Localizing Neutrality Violations

Zeinab Shmeis, Muhammad Abdullah, and Pavlos Nikolopoulos

Net Neutrality is an ongoing debate; regulators in both the U.S. and Europe are going back and forth on whether to enforce net neutrality rules and in what form. One key element in this debate is the lack of transparency regarding the traffic-prioritization strategies deployed by the ISPs.

In an attempt to elevate a level of transparency in the network, Wehe [1], an application already running on M-Lab servers, uses two-sample hypothesis testing to detect end-to-end differentiation. More specifically, Wehe exchanges two types of traffic between a server and a user: one that replays captured data from actual applications (such as Youtube, Netflix, Skype, etc), and another one that transfers randomized data. If the two traffic types experience significantly different throughputs over time, this means that at least one ISP on the end-to-end path between the server and the user applies traffic throttling. The Wehe app has been widely accepted by users (with ~ 300 tests taking place every day), and traffic differentiation has been detected in more than 150K cases.

In this manuscript, we propose extending Wehe by using Network Tomography. The key idea is to leverage the knowledge of the ISP-level topology and design Wehe tests, such that traffic differentiation can be localized to a particular link sequence, not only an end-to-end path. Moreover, our approach can also detect differentiation depending on network-layer (not only content) criteria.

Problem Formulation and Approach

Consider the typical Wehe setting explained above, where one server exchanges two types of traffic with a user, e.g., YouTube data and traffic of random content, and suppose that traffic differentiation is detected, i.e., the end-to-end performance w.r.t. YouTube traffic is significantly worse than the other. We ask: does that differentiation occur inside the access ISP of the user or elsewhere?

To answer this question, we want to use side-information from the network topology and apply network tomography to infer link-level performances w.r.t both traffic types. Obviously, if we are able accurately estimate both performances at each link, then all we need to localize the differentiation is to simply identify the link where the two differ significantly, in a statistical sense. However, as we see next, using only one end-to-end path between the server and the user (as Wehe does) cannot provide accurate link-level estimates [2].

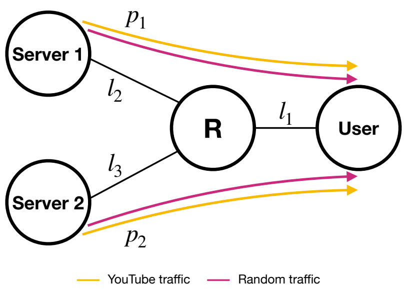

Figure 1: Example Topology

Network-tomography approach: Let us use the term "good" (resp. "bad") to refer to links/paths showing a high (resp. low) forwarding performance w.r.t. some traffic type. Accordingly, let x_i (resp. y_j) denote the expected fraction of time of a link i (resp. path j) being good. Network tomography consists of inferring all link-level x_i from path-level $y_j = \prod_{i \in j} x_i$. This formulation implicitly assumes that link performances are independent; hence the expected fraction of time that a path is good is just the product of the link-level expectations.

With the help of Figure 1, we showcase why the typical Wehe architecture is not always enough to localize differentiation . Suppose that Server 1 and the User run a Wehe test along path p_1. Server 1 sends both YouTube traffic (i.e. the traffic we suspect is throttled) and some traffic of random content. If YouTube traffic experiences policing at router R, the forwarding performance of link l_1 w.r.t. YouTube is significantly worse—formally: $x_{l_1}(\text{YouTube}) < x_{l_1}(\text{rand})$. Wehe is able to detect differentiation at the path level (i.e. $y_{p_1}(\text{YouTube}) < y_{p_1}(\text{rand})$), but unfortunately cannot tell whether it is because of link l_1 or l_2. The reason is that the number of paths it considers ($= 1$) is smaller than the number of the link-state variables we want to infer ($= 2$). Hence, x_{l_1}, x_{l_2} cannot be reconstructed from y_{p_1}—the inference problem is said to be *ill-posed.*

Instead, our approach consists of using two (or more) Wehe servers. Consider again Figure 1, but this time assume that both Servers 1 and 2 send the same traffic to the user. We can now construct two systems of equations, as in [3]:

$$\begin{aligned}
y_{p_1}(\text{YouTube}) &= x_{l_1}(\text{YouTube}) \cdot x_{l_2}(\text{YouTube}) & y_{p_1}(\text{rand}) &= x_{l_1}(\text{rand}) \cdot x_{l_2}(\text{rand}) \\
y_{p_2}(\text{YouTube}) &= x_{l_1}(\text{YouTube}) \cdot x_{l_3}(\text{YouTube}) & y_{p_2}(\text{rand}) &= x_{l_1}(\text{rand}) \cdot x_{l_3}(\text{rand}) \\
y_{p_{1,2}}(\text{YouTube}) &= x_{l_1}(\text{YouTube}) \cdot x_{l_2}(\text{YouTube}) \cdot x_{l_3}(\text{YouTube}) & y_{p_{1,2}}(\text{rand}) &= x_{l_1}(\text{rand}) \cdot x_{l_2}(\text{rand}) \cdot x_{l_3}(\text{rand}),
\end{aligned}$$

where the two equations of the last line describe the expected fraction of time that both paths p_1 and p_2 are jointly (simultaneously) good w.r.t. a traffic type. These are typical 3×3 systems that can be solved to infer all x's from y's.

Research steps

We now describe the challenges of our approach and how we plan to address them in the M-lab/Wehe setting:

Step 1. Identifying Measurement Paths (1 month): As hinted in the previous section, our approach requires the network topology between the Wehe servers and the user to satisfy a specific property: paths p_1 and p_2 have to merge at an intermediate node and then follow a common path up to the user ("Y"-shaped topology). Moreover, no loops are allowed. In practical scenarios, we usually want to identify neutrality violations within the user's access ISP; so, the merging node must be located inside that access network. These, in turn, impose several challenges:
(a) Do such topologies exist in practice? How easy it is to find them, i.e. identify at least 2 Wehe servers, whose paths to the user merge at a common point inside the destination ISP? Our search over M-lab's BigQuery database reveals that such topologies do exist; $\sim 39\%$ of the traceroutes from one day return useful topologies for network tomography.
(b) How easily/fast can we identify Y-shaped topologies without loops by processing the entire BigQuery database? Since new Wehe users may show up and/or existing users may request different tests from various ISPs, our approach should be able to find valid topologies reasonably fast. However, this becomes non-trivial as the number of intermediate nodes along each path increases. In fact, one can show that this scales as an NP hard problem. For that reason, we plan to rely on low-complexity approximation algorithms from theoretical CS or efficient heuristics that are able to quickly identify valid topologies, while at the same time eliminate false positives (i.e. invalid topologies). Our preliminary results on that front are promising; we have come up with a greedy algorithm that can return a large number of valid topologies within seconds. But, further improvement is required to achieve fewer false negatives.
(c) Instead of leveraging the BigQuery database that is frequently, yet not constantly updated, can we collect traceroute data online? Although we have not thoroughly investigated this, it seems possible within the Wehe/M-Lab framework.

Step 2. Neutrality Inference algorithm (3 months): The proposed solution in the previous section offers only a theoretical framework. To apply it in practice, we need to properly define the abstractions of "good" and "bad" paths/links via actual performance metrics. Such metrics must be easily computed from the traffic (e.g., loss, throughput, delay) and should reflect traffic differentiation. So far, we have explored two definitions: a path/link is good (a) if its average packet loss is below a certain threshold, or (b) if average throughput is above a certain rate. Packet loss and throughput are metrics satisfying both above conditions: First, they can be easily measured at the server or user side. Second, traffic policing usually limits the average throughput and causes packet loss to increase.

Our preliminary simulations with these metrics are promising when considering applications running over UDP; however, things are more challenging with TCP. This is because TCP's congestion control algorithm paces the servers' sending rate—hence, packets are not always sent from the servers in a synchronized manner. In our context, this can be regarded as *noise* in the computation of the last-line equations of the above tomography systems of equations. Therefore, we plan to "de-noise" the computation of the metrics by leveraging side-information from the other parameters, such as the RTT, the TCP congestion state, the value of the congestion window, etc. Essentially, in each Wehe test, we want to identify periods of time, where all flows send roughly the same amount of traffic (i.e. TCP behaves like UDP). At the same time, we want to explore the feasibility of collecting these side-information from M-Lab's "TCP INFO" tool, which already collects statistics about TCP connections running on M-Lab servers. Because of the extent to which applications rely on TCP nowadays, we believe that this is an interesting problem on its own.

In addition to TCP-related issues, we need to engineer the packet-loss and throughput thresholds that determine whether a path/link is good or not. Rather than looking for a magic threshold, we propose using multiple threshold values (that can be extracted from the measurements themselves, such as percentiles) and define differentiation-detection criteria based on multivariate analysis. Testing with a prototype of such an approach and without applying any de-noising as mentioned above, we were able to accurately localize traffic policing in 15 out of 20 simulations with TCP traffic.

Step 3. Implementation (1 month): Due to the involvement of multiple Wehe servers in a tomography test, the existing Wehe code and architecture need to be modified. Several challenges need to be addressed: (a) Up-to-date information regarding ISP-level topologies needs to be periodically gathered and then stored. To make it easily accessible, we plan to store it on M-Lab's Google Cloud Storage. (b) Servers have to initiate measurements almost simultaneously, so that the average time that both paths are good is estimated accurately. This requires back-and-forth checks at the client. (c) Measurement data collected from multiple servers have to be relayed to a server that solves the tomography problem. This can be done either directly or via the client acting as a proxy.

All of this has to be accomplished at an acceptable time and bandwidth cost to the Wehe mobile app user. In our preliminary implementation, we have set up 3 local instances of the Wehe server and a client running our modified version of the Wehe app. Since the current Wehe architecture does not have a server-server communication channel, we use the client as a proxy for relaying measurement data. We collect traces for Wehe-supported applications through our 3 local servers simultaneously and then relay them via the client to a local server that performs tomography. Before relaying the traces, we remove the payload and compress them reducing their size by at least 95%. We find that the additional bandwidth overhead with the client acting as a proxy is less than 2% on average, suggesting that the tomographic measurements can be collected without significant changes in the Wehe architecture.

References

[1] Fangfan Li, Arian Akhavan Niaki, David Choffnes, Phillipa Gill, and Alan Mislove. A large-scale analysis of deployed traffic differentiation practices. In *Proceedings of the ACM Special Interest Group on Data Communication*, pages 130–144. 2019.

[2] Hung Xuan Nguyen and Patrick Thiran. The boolean solution to the congested ip link location problem: Theory and practice. In *IEEE INFOCOM 2007-26th IEEE International Conference on Computer Communications*, pages 2117–2125. IEEE, 2007.

[3] Zhiyong Zhang, Ovidiu Mara, and Katerina Argyraki. Network neutrality inference. *ACM SIGCOMM Computer Communication Review*, 44(4):63–74, 2014.