International Journal of Scientific Engineering and Research (IJSER)
www.ijser.in
ISSN (Online): 2347-3878
Volume 1 Issue 1, September 2013

Security in Distributed Hash Table in Peer to Peer Protocols

[1]CH. Sushma, [2]D. Navaneetha

[1, 2]Assistant Professor, Bhoj Reddy Engineering College for Women, Hyderabad, India

Abstract: ***Currently Distributed Hash Tables (DHTs) have a significant role in Internet and peer-to-peer protocols. However, DHTs also contain a number of security issues. This paper provides an overview of security research in context of DHTs from methodological point of view. We discuss about security issues and techniques in DHTs but the main focus is in research methods. Also, high-level model is presented to describe the research process in our context.***

Keywords: distributed hash tables, DHT, security, Sybil attack, peer -to- peer.

1. Introduction

Distributed Hash Tables (DHTs) are scalable and efficient way to implement a decentralized lookup service in a distributed system. DHTs are widely used for example in peer-to-peer networks. Unfortunately, Distributed Hash Tables are vulnerable for several kinds of attacks. In this paper we are focusing in security aspects of DHTs from methodological point of view.

As a distributed and highly scalable system, the Distributed Hash Tables are difficult target of practical study. It is extremely complicated to set up an authentic and realistic testing environment with millions of nodes all around the world. This is usually out of the question because lack of the resources as well. That is why computer simulations and mathematical modeling have a significant role in the research. Experimenting proof-of-concept implementations are usually done by simulating the physical network layer and running the real implementations on virtualized environment.

The purpose of this paper is not to analyze the security itself or provide new solutions. Instead, the goal is to give a clear overview about research methods in the context of security and Distributed Hash Tables. However, at first it might be useful to give a short explanation about security techniques and vulnerabilities in DHTs. To be able to understand the security we need to understand how DHT networks operate and what effects certain aspects of they may cause from security point of view.

2. History of Distributed Hash Table

A distributed hash table is, as its name suggests, a hash table which is distributed among a set of cooperating computers, which we refer to as *nodes*. Just like a hash table, it contains key/value pairs, which we refer to as *items*. The main service provided by a DHT is the *lookup operation*, which returns the value associated with any given key. In the typical usage scenario, a client has a key for which it wishes to find the associated value. Thereby, the client provides the key to *any* one of the nodes, which then performs the lookup operation and returns the value associated with the provided key. Similarly, a DHT also has operations for managing items, such as inserting and deleting items.

The representation of the key/value pairs can be arbitrary. For example, the key can be a string or an object. Similarly, the value can be a string, a number, or some binary representation of an arbitrary object. The actual representation will depend on the particular application. An important property of DHTs is that they can efficiently handle large amounts of data items.

Consistent Hashing which is a hashing scheme for caching web pages at multiple nodes, such that the number of cache items needed to be reshuffled is minimized when nodes are added or removed.

***PRR*2 or Plaxton Mesh** which is a scheme that enables efficient routing to the node responsible for a given object, while requiring a small routing table. Here we list the basic requirements and characteristics of DHTs

A. Scalability

The fundamental requirement for DHTs is scalability. DHTs should be scalable up to millions of nodes and even more key-value pairs. In practice this means that the search and storage complexity should not grow more than by magnitude of O (log N).

B. Decentralization

No central server exists, every node in the network is equally important

C. Availability

All the data should be available from any node in the network despite that the nodes are rapidly joining the network and exiting from the network. This also requires some replication.

D. Load balancing

Node identifiers and data items should be distributed in a way that every node would need to carry roughly equal amount of requests.

3. Distinguishing Features of DHTs

So far, the description of a DHT is similar to the domain name system, which allows clients to query any DNS server for the IP address associated with a given host name. DHTs can be used to provide such a service. There are several such proposals and it has been evaluated experimentally. The initial experiments showed poor performance while recent attempts using aggressive replication, yield better performance results than traditional DNS. Nevertheless, DHTs have properties which distinguish them from the ordinary DNS system.

The property that distinguishes a DHT from DNS is that the organization of its data is self-managing. DNS internal structure is to a large extent configured manually. DNS forms a tree hierarchy, which is divided into zones. The servers in each zone are responsible for a region of the name space. For example, the servers in a particular zone might be responsible for all domain names ending with .com. The servers responsible for those names either locally store the mapping to IP addresses, or split the zone further into different zones and delegate the zones to other servers. For example, the .com zone might contain servers which are responsible for locally storing mappings for names ending with abcd.com, and delegating any other queries to another zone. The whole structure of this tree is constructed manually.

DHTs, in contrast to DNS, dynamically decide which node is responsible for which items. If the nodes currently responsible for certain items are removed from the system, the DHT self-manages by giving other nodes the responsibility over those items. Thus, nodes can continuously *join* and *leave* the system. The DHT will ensure that the routing tables are updated, and items are redistributed, such that the basic operations still work. This joining or leaving of nodes is referred to as *churn* or *network dynamism*.

Another key feature of DHTs is that they are fault-tolerant. This implies that lookups should be possible even if some nodes fail. This is typically achieved by replicating items. Hence failures can be tolerated to a certain degree as long as there are some replicas of the items on some live nodes. Again, as opposed to other systems, such as DNS, fault tolerance and the accompanying replication are self-managed by the system. This means that the system will automatically ensure that whenever a node fails, some other node actively starts replicating the items of the failed node to restore the replication degree.

3.1 Overlay Networks

A DHT is said to construct an *overlay network*, because its nodes are connected to each other over an existing network, such as the Internet, which the overlay uses to provide its own routing functionality. The existing network is then referred to as the *underlay network*. If the underlay network is the Internet, the overlay routes requests between the nodes of the DHT and each such reroute passes through the routers and switches which form the underlay.

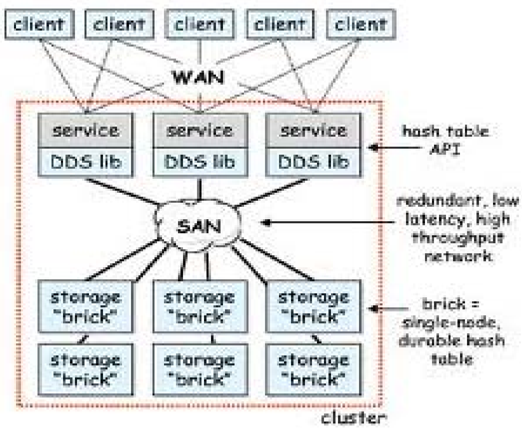

Figure 1

3.2 Routing Latency

The number of hops does not solely determine the time it takes to reach the destination, network latencies and relative node speeds also matter. Closely related to latencies are two properties called *content locality* and *path locality*. Content locality means that data that is inserted by nodes within an organization, confined to a local area network, should be stored physically within that organization. Path locality means that queries for items which are available within an organization should not be routed to nodes outside the organization. These two properties are useful for several reasons. First, latencies are lowered, as latencies are typically low within a LAN.

4. Properties of DHTs

DHTs are *scalable* because:

- Routing is scalable. The typical number of hops required to _nd an item is less or equal than log (*n*) and each node stores log (*n*) routing entries, for *n* nodes.
- Items are dispersed evenly. Each node stores on average *dn* items, where *d* is the number of items in the DHT, and *n* is the number of nodes.
- The system scales with dynamism. Each join/leave of a node requires redistributing on average *dn* items, where *d* is the number of items in the DHT, and *n* is the number of nodes.

5. Security and Trust

There are three main security issues related to DHTs mentioned in literature. These are called Sybil attacks, Eclipse attacks and Routing and Storage attacks. In Sybil attacks the idea is that an attacker generates large amount of nodes in the network in order to subvert the reputation system or mechanisms based on redundancy. These nodes do not necessarily need to be real physical computers but they can be virtual nodes controlled by single attacker. The

Sybil attack is not specific to DHTs but DHT is type of a system which is vulnerable to Sybil attacks. The vulnerability to Sybil attack depends on how cheap it to generate new nodes.

Eclipse attack is based on poisoning the routing tables of honest nodes. As there are many nodes joining and exiting from the DHT all the time, nodes need to actively update and synchronize their routing tables with their neighbors in order to keep lookup system functional. Thus, one malicious node can potentially poison many of its neighbors' routing table by providing false information. If the attacker possesses a "narrow" point in a network, he can utilize the Eclipse attack to potentially isolate the network in two parts.

Routing table and Storage attack is a type of an attack where a single node is not following the protocol. Instead of forwarding the lookup requests, it may drop the messages or pretend being the responsible of the key. Hence, it may provide corrupted or malicious data - such as viruses or trojan horses - as a response

Sybil attacks or Eclipse attacks do not directly break the DHT nor damage the other peers. They are more like tools for attacker to control the routing and data flow in DHT. Instead, Routing table and Storage attacks are something that actively tries to harm the network and the other peers. Thus, effective way to organize attack in DHT would be setting up a malicious node providing corrupted information and then utilizing Eclipse attack or Sybil attack to forward the requests to that node.

6. Security Mechanisms

Much research has been done in order to protect the peers and the whole network against existing security threats. We are not going in details here as the focus of this paper is not in the security itself but research methods. However, some practical examples of security solutions in DHTs are listed below.

1) Sybil attacks: As a defense against Sybil attack, there are several different approaches. Borisov proposes a challenge-response protocol based on computational puzzles. The idea is that every node should periodically send computational puzzle to its neighbors. Solving the puzzle "proves" that the node is honest and trustworthy, but it also requires CPU cycles. The goal is to make organizing Sybil attack more difficult: running one peer client does not require much of CPU power, but running thousands of active virtual nodes is computationally infeasible.

2) Eclipse attacks: An obvious way to shield against Eclipse attacks is to add some redundancy in routing. This approach is utilized by Castro who propose two separate routing table: the optimized routing table and the verified routing table.

3) Routing and Storage attacks: As an example, Ganesh and Zhao propose a solution where nodes sign "proof-of-life" certificates that are distributed to randomly chosen proof managers. The node which is making lookup request, can request the certificates from proof managers and that way detect the possibly malicious nodes.

Security needs to be considered for every distributed system, and DHTs are no exception. One particular type of attack which has been studied is the *Sybil attack* .The attack is that an adversarial host joins the DHT with multiple identities. One way to establish the identity of the nodes of the DHT is to use public key cryptography. Every node in the DHT is verified to have a valid certificate issued by a trusted certificate authority.

7. Conclusion

In this paper we studied security in Distributed Hash Tables from the methodological point of view. The goal of this paper was to present the most important research methods in context of Distributed Hash Tables and their security aspects. Another main objective was to give an overview how those methods are applied in practice. As a conclusion, according to existing surveys and research papers about our topic and what we learnt in previous chapters, the two most important and popular methods were computer simulations and data analysis. Those are utilized almost in every paper in this research field. DHT is basically an overlay network so it is obvious that the network simulation tools are major asset for a researcher. Data analysis is strictly connected to the simulations: as we have much data as outcome of the simulations, we need to understand that data and realize how certain values affect to security properties.

Experimental research is maybe not as big role as one might expect. Some examples exist where proof-of-concept implementations have been run in real environment. However, simulations are the primary tool for gathering data from network behavior. The main reason for this may be that building a real test bed with millions of nodes all around the world is quite demanding and requires huge amount of resources as well. Also, simulations generally give us relatively good picture of network behavior.

References

[1] G. Urdaneta, G. Pierre, and M. V. Steen, "A survey of dht security techniques," ACM Comput. Surv., vol. 43, pp. 8: 1–8:49, February 2011. [Online]. Available: http://doi.acm.org/10.1145/1883612.1883615
[2] H. Sitepu, C. Machbub, A. Langi, and S. Supangkat, "Unohop: Efficient distributed hash table with o(1) lookup performance," in Broadband Communications, Information Technology Biomedical Applications, 2008 Third International Conference..
[3] Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in Proceedings of the ACM SIGCOMM '01 Conference, San Diego, California, August 2001.

[4] Singh, T. Wan Johnny Ngan, P. Druschel, and D. S. Wallach, "Eclipse attacks on overlay networks: Threats and defenses," in In IEEE INFOCOM, 2006.
[5] M. Naor and U. Wieder, "A simple fault tolerant distributed hash table," in Peer-to-Peer Systems II, ser. Lecture Notes in Computer Science, M. Kaashoek and I. Stoica, Eds. Springer Berlin / Heidelberg, 2003, vol. 2735.

Author Profile

CH. Sushma received the B. Tech and M. Tech in Computer Science and Engineering from JNTU in 2007 and 2009 and working as Assistant Professor in Bhoj Reddy Engineering College for Women, Hyderabad, India

D. Navaneetha received the B. Tech and M. Tech in Information Technology from JNTU in 2005 and 2012 and working as Assistant Professor in Bhoj Reddy Engineering College for Women, Hyderabad, India