Secure transaction on the Peer to Peer based Virtual Network

Anil Saroliya, Upendra Mishra and Ajay Rana

***Abstract*—Distributed Hash Tables or the DHT is a very crucial and attention seeking topic as far as the field of P2P network overlays is concerned; since the latter has set a new benchmark in the arena of file sharing. The use of DHTs in P2P network involves the cause of file searching within the network. The DHT protocol works by assigning a key to a single P2P function and finds the node or nodes associated to this key thereby completing the request for file search. Other functions involving the retrieval of information and its storage are facilitated by certain higher layers in the P2P network. Through the research made out in the paper, the goal is to find out various security issues related to the process and resolve them according to the routing protocols of the network. The Chord which is a DHT protocol has been taken as the target for research in this paper for certain reasons that will consequently be covered in the following.**

***Keywords*— Structured P2P networks, distributed hash tables, routing, security, backtracking.**

I. INTRODUCTION

A. Distributed Hash Table's

DHT comprises of certain data structures in a distributed manner that holds consecutive keys and their values as pairs. Each of the pair so arranged is put to a single (node) or limited number of nodes. A mapping technique determines the node onto which a particular exact pair has to be allocated or stored. The joining or disjoining of such nodes does not result in a cause for remapping of all keys at once. The Consistent Hashing Mechanism is used to map keys in case of DHTs that separate a particular key into many parts. The process so initiated employs the 'Concept of Distance' to map a specific key to the respective node. Here, Distance refers to a logical magnitude and is not confined to a physical distance between the nodes in the P2P network. This means that a node which is authentically located in Germany might be closer to the one located in China rather than in the same region (Germany). The mapping function is hence induced with the input of a fresh key-value pair into the hash table oars soon as we desire to search the key. This function makes use of the key in deciding which node will hold the subsequent pair. After this process, if the same key is asked again, then the mapping function resolves the place where the key is found available, thereby making the recovery of value; a quick process.

The DHT Protocols allow resources to be located quickly in the decentralized distributed systems. These Resources may be present in the form of files, directory entries, discussion messages, or other type of objects that can be stored on and retrieved by nodes in a distributed system [1]. A DHT comprises of a cluster of participant nodes, where each node holds information about a subset of other participating nodes in the system and routes lookup requests through the system towards their respective destinations. To every resource in the system, a key is specifically associated. With the help of this specific key, a DHT can easily locate the node allocated for the associated resource instantly, typically within O (log n) hops, where n is the number of available nodes in the system. Apart from these nodes, the number of other nodes in the system that each node needs to be aware of is also typically O (log n). CAN[3], Pastry, and Chord are counted among the DHTs that have received prominent concentration through the course of time and very well known in the regard.

B. Distributed Hash Table's

The chord protocol has been provided with individual numerical identifiers for both nodes and keys. To obtain the key's identifier we just need to hash that key by the particular hash function which is used by each node of the system that returns m bit integers. This identifier is so obtained by the node by hashing of its IP address. Now these identifiers are arranged on an identifier circle (ring) by the order of modulo 2m. Each of the key's value is now matched to the first node whose identifier is equal to that key's identifier in the ring. The aspect has been illustrated vividly in 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 (recognized by N) and 5 keys (recognized by K) being stored in the ring system as shown in figure. The arrangement and assignment of keys on the ring system is clearly depicted in the figure.

Each node holds some routing information to locate the nodes that are typically dependable for keys. The table of allocation so formed is called as the "finger table" in case of Chord.

In the above illustration, id consists of m entries (0 to m-1). The sequence so formed in a consecutive manner is id + 2i. There can be duplicate entries in the same as well. Figure 2 depicts the derivation of the finger table with Node N8. Hence the last entry in the finger table for this node would be calculated as 8+25. The sequence of entries of the finger table are given by i = 0, 1,2,3,4.

As per figure 2, each node holds some information about a particular or specific subset of other nodes in the ring system; therefore it is clear that with the increase in the density or size of the ring system, there would occur a significant decrease in the uniqueness of the number of nodes in each node's own finger table. This aspect has been again very clearly depicted in the figure below. The very advantage of this finger table lies in the fact that we can skip nearly and at least half of the remainder distance between the node responsible for routing and the one responsible for the key.

Manuscript received on January, 2013.

Anil Saroliya, Amity School of Engineering & Technology, Amity University Rajasthan, Jaipur, Rajasthan, India.

Dr. Upendra Mishra, Amity School of Engineering & Technology, Amity University Rajasthan, Jaipur, Rajasthan, India.

Dr. Ajay Rana, Amity University UtterPradesh, Noida, UP, India

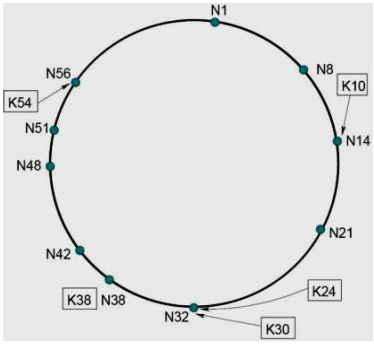

Fig. 1: An illustration of keys mapping to nodes.

The above mentioned approach is largely known as 'Divide and Conquer' and has been depicted in the figure below. It also depicts the use of O (log n) hops for each route. The algorithm so derived illustrates forwarding of the request to the last entry in the table that is actually the preceding entity of the identifier of the key.

The preceding node of the destination node now detects the key in between itself and its succeeding node. It now returns information about its succeeding node to the one doing the lookup. The same has been described vividly through figure 3, where N8 is doing a lookup for K54. A new node can certainly join a Chord network but only if it knows any one of the nodes already involved in the network. This process takes place outside the band. The node used by the joining node is called the 'Bootstrap node' and it performs the lookup process for the joining node's (new node's) own identifier. The resulting node after the lookup becomes the successor of the new node (newly joined node) in the ring system.

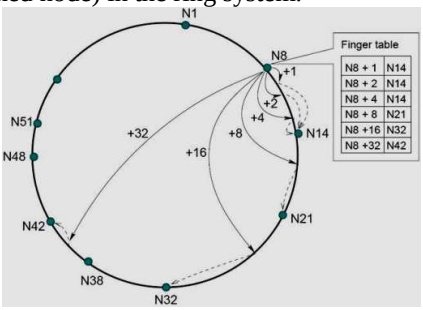

Fig. 2: An example finger table, taken from [4].

The new node now intimates the successor node about its new status as a predecessor and the successor now informs its previous predecessors that the new node (resulted after lookup) has now become its successor. And so on the process continues by performing lookups and filling in entries in the finger table.

Since nodes will be joining and leaving continuously, each node needs to periodically re-perform these lookups in order to keep its finger table up to date.

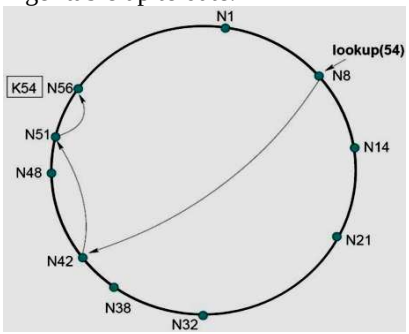

Fig. 3: An example of the route taken by a lookup in a Chord network, taken from [4].

C. *Chord Attacks & Vulnerabilities*

There are several attacks and vulnerabilities ([4]; [5]) under the DHT lookup request which rely on the other nodes too.

One type of vulnerability is routing attacks, and this is the category on which such paper is focused. Routing attacks happens when a node deliberately drops lookup requests or frontwards the lookup request to other node in a way that contravenes the protocol conditions.

There is also another type of attack, Colluding; in colluding wrong forwarding done by the intermediate node against the request targeted for actual destination. Such request can be forwarded to malicious or harmful node. Colluding malicious nodes might run a different Chord structure or a small-ring in an actual Chord group and detain incoming searching requests and move them in further direction into this small-ring. This attack appears like, as if searching requests are mover further correctly in forward direction, it could join the requesting node unknowingly to the malicious partition.

II. PROPOSED DEFENSE MECHANISMS

To make less intense the routing attacks on Chord, we propose an approach of backtracking and make the following key alteration to the protocol:

- The node which is executing the lookup will contact immediately to every node; as an substitute of lookup requests being move forward from node to node and make request the subsequently hop on the route towards destination.
- Every hop will be confirmed by the node which performing the lookup for possible precision by examining the variation among node identifiers in a hop to information regarding amount of network capacity consequent from the finger table.
- In a case when a hop is found as unacceptable as per above assumption, the node which generates the lookup request will backtracking the request to the preceding node on the path and request for other entries of DHT table.

A. *The concept of Backtracking*

Backtracking is a modification of the brute force approximation that methodically find for a clarification to a problem between all existing options. It perform as by assuming the solutions are characterized by vectors (v1, ..., vm) of values and by find the way, in a depth first mode, the domains of the vectors until the solutions are found [6][7].

In general, the hop chosen during the Chord routing process is the hop that most closely precedes the identifier of the key which is sought. In the proposed system, when a faulty node is detected during the routing process, it will fall back to the previous node on the route and use its next closest preceding node to the destination. This offers less progress, but gives us a way to route around the faulty node. If a node runs out of hops to give because it has no more nodes in its routing table that precede the destination, again it will fall back to the previous node on the route and use its routing table and repeat the process. In the following figure, it must be carefully understood that under what circumstances the nodes on the route to the destination must be bypassed. If the request is for a node's finger table and it contains a reference to what appears to be the destination node, it might be tempted to use this reference and bypass the rest of the routing process.

The issue with this method of bypassing is that nodes further away from the destination are more likely to have out

of date successor information, and the destination may have changed and that node has not yet called its fix_fingers() method. Therefore, use bypassing as a last option. Do not immediately bypass the rest of the nodes on a route when a node on the route knows of the destination. Instead, only bypass if that node has run out of any other hops that can be used.

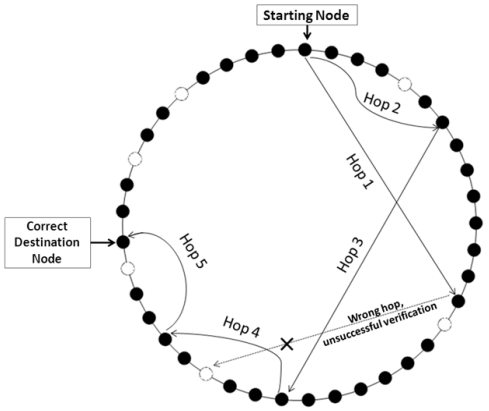

Fig. 4: The concept of backtracking.

B. Backtracking Algorithm

The following algorithm explains the working of Backtracking for securing the Chord transaction.

Algorithm steps of Backtracking:

From the staring node ask the node n to find the successor of available id:

Step 1 : **Start Algorithm,** starts at node **n**, to find the next hop for available id, current node-id, finger-table, and index-value.

Step 2 : Take a counter variable at initial stage, and check to find the id in the range of present node (node-id) finger table. Ask the node **n** to find the successor of available **id.**

Step 3 : If the index_value matches the requirement for suitable preceding node at first position of the finger_table then return the finger_table value for first closest preceding node to the requesting node else return nothing. At this stage bypassing is not introduced.

Step 4 : But if index value matches at second (and more than second) closest preceding node, reverse scanning is started for desired node and find the bypass_node.

Step 5 : Now take the bypass_node as starting node and repeat the Step 2.

Step 6 : If the request is fulfilled then requesting peer can easily move to the requested destination.

Step 7 : End Algorithm.

The above algorithm allows bypassing faulty nodes immediately preceding the destination. Since nodes often have many finger table entries for nearby identifiers, when moving closer to the destination it's a good chance of being able to find a node with the correct destination in its finger table, allowing the node to bypass faulty nodes preceding the destination.

III. EXPERIMENTS AND RESULTS

With the use of backtracking the enhanced secured Chord simulator is programmed to get the assessment in between ordinary peer to peer network & secured DHT oriented Chord protocol. The proposed solution is able to figure out the malicious node attacks and also agree to recognize such stoppage nodes which will not utilized anymore. Such experimentation is associated with the network of many nodes, among these nodes, some nodes are also malicious. For every dissimilar group of malicious nodes, varieties of results were experienced. Following tables and chart explain the relative analysis in between unsecure and secure results:

Table 1. Comparison between Secured Successful (SS) Lookups and Default Successful (DS) lookups

Nodes	SS Lookups	DS Lookups
100,20	85	50
200,40	74	53
300,60	76	45
400,80	79	40
500,100	67	52
600, 120	63	38
700, 140	66	35
800, 160	64	42
900, 180	66	46
1000, 200	53	4

Table 2. Comparison between Secured Failure (SF) Lookups and Default Failure (DF) lookups

Nodes	SF Lookups	DF Lookups
100,20	85	50
200,40	74	53
300,60	76	45
400,80	79	40
500,100	67	52
600, 120	63	38
700, 140	66	35
800, 160	64	42
900, 180	66	46
1000, 200	53	4

Here: e.g. - Nodes (100, 20) means 100 uncompromised nodes and 20 nodes are malicious nodes

In the following statistical investigation, it is cleared that amount of successs rate of secured lookups are higher than unsecured lookups in Chord protocol.

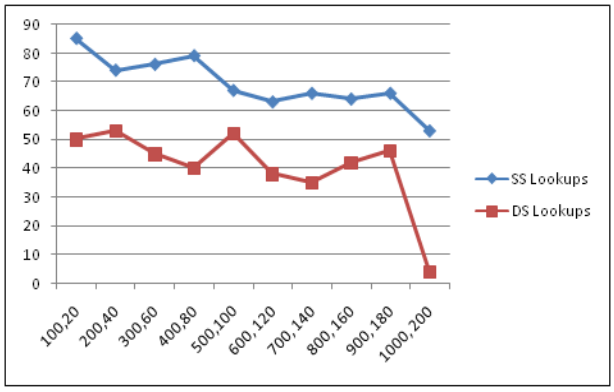

Fig. 5: Comparative analysis Secured-Successful (SS) Lookups and Default-Successful (DS) lookups

IV. CONCLUSION & FUTURE SCOPE

Just small number of malicious node can breach the security in DHT oriented P2P network. Number of security approaches has been targeted with respect to utilize DHT's in condition where users cannot be confident. In this paper, we proposed an algorithm of backtracking for justifying the things of routing attacks.

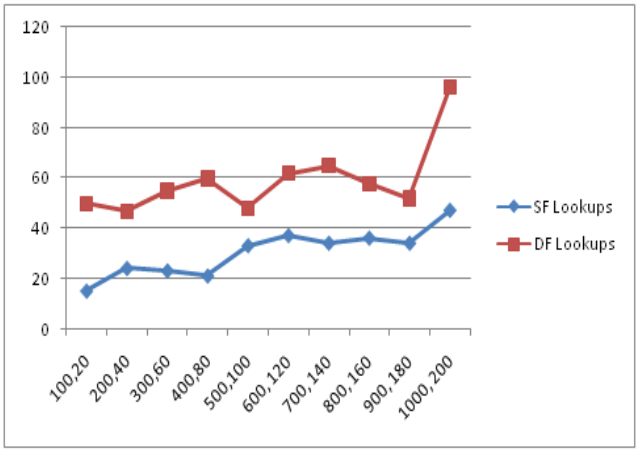

Fig. 6: Comparative analysis Secured-Failure (SF) Lookups and Default- Failure (DF) lookups

Due to the distributed property of structured p2p network it is really difficult to maintain the security but through efficient backtracking as proposed in this paper can improve the efficiency and exactness property of routing.

In the near future, much advancement will be made in this work and a secure solution will be implemented to eradicate such routing attacks.

Just small number of malicious node can breach the security in DHT oriented P2P network. Number of security approaches has been targeted with respect to utilize DHT's in condition where users cannot be confident. In this paper, we proposed an algorithm of backtracking for justifying the things of routing attacks. Due to the distributed property of structured p2p network it is really difficult to maintain the security but through efficient backtracking as proposed in this paper can improve the efficiency and exactness property of routing.

In the near future, much advancement will be made in this work and a secure solution will be implemented to eradicate such routing attacks.

REFERENCES

[1] Dehui Liu, Feng Chen, Gang Yini, HuaiMin Wangl, Peng Zoul: LSB-Chord:Load Balancing in DHT based P2P systems under Churn. In:Proc. IEEE ICCSIT'10, Changsha, China (2010)

[2] Heinbockel, W., and Kwon, M.: Phyllo: A peer-to-peer overlay security framework. The First Workshop on Secure Network Protocols (NPSec), Boston, MA (2005)

[3] Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A scalable content addressable network. In: Proc. ACM SIGCOMM'01, San Diego, CA (2001)

[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)

[6] Gurari, Eitan, Backtracking algorithms "CIS 680: Data Structures: Chapter 19: Backtracking Algorithms" (1999)

[7] Mariem Thaalbi, Nabil Tabbane, Tarek Bejaoui, Ahmed Meddahi: Enhanced Backtracking Chord protocol for mobile Ad hoc networks. In: Proc. IEEE ICCIT'12, Ariana, Tunisia (2012)