This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE Globecom 2011 proceedings.

Peer-to-Peer Content Distribution in Clustered Topologies with Source Coding

Dinh Nguyen, Hidenori Nakazato
Graduate School of Global Information and Telecommunication Studies
WASEDA University, Japan
{nqdinh@fuji., nakazato@}waseda.jp

Abstract— Network coding has been applied successfully in peer-to-peer systems to shorten the distribution time, particularly in extreme conditions where peers in the network are clustered and separated by physical links with limited bandwidth. In this paper, we focus more closely on the use of source coding, i.e. encoding is done only at the source, as an alternative to network coding to facilitate content distribution under such limited bandwidth configuration. We observe analytically and experimentally that, in this specific case, with appropriately chosen expansion factors, source coding can have comparable performance to network coding in terms of distribution time, yet consuming much less computational resources than the latter approach does.

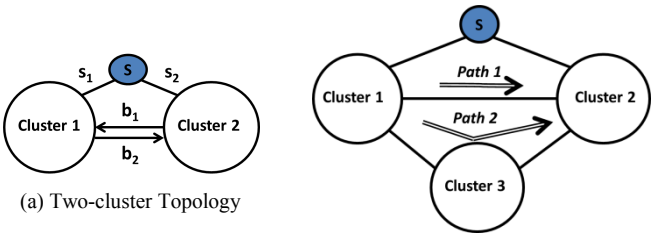
I. INTRODUCTION

Network coding [1], which allows content to be coded at intermediate nodes while being forwarded in the network, has been shown to achieve significantly shorter distribution time in peer-to-peer (P2P) content distribution [4]. One common network coding technique is random linear coding [2] in which the source and network nodes linearly combine the content they have using random coefficients before sending it to other nodes. The price to pay, however, is coding at every node.

As opposed to network coding, source coding [5][6], in the context of content distribution, only requires the source to encode with some degree of redundancy, or *expansion factor*. The encoded content, after that, is forwarded as it is inside the network without further encoding. Given its lower resource consumption, the question is, however, "does source coding performance match that of network coding?"

In this paper, we argue for the use of source coding as an alternative method for shortening content distribution time in clustered networks where peers are severely separated by physical links with limited bandwidth. We show that, with carefully chosen expansion factors, source coding can achieve performance comparable to that of network coding, nevertheless using much less resources than network coding does. We evaluate source coding performance against that of network coding in various clustered network topologies and network sizes.

In the remaining parts, we review related work in section 2. The system model is given in section 3. Section 4 focuses on how we determine the best source coding expansion factors and section 5 presents performance evaluation results. Finally, we conclude the paper in section 6.



(a) Two-cluster Topology

(b) Three-clustered Topology

Figure 1. Clustered Topologies

II. RELATED WORK

BitTorrent [3], a popular P2P file sharing with parallel downloads to accelerate download speed, divides the file into equal-size blocks, i.e. *chunks*, which peers send and receive in parallel, utilizing both available upload and download bandwidth. Each newly joining peer connects to a set of random existing peers, such that to construct a mesh overlay network with random topologies. Furthermore, rarest blocks are chosen first by receiving peers to quickly disseminate the whole file into the system. To encourage peers to contribute uploading bandwidth to the system, a peer uploads to a certain number of neighboring peers at a time, those provide it with best downloading rates.

Source coding or more precisely rateless erasure codes [5][6], which encodes data blocks at the source with some level of redundancy, and network coding [1][3], which, in addition, allows intermediate nodes to encode, have been shown to significantly improve BitTorrent file distribution on bottlenecked topologies [4]. In the specific case of clustered topologies, [4] presents impressive performance improvement of both source coding and network coding in terms of finish time, i.e. the time it takes for peers to collect enough coded blocks for decoding. Though network coding is shown to have much better performance than source coding using a fixed expansion factor by simulations, the paper omits the performance of different source coding expansion factors. In our work, we instead propose a method of determining the best expansion factors which shorten distribution time the most.

Whereas both source coding and network coding can potentially generate infinite encoding, i.e. infinite block redundancy, source coding, which encodes only at the source, has the advantage of eliminating encoding load on all network nodes. Our work goes further from [4] to answer the question how much block redundancy is enough. We examine the best

978-1-4244-9268-8/11/$26.00 ©2011 IEEE

expansion factor for source coding, which itself depends on the bandwidth available at the source and at the bottlenecks between peer clusters, and compare its performance with that of network coding.

III. System Model

A. Peer-to-Peer Content Distribution with Source Coding

We consider a P2P content distribution problem over clustered underlying topologies where peers physically belong to clusters separated by bottlenecked links. A cluster can be identified as a group of peers which have relatively high bandwidth among themselves. One such topology with two clusters is illustrated in Fig. 1(a). Each peer maintains overlay connections to some random peers, i.e. its neighbors, which can be in the same cluster or in other clusters. Peers have abundant bandwidth to neighbors in the same cluster, but narrow bandwidth to neighbors in other clusters.

To capture the essence of coding redundancy in shortening distribution time, we assume a static scenario, i.e. there is no change in the physical topology and the overlay topology during a content distribution session. The insight obtained from this static case is critically important for future work which investigates the dynamic scenario.

A file exists at the source and is distributed to all peers which, at the beginning, do not have any part of the file. The file is divided into k equal blocks, the same as in [3], which are encoded by the source and transferred in the system in parallel. To put some degree of redundancy to the system, the number of the encoded blocks that the source generates, $(1+e)k$, is greater than the number of the original blocks. We refer to the factor $1+e$ as redundancy ratio or expansion factor interchangeably in this paper.

The source encodes in such a way that a receiver can decode the file after it receives any k out of all the encoded blocks, where k is the number of original blocks[1]. One practical way is to apply the random linear coding scheme [2] at the source to combine original blocks linearly using coefficients picked up uniformly at random and send the coefficients along with the encoded block. A peer can recover the original file after it has received any k linearly independent coded blocks.

Since the resources at the source are invaluable, e.g. in case it has several files to serve, the source is assumed to leave right after it has sent all the encoded blocks. A peer finishes when it has collected enough coded blocks required for decoding. We assume an altruistic system where peers stay and forward blocks even after they have finished downloading.

As in BitTorrent systems [3], [4], block exchange complies with two rules: (1) rarest block first selection at the receiver's side: receivers choose rarest blocks within its neighborhood to download, and (2) a certain incentive scheme at the sender's side: senders send blocks to their neighbors reciprocally.

B. Encoding/Decoding Complexity

We note that whereas random linear coding, a method of network coding, is performed by several nodes in the network, in our source coding system, only the source does encode, and the other nodes just forward the coded blocks. This kind of source coding, which uses linear codes, is more computation-intensive than that in [5][6], but apparently is much less resource consuming than network coding solutions which impose coding everywhere in the network [4]. The decoding complexity, however, is the same for both source coding and network coding systems because all nodes are required to decode. Table I compares those encoding/decoding complexities assuming the system uses random linear coding for encoding and Gaussian elimination in the decoding phase[2].

TABLE I. Encoding/Decoding Complexity

	Encoding Complexity	Decoding Complexity
Source Coding	$O(k^2B)$	$O(Nk^2B)$
Network Coding	$O(Nk^2B)$	$O(Nk^2B)$

N is the number of nodes in the system, k is the number of original blocks, and B is the block size.

Notice that in the source coding system the encoding complexity is independent of the network size whereas in network coding systems, it is N times as much and increases linearly with the number of nodes in the system.

IV. Source Coding Expansion Factor

In this section, we describe in detail how to figure the best source coding redundancy ratio which shortens finish time the most, gradually from simple clustered topologies to general ones. As bandwidth within a cluster is, by assumption, abundant, we can safely collapse each cluster to one *cluster node* and determine the best source coding expansion factor on the topology over those *cluster nodes*.

Assuming we know the source bandwidth to each cluster, and the bottlenecked bandwidth between any pair of clusters, our problem can be stated as follows.

Given
- a network topology with one single source and n clusters,
- the source bandwidth to each cluster: $s_1, s_2, .., s_n$, and
- bandwidth of the links between clusters: $u_{i,j}$,

what is the best source coding redundancy ratio to achieve shortest distribution time?

The notations in Table II are used in our analysis. For the sake of presentation, let b_i be the total bottlenecked bandwidth from all the other clusters to cluster i. We have $b_i = maxflow(i) - s_i$ where $maxflow(i)$ is the max-flow from the source to cluster i, which is determined by the max-flow min-cut theorem [7].

A. Topologies where All Clusters Connect to the Source

We first analyze the finish time of source coding on a topology with two clusters (as in Fig. 1(a)). We are interested in minimizing the finish time of peers in the slowest cluster.

Without loss of generality, assume cluster 2 is the slowest cluster, i.e. total bandwidth to cluster 2 is not greater than that of cluster 1: $s_2+b_2 \leq s_1+b_1$. If the source stays forever, cluster 2 will finish at time $T_2 = \frac{k}{s_2+b_2}$ after downloading k blocks through both the bottlenecked link from cluster 1 and the link

[1] Rateless erasure codes may require slightly more than k blocks for successful decoding.

[2] The reasoning is omitted due to the paper length constraint.

TABLE II. NOTATIONS

Notation	Meaning
1, 2,.., n	Cluser IDs
$s_1, s_2, .., s_n$	Bandwidth of the link from the source to each cluster, s_i=0 if cluster *i* does not have a direct link to the source
$u_{i,j}$	Bandwidth of the link from cluster *i* to cluster *j*
maxflow(i)	Max-flow from the source to cluster *i*
$b_1, b_2, .., b_n$	Total bottlenecked bandwidth to each cluser b_i=maxflow(i)-s_i
p_i	Bottlnecked bandwidth of an augmenting path *i* between two particular clusters
k	Number of original blocks
1+e	Redundancy ratio, the *k* original blocks are encoded into *(1+e)k* coded blocks by the source
t_S	The time when the source finishes sending all the encoded blocks and leaves (time starts from 0)

with the source. However, the source does leave after sending all the encoded blocks at time t_S. If cluster 2 does not finish by that time, it will have to download the remaining blocks from cluster 1 through the bottleneck, which takes much longer time. There are two cases.

Case 1: $t_S \le T_2$, the source leaves no later than the time it takes cluster 2 to finish. When the source leaves at t_S, cluster 2 has downloaded $s_2 t_S$ blocks from the source, the remaining $k - s_2 t_S$ blocks have to be downloaded through the bottleneck which finishes at time $\frac{k-s_2 t_S}{b_2}$. The finish time $\frac{k-s_2 t_S}{b_2}$ is minimized when $t_s = T_2$; then we have $\frac{k-s_2 t_S}{b_2} = \frac{k}{s_2+b_2}$.

We need to compute the number of coded blocks that the source has sent at time $t_s = T_2$. Since cluster 1 finishes at time $T_1 = \frac{k}{s_1+b_1}$ which is no later than T_2, the number of blocks that it downloads directly from the source is $\frac{ks_1}{s_1+b_1}$. Likewise the number of blocks that cluster 2 has downloaded directly from the source at time T_2 is $\frac{ks_2}{s_2+b_2}$. We have the total number of blocks the source has distributed $(1+e)k = \frac{ks_1}{s_1+b_1} + \frac{ks_2}{s_2+b_2}$, and the redundancy ratio, therefore, is $1 + e = \frac{s_1}{s_1+b_1} + \frac{s_2}{s_2+b_2}$.

Case 2: $t_S > T_2$, the source leaves after cluster 2 has finished at time $T_2 = \frac{k}{s_2+b_2}$. There is no improvement in finish time, which is $\frac{k}{s_2+b_2}$, compared to case 1, though the redundancy is higher.

Conclusively, the best expansion factor can be determined by
$$1 + e^* = \frac{s_1}{s_1+b_1} + \frac{s_2}{s_2+b_2} \quad (1)$$
where $b_1 = min(s_2, u_{2,1})$, and similarly $b_2 = min(s_1, u_{1,2})$ (which are the values of *maxflow(1)*-s_1 and *maxflow(2)*-s_2 respectively).

In *n*-cluster topologies which have *n* clusters, all connect directly to the source, using the same reasoning, we can generalize the result in (1) to obtain the best expansion factor
$$1 + e^* = \sum_{i=1}^{n} \frac{s_i}{s_i + b_i}. \quad (2)$$

We now move to general clustered topologies in which some clusters have connections to the source, the others do not.

B. General Clustered Topologies

Our idea is to approximate general clustered topologies with *n*-cluster topologies in the previous subsection. Since duplicated blocks might be transferred on multiple paths to a cluster *i*, actual bottlenecked bandwidth to that cluster, which we call *effective* bottlenecked bandwidth b_{ieff}, is smaller than the total bandwidth of all the paths. Once b_{ieff} is determined, we can apply (2) to compute the best expansion factor.

1) Effective Bottlenecked Bandwidth

Fig. 1(b) illustrates two paths between cluster 1 and cluster 2 in a topology with three clusters: the direct path cluster 1 – cluster 2, namely *path 1*, and the indirect path cluster 1 – cluster 3 – cluster 2, namely *path 2*. Consider blocks travelling from cluster 1 to cluster 2. Since some blocks available in cluster 1 are picked up twice by both cluster 2 and cluster 3, the effective bottlenecked bandwidth to cluster 2, namely b_{2eff}, is smaller than the total bandwidth b_2 of the two mentioned paths.

Denote *N(t)* as the number of blocks cluster 1 has downloaded from the source but has not retrieved by cluster 2 and *a(t)* as the number of non-duplicated blocks that cluster 2 obtains from cluster 1 through the two bottlenecked paths at time *t*. The numbers of blocks that cluster 2 downloaded directly from cluster 1 on *path 1* is $p_1 t$ and via cluster 3, on *path 2*, is $p_2 t$, where p_1 and p_2 are the bottlenecked bandwidth of the two paths: $p_1 + p_2 = b_2$. For analysis, assuming blocks are picked up uniformly at random, the number of duplicated blocks is $\frac{p_1 p_2 t^2}{N(t)}$. The total non-duplicated blocks cluster 2 receives from cluster 1 is $a(t) = (p_1 + p_2)t - \frac{p_1 p_2 t^2}{N(t)}$. (3)

The number of blocks available at cluster 1 which cluster 2 has not downloaded at time *t* is $N(t) = s_1 t - b_{2eff} t$ (4)
where s_1 is the bandwidth from the source to cluster 1 and b_{2eff} is the effective bottlenecked bandwidth to cluster 2 which we are to estimate. From (3) and (4), we have
$$b_{2eff} = \frac{a(t)}{t} = p_1 + p_2 - \frac{p_1 p_2 t}{N(t)} = p_1 + p_2 - \frac{p_1 p_2}{s_1 - b_{2eff}}. \quad (5)$$

Equation (5) can be generalized to the case there are *m* paths, *path 1, .., path m*, between clusters 1 and cluster 2:
$$b_{2eff} = p_1 + p_2 + .. + p_m - \frac{p_1 p_2 + p_1 p_3 + .. + p_{m-1} p_m}{s_1 - b_{2eff}} + \frac{p_1 p_2 p_3 + .. + p_{m-2} p_{m-1} p_m}{(s_1 - b_{2eff})^2} - .. - \frac{(-1)^m p_1 p_2 .. p_m}{(s_1 - b_{2eff})^{m-1}}. \quad (6)$$

Denote $s_1 - b_{2eff} = x$, (6) can be solved by using Newton-Raphson method [9] to find a root in $[s_1 - (p_1 + .. + p_m), s_1]$ of
$$f(x) = 2x^m - s_1 x^{m-1} - (x - p_1)(x - p_2)..(x - p_m).$$

2) Expansion Factor in General Clustered Topologies

The best expansion factor in general clustered topologies is
$$1 + e^* = \sum_{i=1}^{n} \frac{s_i}{s_i + b_{ieff}} \quad (7)$$

where b_{ieff} is the effective bottlenecked bandwidth to cluster i computed by the technique given above. Table III summarizes the algorithm to compute the best expansion factor.

TABLE III. ALGORITHM TO COMPUTE EXPANSION FACTOR IN GENERAL CLUSTERED TOPOLOGIES

	Steps
1.	Determine the set of clusters which directly connect to the source (set A). The algorithm operates on this set A.
2.	Compute effective bottlenecked bandwidth to each cluster j in set A.
a.	Determine all augmenting paths from the source to cluster j using Edmonds–Karp algorithm [7].
b.	For each cluster i in set A, $i \neq j$, compute the effective bottlenecked bandwidth from cluster i to cluster j using (6).
c.	The effective bottlenecked bandwidth to cluster j is the sum of all effective bottlenecked bandwidths computed in step 2b.
3.	Compute the best expansion factor based on the source bandwidth and effective bottlenecked bandwidths using (7).

We have to note that the expansion factor given in (7) minimizes the finish time of the slowest cluster which directly connects to the source, not the overall slowest cluster. Clusters which have no direct connections to the source, e.g. cluster 3 in Fig. 1(b), have to download all required blocks through the bottlenecked links to other clusters regardless of how much block redundancy is generated by the source and do not gain much from source coding. We may be able to speed up download by placing additional coding nodes, however. In this paper, we concentrate on speeding up download in clusters with direct connections to the source. Utilizing and extending the findings in this simpler case, we hope to find out the placement of additional coding nodes for clusters without direct connections to the source.

V. PERFORMANCE EVALUATION

We implement a C++ simulator [3] of the P2P content distribution system described in section III and run simulations over various clustered topologies with different network sizes distributing a file from the source to all participating peers in three scenarios: *no coding*, *source coding*, and *network coding*. The file is divided into smaller fix-sized parts, i.e. blocks. The source and all peers exchange blocks until all peers acquire enough blocks to construct the original file; then the simulation finishes. The simulation settings are the same for all three scenarios; the difference however is where encoding is done.

- *No coding* (BT) - coding is not deployed in the system; the original blocks are exchanged as in pure BitTorrent systems [3]. This is the baseline scenario used to evaluate the performance of source coding and network coding.
- *Source coding* (SC) - only the source generates encoded blocks whose number depends on the desired redundancy; the encoded blocks are then exchanged as a whole.
- *Network coding* (NC) – all peers create new encoded blocks when they want to send data to their neighboring peers; encoding at the source is the same as in the source coding scenario.

[3] The simulator can be obtained by contacting the first author.

The simulations are round-based. Blocks are assigned to each peer according to available bandwidth, rarest block first selection, and the incentive scheme in the beginning of each round. The assigned blocks are downloaded by the peer at the end of the round and then the system moves to next round. After a peer has collected enough blocks, it stops downloading but keeps staying in the system to serve other peers. A link capacity is measured by block per round, i.e. how many blocks can be transferred through the link in a round. We disregard the negligible overhead of sending encoding coefficients associated with random linear coding in our simulations.

We implement *mutual exchange* incentive scheme in the simulations: when there is contention for uploading, a sending peer preferably uploads to the neighbors from whom it is also downloading. After such peers are exhausted, other neighbors are chosen for upload. This kind of incentive schemes has previously been used in [4]. In addition, to ensure that the source quickly disseminates new blocks into the network, we enforce at the source a scheme which works like the super seeding scheme [8]. Each time there is a request to download a new block, the source tries to serve with a random block which has never been sent into the network.

For a given network size, we run simulations 100 times, each time with a random overlay topology (which is reused in all three scenarios) and collect the average finish time in each scenario. Within a cluster, peers are arranged in k-regular random topologies where k is from 3 to 6. In case of *source coding*, the source leaves after sending all the encoded blocks. To set the ground for comparison, in the other cases (*no coding* and *network coding*), the source leaves after having sent the same number of blocks as in the case of *source coding*. We use simplified clusters topologies as building blocks to show the effectiveness of source coding. Realistic topologies can be broken down into those simpler ones.

A. Two-cluster Topologies

We first evaluate performance in simple topologies of two clusters. The topology in Fig. 1(a) is used as the underlying physical network in our simulations. Source bandwidth to each cluster is 4 blocks per round. Bottleneck bandwidth between clusters is 1 block/round. Each cluster has 100 to 500 nodes with 8 blocks per round bandwidth among peers within a cluster[4]. The source has a 200-block file which is encoded into an appropriate number of coded blocks depending on the chosen redundancy at the beginning of the distribution section.

In Fig. 2, with the best redundancy ratio 1.6, *network coding* and *source coding* have almost the same finish time in all network sizes, which achieves an improvement of approximately 20% over *no coding*. Fig. 4 compares *source coding* and *network coding* in terms of finish time and improvement over *no coding* with different expansion factors (averaged over 500 runs with 500 overlay topologies in 5 network sizes). In all three scenarios, with higher expansion factors the finish time is shorter; however, there is a knee in finish time of *source coding* and *network coding* at expansion factor 1.6 (the best by analysis) where the improvement over *no coding* systems is highest. The finish time improvement of

[4] Similar results are observed when sizes vary with clusters.

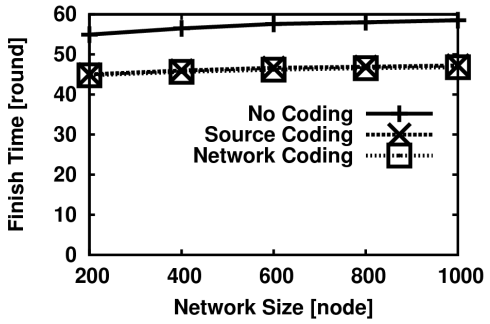


Figure 2. Finish Time with the Best Expansion Factor

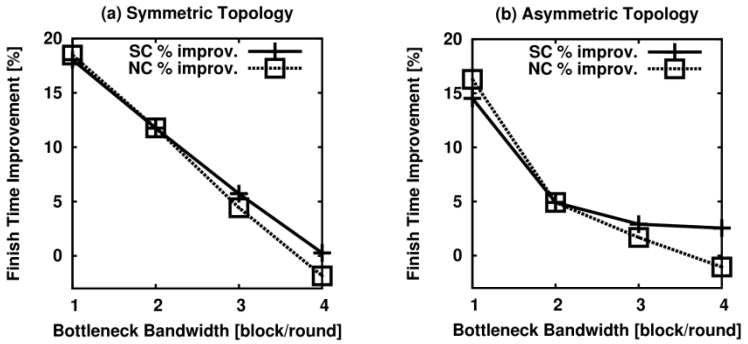


Figure 3. Improvement with Different Bottleneck Bandwidth

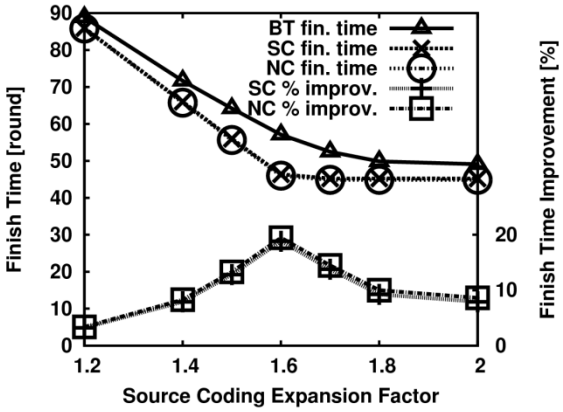


Figure 4. Source Coding vs. Network Coding

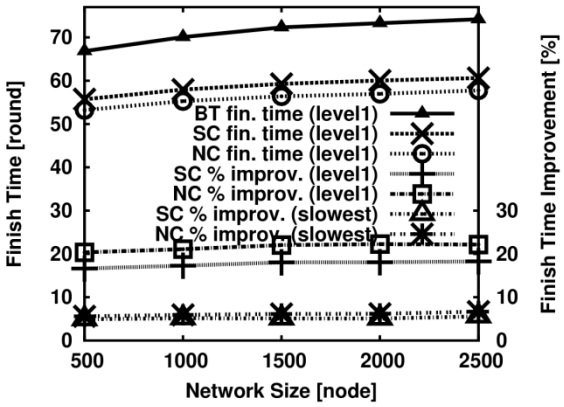


Figure 5. Improvement in a General Clustered Topology

both *source coding* and *network coding* are almost the same and largely depend on the redundancy at the source. We change the bottleneck bandwidth between the 2 clusters (Fig. 3(a)), and moreover, run simulations in an asymmetric topology (Fig. 3(b)) where the 2 links from the source have bandwidth of 5 and 3 blocks per round respectively. The results show that redundancy in both source coding and network coding is less effective with bigger bottleneck bandwidths.

B. General Clustered Topologies

We create more complex, asymmetric clustered topologies by connecting 5 clusters with random bottleneck links whose bandwidth is between 0 (unconnected) and 2 blocks per round. Of the 5 clusters, three have direct connections to the source sharing a total bandwidth of 10 blocks per round. Each individual link from the source is assigned a random bandwidth from 2 blocks per round to 5 blocks per round. The bandwidth within a cluster is set to 10 blocks per round. The file size is 200 blocks. We use the algorithm in Table III to figure the best source coding expansion factors.

Fig. 5 presents results in one such topology. At around 20% improvement, source coding performs almost the same as network coding in terms of finish time improvement of the slowest cluster which connects directly to the source (marked as *level1* in the figure). Source coding redundancy (and also network coding) is less effective to clusters without direct connections to the source with lower than 10% improvement (marked as *slowest*) as we have discussed in section IV.

VI. CONCLUSIONS

We argue for the use of source coding, a much less computation-consuming method compared with network coding, for putting redundancy to the system in order to shorten content distribution time over clustered P2P networks. Source coding, by eliminating the encoding phase at all peers, consumes much less system-wide computational resources than network coding. To ensure an equivalent performance to network coding, we have figured the best source coding expansion factors which shorten the distribution time the most in clustered networks. Evaluation consistently confirms source coding performance, matching that of network coding, in various clustered topologies and different network sizes.

REFERENCES

[1] R. Ahlswede, N. Cai, S. R. Li, and R. W. Yeung, “Network Information Flow”, IEEE Transactions on Information Theory, July 2000.

[2] T. Ho, R. Koetter, M. Medard, D. Karger, and M. Effros, “The Benefits of Coding over Routing in a Randomized Setting”, ISIT, Yokohama, Japan, 2003.

[3] B. Cohen, “Incentives Build Robustness in BitTorrent”, P2P Economics Workshop, 2003.

[4] C. Gkantsidis and P. R. Rodriguez, "Network Coding for Large Scale Content Distribution", IEEE INFOCOM, March 2005

[5] P. Maymounkov and D. Mazires, “Rateless Codes and Big Downloads”, IPTPS’03, February 2003.

[6] M. Luby, “LT Codes”, in the 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002.

[7] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, “Introduction to Algorithms”, second ed., MIT Press and McGraw–Hill, ch. 26.2, 2001, pp. 660–663.

[8] BitTornado, www.bittornado.com.

[9] A. S. Householder, “Principles of Numerical Analysis”, New York: McGraw-Hill, 1953, pp. 135-138.