Detecting General Network Neutrality Violations with Causal Inference

Mukarram Bin Tariq, Murtaza Motiwala, Nick Feamster, Mostafa Ammar
{mtariq,murtaza,feamster}@cc.gatech.edu

ABSTRACT

We present NANO, a system that identifies performance degradations that result from network neutrality violation by an Internet service provider (ISP), such as, differential treatment of specific classes of applications, users, or destinations by the ISP. Existing systems for detecting differential treatment are typically specific to an application or to a particular differentiation mechanism. Because ISPs can change differentiation policies and mechanisms, users need a method to detect differentiation, regardless of the applications that might be subject to differentiation and to the mechanisms used to achieve it. Such a scheme would make detection both robust and difficult to evade. To distinguish differentiation from other causes of degradation (*e.g.*, overload, misconfiguration, failure), NANO uses a statistical method to establish causal relationship between an ISP and observed service performance. NANO agents deployed at participating clients across the Internet collect performance data for selected services and report this information to centralized servers, which analyze the measurements to establish causal relationships between an ISP's policy and performance degradations. We have implemented NANO and deployed clients in a controlled environment on Emulab. We run a combination of controlled (Emulab) and wide-area (PlanetLab) experiments to demonstrate NANO's ability to determine the extent and criteria for differentiation, for a range of current and potential ISP policies on both BitTorrent and HTTP traffic.

1. Introduction

An intense debate has been raging over *network neutrality*, which states that end users must control the content and applications that they use on the Internet, and that the ISPs remain neutral how they forward traffic, irrespective of content, application, or sender [9]. We do not take a stance in this debate; rather, our goal is to make Internet service providers policies more *transparent* to end users. We define any practice by the ISP that degrades performance or connectivity for a service as discrimination or violation of network neutrality. We aim to help users in access networks detect such practices.

Because ISP discrimination can take many forms, detecting it is often difficult. ISPs have been interfering with TCP connections for BitTorrent and other peer-to-peer applications [7]; most recently, Cox Communications said that it planned to begin throttling peer-to-peer traffic [1]. Other types of discrimination include blocking specific ports, throttling bandwidth, or shaping traffic for specific services, or enforcing traffic quotas. Existing detection mechanisms actively probe ISPs to test for specific cases of discrimination: Glasnost detects spurious TCP reset packets of BitTorrent connections [7, 10], Beverly *et al.* present a study of port-blocking [21], and NVLens detects the use of packet-forwarding prioritization by ISPs by examining the type-of-service bits in ICMP time exceeded messages [29]. These tools have several drawbacks. First, they are specific to either the application (*e.g.*, BitTorrent) or the mechanism that the ISP is using to discriminate (*e.g.*, resetting TCP connections). Second, they rely primarily on active probes, which are typically detectable, making it possible for an ISP to either block or prioritize them, thus rendering the detection tools useless. Detecting *general* performance degradations caused by an ISP ultimately requires a method that does not make assumptions about the type or method of discrimination and that relies primarily on observations of *in situ* network traffic (as opposed to exogenous, active probes). This paper presents such a method.

This paper presents the design, implementation, and evaluation of Network Access Neutrality Observatory (NANO), a system that infers the extent to which an ISP's policy causes performance degradations for a particular service. Realizing NANO is challenging because ISP discrimination can take many forms in practice. First, ISPs can discriminate against different types of applications, ranging from BitTorrent to YouTube to Skype. Second, an ISP can apply different discrimination policies (*e.g.*, certain users may receive a different class of service). Third, an ISP can apply different mechanisms to implement the discrimination policy, ranging from outright blocking of traffic to applying different scheduling priorities to different classes of traffic.

To address the diversity and continually changing nature of discrimination, NANO uses a statistical "black-box" approach: In contrast to existing methods, we make no assumptions about the mechanisms for implementing discrimination. Instead, we use statistical analysis primarily based on *in situ* service performance data to quantify the causal relationship between an ISP's policy and the observed service degradation. Establishing such a causal relationship is challenging because many *confounding factors* (or variables) that are unrelated to ISP discrimination can also affect the performance of a particular service or application. For example, a service may be slow (*e.g.*, due to overload at a particular time-of-the-day). A service might be poorly located relative to the customers of the ISP. Similarly, a service may be fundamentally unsuitable for a particular network (*e.g.*, Internet connectivity is not suitable for VoIP applications in many parts of the world).

NANO relies on participating end-system clients that collect and report service performance measurements for a service. Our goal is to make the practices of Internet service providers transparent to end users, regardless of the type of discrimination that an ISP may apply. We aim to detect discrimination by observing the effect on service performance using primarily passive, in-band methods. In this case, the detection mechanism will be more robust because, unlike active measurements, ISPs cannot prioritize, block, or otherwise modify in-band measurements. The challenge is figuring out how to gather the right set of measurements to allow users to detect discrimination, and, in particular, to disambiguate discrimination from other factors that might degrade performance. To help isolate the effects of an ISP's policies from other factors, we gather data from a large number of clients in such a way that helps isolate the effects of each of the confounding variables. We organize the gathered data around strata, so that, for each combination of confounding variables (and appropriate ranges), we can isolate the effects of the client's upstream ISP on observed performance. As NANO makes no assumption about the type of discrimination mechanism, our system is the first "future proof" system for determining whether an ISP is actively causing performance degradation.

A necessary condition for demonstrating a causal relationship between an ISP and discrimination is to show that when all the other factors are equal, a service performs poorly when accessed from an ISP compared to another ISP. We draw inspiration from statistical epidemiology: Just as epidemiologists seek to determine whether a particular drug might be responsible for the improved health of a patient, we seek to determine whether a particular ISP is the cause of performance degradation. The challenge in establishing causality is that many confounding factors may be the underlying responsible cause for the observed outcome. For example, users may experience degraded performance due to the choice of application, operating system, or computer. Our system must isolate these confounding factors to determine when an ISP causing the performance degradation.

The main challenge in designing NANO is to create an environment where all other factors are in fact equal. Creating such an environment requires (1) enumerating the confounding factors; (2) establishing a "baseline" level of performance where all factors besides the confounding variables are equal. The nature of many confounding factors makes it difficult to create an environment on the real Internet where all other factors, except for an ISP's discriminative policy and service, would be equal. Instead, to correctly infer the causal relationship, we adjust for the confounding factor by creating strata of clients that have similar values for all factors except for their access network. Our approach is based on the theory of causal inference, which is applied extensively in other fields, including epidemiology, economics, and sociology.

We have implemented NANO and conducted experiments with a working prototype. We emulate access network ISPs on Emulab, whose clients access HTTP and BitTorrent service from hundreds of PlanetLab nodes on the wide-area Internet. Some of the ISPs in our experimental setup discriminate while others remain neutral. Specifically, we have conducted three experiments with different discrimination criteria and mechanisms, and confounding variables. We demonstrate that the distribution of performance from the discriminating ISPs may look very similar to the distribution of performance from the neutral ISPs due to the confounding variables. We then show that using NANO we can correctly identify the ISPs that are discriminating, estimate the total causal effect on the performance of the services and also infer the discrimination criteria that the ISP is using.

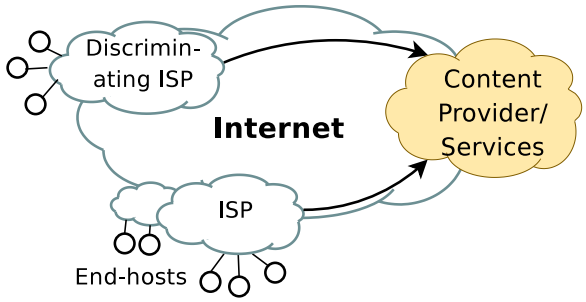


Figure 1: Problem Overview.

Section 2 presents a problem overview and motivates the use of black-box statistical methods for detecting network neutrality violations. Section 3 offers a primer on causal inference and explains confounding variables can make the cause of performance degradation ambiguous. Section 4 describes the architecture of NANO and Section 5 presents some implementation details. Section 6 describes the rationale for our experiments and evaluates the accuracy, sensitivity, and scalability of NANO. Section 7 lists various open issues with NANO. Section 8 discusses related work, and Section 9 concludes.

2. Problem Overview and Motivation

In this section, we describe an overview of the network neutrality violation problem. We also define the key terms.

Problem statement. Our goal is to detect whether a certain practice of an ISP (intentional or accidental) results in degraded performance for a service compared to other similar services or performance for the same service through other ISPs. If an ISP's policy of treating traffic differently does not result in degradation of performance, we do not consider it discrimination. We define some of these terms in the next paragraph.

Definitions. A *service* is the "atomic unit" of discrimination. An ISP may discriminate against traffic for a particular service, e.g., Web search, traffic for a particular domain, or particular type of media, such as video. Such traffic may be identifiable using the URL or the protocol. Similarly, ISPs may target specific applications, e.g., VoIP, or peer-to-peer file transfers. *Performance*, the outcome variable, is specific to the service. For example, we use server response time for HTTP requests, loss, and jitter for VoIP traffic, and average throughput for peer-to-peer traffic. *Discrimination* against a service is a function of ISP policy. The performance for a service depends on both the properties of the ISP's network,

e.g., its location, as well as the policy of treating the traffic differently. Thus, an objective evaluation of ISP discrimination must adjust for the ISP's network as a confounding factor. To differentiate an ISP's network from its discrimination policy, we use the ISP *brand* or *name* as the causal variable referring to the ISP's discrimination policy. In the rest of the paper, when we use ISP as the cause, we are referring to the ISP policy or the brand with which the policy is associated.

Why it's difficult. Detecting discrimination is challenging for several reasons. First, edges do not know what policies or mechanisms an ISP might be implementing to discriminate against traffic. Detecting ISP discrimination is a cat-and-mouse game between discriminating ISPs and the edges (*i.e.*, end-hosts and content providers). End users access content (*e.g.*, video, audio content) or use a particular service (*e.g.*, p2p file sharing, VoIP) via their ISPs, as shown in Figure 1. A discriminating ISP could intentionally reduce the performance of a certain service or throttle connectivity to a particular content in a wide variety of ways, many of which might be difficult for users to detect. Existing tools for detecting network neutrality all assume that either the mechanism for discriminating against traffic or the application being discriminated against is known in advance. Unfortunately, this is generally not the case; users from a wide range of ISPs and countries often do not even know whether an ISP might be discriminating certain subsets of traffic. Instead, these users need *general* methods for detecting discrimination that do not rely on testing for specific discrimination types.

Second, any tool that detects discrimination must positively identify the ISP—as opposed to any other possible factor—as the underlying cause of discrimination. This problem was most recently identified by an unnamed industry source, who expressed skepticism about the effectiveness of existing tools: "However, one ISP industry source, who asked not to be identified, questioned whether the tools would accurately point to the cause of broadband problems. 'Spyware or malware on computers can affect browser performance, and problems with the wider Internet can cause slowdowns, the source said."' [12] It is precisely this problem—adjusting for such external causes and confounding factors—that we seek to solve in the design of a tool for detecting network neutrality violations.

Also, implicit in our discussion so far has been the assumption that we *know* what the non-discriminated service performance is for a given ISP. However, it is not really clear how to determine what is the non-discriminated performance for a given service. We follow one approach for estimating this value and allude to other possible ways of approximating the non-discriminated service performance (or *baseline* performance) in Section 4.1.2.

NANO appears to be the first tool that can isolate such discrimination from other confounding factors, without *a priori* knowledge of an ISP's discrimination policy. At the heart of our approach is "black box" statistical testing. Unlike existing approaches to detecting discrimination, our approach is statistical, not rule-based. As opposed to trying to identify or detect certain behavior, we instead directly observe the performance of traffic and try *to infer the causes* of degraded performance when it does occur. This approach is more general; however, it also faces its own set of challenges, such as ensuring that all possible causes are enumerated, collecting a statistically significant set of data for analysis, etc. The next section offers a primer on causal inference and explains how it might be applied to network performance.

3. Background and Problem Formulation

In this section[1], and basic concepts used we give a brief overview of causal inference, how it relates to association and correlation and approaches for quantifying causal effect. We also formalize the application of causality to detecting ISP discrimination. One of the big problems with causal inference is ensuring that all possible causes are enumerated; thus, we also explain how we test for that our model captures a sufficient set of confounding variables.

3.1 Background for Causal Inference

Statistical methods offer tools for causal inference that have been used in observational and experimental studies [13, 19]. NANO draws heavily on these techniques. In this section, we review basic concepts and approaches for causal inference, and how they relate to inferring ISP discrimination. We begin with a graphical representation of causality and an example that shows correlation and confounding variables.

Graphical Representation and Example. We can express causal relationships between various variables and system performance using a graphical model. We represent the variables as nodes in a graph. An edge in this graph is directed if we can establish the direction of causality between the two nodes (if *x causes y*, then there is an edge from node x to node y), and undirected if we observe correlation but cannot determine the direction of causation (please see [24] for an algorithm for determining the causal structure). In this graph, a variable z is confounding for the relationship between the causal variable (x) and outcome variable (y) if there are paths from z to both x and y that do not go through either x or y.

To illustrate this, consider Figure 2 which shows a simplified causal model for a hypothetical service. The service performance depends on the application that the client uses and the network round-trip time. The client application has a *correlation* with the brand of the ISP, but we are not able to determine the direction of causality. Perhaps the correlation exists because the ISP advises its customers to use a particular application, or perhaps it is the other way around: that the clients using a particular application prefer a particular ISP. Similarly, we observe a correlation between the client location and the brand of the ISP, but we are not able to determine the direction of causal influence. The round-trip time is determined by the relative location of the client and the server. The server location is determined by the service provider and the brand of the ISP (such dependencies are common in content distribution networks). In this model,

[1]Parts of the background from this section are adapted from material that appeared in a workshop paper [23].

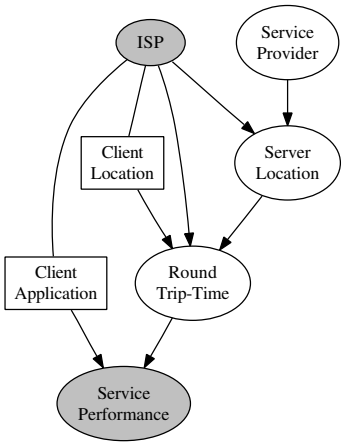


Figure 2: Graphical model for service performance. Causal variable ("ISP") and outcome variable ("Service Performance") are shown shaded. Confounding variables are shown in rectangular nodes.

client location and the client applications become *confounding variables* because these variables have potential paths to both the causal variable (ISP) and the outcome variable (service performance). We describe these concepts more formally in the following.

Causal Effect. The statement *"X causes Y"* means that if there is a change in the value of variable X, then we expect a change in value of variable Y. We refer to X as the *treatment variable* and Y as the *outcome variable*.

In the context of this paper, accessing a particular service through an ISP is our treatment variable (X), and the observed performance of a service (Y) is our outcome variable. Thus, treatment is a binary variable; $X \in \{0, 1\}$, $X = 1$ when we access the service through the ISP, and $X = 0$ when we do not (e.g., access the service through an alternative ISP). The value of outcome variable Y depends on the performance metric and the service for which we are measuring the performance.

The goal of causal inference is to estimate the effect of the treatment variable (the ISP) on the outcome variable (the service performance). Let's define a *ground-truth* value for the outcome random variable as G_X, so that G_1 is the outcome value for a client when $X = 1$, and G_0 is the outcome value when $X = 0$. We will refer to the outcome when not using the ISP ($X = 0$) as the *baseline*—we can define baseline in a number of ways, as we describe in more detail in Section 4.1.2.

We can quantify the *average causal effect* of using an ISP as the expected difference in the ground truth of service performance between using the ISP and the baseline.

$$\theta = \mathbb{E}(G_1) - \mathbb{E}(G_0) \tag{1}$$

Note that to compute the causal effect, θ, we must observe values of the outcome both under the treatment and without the treatment.

Association vs. Causal Effect. In a typical *in situ* dataset, each sample presents only the value of the outcome variable either under the treatment, or under the lack of the treatment, but not both; e.g., a dataset about users accessing a particular service through one of the two possible ISPs, ISP_a and ISP_b, will comprise data of the form where, for each client, we have performance data for either ISP_a or ISP_b, but not both. Such a dataset may thus be incomplete and therefore not sufficient to compute the causal effect, as shown in Equation 1.

Instead, we can use such a dataset to compute correlation or association. Let's define *association* as simply the measure of observed effect on the outcome variable:

$$\alpha = \mathbb{E}(Y|X = 1) - \mathbb{E}(Y|X = 0) \tag{2}$$

It is well known that association is not a sufficient metric for causal effect, and in general $\alpha \neq \theta$.

3.2 Approaches for Estimating Causal Effect

This section presents two techniques for estimating the causal effect, θ. The first, random treatment, involves an active experiment, where we randomly assign the treatment to the clients and observe the association. The second, adjusting for confounding variables, is a passive technique, where we work with only an *in situ* dataset and estimate the overall causal effect by aggregating the causal effect across several small strata.

3.2.1 *Random Treatment*

Because the ground-truth values (G_0, G_1) are not simultaneously observable, we cannot estimate the true causal effect (Eq. 1) from an *in situ* dataset alone. Fortunately, if we assign the clients to the treatment in a way that is independent of the outcome, then under *certain conditions*, association is an unbiased estimator of causal effect. This property holds because when X is independent of G_X, then $\mathbb{E}(G_X) = \mathbb{E}(G_X|X) = \mathbb{E}(Y|X)$; see [25, pp. 254–255] for a proof.

For association to converge to causal effect with random treatment, all other variables in the system that have a causal association with the outcome variable must remain the same as we change the treatment. In the case of the example above, association will converge to true causal effect under random treatment, if and only if the original ISP and the alternative ISP are both similar except for their discrimination policy.

Random treatment is difficult to emulate in the Internet for two reasons. First, it is difficult to make users switch to an arbitrary ISP, because not all ISPs may offer services in all geographical areas, the users may be contractually bound to a particular ISP, and asking users to switch ISPs is inconvenient for users. Second, if changing the ISP brand also means that the users must access the content through a radically different network which could affect the service performance, then we cannot use the mere difference of performance seen from the two ISPs as indication of interference: the association may not converge to causal effect under these conditions because the independence condition is not satisfied. This situation is called *operational confounding*: changing the treatment inadvertently or unavoidably changes a confounding variable.

3.2.2 *Adjusting for Confounding Variables*

Because it is difficult to emulate random treatment on the real Internet and control operational confounding, we need to find a way to adjust for the effects of confounding variables. NANO uses the well-known stratification technique for this purpose [13].

Confounding variables are the extraneous variables in the inference process that are correlated with both the treatment and the outcome variables. As a result, if we simply observe the association between the treatment and the outcome variables, we cannot infer causation or lack of it, because we cannot be certain whether the change is due to change in the treatment variable or a change in one or more of the confounding variables.

With stratification, all samples in a stratum are *similar* in terms of values for the confounding variables. As a result, X and G_X are independent of the confounding variables within the stratum, essentially creating conditions that resemble random treatment. Thus, the association value within the stratum converges to causal effect, and we can use association as a metric of causal effect within a strata.

Challenges. This approach presents several challenges. First, we must enumerate the confounding variables and collect sufficient data to help disambiguate the true causal effect from the confounding effects. Second, we must define the stratum boundaries in a way that satisfies the above conditions. Unfortunately, there is no automated way to enumerate all the confounding variables for a given problem; instead, we must rely on domain knowledge. Section 4 addresses these challenges.

Formulation. In the context of NANO, we have multiple ISPs and services; we wish to calculate the causal effect $\theta_{i,j}$ that estimates how much the performance of a service j, denoted by Y_j, changes when it is accessed through ISP i, versus when it is not accessed through ISP i. Let Z denote the set of confounding variables, and s a stratum as described above. The causal effect $\theta_{i,j}$ is formulated as:

$$\theta_{i,j}(s;x) = \mathbb{E}(Y_j|X_i = x, Z \in \mathbb{B}(s)) \quad (3)$$

$$\theta_{i,j}(s) = \theta_{i,j}(s;1) - \theta_{i,j}(s;0) \quad (4)$$

$$\theta_{i,j} = \sum_s p_{i,j}(s)\theta_{i,j}(s) \quad (5)$$

$\mathbb{B}(s)$ represents the range of values of confounding variables in the stratum s. $\theta_{i,j}(s)$ represents the causal effect within the stratum s and $p_{i,j}(s)$ refers to the probability of the service j under stratum s with ISP i. A key aspect is the term $\theta_{i,j}(s;0)$ in Equation 4: it represents the *baseline* service performance, or the service performance when the ISP is *not* used; we define this concept in more detail in Section 4.1.2. Note that the units for causal effect are same as for service performance, so we can apply simple thresholds to detect discrimination.

3.3 Sufficiency of Confounding Variables

Although there is no simple or automatic way to enumerate all the confounding variables for a problem, we can test whether a given list is sufficient in the realm of a given dataset. To do so, we use the following heuristic. We predict the value of the outcome variable using a non-parametric regression function, $f()$, of the treatment variable, X, and the confounding variables, Z, as $\hat{y} = f(X;Z)$. We then compare the predicted value with the value of outcome variable observed in the given dataset, y, using relative error, $|y - \hat{y}|/y$.

If X and Z are sufficient to define an unbiased predictor for the outcome Y, then the prediction error should be small, with an expected value of zero, and statistically independent of the outcome variable. Recall that the confounding variables correlate with both the causal and the outcome variable, therefore the distribution of the unobserved confounding variable should vary over the range of the outcome variable. Prediction error with an estimator that does not account for this unobserved confounding variable, may not be independent of the outcome variable. We can test the independence between prediction error and outcome using a simple correlation test. We demonstrate this heuristic in Section 6.3.1.

4. Applying Causality to Network Traffic

This section explains how NANO performs causal inference, enumerates the confounding variables required for this inference, and describes the system architecture for collecting and processing the relevant data.

4.1 Establishing Causal Effect

Estimating causal effect for a service degradation involves three steps. First, we must stratify the service performance data reported from the end-hosts to create a number of strata. Next, we estimate the extent of possible ISP effect within each stratum and across the board. Finally, we try to infer the criteria that the ISP is using for discrimination. The remainder of this section describes these three steps.

4.1.1 *Stratifying the data*

To stratify the data, NANO creates bins (i.e., ranges of values) along the dimensions of each of the confounding variables, such that the value of the confounding variable within the bin is (almost) constant. The bin size depends on the nature of the confounding variable. As a general rule, we create strata such that there is a bin for every unique value of categorical variables; for the continuous variables, the bins are sufficiently small, that the variable can be assumed to have essentially a constant value within the stratum. For example, for a confounding variable representing the client browser, all the clients using a particular version and make of the browser are in one stratum. Similarly, we create one hour strata along the time-of-the-day variable.

We use simple correlation to test whether the treatment variable and the outcome variable are independent of the confounding variable within a stratum. We combine adjacent strata if the distribution of the outcome variable conditioned on the treatment variable is identical in each of the stratum; this reduces the total number of strata and the number of samples needed.

4.1.2 *Establishing the baseline and causality*

One challenge with respect to Equation 4 is the term $\theta_{i,j}(s;0)$, which represents the *baseline* service performance, or the service performance when the ISP is *not* used. This aspect presents a serious challenge: What does it mean to *not* use ISP i to access service j? In other words, what is the "baseline" performance that a user could expect when using a particular service? For example, "not using" an ISP might mean using another ISP, k, but if ISP k is also discriminating against service j, then $\theta_{k,j}(s;1)$ will not have the (neutral) ground-truth baseline value. To address this problem, NANO takes $\theta_{i,j}(s;0)$ as the average service performance when not using ISP i, calculated as: $\sum_{k \neq i} \theta_{k,j}(s;1)/(n_s - 1)$, where $n_s > 2$ is the number of ISPs for which we have clients in stratum s.

An important implication of defining the baseline in this way is that NANO is essentially comparing the performance of a service through a particular ISP against the average performance achieved through other ISPs, while adjusting for the confounding effects. If all or most of the ISPs across which NANO obtains measurements are discriminating against a service, it is not possible to detect such discrimination using this definition; in this case, discrimination becomes the *norm*. In such cases, we might consider using other definitions of discrimination, such as the comparing against the best performance instead of the average, or using a performance model of the service obtained from laboratory experiments or mathematical analysis as the baseline.

Another issue with our definition is that it only measures discrimination based on deviation from *average* performance; this definition is simple, but it may be subject to manipulation, whereby an ISP can manipulate the performance of individual flows without affecting the average. We discuss possible defenses against this type of manipulation in Section 7.

4.1.3 *Inferring the discrimination criteria*

NANO infers the discrimination criteria that an ISP uses by using simple decision-tree based classification methods. For each stratum and service where NANO detects discrimination, NANO assigns a negative label, and for each stratum and service where it does not detect discrimination, it assigns a positive label. NANO then uses the values of the confounding variables and the service identifier as the feature set and uses the discrimination label as the target variable, and uses a decision-tree algorithm to train the classifier.

The rules that the decision tree generates indicate the discrimination criteria that the ISP uses, because the rules indicate the boundaries of maximum information distance between discrimination and the lack of it. We note that the causal inference performed by NANO makes no assumptions about discrimination criteria used by the ISP; the determination of the actual sources of discrimination comes as almost as a side effect of stratification, after the fact.

4.2 Enumerating the Confounding Variables

Confounding variables are the extraneous factors in inferring whether an ISP's policy is discriminating against a service; these variables correlate, either positively or negatively, with both the ISP brands and service performance. Because there is no automated way to enumerate these variables for particular problem, we must rely on domain knowledge. In this section, we describe three categories of confounding variables and explain how they correlate with both the ISP brands and the service performance. In Section 5, we describe the specific variables that we collect to adjust for these confounding variables.

Client-based. Client-side applications, as well as system and network setup, can confound the inference. The particular application that a client uses for accessing a service might affect the performance. For example, in the case of HTTP services, certain Web sites may be optimized for a particular Web browser and perform poorly for others. Similarly, certain Web browsers may be inherently different; for example, at the time of this writing, Opera, Firefox, and Internet Explorer use different number of simultaneous TCP connections, and only Opera uses HTTP pipelining by default. For peer-to-peer traffic, various client software may experience different performance. Similarly, the operating system and the configuration of the client's computer and local network, as well as a client's service contract, can affect the performance that the client perceives for a service.

We believe that the above variables also correlate with ISP brand, primarily because the ISP may serve particular communities or localities. As an example, we expect that Microsoft's Windows operating system may be more popular among home users, while Unix variants may be more common in academic environments. Similarly, certain browsers may be more popular among certain demographics and localities than other.

Network-based. Various properties of the Internet path, such as location of the client or the ISP relative to the location of the servers on the Internet, can cause performance degradation for a service; such degradation is not discrimination. Similarly, a path segment to a particular service provider might not be sufficiently provisioned, which could degrade service. If we wish to not treat these effects as discrimination, we should adjust for the path properties.

Time-based. Service performance varies widely with time-of-day due to changes in utilization. Further, the utilization may affect both the ISPs and the service providers, thus confounding the inference.

5. Design and Implementation

This section describes the implementation of NANO, which has two parts: NANO-Agents reside on end hosts, collect data (typically high-level or aggregated performance statistics) for traffic from that host to various destinations, and send aggregate traffic statistics to the centralized NANO-Server. The NANO-Server collects these statistics and performs the inference described in Section 3 to quantify the ISP's effect on performance. The primary source of data for NANO are client-side agents installed on computers of voluntarily participating clients (NANO-Agents). Each agent continuously monitors and reports the data to the NANO servers. The rest of this section describes these

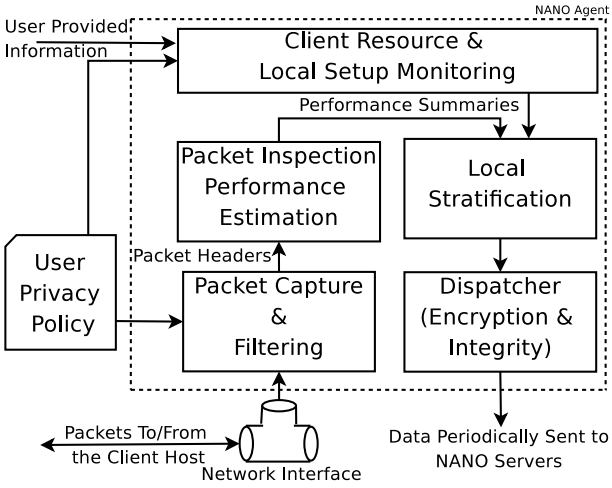


Figure 3: NANO-Agent Architecture.

two components in detail. We also briefly describe our plans to make the code for NANO-Agents availableto allow us to reach a wide audience for gathering data and statistics across a number of heterogeneous clients.

5.1 NANO-Agents

The most convenient place to collect network performance data is using monitoring agents at clients. We have developed NANO-Agent as a packet-level sniffer that can access find-grained information from the client machines including the various system resource utilization and client machine setup information.

Figure 5.1 shows the architecture for the NANO agent. The NANO-Agent collects three types of features for the confounding factors, corresponding to the three classes of confounding variables (Section 4.2). First, the NANO-Agent collects features that help identify the client setup, including the operating system, basic system configuration and resource utilization on the client machine. Second, NANO-Agents determine the topological location of the client and their ISPs. Also, all the data collected is time-stamped to allow adjustment for time-of-day factor. NANO-Agents collect flow statistics and information about the application responsible for the flow to stratify the collected data on the application or protocol type.

Data collection The criteria for data collection has two parts. First, the feature should quantify the treatment variable, the outcome variable, or the values of the confounding factors. Second, the data should be unbiased. The first criterion helps us determine a set of features for which to collect data; this list is explained below. We can collect many of these features through active or passive monitoring. The second criterion, however, suggests that we must take care that the measurements are not biased. As we discussed in Section 1, ISPs may have the incentive to interfere with identifiable active measurements to deter inference of discrimination or improve their rankings. Similarly, we believe that while we could use the data directly from service providers as the "baseline" service performance, such information could be biased in the favor of the service provider. Therefore, to the extent possible, NANO relies on passive measurements to determine the values of the features.

The NANO-Agent analyzes the network, transport and application protocol (*e.g.*, HTTP and RTP) headers to identify the service and assess performance. For the experiments that we describe in Section 6, we focus solely on features that can be extracted from the TCP/IP headers of the packets and associated timing information. In particular, we try to estimate the throughput and latency that the packets experience for a TCP flow. To estimate the throughput agent continuously measures the bytes uploaded and downloaded in a specified (configurable interval). To estimate the latency on a TCP flow, the agent measures the latency between the SYN and SYN/ACK packets for the flows that originate at the client, and the latency between the SYN/ACK and the first subsequent ACK, for the incoming connections that the client receives. In addition, the agent keeps track of connection duration and events such as losses, timeouts (by tracking the TCP duplicate acknowledgements) or unexpected connection terminations, such as, with a TCP reset flag. The agent also infers the application associated with the active flows by gleaning the information in the `proc` file system. In addition, the agent also continuously monitors the average CPU and memory utilization on the client host.

In addition to runtime statistics, the agent also collects the information about the client setup. This includes the client host specification, i.e., the platform, CPU and memory specification, and the active operating system on the host. We rely on the user to provide the agent with information that we cannot infer dynamically. This includes the type of network interface (wired or wireless), the type of contract the client has with the ISP, the user's location (city and country). In the future, we plan to use an IP-Geo Location database in lieu of client provided information.

Protecting user privacy To mitigate privacy concerns, the agent performs local stratification to lower the granularity of information that the client sends to the server. For this, we mask the least significant 8 bits of the source and destination IP addresses of the flows for which we report the data[2], round-off the round-trip time measurements to a (configurable) nearby values. Because this stratification and impact the inference accuracy at the server, we are extending our implementation to allow these parameters to be updated dynamically based on feedback from the server; for now, the agent obtains the stratification parameters from local configuration.

Implementation We have implemented the agent using C++. The agent promiscuously captures the packets that flow from the client host's network interface using the *pcap* library. In order to address the privacy concerns, we allow the users to specify a set of destinations for which the NANO agents monitor performance. Based on this user-preference, the agent filters the packets and analyzes the filtered packets to estimate service performance.

[2]We disable this feature for the experiments described in Section 6 because all of our testbed nodes shared the same subnet.

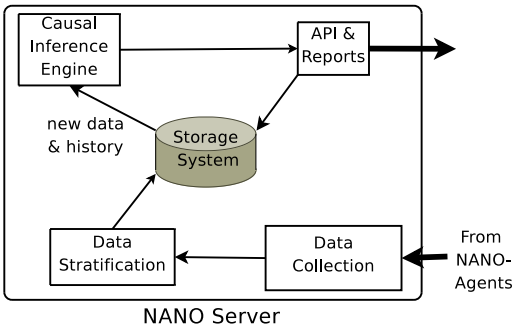


Figure 4: NANO Server Architecture.

NANO-Agent uses protocol buffers [11] to maintain the information that it collects. Protocol buffers offer high speed and compact serialization that allows us to minimize the computational and communication overhead for running the agent at the client. The agent periodically serializes the data that it has collected and sends it to a NANO data collection server. We discuss the overhead further in Section 6. At this point, the NANO agents connect to the server an unsecure channel, however, we are working on extending our implementation to use a secure channel to prevent eavesdropping and ensure integrity, as well as to use a Mixing network, such as TOR, to further obfuscate the client identity from the NANO servers.

5.2 NANO-Server

Figure 4 shows the architecture of the NANO-Server, which receives periodic information from NANO-Agents running on the participating end-hosts. The server receives the data from the client agents and stores it in a database. The server simultaneously stratifies and indexes exes the data using unique identifiers for each strata. This indexing allows for quick retrieval of data when needed. In our present implementation, the stratification process is manually guided: we configure the server with the information about strata boundaries for each feature. Our criteria is that for features with discrete values, such as location, or application identifier, we create a stratum per value. For continuous variables, such as file size, we quantize the value into small bins, and use the quantized value as a stratum.

The server periodically tries to predict the performance for the services that it is tracking for each ISP using the method described in Section 3.3 and computes the prediction error. The NANO-Server uses kernel regression [27] with a radial basis kernel in a piece-wise manner as a regression function. The server uses the error distribution to asses whether it has sufficient data and features to predict performance of a particular service. The server also uses Equation 5 to estimate the extent of causal effect of each ISP for each service. The number of strata that the server deals with can grow exponentially with features. Fortunately, because the computation of effect within each stratum is independent of other strata, the computation is easily parallelizable.

Implementation The server is implemented using a combination of C++ to implement the data collection and demarshalling and using a Python and MySQL backend for analysis and causal inference. Our present implementation can compute causal effect over 20,000 tiles in about one minute using two threads on a dual 3.2 GHz processor Intel Pentium 4 machine with 4 GB of memory. We realize that in a real-world deployment, the number of strata can easily approach a million; for this, we plan to port the NANO-server to a Map-Reduce [5]-based implementation.

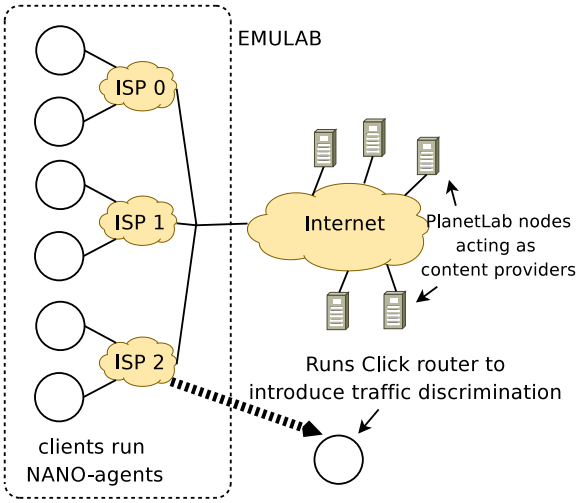


Figure 5: Experimental setup on Emulab. Each ISP is represented as an Emulab node with Click router running to perform discrimination. NANO-agents are run on the client nodes and report information to our server.

6. Evaluation

In this section, we present results from our evaluation of NANO We present four main results:

- NANO can determine whether it has a sufficient set of confounding variables; that is, NANO's heuristic for testing sufficiency of confounders has low relative error. (Section 6.3.1)
- NANO can detect when an ISP is discriminating. We tested NANO's detection algorithms with three different types of discrimination and in the presence of various network and client-side confounding variables. (Section 6.3.2)
- NANO can determine the discrimination criteria used by the ISP with the help of a simple decision-tree based classifier. (Section 6.3.3)
- NANO has low bandwidth overhead, in terms of transferring summaries from the NANO-Agents to the NANO-Server. (Section 6.3.4)

6.1 Experimental Setup

We use Emulab [26] to create a setup where we can control some of the confounding variables on the client side and use various discrimination criteria that might be implemented by an ISP. We use PlanetLab [2] nodes to act as providers of content (or seeds in the case of the BitTorrent setup), which ensures that our experiments use real Internet paths. In the rest of this section, we describe the three aspects of our

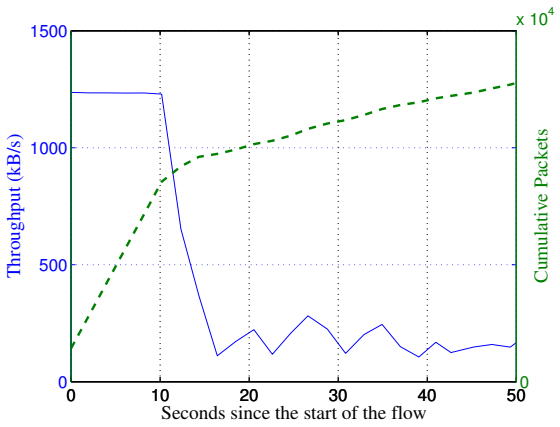


Figure 6: Discrimination using Click

setup: topology, implementing discrimination, and running services.

6.1.1 *Topology and testbed*

We use Emulab to test the discrimination detection abilities of NANO. We use Emulab to create a set of ISPs, each with its set of clients that connect to the ISP using links of configurable characteristics. The ISP provides connectivity to the Internet to its clients. Figure 5, shows this arrangement. We run the Click router software [15] on the routers of the ISPs that perform traffic discrimination. The clients can be configured to be of different physical configurations and run different operating systems on them. In particular, we create five ISPs: two of these discriminate, and we call them discriminating ISPs, ISP D_1 and ISP D_2. The remaining three ISPs use best-effort service for all the packets on their routers; we refer to these as the neutral ISPs and ISP N_1, ISP N_2 and ISP N_3. The exact manner and nature of discrimination varies with the experiment and is described in 6.2.

A potential problem with the experiment setup is that if some ISP in the wide area (outside our Emulab environment) is discriminating traffic between the Emulab clients and the PlanetLab nodes, then NANO will not be able to detect that since NANO only has data from the NANO agents which all use the ISP outside of Emulab. However, even if an external ISP is discriminating traffic from Emulab, it should not affect our results, as the traffic would be discriminated for all Emulab clients and the discriminated performance would be considered as the baseline for our evaluation.

6.1.2 *Emulating ISP discrimination*

Discrimination is performed by running Click on the Emulab node that acts as the ISP router to connect the ISP clients to the Internet (see Figure 5). We pass the client traffic through the Click router running as a kernel module on the router node. We used a combination of Click elements to perform various forms of discrimination including probabilistically dropping packets on all flows, or flows which exceed a certain length, dropping of TCP acknowledgements, dropping packets for a particular service or destination, and sending TCP RST packets back to the client (similar to the practice by Comcast).

We used the available Click router elements like *IPClassifier* to classify packets based on the various IP and TCP fields, *RandomSample* for dropping packets and, *AggregateIPFlows* and a modified version of *AveragePktCounter* to implement classifying flows which exceed a certain length. Running Click router as a kernel module was sufficient to ensure that the Emulab ISP node could sustain a reasonable amount of traffic (maximum of 20Mbps) during the experiments.

Figure 6 shows an example of discrimination using Click router. In this case we have configured the router to probabilistically drop TCP packets of flows which have exceeded 13000 packets. For the flow shown in Figure 6, this happens at around 10 seconds after the start of the flow: we see a clear drop in the throughput (calculated as bytes transferred per ten seconds) as well as the rate of cumulation of packets for the flow.

6.1.3 *Clients and services*

We use a set of PlanetLab nodes, which are geographically distributed and have a wide range of RTTs from the client nodes in Emulab. We have configured two kinds of services on the PlanetLab nodes. First, we configure two Web servers on each of these PlanetLab nodes to represent two different Web services. Second, we have configured the PlanetLab nodes to act as BitTorrent clients.

The clients in the Emulab environment access these services through their emulated ISPs and the service performance can be impacted if the ISP discriminates. Each client also runs an instance of NANO-Agent which periodically reports the performance data to a central NANO-server running elsewhere on the Internet.

6.2 Experiments

We have performed three experiments to evaluate the efficacy of the NANOsystem. Here we describe these experiments. Table 1 summaries the configurations for each of these experiments.

6.2.1 *Experiment 1 (Simple Discrimination).*

In this experiment we emulate a scenario where some ISPs discriminate the HTTP traffic, however, the location of the HTTP servers is a confounding variable, which makes it difficult to identify the ISPs that are discriminating. We use the TCP throughput achieved during the HTTP fetch as the performance variable.

To create this confounding effect, we divide the PlanetLab nodes on which the HTTP servers run into two groups. The *near nodes* are the ones which have a less than 60ms RTT from the Emulab testbed site. The *far nodes* are the ones which have a RTT between 60ms and 120ms from the Emulab testbed site. We have 146 near nodes and 167 far nodes, and 313 servers in all.

Clients in each ISP repeatedly fetch content, each time from a randomly chosen PlanetLab server. We configure the clients in each ISP to use different probabilities for picking a random HTTP server from the set of near or far nodes.

ISPs N_1, N_2, and N_3 are neutral. ISP D_1 and ISP D_2 discriminate in each experiment.

Simple Discrimination. ISPs D_1 and D_2 discriminate the HTTP traffic for all their clients. ISPs D_1 and D_2 drop 0.1% and 0.3% of the packets respectively. Location of the HTTP server is a confounding variable. ISPs N_1, N_2, N_3, D_1, and D_2 access the content from the *near* PlanetLab servers with probabilities, 0.4, 0.1, 0.7, 0.6, and 0.9, respectively, and access the *far* PlanetLab servers with the remaining probability.

Long Flow Discrimination. We have two HTTP services, S_1 and S_2. ISPs D_1 discriminates S_1 and D_2 discriminate the HTTP traffic for S_2 for all their clients *if* the flow from S_1 or S_2 exceeds certain limits. ISPs D_1 and D_2 drop 0.1% and 0.3% of the packets for flows exceeding 10000 and 13000 packets respectively. Location of the servers remains a confounder as in Experiment 1, with same probabilities for the *near* HTTP servers. All HTTP servers provide both S_1 and S_2, albeit on different ports.

BitTorrent Discrimination. ISP D_2 discriminates the BitTorrent traffic for all its clients if the BitTorrent peer is not in certain subset of PlanetLab nodes. ISP D_2 discriminates by dropping 0.3% of the packets of the flows that are established with the non-preferred peers.

Table 1: Summary of Experiments

The exact probabilities are described in Table 1. Because the location of the server also has an impact on the performance and because the distribution of location of servers is different across the ISPs, the location of the sever becomes a confounding variable. Such confounding effects can be common in content distribution environments where the customers of a neutral ISP may experience poorer service because of less ideal servers, where as the aggregate experience for the customers in a discriminating ISP might be better.

In this experiment, the ISPs discriminate by randomly dropping the packets on the TCP flows. Table 1 summarizes the configuration for dropping the packets.

This experiment demonstrates NANO's capability to deal with confounding effects and correctly identify the discriminating ISP.

6.2.2 *Experiment 2 (Long Flow Discrimination)*

This experiment is similar to Experiment 1, except that in this experiment we consider two HTTP services, S_1 and S_2 and two different ISPs discriminate against each under certain conditions. One of the discriminating ISP discriminates traffic between its clients and S_1, if the flow exceeds a certain threshold. The other discriminating ISP discriminates traffic with S_2 if the flow exceeds a certain other threshold. The exact parameters are described in Table 1.

This experiment demonstrates NANO's capability to simultaneously deal with multiple services and also inferring the criteria for discrimination for each.

6.2.3 *Experiment 3 (BitTorent Discrimination)*

In this experiment we consider BitTorrent as a service. The ISPs discriminating ISPs remain neutral if the peer for its client is in one of the *preferred* locations, otherwise, if the peer is in a non-preferred location, then the discriminating ISP discriminates by dropping the packet for the flow. Such a scenario is perceivable because an ISP may not mind if the BitTorrent peers are accessible through a peering link, but it may discriminate if the peer is accessible through a transit link for which the ISP has to pay more. This experiment is similar to Experiment 2, except that the discrimination criteria is different.

We ran each experiment for approximately 6 hours and collected the data at the NANO-server. The results in the following section are based on a run in the week of January 24, 2009.

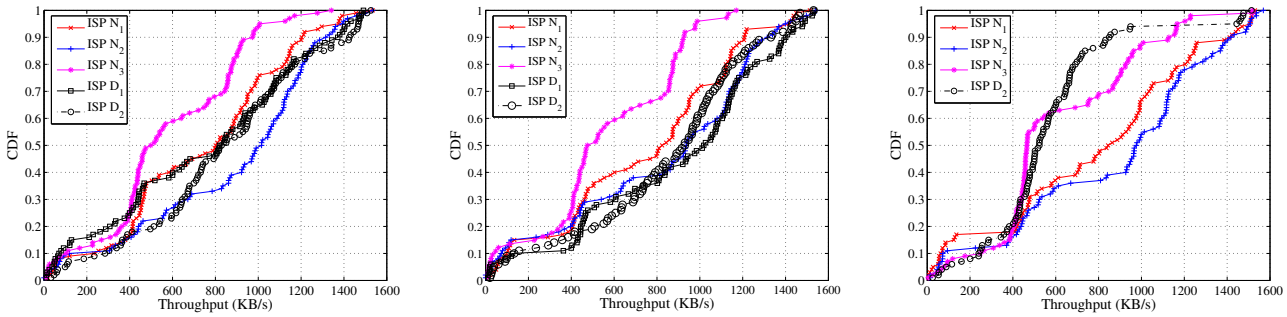
6.3 Results

In this section we present the results of applying NANO in the experiments described in Section 6.2. In Figure 7, we present the aggregate distribution of performance from all the ISPs to show that it is not easy to tell the discriminating ISPs apart. In Section 6.3.1 we show that the variables that we collect are sufficient to predict the performance in each of the three experiments. In Section 6.3.2, we present the causal effect values that NANO computes for each ISP in each of the experiments. Finally, in Section 6.3.3 we demonstrate that NANO can also infer the criteria that the ISP is using.

Figure 7 shows the distribution of performance for the clients of each ISP in all three experiments. For the first two experiments, we compute the average throughput over the life of the flow and use that as a metric. It is difficult to identify the discriminating ISP for any of the experiments in a straight forward manner. In the first two experiments (Figures 7(a,b)), the overall distribution of performance for the discriminating ISPs is similar or better than the distribution of performance in the neutral ISPs because the clients in the discriminating ISPs access the *near* servers with higher probability. Because geographically closer servers are more likely to provide higher throughput, we observe a larger fraction of higher throughput sessions for the discriminating ISPs. Similarly, the throughput for most BitTorrent clients in D_2 is very similar to the throughput in the other ISPs when indeed the ISP is discriminating against a subset of destinations.

6.3.1 *Determining sufficiency of confounders*

In Section 3.3 we presented a heuristic for determining whether the variables we collect are sufficient. We argued that if we have enough variables, then we should be able to predict the performance using a regression function trained on the values for those variables, and that the relative error should be independent of the outcome variable. We demon-



(a) Simple Discrimination. (b) Long Flow Discrimination. (c) BitTorrent Discrimination.

Figure 7: Overall performance distribution for the clients in all the ISPs. The distributions for the discriminating ISPs are not distinguishable.

strate the heuristic here for the Long Flow Discrimination experiment (Experiment 2).

NANO-Agents report the current length of all active flows in the performance summaries that they send to the NANO-server. We used this variable and the server location to predict the performance (throughput). We stratify the data on two dimensions: server location and flow's cumulative packets (current length). For this we consider all the servers sharing a particular 24 subnet as sharing a location and belonging in the same stratum. With this we ended up with 121 stratum on the server location dimension, with an average of 3 servers per stratum. We also created 1000 packet bins for cumulative packets of the flow. We used the average throughput over NANO-agent's reporting interval as the outcome variable. Next, we took a sample of 30 performance summaries falling in each two-dimensional stratum to train a regression function for each stratum. We then used these functions to predict the throughput for the traffic over the next hour and compute the relative error as $|y - \hat{y}|/y$, where y is the ground-truth value of throughput and $\hat{y}$ is the predicted value. We ignored the strata that had fewer than 30 summaries; between 3% and 8% stratum were rejected for each ISP across the three experiments.

Figure 8 shows the distribution of relative error for each of the three experiments. We believe that the error is acceptable because the standard deviation of throughput on the PlanetLab sites is quite large. In addition, we found that the error (without taking the absolute value) was centered at zero and showed small correlation with the outcome value. For the Long Flow Discrimination experiment, the correlation values are 0.01, 0.08, 0.05 0.01 and 0.05 for the ISPs N_1, N_2, N_3, D_1 and D_2, respectively.

6.3.2 *Estimating causal effect*

Using Eq.4 we compute the causal effect of each ISP for each experiment. As discussed in Section 4.1.2, when computing the causal effect of one ISP, we use the average performance of *all other* (discriminating and neutral) ISPs as the baseline. Unfortunately, because our experimental setup is small, we did not have samples from all ISPs for all the stratum. To overcome this problem, we only consider the stratum where at least three of the five ISPs in our experiment had 20 samples or more each. As a result, the baseline

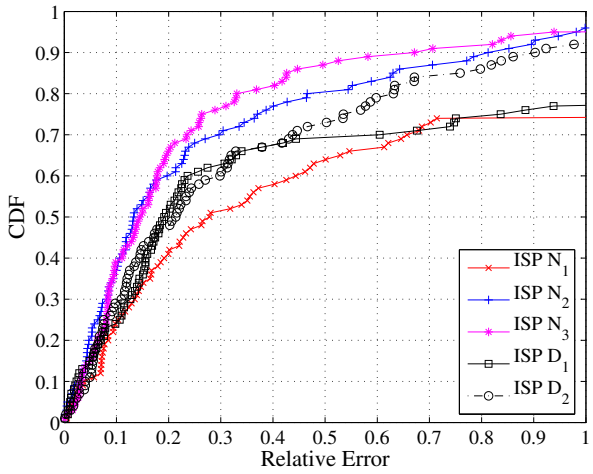


Figure 8: Distribution of relative error in predicting the throughput for the Long Flow Discrimination experiment with the variables that NANO collects.

performance for some of the strata might comprise fewer than four ISPs. For the Long Flow Discrimination experiment, only 4% of strata did not meet the other criteria. For the other two experiments, even fewer strata were rejected.

Because of the large number of strata, it is not possible to list here the causal effect of each ISP for each strata. Instead, we present the overall effect computed using Eq.4 for each ISP, service and experiment in Table 2. Negative numbers indicate relatively adverse causal effect, and we can label sufficiently large negative effect as discrimination. NANO correctly identifies ISPs that are discriminating for all the three experiments. Note that NANO is able to identify in Long Flow Discrimination experiment that ISP D_2 is not discriminating service S_1 and ISP D_1 is not discriminating against service S_2.

Also interesting to note is the extent of causal effect that NANO determines. For the Simple Discrimination experiment, ISP D_1 was dropping packets with lesser probability than ISP D_2; correspondingly, we see that the causal impact of D_1 is less severe than that of ISP D_2. Similarly, in the Long Flow Discrimination experiment, the causal im-

Service	ISP N_1	ISP N_2	ISP N_3	ISP D_1	ISP D_2
	Experiment 1. Simple Discrimination				
HTTP	2.10	8.39	14.65	**-108.64**	**-424.91**
	Experiment 2. Long Flow Discrimination				
HTTP S_1	5.17	4.80	18.9	**-61.20**	5.36
HTTP S_2	2.20	6.1	4.65	3.82	**-40.91**
	Experiment 3. BitTorrent Discrimination				
BitTorrent	1.16	-10.24	1.53	—	**-306.13**

Table 2: Causal effect for each ISP. NANO correctly identifies the discriminating and neutral ISPs. Note: All the numbers are in KB/s units. Negative numbers indicate poorer performance.

pact for both ISP D_1 and D_2 is less severe compared to the Simple discrimination experiment because in the Long Flow Discrimination experiment the two ISPs only start discriminating a flow once it reaches a certain cumulative packets threshold. As a result, part of the flow enjoys undiscriminated performance, dampening the overall causal effect.

Finally, we wish to note that in the course of our experiments, we encountered PlanetLab nodes, paths to which were very lossy to begin with. When the discriminating ISPs dropped packets for the flows with such PlanetLab nodes, we did not notice any appreciable decrease in throughput. As a result, NANO did not detect any significant discrimination for these flows, although some discrimination was in fact happening. This, however, is in line with our original goal of detecting net neutrality violations that result in performance degradation.

6.3.3 *Inferring discrimination criteria*

As discussed in Section 4.1.3, NANO can infer the discrimination criteria that the ISP might be using. Here we demonstrate this capability for the Long Flow Discrimination experiment.

To infer the criteria that ISP D_1 is using, we labelled the stratum on which we detected more than 100KB/s of causal effect as the discriminated strata and the remaining strata as undiscriminated strata. We ran the J.48 decision tree on the labelled dataset, where the data columns included the /24 subnet (`location` of servers and average number of cumulative packets (`cum_pkts`) for a session. The decision tree produces a rule that `cum_pkts <= 10103 --> not_discriminated, cum_pkts > 10103 --> discriminated`, and yields a 89% accuracy. The fact that the tree ignored the `location` variable completely shows that ISP is not discriminating based on destination. Instead, the rule suggests that the ISP is discriminating when the flow's duration is around 10000 packets. Recall from Table 1 that this is indeed the criteria that ISP D_1 uses for discrimination.

We have similarly obtained discrimination criteria for ISP D_2 for Long Flow Discrimination experiment as well as the BitTorrent Discrimination experiment.

6.3.4 *System Overhead*

NANO uses efficient techniques and data structures to reduce the reporting overhead from the NANO-Agents to the NANO-Servers as discussed in Section 5. Our experiments, showed a very low reporting overhead from the NANO-Agents running on Emulab nodes to our NANO-Server. For the Long Flow Discrimination experiment, the experiments transferred a total of 165GB of traffic on the client nodes over a period of over 6 hours. The total reporting overhead for this experiment was only 7MB, with NANO-Agents reporting statistics for a total of more than 14,000 flows along with other system information. This shows that NANO has very low reporting overhead.

7. Discussion

In this section, we address various issues with NANO. We first discuss various strategies for protecting user privacy when collecting data at NANO-Agents. We then discuss how to protect the integrity of the data collected at NANO-Agents (*i.e.*, how to control scenarios where NANO-Agents might lie). We also describe scenarios where ISPs might skew the statistical distributions of various performance metrics to mislead NANO's statistical inference, as well as possible defenses against this evasion.

7.1 Privacy

NANO collects performance-related information from NANO-Agents by passively monitoring traffic; it then passes performance-related statistics from this traffic back to the NANO-Server, which performs inference. This passively collected information may contain potentially sensitive information, in particular destinations a client has visited and content it has downloaded. Unfortunately, standard anonymization techniques, such as anonymizing IP addresses and various other features (*e.g.*, browser type, operating system) would obfuscate the very features used to stratify the data, thus preventing causal inference. Accordingly, we must apply techniques that somehow mask the identities of the clients but preserve the features that NANO uses to stratify the data.

Several methods could help NANO preserve client privacy. First, NANO-Agents could collect data from only the top 'k' Web sites (*e.g.*, according to Alexa) and strip personally-identifiable information from the payloads of this traffic. The data collected from clients would thus only reveal whether they had visited popular sites (not overly sensitive, since many users visit these sites) and the performance they were experiencing to those sites. Second, NANO could use a combination of passive and active measurements to produce the corpus of data used for inference; in such cases, the NANO-Server would receive all measurements, but would not be able to distinguish which traffic was generated solely to probe the network and which traffic was actually initiated by the client. Third, a client might perform some amount of pre-processing before passing data to the NANO-Server. The client could, for example, pre-process the traces to produce relevant statistics, without passing complete traffic traces to the NANO-Server; some

amount of stratification could even be performed at the client itself. Finally, clients using NANO might send their reports through an anonymizing network (*e.g.*, Freenet [4]) that obfuscates the source of the original report. Of course, the IP addresses of the clients would still be contained in the traffic traces, but we envision that in the process of mixing, IP addresses on various traces could be swapped without affecting the stratification. "Mixing" records collected from NANO-Agents is a subject for future work.

7.2 Integrity

NANO-Agents could lie about the data they collected, either by producing false traces, or by modifying the statistics about the traffic observed at the client. In these cases, it may be very difficult to detect when a client is reporting false statistics about its observed network performance. We suggest two possible techniques that could help mitigate this possibility. First, NANO could collect data from NANO-Agents that have similar values for various confounding factors (*e.g.*, same upstream ISP, same portion of the network topology); if, in these cases, reported performance measurements yield continuous discrepancies, NANO could determine that a NANO-Agent was reporting inaccurate results.

On the other hand, it is possible that, given knowledge of the distributions that NANO is measuring, that ISPs might be able to conceal the presence of discrimination by affecting the mean performance values. As described in Section 4, NANO establishes causality by measuring the difference in expected values for response times given the use of a specific ISP and its deviation from baseline measurements. However, an ISP could leave the *mean* value unaffected by giving exceptionally good performance to some clients and degrading performance for others. To defend against this type of attack, we imagine that NANO might be extended in two ways. First, we could modify NANO's causal inference algorithms to operate on *multiple points in the response-time distribution*, as opposed to simply inferring causality based on mean values. Second, presuming that an ISP's attempt to game the detection may vary over a range of time, we could run NANO's inference algorithm over different time granularities to attempt to catch more fine-grained variations in an ISP's policies across users, services, or applications.

8. Related Work

This section surveys previous projects that have attempted to characterize or prevent ISP discrimination.

Measurement tools. Glasnost [7, 10] detects TCP reset poisoning for connections of peer-to-peer applications. It simulates the BitTorrent protocol by running an active experiment from the Glasnost server to the end-host. It detects any spurious TCP RST packets which might be generated by the ISP to throttle the BitTorrent protocol. This approach has the drawback that if the ISPs change the discrimination mechanism then it would be difficult for Glasnost to detect throttling. Similarly, NVLens [29] focuses on detection of performance degradation among backbone ISPs via setting of ToS bits in the IP headers of the packets.

Tripwire uses a fingerprint-based technique to detect modification of in-flight packets, such as for insertion of advertisements [20]. This is an important class of neutrality violation, but we focus on violations that result in discrimination and performance degradation, rather than modification of actual content. These existing measurement techniques rely on active measurements and focus on specific discrimination mechanisms employed by the ISP. In contrast, NANO relies on *in situ* measurements, which allow for more robust detection, and on "black box" causal inference, which is more general in the face of evolving discrimination. Both Network Diagnostic Tool (NDT) [3], which performs test to end-users' machine to diagnose problems near the end-user, and Network Path and Application Diagnostics (NPAD) [8], which diagnoses network performance issues by extrapolating TCP performance to another location given measurements to the client from a server, rely on active client probing to detect network performance issues. These tools are deployed as part of a recently announced network transparency initiative M-Lab [18], a platform for hosting research tools to measure ISP discrimination.

Comparing performance across ISPs. NetDiff [17] detects performance differences between backbone ISPs. NetDiff uses the Geo-location and spread as a normalizing factor for fair comparison between ISPs, and in a sense adjusts for a confounding factor in the assertion that one ISP is better than another. NANO's agenda is detecting per-service discrimination which introduces additional confounding factors. Also, NetDiff uses ICMP packets to probe the paths, however, an ISP can easily thwart the efforts by detecting such probe packets and not discriminating them or the ISP could be specifically discriminating a particular service. NANO overcomes these difficulties by passively monitoring the performance of the various services. NANO can draw on previous work on characterizing ISP networks [16] and monitoring ISP SLA [22] to adjust for ISP topology differences. Keynote [14] is another application that compares performance across backbone ISPs; however, in addition to the above drawbacks it also requires ISP co-operation to place measurement nodes, which may not be possible.

Comparing services within an ISP. Our approach to inferring the effect of an ISP's policy on a service's performance relies on comparing performance of a service across multiple ISPs. Another interesting point in the design space is comparison performance of similar services within an ISP and then determining whether there is a difference in performance among these services; for example, NVLens [29], for instance, compares the latency for BitTorrent packets with the performance for HTTP packets. We believe that while interesting, this choice of comparison presents additional challenges that can be difficult to overcome. The services may differ in ways that naturally affect their performance: For example, a service that sends packets at a higher burst-rate may experience higher loss and latency for similar average transfer rate. Even if two services have similar traffic patterns, (*e.g.*, due to both using TCP), the completion time for a request may depend on additional server side variables, such

as the back-end delays, caching rate, or rate-limiting at the server end. A fair direct comparison between performance would require comparing the performance of services that are *similar* in all aspects that can affect a services performance; this can be difficult to achieve in general.

Architectures and testbeds. Yang *et al.* [28] propose a way to prevent ISPs from discriminating against packets altogether, but they require changes to user traffic (*e.g.*, encryption) unlike NANO which only detects discrimination based on passive measurements. We ultimately hope to use SatelliteLab [6] nodes to directly emulate random-treatment on the Internet, as part of NANO. Unfortunately, at this time, the *satellite* nodes in the SatelliteLab system only support relaying the traffic that the *planet* nodes generate, which introduces an additional confounding factor.

9. Conclusion

This paper presented NANO, a system that applies causal inference techniques to passively collected data from end-hosts to detect service degradation because of discrimination by ISPs. Our evaluation shows that NANO accurately determines when an ISP is causing performance degradation along a network path, even when various confounding factors may complicate the inference. We successfully demonstrated NANO's ability to detect discrimination for various discrimination policies and application types, thus demonstrating that NANO's detection is general, and can detect discrimination even when the ISP's discrimination policies are not known *a priori*.

NANO's generality follows directly from its approach: Rather than applying specific rules to detect specific types of discrimination and actively probing ISPs to try to find cases where those rules are violated NANO observes *in situ* traffic and performs causal inference to determine whether the characteristics of the actual application traffic itself shows any variations that can be attributed to the ISP. Observing the actual application traffic and applying causal inference makes NANO both *robust to evasion* and *future-proof* against discrimination policies and mechanisms that ISPs might deploy.

Our next step is to deploy NANO as an operational system so that anyone can use NANO to determine whether their access network is discriminating. We plan to collect data from a wide range of heterogeneous NANO-Agents deployed across the Internet and offer NANO-Server as a service. We are currently working with a large content provider to deploy NANO on a large, geographically distributed measurement platform as part of a larger initiative to make communication networks more transparent.

REFERENCES

[1] N. Andersen. Cox ready to throttle P2P, non "time sensitive" traffic. http://tinyurl.com/bcexla, Jan. 2009.

[2] A. Bavier, M. Bowman, D. Culler, B. Chun, S. Karlin, S. Muir, L. Peterson, T. Roscoe, T. Spalink, and M. Wawrzoniak. Operating System Support for Planetary-Scale Network Services. In *Proc. First Symposium on Networked Systems Design and Implementation (NSDI)*, San Francisco, CA, Mar. 2004.

[3] R. Carlson. Network Diagnostic Tool. http://e2epi.internet2.edu/ndt/.

[4] I. Clarke. A distributed decentralised information storage and retrieval system. Master's thesis, University of Edinburgh, 1999.

[5] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. In *Proc. 6th USENIX OSDI*, San Francisco, CA, Dec. 2004.

[6] M. Dischinger, A. Haeberlen, I. Beschastnikh, K. Gummadi, and S. Saroiu. SatelliteLab: Adding Heterogeneity to Planetary-Scale Testbeds. In *Proc. ACM SIGCOMM*, Seattle, WA, Aug. 2008.

[7] M. Dischinger, A. Mislove, A. Haeberlen, and K. P. Gummadi. Detecting bittorrent blocking. In *Proc. Internet Measurement Conference*, Vouliagmeni, Greece, Oct. 2008.

[8] M. M. et al. Network Path and Application Diagnosis. http://www.psc.edu/networking/projects/pathdiag/.

[9] E. Felten. Three Flavors of Net Neutrality. http://www.freedom-to-tinker.com/blog/felten/three-flavors-net-neutrality, Dec. 2008.

[10] Glasnost: Bringing Transparency to the Internet. http://broadband.mpi-sws.mpg.de/transparency.

[11] Protocol Buffers. http://code.google.com/apis/protocolbuffers.

[12] G. Gross. Google, partners release net neutrality tools. http://www.thestandard.com/news/2009/01/28/google-partners-release-net-neutrality-tools, Jan. 2009.

[13] N. Jewell. *Statistics for Epidemiology.* Chapman & Hall/CRC, 2004.

[14] Keynote Home Page. http://www.keynote.com/, 1999.

[15] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The Click modular router. *ACM Transactions on Computer Systems*, 18(3):263–297, Aug. 2000.

[16] H. V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. E. Anderson, A. Krishnamurthy, and A. Venkataramani. iPlane: An information plane for distributed services. In *Proc. 7th USENIX OSDI*, Seattle, WA, Nov. 2006.

[17] R. Mahajan, M. Zhang, L. Poole, and V. Pai. Uncovering Performance Differences among Backbone ISPs with Netdiff. In *Proc. 5th USENIX NSDI*, San Francisco, CA, Apr. 2008.

[18] Measurement Lab. http://measurementlab.net, Jan. 2009.

[19] J. Pearl. *Causality: Models, Reasoning, and Inference.* Cambridge University Press, 2000.

[20] C. Reis, S. D. Gribble, T. Kohno, and N. C. Weaver. Detecting in-flight page changes with web tripwires. In *Proc. 5th USENIX NSDI*, San Francisco, CA, Apr. 2008.

[21] S. B. Robert Beverly and A. Berger. The internet's not a big truck: Toward quantifying network neutrality. In *Passive & Active Measurement (PAM)*, Louvain-la-neuve, Belgium, Apr. 2007.

[22] J. Sommers, P. Barford, N. Duffield, and A. Ron. Efficient Network-wide SLA Compliance Monitoring. In *Proc. ACM SIGCOMM*, Kyoto, Japan, Aug. 2007.

[23] M. B. Tariq, M. Motiwala, and N. Feamster. NANO: Network Access Neutrality Observatory. In *Proc. 7th ACM Workshop on Hot Topics in Networks (Hotnets-VII)*, Calgary, Alberta. Canada., Oct. 2008.

[24] M. B. Tariq, A. Zeitoun, V. Valancius, N. Feamster, and M. Ammar. Answering "What-if" Deployment and Configuration Questions with WISE. In *Proc. ACM SIGCOMM*, Seattle, WA, Aug. 2008.

[25] L. Wasserman. *All of Statistics: A Concise Course in Statistical Inference.* Springer, 2003.

[26] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. An integrated experimental environment for distributed systems and networks. In *Proc. 5th USENIX OSDI*, pages 255–270, Boston, MA, Dec. 2002.

[27] J. Wolberg. *Data Analysis Using the Method of Least Squares.* Springer, 2006.

[28] X. Yang, G. Tsudik, and X. Liu. A technical approach to network neutrality. In *Proc. 5th ACM Workshop on Hot Topics in Networks (Hotnets-V)*, Irvine, CA, Nov. 2006.

[29] Y. Zhang, Z. M. Mao, and M. Zhang. Ascertaining the Reality of Network Neutrality Violation in Backbone ISPs. In *Proc. 7th ACM Workshop on Hot Topics in Networks (Hotnets-VII)*, Calgary, Alberta. Canada., Oct. 2008.