HIERAS: A DHT Based Hierarchical P2P Routing Algorithm

Zhiyong Xu, Rui Min and Yiming Hu
Department of Electrical & Computer Engineering and Computer Science
University of Cincinnati, Cincinnati, OH 45221-0030
E-mail: {zxu,rmin,yhu}@ececs.uc.edu

Abstract

Routing algorithm has great influence on system overall performance in Peer-to-Peer (P2P) applications. In current DHT based routing algorithms, routing tasks are distributed across all system peers. However, a routing hop could happen between two widely separated peers with high network link latency which greatly increases system routing overheads.

In this paper, we propose a new P2P routing algorithm —— HIERAS to relieve this problem, 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 rings besides the highest level P2P ring. A P2P ring is a subset of the overall P2P overlay network. We create P2P rings in such a strategy that the average link latency between two peers in lower level rings is much smaller than higher level rings. Routing tasks are first executed in lower level rings before they go up to higher level rings, a large portion of routing hops previously executed in the global P2P ring are now replaced by hops in lower level rings, thus routing overheads can be reduced. The simulation results show HIERAS routing algorithm can significantly improve P2P system routing performance.

1 Introduction

There are many Peer-to-Peer (P2P) applications [1, 2, 3, 4] and research projects [5, 6, 7, 8, 9, 10, 11] deployed on Internet or under development. As a new network architecture, P2P features of decentralized control, self-autonomous and load balancing make it very attractive in some environments. However, its appealing properties also bring more difficult problems in system design than the traditional Client/Server system, especially in large-scale environments.

A fast, efficient routing algorithm is critical for a P2P system to achieve optimal performance. A bunch of research papers have been proposed, including Pastry [12, 13], Tapestry [14], Chord [15] and CAN [8]. All these algorithms are *Distributed Hash Table (DHT)* based routing algorithms and use similar strategy: Each peer is given a unique identifier and each file is associated with a key (using some collision-free hash functions). Routing information of the files are stored in Distributed Hash Tables (DHTs), a routing procedure is accomplished by consulting the DHTs on several peers. A routing procedure is guaranteed to finish within a small number of routing hops (normally $\log(N)$, N is the total number of system nodes). In each routing hop, the message is sent to a node whose nodeid is numerically closer to the request key. However, problems rise in these algorithms, in a large-scale P2P system, peers are spread all over the world, the network link latencies between two peers are widely varied. It is very likely that a routing hop is taken on two peers with long network link latency. Thus, they cannot achieve the best routing performance.

This problem is caused by the negligence of peers' topological characteristics. In most DHT algorithms, routing tables are created according to peers' numerical characteristics but not the topology properties. Some algorithms take this into account, for example, Pastry constructs a peer's routing table in such a way that the topologically adjacent peers have higher probability to be added into the peer's routing table. During a routing task, the current peer always chooses a peer which is relatively closer to it for the next routing hop. However, Pastry data structures are complex, with the same network sizes, much more routing information than other DHT algorithms like Chord need to be maintained.

We use a simple approach other than Pastry to utilize network topology characteristic. In this paper, we propose an efficient DHT based routing algorithm —— HIERAS, it improves P2P routing performance by combining a hierarchical structure with the current DHT based routing algorithms. Unlike current DHT based algorithms which view the whole system as a single P2P ring only, in HIERAS, besides the biggest P2P ring which contains all the peers, we group topologically adjacent peers into other P2P rings in lower layers as well. A peer belongs to several different layer P2P rings simultaneously. A routing procedure is executed from the scratch, the lowest layer ring is searched first, it moves up to a higher layer and eventually reaches the highest P2P ring.

The rest of the paper is organized as follows, we introduce HIERAS hierarchical structure in Section 2. In Section 3, we describe HIERAS system design. We present our simulation environment and evaluate HIERAS routing algorithm efficiency in Section 4. We discuss the related works in Section 5 and give out conclusions and future works in Section 6.

2 Hierarchical P2P Architecture

HIERAS distinguishes itself from other DHT algorithms with the introduction of the hierarchical structure. In this section, we

Proceedings of the 2003 International Conference on Parallel Processing (ICPP'03)
0190-3918/03 $ 17.00 © 2003 IEEE

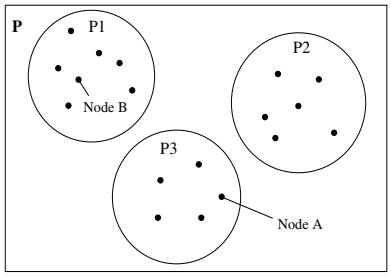

Figure 1: Overview of a Two-Layer HIERAS System

describe HIERAS hierarchical structure in detail.

2.1 Hierarchical P2P Layers

Hierarchical structure is widely used in C/S systems. It is an important mechanism to relieve scalability problem. It is more efficient and easier to manage than broadcasting or pure P2P mechanism. By dividing the whole system into several different layers and solving local tasks inside their own layers, system workloads previously taken by the top layer are greatly reduced.

Current P2P systems do not use hierarchical structure. System peers are deployed on a one-dimension overlay network. HIERAS uses hierarchical structure to improve DHT performance. In HIERAS, a lot of P2P rings coexist in different layers. A P2P ring is a self-organized and relatively independent unit which contains a subset of peers. The members in a P2P ring are equally important and take equal responsibilities for the workloads within this ring. In each P2P layer, all the peers are grouped into several disjointed P2P rings. There's only one ring in the highest layer which contains all the peers. A peer must belong to a P2P ring in each layer, if the system hierarchy depth is k, it belongs to k P2P rings with one in each layer. We organize P2P rings in such a mechanism: the lower the layer, the more topologically adjacent the nodes within a P2P ring in this layer. A simple illustration of a two-layer HIERAS system is shown in Figure 1. It contains 3 layer-2 P2P rings: P1, P2 and P3. P is the Layer-1 (biggest) P2P ring which contains all three layer-2 rings. Node A is a member of P3 and Node B is a member of P1. Both of them are members of P. Every node in the system belongs to P and one of three layer-2 rings.

2.2 Distributed Binning Scheme

An important issue in HIERAS algorithm is how to create the P2P rings and determine to which rings a new node should be added. The decision strategy has great impact on HIERAS efficiency. It must be relatively simple and fast with minimal overhead since nodes may join/leave system frequently. Also, it must be approximately accurate and will not group topologically distant nodes into the same low layer P2P ring. Otherwise, the average link latencies in different layer P2P rings have no big difference and HIERAS is not efficient.

A simple and relatively accurate topology measurement mechanism is the distributed binning scheme proposed by Ratnasamy and Shenker in [16]. In this scheme, a well-known set of machines are chosen as the landmark nodes, system nodes partition themselves into disjoint bins such that nodes that fall within a given bin are relatively closer to each other in terms of network link latency. Although the network latency measurement method (*ping*) is not very accurate and determined by many uncertain factors, it is adequate for HIERAS and we use it for ring creation. Our simulation results show this mechanism is sufficient for HIERAS algorithm to achieve good performance.

Table 1: Sample Nodes in a Two-Layer HIERAS System with 4 Landmark Nodes

Node	Dist -L1	Dist -L2	Dist -L3	Dist -L4	Order
A	25ms	5ms	30ms	100ms	1012
B	40ms	18ms	12ms	200ms	1002
C	100ms	180ms	5ms	10ms	2200
D	160ms	220ms	8ms	20ms	2200
E	45ms	10ms	100ms	5ms	1020
F	20ms	140ms	50ms	40ms	0211

Table 1 shows 6 sample nodes A, B, C, D, E and F in a two-layer HIERAS system with the measured network link latencies to 4 landmark nodes L1, L2, L3 and L4. As [16], we divide the range of possible latency into 3 levels; level 0 for latencies in [0, 20], level 1 for latencies in [20,100] and level 2 for latencies larger than 100. The order information is created according to the measured latencies to the 4 landmark nodes L1, L2, L3 and L4, and this information is used for layer-2 P2P ring decision. For example, Node A's landmark order is 1012. Nodes C and D have the same order: 2200 and they are in same layer-2 ring "2200", all the other nodes are belong to different layer-2 rings.

2.3 Landmark Nodes

We use a well-known set of landmark machines spread across the Internet as [16]. 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. We can also use multiple geographically closest nodes as one logical landmark node to relieve this problem [16].

2.4 Hierarchy Depth

The hierarchy depth also has great influence on HIERAS efficiency. As the hierarchy depth increases, more routing hops previously taken in the highest layer ring will be replaced by hops in low layer rings which results in a better routing performance. However, with an increased hierarchy depth, each node must maintain more routing information, more P2P rings will be created and system overheads of ring maintenance will increase. In our simulation, we found 2-layer or 3-layer is sufficient to achieve good routing performance and will not bring much overheads. System routing performance cannot be improved much more with a bigger hierarchy depth.

Proceedings of the 2003 International Conference on Parallel Processing (ICPP'03)
0190-3918/03 $ 17.00 © 2003 IEEE

3 System Design

HIERAS is a multi-layer DHT based P2P routing algorithm. Like other DHT algorithms, all the peers in HIERAS system form a P2P overlay network on the Internet. However, HIERAS contains many other P2P overlay networks (P2P rings) in different layers inside this overall P2P network. Each P2P ring contains a subset of all system peers. These rings are organized in such a scheme (*by using distributed binning scheme*): the lower the layer of a ring, the smaller the average link latency between two peers inside it. In HIERAS, a routing procedure is first executed in the lowest layer P2P ring which the request originator is located in, it moves up and eventually reaches the biggest P2P ring. A large portion of the routing hops in HIERAS are taken in lower layer P2P rings which have relatively smaller network link latencies. Thus a overall lower routing latency is achieved. In this section, we describe the detailed design of HIERAS algorithm.

3.1 Data Structures

In HIERAS, each node or file is given a unique idenfier generated by a collision free algorithm such as SHA-1. This strategy is used by Pastry, Chord, Tapestry and CAN, so does HIERAS.

All the current DHT based routing algorithms such as Chord have well-organized data structures and efficient routing schemes. HIERAS focuses on improving the efficiency of current DHT based routing algorithms by utilizing peers' topological characteristics. It is built on top of an existing DHT routing algorithm. In each layer, it uses the underlying routing algorithm to perform routing tasks. In our design, we use Chord algorithm as the underlying routing algorithm for its simplicity. However, it is easy to extend HIERAS to other DHT algorithms such as CAN.

Table 2: Node 121 (012)'s Finger Tables in a Two-Layer HIERAS System

Start	Intervals	Layer-1 Successor	Layer-2 Successor
122	[122,123)	124 (“001”)	143 (“012”)
123	[123,125)	124 (“001”)	143 (“012”)
125	[125,129)	131 (“011”)	143 (“012”)
129	[129,137)	131 (“011”)	143 (“012”)
137	[137,153)	139 (“022”)	143 (“012”)
153	[153,185)	158 (“012”)	158 (“012”)
185	[185,249)	192 (“001”)	212 (“012”)
249	[249,121)	253 (“012”)	253 (“012”)

HIERAS routing data structure can be easily created by adding a hierarchical structure on the underlying DHT algorithms. For example, in Chord, each node has a finger table with (at most) k entries (if the identifier length is k bit), the i^{th} entry in node *N*'s finger table contains the nodeid and IP address of the first node, *S*, that succeeds *N* by at least 2^{i-1} on the name space. It is denoted by *N.finger[i].node*. A peer can be added into a node's finger table if it satisfies the numerical requirements for that entry (the details can be found in [15]). This Chord finger table is used as the highest layer finger table in HIERAS without any modification. Besides this highest layer finger table, each node in HIERAS creates *m-1* (m is the hierarchy depth) other finger tables in lower layer P2P rings it belongs to. For a node to generate a lower layer finger table, only the peers within its corresponding P2P ring can be chosen and put into this finger table. A simple illustration is shown in Table 2. The sample system is a two-layer HIERAS system with 3 landmark nodes L1, L2 and L3, all the nodeids are created on a 2^8 name space. Table 2 shows the finger tables on a sample node with the nodeid 121, its second layer P2P ring is “012”. In the highest layer finger table, the successor nodes can be chosen from all system peers. For example, the layer-1 successor node in the range [122,123] is a peer chosen from all system peers, its nodeid is 124 and it belongs to the layer-2 P2P ring “001”. While in the second layer finger table, the layer-2 successor nodes can only be chosen from the peers inside the same P2P ring as node 121 which is “012”. For example, the successor node in the range [122,123] is the node whose nodeid is 143, it also belongs to layer-2 P2P ring “012”.

Besides the data structures inherited from the underlying DHT algorithms, in HIERAS, we use *landmark table* to maintain landmark nodes information, it simply records the IP addresses of all landmark nodes. *Ring tables* are used to maintain information of different P2P rings. The structure of a ring table is shown in Table 3. The ringname is defined by the landmark order information such as “012”. The ringid is generated by using the collision-free algorithm on the ringname. A ring table is stored on the node whose nodeid is the numerically closest to its ringid. It records 4 nodes inside the ring: the node with the smallest nodeid, the node with the second smallest nodeid, the node with the largest nodeid and the node with the second largest nodeid. Ring table is duplicated on several nodes for fault tolerance. The node which stores the ring table periodically checks the status of these nodes, in case of a node failure, a new routing procedure is performed to add a new node into the table. Ring table is used to find a node in this particular P2P ring when a new node is added into system, the details of its usage will be introduced in Section 3.3.

3.2 Routing Algorithm

HIERAS algorithm uses hierarchical routing mechanism. In a *m*-layer HIERAS system, a routing procedure has *m* loops. In the first loop, the routing procedure starts from the lowest P2P ring which the request originator belongs to, the finger table in this ring is used. The routing procedure inside this ring continues until the routing message has been forwarded to a peer whose nodeid is the numerically closest to the requested key than any other peers in this ring. The same operation is repeated in different layers, the only difference is the usage of the different finger table. As the routing procedure goes up, more and more peers are included and the message is forwarded more and more closer to the destination node. At the last loop, the routing procedure is executed on the largest P2P ring which includes all system peers and the routing procedure definitely will end at the destination node. After the message arrives the destination node, the node returns the location information of the requested file to the orig-

Proceedings of the 2003 International Conference on Parallel Processing (ICPP'03)
0190-3918/03 $ 17.00 © 2003 IEEE

Table 3: P2P Ring Table Structure

Ringid	Ringname	Node (Largest Id)	Node (Second Largest Id)	Node (Smallest Id)	Node (Second Smallest Id)

inator and the routing procedure finishes. Predecessor and successor lists can be used to accelerate the process. In each loop, when the routing message reaches the numerically closest node in this ring, HIERAS algorithm checks if the current peer is the destination node or not. If it is, the routing procedure finishes and the result is returned immediately. Only in case of false, the routing procedure will continue and the routing message is forwarded to the upper layer P2P ring.

Clearly, in HIERAS, the same underlying DHT routing algorithm keeps being used in different layer rings with the corresponding finger table. Though we use Chord here, it is easy to transplant to other algorithms. For example, if we use CAN as the underlying algorithm, the whole coordinate space can be divided multiple times in different layers, we can create multi-layer neighbor sets accordingly and use these neighbor sets in different loops during a routing procedure. The algorithm is very simple, it has m steps, and in each step, the underlying routing algorithm is performed with the finger table in that ring. The underlying algorithm is responsible for checking the current peer is the destination node or not. If it is, HIERAS algorithm stops the loop and returns the result to the originator.

Comparing to the current DHT based algorithms, such a hierarchical scheme has several advantages. First, it keeps the scalability property of the current algorithms, a routing procedure definitely finishes within $O(\log N)$ steps. Although it takes several loops, it has the same trend as its underlying algorithm: The routing message keeps moving towards the destination node by reducing nearly half of the distance each time. Second, it greatly reduces the actual routing latency. Suppose in a P2P system with 100000 nodes and the average link latency per hop in Chord is 100ms which is the same as the average link latency in the biggest P2P ring in HIERAS. If the average number of routing hops is 6, the average routing latency in Chord is 6×100 equals 600ms. Assuming in a 2-layer HIERAS system, the average link latency in the lower layer is only one fourth of the higher layer which is 25ms each hop, and 4 hops are executed in the lower layer rings, the average routing latency by using HIERAS is 4×25ms plus 2×100ms which is 300ms. The P2P system average routing latency can reduces by 50%. Third, by using an existing DHT routing algorithm as the underlying algorithm, the well-designed data structure and mechanisms for fault tolerance, load balance and caching scheme of the underlying algorithm are still kept in HIERAS which greatly reduces the design complexity.

3.3 Node Operations

In HIERAS, when a new node n joins the system, operations other than Chord algorithm must be taken. Prior to other operations, it must send a *join* message to a nearby node *n'* which is already a member of the system. This process can be done in different methods, we just omit it and simply assume it can be done quickly. (This is the same assumption as other DHT algorithms.) Then the node n can get the information of landmark nodes from this nearby node *n'* and fulfill its own landmark table. It then decides the distance between itself and the landmark nodes and uses the distributed binning scheme to determine the suitable P2P rings it should join.

In the following steps, it must create routing data structures such as finger tables in each layer. First, it creates the highest layer finger table, the mechanism used in Chord can be introduced without modification. Because node n already knows one nearby node *n'* in the system, it can learn its fingers by asking node *n'* to look them up in the whole P2P overlay network. The detailed process is described in [15]. Thus the highest layer finger table is created. Second, it needs to create finger tables in lower layers. To create the finger table in a specific ring, node n must know at least one node p in that ring. This is done as follows: node n calculates the ringid of this ring and sends a *ring table request* message to node c which stores the ring table of this ring using the highest layer finger table. This particular routing procedure is not a multi-layer process, it is the same as an ordinary Chord routing procedure. As c receives this message, it sends a response to node n with the stored ring table, then node $\hat{n}$ knows several nodes in that ring. To create its own finger table of this ring, node n sends a *finger table creation* request to a node p inside this ring with its own nodeid. Node p modifies its corresponding finger table to coordinate the arrival of the new node and create the finger table for node n using the same mechanism as in the highest layer except that this time, the routing only occurs within this ring. After p generates the finger table of n, it sends back to n. Node n is successfully joined to this specific ring. n then compares its nodeid with the nodeids in the ring table, and if it should replace one of them (larger than the second largerst nodeid or smaller than the second smallest nodeid), it sends a *ring table modification message* back to node c and c modifies the ring table accordingly. In a m-layer HIERAS system, this procedure will repeat m times. After that, all the finger tables of the newly added node n are created. Node n joins system successfully.

In P2P systems, a node may leave the system or fail silently. As in Chord algorithm, the key step in node failure operation is maintaining correct successor pointers. HIERAS can use the same strategy except in HIERAS, a node must keep a "successor-list" of its r nearest successors in each layer. Thus it has more overheads. However, the nodes within the low layer rings are topologically closer, the maintenance overhead is affordable.

3.4 Cost Analysis

The routing performance gain in HIERAS comes at the cost of extra overhead. In Chord, each node only needs to maintain one finger table while in HIERAS, each node must maintain multi-

Proceedings of the 2003 International Conference on Parallel Processing (ICPP'03)
0190-3918/03 $ 17.00 © 2003 IEEE

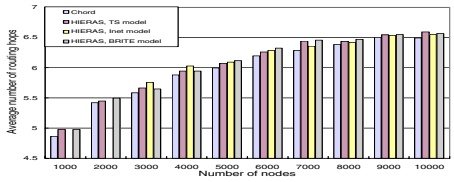

Figure 2: HIERAS and Chord Routing Performance Comparison (Routing Hops)

ple finger tables which increases system maintenance overhead, HIERAS has multiple "successor-lists" while Chord only has one. Also HIERAS needs more operations such as calculating ring information and requesting the ring table when a node joins the system. However, in case of a small number of hierarchy depth like 2 or 3, the cost is affordable, the occupied space by multi-layer finger tables are only hundred or thousands of bytes. Normally, the number of nodes in the lower layer finger table is smaller than the higher layer finger table (as described in Table 2) and the nodes in lower layer rings are topologically closer to each other, then the cost to keep the content of lower layer finger table up-to-date is much smaller compared to the maintenance overhead in the higher layer (the nodes are widely separated).

4 Performance Evaluation

We conducted trace-driven simulation experiments to evaluate HIERAS routing performance. First, we describe network models, workload traces and routing algorithms used in our simulation. Then we present our simulation results.

4.1 Simulation Environment

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 [17]. 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). Besides TS, Inet [18] and BRITE [19] models are used for comparison purpose.

In our simulations, we vary the number of system nodes from 1000 to 10000 in all simulated networks except in Inet the minimal number of nodes is 3000. We use Chord as the underlying DHT algorithm in HIERAS and we compare the routing performance of the original Chord design with HIERAS. We use a two-layer configuration for HIERAS in all simulations except in Section 4.5 when evaluating the effects of hierarchy depth.

4.2 Routing Costs

The primary goal of HIERAS is to reduce the routing cost in current DHT algorithms. In this simulation, we compare HIERAS routing performance with Chord in different models and network sizes. In all experiments, HIERAS algorithm uses 4-landmark nodes configuration, and in each experiment, we use 100000 randomly generated routing requests.

Figure 2 shows the comparison result measured with the metric of "average number of routing hops". Both HIERAS and Chord algorithm have good scalability: as the network size increases from 1000 nodes to 10000 nodes, the average number of routing hops only increases around 32%. HIERAS performance is a little worse than Chord, which means, in all situations, HIERAS has a larger average number of routing hops. However, the difference between HIERAS and Chord algorithm is very small, HIERAS only takes 0.78% to 3.40% more average number of routing hops than Chord.

Figures 3 shows the comparison results measured by metric of "the average routing latency". Although HIERAS has higher average number of routing hops than Chord, it has smaller average routing latency in all experiments. For TS model, the average routing latency in HIERAS is only 51.8% of Chord. For Inet model, the average latency is 53.41% of Chord. Even for BRITE model, the routing cost in HIERAS is still only 62.47% of Chord. Clearly, the average link latency per hop in HIERAS is much smaller than Chord which greatly recduces routing cost. We will show this in the following simulations.

In TS model, the average routing latency in a 7000-node network is even smaller than a 6000-node network. This is caused by configurations of different numbers of transit domains, stub domains and nodes in each stub domain in these two emulated networks. However, this does not violate the overall result.

4.3 Routing Cost Distribution

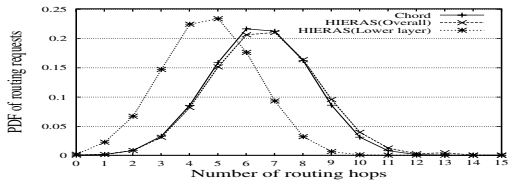

Figure 4: Probability Density Function (PDF) Distribution of the Number of Routing Hops

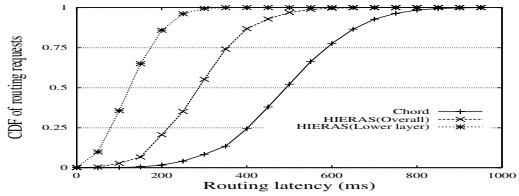

Figure 5: Cumulative Density Function (CDF) Distribution of the Routing Latency

To better understand the effects of using hierarchical structure in HIERAS, we analyze the routing cost distributions. Figure 4 and Figure 4.3 show the simulation results. The data are collected from 100000 randomly generated routing requests on a 10000-node TS network.

Proceedings of the 2003 International Conference on Parallel Processing (ICPP'03)
0190-3918/03 $ 17.00 © 2003 IEEE

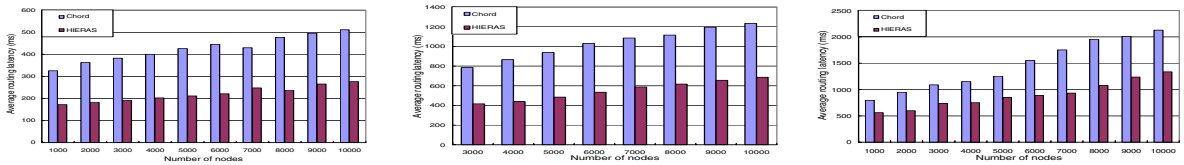

Figure 3: HIERAS and Chord Routing Performance Comparison (Average Latency, TS, Inet and BRITE model)

The probability density function (PDF) distribution curves of the average number of routing hops in Chord and HIERAS are nearly coherent except in HIERAS, the number of requests finishes between 0 and 8 hops are little smaller than Chord, while between 9 and 15 routing hops, the numbers of routing requests in HIERAS are little bigger. Thus the average number of routing hops in HIERAS is 6.5937 while it is 6.4933 in Chord. HIERAS spends 1.55% more hops per request than Chord on average. However, as shown in the third distribution curve, in HIERAS, only 1.887 hops per request are taken in the higher layer, the rest 71.38% routing hops are executed on the lower layer P2P rings.

Figure 4.3 also shows the measured cumulative density function (CDF) distribution curves of the average routing latency in Chord and HIERAS algorithms. The average routing latency in HIERAS is 276.53 ms, while in Chord, it is 511.47 ms. The average routing latency in HIERAS is only 54.07% of Chord. The performance improvement comes from the replacement of a large amount of routing hops from the higher layer P2P ring to the lower layer rings. As a result, the number of requests finishes with a small routing latency in HIERAS is much more than in Chord. In this simulation, the average link delay in the higher layer ring is 79ms while in the lower layer rings, it is only 27.758ms which is only 35.23% of the link delay in the higher layer. In HIERAS, although routing hops taken on the lower layer occupy 71.38% of all routing hops, the network latency in the lower layer only occupy 47.24% of the overall routing latency. The delay difference ratio is only 2.85 in our simulated networks, we can expect an even larger link latency difference exists between local area network (LAN) and wide area network (WAN) in real world, HIERAS algorithm can achieve even higher improvement.

4.4 Landmark Nodes Effect

In previous experiments, only 4-landmark nodes configuration is used. In this simulation, we test the effects of different number of landmark nodes in HIERAS. The experiments are conducted on a 10000-node TS network with 100000 randomly generated routing requests. We vary the number of landmark nodes from 2 to 12. Figure 6 shows the result. As the number of landmark nodes increases, the average number of routing hops changes little. For 2 to 5 landmark nodes configurations, HIERAS has a higher average number of routing hops than Chord. For 6 to 8 landmark nodes configurations, the average number of routing hops in HIERAS reduces and it is even smaller than Chord. After 9 landmark nodes, it becomes higher again. These results indicate that the average number of routing hops in HIERAS depends on the number of landmark nodes which determines the total number of the lower layer P2P rings. With an adequate number of landmark nodes, HIERAS can achieve the minimal number of routing hops. Too few and too many landmark nodes will hurt its efficiency. Another observation is: as the number of landmark nodes increases from 2 to 8, the average number of routing hops that taken on the lower layer P2P rings reduces sharply. But after 8, it becomes constant. The intuition behind this fact is straightforward, as the number of landmark nodes increases, the number of P2P rings also rises, the average number of peers in each ring reduces, hence the routing hops taken on the lower layer reduces.

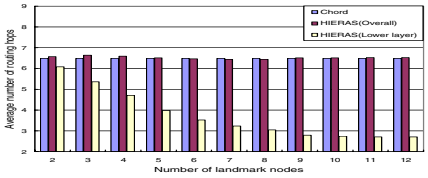

Figure 6: Comparing Average Number of Routing Hops with Different Number of Landmark Nodes in HIERAS Algorithm, TS Model

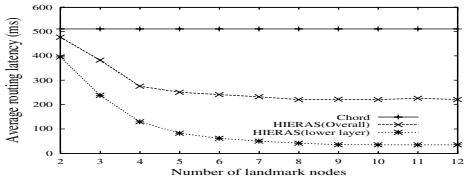

Figure 7: Comparing Average Routing Latency with Different Number of Landmark Nodes in HIERAS Algorithm, TS Model

Figure 7 shows the simulation results measured in average routing latency. With the 2-landmark nodes configuration, the average routing latency in HIERAS is only 7.12% less than Chord. This fact indicates with a small number of landmark nodes, HIERAS is not efficient because system can not generate enough lower layer P2P rings to group adjacent nodes inside. The average routing latency decreases rapidly with the increased number of landmark nodes. Clearly, HIERAS can get more accurate topological information and the distributed binning scheme works much better, large portions of routing

Proceedings of the 2003 International Conference on Parallel Processing (ICPP'03)
0190-3918/03 $ 17.00 © 2003 IEEE

hops are taken between peers with low link latency. With an 8-landmark nodes configuration, HIERAS achieves the highest performance: the average routing latency is only 43.31% of Chord. After that, the performance gain diminishes slightly. However, even with a 12-landmark nodes configuration, HIERAS routing performance is still very good. This indicates while distributed binning scheme can work even better with the decreased number of nodes in each ring, the number of routing hops taken on the lower layer reduces, HIERAS can not get more benefits.

4.5 Hierarchy Depth Effect

We evaluate the effects of hierarchy depth in this simulation. We change hierarchy depth from 2 to 4, the number of nodes in simulated TS network varies from 5000 to 10000, 6 landmark nodes are used. The results are shown in Figure 8 and 9. The average number of routing hops in HIERAS algorithm is getting larger as the hierarchy depth increases, however, the increment is very small. Even in a 4-layer HIERAS system, the average number of routing hops is only 0.29% to 1.65% larger than a 2-layer system. The average routing latency reduces as the depth increases, from 2-layer to 3-layer, the average routing latency reduction is between 9.64% and 16.15% while from 3-layer to 4-layer, it is between 2.12% and 5.42%, the latency even increases for a 7000-node network. While increasing the depth brings more system maintenance overhead, a 2 or 3 layer HIERAS system is the optimal configuration.

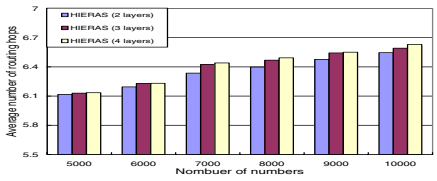

Figure 8: HIERAS Performance with Different Hierarchy Depth (Average Number of Hops), TS Model

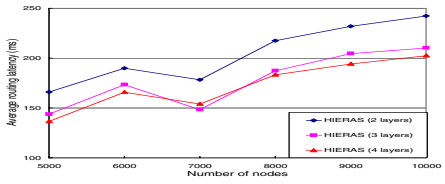

Figure 9: HIERAS Performance with Different Hierarchy Depth (Average Latency), TS Model

5 Related Works

Chord [15] [20], Pastry [12], Tapestry [14] and CAN [8] are all based on distributed hash tables. Chord and CAN algorithms are relatively simply and easy to manage. However, they do not consider network topological character too much. Hence, they can achieve the minimal average number of routing hops but not the real minimum routing latency. In fact, in most cases, a route chosen by these algorithms is not the shortest path measured by link latency. We have already discussed the reasons in Section 1. HIERAS algorithm is simpler and easier to manage than Pastry while it greatly improves routing efficiency for other DHT based algorithms such as Chord.

Ratnasamy and Shenker [16] pointed out that P2P or other large-scale network applications could potentially benefit from some level of knowledge about the relative proximity between its participating nodes. They suggested allocating the topologically adjacent peers with congruent identifiers together, and they applied this idea on CAN with a good performance improvement. HIERAS has the similar objective and also utilizes nodes' topological property to improve P2P system routing performance. We use their distributed binning scheme for P2P ring creation purpose. However, HIERAS differs their work in two aspects: First, the routing procedures in [16] still occur in the one dimension name space while in HIERAS, routing becomes a multi-level procedure. Second, they created topologically sensitive CAN network by modifying the identifier generation mechanism which makes the coordinate space no longer uniformly populated. In HIERAS, we keep the original system architecture and build HIERAS system on top, the underlying network architecture is untouched.

In [21], the authors use probabilistic location and routing scheme to enhance P2P location mechanism in the case where a replica for the queried data exists close to the client, it uses different approach to reduce P2P system routing overhead. [22] uses small-world model to improve Freenet [9] performance, it is aimed on non-DHT based P2P systems. Although it is also very important, we do not address the problem in this paper.

Harvest Cache [23] [24] is a proxy-cache architecture, it creates a hierarchical cache organization and helps distributing load from hot spots. However, in Harvest cache, the roles of clients and servers in each level are still strictly separated, so it is still a traditional C/S application. While in HIERAS, in each level P2P ring, the role and responsibility of each node is the same. By introducing the hierarchical structure into the current DHT based routing algorithms, HIERAS achieves significant routing performance improvement.

6 Conclusions and Future Works

In this paper, we propose a new DHT based P2P routing algorithm —— HIERAS. Besides the biggest P2P ring which includes all system nodes, it creates many small P2P rings in different layers by grouping topologically adjacent nodes together. A node belongs to several different layer P2P rings simultaneously. In each P2P ring, the members have the equal responsibilities for workloads in this ring. A routing procedure is executed in lower layer P2P rings before it goes up, and eventually reaches the global P2P ring. By taking a large portion of routing hops in lower layer P2P rings which have a smaller link latency between any two nodes inside these rings, HIERAS algorithm greatly reduces P2P system routing cost. With simulation results, we can draw the following conclusions:

1. Hierarchical structure does not contradict with P2P ar-

Proceedings of the 2003 International Conference on Parallel Processing (ICPP'03)
0190-3918/03 $ 17.00 © 2003 IEEE

chitecture, by combing them together, an efficient P2P routing algorithm with lower average routing latency is obtained.

2. P2P system topology characteristics can be used to improve performance.

3. Hierarchical structure can improve routing performance of current DHT routing algorithms, no matter the algorithms take network topology into account or not.

4. The number of landmark nodes and the hierarchy depth has great influence on HIERAS efficiency, with an adequate number of landmark nodes and hierarchy depth, HIERAS can achieve the best performance.

In the future, we plan to do a quantitative analysis of HIERAS overheads and compare with current DHT algorithms. We also plan to run more simulations to compare HIERAS performance with other low latency DHT algorithms such as Pastry and Tapestry, and if possible, do the real implementation of HIERAS.

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] J. Kubiatowicz, D. Bindel, P. Eaton, Y. Chen, D. Geels, R. Gummadi, S. Rhea, W. Weimer, C. Wells, H. Weatherspoon, and B. Zhao, "OceanStore: An architecture for global-scale persistent storage," in *Proceedings of the 9th international Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), Cambridge, MA*, pp. 190–201, Nov. 2000.

[6] P. Druschel and A. Rowstron, "Past: A large-scale, persistent peer-to-peer storage utility," in *the 8th IEEE Workshop on Hot Topics in Operating Systems (HotOS), Schoss Elmau, Germany*, May 2001.

[7] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica, "Wide-Area cooperative storage with CFS," in *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP), Banff, Alberta, Canada*, pp. 202–215, Oct. 2001.

[8] 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.

[9] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong, "Freenet: A distributed anonymous information storage and retrieval system," in *Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA*, pp. 46–66, Jul. 2000.

[10] R. Dingledine, M. J. Freedman, and D. Molnar, "The free haven project: Distributed anonymous storage service," in *Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA*, pp. 67–95, Jul. 2000.

[11] A. D. R. Marc Waldman and L. F. Cranor, "Publius: A robust, tamper-evident, censorship-resistant, web publishing system," in *Proceedings of 9th USENIX Security Symposium, Denver, CO*, pp. 59–72, Aug. 2000.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] C. Jin, Q. Chen, and S. Jamin, "Inet: Internet topology generator." Report CSE-TR443-00, Department of EECS, University of Michigan, 2000.

[19] 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.

[20] 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.

[21] S. Rhea and J. Kubiatowicz, "Probabilistic location and routing," in *Proceedings of INFOCOM.*, 2002.

[22] H. Zhang, A. Goel, and R. Govindan, "Using the small-world model to improve freenet performance," in *Proceedings of INFOCOM.*, 2002.

[23] A. Chankhunthod, P. B. Danzig, C. Neerdaels, M. F. Schwartz, and K. J. Worrell, "A hierarchical internet object cache," in *USENIX Annual Technical Conference*, pp. 153–164, 1996.

[24] C. M. Bowman, P. B. Danzig, D. R. Hardy, U. Manber, and M. F. Schwartz, "The Harvest information discovery and access system," *Computer Networks and ISDN Systems*, vol. 28, no. 1–2, pp. 119–125, 1995.

Proceedings of the 2003 International Conference on Parallel Processing (ICPP'03)
0190-3918/03 $ 17.00 © 2003 IEEE