Distributed Internet Paths Performance Analysis through Machine Learning

Sarah Wassermann*, Pedro Casas†
*Inria Paris, †AIT Austrian Institute of Technology
*sarah.wassermann@inria.fr, †pedro.casas@ait.ac.at

***Abstract*—Internet path changes are frequently linked to path inflation and performance degradation; therefore, predicting their occurrence is highly relevant for performance monitoring and dynamic traffic engineering. In this paper we showcase DisNETPerf and NETPerfTrace, two different and complementary tools for distributed Internet paths performance analysis, using machine learning models.**

***Index Terms*—Distributed Active Measurements; Reverse Traceroute; RIPE Atlas Measurement Framework; Machine Learning; M-Lab.**

I. Introduction

Monitoring Internet paths performance is critical for general network management. DisNETPerf [1] and NETPerfTrace [2] serve as powerful tools for tracking and analyzing Internet paths. DisNETPerf is a distributed platform capable of doing reverse `traceroute` measurements, collecting path measurements from any source to any destination, even when the source is not under the control of the experimenter. DisNETPerf uses RIPE Atlas, a largely distributed active measurements platform to perform `traceroute` measurements from any arbitrarily selected server in the Internet. NETPerfTrace is an Internet Path Tracking system capable of forecasting path changes and path latency variations, by analyzing `traceroute` measurements through machine learning models. NETPerfTrace can predict the remaining life time of a path before it actually changes, the number of path changes in a certain time-slot, as well as path latency metrics, providing a system which could not only predict path changes but also forecast their impact in terms of performance variations. Both DisNETPerf and NETPerfTrace are distributed as open software to the network measurement community.

II. Reverse Traceroute with DisNETPerf

The primary goal of DisNETPerf is to compute and monitor the path from a given content server to a specific user. The current version of DisNETPerf locates the closest RIPE Atlas probe to this content server, and gathers information about the path leading from the selected probe to the customer. DisNETPerf is open source and freely available on GitHub (https://github.com/SAWassermann/DisNETPerf).

DisNETPerf uses a combined topology- and delay-based distance notion to locate a RIPE Atlas probe that is as close as possible to a desired target destination, from which reverse `traceroute` measurements should be run. By doing so, DisNETPerf aims at locating probes which offer a very high path similarity to the real reverse path.

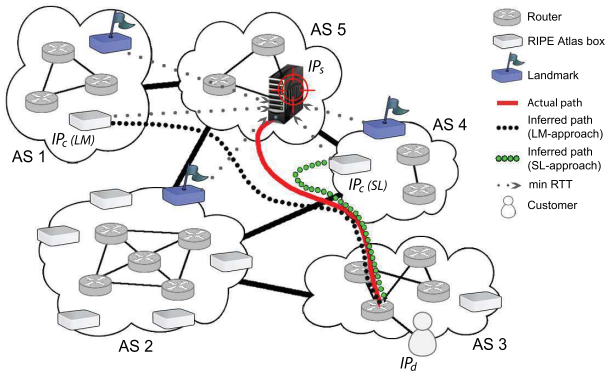

Figure 1. DisNETPerf overview. The first step of DisNETPerf consists of selecting a monitoring point or probe located as close as possible to a target server, to later on perform `traceroute` measurements towards specific destinations.

Fig. 1 describes the overall idea behind the DisNETPerf approach. In a nutshell, given a certain content server with IP address IP_s, and a destination customer with IP address IP_d, DisNETPerf pinpoints the closest box, namely IP_c, using a combined topology- and delay-based distance: probes are located first by AS – using BGP routing proximity to select probes in the same AS as IP_s – and then by propagation delay – for electing the closest probe to IP_s. DisNETPerf then periodically runs `traceroute` measurements from IP_c to IP_d, collecting different path performance metrics such as RTT per hop, end-to-end RTT, etc. This data might then be used to troubleshoot paths from the content server (mimicked by IP_c) to the target customer.

Current DisNETPerf implementation uses two different probe-selection approaches for locating IP_c, partially proposed in the literature for IP geolocation [7], [8], [9]. We called these selection approaches the *smallest latency* (SL) approach and the *landmark* (LM) approach, which we describe next.

A. Probe Selection by Smallest Latency

The SL approach starts by determining whether RIPE Atlas probes are located in the same AS as the targeted content server IP_s. If this is not the case, the SL approach tries to locate probes in the neighbor ASes of IP_s. Neighborhood information is obtained through AS relationships. We use CAIDA's AS relationships dataset [10]. If no probes are found in the neighbor ASes, then the SL approach randomly selects a large (and configurable) set of boxes among all the available

ones. We call these pre-selected probes the "candidate probes". Once the candidate probes have been identified, the selection of IP_c can start.

The SL approach then selects as IP_c the candidate probe with the smallest latency to the target IP_s. Latency is computed on the basis of standard `ping` measurements; more precisely, the SL approach issues 10 `ping` measurements from each of the candidate probes toward IP_s. The candidate probe with the smallest minimum RTT to IP_s is finally elected as the representative probe of the content server, i.e., IP_c. We consider the minimum RTT as it provides a rough estimation of the propagation delay between two IP addresses.

B. Probe Selection using Landmarks

The first step of the LM approach is exactly the same as the one followed by the SL approach, i.e., candidate probes are firstly selected based on their AS. However, the continuation is slightly different. The next step consists of grouping the candidate probes in two different sets: the *landmarks* and the probes that can be elected as IP_c. Landmarks are chosen randomly among all the candidate probes. Then, 10 `ping` measurements are issued from each of the landmarks toward IP_s and toward all the candidate probes belonging to the other set. For each pinged IP address, a feature vector d is computed, containing the minimum RTT from each landmark to this IP address. Finally, IP_c is selected as the probe with the most similar feature vector to the one of IP_s, according to the following normalized distance:

$$D_{ij} = \frac{1}{K} \sum_{l=1}^{K} |d_{il} - d_{jl}|,$$

where K is the number of landmarks providing a RTT for both IP_i and IP_j, and d_{il} is the minimum RTT between IP_i and landmark l. When D_{ij} is small, we assume that IP_i and IP_j are close to each other. Current DisNETPerf implementation uses 20 landmarks for each IP_s.

III. Predicting Internet Path Dynamics with NETPerfTrace

We define a path P as a sequence of links connecting a certain fixed source s to a fixed destination d. At any time t, path $P(t)$ is realized by a specific route r: this route consists of a specific sequence of links connecting s to d, and has an associated initial time t_0 when the route becomes active or in-place, and a final time t_f which corresponds to the time when r changes to another route realization, i.e., when the actual route changes. From now on, we therefore refer to route changes instead of path changes. As such, a path $P(t)$ can be considered as a statistical time process, composed of a set of time-contiguous routes $r_i(t_0^i, t_f^i)$, each one with a duration $D(r_i) = t_f^i - t_0^i$. For the sake of notation, we say that $r_i \in P$.

We additionally define the duration of a route r as $D(r) = t_f - t_0$, its current life time or *route age* at time t as $L_r(t) = t - t_0$, and its remaining life (i.e., time before a route change) at time t as $R_r(t) = t_f - t$. Finally, we define $rc_P(t)$ as the total number of route changes observed so far at time t for path P and $rc_{P_T}(t)$ as the number of route changes observed so far at time t for path P in the current time-slot T.

Given a new `traceroute` measurement at time t, the prediction problem solved by NETPerfTrace includes three prediction targets: (i) the remaining life time $R_r(t)$ of route r, namely $\widehat{R}_r(t)$, (ii) the number of route changes a path P experiences over a specific future time-window of length T, defined as $\widehat{rc}_{P_T}$, and (iii) the average RTT that path P will experience in the next `traceroute` measurement, defined as $\widehat{avgRTT}_P(t+\varepsilon)$, where ε represents the duration until the next measurement. The first two targets correspond to path dynamics prediction, whereas the third target consists of path performance forecasting. In practice, when $\widehat{R}_r(t)$ comes closer to zero, we would increase the sampling rate to better monitor the path performance in the event of a route change. Predicting $\widehat{rc}_{P_T}$ allows to dynamically identify which paths are more prone to frequent changes, and thus better allocate new `traceroute` measurements. Based on previous results on route stability [6], [3] and similar to [4], we focus on predicting the number of daily route changes for the next day, i.e., $T = 24$ hours from now on. At last, predicting the average RTT that a certain path P would experience next becomes highly relevant for dynamic traffic engineering purposes, and when combined with the prediction of route changes, it can provide a very powerful approach to forecast those performance-harmful route changes. To predict these three targets, NETPerfTrace uses a rich set of input features describing the statistical properties of route dynamics and path latency [2]. NETPerfTrace is released as open software to the community - https://github.com/SAWassermann/NETPerfTrace.

By carefully engineering NETPerfTrace underlying model and input features, we show in [2] that NETPerfTrace highly outperforms DTRACK [5], a previous system with the same prediction targets. In particular, NETPerfTrace outperforms DTRACK by a factor of 5 when forecasting the residual lifetime of a path with relative prediction errors below 10%, and by a factor of 7 in correctly predicting daily path changes.

References

[1] S. Wassermann et al., "On the Analysis of Internet Paths with DisNETPerf, a Distributed Paths Performance Analyzer," in *WNM Workshop*, 2016.

[2] S. Wassermann et al., "NETPerfTrace - Predicting Internet Path Dynamics and Performance with Machine Learning," in *Big-DAMA Workshop*, 2017.

[3] I. Cunha et al., "Measuring and Characterizing End-to-End Route Dynamics in the Presence of Load Balancing," in *PAM*, 2011.

[4] I. Cunha et al., "Predicting and Tracking Internet Path Changes," in *SIGCOMM*, 2011.

[5] I. Cunha et al., "DTRACK: A system to Predict and Track Internet Path Changes," in *IEEE/ACM ToN*, vol 22(4):1025–1038, 2014.

[6] V. Paxson, "End-to-end Routing Behavior in the Internet," in *SIGCOMM*, 1996.

[7] E. Katz-Bassett et al., "Towards IP geolocaiton using delay and topology measurements," in *IMC*, 2006.

[8] V. Padmanabhan at al., "An investigation of geographic mapping techniques for Internet hosts," in *SIGCOMM*, 2001.

[9] Y. Liao et al., "A lightweight network proximity service based on neighborhood models," in *IEEE SCVT*, 2015.

[10] The CAIDA UCSD, "AS relationships," 2015, http://data.caida.org/datasets/as-relationships/serial-1/20150601.as-rel.txt.bz2.