Reverse Traceroute with DisNETPerf, a Distributed Internet Paths Performance Analyzer

Sarah Wassermann*, Pedro Casas†

*Université de Liège
sarah.wassermann@student.ulg.ac.be

†AIT Austrian Institute of Technology
pedro.casas@ait.ac.at

***Abstract*—`traceroute` is the most widely used Internet path diagnosis tool today. A major limitation of `traceroute` when the destination is not controllable by the user is its inability to measure *reverse* paths, i.e., the path from a destination back to the source. In this demo session, we showcase DisNETPerf, a new tool to perform reverse `traceroute` measurements. DisNETPerf is able to collect measurements from the server to the user for path performance monitoring and troubleshooting purposes, even when the server 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.**

***Index Terms*—Distributed Active Measurements; Reverse Traceroute; RIPE Atlas Measurement Framework**

I. Why DisNETPerf?

Internet-scale services such as YouTube are provisioned from geo-distributed servers, using large Content Delivery Networks (CDNs). While user requests are normally redirected to the closest servers (in terms of latency), internal CDN load-balancing policies may select servers which lie at hundreds of milliseconds from customers, potentially impacting their Quality of Experience (QoE), especially under congestion of the downlink paths connecting servers providing services (e.g., YouTube servers) to customers. In such a context, it is very difficult for an ISP to find the root cause of the problem, as she should collect path performance data from the server perspective. A normal approach the operator would follow to troubleshoot the problem is to run `traceroute` measurements from some controlled node at the edge of her network (connecting her customers to the Internet) toward some servers provisioning the services which are getting impacted (e.g., YouTube and Facebook servers). Indeed, `traceroute` is still today the de-facto standard tool used by operators to investigate routing failures and performance problems [1]. By assuming path symmetry between their controlled node and the targeted servers, they would be able to get some initial hints on the performance of the end-to-end paths. However, assuming path symmetry at the Internet level is a major mistake [2], [3], as Internet paths often become asymmetric, especially at network boundaries, due to administration policies changes among others. This has been flagged as the number one "plague" of `traceroute` [1], and one can only assume that `traceroute` shows relevant information for the forward path. The reverse path itself is therefore completely invisible, and the only solution to determine the causes of the performance issues is to look at both forward and reverse `traceroute` measurements. However, `traceroute` has a major limitation when the destination is not accessible: it cannot measure the reverse path, i.e., from destination back to the source, as one cannot run measurements from the inaccessible destination. This is exactly the problem we tackle with DisNETPerf: performing *reverse* `traceroute` measurements.

Previous work proposed a tool to perform *reverse* `traceroute` [4], i.e., from the servers to the customers. A major drawback of the proposed approach is that it heavily relies on IP spoofing and IP Record Route Option, both being not necessarily allowed everywhere [5], [6] and potentially leading to security concerns for the case of IP spoofing. Therefore, in [7], [8], we have introduced DisNETPerf, a Distributed Internet Paths Performance Analyzer, which can monitor any Internet path using distributed active measurements. While DisNETPerf is not strictly tied to any particular distributed measurement platform, current DisNETPerf implementation relies on the well-known RIPE Atlas framework [9] for distributed active measurements.

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.

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

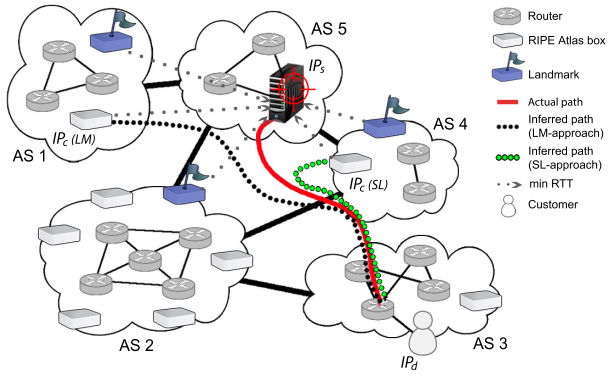

Figure 1. DisNETPerf overview. The first step of DisNETPerf consists of selecting a monitoring point or probe located as close as possible to a target server, to later on perform `traceroute` measurements towards specific destinations.

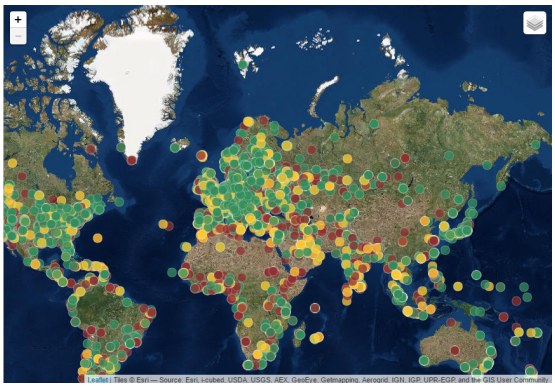

Figure 2. DisNETPerf distributed Internet measurements with RIPE Atlas. RIPE Atlas employs a global network of probes distributed worldwide to run large-scale active measurements.

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 [10], [11], [12]. 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 [13]. 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. Demoing DisNETPerf

We demo DisNETPerf within the use-case scenario described in Sec. I, in which an internal CDN load-balancing policy employed by Google selects YouTube servers which lie at hundreds of milliseconds from customers, impacting their QoE. As we said, it is very difficult for an ISP to find the root cause of the problem in such a context, as she should collect path performance data from the server perspective. Therefore, we run DisNETPerf to locate the best RIPE Atlas probes to diagnose the Internet paths connecting certain YouTube servers to a group of end users.

DisNETPerf runs locally on our laptop, and we assume it receives real-time monitoring information coming from the ISP's monitoring system. At a certain time, the monitoring

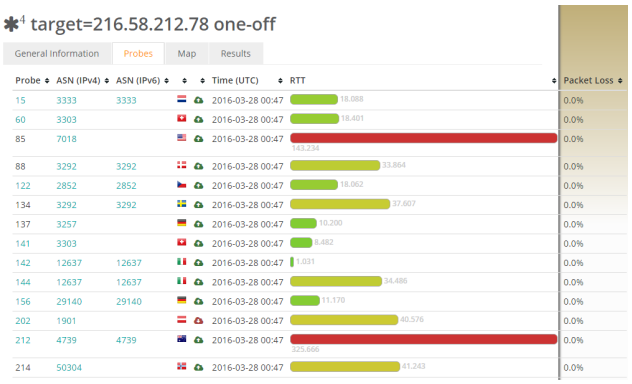

Figure 3. DisNETPerf in action. The first step of the probe-location process consists of the topology- and latency-based proximity estimation, for which RTT measurements are performed.

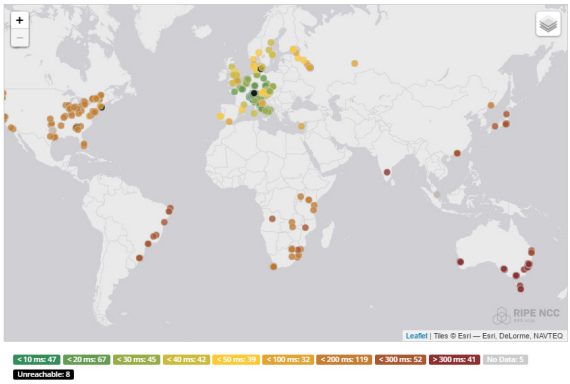

Figure 4. Candidate DisNETPerf probes in the world map. The YouTube IP is geo-located in Italy, and the DisNETPerf selection selects a probe near the city hosting the target.

system of the ISP detects an anomaly for YouTube traffic coming from a set of Google IPs, and immediately sends a report containing these IPs to DisNETPerf, which triggers the aforementioned reverse `traceroute` procedure for these YouTube IPs. The distributed platform used by DisNETPerf runs worldwide on top of RIPE Atlas and it is accessed by DisNETPerf through standard HTTP messages.

The demo of DisNETPerf uses a combination of command line, shell messages showing the interaction between DisNETPerf and the distributed platform, and a powerful web GUI provided by the RIPE Atlas platform. Fig. 2 shows a map reflecting the location of some of the potential RIPE Atlas probes which would be employed by the DisNETPerf demo to perform the reverse `traceroute` probe location.

As an example, let us consider that the YouTube server with IP address 216.58.212.78 is one of the servers involved in the anomaly detected by the ISP. DisNETPerf receives this IP address as input and instantiates an optimal probe location (using the SL approach in this scenario) to perform `traceroute` measurements from the selected probe to the prefix of the ISP customers having QoE issues with YouTube. Fig. 3 shows the starting phase of the DisNETPerf procedure, in which the YouTube IP address 216.58.212.78, geo-located in Milano (Italy), is chosen as target and is pinged from multiple distributed RIPE Atlas probes, selected based on topological-based notions. The candidate DisNETPerf probes are geo-located and displayed in the world map using the RIPE Atlas GUI, as depicted in Fig. 4. The DisNETPerf selection selects finally the closest RIPE Atlas box to the target (located near the city hosting the target in this case) to perform the reverse `traceroute` analysis.

IV. Demo Requirements

To run the DisNETPerf demo, we need a standard monitor (ideally large, 24" or more), a reliable Internet connection (Ethernet capable preferred) and a panel for poster handling. DisNETPerf runs directly on our laptop, which we would bring to IEEE LCN; the DisNETPerf distributed platform runs worldwide and it is accessed by DisNETPerf through standard HTTP messages. Showing a poster explaining the overview of DisNETPerf would help the demo attendees to better understand the demo procedures and the principles of DisNETPerf. For demo setup, we estimate about half an hour.

References

[1] R. A. Steenbergen, "A practical guide to (correctly) troubleshooting with traceroute," in *NANOG 45*, 2009, http://www.nanog.org/meetings/nanog45/presentations/Sunday/RAS_traceroute_N45.pdf.

[2] Y. He, M. Faloutsos, S. Krishnamurthy, and B. Huffaker, "On routing asymmetry in the internet," in *Global Telecommunications Conference, 2005. GLOBECOM '05. IEEE*, vol. 2, Nov 2005, pp. 6 pp.–.

[3] V. Paxson, "End-to-end routing behavior in the internet," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 5, pp. 41–56, Oct. 2006. [Online]. Available: http://doi.acm.org/10.1145/1163593.1163602

[4] E. Katz-Bassett, H. Madhyastha, V. Adhikari, C. Scott, J. Sherry, P. van Wesep, A. Krishnamurthy, and T. Anderson, "Reverse traceroute," in *Proc. USENIX Symposium on Networked Systems Design and Implementations (NSDI)*, June 2010.

[5] R. Beverly, A. Berger, Y. Hyun, and k. claffy, "Understanding the efficacy of deployed Internet source address validation filtering," in *Proc. ACM Internet Measurement Conference (IMC)*, November 2010.

[6] R. Fonseca, G. M. Porter, R. H. Katz, S. Shenker, and I. Stoica, "IP options are not an option," University of California at Berkeley, Technical Report UCB/EECS-2005-24, December 2005.

[7] S. Wassermann, P. Casas, and B. Donnet, "Towards DisNETPerf: a Distributed Internet Paths Performance Analyzer," in *Proc. of the ACM CoNEXT Student Workshop*, 2015.

[8] S. Wassermann, P. Casas, B. Donnet, G. Leduc, and M. Mellia, "On the analysis of internet paths with disnetperf, a distributed paths performance analyzer," in *Proc. IEEE Workshop on Network Measuremnets (WNM)*, 2016.

[9] RIPE NCC Staff, "RIPE Atlas: A Global Internet Measurement Network," Internet Protocol Journal, 2015.

[10] E. Katz-Bassett, J. P. John, T. Anderson, A. Krishnamurthy, Y. Chawathe, and D. Wetherall, "Towards IP geolocaiton using delay and topology measurements," in *Proc. ACM Internet Measurement Conference (IMC)*, October 2006.

[11] V. Padmanabhan and L. Subramanian, "An investigation of geographic mapping techniques for internet hosts," in *Proc. ACM SIGCOMM*, August 2001.

[12] Y. Liao, W. Du, and G. Leduc, "A lightweight network proximity service based on neighborhood models," in *Communications and Vehicular Technology in the Benelux (SCVT), 2015 IEEE Symposium on*, Nov 2015, pp. 1–6.

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