www.ijecs.in
International Journal Of Engineering And Computer Science ISSN: 2319-7242
Volume 5 Issue 4 April 2016, Page No. 16244-16251

Proposed New Architecture Of Overlay Network Of Distributed Hash Table

Jyotsana Sharma[1],Birendra Kumar[2]
Krishna Engineering College, Address-95 Loni Road Mohan Nagar, Rajendra Nagar, Gaziabad.UP-201007.
jyotsanasharma611@gmail.com

ABSTRACT: This paper proposes the new architecture of overlay network of Distributed Hash table (DHT). We introduce a new MultiChord Protocol which is another variant of Chord Protocol defined over overlay network of Distributed Hash table. MultiChord inherits basic properties of Chord protocol with some added new features.

Keywords: DHT, Chord.

I.INTRODUCTION

Distributed hash tables (DHTs) are a class of decentralized distributed systems that provide a lookup service similar to a hash table, i.e., it stores a no of (key, value) pairs on various nodes that are distributed across the internet, and provides an efficient routing algorithm to perform various operations (lookup, node joins, node leave) on that data. Hence, we can say, it is a giant hash table that is cooperatively maintained by a large number of machines worldwide. An overlay network is a virtual network of nodes and logical links that is built on top of an existing network with the purpose to implement a network service that is not available in the existing network DHTs employ an overlay network that is a base for the routing algorithm. In this research, we present MultiChord, a distributed lookup protocol for peer-to-peer networks, that provides hash table like functionality. It employs the concentric circles geometry to distribute the network traffic onto different circles in accordance with their nodeids that are generated by a hash function. Previous protocol, such as Chord, already provides a scalable query latency of O(log N), where 'N' is the no. of nodes in the peer-to-peer network. But in MultiChord, the nodes are distributed among multiple circles, unlike Chord where all the nodes in the system are placed on one circle. Hence, MultiChord retains the O(log N) routing latency, but since now the nodes are distributed among various circles instead of one, effective 'N' is reduced and hence the overall performance of the system improves.

Limitation of Distributed Hash Tables

Distributed Hash Tables are a cornerstone of state-of-the-art Peer-to-Peer systems. They mean a remarkable advancement in solving the issue of scalability and decentralization, with the added value of determinism and high guarantees. However, this has opened a whole set of new questions that need to be addressed. What follows is a summary of those issues, from [SEA-2]. Quoting:

a) **Lack of a Common Framework** Research in DHT systems has been addressed by different research groups. The results were the emergence of systems that are very similar in basic principles. Nevertheless, there is no common framework that allows the common understanding and reasoning about those systems.

b) **Locality** Though accounted for in systems like Pastry and Tapestry, locality remains to be an open research issue. Additionally, the loss of locality due to hashing is not always considered a disadvantage. The

Oceanstore system [OCS-3] which depends on Tapestry for location and routing, considers loss of locality favorable because replicas of items would be stored at physically apart nodes which renders a system resistant to denial of service attacks.

c) **Cost of Maintaining the Structure** Most of the current DHTs depend on the periodic checking and correction (stabilization) for the maintenance of the structure which is crucial to the performance properties of those systems. This periodic activity costs a high number of messages and sometimes unnecessarily in the case of checking stable sections of a routing table. The awareness about this problem motivated research such as e.g., [MAH-4] where a network tries to ―self-tune‖ the rate at which it performs periodic stabilization.

d) **Complex Queries** DHTs assume that for each item, there is a unique key and to retrieve that item one must know the respective key. That is, one cannot search for items matching a certain criteria like a keyword or a regular-expression-specified query. The feasibility of the task is questionable [JLI-5]. Some of the approaches include the insertion of indices [HAR-6] for general queries or using some geometrical constructs that make use of the DHT structure such as space-filling curves [AND-7]. Another approach is to let the hashing be based on keywords or semantic information and not on unique keys [SCH-8].

e) **Heterogeneity** While all DHT systems aim at letting all nodes have equal duties and responsibilities, the heterogeneity in physical connectivity makes them unequal. Consequently, nodes with higher latencies constitute bottlenecks for the operation of structured P2P systems. Two approaches were suggested to cope with those problems:

i) Cloning: The more powerful nodes are cloned so they can act as multiple nodes and receive higher percentage of the uniformly distributed traffic [DAB-9]

ii) Clustering: Nodes of similar latency behavior are clustered together [ZXU-10].

Group Communication Since structured P2P systems offer graphs of known topologies to connect peers, it is natural to start exploiting the structural properties in group communication. The main focus in P2P Group communication is on multicasting. Extensions like [STO-1], [RAT-11], [CAS-12] aim at providing multicast layers to existing DHT systems. Publish-subscribe communication [TAN-13] is also another form of group communication that was researched in P2P systems [BAE-28].

II. Related Work

DHTs were first introduced to the research community in 2001, with the near-simultaneous introduction of four different architectures: CAN, Chord, Pastry, and Plaxton et al. Since that time, there have been an amazing numbers of new DHT architectures proposed, but very few publicly-released, robust implementations.

This area of research has been quite active since it was introduced. Outside academia, DHT technology has been adopted as a component of BitTorrent and in the Coral Content Distribution Network.

DHT research was originally motivated, in part, by peer-to-peer systems such as Napster, Gnutella, and Freenet, which took advantage of resources distributed across the Internet to provide a single useful application

Desired Characteristics of a DHT:

DHTs characteristically emphasize the following properties:

* *Decentralization*: the nodes collectively form the system without any central coordination.

* *Scalability*: the system should function efficiently even with thousands or millions of nodes.

* *Fault tolerance*: the system should be reliable (in some sense) even with nodes continuously joining, leaving, and failing.

In addition to the above mentioned issues, DHTs must deal with more traditional distributed systems issues such as load balancing, data integrity, and performance (in particular, ensuring that operations such as routing and data storage or retrieval complete quickly).

Basic Operation a DHT performs:

1. Store(key, val) *[put operation]*

2. val = Retrieve(key) *[get operation]*

Review of Basic DHT Algorithms

All of them take, as input, a key and, in response, route a message to the node responsible for that key. The keys are strings of digits of some length. Nodes have identifiers, taken from the same space as the keys (i.e., same number of digits). Each node maintains a routing table consisting of a small subset of nodes in the system. When a node receives a query for a key for which it is not responsible, the node routes the query to the neighbor node that makes the most "Progress" towards resolving the query. The notion of progress differs from algorithm to algorithm, but in general is defined in terms of some distance between the identifier of the current node and the identifier of the queried key.

2.1 A Scalable Content-Addressable Network (CAN)

Basic Idea of CAN[CAN-14]: A virtual d-dimensional Coordinate space is considered.
-Each node owns a Zone in the virtual space
-Data is stored as (key, value) pair
-Hash(key) --> a point P in the virtual space
-(key, value) pair is stored on the node within whose Zone the point P locates For routing purpose, each node only need to maintain the information of those nodes that hold coordinate zone adjoining its own zone (neighbors)

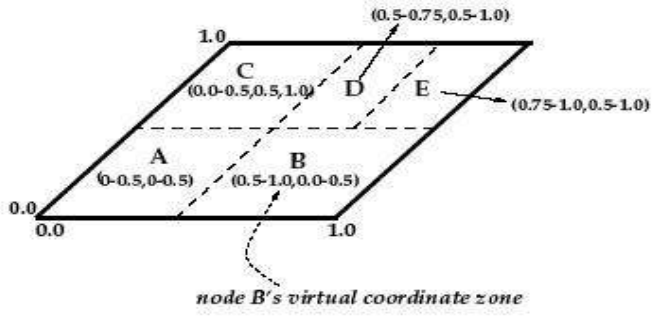

Fig 2.1 : A (sample) 2-d space with 5 nodes

2.2 Chord

The chord algorithm [STO-1] is nothing but a circular, double-linked list. Each node in the list is a machine on the network. Each node keeps a reference to the next and previous nodes in the list, the addresses of other machines. There must be an ordering with which we can determine what the "next" node is for each node in the list. The method used by the Chord DHT to determine the next node is as follows: assign a unique random ID of k bits to each node. Arrange the nodes in a ring so the IDs are in increasing order clockwise around the ring. For each node, the next node is the one that is the smallest distance clockwise away. For most nodes, this is the node whose ID is closest to but still greater than the current node's ID. The one exception is the node with the greatest ID, whose successor is the node with the smallest ID.

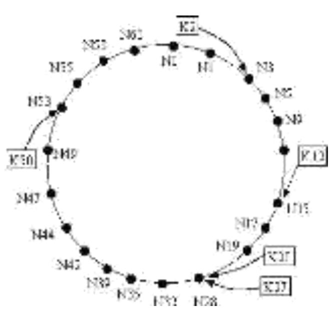

Fig.2.2: circular identifier address space with twenty nodes and five keys

Each node is itself a standard hash table. All we need to do to store or retrieve a value from the hash table is find the appropriate node in the network, then do a normal hash table store or lookup there.

2.3 Pastry

In Pastry [PAS-15] the hash table's keyspace is taken to be circular, like the keyspace in the Chord system, and node IDs are 128-bit unsigned integers representing position in the circular keyspace. Node IDs are chosen randomly and uniformly so peers who are adjacent in node ID are geographically diverse. The routing overlay network is formed on top of the hash table by each peer discovering and exchanging state information consisting of a list of leaf nodes, a neighborhood list, and a routing table. The leaf node list consists of the L/2 closest peers by node ID in each direction around the circle.

2.4 Tapestry

Tapestry [TAP-16] is an extensible infrastructure that provides decentralized object location and routing focusing on efficiency and minimizing message latency. This is achieved since Tapestry constructs locally optimal routing tables from initialization and maintains them in order to reduce routing stretch. Furthermore, Tapestry allows object distribution determination according to the needs of a given application. Similarly Tapestry allows applications to implement multicasting in the overlay network.

2.5 Recent Work

The algorithms described above are all multi hop P2P Networks. In a multi-hop system, a message is routed through several hops in the overlay network, with each intermittent node in the source-destination path contributing to the guidance of the message to its destination. Recently the 1-hop systems have come into play. A 1-hop system (often referred to as single hop) aims to achieve look up operations within O(1) hops

2.5.1 Epichord

EpiChord[EPI-17] is a DHT lookup algorithm that demonstrates that we can remove the O(log n)-state-per-node restriction on existing DHT topologies to achieve significantly better lookup performance and resilience using a novel reactive routing state maintenance strategy that amortizes network maintenance costs into existing lookups and by issuing parallel queries. Our technique allows us to design a new class of unlimited-state per-node DHTs that is able to adapt naturally to a wide range of lookup workloads. EpiChord is able to achieve O(1)-hop lookup performance under lookup-intensive workloads, and at least O(log n)-hop lookup performance under churn-intensive workloads even in the worst case (though it is expected to perform better on average) . Our reactive routing state maintenance strategy allows us to maintain large amounts of routing state with only a modest amount of bandwidth, while parallel queries serve to reduce lookup latency and allow us to avoid costly lookup timeouts.

2.5.2 D1HT

It is a novel single hop DHT that is able to maximize performance with reasonable maintenance traffic over head even for huge and dynamic peer-to-peer (P2P) systems. It detects and notifies any membership change in the system, prove its correctness and performance properties, and present a Quarantine-like mechanism to reduce the overhead caused by volatile peers. It is a one hop P2P network [D1H-18]

III. The Proposed Protocol MultiChord

A MultiChord system is a self-organizing overlay network of nodes, where each node routes client requests and interacts with local instances of one or more applications. It uses consistent hashing which has several good properties. The basic difference between chord and MultiChord is, that in chord, all the nodes in the system are placed on one circle, whereas, in MultiChord, the nodes are placed on multiple circles.

3.1 Structure

As outlined above, MultiChord is organized as multiple concentric circles (that provide as one-dimensional address space). In its structure, the radius of consecutive circles differs by 1. The smallest circle, which will eventually be the innermost circle, will be of radius 1. And then, the radius of each next circle increments by 1, that is, the second circle is of radius 2, third is of radius 3 and so on,

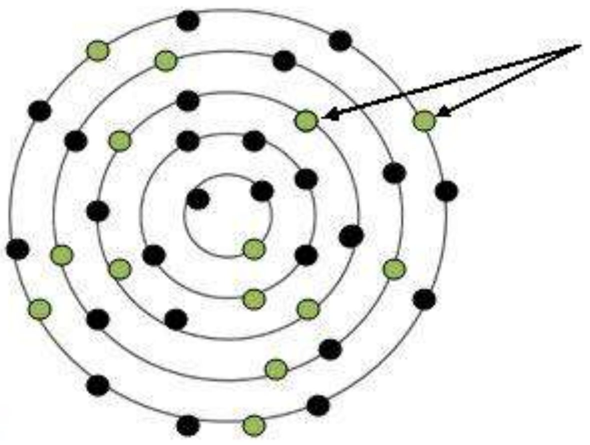

Fig.3.1 The concentric circle structure of MultiChord

In the fig, the dots represent nodes. The black nodes are the normal nodes in the system, however, the green ones are connecting nodes, described below.

3.2 Connecting Nodes

Connecting nodes, as the name implies, are used to connect the circles together. They are used to traverse between the circles. In the finger table of a connecting node, in addition to the list of successors, contains the IP‘s of the connecting node in the immediate outer and the immediate inner circle. Hence, if suppose a node shoots a query for a key placed on a different circle than the one the querying node is itself located. Then it first goes to the connecting node placed on its own circle. This connecting node will contact the connecting node on the immediate next circle, which will again ‘contact‘ the next connecting node on immediate next circle, until the target node is reached.

Each node in MultiChord is assigned an m-bit identifier, that is generated by hashing the node‘s

IP address (This identifier consists of digits only). This identifier or the nodeid is used to indicate a node‘s position in the multi circular address space as follows:
The digits of nodeid are added up. Let this sum be S. Now, the node with sum S is placed on the circle with radius equal to S. Hence, a circle ’contains’ the nodes whose nodeid’s digit sum is equal to its radius.

Let’s see an example …

The node identifier address space is of 5 bits in base 4, i.e., all nodes are hashed to 5 digit numbers in base 4. Thus, the maximum no. of circles possible is 5*4 = 20. Consider the following table ……

S.No	*Nodeids*	*SumofDigits*	*Respective circle*
1	32102	8	8
2	23121	9	9
3	32000	5	5
4	23021	8	8
5	30020	5	5

Thus, node3 and node 5 will be placed on 5th circle (starting from the smallest one), i.e., the circle with radius =5. Similarly, node1 and node 4 will be placed on circle with radius = 8 and node 2 will be placed on circle with radius =9.

On a circle, the node responsible for a key is the node whose identifier most closely follows the key, i.e., the successor (see Fig.)

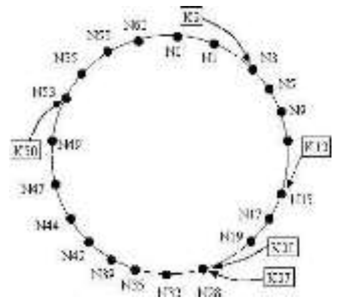

Fig. 3.2 Circular identifier addresses space with twenty nodes and five keys.

3.3 Operation in MultiChord

After having described the basic structure of the overlay network that MultiChord deploys, let's now see the various operations that can be performed on the same:

3.3.1 Lookup Operation

The lookup operation is performed as follows:

1. The key is hashed to obtain its m-bit identifier (Each node and key are assigned an m-bit identifier).
2. The sum of its digits is calculated.
3. The target circle is reached via the connecting nodes.
4. The circle with that radius is identified to be the one containing the particular node.
5. After navigating to the target circle the node with id greater than or equal to the key is identified using the simple chord algorithm.

Hence in MultiChord, the node information required to reach the target node is O(log (M)), where M is average no of nodes per circle (We assume uniform distribution of nodes among the circles)

3.3.2 Join

In a dynamic network, nodes can join (and leave) at any time. The main challenge in implementing these operations is preserving the ability to locate every key in the network.

In MultiChord a node join takes place as follows:

We assume that the new node learns the identity of an existing node by some external mechanism.

The joining node‘s IP is hashed to generate the nodeid. The sum of its digits is obtained, and the node that was 'chosen‘ by the joining node, forwards a query with the nodeid of the new node. After this, the above mentioned process is followed to find the appropriate circle and then the node is placed on the circle according to the simple Chord Algorithm.

3.3.3 Leave

A node departure takes place exactly as happens in the basic chord protocol. However, this operation could lead to a situation where the leaving node was the only node left in the circle. In that situation, the whole connectivity would break. So, in this case, the leaving node would first 'inform'(a special function *'RetainConnectivity'* is invoked) the connecting node of immediate inner and the immediate outer circle, so that the later can update their tables accordingly and the connectivity of the whole system is retained.

3.4 Conclusion

Hence, like chord, MultiChord still requires information about O(log N) other nodes on the circle, but since now the traffic is distributed among various circles, 'n', is reduced and hence the overall performance of the system improves.

We can say, the effective performance if O(log M), where M is the average no of nodes on one circle. (Note that M will be less than N (total no of nodes in the system))

IV.References

[STO-1] ION STOICA, ROBERT MORRIS, DAVID LIBEN-NOWELL, DAVID R. KARGER, M. FRANS KAASHOEK, FRANK DABEK, HARI BALAKRISHNAN, Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications. ACM SIGCOMM (San Diego, CA, 2001) pp. 149-160

[SEA-2] EL-ANSARY, S. A Framework For The Understanding, The Optimization, and Design Of Structured Peer-To-Peer Systems. Licentiate Philosophy Dissertation, Royal Institute of Technology, 2003.

[OCS-3] KUBIATOWICZ, J., BINDEL, D., CHEN, Y., CZERWINSKI, S., EATON, P., GEELS, D., GUMMADI, R., RHEA, S., WEATHERSPOON, H., WEIMER, W., WELLS, C., AND ZHAO, B. OceanStore: An architecture for global-scale persistent storage. In Proceedings of the Ninth international Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2000) (Boston, MA, November 2000).

[MAH-4] MAHAJAN, R., CASTRO, M., ROWSTRON, A. Controlling the cost of reliability in peer-to-peer overlays, LNCS 2735, Proceedings of the Second International Workshop IPTPS 2003 (Berkeley), Springer, 2003.

[JLI-5] LI, J., LOO, B.T., HELLERSTEIN, J., KAASHOEK, F., KARGER, D.R., MORRIS, R. On the Feasibility of Peer-to-Peer Web Indexing

[HAR-6] HARREN, M., HELLERSTEIN, J.M., HUEBSCH, R., LOO, B.T., SHENKER, S., STOICA, I., Complex Queries in DHT-based Peer-to-Peer Networks. In The 1st International Workshop on Peer-to-Peer Systems (IPTPS'02), 2002.
http://www.cs.rice.edu/Conferences/IPTPS02/

[AND-7] ANDRZEJAK, A., XU, Z. Scalable, Efficient Range Queries for Grid Information Services. In 2nd International Conference on Peer-To-Peer Computing, pages 33–40, Linköping, Sweden, September 2002. IEEE Computer Society. ISBN-0-7695-1810-9.

[SCH-8] SCHLOSSER, M., STINEK, M., DECKER, S., NEJDL, W. A Scalable and Ontology-Based P2P Infrastructure for Semantic Web Services. In 2nd International Conference on Peer-To-Peer Computing, pages 104–111, Linköping, Sweden, September 2002. IEEE Computer Society. ISBN-0-7695-1810-9.

[DAB-9] DABEK, F., KAASHOEK, M.F., DARGER, D., MORRIS, R., STOICA, I. Wide-Area Cooperative Storage With CFS. In Proceedings of the 18th ACM Symposium on Operating Systems

Principles (SOSP '01), Chateau Lake Louise, Banff, Canada, October 2001.

[ZXU-10] XU, Z., MALLIK, M., KARLSSON, M. Turning Heterogeneity into an Advantage in

Overlay Routing. Technical Report HPL-2002-126R2, Hewlett-Packard Labs, July 2002. http://www.hpl.hp.com/techreports/2002/HPL-2002-126R2.html

[RAT-11] RATNASAMY, S., HANDLEY, M., KARP, R., SHENKER, S., Application-level Multicast using Content-Addressable Networks. In Third International Workshop on Networked Group
Communication (NGC '01), 2001. http://www-mice.cs.ucl.ac.uk/ngc2001/
[CAS-12] CASTRO, M., DRUSCHEL, P., KERMARREK, A.M., ROWSTRON, A., SCRIBE: A Large-Scale And Decentralized Application-Level Multicast Infrastructure. IEEE Journal on Selected Areas in Communications (JSAC) (Special issue on Network Support for Multicast Communications, 2002.

[TAN-13] TANENBAUM, A.S., VAN STEEN, M., Distributed Systems: Principles and Paradigms. Prentice Hall, Inc., 2002. ISBN-0-13-088893-1.

[CAN-14] RATSANAMY, S., FRANCIS, P., HANDLEY, M., KARP, R., SHENKER, S. A Scalable Content Addressable Network.Technical Report TR-00-010, Berkeley, CA, 2000

[PAS-15] ROWSTRON, A., DRUSCHEL, P. Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. Lecture Notes in Computer Science, 2218, 2001. http://citeseer.nj.nec.com/rowstron01pastry.html

[TAP-16] ZHAO, B.Y., KUBIATOWICZ, J.D., JOSEPH, A.D. Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing. U. C. Berkeley Technical Report UCB//CSD-01-1141, April 2000

[EPI-17] LEONG B., LISKOV B., Eric D DEMAINE,‖Epichord: Parallelizing the Chord Lookup Algorithm with Reactive Routing State Management‖, proceeding of 12th international confrenceon Networks 2004.

[D1H-18] MONNERE L., AMONRIM C.,,
—D1HT: A Distributed one hop hash table‖, in the Proc of 14th
IEEE Dexa, 2006