academicJournals

Vol. 9(13), pp. 619-624, 15 July, 2014
DOI: 10.5897/SRE2013.5740
Article Number: 9EC0F2146025
ISSN 1992-2248 © 2014

http://www.academicjournals.org/SRE

Scientific Research and Essays

Full Length Research Paper

Estimation of domain name system (DNS) server load distribution

Zheng Wang

Computer Network Information Center, Chinese Academy of Sciences, Beijing, China.

Received 18 November, 2013; Accepted 12 June, 2014

Domain name system (DNS) resolution service is usually provisioned by multiple authoritative servers for performance and robustness. Estimating the query load distribution among multiple authoritative servers is one of the key issues arising with DNS server load balancing and optimization. We propose an analytical model of Round-Trip-Time (RTT)-sensitive server selections consisting of cache servers, authoritative servers and clients, which makes it possible to infer DNS server load accurately. A DNS server fingerprint approach is then proposed to identify RTT-sensitive server selections from BIND's. Finally, we present a server load estimation method based on server selection classification. Under BIND server selection algorithm, the solution of the server selection model is obtained using iteration method, which is validated by the simulation results.

Key words: Round-Trip-Time, domain name system (DNS) server fingerprint, server selection, load distribution estimation.

INTRODUCTION

The domain name system (DNS) is one of the most fundamental components of the today's Internet, providing a critical link between human users and Internet locations by mapping host names to IP addresses. To enhance the resilience, reliability and scalability of DNS authoritative service, the DNS authoritative data is usually stored at multiple geographically distant servers. Each authoritative server maintains the same DNS zone data. Requests by clients are first served by their cache servers, which then forward the cache missed requests to one of these authoritative servers.

For each emitting query, the cache server decides which authoritative server is the destination per its server selection algorithm. As the DNS specifications (Mockapetris, 1987) are vague on server selection algorithms, current cache server implementations show different effects in their query distribution among a set of authority servers. BIND is by far the most popular cache server implementation in use. It adopts a Round-Trip-Time (RTT) proportional server selection algorithm favoring the small-latent servers. But most of the alternative implementations exhibits sub-optimal server selection behavior, distributing queries evenly among all authoritative servers.

Given potential DNS request volume and distribution from cache servers, the planning and design of the

*Corresponding author. E-mail: wangzheng@conac.cn; zhengwang09@126.com

number, location, capacity of DNS authoritative servers largely depends on their projected server load distribution. Server load distribution estimation may at least help to achieve a balanced utilization among DNS authoritative servers, preventing the occurrence of such a situation where some servers are overloaded or even overwhelmed while some are underloaded. In case that a particular server is regarded as overloaded and a new server is expected to be added to share its excessive traffic, server load distribution estimation can play a critical role in optimally locating the new server and evaluating its effects of load rebalance. We can also consider the importance of server load distribution estimation when predicting the impacts of DoS/DDoS attacks towards DNS servers. Server load distribution estimation makes it possible to predict precisely how a particular server is flooded by an attack launched from a set of particular cache servers.

This paper provides a server load estimation method based on server selection classification: for RTT-sensitive server selections, an analytical model is proposed; for RTT-sensitive server selections, a DNS server fingerprint approach is proposed to identify them.

RELATED WORK

Many previous efforts on server load estimation were focused on web servers or web services (Zhichun et al., 2010; Vercauteren et al., 2007; Jiani and Laxmi, 2006). While web servers or web services are characterized by handling users' requests for interdependent and dynamic content, communication between cache servers and authoritative servers can be seen as consisting of simple and short queries, therefore DNS server selection plays a more important role in server load estimation.

Previous studies on DNS server selection are limited. Zheng et al. (2010) provided the analytical performance evaluation of BIND DNS selection algorithm. The results are the component basis of this paper. Yingdi et al. (2012) conducted a series of trace-driven measurements to understand the server selection performance of current cache server implementations. The results validate that BIND implements a RTT-proportional server selection algorithm. Supratim et al. (2008) proposed a server selection algorithm using auto-regression models for estimating the server response times. The accuracy of SRTT estimation can be improved by the techniques. Ager et al. (2010) provided measurement study on the responsiveness of DNS in two aspects (1) the latency between clients and DNS cache servers, (2) the content of the DNS cache when the query is issued.

The major security concerns about DNS are the ever-lasting threats imposed by Denial of Service (DoS) attacks. Marios et al. (2013) examined a DNSSEC-powered amplification attack feathered by independence of botnet and undisclosed attackers. However, the impacts of such amplification attack bear little direct relation to server load distribution because the amplified flooding responses from multiple servers are virtually aggregated by cache servers before being forwarded towards victim end users. A robust counter measure against this type of threats is proposed based on Bloom filters (Sebastiano and Dario, 2011). It is deployed at the side of victim end users, thus irrelevant to cache server's and authoritative server's behavior.

The DNS system are extensively measured and examined in recent years. Kyle et al. (2013) presented methodologies for efficiently discovering the complex client-side DNS infrastructure. Craig and Andrew (2013) examined DNS resolver behavior and usage, from query patterns and reactions to nonstandard responses to passive association techniques to pair resolvers with their client hosts. Hongyu et al. (2013) provided some new findings in DNS traffic patterns and proposed a novel approach that detects malicious domain groups using temporal correlation in DNS queries. Thomas et al. (2013) passively monitored DNS and related traffic within a residential network in an effort to understand server behavior--as viewed through DNS responses, and client behavior--as viewed through both DNS requests and traffic that follow DNS responses.

Analytical model of RTT-sensitive server selections

We consider a network consisting of *M* local DNS cache servers (or server selection nodes) and N distributed authoritative servers. Each cache server *j* receives DNS requests generated by its clients and if not hit in the cache forwards them to the authoritative servers. Inspired by Jaeyeon et al. (2003) we assume that the requests sent by each cache server *j* follow a Poisson process with rate λj. It is a simplifying assumption about the inter-arrival times of requests and facilitates our analysis without much loss of generality. The Poisson arrival assumption is also theoretically justified by the fact that it represents the aggregation of requests made by a large population of clients (Karlin and Taylor, 1975). Measurements of request arrivals have been shown to match well a Poisson process, at least over small to moderate timescales (Villela et al., 2007). Each cache server *j* assigns requests to server *i* with proportion p_{ji}, independent of other requests, therefore the arrival process to each authoritative server *i*, $i = 1,2,\ldots,N$ is an independent Poisson process with rate $p_{ji}\lambda_j$. Figure 1 shows the scenario.

We assume that the service time distribution at each server *i* is arbitrary with mean $\bar{x}_i$. The mean service rate of server *i* is $\mu_i = 1/\bar{x}_i$. We assume that each server implements a Processor Sharing (PS) scheduling policy, where all the requests share the server's capacity equally and continuously (Kleinrock, 1976). Under the above

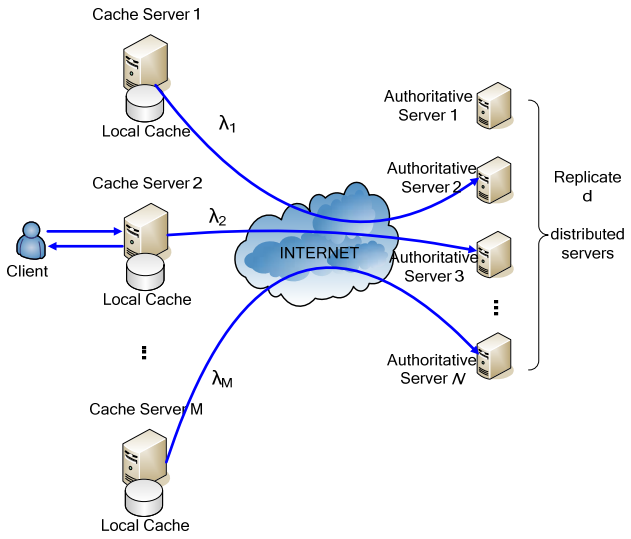

Figure 1. RTT-sensitive server selection model.

assumptions, each server behaves as an-PS queue. Let $\mathbf{P}_i = (p_{1i}, p_{2i}, ..., p_{Mi})'$ denote the server selection proportion vector for authoritative server *i* and $\boldsymbol{\lambda} = (\lambda_{1i}, \lambda_{2i}, ..., \lambda_{Mi})'$ the requests rate vector of the cache servers. Then for a given $\mathbf{P}_i$ in such a network, the average delay of a request forwarded to server is given by the following expression (Kleinrock, 1976):

$$\bar{T}_i(\mathbf{P}_i) = \frac{1}{\mu_i - \sum_{j=1}^{M} p_{ji}\lambda_j} = \frac{1}{\mu_i - \mathbf{P}_i'\boldsymbol{\lambda}} \quad (1)$$

Query delay can be approximated as the sum of network latency (largely dependent of network topology) and processing delay (related to the ratio of the name resolution service capability of the authoritative servers to the query arrival rate). So the overall query delay between the cache sever server *j* and the authoritative server *i* is:

$$rtt'_{ji} = rtt_{ji} + \bar{T}_i(\mathbf{P}_i) \quad i = 1,2,...N \quad j = 1,2,...M \quad (2)$$

Where rtt_{ji} is the network latency. Thus rtt'_{ji} is the function of $\mathbf{P}_i$, or

$$rtt'_{ji} = rtt'_{ji}\left(\bar{T}_i(P_i)\right) = rtt'_{ji}(P_i) \quad i = 1,2,...N \quad j = 1,2,...M \quad (3)$$

To minimize the delays experienced by users, RTT-sensitive server selection algorithm (e.g., BIND) prefers the least-latent authoritative server (for good response time) yet still queries the others (to distribute the load and monitor their performance). Latencies between the cache server and the set of authoritative servers are the only metric on which RTT-sensitive algorithm is based. Therefore, query latency impacts significantly on DNS server load distribution. So the server selection proportion is a function of the overall request delay rtt'_{ji}, $i = 1,2,...N$, $j = 1,2,...M$. Consider an element p_{ji} of $\mathbf{P}_i$, $i = 1,2,...N$, $j = 1,2,...M$, it is dependent of the overall delay from the cache sever server *j* to each authoritative server. Therefore each $\mathbf{P}_i$, $i = 1,2,...N$ is decided by total space of overall delay, or

$$\mathbf{P}_i = \mathbf{P}_i\left(rtt'_{ij}, i = 1,2,...N, j = 1,2,...M\right) \quad i = 1,2,...N \quad (4)$$

Equations (3) and (4) indicate the interrelation among the overall delay and the server selection proportion. Although it is possible to express Equations (3) and (4) in the definite form, which actually establishes equation set, we still find it hard to provide a closed-form expression for the solution. We also note that cache server *j*, $j = 1,2,...M$ has no information other than its overall delay rtt'_{ji}, $i = 1,2,...N$. So its server selection reaction is only based on its own overall delay sector but not the total space of overall delay.

Fingerprint RTT-Insensitive server selections

While BIND is by far the most widely used DNS cache server implementations accounting for 53.9% of the total (Infoblox, 2010), there are still other implementations in use. The most popular implementations other than BIND include DNS Cache, Unbound, and Windows DNS. Unlike BIND which shows kind of RTT-proportional server selection, such implementations are RTT-insensitive. This means queries are distributed evenly among all the authority servers. Such sub-optimal server selection makes the problem of sever load estimation simpler, because load distribution is predetermined irrespective of query processing delay affected by load distribution. As RTT-insensitive implementations behave quite differently from BIND and so do their server load distributions, the overall estimation accuracy may be significantly degraded by neglecting RTT-insensitive sever selection or regarding all as BIND's RTT-sensitive. So a key problem arises on how to fingerprint RTT-insensitive server selections and effectively identify them from BIND's counterparts.

A DNS fingerprinting tool is available on (fpdns. https://github.com/kirei/fpdns), which is very accurate and

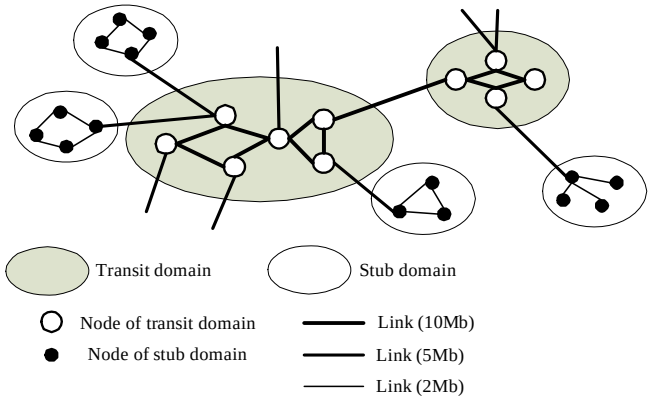

Figure 2. Illustration of network topology.

covers a wide range of DNS implementation types and versions. Its methodology used to identify individual name server implementations is based on "borderline" protocol behaviour. DNS implementations adhere to the well documented standard protocol behaviour in the case of 'common' dns messages, but the DNS protocol also offers a multitude of message bits, response types, opcodes, classes, query types and label types in a fashion that may diversify implementations. The tool uses series of "borderline" query-response messages to identify implementations.

ESTIMATION METHODS

To estimate load distribution, the first step is to identify cache resolvers implementing RTT-insensitive server selection from those implementing BIND's server selection. In this step, a DNS fingerprinting tool may be used to actively probe all cache resolvers that are expected to query the investigated servers. The perceived implementations other than BIND is classified as the RTT-insensitive server selection originators. We then estimate server loads under BIND and RTT-insensitive selection respectively.

For the BIND-based server load estimation, we must integrate BIND server selection algorithm to solve the simultaneous equations consisting of Equations (1), (2) and (4). BIND name servers use RTT to choose between name servers authoritative for the same zone. Each time a BIND name server sends a query to a remote name server, it starts an internal stopwatch. When it receives a response, it stops the stopwatch and makes a note of how long that remote name server took to respond. When the name server must choose which of a group of authoritative name servers to query, it simply chooses the one with the lowest RTT. Before a BIND name server has queried a name server, it gives it a random RTT value, but lower than any real-world RTT. This ensures that the BIND name server queries all of the name servers authoritative for a given zone in a random order before playing favorites. On the whole, this simple but elegant algorithm allows BIND name servers to "lock on" to the closest name servers quickly and without the overhead of an out-of-band mechanism to measure performance. For server S_1, S_2, …, S_n with their RTTs as $rtt_1, rtt_2, \ldots, rtt_n$ respectively, under BIND server selection algorithm, Equation (4) can be instantiated as

$$1 : \log \frac{rtt_1}{rtt_1 * \alpha + rtt_i * (1-\alpha)} / \log \beta : , \ldots , : \log \frac{rtt_1}{rtt_1 * \alpha + rtt_n * (1-\alpha)} / \log \beta \quad (5)$$

The estimated server load distribution under BIND server selection algorithm can be computed by solving simultaneous equations consisting of Equations (1), (2) and (5). For the RTT-insensitive server load estimation, each cache server j assigns requests to server i always with an equal proportion

$$P_{ji} = 1 / N \quad (6)$$

Finally, we can get the overall server load distribution by adding together the BIND-based server load estimation and the RTT-insensitive server load estimation.

Simulations

As RTT-insensitive server load estimation is simple and easy to implement, we only conduct simulations using the BIND server selection algorithm and PS scheduling policy to illustrate the model in this area. To solve the seemingly complicated nonlinear equations, we propose an iteration method to obtain the solution. At the start of iteration, we assume that where the processing delay of the authoritative servers is negligible compared to the network path delay (or in an idle state). So the initial query load distribution can be obtained per Equation (5) given all cache servers' query rates and distances from each authoritative server. Then for the next iteration, each authoritative server's processing delay is taken as calculated in the previous iteration. The query load distribution is updated based on the authoritative server's processing delay and the network path delay between cache servers and authoritative servers. Repeat the process for more iterations until the query load distribution's change is negligible and the server selection reaches a steady state.

We implement the server algorithm in the NS2 simulator. The network topology is generated by GT-ITM Topology Generator provide by NS2. The topology parameters are given as the following (Figure 2): three-level hierarchy: transit domain, stub domain and nodes; 2 transit domains; each transit domain has (on average) five nodes; each transit node connects (on average) to five stub domain; each stub domain has (on average) twenty nodes; each cache server coverage is 10% of the nodes except the 10 transit nodes. We generate

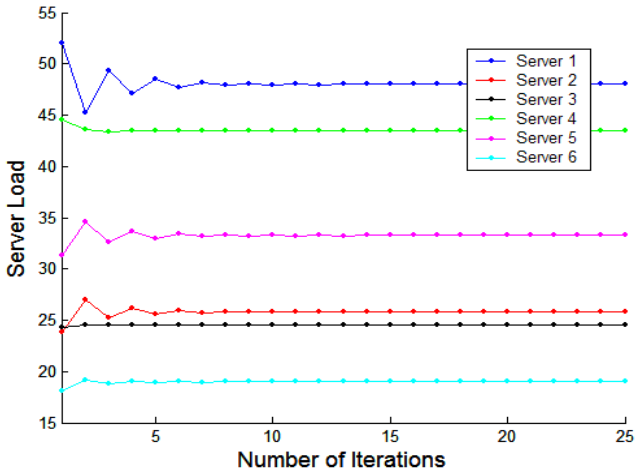

Figure 3. Solving BIND server selection model by iteration.

200 nodes, randomly select 6 nodes as the authoritative servers and the remaining nodes as cache server (Figure 3).

We apply a patch for NS2 (Kostas, 2004) as Poisson traffic generator for cache servers. All cache servers follow the same traffic pattern. The mean packet inter-generation time is set as 1.15 ms, and the size of the packet generated is 256 bytes. Cache servers distribute their queries among all authoritative servers per BIND server selection algorithm. The network latency between each pair of cache server and authoritative server is solely dependent of the network distance in the generated topology. In our simulation, it ranges from 30 to 200 ms. The DNS resolution processing at each authoritative server implements PS scheduling policy and the mean service rate is set as 100 kqps. Once the incoming query rate is aggregated from all cache servers, the query processing delay at an authoritative server is computed and set per PS scheduling policy. Then cache servers may adjust its query load distribution according to the updated RTTs. In response to the updated query load distribution from cache servers, authoritative servers then renew their query processing delays. So the RTTs change accordingly, and so on. The simulation lasts for enough time until we see the dynamic converges to a stable state. The result of iterations is shown in Figure 3. We can see that the query load distribution for each authoritative server quickly converges to a steady value. To see whether the iteration like simulation does find the correct solution formulated in Equations (1) to (5). We plug the convergence values in Figure 3 obtained by simulation into Equations (1) to (5). Their consistency with Equations (1) to (5) shows their correctness.

Conclusions

Estimating the query load distribution among multiple DNS authoritative servers is one of the key issues arising with DNS server load balancing and optimization. The contribution of this paper can be summarized as: (1) An analytical model of RTT-sensitive server selections is proposed; (2) A server load estimation method is proposed based on server selection classification, which uses a DNS server fingerprint approach to identify RTT-sensitive server selections from BIND's; (3) Under BIND server selection algorithm, an iteration method is proposed to solve the server selection model, which is validated by the simulation results.

Conflict of interests

The author(s) have not declared any conflict of interests.

ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation of China (No. 61003239), Beijing Institution of Higher Learning "Young Talents Plan", Beijing Natural Science Foundation (No. 4144084), and the National Science Foundation for Young Scholars of China (No. 61102057).

REFERENCES

Ager B, Muhlbauer W, Smaragdakis G, Uhlig S (2010). Comparing DNS resolvers in the wild. Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement (IMC'10). pp. 15-21.

Craig AS, Andrew JK (2013). Resolvers revealed: characterizing DNS resolvers and their clients. ACM Trans. Internet Technol. 12(4): Article 14.

Hongyu G, Vinod Y, Yan C, Phillip P, Shalini G, Jian J, Haixin D (2013). An empirical reexamination of global DNS behavior. Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM (SIGCOMM'13). pp. 267-278.

Infoblox (2010). DNS SURVEY: OCTOBER 2010.

Jaeyeon J, Arthur WB, Hari B (2003). Modeling TTL-based Internet caches. Proc. IEEE INFOCOM 1:417-426.

Jiani G, Laxmi NB (2006). Load balancing in a cluster-based web server for multimedia applications. IEEE Trans. Parallel Distributed Syst. 17(11):1321-1334. http://dx.doi.org/10.1109/TPDS.2006.159

Karlin S, Taylor H (1975). A first course in stochastic processes. Academic Press.

Kleinrock L (1976). Queueing systems, Wiley. Bus. Econ. 448 pp.

Kostas P (2004). Poisson traffic generator for ns-2. http://www.matlab.nitech.ac.jp/~khpoo/research/index-poisson.htm.

Kyle S, Tom C, Michael R, Mark A (2013). On measuring the client-side DNS infrastructure. Proceedings of the 2013 conference on Internet measurement conference (IMC'13). 77-90.

Marios A, Georgios K, Panagiotis K, Georgios L, Stefanos G (2013). DNS amplification attack revisited. Comput. Secur. 39:475-485. http://dx.doi.org/10.1016/j.cose.2013.10.001

Mockapetris P (1987). Domain Names - Concepts and Facilities. IETF RFC 1034.

Sebastiano DP, Dario L (2011). Protecting against DNS reflection attacks with Bloom filters. Proceedings of the 8th international conference on Detection of intrusions and malware, and vulnerability assessment (DIMVA'11). pp. 1-16.

Supratim D, Anand S, Pavan KP (2008). An improved DNS server selection algorithm for faster lookups. Proceedings of the Third

International Conference on Communication System Software and Middleware (COMSWARE'08). pp. 288-295.
Thomas C, Mark A, Michael R (2013). On modern DNS behavior and properties. SIGCOMM Comput. Commun. Rev. 43(3):7-15. http://dx.doi.org/10.1145/2500098.2500100
Vercauteren T, Aggarwal P, Xiaodong W, Ta-Hsin L (2007). Hierarchical forecasting of web server workload using sequential monte carlo training. IEEE Trans. Signal Proc. 55(4):1286-1297. http://dx.doi.org/10.1109/TSP.2006.889401
Villela D, Pradhan P, Rubenstein D (2007). Provisioning servers in the application tier for e-commerce systems. ACM Trans. Internet Technol. 7(1): Article 7.
Yingdi Y, Duane W, Matt L, Lixia Z (2012). Authority server selection in DNS caching resolvers. ACM SIGCOMM Comput. Commu. Rev. 42(2):80-86. http://dx.doi.org/10.1145/2185376.2185387
Zheng W, Xin W, Xiao-Dong L (2010). Analyzing BIND DNS server selection algorithm. Int. J. Innov. Comput. Inf. Cont. 6(11):5131-5142.
Zhichun L, Ming Z, Zhaosheng Z, Yan C, Albert G, Yi-Min W (2010). WebProphet: automating performance prediction for web services. Proceedings of the 7th USENIX conference on Networked systems design and implementation (NSDI'10). pp. 10-10.