Towards an Emergency Domain Name System Based on a Peer-To-Peer Network

Carolina Del-Valle-Soto, Iván Razo-Zapata, and Carlos Mex-Perera

Center for Electronics and Telecommunications
ITESM, Campus Monterrey
Av. Eugenio Garza Sada 2501 Sur, Col. Tecnológico
Monterrey, N.L., CP 64849 Mexico
{carolinadvs@yahoo.com.mx,carlosmex,razozapata}@itesm.mx

Abstract. In this work the performance of a Domain Name System (DNS) over P2P (Peer-to-Peer) networks is studied. We propose a double ring structure of P2P networks to provide a DNS. The architecture is called eDNS (emergency-DNS). eDNS can be used as an emergency mechanism in case of loss of availability of the original DNS. eDNS provides security and robustness for finding information under attacks. The performance of the proposed architecture was measured with the percentage of lost queries and the average number of hops in each query considering scenarios with a number of compromised nodes. Simulations show how the performance of the DNS is affected under attacks. According to the results, it is observed that the eDNS architecture overcomes the damage caused by attacks improving the availability of the DNS resources.

Key words: Domain Name System, Peer-To-Peer, DNS

1 Introduction

The main purpose of the DNS is to translate domain names into IP addresses. When a user wants to download a web site, he/she types in the Internet browser the corresponding domain, *i.e.* www.google.com, then the DNS will try to obtain the IP address. The DNS structure is based on a hierarchical inverse tree model (see Figure 1), where the origin is known as "root". Below that level, there are domains defined as TLDs (Top Level Domains). Such domains are classified as gTLDs (generic TLDs such as .com, .net, .org, .edu, .gov, .mil) and ccTLDs (country-code TLDs such as .uk, .se, .es, .mx). In order to have data origin authentication and authenticated denial of existence of domain names for DNS, Domain Name System Security Extensions (DNSSEC) can be used. DNSSEC was designed to protect from attacks like DNS cache poisoning [2].

Although DNS is a distributed scheme, it could be affected by some kind of attacks trying to damage critical TLDs, therefore, attenuating the performance.

Received 01/07/08
Accepted 01/08/08
Final version 08/09/08

In this way, it is desirable to apply another distributed schemes, such as Peer-to-Peer (P2P) architectures for supporting and protecting the DNS.

The hierarchical DNS is highly sensitive to latency and it has a significant challenge as to serve efficiently. In addition, the same hierarchical organization makes DNS have a disproportionate burden on the different levels of the hierarchy. The higher nodes are highly susceptible to a denial of service attacks, which causes a vulnerable security system as a whole. Another problem is that DNS servers are required for domestic outlets, incurring high administrative costs because DNS administration has to be manual [11]. That is why we want to establish a hybrid proposal that combines the concepts of a P2P network efficiently to improve the shortcomings that presents the DNS and thus, obtain a search more quickly and securely through a network bit vulnerable. So, our motivation is to make a robust architecture against attacks that combines the advantages of DNS with the advantages that presents a P2P network. This can control various DNS attacks such as denial of existence. Thus, an attack on the root of the DNS or any attack on the second level of the hierarchy, such as domain TLDs can be compensated through a double tier P2P-based structure. Then, this structure is called eDNS and it will be shown that can be a good mechanism for scenarios such above where a fast, secure and reliable DNS is desirable.

P2P architecture is a growing concept in the world of network technologies and the Internet. In a P2P system, distributed computing nodes of equal roles or capabilities exchange information and services directly with each other. The goal of a data-sharing system is to support a search protocol and retrieve data found on user caches. The information searches are a fundamental and problematic aspect of P2P environments because they aim for ensuring that there is not a single point of failure and offering scalability and dynamism [3].

P2P systems can be classified into 3 categories: *Centralized* systems , which have a central directory server to which users (clients) make requests. However, these systems have a single point of failure which makes them highly vulnerable [6]; as an example of these systems is Napster. The *decentralized* systems do not have a central server, but on the contrary, the nodes form a network among themselves and thus petitions are sent. Among decentralized networks designs, some are *structured*; in which there is a close link between P2P topology and the location of data; for example, Pastry [9], Tapestry [14], Skipnet, CAN [8] and Chord [10]. Other decentralized P2P systems are *unstructured*; where there is no link between the topology and location of information [1] that is, there is no precise control between the topology of the network and the place where the data in it, for instance Gnutella and Freenet.

This work is carried out under an architecture platform defined by Pastry, using DHT (Dynamic Hash Tables), where each node has a unique identifier assigned randomly, each object has also allocated a selected identifier and it is stored in the node whose identifier is numerical closest to the object ID, called *"home node"*. The routing is done docking with the prefix resource identifier node. Each request is traveling across the network with a certain number of hops up to find the best coupling of the digits of their ID with a specific node.

Thus, the search space is reduced exponentially. Then, the petition routes is under *O(log(b)N)*, where N is the number of nodes in the table and b is the numerical basis used in the system [11].

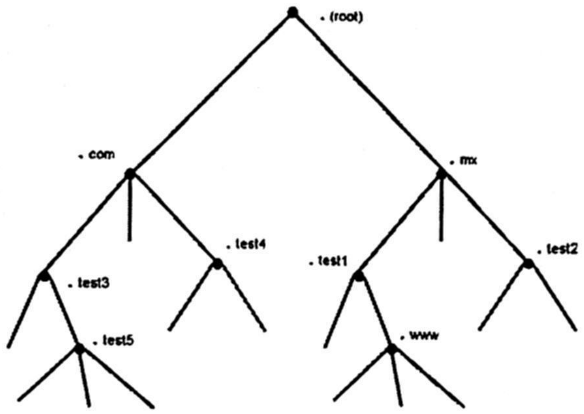

Fig. 1. Structure DNS Tree.

But the most problems with P2P data-sharing is that the networks adds overload, latency and low security. Besides it is susceptible to malicious activity, an attacker peer can insert itself into the network at the various points or a single point and forwards a lot of queries, or discard them altogether too. Also, a secure P2P network is challenging because the network topology is dynamic and nodes are on-line and off-line very often times. Thus we have problems like signatures to verify, legitimate packets and drop packets that do not pass the verification because P2P networks are very susceptible to malicious activity like previously mentioned. For that reason we propose a double tier structure for a better distribution of information combined with DNS sub-trees, this is a hybrid system that takes advantages of both strategies, centralized and decentralized schemes [12].

The rest of this paper is structured as follows. In the next sections, we describe related work and the concepts with peer-to-peer architecture over two tiers with DNS. After that, we present the proposal and then we evaluate the performance of the network through simulations and give results considering scenarios with a number of compromised nodes. Afterwards contributions are summarized. Finally, last section depicts the future work.

2 Concepts

Neither fixed clients nor fixed servers are present into a P2P network, but it contains nodes that behave both as clients and servers. P2P networks manage and optimize the use of bandwidth of all users leading to a better workload balance. Thus, this result gives more efficiency in connections and transfers than with some centralized conventional methods where a small number of servers provides total bandwidth and shared resources for a service or application. Among the most common and potential applications of P2P networks are sharing and search files (most widespread application of such networks), distributed file systems, Internet telephony systems, alternatives to the conventional distribution of films and television programs and scientific calculations which process databases, among others [1].

Some of the most common topologies for networks are: star, mesh, backbone, ring, and combinations between them. In this work the P2P networks are based on ring topology, where the nodes are connected in a closed circuit without a central system in the network, like Figure 2. In addition, P2P networks can work under protocols such as: Tapestry, Pastry, CAN and Chord [8–10, 14], which define how the information is distributed and how the searches must be performed. For instance, in our proposal we have used Pastry.

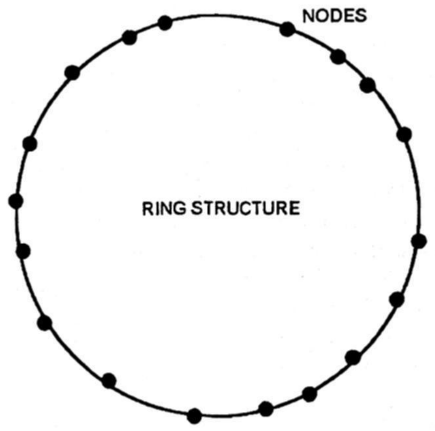

Fig. 2. Structure of a P2P ring.

In recent studies, it was shown that P2P networks were very much employed to share information or give hints as to where said information could be found. In the work presented by Pfeifer *et al.* [5], he shows a more specific architecture for P2P networks, either using DHT tables or hash functions is shown. Besides, information organization and safeguard is achieved in a much easier way through node creation. These may, without harm to the system, connect or disconnect themselves on a random basis. Data and network information caching is often used to reduce response time in P2P networks and this is where the problems of cache attack or cache poisoning become significant. In the work by Yang and Garcia-Molina [12], the respective benefits and drawbacks of centralized and decentralized forms of P2P architectures are presented. As it can be seen in the article, the centralized form of P2P networks has some serious disadvantages, the most conspicuous appear when a node changes places or disconnects itself. The other nodes (except for the central one) are not notified and the queries will receive neither up to date information nor a reliable copy of it.

3 Related Work

Previous investigations were focused on resource record security in DNS and they helped to make the DNS tree safer and more reliable. The article by Ayumu Kubota *et al.* [4] refers the implementation of DNS in a P2P network without loss of hierarchy. Also, it referred to a topology with several tears in a hierarchical tree form that resembles to DNS structure with important features such as redundancy and number of hops to be made to reach a destination, and here is where it comes to a close similarity with our work because the replication in areas seeking to minimize the number of hops to get to satisfy a query. In addition, it refers to possible attacks that can be made on the network, such as a malicious node can know the well-defined destination and therefore it can know their neighbors and thus, all of the neighborhood.

The article by Ramasubramanian and Sirer [7] studies the security problem as an improvement in a specific network rather than as a mandatory requirement for the structuring of that network. Thus, here DNS plays an important role. Besides, this network is better structured thanks to DHT tables. In our work are also used DHT tables for quick search of information, plus giving a better organization of nodes in the ring.

The studies that have been made regarding P2P networks focused their investigation on analysis of a single ring, however, the work presented by Haiyun Luo *et al.* [13] mentions a dual structure that operates separately . So what we want to achieve with our work is maintaining a double-ring, where the two rings operated jointly and the workload of the petitions does not fall on one of them, but they share the job features such as: information capacity , nodes, time to answer, among others. In short, in our proposal we want to provide an efficient solution of two rings that operate in tandem distributing the burden on the network. In addition, we exploit the advantages that provides DNS regarding security and the possible future use of DNSSEC.

4 Proposal

We propose to use an hybrid system where elements of both, pure P2P and client/server systems coexist. Currently, hybrid file-sharing systems have better performance than pure systems because some tasks (like searching) can be done much more efficiently in a centralized manner. We propose a two-tier design with application layered on pastry and dynamic hash tables (DHTs), in our assessment, would comply with this goal. Besides, we distribute the information according to popularity of DNS resources and in this manner we balance the workload among the nodes of both tiers. Also, with this design we provide decentralization and self-organization.

Thus, our proposal consists of the arrangement of two rings which will be the topology for eDNS. In such a way, these rings form two P2P networks in which actions such as data sharing and data update are performed. One tier has servers and the other tier has clients and both tiers are connected. For simplicity, we will refer them as servers and clients rings, respectively. Note that these names come from the original roles of such nodes in the classical DNS, however once they are working in a P2P architecture all of them must behave as peers. Consequently, the workload balance will be improved, once the nodes that form the ring will have replicates of the resources that are more commonly requested. Thus, allowing for more efficient and faster request solution.

The servers ring consists of nodes that are responsible for "small" DNS trees. This helps to improve the information search as it can take advantage of the DNS can employ, for example, DNSSEC for greater security.

Figure 3 shows how eDNS works. When a client tries to solve a domain name, it sends a query to the clients ring, if no answer is given then the query is send to the servers ring. Due to DNS sub-trees in the servers tier, this scheme is more reliable than an architecture with only clients ring.

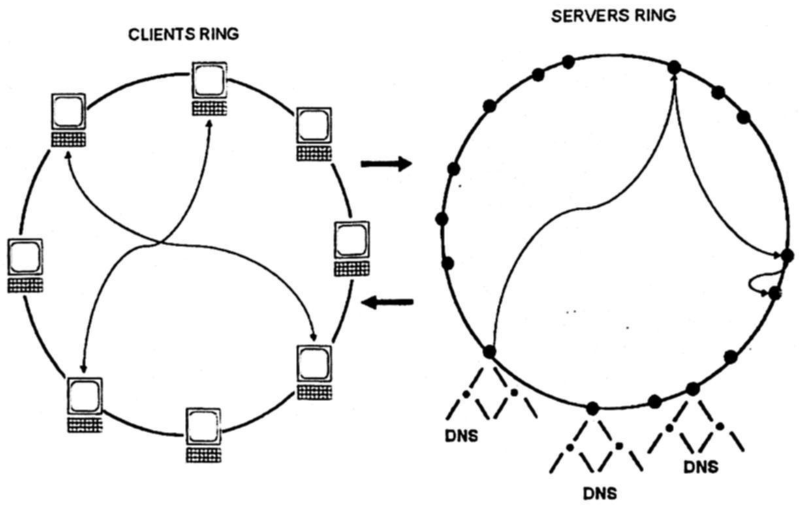

Fig. 3. Clients and servers organize to form double peer-to-peer network with DNS.

Figure 4 shows resources organization in the P2P tier, which is explained with an example, we have a specific domain: *example.com*, and its hash id is 0121, which will be located in a node having a hash id close to itself. So, resources are located according to their neighbors' hash id to form a logic P2P tier. Besides, in Figure 3 we can see the relationship between servers and clients, and servers with DNS tree.

In the servers ring each node is an "DNS island", *i.e.* is a small DNS tree. These nodes are created as characteristic domains based on the ccTLDs (country domains, such as .mx, .co) and gTLDs (generic Internet domains for organizations such as: .com, .net). The advantages that this form of ring organization has is that it provides delegation of areas and the possibility of establishing "islands of trust", all these characteristics of DNSSEC. Then, these trees in each node allow greater ease of finding information, better organization of resources and the possibility of using characteristics and advantages of DNSSEC. As disadvantage may be that keys and information updates could be made only through each island of trust independently.

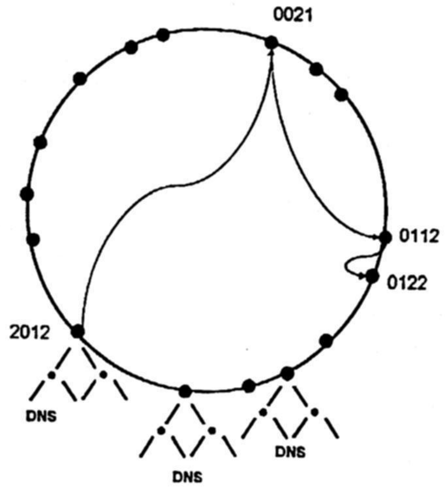

Fig. 4. Self-organization with DHT tables.

Note that servers nodes are normally more robust, while clients belong to computer users, who frequently connect and disconnected all the time.

5 Simulations and Results

5.1 Preliminary Considerations

Due to the importance of illustrating the performance of the eDNS architecture, it was necessary to feed the eDNS with realistic petitions. Therefore, DNS petitions were captured from one DNS server which belongs to one university network. These petitions were captured in the DNS server, and subsequently the set of petitions was computed for getting knowledge about the required domain names and their frequency.

In this way, two files were yield. The first one with the set of petitions sorted by their arriving time (*Time-DNS*), and the second with 13818 domain names sorted by their frequency (*Frequency-DNS*). Consequently, the eDNS was built using the information contained into the second file and the petitions into the eDNS were performed following the same distribution as in the first file.

5.2 Building the eDNS architecture

The DNS information is distributed into the clients tier using DNS lists. Therefore, each node has a list with domain names. Since both, the store and search of domain names into the ring are performed using the Pastry protocol, each node has domain names whose hash id is similar to the node's hash id.

On the contrary, in the server tier, DNS information is stored using the concept of gTLDs and ccTLDs. For domain names with gTLD and ccTLD, such as *www.google.com.mx*, one node will be built using the combination of gTLD, ccTLD and the domain name, in this way the node will have the *google.com.mx* domain. On the other hand, if the domain name has only either gTLD or ccTLD, such as *www.mty.itesm.mx* or *www.network.ieee.org*, the domain name will have only the combination of the domain name with the corresponding gTLD or ccTLD, in this case the nodes will have *itesm.mx* and *ieee.org* respectively. Therefore, the resources that are common domain are stored in a single node, forming a DNS tree. This allows a more quick efficient search, and the possibility of exploiting DNSSEC advantages.

The petitions were conducted with the ring of clients and servers operating jointly. The basic operation of this scheme is as follows: Whenever a query arrives to the eDNS, the query is passed to the clients ring which responds either with a positive answer or with a negative answer. If the clients ring responds positively so the resource was reached, on the other hand if the answer was negative, the clients ring passes the query to the servers ring and the servers ring will respond to the petition.

Furthermore, in order to test the robustness of the eDNS, some malicious nodes were introduced into both rings. The coalitioned nodes are malicious nodes that drop petitions, consequently they make that information can not pass through them, *i.e.* they are a barrier in the searching process and also play like failures into the eDNS architecture.

5.3 Results

Figure 5 shows the percentage of lost queries according to the percentage of coalitioned nodes in each tier, for instance: 0, 0.03, 0.3, 3, 30 and 45 percent of coalitioned nodes. The blue-triangled line shows the performance of the clients tier and the red-circled line shows the performance of the servers tier. As observed, the clients ring lost more messages than servers ring. It is because server-ring nodes clusters more domain names in their sub-trees. For example, if the node with the domain name *mty.itesm.mx* in the clients ring was lost, also the petition asking for *mty.itesm.mx* will be lost, and the message will be redirected to the node with the *itesm.mx* domain in the servers ring.

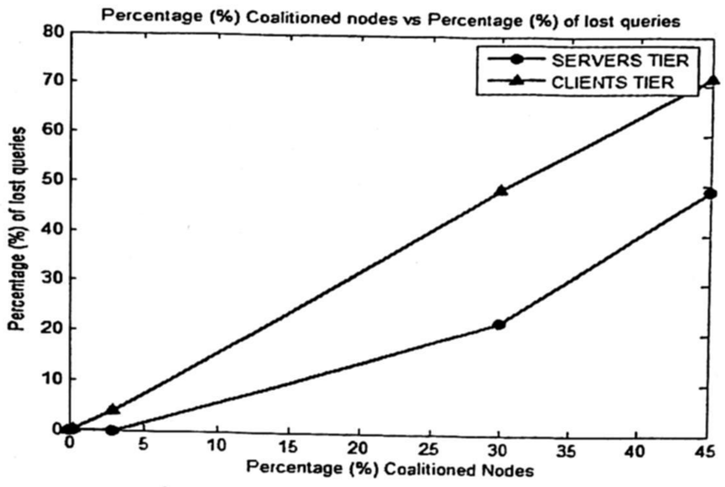

Fig. 5. Percentage of lost queries

On the other hand, Figure 6 shows the average number of hops in the clients tier (triangled line) and the servers tier (circled line) for an equal number of coalitioned nodes, including: 0, 0.03, 0.3, 3, 30 and 45 percent of coalitioned nodes. Also this figure depicts a tradeoff between clients ring activity and servers ring activity, therefore the less activity in clients ring the more activity in the servers ring. In this way, whenever a client-ring node gets a negative answer for a petition, it will ask for the domain name to the servers ring, so increasing the workload in the servers ring.

Figure 7 shows tables 1, 2, 3 and 4. Tables 1 and 2 show the percentage of lost queries according to the percentage of coalitioned nodes in each tier, as well: 0, 0.03, 0.3, 3, 30 and 45 percent of coalitioned nodes. On the other hand, Tables 3 and 4 show the average number of hops in the clients tier and the servers tier for an equal number of coalitioned nodes. Table 1 presents the result of a test with the more frequent resources, *i.e.* more popular requests, and where the nodes in the clients tier share their caches and no shared caches and it shows the percentage of lost queries. In table 2 results obtained from a test with less frequent resources are given, *i.e.* less popular requests, and where the nodes in the clients tier share their caches and no shared caches and it shows the percentage of lost queries. Table 3 shows a test with more frequent resources and where the nodes in the clients tier share their caches and no shared caches

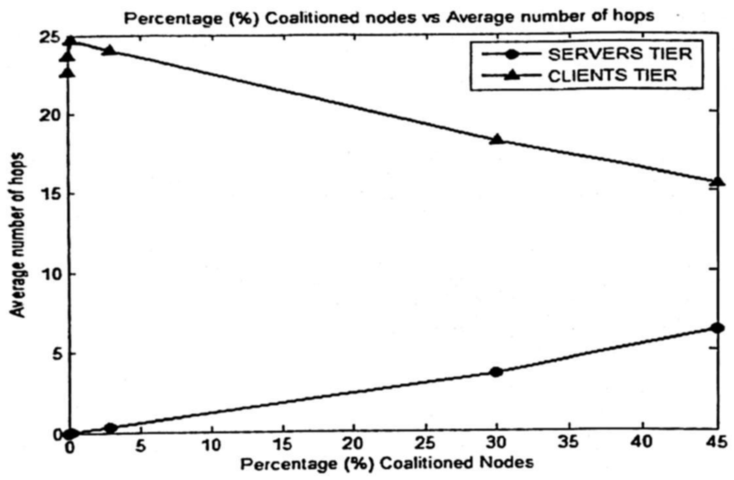

Fig. 6. Percentage of lost queries

and it reports the average of number of hops. Table 4 shows a test with less frequent resources and where the nodes in the clients tier share their caches and not share caches and it presents the average of number of hops.

6 Conclusion

As findings of this study we can say that regarding the results, the client tier always has the largest number of hops of lost queries. This is because the server tier nodes has only two domain levels, maximum three, which makes the search for more general information. Analyzing comparatively tests with more popular and less popular domains with shared caches and not shared caches (Tables 1, 2, 3, 4), we can see that when nodes do not share their caches the delay in meeting a request becomes larger, this shows that the number of hops increases for the graphic when the cache was not shared between the nodes.

7 Future Work

As future work we intend to make an architecture that responds to changes in the cache nodes and also to be robust in terms of requests for emergency. We will try the issue of denial of existence as a form of direct attack to DNS. Through a complete interplay of the two P2P rings will be offset the burden on the network as best as possible. Also we want to do experiments with the cache nodes, making the shared cache to work by geographic regions and the possibility of generating preferences on users. It is also expected to take advantage of the

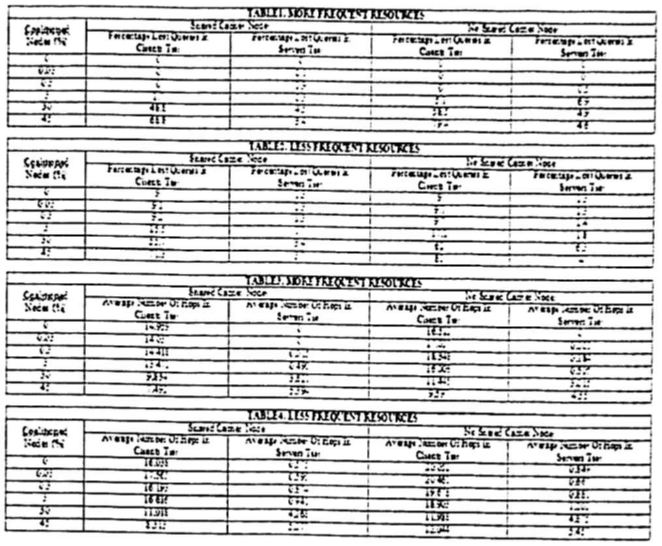

TABLE1. MORE FREQUENT RESOURCES				
[illegible] Nodes (%)	Shared Cache Node		No Shared Cache Node	
	Percentage Lost Queries In Client Tier	Percentage Lost Queries In Server Tier	Percentage Lost Queries In Client Tier	Percentage Lost Queries In Server Tier
0	0	0	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	41.1	[illegible]	[illegible]	[illegible]
[illegible]	61.1	[illegible]	[illegible]	[illegible]

TABLE2. LESS FREQUENT RESOURCES				
[illegible] Nodes (%)	Shared Cache Node		No Shared Cache Node	
	Percentage Lost Queries In Client Tier	Percentage Lost Queries In Server Tier	Percentage Lost Queries In Client Tier	Percentage Lost Queries In Server Tier
0	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

TABLE3. MORE FREQUENT RESOURCES				
[illegible] Nodes (%)	Shared Cache Node		No Shared Cache Node	
	Average Number Of Hops In Client Tier	Average Number Of Hops In Server Tier	Average Number Of Hops In Client Tier	Average Number Of Hops In Server Tier
0	[illegible]	0	[illegible]	0
[illegible]	[illegible]	0	[illegible]	[illegible]
[illegible]	14.411	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

TABLE4. LESS FREQUENT RESOURCES				
[illegible] Nodes (%)	Shared Cache Node		No Shared Cache Node	
	Average Number Of Hops In Client Tier	Average Number Of Hops In Server Tier	Average Number Of Hops In Client Tier	Average Number Of Hops In Server Tier
0	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Fig. 7. Percentage of lost queries

characteristics of DNS along with DNSSEC for changing keys on a chain of trust, providing greater robustness and reliability of the information. Further simulations will work with a more real cache, where nodes have characteristics in common and can be related under specific parameters. All this will lead to a more wisely exploit of the advantages that P2P network offers in conjunction with the advantages of having a DNS chain of trust, where there cryptographic keys and information are updated.

Acknowledgment

Authors would like to thank to Catedra de Biometricas y Protocolos Seguros para Internet and Google.

References

1. Scott Shenker Edith Cohen. Replication strategies in unstructured peertopeer networks. *SIGCOMM'02*, agosto, 2002.
2. Yong Wan Ju, Kwan Ho Song, Eung Jae Lee, and Yong Tae Shin. Cache poisoning detection method for improving security of recursive dns. *ICACT*, 2007.
3. Junjiro KONISHI, Naoki WAKAMIYA, and Masayuki MURATA. Design and evaluation of a cooperative mechanism for pure p2p file-sharing networks. *IEICE TRANS. COMMUN.*, pages 2319–2326, September 2006.

4. Ayumu KUBOTA, Yutaka MIYAKE, and Toshiaki TANAKA. Secure host name resolution infrastructure for overlay networks. *PAPER Special Section on Networking Technologies for Overlay Networks, IEICE TRANS. COMMUN.*, 2006.
5. G. Pfeifer, C. Fetzer, and T. Hohnstein. Exploiting host name locality for reduced stretch p2p routing. *Sixth IEEE International Symposium on Network Computing and Applications (NCA 2007)*, 0, 2007.
6. Edith Cohen Kai Li Scott Shenker Qin Lv, Pei Cao. Search and replication in unstructured peer-to-peer networks. *Department of Computer Science, Princeton University.*
7. V. Ramasubramanian and E. Gun Sirer. Beehive: O(1) lookup performance for power-law query distributions in peer-to-peer overlays. *Dept. of Computer Science, Cornell University.*
8. Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content-addressable network. In *ACM SIGCOMM'01.*, 2001.
9. Antony Rowstron and Peter Druschel. Pastry: Scalable, decentralized object object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM.*, 2001.
10. Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *ACM SIGCOMM'01.*, 2001.
11. Emin Gun Sirer Venugopalan Ramasubramanian. Beehive: O(1) lookup performance for power-law query distributions in peer-to-peer overlays. *Dept. of Computer Science, Cornell University.*
12. Beverly Yang and Hector Garcia-Molina. Comparing hybrid peer-to-peer systems. *Proceedings of the 27th VLDB Conference*, 2001.
13. Hao Yang, Haiyun Luo, Yi Yang, Songwu Lu, and Lixia Zhang. Hours: Achieving dos resilience in an open service hierarchy. *Computer Science Department, University of California, Los Angeles.*
14. Ben Y. Zhao, John Kubiatowicz, and Anthony D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, U.C. Berkeley, April 2001.