# NETPerfTrace – Predicting Internet Path Dynamics and Performance with Machine Learning

Sarah Wassermann
Université de Liège
sarah.wassermann@student.ulg.ac.be

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

Thibaut Cuvelier
Université de Liège
tcuvelier@ulg.ac.be

Benoit Donnet
Université de Liège
benoit.donnet@ulg.ac.be

## ABSTRACT

We study the problem of predicting Internet path changes and path performance using `traceroute` measurements and machine learning models. Path changes are frequently linked to path inflation and performance degradation, therefore the relevance of the problem. We introduce NETPerfTrace, an Internet Path Tracking system to forecast path changes and path latency variations. By relying on decision trees and using empirical distribution-based input features, we show that NETPerfTrace can predict *(i)* the remaining life time of a path before it actually changes and *(ii)* the number of path changes in a certain time period with relatively high accuracy. Through extensive evaluation, we demonstrate that NETPerfTrace highly outperforms DTRACK, a previous system with the same prediction targets. NETPerfTrace also offers path performance forecasting capabilities. In particular, our tool can predict path latency metrics, providing a system which can not only predict path changes, but also forecast their impact in terms of performance variations. We release NETPerfTrace as open software to the networking community, as well as all evaluation datasets.

## CCS CONCEPTS

• **Networks → Network performance modeling**; • **Computing methodologies → Supervised learning by regression**; *Feature selection*;

## KEYWORDS

distributed measurements; machine learning; `traceroute`; M-Lab; DTRACK.

**ACM Reference format:**
Sarah Wassermann, Pedro Casas, Thibaut Cuvelier, and Benoit Donnet. 2017. NETPerfTrace – Predicting Internet Path Dynamics and Performance with Machine Learning. In *Proceedings of Big-DAMA '17, Los Angeles, CA, USA, August 21, 2017,* 6 pages.
https://doi.org/10.1145/3098593.3098599


*Big-DAMA '17, August 21, 2017, Los Angeles, CA, USA*

ACM ISBN 978-1-4503-5054-9/17/08…$15.00
https://doi.org/10.1145/3098593.3098599

**The research leading to these results has been partially funded by the Vienna Science and Technology Fund (WWTF) through project ICT15-129, "BigDAMA".**

## 1 INTRODUCTION

Internet paths change frequently due to inter/intra-domain routing changes, load balancing, and even misconfigurations and failures [8]. Some of these changes can seriously disrupt performance, causing longer round-trip times, congestion, or even loss of connectivity [5]. For example, Google reports that inter-domain routing changes caused more than 40% of the cases in which clients experienced a latency increase of at least 100 ms [12]. These changes could not only impact the end users Quality of Experience (QoE), but also might turn to be quite costly: Amazon claims that every additional 100 ms of page load time could cost them 1% of their sales, and that a page load slowdown of just one second could turn into a $1.6 billion loss in sales each year. As such, predicting the time when a path is likely to change, as well as how such a change would impact end-to-end latency, becomes a highly relevant problem in practice.

To tackle this challenge, and similar to Cunha et al. [3, 4], we predict the time when a path change would occur by relying on `traceroute` measurements and supervised machine learning prediction models. We introduce *NETPerfTrace*, an Internet Path Tracking system allowing to predict the number of path changes in a certain time slot, to forecast the most likely time when these paths would actually change, as well as to predict their future path latency. Extensive evaluations using highly distributed `traceroute` measurements from M-Lab show that NETPerfTrace nearly perfectly predicts *(i)* the remaining life time of a path (i.e., the time before a path change) in about 30% of the cases, *(ii)* the exact number of daily path changes in about 70% to 80% of the cases, and *(iii)* the average RTT of a path in about 50% of the cases. In addition, we show that NETPerfTrace highly outperforms DTRACK [3, 4], a previous system conceived to predict Internet path changes.

NETPerfTrace relies on a standard random forest model for prediction, which provides accurate results with very low computational overhead as compared to other evaluated machine learning models; readers may refer to [9] for benchmarking results. We perform extensive evaluation on the impact of different input features by studying the correlations between the inputs and the prediction

targets, as well as by using feature selection techniques. NETPerfTrace and the datasets used in this paper are freely available[1], making results fully reproducible. We are currently extending our tool DisNETPerf [11] by adding an automatic approach to *dynamically* adapt the sampling rate of a path based on the remaining time until a next path change, similar to [3].

The remainder of this paper is organized as follows: Section 2 briefly reviews the related work. Section 3 describes the main concepts behind NETPerfTrace. Section 4 reports prediction results for NETPerfTrace using M-Lab `traceroute` measurements, including an evaluation on the impact of different input features, using feature selection techniques. Section 5 reports the results obtained in the comparative evaluation of NETPerfTrace and DTRACK. Section 6 concludes this work.

## 2 RELATED WORK

There is a very rich literature on using `traceroute` measurements to track Internet path dynamics and performance. Since the early work of Paxson on the analysis of end-to-end Internet routing behavior [8], multiple research efforts have targeted the study of Internet paths at a large scale. Systems such as DisNETPerf [11], iPlane [7], Reverse `traceroute` [6], and Sibyl [1] are all distributed measurement systems relying on `traceroute` measurements to track and predict Internet paths performance. DisNETPerf and Reverse `traceroute` particularly target the problem of measuring paths from arbitrary selected sources. iPlane and Sibyl both offer a service for predicting the performance of Internet paths, by building a structural model of the Internet using `traceroute` and opportunistic measurements.

While the problem of analyzing path changes at the Internet scale has attracted important attention in the past, only few papers have focused on predicting such path changes [3–5, 12], which is the target of this paper. Papers such as [5, 12] study the potential causes leading to Internet path changes, particularly those causing higher latencies [12]. Close to our work, authors in [3, 4] study the problem of predicting path changes using both `traceroute` measurements and machine-learning-based predictors. In particular, they develop a model based on $k$-nearest neighbors to predict both the remaining time of an established path before a change and the number of changes experienced by a path during a certain time period. Our work builds on these papers, using different modeling techniques and different input features for prediction.

This paper is an extension of our early work on path dynamics and performance prediction [10], presenting preliminary results of the techniques described next. A more complete report of the studies conducted in this work is available in [9].

## 3 PROBLEM STATEMENT

We define a path $P$ as the connectivity from a source $s$ to a 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 $d$. Route $r$ has an associated initial time $t_0$ when it becomes active or in-place, and a final time $t_f$ which corresponds to the time when $P$ 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 $r_i(t_0^i, t_f^i)$. $r_i \in P$ indicates that $r_i$ realizes path $P$.

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 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, denoted by $\widehat{avgRTT}_P(t+\varepsilon)$, where $\varepsilon$ 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 $\widehat{R}_r(t)$ becomes closer to zero, we would increase the sampling rate to better monitor the path performance in case of a route change. Predicting $\widehat{rc}_{P_T}$ allows to dynamically identify which paths are more prone to frequent changes, and thus to better allocate new `traceroute` measurements. Based on previous results on route stability [2, 8] and similar to [3], we focus on predicting the number of daily route changes for the next day, i.e., $T = 24$ hours. At last, predicting the average RTT that a certain path $P$ would experience next becomes highly relevant for dynamic traffic engineering purposes: when combined with the prediction of route changes, it can provide a very powerful approach to forecast those performance-harmful route changes.

In order to predict these three targets, we use a rich set of input features describing the statistical properties of route dynamics and path latency. Table 1 describes these features, separated into three different groups. Note that we compute all these statistical features from the `traceroute` measurements performed in an observation period $T_{learn}$ of the monitored paths for learning purposes. The first group of features, referred to as $F_A$, includes 11 features relevant to the prediction of $R_r(t)$. These features describe the statistical properties of the route duration $D(r)$ observed for each path $P$. $F_A$ also includes information about the currently active route $r$ at time $t$, namely its route age $L_r(t)$. The second group of features, referred to as $F_B$, includes 14 features relevant to the prediction of $rc_{P_T}$. $F_B$ features take into account the statistical properties of $rc_{P_T}$. In addition to that, $F_B$ contains information about the number of route changes observed for path $P$ and a binary feature indicating whether a route change occurred for $P$ in the current time slot $T$. The third group of features, referred to as $F_C$, includes 44 features relevant to the prediction of $avgRTT_P(t+\varepsilon)$. $F_C$ features account for the statistical properties (average, minimum, maximum, and percentiles) of the four RTT metrics reported in `traceroute` measurements, namely the average, minimum, maximum, and standard deviation of the end-to-end RTT. In addition, $F_C$ also includes the current value of end-to-end RTT metrics at time $t$. As we show next, these features are highly correlated to the corresponding prediction targets, resulting in a strong forecasting power.

[1] https://github.com/SAWassermann/NETPerfTrace

| **Residual Life Time $R_r$ feature set ($F_A$)** | 11 |
|---|---|
| average, minimum, and maximum of $D(r_i), \forall r_i \in P$ | 3 |
| 5-, 10-, 25-, 50-, 75-, 90-, 95-percentiles of $D(r_i), \forall r_i \in P$ | 7 |
| route age of route $r$ at time $t$ for $P$ | 1 |
| **# Route Changes $rc_{P_T}$ feature set ($F_B$)** | 14 |
| average, minimum, and maximum of $rc_{P_T}$ in $P$ | 3 |
| 5-, 10-, 25-, 50-, 75-, 90-, 95-percentiles of $rc_{P_T}$ in $P$ | 7 |
| total number of route changes in $P$ | 1 |
| total number of route changes in $P$ in $T$ | 1 |
| number of route changes in $P$ at time $t$ in $T$ | 1 |
| binary indication of a route change in $T$ | 1 |
| **Path Latency $avgRTT_P$ feature set ($F_C$)** | 44 |
| average of $RTT$ metrics in $P$: $mean\,(avg/max/min/dev\ RTT)$ | 4 |
| minimum of $RTT$ metrics in $P$: $min\,(avg/max/min/dev\ RTT)$ | 4 |
| maximum of $RTT$ metrics in $P$: $max\,(avg/max/min/dev\ RTT)$ | 4 |
| 5-, 10-, 25-, 50-, 75-, 90-, 95-percentiles of $RTT$ metrics $(avg/max/min/dev\ RTT)$ in $P$ | 28 |
| current $RTT$ metrics $(avg/max/min/dev\ RTT)$ at time $t$ | 4 |

**Table 1: Feature set used by NETPerfTrace.**

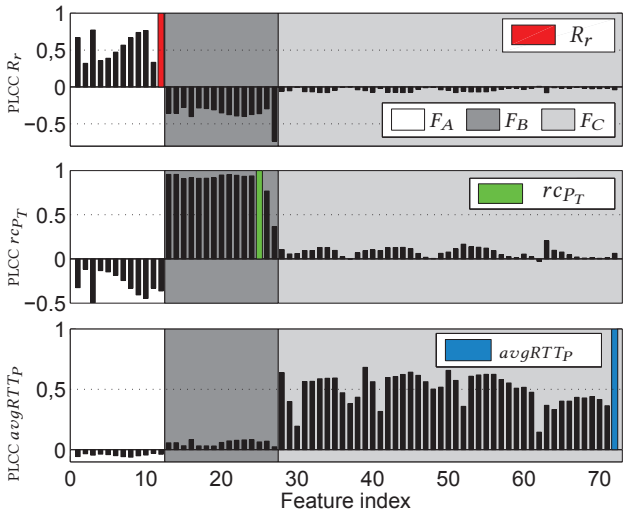



**Figure 1: Linear correlation between features and prediction targets, for feature sets $F_A$, $F_B$, and $F_C$.**

NETPerfTrace uses random forest (RF) as the underlying prediction model. In particular, we select a RF model with 10 trees (RF10). In [9], we present an in-depth benchmark comparing several machine learning models for NETPerfTrace. We based our preference for RF10 on both prediction performance and computational speed; see [9] for full insights.

## 4 NETPERFTRACE EVALUATION

In this section, we study the performance achieved by NETPerfTrace. Firstly, we introduce the evaluation dataset and study the correlation among input features and prediction targets. Next, we assess the prediction power of NETPerfTrace by comparing the real and predicted values for the three targets. Finally, we analyze the relevance of the input features in terms of prediction power, and apply feature selection techniques to select the best ones for each target. To avoid biased results, all evaluations in this paper are done on a 10-fold cross-validation basis.

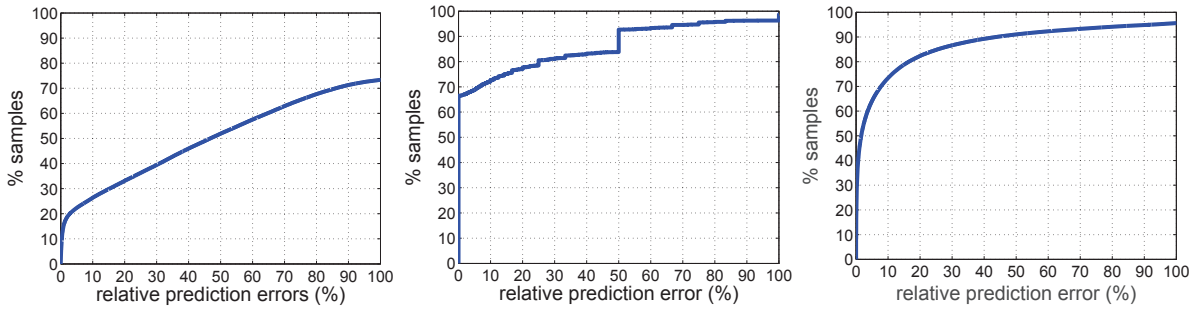
### 4.1 M-Lab Data Description

For the purpose of this study, we analyze a full week of Paris-`traceroute` measurements performed through the M-Lab open Internet measurement initiative[2]. The M-Lab infrastructure consists of a high number of servers distributed globally in multiple provider networks and geographic regions. M-Lab makes all data available, including packet traces and supplementary path measurements data. The raw data files are accessible through Google's BigQuery and Cloud Storage[3].

The analyzed dataset corresponds to the first week of January 2016. During this week, we observe more than 450,000 different paths, sampled through Paris-`traceroute` measurements from more than 180 geo-distributed servers. Unfortunately, not all of these paths are periodically sampled during this week, as M-Lab `traceroute` measurements are normally triggered as part of other experiments; traces are therefore known to be sporadic. Indeed, when analyzing the number of `traceroute` measurements for each of these paths, we found that only 15,725 paths have been sampled more than 10 times, and only 2,346 paths have at least 100 `traceroute` measurements during the analyzed week. We use 100 as threshold to avoid reducing the useful dataset even more, but naturally, the more `traceroute` measurements or samples we have for a path, the higher the visibility on potential route changes. Having 100 samples in a week means an average path sampling rate of one `traceroute` every 100 minutes, which is quite low but still a good starting point for the different analyses. In fact, the time between `traceroute` measurements in the resulting dataset is below 14 minutes for more than 50% of the measurements, and for more than 40% of the paths, the sampling rate is above one `traceroute` every 20 minutes. The total number of `traceroute` measurements in the resulting filtered dataset is above 550,000. Regarding path topology, the resulting 2,346 paths are issued from 82 different sources, distributed in 33 different ASes, and leading to about 2,000 different destinations in 125 different ASes. These paths traverse more than 260 different ASes, and have an average length of ten hops and four ASes. As such, we believe the used dataset is rich and representative of current Internet paths.

For each of these 2,346 paths $P$, we compute the distribution of the aforementioned input features during an observation period $T_{learn}$ = 1 week. As we mentioned before, while we use the full week of measurements to compute the input features for NETPerfTrace, evaluations are done on a 10-fold cross-validation basis, limiting potential bias.

[2] https://www.measurementlab.net/
[3] https://console.cloud.google.com/storage/browser/m-lab/




(a) $R_r(t)$ relative prediction errors. (b) $rc_{P_T}$ relative prediction errors. (c) $avgRTT_P$ relative prediction errors.

**Figure 2: Relative prediction errors for (a) $R_r(t)$, (b) $rc_{P_T}(t)$, and (c) $avgRTT_P(t)$, using input feature sets $F_A$, $F_B$, and $F_C$.**

## 4.2 Feature Correlation

Let us start by analyzing the correlations among input features and prediction targets. This would let us perform a first raw selection of features for each prediction target. Figure 1 depicts the Pearson linear correlation coefficients (PLCCs) between the full set of input features and the three prediction targets, discriminated by the feature sets $F_A$, $F_B$, and $F_C$. The set is extended by adding the three prediction targets, which are flagged by PLCC = 1 in the corresponding plot. As expected, features from each set present high positive correlation to the corresponding prediction target. Features from sets $F_A$ and $F_B$ are inversely correlated to targets $rc_{P_T}$ and $R_r$, respectively, which is coherent with the fact that more stable paths with smaller number of changes have longer life times. In addition, there is negligible correlation between path stability and path performance. Indeed, features from set $F_C$ are very weakly correlated to targets $R_r$ and $rc_{P_T}$, and features from sets $F_A$ and $F_B$ are very weakly correlated to $avgRTT_P$. Based on these initial observations, we consider each set of features $F_A$, $F_B$, and $F_C$ as individual inputs to predict $R_r$, $rc_{P_T}$, and $avgRTT_P$ next. In Section 4.4, we show that a more careful feature selection can improve the performance of NETPerfTrace.

## 4.3 NETPerfTrace Performance

Figure 2 reports the prediction performance achieved by NETPerfTrace using input features sets $F_A$, $F_B$, and $F_C$ for predicting $R_r$, $rc_{P_T}$, and $avgRTT_P$, respectively. Performance is measured in terms of relative prediction errors (RE), i.e., RE = $|\hat{X} - X|/X$, where $X$ and $\hat{X}$ are real and predicted values, respectively. Note that in the case of $rc_{P_T}$ prediction, we might have time slots for which $rc_{P_T} = 0$. Indeed, about 25% of the 24-hours time slots correspond to zero route change slots in the studied dataset. Therefore, RE values are reported separately when it comes to the estimation of $rc_{P_T}$. A first general observation is that predicting both $R_r$ and $avgRTT_P$ is more challenging than predicting $rc_{P_T}$. Indeed, REs are much higher, and according to Figure 1, PLCCs are much smaller. In particular, and as already pointed out by previous work [3, 4], predicting $R_r$ is difficult and error-prone.

Figure 2 (a) reports the distribution of the obtained REs for $R_r(t)$. NETPerfTrace correctly predicts $R_r$ for only about 20% of the samples, and achieves relative errors below 100% for more than 70% of the samples. Results are therefore quite disappointing, but as we show in Section 5, NETPerfTrace actually highly improves previous work for $R_r$ prediction. Finally, we found that NETPerfTrace underestimates $R_r$ for about 40% of the samples, and that prediction errors tend to be higher for shorter residual life times.

Figure 2 (b) reports the distribution of the obtained REs for $rc_{P_T}$. Relative prediction errors are small, with about 70% of the non-zero route-change time slots being perfectly predicted and more than 90% of them with relative errors below 50%. The model correctly predicts 38% of the zero route-change slots, achieving an overall perfect prediction for 60% of the samples.

Finally, Figure 2 (c) reports the distribution of the obtained REs for $avgRTT_P$. In this case, relative prediction errors are almost zero for about 50% of the samples, and below 30% for almost 90% of them. Given that $avgRTT_P$ values are in general very low – below 130 ms for more than 75% of the samples –, such small relative prediction errors are highly satisfactory.

## 4.4 Feature Selection

Based on the initial feature correlation results reported in Figure 1, there is a strong correlation between features of group $F_A$ and $F_B$ for the prediction of both $R_r(t)$ and $rc_{P_T}$. This could be exploited to improve prediction performance. We therefore explore now the performance of NETPerfTrace when using as input the full set of 69 input features $F_A \cup F_B \cup F_C$, and perform wrapper-based feature selection on top of this full set. Wrapper-based selection ranks features based on their prediction power for a specific prediction model. We use RF10 in this case.

Table 2 reports the top five features selected by wrapper-based selection out of the full set of features – we refer to these as 5/69 features – for the three prediction targets. We can easily spot out that the most important features are not necessarily the ones included in the subsets $F_A$, $F_B$, and $F_C$. A striking example are the top five features selected for predicting $R_r(t)$: only two out of the five features were already in the subset $F_A$. The other three are related to the number of route changes, included in $F_B$. We can see that features in $F_A$ also help estimating $rc_{P_T}$. However, as expected, features in set $F_C$ play a significant role only for the prediction of $avgRTT_P$.

To verify the prediction properties of the selected features, we computed the relative prediction errors for $R_r(t)$, $rc_{P_T}$, and $avgRTT_P$ when considering *(i)* the features on each independent set (i.e, $F_A$,

| Top 5 features | Residual life time | # route changes in time slots | average RTT |
|---|---|---|---|
| #1 feature | # route changes in current time slot | head distribution # route changes | $mean(avgRTT_P)$ |
| #2 feature | route age of current route | avg # route changes in time slots | current $avgRTT_P$ |
| #3 feature | max of all $D(r_i)$, $\forall r_i \in P$ | total # route changes | current $maxRTT$ |
| #4 feature | route change binary flag | # route changes in current time slot | current $minRTT$ |
| #5 feature | current # route changes in present time slot | avg of $D(r_i)$, $\forall r_i \in P$ | route age of current route |

**Table 2: Feature selection for the three prediction targets when considering all the 69 features.**

$F_B$, and $F_C$), *(ii)* the full set of 69 features, and *(iii)* the 5/69 features reported in Table 2.

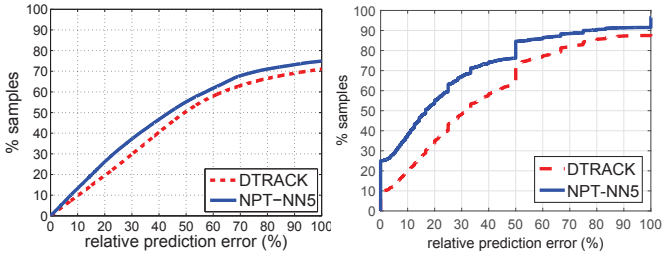
The performance increase for the prediction of $R_r(t)$ with respect to the one achieved with $F_A$ features is impressive: simply by using the 5/69 features, we observe a major reduction in the relative prediction errors. Indeed, the relative prediction errors are almost zero for about 30% of the samples with 5/69 features (versus 20% with $F_A$), and below 100% for almost 90% of the samples (about 70% with $F_A$). Using the full set of 69 features has no significant changes in the relative prediction errors with respect to the $F_A$ set.

Regarding the estimation of $rc_{P_T}$, the 5/69 features do not provide any relevant improvement with respect to $F_B$ features. However, in this case, there is a significant improvement when considering the full set of 69 features. The overall perfect prediction performance increases from 60% to more than 80%, and the distribution of relative prediction errors shows an important decrease. Still, for the sake of reducing the model complexity and the number of input features, the final release of NETPerfTrace uses the 5/69 features as input.

Finally, and as expected, there are no significant changes in the prediction performance of $avgRTT_P$ when using either the 5/69 or the full set of features. As a general conclusion of the feature selection analysis, the final implementation of NETPerfTrace uses the 5/69 features reported in Table 2 as input for the prediction of the three corresponding targets.

## 5 NETPERFTRACE VERSUS DTRACK

We now compare the performance of NETPerfTrace with the state of the art, using RF10 as underlying model and the previously selected 5/69 features as input. In particular, we compare NETPerfTrace to DTRACK [3, 4]. DTRACK predicts only path dynamics and not path performance, as its focus is on the prediction of $R_r(t)$ and $rc_{P_T}$. The system uses a Nearest Neighbors (NN)-based model as underlying prediction model, and takes as input the following four features: *(i)* route age of route $r$, *(ii)* prevalence of route $r$ in the current time slot $T$ (i.e., proportion of time $r$ is active), *(iii)* number of previous occurrences of route $r$ in $T$ for path $P$, and *(iv)* the total number of route changes in $T$ for path $P$, i.e., $rc_{P_T}$. In [3, 4], Cunha et al. also evaluate the usage of another prediction model for DTRACK, called RuleFit. However, the model was finally only used for feature selection, as its computational complexity and running time make it inapplicable in an operational deployment. The RF10 model used by NETPerfTrace is extremely fast [9], even faster than the ones tested for DTRACK. Cunha et al. named DTRACK's underlying algorithm NN4 (detailed in [3, 4, 9]), as it works on top of the four aforementioned features.




(a) NPT-NN5 vs. DTRACK: $R_r(t)$. (b) NPT-NN5 vs. DTRACK: $rc_{P_T}$.

**Figure 3: NETPerfTrace-NN5 vs. DTRACK.**

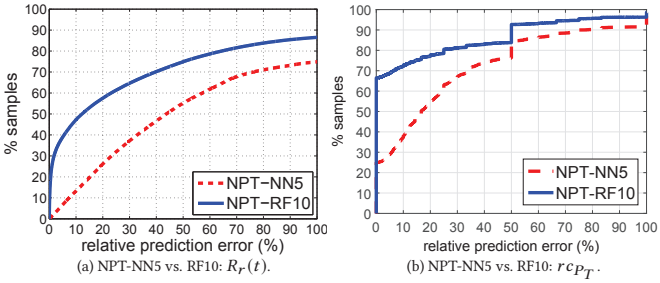
The comparison of NETPerfTrace and DTRACK is performed along three distinct dimensions: features, models, and systems. Firstly, we compare the input features used by both systems, using a NN$X$ model ($X = 4$ for DTRACK and $X = 5$ for NETPerfTrace) and a RF10 model. Secondly, we compare the performance of the underlying prediction models, by using NETPerfTrace input features and the two different prediction models – NN5 and RF10. Finally, we directly compare NETPerfTrace and DTRACK systems on the dataset presented in Section 4.1, using their default configurations (i.e., models and input features).

### 5.1 NETPerfTrace vs. DTRACK: Features

Figure 3 compares the performance of NETPerfTrace and DTRACK using their corresponding input features and NN$X$ as underlying prediction model. Figure 3 (a) reveals only a slight reduction on the relative prediction errors for $R_r(t)$ when using NN$X$ with NETPerfTrace top 5/69 input features (NPT-NN5) as compared to DTRACK features. Figure 3 (b) shows that the performance improvement is much more relevant when considering the prediction of $rc_{P_T}$. NPT-NN5 correctly predicts more than 25% of the non-zero-change time slots, while DTRACK does it for only 10%. The overall perfect prediction rate for NPT-NN5 rounds 47%, whereas it reduces to only 2% for DTRACK. Repeating the same evaluations by using RF10 as underlying prediction model shows better results for both input feature sets, but without relevant comparative difference. As a first conclusion, the 5/69 features used by NETPerfTrace provide in general much better results than those used by DTRACK, for both NN$X$ and RF10.

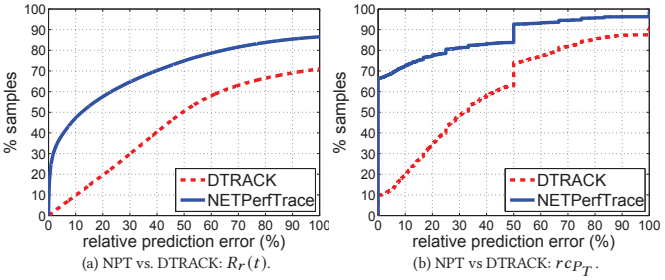
### 5.2 NN5 vs. RF10 with NETPerfTrace

We now compare the prediction power of the two underlying models used by NETPerfTrace and DTRACK, using as input the 5/69 features used by NETPerfTrace by default. Figure 4 shows a significant performance improvement when using the RF10 model




(a) NPT-NN5 vs. RF10: $R_r(t)$. (b) NPT-NN5 vs. RF10: $rc_{P_T}$.

**Figure 4: NETPerfTrace using NN5 vs. RF10.**




(a) NPT vs. DTRACK: $R_r(t)$. (b) NPT vs DTRACK: $rc_{P_T}$.

**Figure 5: NETPerfTrace vs. DTRACK.**

as compared to DTRACK's NN5 model. For example, Figure 4 (a) shows that about 30% of the relative prediction errors are close to 0% when using RF10, whereas almost no zero relative prediction errors are observed for NN5. Figure 4 (b) shows that for nearly 70% of the samples, RF10 predicts the correct number of non-zero route changes, which drops to only 25% for NN5. As a second conclusion, the prediction model used by NETPerfTrace clearly outperforms the one used by DTRACK.

### 5.3 NETPerfTrace vs. DTRACK: Wrap-up

To conclude, we now focus on the performance of both NETPerfTrace and DTRACK systems using their default configurations in terms of model and input features. Figure 5 shows that NETPerfTrace largely outperforms DTRACK for predicting path dynamics. According to Figure 5 (a), NETPerfTrace can predict $R_r(t)$ with relative errors below 10% for about 50% of the samples, whereas DTRACK only does so for 10% of the samples. In addition, almost 30% of the predictions with NETPerfTrace yield a relative error close to 0%, whereas almost no zero relative prediction errors are observed for DTRACK. In terms of daily number of route changes, Figure 5 (b) shows that NETPerfTrace correctly predicts almost 70% of the non-zero route changes, whereas DTRACK correctly forecasts only 10% of the changes. Overall, NETPerfTrace predicts the correct number of route changes for about 65% of the samples (including the zero route changes), whereas DTRACK correctly does so for only 8% of the samples.

As a general conclusion, presented results evidence that NETPerfTrace largely outperforms DTRACK when forecasting both $R_r(t)$ and $rc_{P_T}$, by using only one additional feature to tackle both prediction problems. On the one hand, this is explained by the better prediction power of the selected features. Note that we have selected specific feature sets for the prediction of $R_r(t)$ and $rc_{P_T}$, respectively, whereas DTRACK uses the same set of features for predicting both targets. On the other hand, NETPerfTrace relies on a much more powerful prediction model than DTRACK, which greatly contributes to the overall higher accuracy of the system.

## 6 CONCLUDING REMARKS

We have addressed the problem of predicting Internet path changes and path performance using `traceroute` measurements and machine learning models. We have introduced and evaluated NETPerfTrace, an Internet Path Tracking system allowing to forecast the remaining life time of a path before it actually changes, the daily number of path changes in the next day, and the average RTT of the next `traceroute` measurement, with relatively high accuracy. By carefully engineering the model behind NETPerfTrace and input features, we have shown that NETPerfTrace highly outperforms DTRACK, a previous system with the same prediction targets. Finally, we have released NETPerfTrace as open software to the community.

## REFERENCES


[1] I. Cunha, P. Marchetta, M. Calder, Y. Chiu, B. Schlinker, B. Machado, A. Pescapè, V. Giotsas, H. Madhyastha, and E. Katz-Bassett. 2016. Sibyl: A Practical Internet Route Oracle. In *Proceedings of the 13th Usenix Conference on Networked Systems Design and Implementation (NSDI'16)*. USENIX Association, Berkeley, CA, USA, 325–344.

[2] I. Cunha, R. Teixeira, and C. Diot. 2011. Measuring and Characterizing End-to-end Route Dynamics in the Presence of Load Balancing. In *Proceedings of the 12th International Conference on Passive and Active Measurement (PAM'11)*. Springer-Verlag, Berlin, Heidelberg, 235–244.

[3] I. Cunha, R. Teixeira, D. Veitch, and C. Diot. 2011. Predicting and Tracking Internet Path Changes. In *Proceedings of the ACM SIGCOMM 2011 Conference (SIGCOMM '11)*. ACM, New York, NY, USA, 122–133.

[4] I. Cunha, R. Teixeira, D. Veitch, and C. Diot. 2014. DTRACK: A System to Predict and Track Internet Path Changes. *IEEE/ACM Trans. Netw.* 22, 4 (Aug. 2014), 1025–1038.

[5] U. Javed, I. Cunha, D. Choffnes, E. Katz-Bassett, T. Anderson, and A. Krishnamurthy. 2013. PoiRoot: Investigating the Root Cause of Interdomain Path Changes. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM (SIGCOMM '13)*. ACM, New York, NY, USA, 183–194.

[6] E. Katz-Bassett, H. Madhyastha, V. Adhikari, C. Scott, J. Sherry, P. Van Wesep, T. Anderson, and A. Krishnamurthy. 2010. Reverse Traceroute. In *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation (NSDI'10)*. USENIX Association, Berkeley, CA, USA, 15–15.

[7] H. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani. 2006. iPlane: An Information Plane for Distributed Services. In *Proceedings of the 7th Symposium on Operating Systems Design and Implementation (OSDI '06)*. USENIX Association, Berkeley, CA, USA, 367–380.

[8] V. Paxson. 1996. End-to-end Routing Behavior in the Internet. In *Proceedings of the ACM SIGCOMM 1996 Conference (SIGCOMM '96)*. ACM, New York, NY, USA.

[9] S. Wassermann, P. Casas, T. Cuvelier, and B. Donnet. 2017. *Predicting Internet Path Dynamics and Performance with Machine Learning*. Technical Report.

[10] S. Wassermann, P. Casas, and B. Donnet. 2016. Machine Learning based Prediction of Internet Path Dynamics. In *Proceedings of the CoNEXT Student Workshop 2016*.

[11] S. Wassermann, P. Casas, B. Donnet, G. Leduc, and M. Mellia. 2016. On the Analysis of Internet Paths with DisNETPerf, a Distributed Paths Performance Analyzer. In *2016 IEEE 41st Conference on Local Computer Networks Workshops (LCN Workshops)*. 72–79.

[12] Y. Zhu, B. Helsley, J. Rexford, A. Siganporia, and S. Srinivasan. 2012. LatLong: Diagnosing Wide-Area Latency Changes for CDNs. *IEEE Transactions on Network and Service Management* 9, 3 (September 2012), 333–345.