Effective Server Selection Strategy for Enhanced Network Performance in Smartphones

Lakshmi Tulasi Ambati

IT Consultant, Creative Business Tech, London, England.

Abstract:

The popular programmes' multimedia contents are often obtained via mirror sites. The information is kept in redundant servers to distribute the load. The process of choosing one mirror server over another has an impact on both user experience and performance. The current DNS lookup-based mirror server selection method often selects the server at the top of the lookup list, which is typically not the best option. In this research, we offer a different mirror server selection approach that limits the number of retransmissions, average throughput, Round Trip Time (RTT), and TCP connect time. The ideal server will be determined by selecting the one that has satisfiable values for every metric. Metrics are used based on personal choice. The best server is chosen from the group with the lowest RTT and connection setup time if many servers meet all the requirements.

In the event that no server meets every need, servers meeting a subset of the requirements will be picked, and one suboptimal server with the lowest RTT and connection setup time will be selected from that group. We have tested, implemented, and simplified our suggested solution for Android-based smartphones. We can see from the test measures that our suggested strategy should perform better in terms of lower RTT, retransmission counts, average throughput, and energy use.

I. Introduction

Multimedia files and other streaming items are typically kept on several mirror servers to evenly distribute the load across the servers and provide the best possible response time to the user. Nonetheless, a number of TCP parameters influence the servers' replies; the most often studied parameter in this area is Round Trip Time (RTT) [1]. Shorter links between clients and servers will have a lower RTT than longer connections in a scenario without congestion. In contrast, although if the equivalent distance is shortest, the RTT of the packets that go over the crowded lane will be significantly greater. Due to processing and queuing delays at the router, packet size and processing time will also have an impact on RTT.

The average throughput, number of retransmissions, and TCP connect time, in addition to RTT, all have a significant impact on network performance. Together, these indicators have a significant impact on how long the download takes and how long the radio stays on. RTT

affects each of these metrics (TCP connect time, retransmission count, and average throughput). They are not, however, connected one to the other. For instance, even if more retransmissions result in lower throughput and higher RTT, the congestion control mechanism of TCP makes their relationship less obvious. TCP automatically decreases the congestion window size by a percentage (based on a variation of the TCP congestion management algorithm) when the number of TCP packets transmitted without an ACK exceeds 3. As a result, the throughput decreases multiplicatively and the RTT rises in an intricate way. Thus, there is no clear relationship between these factors. While choosing the best server may seem to mostly depend on RTT, there are several contradictory situations that are discussed below that might result in subpar performance even with a lower RTT.

If the drop rate is more over 10%, TCP/IP does not function well. Even if the equivalent RTT is substantially lower, data transmission at a much slower effective rate (i.e., reduced throughput) occurs when the drop rate is more than 10% because of the requirement to wait for the retransmission of lost packets. Furthermore, even if the delay caused just by network transmission is significantly smaller (RTT), the initial time of connection (TCP connect time) and the server response time itself will be significantly larger with near capacity servers [2].

In addition to being ineffectual, the current DNS lookup-based mirror server selection method is also flawed. To spread the load across the content servers, CDN DNS servers give the client application numerous IP addresses as part of the current mirror server selection strategy.

Nonetheless, out of all the resource entries, the majority of internet client apps select the first DNS resource record. We ran a number of tests to examine content servers' RTT. Using a DNS lookup software on the Android smartphone, we were able to determine the content servers' IP addresses. Next, we determined each content server's RTT.

IP Address	RTT1	RTT2	RTT3
119.161.24.27	139	168	265
203.84.197.27	205	169	*
119.161.24.16	111	110	110
119.161.24.8	133	129	131
203.84.197.9	118	182	198
124.108.101.10	154	163	152
203.84.197.25	107	98	100
203.84.197.26	148	221	*

Figure 1: RTT-based DNS selection experimental results.

A scenario is shown in Fig. 1 where the best record has an RTT of 100 msec and the first record gives an RTT of 265 msec (in column RTT3 in Fig. 1). The figures displayed in Table may be obtained by assuming that we download a file that is 4 MB in size and that the maximum transmission unit (MTU) size for Ethernet is 1500 bytes. I. The difference in download time for a 4 MB file while selecting the server with a 100 msec RTT is around 740.94 279.6 = 265% when compared to the one with a 265 msec RTT. As the upload or download file size grows, this benefit will become much more noticeable. As a result, choosing a mirror server by default using DNS search or by using RTT alone is futile. Instead of focusing just on RTT optimisation, it is crucial to optimise all the metrics, including TCP connect time, throughput, number of retransmissions, and RTT, combined for global optimisation. We suggest choosing a mirror server policy by combining these metrics.

A. Inputs

In this study, we contribute the following:

• We show that the existing method of selecting the best mirror server from the DNS query reply list—based on DNS lookup—is inefficient.

• We suggest integrating Average Round Trip Time, Average Throughput, Average TCP Connect Time, and Number of Retransmissions into a single measure. We also provide a method to select the best available mirror server based on these criteria.

• We demonstrate notable performance gains by putting the suggested approach to the test using real-time measurements. Our suggested method may be quickly put into practice, improve throughput, save power, and provide a positive user experience for smartphones.

B. Use in Practice

• This proposal's results and implications improve end-to-end latency and significantly increase smart phone energy efficiency. For this reason, this idea makes a substantial contribution. This concept can lead to improved user experience by offering longer battery life, higher throughput, and lower latency for all apps.

We provide a client-only solution that doesn't require any server modifications. This approach is applicable to all Linux systems as well as Android. This approach may be applied widely not only to smartphones but also to other surfing devices like tablets and pads, as it chooses the best mirror server.

II. Related Work

Y. Yu et al. implement a DNS server selection strategy based on RTT in [1]. In order to comprehend the current DNS server selection techniques, this paper employs a trace-driven methodology. The use of auto-regression models to estimate server response times for the

DNS server selection process is examined by the authors in [3]. Authors in [4] measure the effect of decreasing DNS TTL (Time To Live) values on Web access latency and demonstrate that doing so can result in a two-fold increase in name resolution delay. A strategy for selecting DNS servers among resolved servers by evaluating last-mile network performance is suggested in [5]. outlines a request routing DNS server's design and first assessment in [6], which separates server selection from the rest of DNS. While the mirror server selection policy covered in this paper is different, all of the previously described work focuses solely on DNS server selection. However, the majority of these publications pick DNS servers using the crucial measure RTT.

A overview of several methods for choosing mirror servers to minimise latency may be found in [7]. employs dynamic network data, such as packet loss rates and RTT between video content servers and edge nodes of wireless networks (such as an RNC node in a 3G network and an Internet Service Provider (ISP) router in a Wi-Fi network), to determine which video content server is best when a video request is made. In [9], a joint server selection and routing method is suggested. A thorough analysis of YouTube's load balancing and server selection techniques can be found in [10]. [11] examines several server selection policy classes in the framework of a basic system model.

In contrast to existing approaches, our suggested server selection policy takes into account all relevant indicators at the same time and offers a technique for selecting both optimum and inferior options.

III. Proposed work

We take a look at the basic system model depicted in Figure 2. Internet-based material is accessed by smart phone users. You may access the Internet via cellular or Wi-Fi networks.

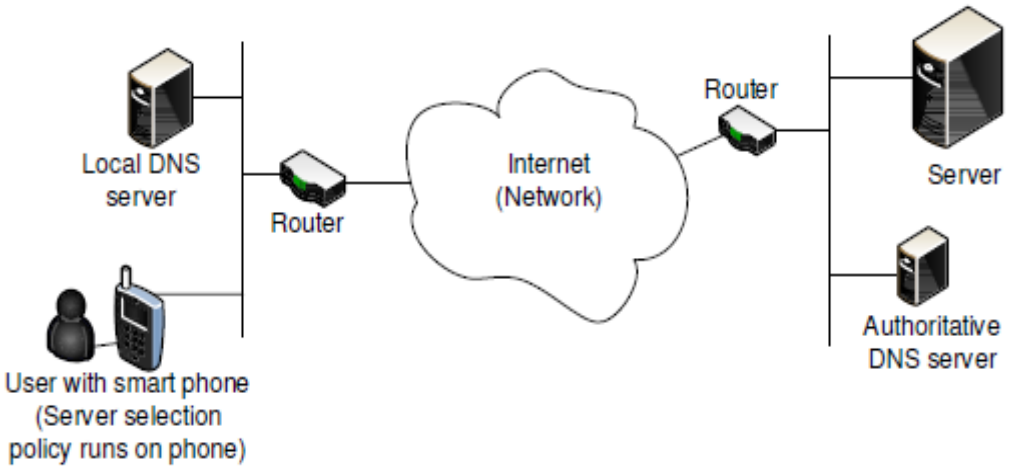
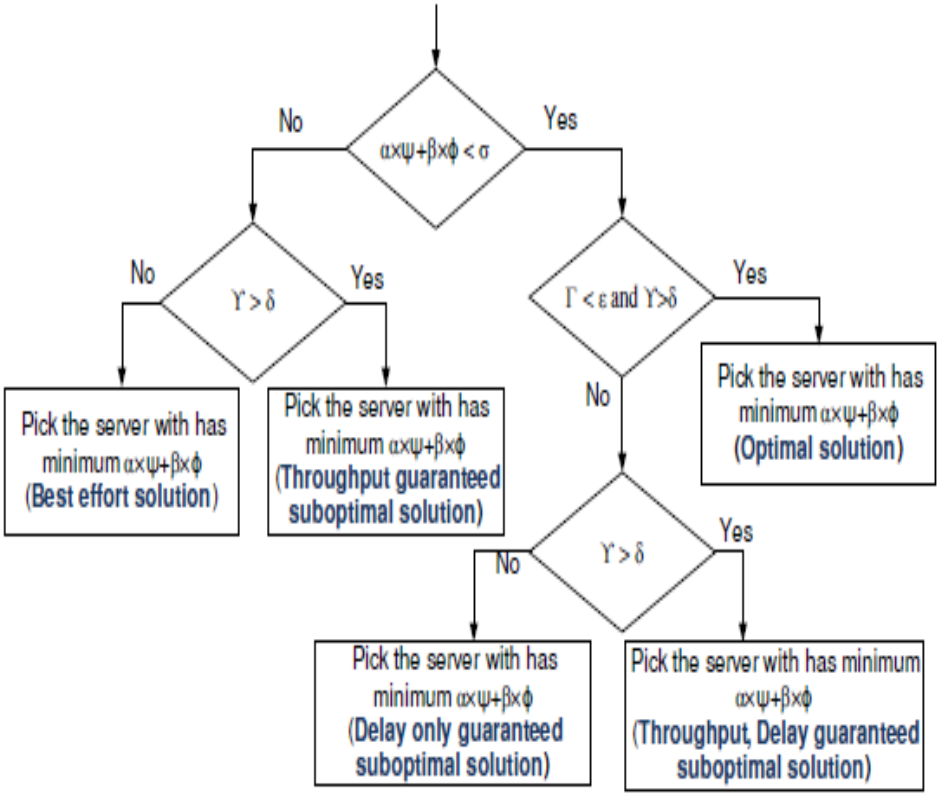


Fig 2: System Model

The smartphone initially talks to a local DNS server in order to resolve the domain name and retrieve the IP list of mirror servers. Using the acquired IP list, our suggested mirror server selection policy operates on the smartphone itself. The server selection strategy first establishes connections with each potential mirror server via the Gateway router in order to gather metrics during the training phase. The smartphone will save these measurements before applying the chosen policy. The policy engine on the smartphone will select the optimal server for all subsequent connections based on an algorithm that is applied to previously gathered measurements and is constantly changing over time. Following that, the client device will be linked to the mirror server that was chosen. This system architecture handles back haul network issues and does not presuppose any specific radio access technology (RAT) for wireless communication, such as 3G, 4G, or Wi-Fi (standards agnostic). Additionally, since mobility dynamics is one of the indicators we take into consideration, our suggested solution does not make any assumptions about the users' mobility. The smartphone is currently thought to run either the Tizen or Android operating systems. Nonetheless, any Linux platform can use the suggested method. Our solution's primary focus is on streaming apps, where users try to download large amounts of multimedia information from servers. The upload situation (uploading to cloud servers) may also be accommodated by this technique.



Algorithm to choose best server

IV. Conclusion

By constraining TCP connect time, average RTT, average throughput, and number of retransmissions, we have suggested a methodology for selecting mirror servers. We might get better performance with our suggested method in terms of average throughput, RTT, energy usage, and the quantity of retransmissions. In this paper, a static situation with a fixed network connection—either WiFi or 3G—was examined. We will focus on creating a policy to choose several mirror servers according to priority in the future. For instance, the user's desired download may contain content that is stored on many servers. We will create a policy in this case by assigning weights to every mirror server. The measures that we suggested in this study will be used to determine the weights. Based on the weights, different amounts of content can be downloaded from different servers, and an HTTP range request can be sent accordingly. As a result, content will be downloaded concurrently from several servers, and using an HTTP range request, it may be mixed on a smartphone. This can further reduce download times and improve user satisfaction.

REFERENCES

[1] Y. Yu, D. Wessels, M. Larson, and L. Zhang, “Authority server selection in dns caching resolvers,” SIGCOMM Comput. Commun. Rev., vol. 42, no. 2, pp. 80–86, Mar. 2012.

[2] Y-Q Zhang, A Kandel, T Y Lin, Y Y Yao, “Computational Web Intelligence Intelligent Technology for Web Applications,” Series in Machine Perception and Artificial Intelligence, vol. 58, 2004.

[3] S. Deb, A. Srinivasan, and S. Pavan, “An improved dns server selection algorithm for faster lookups,” in 3rd International Conference on Communication Systems Software and Middleware and Workshops, COMSWARE 2008, pp. 288–295.

[4] A. Shaikh, R. Tewari, and M. Agrawal, “On the effectiveness of dnsbased server selection,” in INFOCOM 2001, vol. 3, pp. 1801–1810.

[5] U. Goel, M. Wittie, and M. Steiner, “Faster web through client-assisted cdn server selection,” in 24th International Conference on Computer Communication and Networks (ICCCN), 2015, pp. 1–10.

[6] H. A. Alzoubi, M. Rabinovich, and O. Spatscheck, “Myxdns: A resquest routing dns server with decoupled server selection,” in Proceedings of the 16th International Conference on World Wide Web, ser. WWW ’07, 2007, pp. 351–360.

[7] B. Briscoe, A. Brunstrom, A. Petlund, D. Hayes, D. Ros, I.-J. Tsang, S. Gjessing, G. Fairhurst, C. Griwodz, and M. Welzl, “Reducing internet latency: A survey of techniques and their merits,” IEEE Commun. Surveys Tuts., no. 99, 2014.

[8] H. Nam, K.-H. Kim, D. Calin, and H. Schulzrinne, "Towards dynamic network condition-aware video server selection algorithms over wireless networks," in IEEE Symposium on Computers and Communication (ISCC), 2014, pp. 1–6.

[9] S. Narayana, W. Jiang, J. Rexford, and M. Chiang, "Joint server selection and routing for geo-replicated services," in Proceedings of the IEEE/ACM 6th International Conference on Utility and Cloud Computing, 2013, pp. 423–428.

[10] VK Adhikari, S Jain, ZL Zhang, "Where do You "Tube"? Uncovering Youtube Server Selection Strategy," in Proceedings of 20th International Conference on Computer Communications and Networks (ICCCN), 2011, pp. 1–6.

[11] N. Carlsson and D. L. Eager, "Server selection in large-scale videoon- demand systems," ACM Trans. Multimedia Comput. Commun. Appl., vol. 6, no. 1, pp. 1–26, 2010.

[12] Jacobson V, "Congestion avoidance and control," in Proceedings of the ACM SIGCOMM conference on applications, technologies, architectures,
and protocols for computer communication, Aug 1988, pp. 314–329.

[13] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, "The macroscopic behavior of the tcp congestion avoidance algorithm," SIGCOMM Comput. Commun. Rev., vol. 27, no. 3, pp. 67–82, 1997.