Is today's DNS the right solution for middleboxes selection?

A. Silvestro[†‡] R.Bifulco[†] F.Schneider[†] X. Fu[‡] J.Kangasharju[⊥]

NEC Laboratories Europe[†] University of Goettingen[‡] University of Helsinki[⊥]

Abstract

In-network services, often implemented using middleboxes, are key components of today's network applications (e.g., CDN, antiviruses, proxies, etc.). Current solutions for the selection of a serving middebox assume the presence of a single middlebox on the end-to-end network path, leading to not optimal solutions in terms of network performance when multiple middleboxes are inserted i.e., multiple parties are involved. This paper highlights the cost of a non-optimized middlebox selection strategy and suggests directions for further investigation.

1. Introduction

In-network services are important building blocks for today's network applications [1, 2]. Content Distribution Networks (CDNs), antiviruses, privacy protecting proxies, performance enhancers, are just a few examples of such services. However, the Internet's architecture and protocols ignore to a large extent the presence of in-network services, forcing network and service providers in implementing workarounds [3] that may negatively impact the performance delivered to final users. The issue is particularly relevant when several in-network services, provided by third parties, are introduced on the end-to-end path.

Previous work [2] focused on guaranteeing that an in-network service is not hidden to any of the communication's end-points, or on enabling an end-point in deciding whether to use an in-network service or not, when connecting to another end-point [4]. Still, the mentioned works assume that the node provisioning a given service, i.e., the middlebox, is known in advance. However, an in-network service is usually implemented using several middleboxes, typically deployed at different locations [5]. The selection of a specific middlebox is performed by the in-network service provider, which takes into account a number of variables, including system and network loads, end-point locations, local regulation constraints, etc. [6] For instance, IP anycast [5] and DNS redirection [7] are state of the art techniques adopted by service providers.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

CrossCloud'17, April 23, 2017, Belgrade, Serbia.

Copyright © 2017 ACM 978-1-4503-4934-5/17/04…$15.00.

http://dx.doi.org/10.1145/3069383.3069389

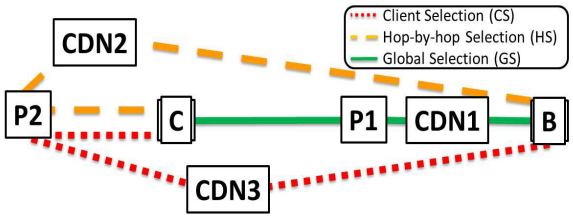

Figure 1: Example scenario.

In this paper, we consider the case of multiple in-network services explicitly inserted within the end-to-end path (Fig. 1), by either the client or the server. Using the current DNS architecture, we show that today's possible middlebox selection approaches may introduce a significant penalty in the transfer time of TCP flows. Then, we highlight the space for possible improvements and future research directions.

2. A Strawman Solution

Consider the example scenario of Fig. 1. The client (C) connects to her bank's web site (B). The client subscribed to a parental control in-network service (P), while the content server subscribed to a CDN service (CDN). Each in-network service is specified by a single domain name (e.g., *parental-ctr.com*, *cdn.com*) while it is deployed at multiple locations in the network, thus specified by multiple IP addresses. The authoritative DNS of each in-network service provider, for each request, selects the middlebox to use, mapping it to the related IP address. The serving middlebox is selected dynamically – among other parameters – to be the closest (in terms of delay) to the requesting client. In the presented scenario, the parental control service uses two middleboxes deployed at different locations: P_2 is the closest to the user, while P_1 is the closest to the bank. The CDN has deployed three middleboxes in the network: CDN_3 is the closest to the user, while CDN_1 and CDN_2 are the closest to P_1 and P_2, respectively. The selection of the serving middleboxes can happen in one of the following ways:

1) ***Client Selection (CS):*** the client sends 3 DNS queries i.e., a first query to resolve *parental-ctr.com*, a second to resolve *cdn.com* and a third to resolve *bank.com*. According to Fig. 1 and assuming each authoritative DNS selects the serving instance based on the client's network location, P_2 and CDN_3 are selected to be the closest to the client.

2) ***Hop-by-hop Selection (HS):*** the client uses the *EDNS0* client-subnet extension of the DNS protocol [8] in order to enhance the selection process for the service chain. Such extension allows a DNS client to specify an IP prefix in a query, in order to provide the DNS server with a hint about the client's location. Therefore, the client's subnet is defined

as source for the first query, the parental control and the CDN for the second and the third query, respectively. In this case, P_2 is selected to be the closest to the user while CDN_2 is selected to be the closest to P_2.

3) ***Global Selection (GS):*** in this case, we assume to have a full visibility of all the middlebox instances and their locations. Therefore, we can solve the *GS* problem modeling it as a shortest path selection problem, providing the best among the possible solutions. This would lead to the selection of P_1 and CDN_1 in the example of Fig. 1.

3. Evaluation

Experiment. In our experiment, we consider service chains composed of 5 in-network services. In order to better understand the implication of the different selection strategies, we assume that TCP is used to establish the connections throughout the chain and the *Time To First Byte (TTFB)* has been selected as a comparison metric. It represents the time required for a client to receive the first byte of a content server's response to a client issued request. Assuming the end-to-end delay of D, TTFB is proportional to it by a factor four ($TTFB = 4D$) [9]. We implemented a simulator in Python that creates weighted network graphs composed of one client, several in-network services' instances and one content server (which we assume to be deployed in a single network location). Each node is placed randomly on a $100x100$ coordinate grid, and the *Euclidean* distance between any two nodes is considered as this edge's weight (i.e., network delay). We implemented *CS* and *HS* while we use the *NetworkX Dijkstra* implementation for *GS*.

We generated network graphs with a number of nodes varying from 27 (i.e., 5 middleboxes per service) to 452 nodes (i.e., 90 middleboxes per service) which represents a wider distribution of the in-network services in the network. For each experiment, we generated 100 different graphs - 1000 graphs in total throughout the whole experiment – and *CS, HS and GS* are executed on the same graph.

Result. Fig. 2 shows a box plots of the results. The x-axis represents the number of middleboxes deployed by each in-network service. The y-axis represents the percentage of TTFB required by *CS* and *HS* compared to *GS*. As we expected, the results show that for all the cases and considering all quantiles GS provides the best solution. In fact, all the medians are greater than 0, which means that *CS* and *HS* always show an increase on the TTFB compared to GS. Observing the graph, we can make two further observations. The first observation is that HS produces better results than CS, both in terms of lower percentages and lower spread (smaller boxes, shorter whiskers). Particularly, considering the medians, for each experiment *CS* selects network paths which show a delay (or TTFB), on average, 49% higher compared to *GS*. *HS*, instead, shows 28% higher costs. The second observation is that even considering the HS strategy, there is space for optimization, as the first quartile is in most cases around and above 20%. This motivates further investigation in this direction, considering that for the considered metric, small increase in the delay may have an important impact. For instance, Amazon estimates that an increase of delay as little as 100ms cuts its revenue by 1% [10].

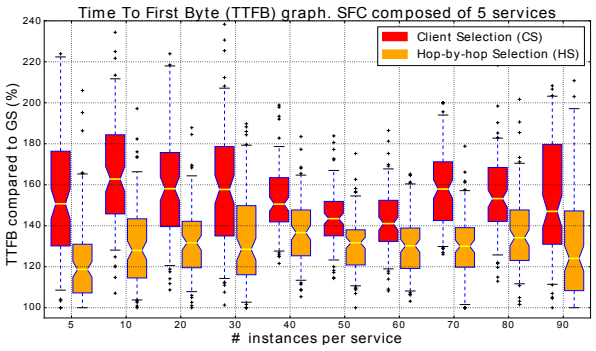

Figure 2: Time To First Byte (TTFB) Graph

4. Future Work

The overheads highlighted by our simulations are rooted in the uncoordinated decisions performed by multiple and independent parties involved in the end-to-end service delivery. However, *GS* is difficult to be applied in practice, since it requires the full knowledge of all the in-network services' instances. In fact, they might not be aware of each other or they might not be willing to exchange information. Even assuming the parties are aware and agree on collaborating with each other, it is still unclear which is the minimal amount of information they would need to share in order significantly improve on the selection process. Answering this question and providing a system that efficiently solves these cases is the focus of our future work.

Acknowledgment

This research work has been partly funded by the joint EU FP7 Marie Curie Actions *CleanSky* Project (G.A.: 607584).

References

[1] Sherry J. et al. Making middleboxes someone else's problem: Network processing as a cloud service. *SIGCOMM CCR*.

[2] M. Walfish et al. Middleboxes No Longer Considered Harmful. OSDI'04.

[3] S. Loreto et al. Explicit trusted proxy in http/2.0, 2014.

[4] D. Naylor et al. Multi-Context TLS (mcTLS): Enabling Secure In-Network Functionality in TLS. ACM SIGCOMM'15.

[5] M. Calder et al. Analyzing the Performance of an Anycast CDN. In *Proceedings of ACM IMC '15*.

[6] H. Liu et al. Efficiently Delivering Online Services over Integrated Infrastructure. In NSDI'16.

[7] B. Maggs et al. Alg. nuggets in content delivery. *CCR*.

[8] C. Contavalli et al. Client Subnet in DNS Queries. RFC 7871.

[9] Siracusano G. et al. On the Fly TCP Acceleration with Miniproxy. In ACM SIGCOMM HotMIddlebox '16.

[10] T. Flachet al. Reducing web latency: the virtue of gentle aggression. ACM SIGCOMM CCR.