Applying Network Coding to Peer-to-Peer File Sharing

Min Yang and Yuanyuan Yang, *Fellow*, *IEEE*

Abstract—Network coding is a promising enhancement of routing to improve network throughput and provide high reliability. It allows a node to generate output messages by encoding its received messages. Peer-to-peer networks are a perfect place to apply network coding due to two reasons: the topology of a peer-to-peer network is constructed arbitrarily, thus it is easy to tailor the topology to facilitate network coding; the nodes in a peer-to-peer network are end hosts which can perform more complex operations such as decoding and encoding than simply storing and forwarding messages. In this paper, we propose a scheme to apply network coding to peer-to-peer file sharing which employs a peer-to-peer network to distribute files resided in a web server or a file server. The scheme exploits a special type of network topology called combination network. It was proved that combination networks can achieve unbounded network coding gain measured by the ratio of network throughput with network coding to that without network coding. Our scheme encodes a file into multiple messages and divides peers into multiple groups with each group responsible for relaying one of the messages. The encoding scheme is designed to satisfy the property that any subset of the messages can be used to decode the original file as long as the size of the subset is sufficiently large. To meet this requirement, we first define a deterministic linear network coding scheme which satisfies the desired property, then we connect peers in the same group to flood the corresponding message, and connect peers in different groups to distribute messages for decoding. Moreover, the scheme can be readily extended to support link heterogeneity and topology awareness to further improve system performance in terms of throughput, reliability and link stress. Our simulation results show that the new scheme can achieve 15%–20% higher throughput than another peer-to-peer multicast system, Narada, which does not employ network coding. In addition, it achieves good reliability and robustness to link failure or churn.

Index Terms—Network coding, peer-to-peer networks, web-based applications, file sharing, multicast

1 Introduction and Related Work

In the last several years, the Internet has witnessed tremendous increase of different types of web-based applications, ranging from web-based file sharing to video broadcasting/conferencing. Web-based applications have gained more and more interests due to the flexibility and easy accessibility. Many such applications involve one source (server) and multiple destinations (receivers). However, due to lack of multicast support over the Internet, these applications usually suffer from the scalability problem, which limits the number of receivers involved. *Peer-to-peer* is a promising technology that can implement multicast at the application layer, where receivers (peers) not only receive data, but also forward data. By incorporating peer-to-peer technology into web-based applications, the scalability problem can be eliminated, i.e., the system performance (throughput, latency, etc.) will not be impaired when there are more users in the system.

In this paper, we consider applying peer-to-peer technology to file sharing services, in which a web server or a file server holds a file that is requested by multiple clients (receivers). In most peer-to-peer systems, peers usually are end users' personal computers which may have limited resources or even be unstable. It is critical for the file sharing system to be reliable and resilient while achieving good throughput at the same time.

Network coding is another promising technology that can be employed to improve system throughput and reliability. Here we give a brief introduction on network coding. In today's network, messages are generally transferred by routing through intermediate nodes between the source and the destination, i.e., by having intermediate nodes store and forward messages. In fact, routing is not the only operation that can be performed at a node. Recently, network coding has emerged as a promising enhancement of routing to improve network throughput and provide high reliability. Network coding refers to a scheme where a node is allowed to generate output messages by encoding (i.e., computing certain functions of) its received messages. Thus, network coding allows information to mix, in contrast to the traditional routing approach where each node simply forwards received messages. Network coding has a wide range of applications from wireless networks [16], [28] to network tomography [17]. Network coding can greatly improve the throughput of a multicast network. Let's first use the well-known butterfly network in Fig. 1 to demonstrate the advantage of network coding for multicast. In the figure, node s is the source, nodes r_1 and r_2 are two receivers, all edges have capacity 1, which means that the edge can only transmit 1 unit of data (bit) per unit time (second), and source s has two bits, b_1 and b_2 to multicast to both r_1 and r_2. First we use the traditional multicast without network coding as shown in Fig. 1(b). Without loss of generality, we use the dashed line (red) to

• *The authors are with the Department of Electrical and Computer Engineering, Stony Brook University, Stony Brook, NY 11794. E-mail: myang@ece.sunysb.edu, yuanyuan.yang@stonybrook.edu.*

Manuscript received 06 Jan. 2011; revised 19 Dec. 2012; accepted 26 Mar. 2013. Date of publication 10 Apr. 2013; date of current version 15 July 2014. Recommended for acceptance by D. Avresky.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TC.2013.88

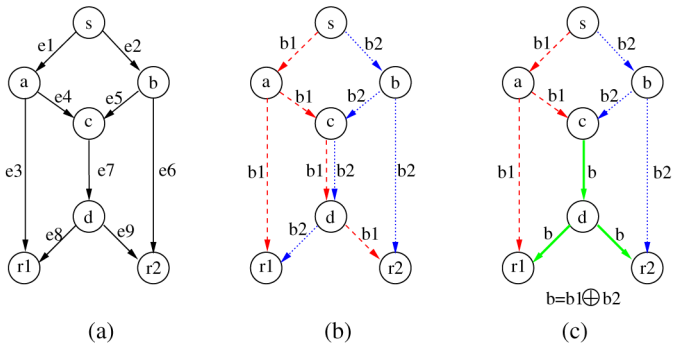


Fig. 1. Illustration of the advantage of network coding for multicast. (a) The butterfly network represented by a DAG. (b) Multicast without network coding. (c) Multicast with network coding.

represent bit b_1, the dotted line (blue) to represent bit b_2 and the bold line(green) to represent both bits b_1 and b_2. Bit b_1 can reach r_1 in two seconds, and bit b_2 can reach r_2 in two seconds. When node c receives both bits, it forwards them in sequence. Suppose it forwards bit b_1 first. Then r_1 receives both bits in 4 seconds and r_2 receives both bits in 5 seconds. Now consider using network coding on link c–d as shown in Fig. 1(c). When node c receives both bits, it first mixes them by performing an exclusive OR (XOR) operation. Then it sends the mixed bit b to node d. When nodes r_1 or r_2 receive the mixed bit, it can recover the original bits b_1 and b_2 by XORing the mixed bit and the other received bit. The entire transmission can be completed in 4 seconds.

The above example reveals that network coding can increase the throughput of multicast. In general, given a network topology, a source node s and a set of receiver nodes $R = \{r_1, r_2, \ldots\}$, *multicast capacity* is defined as the maximum rate at which the source can send messages to the receivers, that is, it is equal to $\min_{r_i \in R}\{MaxFlow(s, r_i)\}$ where $MaxFlow(s, r_i)$ is the maximum flow between source s and receiver r_i. In the example in Fig. 1, multicast capacity of the network is 2. Without network coding the throughput is 1.5, while with network coding the throughput is 2. In fact, Ahlswede et al. [1] has generally proved that if network coding is allowed in network nodes, then multicast capacity can be achieved, that is, network capacity can be fully utilized in multicast.

A multicast network [26], [27], [29] can be modeled by a directed graph where each node has multiple incoming edges and outgoing edges. Constructing a *network coding scheme* for a multicast network is equivalent to assigning a function to each edge which defines the mix operation for that edge. This function is called *edge function*. Suppose the edge is one of the outgoing edges of node v. Its edge function takes the messages on the incoming edges of node v as input and outputs a message to be sent on itself. A *network coding scheme* is the union of the edge functions of all the edges in the multicast network. A *valid network coding scheme* is a network coding scheme in which all the receivers can decode the original messages based on the messages they receive.

Let messages be represented by symbols over a Galois field $GF(q)$, where q is a prime number or a power of a prime number. *Linear network coding* refers to network coding schemes where the edge functions are linear functions over $GF(q)$. If node w has c incoming edges $\{e_1, e_2, \ldots, e_c\}$ and one of the outgoing edges is e, then the edge function of edge e in a linear network coding scheme can be denoted by a vector $V_e = (v_1, v_2, \ldots, v_c)$. The message on edge e is generated by the linear function $M_e = v_1 M_{e_1} + v_2 M_{e_2} + \cdots + v_c M_{e_c}$, where M_{e_i} represents the message transmitted on edge e_i.

In a network with multicast capacity k, the source can transmit k messages simultaneously. If we represent the k messages by a k dimensional vector M over $GF(q)$, the messages transmitted over edge e can be represented by the product of vector M and another k dimensional vector V'_e. Vector V'_e is called the *edge vector* of edge e. We have $V'_e = v_1 V'_{e_1} + v_2 V'_{e_2} + \cdots + v_c V'_{e_c}$. A *valid linear network coding scheme* is a linear network coding scheme where the rank of the matrix composed by the edge vectors of the receiver's incoming edges is equal to the dimension of M, which is k in this case.

In recent years, there has been some work on linear network coding in the literature. Li, et al. [2] showed that linear network coding over a finite field is sufficient to achieve multicast capacity. Kotter, et al. gave an algebraic characterization for a linear network coding scheme in [3]. They also gave an upper bound on the field size and a polynomial time algorithm to verify the validity of a network coding scheme. Ho, et al. presented a random linear network coding approach in [4], [5] in which nodes generate edge vectors randomly. Clearly, the linear network coding scheme generated by this approach is not always valid. They proved that the probability of failure is $O(1/q)$ where q is the size of the finite field. In [6] [7], Lun, et al. proposed a distributed algorithm to find a subgraph of the original topology such that the link cost can be minimized without sacrificing multicast capacity. In contrast to the random network coding, Jaggi, et al. proposed a polynomial deterministic algorithm in [19] that can construct deterministic linear network coding schemes for multicast networks.

Peer-to-peer (overlay) networks are a perfect place to apply network coding due to two reasons: the topology of a peer-to-peer network is constructed arbitrarily. It is easy to tailor the topology to facilitate network coding; the nodes in a peer-to-peer network are end hosts which can perform more complex operations such as decoding and encoding than simply storing and forwarding messages. In [8], linear network coding was applied to application layer multicast, in which a rudimentary mesh graph is first constructed, and on top of it a rudimentary tree is formed. Then a multicast graph is constructed, which is a subgraph of the rudimentary mesh and a supergraph of the rudimentary tree. The multicast graph constructed this way is 2-redundant, which means that each receiver has two disjoint paths to the source. By taking advantage of the 2-redundancy property of the multicast graph, a light-weight algorithm generates a sequence of 2-dimensional transformation vectors which are linearly independent. These vectors are assigned to the edges as their edge functions. However, the paper did not discuss how to process dynamic joining or leaving of peers, while dynamic membership is a common phenomenon in peer-to-peer networks. Moreover, the 2-redundancy property limits the minimum cut of the multicast graph, which in turn limits network throughput.

In [20], random network coding was applied to content distribution, in which nodes encode their received messages

with random coefficients. Compared to deterministic network coding, random network coding is inherently distributed. In random network coding, nodes can determine the edge functions of its outgoing edges independently by generating random coefficients for the edge functions. The advantage of random network coding is that there is no control overhead to construct and maintain a linear coding scheme among nodes. However, the edge vectors of a receiver's incoming edges may not be linearly independent. In other words, a receiver may not recover the original messages even it receives k or more messages (here k is the multicast capacity of the multicast network). To reduce the probability of failing to decode messages, it is required to encode over a very large field. Another drawback of random network coding is the increased data traffic. As there is no deterministic path for data delivery, all the nodes take part in relaying the data to the receivers even if it is not necessary. As a result, the same message may be transmitted through the same link multiple times.

The motivation of the paper is to design an efficient and reliable file sharing service over peer-to-peer networks by taking advantage of the good properties of network coding and applying it to peer-to-peer networks in a proper way. In this paper, a new peer-to-peer file sharing scheme is proposed, which is called *Peer-to-Peer FilE sharing based on nEtwork coDing*, or *PPFEED* for short. We utilize a special type of network with a regular topology called *combination network*. It was demonstrated in [15] that when the network size increases, this type of network can achieve unbounded network coding gain measured by the ratio of network throughput with network coding to that without network coding. The basic idea of PPFEED is to construct an overlay network over the source, i.e., the file server, and the receivers such that it can be decomposed into multiple combination networks. Compared to [8], our approach can accommodate dynamic membership and construct a much simpler overlay network topology in different k values. Compared to [20], our network coding scheme is deterministic, which means that the validity of the coding scheme is guaranteed. The data traffic is then minimized so that the same messages are transmitted through an overlay link at most once. Also, system reliability is improved dramatically with little overhead. In addition, PPFEED can be extended to support link heterogeneity and topology awareness.

The rest of the paper is organized as follows. In Section 2, we describe the topology and the properties of the combination network. We also design a simple deterministic linear network coding scheme for the combination network. In Section 3, we describe how PPFEED works in detail, how it handles topology mismatch and link heterogeneity and discuss its good fault tolerance ability. We give our simulation results and compare the performance of the scheme in Section 4. Finally, Section 5 gives the concluding remark and future work.

2 Deterministic Linear Coding over Combination Networks

One of the advantages of network coding is that it can increase network throughput by achieving multicast capacity. *Network coding gain* is defined as the ratio of throughput with network coding to that without network coding. Clearly, it is always greater than or equal to 1. When applying network coding to a multicast network, the gain depends on the topology of the multicast network. For example, if the topology of a multicast network is a tree, i.e., the source is the root of the tree and the receivers are the leaves of the tree, network coding gain is 1. Intuitively, if each receiver has roughly the same number of disjoint paths to the source, the more overlap among the paths, the larger network coding gain can be achieved.

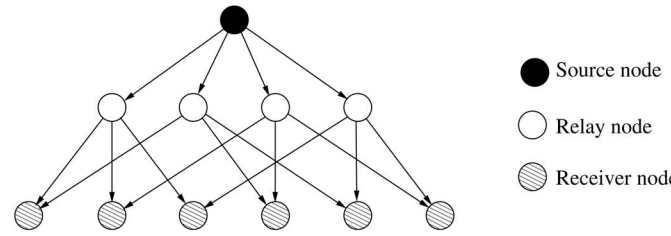


Fig. 2. Combination network C_4^2.

A combination network is a multicast network with a regular topology. The topology of a combination network is a regular graph which contains three types of nodes: *source node*, *relay node* and *receiver node*. A combination network contains a source node which generates messages, n relay nodes which receive messages from the source node and relay them to the receiver nodes, and C_n^k receiver nodes which receive messages from the relay nodes. There are n links connecting the source node to the n relay nodes respectively. For every k nodes out of the n relay nodes, there are k links connecting them to a receiver node. Since there are a total of C_n^k different combinations, the number of receiver nodes is C_n^k. The capacity of each link is 1. Fig. 2 shows a combination network for $n = 4$ and $k = 2$, usually denoted as a C_4^2 combination network. For clarity, the nodes are arranged into three layers: the first layer contains only one node, the source node; the second layer contains n relay nodes; the third layer contains C_n^k receiver nodes.

Combination networks have good performance with respect to network coding gain. It was proved [15] that network coding gain is unbounded when both n and k approach infinity. Although it is impossible to satisfy this condition in reality, it is conceivable that there is a great potential to combine those two. For a C_n^k combination network, the multicast capacity is k as each receiver node has k disjoint paths to the source node. This implies that to achieve the multicast capacity in a combination network, the minimum subgraph to deploy network coding is the entire topology of the combination network.

We have discussed the good properties of combination networks. The next issue is how to design a valid linear network coding scheme on a combination network. As mentioned earlier, existing network coding methods can be classified into two categories: random method and deterministic method. Random network coding [4] can only obtain a valid coding scheme with a probability less than 1. In [19], a polynomial deterministic algorithm was proposed to construct a valid linear coding scheme for any multicast network. However, it is a centralized algorithm which needs to know the complete topology information before constructing the linear coding scheme. Moreover, since it is designed for general topologies, it involves many complex processes such

as keeping track of the paths between the source and the receivers and maintaining an extra array to facilitate linear independence test. On the other hand, for a multicast network with a regular topology like a combination network, it is possible to design a tailored linear network coding scheme which takes advantage of the regular topology to achieve both effectiveness and efficiency. In this paper, we propose a simple deterministic linear coding scheme which can guarantee its validity and achieve $O(1)$ time complexity.

Before we go into details of our linear network coding scheme construction, we first determine an appropriate value for k, which is the multicast capacity of a combination network. On one hand, given the number of relay nodes n, the network coding gain reaches its maximum when $k = n/2$ [15]. On the other hand, k should not be too large because a large k will increase the link stress and overhead. Therefore, as a tradeoff, in this paper we only consider combination networks with $n = 4$, $k = 2$ or $n = 6$, $k = 3$ where the network coding gain is 1.5 and 2, respectively. The n and k determine the size of the combination network. As we will see later, the size of the overlay network is independent of the size of the combination network. If we apply the same n and k to different overlay networks of various sizes, the performance difference is very small.

As mentioned earlier, constructing a valid linear network coding scheme is equivalent to assigning each edge an edge vector such that the vectors of the incoming edges of a receiver are linearly independent. Suppose the source has k symbols to multicast to the receiver nodes. In a C_n^k combination network, there are a total of $n + kC_n^k$ edges. The first n edges connect the source node to n relay nodes with edge vectors $V_1', V_2', \ldots, V_n'$. For each relay node, there are kC_n^k/n outgoing edges connecting it to kC_n^k/n receiver nodes. As each relay node has only one incoming edge, the edge vector of its outgoing edge is the edge vector of its incoming edge multiplied by a constant. Since the constant does not change the linear independence property, we only consider the case that the constant is 1. Each receiver has k incoming edges whose edge vectors are k vectors out of the n vectors $V_1', V_2', \ldots, V_n'$. For a receiver node to decode the original messages, the k edge vectors must be linearly independent. Thus the main issue is how to find a set of n k-dimensional vectors such that every k vectors of the n vectors are linearly independent.

Suppose $GF(q)$ is a given Galois field, where $|GF(q)| = q, GF = \{0, 1, 2, \ldots, q-1\}, q \geq n$. We define the linear network coding schemes for C_n^2 and C_n^3 combination networks as follows.

Definition 1. *The linear network coding scheme for C_n^2 combination network is to assign vectors $(1, \alpha_1), (1, \alpha_2), \ldots, (1, \alpha_n)$, where $\alpha_1, \alpha_2, \ldots, \alpha_n$ are different symbols in $GF(q)$, to n edges connecting to n relay nodes as edge vectors. The edge vectors of the edge outgoing from one relay node is the same as the edge vector of the edge incoming to the relay node.*

Definition 2. *The linear network coding scheme for C_n^3 combination network is to assign vectors $(1, \alpha_1, \alpha_1^2 \bmod q)$, $(1, \alpha_2, \alpha_2^2 \bmod q), \ldots, (1, \alpha_n, \alpha_n^2 \bmod q)$, where $\alpha_1, \alpha_2, \ldots, \alpha_n$ are different symbols in $GF(q)$, to n edges connecting to n relay nodes as edge vectors. The edge vectors of the edge outgoing from one relay node is the same as the edge vector of the edge incoming to the relay node.*

We have the following theorem concerning the linear coding scheme.

Theorem 1. *The proposed linear coding schemes defined in Definitions 1 and 2 are valid.*

Proof[1]. Case 1: $k = 2$. For any two vectors $(1, \alpha), (1, \beta)$ where α and β are two different symbols in a Galois field, we evaluate the determinant of matrix $\begin{pmatrix} 1 & \alpha \\ 1 & \beta \end{pmatrix}$. We have

$$\begin{vmatrix} 1 & \alpha \\ 1 & \beta \end{vmatrix} = \beta - \alpha \tag{1}$$

Since $\alpha \neq \beta$, the determinant is not equal to 0. We conclude that the two vectors are linearly independent.

Case 2: $k = 3$. For any three vectors $(1, \alpha, \alpha^2 \bmod q)$, $(1, \beta, \beta^2 \bmod q)$ and $(1, \gamma, \gamma^2 \bmod q)$, where α, β and γ are three different symbols in a Galois field and $\gamma > \beta > \alpha$, the determinant of matrix $\begin{pmatrix} 1 & \alpha & \alpha^2 \\ 1 & \beta & \beta^2 \\ 1 & \gamma & \gamma^2 \end{pmatrix}$ is

$$\begin{aligned} \begin{vmatrix} 1 & \alpha & \alpha^2 \\ 1 & \beta & \beta^2 \\ 1 & \gamma & \gamma^2 \end{vmatrix} &= \beta\gamma^2 + \alpha\beta^2 + \alpha^2\gamma - \alpha^2\beta - \alpha\gamma^2 - \beta^2\gamma \\ &= \beta\gamma(\gamma - \beta) + \alpha\beta(\beta - \alpha) + \alpha\gamma(\alpha - \beta + \beta - \gamma) \\ &= (\beta - \alpha)(\gamma - \beta)(\gamma - \alpha) \end{aligned} \tag{2}$$

Since $\alpha \neq \beta \neq \gamma$, the determinant is not equal to. We conclude that the three vectors are linearly independent. □

In the rest of the paper, the value of k is either 2 or 3.

3 Peer-to-Peer File Sharing Based on Network Coding (PPFEED)

In this section, we present the PPFEED scheme that provides a peer-to-peer file sharing service over the Internet. We first give an overview of PPFEED and describe the basic idea of constructing an overlay network composed of multiple combination networks and applying network coding on the overlay network. Then we go into details by describing the processes of peer joining/leaving and data dissemination. Finally, we discuss some optimizations which can improve the reliability and resilience of the system.

3.1 Overview of PPFEED

We assume that there is a server holding the file to be distributed. Peers interested in the file form an overlay network through which the file is distributed. The construction of the overlay network is based on the idea of combination networks. Similar to the combination networks, there are two system parameters n and k in the overlay network. The server encodes the file into n different messages using the linear coding scheme given Section 2. Thus any k messages out of the n messages can be used to decode the original file. The parameters of the encoding functions are predetermined once n and k are determined. There is no real-time parameter generation as in random or other deterministic linear network

1. Since the mod operation is distributive over addition and multiplication, we omit $\bmod q$ in the equations.

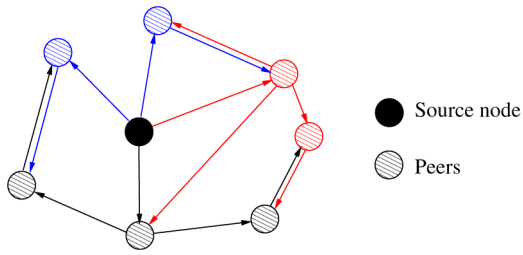


Fig. 3. An example overlay network constructed by PPFEED.

coding schemes. This makes it take constant time ($O(1)$) to generate the encoding function. The peers are divided into n disjoint groups and each group is assigned a unique *group ID*. Each group of peers is responsible for relaying one of the n encoded messages. To accelerate the distribution of the messages, peers in the same group are connected by an unstructured overlay network according to some loose rules. Each peer in a group is connected to at least other $k-1$ peers which are in $k-1$ different groups respectively.

Although the topology is constructed based on combination networks, it is more a mesh network than a multi-tree network. If we color peers with n different colors, the entire overlay topology can be looked as a mesh with the constraint that each peer must be connected to at least k peers with k different colors. Therefore, PPFEED enjoys the scalability and resilience of a mesh network. In Section 4, we will quantify its performance through simulations.

Fig. 3 shows an example of the overlay network constructed by PPFEED for $n=3$ and $k=2$. There are three groups of peers which are colored in black, blue and red, respectively. The colored links represent corresponding messages with arrows pointing to the transmission directions. We can see that each peer has at least two links with different colors pointed to itself. All the peers are able to decode the received messages.

The overlay links are added to the system on demand as new peers join the system. There are two key issues in the overlay network construction: first, how to connect the peers in the same group; second, how to connect the peers in different groups.

The first issue involves how to distribute the n messages in a scalable way. When the number of peers is small, it is acceptable to connect all the peers to the source node, i.e., the server. When the number of peers increases, the peers resort to each other instead of the server to distribute the messages. Here we adopt some loose rules similar to the unstructured peer-to-peer network Gnutella [23]: a newly joined peer first contacts the server which will respond with a random list of the existing peers, and the new peer creates connections with these peers. The reason is three folds. First, it is resilient to peer join. A new peer can be connected to any existing peer. Second, the overlay construction and maintenance overhead is low. There are no constraints on the overlay topology, such as node degrees or graph diameter, as long as it is connected. If some peers leave the system, it is convenient for the rest of the peers to reconnect through the server. Third, it is easy to flood messages on the resulting overlay network.

The second issue involves how the peers receive other $k-1$ messages to decode the messages. As any k different messages can be used to decode, we only need to find $k-1$ peers in different groups. Since there are $n-1$ eligible groups, we first connect the peer to $k-1$ random peers in different groups. Then perform a local topology adjustment (discussed in III-E) to find the $k-1$ peers such that the average latency between the peer and $k-1$ peers is minimized.

Some erasure codes, such as Reed-Solomon or Tornado, have similar property to PPFEED in terms of decoding the original messages from a group of encoded messages. In [10], the authors have applied Tornado code to reliable distribution of bulk data to a variety of heterogeneous population of receivers. The erasure codes encode the original messages into a sequence of encoded messages such that any subset of the encoded messages can be used to reconstruct the original messages as long as the size of the subset is sufficiently large. However, they have different objectives and applications. The erasure codes are designed to avoid retransmissions, particularly in unicast communication, while network coding is used to improve system throughput for multicast networks. In some applications, retransmissions may be impossible or undesirable. For example, in multicast applications, retransmission requests may greatly overwhelm the sender if the group size is large. In satellite networks, where the back channel has high latency and limited capacity, retransmission requests are costly and unreliable. With the erasure codes, the sender can keep sending the sequence of encoded messages regardless of whether the receivers receive them or not. As long as the receivers receive enough encoded messages, they can extract the original messages from the received messages. On the other hand, network coding is applied when multiple flows share a common link where the messages from different flows can be mixed together to save bandwidth. Network coding can be applied to multicast networks or wireless networks where messages sharing common links can be encoded together to improve system throughput.

As mentioned earlier, we consider the combination networks with $n=4$, $k=2$ or $n=6, k=3$ in this paper. When we apply PPFEED to a large overlay network, as n and k are fixed, the number of peers in the same group increases. Each peer is still connected to the other $k-1$ peers in different groups. The scalability of PPFEED is derived from the scalability of the overlay network composed of peers within the same group. As described earlier, the overlay network is constructed by some loose rules to accommodate scalability. Therefore, PPFEED can be applied to overlay networks of various network sizes without performance degradation.

3.2 Topology Awareness

A common problem for overlay networks is topology mismatch between the logical network and the physical network. Since overlay networks are logical networks on top of physical networks, the overlay links are logical links. Each logic link is composed of one or more physical links. The overlay links are added arbitrarily as needed. As a result, the topology of the overlay network may be different from the topology of the physical network. Two nodes which are close to each other in the overlay network may be far away in the physical network. Such topology mismatch may greatly increase the *link stress* and degrade the performance. Link stress is defined as follows:

Definition 3. *A physical link's link stress is the number of times that the same message is transmitted over a certain physical link.*

We address this problem by constructing a topology aware overlay network. We propose a *topology clustering scheme* by adopting the idea of the binning scheme introduced in [24]. Some terminologies deserve to be clarified before we go into details. *Landmarks* are a set of peers chosen to be the reference points for other peers to determine their "locations." Here location represents the position of a peer in the network with regard to latencies between peers. *Coordinate* is a list of numbers which represent the location of a peer. A *cluster* is a set of peers with the same coordinates. In this scheme, the server is responsible for choosing some peers as landmarks. Each new peer will receive the list of landmarks before it receives the list of peers. The new peer sends probe messages to the landmarks to learn the distances between them and itself. The landmark peers are listed in an ascending order of distances. The ordered list acts as the coordinate of the peer in the system. The peers with the same coordinates are considered close to each other. The coordinate is sent to the server, then the server assigns the new peer a group ID based on its coordinate and some other factors which we will discuss later. Peers with the same coordinates form a cluster. Each peer is assigned to a group based on the network coding scheme discussed in III-A. As group division and cluster division are interacting, we use two heuristic rules to guide the group or cluster assignment of peers to optimize the system performance. First, each cluster has at least k different groups; Second, the peers in the same group should span as few clusters as possible. The first rule is to ensure that peers can receive enough messages to decode within its cluster. The second rule is to minimize the number of links across clusters. However, the rules are not mandatory. In reality, it is possible that the number of peers in the same cluster is less than k. In this case, the peers in the cluster have to connect to more peers in other clusters to receive a sufficient number of messages to decode.

To increase the accuracy of the coordinates of peers, no two landmark peers have the same coordinate: a new peer cannot be a landmark if its coordinate is the same as one of the existing landmark peers; a landmark peer is removed from the landmark peers if it has the same coordinate as another landmark peer. To further improve the accuracy, two landmark peers should not be "too close" to each other. A threshold value can be used to quantify the "closeness" of two landmark peers, i.e., if the number of equal numbers in the two coordinates is larger than some threshold, the two landmark peers are considered to be too close to each other.

3.3 Link Heterogeneity

Link heterogeneity is a common phenomenon in networks. In peer-to-peer networks, link heterogeneity refers to the fact that peers have different access link capability. Common access links include dial-up connections (56 Kbps modem), DSL (128 Kbps to 10 Mbps) and high speed cable/LAN (10–100 Mbps). The ratio of the link capacities of the fastest link to the slowest link can be more than 1000.

Due to the link capacity imbalance, if a peer with high link capacity is receiving messages from a peer with low link capacity, its download speed is upper bounded by the download speed of the low link capacity peer. In other words, the bandwidth of the high link capacity peer is wasted.

In PPFEED, we consider how to maximize the utilization of link capacity of each peer. Suppose the bottleneck is always on the access links. Thus we can construct as many as possible overlay links connected to the peer as long as its access link capacity permits. As mentioned earlier, the overlay construction between the peers in the same group is arbitrary. We take advantage of this property to construct an overlay network such that the number of links of a peer is proportional to its link capacity. Peers attach their upload bandwidth to the JOIN request packet. The server will maintain a list of peers with their respective upload bandwidths in each group. When the server receives a JOIN request, it will first assign the peer to a group and then respond with a list of peers with most residue upload bandwidths to the new peer. The new peer will connect to the peers in the list. The residue upload bandwidth is the available upload bandwidth for each peer. It is updated by peers and reported to the server. The update frequency is throttled to avoid overwhelming the server. The scheme described above is simple and resilient to dynamic peer join/leave. Our purpose is to alleviate the negative effect due to link heterogeneity without imposing too much overhead on the existing system.

Next, we discuss how PPFEED works in more detail.

3.4 Peer Joining

We assume that the server is well-known whose IP address is known to all the peers by some address translation service such as DNS. When a peer wants to retrieve a file hosted by the server, it initiates a join process by sending a JOIN request to the server.

The server maintains several counters and lists. For each group G_i, the server maintains a counter gc_i to store the number of peers in the group and a list gl_i to store the list of clusters which include the peers in the group. For each cluster C_i, the server maintains a list cl_i to store the groups which include peers in the cluster. Besides, the server maintains a list of existing peers and their respective residue upload bandwidths and IP addresses for each group. As more and more peers join the system, it is resource consuming to maintain a full list of peers for each group. The server will keep a partial list of peers with the largest residue upload bandwidths. Meanwhile, peers will report to the server their updated residue upload bandwidths periodically to update the partial list on the server. Although the server is responsible for bootstrapping the peers, it will not be the bottleneck of the system because once each peer receives the list, it communicates with other peers for topology construction and data dissemination. When the server receives a peer's join request, it assigns the peer to a group. Then the server sends the list of peers of that group to the joining peer and updates the number of peers in that group.

The server will assign the new peer to a group based on following factors. First, the peer is assigned to a cluster based on its coordinate. If the number of groups which include peers in the cluster is less than k, the new peer will be assigned to one of the groups which include no peers in the cluster. When there are multiple such groups, the group spans the least number of clusters is selected. Tie is broken by picking the group with less number of peers in the group. The rationale behind this is that we want to minimize the logical links between different clusters and ensure that peers can receive sufficient "innovative" messages, i.e., messages from different groups, within the cluster to perform decoding.

TABLE 1
Peer Joining Algorithm

```
INPUT: joining peer v
OUTPUT: updated overlay network
BEGIN
  //Suppose the cluster corresponding to peer v is C_i
  if |cl_i| < k
    S_i = the set of groups not in cl_i;
  else
    S_i = the set of groups in cl_i;
  pick a group g_i in S such that gl_i is the smallest;
  if multiple groups have the same smallest gl, pick a group g_i
   with less gc_i;
  peer v is assigned to group g_i.
END
```

After receiving the list of peers, the new peer will contact them and create overlay links with them. These peers are called intra-neighbors of the new peer because they are within the same group. In contrast, the neighbors which are in different groups are called inter-neighbors. The new peer asks one of its intra-neighbors to provide a list of its inter-neighbors. When picking the intra-neighbor, higher priority is given to the peer in the same cluster. The new peer then takes the list of peers as its inter-neighbors.

The topology of the peer-to-peer network can be considered as a combination of multiple unstructured peer-to-peer networks, each of which is composed of the peers within the same group. The topology within one group is arbitrary as long as it is connected. The only constraint is on the edges between different groups. It is required that each peer is connected to at least $k-1$ peers in $k-1$ different groups respectively. When more than $k-1$ peers are connected, the system reliability can be improved significantly which we will discuss later in this section.

The pseudo-code for peer joining is listed as Table 1 shows.

3.5 Local Topology Adjustment

In the above we describe a landmark based solution to solve the topology mismatch problem. However, the systematic approach has some limitations. First, it is a centralized algorithm which does not scale well with the growing size of the system. Second, it can not provide optimal topology solution for peers within the same cluster. In this section, we introduce local topology adjustment to adjust the topology locally in a distributed way.

The idea of local topology adjustment is to replace the direct neighbors with peers with better performance through a local search. Assume the clocks of all peers are synchronized by some mechanism, such as NTP. Each peer periodically sends QUERY messages to its neighbors. There are two types of QUERY messages: one for intra-neighbor searching and one for inter-neighbor searching. Suppose peer u sends out a QUERY message to one of its neighbors, say, v. After receiving the QUERY message, node v will send a RESPONSE message back to u and a QUERY-2 message to each of its neighbors except for u. The RESPONSE message includes a TIMESTAMP field which records the time when node v sends the RESPONSE message. After node u receives the message, it calculates the latency between u and v by the difference between the time when the RESPONSE message is received and the time indicated by the TIMESTAMP in the RESPONSE message. The QUERY-2 message includes the IP address of node u. After a neighbor of v, say, w, receives the QUERY-2 message, it sends a RESPONSE-2 message to node u. It also includes a TIMESTAMP field which records the time when w sends the RESPONSE-2 message. After node u receives the message, it calculates the latency between u and w by the difference between the time when the RESPONSE-2 message is received and the time indicated by the TIMESTAMP in the RESPONSE-2 message. If the latency between u and w is less than that between u and v, node w will replace node v and become the neighbor of node u.

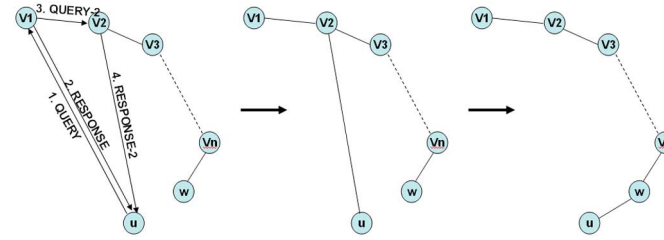


Fig. 4. Illustration of local topology adjustment.

Fig. 4 illustrates an example for local topology adjustment. In the figure, an overlay topology with $n+2$ nodes connected as a list. We use the distance between nodes to represent the latency roughly. At the beginning, node v_1 is the neighbor of node u. After n steps of local topology adjustment, node w becomes the neighbor of node u as the topology on the right shows.

Local topology adjustment is a distributed light-weight overlay topology optimization process. It can detect a better neighbor after searching the nodes nearby with a search radius of 2. The process is done periodically as the topology evolves. However, in some cases, local topology adjustment cannot find the best neighbor with the shortest latency. Fig. 5 shows one of these cases, where although node u and node w are close to each other, they cannot detect each other through local topology adjustment. In this case, we need the landmark based solution. Therefore, local topology adjustment is a complement to the landmark based solution.

3.6 Peer Leaving

There are two types of peer leaving: friendly or abruptly. Friendly leaving means that the leaving peer initiates a leaving process so that the system is aware of its leaving and can make necessary updates accordingly. Abruptly leaving means that the leaving peer leaves the system without any notification, mainly due to link crash or computer crash.

For the friendly leaving, the leaving peer will initiate a leaving process by sending LEAVE messages to both of its

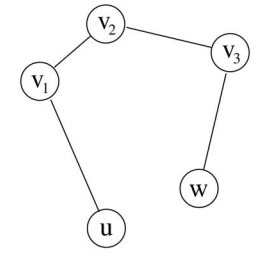


Fig. 5. An example that local topology adjustment does not work.

intra-neighbors and inter-neighbors. The leaving of the peer may impair the connectivity between peers in the same group. To rebuild the connectivity, the intra-neighbors will initiate the join process with the group ID in the join request. After receiving a join request with a designated group ID, the server temporarily acts as one of the intra-neighbors of the peer to guarantee the connectivity. The topology is further adjusted by the local topology adjustment process. Here the server acts as a connection "hub" for the peers that were connected to the leaving peer. This may increase the data forwarding burden on the server temporarily. Nevertheless, the situation will be improved after the local topology adjustment process is completed. Moreover, it can achieve strong robustness with little control overhead. For example, it can handle concurrent peer leavings. The connectivity is rebuilt after the server receives all the join requests from the intra-neighbors of the leaving peers.

After receiving the LEAVE message from the leaving peer, the inter-neighbors will ask one of its intra-neighbors for a new inter-neighbor to replace the leaving one. In addition, the leaving peer will also send a LEAVE message to the server. This LEAVE message will make the server update the number of peers in the group.

For the abruptly leaving, peers send HELLO messages to its neighbors (both intra and inter-neighbors) periodically and maintain a HELLO timer for each neighbor. Receiving a HELLO message from one of its neighbors triggers a reset of the corresponding HELLO timer. When a peer is disconnected from the network due to some reason, the neighbors detect the abruptly leaving by the timeout of the corresponding HELLO timer. Similarly, the intra-neighbors initiate the join process after detecting the sudden leave. The inter-neighbors ask their intra-neighbors for a replacement of the left peer. Moreover, one of the neighbors is chosen to send a LEAVE message to the server on behalf of the abruptly leaving peer so that the server can update the number of the peers in the group. To minimize the selection overhead, we choose the inter-neighbor whose group ID is next to that of the leaving peer to perform this task. For example, if we assume the encoding vector for the group corresponding to the leaving peer is $(1, \alpha)$ (or $(1, \alpha, \alpha^2 \, mod \, q)$), inter-neighbor corresponding to the group whose encoding vector is $(1, (\alpha + 1) \bmod n)$, or $(1, (\alpha + 1) \bmod n, ((\alpha + 1) \bmod n)^2 \bmod q)$ is chosen.

3.7 Data Dissemination

Before sending out the file, the server needs to encode the file. The encoding is over a Galois field $GF(q)$. The file is divided into multiple blocks with each block represented by a symbol in $GF(q)$. The first k blocks are encoded and then the second k blocks, and so on. In the case that there are not enough blocks to encode, the file is padded with zero string. Assuming the field size is $q(> n)$, each k blocks are encoded into n different messages using the linear coding scheme given in Section 2. Therefore, any k messages of these n messages can be used to decode the original k blocks.

After encoding, the server sends the encoded n messages to the peers in the n groups respectively. The group ID and the encoding function form a one-to-one mapping. Peers can learn which messages they receive based on the sender of the messages: if the sender is the server, the messages correspond to the group ID of the peer itself; if the sender is a peer, the messages correspond to the group ID of the sender.

The peers forward all the messages they receive based on the following rules:

Rule 1. If the message comes from the server, the peer forwards it to all its intra-neighbors and inter-neighbors.

Rule 2. If the message comes from one of its intra-neighbors, the peer forwards it to other intra-neighbors except for the sender.

Rule 3. If the message comes from one of its inter-neighbors, the peer does nothing.

Peers forward messages in a push style, which means that the messages are forwarded under the three rules as soon as they arrive at a peer. A peer decodes the messages right after it receives k different messages. Thus data dissemination is fast and simple in PPFEED.

3.8 Improving Reliability and Resilience to Churn

Besides the throughput improvement, PPFEED can provide high reliability and high resilience to *churn* which refers to frequent peer joins or leaves. All we need to do is to add a redundant overlay link for each peer.

In the previous sections, each peer is connected to $k-1$ inter-neighbors. The $k-1$ messages received from them plus the message received from the source or the intra-neighbors should be sufficient to decode the original file blocks. However, no links are 100% reliable. In case that the messages are lost or damaged, the sender has to retransmit the messages until they are received correctly. Clearly, retransmission will cause longer latency, larger buffer size and reduce system throughput. PPFEED can reduce the retransmission probability by introducing a redundant link to each inter-neighbor. Now each peer is connected to k instead of $k-1$ peers in different groups. If one of the links fails, the peer can still decode the original file blocks based on the remaining $k-1$ messages. The failure probability of this scheme can be quantitatively analyzed as follows. Suppose each overlay link will fail with probability $1-p$. In the original scheme, the peer can successfully decode the message only if it receives the $k-1$ messages from the $k-1$ neighbors, which has a probability of p^{k-1}. Therefore the probability it fails to decode is $P_{failure} = 1 - p^{k-1}$, while in the improved scheme, the peer fails to decode with probability $P'_{failure} = 1 - (p^k + k(1-p)p^{k-1})$. The ratio of the two probabilities is $P_{failure}/P'_{failure} = (1-p^{k-1})/(1-(p^k + k(1-p)p^{k-1}))$. Fig. 6 plots the curves of the ratio for different p values when $k = 2$ and $k = 3$. We can see that when $p = 0.9$, the failure probability of the original scheme is about 10 times of the improved scheme.

Another advantage of connecting peers with more inter-neighbors than necessary is that it can improve the system resilience to churn. The rationale is similar to the reliability improvement. For example, if we connect a peer with k other peers, it can tolerate one peer's leave without affecting the download speed. Peers experiencing unstable neighborhood should be connected to more than one redundant inter-neighbors to alleviate the performance degradation.

Finally, connecting peers with more inter-neighbors can enable peer to detect "malicious" node which is "polluting" the messages. If one "malicious" peer modifies the encoded messages, the peer receiving the "polluted" message will get different decoded messages if it uses the "polluted" message

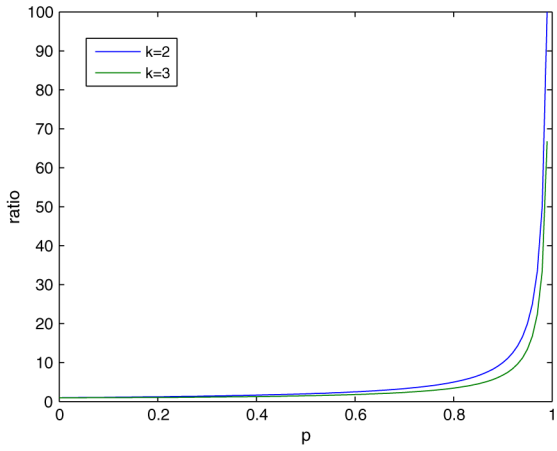


Fig. 6. Failure probability ratio of the original scheme to the improved scheme.

for decoding. A checksum is attached to each message to facilitate the detection of corrupted messages. In this case, it can ask the server to send the correct message directly and detect which neighbor is the "malicious" peer. If there are more than one "malicious" peers, peers should connect to more than one redundant neighbors as well. This may increase the server's workload temporarily. After the "malicious" peer is detected, the neighbors can ignore the messages from the "malicious" peer.

4 Performance Evaluations

In this section, we study the performance of PPFEED through simulations. Among a number of existing related works, we choose Narada [18], a peer-to-peer multicast system, and Avalanche [20], a peer-to-peer file sharing system, as the performance comparison counterparts. Narada is a representative work in application layer multicast. It first constructs an overlay mesh spanning over all the peers. The overlay mesh is a richer connected graph which satisfies some desirable performance properties, such as scalability and partition avoidance. The multicast tree is a spanning tree on top of the mesh and is constructed on demand of the source peer. Avalanche is a well-known peer-to-peer file sharing system based on random network coding. BitTorrent [9] is another well-known peer-to-peer file sharing system. It breaks a file to small pieces and peers help each other to download the whole file. The rarest pieces are given the highest weight and there is an incentive mechanism to encourage uploading. However, it lacks a comprehensive solution to handle cases such as topology awareness or link heterogeneity. Given that Avalanche is based on network coding and its performance is better than BitTorrent [20], it is more appealing to compare to Avalanche. Thus, the two comparison counterparts are chosen to evaluate the benefit of network coding in general as well as that of a tailored deterministic network coding.

The simulation adopts following three performance metrics:

Throughput: throughput is defined as the service the system provides in one time unit. Here we let different systems transmit the same file, thus throughput can be simply represented by the time consumed by the transmission. The shorter the time consumed, the higher the throughput. We start transmitting the file from time 0. Then the consumed time is the time when the peers finish receiving the file, denoted by *finish time*.

Reliability: this performance metric is used to evaluate the ability of the system to handle errors. We use the *number of retransmissions* to characterize this ability. A system with higher reliability will have a smaller number of retransmissions, and thus higher throughput.

Link stress: link stress is defined as the number of copies of the same message transmitted through the same link. It is a performance metric that only applies to an overlay network due to the mismatch between the overlay network and the physical network. We use it to evaluate the effectiveness of the topology awareness improvement and the efficiency of the system.

(i) We study the performance of the system in four different configurations:

(ii) Baseline configuration. In this configuration, peers have uniform link capacities, and overlay links are constructed randomly. The file is sent after the overlay network is formed.

(iii) Dynamic peer join/leave configuration. In this configuration, peers have uniform link capacities, and overlay links are constructed randomly. Peers join the system during the file transmission and stay in or leave the system after downloading the whole file.

(iv) Heterogeneity configuration. In this configuration, peers have heterogeneous link capacities, and overlay links are constructed by taking into consideration of link heterogeneity. The file is sent after the overlay network is formed.

(v) Topology awareness configuration. In this configuration, peers have uniform link capacities, and overlay links are constructed by taking into consideration of topology mismatch. The file is sent after the overlay network is formed.

The network topologies are random graphs generated by GT-ITM [25]. For each simulation, we usually run 10 times and calculate the average to eliminate the performance variations between different simulation runs. For throughput and reliability, the simulation graphs are drawn based on the average values of 10 simulation runs under the same constraints: the same network topology, the same set of peers and the same transmitted file. For link stress, each point in the graph represents the average value of 10 different sets of random peers in the same network topology.

We conducted the same simulations for $k = 2$ and $k = 3$. In the rest of this section, we will omit the result for $k = 2$ when it is similar to that of $k = 3$.

4.1 Baseline Configuration

In the baseline configuration, peers have uniform link capacities. The simulation is divided into two steps. First, overlay construction period: A number of random nodes are picked to join the system in sequence. Second, file transmission period: The server sends the file to the overlay network.

We plot the finish time curves of PPFEED and Narada in Fig. 7(a). We simulate PPFEED with different k values as shown in the figure. The finish time of nodes is sorted in an ascending order. It can be seen that the average finish time of PPFEED is 15–20% shorter than that of Narada and 8–10% shorter than Avalanche. We notice that the finish times of Narada spans a broader range which leads to a larger variance than that of Avalanche and PPFEED. This is because that in

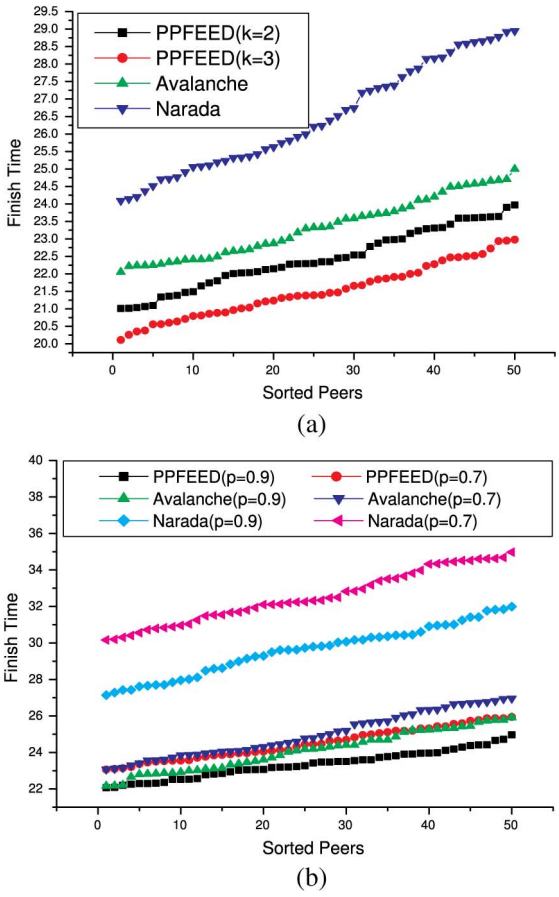


Fig. 7. (a) Finish time of the baseline configuration. (b) Finish time of the baseline configuration with link failures.

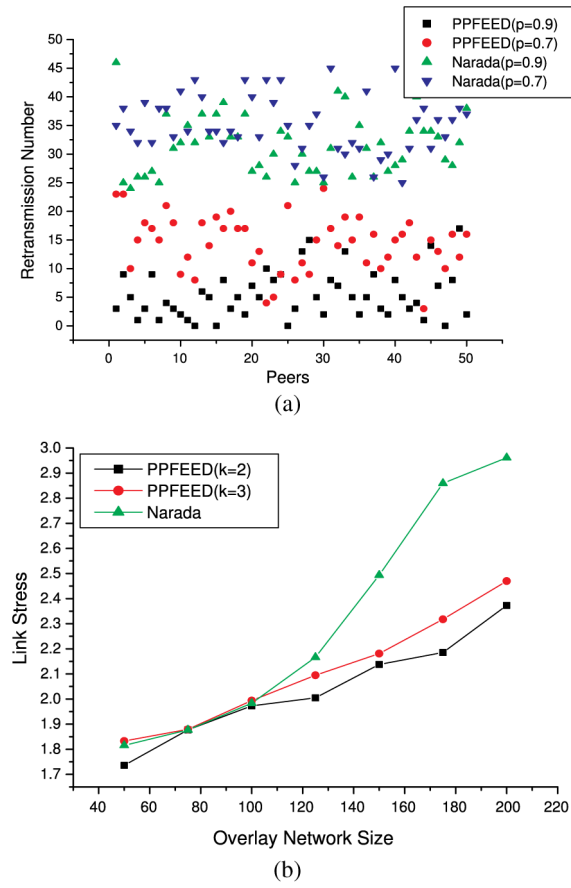


Fig. 8. (a) The number of retransmissions of the baseline configuration with link failures. (b) Link stress of the baseline configuration.

Narada, the two peers with the biggest difference in finish time are a child of the root and a leaf, respectively. This difference may be very large depending on the overlay topology. On the contrary, Avalanche and our scheme construct a mesh to distribute the file. As a result, the distance between the highest level peer and the lowest level peer is shortened. The reason for PPFEED outperforming Avalanche with respect to throughput is two folds. First, Avalanche adopts random network coding which may generate some messages without innovative information. This will cause some delay for receivers to perform decoding. PPFEED adopts a deterministic network coding scheme with low complexity and control overhead to guarantee that each message is innovative. Second, Although PPFEED imposes some topology constraints to the overlay network due to deterministic network coding, the constraints are very flexible and resilient. For example, the topology within the same group is arbitrary. Each peer can choose arbitrary $k-1$ peers in other groups as its inter-neighbors. Therefore, these constraints have little impact on the performance. From the figure we can see that the throughput of PPFEED is higher for $k=3$ than that for $k=2$. This can be explained by the fact that when $k=3$, each peer is connected to more peers. Thus the download capacity of peers can be better utilized.

Fig. 7(b) shows the finish time when we set the physical link failure probability to $1-p$. Note that the link failure probabilities of different physical links are independent. In the analysis in Section 3.8, we simply assumed that the link failure probability of an overlay link is $1-p$ to simplify the analysis. However, the link failure probabilities of overlay links are dependent due to sharing common physical links. Here we use link failure probabilities of physical links to simulate real networks more accurately. We can see that the finish times of PPFEED and Avalanche are much shorter than that of Narada. With the same link failure probability, the finish time of PPFEED is slightly shorter than that of Avalanche. Compared to the previous simulation result without link failure probabilities, the increase of finish time is the least for Avalanche and the largest for Narada. The reduction of throughput is small for both Avalanche and PPFEED when physical links are unstable.

The number of retransmissions is shown in Fig. 8(a). As Avalanche does not require retransmission, we only plot the curves for PPFEED and Narada. We use colored dots to denote the numbers of retransmissions of peers. We can see that Narada needs more retransmissions than PPFEED. The less the p is, the more retransmissions are needed. The average number of retransmissions of PPFEED is about 5 when $p=0.9$ while that of Narada is about 30. Both Figs. 7(b) and 8(a) reveal that PPFEED has a good fault tolerance ability due to the redundant link.

Each packet in Avalanche is unique, the link stress of Avalanche is always 1. We compare the link stresses of PPFEED and Narada as shown in Fig. 8(b). When the number of peers is small, PPFEED and Narada have similar link stress. However, when the number of peers is beyond 100, the links stress of PPFEED is less than Narada. The reason for this is similar to that for the larger finish time variance of Narada. If we track a message in Narada, the paths it travels through

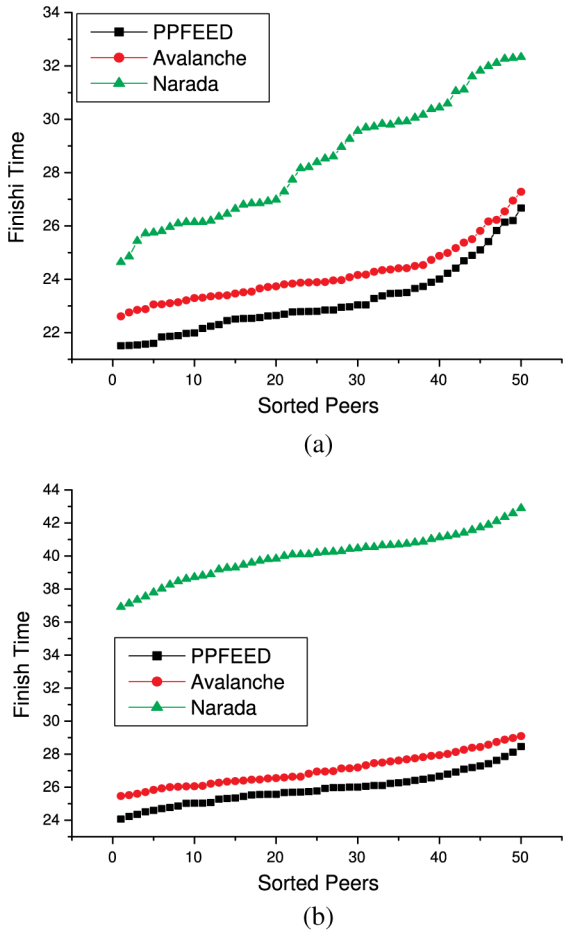


Fig. 9. Finish time of the dynamic peer join/leave configuration. (a) Peers stay in the system after receiving the file. (b) Peers leave the system after receiving the file.

form a tree spanning all the peers. If we track a message in PPFEED, the paths it travels through form a mesh spanning a portion of the peers (roughly peers if the overlay network is well balanced, where $1/n$ is for the peers within the same group, $(k-1)/n$ is for the peers in different groups). Fewer overlay links will reduce the probability that the same message travels through the same physical link. The link stress of PPFEED is higher when $k = 3$ than $k = 2$. When k is larger, the number of peers in the same groups is reduced. On the other hand, each peer is connected to more peers in different groups. As a result, the increased links between different groups outnumber the reduced links due to the reduced group size. Thus the link stress is increased.

4.2 Dynamic Peer Join/Leave Configuration

In this configuration, we allow peers to join and leave the system during the file transmission to evaluate the system resilience to churn. The simulations are conducted in two scenarios: First, we let peers join the system randomly and the file transmission starts as long as there are peers requesting it. After receiving the file, peers stay in the system; Second, the peers still join the system randomly. When a peer successfully receives the file, it leaves the system right away. In both scenarios, we calculate the finish time of a peer by the difference between the time it finishes downloading the file and the time it joins the system.

Fig. 9(a) shows the finish time comparison when peers stay in the system after receiving the file. We can see that the average finish time of all three schemes increases compared to the baseline configuration. For PPFEED and Avalanche, this is due to the lack of peers which help forwarding the file. For Narada, this is due to the join latency of peers. The increase of average finish time for Narada is more than that of PPFEED or Avalanche, which indicates that peer-to-peer systems achieve better resilience to dynamic joins than tree-based approaches. The peers with different finish times are not evenly distributed as in the baseline configuration. The number of peers with larger finish time increases. The largest finish time for Avalanche is close to that of PPFEED. This is because PPFEED needs to download the file from k peers from k different groups while Avalanche has no such requirements.

Fig. 9(b) shows the finish time comparison when peers leave the system after receiving the file. We can see that the average finish times of PPFEED and Avalanche are increased by around 15% while that of Narada is increased by around 50%. Tree-based approaches are extremely vulnerable to churn as each departure will disconnect all the downstream peers and the tree needs to be rebuilt. Both PPFEED and Avalanche have similar resilience under the churn. However, PPFEED achieves slightly higher throughput. It indicates that PPFEED achieves great resilience under dynamic peers join/leave. Although PPFEED adopts deterministic network coding, the overlay topology in PPFEED is quite flexible. In addition, by adding redundant links, the resilience can be improved dramatically.

4.3 Heterogeneity Configuration

Now we evaluate the ability of PPFEED to handle peers with heterogeneous link capacities. The random topologies generated by GT-ITM are flat random graphs with high speed links. Peers are added to the nodes randomly with each peer connected to one node by an access link and the access link is set to a certain link capacity. We set the peers with heterogeneous link capacities such that 1/3 with the highest link capacities, 1/3 with the lowest link capacities and 1/3 with the medium link capacities. The highest link capacity is 10 times of the lowest one, and 3 times of the medium ones.

Fig. 10(a) shows the finish time comparison of the heterogeneity configuration between the overlay construction with heterogeneity consideration and without heterogeneity consideration. We can see that the finish time with heterogeneity consideration is much shorter than the baseline which does not consider heterogeneity. However, the variance of the finish time is almost the same. It indicates that the peers with higher link capacities are helpful to increase system throughput, but they cannot reduce the finish times of themselves.

Fig. 10(b) shows the link stress comparison. In contrary to the finish time, the link stress with heterogeneity consideration is larger than that without heterogeneity consideration. One reason is that the access links of the high capacity peers are used by many other peers to construct overlay links. As a result, the messages sent by a high capacity peer are more likely transmitted through the same physical link. When the size of the overlay network is small, peers are distributed around the network sparsely. The link stress is mainly determined by the positions of the peers. In some cases, the link stress when $k = 2$ is even greater than that when $k = 3$.

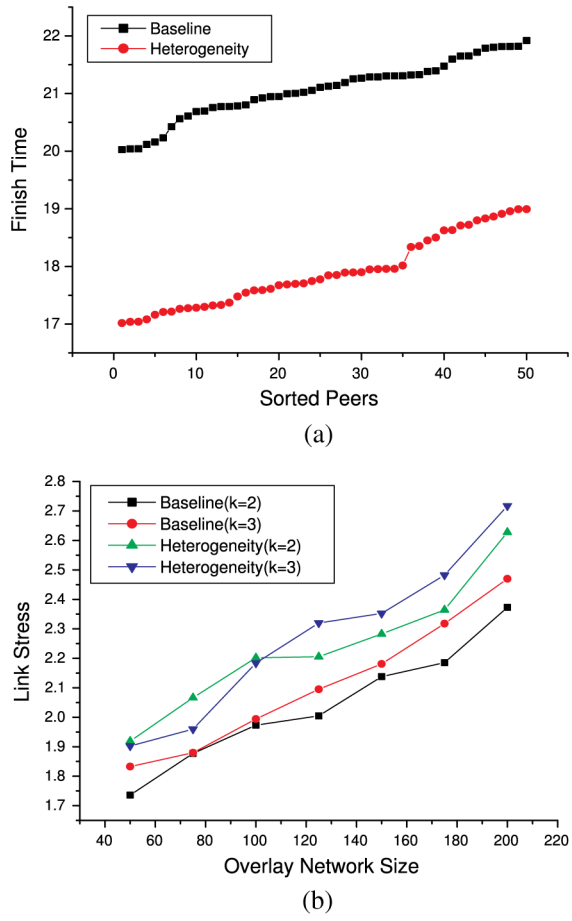


Fig. 10. (a) Finish time of the heterogeneity configuration. (b) Link stress of the heterogeneity configuration.

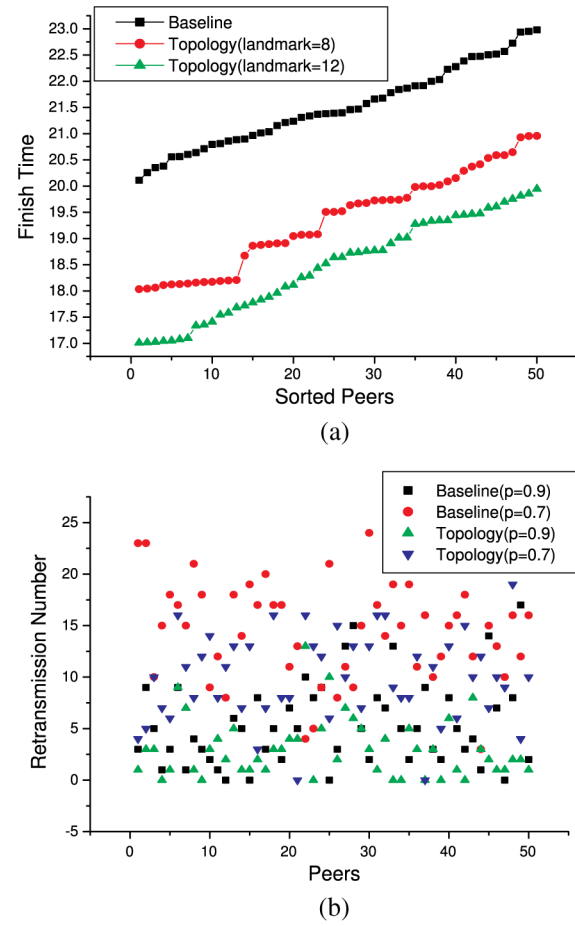


Fig. 11. (a) Finish time of the topology awareness configuration. (b) The number of retransmissions of the topology awareness configuration.

4.4 Topology Awareness Configuration

We now study the performance of PPFEED when considering the physical network topology during the overlay network construction.

Fig. 11(a) shows the finish time at different number of landmarks. The highest curve is the same curve in the baseline configuration when $k = 3$. We can see that topology clustering reduces the finish time by about 10% compared to that without topology clustering. Increasing landmarks can increase the accuracy of topology clustering, thus the finish time is shortened. We notice that the curve of the finish time when the number of landmarks is 8 has a staircase shape. This is because that the peers in the same cluster may finish receiving the file at roughly the same time. While the finish times between different clusters may be longer. When the number of landmarks is 12, the cluster size is reduced. Peers may not receive $k - 1$ different messages within the same cluster, thus the staircase disappears.

We set the physical link failure probability to $1 - p$. Fig. 11(b) shows the number of retransmissions. Compared to the baseline configuration, the average number of retransmissions of the topology awareness configuration is slightly smaller. Generally speaking, the improvement of topology clustering on the number of retransmissions is small. The main reason of the improvement is due to redundant links.

One of the biggest advantages of topology clustering is to reduce the link stress. From Fig. 12, we can see that the link stress of the topology awareness configuration is reduced by about 37% compared to that of the baseline configuration when the number of landmarks, which are used as reference points for other peers as discussed in III-B, is 12. With the increase of the overlay network size, the difference is even bigger. Increasing landmarks makes the link stress smaller.

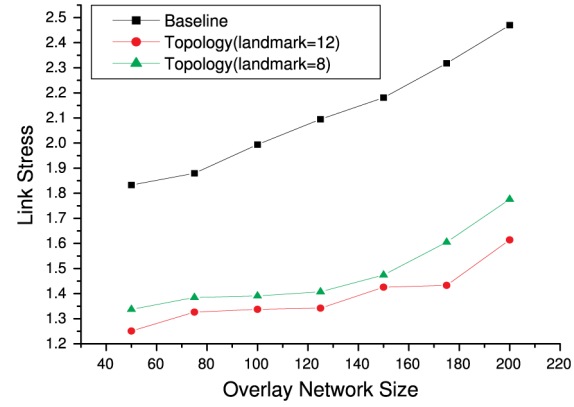


Fig. 12. Link stress of the topology awareness configuration.

5 Conclusion

In this paper, we have proposed a peer-to-peer file sharing scheme based on network coding, PPFEED. The scheme can serve as a peer-to-peer middleware created within the web services framework for web-based file sharing applications. Compared to other file sharing schemes, the advantages of

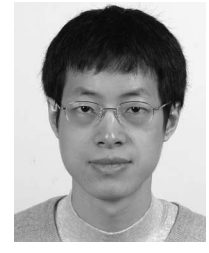
our scheme can be summarized as follows. (a) Scalability. Files are distributed through a peer-to-peer network. With the increase of the network size, the total available bandwidth also increases. (b) Efficiency. The linear network coding scheme is deterministic and easy to implement. There is no requirement for peers to collaborate to construct the linear coding scheme on demand. All the peers need is the mapping between the group ID and the encoding function, and this mapping does not change with time. Compared to random network coding, the receiver can always recover the original messages after receiving k different messages and the data dissemination is more efficient as data messages are transmitted through the same overlay link at most once. (c) Reliability. The redundant links can greatly improve the reliability of the system with little overhead. (d) Resilience. Churn is a common problem in overlay networks. By adding redundant links, the negative effect of churn is alleviated. (e) Topology awareness. Simulation results show that the proposed topology clustering scheme can greatly reduce link stress and improve throughput. (f) Heterogeneity support. In case that links have different link capacities, PPFEED can arrange the overlay topology to maximize the utilization of each peer's link capacity. Our future work includes how to optimize the system under unstable network status such as congestion.

Acknowledgment

The research was supported in part by the U.S. National Science Foundation under grant number CCF-0744234.

References

[1] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. Inf. Theory*, vol. 46, no. 4, pp. 1204–1216, Jul. 2000.
[2] S.-Y. R. Li, R. W. Yeung, and N. Cai, "Linear network coding," *IEEE Trans. Inf. Theory*, vol. 49, no. 2, pp. 371–381, Feb. 2003.
[3] R. Koetter and M. Medard, "An algebraic approach to network coding," *IEEE/ACM Trans. Netw.*, vol. 11, no. 5, pp. 782–795, Oct. 2003.
[4] T. Ho, M. Medard, J. Shi, M. Effros, and D. R. Karger, "On randomized network coding," in *Proc. Annu. Allerton Conf. Commun. Control Comput.*, 2003, pp. 4413–4430.
[5] T. Ho, M. Medard, R. Koetter, D. Karger, M. Effros, J. Shi et al., "A random linear network coding approach to multicast," *IEEE Trans. Inf. Theory*, vol. 52, no. 10, pp. 4413–4430, Oct. 2006.
[6] D. S. Lun, N. Ratnakar, R. Koetter, M. Medard, E. Ahmed, and H. Lee, "Achieving minimum-cost multicast: A decentralized approach based on network coding," in *Proc. IEEE INFOCOM'05*, Mar. 2005, pp. 1607–1617.
[7] D. S. Lun, M. Medard, T. Ho, and R. Koetter, "Network coding with a cost criterion," in *Proc. Int. Symp. Inf. Theory Appl. (ISITA'04)*, Oct. 2004, pp. 1232–1237.
[8] Y. Zhu, B. C. Li, and J. Guo, "Multicast with network coding in application-layer overlay networks," *IEEE J. Sel. Areas Commun.*, vol. 22, no. 1, pp. 107–120, Sep. 2004.
[9] BitTorrent. (2004) [Online]. Available: http://bittorrent.com
[10] J. W. Byers, M. Luby, and M. Mitzenmacher, "A digital fountain approach to asynchronous reliable multicast," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 3, pp. 1528–1540, Oct. 2002.
[11] A. G. Dimakis, P. B. Godfrey, M. J. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," in *Proc. IEEE INFOCOM'07*, May. 2007, pp. 4359–4551.
[12] D. M. Chiu, R. W. Yeung, J. Huang, and B. Fan, "Can network coding help in P2P networks?," in *Proc. Int. Symp. Model. Optimiz. Mobile Ad Hoc Wireless Netw.*, 2006, pp. 1–5.
[13] M. Kim, C. W. Ahn, M. Medard, and M. Effros, "On minimizing network coding resources: An evolutionary approach," in *Proc. NetCod*, 2006.
[14] K. Bhattad, N. Ratnakar, R. Koetter, and K. R. Narayanan, "Minimal network coding for multicast," in *Proc. IEEE Int. Symp. Inf. Theory*, Sep. 2005, pp. 1730–1734.
[15] C. K. Ngai and R. W. Yeung, "Network coding gain of combination networks," in *Proc. IEEE Inf. Theory Workshop*, Oct. 2004, pp. 283–287.
[16] C. Fragouli, J. Y. Le Boudec, and J. Widmer, "On the benefits of network coding for wireless applications," in *Proc. NetCod*, 2006, pp. 1–6.
[17] C. Wu and B. Li, "Echelon: Peer-to-peer network diagnosis with network coding," in *Proc. IEEE Int. Workshop Quality Service (IWQoS)*, Jun. 2006, pp. 20–29.
[18] Y. H. Chu, S. G. Rao, S. Seshan, and H. Zhang, "A case for end system multicast," *IEEE J. Sel. Areas Commun., Special Issue on Networking Support for Multicast*, vol. 20, no. 8, pp. 1456–1471, Oct. 2002.
[19] S. Jaggi, P. Sanders, P. A. Chou, M. Effros, S. Egner, K. Jain et al., "Polynomial time algorithms for multicast network code construction," *IEEE Trans. Inf. Theory*, vol. 51, no. 6, pp. 1973–1982, Jun. 2005.
[20] C. Gkantsidis and P. R. Rodriguez, "Network coding for large scale content distribution," *IEEE INFOCOM 2005*, Miami, FL, USA, Mar. 2005.
[21] M. Yang and Y. Yang, "An efficient hybrid peer-to-peer system for distributed data sharing," *IEEE Trans. Comput.*, vol. 59, no. 9, pp. 1158–1171, Sep. 2010.
[22] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," in *Proc. ACM SIGCOMM*, 2001, pp. 149–160.
[23] (2003). Gnutella Protocol Development, the gnutella v0.6 protocol [Online]. Available: http://rfc-gnutella.sourceforge.net/developer/index.html
[24] S. Ratnasamy, M. Handley, R. M. Karp, and S. Shenker, "Topologically-aware overlay construction and server selection," *IEEE INFOCOM'02*, New York, NY, USA, Jun. 2002.
[25] GT ITM: Georgia Tech Internetwork Topology Models. (2009) [Online]. Available: http://www.cc.gatech.edu/projects/gtitm/
[26] M. Yang and Y. Yang, "A hypergraph approach to linear network coding in multicast networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 7, pp. 968–982, Jul. 2010.
[27] Y. Yang, J. Wang, and M. Yang, "A service-centric multicast architecture and routing protocol," *IEEE Trans. Parallel Distrib. Syst.*, vol. 19, no. 1, pp. 35–51, Jan. 2008.
[28] X. Deng, Y. Yang, and S. Hong, "A flexible platform for hardware-aware network experiments and a case study on wireless network coding," *IEEE/ACM Trans. Netw.*, vol. 21, no. 1, pp. 149–161, Feb. 2013.
[29 Y. Yang and J. Wang, "A new self-routing multicast network," *IEEE Trans. Parallel Distrib. Syst.*, vol. 10, no. 12, pp. 1299–1316, Dec. 1999.



Min Yang received the BEng and MS degrees in computer science from Beijing University of Posts and Telecommuncations, China and Tsinghua University, Beijing, China, in 1999 and 2002, respectively, and the PhD degree in electrical engineering from Stony Brook University, New York, in 2009. Currently, he is working in Google Inc. on data center networks. His research interests include multicast routing, data center networks, network coding, and peer-to-peer networks.

Yuanyuan Yang received BEng and MS degrees in computer science and engineering from Tsinghua University, Beijing, China, and MSE and PhD degrees in computer science from Johns Hopkins University, Baltimore, Maryland. She is a professor of computer engineering and computer science at Stony Brook University, New York, and the director of Communications and Devices Division, New York State Center of Excellence in Wireless and Information Technology. Her research interests include wireless networks, optical networks, data center networks, high-speed networks, and parallel and distributed computing systems. She have published over 280 papers in major journals and refereed conference proceedings and holds 7 US patents in these areas. Currently, she is an associate editor-in-Chief for the *IEEE Transactions on Computers* and an associate editor for the *Journal of Parallel and Distributed Computing*. She has served as an associate editor for the *IEEE Transactions on Computers* and *IEEE Transactions on Parallel and Distributed Systems*.