DYNAMIC PEER-TO-PEER (P2P) SOLUTION TO COUNTER MALICIOUS HIGHER DOMAIN NAME SYSTEM (DNS) NAMESERVERS

Marwan Abu-Amara, Farag Azzedin, Fahd A. Abdulhameed, Ashraf Mahmoud, Mohammed H. Sqalli*

Computer Engineering Department, *Information and Computer Science Department
King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia
{marwan, fazzedin, fahd, ashraf, sqalli}@kfupm.edu.sa

ABSTRACT

The Domain Name System (DNS) is a critical service in the Internet infrastructure which provides user-friendly name to Internet IP address mapping. The absence of the DNS has a severe impact on several Internet services such as email. To avoid an intentional blocking from a malicious higher name server, a dynamic round-robin peer-to-peer (P2P) solution that is built over the Chord protocol is proposed as a secondary path to resolve the DNS queries. An evaluation of the proposed solution based on load balancing, failure, timeout, and number of hops is also conducted through simulation experiments.

Index Terms— Intentional blocking, malicious nameserver, P2P, round-robin, Chord protocol

1. INTRODUCTION

Domain name system (DNS) plays a major role in the Internet connectivity by mapping a user-friendly host name to an IP address that can be dealt with by network devices. The DNS consists of a hierarchy of DNS servers with 13 root nameservers at the top of the hierarchy. The IP addresses of the 13 root nameservers are hard coded in root hints file in every recursive or caching DNS server [1]. DNS provides a critical and a core service and its absence has a severe impact on other application-layer protocols such as HTTP, FTP, and SMTP. Thus, for any network, DNS provides a vital service, and without it the network is deemed isolated and blocked from using Internet services.

The DNS service could be impacted by misconfiguration causing unavailability of the DNS servers [2]. It can also be interrupted as a result of denial of service (DoS) attacks, cache poisoning, and compromised data [3]. To countermeasure such attacks, researchers proposed several remedies spanning a wide range of solutions such as using DNS security extensions [4], using "Anycast routing" [5], manipulating the time-to-live (TTL) value [6][7], and increasing the efficiency of DNS caching [8].

In this paper, we propose a countermeasure to the intentional denial of service to the DNS by higher nameservers. To the best of our knowledge, there has been no research conducted to address this problem. Our proposed countermeasure is motivated by the fact that the resolvers need to go through the higher nameserves (i.e., root and/or top-level-domain (TLD)) to resolve the client's request. The nameserver could respond to the resolver's query with a *Refused* error code just because the resolver is not welcomed in this nameserver [9]. Hence, the nameserver could maliciously block a specific domain name.

Although the idea of malicious and intentional denial of service by higher nameservers seems unlikely at first, there are several reasons that may force a higher nameserver to become malicious and perform intentional denial of service against a specific organization or country. For example, intentional denial of service by higher nameservers can be driven by political motivations, as governments may force higher nameservers to block Internet access in order to establish an Internet ban on the targeted region. Many large services and networks have been attacked recently for political motivations. For example, on December 2009, Gmail had many attacks targeting email accounts of Chinese human rights activists [10]. Twitter, has also been attacked during 2009 by hackers from Iran [11]. Another prime example of political motivations of a higher name server to perform intentional denial of service against an organization are the recent attempts by many governments to pressure service providers to block access to WikiLeaks [12].

2. RELATED WORK

The DNS system can be attacked in many different ways as described by Cheung [5]. Cheung divided the DNS system into three sections; the low level servers (resolvers), the communications between the servers and the clients, and the top level servers (nameservers). In the resolvers, the attackers try to get access to the DNS system through the available vulnerabilities which could lead to damaging the resolver servers or misbehavior of the resolver servers. By flooding the DNS servers with a massive number of forged queries, the normal DNS queries could be dropped by the routers. This could also happen if the attacker was able to take advantage of a critical router along the DNS server

CCECE 2011 Niagara Falls, Canada
978-1-4244-9789-8/11/$26 ©2011 IEEE

path to cause it to misbehave. Finally, the nameservers could be damaged by directly accessing the nameservers or through damaging other necessary services to the nameservers. Other malicious attacks that take advantage of the DNS caching aspect include cache poisoning. In such an attack, the attacker inserts an incorrect DNS record or modifies an available DNS record in the DNS cache server so that it redirects the client to the attacker's website [3][4].

To countermeasure malicious attacks on the DNS several solutions were proposed. The robustness of the name servers could be increased by using "Anycast routing" that counters the flooding attacks [5]. Another countermeasure uses the DNS security extensions to provide a secure integrity and authenticity to the DNS traffic [13]. Researchers in [6] and [7] discussed how to manipulate the time-to-live (TTL) value of the DNS records to improve the availability of the DNS system against DoS attacks. A solution provided in [8] increases the efficiency of the DNS caching by suggesting two policies. The first policy is renewal cache refreshment where the DNS record is refreshed upon the expiration of the TTL. The second policy is simultaneous validation where the expired DNS record is refreshed whenever a DNS lookup requests this record.

To ensure the availability of the root DNS servers, a secondary DNS lookup service through a cooperative cache sharing network that is based on Pastry and Chord P2P protocols was proposed in [3]. It was found that the effort to attack a domain becomes harder when the resource records of the domain are distributed over different cache resolvers.

A new platform was proposed in [14] that implements a Cooperative Domain Name System (CoDoNS) based on a structured P2P Pastry protocol with analytically-informed proactive caching used to distribute the popular records in the nodes near the home node. An alternative solution uses Chord P2P protocol to replace the existing DNS [15]. In addition to implementing all the functionality of the existing DNS, the new domain record requires authentication from the higher level. Although the design had more availability and better fault-tolerance, it suffered from high latency.

3. PROPOSED SOLUTION

To solve the intentional DNS blocking, a simple forwarding of the DNS traffic to a neighbor DNS resolver will help. However, the sudden increase of the DNS traffic from the neighbor resolver could trigger the higher DNS servers to investigate and block the neighbor resolver as well. An alternative solution can use more neighbors so as to distribute the load among the participating neighbors. The communications between these neighbors could be established through a structured P2P network or through the use of a round-robin approach. Using the structured P2P network will create a scalable and stable network but with an extra latency. On the other hand, a round-robin approach provides a fast path to resolve queries but it will suffer when the network size is increased because of the need to maintain the routing information for every network node when a node joins or leaves the network. Another solution can be created by combining the last two solutions. Such a solution will be referred to as the *Dynamic Round-Robin P2P* solution.

The proposed solution uses both the Chord P2P protocol [16] and the round-robin approach. The Chord protocol arranges the peers (nodes) in a ring, and constructs a *finger table* that is made of a number of selective nodes so as to minimize the number of hops to the next responsible resolver node. Accordingly, the proposed solution requires each resolver to construct a list of unique nodes from the finger table. Such a list is referred to as a *routing table*. In addition to using the normal DNS lookup procedure, the routing table will be used to forward the query to one node in a round-robin fashion. When the query is forwarded to the next hop node, it will resolve the query through the existing DNS lookup procedure. If the next hop node is BLOCKED (i.e., the node is being intentionally denied DNS service by higher nameservers), the query will not be resolved. To keep the routing table updated with ACTIVE nodes (i.e., nodes that can resolve queries through higher nameservers) and minimize the number of unresolved queries, there are two modifications to the normal round-robin approach; using Chord as underlying layer, and using an enhanced round-robin. Chord protocol is used as an underlay infrastructure to solve the scalability issue in the round-robin approach by updating (i.e., stabilizing) the finger table in the Chord protocol on a periodic basis. Also, the stabilization process is modified to support the scenario of a BLOCKED node. An enhancement to the round-robin technique is to go through the entire routing table if there are FAILED nodes, which guarantees a successful lookup.

3.1. Modified Chord P2P protocol

The Chord P2P protocol constructs a *Successor List* that is used to provide better lookup and high stability when there are failures in the network. Moreover, the use of *virtual nodes* was proposed in [16] to increase the fairness between the nodes. Also, the Chord P2P protocol uses two main procedures, *Find Successor* and *Closest Predecessor*, to submit a query to nodes. Accordingly, there are three modifications to the Chord P2P protocol introduced in [16].

The first modification introduces a new condition to the Find Successor procedure that checks if the entry in the Successor List is BLOCKED or DEAD. Since the BLOCKED node cannot resolve any query, it will be flagged in the Find Successor procedure. However, the BLOCKED node will not be removed from the Successor List so as to help in the Closest Predecessor procedure. The BLOCKED node does not cause a timeout since it continues to participate in the stabilization process.

The second modification extends the condition in the Closest Predecessor procedure that checks if the entry is ALIVE or not to consider the BLOCKED node as an ALIVE node. Each entry in the finger table or the Successor List will be checked with the condition "*Entry*[*k*] ∈ (*ID*, *KEY*)" if the node is either ACTIVE or BLOCKED, otherwise the node will be removed from the routing table.

The third modification ensures that the feature of notifying the predecessor and successor when a node voluntary departs that was suggested in [16] will be used also when a node is BLOCKED.

3.2 Modified round-robin

To enhance the round-robin performance, the round-robin algorithm will check a window of the routing table in each round. This window could be one entry resulting in the basic round-robin, portion of the routing table, or the complete routing table. This enhancement is applied only when the first entry in the window is DEAD or BLOCKED.

Using the lookup window, the lookup procedure will start with the first entry in the window. If this entry is either DEAD or BLOCKED, the algorithm will remove that entry from the routing table. In addition, there will be a timeout when the entry is DEAD. If all entries in the window have been removed, the query message will be dropped. On the other hand, the message will be considered successfully resolved if there is at least one ACTIVE entry in the window. A pointer is used in order to help in determining the window's starting point in the next round. The pointer is incremented in a round-robin fashion after each check. Notice that the window size will be adjusted whenever the routing table size is changed.

4. SOLUTION EVALUATION

The proposed dynamic P2P solution is evaluated and compared against the modified Chord P2P protocol by means of simulations. The simulation environment is built using NetBeans IDE 6.8 with Java 1.6.0. The Chord behavior in the simulator was verified by reproducing the results in [16]. In general, the simulator has the same behavior as the Chord protocol presented in [16].

The following describes the analysis metrics used for comparison between the modified Chord P2P protocol and the proposed round-robin P2P approach. Note that the unit of the analysis metric, if any, appears between parentheses after the name of the analysis metric.

1. *Load (keys/node)* – The average number of keys per node that only appear in the lookup (query) messages.
2. *Traffic (message/node)* – The average number of forwarded query messages per node.
3. *Fairness* – Reflects how uniformly the query resolving is distributed among the nodes (i.e., *load fairness*). Also, fairness is used to identify the uniformity of the number of messages forwarded by each node (i.e., *traffic fairness*). Note that the messages in the *load fairness* case refer to the resolved queries that only appear in the lookup messages, and in the *traffic fairness* case refer to any forwarded queries.
4. *Percentage of Failed Lookup* – Percentage of dropped queries out of the total queries sent.
5. *Message Path Length (hops)* – The number of hops needed to identify the resolver.
6. *Timeout (seconds)* – The average timeout when a node does not respond (i.e., failed or departed) in a successful lookup.

5. RESULTS AND ANALYSIS

Two scenarios were simulated and analyzed; *Blocking* and *Failure*. The blocking scenario considers simulating the situation when a number of nodes become BLOCKED, whereas the failure scenario simulates when a number of nodes FAIL. Each scenario consists of two network configurations. The two configurations examined are:

1. *Chord-20:* it is the basis of the other configuration and is used to update the routing information of the other configuration. The Successor List size in Chord-20 is $2\times(\log_2 N)$, where N is the total number of virtual nodes in the network. So, with 1,000 nodes and one virtual node per real node, the size equals 20. Chord-20 was picked for implementation similar to what is used in [16].
2. *Round-10:* changing the lookup process from Chord to round-robin with the lookup window size set to the size of the complete routing table. Thus, for a network with 1,000 nodes, the window size = $(\log_2 1{,}000) = 10$.

Table 5.1 summarizes the common parameters and the corresponding values that are used. Most of the values used are derived from [16].

Table 5.1: Default parameters

Parameter	*Value*
Iterations	10 iterations per sample
Initial Network Size	1,000 real nodes
Virtual Nodes	4 virtual nodes/real node
Successor List Size	$2\times(\log_2 N)$
Lookup Window Size	full routing table size
Query Message	key & live source are chosen randomly
Number of Queries	10,000 lookup queries
Query Inter-arrival Time	Exponential with mean of 50 ms
Timeout	500 ms
Stabilization Period	Uniform [15, 45] seconds

In Figures 1 to 5, after a network of 1,000 real nodes is stable, the state of 50% of the nodes is changed from ACTIVE to BLOCKED (i.e., blocking scenario), and from ACTIVE to FAILED (i.e., failure scenario). Then, uniform random queries are generated using the default parameters.

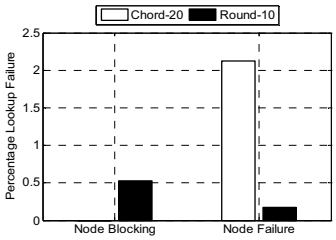


Figure 1: Lookup failure.

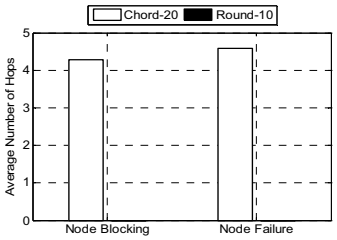


Figure 3: Message path length.

5.1 Lookup failure

Figure 1 shows the lookup failure for the blocking and the failure scenarios. In case of the blocking scenario, there is no lookup failure when using Chord-20. That is because the blocked node will notify its neighbor. Accordingly, the neighbor updates its Successor List. In contrast, Round-10 depends on the routing table that is only updated during the stabilization process (i.e., finger table updating). Hence, it incurs a negligible lookup failure (about 0.5%).

For the failure scenario, we note that the Chord protocol is stable whenever the Successor List is accurate. However, when failed nodes are 50%, the network stability decreases and the dropped messages increase. In contrast, Round-10 suffers from a minor lookup drop similar to its behavior in the blocking scenario.

5.2 Load and load fairness

The load and the load fairness results are shown in Figures 2(a) and 2(b), respectively. For both the blocking and the failure scenarios, each non-BLOCKED node receives an equal load (i.e., keys) for both Chord-20 and Round-10. Thus, both Chord-20 and Round-10 distribute queries equally among the remaining nodes. In contrast, it is observed that for both cases of the blocking and the failure scenarios, Round-10 provides higher fairness between the nodes than Chord-20 due to its natural behavior.

5.3. Message path length

For the blocking scenario, we note that Round-10 does not forward the queries to intermediate nodes, so there is no extra hop to resolve the queries as noticed from Figure 3. In Chord-20, the path length follows the equation *Path Length* = $(log_2\ M)/2 - (log_2\ r)/2 + 1$ provided in [16], where M = network size, and r = virtual nodes per real node.

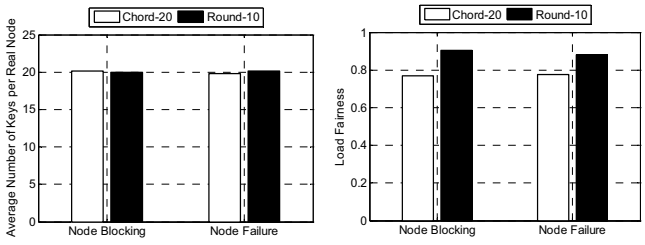
In the node failure scenario, Chord-20 experiences an increase in the path length because of the failed nodes. This is because the node attempts to submit the query to the failing node, which results in a timeout. Then the node will utilize the closest predecessor procedure and that results in an additional submission of the same query, resulting in an increase in the path length. On the other hand, there are no hops in resolving queries in the Round-10.

5.4. Message timeout

In the case of the blocking scenario, it was stated earlier that the BLOCKED nodes are still considered as live nodes, and that results in no timeout as illustrated in Figure 4. Whereas in the failure scenario, a failing node obviously does not notify others about its condition, then it is possible that some of the nodes along the path to the responsible resolver node are DEAD. Hence, Chord-20 experiences longer timeouts than Round-10 as shown in Figure 4.

5.5. Traffic and traffic fairness

Figures 5(a) and 5(b) illustrate the traffic and traffic fairness, respectively. For the blocking scenario, the average number of traffic increases due to the additional traffic introduced by the stabilization process. The difference between Chord-20 and Round-10 is due to the fact that Round-10 does not forward messages during the lookup process. For the failure scenario, a slight increase in the average traffic between the nodes is observed when compared with the average traffic in the blocking scenario presented in Figure 5(a). This occurs in the failing scenario because the only ALIVE nodes participating in the system are not BLOCKED. In contrast, in the blocking scenario, BLOCKED nodes are part of the ALIVE nodes but do not participate in forwarding messages.



(a) (b)

Figure 2: (a) Load, and (b) load fairness.

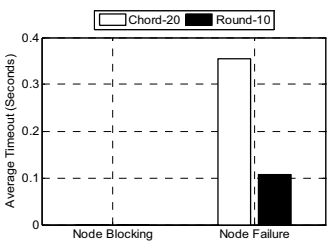
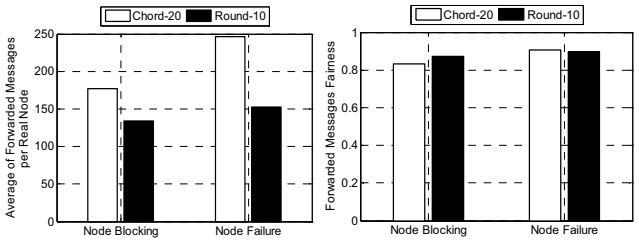


Figure 4: Message timeout.



(a) (b)

Figure 5: (a) Traffic, and (b) traffic fairness.

Also, as shown in Figure 5(b), the drop in traffic fairness between the nodes, especially for Chord-20, is a result of updating the finger table during the stabilization process. Accordingly, when the stabilization process is completed, there will be no BLOCKED nodes in the table. That causes the BLOCKED nodes to only generate queries, and to stop helping in forwarding the messages. As a result, the traffic fairness among all ALIVE nodes is affected.

6. CONCLUSIONS

This paper proposes a P2P solution that provides an alternative path to resolve the query which increases the availability of the DNS to any specific region. The proposed solution minimizes the effects of the intentional blocking by a higher nameserver. The P2P system is based on the Chord protocol and uses the round-robin scheme when communicating with other nodes. This solution provides a load balance between the nodes in the system and provides an answer for the DNS query through two parallel paths, the standard path (i.e., through root/TLD DNSs) and the P2P system. The proposed solution was evaluated using a custom-built simulator, and it was concluded that Round-10 with 4 virtual nodes has the best performance.

7. ACKNOWLEDGEMENT

The authors acknowledge the support provided by King Fahd University of Petroleum and Minerals (KFUPM). This project is funded by King Abdulaziz City for Science and Technology (KACST) under the National Science, Technology, and Innovation Plan (project No. 08-INF97-4).

8. REFERENCES

[1] "Root hints," *ICANNwiki*, http://icannwiki.org/Root_hints

[2] D. Barr, "Common DNS Operational and Configuration Errors," RFC1912, February 1996.

[3] N. Poolsappasit and I. Ray, "Enhancing Internet Domain Name System Availability by Building Rings of Cooperation Among Cache Resolvers," *Proceedings of 2007 Information Assurance and Security Workshop*, West Point, NY, pp.317-324, 20-22 June 2007.

[4] S. Ariyapperuma and C.J. Mitchell, "Security Vulnerabilities in DNS and DNSSEC," *Proceedings of the Second International Conference on Availability, Reliability and Security*, Vienna, pp. 335-342, 10-13 April 2007.

[5] S. Cheung, "Denial of Service against the Domain Name System: Threats and Countermeasures," *IEEE Security and Privacy*, vol. 4, no. 1, p. 40, January 2006.

[6] H. Ballani and P. Francis, "Mitigating DNS DoS Attacks," *Proceedings of the 15th Conference on Computer and Communications Security*, Alexandria, Virginia, pp. 189-198, 27-31 Oct. 2008.

[7] V. Pappas, D. Massey, and L. Zhang, "Enhancing DNS Resilience against Denial of Service Attacks," *Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, Edinburgh, pp. 450-459, 25-28 June 2007.

[8] E. Cohen and H. Kaplan, "Proactive Caching of DNS Records: Addressing a Performance Bottleneck," *Proceedings of the 2001 Symposium on Applications and the Internet*, San Diego, CA, USA, pp.85-94, 8-12 Jan. 2001.

[9] P. Mockapetris, "Domain Names - Implementation and Specification," RFC1035, November 1987.

[10] D. Drummond, "A new approach to china," *The Official Google Blog*, http://googleblog.blogspot.com/2010/01/new-approach-to-china.html, Jan. 2010.

[11] J. Finkle and D. Bartz, "Twitter hacked, attacker claims Iran link," *Reuters*, http://www.reuters.com/article/idUSTRE5BH2A620091218, December, 2009.

[12] "WikiLeaks," *Wikipedia*, http://en.wikipedia.org/wiki/WikiLeaks#cite_note-197, 2011.

[13] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, "DNS security introduction and requirements," RFC 4033, Mar. 2005.

[14] V. Ramasubramanian and E.G. Sirer, "The Design and Implementation of a Next Generation Name Service for the Internet," *Proceedings of applications, technologies, architectures, and protocols for computer communications*, Portland, Oregon, USA, pp. 331 - 342, 30 Aug.-3 Sep. 2004.

[15] R. Cox, A. Muthitacharoen, and R. Morris, "Serving DNS Using a Peer-to-Peer Lookup Service," *Proceedings of first International Workshop on Peer-to-Peer Systems*, London, UK, pp. 155 – 165, 2002.

[16] I. Stoica, R. Morris, D. Liben-Nowell, D.R. Karger, M.F. Kaashoek, F. Dabek, H. Balakrishnan, "Chord: a scalable peer-to-peer lookup protocol for Internet applications," IEEE/ACM Transactions on Networking, vol. 11, no. 1, pp. 17- 32, Feb 2003.