Glasnost: Enabling End Users to Detect Traffic Differentiation

Marcel Dischinger
MPI-SWS

Massimiliano Marcon
MPI-SWS

Saikat Guha
MPI-SWS, Microsoft Research

Krishna P. Gummadi
MPI-SWS

Ratul Mahajan
Microsoft Research

Stefan Saroiu
Microsoft Research

Abstract

Holding residential ISPs to their contractual or legal obligations of "unlimited service" or "network neutrality" is hard because their traffic management policies are opaque to end users and governmental regulatory agencies. We have built and deployed Glasnost, a system that improves network transparency by enabling ordinary Internet users to detect whether their ISPs are differentiating between flows of specific applications. We identify three key challenges in designing such a system: (a) to attract many users, the system must have low barrier of use and generate results in a timely manner, (b) the results must be robust to measurement noise and avoid false accusations of differentiation, which can adversely affect ISPs' reputation and business, (c) the system must include mechanisms to keep it up-to-date with the continuously changing differentiation policies of ISPs worldwide. We describe how Glasnost addresses each of these challenges. Glasnost has been operational for over a year. More than 350,000 users from over 5,800 ISPs worldwide have used Glasnost to detect differentiation, validating many of our design choices. We show how data from individual Glasnost users can be aggregated to provide regulators and monitors with useful information on ISP-wide deployment of various differentiation policies.

1 Introduction

A confluence of technical, business, and political interests has made "network neutrality" a hot button issue [18, 19]. The debate revolves around whether and to what extent Internet service providers (ISPs), who own and operate data networks, should be allowed to differentiate one class of traffic from another. Many ISPs want to restrict bandwidth-hungry applications that can hurt other applications in the network. Some also want to control applications such as VoIP that reduce ISPs' ability to profit from competing services of their own. In contrast, many content providers are against traffic differentiation because it gives the ISPs arbitrary control over the quality of service experienced by users. In parallel, regulatory bodies and politicians are trying to devise policies that balance competing concerns [20, 21].

As this debate rages, ordinary Internet users are often in the dark, even though they are directly affected. The information sources available to users today are media reports, blogs, and statements made by ISPs; such information sources are imprecise at best and incorrect at worst. As a result, much traffic differentiation occurs without their knowledge. However, when ISPs traffic management practices come to light, user outrage forces regulatory bodies to conduct public hearings on prevalent practices [20, 21].

This situation led us to build and deploy a system, called Glasnost, that enables users to detect if they are subject to traffic differentiation. We make no judgment about whether traffic differentiation should be permitted by regulatory policy. Rather, our motivation is to make any differentiation along their paths *transparent* to users.

While other recent research efforts also aim to detect traffic differentiation [27, 31], Glasnost is unique in its focus on users. Instead of providing only a broad characterization of differentiation in the Internet, our goal is to let individual users determine if they experience differentiation and quantify its impact at the time they use our system.

Our focus is on enabling individuals who are not technically savvy. This creates design constraints that are typically not present in other measurement systems. First, the bar to using the system must be low. For instance, it is undesirable to require the installation of special software on client machines, especially if such software needs privileged access. This constraint hinders our ability to collect high-fidelity data (e.g., packet traces) or to finely control packet transmissions. We must limit ourselves to coarse-grained data obtained through unprivileged client operations. Second, the results for an individual user must be accurate and simple to interpret. For example, we cannot return results that

rely on inferences derived from data aggregated across users. While such results are accurate on aggregate, they could be incorrect when applied to an individual user. Third, the system must evolve with ISP practices. Otherwise, Glasnost would gradually become unable to detect the presence of differentiation and users would stop trusting the system.

We based our Glasnost design on these constraints. The result is a system that is effective and easy to use. A user can detect differentiation by simply pointing her browser to a Web page. The browser downloads and runs a Java applet which exchanges traffic with our measurement server. The client-server nature of our architecture helps to avoid many of the operational issues with network measurements, such as traversing NATs and firewalls, or raising alarms in network intrusion detection systems. The traffic exchange is designed to accurately and quickly detect any differentiation. We also build a simple flow emulation tool that simplifies the incorporation of tests to detect new differentiation techniques that emerge in the Internet.

The diversity of ISP practices makes it challenging to detect traffic differentiation reliably. For instance, an ISP might employ differentiation only at specific times (e.g., in the evenings), or only under high loads, or only for flows that send too much traffic. These factors led us to design an on-demand system. Each time a user uses Glasnost, she performs an *individual* test that detects the presence of traffic differentiation for her Internet connection *at the time of the test*. This provides a more reliable answer to this user than extrapolating the results from other testing times or other users.

Glasnost has been operational since March 2008, enabling users to detect BitTorrent differentiation. Between March 2008 and September 2009, more than 350,000 users from over 5,800 ISPs worldwide have used the system. Several individuals and corporations volunteered to host Glasnost measurement servers on their own infrastructure in order to allow operations on an even larger scale. We believe that our design principles have directly contributed to the success of Glasnost.

In addition to the design and evaluation of Glasnost, we also present a detailed analysis of BitTorrent differentiation in the Internet. We find that about 10% of our users experience differentiation of BitTorrent traffic. We also study ISPs' BitTorrent differentiation policies in detail over a period of two months (from January to February 2009) using data from the Glasnost tests. We find, for instance, that it is more common for ISPs to differentiate against file uploads than downloads and to differentiate throughout the day rather than only during peak hours.

2 Traffic Differentiation

Traffic differentiation refers to an ISP treating the packets of one flow differently than those of another flow. Based on information published by ISPs, researchers, and equipment vendors [5, 10, 22], we characterize traffic differentiation along three dimensions.

1. Traffic differentiation based on flow types. To differentiate between flows of different types, i.e., belonging to different applications, ISPs must distinguish the packets of one flow from those of other flows. This can be done by examining one of the following:

(a) *The IP header.* The source or destination addresses can determine how an ISP treats a flow. For example, universities routinely rate-limit only traffic that's going to or coming from their student dorms.
(b) *The transport protocol header.* ISPs can use port numbers or other transport protocol identifiers to determine a flow's treatment. For example, P2P traffic is sometimes identified based on its port numbers.
(c) *The packet payload.* ISPs can use deep-packet inspection (DPI) to identify the application generating a packet. For example, ISPs look for P2P protocol messages in packet payload to rate-limit the traffic of P2P applications, such as BitTorrent.

2. Traffic differentiation independent of flow type. In addition to features of a flow itself, an ISP may use other criteria to determine whether to differentiate. Some of these include:

(a) *Time of day.* An ISP may differentiate only during peak hours.
(b) *Network load.* An ISP may differentiate on a link only when the network load on that link is high.
(c) *User behavior.* An ISP may differentiate only against users with heavy bandwidth usage.

3. Traffic manipulation mechanisms. There are a number of ways in which an ISP can treat one class of packets differently.

(a) *Blocking.* One form of differentiation is to terminate a flow, either by blocking its packets or by injecting a connection termination message (e.g., sending a TCP FIN or TCP RST packet).
(b) *Deprioritizing.* Routers can use multiple priority queues when forwarding packets. ISPs can use this mechanism to assign differentiated flows to lower priority queues and to limit the throughput of certain classes.

(c) *Packet dropping.* Packets of a flow can be dropped either using a fixed or variable drop rate.

(d) *Modifying TCP advertised window size.* ISPs can lower the advertised window size of a TCP flow, prompting a sender to slow down.

(e) *Application-level mechanisms.* ISPs can control an application's behavior by modifying its protocol messages. For example, transparent proxies [28] can redirect HTTP or P2P flows to alternate content servers.

What kinds of traffic differentiation does Glasnost detect?
Our current implementation of Glasnost detects traffic differentiation that is triggered by transport protocol headers (e.g., ports) or packet payload. These triggers are more common than IP headers [1, 5].

We designed Glasnost to be an on-demand system. Each time a user uses Glasnost, we detect traffic differentiation between flows of the user at the time of the test. While Glasnost has not been designed detect traffic shaping that affects all flows of a user, e.g., based on time of day or network load or user behavior, it is possible to infer such shaping policies by aggregating and comparing the results of Glasnost tests conducted at different times of the day by different users on different networks.

Instead of inferring differentiation based on a particular manipulation mechanism, Glasnost detects the presence of differentiation based on its impact on application performance.

3 Design Principles

In the process of developing Glasnost we identified several key design principles. Although in Glasnost our focus is traffic differentiation, the design principles we identified are more general and apply to many measurement systems that want to attract a large number of users. In this section, we discuss these principles in detail and argue why they are generally useful when designing measurement systems for Internet users at large.

Our goal was to build a system that lets ordinary Internet users determine if they are affected by traffic differentiation. Because of its focus on end users and the nature of its measurements, Glasnost must satisfy certain design requirements that are typically not present in other measurement systems. We distill these requirements into three design principles. These principles dictate that the system must be easy to use so that it can serve any Internet user, its inferences must be robust and simple to interpret, and it must be extensible to allow detection of new network policies as they evolve.

We explain these principles in detail below and also describe the consequences they have on the design of Glasnost. These consequences motivate certain design choices and rule out many others.

Principle #1: Low barrier of use
Attracting a large number of users to a measurement system requires having a low barrier of use. Although this challenge appears obvious, solving it is the key to success. As we discuss later, it complicated the design of other aspects of the system. But at each step we resisted the temptation to compromise in the interest of other desirables such as efficiency and higher-fidelity data.

Design consequences. There are four design consequences of this principle. First, because most users are not technically savvy, the interface must be simple and intuitive. Second, we cannot require users to install new software or perform administrative tasks. Many network measurement techniques require installing drivers (e.g., the WinPcap library for Windows) or running privileged code (e.g., raw sockets) on users' machines. Such code can provide detailed, low-level data (e.g., packet traces) that simplifies the measurement task. But in our experience, users are often unwilling to use systems with such requirements. For example, one of our earlier attempts required users to run code with administrator privileges on their machines and to leave a port open in their firewalls and NATs. These obstacles greatly limited adoption; we attracted fewer than fifty users. Third, because many users have little patience, the system must complete its measurements quickly. Fourth, to incentivize users to use the system in the first place, the system should display per-user results immediately after completing the measurements.

In order to satisfy above the requirements, our current client-side implementation uses a small-size Java applet (21 KBytes) that users download off our webpage. The applet exchanges traffic with our servers, which we then analyze to detect differentiation (we explain the nature of this traffic below). The test runs for about 6 minutes. Immediately after the test is finished Glasnost whether the user is affected by traffic differentiation.

Our quick and simple test methodology is inspired by non-research-oriented web sites for broadband speed tests [2] and represents a departure from other research systems. For instance, Scriptroute [25] requires users to write their own measurement scripts, and thus its use has been limited to researchers and other experts.

Principle #2: Measurement accountability
Because the system is designed for ordinary users, it is essential that the measurements are accurate and that the results cannot be misinterpreted. For instance, consider the results of an experiment to infer path capacities in

the Internet. Since the measurements can be affected by transient noise, researchers will know that the answer computed along an individual path cannot be trusted but the answers can be aggregated to provide an accurate estimate of path capacity. But an ordinary user that is interested in the capacity of her own path might not be in a position to make that distinction.

When detecting traffic differentiation, accurate interpretation of results is critical due to the controversial nature of traffic management in the Internet: there is still a heated debate whether it is legal for an ISP to employ traffic management. In addition, if people were to falsely interpret results as their ISP performing traffic differentiation when in fact it is not, the system would quickly lose credibility. In fact, in the past there have been instances when some widely publicized studies have mistakenly accused ISPs of using policies they never deployed [26, 29].

Design consequences. Maintaining measurement accountability has three design consequences. First, the test to detect differentiation should, to the extent possible, marginalize any factors that add uncertainty. The performance of an Internet flow can be affected by many confounding factors. This includes the operating system, especially its networking stack and its configuration. Additionally, directly using application client software is problematic as it does not give full control over the measurement traffic. Short-term throughput of such "natural" flows can differ because of differences in packet sizes and burstiness. Finally, we have to consider transient noise, as, e.g., caused by background traffic.

With passive measurement tools, it is often not easy to isolate these factors. These tools must take into account for a large number of confounding factors in their inference. The complexity of this analysis can lead to inaccurate results. In contrast, active measurements can be designed to avoid most confounding factors. Having full control over the traffic that is sent to measure performance simplifies the analysis. Further, active measurements allow to run all measurements between the same pair of hosts, removing factors like OS and networking stack. The only remaining confounding factor is transient noise, which can be dealt with using simple techniques such as repeating measurements multiple times.

Second, because not all uncertainty can be removed from the inference, the result presented to the user must be conservative, with a near-zero false positive rate. In the context of traffic differentiation, a false positive means that the system falsely claims that the user is experiencing traffic differentiation. Minimizing false positives is challenging because it results in an increase in the false negative rate. This trade-off is inherent.

Because of the concerns above, our testing primitive is based on comparing the throughput of a pair of flows. One flow in the pair belongs to the potential victim application. The second is a reference flow that belongs to a different application. The flows are identical except for the trigger that we want to test for differentiation, such as port number or payload. The flows are generated back-to-back and multiple pairs are run to reduce and calibrate the effect of noise.

Third, we must be prepared to provide the data and the evidence behind our inferences when requested. We retain the data of all measurements in which Glasnost detects traffic differentiation. We treat this data as evidence. If we are challenged to justify our findings, the stored data will help us explain on what basis Glasnost declared that an ISP is using traffic differentiation.

Principle #3: Easy to evolve

To remain relevant, a system that wants to detect traffic differentiation must be able to evolve as ISPs evolve their traffic management policies. For example, in Fall 2008, Comcast blocked BitTorrent uploads for some of its customers [10]. Several months later, they started replacing this practice with less severe forms of differentiation [7]. In fact, our recent measurements indicate that BitTorrent traffic blocking is rare today unlike in 2008. A system with a fixed set of capabilities will have a limited shelf life in such an evolving environment.

Design consequences. This principle mandates incorporation of mechanisms that help the system evolve with the network. Network evolution may be incidental or adversarial. In an incidental evolution, ISPs might target new applications in the future or use new traffic manipulation mechanisms. A detection system should be extensible, to add tests that detect traffic differentiation against popular new applications or based on new shaping techniques. Glasnost enables advanced users to submit packet-level traces of applications that they suspect are being targeted by their ISPs. User suspicion is powerful; it was how many of the currently known ISP differentiation behaviors came to light. We do not expect all users to be able to submit traces but there are many enthusiastic users that are capable of collecting (with our help if needed) and sharing traces. Glasnost then makes it easy to use these network traces to construct new detection tests. These tests help us keep pace with new traffic differentiation techniques and applications that may be targeted.

Adversarialy, ISPs could begin whitelisting traffic from measurement servers in an attempt to evade detection. A successful system must be aware of this problem and find ways to minimize whitelisting. Our solution was to make our server code publicly available. Anyone can setup Glasnost on a well-provisioned server and

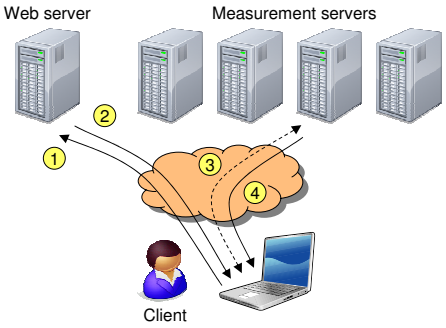


Figure 1: The Glasnost system. (1) The client contacts the Glasnost webpage. (2) The webpage returns the address of a measurement server. (3) The client connects to the measurement server and loads a Java applet. The applet then starts to emulate a sequence of flows. (4) After the test is done, the collected data is analyzed and a results page is displayed to the client.

other users can start measuring to new servers. Making our code publicly available allowed other Glasnost servers to appear on the Internet, which makes it hard for ISPs to evade detection. However, this method is not foolproof; a determined ISP may choose to stay up-to-date with the list of Glasnost servers. We doubt that many ISPs would be willing to invest significant effort in evading detection. As much as an ISP would like to conceal its traffic management practices from the public, denying those practices or making blatant attempts to hide them is risky. Such behavior, if detected, would attract intense scrutiny from telecom regulators and would severely damage the ISP's reputation. For example, when Comcast's BitTorrent blocking practices were revealed to the public [1], Comcast was fined by the FCC and was subjected to highly critical media coverage.

4 Design of Glasnost

We now present the design of Glasnost based on the requirements outlined above.

4.1 System architecture

Glasnost is based on a client-server architecture. Clients connect to a Glasnost server to download and run various tests. Each test measures the path between the client and the server by generating flows that carry application-level data. This data is carefully constructed to detect traffic differentiation along the path.

Figure 1 presents a high-level description of how clients measure their Internet paths. A client first contacts a central webpage that redirects to a Glasnost measurement server. This dynamic redirection enables load balancing across measurement servers and makes it easy to incorporate new servers by adding them to the redirection list.

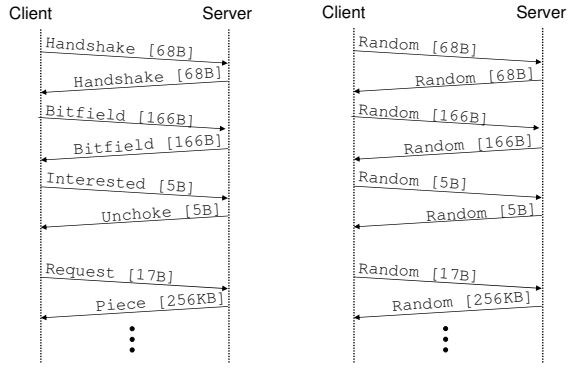
Figure 2: The Glasnost web interface.

After the client is redirected, the measurement server presents a simple interface to the user. As shown in Figure 2, the user selects the application traffic she would like to test and starts the test by just clicking the "Start testing" button. The client's browser downloads a Java applet that starts exchanging packets with the server. We elaborate on the Glasnost measurement tests next.

4.2 Measurement tests

The key primitive behind the Glasnost measurement tests is the *emulation* of a pair of flows that are identical except in one respect that we suspect triggers differentiation along the path. Comparing the performance of these flows helps to determine if differentiation is indeed present.

Figure 3 shows two flows designed to detect whether differentiation based on BitTorrent protocol content is present along a path. The exchange on the left corresponds to the first flow. The client opens a TCP connection to the measurement server and starts exchanging packets that implement the BitTorrent protocol: the packet payloads carry BitTorrent protocol headers and content. The exchange on the right corresponds to the second flow. The client opens another TCP connection and performs the same packet exchange, but the packets contain random bytes instead of BitTorrent headers or data. An ISP that differentiates against BitTorrent based on protocol messages would impact only the first flow. Thus, significant differences in the flows' performance is likely to be caused by the differences in their payloads and lets us detect whether differentiation is present along the path. Transient noise can also lead to differences in flows' performance; we describe in the next section how we handle noise.



(a) BitTorrent flow (b) Reference flow

Figure 3: A pair of flows used in Glasnost tests. The two flows are identical in all aspects other than their packet payloads, which allows us to detect differentiation that targets flows based on their packet contents.

During the test, the measurement server records a packet-level trace of all emulated flows and the client applet records ancillary information including exceptions caused by network errors. Once the transfers end, the client uploads the recorded information to the server. The server analyzes this information together with the traces collected on the server-side and shows the findings to the client.

Glasnost's emulation methodology leads to measurement robustness. As Figure 3 shows, application-level data is the only difference between the two emulated flows. The two flows traverse the same network path and have the same network-level characteristics, such as port numbers, packet sizes, etc. In contrast, passive measurement, a different technique, may have many factors differ across these measured flows. Correctly accounting for all such differences is challenging.

Another benefit of active measurement is the ability to carefully control the measurement test. For example, we can repeat flows with different payloads or port numbers. This ability allows Glasnost to precisely identify the specific factors that trigger differentiation.

In the next section, we describe our measurement test in more detail and how we make it robust to transient noise. We describe how we make the system easy to evolve using a trace replay based tool for constructing measurement tests in Section 6.

5 Robust Detection of Differentiation

As described earlier, Glasnost emulates a pair of flows and determines the presence of traffic differentiation by comparing their performance. When comparing the performance of a pair of flows, we must ensure that their difference is indeed due to the differences in their content and not due to some changes in the test environment. Our measurement tests are constructed in a way that eliminates all major confounding factors except one – transient noise due to interference from cross-traffic (background traffic) along the measurement path. In this section, we discuss techniques to robustly detect traffic differentiation in the face of transient noise.

The primary challenge in this task stems from the fact that the noise can vary at small time-scales. Thus, two flows can be affected differently even if run back-to-back. As one egregious example, we found that the throughput of two back-to-back flows differed by a factor of three even though the flows were identical. A simplistic detection method will mistakenly detect differentiation in this case. It might appear that the differential impact of noise could be reduced by running the flows simultaneously. But we find that setup to be even worse because of self-interference among the two flows.

Our basic strategy for robust detection is to run each flow type multiple times. We use the variance in the performance of the flows of the same type to identify paths that are too noisy to enable reliable detection. For the remaining paths, we can then detect differentiation by comparing the flows of different types. We first describe how we apply this strategy when tests are run long enough that we do not have to worry about having too little data. As we found that many users are too impatient to run long tests, we adapted our strategy to tests that run for a shorter duration.

We describe our method using throughput as the measure of flow performance[1], since it is of prime interest to many applications and is the target of many ISPs looking to reduce their network load. Because of TCP dynamics, throughput is directly affected by any differentiation that impacts flow latency or loss.

5.1 Filtering tests affected by noise

To detect the level of transient noise, we repeat the runs of the two flow types multiple times back-to-back. Unlike active ISP differentiation, transient noise does not discriminate based on flow content; it would not affect multiple runs of the same flow type and thus can be detected by comparing their performance.

To understand transient noise patterns and the extent to which they affect flow throughput, we configured our Glasnost deployment to run a BitTorrent flow and a reference flow with random bytes, five times each. The runs of the two flow types were interspersed and each flow lasted for 60 seconds to allow sufficient time for TCP to achieve stable throughput. Over a period of one

[1]Our method can be extended to other measures of performance such as jitter.

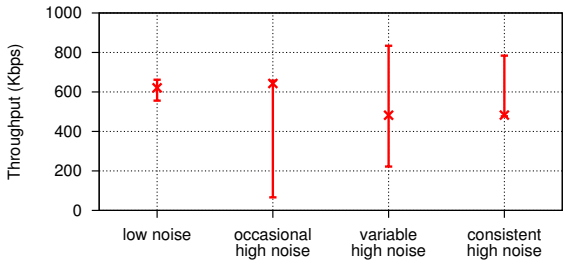


Figure 4: The four classes of noise we observed in our analysis. The graph shows the minimum, median, and maximum throughputs observed in example tests affected by each class of noise.

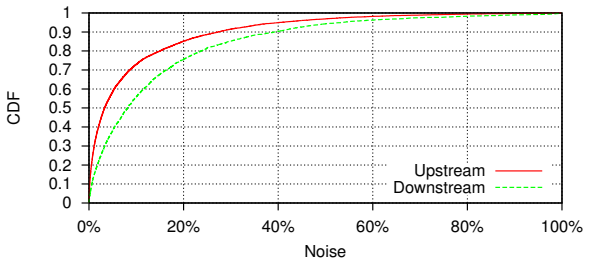


Figure 5: Noise observed in our 3,705 sample dataset. 85.2% of upstream flows and 75.7% of downstream flows have less than 20% of noise. Noise is measured as the difference between maximum and median throughput calculated as a percentage of maximum throughput.

month, we collected measurements of 3,705 residential broadband hosts, 2,871 in the upstream and 834 in the downstream direction.

We compared the throughput obtained by the five runs of each flow type with each other. Our analysis of the maximum, median, and minimum throughput reveals the four distinct patterns shown in Figure 4, corresponding to four different cross-traffic levels:

1. **Consistently low cross-traffic:** all throughput measurements belonging to the same flow type fall within a narrow range (i.e., min is close to max).

2. **Mostly low but occasionally high cross-traffic:** a majority of throughput measurements are clustered around the maximum but a few points are farther away (i.e., max and min are far apart but median is close to max).

3. **Highly variable cross-traffic:** the throughput measurements are scattered over a wide range (i.e., max and min are far apart and median is far apart from both).

4. **Mostly high but occasionally low cross-traffic:** a majority of throughput measurements are clustered around the minimum but a few measurements are farther away (i.e., max and min are far apart but median is close to min).

Our categorization of the level of cross-traffic in each case is based on two key observations about the nature and impact of cross-traffic. First, cross-traffic only lowers throughput and never improves it. Thus, when a majority of throughput measurements are close to min but far apart from max (as in category 4 above), it is more likely that the noise-free throughput is closer to max than min.

Second, cross-traffic is unlikely to be consistently high over a long period of time. In theory, measurements in category 1 above could be explained by consistently high cross-traffic. But, this would require the cross-traffic to remain high and consistent (without changing) over the duration of the entire experiment, which is ten minutes. We believe that this is unlikely.

For robust detection of differentiation, we discard all tests where a majority of flows are affected by high noise (i.e., categories 3 and 4). For these tests, we cannot determine whether the difference in throughput is caused by differentiation or transient noise. We analyze only the remaining tests, for which a majority of runs experience low noise (i.e., categories 1 and 2).

To help determine which tests belong to the predominantly low noise category, we plot the difference between maximum and median throughput as a percentage of maximum throughput in Figure 5. We found that for a large majority of tests (85.2% of upstream tests and 75.7% of downstream tests) the median throughput is within 20% of the maximum. The difference between median and maximum throughput is considerably larger for the remaining flows. We thus use the 20% difference between median and max throughputs as a threshold to discard tests that are significantly affected by noise. Next, we describe how we detect traffic differentiation within the remaining tests.

5.2 Detecting differentiation in low-noise tests

To detect traffic differentiation among tests that are identified as low noise, we compare the maximum througput of each flow type. Our decision to use the maximum is based on the observations that (a) in low-noise cases, most measurements lie close to the maximum throughput and (b) because noise tends to lower throughput, the maximum throughput is a good approximation for what the flows would achieve without cross-traffic.

We infer that the two flow types are being treated differently if the maximum throughput of one differs from

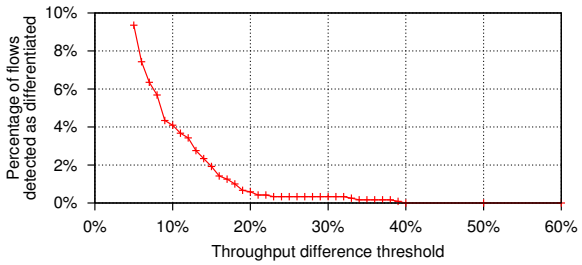


Figure 6: Selecting a good throughput difference threshold. Thresholds smaller than 20% tend to produce a significant number of false positives.

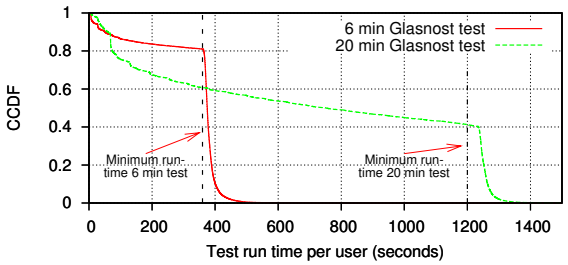


Figure 7: Duration users run the Glasnost test. Longer duration tests are aborted by a larger fraction of users.

that of the other by more than a threshold δ. Selecting a good δ involves a trade-off. With high values, we cannot detect differentiation unless the impact on throughput is high. For instance, with δ=50%, we would only detection differentiation that halves the flow throughput. Thus, high values raise the false negative rate. On the other hand, with low values of δ (say 5%), we risk false positives, i.e., declaring that ISPs are employing traffic differentiation while they actually do not.

To understand how the false positive rate varies with δ, we selected 302 test runs from users from ISPs that we know do not differentiate. Figure 6 plots the percentage of tests that are falsely marked as being differentiated for different threshold values. The plot shows an interesting trend; the false positive rate drops steeply until δ reaches 20%. Beyond this threshold, there are a handful of hosts (0.58%) that pass our noise tests but are still falsely marked as differentiated. To avoid any false positives, we would need to raise the threshold to 40%, which increases the false negative rate.

We thus set δ to 20%. With this value we maintain a low false positive rate (under 0.6%), but we fail to detect differentiation that reduces a flow's throughput by less than 20%. We consider this an acceptable trade-off.

5.3 User impatience with long tests

As described above, we configured Glasnost to run a pair of one-minute-long flows five times, resulting in a total test time of 10 minutes. The tests we originally deployed also detected whether the differentiation was based on port number or payload, extending the test duration to 20 minutes. While this test configuration enables us to detect differentiation with high confidence, we noticed that a considerable fraction of users were aborting the tests before completion.

Figure 7 shows how long users keep their Glasnost test running. The plot for 20 minute long tests shows an alarming decline in the percentage of users as the test progresses. Only 40% of the users stay till the end and nearly 50% aborted their tests within the first 10 minutes. The sudden drop near the 20 minute point corresponds to successfully completed tests.

Our results show that users are impatient. Most are not willing to use tests that take more than a few minutes. To confirm this, we reconfigured Glasnost to use shorter-duration tests. We reduced the number of times we repeat each flow type to two (from five), and we decreased the duration of each flow to 20 seconds (from 60 seconds), which is still sufficient for TCP to exit slow-start and achieve stable throughput. We bundled the tests for both upstream and downstream directions, and the resulting test takes 5.33 or roughly 6 minutes.

Figure 7 shows also how long users keep the 6 minute Glasnost test running. More than 80% of the users stay till the end, confirming that shorter tests on the order of a few minutes are more effective at retaining users.

5.4 Detecting differentiation with short tests

Short duration tests are challenging for detecting differentiation robustly because they gather few measurement samples. To estimate the impact of this reduction in data on detection accuracy, we consider data from the longer tests for which we have a result, i.e., for which we know whether or not the ISP is differentiating. We prune the data to include only what would be gathered by the short test and run our analysis on the pruned data. We compare the results from this shorter test data with those obtained before.

We find that nearly 25% of the long tests that we were able to successfully analyze before, were discarded as too noisy after pruning. We find the false positive rate (i.e., cases when the long test found no traffic differentiation but the short test did) to be 2.8% and the false negative rate (i.e., cases when the long test found traffic differentiation but the short test did not) to be 0.9%.

We also find that we can achieve a four-fold reduction in false positive rate, to 0.7% (which is comparable

to the false positive rate of long tests), by raising the δ threshold from 20% to 50%. While this increases the false negative rate to 1.7%, we consider it an acceptable trade-off.

6 Facilitating New Test Construction

Manually implementing Glasnost tests for a new application is a laborious and error prone task. It requires detailed knowledge of the application's protocols and their common implementations. This creates a high barrier for new test construction, making it difficult to keep pace with the evolution of ISPs' policies.

In this section, we present a tool called `trace-emulate` that simplifies the construction of new tests by automating most of the process. We also present a validation of the tests constructed by `trace-emulate` using the open source DPI engine of a commercial traffic shaper [22].

Our `trace-emulate` tool automatically generates a new Glasnost test from the packet-level trace of an application. It extracts the essential characteristics of the application flows. These include packet sizes and payloads as well as the order of packets with protocol messages and the inter-packet timing.

The test configuration that `trace-emulate` outputs is then used by the Glasnost Java applet to run the test. When run against the server, the applet exchanges two flows. The first flow has the same characteristics as the original trace. For example, assume that in the original trace the client performed the following operations: (1) sent packet A, (2) received packet B, (3) sent packet C after t seconds. These operations occur in the same order and relative times in the generate flow. In some cases, simultaneously preserving packet ordering and inter-packet timing is impossible. Such cases arise when an endpoint is waiting to receive a packet that gets delayed in the network. We make the endpoint (client or the server) wait until the packet is received before continuing the emulation, even though it increases the inter-packet time. Our decision to preserve ordering at the expense of timing is motivated by the observation that ISPs often use the sequence of protocol messages to identify applications, rather than their relative timing. The second flow exchanged by the applet is a reference flow with the same characteristics but uses different payloads and ports. The user uploading the trace can set the ports to specific values, e.g., the application's default port; otherwise, random ports are used.

Our experiments confirm that the replaying method of `trace-emulate` produces the same packet sizes, payloads, and ordering as the original trace. We omit detailed results.

Validating tests generated by trace-emulate. We validate that `trace-emulate` captures the essential characteristics that an ISP might use to identify an application flow in practice. While ISPs can, in theory, use arbitrarily complex mechanisms, in practice they are limited to using mechanisms that can scale to at least multiple Gbps. We are therefore interested in validating `trace-emulate` against *practical* detection mechanisms used by ISPs.

As one might imagine, ISPs use traffic classification solutions from third-party vendors such as Sandvine, BlueCoat, and Arbor Networks; most ISPs do not build their own system. Fortunately, pressure from privacy watchdogs compelled one of these vendors – Ipoque – to release the code it uses to inspect user traffic [22]. This release gives the research community, for the first time, access to production code that ISPs use to detect the application that a user is running.

The Ipoque code allows us to realistically validate `trace-emulate`. By inspecting the code we discover what applications are detected. We run the application and check whether the Ipoque detector detects the application from the packet flow. We then use `trace-emulate` to generate a Glasnost test for that application. We run the test and check whether the result is the same. If Ipoque detects our emulated flow as the target application, then we have successfully captured the essential characteristics that are necessary for detection by a commercial traffic classifier.

Ipoque's detector can identify traffic from more than 90 widely-used applications broadly classified as peer-to-peer, video streaming, instant messaging, online gaming, and other applications (email, web, etc.). It took us less than two hours to generate Glasnost tests for 10 representative applications in all five of the above categories. This included eMule, Gnutella, and BitTorrent (all P2P); YouTube (streaming video); World of Warcraft (online game); IRC (instant messaging); and HTTP, FTP and IMAP. For eMule and Gnutella, Ipoque separately identifies their control and data connections; consequently, we used `trace-emulate` to generate the corresponding two tests. That we were able to generate all tests in a matter of hours is a testament to the simplicity of `trace-emulate`.

In every single case, Ipoque identified the test generated by `trace-emulate` as the target application. To the extent Ipoque is representative of other similar vendors, we can claim that `trace-emulate` captures the essential flow characteristics for applications that do not encrypt traffic. However, without knowledge of how Ipoque detects applications from encrypted traffic, we cannot make any claims in that regard.

To convince ourselves that the Ipoque result holds in the real-world, we further validated `trace-emulate`

Application	Port-based	Content-based
BitTorrent	6881, down	down
eMule data	4662, down	down
Gnutella control	6346, down+up	down+up
Gnutella data	6346, down+up	down
HTTP	no	no
IMAP	no	no
SSH	no	no

Table 1: Results from running new Glasnost tests on a host connected via Kabel Deutschland. We identified instances of port-based and content-based traffic differentiation both the downstream (down) and upstream (up) directions.

against Kabel Deutschland, the biggest cable ISP in Germany. Kabel Deutschland targets P2P filesharing applications between 6pm and midnight [12]; their chosen vendor for traffic shaping equipment is unknown. In any event, since we know their policy, validating `trace-emulate` is straightforward. We run tests we generated for BitTorrent, eMule, Gnutella, HTTP, IMAP, and SSH from a Kabel Deutschland user, and check if Glasnost detects traffic differentiation.

Glasnost detected traffic differentiation for *each* of the P2P applications, and *none* of the non-P2P applications. In fact, by running the tests in both directions (downstream and upstream) and using different ports (default application port, random port), we were able to refine the policy published by Kabel Deutschland. Table 1 shows that P2P traffic is differentiated regardless of the port number used (i.e., based on the packet content). Next, we ran the HTTP, IMAP, and SSH tests on the ports typically used by the three P2P applications and found the flows achieved significant lower throughput. Running the same tests on random ports resulted in normal throughput. This is precisely what one might expect if Kabel Deutschland additionally uses port-based detection, which naturally has false-positives. Regardless of whether Kabel Deutschland sought to omit mention of side effects of their differentiation policy or we have identified a misconfiguration, our finding demonstrates the value of network transparency tools such as Glasnost.

7 Deployment Experiences

We deployed Glasnost publicly on the Internet on March 18th, 2008 and it has been operational ever since. It can be accessed at *http://broadband.mpi-sws.org/transparency/glasnost.php*. Initially, Glasnost was deployed on eight servers at MPI-SWS. Over the last year, the number of servers has grown to eighteen with the use of Measurement Lab (M-Lab) [16], an open platform for the deployment of Internet measurement tools to enhance network transparency. Eleven servers are in Europe, three on the west coast of the USA, and four on the east coast of the USA.

In the beginning, we chose to focus on one application as we developed our system and refined its techniques. We picked BitTorrent because it is widely suspected of being manipulated by ISPs [15]. However, our differentiation detection techniques are not specific to BitTorrent and can be applied to other applications as well. Because we have only recently deployed tests for other applications, an overwhelming majority of our data is from BitTorrent. We thus limit most of the discussion below to BitTorrent.

Details of deployed tests. In this paper, we present results for four BitTorrent tests deployed on Glasnost. These tests detect port- and content-based differentiation in the upstream as well as the downstream direction. Each test involves emulating BitTorrent and reference flows. For detecting content-based differentiation, we replace BitTorrent packet payloads with random bytes in the reference flows, while keeping other aspects identical. For detecting port-based differentiation, only the port of the reference flow is switched from a well-known BitTorrent port (e.g., 6881) to a neutral port that is not associated with any particular application (e.g., 10009). We emulate flows in both upstream and downstream directions to check for manipulation of both BitTorrent uploads and downloads. As described in Section 5, we configured Glasnost to offer a 6-minute long test to users with each flow running for 20 seconds. Also, each flow type is repeated once.

Usage. Between March 18th, 2008 and September 21st, 2009, 368,815 users[2] from 5,846 ISPs used Glasnost to test for traffic differentiation. We believe that our large user base is a result of our focus on lowering the barrier of use such that even lay users can use our system.

Figure 8 shows that our users have a wide geographical footprint. They come from North America (38%), Europe (36%), South America (11%), Asia (12%), Oceania (3%), and Africa (<1%).

Table 2 lists the top 20 access ISPs to which our users belonged. Users' IP addresses are mapped to ISPs using *whois* information from the Regional Internet Registries. We see that a large fraction of our users are from some of the largest residential ISPs in their respective countries, such as Comcast in the USA, Bell Canada in Canada, or BT in the UK.

[2]In this section, we use the terms tests, IP addresses, and users interchangeably. There are very few IP addresses from which we saw repeat tests and a vast majority of tests correspond to an unique IP address. The same end user may be associated with different IP addresses during the course of our study. By overlooking this, we may be over-counting the number of unique end users.

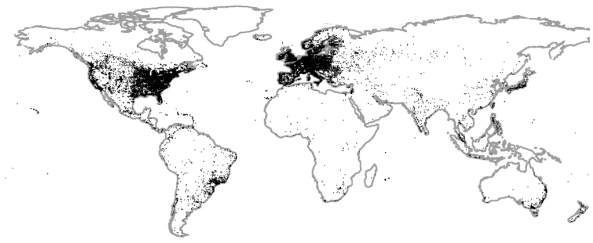


Figure 8: Location of Glasnost users.

ISP	Tests	ISP	Tests
Comcast (US)	29,464	BT (UK)	5,192
RoadRunner (US)	16,257	Chunghwa T. (TW)	5,084
AT&T (US)	10,884	Shaw (CA)	4,933
UPC (NL)	8,871	Brasil Telec. (BR)	4,862
Verizon (US)	7,611	Rogers (CA)	4,499
Cox (US)	4,194	Telefonica (BR)	4,408
Net Virtua (BR)	7,207	Telefonica (ES)	4,229
Telecom Italia (IT)	6,955	NTL (UK)	3,852
Charter (US)	3,634	Vivo (BR)	3,723
Bell Canada (CA)	5,233	GVT (BR)	3,723

Table 2: Top 20 ISPs based on the number of Glasnost tests conducted by their users.

7.1 Characterizing BitTorrent Differentiation

We now use the data collected during our deployment to characterize BitTorrent differentiation in the Internet. To our knowledge, such detailed characterization was not available before.

Figure 9 shows the percentage of users for whom we detected differentiation in at least one of the four tests that we widely deployed on Glasnost. Aside from a few weeks in the beginning when we did not have enough users, this percentage has stayed roughly constant around 10%. Thus, a non-negligible fraction of our testers are subject to differentiation.

We do not, however, claim that 10% of all Internet users experience differentiation. Glasnost users are self-selecting, and our data may be biased towards users that suspect their ISP to be differentiating against BitTorrent.

7.2 Understanding ISP behaviors

Our Glasnost deployment was so popular that we had hundreds of users from some of the largest ISPs worldwide. Aggregating results from all the users belonging to an ISP can provide an understanding of the extent to which the ISP differentiates traffic. Such ISP-wide perspectives are especially useful for policy makers and government regulators responsible for monitoring ISP behavior. Further, end users can compare the state of

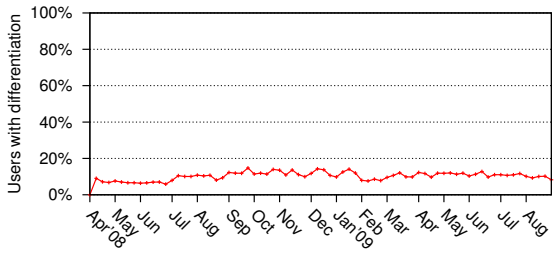


Figure 9: Percentage of tests in which we detected differentiation since March 2008.

differentiation across different ISPs to make a more informed choice when selecting their ISP.

We now turn our attention to understanding the policies of individual access ISPs. For this analysis, we map users to their access ISPs (using *whois*) and assume that the access ISP is responsible for any observed differentiation. While it is possible that the responsibility lies with a transit ISP along the path, differentiation is a more common practice amongst access ISPs [1, 5].

We limit the analysis in this section to the tests conducted in the two-month period that covers January and February 2009 because the differentiation behavior of an ISP can change over time. We select the two-month period for which we have the most data. Further, we consider only ISPs for which we have at least 100 tests in this time period. There are 140 such ISPs.

7.2.1 Basis for differentiation

Table 3 shows the list of the top-30 ISPs ranked based on the fraction of hosts that detected differentiation. Table 3 also shows how traffic is differentiated. More than half the ISPs differentiate only in the upstream direction and 7 ISPs only in the downstream direction. 20% of ISPs (e.g., Clearwire, TVCABO) differentiate in both directions. We also find that most differentiating ISPs use both content- and port-based differentiation. For only four ISPs (Free, GVT, Pipex, and Tiscali UK) do we observe an exclusive use of port-based differentiation (which is easier to evade). And only one ISP, Oi, uses content-based differentiation exclusively.

Our results show that Glasnost can shed light on how ISPs identify the traffic they differentiate.

7.2.2 Fraction of users impacted

ISPs that differentiate against BitTorrent traffic do not do so for every user. For each ISP in Table 3, Figure 10 shows the fraction of users that tested positive for differentiation. We see that in the median case only 21% of users are affected. Given our tests' low false posi-

ISP	Loc.	Upstream		Downstream	
		app	port	app	port
Bell Canada (D)	CA	×	×		
Brasil Telecom (D)	BR			×	×
BT (D)	UK	×	×		
Cablecom (C)	CH	×	×		
Canaca (D)	CA	×	×		
City Telecom (F)	HK	×	×	×	×
Clearwire (W)	US	×	×	×	×
Cogeco (C)	CA	×	×		
EastLink (C)	CA	×	×		
Free (D)	FR				×
GVT (D,F)	BR		×		
Kabel Deutschland (C)	DE	×	×	×	×
Magix (D)	SG			×	×
Oi (D)	BR			×	
ONO (C)	ES			×	×

ISP	Loc.	Upstream		Downstream	
		app	port	app	port
PCCW (D)	HK	×	×	×	×
Pipex (D)	UK		×		
Rogers (C)	CA	×	×		
Shaw (C)	CA	×	×		
TekSavvy (D)	CA	×	×		
Tele2 (D)	IT	×	×		
Telenet (D)	BE		×	×	
TFN (D)	TW			×	×
Tiscali Italia (D)	IT			×	×
Tiscali UK (D)	UK		×		
TM Net (D)	MY	×	×		
TVCABO (C)	PT	×	×	×	×
UPC NL (C)	NL	×	×		
UPC Poland (C)	PL	×	×		
UPC Romania (C)	RO	×	×		

Table 3: Top 30 ISPs based on the fraction of users that are affected by traffic differentiation during January and February 2009. The table shows if the flows are differentiated based on application content (app), TCP ports, or both. The letter in parenthesis gives the type of access network the ISP runs, i.e., DSL (D), Cable (C), Fiber-To-The-Home (F), and WiMax (W).

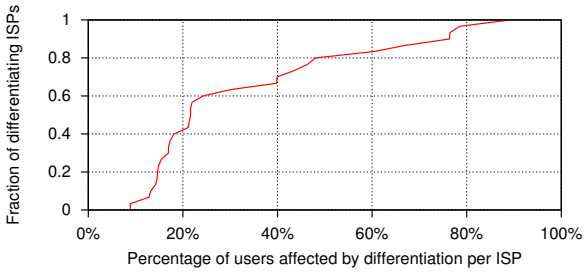


Figure 10: Typically, we detected traffic differentiation for only a fraction of an ISP's users.

tive and negative rates, this inconsistent impact within an ISP cannot be explained by inference errors alone.

Our data does not allow us to infer why only a fraction of users of an ISP experience traffic differentiation. There are many possible reasons. An ISP might choose to target only customers who generate a lot of P2P traffic, the traffic shapers might be deployed in only a portion of the ISP network, or an ISP might differentiate only during peak hours or periods of high load.

7.2.3 Dependence on time of day

One potential explanation for why only some users experience differentiation is that ISPs may differentiate only during peak hours, when the network is experiencing the greatest load. To investigate the dependence on time of day we divided our dataset into two time periods based on the local time of the user[3]. The *peak* period is 8pm–12am, and the *off-peak* period is 5–9am. These periods are strict subsets of the peak and off-peak durations for access ISPs [5, 14].

For each period we infer if an ISP differentiated traffic. Our analysis excludes ISPs that have fewer 100 measurements for either of the two time periods. This leaves us with 30 ISPs. We find that slightly more than half of these ISPs to differentiate during both peak and off-peak hours. The other ISPs, e.g., BT, Bell Canada, Kabel Deutschland, ONO, and Tiscali UK, restrict traffic differentiation to the peak period.

Our results in the last two sections show the importance of enabling end users to detect differentiation for themselves and at particular points in time. Many existing tools attempt to discover whether or not a ISP differentiates traffic [27, 30]. Since not all users of an ISP are affected by differentiation all the time, ISP-wide information alone is not sufficient for a user to determine if she experiences differentiation.

7.3 User feedback

Since our system became operational, we have received more than one hundred e-mails from users. The feedback is overwhelmingly positive, and it reveals two pieces of information. First, we find evidence of false negatives in our results. Around 6% of our emailers were skeptical when Glasnost did not discover traffic differentiation. They were convinced that their ISP differentiates, sometimes based on information their ISP publishes. If these users are right, their cases con-

[3]We used an IP-to-geolocation tool to infer the timezone of each user.

firm that our decision to minimize the false positive rate comes at the cost of false negatives. While we continue to investigate ways to reduce the false negative rate, we are pleased to report that no user has complained about the presence of a false positive.

Second, some emails requested Glasnost tests for other P2P applications such as eMule as well as non-P2P applications such as FTP, SSH, and HTTP. The constant stream of such requests motivated us to open the Glasnost platform and allow users to contribute new Glasnost tests. We describe this extension in the following section.

7.4 User-contributed Glasnost tests

It is not feasible for us to create Glasnost tests for each of the large number of applications and possible traffic differentiation policies that are of interest to users. Hence, we decided to allow users to create their own Glasnost tests using the `trace-emulate` tool that we described earlier. To create a new test, users need to capture a packet trace of their target application using `tcpdump` and then use `trace-emulate` to create a new Glasnost test from the trace. These new tests can be uploaded to our measurement servers using the Glasnost webpage. Our interface for creating new tests is targeted not at lay users, but at advanced users who have some familiarity with capturing network traces.

We have deployed this interface only recently, and we do not yet have a lot of experience with it. However, we asked a handful of our colleagues, who are doctoral students not associated with our project, to use the interface to create new Glasnost tests: they were able to create new tests quite easily.

8 Related Work

This section describes Glasnost in the context of existing work on traffic differentiation, trace replay, and measurement systems.

Traffic Differentiation. Three early studies investigated the prevalence of blocking for BitTorrent [10, 11] or for general traffic based on port numbers [4]. They found blocking to be relatively common. Our results show that gentler forms of differentiation are now much more prevalent than outright blocking.

Three recent efforts proposed techniques for detecting traffic differentiation. NetPolice [31] (previously named NVLens [30]) compares the aggregate loss rates of different flows to infer the presence of "network neutrality violations" in backbone ISPs. In contrast, Glasnost focuses on enabling individuals to detect whether they are subject to traffic differentiation.

NANO [27] uses causal inference to infer the presence of traffic performance degradation. NANO relies on a vast amount of passively collected traces from many users to infer if traversing a particular ISP leads to poorer performance for certain kinds of traffic. In contrast, Glasnost uses active measurements and a simple head-to-head comparison of two flows to quickly inform users whether they face traffic differentiation — without relying on other users. However, adding passive measurement techniques to Glasnost might enable it to detect time- or usage-dependent traffic differentiation.

DiffProbe [13] detects whether traffic differentiation based on active queue management (AQM), such as RED and weighted fair queueing, is deployed in the network path. DiffProbe complements Glasnost as it can detect differentiation that leads to small increase in latency and can identify the AQM technique used. If AQM affects application throughput, Glasnost can also detect this.

Trace replay. Monkey [6] is a TCP replay tool that takes a packet-level trace as input and generates a new trace with similar network-level properties, such as latency and bandwidth. More recent work [8] investigates ways to infer higher-level protocols from low-level packet traces. Our `trace-emulate` tool is an adaptation of such methods.

Measurement systems. Many researchers use network testbeds, such as PlanetLab [24], RON [3], and NIMI [23], to conduct measurement studies. Unlike Glasnost, these testbeds are designed explicitly for use by researchers. There are a number of tools deployed on M-Lab [16] with the goal of enhancing Internet transparency. Most of them are generic measurement tools that characterize certain features of the Internet

The DIMES project [9] is based on the SETI@home model. It uses volunteer-contributed hosts to run `traceroute` measurements that are used to map the connectivity of edge networks. The two systems, DIMES and Glasnost, offer an interesting (if unfair due to different goals) comparison of user models. DIMES relies on the ability to run arbitrary code on users' computers. It was deployed over four years ago and has about 8,000 users.

Finally, Netalyzr [17] is a web-based measurement tool that mostly focuses on the detection of networking problems. Like Glasnost, it targets lay users with an easy-to-use interface and allows them to detect, for instance, manipulation of web content by a HTTP proxy in the path or blocking of traffic on some prominent ports.

9 Conclusion

We described Glasnost, a system that we deployed more than a year ago to let ordinary users detect traffic dif-

ferentiation along their paths. More than 350,000 users from over 5,800 ISPs worldwide have used it to detect BitTorrent differentiation. We believe that our focus on making it easy for lay users to use the system and to understand its results have led to its success. Using the data gathered by Glasnost, we also presented what to our knowledge is the first detailed analysis of BitTorrent differentiation practices in the Internet. The data collected by Glasnost is available through M-Lab [16].

Over the past year, we have encountered many researchers who were skeptical about the benefits of measuring traffic differentiation. Even some of this paper's authors were initially skeptical. A common argument is that, since traffic differentiation is attracting so much attention from industry and the government, the permissible practices would soon be standardized and apparent. The skeptics might or might not be right. But the popularity of Glasnost and the positive feedback shows that many users are curious about the behavior of their Internet paths. Indeed, Glasnost's impact goes beyond traffic differentiation in our view. Its design shows one effective way to build and deploy a measurement system that satisfies such curiosities and makes the network more transparent to its users.

10 Acknowledgments

We thank Andreas Haeberlen and Alan Mislove for their contributions during the early stages of the Glasnost project. We also thank our shepherd Nick Feamster, and the anonymous reviewers for detailed feedback on this paper. Finally, we thank the people and the organizations supporting the M-Lab platform for hosting Glasnost on M-Lab servers.

References

[1] Comments of Comcast Corporation before the FCC. http://fjallfoss.fcc.gov/prod/ecfs/retrieve.cgi?native_or_pdf=pdf&id_document=6519840991.

[2] The Global Broadband Speed Test. http://www.speedtest.net.

[3] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. Resilient Overlay Networks. In *Proc. of SOSP*, 2001.

[4] R. Beverly, S. Bauer, and A. Berger. The Internet's Not a Big Truck: Toward Quantifying Network Neutrality. In *Proc. of the Passive and Active Measurement Conference (PAM)*, 2007.

[5] Canadian Radio-television and Telecommunications Commission. Review of the Internet traffic management practices of Internet service providers. http://crtc.gc.ca/PartVII/eng/2008/8646/c12_200815400.htm.

[6] Y.-C. Cheng, U. Hoelzle, N. Cardwell, S. Savage, and G. M. Voelker. Monkey See, Monkey Do: A Tool for TCP Tracing and Replaying. In *Proc. of the USENIX Technical Conference*, 2004.

[7] Comcast: Description of planned network management practices. http://downloads.comcast.net/docs/Attachment_B_Future_Practices.pdf.

[8] W. Cui, M. Peinado, K. Chen, H. J. Wang, and L. Irun-Briz. Tupni: automatic reverse engineering of input formats. In *Proc. of CCS*, 2008.

[9] The DIMES Project. http://www.netdimes.org/.

[10] M. Dischinger, A. Mislove, A. Haeberlen, and K. P. Gummadi. Detecting BitTorrent Blocking. In *Proc. of IMC*, 2008.

[11] EFF. "Test Your ISP" Project. http://www.eff.org/testyourisp.

[12] J. Röttgers, Focus Online, 6.03.2008. Internetanbieter bremst Tauschbörsen aus. http://www.focus.de/digital/internet/kabel-deutschland_aid_264070.html.

[13] P. Kanuparthy and C. Dovrolis. DiffProbe: Detecting ISP Service Discrimination. In *Proc. of INFOCOM*, 2010.

[14] N. Laoutaris and P. Rodriguez. Good Things Come to Those Who (Can) Wait – or how to handle Delay Tolerant traffic and make peace on the Internet. In *Proc. of HotNets*, 2008.

[15] List of ISPs suspected to traffic shape BitTorrent. http://www.azureuswiki.com/index.php/Bad_ISPs.

[16] Measurement Lab. http://www.measurementlab.net.

[17] The ICSI Netalyzr. http://netalyzr.icsi.berkeley.edu.

[18] New York Times. 'Neutrality' Is New Challenge for Internet Pioneer, September 2006. http://nytimes.com/2006/09/27/technology/circuits/27neut.html.

[19] New York Times. Comcast: We're Delaying, Not Blocking, BitTorrent Traffic, October 2007. http://bits.blogs.nytimes.com/2007/10/22/comcast-were-delaying-not-blocking-bittorrent-traffic.

[20] New York Times. F.T.C. Urges Caution on Net Neutrality, June 2007. http://www.nytimes.com/2007/06/28/technology/28net.html.

[21] New York Times. F.C.C. Chairman Favors Penalty on Comcast, July 2008. http://www.nytimes.com/2008/07/11/technology/11fcc.html.

[22] OpenDPI. http://www.opendpi.org.

[23] V. Paxson, A. K. Adams, and M. Mathis. Experiences with NIMI. In *Proc. of the SAINT Workshop*, 2002.

[24] PlanetLab. http://www.planet-lab.org/.

[25] N. Spring, D. Wetherall, and T. Anderson. Scriptroute: A Public Internet Measurement Facility. In *Proc. of USITS*, 2003.

[26] Systems Research Lab, University of Colorado at Boulder. Broadband network management. http://systems.cs.colorado.edu/mediawiki/index.php/Broadband_Network_Management.

[27] M. B. Tariq, M. Motiwala, N. Feamster, and M. Ammar. Detecting Network Neutrality Violations with Causal Inference. In *Proc. of the CoNEXT Conference*, 2009.

[28] VELOCIX: New Generation Content Delivery Network. http://www.velocix.com.

[29] Vuze Network Status Monitor. http://azureus.sf.net/plugin_details.php?plugin=aznetmon.

[30] Y. Zhang, Z. M. Mao, and M. Zhang. Ascertaining the Reality of Network Neutrality Violation in Backbone ISPs. In *Proc. of ACM HotNets-VII Workshop*, 2008.

[31] Y. Zhang, Z. M. Mao, and M. Zhang. Detecting Traffic Differentiation in Backbone ISPs with NetPolice. In *Proc. of the Internet Measurement Conference (IMC)*, 2009.