A Study of the Impact of DNS Resolvers on Performance Using a Causal Approach

Hadrien Hours*, Ernst Biersack*, Patrick Loiseau*, Alessandro Finamore†, Marco Mellia†
*EURECOM, email: firstname.lastname@eurecom.fr
†Politecnico di Torino, email: firstname.lastname@polito.it

***Abstract*—For a user to access any resource on the Internet, it is necessary to first locate a server hosting the requested resource. The Domain Name System service (DNS) represents the first step in this process, translating a human readable name, the resource host name, into an IP address. With the expansion of Content Distribution Networks (CDNs), the DNS service has seen its importance increase. In a CDN, objects are replicated on different servers to decrease the distance from the client to a server hosting the object that needs to be accessed. The DNS service should improve user experience by directing its demand to the optimal CDN server. While most of the Internet Service Providers (ISPs) offer a DNS service to their customers, it is now common to see clients using a public DNS service instead. This choice may have an impact on Web browsing performance. In this paper we study the impact of choosing one DNS service instead of another and we compare the performance of a large European ISP DNS service with the one of a public DNS service, Google DNS. We propose a causal approach to expose the structural dependencies of the different parameters impacted by the DNS service used and we show how to model these dependencies with a Bayesian network. This model allows us to explain and quantify the benefits obtained by clients using their ISP DNS service and to propose a solution to further improve their performance.**

I. Introduction

Each time an Internet user wants to access a resource, he uses an human readable name called Uniform Resource Locator (URL), containing the domain name of the administrative entity hosting this resource. However, a domain name is not routable and needs to be translated into the (IP) address of a server hosting the resource the client wants to access. It is the role of the DNS service to translate the domain name to the corresponding IP address. Many popular services such as Youtube, Itunes, Facebook, Twitter, rely on CDNs where objects are replicated on different servers and in different geographical locations to optimize the performance for their users. When a client wants to access one of these services, its default DNS server contacts the authoritative DNS server for the domain hosting the resource the client is requesting. Based on the location the request is coming from, the authoritative DNS redirects the client to the optimal server. Most of the ISPs provide a DNS service, but it is now common to see customers using a public DNS service instead [1]. Clients using the DNS service of their ISP are served by a local DNS server. Local DNS servers are closer to clients, their usage provides a more accurate location information to the CDN compared to the locations represented by the few DNS servers offered by a public DNS service such as the Google DNS service. There have been several studies suggesting that public DNS services do not perform as well as local DNS services provided by ISPs, mainly because of the impossibility of public DNS to correctly communicate the location of the clients originating the request [2], [3].

However, studying the performance of the users accessing Internet resources is a complex task. Many parameters influence the end user experience and the relationships between these parameters is not always observable or intuitive. It is therefore necessary to adopt a simple, yet formal, model that allows us to understand the role of a given parameter and its dependencies with other parameters. Bayesian networks offer a simple and concise way to represent complex systems [4]. In this paper, we use a Bayesian network to represent the causal model of the impact of the DNS service on the throughput performance experienced by clients accessing resources via the Akamai CDN and to capture the dependencies between the different parameters impacting the throughput of the clients. We further use this model to predict the effect on throughput performance had the client used another DNS service. This prediction allows us to understand the impact of choosing one DNS service instead of another. From the same model, we are also able to indicate how to improve the performance of the users of the local DNS.

This work differs from previous studies of DNS services in several points:

- We use a causal approach that formally models the structural dependencies of the different parameters influencing the user experience (throughput).
- After observing that the users of the local DNS experience better performance than the Google DNS users, we show that such improvement is made through a redirection of the local DNS users to closer servers and we are able to quantify such performance improvement.
- The causal model of our system shows that the TCP parameterization of the servers accessed by the users of the Google DNS plays a key role in improving their performance. Besides fully explaining the observed performance, this result also indicates a solution to further improve the performance of the users of the local DNS.

Overall, the main contribution of this work resides in the methodology that is presented, and in its use of counterfactuals to understand the causal dependencies of complex systems.

In Section II, we present causal models and their use to predict interventions, summarizing some of the main concepts from [5], [6]. Section III presents the environment of our study and the description of the parameters constituting our

system. Section IV presents our study of the DNS impact on the throughput. In particular we present the causal model of our system where we can observe the impact of the DNS choice on the throughput. Our approach also allows us to predict the improvement that could be achieved by modifying the parameterization of the servers accessed by the users of the local DNS service. Section V compares our approach to other works and Section VI summarizes the method presented and proposes some directions for exploiting further this method.

II. Causal model definitions and usage

It is very challenging to model complex systems and to structure knowledge obtained from its passive observations. Many of the existing works rely on the presence of correlation between different events observed simultaneously (see [7] and references therein). However, correlation is not causation and the detection of correlation between two parameters does not inform us on how they are related. One can impact the other, or the opposite, or an unobserved parameter can impact these two parameters simultaneously. The difference between correlation and causation plays an important role if we want to use this knowledge to decide on how to improve our system by partly modifying its behavior. A causal approach tries to solve this issue by uncovering the structural dependencies between the parameters of the system one wants to understand and to manipulate. The capacity to predict the effects of a manipulation on the system parameters is an important strength of causal models as they are *stable under intervention*. The stability under intervention means that a causal model, inferred from the observations of a system in a given situation, is still valid if we manually change the system mechanisms, redefining the systems laws. The manual modification of the system parameters is called an *intervention*. Interventions consist in modifying the behavior of a component of the system, removing the influence from its direct and remote causes and manually fixing its variations. Causal models, and the causal theory [5], [6], allow us to predict the behavior of the different parameters of the inferred model after an intervention where we modify the system, using passive observations made prior to this intervention.

In this section we present the PC algorithm [8] that is used to infer the causal model of our system. We also describe the different properties of a causal model as described in [5], [6].

A. Causal model inference

For our work, we use the PC algorithm [8] to build the Bayesian graph representing the causal model of our system. This algorithm takes as input the observations of the different parameters that form our system and infers the corresponding causal model. In our representation of a causal model as a Bayesian network, each vertex represents one parameter of our system and the presence of an edge from a vertex to another vertex ($X \rightarrow Y$) represents the existence of a causal dependence of a parameter (corresponding to vertex Y) on another parameter (corresponding to vertex X).

The first step of the PC algorithm consists in building a fully connected and unoriented graph, called *skeleton*, where each parameter is represented by a vertex and connected to any other parameter. In a second step, all the unconditional independences ($X \perp\!\!\!\perp Y$) are tested for all couples of parameters. The edges between two nodes whose corresponding parameters are found independent are then removed. For the parameters whose nodes are still adjacent, we test if there exists a conditioning set of size 1 that makes two adjacent nodes independent. If such set exists we remove the edge connecting the corresponding two nodes, otherwise the edge is not removed. This step is repeated, increasing the conditioning set size by one at each step, until the size of the conditioning set reaches the maximum degree of the current skeleton (the maximum number of adjacent vertices for any vertex in the current graph), which means that no more independence can be found. The final step consists in the orientation of the edges. First, we orient all the V-structures (subgraphs $X-Z-Y$ where X and Y are not adjacent) and then orient as many edges as possible without creating new colliders[1] or cycles [5].

It is worth noting that the PC algorithm is able to infer a causal model up to its independence equivalence class (called the Markov Equivalence Class and represented by a partially oriented graph). It often happens that several graphs exist that imply the detected independences. See Section II-B for the equivalence between graphical connectivity and dependence. As a consequence, the output of the PC algorithm is a graph where only the independences leading to a unique orientation of the edges are represented. The other edges are left unoriented. In our work we use domain knowledge to select the member of the equivalence class inferred by the PC algorithm closest to our understanding of the system under study. Note that the Tetrad software [6] offers the possibility to represent all the members of a given Markov Equivalence Class.

B. D-separation

The d-separation criterion is a graphical criterion to judge, from a graph, the independence between two parameters, represented by their corresponding nodes. D-separation associates the notion of connectedness with dependence. Therefore, if there exists a directed path between two nodes, they are said to be d-connected and their corresponding nodes are dependent. On the other hand, if we condition on one of the nodes of this path then this node is said to block the path and the parameters are conditionally independent. When studying d-separation, an important notion is the one of **collider** (see footnote 1). The presence of a collider on a (undirected) path blocks this path and conditioning on a collider unblocks the path. Intuitively, two independent causes become dependent if one conditions on their common consequence.

C. Causal model properties and theorems

In this section we suppose that we have already a causal model of our system represented by a Bayesian network and we focus on two parameters X and Y, where Y represents the performance of our system. We are interested in the global effect on Y when intervening on X, including the effects mediated by external parameters also impacted by this intervention. We call this causal effect the *total causal effect*.

[1] a collider, Z, is a vertex part of an oriented subgraph $X \rightarrow Z \leftarrow Y$ where X and Y are not adjacent.

We denote by $do(X = x)$ (or $do(x)$) the intervention that consists in intervening on the the parameter X by fixing its value to be x.

If we use G to denote the Bayesian graph that represents the causal relationships between the parameters of our system, we use $G_{\overline{X}}$ to denote the sub-graph of G where all the edges entering X are removed and $G_{\underline{X}}$ the sub-graph of G where all the edges exiting X are removed. We can use the rules of *do-calculus* from [5] to estimate the distributions of the parameters of our system after an intervention based on their distributions prior to this intervention. Note that these rules do not rely on any assumption regarding the distributions or functional dependencies of the parameters. In particular, P represents the (possibly multinomial) probability distribution specified by the probability mass function or probability density function depending on the nature of the parameters.

Theorem 1 (3.4.1 from [5]): (Rules of do calculus) Let G be the directed acyclic graph associated with a causal model [...] and let $P(\cdot)$ stand for the probability distribution induced by that model. For any disjoint subsets of variables X, Y and Z we have the following rules.

Rule 1(Insertion/deletion of observation):

$$P(y|do(x), z, w) = P(y|do(x), w) \text{ if } (Y \perp\!\!\!\perp Z \mid X, W)_{G_{\overline{X}}} \quad (1)$$

Rule 2(Action/observation exchange):

$$P(y|do(x), do(z), w) = P(y|do(x), z, w) \text{ if } (Y \perp\!\!\!\perp Z \mid X, W)_{G_{\overline{X}\underline{Z}}} \quad (2)$$

Rule 3(Insertion/deletion of intervention):

$$P(y|do(x), do(z), w) = P(y|do(x), w) \text{ if } (Y \perp\!\!\!\perp Z \mid X, W)_{G_{\overline{XZ(W)}}}, \quad (3)$$

where Z(W) is the set of Z-nodes that are not ancestor of any W-nodes in $G_{\overline{X}}$.

D. Density estimation

As no assumption can be made on the distribution of the parameters, we estimate the multidimensional probability density functions via Copulae [9], using the Sklar theorem.

The Sklar theorem stipulates that, if F a is multivariate cumulative distribution function with marginals $(F_1, \ldots, F_i, \ldots, F_n)$, there exists a copula C such that

$$F(x_1, \ldots, x_i, \ldots, x_n) = C(F_1(x_1), \ldots, F_i(x_i), \ldots, F_n(x_n)). \quad (4)$$

In our work, we chose Gaussian copulae as they are smoother and cope better with the constraint of limited data. In the bivariate case, the Gaussian copula is defined as:

$$C_\rho(u, v) = \Phi_\rho(\Phi^{-1}(u), \Phi^{-1}(v)), \quad (5)$$

where ρ represents the correlation matrix and Φ the CDF of the standard normal distribution. The marginals, $F_i(x_i)$, are estimated using normal kernels.

III. Experimental set up

A. Experiment design

To observe the local DNS queries and answers for the clients using their ISP DNS service, we place ourselves at a Point of Presence (PoP) of a large European ISP and observe the traffic directed to or coming from the Akamai CDN. To model the impact of the DNS choice on the ISP client throughput, we make three choices: i) We only focus on the traffic carried by the TCP protocol. ii) To better capture the DNS and network impact on performance, we restrict ourselves to voluminous connections that go beyond the TCP slow start phase and carry at least 2MB. iii) As more than 90% of the observed connections use either Google DNS (*GDNS*) or the DNS of the local ISP(*LDNS*) we consider only these two DNS services.

The probe is placed between the client and the server. We call internal network, denoted as *isp network*, the part of the network between the client and the probe. On the opposite, we call external network the part of the network between the probe and the server, assimilated to the Internet network and denoted *inet network*. The traffic is observed on two different days, a Thursday and a Sunday, from 5.30 pm to 9.30 pm (peak time).

B. Model parameters

We use the Tstat software [10] to passively measure per-flow statistics. We select a subset of statistics representing the parameters of our system that are known, from domain knowledge, to impact the throughput. We obtain a dataset where each sample represents a single connection and for each connection several parameters are observed. We have about 7000 connections, which represent our sample size.

We measure several parameters in addition to the Tstat statistics [10]: the DNS service, the number of hops between the client and the server and the server IP address, represented by its corresponding Autonomous System (AS) number.

C. Observation summary

For each connection, we record 19 parameters. In Table I, we present the average (μ), minimum (*min*), maximum (*max*), standard deviation (σ) and coefficient of variation ($CoV = \frac{\sigma}{\mu}$) of each of the 19 parameters over the 7000 connections. As this work focuses on the comparison between the performance of LDNS users and GDNS users, Table II presents the statistics for the connections where the LDNS is being used and for the connections where the GDNS is used separately.

We use the following notations:

- Parameters with the prefix *isp* represent the *isp network* statistics while the ones with the prefix *inet* represent the *inet network* statistics.
- The suffix *avg* represents the average value of a given parameter over a connection
- The suffix *std* represents the standard deviation of a given parameter over a connection

TABLE I. SUMMARY OF THE DIFFERENT PARAMETERS

Parameter	μ	*min*	*max*	σ	*CoV*
dstip	N.A.	1300	34000	N.A.	N.A
dns	N.A	1	3	N.A.	N.A.
dow	N.A.	4	7	N.A.	N.A.
tod (s)	7100	52000	78000	4400	0.1
isprttavg (ms)	76	0	19000	460	6.1
isprttstd (ms)	100	0	37000	960	9.2
ispnbhops	1.8	1	3	0.51	0.3
inetrttavg(ms)	26	0.48	660	27	1.0
inetrttstd (ms)	8.2	0	4700	61	7.5
inetnbhops	9.4	2	21	2.8	0.3
rwin0	0.83	0	360	11	13
rwinmin (kB)	31.3	0.004	65	22.5	0.9
rwinmax (kB)	213	17.5	2625	150	0.7
cwinmax (kB)	150	7.3	1625	103	0.7
cwinmin (kB)	0.9	0.001	1.5	0.6	0.7
retrscore	0.005	0	0.19	0.009	1.9
rto (bool)	0.11	0	1	0.32	2.8
nbbytes (MB)	23.8	2.1	3875	138	5.7
tput (Mbps)	3.2	0.006	35	2.6	0.8

- The *rto* parameter represents the presence of at least one packet retransmission due to a time out and *retrscore* the fraction of retransmitted packets.
- The parameters *rwin** and *cwin** represent receiver window and congestion window metrics respectively.
- The day of the week and time of the day are captured by the variables *dow* and *tod* respectively.

It should be noticed that the congestion window is a sender parameter. Tstat estimates the congestion window by looking at the amount of data that has not been acknowledged yet, namely the in-flight size. As the sender-to-the-probe path is typically very high speed (~10Gbps), and the bottleneck is after the probe (e.g., the ADSL link), the in-flight-size is a very good approximation of the sender congestion window.

Destination IP (*dstip*), DNS (*dns*) and days (*dow*) are categorical data for which the average value, standard deviation or coefficient of variation do not exist.

Two observations can be made when looking at the variation of the different parameters in Table I and will be referred to in the discussion of the causal model of our system.

First, we can observe that the value of the average RTT inside the ISP network (*isprttavg*) is almost three times as high as the average RTT in the "Internet" network (*inetrttavg*). This difference is due to the use of an ADSL link as access link and the latency experienced by the packets entering or exiting the ISP network. When many packets are being sent at the same time they can create some congestion that implies high values in the RTT. Such changes and variations can also be observed in the standard deviation of the RTT that is more than ten times bigger in the internal network than in the external network.

Second, another information that does not appear in Table I, 64% of the connections experience at least one loss but only 27% of them show a retransmission score bigger than 0.5%. This value suggests that, in cases of congestion, packets queue in buffers but are not necessarily dropped.

IV. CAUSAL STUDY OF DNS IMPACT ON THE THROUGHPUT

A. Modeling causal relationships

Using the PC algorithm [8] and the kernel based independence test from [11] we obtain the Bayesian network representing the causal model of our system, presented in Figure 1. In this section we briefly discuss some of the dependencies exhibited by this model.

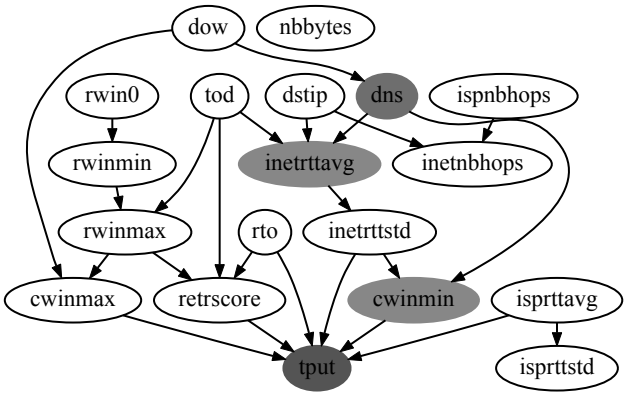

Fig. 1. Bayesian network representing the causal model of Web performance using two different DNS: the public Google DNS and the DNS of the local ISP

We can observe that the day of the week (*dow*) and the time of the day (*tod*) are two parentless nodes, which is not surprising, and that the time of the day (*tod*) influences the RTT between the probe and the server (*inetrttavg*), which captures the *peak hour effect*.

The observation concerning the two rtt standard deviations in Section III-C suggested that the internal rtt would have the stronger impact on the throughput. However, we can observe a direct dependence between the standard deviation of the external RTT (*inetrttstd*) and the throughput (*tput*) but not between the standard deviation of the internal RTT (*isprttstd*) and the throughput. This observation illustrates the ability of causal model to exhibit non intuitive dependencies.

On the other hand, we observe that the day of the week (*dow*) influences the DNS service used by the clients (*dns*). As our observations are made on two days only (a Thursday and a Sunday), our conclusions are limited. However, looking at the data, it appears that on Thursday 72% of the connections use the LDNS service against 28% using the GDNS service, while on Sunday 93% of the connections use the LDNS service against 7% using the GDNS service. It would be interesting to identify the clients using one DNS service and compare their locations with the ones of the clients using the the other DNS service to better understand this dependence. The day of the week may capture the difference in the Internet usage and the devices used at home and at work. However, for privacy reasons, the IP addresses of the clients are obfuscated, which prevents us from investigating this hypothesis.

One of the most interesting dependencies, which motivated this work, is the one present between the DNS (*dns*) and the external RTT (*inetrttavg*). We could observe in Tstat logs that the servers accessed by the clients using the LDNS service are often located inside the autonomous system of the ISP. This is not the case for the clients using the GDNS service. As mentioned in the introduction and in [2], clients using the local DNS service benefit from a redirection to servers closer than the ones of the clients using a public DNS service. We observe an average external RTT of 20 ms for the LDNS service users,

TABLE II. SUMMARY OF THE DIFFERENT METRICS FOR THE TWO DNS: LOCAL DNS (LD) AND GOOGLE DNS (GD)

Par	μ		*min*		*max*		σ		*CoV*	
	LD	GD	LD	GD	LD	GD	LD	GD	LD	GD
isprttavg (ms)	80	61	0	0	19000	15000	470	440	5.9	7.2
isprttstd (ms)	1100	76	0	0	32000	37000	920	1100	8.3	14.0
ispnbhops	1.8	1.9	1	1	3	3	0.53	0.4	0.3	0.2
inetrttavg (ms)	20	48	0.48	11	510	660	20	38	1.0	0.8
inetrttstd (ms)	8.6	6.5	0	0	4700	1400	65	44	7.6	6.8
inetnbhops	8.7	12	2	5	17	21	2.4	2.7	0.3	0.22
rwin0	0.97	0.29	0	0	330	360	12	9.2	12.0	32.0
rwinmin (kB)	35	12	0.004	0.03	65	65	28	14	0.8	1.1
rwinmax (kB)	213	213	18	18	2625	2000	150	138	0.3	0.7
cwinmax (kB)	163	118	7.3	7.8	1625	738	108	72	0.7	0.6
cwinmin (kB)	0.9	1.2	0.001	0.001	1.5	1.5	0.6	0.5	0.7	0.4
retrscore	0.005	0.004	0	0	0.19	0.06	0.01	0.01	1.9	1.8
rto (bool)	0.11	0.11	0	0	1	1	0.32	0.31	2.8	2.9
nbbytes (MB)	29	7	2.1	2.1	3875	1375	150	44	5.3	6.5
tput (Mbps)	3.2	3	0.006	0.007	35	29	2.7	2	0.9	0.7

while the users of the GDNS service experience an average external RTT of 48 ms (see Table II).

We can also see that congestion window metrics (*cwinmin*, *cwinmax*) have a direct impact on the throughput (*tput*). Additionally, the minimum congestion window (*cwinmin*) has the DNS (*dns*) as direct parent. Its average value for clients using Google DNS is 1.2kB against 0.9kB for users served by the local DNS, see Table II.

As explained in Section IV-B2, a parameter present in a causal model represents also the mechanisms captured by such parameter. This is the case of the *cwinmin* that also captures the tuning of the server TCP parameters, such as the initial congestion window.

Clients using the local DNS often access their objects from servers that are located *inside* the ISP network. These servers could have a configuration different from the servers accessed by the users of the GDNS. This hypothesis could also explain the fact that both DNS services result in a similar throughput despite a different RTT. Two other reasons could be the impact of losses on the congestion window or the load of the servers being accessed by the clients.

To capture the server load, we estimate the server processing time defined from the time at which a server sends the acknowledgment of the client HTTP/GET message and the time at which it sends the first data packet. However, the server processing time shows an expected value of 43 ms for the LDNS users against 64 ms for the GDNS users. A higher processing time for the servers accessed by the GDNS users suggests that they are more loaded. On the other hand, the congestion window is impacted by the loss but, as mentioned in Section III-C, few losses actually happen during the period of observation of the system and no significant dependence is found between the loss (*retrscore*) and the DNS service (*dns*).

It comes with no surprise that the internal RTT (*isprttavg*) is a parent of the throughput. The absence of a dependence between the time of the day (*tod*) and the internal RTT can be explained by the fact that all the observed users are using the same "internal" path (the path from the users to the probe).

We see that the maximum receiver window advertised by the client (*rwinmax*) has the time of the day as one of its parent (*tod*). Such fact may be due to the TCP buffer auto tuning mechanism [12] that adjusts the receiver window according to the quantity and frequency of data received by the client, which is influenced by the time of the day.

We can notice the absence of an edge between the DNS (*dns*) and the destination IP address (*dstip*) and the absence of any adjacency with the object size (*nbbbytes*). An explanation of the absence of these edges could be the unequal distribution of the servers accessed by the users of the LDNS service when compared to the distribution of the servers accessed by the users of the GDNS service. A solution to detect weaker dependences is to increase the acceptance rate in the independence tests. Increasing the acceptance rate implies a higher risk in failing to reject weak independences and should be used with caution. The independence of the object size from other parameters influencing the throughput is not necessarily surprising as we consider long connections. The absence of a dependence between the DNS (*dns*) and the destination IP (*dstip*) of the servers could also be due to a lack of granularity in the representation of IP address by its AS number. This last point would not have an important impact on the results presented in this paper.

The loss parameters (*retrscore* and *rto*) and RTT parameters (*inetrttstd* and *isprttavg*) are four of the six direct parents of the throughput, which is in line with our domain knowledge of TCP. The additional parents are the congestion window metrics of the server (*cwinmin* and *cwinmax*).

The fact that none of the receiver window metrics (*rwin**) is a direct parents of the throughput (*tput*) is not surprising. The study of the client receiver window limitation suggests that the clients are never limiting the throughput (not showed here for space reasons).

B. Estimation of the impact of a given parameter in the difference of performance experienced by two different users

We have seen that the Bayesian network derived in the previous section reveals a rich set of causal relationships that indicate how the different parameters impact the throughput. We will now use this model to answer *what-if questions* from the existing data without the need to collect more data or perform additional experiments.

In this section, as dealing with probabilities, we compare the expected values of the throughput ($\mathbb{E}[TPUT] = \int f_{TPUT}(tput)tput\mathrm{d}tput$) instead of its average values (μ_{TPUT}).

1) Distance and delay: In this section we use our causal model to answer the question: "*What is the gain, in terms of performance, induced by choosing the local DNS service and being redirected to servers closer to the client?*".

This question can be reformulated as: "*What would have been the performance of a user served by the local DNS if it would have been redirected to a server whose inetrtt corresponds to the one the Google DNS service would have redirected him to ?*".

In our case, this question is equivalent to predict the effect of an intervention where we modify the external delay (RTT) experienced by clients served by the LDNS and give this delay the distribution of the delay experienced by clients served by the GDNS; the distribution of the rest of the parameters being kept identical for the LDNS service users.

More formally, if we denote by RTT the *inetrttavg* parameter, by *LD* the local DNS and by *GD* the Google DNS, we want to estimate the following distribution:

$$f\left(TPUT = tput|DNS = LD, do(RTT \sim f_{RTT|do(DNS)}(\cdot, GD))\right). \tag{6}$$

We can observe from the causal graph Figure 1 (cf the explanation of d-separation in Section II-B) that $(RTT \perp\!\!\!\perp DNS)_{G_{\underline{DNS}}}$ which implies (Rule 2 from Theorem 1):

$$f_{RTT|do(DNS)}(rtt, GD) = f_{RTT|DNS}(rtt, GD). \tag{7}$$

From [5, Section 4.2], if we want to predict how an intervention on X affects Y, where the intervention on X is enforced with the probability $f^*(X|Z)$, we obtain:

$$f(y)_{|f^*(x|z)} = \int_{D_X}\int_{D_Z} f_{Y|do(X),Z}(y, x, z)f^*(x|z)f(z)\mathrm{d}x\mathrm{d}z. \tag{8}$$

On the other hand, we can read from the causal graph in Figure 1 (cf the explanation of d-separation in Section II-B) that $(RTT \perp\!\!\!\perp TPUT|DNS, TOD)_{G_{\underline{RTT}}}$. It follows, from Rule 2 of Theorem 1 that

$$\begin{aligned} &f_{TPUT|do(RTT),TOD,DNS}(tput, rtt, tod, dns) = \\ &\quad f_{TPUT|RTT,TOD,DNS}(tput, rtt, tod, dns). \end{aligned} \tag{9}$$

As a consequence we can rewrite Equation (6) as:

$$\begin{aligned} &f(tput|LD)_{f(rtt|do(GD))} = \\ &\int_{D_{RTT}}\int_{D_{TOD}} f(tput|do(rtt), LD, tod)f(tod)f(rtt|do(GD))P(GD) = \\ &\int_{D_{RTT}}\int_{D_{TOD}} f(tput|rtt, LD, tod)f(tod)f(rtt|GD)P(GD) \end{aligned} \tag{10}$$

using Equation (7) and Equation (9).

Limitations: Figure 2 represents the distribution of the external RTT for GDNS users and LDNS users. The prediction represented by the Equation (6) is possible as, roughly, the range of the external RTT values observed for GDNS users represents a subset of the values observed for the LDNS users. For the opposite intervention, where GDNS service users would be given access to servers placed at the locations the LDNS service users are redirected to, the prediction presents an important challenge because we do not know $f(tput|rtt, GD, tod)$ for some of the RTT values for which $f(RTT|LDNS) > 0$. This point is a limitation that is common to many machine learning problems, where the amount of available information limits the range of predictions we can make. We are currently working on the extension of the method to the case where less information is available. This extension would require a parametric model to extrapolate the different pdfs out of the domains where the variables of our system are observed. For the purpose of this paper we do not need such extrapolations and manage to obtain the predictions we want directly from our observations of the system.

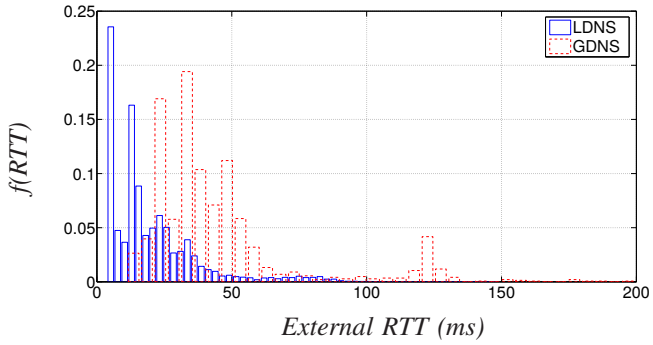

Fig. 2. Histogram of the external RTT for the local DNS (LDNS) and Google DNS (GDNS)

Results: The result of the intervention is presented in Figure 3 with cumulative distribution functions (CDFs). The CDF of the throughput for the LDNS before intervention is plotted (blue solid line) with the CDF of the throughput for the LDNS service users after an intervention setting their external delays distribution to the delay distribution seen by the GDNS users (red dotted line). We can observe the distribution being clearly shifted towards lower values of the throughput.

The expected throughput for clients using the local DNS service prior to intervention is **3.5 Mbps** and **3.0 Mbps** after intervention (14% decrease). This result shows the gain in performance that the redirection to closer CDN servers, provided by the use of the local DNS service, represents. We can assign the 14% decrease in the throughput to the higher values of RTT given to LDNS users in the intervention. These higher values correspond to the RTT of the GDNS users and reflect the suboptimal redirections of user demands to server hosting the requested resources.

Interpretations: The results we obtain cannot be verified in practice as this manipulation would require moving the servers hosting the resources that the local DNS users want to access to locations the Google DNS service redirects his users to and force the local DNS to redirect its users the way Google DNS would. In fact, this is precisely the benefit of the causal approach we adopted: our model offers the possibility to predict the effect of interventions that are difficult to perform experimentally. From the prediction of such interventions we can derive and quantify the impact of the DNS service choice on user performance (throughput).

The results show that, the local DNS obtains a gain in

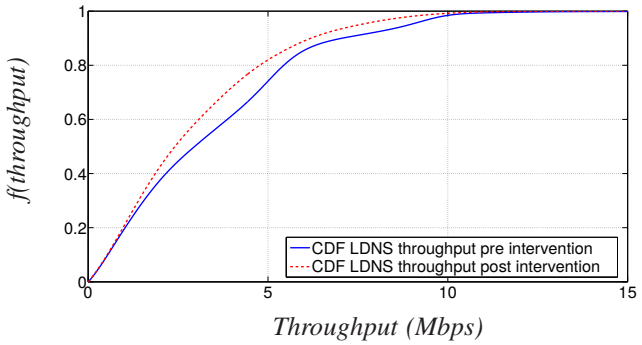

Fig. 3. Evolution of the throughput distribution before and after intervening on the external delay experienced by Local DNS (LDNS) clients

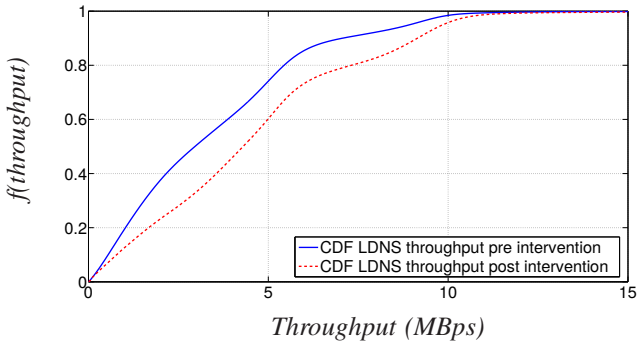

Fig. 4. Evolution of the throughput distribution before and after intervening on the minimum congestion window of servers of the users of the local DNS

performance by decreasing the RTT delay experienced by its users. This difference in the external delay is due to the non-optimal redirection to CDN servers by the Google DNS service. The gain was quantified by comparing the original performance to the one the users of the local DNS would have experienced if a different strategy was used.

The previous results did not consider the impact of the servers themselves, captured by the minimum congestion window in our problem, or other parameters such as the loss (*retrscore*) that are different between the two DNS services and could explain the throughput experienced, in the original dataset, by the users of the GDNS service that is only 7% smaller. As we can see from the causal model, Figure 1, the loss parameters (*retrscore* and *rto*), internal delay parameters (*isprttavg* and *isprttstd*) or maximum congestion window (*cwinmax*) are not influenced by the choice of the DNS service (*dns*). Therefore, the next section focuses on the impact of the minimum congestion window on the performance. Note that *cwinmin* is a direct parent of the throughput (*tput*) and is influenced by the DNS service choice (*dns*).

2) Minimum congestion window: As mentioned previously, the minimum congestion window (*cwinmin*) is a direct parent of the throughput (*tput*), see Figure 1. Its average value is higher for the clients using the GDNS service than for the clients using the LDNS service (1.2kB and 0.9kB respectively). The difference in the expected value of the throughput of LDNS users (3.5 Mbps) and GDNS users (3.3 Mbps) is 6%, smaller than the gain for the LDNS users being redirected to closer server, that is estimated to be 14%. We make the hypothesis that the minimum congestion window represents a difference in the configuration of the servers accessed by the LDNS users and the configuration of the servers accessed by the GDNS users. To study this hypothesis we estimate the causal effect of the minimum congestion window on the throughput, mediated by the choice of the DNS service.

In this section we are interested in estimating the influence of the DNS service on the throughput via its impact on the minimum congestion window. It is equivalent to ask the question: "*What would be the throughput for the clients using the local DNS if the servers they are redirected to would present the same minimum congestion window as the ones Google DNS users are redirected to ?*".

For space reasons, and because the method is the same for the minimum congestion window (*cwinmin*) as it was for the external delay (*inetrttavg*), we only write down the final equation corresponding to our problem.

We also observe, from the causal graph of Figure 1 (cf the explanation of d-separation in Section II-B):

- $(CWINMIN \perp\!\!\!\perp DNS)_{G_{\underline{DNS}}}$
- $(CWINMIN \perp\!\!\!\perp TPUT|DNS, INETRTTSTD)_{G_{\underline{CWINMIN}}}$

For the sake of brevity, we denote *cmin* the minimum congestion window (*cwinmin* in our model) and σ_{rtt} the standard deviation of the external rtt (*inetrttstd* in our model). We still use *LD* when referring to the local DNS and *GD* for the Google DNS. Therefore, we obtain the following equation:

$$f(tput \mid LD)_{f(cmin|do(GD))} = \int_{D_{CMIN}} \int_{D_{\sigma_{RTT}}} f(tput|do(cmin), LD, ts) f(\sigma_{rtt}) f(cmin|do(GD)) P(GD) = \int_{D_{CMIN}} \int_{D_{\sigma_{RTT}}} f(tput|cmin, LD, \sigma_{rtt}) f(\sigma_{rtt}) f(cmin|GD) P(GD) \quad (11)$$

Results: From Equation (11) we can predict the distribution of the throughput for the LDNS users after an intervention where we give to the minimum congestion window the distribution seen by GDNS users. The CDFs of the pre-intervention throughput (solid line) and post-intervention throughput (dotted line) are presented in Figure 4 where we can see the gain in throughput corresponding to the intervention on the LDNS server minimum congestion windows. The expected throughput for LDNS service users after the intervention is **4.6 Mbps** (against **3.5 Mbps** prior to intervention) and represents a throughput gain superior to 32%. The gain observed for this intervention is due to the fact that the servers GDNS service users are redirected to use higher values for their minimum congestion window.

Remark: The study of the opposite intervention, where GDNS service users are redirected to servers with a minimum congestion window following the distribution of the minimum congestion window seen by the LDNS service users, in the original dataset, is more complex. The reasons are the same as the ones mentioned in Section IV-B1. If we observe the distribution of the minimum congestion windows for LDNS service users and GDNS service users, Figure 5, we can notice the absence of *cmin* values for GDNS users to estimate $f(tput|cmin, GD, \sigma_{rtt})$

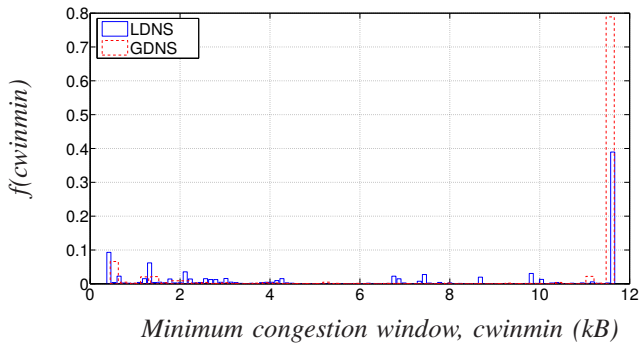

Fig. 5. Histogram of the minimum congestion window for the Local DNS (LDNS) and Google DNS (GDNS)

for values of *cmin* where $f(cmin|LD) > 0$. Such limitation is common to many machine learning problems, where the amount of available information limits the range of predictions.

The results obtained for the intervention on the external RTT were supported by the observations of the system that showed that the local DNS redirects its users to closer servers and, by doing so, improves their throughput (relatively to the situation where the local DNS would not do so). These section findings verify a hypothesis that was made when looking both, at the data and the causal graph. It appears that, while the proximity of the server has an important impact on the throughput, the configuration of the server hosting the content a client wants to access also has an important impact. The impact of a server configuration on its client throughput can be noticed from two different observations. First, from the data, the GDNS service users obtain an expected throughput value (3.3 Mbps) similar to the throughput of the LDNS service users (3.5 Mbps) while experiencing a bigger external delay (48 ms vs 20 ms). Second, by comparing the expected gain in performance corresponding to the two interventions we studied in our causal study. On the one hand, we could observe that the users of the LDNS service are redirected to closer servers, this redirection was estimated to represent a 14% gain of performance. On the other hand, the expected gain in performance when we modify the minimum congestion window of the servers accessed by the LDNS users was estimated to a 30% additional gain in performance. Comparing these two gains allows us to explain the throughput experienced by the GDNS service users and indicates how to improve the throughput of the users of the LDNS service.

The impact of the server TCP stack parameter tuning of the servers is studied in [13]. This work mainly focuses on short connections but shows the improvements that tuning the initial congestion window will have on performance. It is important to notice that, in causal models such as the one presented in Figure 1, a given node named according to a parameter X also represents the influence that external factors impacting *only this parameter* have on the rest of the system. Having said so, we can extrapolate our hypothesis on the minimum congestion to the TCP stack tuning parameters (such as the additive increase value for each acknowledged packet) and suppose that the servers that the Google DNS redirects its clients to have a more aggressive TCP parameterization that allows the servers to reach faster a higher value of their *sending rate* (number of packets sent at each TCP round).

V. Related work

This paper is in line with our two previous works [14], [15] where causal model inference and atomic predictions were presented. However, in this study of the causal impact of DNS on client performance, we do not focus on the technical details of adapting existing tools to the constraints of the system (see [14], [15]) and focus on the exploitation of causal theory. We present a case where more parameters are present (including categorical data) and use the causal model to explain non intuitive observations (namely a similar throughput for connections experiencing a different RTT). Most importantly we study interesting, while more complex, scenarios based on counterfactuals that give a very deep understanding of the causal mechanisms at play.

The two works closest to ours are the WISE system [16] and the Nano system [17]. The first one also uses the PC algorithm [8] to infer a graphical causal model from which interventions are then predicted. In this sense this work is very similar to ours. However, it differs a lot in the approach being used and the independence criterion adopted. An important domain knowledge is required in this approach and the independence criterion used, based on linear dependencies, gives very poor results for our problem. In addition, their study focuses on simpler scenarios of intervention while relying on more important quantity of data and resources. Our approach takes full advantage of the causal theory from [5], [6] to predict interventions and counterfactuals. Counterfactuals are very interesting questions for understanding the causal role of the different parameters of a system and, to our knowledge, scenarios such as the ones presented in Section IV-B have not been treated so far.

On the other hand, Nano tries to detect network neutrality violation by assessing the causal direct effect between the quality of experience of a user from a given ISP and the type of content being accessed. A performance baseline is defined based on observations made for different ISPs sharing similar configurations and then compared to the one observed for a particular scenario. Again, this approach uses domain knowledge to define the possible confounders and condition on these variables to remove spurious associations. Without a formal causal model this approach presents some risks. One of the confounders could be a collider in the corresponding causal graphical model. Conditioning on a common effect induces a dependencies between two independent causes whose influence tries to be canceled, questioning the obtained results.

Many studies have been made on DNS caching strategies, see [18] and references therein, and represent a different aspect of DNS performance study. In [19] the authors also present the impact of different replicas strategies in CDNs and client usage of DNS response. These works, while differently tackling the same question, present complementary results. The main contribution of this paper resides in the presented methodology based on the inference and usage of a causal model that allow us to estimate the causal effect of the DNS on user performance and goes beyond the simple parameter observations. Several works have been made concerning the study of DNS service choice and its impact on client performance [3], [2], [1]. These works mainly rely on active measurement and, while supporting the results found in this paper, differs greatly in their approach and objectives.

VI. Concluding remarks

In this paper, we present a new approach to study the effect of the DNS service on throughput performance. Using a causal approach, supported by the inference of causal model represented by a Bayesian graph, we are able to study the causal effect of a DNS service on the TCP throughput. We compare the performance of clients using their ISP local DNS service to the performance of clients using the Google DNS service. The causal model we obtain allows to unveil dependencies that would be very difficult to extract otherwise from data such as RTT, loss frequency, DNS service, throughput, and so on. This study only focuses on clients downloading files bigger than 2 MB from Akamai servers. However, it appears that the choice of the DNS server has a strong impact on the location of the servers to which the clients are redirected to, which impacts both the distance from clients to servers as well as the servers configuration. These two facts are captured by the dependence between the DNS and the RTT as well as the dependence between the DNS and the server minimum congestion window in the inferred causal model.

A very interesting property of causal models is their "*stability under intervention*". The model inferred from data following a given distribution is still valid when we want to predict the effect of modifying this distribution. We predict, in Section IV-B, the performance a client using the local DNS service would have experienced if redirected to servers the Google DNS service would have redirected it to. In this way, we can estimate the impact of the local DNS service redirection strategy on the client performance and quantify the corresponding performance difference.

When comparing the performance of the local DNS and the Google DNS users, we can observe that Google DNS users experience a throughput whose difference with the one of the local DNS service users cannot simply be explained by the redirection of Google DNS users to more distant servers. Based on the causal graph obtained in Section IV-A, we can formulate the hypothesis that the configurations of such servers allow the Google DNS users to eventually experience a performance similar to the one of the local DNS service users. This hypothesis is verified by our prediction consisting in giving to servers serving the local DNS users a minimum congestion window equivalent to the one of the servers serving Google DNS users. We estimate the gain in throughput corresponding to this intervention to be 32%. By comparison, the gain in terms of throughput corresponding to the better redirection of the local DNS users is estimated to 14%.

Compared to our previous works [14], [15], we demonstrated the real potential of adopting a causal approach. Counterfactuals are one of the possible way to approach Causality and this work follows this line. We evaluate the effect of a parameter on the system performance by predicting the effect that changing its parent would have with the rest of the system parameters left unchanged. We manage to answer questions such as "How would the system behave under the condition *C1* if one of his parameter was behaving as it would have under the condition *C2*, knowing that these two conditions are exclusive ?". The ability to predict such scenarios is a very powerful usage of the inherent mechanisms underlying the development of Causality. Counterfactuals are relatively complex to study, explain and even more to predict. However, with the usage of Bayesian networks as representation of the causal model of our system, we make the prediction of counterfactuals simpler to understand and to estimate.

Complex interventions where many parameters are modified simultaneously require important resources in terms of data and computational power. The results presented in this paper represent the first successful attempt to perform such a study. Based on this work, we are confident in the fact that the underlying tools and methods can be improved to reduce the required resources and increase both, the accuracy of such predictions and the range and complexity of the interventions that one can consider. In particular, the extension of prediction of such counterfactuals in cases where the two conditional probabilities have only partial overlap is a limitation of our approach and studies are currently being made to fit parametric models to overcome this issue.

Acknowledgment

The authors would like to thank Elias Bareinboim, from UCLA, for his advice and support developing the methodology for predicting counterfactuals from passive observation and our causal model. The research leading to these results has received fundings from the European Union under the FP7 Grant Agreement n. 318627 (Integrated Project "mPlane").

References

[1] J. S. Otto, M. A. Sánchez, J. P. Rula, and F. E. Bustamante, "Content delivery and the natural evolution of dns: Remote dns trends, performance issues and alternative solutions," in *IMC*, 2012.

[2] C. Huang, D. A. Maltz, J. Li, and A. G. Greenberg, "Public DNS system and global traffic management." in *INFOCOM*. IEEE, 2011.

[3] B. Ager, W. Mühlbauer, G. Smaragdakis, and S. Uhlig, "Comparing DNS resolvers in the wild," in *IMC*, 2010.

[4] A. Darwiche, *Modeling and Reasoning with Bayesian Networks*, 1st ed. Cambridge University Press, 2009.

[5] J. Pearl, *Causality: Models, Reasoning and Inference*. Cambridge University Press, 2009.

[6] P. Spirtes, C. Glymour, and R. Scheines, *Causation, Prediction, and Search*, 2nd ed. The MIT Press, Jan. 2001.

[7] V. Mayer-Schönberger, *Big Data: A Revolution That Will Transform How We Live, Work and Think*. John Murray Publishers, 2013.

[8] P. Spirtes and C. Glymour, "An Algorithm for Fast Recovery of Sparse Causal Graphs," *Social Science Computer Review*, vol. 9, 1991.

[9] P. Jaworski, F. Durante, W. Härdle, and T. Rychlik, "Copula theory and its applications," in *Lecture Notes in Statistics, Springer*, 2010.

[10] A. Finamore *et al.*, "Experiences of internet traffic monitoring with tstat," *Network, IEEE*, vol. 25, no. 3, May 2011.

[11] A. Gretton, K. Fukumizu, C. H. Teo, L. Song, B. Schölkopf, and A. J. Smola, "A kernel statistical test of independence." in *NIPS*, 2007.

[12] P. Ford, A. Shelest, and N. Srinivas, "Method for automatic tuning of TCP receive window," Aug. 15 2002, uS Patent App. 09/736,988.

[13] N. Dukkipati *et al.*, "An argument for increasing TCP's initial congestion window," *SIGCOMM*, vol. 40, no. 3, 2010.

[14] H. Hours, E. Biersack, and P. Loiseau, "A causal study of an emulated network," in *10ème Atelier en Evaluation de Performances*, Jun. 2014.

[15] ——, "Causal study of network performance," in *ALGOTEL*, Jun. 2014.

[16] M. Tariq *et al.*, "Answering what-if deployment and configuration questions with wise," in *SIGCOMM*, 2008.

[17] M. Tariq, M. Motiwala, N. Feamster, and M. Ammar, "Detecting network neutrality violations with causal inference." in *CoNEXT*, 2009.

[18] N. Fofack and S. Alouf, "Modeling modern DNS caches," in *ValueTools*. ICST, 2013.

[19] T. Callahan, M. Allman, and M. Rabinovich, "On modern DNS behavior and properties," *SIGCOMM*, vol. 43, no. 3, 2013.