Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.12785/amis/070613

Optimizing DNS Server Selection

Zheng Wang[1,2,*] *and Rui Wang*[2]

[1]Computer Network Information Center, Chinese Academy of Sciences, Beijing 100190, China
[2]China Organizational Name Administration Center, Beijing 100028, China

Received: 4 Mar. 2013, Revised: 7 Jul. 2013, Accepted: 8 Jul. 2013
Published online: 1 Nov. 2013

Abstract: In DNS zone delegation, the NS resource records of the delegated domain appear in both the parent and child zones, which complicates the domain name resolution procedure especially for server selection. In this paper, we investigate the recursive resolution mechanism for the delegated domain and the necessary operations. As there are no specifications for such resolution, we provide some recommendations such as NS record substitution mechanism. Furthermore, we analyze the delegation impacts on server selection as the NS records are obtained from both parent and child zone. We propose an integrated sever selection procedure to handle the lame delegation and enhance server selection efficiency. It makes use of the two stages of server probing information and can help save the initial server probing in the server selection algorithm. Several proposals as to the disposal of NS resource records and the maintenance of the rtt information of servers are also presented.

Keywords: domain name system, zone delegation, server selection

1 Introduction

The Domain Name System (DNS) is a fundamental component of the modern Internet [1], providing a critical link between human users and Internet routing infrastructure by mapping host names to IP addresses.

The Domain Name System uses a tree (or hierarchical) name structure. At the top of the tree is the root node followed by the Top-Level Domains (TLDs), then the Second-Level Domains (SLD) and any number of lower levels, each separated with a dot.

Each node within the domain name hierarchy is assigned to an authority-an organization or person responsible for the management and operation of that node [2]. Such an organization or person is said to administer the node authoritatively. The authority for a particular node can in turn delegate authority for lower levels of that node within the domain name hierarchy. The rules and limitations of the authority are covered by agreements that flow through the various nodes in the hierarchy.

When a parent zone P delegates part of its namespace to a child zone C, P stores a list of NS resource records for the authoritative servers of zone C. This list of NS resource records are kept both at the parent and the child zone. As shown in Fig.1, com. zone delegates example.com. zone to a child zone. The authoritative servers of the child zone are listed in a set of NS resource records. And normally the same set of NS resource records are also contained in the zone file of the child zone- example.com. zone here.

```
$ORIGIN com.
example.com.   IN   NS   ns1.example.com.
example.com.   IN   NS   ns2.example.com.
example.com.   IN   NS   ns3.example.com.
$ORIGIN example.com.
example.com.   IN   NS   ns1.example.com.
example.com.   IN   NS   ns2.example.com.
example.com.   IN   NS   ns3.example.com.
```

Fig. 1: Zone delegation example.

In order to enhance availability, robustness and reliability, each domain is usually handled by multiple DNS servers. For example, there are 13 root servers distributed around the world, and some servers also have multiple instances using anycast. And for the Top Level Domains, which usually see a very large number of queries, it is not unusual to have five or six servers. A local DNS server, which is also called iterative resolver, receives the IP addresses of all the name servers for the

[*] Corresponding author e-mail: zhengwang09@126.com

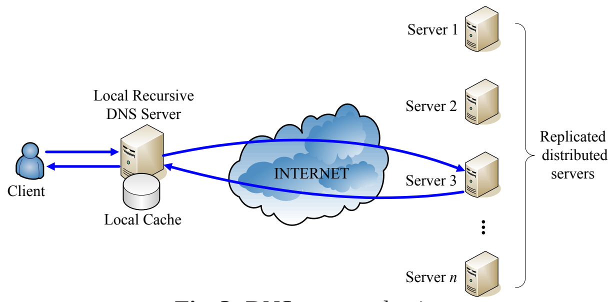

Fig. 2: DNS server selection.

domain to which the query was directed. These IP addresses are the items contained in the NS resource records in DNS specification. And all of the NS resource records corresponding to the same domain are called NS RR sets. The local DNS server goes ahead and caches all these IP addresses. Thus, the next time there is a query for this domain, the DNS server chooses from the cached NS records listed in the domain's NS RR set. For instance, if a local DNS server knows (via caching) the IP addresses of more than one .com server, then it needs to select one of these servers to forward the query. This is referred to as the DNS server selection problem (see Fig. 2).

There have been some analysis and measurement studies of DNS server selection. In general, many efforts focus on DNS performance evaluation and identification of possible DNS misconfigurations, errors or implementation bugs [17–19]. Another hot field is related to the security issues such as DNS traffic anomaly detection [20, 21], counter measures of DNS DoS or DDoS attacks [22, 23] etc. Our interest lies in characterizing the performance of DNS server selection algorithm implemented in BIND.

What complicates DNS server selection problem is that the NS records are obtained respectively from the parent and child zone, rather than once in the domain name resolution. In other words, the resolver first gets the referral to child zone from the parent zone. And then it selects one of the name servers to fetch the authoritative name servers from the child zone. Finally, it performs the server selection again among the authoritative name servers to send out the DNS query. This name resolution procedure introduces a coupling between the two stages of server selection, which the current DNS implementations fail to exploit to facilitate the server selection efficiency. We present in this paper an integrated server selection procedure through which the rtt information acquired in the first stage can be maintained for the second stage, thereby unnecessary queries for the duplicate name servers are saved. Another impact of DNS specifications on the server selection comes from the lame delegation. Previous analysis [3] shows that on average 15% of the DNS zones suffer from a specific misconfiguration called lame delegation, in which the parent of a DNS zone points to wrong name servers for

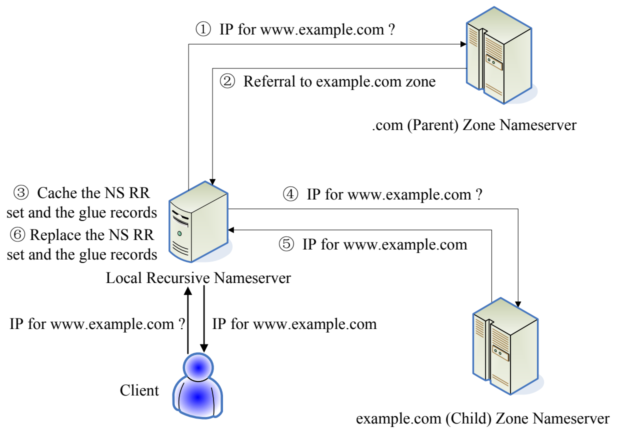
① IP for www.example.com ?
② Referral to example.com zone
.com (Parent) Zone Nameserver
③ Cache the NS RR set and the glue records
⑥ Replace the NS RR set and the glue records
④ IP for www.example.com ?
⑤ IP for www.example.com
Local Recursive Nameserver
IP for www.example.com ?
IP for www.example.com
Client
example.com (Child) Zone Nameserver

Fig. 3: Recursive resolution procedure.

the child zone. Lame delegation may translate to increased query time when DNS queries sent to non-existing servers timeout, and have to be resent to a different server. As lame delegation, especially in the server selection, has not been handled by DNS implementations and specifications, we propose an optimized server substitution mechanism for server selection in this paper, which eliminates wrong name servers for the second stage server selection while retaining correct name servers and their rtt information. DNS implementations provide varied choices on the NS RR set to which DNS queries are sent. And the NS RR set are cached for the afterwards DNS queries until its TTL expires. We clarify that NS RR set from the parent zone should not be used for DNS queries and server selection algorithm should be based on IP (glue record) rather than domain name (NS record). We also derive the TTL of name server's rtt information according to the query fairness, and conclude that the TTL should be set as two times that of NS RR set.

The rest of the paper is organized as follows: In Section II, we will analyze the effects of delegation on DNS server selection. A new server selection procedure is proposed in Section III. In Section IV, other proposals for DNS server selection are provided. We develop an experimental test on the server selection algorithm of BIND implementation and provide performance evaluation of parent zone selection algorithm via simulation study in Section V. Finally, we conclude the paper in Section VI.

2 The effects of delegation on DNS server selection

Fig. 3 illustrates the recursive resolution procedure. The client's browser uses a resolver and queries a local recursive server for a name (say example.com). The query may miss the DNS cache in this server, that is there is no

```
/* Authority section */
example.com.   IN  NS  ns1.example.com.
               IN  NS  ns2.example.com.
               IN  NS  ns3.example.com.
/* Additional section – “glue” records */
ns1.example.com.   IN  A  111.0.0.1
ns2.example.com.   IN  A  111.0.0.2
ns3.example.com.   IN  A  111.0.0.3
```

Fig. 4: Authority and additional section of DNS messaget

cached A records for ”www.example.com”. Moreover, if the NS record set for the queried domain also expired at this time (otherwise, the server can go to the authoritative server of the ”example.com” zone straightly), the recursive server has to request the parent zone of ”example.com” by contacting the authoritative server of ”.com” zone. The ”.com” authoritative server answers with a referral to the servers responsible for the example.com domain. This is in the form of NS records of servers in the authority section of the DNS message (see Fig. 4). Though technically we asked only for the NS records, the servers also give us the IP address of each in the additional section of the DNS message: this is known as ”glue” and is provided to avoid ”query loop” and save us from having to look it up. The recursive server chooses one of the authoritative servers and sends off the same query: ”what’s the A record for www. example.com?”. The authoritative server’s reply message contains the A record in the answer section, the NS records and glue records in authority and additional section respectively.

Note that here the NS records and glue records are not the replication of the previous referral message but those stored in the child zone. Therefore the problem is which NS resource records, those in parent zone or the child zone, are finally cached by the recursive server and used for afterwards requests? Ideally, when the operator of zone C makes changes to one or more of C’s authoritative servers, he must coordinate with the operator for zone P to update P accordingly. In reality, there are cases where changes made at the child zone are not re?ected at the parent zone, usually due to ”bad” coordination between them. So the NS records in the child zone are more likely to represent the real delegation. So it is highly recommended that the lately fetched NS records from the child zone should be adopted as the cached ones. However, the NS records from the parent zone should also be temporarily cached until they are subsequently substituted by the ones in the child zone. This is necessary because the first request for one of the child zone authoritative servers may fail and the recursive server can also try other servers in the cache in the afterwards requests. The worst case is that all servers can not be successfully requested until the TTL expires, the NS records are dropped from the cache and a new round begins when a new request comes.

However, the NS records from the parent zone should also be temporarily cached until they are subsequently substituted by the ones in the child zone. This is necessary because the first request for one of the child zone authoritative servers may fail and the recursive server can also try other servers in the cache in the afterwards requests. The worst case is that all servers can not be successfully requested until the TTL expires, the NS records are dropped from the cache and a new round begins when a new request comes.

In terms of the DNS server selection algorithm, as discussed above, there are two problems to be considered:

1) How to choose one server from the NS records from the parent zone?

2) How to choose one server from the NS records from the child zone?

Under the above proposal (temporarily cache the records from parent zone and then replace them by those from child zone), since there is no historical server performance information (eg. rtt) available when getting the NS records from the parent zone, the best choice is to choose one server at random, hopefully each server having equal chance to be selected. If the first request is successful, the cache substitution is done and the next problem is problem 2. If the first request fails, as the NS records are still in the cache, we can penalize this server and favor other servers in the afterwards selections. We can use various punishment techniques and the following algorithm is similar to the BIND 9 algorithm [24].

Set the initial virtual rtt values for all servers, the server selection algorithm always choose the server with the smallest rtt value. The initial virtual rtt values should be different and significantly less than the real ones.

If the query request fails when contacting a server, its rtt is updated as follows:

$$rtt_{new} = min(rtt_{previous} + 0.2, 10)(second) \tag{1}$$

Where $rtt_{previous}$, and rtt_{new} are the previous rtt maintained by the algorithm and the updated rtt respectively. For other servers not selected in the round, their rtts are updated as follows:

$$rtt_{new} = rtt_{previous} * 0.98 \tag{2}$$

Simulation results show later that when all servers can not be successfully requested, the algorithm degenerates to round robin. Once a server’s request gets successful answer, we come to problem 2.

For problem 2, BIND 9 uses the following algorithm. Go through all servers in the first round to obtain their initial rtts. Then always select the server with the least rtt maintained by the algorithm. If the query request gets response successfully from a server, update its rtt as follows:

$$rtt_{new} = rtt_{previous} * 0.7 + rtt_{current} * 0.3 \tag{3}$$

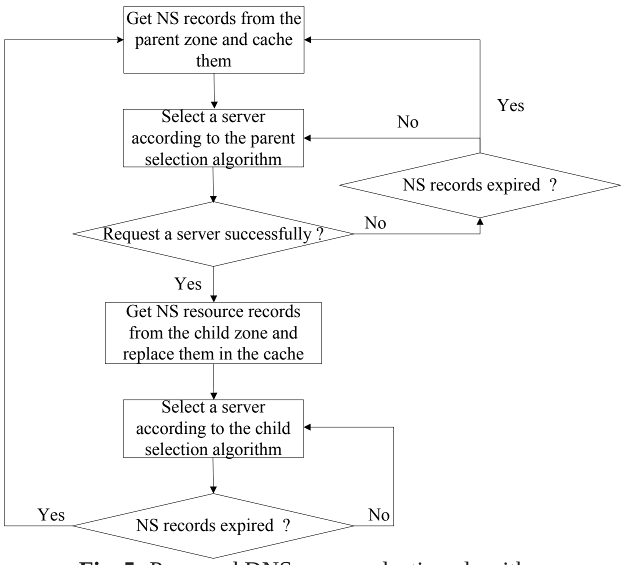
Get NS records from the parent zone and cache them
Select a server according to the parent selection algorithm
No
Yes
NS records expired ?
Request a server successfully ?
No
Yes
Get NS resource records from the child zone and replace them in the cache
Select a server according to the child selection algorithm
Yes
NS records expired ?
No

Fig. 5: Proposed DNS server selection algorithm

Where $rtt_{current}$ is the current measured rtt.

For the failed query and other servers not selected, the update of rtt is as Equation (1) and (2).

3 Proposed DNS server selection procedure

Given that BIND 9 server selection algorithm performs fairly well for problem 2, careful thought tells us that problem 1 and 2 are actually the two stages of domain name resolution, thus if the delegation is consistent, their NS records to be selected are the same set. The server probing information in the stages should be shared and integrated to enhance the efficiency of server selection algorithm. However, BIND 9 implementation does not seem to see this point, and adopts the ”divide and conquer” policy, which takes the NS records from the child zone as the new start of the server selection algorithm and neglects those from the parent zone. We believe that this is a kind of probing information waste. Therefore we propose the following server selection solution.

The modified procedure of domain name resolution can be expressed as follows (see Fig. 5). First the recursive server gets NS records from the parent zone, caches them and then selects a server according to the parent selection algorithm. The selection algorithm runs until any of two conditions is satisfied. One is that the NS records expire, and then the recursive server has to request the parent zone again. The other is that a server is successfully requested and the procedure goes to the child zone selection stage. In this stage, the recursive server first checks the returned NS records and compares them with those already cached. If the two sets of NS records are partly different, there are some NS records in the cache which match the ones in the child zone and also have been requested by the parent selection algorithm. Thus remain only those name servers’ rtts maintained by the parent selection algorithm in the previous stage. Meanwhile, replace all NS records in the cache by the ones in child zone. Then perform the child zone selection algorithm. Ordinarily, those servers already requested by the parent selection algorithm are unlikely to be requested at first. This means that the initial polling stage of the child zone selection algorithm excludes those servers. Note that those servers are sure to include one successfully requested server and none or more unreachable servers in the parent selection stage. Child zone selection algorithm is stopped only by the event of NS records expiration. Afterwards, the recursive server contacts the parent zone and a new round starts.

4 Other proposals

4.1 On the use of cached parent zone NS records

As mentioned in Section 2, the parent zone NS records can only be utilized for the selection of authoritative servers (child zone servers), which means they should not be cached to be replied as authoritative servers in the form of NS records in the DNS message. Thus the kind of cache is a little different from the common sense. This is due to the consideration that only the NS records in the authoritative servers can be regarded as the real authority. Usually the NS records in the child zone file are first altered while those in the parent file may fail to change accordingly due to lack of coordination. Therefore it is not until the NS records in the child zone are lately obtained that the NS records can really be cached and used in the authority section of the DNS message.

4.2 Base the server selection algorithm on IP rather than domain name

Section 2 and 3 do not specify what does the ”server” means. Does it denote the domain name in the RDATA of the NS record? Or the IP address in the RDATA of the glue record? The question is raised especially when one NS RNAME corresponds to multiple glue records thus multiple IP addresses. We argue that only server’s IP is the ultimate target of server selection algorithm. So we need to expand all relevant IPs according to the NS and glue RRs before performing the server selection algorithm. One example is illustrated in Fig. 6. ns2.example.com has two glue records and ns3.example.com has three glue records. The total 6 IP addresses have the same rank and should be listed for the server selection algorithm.

```
/* Authority section */
example.com.   IN  NS  ns1.example.com.
               IN  NS  ns2.example.com.
               IN  NS  ns3.example.com.
/* Additional section – "glue" records */
ns1.example.com.   IN  A  111.0.0.1
ns2.example.com.   IN  A  111.0.0.2
ns2.example.com.   IN  A  111.0.0.3
ns3.example.com.   IN  A  111.0.0.4
ns3.example.com.   IN  A  111.0.0.5
ns3.example.com.   IN  A  111.0.0.6
```

Fig. 6: Authority and additional section of DNS message (multiple glue RRs)

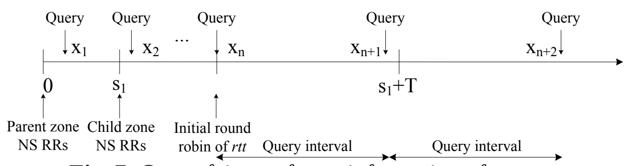

Fig. 7: Query fairness for rtt information of servers

4.3 The maintenance of rtt information of servers

Section 2 and 3 have discussed how should the rtt information of servers be updated, but have not detailed how long should they be kept in the recursive server or do they also have time-to-live just as RR? Instant thought may tell us that the rtt information of servers should be retained and maintained as long as the NS RRs still exist in the recursive server. This idea, though seemingly logical, fails to pay regard to the query fairness. Here we use the query fairness to express the principle that if two queries have the same interval to their previous ones, they should be granted the equal rtt information.

The query fairness for rtt information of servers is shown Fig. 7. The start time is the time when the recursive server gets the parent zone NS RRs and it sends the first request for the authoritative server at x_1. Then at s_1, the child zone NS RRs are obtained. After subsequent queries from x_2 to x_{n-1}, the initial round robin of rtt probing is completed. Afterwards, a query comes at x_n. The recursive server receives the last query before the time-to-live (TTL) of NS RR set expires. Here we let the TTL is T. Finally, the first query after the TTL expires comes at x_{n+2}. For the sake of completeness, we consider two cases, discuss the extreme of their query intervals and infer how should the TTL of rtt be redefined rather than comply to that of NS RRs according to the fairness assumption.

1) Let the queries from x_2 to x_n do not exist and take the first query after having the parent zone RRs as the starting one and the query at x_{n+1} as the following one. Then the query interval is

$$I_1 = x_{n+1} - x_1 \tag{4}$$

We have

$$0 < x_1 < s_1 \tag{5}$$

And let the second query approximates the time of NS RRs expiration or

$$x_{n+1} \to s_1 + T \tag{6}$$

Usually the interval between two times of acquiring the NS RRs is relatively small comparing to their TTL. For a successful request for the authoritative server, that interval is actually its rtt. So we have

$$s_1 \ll T \tag{7}$$

By Equation (4)-(7), we have

$$I_1 \to T \tag{8}$$

For the fairness, if the query interval between x_{n+1} and x_{n+2} is no more than I_1, the query at x_{n+2} should also be served with the rtt information of servers. Therefore the rtt information should be kept at least until the query at x_{n+2} comes. Thus the TTL of the rtt information is

$$T_{rtt1} = 2T \tag{9}$$

2) We let the completion of the initial round robin of rtt probing as the fairness precondition. Then the query at x_n is the starting one and the query at x_{n+1} is as the following one. Then the query interval is

$$I_2 = x_{n+1} - x_n \tag{10}$$

We have

$$s_1 < x_n < s_1 + T \tag{11}$$

And let the query at x_n approximates s_1

$$x_n \to s_1 \tag{12}$$

Usually the interval between two times of acquiring the NS RRs is relatively small comparing to their TTL. For a successful request for the authoritative server, that interval is actually its rtt.

By Equation (6), (7), (10) and (12), we have

$$I_2 \to T \tag{13}$$

Similar to Case 1, the TTL of the rtt information is

$$T_{rtt2} = 2T \tag{14}$$

In both cases, we can infer that the TTL of the rtt information should be two times that of the NS RRs.

```
; <<>> DiG 9.6.0 <<>> @218.241.99.50 sina.com.cn +trace
; (1 server found)
;; global options: +cmd
.                          510300IN      NS      H.ROOT-SERVERS.NET.
.                          510300IN      NS      D.ROOT-SERVERS.NET.
.                          510300IN      NS      F.ROOT-SERVERS.NET.
.                          510300IN      NS      A.ROOT-SERVERS.NET.
.                          510300IN      NS      B.ROOT-SERVERS.NET.
.                          510300IN      NS      K.ROOT-SERVERS.NET.
.                          510300IN      NS      I.ROOT-SERVERS.NET.
.                          510300IN      NS      J.ROOT-SERVERS.NET.
.                          510300IN      NS      G.ROOT-SERVERS.NET.
.                          510300IN      NS      M.ROOT-SERVERS.NET.
.                          510300IN      NS      L.ROOT-SERVERS.NET.
.                          510300IN      NS      C.ROOT-SERVERS.NET.
.                          510300IN      NS      E.ROOT-SERVERS.NET.
;; Received 500 bytes from 218.241.99.50#53(218.241.99.50) in 3 ms

cn.                        172800IN      NS      a.dns.cn.
cn.                        172800IN      NS      b.dns.cn.
cn.                        172800IN      NS      c.dns.cn.
cn.                        172800IN      NS      d.dns.cn.
cn.                        172800IN      NS      e.dns.cn.
cn.                        172800IN      NS      ns.cernet.net.
;; Received 294 bytes from 128.63.2.53#53(H.ROOT-SERVERS.NET) in 288 ms

sina.com.cn.               21600  IN     NS      ns3.sina.com.cn.
sina.com.cn.               21600  IN     NS      ns2.sina.com.cn.
sina.com.cn.               21600  IN     NS      ns1.sina.com.cn.
;; Received 131 bytes from 203.119.25.1#53(a.dns.cn) in 2 ms

sina.com.cn.               60     IN     A       202.108.33.32
sina.com.cn.               86400  IN     NS      ns2.sina.com.cn.
sina.com.cn.               86400  IN     NS      ns3.sina.com.cn.
sina.com.cn.               86400  IN     NS      ns1.sina.com.cn.
;; Received 147 bytes from 61.172.201.254#53(ns2.sina.com.cn) in 28 ms
```

Fig. 8: Dig output for ”sina.com.cn”

5 Experimental and simulation results

5.1 Test of server selection method when NS cache misses

As shown in Section 2, when a recursive name server contacts the parent zone for NS records, it does not know its previous selection for the substitution by the child zone. What selection routine does the DNS software follow? To answer this question we analyze the selection results through a DNS request test.

We utilize dig - a utility for interrogating DNS servers which is typically only available on BIND-supported platforms. We use the trace option of dig to inhibit its default recursion and issue queries for the requested name to the root-servers and follow (and print) all referrals until an authoritative name server for the domain name is reached. This is similar to the cache miss effect of NS records. We generate a large enough workload of requests for a domain name by dig, record the selection results and count the number of selection for different servers.

The test is carried out under Linux and the caching NS software tested is BIND 9.2.1. We write a shell script to call dig command for ”sina.com.cn” 10,000 times and output the results to a file for statistics. The output of dig for one request is shown in Fig. 8.

In Fig. 8, we can see that there are three selections in the domain name resolution procedure: one for the root server, one for the .cn domain, and one for the sina.com.cn domain. Thus we can obtain three set of selection results in Fig. 9, 10 and 11 respectively.

Fig. 9, 10 and 11 show an approximately uniform distribution of queries to different name servers for all three NS resource record sets, which indicates that BIND 9.2.1 selects the server randomly when NS records cache misses.

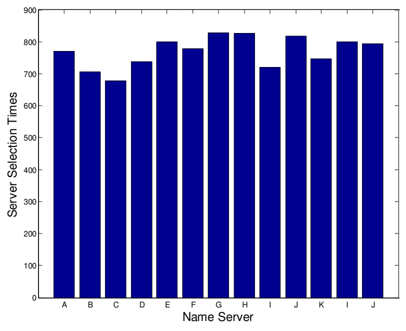

Fig. 9: Distribution of queries to root servers

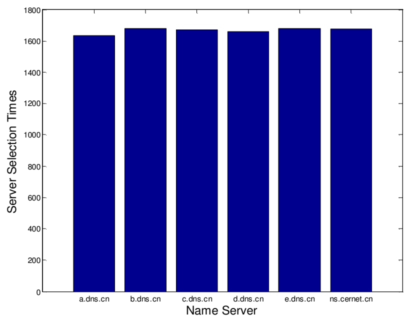

Fig. 10: Distribution of queries to cn servers

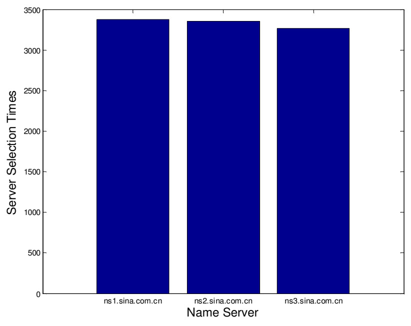

Fig. 11: Distribution of queries to sina.com.cn servers

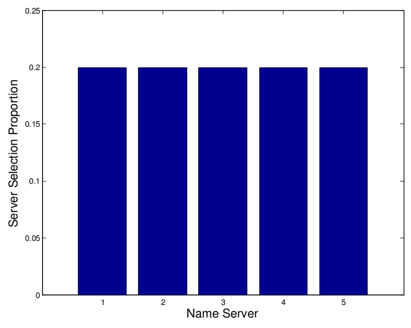

Fig. 12: Server selection proportion

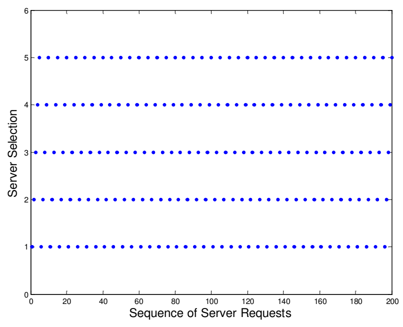

Fig. 13: Server selection sequence

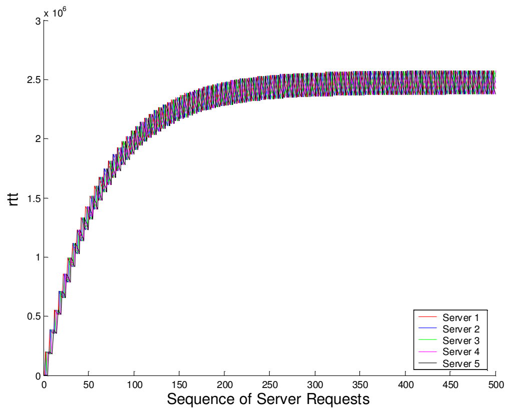

Fig. 14: rtt change (500 selections)

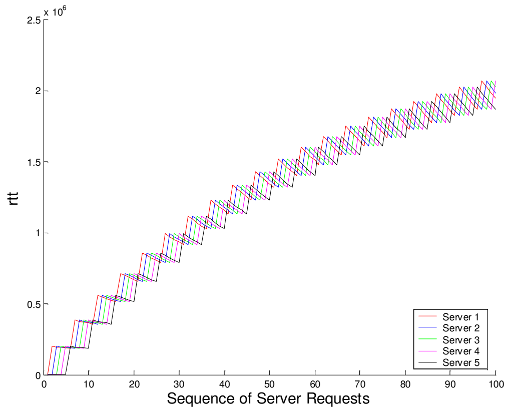

Fig. 15: rtt change (100 selections)

5.2 Parent zone selection algorithm simulation

To illustrate how the selection algorithm of the parent zone performs when all servers are unreachable, we present the simulation results such as server selection proportion, server selection sequence and rtt change. If the initial virtual rtt of server 1, 2, ..., n is rtt_1, rtt_2, ..., rtt_n respectively, let $n = 5$ and rtt_1, rtt_2, ..., rtt_5 is 1 ,2, 3, 4, 5 microsecond respectively which are relatively small values compared to the real ones.

Fig. 12 shows that the server selection proportions for 500 requests, which indicates that each server has an equal frequency of requests. Fig. 13 shows the result of the first 200 selections. And we can see that all servers are requested in round robin. This is due to the rtts maintained by the algorithm, which is shown in Fig. 14. In Fig. 14, after a round of polling all servers, the rtts climb steeply at a step of 0.2. The trend of rtts ascending continues until rtts reach 10 and then they fluctuate around 10. The rtts in the first 100 requests are shown in Fig. 15, which better illustrates the rtts ascending.

6 Conclusion

The Domain Name System (DNS) is a fundamental component of the modern Internet. In the DNS zone delegation, both parent and child zone file list the NS resource records for the delegated domain and this complicates the domain name resolution procedure especially for server selection.

In this paper, we analyzed the recursive resolution for the delegated domain and provide some recommendations such as NS record substitution mechanism. Furthermore, we analyzed the delegation effects on server selection as NS records are obtained through both parent and child zone and proposed an integrated sever selection solution as well as some recommendations as to the disposal of NS resource records and the maintenance of the rtt information of servers.

Acknowledgement

This work was supported in part by the National Key Technology R&D Program of China under the grant number 2012BAH16B00 and the National Science Foundation for Distinguished Young Scholars of China under the grant number 61003239.

References

[1] P. Mockapetris, Internet Request for Comments **RFC 1034**, (1987).
[2] P. Albitz and C. Liu, DNS and BIND, O'Reilly and Associates, (1998).
[3] H. Rood, Telecommunications Policy, **24**, 533-552 (2000).
[4] Name server DoS Attack October (2002), http://www.caida.org/projects/dns-analysis/.
[5] UltraDNS DOS Attack (2002), http://www.theregister.co.uk/2002/12/14/.
[6] Zheng Wang, KSII Transactions on Internet and Information Systems, **6**, 2750-2763 (2012).
[7] ICANN Factsheet for the February 6, 2007 Root Server Attack (2007), http://www.icann.org/announcements/factsheet-dns-attack-08mar07.pdf.
[8] Events of 21-Oct-2002 (2002), http://d.root-servers.org/october21.txt.
[9] DNS FAQ (2004), http://www.cs.cornell.edu/People/egs/beehive/faq.html.
[10] M. Handley and A. Greenhalgh, Proceedings of HotNets, **2005**, (2005).
[11] H. Yang, H. Luo, Y. Yang, S. Lu, and L. Zhang, Proceedings of DSN, **2004**, 83-93 (2004).
[12] K. Parka, V. Pai, L. Peterson, and Z. Wang, Proceedings of OSDI, **2004**, (2004).
[13] H. Ballani and P. Francis, Proceedings of HotNets, **2006**, (2006).
[14] R. Cox, A. Muthitacharoen, and R. Morris, Proceedings of IPTPS, **2002**, 155-165 (2002).
[15] V. Ramasubramanian and E. Sirer, Proceedings of SIGCOMM, **2004**, 331-342 (2004).
[16] T. Deegan, J. Crowcroft, and A. Warfield, Proceedings of CCR, **2005**, 5-13 (2005).
[17] J. Kangasharju and K. Ross, Proceedings of INFOCOM, **2000**, 660-669 (2000).
[18] E. Cohen and H. Kaplan, Proceedings of SAINT, **2001**, 85-94 (2001).
[19] J. Jung, E. Sit, H. Balakrishnan, and R. Morris, IEEE/ACM Transactions on Networking, **10**, 589-603 (2002).
[20] Million-PC botnet threatens consumers (2006), http://www.infomaticsonline.co.uk/vnunet/news/2167474/million-pc-botnet-threatens.
[21] L. Kleinrock, Queueing Systems, Wiley-Interscience, **2**, (1976).
[22] Ziqian Liu, Presentation at the 2nd DNS-OARC Workshop (2009), https://www.dns-oarc.net/files/workshop-200911/Ziqian_Liu.pdf.
[23] Bind website, http://www.isc.org/products/BIND/.
[24] Zheng Wang, Xin Wang and Xiao-Dong Li, International Journal of Innovative Computing, Information and Control, **6**, 5131-5142 (2010).

Zheng Wang received the MS degree in Electrical Engineering from Institute of Acoustics, Chinese Academy of Sciences in 2006, and the PhD degree in Computer Science from Computer Network Information Center, Chinese Academy of Sciences in 2010. He is currently the director of Joint Labs in China Organizational Name Administration Center. His research interests are in the areas of network architecture, Domain Name System, and information systems.

Rui Wang received the BE degree in Electrical Engineering from Beihang University in 2002, and the MS degree in Electrical Engineering from Peking University in 2012. He is currently the vice director of China Organizational Name Administration Center. His research interests includes network system and applications, e-goverment and Domain Name System.