

# Optimal Layered Multicast with Network Coding: Mathematical Model and Empirical Studies

Ajay Gopinathan, Zongpeng Li  
Department of Computer Science  
University of Calgary  
{agopinat, zongpeng}@ucalgary.ca

**Abstract**—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 further generalize our model for 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.

## I. 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-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 [1]–[5]. Computing optimal multicast flows, however, was shown to be equivalent to packing Steiner trees [2], [6], which is NP-hard [7], [8]. However, recent breakthroughs in network coding research [9]–[12] 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 op-

timal 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 [13]–[16]. 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 [17], 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 [18], [19]. 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 [20], [21] 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 correct 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. I (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 receivers are allowed to receive layers only in entirety, the maximum throughput is

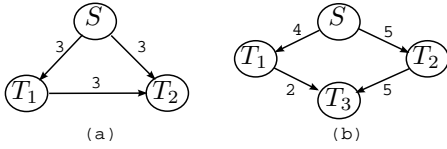


Fig. 1. In (a), layer sizes are (2,2,2). Receiver  $T_2$  is unable to achieve maximum throughput unless  $T_1$  receives and forwards partial layers. In (b), layer sizes are (2,3,2).  $T_3$  can only obtain all three layers if  $T_1$  receives and forwards non-continuous layers.

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 a correct optimal multicast algorithm should both (a) take into account the cumulative decoding nature of layered data, by screening out undecodable 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 [6], [11], [12]. 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 (lower cost) can be achieved, as compared to previous layered multicast models [20], [21] or one-layer only optimal multicast [11], [12].

We further study the related problem: how achievable multicast throughput depends upon the layer size progression. 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 [22] 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 II, we discuss previous research related to this work. We discuss our solution and mathematical program for optimal layered multicast in section III. In section IV, we seek to optimize layer sizes for layered multicast using simulated annealing. We then conclude our findings in section V.

## II. RELATED WORK

Traditional multicast algorithm design are mostly based on multicast trees or meshes that can be decomposed into a set of trees [1]–[5]. 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 [6], and are NP-hard [7], [8]. 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* [9], [10]. 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 [11], [12], [23]–[26]. 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 [13] studied optimal routing and error control mechanisms for layered coding with layered transmission rates. McCanne *et al.* [14] 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. In [15], the authors propose two distributed and scalable algorithms for achieving optimal rates that maximize receiver utility in multirate multicast. Li *et al.* [16] proposed a system for distribution of layer coded video over the Internet with a focus on improving reception quality, using efficient retransmission mechanisms. An overview of layered multicast for video applications on the Internet can also be found in [27].

In [28], the authors 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 [20], [21] and in this work.

Zhao *et al.* present a model for layered multicast in [20]. 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. Xi *et al.* proposed a similar model in [21], 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.

Non-linear integer programming is NP-hard in general. Previous solution methods include branch and bound [29] [30], outer approximation [31] and generalized bender's decomposition [32]. 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 [22]. 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 [33].

### III. 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.

#### A. Network Model and Notations

We model the network as a directed, capacitated graph. 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 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 [9], [10]. 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 [6]; 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.

#### B. 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  $\vec{T_i S}$  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  $\vec{T}_i S$  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  $f_k^i(\vec{T}_i S)$  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:

$$f_k^i(\vec{T}_i S) = l_k \cdot y_k^i \quad \forall i, \forall k \quad (5)$$

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:

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

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

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

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 [20], [21], 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  $x_{k+1}^i \leq x_k^i \leq \frac{f_k^i(\vec{T}_i S)}{l_k}, \forall i, \forall k \neq L$ .

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).

### C. 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) [34], 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. In all cases, 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 (glpk 4.16) [35].

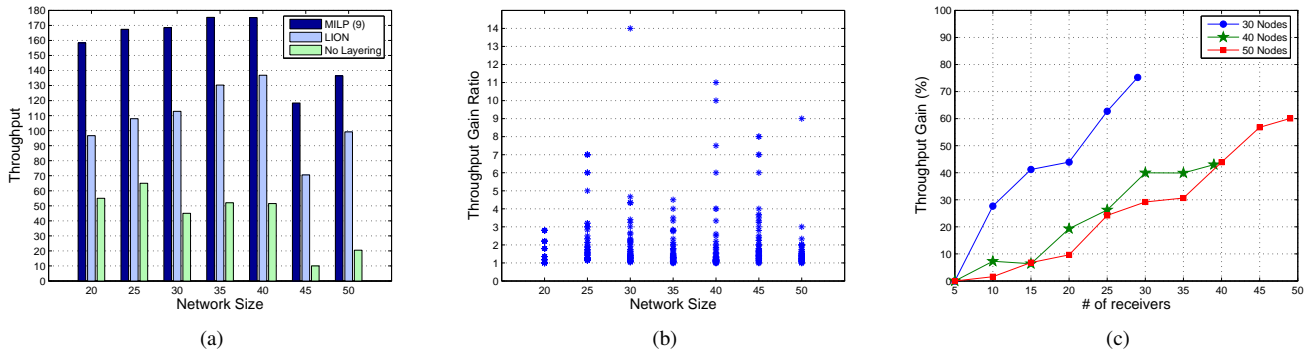


Fig. 2. The bar chart in (a) shows the average throughput obtained by our model as compared to LION and multicast without layering for varying network sizes and a fixed number of receivers. In (b), we plot the throughput gain ratio observed throughout our simulations, while (c) shows the throughput gain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

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. We denote by LION the model presented in [20], [21], while the “No-layering” model represents optimal single-rate multicast with network coding [6], [11]. 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. 2c, 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. 3 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.

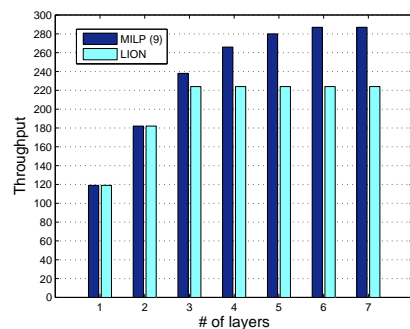


Fig. 3. Throughput obtained by our model as compared to LION with varying number of layers. As the number of layers increase, LION is unable to improve after 3 layers, while our model is able to more efficiently utilize bandwidth to route the extra data as the number of layers increases.

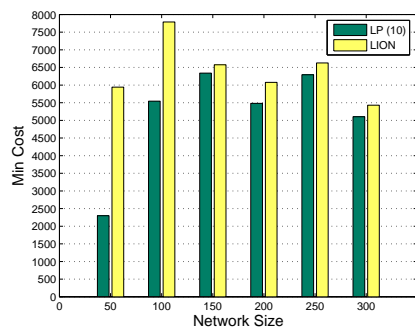


Fig. 4. Average min-cost for different sizes of networks as computed by our linear program in (10) as compared to the model in LION.

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:

$$\begin{aligned}
& \text{Minimize} && \sum_{\vec{uv}} \sum_k f_k(\vec{uv}) c(\vec{uv}) && (10) \\
& \text{Subject to:} && \\
& \left\{ \begin{array}{ll} \sum_{v \in N(u)} [f_k^i(\vec{uv}) - f_k^i(\vec{vu})] = 0 & \forall k, \forall i, \forall u \\ f_k^i(\vec{uv}) \leq f_k(\vec{uv}) & \forall k, \forall i, \forall \vec{uv} \\ \sum_k f_k(\vec{uv}) \leq C(\vec{uv}) & \forall \vec{uv} \\ x_k^i \leq \frac{f_k^i(T_i S)}{l_k} & \forall k, \forall i \end{array} \right. \\
& f_k(\vec{uv}), f_k^i(\vec{uv}) \geq 0 && \forall k, \forall i, \forall \vec{uv}
\end{aligned}$$

We examine the effect of our min-cost multicast formulation for layered multicast in Fig. 4. We generated topologies of various sizes, and assigned costs on each link randomly from the range  $[20, 40]$ . Each topology consists of 20 randomly 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.

#### IV. THE EFFECT OF LAYER SIZE PROGRESSION ON MULTICAST THROUGHPUT

In section III, 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 IV-B), 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 [29], [30], outer approximation [31] or generalized bender's decomposition [32]. These methods are able to guarantee optimality only under generalized convexity. Additionally, they are only capable of handling small problem sizes. Fig. 5 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

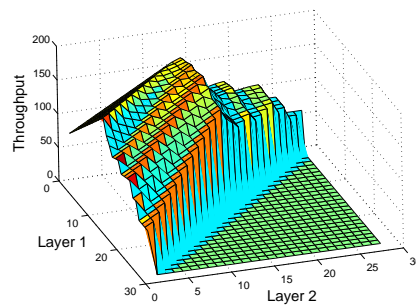


Fig. 5. This figure plots the throughput obtained by exhaustively enumerating all possible layer size progressions for a 3-layer multicast in a network with 50 nodes, 20 receivers. The solutions obtained clearly illustrate the non-convex nature of the optimization of layer size progression.

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 [22] is loosely based on the idea of 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 “up-hill” 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.

#### A. 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,  $f(1)$  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

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

At each iteration, a *neighbour* solution is generated by introducing small and random changes to the 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*,  $\tau$ :

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

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  $\tau$  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  $\tau$ , the *cooling schedule* that determines how  $\tau$  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  $\tau$  slowly results in more accurate solutions, but increases the runtime of the algorithm; decreasing  $\tau$  too quickly could result in sub-optimal solutions. For our simulations, we chose an exponentially decreasing cooling schedule

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

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

The effectiveness of the cooling schedule also depends on the initial temperature,  $\tau_0$ . During the early stages of the algorithm, we would like some probability  $p_0$  of accepting

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

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,  $\delta f$ , 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.

## B. Simulation Results and Discussion

Having designed our simulated annealing algorithm, we now investigate how well the throughout 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 [34], 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

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

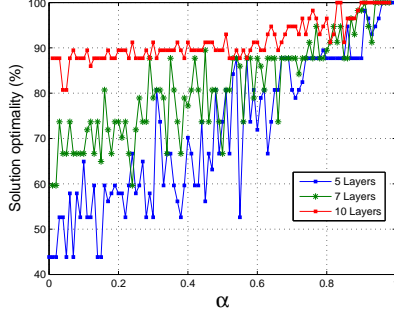


Fig. 6. The value of  $\alpha$  can be tuned to obtain a tradeoff between optimality and running time

process, we plot different values of the parameter  $\alpha$  against the optimality of the solution obtained, for a network of 50 nodes using 5, 7 and 10 layers respectively, in Fig. 6. The values of  $\alpha$  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  $\alpha$  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  $\alpha$  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. 6. As the number of layers increases, the solution for a given value of  $\alpha$  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  $\alpha$ , the solution obtained is not too far from optimality.

Fig. 7 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. 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

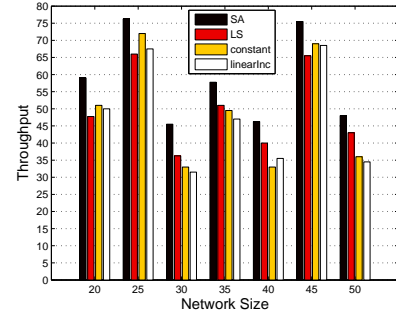


Fig. 7. Simulated annealing obtains higher throughput than local search or fixed layer size progressions for varying network sizes

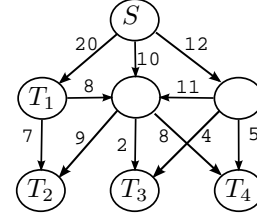


Fig. 8. The cumulative sum of the optimal layer sizes obtained for this network match the receiving capacities in the multicast group

size configurations and the receiving capacity of nodes in the multicast group. Fig. 8 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 9 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.

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

TABLE III  
OPTIMAL LAYER SIZE PROGRESSION VS. DISTINCT RECEIVER CAPACITIES

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.

## V. 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 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. Such insight may serve as a valuable insight in the future design of layered coding schemes.

## REFERENCES

- [1] J. Feigenbaum, C. Papadimitriou, and S. Shenker, "Sharing the Cost of Multicast Transmissions," *Journal of Computer and System Sciences*, vol. 63, pp. 21–41, 2001.
- [2] S. Chen, O. Günlük, and B. Yener, "The Multicast Packing Problem," *IEEE/ACM Transactions on Networking*, vol. 8, no. 3, pp. 311–318, 2000.
- [3] W. Wang, X.-Y. Li, and Z. Sun, "Sharing the Multicast Payment Fairly," in *Proceedings of the 11th International Computing and Combinatorics Conference (COCOON)*, 2005.
- [4] N. Garg, R. Khandekar, K. Kunal, and V. Pandit, "Bandwidth Maximization in Multicasting," in *Proceedings of the 11th European Symposium on Algorithms (ESA)*, 2003.
- [5] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable Application Layer Multicast," in *Proc. of ACM SIGCOMM*, August 2002.
- [6] Z. Li, B. Li, and L. C. Lau, "On Achieving Optimal Multicast Throughput in Undirected Networks," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2467–2485, June 2006.
- [7] 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.
- [8] M. Thimm, "On The Approximability Of The Steiner Tree Problem," in *Mathematical Foundations of Computer Science 2001, Springer LNCS 2136, 678-689*, 2001.
- [9] 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.
- [10] 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.
- [11] Z. Li, B. Li, D. Jiang, and L. C. Lau, "On Achieving Optimal Throughput with Network Coding," in *Proceedings of IEEE INFOCOM*, 2005.
- [12] D. S. 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.
- [13] N. Sacham, "Multipoint communication by hierarchically encoded data," in *Proceedings of IEEE INFOCOM*, 1992.
- [14] S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-driven layered multicast," in *ACM SIGCOMM*, vol. 26, 1996, pp. 117–130.
- [15] K. Kar, S. Sarkar, and L. Tassiulas, "Optimization based rate control for multirate multicast sessions," in *Proceedings of IEEE INFOCOM*, 2001.
- [16] X. Li, S. Paul, P. Pancha, and M. Ammar, "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*, 1997.
- [17] A. A. E. Gamal and T. M. Cover, "Achievable rates for multiple descriptions," *IEEE Trans. Inform. Theory*, vol. 28, pp. 851–857, 1982.
- [18] ISO/IEC, "Generic coding of moving pictures and association audio information," *ISO/IEC*, pp. 13 818–2, 1995.
- [19] ITU, "Video coding for low bit rate communication," *ITU-T Recommendation H.263*, 1998.
- [20] J. Zhao, F. Yang, Q. Zhang, Z. Zhang, and F. Zhang, "Lion: Layered overlay multicast with network coding," *IEEE Transactions on Multimedia*, vol. 8, p. 1021, 2006.
- [21] X. Chenguang, X. Yinlong, Z. Cheng, W. Ruizhe, and W. Qingshan, "On network coding based multirate video streaming in directed networks," in *Proceedings of IEEE Professional Communication Conference*, 2007.
- [22] S. Kirkpatrick, C. D. G. Jr, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, p. 671, 1983.
- [23] Z. Li and B. Li, "Efficient and Distributed Computation of Maximum Multicast Rates," in *Proceedings of IEEE INFOCOM*, 2005.
- [24] Y. Wu, P. A. Chou, Q. Zhang, K. Jain, W. Zhu, and S. Y. Kung, "Network Planning in Wireless Ad Hoc Networks: A Cross-layer Approach," *Journal on Selected Areas in Communications (JSAC)*, vol. 23, no. 1, pp. 136–150, January 2005.
- [25] J. Yuan, Z. Li, W. Yu, and B. Li, "A Cross-Layer Optimization Framework for Multihop Multicast in Wireless Mesh Networks," *Journal on Selected Areas in Communications (JSAC), Special Issue on Multihop Wireless Mesh Networks*, vol. 24, no. 11, pp. 2092–2103, November 2006.
- [26] Z. Li, "Min-Cost Multicast of Selfish Information Flows," in *Proceedings of IEEE INFOCOM*, 2007.
- [27] B. Li and J. Liu, "Multirate video multicast over the internet: an overview," *IEEE Network*, vol. 17, no. 1, pp. 24–29, 2003.
- [28] N. Sundaram, P. Ramanathan, and S. Banerjee, "Multirate media stream using network coding," in *Proceedings of 43rd Annual Allerton Conference on Communication, Control, and Computing*.
- [29] I. Quesada and I. Grossmann, "An lp/nlp based branch and bound algorithm for convex minlp optimization problems," *Computers & chemical engineering*, vol. 16, pp. 937–947, 1992.
- [30] O. K. Gupta and A. Ravindran, "Branch and bound experiments in convex nonlinear integer programming," *Management Science*, vol. 31, pp. 1533–1546, 1985.
- [31] R. Fletcher and S. Leyffer, "Solving mixed integer nonlinear programs by outer approximation," *Mathematical Programming*, vol. 66, pp. 327–349, 1994.
- [32] A. M. Geoffrion, "Generalized benders decomposition," *Journal of Optimization Theory and Applications*, vol. 10, pp. 237–260, 1972.
- [33] B. Hajek, "Cooling schedules for optimal annealing," *Mathematics of Operations Research*, vol. 13, pp. 311–329, 1988.
- [34] "Boston University Representative Internet Topology generator," <http://www.cs.bu.edu/brite/>.
- [35] "GNU Linear Programming Kit," <http://www.gnu.org/software/glpk/>.