Topology Affects the Efficiency of Network Coding in Peer-to-Peer Networks

Tara Small, Baochun Li, and Ben Liang
Department of Electrical and Computer Engineering
University of Toronto
{tsmall, bli}@eecg.toronto.edu, liang@comm.toronto.edu

***Abstract*—With network coding, intermediate nodes between the source and the receivers of an end-to-end communication session are not only capable of relaying and replicating data messages, but also of *coding* incoming messages to produce coded outgoing ones. It has been the traditional wisdom in information theory that network coding improves the capacity of multicast sessions in directed networks. Recent studies have also shown that network coding is beneficial for content distribution in peer-to-peer networks, since it resolves the "last block" problem, and eliminates content reconciliation. In this paper, we show that such benefits of network coding does not come without costs and trade-offs. In particular, we refute the previous claim that peers receive linearly independent coded blocks with very high probabilities. Using example scenarios and extensive simulations, we show that it is very likely for peers to receive linearly dependent non-innovative blocks, thus decreasing their efficiency as these redundant blocks consume bandwidth. We observe that such redundancy of network coding is critically dependent on the *randomness* and *sparsity* of the P2P topology. We conclude with suggestions on topologies of certain characteristics that are preferred over others, in order to minimize the network coding redundancy, the time to distribute data, and the server cost.**

I. Introduction

Network coding has recently been proposed in information theory [1], and has since received extensive research attention. In contrast to channel coding, the essence of network coding is a paradigm shift to allow coding at intermediate nodes between the source and the receivers in multicast communication sessions, assuming that communication links are free of errors. The fundamental assumption of error-free links is realistic in peer-to-peer networks (due to retransmission-based transport layer protocols). It has relieved the research on network coding from addressing the challenges of *interference*, which often lead to the most difficult problems in the field of network information theory.

The fundamental insight of network coding is that information to be transmitted from the source in a session can be *inferred*, or decoded, by the intended receivers, and does not have to be transmitted verbatim. It has also focused on the *coding* capabilities of intermediate nodes, in addition to forwarding and replicating incoming messages. With the ability to code at relay nodes in a session, we may forward, replicate and code information flows, as opposed to traditional commodity flows, where only forwarding is allowed. In recent research literature on network coding that is rapidly expanding, it is a well known result that network coding — by using linear codes only — may achieve better network throughput in some of the network topologies.

In recent years, peer-to-peer (P2P) architectures have also been shown to offer high performance, better scalability, as well as superb resilience to peer failures and departures. It has been increasingly natural to design Internet applications using the peer-to-peer architecture, the most important application being bulk content distribution (*e.g.*, BitTorrent [2]). As end hosts at the edge of the Internet possess abundant computational resources with current-generation processors, it is natural to consider taking advantage of the power of network coding in peer-to-peer applications, by allowing end hosts to not only forward and replicate, but to code as well.

Recent work on network coding has gradually shifted its focus from a more theoretical point of view to a more practical setting. The following critical question naturally emerges: Given a content distribution session in peer-to-peer networks, is network coding indeed able to offer a better throughput — best measured in the time to complete downloading at the peers, as compared to using a protocol without coding (such as BitTorrent)? Recent studies (most notably the Avalanche project [3]) have shown that network coding is beneficial for content distribution in peer-to-peer networks, since it resolves the "last block" problem, and eliminates content reconciliation. These observations are derived from the insight that *all coded blocks are treated equally*, without the need of finding the rarest blocks that can be downloaded first. The conclusion seems to be certain: network coding leads to shorter downloading times due to these benefits.

In this paper, we show that such benefits of network coding do not come without costs and trade-offs. In particular, we refute the claim from previous work that peers receive linearly independent coded blocks (usually referred to as *innovative* blocks) with very high probabilities. Theoretically, this claim is correct, provided that all coding is performed at the source peer, or at intermediate peers after complete decoding to recover the original blocks. However, we show that, when peers code outgoing blocks before they fully decode and recover original blocks in realistic P2P topologies, it is very likely for peers to receive linearly dependent *non-innovative* blocks, thus (sometimes significantly) decreasing their efficiency as these redundant blocks consume bandwidth.

This work was supported in part by Bell Canada through its Bell University Laboratories R&D program.

With analysis of two example networks and extensive simulations with a common small-world topology, we study how the redundancy introduced by network coding is affected by the *sparsity* of the topology, quantitatively represented by the average number of neighbors that peers have, as well as the *randomness* of the topology, quantitatively characterized by the *rewiring probability* of a small-world topology. We also study other vital system performance metrics, including the time delay in data distribution and the bandwidth cost to the P2P server. Finally, we seek to quantitatively identify the types of topologies to optimize system performance should network coding be applied.

The remainder of this paper is organized as follows. Sec. II reviews related work. Using examples and intuitive explanations, Sec. III illustrates why network coding leads to linearly dependent blocks. The effects of randomness and network sparsity are analyzed in Section IV with empirical studies. In Sec. V, we further provide insights on preferred topologies that optimizes system performance when network coding is used.

II. Related Work

The pioneering work by Ahlswede *et al.* [1] and Koetter *et al.* [4] proves that, in a directed network with network coding support, a multicast rate is feasible if and only if it is feasible for a unicast from the sender to each receiver. Li *et al.* [5] has further proved that linear coding usually suffices in achieving the maximum rate. These results are significant in the sense that, with network coding, the cut-set capacity bounds of unicast flows from the source to each of the receivers can be achieved in a multicast session. In other words, network coding helps to alleviate competition among flows at the bottleneck, thus improving session throughput in general.

To practically implement the paradigm of network coding, one needs to address the challenges of computing *coding coefficients* to be used by each of the intermediate nodes in the session, so that the coded messages at the receivers are guaranteed to be decoded. This process is usually referred to as *code assignment*. Although deterministic code assignment algorithms have been proposed and shown to be polynomial time algorithms (*e.g.*, [6]), they require extensive exchanges of control messages, which may not be feasible in dynamic peer-to-peer networks. As an alternative, Ho *et al.* [7] has been the first to propose the concept of *randomized network coding*. With randomized network coding, an intermediate node transmits on each outgoing link a linear combination of incoming messages, specified by independently and randomly chosen *code coefficients* over some finite field. Ho *et al.* show that by allowing peers to locally encode data using coefficients from sufficiently large Galois fields, received coded blocks at downstream peers are decodable with a very high probability, on the order of the inverse of the size of the finite field. For example, if the field size is 2^8, the lower bound of this probability is ≥ 0.989.

Since the landmark paper on randomized network coding by Ho *et al.*, there has been a gradual shift in research focus in the area of network coding, from theoretical studies on achievable flow rates and code assignment algorithms, to more practical studies on applying network coding in a practical setting. Such a shift of focus has been marked by the work by Wu *et al.* [8], in which the authors have concluded that randomized network coding can be designed to be robust to random packet loss, delay, as well as any changes in network topology and capacity. It was shown that sessions with randomized network coding can achieve close to the theoretically optimal performance.

The *Avalanche* project by Microsoft Research [3], [9] has further proposed that randomized network coding can be used for bulk content distribution, in competition with *BitTorrent*, one of the most successful P2P content distribution protocols at the time of this writing. The work has made the claim that performance benefits provided by network coding in terms of throughput can be more than two to three times better than transmitting original blocks. In this sense, one may conclude that network coding can indeed be practically implemented, and does offer significant advantages as compared to BitTorrent. However, Wang *et al.* [10] has focused on the computational complexity of network coding, and has shown that coding complexity may lead to significant increases with respect to downloading times in content distribution sessions, especially as the number of blocks increases.

In this work, we show that the likelihood of receiving linearly dependent blocks is much higher, leading to a lower level of efficiency when network coding is used. Such *redundancy* introduced by network coding depends on the topology, but nevertheless leads to higher bandwidth consumption and, inevitably, longer downloading times.

III. The Problem of Linearly Dependent Blocks

In this paper, the peer-to-peer session that we intend to study is modeled as a collection of N peers, self-organized into a peer-to-peer *topology* with application-layer links. One of the peers is the *server*, or the *source* of content distribution. The original content on the source is segmented into n *original blocks* $[b_1, b_2, \ldots, b_n]$, each b_i has a fixed number of bytes k (referred to as the block size). All other peers intend to complete their downloads of the original content within the constraints of the peer-to-peer topology. We make the realistic assumption that a fraction p_s of the peers serves as direct downstream peers of the server. The server sends coded blocks to these direct downstream peers with a period t_s, *i.e.*, the server upload bandwidth to each peer is k/t_s. Upon receiving new coded blocks, a peer produces new coded blocks for its downstream peers in the topology.

A. Randomized Network Coding

We briefly summarize the concept of randomized network coding [3], [7], [8], [11]. At the time of encoding for downstream peer p, a peer (including the source) independently and randomly chooses a set of coding coefficients $[c_1^p, c_2^p, \cdots, c_m^p](m \leq n)$ in the Galois field GF(2^8) for the downstream peer p. It then randomly chooses m blocks —

$[b_1^p, b_2^p, \ldots, b_m^p]$ — out of all the blocks it has received so far (all the *original* blocks if it is a source of the session), and produces one coded block x of k bytes:

$$x = \sum_{i=1}^{m} c_i^p \cdot b_i^p$$

The ratio m/n is referred to as *density* in this paper, as a low ratio leads to sparse decoding matrices. A coded block x is *self-contained*, in that the coding coefficients used to encode *original blocks* to x are embedded in the header of the coded block. Since the embedded coding coefficients are related to the original blocks, we need a total of n coefficients, leading to a header overhead of n bytes per coded block (if uncompressed). These n coding coefficients to be embedded can easily be computed by multiplying $[c_1^p, \cdots, c_m^p]$ with the $m \times n$ matrix of coding coefficients embedded in the incoming blocks $[b_1^p, b_2^p, \ldots, b_m^p]$.

As the session proceeds, a peer accumulates coded blocks from its upstream peers into its local buffer, and encodes new coded blocks to serve its downstream peers. When serving multiple downstream peers, it needs to independently and randomly choose a new set of coding coefficients for each of its downstream peers. In order to reduce the delay introduced by waiting for new coded blocks, the peer produces a new coded block upon receiving $a \cdot n$ coded blocks ($0 < a \leq 1$), in which the tunable parameter a is referred to as *aggressiveness* in this paper. A smaller a leads to a shorter waiting time and, potentially, shorter delay in the process of content distribution. In other words, the peer is more "aggressive."

As soon as a peer has received a total of n coded blocks $\mathbf{x} = [x_1, x_2, \ldots, x_n]$, it starts the decoding process. To decode, it first forms a $n \times n$ matrix $\mathbf{A}$, using the n coding coefficients embedded in each of the n coded blocks it has received. Each row in $\mathbf{A}$ corresponds to n coded coefficients of one coded block. If vectors in all the rows are *linearly independent*, it may then recover the original blocks $\mathbf{b} = [b_1, b_2, \ldots, b_n]$ by

$$b = \mathbf{A}^{-1}\mathbf{x}^T$$

In this equation, it first needs to compute the inverse of $\mathbf{A}$, using Gaussian elimination. It then needs to multiply $\mathbf{A}^{-1}$ and $\mathbf{x}$, which takes $n^2 \cdot k$ multiplications of two bytes in GF(256). The inversion of $\mathbf{A}$ is only possible when its rows are linearly independent, *i.e.,* $\mathbf{A}$ is full rank.

B. The Problem of Linearly Dependent Blocks

We concur with the claims in previous work that peers would receive linearly independent (*innovative*) blocks with a very high probability, provided that all coding is performed at the source peer, or at intermediate peers after they have completed the decoding process and recovered all original blocks (so that they become source peers). In other words, we assume that peers wait for n coded blocks to arrive before producing coded blocks, *i.e.,* the aggressiveness is 1. However, this assumption is made against the intuitive benefit of network coding — the ability to code as peers receive. We now show two examples — with smaller and larger topologies — that peers may easily receive linearly dependent (*non-innovative*) blocks when aggressiveness $a < 1$. When this phenomenon occurs, we claim that the *efficiency* of network coding is lower, as it introduces *redundancy*. Naturally, such redundancy is undesirable as it consumes bandwidth.

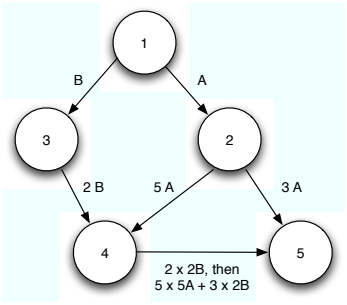
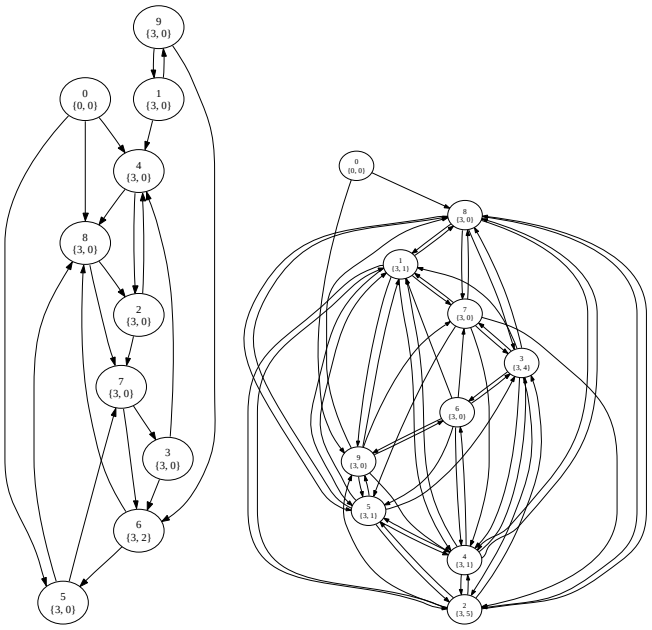


Fig. 1. Network coding leads to linearly dependent blocks: the first example with a small topology.

Our first example involves a network topology that resembles the structure in Fig. 1. We show that, when peers become more aggressive and code as they receive, network coding leads to linearly dependent blocks. Let us assume that peers forward data to their downstream neighbors with aggressiveness $1/n$; that is, data is forwarded immediately upon reception of an innovative block. Further, we let peer 1 send two linearly independent coded blocks A and B to its downstream peers 2 and 3, respectively.

Peer 2 sends coded forms of A using randomly generated coefficients, say $3A$ and $5A$, to its peers 4 and 5. Similarly, peer 3 encodes B and sends $2B$ to its downstream peer 4. Due to the stochastic nature of arrivals, peer 4 may receive either coded block before the other. If peer 4 receives $5A$ first, then it sends $k_1 \cdot 5A$ to peer 5 (for some random coefficient k_1). Since peer 5 may already have received $3A$ from peer 2, $k_1 \cdot 5A$ is linearly dependent (redundant). On the other hand if peer 4 receives $2B$ first, it then sends $k_2 \cdot 2B$ (for some random k_2) to peer 5. Though peer 5 has already received two linearly independent blocks containing A and B, when the number of blocks $n > 2$ (which implies that peer 5 has not received all n coded blocks yet), peer 4 would further send $k_3 \cdot 5A + k_4 \cdot 2B$ to peer 5 upon receiving $5A$, since it is to produce one new coded block to its downstream peers upon receiving a new innovative block. The third block from peer 4 is obviously linearly dependent and redundant on peer 5. It appears that the problem of linearly dependent blocks comes from the fact that both peer 2 and 4 are direct upstream peers of peer 5, but peer 2 serves peer 4 at the same time. Such a "shortcut" from peer 2 to 5 appears to be the cause of our problem in this particular example.

The redundancy of using bandwidth with network coding, shown in the example above on peer 5, is categorically different from the problem of *rank deficiency* of the decoding matrix $\mathbf{A}$, when the min-cut between the sender and the receiver is not large enough. In Fig. 1, peer 3 only has one link from peer 1, so it is not able to fully decode within one time slot (the time to transmit a coded block), as its decoding matrix



(a) Some common neighbors in sparse topologies (b) Many common neighbors in dense topologies

Fig. 2. Network coding leads to linearly dependent blocks: the second example with random topologies.

$\mathbf{A}$ is rank deficient. Peer 5, however, suffers from a different problem: over a period of one or two time slots (depending on the stochastic progress of block propagation), it has received *more* blocks than it needs.

The second example involves a larger topology, shown in Fig. 2, which is an example of a random topology that is often used in P2P networks today. In this figure, an identifier is indicated for each peer, as well as the pair of the numbers of $\{$independent, dependent$\}$ blocks received by each peer after every peer receives $n = 3$ coded blocks to successfully decode the desired data. This example illustrates the results obtained from our simulations.

In Fig. 2(a), peer 7 receives two independent blocks in succession from peer 8 and forwards them to its downstream neighbors, peers 3 and 6. Peer 3 is an upstream neighbor of peer 6 and a downstream neighbor of peer 7, so the information sent from peer 3 to peer 6 is almost always redundant. In more densely-connected topologies, peers are even more likely to have direct downstream neighbors in common. In Fig. 2(b), peer 7 produces coded blocks to peer 3, its downstream neighbor; however, the existing blocks on peer 7 may be received from peer 8, making it likely for the freshly produced blocks from peer 7 to be linearly dependent on the coded blocks from peer 8, who has peer 3 as one of its downstream neighbors as well. In addition to peer 3, we show from simulation results that peer $2, 4$ and 5 have also received various numbers of non-innovative blocks as well.

To summarize our discoveries so far, we have observed from both smaller and larger topologies that, if we allow peers to produce new coded blocks as they receive from the upstream peers, it is very likely that network coding leads to linearly dependent blocks that bring redundant traffic to peer-to-peer topologies, consuming bandwidth. The fundamental insights in our observations are the following.

- ▷ Redundancy in network coding may be introduced by the stochastic nature of overlay link delays, such that peers receive linearly dependent blocks from some of their upstream peers first, before they receive innovative blocks from others.
- ▷ Redundancy in network coding is heavily dependent on the topology itself. From our examples, it appears that topologies with higher densities are more likely to induce redundancy with network coding. Further, we have also shown that neighborhoods that contain many "shortcuts" — where direct upstream peers serve one another as well — may be the culprit that causes problems of redundancy, simply because they exchange coded blocks faster than the rate of sending innovative blocks into the neighborhood!

Is a sparse topology any better? On second thought, this may not be the case. Though dense topologies may lead to additional redundancy, they may also be helpful to rapidly disseminate innovative blocks across the topology, simply because the distance of travel (in terms of the link delays) is much shorter. On the other hand, if topologies are too sparse, coded blocks may not be able to travel effectively through the topology, and ineffective travel may lead to redundancy in small clusters of peers that are unlikely to receive new innovative blocks. What constitutes a "good" topology that minimizes redundancy introduced by network coding? The rest of this paper is to explore, through extensive simulation studies, two important characteristics of topologies, *sparsity* and *randomness*.

IV. Topology Effects on the Efficiency of Network Coding

We developed an event-driven simulator in C++ to evaluate the performance of network coding in a broad spectrum of topologies with various levels of sparsity and randomness. We seek to understand the way that redundancy, block distribution times, and server costs vary. The *block redundancy* of a peer is the quotient of the number of coded blocks it receives and the number needed to successfully decode the segment. Redundancy of 1 implies that only linearly independent blocks are received at the peer. The *distribution time* is the time interval from initial forwarding of a block from the server to any of its downstream peers until all peers in the network have successfully received n independent coded blocks. The *server cost* is the number of blocks forwarded from the server to any of its downstream peers. Note that the server stops sending when all of its downstream peers (though not necessarily all peers in the network) have received n independent blocks. The remaining peers that have not yet received n independent blocks request new coded blocks from their upstream neighbors until n independent blocks have been received.

We vary the network topology randomness by adjusting the rewiring probability in small-world topologies. Small-world topologies are graphs that have been studied for many years

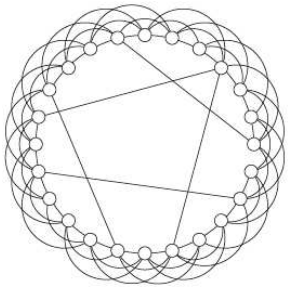


Fig. 3. Small-world topology with some rewired links.

to explain social networks [12][1]. As explained in the seminal paper by Watts and Strogatz [13], one way to construct a small-world graph is by organizing the peers into a ring, connecting each peer to d local neighbors, then rewiring each link to a random peer in the network with probability p, as shown in Fig. 3. Choosing $p = 0$ results in a completely regular graph where each peer has the same number of downstream and upstream neighbors. Peers in a $p = 1$ small-world graph chooses each of its d downstream links uniformly at random from all other peers in the network. The flexibility of adjusting the parameter p between these two extremes allows us to smoothly alter the randomness in the network.

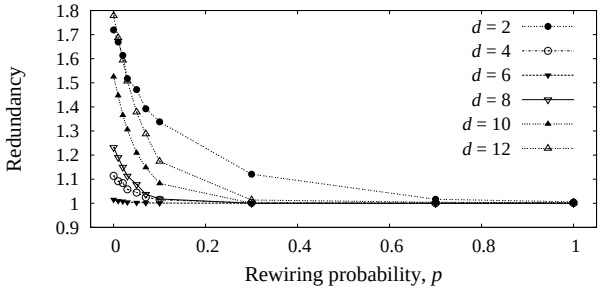
When the regularity of the graphs varies, the path lengths and clustering in the graphs change the structure. Random graphs have low clustering, and since peers of degree d are likely to have d 1-hop neighbors, d^2 2-hop neighbors, d^3 3-hop neighbors, *etc*., it is likely that any two peers will have a short path length between them. Alternatively, regular graphs are likely to have long paths between peers and significant clustering. Some analytical and numerical results for these metrics are presented in [14]. Clearly, the number of neighbors d of a peer characterizes the sparsity of the topology.

A. Impact of Randomness and Sparsity

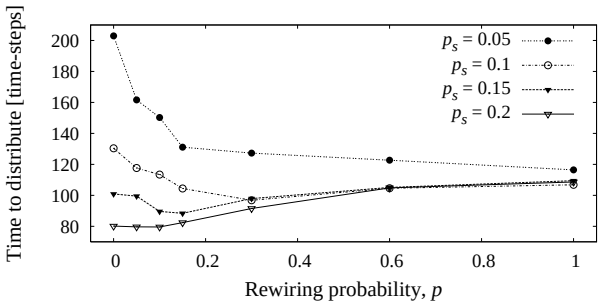
We first evaluate a network of 100 peers using segments of 100 data blocks. Each peer forwards a coded block constructed from $m = 6$ coded blocks with aggressiveness $a = 0.04$, server connectivity $p_s = 0.15$, and the segment contains 100 data blocks. Link delays follow a uniform random distribution in $[0.75t_s, 1.25t_s]$. The impact of both randomness and sparsity on redundancy and distribution times is exhibited quantitatively in Fig. 4.

We first examine the effect of randomness. Recall that rewiring with $p = 0$ forms a regular topology where we see high clustering and long path lengths between peers. In a regular network, we expect redundancy because the topology is designed so that peers are likely to share neighbors. The long path lengths also tend to increase the time to distribute n independent data blocks to all peers. On the other hand, rewiring with $p = 1$ creates a totally random topology with very low clustering. Since it is unlikely for peers to share neighbors, we expect lower redundancy and short path lengths should decrease the distribution times. Fig. 4 confirms the above intuition. Furthermore, we observe that, in most cases, the effects of randomness decrease significantly from $p \approx 0.1$ to $p = 1$.

[1]These graphs are also sometimes labelled as having "six degrees of separation."



(a) Average redundancy experienced at a peer



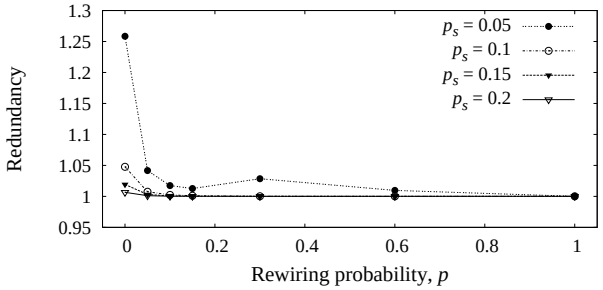
(b) Time to complete block forwarding

Fig. 4. Performance experienced in a network with different levels of randomness and sparsity.

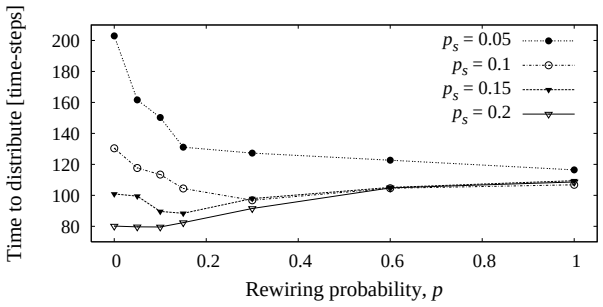
With respect to sparsity, we observe that the degree, d, of the peers has a significant impact on redundancy introduced by network coding. Fig. 4 shows that choosing $d = 6$ as the degree of peers leads to the best performance of the network when the rewiring probability p is not too high. This observation conforms to our intuition that, too few neighbors result in topologies where there is infrequent introduction of new information, leading to redundant blocks sent between peers. Too many neighbors, however, also lead to common downstream peers receiving the same information from multiple sources.

Server cost is another key concern when scaling any P2P network to large numbers of nodes. Clearly, the fraction of peers connected to the server p_s has a direct impact on the server cost. The cost is also indirectly affected by the connectivity of the peers themselves. Networks may experience more forwarding of independent blocks due to the choice of server distribution period or due to the available links between the peers. In either of these cases, the server cost is reduced.

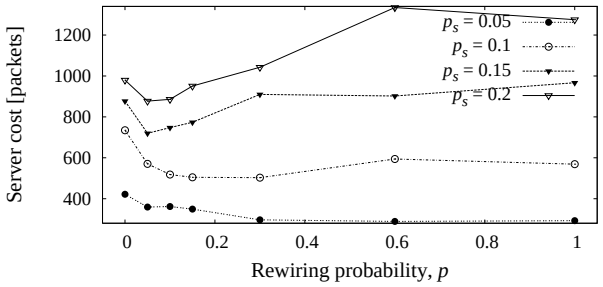
Fig. 5 allows us to consider the server cost in terms of the redundancy and distribution time metrics for four values of server connection probability p_s. The parameter values are similar to those in the previous figure except where noted. As is intuitively clear, we see a minimum in the server cost when redundancy is low and the distribution time is low for any p_s. However, the lowest server cost is achieved (at the expense of distribution time) when the server has the fewest downstream peers.



(a) Average redundancy experienced at a peer

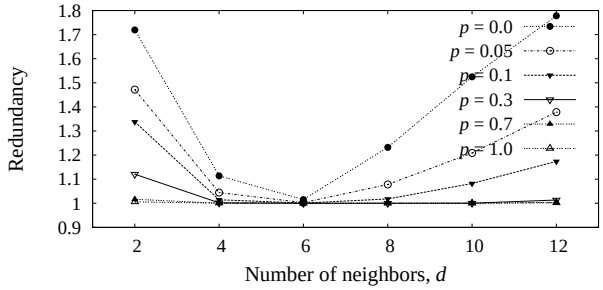


(b) Time to complete block forwarding

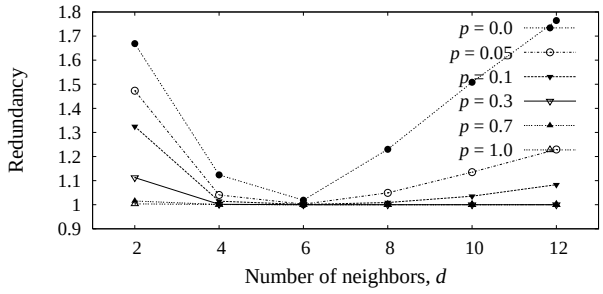


(c) Cost to the server

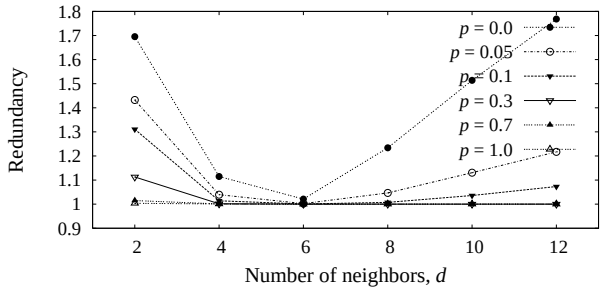
Fig. 5. The choice of p_s affects redundancy and server cost.



(a) Network of 100 peers



(b) Network of 500 peers



(c) Network of 1000 peers

Fig. 6. Regardless of the number of peers, 6 neighbors show the best performance.

B. Impact of Network Size

Up to this point, the sparsity of network topologies has been varied in the context of the degree d of each peer. Another notion of sparsity is related to the number of peers in the network, N. In a global sense, increasing the number of peers in the network decreases the overall connectivity; that is, the likelihood that a particular peer is directly linked to another particular peer is smaller. Hence it may appear that the ratio between d and N has a significant effect on the coding efficiency.

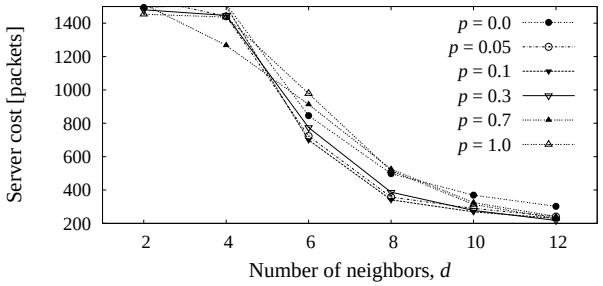
Fig. 6 shows that it is not the global connectivity that has the most significant impact on network coding redundancy. It is instead the local connectivity (the peer degree) that dictates the redundancy in the network. The values of redundancy are nearly identical as the number of network peers varies for a large range of rewiring probabilities p. Again, we observe that, as long as the topology is not completely random, the optimal number of downstream peers is 6.

Other performance metrics differ considerably, however. Perhaps the most notable scaling metric as N increases is the server cost. Shown in Fig. 7, the server cost increases approximately linearly with N, but is lower for smaller (non-zero) values of the rewiring probability. In other words, regular topologies with a few long-distance links are preferred over purely random topologies. Topologies with peers of degree six to ten experience more than twice as much server cost in a purely random network than a small-world network with rewiring probability 0.05.

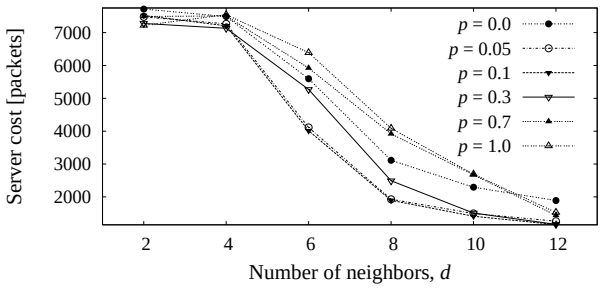
As one would expect, the distribution time increases sub-linearly as N increases. Path lengths increase between peers as $\log N$, and distribution times increase even more slowly than that because it is not necessary for the same blocks to reach every part of the network. Since network coding requires any n independent blocks for decoding, the distribution time scales more effectively. Our simulation results confirm this conclusion. However, we have omitted the graphs due to space constraint.

V. Conclusion on Preferred Topologies

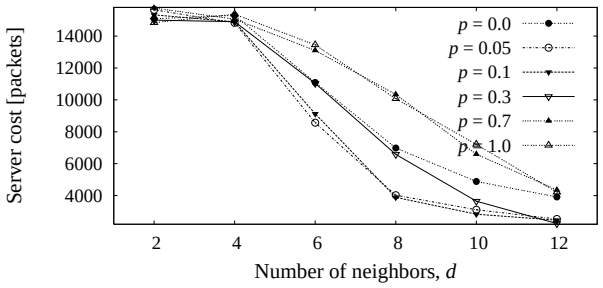
We have observed — using both examples and empirical studies — that peer-to-peer networks with network coding experience inefficiencies due to the timing of arrivals of



(a) Network of 100 peers



(b) Network of 500 peers



(c) Network of 1000 peers

Fig. 7. Server cost increases approximately linearly with N, and much better performance is observed for small p as N scales up.

relayed blocks and due to the links formed between peers. Although these inefficiencies naturally present themselves, we have found that network parameters can be chosen to intentionally minimize redundancy and optimize performance.

Based on extensive studies in the previous section, we have shown that the peer-to-peer topologies offering the best overall performance are small-world topologies with low rewiring probability (around $p = 0.1$) with peer degree of six[2]. These networks enjoy low redundancy, comparable with purely random networks, but have lower distribution times. Furthermore, in such preferred topologies, peers have a sufficient number of neighbors for effective distribution, without too many peers sharing downstream neighbors. Last but not the least, small-world networks with low rewiring probability exhibit significantly lower server cost than corresponding networks with more randomness.

Small-world topologies with low p also benefit from the ability to achieve better overhead and load balance than purely random networks without overwhelming cost to the server or to the peers. Finding random neighbors for each peer is usually a costly operation that requires global knowledge in the network. When the rewiring probability is small, messaging overhead is only needed for a small fraction of the links, and local links are straightforward to acquire.

The above observations can be applied to generate design guidelines for P2P topology design [15] when network coding is used. In particular, in the case when multiple data segments are to be distributed (*e.g.,* peer-to-peer streaming), good load balance can be achieved by changing the rewired links for different segments of the data stream. Clearly, data arriving at a peer from a different part of the network (along a rewired "long-distance" link) is more likely to be independent from coded blocks acquired locally. Changing the rewired links will alter the flow of data through the network, so that peers would appear to have different positions in the topology. Averaging over many data segments, each peer should serve approximately the same number of coded blocks to the other peers.

[2]We have also observed that a suitable aggressiveness factor is around $0.04 - 0.05$ to balance the distribution time and redundancy. An extensive study on the optimal aggressiveness factor is omitted due to the limitation on paper length.

REFERENCES

[1] R. Ahlswede, N. Cai, S. R. Li, and R. W. Yeung, "Network Information Flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, July 2000.

[2] B. Cohen, "Incentives Build Robustness in BitTorrent," in *Proc. of Workshop on Economics of Peer-to-Peer Systems*, June 2003.

[3] C. Gkantsidis and P. Rodriguez, "Network Coding for Large Scale Content Distribution," in *Proc. of IEEE INFOCOM 2005*, March 2005.

[4] R. Koetter and M. Medard, "An Algebraic Approach to Network Coding," *IEEE/ACM Transactions on Networking*, vol. 11, no. 5, pp. 782–795, October 2003.

[5] S. Y. R. Li, R. W. Yeung, and N. Cai, "Linear Network Coding," *IEEE Transactions on Information Theory*, vol. 49, p. 371, 2003.

[6] P. Sanders, S. Egner, and L. Tolhuizen, "Polynomial Time Algorithm for Network Information Flow," in *Proc. of the 15th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA 2003)*, June 2003.

[7] T. Ho, R. Koetter, M. Medard, D. Karger, and M. Effros, "The Benefits of Coding over Routing in a Randomized Setting," in *Proc. of International Symposium on Information Theory (ISIT 2003)*, 2003.

[8] P. Chou, Y. Wu, and K. Jain, "Practical Network Coding," in *Proc. of Allerton Conference on Communication, Control, and Computing*, October 2003.

[9] C. Gkantsidis, J. Miller, and P. Rodriguez, "Anatomy of a P2P Content Distribution System with Network Coding," in *Proc. of the 5th International Workshop on Peer-to-Peer Systems (IPTPS 2006)*, 2006.

[10] M. Wang and B. Li, "How Practical is Network Coding?" in *Proc. Fourteenth IEEE International Workshop on Quality of Service*, June 2006, pp. 274–278.

[11] T. Ho, M. Medard, J. Shi, M. Effros, and D. Karger, "On Randomized Network Coding," in *Proc. of Allerton Conference on Communication, Control, and Computing*, October 2003.

[12] S. Milgram, "The Small World Problem," in *Psychology Today*, vol. 2, 1967, pp. 60–67.

[13] D. Watts and S. Strogatz, "Collective dynamics of 'small-world' networks," in *Nature*, vol. 393, June 1998, pp. 440–442.

[14] M. E. J. Newman, "Models of the Small World," *Journal of Statistical Physics*, vol. 101, no. 3-4, pp. 819–841, November 2000.

[15] T. Small, B. Li, and B. Liang, "Outreach: Peer-to-Peer Topology Construction towards Minimized Server Bandwidth Costs," *IEEE Journal on Selected Areas in Communications, Special Issue on Peer-to-Peer Communications and Applications*, vol. 25, no. 1, pp. 35–45, May 2007.