LION: Layered Overlay Multicast with Network Coding

Jin Zhao*[1], Fan Yang[2], Qian Zhang[2], Zhensheng Zhang[3], Fuyan Zhang[1]

[1]Dept. of Computer Science, Nanjing University, Nanjing, China

[2]Microsoft Research Asia, Beijing, China

[3]San Diego Research Center, San Diego, CA, USA

Abstract

Recent advances in information theory show that the throughput of a multicast session can be improved using network coding. In overlay networks, the available bandwidth between sender and different receivers are different. In this paper, we propose a solution to improve the throughput of an overlay multicast session with heterogeneous receivers by organizing the receivers into layered data distribution meshes and sending substreams to each mesh using layered coding. Our solutions utilize alternative paths and network coding in each mesh. We first formulate the problem into a mathematical programming, whose optimal solution requires global information. We therefore present a distributed heuristic algorithm. The heuristic progressively organizes the receivers into layered meshes. Each receiver can subscribe to a proper number of meshes to maximize its throughput by fully utilizing its available bandwidth. The benefits of organizing the topology into layered mesh and using network coding are demonstrated through extensive simulations. Numerical results indicate that the average throughput of a multicast session is significantly improved (up to 50% to 60%).

Keywords

Overlay multicast, network coding, heterogeneity

1. INTRODUCTION

Recently, overlay multicast, or end system multicast, has received many research interests [1-4]. Instead of relying on the IP multicast support in routers, an end host can act as "router" to forward data to other end hosts.

The challenge to improve multicast throughput lies in the observation that the available bandwidth between end hosts are heterogeneous. A measurement study in peer-to-peer overlay networks [16] reveals that the bottleneck bandwidth between the end hosts exhibits extremely heterogeneity. To handle heterogeneity, layered multicast is proposed in IP multicast [13] and overlay multicast [14], respectively. The basic idea is to encode source data into several layers. Thus a receiver can subscribe to a proper number of layers to maximize the throughput.

*Work performed when Jin was a visiting student at Microsoft Research Asia.

A significant amount of research efforts has been directed toward overlay multicast throughput improvement. Existing works can be roughly classified into single-path and multi-path schemes. Many of the existing works have advocated building single data distribution tree rooted at sender. Therefore each receiver has only one path from the sender along the tree. Narada [1], which constructs a spanning tree for data delivery on the initial mesh structure, is a typical single-path scheme. Recent Internet measurement [17] shows the redundant IP routes between hosts are quite common. Study in overlay networks [18] also demonstrates that a better path other than the default one may exist. The throughput attained by single multicast tree is therefore suboptimal and can be improved.

Multi-path schemes construct multiple paths between sender and each receiver and use multi-stream coding or erasure coding to address heterogeneity. Both CoopNet [2] and SplitStream [3] build multiple trees among overlay nodes and send data to each tree using multiple description coding. Each receiver has multiple paths from sender along different trees. More recently, oEvolve [4] is proposed to improve the receiver's throughput by measuring each receiver's available bandwidth periodically. The scheme dynamically adds new trees spanning the receivers which have residual bandwidth. Receivers with residual available bandwidth can improve throughput by joining more trees.

The schemes mentioned above assume that all the nodes in overlay network are receivers. Recent advances in network coding [5] showed that with the presence of relay nodes, the multicast throughput can be further improved by allowing coding operation at intermediate nodes in the network. Notably, Zhu et al. [8] apply network coding to overlay network to improve throughput. By constructing a 2-reduandant multicast graph, each receiver has two disjoint paths from the sender. However, the scheme needs to synchronize the rate of each receiver's two paths to be exactly same. In addition, the rates in all the paths that share some common links also have to be synchronized to the rate in the bottleneck link. Furthermore, they do not address the heterogeneity issue. Chou et al. proposed a practical network coding scheme [12]. In their work, each node randomly codes the incoming packets and floods them to all the neighbors. They also utilize the Priority Encoding Transmission (PET) technique to provide loss protection at the expense of introducing redundancy. However, flooding packets among nodes will generate much unnecessary data that results in performance degradation.

In this paper, we propose LION, a layered overlay multicast framework with network coding to address end system heterogeneity. As in layered IP multicast, we assume the sender can provide layered data at different rates. We seek to improve the multicast session throughput by organizing the heterogeneous receivers into layered meshes and using network coding in each mesh. We first formulate the problem using mathematical programming. As obtaining an optimal solution requires global information, we also propose a distributed heuristic algorithm to approximate the optimal solution. Unlike earlier works which build single or multiple data distribution trees, LION fully leverages multi-path property in a network and builds multiple data distribution meshes. A receiver subscribes to a proper number of meshes to maximally utilize its available bandwidth. A source data layer is sent to a corresponding mesh. In each mesh, a receiver has multiple paths to receive the source data. *This uniqueness in data distribution mesh distinguishes LION from existing*

works in layered multicast. The challenge in constructing layered mesh lies in how to build lower layer meshes to maximally utilize the advantage of network coding and to leave more residual bandwidth for higher layer meshes. Existing mesh construction methods can not be directly applied to the layered overlay multicast to achieve the above goal. In this paper we propose a path-overlapping method that takes advantage of network coding to address the problem. Simulation results show that the proposed heuristic of layered overlay multicast handles heterogeneity very well and improves multicast session's throughput up to 50% to 60%.

The rest of this paper is organized as follows. Section 2 gives some insights of the motivation in layered multicast with network coding. Section 3 formulates the problem and presents a distributed heuristic approach to solve the resultant mathematical programming. In section 4, we extend the heuristic approach to overlay multicast. In section 5, we evaluate our heuristic through simulations which show that our heuristic is quite encouraging. We conclude this paper in section 6.

2. BACKGROUND AND MOTIVATION

The maximal achievable throughput of a multicast session has been considered as a fundamental problem for many years in graph theory. Menger [19] proved that the maximal unicast capacity from a sender to a receiver equals the minimum cut capacity, or min-cut, separating the sender from the receiver. A cut of a graph is an edge set which partitions the graph into two parts. The capacity of a cut is the sum of edge capacities in the cut. Furthermore, Ford and Fulkerson [20] developed an efficient algorithm to find the min-cut. When all nodes in the network are receivers (except the sender), Edmonds [21] proved that the maximal broadcast capacity is the minimal min-cut among all receivers. However, when there exist Steiner nodes, which are not multicast receivers and only act as data relay, finding the maximal multicast capacity is NP-hard [22][23]. In their seminal work, Ahlswede et al.[5] proved that by means of network coding, the minimal min-cut of a multicast session is also achievable even if there are Steiner nodes. Ideally, a sender can multicast to all receivers at the rate of minimal min-cut among the receivers. Li et al. [6] and Koetter et al. [7] further proved that linear network coding is enough to achieve the capacity. Consider the network in Fig. 1(b). Node S encodes the data (a1+b1) and sends to R3. Both receiver T2 and T3 can receive a1, b1 from sender simultaneously. Operation "+" refers to operation over Galois Field.

With network coding the throughput of single-rate multicast can achieve the minimal min-cut of all receivers. However, when the receivers are heterogeneous, single-rate multicast for all receivers is not efficient. In case of multi-rate multicast, network coding alone may be not enough. Adding layers to receivers with higher min-cut may further improve the throughput. In this paper we seek to maximize the throughput of a multicast session with heterogeneous receivers. The problem can be regarded as multi-rate multicast using network coding.

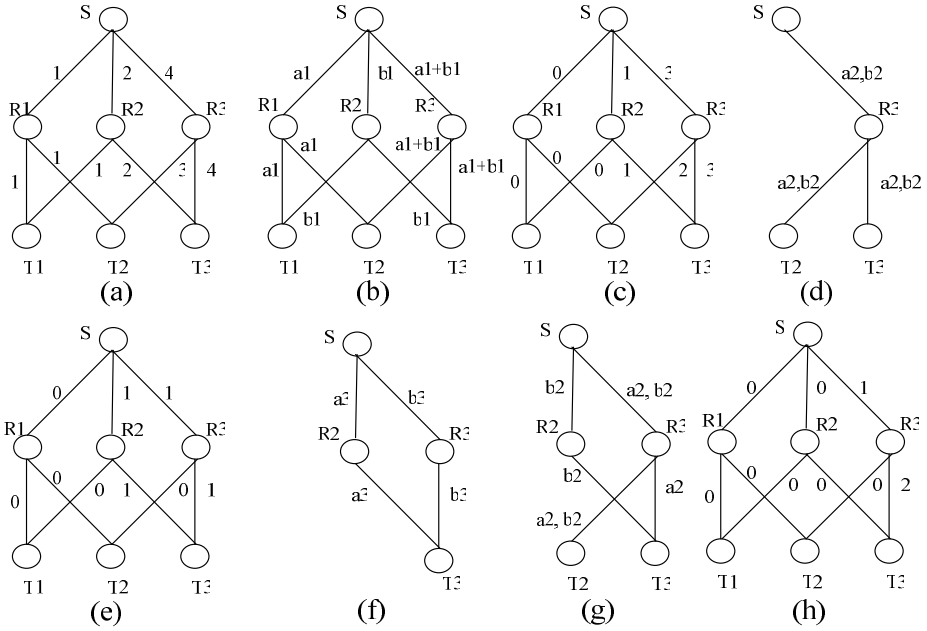


Fig.1 An example of adding layers to multicast session.

We use an example in Fig. 1 to illustrate our conjecture. Consider the network in Fig 1 (a) with S as the sender, T1,T2, T3 as receivers and R1,R2,R3 as Steiner nodes, i.e., relays. The number on each link is the link's available bandwidth. Suppose we have encoded the source data into 3 layers. We further encode each layer into small-grained stripes. Each stripe has unit bit-rate. Layer 1's stripes are (a1,b1), layer 2's stripes are (a2,b2) and layer 3's stripes are (a3,b3). Therefore each layer has 2 unit bit rate. T1 has min-cut 2, T2 has min-cut 4, T3 has min-cut 6 from sender, respectively. With single-rate multicast, the maximal through-put is 2. T2 and T3's available bandwidth can not be fully utilized. If we properly organize the receivers into layers, T2 and T3's throughput can be further improved. Layer 1's data distribution mesh is illustrated in Fig 1 (b). With *network coding*, T1, T2 and T3 all can receive layer 1's content. The residual bandwidth after removing layer 1's bandwidth is shown in Fig 1(c). Both T2 and T3 have residual bandwidth and therefore can join layer 2. Only T3 can join layer 3. The data distribution meshes of each layer are shown in Fig. 1 (b), (d), and (f), respectively. Receivers T2 and T3's throughputs are improved by joining additional higher meshes. Here whether a receiver can join another higher layer depends on how the current layer is constructed. For example, after constructing layer 1 (Fig. 1(b)), if layer 2 is constructed as in Fig 1. (g), then the remaining bandwidth is not enough to construct layer 3 as shown in Fig 1(h).

Therefore, when we construct lower layers, while we first need to select enough paths for each receiver, we shall also need to leave as much residual bandwidth as possible for higher layers. An intuition is that we need encourage the paths for different receivers overlap. *The overlapped links can serve more receivers simultaneously using network coding* (Fig. 1(b)). In Fig 1(g) the paths for receiver T2 and T3 overlap only once at link S-R3. However, the paths for T2 and T3 overlap twice at link S-R3 in Fig. 1 (d) and (f).

In next section, we formulate the problem into a mathematical programming based on these observations.

3. PROBLEM FORMULATION

3.1. Notations

Suppose the source data are encoded into L layers $\{l_1, l_2, ..., l_L\}$. Layer i has bit-rate B_i. Layer i can be decoded only when layer 1 to layer i-1 are all available.

We model the network as a directed graph $G(V, E)$ where V is the set of nodes and E is the set of edges. There are 3 disjoint node sets in V, namely, S, R and T, which denote sender, relays and receivers, respectively. $|S|=1$, $|R|=N_R$, $|T|=N_T$, $V=S \bigcup R \bigcup T$, $T=\{t_1, t_2, \ldots, t_{N_T}\}$. Suppose there are total M edges in the network, $E=\{e_1, e_2, \ldots, e_M\}$, $|E|=M$. The available bandwidth of edge e_m is C_m. A multicast session is identified by a tuple (S, T). Receivers with different available bandwidth from S can receive different number of layers.

We denote each receiver's layer subscription as matrix Z.

$$z_{k,i} = \begin{cases} 1, & receiver\ i\ can\ \ receive\ layer\ k \\ 0, & otherwise \end{cases}, \quad 1 \le i \le N_T, 1 \le k \le L. \tag{1}$$

Suppose receiver t_i has N_i possible paths from the sender, the set of these paths is denoted by

$$Path(t_i) = \{P(t_i,1), P(t_i,2), \ldots, P(t_i, N_i)\}. \tag{2}$$

As a path consists of consecutive edges, matrix $X(t_i)$ is used to denote the edges which are included in t_i's paths.

$$x_{j,m}(t_i) = \begin{cases} 1, if\ \ e_m \in P(t_i, j) \\ 0, otherwise \end{cases}, 1 \le i \le N_T, 1 \le m \le M, 1 \le j \le N_i. \tag{3}$$

Note that the elements in matrix X are constants. They are determined once all the paths from sender to receivers in the directed graph $G(V, E)$ are specified.

We use matrix $Y(t_i)$ to denote the assigned flow rate on t_i's paths in all layers. Element $y_{k,j}(t_i)$ in $Y(t_i)$ denotes receiver t_i's flow rate of path j in layer k.

$$y_{k,j}(t_i) \ge 0, \ 1 \le k \le L, 1 \le j \le N_i \tag{4}$$

We use matrix U to denote the consumed bandwidth on all edges in all the layers. An element $u_{k,m}$ of U denotes the consumed bandwidth of edge m in layer k.

$$u_{k,m} \ge 0, 1 \le k \le K, 1 \le m \le M \tag{5}$$

Note that U defines the data distribution meshes for all the layers.

3.2. Multi-layer Formulation

Our objective is to maximize the multicast session's throughput. In our setup, it is to maximize the total bit-rate of all receivers' subscription. Each receiver joins layers in an incremental order. Before receiver t_i joins

layer k, it needs to join all the lower layers (1 to *k*-1) first. If t_i*'s* paths have additional available bandwidth that can support B_k, it can join layer k. Network coding guarantees that all receivers which joined layer *k* can have bit-rate B_k since each receiver in layer *k* has available bandwidth B_k from the sender [5]. The data of layer *k* are distributed in a mesh rather than a tree since each receiver may have multiple paths from the sender. Here we specify that network coding is only allowed within the same layer. Though it is possible to code the data from different layers, combining data belonging to different layers makes it difficult to recover all original data for receivers that only receive partial layers.

The problem can now be formulated as the following mathematical programming.

$$\text{Maximize} \sum_{i=1}^{N_T} \sum_{k=1}^{L} B_k \cdot z_{k,i} \tag{6}$$

Subject to:

1) $\sum_{j=1}^{N_i} y_{k,j}(t_i) = B_k \cdot z_{k,i},\ 1 \le i \le N_T, 1 \le k \le L$

2) $z_{k,i} \ge z_{k+1,i},\ 1 \le k \le L-1, 1 \le i \le N_T$

3) $u_{k,m} = \max_{1 \le i \le N_T} \left\{ \sum_{j=1}^{N_i} \left[x_{j,m}(t_i) \cdot y_{k,j}(t_i) \right] \right\}, 1 \le m \le M, 1 \le k \le L$

4) $\sum_{k=1}^{L} u_{k,m} \le C_m,\ 1 \le m \le M$

Constraints 1) ensure that receiver t_i must assign total B_k bandwidth on all its paths for layer *k* if $z_{k,i}$=1. Constraints 2) ensure that each receiver subscribes to layers in an incremental order, layer *k*+1 is not decodable without any layer less than *k*+1. Constraints 3) specify the required flow rate on each edge for each layer. Receiver t_i's bandwidth consumption for layer *k* on edge m is the sum of the specified layer-*k* flow rate on t_i's all paths which pass edge *m*. With network coding, different receivers will not compete for edge bandwidth within one layer, therefore the required flow rate on edge m for layer k is equal to the largest bandwidth on edge *m* consumed among all the receivers in layer *k*. Constraints 4) ensure that an edge can support all the flows on all layers. Matrix *U* forms the data distribution meshes. *U*'s column k denotes the mesh for layer *k*.

When we consider heterogeneous receivers, fairness should be also considered. There may exist cases where one receiver's throughput is maximized at the cost of other receivers' starvation. We introduce a weight value, w_k, on layer *k*, (*k*=1, .., *L*) to ensure certain fairness that the available bandwidths are allocated to receivers which subscribe to lower layers first. Recall that N_T denotes the number of receivers. To ensure fairness, the weight value of each layer should satisfy:

$$w_k > (N_T - 1) \sum_{i=k+1}^{L} w_i,\ 1 \le k \le L \tag{7}$$

Suppose layer L (i.e., highest layer) has the lowest weight 1, we determine the value of weight recursively.

$$w_k = (N_T)^{L-k} \tag{8}$$

Note that other values of w_k can be used as well.

Then the objective in (6) can be rewritten as

$$\text{Maximize} \sum_{i=1}^{N_T} \sum_{k=1}^{L} w_k \cdot z_{k,i} , \tag{9}$$

to ensure fairness.

The above formulation considers all receivers and all layers simultaneously. Therefore it can achieve optimal value. However, the problem is a nonlinear optimization (NLP), which is hard to solve. By changing the form of *max* function in constraints 3), we can transform the problem into a linear one as defined in (10).

$$\text{Maximize} \sum_{i=1}^{N_T} \sum_{k=1}^{L} w_k \cdot z_{k,i} \tag{10}$$

Subject to:

1) $\sum_{j=1}^{N_i} y_{k,j}(t_i) = B_k \cdot z_{k,i} \ , 1 \le i \le N_T , 1 \le k \le L$

2) $z_{k,i} \ge z_{k+1,i} , 1 \le k \le L-1, 1 \le i \le N_T$

3) $\sum_{j=1}^{N_i} (x_{j,m}(t_i) \cdot y_{k,j}(t_i)) \le u_{k,m} , 1 \le m \le M , 1 \le k \le L , 1 \le i \le N_T$

4) $\sum_{k=1}^{L} u_{k,m} \le C_m , \ 1 \le m \le M$

As we can see in constraints 3) in (10), the *max* function is transformed into N_T-1 more inequations. The problem now becomes an integer linear programming (ILP). We refer to the problem as m-Layer problem. Though linear, the problem requires global information on all receivers and all layers to get the optimal value, which is not practical for overlay multicast.

3.3. Layer Decomposition

As m-Layer problem is computationally intractable, to reduce computation complexity, we decompose the problem into multiple one-layer problems, i.e., solve the optimization problem within each layer in isolation. In this case constraints 2) disappear and constraints 4) can be merged into constraints 3), the optimization problem in layer *k* can be formulated as:

$$\text{Maximize} \sum_{i=1}^{N_T} w_k \cdot z_{k,i}$$

$$\text{subject to}:$$

$$1) \sum_{j=1}^{N_i} y_{k,j}(t_i) = B_k \cdot z_{k,i}\ , 1 \le i \le N_T \qquad (11)$$

$$2) \sum_{j=1}^{N_i} (x_{j,m}(t_i) \cdot y_{k,j}(t_i)) \le C_m, 1 \le m \le M, 1 \le i \le N_T$$

Note that (11) is also an ILP. We construct the mesh for each layer iteratively to ensure more receivers join lower layer first. We first solve the problem in (11) by letting k=1, i.e., let receivers first join layer 1. After the flows are assigned on each edge in layer 1, we update the available bandwidth on all the edges. Edge with no available bandwidth will be removed. The paths containing these removed edges will be deleted too. We also remove the receivers which can not receive layer 1 if any. Then we solve the resultant ILP (for k=2) again and obtain a mesh for layer 2. We iteratively use the approximation until there are no receivers in the residual graph. As each receiver adds layers in an incremental order, adding higher layer not only does not cause lower layer's quality decrease but also increases throughput for receivers with available bandwidth, therefore fairness is satisfied.

However, there may be multiple solutions to each ILP. Among those multiple solutions, we choose the one with minimal consumed edge bandwidth, which will maximize the available bandwidth as well as the connectivity in the residual network. More formally, suppose we have N possible solutions to (11) for layer k, we choose the j_0th solution, where j_0 is given by:

$$j_0 = \arg\min_{1 \le j \le N} \left\{ \sum_{m=1}^{M} u_{k,m}(j) \right\}. \qquad (12)$$

Recall that $u_{k,m}(j)$ is the consumed bandwidth of edge m in layer k in the jth solution (among the N solutions) to (11).

We refer to the decomposed ILP as 1-Layer problem. The 1-Layer problem still needs to consider all receivers simultaneously such that the consumed edge bandwidth is minimal. In the remaining of the paper, we describe distributed heuristics to obtain the solutions.

3.4. Heuristic Approach

As stated in the previous section, to obtain the optimal solution even within one layer we still need to coordinate all receivers, which is still not practical. Therefore we propose a heuristic algorithm which can be implemented in a distributed manner to approximate the optimal solution. The heuristic not only considers each layer in isolation, but also considers each receiver in isolation. The receivers join layers in an incremental order. The basic idea of the heuristic is to encourage overlapping among the paths of different receivers. During the construction of data distribution mesh for a layer, each receiver selects paths independ-

ently but tries to select as many paths as possible such that the probability of overlapping with other receiver's paths is high so that the network coding can be applied.

Each receiver t_i first runs the Ford-Fulkerson algorithm [20] to find the maxflow (maximum achievable flow rate) from the sender as well as the flow rate on the paths to achieve the maxflow. We denote the obtained path set as *maxflow paths* $MFP(t_i)$. Suppose there are total $N_P(t_i)$ paths in t_i's *maxflow paths*. The flow rate assigned to path j should be $MF_j(t_i)$ in order to achieve the maxflow. Assume that layer k's bit-rate is B_k. The receivers with maxflow larger than or equal to B_k will join layer k. The basic idea of our heuristic in constructing data distribution mesh for layer k is that it tries to evenly assign the flow of layer k on all the maxflow paths. In this way, the probability that a receiver's data paths overlap with other receivers' paths can be maximized. Therefore each path in $MFP(t_i)$ can approximately carry $B_k/N_P(t_i)$ flow. However, paths with $MF_j(t_i)$ less than $B_k/N_P(t_i)$ may carry less flows while paths with $MF_j(t_i)$ larger than $B_k/N_P(t_i)$ will be allocated more flows. Suppose path j is assigned with $y_j(t_i)$ flow rate. The pseudo-code of flow rate assignment in a layer is summarized in Fig. 2.

```
if ∑_{j=1}^{N_P(t_i)} MF_j(t_i) < B_k   /* maxflow paths can not afford layer k */
        stop adding layer k and the algorithm terminates
else
{
        for each path j in MFP(t_i)
            y_j(t_i)=0;
        ResidualPath(t_i)=MFP(t_i); /*ResidualPath(t_i) denotes the paths with
MF_j(t_i)>0 */
        while ∑_{j=1}^{N_P(t_i)} y_j(t_i) < B_k
        {
            pathnum=|ResidualPath(t_i)|;
            minflow= min{MF_j(t_i)} in ResidualPath(t_i);
            unassigned = B_k − ∑_{j=1}^{N_P(t_i)} y_j   /*unassigned flow */
            if (minflow*pathnum>=unassigned) /*flow increment in this round */
                  increment = unassinged / pathnum
            else
                  increment=minflow;
            for each path j in ResidualPath(t_i)
            {
                  y_j(ti)=y_j(ti)+increment;
                  MF_j(ti)=MF_j(ti)- increment;
                  if (MF_i(ti)<=0)
                      Remove path j from ResidualPath(ti);
            }
        }
}
```

Fig. 2. Pseudo-code for flow rate assignment in layer k

The calculation of the required flow rate on each edge e_m is similar to the constraints 3) in equation (6). Each receiver's assignment on an edge is the sum of the assigned flow of its all paths which pass the edge. Suppose receiver t_i ($1 \le i \le N_T$) assigned a flow rate of $edgeflow_{k,m}(t_i)$ edge m ($1 \le m \le M$) for layer k.

The required flow rate on edge m for layer k is

$$u_{k,m} = \max_{1 \le i \le N_T} \{edgeflow_{k,m}(t_i)\}. \quad (13)$$

The edges with required flows form the data distribution mesh for layer k. After layer k is constructed, each receiver updates the available bandwidth of the maxflow paths. Iteratively, each receiver uses the heuristic again to construct layer k+1.

We will present the numeric results for the throughput obtained using m-Layer, 1-Layer and the heuristic approaches and discuss the throughput gaps among the three approaches in section 5.

4. Distributed Approach to Overlay Multicast

The challenge in applying network coding to current Internet lies in that routers usually do not support additional coding operation. As end hosts in overlay networks can provide additional computing capabilities and storage, which is potentially suitable for network coding, we also extend our heuristic to overlay multicast. In this section, we present LION, a layer distributed implementation of the heuristic approach in overlay network.

4.1. Basic Overlay Network Construction

We follow the approach used in [8] to construct the basic overlay network. We do not focus on how to construct the overlay network. Instead, we use existing bootstrapping techniques, e.g. Narada, to form a well-constructed basic overlay network. Suppose the overlay network is relatively densely connected so that there exist multiple paths between any nodes. We build layered meshes on top of the basic overlay network. The links in the basic overlay network have two weights (B, D), representing available bandwidth and delay, respectively. Each end host periodically probes other non-neighbors and measures (B, D) to see if new links can be added. If (B, D) satisfy a pre-defined threshold, the link is added, otherwise, the link is dropped. As in [8], we assume there are dedicated pure relay nodes in the overlay network. We also impose a node degree constraint on each relay node such that the number of links passing a node is limited, by which we restrict the *stress* of the overlay network [8].

The construction of data distribution mesh within each layer is carried in two steps: selecting path and reserving path. We first use flooding to find all possible paths for each receiver, and then each receiver selects a number of paths to construct a data distribution mesh. Each receiver then sends a request back to sender to reserve the selected paths and assign the amount of stripes along each path. As the mesh is constructed, each receiver update all the (B, D) of its paths.

By iteratively repeating the above approach, we can construct layered meshes on top of the basic overlay network.

4.2. Finding Path

Before constructing layered meshes, each receiver needs to exploit multiple paths from the sender which are potentially suited for network coding. *Finding edge-disjoint paths for each sender-receiver pair and select-*

ing edge-overlap paths among different receivers can increase the possibility for network coding and leave more available bandwidth for next layer. We use a greedy method to find disjoint paths for each receiver. The sender floods a Finding Path packet, *FP*, to all its neighbors. When a relay node *R*1 receives a *FP* packet from relay node *R*2, it first checks if its ID is already included in the packet. If so, node *R*1 drops the packet to avoid loop. Otherwise, it appends its node ID and the last hop link weights (*B*, *D*) to the *FP* packet and sends the revised *FP* packet to all other neighbors excluding *R*2. A node may receive multiple *FP* packets from different incoming links. The *FP* packets are terminated at receiver nodes. Each *FP* packet represents a path from the sender to the receiver. A receiver just checks the *FP* packets and records them as *available paths.*

Some optimization can be enforced to further reduce flooding traffic. We can define a delay and bandwidth threshold. If an *FP* packet exceeds the delay threshold, it will be discarded. If a link's available bandwidth does not meet the minimal requirement, the path will also be excluded.

The pseudo-code for finding path phase is summarized in Fig. 3.

```
Sender S:
        Send FP to all neighbors
Upon receiving a FP, Relay R:
        Check if it is included in FP and if the last hop link satisfies the delay and bandwidth requirements
        if FP satisfied all requirements
                forwards FP to all neighbors
        else
                drop FP
Upon receiving a FP, Receiver T:
        Record the path
        Discard FP
```

Fig. 3. Pseudo-code for finding path using flooding

4.3. Constructing Layered Mesh

As different overlay links may traverse a same IP link, it is not practical to directly apply the Ford-Fulkerson algorithm to overlay network to find the maxflow for a receiver. Since each receiver needs to select enough paths to join a layer and tries to have as many overlapping paths with other receivers as possible, we use a heuristic that each receiver selects the maximal number of disjoint paths from the discovered available paths. By temporarily setting each overlay link to the same bandwidth, finding maximal number of disjoint paths is equivalent to finding maximal capacity in the available paths. Therefore, each receiver can use the Ford-Fulkerson algorithm to obtain the maximal number of disjoint paths. We denote the obtained path set as *max-disjoint paths*. Suppose there are N_{MD} disjoint paths in *max-disjoint paths*. We further packetize the source data into small-grained stripes. Suppose layer k has K_k stripes, therefore the bit-rate of a stripe is B_k/K_k.

Suppose path j has available bandwidth C_j. Therefore path j can carry $NM_j=C_j/(B_k/K_k)$ stripes for layer k. NM_j is rounded down to the nearest integer. Only the receivers whose *max-disjoint paths* can afford all stripes in layer k can join layer k.

During constructing the mesh for layer k, each node selects paths from *max-disjoint paths* in a distributed manner using the heuristic: evenly assigning the stripes in layer k on all paths in *max-disjoint paths*. This is to distribute the stripes as wide as possible to maximize the possibility of path overlapping. This is similar with the approach in table I except that the flow is assigned in stripes in overlay multicast. If the number of *max-disjoint paths* is larger than the stripe amount in layer k, each receiver chooses paths with smaller delay. Otherwise, choose all paths. Assume that the assigned number of stripes on path j is SN_j. After each receiver determines how many stripes a path carries, it sends a reservation packet along the path back to the sender. Assume that receiver t_i ($1\leq i\leq N_T$) is assigned $N_{i,m}$ stripes on link m ($1\leq m\leq M$). Each relay node aggregates the reservations from its downstream link m to calculate the required amount of stripes $N_m=\max(N_{i,m})$ ($1\leq i\leq N_T$) on link m. The reservations terminate at the sender. The sender also calculates the required amount of stripes on its downstream links. The reserved overlay links form the data distribution mesh in this layer. The pseudo-code for constructing mesh is shown in Fig. 4.

```
Receiver T:

        Update the available bandwidth of each path in maximal disjoint paths
        Assign stripes on the max-disjoint paths
        Reserve the assigned stripes back to the sender along the path
Relay R:
        Aggregate the reservation from different receivers
        Forward the reservation along the path
Sender S:
        Send required number of stripes to its next hops
```

Fig. 4. Pseudo-code for constructing a mesh for a layer

Iteratively, each receiver will update the available bandwidth of its *maximal-disjoint paths* for the next layer and repeat the reservation to get next layer. The procedure repeats till there are no receivers who can join higher layer.

4.4. Network Coding

After each layer's mesh is constructed, the sender sends corresponding layer's data to each downstream link at the reserved rate. During the data transmission, a relay node performs network coding for the stripes (in the same layer) from different upstream links and sends the reserved amount of coded stripes to downstream links.

There exist polynomial-time algorithms [9][10] to assign the deterministic coding vectors on the distribution mesh. However, random network coding [11] provides less complexity since it avoids the need to determine coding vectors before data distribution as well as the need to decide which data should be delivered along which path. We adopt the idea of practical random coding from Chou et al.[12]. After each layer is con-

structed, we use random linear coding in the layer. The stripes from different upstream links are combined with random operation over a large Galois Field.

To ensure easy operation, we assume that network coding is only performed within each layer. Combining stripes using network coding among different layers makes it difficult to decode for receivers with low available bandwidth.

Similar to previous layered multicast solutions [13], LION adapts to varying network conditions in a distributed manner. The meshes are constructed and removed dynamically over time. In case of congestion, data in higher layers are dropped first. When the packet loss in a layer exceeds a threshold, a receiver will notify its upstream parents and leave the corresponding mesh. If a relay node has no downstream children, it will also notify its parents and stop forwarding the data in the layer. In this way, congestion is alleviated. If there are no receivers being able to join a certain layer, the mesh of that layer is removed. To avoid fluctuation, a receiver will postpone its next attempt to join layers after it leaves a mesh*.

5. SIMULATIONS

In this section, we first show the numerical results for our heuristic on a physical network, followed by simulation results on an overlay network.

5.1. Throughput Gaps

In this section, we present the numerical results that compare the throughputs among the m-layer, 1-layer and the heuristic approaches.

The underlying networks were generated using the transit-stub model of GT-ITM [15]. The number of routers is 30. There are 6 to 22 receivers that are randomly attached to the routers. The available link bandwidth between transit routers is randomly set at 10-50Kbps. The available link bandwidth between transit routers and stub routers is 100Kbps. The sender's last hop link is set at 500Kbps, receiver's last hop link bandwidth is randomly set from 60 to 500Kbps. The source data has 20 layers, each layer has uniform bit-rate of 10Kbps.

We compare the throughput of m-Layer ILP, 1-Layer ILP, the heuristic approach, and the most prevalent IP multicast scheme using DVMRP[24] and SWP (Shortest Widest Path) [25]. DVMRP builds a shortest tree while SWP builds a widest tree rooted at the sender. Note that both DVMRP and SWP use multi-rate, each receiver chooses maximal number of layers to satisfy its available bandwidth.

We solve the m-Layer and 1-Layer ILP using Lingo [26]. The performance of the heuristic is obtained through simulation in C++ code. The results are averaged across 5 runs. The throughput gap is denoted by the ratio of the throughput to m-Layer's throughput. We summarize the results in Table I.

Table I. Throughput Gap among m-Layer, 1-Layer and heuristic approaches

Receiver Size	m-Layer	1-Layer	Heuristic	DVMRP	SWP
6	1	0.99245	0.91562	0.49026	0.60354
10	1	0.96553	0.92372	0.58331	0.65919
14	1	0.96838	0.87113	0.52223	0.57147
18	1	0.97019	0.89223	0.56225	0.61253
22	1	0.99806	0.88355	0.56879	0.60794

Since the m-Layer formulation jointly considers all the layers and all the receivers, the resultant solution is global optimal. While the 1-Layer formulation just considers the optimal solution within one layer, the performance is sub-optimal comparing to the m-Layer formulation. The proposed heuristic obtains the solution in a distributed manner which does not need information about all the receivers and consequently may result in worse performance than 1-Layer formulation. From the numerical results in Table I, we note that the throughput gaps between the *m*-Layer and the heuristic approaches are reasonably small (varies from 8% to 13% under different receiver size), given that the heuristic does not require global information and is a distributed approach. This demonstrates the effectiveness of the proposed heuristic. Furthermore, the gap between the *m*-Layer and the heuristic approaches increases as the receiver size increases. Even though there are some gaps between the m-Layer and the heuristic approaches, the heuristic approach still outperforms both DVMRP and SWP by about 50%, which is very significant. This is mainly due to the fact that the proposed heuristic exploits multiple paths between sender and receivers with the help of network coding.

5.2. Overlay Network

We evaluate the performance of LION, the proposed heuristic algorithm in an overlay network, through extensive simulations. The simulations were conducted in a simulator written in C++. The underlying router-level networks were generated using the transit-stub model of GT-ITM. We create end hosts and randomly connect them to the routers chosen from the stub domain. We compare LION with Narada [1] and Coded Multicast [8] using the following metrics.

1) *Throughput*: we measure the application layer throughput at each receiver. The multicast session's throughput is averaged over all receivers.
2) *Delay*: The end-to-end delay is defined as the time interval between the time a packet is being sent at the sender and the time the packet is correctly decoded at the receiver. As for Coded Multicast and LION, the end-to-end delay is defined as the longest delay among all the paths involved. The session's delay is averaged over all receivers.

---

* There already have large amount of work addressing how much time receivers have to wait for next attempt to join a certain layer to avoid message explosion [13]. We rely on existing work to solve this issue.

3) *Normalized Resource Usage Ratio (NRUR)*. Resource usage ratio (RUR) is defined as $\sum_{i=1}^{M} b_i / \sum_{j=1}^{N_T} T_j$, where b_i is the consumed bandwidth on IP link i which is involved in the session and T_j is receiver j's throughput. RUR reflects the network resource's efficiency that is, how much network resource is consumed to achieve a unit throughput. The higher the value of RUR is, the more network resource the multicast session consumes to deliver a bit of data. Normalized RUR (NRUR) is defined as the ratio of each scheme's RUR over DVMRP[24]'s RUR. NRUR reflect the overlay multicast's penalty compared with traditional IP multicast.

Stress, which is defined as the number of identical copies of a packet carried by a physical link[1], is a commonly used metric to evaluate overlay multicast's performance. Similar to NRUR, stress also reflects the overlay multicast's resource consumption penalty on each physical link compared with IP multicast. However, as network coding involves coding the packets in the network, it is not intuitive to directly calculate stress for a physical link under Coded Multicast and LION. Instead, we use NRUR to reflect the overlay multicast's overall resource consumption penalty compared with IP multicast.

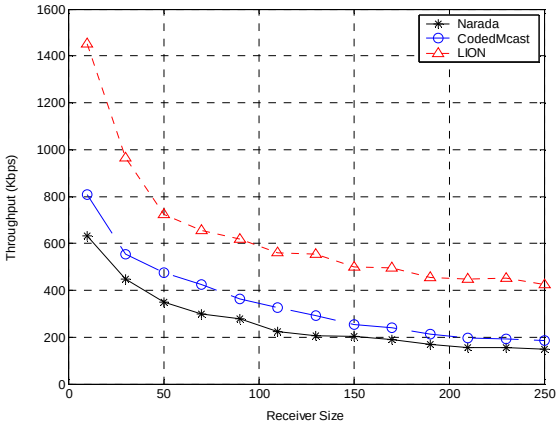
Three sets of simulations, namely scenario 1, 2 and 3, were conducted using the simulator.

Table II Link bandwidth between routers and end-systems chosen for scenario 1, 2 and 3

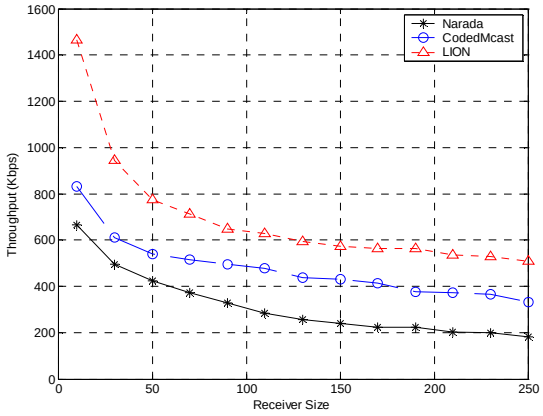
Parameter	Value
Routers in different transit	1-5Mbps
Routers in same transit	5-10Mbps
Transit router-Stub router	10Mbps
Stub router-Stub router	5Mbps
Stub Router-sender	5Mbps
Stub Router-relay	5Mbps
Stub Router-receiver	0.5-5Mbps

Scenario 1's underlying network consists of 800 routers. Scenario 2's underlying network consists of 1500 routers. Scenario 3's underlying network consists of 2000 routers. Other setups are summarized in Table II. The link bandwidth between transit domain routers is randomly set at 5-10Mbps. The link bandwidth between stub domain routers is set at 10Mbps. We set receiver's last hop link bandwidth randomly from 500Kbps to 5Mbps. The delay of link is randomly set at 5-20ms. Without loss generality, we do not model the queuing delay. The number of receivers ranges from 10 to 250. Suppose we have dedicated relay nodes in the overlay network. We set the ratio of relay size to receiver size 1.2.

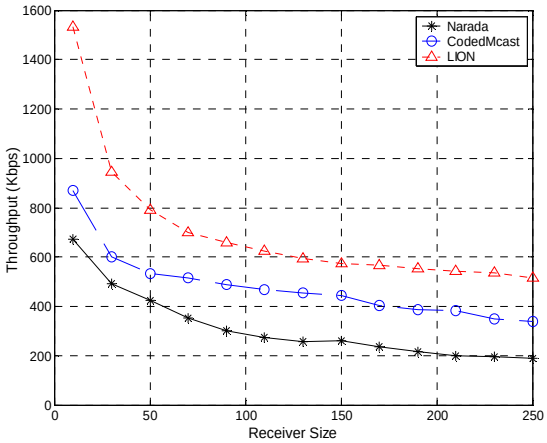
To ensure fair comparison, Narada, Coded Multicast and LION have the same physical network and participant end hosts in each scenario. Each curve in all the plots is an average over 50 simulation runs.



(a)



(b)

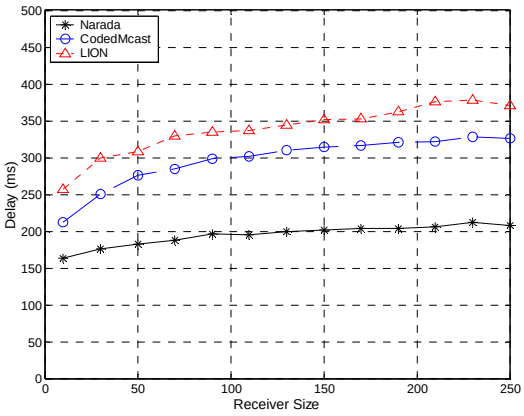


(c)

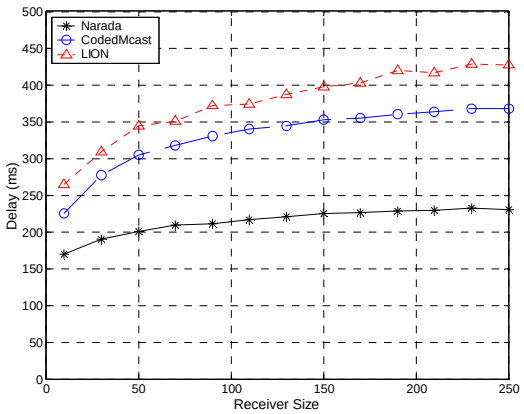
Fig. 5. Average throughput versus receiver size. (a) scenario 1 (800 routers). (b) scenario 2 (1500 routers). and (c) scenario 3 (2000 routers).

Fig 5. plots the average throughput of Narada, Coded Multicast and LION as a function of the receiver size for scenario 1, 2 and 3, respectively.

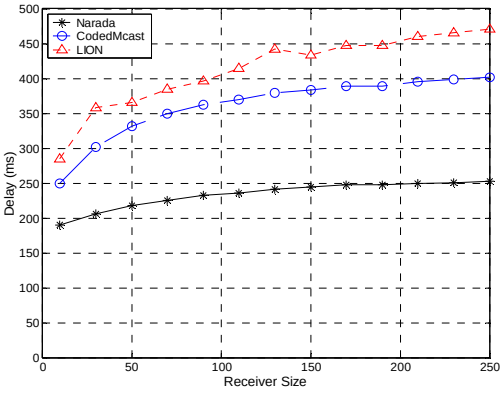
It can be observed from Fig. 5 that LION outperforms both Narada and Coded Multicast at all receiver sizes. As Coded Multicast uses network coding, the throughput is higher than that of Narada. Because LION considers each receiver's heterogeneity the throughput of receivers with higher available bandwidth can be improved by joining more layers. Hence the session's average throughput is improved. This highlights the benefits of LION. With the increase of the number of receivers, the throughput of the session decreases. This is due to the imperfect construction of all three kinds overlay networks. With the increase of the number of receivers, more overlay links will map to the same physical links. The stress of the shared physical link will increase, which indicates more wastage of network resource. The increase of receiver size also introduces fiercer competition among receivers to the limited bandwidth shared among them. In consequence, the session throughput decreases. However, LION still outperforms both Narada and Coded Multicast.



(a)



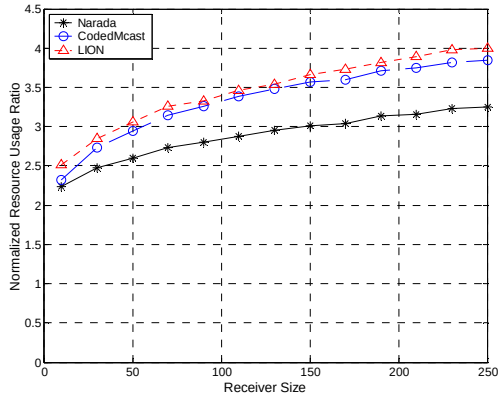
(b)



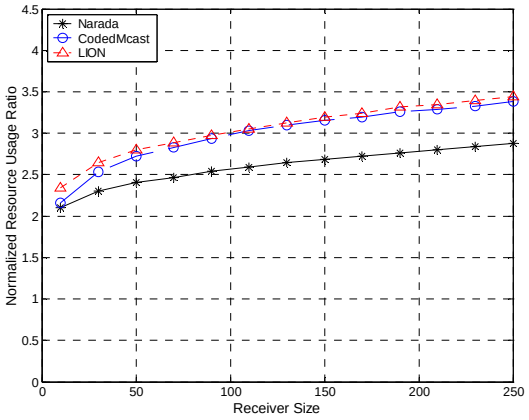
(c)

Fig 6. Average delay versus receiver size. (a) scenario 1 (800 routers). (b) scenario 2 (1500 routers). and (c) scenario 3 (2000 routers).

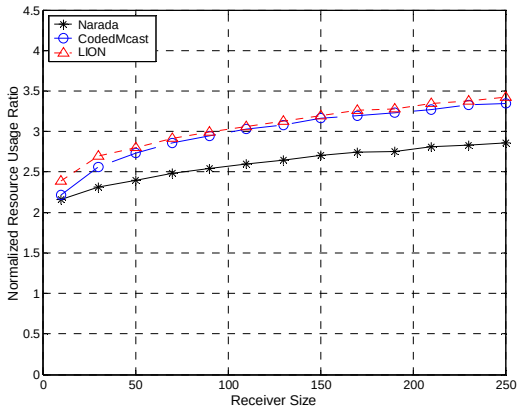
Fig. 6 plots the average delays as a function of the receiver size for Narada, Coded Multicast and LION, respectively. It is clear that Narada has the lowest delay since it uses single path with smallest delay (among all the possible paths from sender to a receiver). LION and Coded multicast have higher delays than Narada since they use multiple paths and the data transmission delay is aligned with the path with highest delay. Furthermore LION and Coded multicast can carry more data and the delay is averaged over not only those data normally accepted under Narada but also those additional data under LION or Coded multicast. As LION selects more paths than both Narada and coded multicast, it is reasonable that LION has the highest delay. We believe that the degradation of delay is compensated by throughput improvement. From Fig. 6 we can also observe that the increase in delay is not proportional to the increase in receiver size.



(a)



(b)



(c)

Fig. 7 Normalized Resource Usage Ratio versus receiver size. (a) scenario 1 (800 routers). (b) scenario 2 (1500 routers). and (c) scenario 3 (2000 routers).

Fig. 7 displays Normalized Resource Usage Ratio (NRUR) versus receiver size for Narada, Coded Multicast and LION. We observe in Fig. 7 that Narada has the lowest NRUR. This is because that multi-path transmission inherently results in larger RUR since it involves more links. LION has a slightly higher NRUR than Coded Multicast. It is reasonable since LION selects more paths and utilizes the links which have available bandwidth to improve throughput. The increase in LION's NRUR is not significant and not sensitive to the variation of receiver size.

We also investigate the impact of relay size on performance. We fix the receiver size to 100, and change the relay size from 10 to 200. Other simulation setups are same as in scenario 1.

Fig 8 shows the throughput as a function of relay size for Narada, Coded Multicast and LION. It is clear that since both Coded Multicast and LION need dedicated relays nodes, the session's average throughput increase with relay size. When relay size is small, more receivers will connect to a same relay, therefore the relay will become the bottleneck. Meanwhile some receivers can not find enough relay to connect to sender due to the relay's degree constraint. Therefore, the session's throughput increases with relay size. However,

when the relay size is larger than receiver size (100), the throughput does not increase anymore as the relay size increases. This is because when relay size is large enough, relay will not be a bottleneck. Hence additional relays will not necessarily increase the session's throughput.

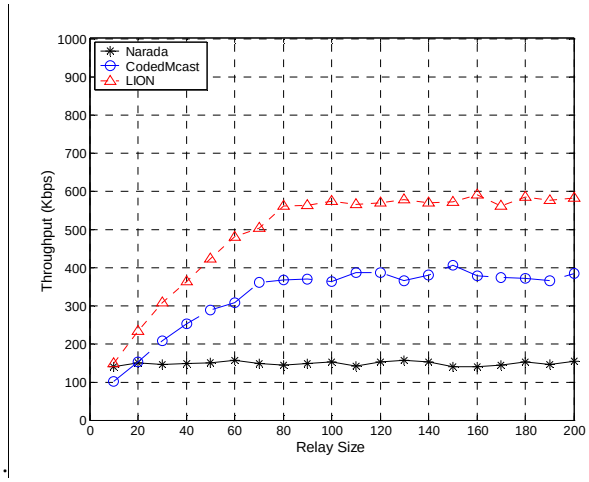


Fig. 8. Average Throughput versus relay size for Narada Coded Multicast and LION.

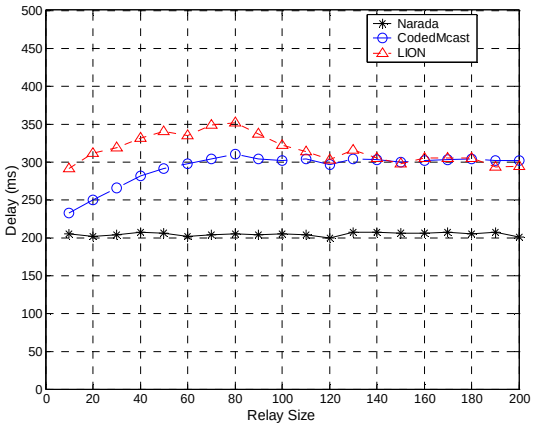


Fig. 9. Average Delay versus relay size for Narada Coded Multicast and LION.

Fig. 9 plots the average delay as a function of relay size for Narada, Coded Multicast and LION. As can be seen, the changes in delay under Narada are small. Coded Multicast and LION's delays first increase with relay size when the relay size is small. However the delay decreases or remains the same when the relay size is larger than receiver size, since receivers may find better relays in the paths which results in smaller delay.

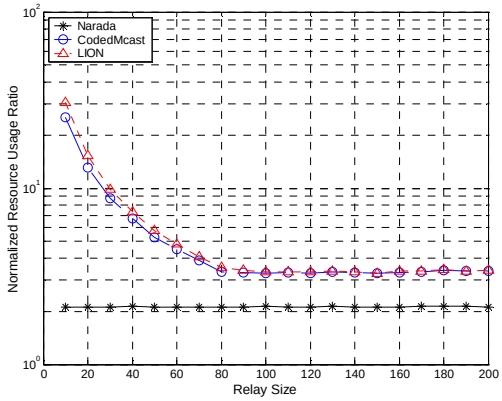


Fig. 10. Normalized Resource Usage Ratio versus relay size for Narada Coded Multicast and LION.

We observe in Fig. 10 that the NRURs of Coded Multicast and LION are much higher than that of Narada when the relay size is small. This is because, due to the node degree constraint enforced when constructing the basic overlay network [8], when relay size is small, some receivers can not find relay nodes and therefore can not join the multicast session which results in zero throughput. The links involved have low efficiency since they can only serve part of the receivers. Hence the NRURs of LION and Coded Multicast are high when relay size is small. The NRUR of Narada is almost constant for all relay size since Narada does not require dedicated relay nodes. For relay size larger than receiver size (100), the NRURs of Coded Multicast and LION are nearly constants and in the same order with that of Narada.

In summary, LION improves the session's throughput significantly while keeping the delay and network resource consumption under a reasonable value.

6. CONCLUSIONS

In this paper, we seek to improve multicast session throughput in heterogeneous overlay networks using network coding with the presence of the relay nodes. Given that the bandwidth between multicast sender and receivers is heterogeneous in nature, we propose a layered overlay multicast scheme to improve the throughput. Instead of building single or multiple trees, we construct the data distribution paths as multiple layered meshes. We first formulate the problem and then propose a distributed heuristic to approximate the optimal solution. Simulation results demonstrate that the overall throughput of a multicast session is significantly improved compared with existing works. Note that one of the limitations LION may have is that it's sensitive to the event of node join and leave. It may introduce certain overhead since sender may flood the network to find a better solution when node join or leave happens. However, we believe LION still has some applicable scenarios such as small scale overlay networks or a network which has relative static node membership.

Future works include implementing LION in Internet and combining LION with layered coding, say, FGS, PFGS, to provide scalable video delivery in a heterogeneous environment.

REFERENCES

[1] Yang-hua Chu, Sanjay G. Rao, Hui Zhang, "A case for end system multicast," in Proc. *ACM SIGMETRICS* 2000.

[2] V. Padmanabhan, H. Wang, P. Chou, K. Sripanidkulchai, "Distributing streaming media content using cooperative networking," in Proc. *ACM NOSSDAV*, May 2002.

[3] M. Castro, P. Druschel, A-M. Kermarrec, A. Nandi, A. Rowstron, A. Singh, "SplitStream: high-bandwidth multicast in a cooperative environment," in Proc. *ACM SOSP*, Oct. 2003

[4] Y. Zhu, J. Guo, B. Li, "oEvolve: towards evolutionary overlay topologies for high bandwidth data dissemination," *IEEE JSAC*, Sep. 2004.

[5] R. Ahlswede, N. Cai, S. Li, R. Yeung, "Network information flow," *IEEE Trans. on Information. Theory*, vol. 46, pp. 1204-1216, 2000.

[6] S. Li, R. Yeung, N. Cai, "Linear network coding," *IEEE Trans. on Information. Theory*, Vol. 49(2) pp.371-381, Feb. 2003.

[7] R. Koetter, M. Medard, "An algebraic approach to network coding," *IEEE/ACM Trans. on Networking*, Vol. 11(5), pp.782-795 Oct. 2003.

[8] Y. Zhu, B. Li, J. Guo, "Multicast with network coding in application-layer overlay networks," *IEEE JSAC*, Jan. 2004.

[9] P. Sanders, S. Egner, L. Tolhuizen. "Polynomial Time algorithms for Network information flow". *In Proc. of ACM SPAA*, pp.286-294, June 2003

[10] S. Jaggi, P.A. Chou, K. Jain. "Low complexity algebraic multicast codes. In: *Proc of IEEE ISIT* 2003

[11] T. Ho, R. Koetter, M. Medard, D. Karger and M. Effros, "The Benefits of Coding over Routing in a Randomized Setting", IEEE ISIT 2003

[12] P. Chou, Y. Wu, K. Jain, "Practical network coding," in Proc. 41st *Allerton Conf. on Communication Control and Computing*, 2003.

[13] S. McCanne, V. Jacobson, M. Vetterli, "Receiver-driven layered multicast," in Proc. *ACM SIGCOMM* , Sept. 1996.

[14] Y. Cui, K. Nahrstedt, "Layered peer-to-peer streaming," in Proc. *ACM NOSSDAV* 2003.

[15] E. Zegura, K. Calvert, S. Bhattacharjee, "How to model an internetwork," in Proc. *IEEE INFOCOM*, 1996.

[16] S. Saroiu, P. Gummadi, S. Gribble, "A measurement study of peer-to-peer file sharing systems," in *Proc. ACM*/SPIE *MMCN*, Jan. 2002.

[17] S. Savage, A. Collins, E. Hoffman, J. Snell, T. Anderson, "The end-to-end effects of Internet path selection," in Proc. *ACM SIGCOMM*, Aug. 1999.

[18] D. Andersen, H. Balakrishnan, M. Kaashoek, R. Morris, "Resilient overlay networks," in Proc. *ACM SOSP*, Oct. 2001.

[19] K. Menger, "Zur allgemeinen Kurventheorie," *Fund. Math*.,vol. 10, pp.95-115, 1927.

[20] L. Ford, D. Fulkerson, “Maximal flow through a network,” *Canadian J. Mathematics*, vol. 8: 399-404, 1956.

[21] J. Edmonds, “Edge-disjoint branchings,” in *Combinatorial Algorithms*, Ed. R. Rustin, Algorithmics Press, New York, 1973, 91-96.

[22] K. Jain, M. Mahdian, M. Salavatipour, “Packing Steiner Trees,” *ACM-SIAM SODA* 2003.

[23] F. K. Hwang, D. S. Richards, and P. Winter. “ The Steiner Tree Problem” In: Annals of Discrete Mathematics series 53, North-Holland, 1992

[24] S. Deering, D. Cheriton, “Multicast routing in datagram internetworks and extended LANs,” *ACM Trans. on Computer Systems*, vol. 8(2), pp. 85-111, May 1990

[25] Zheng Wang, Jon Crowcroft, "Quality of Service Routing for Supporting Multimedia Applications", IEEE JSAC, Vol. 14, No. 7, September 1996, pp.1288-1234.

[26] Lingo optimization solver, http://www.lindo.com