The 28th International Conference on Distributed Computing Systems

Peer-to-peer File Sharing Based on Network Coding

Min Yang and Yuanyuan Yang
Department of Electrical & Computer Engineering
State University of New York, Stony Brook, NY 11794, USA

***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 is 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. The 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 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 Narada which does not employ network coding. In addition, it achieves good reliability and robustness to link failure or churn.**

Keywords: Network coding, peer-to-peer networks, web-based applications, file sharing, multicast.

I. 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). Due to lack of multicast support over the Internet, these applications suffer from the scalability problem, which limits the number of receivers involved. *Peer-to-peer* is a new technology that can implement multicast at the application layer. By incorporating peer-to-peer technology into web-based applications, the scalability problem can be eliminated. In this paper, we consider applying peer-to-peer technology to file sharing services, for example, a web server or a file server holds a file that is requested by multiple clients (receivers). The topology of such an application can be represented by a multicast network. When peer-to-peer technology is used, the receivers help forward the file for each other besides receiving the file.

Network coding is another promising technology that is employed to improve system throughput and reliability. Multicast is a typical application which can benefit from network coding. In fact, Ahlswede et al. [1] has generally proved that if network coding is allowed in network nodes, then multicast capacity (i.e., the minimum of the max-flows between the source and every receiver, which is considered as the maximum rate the source can send messages to the receivers) can be achieved, in other words, network capacity can be fully utilized in multicast.

A multicast network can be modeled by a directed graph where each node has multiple incoming edges and outgoing edges. Then 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, then 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. *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. pro-

The research was supported in part by the U.S. National Science Foundation under grant numbers CCR-0207999 and CCF-0744234.

1063-6927/08 $25.00 © 2008 IEEE
DOI 10.1109/ICDCS.2008.52

IEEE computer society

posed 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 [14] which 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 [15], 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.

In this paper, we aim at providing an efficient and reliable file sharing service over peer-to-peer networks by utilizing network coding. We call it *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 [12] 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 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 [15], 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 topology awareness.

The rest of the paper is organized as follows. In Section II, 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 III, we describe how PPFEED works and discuss its good fault tolerance ability. In Section IV, we extend our approach to handle topology mismatch. We give our simulation results and compare the performance of the scheme in Section V. Finally, Section VI gives the concluding remark and future work.

II. Deterministic Linear Coding over Combination Networks

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. 1 shows a combination network for $n = 4$ and $k = 2$, usually denoted as a C_4^2 combination network.

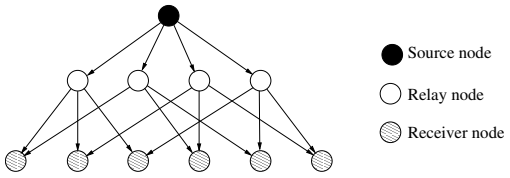

Fig. 1. Combination network C_4^2.

Combination networks have good performance with respect to network coding gain. It was proved [12] that network coding gain (i.e., the ratio of throughput with network coding to that without network coding) is unbounded when both n and k approach infinity. 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. Before we go into

details of the 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 $k = n/2$ [12]. On the other hand, k should not be too large because a large k will increase the link stress and overhead. Hence, 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.

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. Thus 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. Therefore, 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 give the rules for our linear network coding scheme construction for C_n^2 and C_n^3 combination networks as follows.

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.
2. The linear network coding scheme for C_n^3 combination network is to assign vectors $(1, \alpha_1, \alpha_1^2 \mod q)$, $(1, \alpha_2, \alpha_2^2 \mod q), \ldots, (1, \alpha_n, \alpha_n^2 \mod q)$, 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 scheme for combination networks is valid.

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

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

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

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, α, $\alpha^2 \mod q$), (1, β, $\beta^2 \mod q$) and (1, γ, $\gamma^2 \mod q$), where $\gamma > \beta > \alpha$, the determinant of matrix $\begin{pmatrix} 1 & \alpha & \alpha^2 \\ 1 & \beta & \beta^2 \\ 1 & \gamma & \gamma^2 \end{pmatrix}$ is

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

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

III. 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 that can improve the reliability and resilience of the system.

A. 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 in the previous section. Thus any k messages out of the n messages can be used to decode the original file. 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. The entire overlay network can be considered as a union of multiple combination networks (or subgraphs of combination networks). Different combination networks may overlap. A peer that is a relay node in a combination network may be a receiver node in another combination network. Each peer is a receiver node in one combination network. The server is always the source node in any combination network.

Fig. 2 shows an example of the overlay network constructed by PPFEED for $n = 3$ and $k = 2$. There are three groups of peers 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.

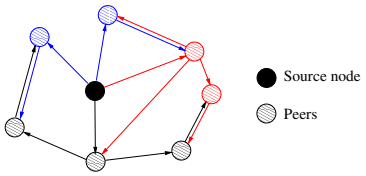

Fig. 2. An example overlay network constructed by PPFEED.

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. Here we adopt some loose rules similar to the unstructured peer-to-peer network Gnutella [17]: for a newly joined peer, it is responded by 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 is no constraint on the overlay topology 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 it performs a local topology adjustment to find the $k-1$ peers such that the average latency between the peer and $k-1$ peers is minimized.

Next, we discuss how PPFEED works in more detail.

B. 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 keeps track of the number of peers in each group and maintains a partial list of existing peers in the group and their IP addresses. The purpose of maintaining a partial list instead of a full list is to achieve a balance between scalability and efficiency. On one hand, when the network size is large, it is resource-consuming to maintain a full list of peers in the server. On the other hand, if the list is too short, it may cause much longer latency for new peers to join when the peers in the list crash. Note that although the server is responsible for bootstrapping the peers, it will not be the bottleneck of the system since once the 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 such that the numbers of peers in different groups are balanced. Then the server sends the list of peers of that group to the joining peer and updates the number of peers in that group.

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 that are in different groups are called inter-neighbors. The new peer asks one of its intra-neighbors, which is picked randomly, to provide a list of its inter-neighbors. 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.

As the overlay topology is formed by always connecting the new peers to a random list of existing peers in the network, the overlay links between peers may not be good with respect to latency or link stress. To alleviate the performance degradation, we optimize the overlay topology by a process called *local topology adjustment*. The basic idea is to replace the direct neighbors by peers with better performance through local search. Each peer sends periodic QUERY messages with a search radius of 2. Once it finds a peer with shorter latency than the current neighbor, it switches its neighbor to the new peer. The local topology adjustment is done periodically such that the peer can find the best neighbor gradually and the overlap topology can adapt to the dynamic joins/leaves.

C. 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 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 acts as one of the intra-neighbors of the peer temporarily 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 done. 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 periodically and maintain a HELLO timer for each neighbor. Receiving a HELLO message triggers a reset of the corresponding HELLO timer. The neighbors detect the abruptly leaving by the timeout of the HELLO timer. Similarly, intra-neighbors initiate the join process after detecting the sudden leave. 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 the encoding vector for the group corresponding to the leaving peer is $(1,\ \alpha)$ (or $(1,\ \alpha,\ \alpha^2 \mod q)$), then inter-neighbor corresponding to the group whose encoding vector is $(1,\ (\alpha+1) \mod n)$ (or $(1,\ (\alpha+1) \mod n,\ ((\alpha+1) \mod n)^2 \mod q)$) is chosen.

D. 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 II. 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.

E. Improving Reliability

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

In the previous subsections, every peer is connected to $k-1$ inter-neighbors. These $k-1$ messages 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 an 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 old scheme, the peer fails to decode with probability $P_{failure} = 1-p^{k-1}$, 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. 3 plots the curves of the ratio for different p values when $k=2$ or $k=3$. We can see that when $p=0.9$, the failure probability of the old scheme is about 10 times of the improved scheme.

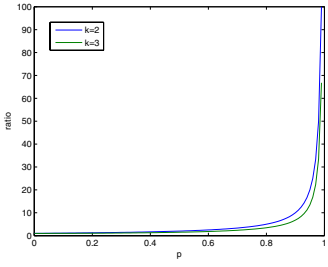

Fig. 3. Failure probability ratio of the old scheme to the improved scheme.

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.

IV. Extension to Support Topology Awareness

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 will greatly increase the *link stress* and degrade the performance. Here link stress is defined as the number of copies of a message transmitted over a certain physical link.

In Section III-B, we mentioned that peers can use local topology adjustment to find peers with shorter latency such that peers can dynamically adjust the local overlay topology to alleviate the mismatch. However, a problem with this method is that its convergence speed is slow and its accuracy is limited. We need a more efficient and accurate method to minimize the mismatch. Here we propose a *topology clustering scheme* by adopting the

idea of binning scheme introduced in [18] to construct a topology aware overlay network.

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 coordinate is sent to the server, then the server assigns the new peer a group ID based on its coordinate. Peers with the same coordinates form a cluster. The heuristic rules of assigning peers to groups are: each cluster has at least k different groups; the peers in the same group should span as few clusters as possible. The first rule is to guarantee that peers can receive enough messages to decode within its cluster. The second rule is to minimize the number of links across clusters. To implement this, the server should keep track of the numbers of different groups in each cluster. After receiving a peer's join request and its coordinate, the server first checks whether the corresponding cluster has k different groups. If yes, the peer is assigned to one of the groups such that the numbers of peers of different groups are as balanced as possible. Otherwise, the peer is assigned to a group which is different from the existing groups in the cluster.

In addition, every two landmark peers should not be too close to each other. 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.

V. Performance Evaluations

In this section, we study the performance of PPFEED through simulations. We compare our scheme with a peer-to-peer multicast system called Narada [13]. Narada first constructs an overlay mesh spanning over all the peers. The overlay mesh is a richer connected graph which satisfies some desirable performance properties. The multicast tree is a spanning tree on top of the mesh and is constructed on demand of the source peer. We choose Narada as the comparison counterpart because it is a representative overlay multicast scheme without network coding so that we can evaluate the benefit network coding brings.

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.

We study the performance of the system in three different configurations:

(i) Baseline configuration. In this configuration, the overlay links are constructed randomly. The file is sent after the overlay network is formed.

(ii) Topology awareness configuration. In this configuration, the overlay links are constructed by taking into consideration of topology mismatch. The file is sent after the overlay network is formed.

(iii) Dynamic peer join/leave configuration. In this configuration, the overlay links are constructed randomly. Peers join and leave the system during the file transmission.

The network topologies are random graphs generated by GT-ITM [19]. 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 $k = 3$.

A. 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.

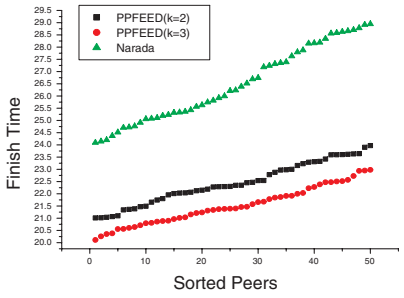

Fig. 4. Finish time of the baseline configuration.

We plot the finish time curves of PPFEED and Narada in Fig. 4. 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. We notice that the finish time of Narada has a larger variance. This is because that 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, our scheme constructs a mesh to distribute the file. As a result, the distance between the highest level peer and the lowest level peer is shortened. 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. 5 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 III-E, 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

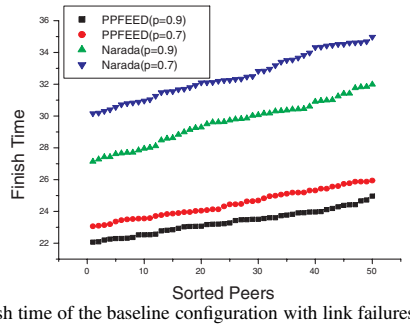

Fig. 5. Finish time of the baseline configuration with link failures.

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 time of PPFEED is much shorter than that of Narada. The gap between them becomes wider when p decreases. Compared to the previous simulation result without link failure probabilities, the finish time of PPFEED increases less than that of Narada.

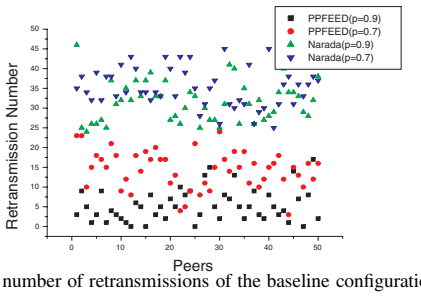

Fig. 6. The number of retransmissions of the baseline configuration with link failures.

The number of retransmissions is shown in Fig. 6. 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 Fig. 5 and Fig. 6 reveal that PPFEED has a good fault tolerance ability.

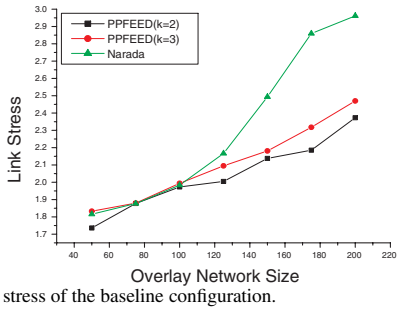

Fig. 7. Link stress of the baseline configuration.

We compare the link stress in Fig. 7. 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 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 (roughly k/n if the overlay network is well balanced, where $1/n$ is for peers within the same group, $(k-1)/n$ is for peers in different groups) of all the peers. 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 reduced group size. Thus the link stress is increased.

B. Topology Awareness Configuration

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

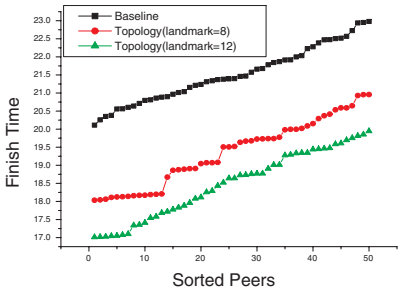

Fig. 8. Finish time of the topology awareness configuration.

Fig. 8 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.

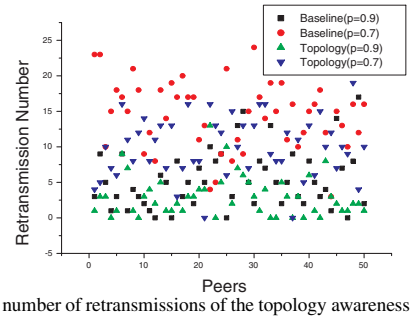

Fig. 9. The number of retransmissions of the topology awareness configuration.

We set the physical link failure probability to $1-p$. Fig. 9 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.

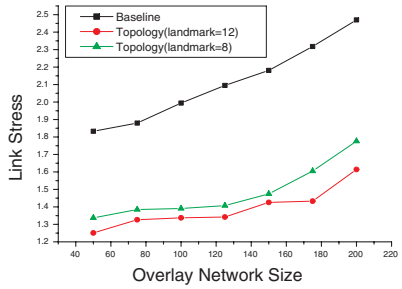

Fig. 10. Link stress of the topology awareness configuration.

One of the biggest advantages of topology clustering is to reduce the link stress. From Fig. 10, 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 is 12. With the increase of the overlay network size, the difference is even bigger. Increasing landmarks makes the link stress smaller.

C. Dynamic Peer Join/Leave Configuration

In this configuration, we allow peers to join and leave the system during the file transmission. Peers join and leave the system in a smooth way. Peers stay in the system to serve other peers even after they finish receiving the file. Peers may leave the system before they finish receiving the file.

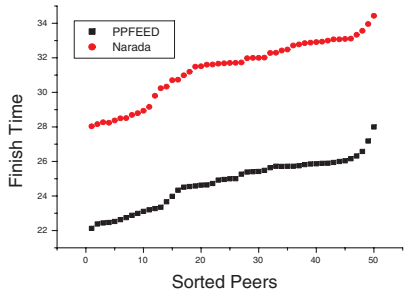

Fig. 11. Finish time of the dynamic peer join/leave configuration.

Fig. 11 shows the finish time comparison between PPFEED and Narada. We can see that the average finish time of both schemes increases compared to the baseline configuration. The peers with different finish times are not evenly distributed as in the baseline configuration. The number of peers with longer finish time increases. Compared to the baseline configuration, the average finish time of Narada increases by about 25%, while PPFEED increases by about 18%, which indicates that PPFEED is more robust under dynamic peer join or leave. We can see that there are some peers with very long finish times. This is due to that the peers providing messages to them leave the system with high probability.

VI. Conclusions

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. Our future work includes how to optimize the system under unstable network status such as congestion.

References

[1] R. Ahlswede, N. Cai, S.-Y. R. Li and R.W. Yeung, "Network information flow," *IEEE Trans. Information Theory*, vol. 46, 2000, pp. 1204-1216.

[2] S.-Y.R. Li, R.W. Yeung and N. Cai, "Linear network coding," *IEEE Trans. Information Theory*, vol. 49, 2003, pp. 371-381.

[3] R. Koetter and M. Medard, "An algebraic approach to network coding," *IEEE/ACM Trans. Networking*, vol. 11, no. 5, 2003, pp. 782-795.

[4] T. Ho, M. Medard, J. Shi, M. Effros and D.R. Karger, "On randomized network coding," *Proc. Annual Allerton Conference on Communication, Control, and Computing*, 2003.

[5] T. Ho, M. Medard, R. Koetter, D. Karger, M. Effros, J. Shi and B. Leong, "A random linear network coding approach to multicast," *IEEE Trans. Information Theory*, vol. 52, 2006, pp. 4413- 4430.

[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," *Proc. IEEE INFOCOM 2005*, Mar. 2005.

[7] D. S. Lun, M. Medard, T. Ho and R. Koetter, "Network coding with a cost criterion," *Proc. 2004 International Symposium on Information Theory and its Applications (ISITA 2004)*, Oct. 2004.

[8] Y. Zhu, B.C. Li and J. Guo, "Multicast with network coding in application-layer overlay networks," *IEEE Journal on Selected Areas in Communication*, Sep. 2004.

[9] A.G. Dimakis, P.B. Godfrey, M.J. Wainwright and K. Ramchandran, "Network coding for distributed storage systems," *Proc. IEEE INFOCOM 2007*, May. 2007.

[10] M. Kim, C.W. Ahn, M. Medard and M. Effros, "On minimizing network coding resources: An evolutionary approach," *Proc. NetCod*, 2006.

[11] K. Bhattad, N. Ratnakar, R. Koetter and K.R. Narayanan, "Minimal network coding for multicast," *Proc. IEEE International Symposium on Information Theory* , 2005.

[12] C.K. Ngai and R.W. Yeung, "Network coding gain of combination networks ," *IEEE Information Theory Workshop*, Oct. 2004, pp. 283-287.

[13] Y.H. Chu, S.G. Rao, S. Seshan and H. Zhang, "A case for end system multicast," *IEEE Journal on Selected Areas in Communication, Special Issue on Networking Support for Multicast*, vol. 20, no. 8, 2002.

[14] S. Jaggi, P. Sanders, P.A. Chou, M. Effros, S. Egner, K. Jain and L. Tolhuizen, "Polynomial time algorithms for multicast network code construction," *IEEE Trans. Information Theory*, vol 51, no. 6, June 2005, pp. 1973-1982.

[15] C. Gkantsidis and P. R. Rodriguez, "Network coding for large scale content distribution," *IEEE INFOCOM 2005*, Miami, FL, March, 2005.

[16] S. Ratnasamy, M. Handley, R. Karp and S. Shenker, "A scalable content-addressable network," *Proc. ACM SIGCOMM*, 2001, pp. 149-160.

[17] Gnutella Protocol Development, the gnutella v0.6 protocol. Available: http://rfc-gnutella.sourceforge.net/developer/index.html. 2003.

[18] S. Ratnasamy, M. Handley, R. M. Karp and S. Shenker, "Topologically-aware overlay construction and server selection," *IEEE INFOCOM 2002*, New York, NY, June, 2002.

[19] http://www.cc.gatech.edu/projects/gtitm/.