Network Coding for Multi-Resolution Multicast

MinJi Kim, Daniel Lucani, Xiaomeng Shi, Fang Zhao, Muriel Médard
Massachusetts Institute of Technology, Cambridge, MA 02139, USA
Email: {minjikim, dlucani, xshi, zhaof, medard}@mit.edu

***Abstract*—Multi-resolution codes enable multicast at different rates to different receivers, a setup that is often desirable for graphics or video streaming. We propose a simple, distributed, two-stage message passing algorithm to generate network codes for single-source multicast of multi-resolution codes. The goal of this *pushback algorithm* is to maximize the total rate achieved by all receivers, while guaranteeing decodability of the base layer at each receiver. By conducting pushback and code assignment stages, this algorithm takes advantage of inter-layer as well as intra-layer coding. Numerical simulations show that in terms of total rate achieved, the pushback algorithm outperforms routing and intra-layer coding schemes, even with field sizes as small as 2^{10}(10 bits). In addition, the performance gap widens as the number of receivers and the number of nodes in the network increases. We also observe that naïve inter-layer coding schemes may perform worse than intra-layer schemes under certain network conditions.**

I. Introduction

Many real-time applications, such as teleconferencing, video streaming, and distance learning, require multicast from a single source to multiple receivers. In conventional multicasts, all receivers receive at the same rate. In practice, however, receivers can have widely different characteristics. It becomes desirable to serve each receiver at a rate commensurate with its own demand and capability. One approach to multirate multicast is to use multi-description codes (MDC), dividing source data into equally important streams such that the decoding quality using any subset of the streams is acceptable, and better quality is obtained by more descriptions. A popular way to perform MDC is to combine layered coding with the unequal error protection of a priority encoding transmission (PET) system [1]. Another approach for multirate multicast is to use multi-resolution codes (MRC), encoding data into a base layer and one or more refinement layers [2], [3]. Receivers subscribe to the layers cumulatively, with the layers incrementally combined at the receivers to provide progressive refinement. The decoding of a higher layer always requires the correct reception of all lower layers including the base layer.

In this paper, we consider multirate multicast with linear network coding. Proposed in [4], network coding allows and encourages mixing of data at intermediate nodes. It has been shown that for a single rate multicast, network coding achieves the minimum of the maximum flow to each receiver; this limit is generally not achievable through traditional routing schemes. Kötter and Médard also studied multirate multicast, deriving necessary algebraic conditions for the existence of network coding solutions for a given network and receiver requests [5]. For n-layer multicast, linear network codes can satisfy requests from all the receivers if the n layers are to be multicasted to all but one receiver. If more than one subscribe to subsets of the layers, linear codes cease to be sufficient.

Previous work on multirate multicast with network coding includes [6], [7], [8], [9], [10], [11]. For the MDC approach, [6] and [7] modified PET at the source to cater for network coded systems. Recovery of some layers is guaranteed before full rank linear combinations of all layers are received, and this is achieved at the cost of a lower code rate. Wu *et al.* studied the problem of Rainbow Network Coding, which incorporates linear network coding into multi-description coded multicast [8]. For the MRC approach, [9] studied multi-resolution media streaming, and proposed a polynomial-time algorithm for multicast to heterogeneous receivers. Zhao *et al.* considered multirate multicast in overlay networks [10], organizing receivers into layered data distribution meshes, and utilizing network coding in each mesh. Xu *et al.* proposed the Layered Separated Network Coding Scheme to maximize the total number of layers received by all receivers [11].

Note that if no additional coding at the source such as modified PET is used, the aggregate rate to all receivers is maximized by solving the linear network coding problem separately for each layer [8], [9], [10], [11]. Specifically, for each layer, a subgraph is selected for network coding by performing linear programming. In other words, only *intra-layer* network coding is allowed. On the other hand, *inter-layer* network coding, which allows coding across layers, often achieves higher throughput. Incorporating inter-layer linear network coding into multirate multicast, however, is significantly more difficult, as intermediate nodes have to know the network topology and the demands of all downstream receivers before determining its network codes.

Reference [12] considers inter-layer network coding by performing integer-programming (IP) flow optimization on "multicast layers i" defined as the combinations of layers from 1 up to i. In addition to IP, which is NP-Hard, this algorithm requires several computations of edge disjoint paths, which is also NP-Hard. It also requires centralized knowledge of the

This work was supported in part by the National Science Foundation under grants No. 0831728, CNS-0627021, and CCR-0325496, and by ONR MURI Grant No. N00014-07-1-0738, by subcontracts # 060786/069145 issued by BAE Systems National Security Solutions, Inc. In addition, this work was partially supported by the Defense Advanced Research Projects Agency (DARPA) and the Space and Naval Warfare System Center (SPAWARSYSCEN), San Diego under Contract Nos. N66001-06-C-2020/00151, and by subcontract #18870740-37362-C issued by Stanford University and supported by DARPA.

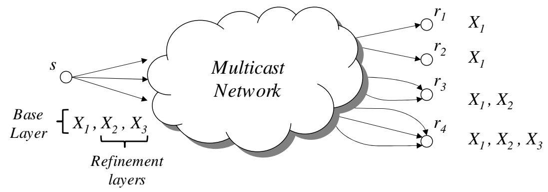

Fig. 1. A network with a source s with multi-resolution codes $\mathcal{X}_1, \mathcal{X}_2$, and $\mathcal{X}_3$, and receivers r_1, r_2, r_3, r_4.

network topology. Such centralized algorithms are difficult to perform on general networks. On the other hand, the algorithm we study has polynomial complexity and is fully distributed.

In this paper, we propose a simple, distributed, two-stage message passing algorithm to generate network codes for single source multicast of multi-resolution codes. Unlike previous work, this algorithm allows both intra-layer and inter-layer network coding at all nodes. It guarantees decodability of the base layer at all receivers. In terms of total rate achieved, with field size as small as 2^{10}, it outperforms routing as well as network coding schemes that allow intra but not inter-layer coding. The performance gain increases as the number of receivers increases and as the network grows in size, if appropriate criterion is used. Otherwise, naïve inter-layer coding may lead to an inappropriate choice of network code, which can be worse than intra-layer network coding.

The rest of this paper is organized as follows. Section II presents the network model and the network coding problem of multicast of multi-resolution codes. The pushback algorithm is proposed in Section III, and analyzed in Section IV. Simulation results are presented in Section V, while discussions on future work conclude the paper in Section VI.

II. Problem Setup

We consider the network coding problem for single-source multicast of multi-resolution codes, as illustrated in Figure 1. A single-source multicast network is modeled by a directed acyclic graph $G = (V, E)$, V being the set of nodes, and E the set of links. Each link is assumed to have unit capacity, while links with capacities greater than 1 are modeled with multiple parallel links. The subset $R = \{r_1, r_2, ...r_n\} \subseteq V$ is the set of receivers which wish to receive information from the source node $s \in V$. The source processes, $\mathcal{X}_1, \mathcal{X}_2, ..., \mathcal{X}_L$, constitute a multi-resolution code, where $\mathcal{X}_1$ is the *base layer* and the rest are the *refinement layers*. It is important to note that layer $\mathcal{X}_i$ without layers $\mathcal{X}_1, \mathcal{X}_2, ..., \mathcal{X}_{i-1}$ is not useful for any i. For simplicity, we assume each layer is of unit rate. Thus, given a link $e \in E$, we can transmit one layer (or equivalent coded data rate) on e at a time. The min-cut between s and a node v is denoted by $minCut(v)$, and we assume that every node v knows its $minCut(v)$. Note that there are efficient algorithms, such as Ford-Fulkerson algorithm, that can compute $minCut(v)$.

Our goal is to design a simple and distributed algorithm that provides a coding strategy to maximize the total rate achieved by all receivers with the reception of the base layer guaranteed to all receivers. By Min-Cut Max-Flow bound, each receiver r_i can receive at most $minCut(r_i)$ layers ($\mathcal{X}_1$, $\mathcal{X}_2$, .., $\mathcal{X}_{minCut(r_i)}$). We present the *pushback* algorithm, and compare its performance against other existing algorithms and against the theoretical bound of Min-Cut Max-Flow.

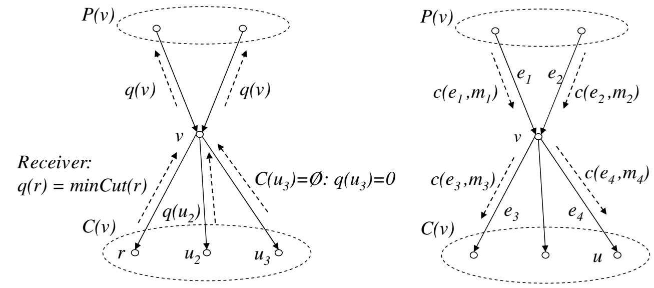

Fig. 2. Pushback stage and code assignment stage at node v.

III. Pushback Algorithm

The pushback algorithm is a distributed algorithm which allows both intra-layer and inter-layer linear network coding. It consists of two stages: *pushback* and *code assignment*.

In the pushback stage, messages initiated by the receivers are pushed up to the source, allowing upstream nodes to gather information on the demand of any receiver reachable from them. Messages are passed from nodes to their parents. Initially, each receiver $r_i \in R$ requests for layers $\mathcal{X}_1, \mathcal{X}_2, ..., \mathcal{X}_{minCut(r_i)}$ to its upstream nodes, *i.e.*, the receiver r_i requests to receive at a rate equal to its min-cut. An intermediate node $v \in V$ computes a message, which depends on the value of $minCut(v)$ and the requests from its children. Node v then pushes this message to its parents, indicating the layers which the parent node should encode together.

The code assignment stage is initiated by the source once pushback stage is completed. Random linear network codes [13] are generated in a top-down fashion according to the pushback messages. The source s generates codes according to the messages from its children: s encodes the requested layers and transmits the encoded data to the corresponding child. Intermediate nodes then encode/decode the packets according to the messages determined during the pushback stage.

To describe the algorithm formally, we introduce some additional notations. For a node v, let $P(v)$ be its set of parent nodes, and $C(v)$ its children as shown in Figure 2. $P(v)$ and $C(v)$ are disjoint since the graph is acyclic. Let $E_v^{in} = \{(v_1, v_2) \in E \mid v_2 = v\}$ be the set of incoming links, and $E_v^{out} = \{(v_1, v_2) \in E \mid v_1 = v\}$ the set of outgoing links.

A. Pushback Stage

As shown in Figure 2, we denote the message received by node v from a child $u \in C(v)$ as $q(u)$, and the set of messages received from its children as $q(C(v)) = \{q(u) | u \in C(v)\}$. A message $q(u)$ means that u requests its parents to code across layers 1 to *at most* $q(u)$. Once requests are received from all children, v computes its message $q(v)$ and sends the same $q(v)$ to all of its parents. The request $q(v)$ is a function of $q(C(v))$ and $minCut(v)$, *i.e.* $q(v) = f(q(C(v)), minCut(v))$. A pseudocode for the pushback stage at a node $v \in V$ is shown in Algorithm 1. It is important to note that the choice of $f(\cdot)$ is a

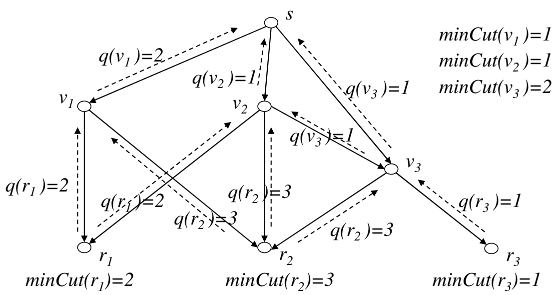

Fig. 3. An example of pushback stage with min-req criterion.

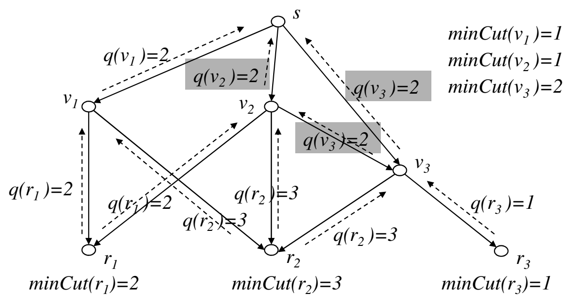

Fig. 4. An example of pushback stage with min-cut criterion. Highlighted are the messages different from that of min-req criterion (Figure 3).

key feature of the algorithm as it determines the performance. We present two different versions of $f(\cdot)$: *min-req criterion* and *min-cut criterion*, which we discuss next.

1) Min-req Criterion: The min-req criterion, as the name suggests, defines $q(v) = f(q(C(v)), minCut(v))$ as follows:

$$q(v) = \begin{cases} 0 & \text{if } q(u) = 0 \text{ for all } u \in C(v), \\ q_{min} & \text{otherwise,} \end{cases}$$

where $q_{min} = \min_{q(u)\neq 0,\ u\in C(v)} q(u)$ is the minimum non-zero $q(u)$ from $u \in C(v)$.

This criterion may seem very pessimistic and naïve, as the intermediate nodes serve only the minimum requested by their downstream receivers to ensure the decodability of the base layer. Nonetheless, as we shall see in Section V, the performance of this criterion is quite good. An example of pushback with min-req is shown in Figure 3. Receivers r_1, r_2, and r_3 request their min-cut values 2, 3, and 1, respectively. The intermediate nodes v_1, v_2, and v_3 request the minimum of all the requests received, which are 2, 1, and 1, respectively.

```
if v is a receiver then
|  q(v) = minCut(v);
end
if v is an intermediate node then
|  if C(v) = ∅ then
|  |  q(v) = 0;
|  end
|  if C(v) ≠ ∅ then
|  |  q(v) = f(q(C(v)), minCut(v));
|  end
end
```

Algorithm 1: The pushback stage at node v.

2) Min-cut Criterion: The min-cut criterion defines the function $q(v) = f(q(C(v)), minCut(v))$ as follows:

$$q(v) = \begin{cases} q_{min} & \text{if } minCut(v) \leq q_{min}, \\ minCut(v) & \text{otherwise,} \end{cases}$$

where $q_{min} = \min_{q(u)\neq 0,\ u\in C(v)} q(u)$.

Note if a node v receives $minCut(v)$ number of linearly independent packets coded across layers 1 to $minCut(v)$, it can decode layers $\mathcal{X}_1, \mathcal{X}_2, ..., \mathcal{X}_{minCut(v)}$ and act as a secondary source for those layers. Thus, if there is at least one child $u \in C(v)$ that requests fewer than $minCut(v)$ layers, *i.e.* $minCut(v) > q_{min}$, node v sets its request $q(v)$ to $minCut(v)$. However, if all children request more than $minCut(v)$ layers, node v does not have sufficient capacity to decode the layers requested by its children. Thus, it sets $q(v) = q_{min}$. An example of pushback with min-cut is shown in Figure 4. The network is identical to that of Figure 3. Again, the nodes r_1, r_2, r_3, and v_1 request 2, 3, 1, and 2, respectively. However, node v_2 requests 2, which is the minimum of all the requests it received, and node v_3 requests $minCut(v_3) = 2$.

B. Code Assignment Stage

This stage is initiated by the source after pushback is completed. As shown in Figure 2, $c(e, m)$ denotes the random linear network code v transmits to its child $u \in C(v)$, where $e = (v, u)$, and m means that packets on e are coded across layers 1 to m. Note m may not equal to $q(u)$, which we discuss in more detail in Section IV. Algorithm 2 presents a pseudocode for the code assignment stage at any node $v \in V$.

Algorithm 2 considers source, intermediate, and receiver nodes separately. The source always exactly satisfies any requests from its children, while the receivers decode as many consecutive layers as they can. For an intermediate node v connected to the network ($P(v) \neq \emptyset$), v collects all the codes $c(e_i, m_i)$ from its parents and determines m^*, the number of layers up to which v can decode. It is possible that v cannot decode any layer, leading to an m^* equal to zero. For $m^* \neq 0$, v can act as a secondary source for layers 1, 2, ..., m^* by decoding these layers. In the case where $q(u) \leq m^*$, $u \in C(v)$, v can satisfy u's request exactly by encoding just the layers 1 to $q(u)$. If $q(u) > m^*$, v cannot decode the layers u requested; thus, cannot satisfy u's request exactly. Therefore, v uses a best effort approach and delivers a packet coded across m_{max} layers, where m_{max} is the closest to $q(u)$ node v can serve without violating u's request, *i.e.* $q(u) \geq m_{max}$. The code assignment stage requires that every node checks its decodability to determine m^*. This process involves Gauss-Jordan elimination, which is computationally cheaper than matrix inversion required for decoding. Note that only a subset of the nodes need to perform (partial) decoding.

Figures 5 and 6 show the code assignment stage for the examples in Figures 3 and 4. The algorithm for code assignment stays the same, whether we use min-req or min-cut criterion during the pushback stage, but the resulting code assignments are different. Although the throughput achieved in these examples are the same, this is usually not the case.

if v *is the source* s **then**
  foreach *edge* $e=(v,u)\in E_v^{out}$ **do**
    v transmits $c(e,q(u))$;
  end
end
if v *is an intermediate node* **then**
  if $P(v)=\emptyset$ **then**
    v sets $c(e,0)$ for all $e\in E_v^{out}$;
  end
  if $P(v)\neq\emptyset$ **then**
    v receives codes $c(e_i,m_i)$, $e_i\in E_v^{in}$;
    v determines m^*, which is the maximum m such that $\mathcal{X}_1,\mathcal{X}_2,...\mathcal{X}_m$ are decodable from $c(e_i,m_i)$'s;
    foreach *child* $u\in C(v)$ **do**
      Let $e=(v,u)$;
      if $q(u)\le m^*$ **then**
        v decodes layers $\mathcal{X}_1,\mathcal{X}_2,...,\mathcal{X}_{m^*}$;
        v transmits $c(e,q(u))$;
      end
      if $q(u)>m^*$ **then**
        Let $m_{max}=\max_{m_i\le q(u)} m_i$;
        v transmits $c(e,m_{max})$;
      end
    end
  end
end
if v *is a receiver* **then**
  v receives codes $c(e_i,m_i)$, $e_i\in E_v^{in}$;
  v decodes m^* layers, which is the maximum m such that $\mathcal{X}_1,\mathcal{X}_2,...\mathcal{X}_m$ are decodable from $c(e_i,m_i)$'s;
end

Algorithm 2: The code assignment stage at node v.

Generally the min-cut criterion achieves higher throughput than the min-req criterion.

C. Complexity analysis

Given a single-source multicast network with a L-layer multi-resolution code, each pushback message $q(\cdot)\in\{1,\dots,L\}$ is $\lceil\log_2 L\rceil$-bits long. On the other hand, the code assignment message $c(\cdot)$ from node v to $u\in C(v)$ is of length $O(q(v)\log_2(p))$-bits where p is the field size. Note that $\log_2 p\approx 10$ is sufficient (see Section V); therefore, $c(\cdot)$ is $O(L)$-bits long. Thus, a node v transmits a total of $O(\lceil\log_2 L\rceil|P(v)|+L|C(v)|)$-bits. Given L, the message complexity is linear in the number of neighbors v has.

To compute $q(v)$, node v performs $O(|C(v)|)$ comparisons. Since v transmits the same $q(v)$ to all its parents, v only needs to compute $q(v)$ once. To compute $c(v)$, node v first performs a Gauss-Jordan elimination to determine m^* (see Section V for more detail). Note that given a $k\times l$ matrix, Gauss-Jordan elimination has complexity $O(kl^2)$. By Lemma 4.2, v receives a coding matrix of size $|P(v)|\times q(v)$, which is at most a matrix of size $|P(v)|\times L$. Thus, computing m^* requires at most $O(|P(v)|L^2)$ operations. In addition, v may need to decode m^* layers, where $m^*\le\min\{|P(v)|,L\}$. Decoding involves a matrix inversion, with complexity $O(\min\{|P(v)|,L\}^3)$. Lastly, re-encoding requires a vector-matrix multiplication, which has complexity $O(|P(v)|L)$. Note that v needs to re-encode a separate message $c(\cdot)$ for each $u\in C(v)$. Thus, re-encoding requires $O(|C(v)||P(v)|L)$ operations. The total complexity at v is $O(|P(v)|L^2)+O(\min\{|P(v)|,L\}^3)+O(|C(v)||P(v)|L)=O(|P(v)|L^2+|C(v)||P(v)|L)$.

The overall time required for each stage is linearly proportional to the maximum distance from the source to any of the sinks, assuming that each link incurs unit delay. If we assume the network is static, this time delay can be well amortized over multiple transmissions.

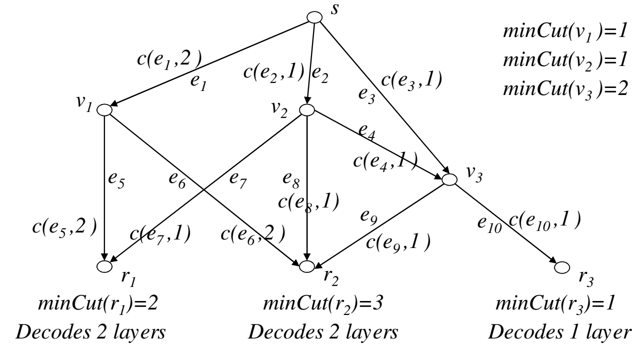

Fig. 5. An example of code assignment stage with min-req criterion.

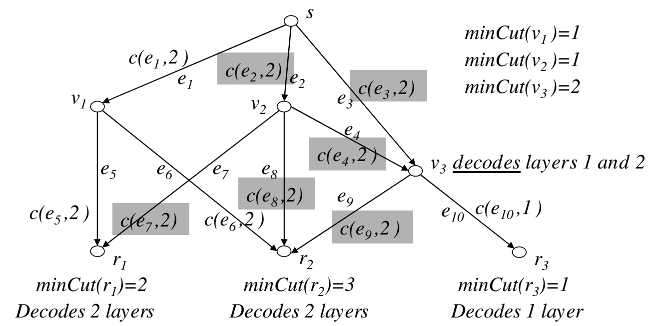

Fig. 6. An example of code assignment stage with min-cut criterion. High–lighted are the messages different from the min-req criterion (Figure 5).

IV. Analysis of Pushback Algorithm

In general, not all receivers can achieve their min-cuts through linear network coding. Nonetheless, we want to guarantee that no receiver is denied service, *i.e.* although some nodes may not receive up to the number of layers desired, all should receive at least layer 1. In this section, we prove that the pushback algorithm guarantees decodability of the base layer, $\mathcal{X}_1$, at all receivers. Two related lemmas are presented to prove Theorem 4.3.

Lemma 4.1: Assume $minCut(v)=n$ for a node v in G. In the pushback algorithm, if $m_i\le n$ for all $c(e_i,m_i)$, $e_i\in E_v^{in}$, then v can decode at least layer 1 with high probability. In other words, if all received codes at v are combinations of at most n layers, v can decode at least layer 1.

Proof: Recall that a code $c(e_i,m_i)$ represents coding across layers 1 to m_i; if the field size is large, with high probability, the first m_i elements of this coding vector are non-zero, whereas the rest are zeros.

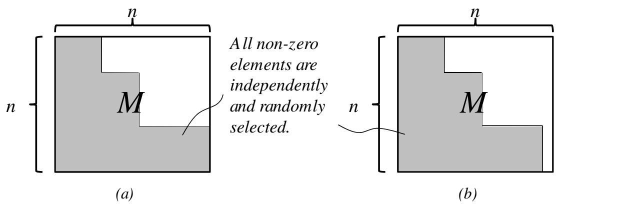

Fig. 7. Coding matrix M; each row represents a code received, and columns represent the layers. The maximum number of non-zero columns in M, c_k, can be equal to n (as shown in (a)), or less than n (as shown in (b)).

Since $minCut(v) = n$, there exist n edge-disjoint paths from the source s to v, for all links are assumed to have unit capacity. Therefore, v receives from its incoming links at least n codes, each of which can be represented as a row coding vector of length n, since $m_i < n$ for all i. Pick the n codes corresponding to the edge-disjoint paths to obtain an $n \times n$ coding matrix. For the square coding matrix, sort its rows according to the number of non-zero elements per row, obtaining the structure shown in Figure 7. We denote this sorted matrix by M, and the unique numbers of non-zero elements in its rows by $c_1, c_2, ..., c_k$, in ascending order. Since the rows of M are generated along edge-disjoint paths using random linear network coding, the non-zero elements of M are independently and randomly selected.

Next, define upper-left corner submatrices M_1, M_2, ..., M_k as shown in Figure 8, where each submatrix M_i is of size $r_i \times c_i$. Specifically, the rows of M with exactly c_1 non-zero elements form a $r_1 \times c_1$ submatrix M_1; the rows of M with exactly c_1 or c_2 non-zero elements form the $r_2 \times c_2$ submatrix M_2. M_k is of size $r_k \times c_k$, where $r_k = n$, and $c_k \leq n$. Note for any i, if $rank(M_i) = c_i$, node v can decode layers 1 to c_i, *i.e.*, the base layer is decodable. In other words, layer 1 is *not decodable* at node v only if $rank(M_i) < c_i$ for all i.

With these definitions, we assume layer 1 is not decodable at node v, and prove the lemma by contradiction. Specifically, we prove by induction that this assumption implies $r_i < c_i$ for all i, leading to the contradiction $r_k < c_k$.

For the base case, first consider M_1. If layer 1 is not decodable, $rank(M_1) < c_1$. Recall that elements in M_1 are independently and randomly selected [13]; if $r_1 \geq c_1$, with high probability, $rank(M_1) = c_1$. Therefore, the above assumption implies $r_1 < c_1$ and $rank(M_1) = r_1$. Next consider M_2. Under the assumption that layer 1 is not decodable, $rank(M_2) < c_2$. Since $rank(M_1) = r_1$ and M_2 includes rows of M_1, $rank(M_2) \geq r_1$. Rows $r_1+1, r_1+2, ..., r_2$ are called the *additional rows* introduced in M_2. If there are more than $c_2 - r_1$ additional rows, M_2 has full rank, *i.e.* $rank(M_2) = c_2$, with high probability. Hence, the number of additional rows in M_2 must be less than $c_2 - r_1$, implying $r_2 < c_2$.

For the inductive step, consider M_i, $3 \leq i \leq k$. Assume that $r_j < c_j$ for all $j < i$. If layer 1 is not decodable, $rank(M_i) < c_i$. By similar arguments as above, $rank(M_{i-1}) = r_{i-1}$, and there must be less than $c_i - r_{i-1}$ additional rows introduced in M_i. Thus, $r_i < c_i$. By induction, the total number of rows $r_k = n$ in M is strictly less than $c_k \leq n$, which is a contradiction. We therefore conclude that node v can decode the base layer. In fact, v can decode at least c_1 layers. ∎

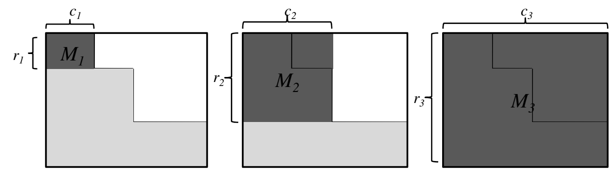

Fig. 8. Upper-left corner submatrices M_1, M_2, and M_3.

Lemma 4.2: In the pushback algorithm, for each link $e = (v, v')$, assume that node v' sends request $q(v') = q$ to node v. Then, the code $c(e, m)$ on link e (from node v to v') is coded across at most q layers, *i.e.* $m \leq q$.

Proof: First, define the notion of *levels*. A node u is in level i if the longest path from s to u is i, as shown in Figure 9. Since the graph is acyclic, each node has a finite level number. We shall use induction on the levels to prove that this lemma holds for both min-req and min-cut criteria.

For the base case, if v' is in level 1, it is directly connected to the source, and receives a code across exactly q layers on e from s. For the inductive step, assume that all nodes in levels 1 to i, $1 \leq i < k$, get packets coded across layers 1 to at most their request. Assume v' is in level $i+1$.Let $v \in P(v')$; therefore, v is in level $j \leq i$. Let q_{min} be the smallest non-zero request at v, that is $q_{min} = min_{q(u) \neq 0, u \in C(v)} q(u)$.

For the min-req criterion, v always sends request $q(v) = q_{min}$ to its parents, and the codes v receives are linear combinations of at most q_{min} layers. Therefore, the code v sends to its children is coded across at most q_{min} layers, where $q = q(v') \geq q_{min}$. In other words, the code received by v' is coded across at most q layers.

For the min-cut criterion, if $q_{min} > minCut(v)$, node v requests $q(v) = q_{min}$ to its parents. By the same argument as that for the min-req criterion, v' receives packets coded across at most q layers. If $q_{min} \leq minCut(v)$, v requests $q(v) = minCut(v)$. According to the code assignment stage, if v cannot satisfy request q exactly, it will send out a linear combination of the layers it can decode. Since v is in level $j \leq i$, v receives codes across layers 1 to at most $minCut(v)$. By Lemma 4.1, node v can decode at least the base layer. Thus, we conclude that node v is always able to generate a code for node v' such that it is coded across layers 1 to at most q. ∎

Theorem 4.3: In the pushback algorithm, every receiver can decode at least the base layer.

Proof: The receiver with min-cut n receives linear combination of at most n layers by Lemma 4.2. From Lemma 4.1, the receiver, therefore, can decode at least the base layer. ∎

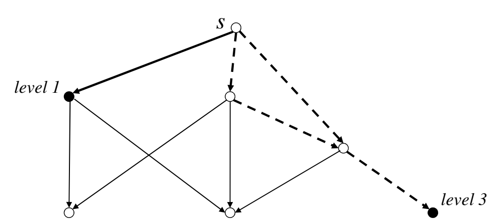

Fig. 9. Level 1 and level 3 node in a network.

V. SIMULATIONS

To evaluate the effectiveness of the pushback algorithm, we implemented it in Matlab, and compared the performance with both routing and intra-layered network coding schemes. Random networks were generated, with a fixed number of receivers randomly selected from the vertex set. We consider two metrics to evaluate the performance:

$$\textit{\% Happy Nodes} = \frac{100}{\text{\# of trials}} \sum_{\text{all trials}} \frac{\text{\# of receivers that achieve min-cut}}{\text{\# of receivers}},$$

$$\textit{\% Rate Achieved} = 100 \frac{\sum_{\text{all trials}} \text{total rate achieved}}{\sum_{\text{all trials}} \text{total min-cut}}.$$

The *% Happy Nodes* metric is the average percentage of receivers that achieved their min-cuts, *i.e.* receivers that received the service they requested. The *% Rate Achieved* metric gives a measure of what percentage of the total requested rate (equal to sum of min-cuts) was delivered to the receivers over all trials. It is important to note that the optimal achievable rates (denoted OPT) for multirate multicast are generally unknown for a network. The min-cut is the theoretical upper bound on OPT. Therefore, *% Rate Achieved* is a lower bound on the total rate achieved in terms of OPT, *i.e.*

$$\textit{\% Rate Achieved} \leq 100 \frac{\sum_{\text{all trials}} \text{total rate achieved}}{OPT}.$$

Recall that this total rate achieved is what we intend to maximize with the proposed algorithm.

As an example, consider two possible cases where the (min-cut, achieved-rates) pairs are $([1, 1, 2], [1, 1, 1])$ and $([2, 2, 3], [2, 2, 2])$. In both cases, the demand of a single receiver is missed by one layer, but the corresponding fractions of rates achieved are $3/4$ and $6/7$ respectively. Using only the *% Happy Nodes* metric would tell us that $1/3$ of the receivers did not received all requested layers. However, the *% Rate Achieved* metric provides a more accurate measure of how 'unhappy' the overall network is.

A. Algorithms for comparison

1) Point-to-point Routing Algorithm: the point-to-point routing algorithm considers each multicast as a set of unicasts. The source node s first multicasts the base layer $\mathcal{X}_1$ to all receivers. To determine the links used for layer $\mathcal{X}_1$, s computes the shortest path to each of the receivers separately. Given the shortest paths to all receivers, s then uses the union of the paths to transmit the base layer. Note the shortest path to receiver r_i may not be disjoint with the shortest path to receiver r_j. After transmitting layer $\mathcal{X}_{i-1}$, $2 \leq i \leq L$, source s uses the remaining network capacity to transmit the next refinement layer $\mathcal{X}_i$ to as many receivers as possible. First, s updates the min-cut to all receivers and identifies receivers that can receive $\mathcal{X}_i$. Let $R' = \{r_{i_1}, r_{i_2}, ...\}$ be the set of receivers with updated min-cut greater than 1 and, therefore, can receive layer $\mathcal{X}_i$. The source s then computes the shortest paths to receivers in R'. The union of these paths is used to transmit the refinement layer. This process is repeated until no receiver can be reached or the layers are exhausted.

2) Steiner Tree Routing Algorithm: the Steiner tree routing algorithm computes the minimal-cost tree connecting source s and all the receivers. We assume that each link is of unit cost. For the base layer $\mathcal{X}_1$, s computes and transmits on the Steiner tree connecting to all receivers. For each new refinement layer $\mathcal{X}_i$, s computes a new Steiner tree to receivers with updated min-cuts greater than zero. This process is repeated until no receiver can be reached or the layers are exhausted.

It is important to note that Steiner tree routing algorithm is an optimal routing algorithm – it uses the fewest number of links to transmit each layer. Unlike the point-to-point algorithm, this algorithm may make routing decisions that is not optimal to any single receiver, *i.e.* the source may use a non-shortest path to communicate to a receiver, but it uses fewer links globally. However, this optimality comes with a cost: the problem of finding a Steiner tree is NP-complete.

3) Intra-layer Network Coding Algorithm: the intra-layer network coding algorithm uses linear coding on each layer separately. It iteratively solves the linear programming problem for linear network coding for layer $\mathcal{X}_i$ with receivers $R_i = \{r \in R \mid minCut(r) \geq 1\}$, where $i = 1$ and $R_1 = R$ initially [14]. After solving the linear program for layer $\mathcal{X}_i$, the algorithm increments i, updates the link capacities, and performs the next round of linear programming. References [8] and [9] are examples of this intra-layer coding approach.

B. Implementation of Pushback Algorithm

The pushback algorithm was implemented with two different message passing schedules.

1) Sequential: for the pushback stage, each node in the network sends a request to its parents after request messages from all its children have been received. For the code assignment stage, each node sends a code to its children after receiving codes from all its parents. This corresponds to the algorithms explained in Section III.
2) Flooding: for the pushback stage, each node updates its request to its parents upon reception of a new message from its children. For the code assignment stage, each node sends a new code to its children after receiving a new message from any of its parent nodes. This allows an update mechanism that converges to the same solution as sequential message passing, where the convergence time depends on the diameter of the graph.

Another important issue is the procedure to check decodability at each node. In general, Gauss-Jordan elimination on the coding matrix in a field of size p is necessary to determine which layers are decodable at a node after the codes are assigned. However, this is not the case for 2-layer multi-resolution codes ($L = 2$). We define *pattern of coding coefficients* for a node with δ incoming links as $[a_1, a_2, ..., a_\delta]$, where a_i represents the number of layers combined in the i-th incoming link. If a node receives only the base layer on all incoming links, *i.e.* the pattern of coding coefficients is $[1, 1, ..., 1]$, it can decode the base layer. If at least one of the incoming links contains a combination of two layers,

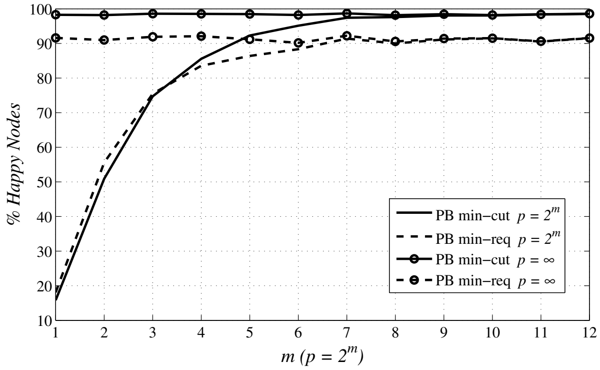

Fig. 10. Varying field size p in a network with 5 receivers and 25 nodes.

i.e. the pattern of coding coefficients is one of the following: $[1, ..., 1, 2], [1, ..., 1, 2, 2], ..., [2, ..., 2]$, both layers can be decoded. Thus, for $L = 2$, the pattern of coding coefficients indicates decodability. Note that using the pattern of coding coefficients is equivalent to using Gauss-Jordan elimination with infinite field size.

In more general cases with $L > 2$, the pattern of coding coefficients is no longer sufficient. For example, a node with 4 incoming links of unit rate can have a min-cut of at most 4. Assume that this node has a min-cut of 3, and that this node is assigned a coding-coefficient pattern of $[1, 1, 3, 3]$. If coding vectors are linearly independent, all layers are decodable. However, it is possible that the third and the fourth links, both combining three layers, are not from disjoint paths, *i.e.* linearly dependent combinations. Then, Gauss-Jordan elimination is necessary to check that only the first layer is decodable.

In subsequent sections, we present simulation results for 2 and 3-layer multi-resolution codes. However, our algorithm is not limited to 2 and 3-layers; it can be applied to general n-layer multi-resolution codes.

C. Simulation results for 2-layer multi-resolution code

The simulations for 2-layer multi-resolution code were carried out for random directed acyclic networks. We averaged 1000 trials for each data point on the curves plotted in this section. The networks were generated such that the min-cuts and the in-degrees of all nodes were less than or equal to 2.

As stated in Section V-B, the patterns of coding coefficients are sufficient to check decodability for 2-layer multi-resolution codes, and it is equivalent to using Gauss-Jordan elimination with an infinite field size. Figure 10 shows the effect of field size in a network with 25 nodes and 5 receivers by performing Gauss-Jordan elimination at every node during the code assignment stage with varying field size p. It also shows the average performance in terms of *% Happy Nodes* when using the pattern of coding coefficients to check decodability. In essence, we are comparing the performance of our system using a field size p to that of an infinite field size. Observe that even for moderately small field sizes, such as $p \geq 2^8$, the pushback algorithm performs close to that of the system operating in an infinite field.

From Figure 10, we see that the min-cut criterion performs considerably better than the min-req criterion for large field

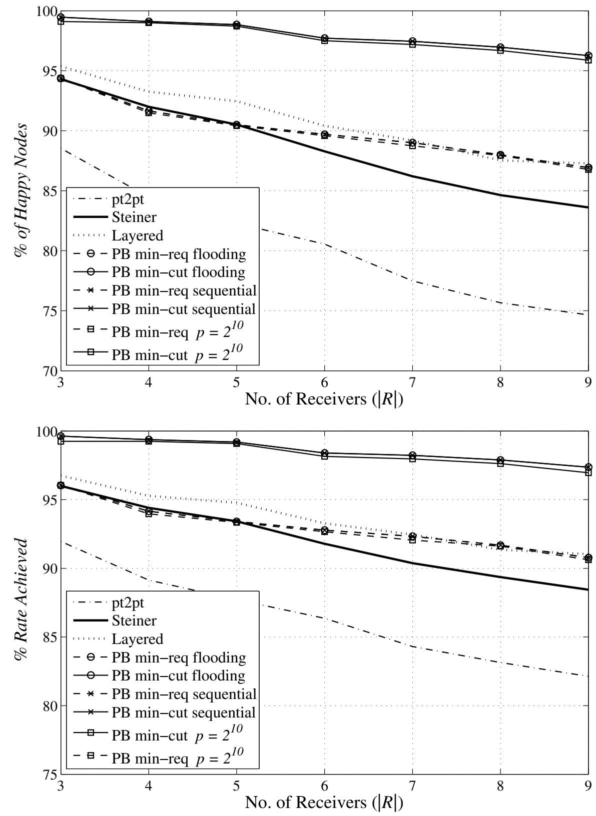

Fig. 11. Varying number of receivers in a network with 25 nodes.

sizes. However, for small field sizes ($p \leq 2^3$), the min-req criterion is slightly better. This is because it forwards the minimum of the requests received at any node. For $L = 2$, there will be more nodes requesting only the base layer when using the min-req criterion than when using the min-cut criterion. Thus, networks using the min-req criterion will have more links carrying only the base layer, which helps improve redundancy for the receivers. This allows several paths to carry the same information, ensuring the base layer is decodable at the receivers. By comparison, the min-cut criterion tries to combine both layers on as many links as possible. When the field size is large, both layers are decodable with high probability; when the field size is small, the probability of generating linearly dependent codes is high, consequently preventing decodability of both layers at a subset of receivers.

Figures 11 and 12 compare the performance of the various schemes in terms of the two metrics *% Happy Nodes* and *% Rate Achieved*. The pushback algorithm is compared to the Point-to-point Routing Algorithm ('pt2pt'), the Steiner Tree Routing Algorithm ('Steiner'), and the Intra-layer Network Coding Algorithm ('Layered'). We implement two versions of pushback: flooding and sequential message passing approaches. The flooding schemes with an infinite field size are labeled 'PB min-req flooding' and 'PB min-cut flooding' for the min-req and min-cut criteria, respectively. The sequential message passing schemes with an infinite field size are labeled 'PB min-req sequential' and 'PB min-cut sequential.' Finally, pushback using a moderate field size of $p = 2^{10}$ are labeled 'PB min-req $p = 2^{10}$' and 'PB min-cut $p = 2^{10}$.'

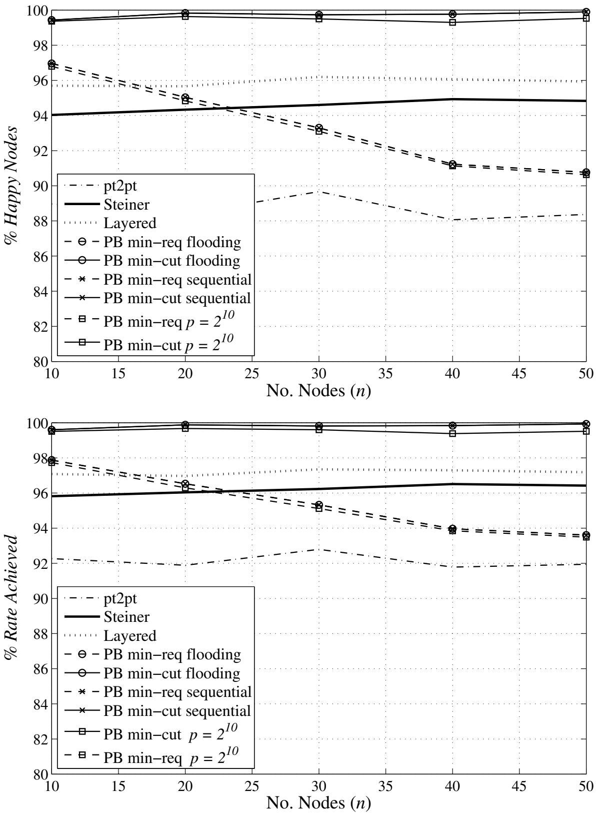

Fig. 12. Varying number of nodes in a network with 3 receivers.

Figure 11 shows the performance of the various schemes when the number of receivers is increased. 'PB min-cut' has the best performance overall. Both flooding and sequential message passing approaches behave similarly. Furthermore, using a moderate field size of $p = 2^{10}$ yields results close to that of an infinite field size (for both min-cut and min-req). The performance of the various schemes follow a similar trend for both metrics *% Happy Nodes* and *% Rate Achieved*. In addition, Figure 11 illustrates that the gaps between the min-cut criterion and 'pt2pt', 'Steiner' and 'Layered' increase with the number of receivers. Note that the gap between the min-cut and the min-req criteria increases more slowly than the gap between the min-cut and the other schemes.

Figure 12 compares the different schemes when the number of receivers is fixed, but the network grows in size. 'PB min-cut' outperforms the intra-layer network coding and the routing schemes; it also consistently achieves close to 100% for both *% Happy Nodes* and *% Rate Achieved* while the second best scheme ('Layered') achieves at most 96% and 97% for the two metrics.

Figure 12 also shows that the min-cut criterion is very robust to the size of the network. In fact, the performance improves as more nodes are added. However, the min-req degrades with the number of nodes. This is because with min-req, the requests from receivers with min-cut equal to 1 limits the rate of other receivers. As the network becomes larger, this flooding of base layer requests has a more significant effect on the throughput since more resources are wasted in delivering just the base layer. This indicates that the choice of network code can greatly impact the overall network performance, depending on its topology and demands. An inappropriate choice of network code can be detrimental, as shown by 'PB min-req'; however, an intelligent choice of network code can improve the performance significantly, as shown by 'PB min-cut'.

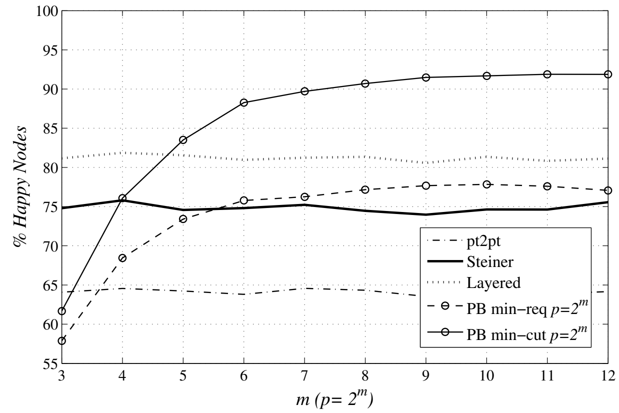

Fig. 13. Varying field size in a network with 9 receivers 25 nodes.

D. Simulation results for 3-layer multi-resolution code

Similarly to the 2-layer case, for 3-layer multi-resolution codes, we generated random networks to evaluate the pushback algorithm. For each data point in the plots, we averaged 1000 trials. The min-cuts and the in-degrees of all nodes were less than or equal to 3. Recall that with 3 layers, the patterns of coding coefficients are not sufficient for checking the decodability of incoming packets. Instead, Gauss-Jordan elimination is necessary at every node during the code assignment stage.

Figure 13 shows the effect of field size in a network of 25 nodes and 9 receivers. 'PB min-cut' outperforms routing and intra-layer coding schemes with a field size of $p = 2^5$. In terms of *% Happy Nodes*, 'PB min-cut' achieves roughly 92% when the field size is large enough, while the intra-layer coding scheme only achieves about 82%. Figure 13 also shows that intra-layer coding scheme still outperforms the routing schemes, even when optimal multicast routing is used for each layer. Our pushback algorithm achieves considerably higher gains by using inter-layer in addition to intra-layer coding.

As the number of receivers increases, more demands need to be satisfied simultaneously. It is therefore expected that the overall performance of multicast schemes will degrade with the number of receivers. This can be observed in Figures 14. However, the performance gap between 'PB min-cut' and 'PB min-req' is approximately constant, while the performance gap over other schemes increases. This means that our algorithm is more robust to changes in the number of receivers than the other schemes, an important property for systems that aim to provide service to a large number of heterogeneous users.

Figure 15 shows the performance of the different schemes as the network grows in size. As more nodes are added, there are more disjoint paths within the network for Steiner tree routing and intra-layer coding to use. Hence the performance of these schemes improves. The opposite behavior occurs for 'PB min-req', *i.e.* the *% Happy Nodes* decreases as the network size increases. This result is similar to that of Figure 12 for 2-layer

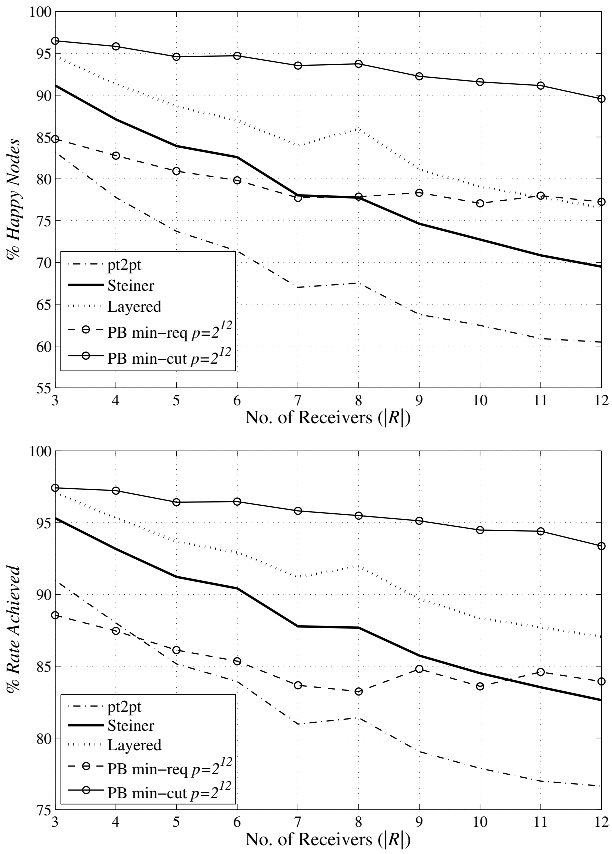

Fig. 14. Varying number of receivers in a network with 25 nodes using field size of 2^{12}.

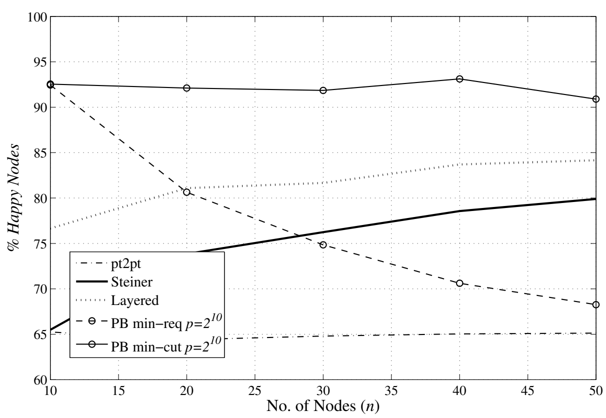

Fig. 15. Varying number of nodes in a network with 9 receivers using field size of 2^{10}.

case. As the network size increases, it becomes more likely that a small request by one receiver suppresses higher requests by many other receivers. Hence, pushback with the min-req criterion quickly deteriorates in terms of *% Happy Nodes*.

VI. Conclusions and Future work

A simple, distributed message passing algorithm, called the *pushback algorithm*, has been proposed to generate network codes for single source multicast of multi-resolution codes. With two stages, the pushback algorithm guarantees decodability of the base layer at all receivers. In terms of total rate achieved, this algorithm outperforms routing schemes as well as intra-layer coding schemes, even with small field sizes such as 2^{10}. The performance gain increases as the number of receivers increases and as the network grows in size as shown by numerical simulations.

Possible future work includes the addition of a third *complaint* stage, in which receivers whose requests have not been satisfied pass another set of requests to their parents, signaling their desire for more. In generating new codes, parent nodes must take into account the new updated requests, while maintaining decodability at receivers which did not participate in the complaint stage. It is important to determine what the complaint messages should be, and to assess the improvements that can be achieved with such an additional stage.

Another possible extension is to apply this algorithm in wireless/dynamic multicast settings. The *flooding* message passing approach is applicable then, as changes in the network can be handled by new messages to the neighboring nodes.

Lastly, in the pushback algorithm, *rate* is the message sent by nodes to their parents, *i.e.* each node signals how many layers down-stream receivers can or want to receive. It may be possible to extend the message to include other constraints, such as power (decoding power), delay, and reliability.

References

[1] A. Albanese, J. Blömer, J. Edmonds, M. Luby, and M. Sudan, “Priority encoding transmission,” *IEEE Transactions on Information Theory*, vol. 42, no. 6, pp. 1737–1744, 1996.

[2] N. Shacham, “Multipoint communication by hierarchically encoded data,” in *Proceedings of INFOCOM '92*, 1992, pp. 2107–2114.

[3] M. Effros, “Universal multiresolution source codes,” *IEEE Transactions on Information Theory*, vol. 47, no. 6, pp. 2113–2129, 2001.

[4] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, “Network information flow,” *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, July 2000.

[5] R. Kötter and M. Médard, “An algebraic approach to network coding,” *IEEE/ACM Transactions on Networking*, vol. 11, no. 5, pp. 782–795, 2003.

[6] D. Silva and F. R. Kschischang, “Rank-metric codes for priority encoding transmission with network coding,” in *Proceedings of 10th Canadian Workshop on Information Theory (CWIT'07)*, June 2007.

[7] J. M. Walsh and S. Weber, “A concatenated network coding scheme for multimedia transmission,” in *Proceedings of 4th Workshop on Network Coding, Theory and Applications (NetCod'08)*, Jan 2008.

[8] X. Wu, B. Ma, and N. Sarshar, “Rainbow network flow of multiple description codes,” *IEEE Transactions on Information Theory*, vol. 54, no. 10, pp. 4565–4574, 2008.

[9] S. B. Niveditha Sundaram, Parameswaran Ramanathan, “Multirate media stream using network coding,” in *Proceedings of 43rd Annual Allerton Conference on Communication, Control, and Computing*, 2005.

[10] J. Zhao, F. Yang, Q. Zhang, Z. Zhang, and F. Zhang, “Lion: Layered overlay multicast with network coding,” *IEEE Transactions on Multimedia*, vol. 8, no. 5, pp. 1021–1032, Oct. 2006.

[11] C. Xu, Y. Xu, C. Zhan, R. Wu, and Q. Wang, “On network coding based multirate video streaming in directed networks,” in *Proceedings of IEEE International Conference on Performance, Computing, and Communications, IPCCC'07*, April 2007.

[12] S. Dumitrescu, M. Shao, and X. Wu, “Layered multicast with interlayer network coding,” in *The 28th IEEE Conference on Computer Communications (INFOCOM'09)*, April 2009.

[13] T. Ho, R. Koetter, M. Médard, M. Effros, J. Shi, and D. Karger, “A random linear network coding approach to multicast,” *IEEE Transactions on Information Theory*, vol. 52, no. 10, pp. 4413–4430, October 2006.

[14] D. S. Lun, N. Ratnakar, M. Médard, R. Koetter, D. R. Karger, T. Ho, E. Ahmed, and F. Zhao, “Minimum-cost multicast over coded packet networks,” *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2608–2623, June 2006.