PROD: Relayed File Retrieving in Overlay Networks

Zhiyong Xu, Dan Stefanescu, Honggang Zhang
Suffolk University
{zxu, dan, zhang}@mcs.suffolk.edu

Laxmi Bhuyan
University of California, Riverside
bhuyan@cs.ucr.edu

Jizhong Han
Chinese Academy of Science
jzhan@ict.ac.cn

***Abstract*—**

To share and exchange the files among Internet users, Peer-to-Peer (P2P) applications build another layer of overlay networks on top of the Internet Infrastructure. In P2P file sharing systems, a file request takes two steps. First, a routing message is generated by the client (request initiator) and spread to the overlay network. After the process finishes, the location information of the requested file is returned to the client. In the second step, the client establishes direct connection(s) with the peer(s) who store a copy of that file to start the retrieving process.

While numerous research projects have been conducted to design efficient, high-performance routing algorithms, few work concentrated on file retrieving performance. In this paper, we propose a novel and efficient algorithm — PROD to improve the file retrieving performance in DHT based overlay networks. In PROD, when a file or a portion of a file is transferred from a source peer to the client, instead of creating just one direct link between these two peers, we build an application level connection chain. Along the chain, multiple network links are established. Each intermediate peer on this chain uses a store-and-forward mechanism for the data transfer. PROD also introduces a novel topological based strategy to choose these peers and guarantees the transmission delay of each intermediate link is much lower than the direct link. We conducted extensive simulation experiments and the results shown that PROD can greatly reduce the transfer time per file in DHT base P2P systems.

I. Introduction

In recent years, Internet experienced fast growing usage of Peer-to-Peer (P2P) applications such as Napster, Gnutella, Kazaa, Edonkey and BitTorrent, [1], [2], [3], [4], [5] etc. Unlike Client/Server (C/S) architecture, in P2P systems, there's no strict separation between clients and servers. Every peer gets involved in the business and takes a portion of system tasks. Due to the fascinating characteristics such as decentralized control, self-organization, fault tolerance and load balancing, P2P systems are very attractive for certain applications, such as file sharing, online gaming and online streaming services. Currently, P2P applications are the dominant file sharing and swapping tools among Internet users. They are No. 1 bandwidth consumer on the Internet, the data transferred on these applications counted for more than half of the total Internet traffic. However, as a relatively new architecture, many open questions such as the routing algorithm, the retrieving mechanism, load balancing strategy, cache and storage management, consistency control, fault tolerance and security, etc. need further investigation. In the past several years, numerous research papers have been published to address these issues.

In P2P systems, if a peer wants to download a file (We use the client to denote this peer hereafter), two steps are needed. First, in order to search the file location information, it has to initiate a routing process to find out the peer which keeps the information about the peers who have a copy of the requested file. According to the methodologies used in the routing process, P2P applications can be divided into two categories: unstructured P2P systems and structured P2P Systems. In unstructured P2P systems, a flood-based routing algorithm (e.g. Gnutella) has to be executed or a central facility has to be contacted (e.g. Napster). For the later approach, the hotspot and single point of failure problems in C/S architecture still exist. If the central facility fails, the whole system becomes out of service. Such a structure greatly reduces the benefits of using P2P model and can not be considered as the pure P2P system. On the other hand, a flood-based routing algorithm will inject a large amount of unnecessary routing messages in the system and waste the precious Internet bandwidth. Thus, these architectures do not work well for large-scale P2P applications.

Structured P2P systems introduced well-organized routing data structures and fully distributed management strategy to solve the above problems. Distributed Hash Table (DHT) based algorithms are the dominant mechanism in this category. DHT systems such as Chord [6] [7], Pastry [8], Tapestry [9] and CAN [10] attracted great attentions in research community. They also got attention from the real world applications. For example, Bit Torrent is now using DHT data structure in its file sharing protocol. All these systems use the similar strategy: each peer is given a unique identifier (peerid) and each file is also associated with a key (fileid). Both ids are generated on a large name space (2^n) using a collision-free hash function [11]. Routing and file location information are maintained in hash tables and distributed on all the peers. Each peer only stores a small portion which determined by the routing mechanism.

In DHT systems, a routing request is accomplished with the collaboration among a group of peers. In each routing hop, the message is sent to a peer whose peerid is numerically closer to the requested key (fileid). This process continues until finally, the message arrives the peer whose peerid is the numerically closest to the key. This peer is the destination peer of the routing process which stores the location information of the re-

978-1-4244-1694-3/08/$25.00 ©2008 IEEE

quested file. It sends this information back to the client. With such a well designed strategy, a routing procedure is guaranteed to be finished within a small number of routing hops (normally $\log(N)$, N is the total number of system peers). Thus, those DHT algorithms can greatly improves the routing efficiency in large-scale P2P applications.

In realty, the efficiency of the second step- file retrieving procedure is even more important because it is the actual data transmission from which clients get what they really want. Furthermore, the routing process only needs to execute once, while a contiguous retrieving connection is needed to download the entire file. However, in both unstructured and structured P2P systems, the file retrieving procedure does not receive enough attention. To accelerate the processing, in most P2P applications, the client sends the file downloading requests to the set of source peers (the peer who stores the copy of the file, we use this name hereafter), a network connection is established between the client and each source peer. If a source peer is topologically far away from the client, such a link has to pass through multiple network and routers. From time to time, the available bandwidth will be greatly affected by the network traffic along the route. The end client is very likely facing long access delay and get very poor throughput. Thus, it may take hours or even days to download a file (if the file is big and the number of source peers is small). A fast, efficient file retrieving algorithm to achieve high throughput per connection (between the source peer and the client) is in great need. Unfortunately, few efforts have been conducted on this issue.

In this paper, we attempt to address the retrieving problem in DHT based overlay networks. We develop a novel and efficient retrieving algorithm — PROD (Peer Relay on file Download) for this purpose. PROD can significantly speed up the file retrieving procedure and improve the system throughput. In our algorithm, instead of setting up a direct network connection between a source peer and the client, we break the connection and create multiple network connections by introducing multiple intermediate relay peers. Along this chain, each relay peer uses a store-and-forward strategy to assist the retrieving process. It receives data packages from the upstream, and forwards it to the next peer in the downstream. Clearly, the actual data transmission delay is determined by the largest delay in these intermediate connections. We design an efficient peer selection mechanism in which the peers' topological information is taken into account. In most scenarios, our strategy can guarantee the access delay of any intermediate connection in this chain is much smaller than the original direct link between the source peer and the client. Thus, PROD is able to achieve much higher throughput.

PROD algorithm considers to improve the throughput of a single source peer-client connection only. Thus, it is orthogonal to multiple source peer-client connection mechanisms used in today's P2P applications. PROD can be applied to each source peer-client connection to further increase the file retrieving throughput.

We organize the rest of the paper as follows: In Section II, we briefly review the fundamental concepts in DHT based overlay networks and describe the routing/retrieving problems. In Section III and IV, we present the motivation of our algorithm, system overview, and the relay peer selection algorithm. In Section V and VI, we conduct the trace based simulation experiments to evaluate and compare the performance of PROD with current DHT algorithms. In Section VII, we discuss the related works. Finally, in Section VIII, we conclude the paper and give the future work.

II. DHT Based Overlay Networks

Our target is to improve the retrieving performance in structured P2P systems. In this section, we briefly review the fundamental techniques in DHT systems, and then we describe the routing/retrieving problem. For description purpose, we choose Chord as the baseline DHT algorithm, and build our solution on top of Chord. However, our scheme can also be applied to other DHT systems such as Pastry, Tapestry, and CAN, etc.

A. Chord Algorithm

In Chord, to uniquely identify a peer, consistent hashing [12] is used. An n-bit identifier (peerid) is assigned to each peer on the circular name space [0, 2^n]. A collision-free algorithm such as SHA-1 [13] is used to generate identifies to avoid the possible conflict problem. The peerid represents a peer's numerical position on the name space. Each file is also assigned a fileid with the same algorithm which reflects the file's location on the name space as well. Chord uses *finger tables* to store the information of other peers. On any Peer *P*, a finger table has (at most) n entries, with the i^{th} entry contains the peerid and IP address information of the first peer, S, that succeeds P by at least 2^{i-1} on the name space. It is denoted as P.finger[i].node. In Chord, when a peer joins the system, its finger table is created with all the peers are used as candidates. The peers who satisfy the numerical requirements are selected. Clearly, with this approach, only the numerical characteristics are considered on the finger table construction. Though not specified directly, Chord has another table: *file location table* on each peer to store the location information of the files whose fileids are numerically closest to this peer's peerid. An entry in this table has a file name, its fileid, and the peerid and IP addresses of a peer who has a copy of this file.

Figure 1 shows a sample finger table and a file location table of the peer (peerid: 121). Here, if Peer 121 sends a routing request for a file with the fileid 168 (key), the first routing hop goes to Peer 158 who is responsible for the name space interval [153, 185). Peer 158 searches its own finger table, and if another peer whose peerid is numerically closer to the key exists, Peer 158 forwards the request to that peer. This process continues until eventually, the request arrives the peer whose peerid is the numerically closest to the key than all the other peers. No further routing hops are needed. The location information of File 168 will be returned to Peer 121. It can retrieve the file after receiving this message. For another example, a request for a file with the fileid 102, 107 or 121 from any peers will reach Peer 121 after several hops, and it will check the file location table and send the corresponding location information back to the client. More details of Chord protocol can be found in [6].

B. Routing and Retrieving Problems

According to the previous description, in Chord, the routing data structure records the peers' numerical characteristics only, the network topological information is not incorporated into the routing algorithm. It is very likely that a routing hop is taken between two peers which are topologically separated. For example, as shown in Figure 2, Client A in Boston, MA may traverse several distant peers in Europe, Asia and Africa before its routing request reaches the destination: Client E in Worcester, MA. In this extreme scenario, the resulting routing latency may be tens of times higher than the actual topological distance between Client A and E. However, there's no way to detect and fix this problem in Chord.

Finger Table

start	intervals	Successor
122	[122,123)	124
123	[123,125)	124
125	[125,129)	131
129	[129,137)	131
137	[137,153)	139
153	[153,185)	158
185	[185,249)	192
249	[249,121)	253

File Location Table

Name	Fileid	Nodeid	NodeIP
File1	102	215	X.X.X.X
File2	107	131	X.X.X.X
File2	107	69	X.X.X.X
File2	107	237	X.X.X.X
⋮	⋮	⋮	⋮
FileK	121	214	X.X.X.X

Fig. 1. Sample finger table and file location table for the Peer 121 on name space [0, 256)

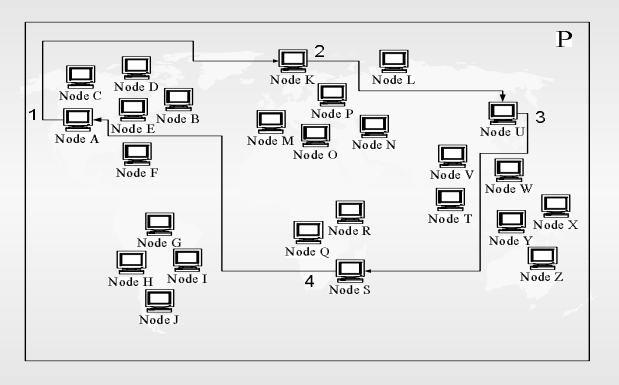

Fig. 2. A sample routing procedure in Chord, the routing request is denoted by arrows. The number of routing hops is 4.

The reason for the routing problem is: we generate the peerid for each peer using SHA-1 algorithm which can not reflect the peer's topological characteristic. A mismatching between the system logical organization and the peers' topological distribution occurs. For example, two peers with the adjacent peerids could be located in different continents, while two peers which are topologically close to each other are very likely have separated numerical peerids. To achieve better performance, we have to put topological information into account when we make the routing decisions.

Furthermore, Chord also has the retrieving problem: After the routing process finishes, the client was acknowledged of the information about the peer or the group of peers that store the requested file. It contacts one or several of them to start download process. A single direct retrieving connection is created between the client and a source peer. The bandwidth available between these two peers is limited by the round trip propagation delay [14]. If two peers are topologically far away from each other, the client will experience long access delay. It may take hours or even days to download the entire file if the file is big. In this stage, no other peers can get involved in this transaction, even if many of them are not busy and could offer some helps. This problem is more serious for the files with small number of copies. This is because the client might have no choice but to download the file from a single remote peer.

III. System Overview

The above routing and retrieving problems exist in all DHT algorithms. They reduce the system throughput significantly. To address these issues, we have to build a bridge between the logical DHT data structures and the underlying physical network. In this section, we describe our solution.

A. Motivation

Recent studies shown that the performance of end-to-end long-haul transmission degrades over lossy links [15], [16]. This phenomenon seriously hurts the overall performance in large-scale distributed systems, such as P2P overlay networks. We propose the Peer Relay On file Downloading — PROD: A novel, fast and efficient algorithm to boost the file retrieving performance in DHT based P2P system. Our approach is motivated by the following observations:

- First, today's personal computer is very powerful, and it can take some responsibilities previously taken by the servers. This is the motivation of P2P overlay networks. However, current DHT algorithms did not fully utilize these resources. The majority of peers are not busy most of the time. For example, under the current service scenarios, for a file retrieving procedure, a direct data transmission connection is established on the source peer and the client only. The idle resources on some other peers can not be utilized to speed up the process. A new strategy is needed to take advantage of these resources.
- Second, in general, retrieving a file from a nearby peer is much fast than retrieving from a remote peer. The reason is the end-to-end link between two close-by peers always has higher bandwidth and lower access latency than the link between two remote peers. In case there's no adjacent peer stores the requested file, if we break a long end-to-end link into several short links, the performance can also be improved.
- Finally, in order to find out the set of relay peers which can be used to generate short links, a relay peer searching algorithm has to be designed. In HIERAS [17], we take the peers' topological information into consideration, and created a hierarchical DHT based routing algorithm. It solved the routing problem in Chord. In PROD, we believe the topological information can also be used to improve the retrieving performance, and it can be used to select relay peers.

Suppose in an overlay network, we have N peers distributed all around the world. Assume Peer P_c finished a routing procedure. Now, it starts a retrieving procedure to download a file

from Peer P_s (who has a copy of that file or a portion of that file). Under current DHT algorithms, a direct end-to-end connection is created. While in PROD, we generate a connection chain. Figure 3 compares a normal file retrieving procedure used in Chord and PROD. As shown in the figure, a single connection between the client and the source peer is created using Chord algorithm (noted as route R1). With PROD algorithm, our relay connections are created among peers V_1, V_2,and V_3, (noted as route R2). For each peer in route R2, it stores the data package in the reserved buffers upon receipt, and it delivers the packages to the downstream peer. This process continues until the data packages arrive the final peer (the client). Clearly, in route R2, the propagation delay of the entire chain is determined by the slowest among four relay connections.

To achieve the satisfactory performance, picking the right intermediate relay peers is of great importance. If we can guarantee the propagation delays on all the four relay connections are much shorter than the delay on the direct link, we can achieve higher throughput. Thus, these relay peers should be chosen as they are approximately distributed along the direction of the physical link from P_c to P_s. For example, suppose the transmission delay d_o (two way propagation delay, plus possible queuing delays due to congestions) on the direct link between P_c and P_s is 100ms, and for i = 0, 1, 2, 3, the transmission delays d_is between V_is and V_{i+1}s are 30, 35, 40, 30ms, respectively. Thus, the maximum package delay on route R2 is only 40ms. If we assume there's no package lose (If it is not 0, we still can get the same conclusion, the analysis can be found in [14]). The throughput of route R2 (T_{R2}) can be approximately characterized by the following formula [18],

$$T(R2) = \frac{C}{MAX(d_i), 0 \leq i \leq 3} \quad (1)$$

While in route R1, the throughput T(R1) is:

$$T(R1) = \frac{C}{d_o} \quad (2)$$

C is a constant and used as a metrics to describe the bandwidth characteristic of the underlying physical network. Since the maximum propagation delay on route R2 is 40ms, PROD algorithm can achieve great performance improvement. The throughput could be improved more than doubled than the direct link. However, if we failed to select the relay peers appropriately, (let's say one of the propagation delays is 85ms, or even higher than 100ms), then few benefits we can obtain, or PROD may even have lower throughput than the original route R1.

To make our approach succeed, we have to design an efficient relay peer searching algorithm which can find out the right relay peers, and guarantee the link latency of every relay connection is much shorter than the direct link between the source peer and the client. In our algorithm, the topological information of peers is used for this purpose. For each intermediate connection, the peers on both ends are carefully chosen that they are relatively topologically close to each other.

B. Distributed Binning Scheme

In order to solve the above problem, our algorithm should be able to estimate the relative topological distance among peers. Thus, we have to design a schema which can figure out the approximate topological distribution of the peers. A simple mechanism to do this is the distributed binning scheme proposed by Ratnasamy and Shenker [19]. It used the link latency as the metric to measure the distance between two peers. In this scheme, a well-known set of machines are chosen as the landmark nodes, and they are well selected and evenly distributed in the system to ensure accuracy of the measurement results. If *k* landmark nodes L1, L2, ... , Lk are picked, when a peer joins the system, it measures and records the latencies to these nodes in the order of 1, 2, ..., k. Then we can generate an ordering link latency information (denote as order information hereafter) for this newly joined peer. The range of the possible latencies between each [peer, landmark node] pair can be divided into different levels. A set of numbers $Z_0, Z_1, ...Z_t$ can be used, and t+1 zones will be defined. For example, as shown in Figure I, we defined 3 numbers: 20, 40, 60ms. Then, four levels are created: level 0 represents the link latencies within [0, 20ms], level 1 for the latencies within (20, 40ms], level 2 for the latencies between (40, 60ms], and level 3 for the latencies greater than 60ms. Thus, we can use a single digit value such as 0, 1, 2 or 3 to represent the relative distance between a peer and a landmark node. For k landmark nodes, on each peer, we can generate k digits to represent the order information. This information can be viewed as the approximate topological position of a peer in a k-dimensional space. For any two peers, the more number of digits are in common (which means they have similar link latencies to more landmark nodes), the topologically closer these two peers are. The detailed information about the distributed binning scheme can be seen in [19].

We use the order information as part of the peer identification information. For example, Peer 121:1012 represents the peer with peerid 121. Four landmark nodes (L1, L2, L3 and L4) are used in the system, and the link latencies from this peer to each of the landmark nodes are within zones (20, 40], [0,20], (20,40] and (40, 60], respectively.

TABLE I

SAMPLE PEERS WITH 4 LANDMARK NODES

Peer	Dist -L1	Dist -L2	Dist -L3	Dist -L4	Order
N1	25ms	5ms	30ms	100ms	1013
N2	40ms	18ms	12ms	200ms	1003
N3	100ms	180ms	5ms	10ms	3300
N4	160ms	220ms	8ms	20ms	3300
N5	45ms	10ms	100ms	5ms	2030
N6	20ms	140ms	50ms	40ms	0321

Table I shows 6 sample peers N1, N2, N3, N4, N5 and N6. The order information is created according to the measured latencies to the 4 landmark nodes L1, L2, L3 and L4. For example, Peer N1's landmark order is 1013. Peers N3 and N4 have the same ordering information: 3300. According to the ordering information of these peers, we can estimate that Peers N3 and N4 are topologically close while N3 and N1 are topologically apart. Distributed Binning Scheme can only reflect the approximate topological information, it is not very accurate. However, from our simulation results, we found it is good enough to achieve significant performance improvement.

C. Hierarchical Architecture

We introduce the hierarchical architecture to represent the topological information. We define a P2P *circle* as a collection of peers with associated routing and file location data structures. A P2P circle is a self-organized and relatively independent unit, the members in a P2P circle are equally important and take the

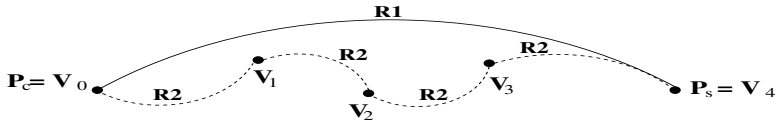

Fig. 3. Comparison of the connection model

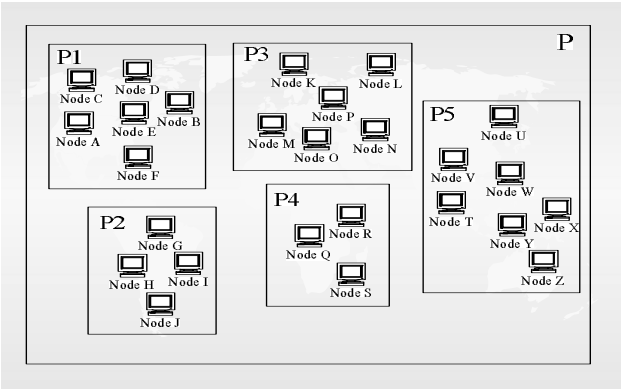

Fig. 4. A Two-layer Hierarchical System, P is the layer-1 circle, P1, P2, P3, P4 and P5 are layer-2 circles.

equal responsibility for the workloads within this circle. In current DHT systems, there's only one P2P circle exist, it contains all the peers. In our system, we create a hierarchical P2P infrastructure: Besides the biggest circle which consists of all the peers, many other P2P circles in different layers which contain different number of peers are generated as well. These circles are created in such a mechanism: the lower the layer, the closer the peers in this circle. Thus, the average link latency between two peers in a lower layer circle is much smaller than the peers in a higher layer circle. We define the number of layers as the *hierarchy depth*. In a *m*-depth P2P system, each peer belongs to m P2P circles with one in each layer. Clearly, the lowest layer circles consist of the set of peers which are the topologically closest to each other.

A simple illustration of a two-layer hierarchical P2P organization is shown in Figure 4. P is the Layer-1 (biggest) circle which contains all the peers. This global layer circle has five layer-2 circles: P1, P2, P3, P4 and P5. Each peer in the system belongs to the layer-1 circle P and one of the five layer-2 circles. For example, Peer A is a member of circle P1 and Peer Z is a member of circle P5. At the same time, both of them are also members of the global circle P. Topologically adjacent peers are grouped in the same layer-2 circle. For example, peers A, B, C, D, E and F are located on the same continent, they are grouped together in layer-2 circle P1.

In our system, we use the peers' order information to generate circles. For example, if we want to create a two-layer system, we can group peers which have the same order information into one lower layer P2P circle. Thus, Peer 121: 0123 is in a circle with all the members which have the same ordering information – 0123. We denote "0123" as the *circleid* for this circle. Peer 121: 0123 and Peer 047: 0123 belongs to the same lower layer circle, but it does not belong to the same lower circle as Peer 221: 0110. However, if a three-layer system is needed, we can create an intermediate layer by taking the ordering information of landmark nodes L1 and L2 only. Peers who have the ordering information — 01XX belong to the same intermediate layer circle, although they might not in the same lowest layer circle. For example, Peer 121: 0123 and 221: 0110 are in the same intermediate layer circle but not in the same lowest layer one.

We can use a well-known set of landmark machines spread across the Internet as [19]. In case of a landmark node failure, newly added nodes are binned using the surviving landmarks while previous binned nodes only need to drop the failed landmark(s) from their order information. In this case, performance degrades. To relieve the landmark nodes failure problem, we can use multiple geographically closest nodes as one logical landmark node [19].

The number of landmark nodes has great impact on the system performance. As we increases the number of landmark nodes, the number of lower layer P2P circles will increase and the average link latency in each circle will reduce. System has more accurate information about a peer's topological location and can add it to a circle with closer neighbors. On the other hand, as the number of landmark nodes increases, the number of circles also increases, thus the average number of peers within each circle will decrease.

In HIERAS [17], we proposed multi-layer routing algorithms, the main idea is to replace a large amount of long-latency routing hops which occurred in the global layer (in Chord) with the low-latency hops in the lower layers (in HIERAS), and it greatly reduced the routing overhead. In PROD, we inherit the algorithm and solve the routing problem. The details of HIERAS algorithm can be found in [17].

Although HIERAS can successfully improve the routing performance, it can not solve the retrieving problem. As traditional DHT algorithms, in HIERAS, after the routing process, the client has to download the file from a peer which stores a copy no matter how far it is. New solution is needed to address the retrieving problem. We extend the idea in HIERAS and design an efficient retrieving mechanism using the topological information.

IV. File Retrieving Algorithm

In this section, we first describe the data structures to be used in PROD, then we present the relay peer searching algorithm. We also discuss the failure tolerance issue related to our algorithm.

A. Data Structures

In our system, the ordering information is stored as part of the peer's characteristics. Since we created a multiple layered system, to support the routing and retrieving operations in different layers, we have to modify the structure of the Chord finger table. If we create an m-depth PROD system, for each peer, m-layer tables are needed with one in each layer. Figure 5 shows a two-layer finger table for Peer 121: 0123. In the higher layer finger table, as Chord algorithm, for each entry, the successor is selected from all system peers if it has the smallest peerid in that interval. However, for the lower layer finger table construction, we only pick the peers from the same circle "0123" as Peer 121.

start	intervals	layer 1 successor	layer 2 successor
122	[122,123)	124: 0011	143: 0123
123	[123,125)	124: 0011	143: 0123
125	[125,129)	131: 1121	143: 0123
129	[129,137)	131: 1121	143: 0123
137	[137,153)	139: 0220	143: 0123
153	[153,185)	158: 0123	158: 0123
185	[185,249)	192: 0012	212: 0123
249	[249,121)	253: 0123	253: 0123

Fig. 5. Peer 121: 0123's finger tables in a two-layer Hierarchical system with 4 landmark nodes, name space [0, 256)

Layer 1 File Location Table

Name	Fileid	Nodeid	NodeIP
File1	102	215: 2103	X.X.X.X
File2	107	131: 1120	X.X.X.X
File2	107	69: 0123	X.X.X.X
File2	107	237: 0201	X.X.X.X
⋮	⋮	⋮	⋮
FileK	121	214: 1111	X.X.X.X

Layer 2 File Location Table

Name	Fileid	Nodeid	NodeIP
File2	107	69: 0123	X.X.X.X
File3	118	143: 0123	X.X.X.X
File3	118	212: 0123	X.X.X.X
⋮	⋮	⋮	⋮
FileX	98	158: 0123	X.X.X.X

Fig. 6. The Two-layer file location tables on Peer 121: 0123

We also modify the structure of the file location table. Figure 6 shows the two-layer file location table for Peer 121: 0123. The contents in the higher layer table are the same as Chord. It records the information of the peers which have copies of the corresponding files, no matter which lower layer circles these peers are in. For the lower layer table, it only records the information of the peers within the same layer-2 circle which store the corresponding files. We introduce the topological information into the file location table, thus a client can always choose the closest peer who has a copy to start retrieving.

Besides the data structures inherited from the underlying DHT algorithms, in PROD, we use *landmark table* to maintain landmark nodes information, it simply records the IP addresses of all landmark nodes and list them in order. We omit it here. We use *Circle tables* to maintain the information of P2P circles. The structure of a circle table is shown in Table II. The circlename is defined by the landmark order information such as "0123". The circleid is generated by using the collision-free algorithm on the circlename. A circle table is stored on the peer whose peerid is the numerically closest to its circleid. It records 4 peers inside the circle: the peer with the smallest peerid, the peer with the second smallest peerid, the peer with the largest peerid and the peer with the second largest peerid. A circle table is duplicated on several numerically close peers (these peers can be get easily) for fault tolerance. A peer which stores the circle table periodically checks the status of these nodes. In case of a peer fails or leaves, a new routing procedure is performed to add a new peer into the table. If the system has more layers, the circleids for intermediate circles are generated by using the partial ordering information. For example, for Circle "01XX", 01 will be used to generate the circleid. And the circle table for "01XX" will be stored on the peer whose peerid is numerically closest to "0100". Circle tables are used by the relay peer searching algorithm to find a peer in a particular P2P circle during the searching process of the next intermediate relay peer. The details of its usage are introduced later.

B. Searching Relay Peers

Finding the right relay peers is critical for PROD to achieve optimal performance. We develop a relay peer searching algorithm based on the peers' ordering information. Figure 7 shows the pseudo code of this algorithm. Here, we assume for a client P_c, the routing process is already finished. It has to create a connection chain to the source peer P_s who stores the requested file. Our algorithm is used to find out the set of the intermediate relay peers.

In our algorithm, we first calculate the sum of the digit differences *K* of P_c and P_s's ordering information. Then, we start the relay peer searching process from P_c. If we do not limit the number of relay peers during the searching process, then, each time, we change the ordering information that the sum of the digit difference will reduce by 1 and use it as the circleid which to search for the next relay peer. For example, if the ordering information of the client 123 and the source peer 056 are 0123 and 2101, respectively. Then the sum of digit difference is 6. Starts from client 123, we are looking for the peer information of the circles with the circleids 1123, 0113, or 0122.

In order to find a peer in these circles, a DHT routing process is performed for each of them. For a particular circle, such a routing request will arrive the peer who stores the circle table for that circleid as other normal routing requests. The result of a set of peers who belong to that circle will be returned. We select one peer from all the candidate peers according to a certain requirement (most cases, the peer generates the lowest delay). It will be added to the set of relay peers. Starts from this newly added relay peer, we reduce the sum of the digit differences of the ordering information again, and the searching process continues, until finally, we reach the source peer. A message containing the information of all the relay peers selected will be returned to the client. Then, a connection chain between the client and the source peer is established to start the file retrieving.

The number of intermediate peers is adjustable and can be defined as a parameter. If the maximum number defined is smaller

```
P_c is the client peer
P_s is the source peer which stores the file
L is the number of Landmark nodes
K is the sum of the digit number differences of the ordering information
R is the maximum number of relay peers
S is the set of relay peers. Originally, it is empty

Function Searching_Relay_Peers(P_c,P_s)
{
    O_c = O_c1, O_c2, ...O_cL is the ordering information of P_c
    O_s = O_s1, O_s2, ...O_sL is the ordering information of P_s
    Set S = Ø
    i = 0
    K =sum of the digit number differences of the ordering information
    S_i = O_c
    E_i = O_s
    N = P_c /* the process starts at the client */
    m = ⌈K/R⌉
    while i ≤ R
    {
        if (m¿=K) /* PROD algorithm failed */
            exit(0);
        C,O_n = Choose_next_relay_peer(N, S_i, E_i,m);
        if (such C does not found) /* no circle exist which satisfy the requirement */
        {
            m=m+1; /* loose the condition*/
            continue;
        }
        i = i + 1;
        S = S + C; /* add a new relay peer to S */
        N = C;
        S_i = O_n
    }
    return(S);
}

Function Choose_next_relay_peer(N, O_S, O_E, m)
{
    O_Diff = the sum of digit differences of the circles
    O_S and O_E
    if (O_Diff ≥ 0)
    {
        generate a set of circleids O = O_n1, O_n2, O_nj
        O_ni is generated by changing any m digits in O_S
        and make the sum of digits difference of O_ni and
        O_E equals O_Diff - m
    }
    Y = Ø, /* Y is the set of candidate relay peers */
    for (each circleid O_ni in O)
    {
        Y_i = Chord(O); /* Seaching the circle using Chord Algorithm */
        if (Y_i does not exist) /* no such circle exist */
            continue; /* try another one */
        else
        {
            Y = Y_i → Y
            P = Y_min; /* the peer has the smallest delay to N */
            O_p is the circleid of P
            return(P, O_p)
        }
        return(0,0) /* no candidate circle exist */
}
```

Fig. 7. Pseudo Code of Searching Relay Peers Algorithm

TABLE II
P2P CIRCLE TABLE STRUCTURE

Circleid	Circlename	Peer (Largest Id)	Peer (Second Largest Id)	Peer (Smallest Id)	Peer (Second Smallest Id)

than the sum of the digit differences of the ordering information, we can reduce the sum by 2 or even more each step. In our simulations, we evaluate its effects.

Our algorithm is based on the following assumption, the closer the two peers, the smaller the difference of the sum of digits on the ordering information. However, it has the weighting problem. The values in the ordering information have different weights, the result might not accurate. For example, if we have two peers with ordering information 0XXX and 1XXX (assume the last three digits are the same), the maximal two way propagation delay between these two peers is $2 \times 20 = 40$ms. However, if the ordering information of these two peers are 2XXX and 3XXX, the maximal delay becomes 120ms. To relieve this problem, we can divide the delay into more zones, such as for [0,100]ms , we can create 0,1,...,9 zones to represent the ordering information. Each one is used to represent only 10ms-sized zone.

During the searching process, we might find a circle with a certain circleid does not exist. In this case, we try another candidate circleid. If none of the candidate circles are found, we can change the circleid generation criteria, say, reduce the sum of the digit difference even more, or take out the ordering information of some certain landmark nodes. If none succeed, we claim we can not find a suitable connection chain. In our simulations, we found such situations seldom happen.

We also observe for two peers, if they have similar ordering information for most landmark nodes, it is not likely they have quite different digits in the ordering information for the remaining landmark nodes. For example, two peers with the ordering information "0123" and "0120" are not likely to exist. It means the weighting problem is not as severe as we thought. Furthermore, in our algorithm, we measure the delays for all the possibilities and choose only the peer with the smallest delay as the relay peer. Our simulation experiments shows, although they are not always the best choices, our relay peer searching algorithm is able to find out the suitable set of peers which can boost the retrieving performance significantly.

After determining the set of relay peers, the multi-connection chain can be created easily.

C. Failure Recovery

In PROD, a retrieving process needs the collaboration among a set of relay peers. Though it can reduce the propagation delay, it introduces other issues. For example, the possibility of a peer failure during the file transmission is higher than the direct link. We use the following two mechanisms to reduce the effect:

- For each peer, we record the alive time, and based on that, we can choose peers who are likely to stay longer in the system as relay peers;
- For each relay connection along the chain, both peers keep the IP addresses of some other peers which belong to the same circle as the relay peer on the other side. This information can be obtained easily from the circle table. If a peer failure is detected, an alternative peer can be selected quickly and replace the role of the dead one.
- When we choose the relay peers, we always choose the peer with more resources and few current workload. These peers have more computer resources to be used for the file retrieving process.

Overall, PROD consumes more network resources since it needs a group of relay peers. This problem can be relieved by limit the maximum number of relay peers. Furthermore, since the majority peers are not busy most of time, it is not harmful to take these idle resources. We can also design an adaptive algorithm, in case of the system is not busy, we can increase the number of relay peers. Otherwise, we decrease the number. For example, during the night, few people are using their computers, more relay peers can be used.

V. EXPERIMENT ENVIRONMENT

To evaluate the performance of PROD algorithm, we conducted trace-driven simulations. In this section, we describe the simulation environment.

A. The Network Model

We choose GT-ITM Transit-Stub (TS model) as the primary network topology model. TS is an internetwork topology model proposed by E. Zegura in [20]. It is a two-layer internetwork model, a TS network is composed of interconnected transit and stub domains. Transit domains function more like Internet backbone while stub domains work more like local area networks. TS model reflects the hierarchical character of internetwork and it is more accurate than a random model. In our simulation, the delays of intra-transit domain links, stub-transit links and intra-stub domain links are set to 100, 20 and 5ms respectively (We also use other distributions but our conclusion does not change).

In our simulations, we vary the number of peers from 1000 to 10000. The number of peers in each stub domain varies from 12 to 20 depends on different total number of peers. We choose Chord as the baseline DHT algorithm and we compare the file retrieving performance of Chord with PROD. We use Chord to represent the original algorithm, if multiple copies exist, the source peer for retrieving is selected randomly, Chord-T to represent Chord algorithm which downloads the file from the closest source peer, PROD-R represents PROD algorithm without the limitation on the number of relay peers. PROD-L represents PROD with fixed maximum number of relay peers. In our simulation experiments, unless specified, we use a two-layer hierarchy system with four landmark nodes used to generate circleids for lower layer P2P circles.

B. Workload Traces

Due to the lack of appropriate P2P system traces, we use web proxy logs obtained from the National Laboratory for Applied Network Research (NLANR) as P2P workload traces in our simulation. This method has been used in many other research projects [21] [19]. The trace data we use are collected from eight individual servers between July 8, 2004 and July 14, 2004.

VI. PERFORMANCE EVALUATION

In this section, we present the evaluation results for the simulations we conducted on the above environment.

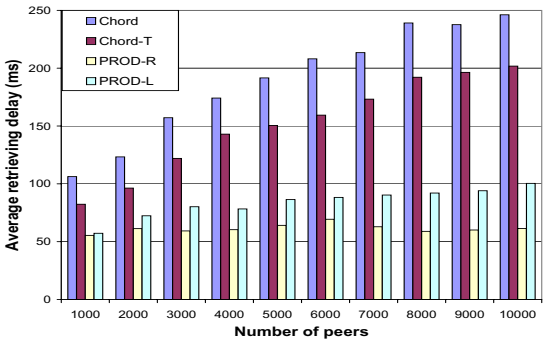

Fig. 8. Propagation Delay Comparison

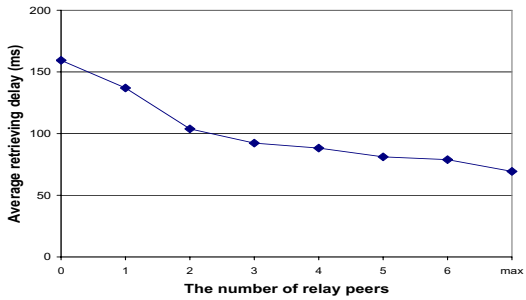

Fig. 9. The Effect of the different number of relay peers in PROD

A. *Average retrieving propagation delay*

Our purpose is to reduce the retrieving access delay in order to improve the file retrieving throughput in DHT overlay networks. In the first experiment, we compare the retrieving performance of PROD with the Chord algorithm. The results are shown in Figure 8.

From the results, we can get the conclusion that among all these algorithms, PROD-R achieves the best performance. The average retrieving delay is only 24.9% to 52.1% of the delay measured in Chord. As the network becomes bigger, the retrieving delay does not change too much. This is because there's no number of relay peers limitation in PROD-R, it can always use more intermediate peers for relay traffic. Thus, PROD-R is perfect for large-scale P2P overlay networks. One deficit of this approach is, as more relay peers introduced, the peer failure problem becomes more severe. PROD-L also provides very good performance. The retrieving delay is only about 40.5% to 58.7% of the delay in original Chord algorithm. However, as the network size increases, due to the limitation on the maximum number of relay peers, the access delay increases. The delay increases is much slower than the original Chord. Thus, for large-scale P2P overlay networks, it is still a good alternative of Chord. For both PROD-R and PROD-L algorithms, the retrieving delay is reduced significantly. If the overlay network is very dynamic, which means the peers join and leave frequently, PROD-L is the best choice.

Obviously, the original Chord algorithm has the worst performance. In Chord-T, we added the topological information into the file location table, the client can always download the file from the peer with the lowest round trip delay, while in Chord, the client only select a source peer randomly. Though it is not as good as PROD, Chord-T can improve the retrieving performance in Chord by 17.4% to 23.5%.

B. *Effects of the maximal number of relay peers*

From the above simulation results, we observer that the maximal number of relay peers allowed in PROD will affect its performance. In relay peer searching algorithm, this number will determine, to generate the circleid the next relay peer belongs to, how many digits we have to change. Then, we also conduct the experiment to evaluate its effect. This simulation is conducted on a 6000-peer network, we vary the maximal number of relay peers from 0 to 6 (PROD-L), and compare it with the result of unlimited relay peers (PROD-R).

Figure 9 shows the result. From the figure, we can see that, with more relay peers introduced, the propagation delay for the file retrieving process reduces. When the number of relay peers is 0 (the original direct link), we got the highest propagation delay. As one relay peer introduced, the delay is reduced by 14.1%. As more relay peers introduced, we can reduce the overall retrieving delay even more. If the number is 6, the delay is reduced by 50.5%. With the unlimited number of relay peers, the propagation delay is cut by 56.5%. We can also observe, as the number of relay peers increased from 0 to 3, the performance improved sharply. But when we keep increase the number of relay peers, although we can still gain some benefit, the performance gain diminished. As we know, more relay peers will increase the risk of peer failure problem, more peers may not be a good choice. Thus, choosing the suitable number of relay peers is of great importance. In this experiment, for a 6000-peer TS network, 4 to 6 relay peers are the optimal configurations.

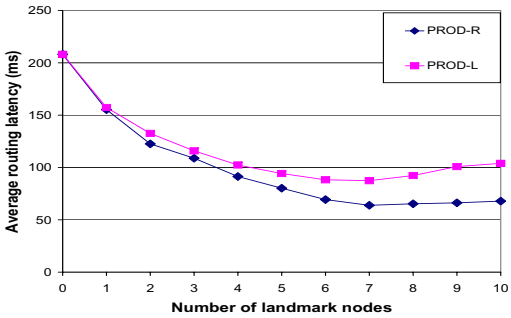

Fig. 10. The Effect of the different number of landmark nodes

C. *Effects of different number of landmark nodes*

Landmark nodes also have great effect on the retrieving performance of our PROD algorithms. With the different number of landmark nodes, the peers' ordering information will have different number of digits. This will affect the relay peer searching algorithm. In the third experiment, we evaluate its effects. The simulation is conducted on a 6000-peer TS network, the number of landmark nodes varies from 0 to 10. For PROD-L, the maximal number of relay peers is 4. The results are shown in Figure 10.

The results show that PROD-R can achieve better performance than PROD-L. When there's no landmark nodes used, no relay peers can be chosen, our system becomes the original Chord system. With one landmark nodes used, both PROD-R

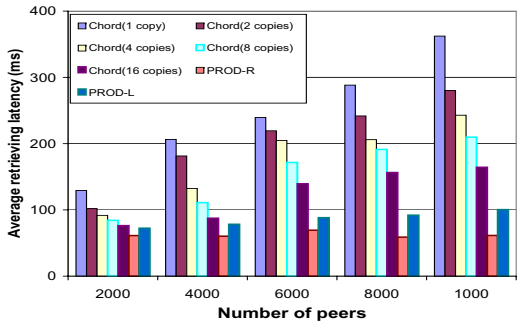

Fig. 11. The Performance Improvement for Different Files

and PROD-L improve the performance greatly. As more landmark numbers introduced, our PROD algorithms continue gain the benefits. However, PROD-R is always better than PROD-L because it has more relay peers. When selecting the circleid for the next relay peer, it changes one digit only, thus it can always pick the best candidate as the next relay peer. PROD-L performance is limited by the maximal number of relay peers. When deciding the next circle, it might have to change multiple digits, thus it can only pick sub-optimal candidates. As we can see from the Figure 10, with more landmark nodes are introduced, the performance gap between PROD-R and PROD-L becomes larger. Here, since the ordering information contains more digits, when choosing the next circle, PROD-L has to increase the number of digits it has to change each time, and this makes the selection of best relay peer candidate more difficult. For PROD-R, this problem does not exist, it can always choose the best candidate by changing only 1 digit in the previous circleid.

Another phenomenon we can observer from Figure 10 is, as more and more landmark nodes are introduced, the performance improvement benefits diminishes or it may have negative effect. We investigate this problem, we found that with the increased number of landmark nodes, the number of lowest layer circles also increases, and the average number of peers in each of these circles reduces. Although the two way delay of a network link within the lowest layer reduces, the delay on the higher layers may increase, and it will cause the whole system performance degradation. Thus, picking the right number of landmark nodes is also important. For this 600-peer network, 6 or 7 is the optimal configuration. But this effect is not serious in PROD. For PROD-L, the propagation delay with 10 landmark nodes is only about 10.5% higher than the best configuration. For PROD-R, this number is 6.3%.

D. Accessing files with different number of copies

In the forth experiment, we compare the performance of PROD and Chord on the files with different number of copies in the system. The results are shown in Figure 11.

As we can see from the result, although for the popular files, the retrieving delay is not very high in Chord-T since it can always download from the nearest peer who has a copy. For those files which have small number of copies in the system, Chord failed to achieve satisfactory performance. Both PROD-L and PROD-R have better performance than Chord. No matter how man copies a file have, our algorithms can always find out an optimal route to relay file retrieving traffic. This means, our algorithm can benefit the clients who wants to access the files which are not frequently used. For those popular files (files with more copies), PROD algorithms still outperform Chord-T. Furthermore, as the network becomes bigger, PROD algorithms can achieve even more performance improvement over Chord. This result proves PROD is a good alternative file retrieving algorithm for current DHT based overlay networks.

VII. Related Works

Due to the deficits of routing architectures used in unstructured P2P systems, a lot of efforts have been made to improve the routing performance in P2P applications. Chord [6] [7], Pastry [8], Tapestry [9] and CAN [10] are DHT based routing algorithms. In these systems, an elegant and efficient routing data structure is generated. However, the negligence of the peers' topological information seriously hurts the routing and retrieval performance. Although DHT algorithms can achieve the optimal routing performance in terms of routing hops, they can not achieve the minimal routing latency.

Furthermore, in these DHT overlay networks, the retrieving problem is not well considered. For both unstructured and structured P2P systems, after locating the requested file, the client will create a direct connection with the peer(s) who store that file and start downloading process. If that peer is topologically far away from the client, the system will suffer long access latency.

Utilizing topological information to boost P2P system performance is a hot topic in recent years [22]. In [17], we propose a new P2P routing algorithm — HIERAS, it keeps scalability property of current DHT algorithms and improves system routing performance by the introduction of hierarchical structure. In HIERAS, we create several lower level P2P circles besides the highest level P2P circle. A P2P circle is a subset of the overall P2P overlay network. We create P2P circles in such a strategy that the average link latency between two peers in lower level circles is much smaller than higher level circles. Routing tasks are first executed in lower level circles before they go up to higher level circles, a large portion of routing hops previously executed in the global P2P circle are now replaced by hops in lower level circles, thus routing overheads can be reduced. However, HIERAS does not attack retrieving problem.

In [23], we utilize the topological information to boost retrieving performance. We introduce a new DHT based P2P algorithm to solve these problems. We use distributed binning scheme to figure out the topological information of each peer, and add this information to the routing and retrieving data structures. Furthermore, by adding the peers' topological characteristics to the file location information, our algorithm can guarantee the closest copy of a requested file to be found. The retrieving problem in DHT algorithms is relieved.

However, both approaches did not use relay transmission to further improve the performance. In [24], Wolf et al. have introduced a transparent TCP acceleration technique that can speedup TCP connections without end-system support. They evaluate that a single acceleration node can speedup TCP connection twofold on lossless and more on lossy links. Multiple acceleration nodes can further increase the attainable throughput. They have discussed how such an acceleration system can be implemented on a network processor. However, their target is at TCP layer, not the application layer.

In [14], Liu et al. explore the flexibility of control at the application layer and propose various application level data relay schemes to largely improve the data throughput by optimally integrating application level routing and transport layer control. They formulate the problems that how to do the TCP pipelines for overlay networks etc. Unfortunately, the authors did not give the detailed instructions on how to build such a route.

In this paper, we also adapt the idea of creating multiple store-forward links to improve P2P system performance. We also design an efficient, low overhead strategy to find the suitable relay peers to build this application level connection chain.

VIII. Conclusions and Future Work

The ultimate goal for P2P overlay networks is to share and exchange files. Though the routing procedure is important, an efficient and fast file retrieving mechanism is more critical. In this paper, we present a novel algorithm — PROD to address the file retrieving issue in DHT based overlay networks. PROD views each peer's topological information as its characteristic, and incorporated this information with the routing and retrieving data structures and algorithms. PROD inherits the hierarchical routing in HIERAS, and extends the idea to solve the retrieving problem. Unlike the current DHT algorithms which only create a single connection between the client and the source peer, a connection chain is established to relay the file download traffic. An efficient relay peer searching algorithm is developed to determine the appropriate set of relay peers by taking the topological information into consideration. We conduct extensive simulation experiments and the results show PROD can greatly improve the file retrieving throughput.

In the future, for comparison purpose, we plan to conduct more simulation experiments by using other network models such as Inept [25] and BRITE [26]. We plan work more on relay peer selection mechanism. We are also considering to do the real implementation and evaluate the efficiency of PROD on Planet Lab [27], the real world testbed. Since most commercial P2P applications belong to the unstructured category, it is also worth considering on how to apply this idea to boost the retrieving performance in unstructured P2P systems.

IX. acknowledgement

This research has been supported by NSF Grant CNS-0509207, 0509440 and National Basic Research Program of China (973 Program) under Grant No. 2004CB318202. We gratefully acknowledges the support of K. C. Wong Education Foundation, Hong Kong. We would also like to thank the anonymous reviewers for their valuable comments.

References

[1] Napster, "http://www.napster.com."
[2] Gnutella, "http://www.gnutella.wego.com."
[3] KaZaA, "http://www.kazaa.com/."
[4] Edonkey, "http://www.edonkey2000.net/."
[5] BitTorrent, "http://www.bittorrent.com/."
[6] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications." Technical Report TR-819, MIT., Mar. 2001.
[7] F. Dabek, E. Brunskill, M. F. Kaashoek, D. Karger, R. Morris, I. Stoica, and H. Balakrishnan, "Building Peer-to-Peer Systems With Chord, a Distributed Lookup Service," in *the 8th IEEE Workshop on Hot Topics in Operating Systems (HotOS), Schoss Elmau, Germany*, pp. 195–206, May 2001.
[8] A. I. T. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, (Heidelberg, Germany), pp. 329–350, Nov. 2001.
[9] B. Zhao, J. Kubiatowicz, and A. Joseph, "Tapestry: An infrastructure for fault-tolerant widearea location and routing." Technical Report UCB/CSD-01-1141, U.C.Berkeley, CA, 2001.
[10] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content addressable network." Technical Report, TR-00-010, U.C.Berkeley, CA, 2000.
[11] F. I. P. S. Publication, "Secure hash standard, http://www.itl.nist.gov/fipspubs/fip180-1.htm," 1995.
[12] D. R. Karger, E. Lehman, F. T. Leighton, R. Panigrahy, M. S. Levine, and D. Lewin, "Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web," in *Proceedings of the 29th annual ACM symposium on Theory of computing ACM Symposium on Theory of Computing (STOC), El Paso, TX*, pp. 654–663, May. 1997.
[13] N. I. of Standards and Technology, "http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf."
[14] Y. Liu, Y. Gu, H. Zhang, W. Gong, and D. Towsley, "Application Level Relay for High-bandwidth Data Transport," in *The First Workshop on Networks for Grid Applications (GridNets'04)*, (San Jose, CA), October 2004.
[15] C. Jin, D. Wei, and S. Low, "Fast tcp: Motivation, architecture, algorithms, performance," in *Proceedings of IEEE INFOCOM*, (Hong Kong, China), Mar. 2004.
[16] D. Katabi, M. Handley, and C. Rohrs, "Congestion control for high bandwidth-delay product networks," in *SIGCOMM '02: Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, (New York, NY, USA), pp. 89–102, ACM Press, 2002.
[17] Z. Xu, R. Min, and Y. Hu, "HIERAS: A DHT-Based Hierarchical Peer-to-Peer Routing Algorithm," in *the Proceedings of the 2003 International Conference on Parallel Processing (ICPP'03)*, (Kaohsiung, Taiwan, ROC), October 2003.
[18] J. Padhye, V. Firoiu, D. Towsley, and J. Krusoe, "Modeling TCP throughput: A simple model and its empirical validation," *Proceedings of the ACM SIGCOMM '98 conference on Applications, technologies, architectures, and protocols for computer communication*, pp. 303–314, 1998.
[19] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Topologically-Aware Overlay Construction and Server Selection," in *Proceedings of IEEE INFOCOM'02*, (New York, NY), Jun. 2002.
[20] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *Proceedings of the IEEE Conference on Computer Communication, San Francisco, CA*, pp. 594–602, Mar. 1996.
[21] A. Rowstron and P. Druschel, "Storage Management and Caching in PAST, A Large-Scale, Persistent Peer-to-peer Storage Utility," in *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP)*, (Banff, Alberta, Canada), pp. 188–201, Oct. 2001.
[22] B. Zhao, Y. Duan, L. Huang, A. Joseph, and J. Kubiatowicz, "Brocade:landmark routing on overlay networks," in *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS), Cambridge, MA*, March 2002.
[23] Z. Xu, X. He, and L. Bhuyan, "Efficient File Sharing Strategy in DHT-based P2P Systems," in *Proceedings of the 24th IEEE International Performance, Computing, and Communications Conference(IPCCC'05)*, (Phoenix, AZ), April 2005.
[24] T. Wolf, S. You, and R. Ramaswamy, "Transparent TCP acceleration through network processing," in *Proc. of IEEE Global Communications Conference (GLOBECOM)*, (St. Louis, MO), Nov. 2005.
[25] C. Jin, Q. Chen, and S. Jamin, "Inet: Internet topology generator." Report CSE-TR443-00, Department of EECS, University of Michigan, 2000.
[26] A. Medina, A. Lakhina, I. Matta, and J. Byers, "Brite: An approach to universal topology generation," in *Proceedings of the International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems (MASCOTS'01), Cincinnati, OH*, Aug. 2001.
[27] "PlanetLab: An open platform for developing, deploying and accessing planetary-scale services." "http://www.planet-lab.org".