IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 16, Issue 1, Ver. II (Jan. 2014), PP 70-74
www.iosrjournals.org

Routing Protocols of Distributed Hash Table Based Peer to Peer Networks

S.Rajalakshmi[1], S.Balaji[2], T.Godwin Selva Raja[3]
(M.Tech-Final year) Department of Computer Science & Engineering, Christ College of Engineering and Technology (affiliated to Pondicherry University) Puducherry, India

***Abstract**: Distributed hash tables (DHTs) is an extremely attractive study theme during the part of P2P networks; such networks be fetching especially admired in functions similar to file sharing. The idea of the Distributed Hash Table is given that the technique to explore the resources (especially files) within a P2P network. A DHT protocol usually affords a solitary task to the P2P function: afford a key and find out the node (or may be nodes) which is responsible for such key [1][3]. Each and every one function (such as really recover the resource or storing the resource on the node afford for it) is offered by superior levels of the P2P function. In such article our objective is to discover the security measures and determine them on accessible routing procedures of such networks. The Chord [4] (a DHT protocol) is selected as the objective approach for a variety of reasons it resolve be enclosed in this paper.*
***Index Terms:** routing protocol, peer to peer networks, distributed hash table*

I. INTRODUCTION

DHT is able to achieve two of our most important wishes. DHT is the distributed data structures to clutch the key and the value as couples in the entirely distributed way. It as well set each key-value pair barely on the single or narrow node. To make a decision on which node an correct pair have to be hoarded we need a mapping method. The arrival and departure a node does not cause the remapping of all the keys. A precise hashing technique (consistent hashing) is worn in DHT to map the key. Such hashing technique divides the key in several divisions. This method works the distance model to map a key to an assured node. Distance is a logical feature and which is not associated to or enclosed to physical distance of the nodes in P2P network. In this network the node which is essentially in England may be nearer to the node in Japan than a node accessible in the same area. The mapping purpose will be in utilize while the placing of the new key-value pair into the hash table will get place and also as we need to search the key. This purpose uses simply the key to make a decision the suitable node which will clutch the pair. Behind that, if the same key will be raised, then the above mapping function will decide the position where the key is accessible, this procedure builds the revival of the value closer.

Distributed hash table (DHT) protocols permit resources to be positioned rapidly in decentralized distributed service systems. Resources are able to contain things such as files, index entries, chat messages, or any extra type of entity that can be hoarded on and recovered by nodes in a distributed system. A DHT consists of a set of contributing nodes, where each node keep up a tiny quantity of information about a subset of other nodes in the system and routes lookup requests through the system towards their destinations. Each resource has a key related with it. Given a key, a DHT can set the node liable for the related resource rapidly, normally within O(log n) hops, here n is the number of accessible nodes in the system. The amount of other nodes in the system that each node wants to be awake of is also usually O (log n). Well-known DHTs that established a vast sum of absorption consist of CAN, Pastry, and Chord.

II. ALGORITHM OF CHORD ROUTING PROTOCOL

Being statistical identifiers are provided for both nodes and keys in Chord approach. To obtain the key's identifier just hash that key by exacting hash function which is exploited by every nodes of the system that precede *m* bit integers. The node discovers its identifier by hashing of its IP address. Now these identifiers are planned on an identifier circle (ring) modulo 2^m. Each key's value is allocated to the first node whose identifier is equivalent to or pursue that key's identifier in the ring. This feature is demonstrated following figure 1.

In the Chord ring shown in figure 1[4], the hash bit length *m* is 6. There are 10 nodes in the network (shown with N prefixes followed by the node's identifier) and 5 keys (shown with K prefixes followed by the key's identifier) are being hoarded. Each key is exposed being hoarded on the first node that succeeds the identifier of key in circle, as point out by the arrows.

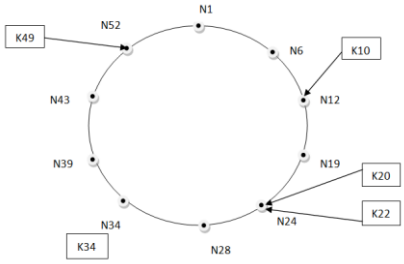

Figure 1: A design of keys mapping to nodes

Each node provisions some routing information to establish nodes which are lawfully reliable for keys. The Chord finger table for a node with identifier *id* holds *m* entries (0 to *m*-1 entries). For finger table entry *i*, the node hoards in that entry is the first node whose identifier succeeds $id + 2^i \pmod{2^k}$. It is probable to contain copy entries in the finger table. Figure 2 shows a model finger table with a design of how the finger table is derivative for node N6. N6's last finger table entry must be the node that succeeds $6+2^5$. This node is N38, so a reference to N39 is stored in the last finger table entry of N6's finger table. The rest of the finger table entries are crammed in with the same process for $i = 0, 1, 2, 3,$ and 4.

As figure 2 demonstrates, each node only has information about a subset of the nodes in the overall system. As the system obtain much larger, the number of exclusive nodes in each node's finger table becomes a smaller part of the overall number of nodes. The size of the finger table has been exposed by [4] to be O(log n) where *n* is the number of nodes in the system. The advantage of the routing table is that when performing a lookup we can bind about half of the remaining distance between the node doing the routing and the node answerable for the key. This divide and conquer technique to routing lookup requests has been shown by [4] to use O(log n) hops for each route. The algorithm for routing a lookup request from a node is simple: forward the request to the last routing table entry that leads the identifier of the key.

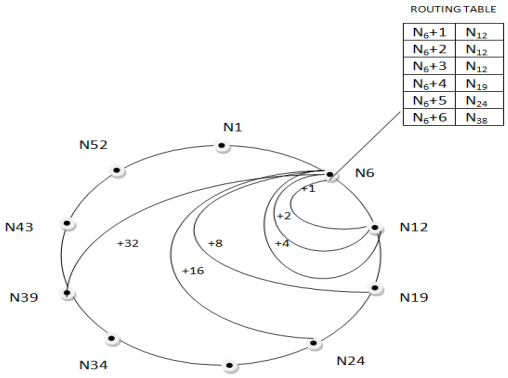

Figure 2: Example of routing table

The node leading the destination node will identify that the key cataract between itself and its successor and return information about its successor to the node performing the lookup. Figure 3 presents an example of the route a lookup request may take throughout a Chord system. In this figure, N6 is performing a lookup request for key K38. For a new node to join a Chord network, it needs to know of any one node that is already in the network.

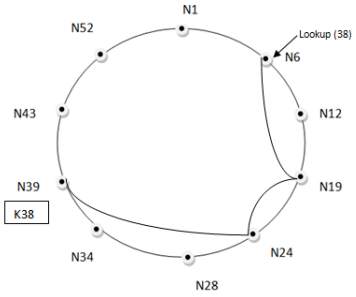

Figure 3: route taken by a lookup in a chord network

Finding a node that is previously in the network is done out of crew. The joining node will then use this "bootstrap" node to perform a lookup on its own identifier. The node returned by this lookup will be the new node's successor in the Chord ring. The new node will send a message to its successor alerting it that it is now that node's predecessor and the successor will inform its previous predecessor that the new node is now its successor. The new node will then use its successor to perform the suitable lookups to fill in its finger table. As nodes will be arriving and departure continuously, each node wants to occasionally re- perform these lookups in order to keep its router table up to date.

III. Vulnerabilities of Chord Attack

As DHT lookup requests rely on other nodes in the system to track the protocol properly, they are susceptible to various kinds of attacks ([4]; [5]). One type of attack is routing attacks, and this is the kind that this manuscript will hub on protecting against.

A routing attack happens when a node purposely dives lookup requests or forwards the lookup request to another node in a method that abuses the protocol condition. Instances of inaccurate forwarding would be to forward the request to nodes further away from the destination, to random nodes, or to other scheming cruel (malicious) nodes. Scheming cruel nodes may run a part Chord separation or a "sub-chord-ring" in a valid Chord network and confine lookup requests and forward them into this sub-chord-ring. This attack creates it appear as if lookup requests are being forwarded properly and it could even cause nodes joining the system to innocently join the cruel separation.

Another type of attack is an attack where the node liable for a key proceeds mistaken values for that key. It is hard for an attacker to aim specific keys in Chord as a cruel node's identifier is a hash of its IP address which services a node into a specific region of the network and creates it simple for other nodes to confirm that a node is using its right identifier.

It is gone to upper levels in the P2P function to confirm that the recovered data from nodes is right once the lookup process finishes effectively. Chord permits for a key's equivalent value to be hoarded on multiple nodes by using multiple hash functions to get multiple identifiers for keys. This paper will not hub on attacks where nodes liable for keys behave badly; as an alternative we focus on avoiding cruel nodes from minding lookups from getting the node(s) liable for them.

However a different technique of attacking a Chord network is for a bootstrapping node to bootstrap a joining node into a cruel network as a substitute of the planned network. Bootstrapping is out of crew, and there is tiny that can be completed if a cruel node is used to bootstrap. We will hence presume that the node worn for bootstrapping is hoped.

IV. Proposed System Methods

To alleviate routing attacks on Chord, we recommend the following main modifies to the protocol:

- As a substitute of lookup requests being forwarded from node to node, the node performing the lookup will openly make contact with each node and request the next hop on the route to the destination.
- Every hop will be confirmed for possible rightness by checking the geometrical distinct between node identifiers in the hop to numerical information about network solidity derived from the routing table of the node performing the lookup.
- If a hop is resolute to be unacceptable, the node performing the lookup will backpedal to the prior node on the route and ask for a different routing table entry.

Each one of these modify is illustrates in more aspect in the following sectors.

V. Routing In Source Node

In the Chord protocol, a node performing a lookup forwards the lookup request to the closest node in its routing table. As an alternative of forwarding our lookup request out into the un- trusted network, we will request each hop in the route for the next hop ourselves. This is possible in overlay networks as we can create a "direct" overlay association to any node on the physical network. This modify is uncomplicated to implement.

When we carry out lookups from the source, when we notice a cruel node beside our route we can go reverse to the last fine node on that route and request for an another next hop. Again, we cannot rely on other nodes to execute these events as other nodes are not hoped.

VI. Route Step Validation

Route hop certification is the for the most part significant modify being made to the protocol. The aim of hop authentication is to respond this issue: node a revisited a reference to node b as the next hop on the path to some key. Is this hop correct?

The major plan is to look at the reserve between the identifier of the next hop returned by node A and the "pointer" used by node A for the routing table entry of that next hop and establishes if this distance is liable known the solidity of the network. A routing table entry pointer for routing table entrance i of a node with identifier id is id + 2i (mod k). This is the identifier that a node looks up when it is satisfying in routing table entry i. We know that the router table pointer must drop between two nodes in the Chord ring, so the reserve between an entry's pointer and the identifier of the real node hoarded in that entry is fewer than the reserve between that node and its predecessor in the ring. By comparing the mathematical reserve among the entry pointer and the entry node's identifier to the average numerical distance between nodes, we can establish how likely it is that a node is using a appropriate node for a exacting routing table entry.

Each node will approximate the average mathematical distance between nodes in the ring from its own routing table. Nodes in this modified edition of Chord will hoard additional information about its routing table entries for this function. When performing episodic routing table updates, nodes will query the nodes in its routing table for the identifiers of those nodes' predecessors and successors. This gives a node up to two exclusive preserve samples per routing table entry. From these samples each node will calculate it's approximate of the average reserve between nodes and the standard variation of those distances from the regular.

VII. Protection Policy

The design of source routing, detecting malicious nodes on the path to the destination, and routing about those cruel nodes ought to obtain us to the legal destination for a known key.

If a node is only sinking lookup request, we can backtrack and go about it and maintain to the destination. If node A refers a lookup requests to inaccurate node B, we can prove that node B ought to not be current in the routing table entry revisited by node A. If a set of cruel nodes form a sub ring in the Chord network and run the Chord protocol along with them and only return routing references to other malicious nodes, the malicious network will have a lower node density than the rest of the Chord network and we will detect this as higher than expected node reserves when we query the cruel nodes. We can then route around these cruel nodes.

VIII. Common Plan Of Method

The customized Chord protocol is realized in a simulator written in Java. The simulator permits the user to run simulated Chord networks of unreliable sizes. The consumer is capable to decide how many nodes will be negotiated. The user will have the capability to denote the typical deviation factor, which is illustrates in section 2.2. While the system should contain the ability to test for cruel nodes that fall packets and revisit arbitrary mistaken next hops, the additional appealing check the method will run will be when a group of nodes are scheming by operation a sub-ring of cruel nodes and only recurred other nodes within this sub-ring as hops when getting a lookup request. The software simulator will execute lookups of arbitrary identifiers from random, non-compromised nodes in the system. The user will have the capability to indicate how many tests will be performed.

IX. Postulation And Experiments

This segment holds a list of supposition about the proposed system that we plan to explore, and how those assumptions will be experienced.

i. Nodes performing lookups must be capable to route approximately cruel nodes that basically drop lookup requests. This resolve be tested by changeable the division of cruel nodes that drop lookup requests, performing a quantity of lookups of arbitrary keys from arbitrary nodes, and copying the lookup success rate. The lookup success rate resolve then be conspired beside the division of cruel, lookup request dropping nodes.

ii. Nodes performing lookups must be able to perceive non-scheming cruel nodes that revisit false after that hops. This resolve be experienced in the same approach as hypothesis1, apart from the cruel nodes will be locate up to revisit random, inaccurate nodes during the lookup requests as an alternative of basically dropping the packet. The same data will be conspired. This has the consequence of testing the authentication and the backtracking algorithm, but this kind of attack ought to be easier for the authentication algorithm to perceive than the attack in the next hypothesis.

iii. Nodes performing lookups must be able to perceive conspires cruel nodes that are running a sub-ring and are recurring positions to further cruel nodes during the lookup method. Another time, this will be tested in the same approach as hypothesis1 and 2, apart from the cruel nodes will be operating an vary routing table that consists of only cruel nodes and via that routing table for the duration of lookup requests. This has the cause of testing the authentication algorithm under the attack system that is the most tricky to perceive.

As the number of nodes in the system increases, the victory rate in keep away from all three attacks illustrated over must raise. The information in use from the tests for i, ii, and iii will test this hypothesis. Tests resolve be performed.

X. Conclusion & Future Work

Roguish nodes in DHT's can establish rigorous disturbances, still when they only survive in tiny information. A number of security worries should be goal in classify to use DHT's in location where consumers cannot be hoped. In this manuscript, we proposed a method for extenuating the property of lone of those concerns: routing threats.

The future work of concerning this feature is, the method conversed at this point ought to move to further DHTs protocols that build exploit of embarrassed routing, and can serve up as a critical portion to a entire security result.

References

[1] Heinbockel, W., and Kwon, M.: Phyllo: A peer-to-peer overlay security framework. The First Workshop on Secure Network Protocols (NPSec), Boston, MA (2005)
[2] Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A scalable content addressable network. In: Proc. ACM SIGCOMM'01, San Diego, CA (2001)
[3] Saroliya Anil, Shrivastava Vishal: Analysis of Routing Attacks in Peer to Peer Overlay Networks, Jaipur, In: National Conference on "Recent Trends in IT: Opportunities and Challenges", S.S. Jain Subodh MCA Inst., Kukas, Jaipur (2010)
[4] Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for Internet applications. In: Proc. ACM SIGCOMM'01, San Diego, California (2001)
[5] Wallach, D.: A survey of peer-to-peer security issues, International Symposium on Software Security, Tokyo, Japan (2002)