CENTRALITY-BASED NETWORK CODER PLACEMENT FOR PEER-TO-PEER CONTENT DISTRIBUTION

Dinh Nguyen and Hidenori Nakazato

Graduate School of Global Information and Telecommunication Studies
Waseda University, Tokyo, Japan

{nqdinh@fuji., nakazato@}waseda.jp

ABSTRACT

Network coding has been shown to achieve optimal multicast throughput, yet at an expensive computation cost: every node in the network has to code. Interested in minimizing resource consumption of network coding while maintaining its performance, in this paper, we propose a practical network coder placement algorithm which achieves comparable content distribution time as network coding, and at the same time, substantially reduces the number of network coders compared to a full network coding solution in which all peers have to encode, i.e. become encoders. Our algorithm is derived from two key elements. First, it is based on the insight that coding at upstream peers eliminates information duplication to downstream peers, which results in efficient content distribution. Second, our placement strategy exploits centrality characteristics of the network topology to quickly determine key positions to place encoders. Performance evaluation using various topology and algorithm parameters confirms the effectiveness of our proposed method.

KEYWORDS

peer-to-peer, content distribution, network coding

1. INTRODUCTION

In contrast with the current store-and-forward routing model, network coding [1, 2], a new approach to maximize throughput in a multicast network,allows network nodes to code, i.e. generating new information from what they have received, and after that, forward the coded information into the network. Each time a node wants to send data, it has to, for example, linearly combine currently available data by a series of numerical multiplications and additions, consuming certain amount of computational resources. Although having been shown to achieve maximum multicast throughput [1–3], network coding incurs an expensive computation cost: practically every node in the network has to code.

In this paper, we study the use of network coding to speed up content distribution in peer-to-peer (P2P) networks. Our goal is to get the underlying reason for network coding's improved performance in P2P content distribution and to optimize resource consumption of network coding.When all peers are allowed to encode, network coding has been shown to significantly shorten distribution time in peer-to-peer content distribution [4]. As encoding's complexity and resource consumption are proportional to the number of encoders[1], the question is "can we minimize the number of network coders while maintaining the optimal performance of network coding?"

1 We use the terms *network coder* and *encoder* interchangeably in this paper.

DOI : 10.5121/ijcnc.2013.5312

Instead of requiring all peers to encode, i.e. placing encoders at every peer, we propose a novel practical algorithm to selectively place network coders at only key locations inside a P2P content distribution network. The objective is shortest distribution time with constraints on the number of encoders.

Our placement algorithm, on the one hand, has its roots in the advantage of coding to eliminate content duplication, which is the cause of performance improvement by network coding. Coding at an upstream peer can eliminate content duplication over multiple paths to a downstream peer and result in efficient content distribution. To identify nodes which lie on multiple paths to other nodes to place network coders, our proposed method, on the other hand, exploits *betweenness centrality* [5] and *flow centrality*[6]to effectively pinpoint the desired key positions in the network.

Our contributions are

1. we demonstrate that a considerable number of encoders, which means computational resources, can be saved while still achieving performance comparable to full network coding where coders are placed at all peers, and
2. we confirm that betweenness centrality and flow centrality are good indicators of where we can effectively place encoders in a P2P network.

The remaining parts of this paper are organized as follows. We review related work in section 2 and describe our system model and assumptions in section 3. We state our coder placement problem in section 4 and then go on to investigate block duplication in P2P content distribution in section 5. Our proposed coder placement algorithm comes in section 6. Section 7 presents evaluation results. Finally, we conclude the paper in section 8.

2. RELATED WORK

BitTorrent [7], a popular P2P file sharing with parallel downloads to accelerate download speed, divides the file into equal-size pieces, i.e. *blocks*, 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. 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.

Network coding [1–4], which allows intermediate nodes to code, have been shown, in theory, to achieve asymptotically optimal content distribution time. In practice, experimental evidence in [4] confirms that network coding can remarkably improve BitTorrent file distribution, especially in clustered topologies where there is limited bandwidth between sets of peers. One interesting observation is that substantial performance gain is evident even when only the source is allowed to code, i.e. *source coding*. Nevertheless, the paper omits concrete explanation for what underlies network coding's good performance, and, more interestingly what we can expect if a constrained number of peers are allowed to encode. In [9, 10], source coding is also applied to improve BitTorrent without incurring encoding at intermediate peers. Those results suggest full-scale network coding where all nodes are required to code might be more than what we need to achieve such performance.
Motivated by the question "can we practically achieve the performance of network coding without requiring all nodes to encode", we first identify topological conditions under which

coding can increase throughput, and then, given that insight, we furthermore propose a fast, practical coder placement method that achieves comparable performance in terms of finish time as network coding while using much less computational resources than network coding does.

Closely related to our work, Kim et al. [11] proposed algorithm to determine a minimal set of nodes where coding is required to achieve the maximum multicast rate. Their method, nevertheless, is based on a genetic algorithm, which barely offers any insight into how network coding improves performance. Bhattad et al. [12] decomposed a multicast solution into flows to subsets of receivers and construct a linear programming problem for minimal network coding. Their approach is applicable only in multicast networks with a small number of receivers since the complexity grows exponentially with the number of receivers. Lun et al. [13] present methods for computing subgraphs over which network coding is deployed. Their primary concern is to minimize the cost associated with bandwidth utilization on network links. Moreover, the model assumes full network coding deployment to achieve maximum multicast rate which is not suitable in case only a subset of network nodes is allowed to code.Recently, Martalo et al. [14] figure network coding complexity, i.e.the minimal number of coding nodes, and its relation to the multicast capacity and the number of receivers in random network topologies. Their evaluation, however, is limited to the case of acyclic networks which is not applicable in P2P overlay networks where circles and loops prevail.

In another direction, Small and Li [15], Niu and Li [16], and Crisostomo et al. [17] study network coding efficiency in different topology settings. More recently, Maheshwar et al. [18] study network coding in a combination network topology and show that the coding advantage, i.e. improving multicast throughput, and the cost advantage, i.e. reducing multicast cost, are upper-bounded by a constant.Unlike them, we aim to locate the best places to assign encoders in any topology in order to improve performance.

Cleju et al. in [19] propose coder placement algorithms to minimize streaming delay in a push-based, sender-driven overlay network. The intermediate nodes in their system, however, are not interested in the content and only act as helpers to the system and their problem is limited to direct acyclic network topologies. In our system, all peers are receivers who actively select which parts of the content they want to download. We also do not impose any constraint on the topologies which practically are random meshes where one peer connects to others at random. Maymounkov et al. [20], Champel et al. [21], and Silva et al. [22] devise network coding methods to achieve better computational efficiency. Our solution, nevertheless, will further save computational resources by reducing the number of encoders.

3. SYSTEM MODEL

3.1. P2P Content Distribution

We consider a P2P content distribution problem from one source to many peers where each peer maintains overlay links to some other peers at random, i.e. its neighbors, over which data are transferred.

Since our placement algorithm works on the network topology to find the best places for network encoders, we assume complete knowledge of the overlay topology and bandwidth capacity of each overlay link. To capture the essence of network coding in shortening distribution time, we assume a static scenario, i.e. there is no change in both the physical topology and the overlay topology during a content distribution session. The insight obtained from this static, centralized

case is critically important for future work which investigates the dynamic and distributed scenarios.

A file exists at a single 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 [7], which are transferred in the system in parallel.

Whenever a peer has available bandwidth to download, it chooses a number of rarest blocks within its neighborhood and requests each block from the corresponding neighbor. Based on its available upload bandwidth and the incentive scheme, the neighbor will permit the download or not. If a peer fails to request a block from a neighbor, it tries with other neighbors who also have the block it interests in. If that also fails, the peer will pick up the next rarest block as a substitute. As in BitTorrent systems [4, 7], 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) an incentive scheme at the sender's side: senders send blocks to their neighbors reciprocally.

We consider three scenarios.

1. *No coding*– coding is not allowed in the system, i.e. all peers send and receive original blocks as in a pure P2P system. A peer finishes when it has collected all the original blocks.
2. *Network coding*– all peers, including the source, are allowed to encode, i.e. combine downloaded blocks to make new encoded blocks and send to other peers. A peer finishes when it has collected enough coded blocks required for decoding.
3. *Selective coding* (proposed) – the same as network coding except that only some chosen peers, including the source, are allowed to encode.

Selective coding may require a centralized server for coder assignment. We leave that implementation issue for future work and focus hereafter on where to place coders.

We assume an altruistic system where peers stay and forward blocks even after they have finished downloading. The source does also stay in the system until all peers finish.

3.2. Random Linear Network Coding

When a chosen peer is allowed to code, or in other words, a coder is placed at the selected peer, the peer uses random linear network coding (RLNC) [3, 8] to create new coded blocks from the blocks it has received.

Using RLNC, an *encoding vector* of K coefficients is attached to each coded block to specify how that coded block is generated from the K original blocks. The coded block together with its encoding vector is then sent to the requesting peer. As in [4], in our system, before requesting a coded blocks, peers check the encoding vector of that block to verify if it is linearly independent to what they have in order to avoid downloading duplicated meaningless blocks.

In case of *selective coding* where some nodes encode and the others do not, receiving peers prefer coded blocks over non-coded, original ones. The reason is because coded blocks, created by randomly combining multiple blocks, can improve throughput by eliminating block duplication as we discuss later.

After a peer collects K independent coded blocks, i.e. the K associated encoding vectors form a full-rank matrix, it can decode to get the K original blocks, i.e. the original file, by solving the set of K linear equations.

4. CODER PLACEMENT PROBLEM

Network coding's optimality does not come without a price: every node has to encode, i.e. become a coder. Our goal is to maintain the optimality of network coding while substantially reducing the number of coders.

Assuming the system uses random linear coding, the encoding complexity of the system is $O(CFK)$ where C is the number of encoders in the system, K is the number of original blocks, and F is the file size. Since F is fixed for a given file, and K cannot be set too low, it is important to minimize the number of coders to reduce the encoding complexity.

Peers in the P2P network form a directed overlay topology, i.e. directed graph $G = \{V, E\}$ where V is the set of peers, or nodes, and E is the set of directed overlay links between peers. Our coder placement problem can be stated as follows.

Given a P2P content distribution which is defined by

- a network topology $G = \{V, E\}$,
- a source in V with a file of size F to be distributed to all peers, and
- a number C ($1 \leq C \leq |V|$),

where in the network topology can we place C coders in order to shorten distribution time the most?

Since our problem is as hard as the problem of finding a placement with shortest finish time and minimum number of coders which is proved to be NP-hard [23], we aim at heuristic placements which exploit the characteristics of the network topology.

In the next section, we discuss the block duplication phenomenon observed in pure, non-coding BitTorrent systems which is the rationale behind our proposed coder placements to eliminate the duplication.

5. BLOCK DUPLICATION IN BITTORRENT

Peers in pure BitTorrent choose blocks to download in a distributed manner based on their own perception that those blocks are rare in the neighborhood. Without a global knowledge, when there are multiple downstream paths to a particular node, some blocks are downloaded multiple times by upstream peers on those paths, which results in insufficiency of new information flow coming to the downstream node. Because of duplicated data, the downstream node cannot utilize its full downloading capacity. This duplication phenomenon has been illustrated in [1, 4]. Nevertheless, in this section, we distinctively figure how much duplication a given node generates, which is the foundation of our proposed coder placement.

A path, without circles or loops, from node i to node j is a sequence of nodes starting from i and terminating at j in which two adjacent nodes are connected by a link. A flow on a path from node i to node $j$is a mapping $E \rightarrow \boldsymbol{R}^+$ which conforms to capacity constraint of each link and flow

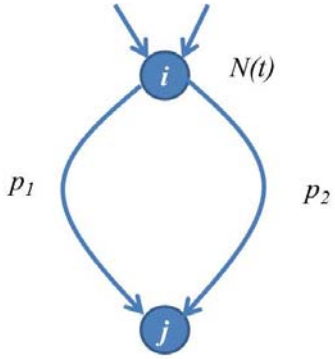


Figure 1.Two paths connect node *i* and node *j*.

conservation at each node on the path. A *max-flow* is the flow with maximum value. Figure 1 illustrates two paths connecting node *i* and node *j*: path 1 and path 2 with two respective flows of p_1 and p_2.

Denote *N(t)* as the total number of blocks available at node *i* by time *t*. Since nodes on one path do not know which blocks have been chosen by nodes on the other path, we can assume blocks are picked up at random: p_1t random blocks are chosen from *N(t)* to transmit on path 1, and likewise, p_2t random blocks are transmitted on path 2 by time *t*. The expected number of duplicated blocks transmitting on the two paths, therefore, is $\frac{p_1t.p_2t}{N(t)}$.

The total number of non-duplicated blocks from node *i* which are delivered to node *j* by time *t* is

$$a(t) = p_1t + p_2t - \frac{p_1p_2t^2}{N(t)} \tag{1}$$

Let s_i be the rate at which blocks coming to node *i*, we have the number of blocks available at node *i* by time *t*: $N(t) = s_it$. From (1), the effective throughput (averaged over time *t*) from node *i* to node *j* is

$$\begin{aligned} p_{eff} &= \frac{a(t)}{t} \\ &= p_1 + p_2 - \frac{p_1p_2}{s_i} \end{aligned} \tag{2}$$

By the same reasoning, (2) can be generalized to get the effective bandwidth in case there are *m* paths connecting node *i* and node *j*

$$\begin{aligned} p_{eff} = p_1 + p_2 + .. + p_m - \frac{p_1p_2 + p_1p_3 + .. + p_{m-1}p_m}{s_i} \\ + \frac{p_1p_2p_3 + .. + p_{m-2}p_{m-1}p_m}{{s_i}^2} - .. - \frac{(-1)^m p_1p_2..p_m}{{s_i}^{m-1}}. \end{aligned} \tag{3}$$

Equation (3) reveals that due to duplicated blocks on the paths, the effective throughput p_{eff} is smaller than the total flows on all paths from node *i* to node *j*:

$$p_{eff} = p_1 + p_2 + .. + p_m - r \tag{4}$$

where$r$>0 is the duplication rate.

From (3) and (4), we have

$$r = \frac{p_1p_2 + p_1p_3 + .. + p_{m-1}p_m}{s_i} - \frac{p_1p_2p_3 + .. + p_{m-2}p_{m-1}p_m}{{s_i}^2} + .. + \frac{(-1)^m p_1p_2..p_m}{{s_i}^{m-1}}. \quad (5)$$

There are two observations on the correlations of duplication rate r with consisting flows which contribute to the creation of our coder placement algorithms.

First, duplication rate is higher with larger consisting flows. If we consider a given flow p_i separately and fix all other flows, (5) can be converted to

$$r = A_i p_i + B_i \quad (6)$$

whereA_i and B_i are independent from p_i, and A_i>0, B_i>0. Equation (6) shows the correlation of duplication rate and each separate flow from node i to node j: when a given flow p_i increases, duplication rate r also increases.

Second, duplication rate is higher if there are more flows from node i to node j. Let $r(m)$ and $p_{eff}(m)$ respectively be the duplication rate and effective throughput with m flows from node i to node j: $p_1, p_2, .., p_m$ and $r(m+1)$ be the duplication rate when a new flow $p_{m+1}$is added. It is easy to see that

$$r(m+1) = r(m) + \frac{p_{eff}(m).p_{m+1}}{s_i} \quad (7)$$

which means $r(m+1) > r(m)$.

Therefore, we have

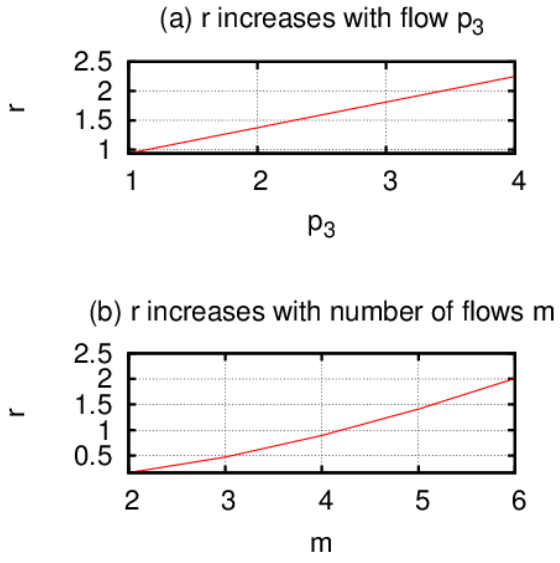


Figure 2. Duplication rate increases with flow size and number of flows.

$$r(l) > r(m)\ \forall l > m. \quad (8)$$

We illustrate the correlation in Figure 2(a) when there are 3 flows p_1, p_2, and p_3 from node i to node j: s_i=8, p_1= p_2=2 and p_3 changes from 1 to 4. In Figure 2(b), we fix s_i=6, p_1= p_2=1 and add more flows with bandwidth equal to 1 to change the number of flows m from 2 to 6.

6. PROPOSED CODER PLACEMENTS

6.1. Coding at the Network Centrality

As shown in section 5, we can achieve high throughput, and thus, short distribution time by eliminating block duplication. Since block duplication happens on multiple delivery paths, to optimize the whole system, our job is to find a set of C nodes, from which duplicated blocks slow down throughput to other nodes the most. One such algorithm (Algorithm 1), which figures all possible path from the source to a downstream node, is given for reference. Algorithm 1's running time, however, is prohibitive due to the exponential number of paths.

Algorithm 1. Multi-path Coder Placement Algorithm

1. For each node i, $R(i)$=0.
2. For each node j, figure all paths from the source to j.
    a. Get the set A of all node i, i is on multiple paths to j.
    b. For each $i \in A R(i)= R(i)+r_i$, where r_i is computed using (5)
3. Assign a set of C nodes with highest $R()$ value as coders.

We propose a heuristic approach instead by quickly looking for nodes which lie on more paths with wider bandwidth. By ensuring that chosen coders lie on paths to many nodes with wider bandwidth we can avoid duplicated blocks from transferring to those nodes as shown in section 5. Our idea is to use *betweenness centrality* (we will explain shortly) to evaluate how important a node to other nodes if that node becomes a coder. The higher a node's betweenness centrality value, the more paths or wider paths between other nodes it stands on, and the more appraisable for it to become a coder. Algorithm 2 summarizes our coder placement strategy.

Note that the source is always chosen as a coder because it stands on paths to every peer. The algorithm utilizes either *betweenness centrality* [5], or its variant, *flow centrality* [6] to figure the importance of a given network node which we explain in the following section.

6.2. Betweenness and Flow Centrality

Originated in social networks studies, centrality is an essential tool for graph analysis which measures the importance of a node within the graph. Depending on the kind of measures, there are various centrality indexes. In this paper, however, we are interested in betweenness centrality, and one of its variants: flow centrality.Betweenness centrality expresses the degree a node locates on the paths between other nodes, and flow centrality expresses the total bandwidth

Algorithm 2. Proposed Coder Placement Algorithm

1. Compute centrality of every peer using either of following algorithms
    a. Brandes's betweenness centrality algorithm [27]
    b. Flow centrality algorithm based on Edmond-Karp's max flow (Algorithm 3)
2. Appoint *C* peers with highest centrality values as coders.

of flows going through a node. That is, betweenness centrality and flow centrality are indicators of nodes fromwhich duplication occurs. Given that network coding, by generating fresh encoded data to send onto each path, can effectively eliminate such duplication to increase effective bandwidth, placing coders in nodes with high centrality values will speed up content delivery.

Betweenness centrality [5] measures the degree that a node stands on the shortest paths between other nodes, which has been applied in different contexts such as routing and cache placement [24, 25] to place a network function in a set of nodes. Since we are interested in distributing data from the source to all peers, all shortest paths under consideration are from the source to other nodes. Denote σ_k as the number of the shortest paths from the source to node *k* and $\sigma_k(i)$ as the number of the shortest paths from the source to node *k* which go through node *i*. Betweenness centrality of node *i* is measured by

$$C_B(i) = \sum_{k \neq i \in V} \frac{\sigma_k(i)}{\sigma_k}. \tag{9}$$

Nodes with high betweenness centrality locate on more shortest paths to other downstream nodes, and thus, likely generate more duplication as we observe in section 5, (8). If those nodes encode, more duplication can be avoided to speed up content distribution. The limitation of betweenness centrality is that it only considers the number of shortest paths to downstream nodes which does not always reflect correctly the importance of a node in eliminating duplication.

Flow centrality [6], on the other hand, measures the portions of *max-flows* between all pairs of other nodes which go through a given intermediate node. Like betweenness centrality, we are interested in flows from the source to all peers. In our study, we compute flow centrality of node *i* as the total amount of flows from the source to all other node *k* which pass through node *i*:

$$C_F(i) = \sum_{k \neq i \in V} f(S, i, k) \tag{10}$$

where$S$ is the source, and $f(S, i, k)$ is the portion of *max-flow* from the source to node *k* that passes through node *i*. As we are interested in both the value of the flow and the shortness of thepath, the *max-flow* mentioned above is the one consists of paths with shortest lengths among all paths from *S* to node *k*.[2]

[2] This kind of flow centrality is slightly different from what has been originally proposed in [6] where $f(S, i, k)$ means the portion of *max-flow* from node *S* to node *k* that must pass through node *i* in order that

High flow centrality nodes have larger aggregate flows to downstream nodes. They, therefore, have high probability to stand on more paths to a given downstream node, and in addition, the flows on the paths are likely larger. Since larger duplication results from larger flows as stated in section 5, we expect high flow centrality nodes to generate more duplication to downstream nodes, which justifies the need to place encoders there.

We compute flow centrality using Algorithm 3 which is basically Edmond-Karp's max-flow algorithm [26] with the addition of line 31 where the augmenting flow value (*flow*) is updated to the flow centrality C_F[i] of each node *i* on that augmenting path.

Algorithm 3. Flow Centrality Algorithm based on Edmond-Karp's Max Flow

```
1   C_F[i] = 0, i∈V;
2   u[i,j] = capacity of link (i,j), (i,j) ∈ E;
3   s = 0;
4   for t∈V do
5       f[i,j] = 0, (i,j) ∈ E;
6       do
7           Q ← empty queue;
8           prev[0] = -1;
9           color[i] = not visited, i∈V;
10          enqueue s → Q;
11          while Q not emptydo
12              dequeue i ← Q;
13              color[i] = visited;
14              foreach neighbor j of i do
15                  if(color[j]==not visitedand (u[i,j]-f[i,j])>0)
16                      color[j] = queued;
17                      prev[j] = i;
18                      enqueue j → Q;
19                  end
20              end
21          end
22          if(color[t]==visited)
23              for(j=t; prev[j]>=0;j=prev[j]) do
24                  i = prev[j];
25                  flow = min{u[i,j]-f[i,j]};
26              end
27              for(j=t; prev[j]>=0;j=prev[j]) do
28                  i = prev[j];
29                      f[i,j] = f[i,j]+ flow;
30                      f[j,i] = f[j,i]- flow;
31                      C_F[i] = C_F[i]+ flow;
32                  end
33          end
34      while(color[t]==visited)
35  end
```

node *k* achieves its *max-flow*. In other words, if node *i* is removed, the *max-flow* from node *S* to node *k* decreases by $f(S, i, k)$.

Betweenness centrality of all nodes can be computed with Brandes's$O(VE)$ algorithm [27]. Since we are only interested in shortest paths from the source, the complexity is $O(E)$. Flow centrality, however, is more expensive to compute. In Algorithm 3, the complexity to visit each source-sink pair is $O(VE^2)$. Therefore, to find out flow centrality of all V nodes, the algorithm takes $O(V^2E^2)$ time.

6.3. Discussion on Algorithms for Dynamic Networks

One straightforward way to extend our proposed placement to the dynamic case, where peers keep joining and leaving the system, is to redeploy encoders periodically.

A more elegant extension is to use a distributed approximation algorithm, such as [28, 29], to compute the centrality value at each peer. Using the method in [28], for example, an alternativecentrality value called *second order centrality* is computed by letting each node keep track of the time elapsed between visits by a random walk. High centrality nodes see more frequent visits compared with other nodes. The approach in [29], on the other hand, figures the centrality level of a node by means of a localized spectral analysis on a small-size neighborhood of the node.

In this paper we focus more on finding the feature of network coding which helps accelerate content distribution rather than applying it in a real application. We thus leave the extension to dynamic networks for future work.

7. EXPERIMENTAL EVALUATION

We implemented a C++ simulator of the P2P content distribution system described in section 3 and run simulations over generated topologies distributing a file from the source to all participating peers. 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 simulations are round-based. Each peer chooses blocks to download according to its available bandwidth, rarest block first selection, and the incentive scheme in the beginning of each round. The chosen 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 implemented 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 [30]. Each time there is a request for download, the source tries to serve with a random block which has never been sent into the network.

For each overlay topology, with the same simulation parameters, we run simulations 100 times[3] distributing a 200-block file from the source and collect the average finish time of all peers in each of the 3 scenarios: *no coding*, *network coding*, and selective coding when we use *betweenness centrality* and *flow centrality* to place coders.

7.1. BitTorrent Block Duplication Validation

We first set up an experiment to verify block duplication phenomenon in BitTorrent and to show network coding effectiveness in eliminating that duplication. The topology given inFigure 3 is used for simulations to investigate block duplication on the two paths: path 1 is from node *i* to cluster 1 then cluster 2, and path 2 starts at node *i* to cluster 2 then cluster 1. Each cluster has 1000 peers which are arranged in a regular random topology. All other overlay links within a cluster have the same capacity of 1 block per round. We change the bandwidth *p* from node *i* to each cluster and the bottleneck bandwidth *b* between clusters. The source bandwidth is twice the bandwidth from node *i* to each cluster.

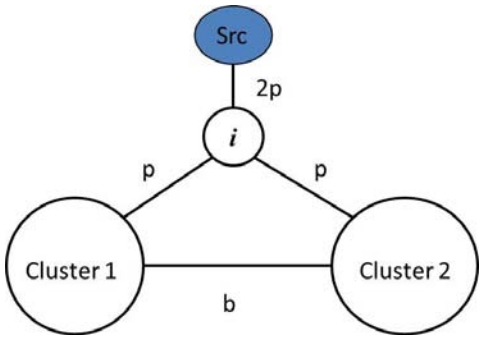


Figure 3.A topology with 2 paths from node *i*

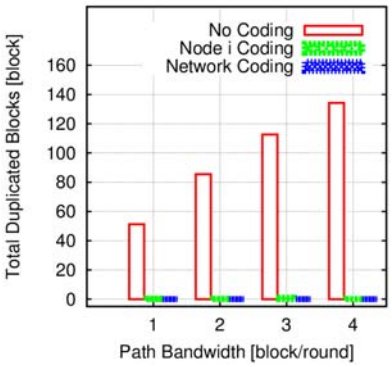
Every block sent from node *i*is reported by the simulator. Using the same simulation parameters, we measure the number of duplicated blocks in 3 cases:

- *no coding*: if a block is transferred *c* times from node *i*, it means there are (*c-1*) duplicated blocks,
- *node i coding* (an encoder is placed at node *i*): if a (coded) block sent by node *i* is linearly dependent on blocks which it has already sent, that block is considered a duplicated one, and
- *network coding* (encoders are placed at all nodes): duplicated blocks are measured as in the case node *i* codes.

As we expected, block duplication only happens in *no coding* case (Figure 4). There are always a large number of duplicated blocks transmitting through the two paths from node *i* while in the other two cases, that number is almost zero. One observation is that the number of duplicated blocks is higher with wider bandwidth from node *i* (Figure 4(a)). Block duplication is also more severe when bottleneck between the two clusters has narrow bandwidth (Figure 4(b)).

[3] Although simulation parameters are the same for 100 runs, due to randomness in downloader selection by the sending peers and block selection by the receiving peers, the result changes with each run.

(a) Number of duplicated blocks (p varies)



(b) Number of duplicated blocks (b varies)

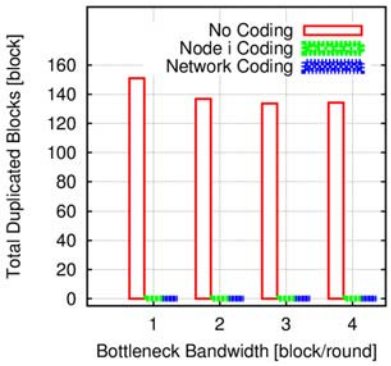


Figure 4. Number of duplicated blocks

7.2. P2P Network Topology

We use Watts and Strogatz*small-world network* model [31] to generate topologies for simulations. The reason is twofold. First, several real-life networks, including P2P overlays, have been reported to exhibit properties of small-world networks [32, 33]. Second, small-world model has a parameter to tune the severity of bottleneck links as explained below.

Nodes in a small-world network are, at the beginning, organized in a ring lattice, each node connects to a predetermined number of nearby nodes, i.e. degree d. The links between nodes are then rewired with some probability p_{rw}, i.e. one endpoint of the (randomly chosen) link is rewired to a new random node, to create shortcuts.

Shortcuts connect different parts of the networks where quite different collections of data blocks exist. Those shortcuts, to a certain extent, equivalent to bottleneck links because the total flow of nearby regular links is bigger than the capacity of the single shortcut link. By adjusting the rewiring probability, we can tune the severity of the bottlenecks. Lower rewiring probabilities mean fewer shortcuts, which in turn mean the bottlenecks are more severe because there are fewer shortcut links for transferring data between different parts of the network. In simulations, we set degree d=6 and change the rewiring probability. All overlay links have capacity of 1 block per round.

7.3. Coder Placement Evaluation Results

Figure 5 compares the performance of the proposed placement algorithm with full network coding. Assigning $C=1000$ peers with highest flow centrality as coders, we can achieve comparable performance to network coding. The average finish time is consistently under 5% longer than network coding. Using betweenness centrality to appoint the same number of coders with highest betweenness centrality values, the finish time is nearly 15% longer than network coding when the topologies have low rewiring probability ($p_{rw}=0.02$ and $p_{rw}=0.1$) and 5% longer than network coding with higher rewiring probabilities. The reason for flow centrality's good performance is that, by taking max-flow into account, it reflects more accurately the characteristics of the topology than betweenness centrality. The complexity of flow centrality is, however, higher than betweenness centrality. For comparison, degree-based placement, i.e. encoders are placed at high-degree nodes first, has poor performance. We notice that the

performance gain due to coding is negligible (even with full network coding) in regular topologies ($p_{rw}=0$) and highly random topologies ($p_{rw}>0.5$) because there are virtually no bottlenecks in such networks.

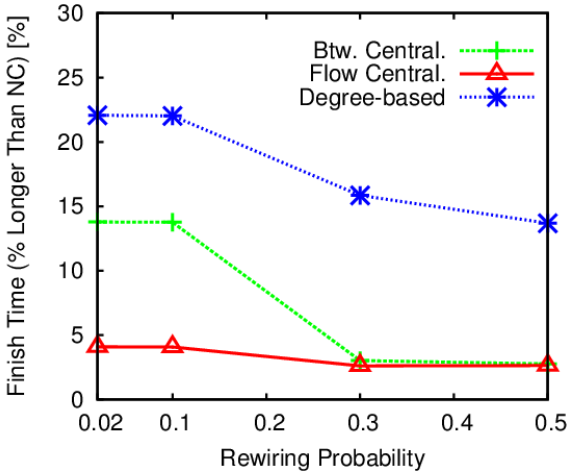


Figure 5.Finish time compared with full network coding.

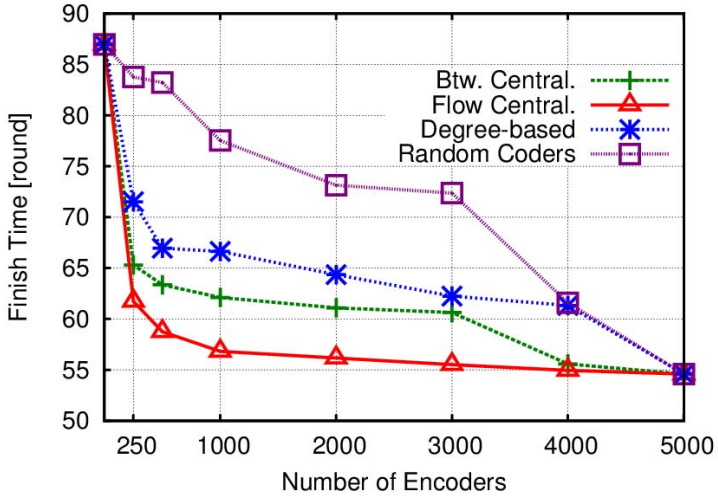


Figure 6.Finish time with different numbers of encoders.

We change parameter C to appoint different numbers of peers with highest centrality values as coders in a topology with 5000 peers and rewiring probability $p_{rw}=0.02$. Figure 6 gives the average finish time when no coders (*no coding*) to 5000 coders (full *network coding*)[4] are deployed. We also include the finish time of random placement, which assigns the encoders at random, and degree-based placement for reference. Betweenness centrality and flow centrality placements always achieve improved performance compared with degree-based and random

[4] Using a given placement method, increasing the number of encoders to 5000 means that all peers in the network encode, i.e. network coding. Therefore, in Figure 6, finish time is the same for all 4 placement methods when the number of encoders is 5000.

placements. Of the two former methods, flow centrality reaches finish time almost equal to network coding with only 1000 encoders and the performance is nearly constant when we increase the number of coders from 1000 to 5000. The result confirms that centrality is good tool to locate a small subset of important nodes for coder placement.

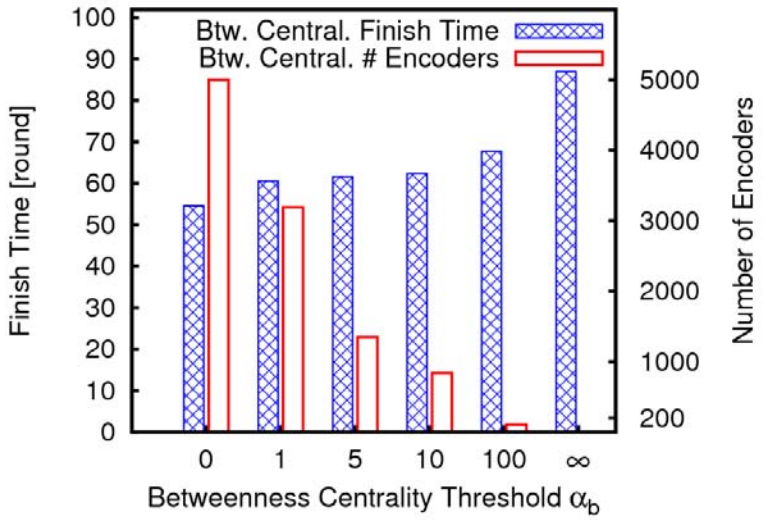


Figure 7. Finish time and number of assigned encoders with different betweenness centrality thresholds.

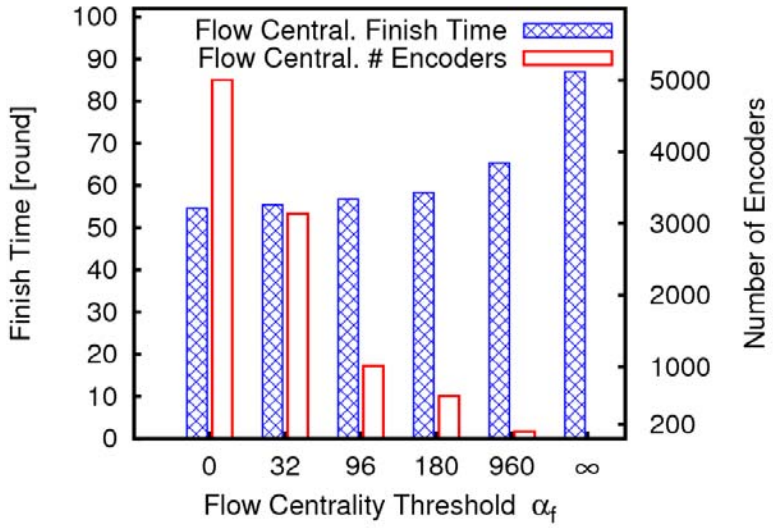


Figure 8. Finish time and number of assigned encoders with different flow centrality thresholds.

We furthermore evaluate the placement method in terms of total number of required encoders and average finish time, varying the centrality threshold. With each threshold, all nodes with centrality value higher than or equal to the threshold are chosen as encoders. The results are given in Figure 7 and Figure 8 in which threshold α=0 means all nodes are chosen as encoders, i.e. *network coding*, and threshold infinity means no nodes code, i.e. *no coding*. In the given topology

(d=6, p_{rw}=0.02), choosing all nodes with betweennesscentrality higher or equal to α_b=10, which means the total fractions of shortest paths to downstream peers the selected encoders locate on are equal to or greater than 10, gives an average finish time 14% longer than network coding, however, with a saving of nearly 85% in the number of required encoders compared to network coding (Figure 7). The performance is better using flow centrality. When threshold is set to α_f=180, i.e. each chosen encoder stands on a total flow of 180 to its downstream peers, with nearly 90% saving in encoders, flow centrality placement achieves short finish time, just 7% longer than network coding finish time (Figure 8). With lower thresholds, i.e. more nodes are chosen as encoders, there is not much improvement in finish time, even though the number of encoders is much higher. The result means that while encoders at high centrality nodes can effectively improve performance, those at low centrality nodes are redundant and can be removed to save resources.

8. CONCLUSIONS

In this paper, we have proposed a novel practical algorithm to place coders within a P2P network to shorten distribution time. Unlike previous approaches which justify coding over the whole network topology, our algorithm, in evaluating the centrality value of each node within the topology, looks inside the network to find particular places which require network coding. The idea works on the basis that coding at an upstream peer can improve data transmission on multiple paths to downstream peers located behind bottlenecks. Data redundancy generated by a coder eliminates duplicated downloads, which otherwise unnecessarily consume bandwidth resources and slow down content delivery over the bottleneck.

Our content distribution has comparable performance to network coding, yet with far fewer coders, using much less computational resources compared to network coding which excessively codes everywhere.

We have confirmed that betweenness centrality and flow centrality are good indicators to locate important nodes, which lie on multiple paths to other nodes, in a network and deploy them as coders. Flow centrality, by taking flow information into account, achieves a performance closely matched that of full network coding. Betweenness centrality placement, although less effective, has the advantage of much lower complexity which is suitable when encoder placement is frequently computed.

For our future work, we plan to extend our algorithm to the dynamic case when peers join and leave the system. As we have discussed in section 6.3, that can be done by, first, devising a distributed algorithm to allow each peer to figure an approximate centrality value by itself which is then used to decide if the peer should encode or not. Second, we would need to determine the centrality threshold which both achieves short finish time and reduces the number of encoders. Peers with centrality values higher than the threshold will become network coders to improve the system performance.

ACKNOWLEDGEMENTS

This work was supported by JSPS KAKENHI Grant Number (24500098).

REFERENCES

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

[2] S. Li, R. Yeung, and N. Cai, "Linear network coding," IEEE Transactions on Information Theory, 2003.

[3] 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.

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

[5] L.C. Freeman, "A set of measures of centrality based on betweenness," 1977, Sociometry, vol. 40, No. 1, 35-41.

[6] L.C. Freeman, S.P. Borgatti, D.R. White, "Centrality in valued graphs: a measure of betweenness based on network flow," Social Networks 13, 141–154, 1991.

[7] B. Cohen, "Incentives Build Robustness in BitTorrent," P2P Economics Workshop, 2003.

[8] P. Chou, Y. Wu, and K. Jain, "Practical network coding," in Proc. Annual Allerton Conference on Communication, Control and Computing, 2003.

[9] T. Locher, S. Schmid, and R. Wattenhofer, "Rescuing Tit-for-Tat with Source Coding," IEEE P2P, Sep. 2007.

[10] D. Nguyen, H. Nakazato, "Peer-to-Peer Content Distribution in Clustered Topologies with Source Coding," IEEE GLOBECOM 2011, Houston, USA, Dec. 2011.

[11] M. Kim, M. Medard, V. Aggarwal, U.-M. O'Reilly, W. Kim, C. W. Ahn, and M. Effros, "Evolutionary Approaches to Minimizing Network Coding Resources," Proc. IEEE INFOCOM 2007, May 2007.

[12] K. Bhattad, N. Ratnakar, R. Koetter, and K. R. Narayanan, "Minimal network coding for multicast," Proc. IEEE ISIT 2005, Sep. 2005.

[13] Desmond S. Lun, NiranjanRatnakar, Ralf Koetter, Muriel M´edard, Ebad Ahmed, and Hyunjoo Lee, "Achieving Minimum-Cost Multicast: A Decentralized Approach Based on Network Coding," IEEE INFOCOM 2005, Mar. 2005.

[14] M. Martalo, M. Mohorovicich, G. Ferrari, C. Fragouli, "Network-coded multihop multicast: Topology and encoding complexity," IEEE International Conference on Communications (ICC), June 2012.

[15] T. Small, B. Li, "Topology Affects the Efficiency of Network Coding in Peer-to-Peer Networks," ICC 2008, May 2008.

[16] D. Niu, B. Li, "Topological Properties Affect the Power of Network Coding in Decentralized Broadcast," IEEE INFOCOM 2010, Mar. 2010.

[17] S. Crisostomo, J. Barros, C. Bettstetter, "Network Coding with Shortcuts," IEEE ICCS 2008.

[18] S. Maheshwar, Zongpeng Li, Baochun Li, "Bounding the Coding Advantage of Combination Network Coding in Undirected Networks," IEEE Transactions on Information Theory, vol.58, no.2, pp.570–584, Feb. 2012.

[19] N. Cleju, N. Thomos, P. Frossard, "Network Coding Node Placement for Delay Minimization in Streaming Overlays," ICC 2010, May 2010.

[20] P. Maymounkov, N.J.A. Harvey, D.S. Lun, "Methods for efficient network coding," in Proc. 44th Annu. Allerton Conf. Communication, Control, and Computing, Monticello, IL, September 2006.

[21] M.L. Champel, K. Huguenin, A.M. Kermarrec, N. Le Scouarnec, "LT Network Codes," Distributed Computing Systems (ICDCS), June 2010.

[22] D. Silva, Weifei Zeng, F.R. Kschischang, "Sparse network coding with overlapping classes,"NetCod '09, June 2009.

[23] M. Langberg, A. Sprintson, and J. Bruck, "The encoding complexity of network coding," IEEE Trans. on Information Theory, pp. 2386–2397, June 2006.

[24] Ali Tizghadam, and Alberto Leon-Garcia, "AORTA: Autonomic Network Control and Management System," 1st IEEE Workshop on Automated Network Management, Apr. 2008.

[25] Diliang He, Wei K. Chai and George Pavlou, "Leveraging In-network Caching for Efficient Content Delivery in Content-centric Network," London Communication Symposium, Sep. 2011.

[26] 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.

[27] U. Brandes, "A Faster Algorithm for Betweenness Centrality,"Journal of Mathematical Sociology, 2001.

[28] A.-M. Kermarrec, E.L. Merrer, B. Sericola, and G. Trédan, "Second order centrality: Distributed assessment of nodes criticity in complex networks," Computer Communications 34 (2011) 619–628.

[29] K. Wehmuth and A. Ziviani, "Distributed Location of the Critical Nodes to Network Robustness based on Spectral Analysis," the 7th Network Operations and Management Symposium (LANOMS), Oct. 2011.

[30] BitTornado. About Super-seed Mode [Online]. Available: http://www.bittornado.com/docs/superseed.txt

[31] Watts, Duncan J., Strogatz, Steven H. (June 1998) "Collective dynamics of 'small-world' networks," Nature 393 (6684): 440–442.

[32] L. A. Adamic, "The Small World Web," in proceedings of ECDL '99, Springer-Verlag, London, UK, 443–452.

[33] N Leibowitz, M. Ripeanu, A. Wierzbicki, "Deconstructing the kazaa network," in proceedings of WIAPP 2003.

Authors

Dinh Nguyen received his Bachelor of Electronics and Telecomm. degree from Hanoi University of Technology, Vietnam, in 1999. From 1999 he was with NetNam ISP Corporation. He received his MSc and currently is a Ph.D. candidate at Graduate School of Global Information and Telecommunications Studies, Waseda University, Tokyo, Japan. His research interests include peer-to-peer systems, network coding, and content distribution systems.

Hidenori Nakazato received his B. Engineering degree in electronics and telecommunications from Waseda University in 1982 and his MS and Ph.D. degrees in computer science from University of Illinois in 1989 and 1993, respectively. He was with Oki Electric from 1982 to 2000. Since 2000, he has been a faculty member of Graduate School of Global Information and Telecommunications Studies, Waseda University. He served as the editor of IEICE Transactions on Communications from 1999 to 2002 and other positions in the executive committee of IEICE Communication Society from 1997 to 2004, and from 2008 to now. He also served as an executive committee member of IEEE Region 10 and is serving as a member of several IEEE Member and Geographic Activity committees. His research interests include performance issues in distributed systems and networks. He is a member of ACM, IEEE, and IPSJ.