Detecting Traffic Differentiation in Backbone ISPs with NetPolice

Ying Zhang
University of Michigan
2260 Hayward Street
Ann Arbor, MI, USA
wingying@umich.edu

Zhuoqing Morley Mao
University of Michigan
2260 Hayward Street
Ann Arbor, MI, USA
zmao@umich.edu

Ming Zhang
Microsoft Research
One Microsoft Way
Redmond, WA, USA
mzh@microsoft.com

ABSTRACT

Traffic differentiations are known to be found at the edge of the Internet in broadband ISPs and wireless carriers [13, 2]. The ability to detect traffic differentiations is essential for customers to develop effective strategies for improving their application performance. We build a system, called NetPolice, that enables detection of content- and routing-based differentiations in backbone ISPs. NetPolice is easy to deploy since it only relies on loss measurement launched from end hosts. The key challenges in building NetPolice include selecting an appropriate set of probing destinations and ensuring the robustness of detection results to measurement noise.

We use NetPolice to study 18 large ISPs spanning 3 major continents over 10 weeks in 2008. Our work provides concrete evidence of traffic differentiations based on application types and neighbor ASes. We identify 4 ISPs that exhibit large degree of differentiation on 4 applications and 10 ISPs that perform previous-AS hop based differentiation, resulting in up to 5% actual loss rate differences. The significance of differences increases with network load. Some ISPs simply differentiate traffic based on port numbers irrespective of packet payload and the differentiation policies may only be partially deployed within their networks. We also find strong correlation between performance differences and Type-of-Service value differences in the traffic.

Categories and Subject Descriptors: C.2.5 COMPUTER-COMMUNICATION NETWORKS: Local and Wide-Area Networks

General Terms: Measurement, Experimentation

Keywords: Internet measurement, Traffic differentiation

1. INTRODUCTION

Since its early days, Internet is designed under the end-to-end principle which argues for intelligent end systems and a "simple" network. Under this principle, networks deliver traffic with best effort and do not treat traffic preferentially based on various properties such as IP address, port number, or packet content [27]. In recent years, a variety of new applications have emerged and proliferated on the Internet. Some require high bandwidth (*e.g.*, peer-to-peer file sharing and video streaming) while others require low latency and loss rate (*e.g.*, voice-over-IP and online gaming). Such trend has inspired ISPs to perform various types of traffic shaping to manage network resource usage and introduce tiered services to meet the requirements of different customers and applications.

Residential broadband ISPs are known to treat traffic differently, *e.g.*, by limiting the bandwidth usage of peer-to-peer file sharing applications [13]. Cellular network carriers have also been reported to restrict the usage of video streaming services to preserve their limited wireless spectrum [2]. Researchers have proposed various techniques for detecting traffic differentiation. Beverly *et al.* presented one of the first measurement studies of port blocking behavior from the edge of the Internet [8]. POPI is another tool for determining router traffic differentiation policy based on port numbers via end-host measurements [19]. More recently, Dischinger *et al.* developed tests for detecting whether broadband ISPs rate-limit or block BitTorrent traffic [13]. Besides these active measurement techniques, Tariq *et al.* proposed to identify differentiation by applying statistical method to passive measurements from end hosts [32]. Yet so far, there has been no detailed and comprehensive study on the current practice of traffic differentiation inside the Internet core. Traffic differentiation in the core arguably has a much wider scope of impact, as such policies affect much more traffic compared to the policies near the edge of the Internet.

In this paper, we consider the problem of detecting traffic differentiation in backbone ISPs. Different types of traffic may experience different performance within the same ISP network due to various reasons. An ISP may "passively" throttle the traffic from a neighbor (*e.g.*, a peer) by carrying the traffic over a low-capacity link, since it may not have the economic incentive to provision or upgrade the link [4]. It may also "actively" prevent the traffic of an application (*e.g.*, BitTorrent) from disrupting other traffic via weighted fair queuing when the network is congested.

Regardless of the actual reasons behind the performance differences, it is important for customers to be able to reason about the behaviors of their ISPs. The ability to detect traffic differentiation enables customers to develop appropriate strategies for improving their application performance. For instance, large content providers strive to ensure their Internet applications outperform those offered by their competitors. If a content provider knows the average loss rate of its traffic traversing a particular ISP is twice that of its competitor, it may want to negotiate better service level agreements (SLA) with that ISP. Small customers will also benefit from such differentiation information. For instance, they may change port numbers or encrypt packets to circumvent content-based differentiation employed by their ISP.

Most ISPs do not reveal the details of their network policies and configurations. Realizing this problem, we aim to develop an end-host based system that can detect traffic differentiation without any

IMC'09, November 4–6, 2009, Chicago, Illinois, USA.

Type	Examples
Packet headers	source & destination port numbers, protocol type
Application layer info	application headers (*e.g.*, HTTP header, BitTorrent header), application payload
Traffic behavior	flow rate, flow duration, packet size, packet interval
Routing info	previous-hop AS, next-hop AS, source & destination IP addresses
Available resources	queue length, link utilization, router load & memory

Table 1: ***Information commonly used for traffic differentiation.***

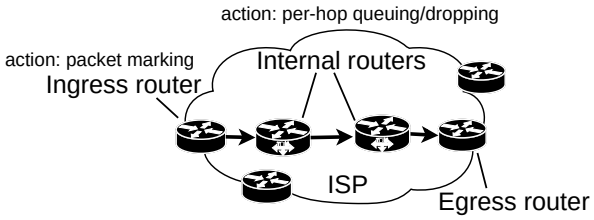

Figure 1: ***One common implementation of differentiation.***

ISP cooperation. Such a system is not only easily deployable but also applicable to many different ISPs. To build such a system, we face two key challenges: i) unlike in the case of broadband ISPs, most end hosts are not directly connected to backbone ISPs. We need to intelligently select probing destinations to cover the relevant internal paths of backbone ISPs while complying with the requirement of limited network and CPU resources on end hosts; ii) measurement data taken from end host is susceptible to various types of noise on the host or in the network. We need to ensure our detection results are not distorted by noise.

NetPolice is the first operational system that can detect traffic differentiation in backbone ISPs by accurately and scalably monitoring packet loss behavior. It relies on an intelligent path selection scheme to detect both content- and routing-based differentiation while systematically balancing path coverage and probing overhead. It leverages statistical hypothesis tests to identify significant loss rate differences between different types of traffic measured along the same ISP internal paths after discounting the effects of measurement noise. Furthermore, it uses a novel technique for cross-validating the statistical test results and the Type-of-Service (TOS) value set by ISPs.

By studying 18 large ISPs spanning 3 major continents over a period of 10 weeks in 2008, NetPolice provides concrete evidence of traffic differentiation based on application types and neighbor ASes. We identified 4 ISPs that exhibit large degree of differentiation on VoIP, BitTorrent, PPLive, and SMTP traffic compared to HTTP traffic. We also identified 10 ISPs that treat traffic differently based on its previous-hop ASes, reflecting different business contracts. The significance of differentiation increases with network load, suggesting that differentiation is likely to be triggered by resource competition. The actual loss rate difference between certain pairs of applications or previous-hop ASes can exceed 5%, large enough to impair the performance of many TCP-based applications. Interestingly, we find a few ISPs simply rely on port numbers to perform traffic differentiation irrespective of actual payload. These ISPs may apply differentiation policies only to a subset of routers in their networks. We further validate our detection results on paths where we have two-ended control.

2. TRAFFIC DIFFERENTIATION

An ISP may use various information in traffic and routers to construct differentiation policies. Table 1 enumerates a list of such potential factors [35]. First, an ISP may provide differentiated services based on the application type for security or business reasons. It is well-known that broadband ISPs drop certain SMTP traffic to fight spams and throttle P2P traffic to manage bandwidth usage. Application types can be determined from packet header fields or application layer information [24]. Even with encrypted traffic, there are sophisticated techniques that can infer application types by identifying certain traffic behavior [34]. Second, an ISP can differentiate traffic according to routing information, reflecting distinct business contracts with its customers and peers. An ISP may assign high priority to traffic from customers who pay for premium services or assign low priority to traffic from peers. This type of differentiation can be applied based on the previous-hop or next-hop ASes, which can be easily extracted from packet headers and routing state. Third, an ISP may enforce differentiation policies according to available resources. Using the link utilization information readily available from SNMP [11], it may slow down traffic with low priority to preserve sufficient bandwidth for other traffic.

It is feasible to implement traffic differentiation in a backbone network with many high-speed links. Today's router already supports various queuing mechanisms to fulfill the need of traffic engineering, quality of service, and security guarantees. Figure 1 illustrates a common architecture for implementing differentiation within a backbone ISP. The ingress routers perform traffic classification by marking packets according to packet header fields and routing information, such as port numbers and previous-hop ASes. The marking is usually applied to the Type-of-Service (TOS) field in the IP header. The internal routers perform traffic shaping according to the TOS value in the packets [17]. There are various queuing and dropping mechanisms that provide different levels of service to traffic, *e.g.*, priority queuing, proportional share scheduling, and policing [10]. These mechanisms differ in details of how and when differentiation is carried out. In §6.7, we demonstrate traffic differentiation can be easily implemented on today's commercial routers in testbed experiments.

Other than the router marking-based mechanisms using packet header information, ISPs may perform deep packet inspection (DPI) [14] to classify application types according to packet content. Some DPI devices can perform pattern matching in packet payload with hardware support for 100 Gps links [1, 12]. Because DPI devices can be quite expensive, they are usually deployed only at selected locations.

In this work, we examine all types of differentiation listed in Table 1 except for the one based on traffic behavior (Table 1 row 4) due to limitations of end-host based probing (§3.2). In fact, behavior-based differentiation could be expensive to implement by ISPs due to the required per-flow state information and potentially high false positives. Our goal of detecting these four types of differentiation guides the design of path selection and probe packet composition in NetPolice. By providing concrete evidence of differentiation, we hope to stimulate more research to fully understand possible differentiation policies in backbone ISPs.

3. METHODOLOGY

NetPolice detects traffic differentiation inside a particular ISP by launching probes from a distributed set of end systems. For this purpose, we have to decide what paths to measure, how to measure

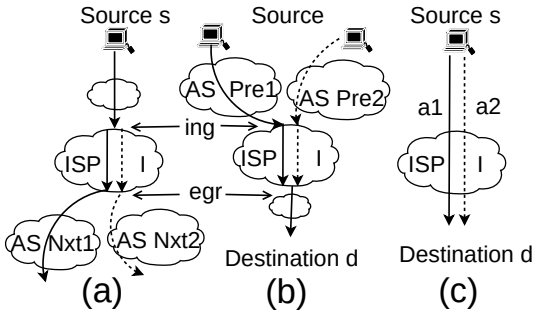

Figure 2: ***Detecting various types of differentiation with end-host based probing***

each path, and how to identify differentiation based on measurement results. We address these three issues below.

3.1 Path selection

NetPolice is designed to detect traffic differentiation based on packet headers, application layer information, and routing information (described in Table 1). Figure 2 illustrates how NetPolice uses measurements from end systems to identify differentiation in *ISP I*. In Figure 2(a), an end host probes two paths to different destinations, sharing the same ingress and egress within *ISP I*, but diverging into two distinct next-hop ASes after leaving the egress. By comparing the performance of the two internal paths between the ingress and egress of *ISP I*, NetPolice can determine whether *ISP I* treats traffic differently based on the next-hop ASes. Similarly, Figure 2(b) shows how NetPolice detects differentiation based on previous-hop ASes. In Figure 2(c), an end host probes a path that traverses the same ingress and egress of *ISP I* to the same destination. By comparing the internal path performance measured by packets of different applications (*e.g.,* a_1 *vs.* a_2), NetPolice can detect differentiation based on content, such as packet headers and application layer information. We leave the discussion of resource-based differentiation to §6.5.

To detect traffic differentiation inside an ISP, we devise an intelligent path selection strategy to ensure good coverage and low overhead. On the one hand, a backbone ISP typically consists of multiple PoPs (Points of Presence) at many geographic locations. We want to cover as many distinct PoP pairs as possible in order to quantify the scope of traffic differentiation policies inside the ISP. On the other hand, NetPolice relies on end hosts to perform measurements. While this makes NetPolice easily deployable and applicable to different ISPs, we must aggressively reduce the measurement overhead to comply with the requirement of limited CPU and network resources at each host.

Given a target ISP, a list of probing sources, and all the destination prefixes on the Internet, a naive approach is to probe all the prefixes from all the sources. This may lead to both wasteful probes that do not traverse the target ISP and redundant probes that traverse the same internal paths multiple times. To avoid these two problems, we frame the path selection problem as follows.

1. Each three-tuple $(src, ingress, egress)$ is traversed at least R times by probes to different destinations.

2. Each three-tuple $(ingress, egress, dst)$ is traversed at least R times by probes from different sources;

3. A probing source does not send more than m probes.

Here, src is a probing source, dst is a destination prefix, and $ingress$ and $egress$ are the PoPs in the ISP.

Conditions 1 and 2 allow us to detect differentiation based on routing information, *e.g.,* previous-hop and next-hop ASes. We can also detect content-based differentiation by probing the same path with packets of different applications. R is a tunable *redundancy factor* that determines the tradeoff between probing overhead and coverage. A larger R will increase not only the chance of detecting routing-based differentiation but also the amount of probing traffic. Condition 3 restricts the total number of probes from each source. Because a source needs to probe each three-tuple many times for reliably detecting differentiation (explained in §3.3), this condition ensures it will not take too much time for a source to complete all the probes.

This problem is an instance of the set covering/packing problem [18, 23]: given multiple sets over a universe of elements, pick a subset of input sets such that each element is included at least R times (covering constraint), and no element is included more than m times (packing constraint). In our case, the input sets are the probes between source-destination pairs, and the elements are the probers and the three-tuples of $(src, ingress, egress)$ and $(ingress, egress, dst)$. A probe typically contains all three element types. This formulation enables us to perform both redundancy elimination and probing load assignment systematically. While this problem is NP-hard, we use a greedy based approximation: at each step, we select the probe that covers the most uncovered elements without exceeding the probing threshold m. This process continues until all the elements are covered at least R times.

3.2 Loss rate measurement

NetPolice focuses on detecting traffic differentiation that degrades application performance. Currently, it measures loss rate in order to detect differentiation schemes based on rate-limiting in backbone ISPs. We may extend it to measure other performance metrics, *e.g.,* delay, by applying the probing techniques developed in Tulip [22]. We may also extend it to detect the differentiation schemes used by broadband ISPs, *e.g.,* traffic blocking and TCP SYN/RST [13].

Given a path, NetPolice measures the loss rate as follows. First, to reduce probing overhead, NetPolice only probes the hops that map to an ingress or an egress of a target ISP instead of all the hops along the path, given that we are only interested in detecting differentiation inside the ISP. We will describe the details of identifying the ingress and egress of an ISP in §4. Second, to measure the loss rate to a particular hop, NetPolice sends probe packets with pre-computed TTL (Time-to-Live) value which will trigger ICMP time exceeded response from that hop. In essence, these packets are similar to traceroute probes. Although an ICMP packet may be forwarded on a slow path, it will not affect the loss measurement as long as the packet is not dropped.

Because packet loss may occur in either direction, we use large probe packets to ensure the measured loss is mostly due to forward path loss. The assumption is that large probe packets are more likely to be dropped than small ICMP packets on the reverse path. This has also been adopted in previous work [22, 20]. To avoid triggering ICMP rate limiting, NetPolice probes each hop once per second for 200 times, allowing us to detect loss rate as small as 0.5%. Probing each hop more times increases the sensitivity of loss rate detection but also the probing overhead. We subtract the measured loss rate of the ingress from that of the egress to obtain the loss rate of the internal path. In §5, we describe how to mitigate the impact of reverse path loss and ICMP rate-limiting on our loss rate measurements.

To detect content-based differentiation, we measure loss rate of an internal path using different application traffic. We select five representative applications with distinct QoS (Quality of Service) requirements: HTTP (default port 80), BitTorrent (P2P file sharing, port 6881), SMTP (email, port 25), PPLive (video streaming, port 4004), and VoIP (port 5060). Except for HTTP, the remaining four applications are selected based on how likely they will be treated differently by backbone ISPs. HTTP, one of the most commonly-used application, is used as the baseline to compare performance with other applications. ISPs may slow down BitTorrent and PPLive traffic due their high volumes. Similarly, ISPs may disfavor SMTP traffic due to email spam concerns. We also test VoIP traffic because many ISPs provide their own VoIP service, raising incentives for preferential treatment.

We construct probe packets with application-specific content captured from real application traces. This eliminates any need to understand the protocols of proprietary applications, such as PPLive or VoIP. To enable fair comparison between the loss rate of different applications, all probe packets are chosen to have the same size.

Because NetPolice relies on TTL-based probes to measure path performance, it cannot fully mimic the temporal behavior of real application traffic. If it probes as fast as the packet rate of applications, it may easily trigger ICMP rate-limiting on routers. An alternative is to run applications on end hosts and detect differentiation based on observed application performance [32]. However, such approach requires the participation of a large number of hosts to cover the internal paths of the backbone ISPs of interest. Furthermore, without directly probing routers, it is challenging to infer the performance of ISP internal paths purely based on end-to-end measurements.

3.3 Differentiation detection

NetPolice detects differentiation by observing the performance differences measured along the same ISP internal path using different types of probe traffic. Due to load variations on a path, the same type of probes may experience different loss rates at different times. This suggests we need to take a sufficiently large number of loss rate measurements to ensure that the observed performance differences accurately reflect how an ISP treats different types of traffic.

We first introduce a few notions before describing the details of our differentiation detection scheme. For a target ISP I, we define $l_{\{s,d,a,t\}}$ to be a loss rate sample measured along an internal path of ISP I from a probing source s to a destination d, using probes of application a at time t. We use the term set to denote a set of samples that are measured with a particular type of probes. For example, $set_{\{s,d,a\}}$ includes all the samples measured along a path from s to d using probes of application a. Similarly, $set_{\{pre,ing,d,a\}}$ includes all the samples measured along the paths traversing previous-hop AS pre and ingress ing to destination d, using probes of application a.

Our basic assumption is that the loss rate samples in a set follow a particular underlying distribution. This distribution reflects how an ISP treats the corresponding type of traffic. We can then detect differentiation between two types of traffic by comparing the two corresponding distributions. We use $pair_{\{s,d,a1,a2\}}$ to denote two candidate $set_{\{s,d,a1\}}$ and $set_{\{s,d,a2\}}$ (see Figure 2(c)). We can compare the two distributions of an *application* $pair_{\{s,d,a1,a2\}}$ to detect content-based differentiation between $a1$ and $a2$. Similarly, we use $pair_{\{pre1,pre2,ing,d,a\}}$ to denote two candidate $set_{\{pre1,ing,d,a\}}$ and $set_{\{pre2,ing,d,a\}}$ (see Figure 2(b)). We can compare the two distributions of an *AS* $pair_{\{pre1,pre2,ing,d,a\}}$ to detect previous-hop AS based differentiation between $pre1$ and $pre2$ at ingress ing. As long as the underlying distributions are stable and the two candidate sets include enough samples, we should be able to reliably detect differentiation between two types of traffic.

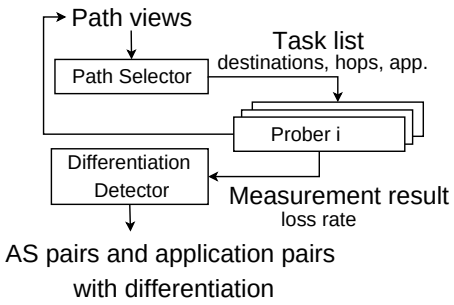

Figure 3: ***The NetPolice system***

Given a pair of input sets, we apply statistical hypothesis tests to determine if there are significant differences between them. Several commonly-used hypothesis tests exist to compute the statistical significance of differences between two input sets. Since the distribution of the loss rate samples in an input set is unknown, we choose the Kolmogorov-Smirnov (K-S) test [31] which makes no assumption about the input sample distribution. The K-S test compares the distance of the two empirical cumulative distribution functions F_1 and F_2 corresponding to the two input sets. It computes the Kolmogorov-Smirnov statistic $D_{1,2} = \sup_x |F_1(x) - F_2(x)|$, where sup is the supremum, under the null hypothesis that the two sets of samples are collected from the same distribution. The null hypothesis test is rejected at significance level α if $\sqrt{\frac{n_1 n_2}{n_1+n_2}} D_{1,2} > K_\alpha$. Here n_1 and n_2 denote the size of the input sets and K_α is the critical value in the K-S statistic table.

As we just discussed, the validity of a K-S test statistic depends not only on whether the distributions of the input sets are stable but also on whether the input sets contain enough samples. We use Jackknife [33], a commonly-used non-parametric resampling method, to verify the validity of the K-S test statistic. The idea is to randomly select half of the samples from the two original input sets and apply the K-S test on the two new subsets of samples. This process is repeated r times. If the results of over $\beta\%$ of the r new K-S tests are the same as that of the original test, we conclude that the original K-S test statistic is valid. We use $r = 400$, $\alpha = 95\%$, and $\beta = 95$ in this paper to ensure 95% confidence interval. In §6.1 and §6.3, we will show that the choice of these parameters makes our differentiation detection results robust against noise in loss rate samples.

4. IMPLEMENTATION

The implementation of NetPolice is illustrated in Figure 3. It has three major components:

Path selector takes *path views* as input and compute a task list of probing destinations for each prober. The path views are the traceroute measurements conducted from all the probers to all the destination prefixes on the Internet. The path selector uses the path views to learn the ingress and egress of the target ISPs that each path traverses. It identifies the ingress and egress by attempting to map each IP hop to an ISP and a PoP based on the DNS name of the IP hop [29]. We extend the set of naming rules in *undns* [29] to increase the number of names that can be successfully mapped. The

ISP	PoP	Ingress-Egress	PoP-AS
ISP_1	49	716	337
ISP_2	139	2125	806
ISP_3	57	1498	1170
ISP_4	25	232	102
ISP_5	46	501	351
ISP_6	71	1750	653
ISP_7	59	677	371
ISP_8	38	502	195
ISP_9	112	822	430
ISP_{10}	45	539	176
ISP_{11}	32	419	119
ISP_{12}	30	267	138
ISP_{13}	64	115	195
ISP_{14}	23	303	82
ISP_{15}	19	137	66
ISP_{16}	44	538	208
ISP_{17}	69	1787	152
ISP_{18}	44	261	316

Table 2: ***18 ISPs being studied***

path views are updated daily to keep up with the evolution of ISP topologies. Some path views may become temporarily out-of-date due to routing changes. We detect routing changes by observing the mismatch between the IP hops seen in loss measurements and the corresponding hops in path views. We simply discard all the loss rate samples affected by routing changes.

The path selector implements the greedy algorithm described in §3.1. Note that path selection is performed for multiple target ISPs simultaneously. This significantly reduces probing overhead by leveraging the fact that a single probe often traverses multiple target ISPs, allowing us to cover the same set of three-tuple elements (defined in §3.1) with fewer probes compared to probing each ISP separately. For each of the target ISPs traversed by a probe, we measure its internal loss rate between an ingress and egress following the method described in §3.2.

Probers run on a distributed set of end hosts, probing all the destinations in their task list periodically. After completing each round of probing to all the destinations, the probers send their measurement results to the differentiation detector for further processing. Probing is conducted with a customized version of traceroute that probes multiple hops of a path and multiple destinations in parallel. The probe packets are constructed to reduce the probability that different probe packets from the same source to the same destination take different IP-level paths due to load-balancing [6].

Differentiation detector first filters the noise in the measurement results due to overloaded probers or reverse path losses. It then tries to detect differentiation based on content, previous-hop AS, or next-hop AS, following the process described in §3.3. Finally, it performs detailed analysis on differentiation policies, such as what input information they use, whether they are affected by network load, and how significant their impact is.

We deployed NetPolice on the PlanetLab testbed [26]. It uses all the PlanetLab hosts across about 200 distinct sites. Each round of probing takes roughly two hours to complete. The results in the paper are based on 74 days of data collected during a period between August 2008 and October 2008. Each set includes around 1,000 loss rate samples. We run multiple instances of NetPolice to take measurements of the five applications described in §3.2 in parallel. We randomize the order of destinations to probe in each round to reduce the chance of a path being simultaneously measured by multiple instances. We studied 18 large ISPs covering major continents including North America, Europe, and Australia, consisting of 9 Tier-1 ISPs, 8 Tier-2 ISPs, and 1 Tier-3 ISP. Table 2 shows NetPolice has a decent coverage of internal paths and interconnections, traversing 115 to 2125 ingress-egress pairs and 66 to 1170 PoP-AS pairs for each ISP. A PoP-AS pair represents an interconnection between a neighbor AS and the target ISP at the corresponding PoP.

5. REDUCING NOISE EFFECTS

Loss rate measurements taken by end-hosts are susceptible to various types of noise on the host and in the network. As mentioned in §3.2, the inaccuracy of loss rate measurements is likely to be caused by three main factors: i) overloaded prober; ii) ICMP rate limiting at router; and iii) loss on reverse path. In this section, we investigate the effects of these three factors and develop techniques to mitigate their impact. We emphasize that these techniques cannot completely eliminate all the noise. However, as shown in the next section, the remaining noise will have little impact on the differentiation detection results.

Many ISPs perform load balancing using equal-cost multi-paths (ECMP) to ensure effective utilization of network resources [5]. Per-flow load balancing is usually performed based on the five tuple $(srcip, dstip, srcpt, dstpt, proto)$. Thus, different application packets, *e.g.*, BitTorrent and HTTP, may take different internal IP-level paths between the same ingress and egress, given their different destination ports (*e.g.*, 6881 *vs.* 80). We do not observe any per-packet load balancing in the 18 ISPs being studied. In this section, we carefully design experiments to ensure our differentiation detection is not affected by potential performance difference of ECMP paths.

5.1 Overloaded prober

Previous work has shown measurement inaccuracies caused by resource contention, in particular CPU load, on probing hosts in PlanetLab experiments [28]. To deal with this problem, we continually monitor the CPU utilization on each prober by running the `top` command and compute the average CPU utilization using three instantaneous load samples in each minute. We can then investigate the relationship between CPU utilization and measured loss rate by temporally correlating these two types of samples. This allows us to identify and discard abnormal loss rate samples that could be affected by high CPU utilization.

To determine an appropriate cut-off threshold of high CPU utilization, we design the following controlled experiment to study the effects of CPU utilization on loss rate measurements. We select a pair of lightly-loaded PlanetLab machines at the same site. One machine acts as a "prober" to transmit one 1000-byte probe packet per second. The other machine acts as an "acker" to receive probe packets and return 40-byte ACKs. In essence, the "prober" behaves just like a real NetPolice prober that measures loss rate. We then run a computation-intensive program to gradually increase the CPU utilization on the "prober" while keeping the acker lightly loaded.

Figure 4 illustrates the relationship between CPU utilization and loss rate measured by the "prober". Because loss is unlikely to occur on the light-loaded acker or on the local area network between the "prober" and the acker, the measured loss rate is almost certainly due to the CPU load on the "prober." Clearly, the loss rate jumps up when the CPU utilization goes above 65%. We repeat this experiment on ten pairs of PlanetLab hosts across different sites and find the loss rates induced by CPU load are consistently smaller than 0.2% when CPU utilization is under 65%. In §6.6, we will show that such loss rates are negligible compared to the

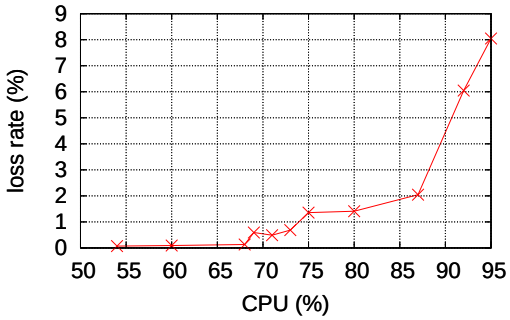

Figure 4: ***Impact of CPU utilization on loss rate.***

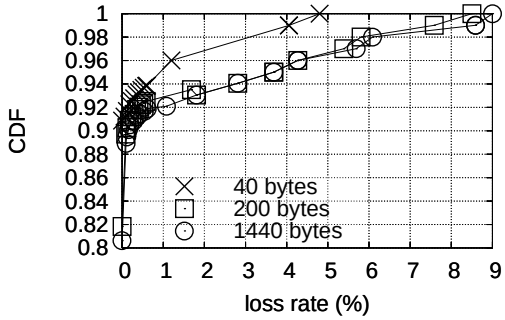

Figure 6: ***Impact of probe packet size on loss rate.***

observed loss rate differences due to traffic differentiation. By applying the 65% cutoff threshold on CPU utilization, 15% of the samples in our data are discarded.

5.2 ICMP rate limiting

ICMP rate limiting is often configured on a per-router basis to prevent router overload. If triggered, it may significantly inflate the measured loss rate. To prevent this, we deliberately keep a large probing interval, *e.g.*, only one probe packet is sent on a given path per second. We use the following experiments to confirm that this probing interval is large enough to avoid triggering ICMP rate limiting.

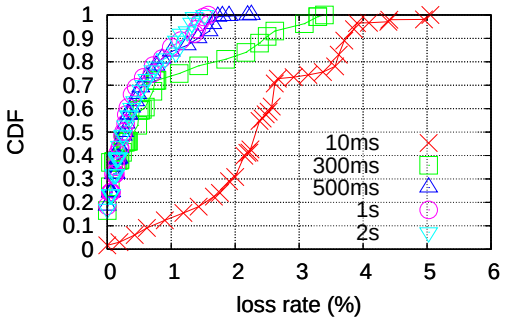

Figure 5: ***Impact of probing interval on loss rate.***

We conducted five sets of experiments by measuring the loss rate of all the internal paths of the 18 target ISPs from all the probers. We gradually increase the probing interval for each set of experiments from 10ms to 2s. The smaller the interval is, the more likely a router along a path may rate-limit the ICMP time-exceeded reply. As shown in Figure 5, the measured loss rates on 30% of the paths increase significantly when the probing interval changes from 500ms to 300ms. This indicates the rate-limiting threshold of the routers along those paths is between 300ms and 500ms. The loss rate curves of 1s, 2s, and 500ms are almost indistinguishable, suggesting that the 1-second interval is sufficiently large to avoid triggering ICMP rate-limiting on most routers. Otherwise, we would have observed the loss rates measured by the 2-second interval to be much smaller than those measured by the 1-second interval.

5.3 Loss on reverse path

NetPolice relies on single-ended probes to measure loss rate. The measured loss rate can be inflated due to reverse path loss. Since large packets are more likely to be dropped [20], we use 1000-byte probe packets to ensure the measured loss is mostly on forward paths. We study the effect of packet size on measured loss rate using controlled experiments. We conducted three sets of experiments by measuring the loss rate of all the ISP internal paths using probe packets of 40 bytes, 200 bytes, and 1440 bytes. As shown in Figure 6, the measured loss rate increases with probe packet size. Since the size of the ICMP responses is always the same, this confirms that bigger probe packets are more likely to encounter losses on forward path. Nonetheless, the loss rates measured by 200-byte and 1440-byte packets are roughly the same, suggesting the effects of packet size on forward path loss diminish when packet size exceeds 200-byte.

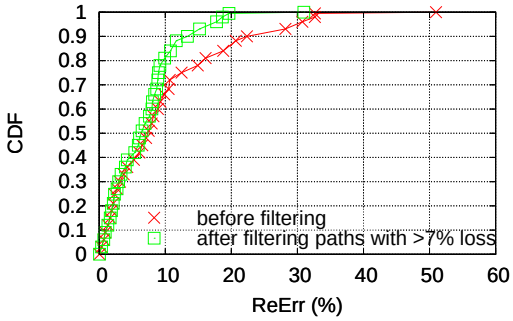

Figure 7: $ReErr$ ***before & after filtering.***

The loss rate measured by 40-byte probe packets is much smaller. In fact, we can use this loss rate as the upper bound of the loss rate on reverse path. To further limit the impact of reverse path loss, we compute $ReErr$ as the ratio of the loss rate measured by 40-byte packets and that by 1000-byte packets on each path. This $ReErr$ is a conservative estimate of the relative error of loss rate measurements induced by reverse path loss. Figure 7 shows that $ReErr$ is less than 10% on 70% of the paths. We find $ReErr$ tends to be large on paths with high loss rate, *e.g.*, $ReErr$ exceeds 10% on most of the paths with loss rate $>$7%. By discarding 6% of the paths with such abnormally high loss rate, $ReErr$ is within 10% on 80% of the remaining paths. In essence, we sacrifice path coverage a little for higher measurement accuracy.

5.4 Load balancing

Per-flow load balancing is observed extensively in our measurements, *e.g.*, BitTorrent traffic and HTTP traffic take different internal IP-level paths between 48% of the source-destination pairs. To eliminate the effect of load balancing, we take a conservative approach in detecting content-based differentiation. We first detect potential differentiation for each application $pair$ from the initial

ISP	App	Paths (%)	Δ_{tos}	TOS_δ	FP (%)
ISP_{12}	BitTorrent	3794 (19)	100	99	12 (0.06)
ISP_{12}	PPLive	825 (4.1)	100	85	24 (0.1)
ISP_2	VOIP	172 (3.2)	100	68	11 (0.2)
ISP_2	SMTP	573 (11)	100	93	9 (0.02)
ISP_3	VOIP	203 (2.1)	100	96	25 (0.2)
ISP_5	SMTP	388 (7.2)	100	97	52 (0.9)

Table 3: ***Test results for content-based differentiation.***

measurement data. We then verify that the detected differentiation still exists when the probe packets of the two applications traverses the same internal IP-level path. Since per-flow load balancing algorithms use the five tuple $(srcip, dstip, srcpt, dstpt, proto)$ to choose an internal path, we fix the five tuple of one application while only changing the source port of the other application until the probe packets of both applications follow the same internal IP-level path. The results in §6 are obtained after applying this controlled procedure to each application pair.

6. EXPERIMENTAL RESULTS

In this section, we provide concrete evidence of traffic differentiation based on content (§6.1) and routing (§6.3) in backbone ISPs. We study the types of information used to construct content-based differentiation policies and the scope of such policies in an ISP network (§6.4). Without access to ISPs' proprietary policy configurations, we leverage both TOS value in probe packets (§6.4) and two-ended controlled probing (§6.2) to validate the detected differentiations. We also provide insight into when differentiations occur (§6.5) and how significant they are (§6.6) in the large ISPs being studied. Finally, we demonstrate that content and routing based differentiation can be easily implemented on today's commercial routers (§6.7).

6.1 Content-based differentiation

Table 3 presents the detection results of content-based differentiation. We only listed the 4 ISPs that exhibits large degree of differentiation. We use the performance of HTTP as a baseline in comparison with the performance of each of the 4 remaining applications. For a particular application, the "Paths" column lists the number and percentage of ISP internal IP-level paths on which differentiation of the application is detected. Surprisingly, these 4 large ISPs show clear evidence of differentiation of applications such as BitTorrent, PPLive, SMTP, and VoIP in Table 3. For instance, BitTorrent experiences higher loss rate on 3794 (19%) paths in ISP_{12}. This is also true for SMTP on 573 (11%) paths in ISP_2. In contrast, ISP_2 and ISP_3 treat VoIP preferentially on 172 (3.2%) and 203 (2.1%) internal paths. While content-based differentiation is known to exist in broadband ISPs [13], we are the first to detect such differentiation in backbone ISPs.

The percentage of internal paths with detected differentiation is relatively small for some applications. This can be explained by two reasons: i) the differentiation policies are not universally deployed within the ISPs. By analyzing the TOS marking behavior of these ISPs (explained in §6.4), we find the differentiation policies are deployed only at certain routers. If we only consider the internal paths traversing those routers, the percentage of paths with detected differentiation will become much higher as shown in the "TOS_δ" column in Table 3; ii) traffic differentiation may happen only during certain periods, *e.g.*, when network is congested. Since we can only measure the loss rate of a path once every two hours (explained in §4), we may not observe any loss rate differences on a path even if it is configured with differentiation polices. In §6.5, we will show that there is indeed a strong correlation between differentiation and network load.

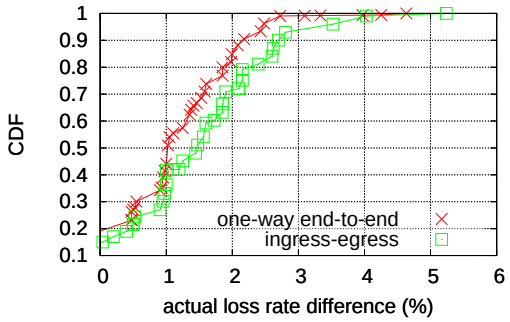

Figure 8: ***Validation using two-ended controlled probing***

The loss rate samples may contain noise even after the filtering process described in §5. The remaining noise may still lead to false positives in the detection results. Because we do not have the ground truth, we cannot quantify false positives directly. Instead, we use the following analysis to show that the detection results indeed reflect content-based differentiation performed by the ISPs. The main idea is to compare for the same paths the detected differentiation across different content with any observable differences due to noise for the same content. Strong evidence of content-based differentiation is manifested if the former is much more prominent than the latter.

We first randomly divide the loss rate samples of *the same application* measured on the same path into two equally-sized subsets and apply the K-S test to these two subsets. Since the samples in both subsets are drawn from the same distribution, an acceptance of the test indicates a false positive due to noise. The test results are in the "FP" column in Table 3. We then make the conjecture that ISPs do not carry out any content-based differentiation. Under this conjecture, we deduce that: i) the detection results in the "Paths" column are all "false positives"; ii) the number of false positives results from the K-S test should be independent of whether it is conducted between the same application (column "FP") or between different applications (column "Paths"). Nonetheless, the numbers in the "Paths" column are mostly over an order of magnitude larger than those in the "FP" column, contradicting our conjecture. This therefore suggests the detection results do reflect content-based differentiation performed by the ISPs.

In §6.4, we will further cross-validate the detection results and the TOS values marked by the ISPs (columns "Δ_{tos}" and "TOS_δ" in Table 3).

6.2 Validation with two-ended controlled probing

As mentioned in §5, loss rate measured by TTL-based probing could be affected by various types of noise. We perform two-ended controlled probing to partially validate the content-based differentiation results presented in the previous section. Given all the PlanetLab node pairs, we first select a subset of them that traverse the ISP internal paths with detected differentiation. In total, we found 13 such pairs, all traversing the internal paths of ISP_{12} with differentiation against BitTorrent. Between each pair of nodes, we simultaneously measure the one-way end-to-end loss rate as well as the loss rate between ingress and egress of ISP_{12} with TTL-based probing, using both HTTP and BitTorrent probes. In Figure 8, the two curves labeled "one-way end-to-end" and "ingress-

egress" correspond to the CDF of actual loss rate differences between HTTP and BitTorrent measured by two-ended controlled probing and TTL-based probing respectively. Clearly, the two curves match quite well, implying that the differentiation between HTTP and BitTorrent can also be confirmed by one-way loss rate measurements.

6.3 Routing-based differentiation

Table 4 summarizes our findings for the 10 ISPs which appear to carry out routing-based differentiation. For previous-hop AS based differentiation, the "AS pairs" column shows the number and percentage of previous-hop AS pairs in which differentiation is detected. Clearly, previous-hop AS based differentiation is commonly used by many ISPs, reflecting the fact that ISPs usually maintain different business contracts with their customers and peers. The number of previous-hop AS pairs exhibiting differentiation can be as large as 1511 (30%) in ISP_{16} and 1086 (21%) in ISP_3. In contrast, next-hop AS based differentiation is far less prevalent. Except for ISP_3, all the other ISPs studied show few cases of next-hop AS based differentiation. This is likely due to the clear advantage of previous-hop AS based approach in enabling an ISP to manage its internal resources to meet its SLAs with customers and peers.

Following the similar logic in §6.1, we show that the detection results indeed reflect previous-hop AS based differentiation performed by the ISPs. We apply K-S test to path pairs that traverse the same $(preAS, ingress, egress, nxtAS)$, which are not subject to any routing-based differentiation. Thus, an acceptance of the test indicates a false positive due to noise, as shown in the "FP" column in Table 4. Under the conjecture that ISPs do not carry out any previous-hop AS based differentiation, we deduce that: i) the detection results in the previous-hop "AS pairs" column are all "false positives"; ii) the percentage of false positive pairs results from the K-S test should be independent of whether it is conducted between the same previous-hop ASes (column "FP") or between different previous-hop ASes (column "AS pairs"). Nonetheless, the percentage numbers in the "FP" column are negligibly small compared to those in the previous-hop "AS pairs" column, contradicting our conjecture. This again implies that the detection results indeed reflect previous-hop AS based differentiation performed by the ISPs, which we will further cross-validate in the next section (columns "Δ_{tos}" and "TOS$_\delta$" in Table 4).

The neighbors of an ISP can generally be classified into customers and peers based on whether the ISP receives payments from them. ISPs may have incentives to give customer's traffic high priority. We employ the commonly-used relationship inference results by Gao [16] to classify the previous-hop ASes into customers and peers. Among all the previous-hop AS pairs consisting of one customer and one peer, the "Customers" and "Peers" columns in Table 4 shows the number of cases where customer's traffic receives better or worse treatment respectively. Seven of the ten ISPs either consistently or mostly give customer's traffic higher priority, confirming our conjecture.

6.4 Correlation with TOS value

As previously illustrated in §2, traffic differentiation can be implemented in the router by marking the TOS field in the IP header. We develop a method to reveal the TOS field marked by the routers along a path. We then study whether the observed traffic differentiation can be explained by different TOS values.

Our probe packets trigger ICMP time exceeded messages from routers. These ICMP messages contain the IP header of the original probe packets, including the TOS values set by the routers. Table 5 illustrates an example of the TOS marking behavior of content-based differentiation. It shows the traceroute output from a PlanetLab node in University of Arizona traversing ISP_{12}. The "TOS" column shows the TOS value of original probe packets extracted from ICMP replies. It is clear that the TOS value of BitTorrent probes is set to 128 by the router at the sixth hop while that of HTTP probes is always 0.

To correlate the loss rate differences with TOS value differences in the traffic, we first need to infer the relationship between TOS values and priorities. We assume an ISP has a consistent policy of associating a TOS value with a fixed priority. However, we do not assume that a large TOS value is always associated with a high priority. Starting with all the pairs that pass K-S test, we compile a list of all the distinct TOS values observed in a target ISP. We then construct a mapping from TOS values to priorities in a way such that the loss rate differences between the pairs with differentiation can be best explained. More specifically, given a pair with differentiation, if the first set has lower loss rates than the second set, the TOS value of the former should map to higher priority.

Can TOS difference explain detected differentiation? Once a mapping is constructed for each ISP, we compute Δ_{tos}, which is the percentage of pairs with detected differentiation that can be explained by differences in priorities inferred from TOS values. The results are in the "Δ_{tos}" columns in Tables 3 and 4, where "-" means no TOS marking is used. Clearly, a large percentage of pairs with detected differentiation can be explained by the priority differences inferred from TOS values. Δ_{tos} is 100% for all the pairs with content-based differentiation (Table 3). For the pairs with previous-hop AS based differentiation, Δ_{tos} is over 80% in 5 ISPs (Table 4). Note that Δ_{tos} is not 100% in some ISPs, which could be caused by ISP's "passive" differentiation. For instance, an ISP may route the traffic from a neighbor through an under-provisioned link, persistently causing high loss rates, even though the ISP does not "actively" treat the traffic with low priority.

Are differentiation policies applied to all the routers? Interestingly, we observe that some ISPs selectively deploy content-based differentiation policy within their networks. Among the 4 ISPs in Table 3, ISP_2 and ISP_3 only mark the TOS field of VoIP traffic that traverses the PoP in Utah and Virginia respectively. The differentiation policy of BitTorrent is much more widely deployed than that of PPLive in ISP_{12}. The TOS marking of BitTorrent is found on nearly 4 times more paths than that of PPLive, which also matches the numbers in the "Paths" column in Table 3. In fact, many routers that perform TOS marking of BitTorrent traffic pay no attention to PPLive traffic. Given that an ISP may not apply the same differentiation policy to all the routers, it is important to cover a reasonable number of ISP internal paths to avoid drawing biased conclusion.

Why the percentage of pairs with detected differentiation is small? As shown in Table 3 and 4, the percentage of application and AS pairs with detected differentiation is relatively small in some ISPs. One major reason is the differentiation polices are not universally deployed within these ISPs. To illustrate this, we compute the percentage of pairs with detected differentiation by only considering the pairs that are confirmed to have differentiation policies based on different TOS values. The results are in the "TOS$_\delta$" columns in Tables 3 and 4. Compared to the corresponding percentage numbers in the "Paths" and "Pairs" columns, TOS$_\delta$ is much higher. Only VoIP in ISP_2 has a TOS$_\delta$ smaller than 80% in Table 3. For previous-hop AS based differentiation, TOS$_\delta$ exceeds 80% in 6 ISPs in Table 4. The reason that TOS$_\delta$ is not 100% is likely due to the fact that differentiation is performed only under certain conditions, *e.g.*, when there is resource competition. As a result, we may not observe any loss rate differences between cer-

ISP name	Previous-hop AS pairs (%)	Δ_{tos}	TOS_δ	Previous-hop Customers (%)	Previous-hop Peers (%)	Next-hop AS pairs (%)	False positive (FP) (%)
ISP_1	480 (11)	85	25	58 (10)	7 (1.2)	97 (1.6)	6 (0.1)
ISP_2	440 (2.4)	48	94	406 (2.6)	0	130 (0.7)	90 (0.5)
ISP_3	1086 (21)	89	86	362 (12)	541 (19)	3159 (15)	11 (0.05)
ISP_5	158 (6)	21	65	36 (4)	22 (2.4)	0	0
ISP_6	559 (16)	98	79	98 (13)	13 (1.7)	164 (4.9)	10 (0.3)
ISP_8	670 (10)	71	41	569 (15)	0	103 (1.3)	33 (0.4)
ISP_9	501 (9)	77	81	365 (12)	0	109 (1.5)	5 (0.07)
ISP_{11}	662 (17)	99	80	99 (5)	232 (12)	93 (2.3)	39 (1)
ISP_{16}	1511 (30)	67	90	134 (12)	243 (23)	102 (2)	5 (0.1)
ISP_{18}	51 (9)	94	91	15 (10)	0	0	0

Table 4: ***Test results for routing-based differentiation.***

Hop	DNS name	TOS	
		BitTorrent	HTTP
2	tuco.telcom.arizona.edu	0	0
3	morgan.telcom.arizona.edu	0	0
4	static.twtelecom.net	0	0
5	-	0	0
6	$HOP_a.ISP_{12}$	128	0
7	$HOP_b.ISP_{12}$	128	0
8	$HOP_c.ISP_{12}$	128	0
9	$HOP_d.ISP_{12}$	128	0

Table 5: ***An example of content-based differentiation confirmed with TOS***

ISP	Destination ports
ISP_{12}	1214 (Napster), 4004 (PPLive), 4662 (eDonkey), 6881-6889 (BitTorrent), 6946, 6961-6969, 6999
ISP_3	10, 5060 (VoIP)
ISP_5	179 (BGP), 16384 (VoIP), 25 (SMTP), 2525 (mail)
ISP_2	25 (SMTP), 53 (DNS), 109 (POP3), 443 (IMAP), 1575, 5060 (VoIP)

Table 6: ***Destination ports used for TOS marking.***

tain pairs even if they are configured with differentiation policies. We will study its correlation with network load in §6.5.

How content-based differentiation policy is constructed? For the 4 ISPs verified to use TOS markings for content-based differentiation, we further analyze which packet fields are used to perform TOS marking. Such information is especially useful for customers who want to circumvent ISP's differentiation policy. We conduct controlled experiments by changing packet headers and application payloads in probe packets. Surprisingly, we found all the 4 ISPs simply use destination port to mark TOS field despite the fact that some applications may change their port numbers, *e.g.*, packets with the default BitTorrent port and fake payloads are still marked. By enumerating different destination ports, we can clearly observe changes in TOS markings. Table 6 lists all the destination ports which are used by the 4 ISPs for TOS marking. For instance, besides PPLive and BitTorrent, ISP_{12} marks the TOS field of Napster and eDonkey (both are P2P applications). It also marks the TOS field of traffic destined to all the default BitTorrent ports between 6881 and 6889. Similar to VoIP, BGP traffic seems to receive preferential treatment by ISP_5, likely reflecting operator's desire to maintain the stability of BGP sessions. We plan to comprehensively study on whether ISPs use other factors rules other than destination port to construct their differentiation policy as future work.

ISP	App	High loss (%)	Low loss (%)
ISP_{12}	BitTorrent	3642 (18)	1707 (8.5)
ISP_{12}	PPLive	825 (4.1)	511 (2.5)
ISP_2	VOIP	182 (3.3)	111 (2)
ISP_2	SMTP	573 (11)	304 (5.8)
ISP_3	VOIP	203 (2.1)	103 (1.07)
ISP_5	SMTP	388 (7.2)	54 (1)

Table 7: ***Effects of network load on content-based differentiation.***

ISP	High loss (%)	Low loss (%)
ISP_1	437 (10)	115 (2.6)
ISP_2	440 (2.4)	308 (1.68)
ISP_3	1108 (21.4)	489 (9.5)
ISP_5	158 (6)	32 (1.2)
ISP_6	559 (16)	414 (11.8)
ISP_8	643 (9.2)	107 (1.6)
ISP_9	501 (9)	115 (2)
ISP_{11}	662 (17)	311 (8)
ISP_{16}	1299 (25.8)	982 (19.5)
ISP_{18}	48 (8.5)	20 (3.5)

Table 8: ***Effects of network load on previous-hop AS based differentiation.***

6.5 Load-sensitive differentiation

Given the strong evidence of traffic differentiation performed by some large ISPs using packet content and routing information, we now investigate whether there exists other factors that may affect traffic differentiation. In particular, if ISPs intend to use differentiation to conserve limited resource in their networks, we should be able to observe a strong correlation between network load and traffic differentiation. For instance, an ISP may throttle BitTorrent traffic only when its bandwidth usage exceeds 100Mbps.

Although we cannot measure network load directly, we can observe its effects in terms of loss rate. High loss rate usually indicates heavy load, given that we have discarded the samples affected by routing changes and failures (in §4). For the two sets in each application or AS pair, we sort the samples in each set based on loss rate value and partition the samples into two equally-sized groups: high-loss *vs.* low-loss. We then perform K-S test both between the two high-loss groups and between the two low-loss groups. Tables 7 and 8 summarize the number and percentage of application and AS pairs that pass the tests. The numbers in the high-loss group are significantly higher than those in the low-loss group, clearly supporting our conjecture that ISPs perform load-sensitive traffic differentiation.

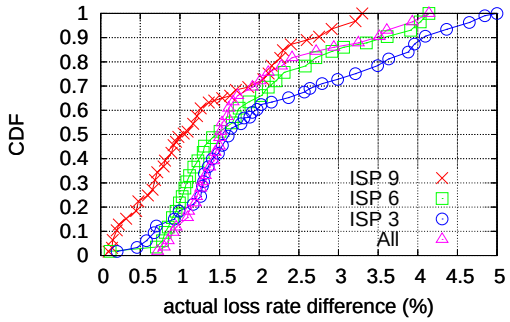

Figure 9: ***Loss rate differences in previous-hop AS based differentiation.***

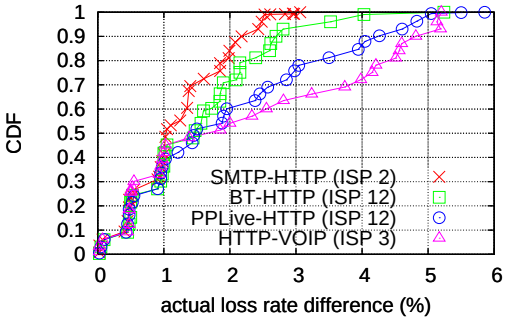

Figure 10: ***Loss rate differences in content-based differentiation.***

6.6 Degree of differentiation

The statistical tests we devise can systematically detect whether there exists differences between two loss rate distributions. We now study whether the actual loss rate differences are significant enough to affect the perceived performance of TCP-based applications. For each AS pair with previous-hop AS based differentiation, we first compute the mean loss rate of each set. We then compute the actual loss rate difference between the two mean loss rates. Figure 9 plots the CDF of actual loss rate differences of all the AS pairs in three target ISPs. Among them, the AS pairs of ISP_9 have the smallest loss rate differences, mostly under 3%. In contrast, the differences are much more evident for AS pairs in ISP_3. Nearly 10% of them have loss rate difference over 4%. Such large loss rate difference will certainly lead to perceptible performance difference for many TCP-based applications.

Figure 10 illustrates the CDF of actual loss rate differences of the application pairs included in Table 3. For each application pair, the actual loss rate difference is computed as the difference between the mean loss rate of an application (*e.g.*, BitTorrent) and that of HTTP. Clearly, the degree of content-based differentiation varies significantly across different applications and ISPs. For instance, ISP_2 treats SMTP only slightly worse than HTTP. Their loss rate differences are smaller than 2% on nearly 90% of paths. In comparison, ISP_3 gives VoIP much higher priority than HTTP, possibly reflecting their desire to meet the QoS requirements of the VoIP service provided by themselves. Interestingly, although both BitTorrent and PPLive experience worse performance than HTTP in ISP_{12}, the loss rates of PPLive are even higher than those of BitTorrent. This is because the paths with PPLive differentiation are only a subset of those with BitTorrent differentiation (explained in §6.4) and this subset of paths tend to have higher loss rates than other paths.

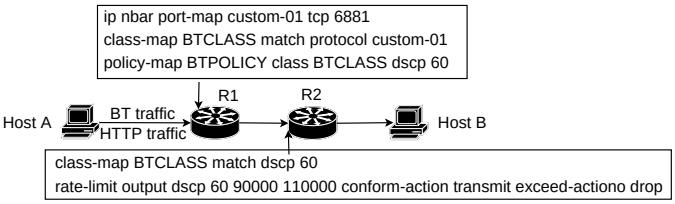

Figure 11: ***Router testbed setup***

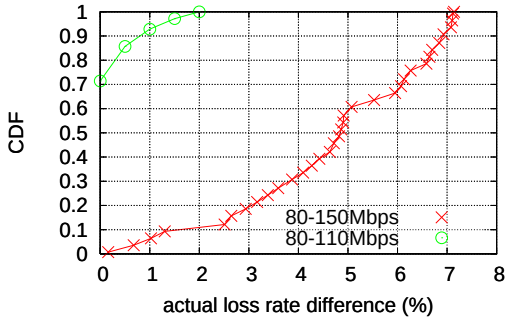

Figure 12: ***Loss rate differences in testbed experiment***

6.7 Implementation of differentiation in router testbed

In this section, we demonstrate the feasibility of implementing and enforcing traffic differentiation in today's commercial routers. As shown in Figure 11, we set up our own experimental testbed using two high-end routers (Cisco 7300 and 12000) running the latest IOS 12.3 from the Schooner testbed [3]. Host A transmits BitTorrent and HTTP traffic to host B via R_1 and R_2. All the machines and routers are connected using Gigabit Ethernet links. To configure the routers for port-based differentiation, we define a port-map on R_1 to capture all the packets with the default BitTorrent port and mark their TOS field using policy map. Interestingly, we found the default router configurations already include pre-defined port-maps for applications such as Napster, Kazaa, SMTP, *etc.* [9], which greatly simplifies the work of configuring differentiation for these applications. The actual router commands used in the Cisco command line interface (CLI) are shown in Figure 11. Similarly, to implement previous-hop AS based differentiation, we can easily mark packets based on incoming interfaces by changing the definition of *class-map* to *class-map NEIGHBOR match interface GigabitEthernet 1/0*. We configure R_2 to prioritize traffic on its incoming interface using weighted random early drop (WRED) queuing.

In § 6.5, we observed that the effects of traffic differentiation are more perceptible when network load is high. To illustrate this, we measure the loss rate differences between HTTP and BitTorrent as we control the sending rate on A. The configurations of R_1 and R_2 remain the same throughout the experiments. R_2 will restrict the BitTorrent bandwidth to be within 110Mbps. Figure 12 shows the actual loss rate differences between BitTorrent and HTTP under two different ranges of sending rates. When the sending rate is high (80 - 150Mbps), the loss rate differences can go up to 7%. In contrast, when the sending rate is below the bandwidth limit (80 - 110 Mbps), the loss rate differences become negligibly small. We also measure the overhead induced by the differentiation configurations on R_1 and R_2. From the SNMP logs, we observed little changes in the CPU utilization on R_1 and R_2 when we disable or enable

the differentiation configurations. This indicates the overhead of enforcing both types of differentiation is small.

7. SYSTEM EVALUATION

In this section, we study the parameter settings and system performance in NetPolice. We will explain the choice of redundancy factor and maximum probing threshold (defined in § 3.1). We will also evaluate the resource usage of NetPolice in terms of network, memory, and CPU. Our evaluation results demonstrate the feasibility of deploying NetPolice as a lightweight tool for continually detecting traffic differentiation in multiple large ISPs simultaneously.

Parameter settings The path selection process of NetPolice is controlled by two pre-defined parameters: the redundancy factor R and the maximum probing threshold m (§ 3.1). R determines the number of distinct paths that will traverse each element. An element can be a three-tuple of $(src, ingress, egress)$ or $(ingress, egress, dst)$. Figure 13 shows the maximum number of destinations assigned to a prober increases with R and remains the same once R exceeds 100. This means when $R > 100$, the redundancy of each element is no longer determined by R but by the set of destinations the probers can probe. We set $R = 100$ to obtain the best coverage.

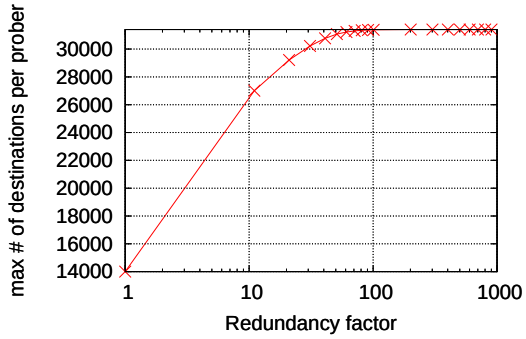

Figure 13: ***How to select redundancy factor.***

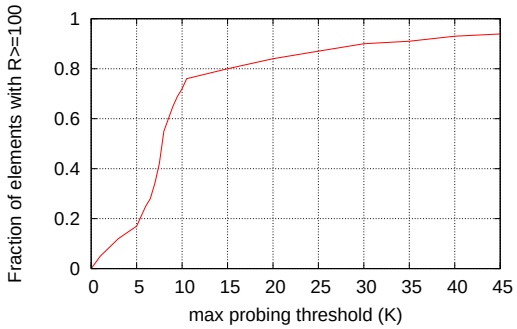

Figure 14: ***How to select maximum probing threshold.***

NetPolice imposes a maximum probing threshold m to prevent a prober from being assigned too many probing destinations. This may cause the actual redundancy of certain elements to be smaller than R. Figure 14 shows the fraction of elements whose actual redundancy reaches $R \geq 100$ under different m. The fraction number grows slowly when m exceeds 10K. We choose $m = 10K$ to attain a reasonable balance between element redundancy and maximum prober overhead. Note that the redundancy of certain elements can never reach R because the number of distinct paths traversing an element is inherently limited by the set of source-destination pairs covered by NetPolice.

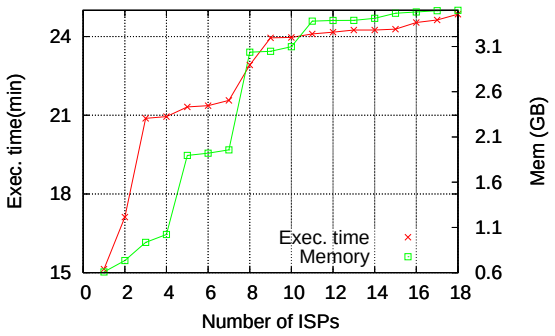

Figure 15: ***Execution time and memory usage of path selector.***

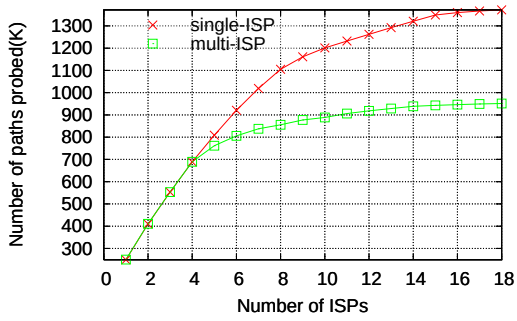

Figure 16: ***Probing overhead under single-ISP vs. multi-ISP path selection.***

Performance evaluation In NetPolice, the number of destinations probed by each prober ranges from 6K to 10K. This corresponds to a bandwidth usage from 17Kbps to 443Kbps per prober. The multi-ISP path selection consumes most of the execution time and memory compared to other components in NetPolice. Since a path can only traverse a limited number of elements, the time and space complexity of the path selection is $O(p^2)$ and $O(ep)$. Here, p is the number of source-destination pairs and e is the number of elements.

We evaluate NetPolice on a commodity server with eight 3.0 GHz Xeon processors and 8 GB memory running Linux 2.6.18 SMP. Figure 15 illustrates the execution time and memory usage of NetPolice as the number of ISPs increases. At 18 ISPs, NetPolice measures 13K unique ingress-egress pairs from 186 sources to 57K destinations. It takes 3.5GB memory for NetPolice to store 182M three-tuples of $(src, ingress, egress)$ and $(ingress, egress, dst)$. The execution time of each run of path selection is around 25 minutes, which is only 20% of one round of probing. This means the path selection process can keep up with measurement speed. To demonstrate the benefit of multi-ISP path selection, Figure 16 compares the total number of paths probed under single-ISP *vs.* multi-ISP path selection. The latter reduces the probing overhead by almost a third when 18 ISPs are being measured.

8. RELATED WORK

Traffic differentiation detection has drawn significant attention among the research community in the last few years. One recent study leverages active measurement launched from end-host

to identify traffic differentiation using port blocking [8]. Evidence of differentiation against P2P traffic has been found in many broadband ISPs by BTTest [13]. Unlike the existing works that study broadband ISPs, NetPolice focuses on differentiation detection in backbone ISPs. This requires us to devise an intelligent path selection algorithm to scalably measure a large number of ISP internal paths.

Our previous work has demonstrated some initial evidence of traffic differentiation in backbone ISPs [35]. In this paper, we significantly extend our prior work by comprehensively presenting, validating, and analyzing the differentiation detection results collected over 10 weeks. NANO [32] targets a similar goal but takes a passive monitoring approach. Without coordinating active measurement from multiple end hosts as NetPolice does, its main difficulty lies in collecting sufficient samples across different hosts and ISPs that enables adjustment for each of the many confounding factors, *e.g.*, congestion and time-of-day effect.

Our work draws heavily on a broad class of measurement studies to reverse engineer the Internet using end-host based probing [30, 21]. Rocketfuel [29] infers ISP topologies by launching traceroute from a set of end-hosts. We extends its DNS naming rules to map IP addresses to geographic locations. There have been a few network-wide systems that measure and predict the performance of various Internet paths [20, 15, 25]. The closest work on monitoring ISP performance is Netdiff [23], which enables cross ISP latency comparison. Similar to NetPolice, the systems above must carefully manage measurement overhead for scalable probing. However, none of them has been used for systematically detecting traffic differentiation.

9. CONCLUSION & FUTURE WORK

In this paper, we presented the NetPolice system to detect content- and routing-based traffic differentiation in backbone ISPs by taking loss measurement from end hosts. NetPolice employs an intelligent probing scheme to attain rich coverage of ISP internal paths while maintaining reasonable measurement overhead. It identifies significant performance gap between different types of traffic using statistical hypothesis tests.

We deployed NetPolice on PlanetLab to study 18 large ISPs across 3 continents over 10 weeks in 2008. We find 4 ISPs that perform differentiation on 4 distinct applications and 10 ISPs that perform previous-hop AS based differentiation, evidenced by up to 5% actual loss rate differences. The degree of differentiation increases with network load. Some ISPs appear to carry out content-based differentiation simply based on port numbers irrespective of packet content. These ISPs may deploy differentiation policies only to a subset of routers in their networks. The loss rate differences are often associated with different TOS values in the traffic marked by the ISPs.

Our work serves as an important step towards increasing the transparency of the Internet. If an ISP blacklists the source IP addresses of NetPolice or disables ICMP response completely, we could get NetPolice deployed on hosts spanning educational, commercial and residential networks to counteract blacklisting of source IP addresses. We could also leverage end-to-end in-band probing techniques [7] to detect performance degradation due to differentiation without requiring router responses. We plan to explore ways to improve the robustness of differentiation detection in NetPolice in the future.

10. REFERENCES

[1] Arbor Ellacoya e100 . http://www.arbornetworks.com.

[2] AT&T Continues to Adjust TOS to Limit 3G Video. http://newteevee.com/2009/04/29/att-continues-to-adjust-tos-to-limit-3g%-video.

[3] Schooner User-Configurable Lab Environment. http://www.schooner.wail.wisc.edu/index.php3?stayhome=1.

[4] A. Akella, S. Seshan, and A. Shaikh. An empirical evaluation of wide-area internet bottlenecks. In *IMC*, 2003.

[5] B. Augustin, M. Curie, T. Friedman, and R. Teixeira. Measuring Load-balanced Paths in the Internet. In *Proc. ACM SIGCOMM IMC*, 2007.

[6] B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, M. Latapy, C. Magnien, and R. Teixeira. Avoiding Traceroute Anomalies with Paris Traceroute. In *Proc. of ACM IMC*, 2006.

[7] I. Avramopoulos and J. Rexford. Stealth probing: Efficient data-plane security for IP routing. In *Proceedings of USENIX Annual Technical Conference*, 2006.

[8] R. Beverly, S. Bauer, and A. Berger. The Internet's Not a Big Truck: Toward Quantifying Network Neutrality. In *Proceedings of the 8th Passive and Active Measurement (PAM) Conference*, 2007.

[9] Cisco Systems. Configuring Port to Application Mapping. http://www.cisco.com/en/US/products/sw/iosswrel/ps1835/products_configu%ration_guide_chapter09186a00800ca7c8.html.

[10] Cisco Systems. Configuring Priority Queueing. http://www.cisco.com/en/US/docs/ios/12_0/qos/configuration/guide/qcpq.h%tml.

[11] Cisco Systems. Simple network management protocol.

[12] cPacket Networks Inc. Complete Packet Inspection on a Chip. http://www.cpacket.com/.

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

[14] Deep packet inspection. www.networkworld.com/details/6299.html.

[15] P. Francis, S. Jamin, V. Paxson, L. Zhang, D. Gryniewicz, and Y. Jin. An Architecture for a Global Internet Host Distance Estimation Service. In *Proceedings of IEEE INFOCOM*, 1999.

[16] L. Gao. On Inferring Autonomous System Relationships in the Internet. In *Proc. IEEE Global Internet Symposium*, 2000.

[17] V. S. Kaulgud. IP Quality of Service: Theory and best practices. www.sanog.org/resources/sanog4-kaulgud-qos-tutorial.pdf, 2004.

[18] S. G. Kolliopoulos and N. E. Young. Approximation algorithms for covering/packing integer programs. *Journal of Computer and System Sciences*, 71(4), 2005.

[19] G. Lu, Y. Chen, S. Birrer, F. E. Bustamante, C. Y. Cheung, and X. Li. End-to-end inference of router packet forwarding priority. In *Proc. IEEE INFOCOM*, 2007.

[20] H. V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani. iPlane: An Information Plane for Distributed Services. In *Proc. Operating Systems Design and Implementation*, 2006.

[21] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson. Inferring Link Weights Using End-to-End Measurements. In *Proc. ACM SIGCOMM IMW*, 2002.

[22] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson. User-level Internet Path Diagnosis. In *Proceedings of ACM SOSP*, 2003.

[23] R. Mahajan, M. Zhang, L. Poole, and V. Pai. Uncovering Performance Differences in Backbone ISPs with Netdiff. In *Proceeding of NSDI*, 2008.

[24] A. W. Moore and D. Zuev. Internet traffic classification using bayesian analysis techniques. *SIGMETRICS Perform. Eval. Rev.*, 33(1), 2005.

[25] V. Paxson, J. Mahdavi, A. Adams, and M. Mathis. An architecture for large-scale Internet measurement. In *IEEE Communications*, 1998.

[26] L. Peterson, T. Anderson, D. Culler, and T. Roscoe. A Blueprint for Introducing Disruptive Technology Into the Internet. In *Proc. of ACM HotNets*, 2002.

[27] J. Saltzer, D. Reed, and D. Clark. End-to-end arguments in system design. *ACM Transactions on Computer Systems*, 2(4), 1984.

[28] J. Sommers and P. Barford. An Active Measurement System for Shared Environments. In *Proc. ACM SIGCOMM IMC*, 2007.

[29] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson. Measuring isp topologies with rocketfuel. *IEEE/ACM Trans. Netw.*, 12(1), 2004.

[30] N. Spring, D. Wetherall, and T. Anderson. Reverse-Engineering the Internet. In *Proc. ACM HotNets*, 2002.

[31] A. Stuart, K. Ord, and S. Arnold. *Kendall's Advanced Theory of Statistics*. Wiley, 1999.

[32] M. B. Tariq, M. Motiwala, and N. Feamster. NANO: Network Access Neutrality Observatory . In *Proceedings of ACM HotNets*, 2008.

[33] J. Tukey. *Bias and confidence in not quite large samples*. Ann. Math. Statist, 1958.

[34] C. Wright, F. Monrose, and G. Masson. On inferring application protocol behaviors in encrypted network traffic. In *Journal of Machine Learning Research (JMLR): Special issue on Machine Learning for Computer Security*, 2006.

[35] Y. Zhang, M. Mao, and M. Zhang. Ascertaining the Reality of Network Neutrality Violation in Backbone ISPs. In *Proceedings of ACM HotNets*, 2008.