Quality of Service on Services Selection using Anycast Techniques: a Convergence Analysis

Lucas J. Adami *, Julio C. Estrella *, Stephan Reiff-Marganiec † Rafael M. de O. Libardi *,
* *University of São Paulo (USP)*
Institute of Mathematics and Computer Science (ICMC), São Carlos-SP, Brazil
Email: {ljadami, jcezar}@icmc.usp.br
† *University of Leicester*
University Road, Leicester, LE1 7RH - UK
Email: srm13@le.ac.uk

***Abstract*—Anycast is an effective technique to select system servers among many and by spreading client requests achieve high performance and scalability. In our previous work, we presented the Global Application Layer Anycast (GALA) system. By combining real time network distances measures and geolocation, it performed better than its inherited algorithm, GAA. In this work we analyze how fast the GALA algorithm can find the best server to attend a client request and comparing its convergence to GAA. Using simulations, we show that it converges much faster and we propose a maximum selection time metric to be used in the selection process. Experimental results reveal that the GALA algorithm is two times better than the GAA considering the metric proposed.**

***Keywords*-anycast; service selection; quality of service; distributed systems;**

I. Introduction

In systems with high demand, a set of servers are used to attend to the clients requests. That way, the load is spread through these nodes and the systems scalability is significantly improved. Among the techniques used for this goal, anycast as proposed by Partridge et al. [1] is an effective one. The main idea is to map servers to the same IP address. Then, the clients send requests to the mapped IP and the network is responsible to route the packets to the best server using metrics, such as router hops, as parameters.

Some works [2] [3] [4] exploring the application layer anycast predict the distance between the client and a server to select a node. However, researches [5] [6] show that these predictions can influence the selection accuracy resulting in inaccurate choices. Some works [7] [8] use real measures (round trip time, RTT) to find nearby servers, but they also have problems that affect the overall performance. Yet, the flexibility of the metrics used is not explored by any of the cited works. They are limited by the node distances and server loads.

In previous work [9], we proposed an application layer anycast system that combined both predictions and real time measures of the distance between nodes. It is called GALA, Global Application Layer Anycast. The results showed that our system was better than a previous one, GAA, that uses only real time measures. In this paper, our contribution is an analysis of the convergence speed of the GALA algorithm and a new metric proposal for the application layer anycast systems, the maximum selection time. The simulation results show that the majority of the requests in GALA already converge in 150 milliseconds while in GAA it takes approximately 450 milliseconds to reach this state. The results show that under this metric GALA significantly outperforms GAA due to a much faster convergence.

II. Related Work

We classify available application layer anycast systems in prediction systems and real measures systems. Many systems predict server distances to select the best node available. OASIS [2] (Overlay-based Anycast Service Infrastructure) classifies a server locality using IP prefixes. Proxima [3] is based on network coordinates (NCs). NCs are an estimate of the location of a node in a network. DOAT [4] (Distributed Overlay Anycast Table) uses spacial filling curves combined with NCs to find close system nodes. As cited before, even though the prediction can improve the anycast system performance, it can result in inaccurate server selections [5]. According to the research of Xing et al., the network topology further influences the prediction accuracy [6].

There are also anycast systems that use real time measures to find nearby system nodes. They use RTT as a metric. AICN [7] (Anycast for Information Centric Network) combines IP layer anycast and application layer anycast, also considering server loads in the server search. However, by using IP layer anycast, the system inherits all its drawbacks, such as the need of special routers for anycast packets routing. GAA [8] (Global Application-layer Anycast) groups its servers in concentric rings. The main problem of this architecture lays in the initial server selection. When a client requests a service, a random server node is chosen to route their request. Because the servers are globally spread, if a

distant initial node is chosen, the algorithm takes more steps to converge. Thus, the total request time increase.

III. Design Details

Figure 1 illustrates the GALA system. An anycast query comprises six main steps. First, the client retrieves a list of available resolvers (1). Then, using this list, it sends a query to the first resolver (2) (in order they appear on the list). After that, the resolver searches for the queried name in a domain names server (3). The answer contains an IP address of a valid system server. Using this initial IP, the resolver iteratively probes this server and its neighbours until a good server is found (4). The chosen server is returned to the client which invokes the service directly to the selected node (5). Finally the node will receive the request and process the result consulting service data if necessary (6).

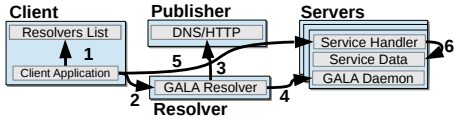

Figure 1. GALA system overview.

A. Clients

The client is responsible to send anycast requests to the GALA system and receive the responses in a transparent way. An anycast request is a domain name query for a name in the form **.any*. The client application is responsible to generate such requests.

B. Publishers

Publishers are nodes that contain the system servers location data. In our proposal, we use a DNS server.

We know that the geographic distance between nodes is correlated to the network distance (RTT) [9]. Therefore, to map the servers, we use geolocation. There are many available geolocation databases on the Internet. For GALA, we chose MaxMind. Comparative studies [10] [11] showed that MaxMind has the best results comparing to the other free databases.

The servers are mapped into regions (grid). This mapping is done using the formulas described in [9]. Each area has a representative name in the domain name system, in the form *xXyY.service.url*. System servers are aware of this mapping and know which area they belong to. Thus, once they enter the system, they send register requests to the DNS server in these mapped areas. That way, each area name registry will contain all servers within that region.

C. Resolver

The GALA resolver application receives domain name queries from clients and answers them. If the name to be resolved is a traditional one, the query is resolved using the system traditional routines. If the name to be resolved is an anycast name, it processes it in four main steps.

First, the GALA resolver consults the geolocation database that contains IP addresses and their respective latitude and longitude locations. By knowing the client location, the resolver builds a domain name based on these values, using the equations cited before. Then, the GALA resolver asks the publisher for the mapped domain name records. Using the list returned, the resolver randomly chooses an initial one and starts the search algorithm described in [9].

Then, after selecting a server, the resolver returns it as an answer to the client application, which invokes the service directly on the server.

D. Servers

Servers offer the service to the clients. They typically have three modules: the service handler, the service data and the GALA daemon. The service handler is the service application itself. The service data is the information stored in the node that is required for the execution of the service. Finally, the GALA daemon is responsible for integrating the node with the entire system.

The GALA daemon is responsible for registering the server into the system, update neighborhood information and answer resolver messages.

GALA servers only keep information of a portion of system nodes. These nodes are called neighbors. The neighborhood is formed by concentric rings. Further servers fall in outer rings and closer servers fall in inner rings. The distance is defined by the RTT.

IV. Performance Evaluation

In our previous work [9] we showed that GALA results in better service execution, decreasing the number of hops and total request time. Also, it maintained the selection efficiency if compared with GAA. We consider the efficiency as the ratio of the real closest server distance and the distance of the server found by the system.

A. Experimental Setup

We executed a complete factorial experiment, with two levels for each factor. For the results analysis, we used linear regression. Also, we generated the graphics using the Minitab software [1].

With the performance evaluation, we wanted to show the convergence of the GALA algorithm compared to the GAA algorithm. The convergence is important because the faster an algorithm converges, the faster it can resolve requests within a time limit, respecting possible request time limit constraints. To do so, we executed many experiments using the same simulator of our previous work. We used two factors, each one with two levels. The factors used are

[1]Minitab: http://www.minitab.com

algorithm and *input data* with levels *GAA/GALA* and *PlanetLab/Geonames*, respectively. On the first input configuration, we use PlanetLab nodes as servers and Geonames as clients. On the second, we swap their roles.

To show convergence, we consider histograms to analyze the distribution of the requests total times. In addition, by using a cumulative histogram, we plot a fitted normal demonstrating the percentage of requests that already finished over time.

B. Results

Figure 2 shows the histogram of the request total latencies. The first panel shows the result for the first input configuration, while the second shows the result for the second input configuration.

In our simulation, we do not simulate service execution. Thus, we consider total latency as the time from client requesting a service until the answer of the system with the best node available. The x axis represents the observed latencies. The y axis shows the total amount of requests that took the observed time to select the best server, as percentage.

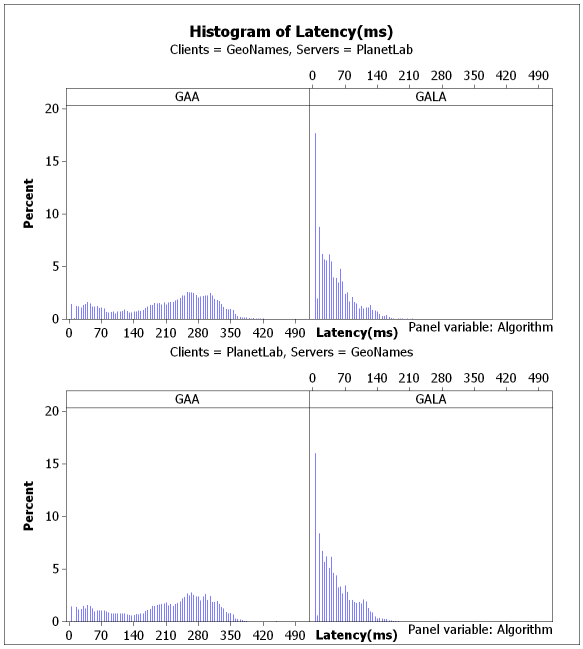

Figure 2. Histogram of latency.

It is possible to observe the randomness of the request times for the GAA algorithm. This happens because of the initial server selection, which is done randomly. On the other hand, GALA observations tend to accumulate near the origin. This behaviour is the expected one, as the GALA algorithm tries to use nearby servers to start the search process. Also, the input data change does not affect the overall observations much.

Figure 3 shows the accumulated frequencies of the request latencies. The graph curves are fitted normals and were generated by Minitab. Each panel shows the accumulated frequencies for an input configuration. The x axis represents the observed latencies. The y axis shows the total amount of requests that took the observed time or less to select the best server, as percentage.

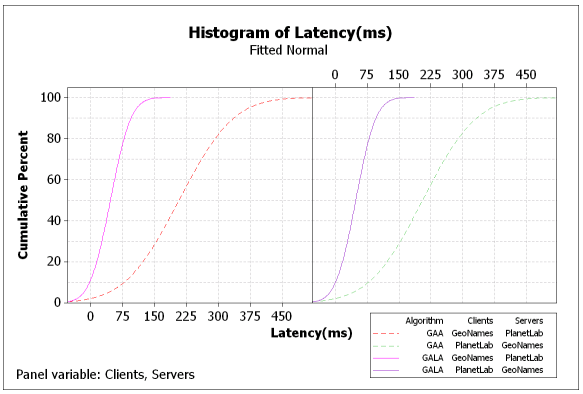

Figure 3. Accumulated histogram of latency.

This Figure illustrates clearly that GALA reaches its convergence in much less time than GAA. For example, considering the left panel, 100% of the requests already converges by 150 milliseconds, while, in GAA, this percentage is near 30%. This fact repeats for the right panel. Also, the input once again did not influence the algorithms behaviour significantly.

C. Limiting the Maximum Selection Time

Service Level Agreements (SLAs) are a contract between the server and the client involving some quality of service parameter.

In the application layer anycast systems studied [4] [2] [3] [7] [8] [12], the authors limited the use of the metrics for selecting a server by means of network distance and server load. In our system, we propose the use of a well-known SLA, the maximum server selection time. This metric is defined by the client and tells how much time the system has to select a server to attend the client request.

This new metric fits well on our system because it is based on iteration. Also, the system transparency achieved does not prevent its usage. Clients send anycast queries using the domain names in the format of *domain.name.any*. If they want to define a maximum selection time, they query the following: *domain.name.any%mst=1000*. MST stands for maximum selection time and 1000 means that the algorithm has one second to select the best server.

The system resolver will parse the query string and will notice the SLA appended to the anycast domain name. Then, on the search algorithm, it will take action so a server is returned before the time expires. To do so, it sets a timer that

expires on the given MST. When the time has passed, the currently selected server (in the current iteration) is returned.

We analyzed how the GALA system behaves using this SLA, compared to GAA. Table I shows possible maximum selection times and how many requests GALA and GAA systems would attend with the best known server. It considers both input data configurations.

MST (ms)	% of Requests GALA	% of Requests GAA
25	36.1119	4.3399
75	79.9391	17.0272
125	95.5922	24.5248
175	99.7261	32.5492
225	100	48.0782
325	100	91.7416
350	100	97.3455
375	100	99.3375

Table I
MAXIMUM SELECTION TIMES.

It is important to note that the best known server does not necessarily mean the best server. In our previous work we showed that both GALA and GAA have approximately 95% of accuracy. In addition, if the server is unable to converge before the selection time expires, it does not mean that the selection failed. It will return the currently selected server, which is the best one for the current algorithm iteration.

Analyzing the Table, it becomes clear that GALA can attend a client SLA better than GAA. By a 125 MST value, it can comply with the quality of service on 95% of the client requests, while in GAA, this value is under 25%. GAA will reach its optimal state with an approximate 350 MST value, being able to attend 99% of the requests. GALA will offer the same quality with a 175 MST value, exactly half of the time for GAA.

V. CONCLUSION

As we discussed, the GALA algorithm converges much faster than GAA. This happens because GALA will always consider close servers to the client by using geolocation, so the probed servers by GALA are closer which decreases the resulting request total time.

Yet, we added an SLA metric on the server selection, the maximum selection time. Knowing that our algorithm converges faster than GAA, we only confirmed its results by presenting a table with MST examples. Overall, GALA was two times better than GAA considering this metric. Furthermore, we affirm that the metric added to the system fits well on application layer anycast systems, especially ours.

For future work, we plan to deploy the GALA system in a real environment. Then, we will also consider server loads on the selection algorithm. The deployment experiment will help us confirm our algorithm efficiency.

ACKNOWLEDGMENT

We would like to thank FAPESP (Fundacão de Amparo à Pesquisa do Estado de São Paulo) for the support of this research. Also, we would like to thank ICMC-USP (Institute of Mathematics and Computer Science) and the LaSDPC (Distributed Systems and Concurrent Programming Laboratory) for offering the necessary equipments and laboratories for this study.

REFERENCES

[1] T. M. C. Partridge and W. Milliken, "Host Anycasting Service," 1993, rFC 1546.

[2] M. J. Freedman, K. Lakshminarayanan, and D. Mazières, "OASIS: anycast for any service," in *Proceedings of the 3rd conference on Networked Systems Design & Implementation - Volume 3*, ser. NSDI'06. Berkeley, CA, USA: USENIX Association, 2006, pp. 10–10.

[3] G. Wang, Y. Chen, L. Shi, E. K. Lua, B. Deng, and X. Li, "Proxima: Towards Lightweight and Flexible Anycast Service," in *INFOCOM Workshops 2009, IEEE*, 2009, pp. 1–2.

[4] E. Mykoniati, L. Latif, R. Landa, B. Yang, R. Clegg, D. Griffin, and M. Rio, "Distributed overlay anycast tables using space filling curves," in *INFOCOM Workshops 2009, IEEE*, 2009, pp. 1–6.

[5] R. Zhang, C. Tang, Y. Hu, S. Fahmy, and X. Lin, "Impact of the inaccuracy of distance prediction algorithms on internet applications - an analytical and comparative study," in *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, 2006, pp. 1–12.

[6] C. Xing and M. Chen, "Impact of network topology on distance prediction accuracy," in *Networking, Sensing and Control, 2008. ICNSC 2008. IEEE International Conference on*, 2008, pp. 1425–1429.

[7] S. Yuan, T. Lin, G. Zhang, Y. Li, H. Tang, and S. Ci, "A future anycast routing scheme for information-centric network," in *Communications (APCC), 2012 18th Asia-Pacific Conference on*, 2012, pp. 173–178.

[8] Z.-Y. Ma, J. Zhou, and L. Zhang, "A Scalable Framework for Global Application Anycast," in *Information and Computing Science, 2009. ICIC '09. Second International Conference on*, vol. 1, 2009, pp. 250–253.

[9] L. J. Adami, J. C. Estrella, E. M. de Oliveira, and S. Reiff-Marganiec, "Providing Quality of Service on Services Selection using Anycast Techniques," in *IEEE SCC, 2014. 11th IEEE International Conference on Services Computing*, 2014.

[10] Y. Shavitt and N. Zilberman, "A geolocation databases study," *Selected Areas in Communications, IEEE Journal on*, vol. 29, no. 10, pp. 2044–2056, 2011.

[11] B. Huffaker, M. Fomenkov, and K. Claffy, "Geocompare: a comparison of public and commercial geolocation databases," May 2011.

[12] B. Wong, A. Slivkins, and E. G. Sirer, "Meridian: a lightweight network location service without virtual coordinates," New York, NY, USA, pp. 85–96, 2005.