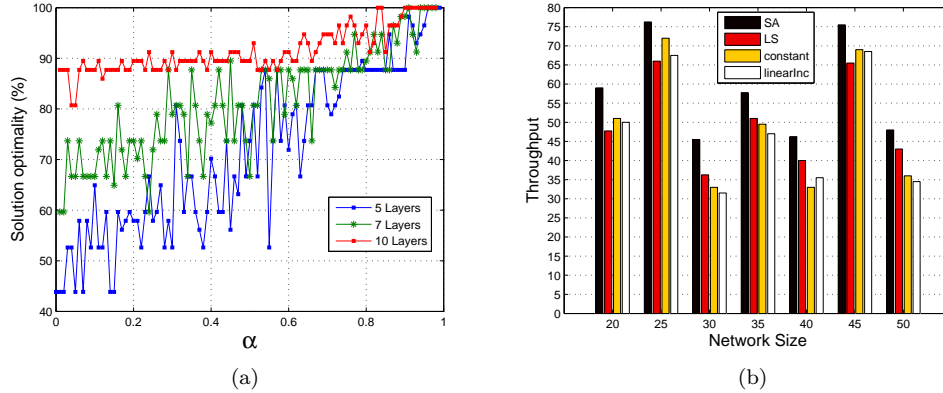


Fig. 10. The value of α can be tuned to obtain a tradeoff between optimality and running time, as shown in (a). In (b), we observe that simulated annealing obtains higher throughput than local search or fixed layer size progressions for varying network sizes.

values of the parameter α against the optimality of the solution obtained, for a network of 50 nodes using 5, 7 and 10 layers respectively, in Fig. 10a. The values of α represent a tradeoff between the running time of the algorithm and the accuracy of the final solution. This allows us to tune the algorithm to the level of optimality that is desired, subject to running time restrictions. As the value of α increases, there is a general trend of increase in the optimality of the solution obtained. However, this trend is visible only in the long run. In the short run, a smaller increase in α does not necessarily lead to a better solution than the solution previously obtained. This can be explained by the random nature of simulated annealing, which necessitates accepting worse solutions occasionally. The effect of different number of layers on the algorithm’s performance is also shown in Fig. 10a. As the number of layers increases, the solution for a given value of α is closer to optimal. This can be explained by the different granularity in the solution space. Since the total layer size is fixed, increasing the number of layers reduces the number of possible choices for each layer size, which means that even with low values of α , the solution obtained is not too far from optimality.

Fig. 10b shows the results of optimizing layer sizes with simulated annealing and three other approaches. As we can see, simulated annealing is able to achieve the highest throughput. In contrast, local search does not perform as well, and in fact in some cases, performs even worse than the fixed layer size configurations.

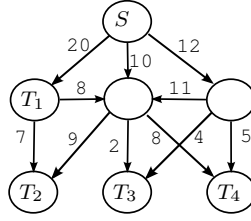


Fig. 11. The cumulative sum of the optimal layer sizes obtained for this network match the receiving capacities in the multicast group

Topology	Optimal Layer Size	Receiver Capacities
50a	(5, 4, 6, 5, 10,)	5, 9, 10, 15, 16, 20
50b	(7, 5, 1, 16, 1,)	7, 8, 9, 12, 13, 15, 42, 48
50c	(13, 4, 6, 3, 4,)	13, 14, 17, 23, 43
50d	(6, 4, 9, 9, 2,)	6, 10, 12, 19, 32, 33, 36, 41, 46
50e	(11, 2, 1, 4, 12,)	6, 11, 13, 15, 18, 37, 49
50f	(11, 2, 9, 5, 3,)	11, 13, 14, 15, 16, 22, 27
50g	(10, 3, 4, 11, 2,)	10, 13, 14, 31, 33, 39, 41

Table III. Optimal layer size progression vs. distinct receiver capacities

Again, this can be attributed to the fact that local search may get trapped in a locally optimal solution that is arbitrarily far or close from the global optimal in non-convex optimization.

Next, we investigated the connection between optimal layer size configurations and the receiving capacity of nodes in the multicast group. Fig. 11 shows an example topology where multicast receivers T_1 , T_2 , T_3 and T_4 each has max-flow values of 20, 16, 6 and 13 respectively, from S . For a total layer size of 20, using a constant layer configuration with 5 layers gives a throughput of 48. Optimizing layer sizes using simulated annealing allows us to achieve a higher throughput of 51. The optimal layer sizes obtained were not unique, and the layer size configurations (6,3,7,1,3), (6,1,2,7,4), (6,3,6,1,4) and (6,3,4,3,4) all give optimal throughput. At first glance, these layer sizes have no discernible progression; but closer examination reveals that the *cumulative* layer sizes match the max-flow values for the multicast receivers. This suggests that optimal layer size progressions should attempt to fit the receiving capacity of multicast receivers.

In order to investigate this phenomenon in larger networks, we ran the simulated annealing algorithm on seven different networks with 50 nodes and 10 receivers. Table III shows the optimal layer sizes as well as the distinct receiving capacities or max-flow values in each network. We denote in bold the receiver capacities that match the cumulative sum of the optimal layer sizes. As can be seen, the optimal layer sizes fit most of the receiver capacities that are below 30, which is the total layer size. Max-flows higher than 30 match trivially. However, it is not possible to fit every receiving capacity, due to the number of distinct capacity values as compared to the number of layers. This shows that finding the optimal layer size progression is not as trivial as exact matching to receiver capacities, and computing the optimal size progression using the proposed simulated annealing algorithm is

still necessary.

5. CONCLUSIONS

Layered multicast is a new paradigm that differs from traditional multicast in both obvious and subtle ways. Previous models, while taking into account the former, fail to deal with the latter. In this paper, we presented a complete and accurate model for optimal layered multicast. Our model makes the distinction between data flows in the application layer and the network layer. We show that this distinction is crucial to obtain an optimal multicast routing scheme. Simulation results indicate that even in medium sized networks, our model achieves significantly higher throughput compared to previous proposals. We then formally proved that the layered multicast problem is NP-complete, hence, our mixed integer linear programming formulation cannot be improved. We designed an algorithm to approximate the optimal layered multicast. Through simulations, the randomized rounding based algorithm was shown to be close to optimal for computing the end-to-end throughput in layered multicast. We also extended our model to study the effect on multicast throughput of different layer size progressions, with a simulated annealing algorithm given for computing the optimal layer size progression. We observed that no fixed layer size progression is always optimal for layered multicast in heterogeneous networks. Our simulation results show that simulated annealing tries to massage the cumulative layer size progression to fit with each receiver's max flows as much as possible. Due to the varying number of distinct receiving capacities and the fixed nature of the total layer size and number of layers, perfect matching is not usually possible. These observations may serve as a valuable insight in the future design of layered coding schemes.

Our results also point to new avenues for future research in data dissemination protocols for layered media streams. An important assumption in our work is that the nodes in the network were assumed to be cooperative. Our findings have shown that it is imperative for upstream nodes to route partial and non-sequential layers in order to achieve the absolute optimal throughput. In the non-cooperative scenario, the design of an appropriate incentive and pricing based scheme to ensure the optimal throughput is achieved constitutes an interesting challenge, which we hope to address in our future work.

ACKNOWLEDGEMENTS

The authors would like to thank the anonymous referees for their insightful comments and suggestions. This research was supported in part by NSERC and the Alberta Ingenuity Fund.

REFERENCES

- AHLWEDE, R., CAI, N., LI, S. R., AND YEUNG, R. W. 2000. Network Information Flow. *IEEE Transactions on Information Theory* 46, 4 (July), 1204–1216.
- BANERJEE, S., BHATTACHARJEE, B., AND KOMMAREDDY, C. 2002. Scalable Application Layer Multicast. In *Proc. of ACM SIGCOMM*.
- BRITE. Boston University Representative Internet Topology generator. <http://www.cs.bu.edu/brite/>.
- CHEN, S., GÜNLÜK, O., AND YENER, B. 2000. The Multicast Packing Problem. *IEEE/ACM Transactions on Networking* 8, 3, 311–318.

- DUMITRESCU, S., SHAO, M., AND WU, X. 2009. Layered multicast with inter-layer network coding. In *Proceedings of IEEE INFOCOM*.
- FEIGENBAUM, J., PAPADIMITRIOU, C., AND SHENKER, S. 2001. Sharing the Cost of Multicast Transmissions. *Journal of Computer and System Sciences* 63, 21–41.
- FLETCHER, R. AND LEYFFER, S. 1994. Solving mixed integer nonlinear programs by outer approximation. *Mathematical Programming* 66, 327–349.
- GAMAL, A. E. AND COVER, T. M. 1982. Achievable rates for multiple descriptions. *IEEE Transactions on Information Theory* 28, 851–857.
- GAREY, M. AND JOHNSON, D. 1979. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W.H. Freeman and Company, San Francisco.
- GARG, N., KHANDEKAR, R., KUNAL, K., AND PANDIT, V. 2003. Bandwidth Maximization in Multicasting. In *Proceedings of the 11th European Symposium on Algorithms (ESA)*.
- GEOFFRION, A. M. 1972. Generalized benders decomposition. *Journal of Optimization Theory and Applications* 10, 237–260.
- GLPK. GNU Linear Programming Kit. <http://www.gnu.org/software/glpk/>.
- GUPTA, O. K. AND RAVINDRAN, A. 1985. Branch and bound experiments in convex nonlinear integer programming. *Management Science* 31, 1533–1546.
- HAJEK, B. 1988. Cooling schedules for optimal annealing. *Mathematics of Operations Research* 13, 311–329.
- HO, T., MEDARD, M., KOETTER, R., KARGER, D., EFFROS, M., SHI, J., AND LEONG, B. 2006. A random linear network coding approach to multicast. *Information Theory, IEEE Transactions on* 52, 10 (Oct.), 4413–4430.
- ISO/IEC. 1995. Generic coding of moving pictures and association audio information. *ISO/IEC*, 13818–2.
- ITU. 1998. Video coding for low bit rate communication. *ITU-T Recommendation H.263*.
- JAIN, K., MAHDIAN, M., AND SALAVATIPOUR, M. R. 2003. Packing Steiner Trees. In *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*.
- KAR, K., SARKAR, S., AND TASSIULAS, L. 2001. Optimization based rate control for multirate multicast sessions. In *Proceedings of IEEE INFOCOM*.
- KARP, R. 1972. Reducibility among combinatorial problems. Complexity of Computer Computations. *E. Miller and JW Thatcher, New York*.
- KIRKPATRICK, S., GELATT, C. D., AND VECCHI, M. P. 1983. Optimization by simulated annealing. *Science* 220, 4598, 671.
- KOETTER, R. AND MÉDARD, M. 2003. An Algebraic Approach to Network Coding. *IEEE/ACM Transactions on Networking* 11, 5 (October), 782–795.
- LI, B. AND LIU, J. 2003. Multirate video multicast over the internet: an overview. *IEEE Network* 17, 1, 24–29.
- LI, X., PAUL, S., PANCHAL, P., AND AMMAR, M. 1997. Layered video multicast with retransmission (LVMR): Evaluation of error recovery schemes. In *Proceedings of the Sixth International Workshop on Network and Operating System Support for Digital Audio and Video*.
- LI, Z. 2007. Min-Cost Multicast of Selfish Information Flows. In *Proceedings of IEEE INFOCOM*.
- LI, Z. AND LI, B. 2005. Efficient and Distributed Computation of Maximum Multicast Rates. In *Proceedings of IEEE INFOCOM*.
- LI, Z., LI, B., JIANG, D., AND LAU, L. C. 2005. On Achieving Optimal Throughput with Network Coding. In *Proceedings of IEEE INFOCOM*.
- LI, Z., LI, B., AND LAU, L. C. 2006. On Achieving Optimal Multicast Throughput in Undirected Networks. *IEEE Transactions on Information Theory* 52, 6 (June), 2467–2485.
- LUN, D. S., RATNAKAR, N., KOETTER, R., MÉDARD, M., AHMED, E., AND LEE, H. 2005. Achieving Minimum-cost Multicast: a Decentralized Approach Based on Network Coding. In *Proceedings of IEEE INFOCOM*.
- MCCANNE, S., JACOBSON, V., AND VETTERLI, M. 1996. Receiver-driven layered multicast. In *ACM SIGCOMM*. Vol. 26. 117–130.

- PAPADIMITRIOU, C. AND STEIGLITZ, K. 1998. *Combinatorial optimization: algorithms and complexity*. Dover Publications.
- QUESADA, I. AND GROSSMANN, I. 1992. An lp/nlp based branch and bound algorithm for convex minlp optimization problems. *Computers & chemical engineering* 16, 937–947.
- SACHAM, N. 1992. Multipoint communication by hierarchically encoded data. In *Proceedings of IEEE INFOCOM*.
- SUNDARAM, N., RAMANATHAN, P., AND BANERJEE, S. 2007. Multirate media stream using network coding. In *Proceedings of 43rd Annual Allerton Conference on Communication, Control, and Computing*.
- THIMM, M. 2001. On The Approximability Of The Steiner Tree Problem. In *Mathematical Foundations of Computer Science 2001, Springer LNCS 2136*, 678–689.
- WANG, W., LI, X.-Y., AND SUN, Z. 2005. Sharing the Multicast Payment Fairly. In *Proceedings of the 11th International Computing and Combinatorics Conference (COCOON)*.
- WU, Y., CHOU, P. A., ZHANG, Q., JAIN, K., ZHU, W., AND KUNG, S. Y. 2005. Network Planning in Wireless Ad Hoc Networks: A Cross-layer Approach. *Journal on Selected Areas in Communications (JSAC)* 23, 1 (January), 136–150.
- XI, C., XU, Y., ZHAN, C., WU, R., AND WANG, Q. 2007. On network coding based multirate video streaming in directed networks. In *Proceedings of IEEE Professional Communication Conference*.
- YUAN, J., LI, Z., YU, W., AND LI, B. 2006. A Cross-Layer Optimization Framework for Multihop Multicast in Wireless Mesh Networks. *Journal on Selected Areas in Communications (JSAC), Special Issue on Multi-hop Wireless Mesh Networks* 24, 11 (November), 2092–2103.
- ZHAO, J., YANG, F., ZHANG, Q., ZHANG, Z., AND ZHANG, F. 2006. Lion: Layered overlay multicast with network coding. *IEEE Transactions on Multimedia* 8, 1021.

Received XXX; XXX; accepted XXX