Chkdiff: Checking Traffic Differentiation at Internet Access

Riccardo Ravaioli *
I3S/CNRS UMR 7271
Sophia Antipolis, France
ravaioli@i3s.unice.fr

Chadi Barakat
Inria
Sophia Antipolis, France
chadi.barakat@inria.fr

Guillaume Urvoy-Keller
I3S/CNRS UMR 7271
Sophia Antipolis, France
urvoy@i3s.unice.fr

ABSTRACT

In this paper we introduce Chkdiff, a novel method for the detection of traffic differentiation. By aiming at application and differentiation technique agnosticism, we choose an approach that is not specific to the currently most popular applications or to the discrimination mechanisms in use at the time of our writing. The design of such method is carefully described, for both upstream and downstream traffic.

Categories and Subject Descriptors

C.2.3 [**Computer-Communication Networks**]: Network Operations—*Network monitoring*

Keywords

Internet neutrality, traffic differentiation, active measurements

1. INTRODUCTION

In the last few years, ISPs have been reported to discriminate against specific user traffic, especially if generated by bandwidth-hungry applications. The so-called *network neutrality*, advocating that an ISP should treat all incoming packets equally, has been a hot topic ever since.

We propose here Chkdiff, a novel method to detect network neutrality violations that takes a radically different approach from existing work: it aims at both application and differentiation technique agnosticism. We achieve this in three steps. Firstly, we perform measurements with the user's real traffic instead of using specific application traces. Secondly, we do not assume that discrimination takes place on any particular packet field, which requires us to preserve the integrity of all the traffic we intend to test. Thirdly, we detect differentiation by comparing the performance of a traffic flow against that of *all* other traffic flows from the same user, considered as a whole.

In this paper we introduce the design of Chkdiff for upstream and downstream traffic at an Internet access and we evaluate its potentials.

2. RELATED WORK AND OBJECTIVE

The several tools found in literature vary in their methodology, goal and assumptions. Some [3, 4, 7] analyze the performance of flows originating from traces collected by the tool's developer. These tools are in theory application-independent but can only test one application at a time. In practice, they are deployed only with a very small set of application traces, which greatly reduces their effectiveness. Glasnost [2] addresses this by relying on advanced users to add traces for new applications. Nano [6] instead takes a more general approach and passively collects traffic performance information from all users. In light of this, we wanted to get rid of any dependence to specific applications, while at the same time not assume *a priori* that an ISP deploys certain packet scheduling or buffer management techniques. In short, we wanted our method to be robust to i) the rise and fall of popular applications (*application agnosticism*) ii) the evolution of differentiation mechanisms applied by ISPs (*technique agnosticism*).

*The first author is supported by Labex UCN@Sophia.

CoNEXT Student'12, December 10, 2012, Nice, France.

3. KEY DESIGN IDEAS

Throughout the whole design of Chkdiff, we adhered to three key ideas.

IDEA 1: USE REAL USER TRAFFIC. We want to test the existence of traffic discrimination for the exact set of applications run by the end user. Hence, we only consider user-generated traffic.

IDEA 2: LEAVE USER TRAFFIC UNCHANGED, OR ALMOST. All methods performing active measurements send probes made of real application packets and of packets that are similar, but slightly modified, so that they do not get discriminated along their path. This is quite an assumption, as we do not know exactly what ISPs do behind the scenes. In the extreme case, ISPs could even white-list traffic generated by differentiation detecting tools. It is therefore crucial to preserve as much of the original packets as possible, as well as their original per-flow order. We will see that the modifications introduced by our tool affect only the ordering of packets, their TTL value or their IP identification field.

IDEA 3: BASELINE IS THE ENTIRE TRAFFIC PERFORMANCE. Since we do not want to make any hypothesis in advance on what kind of mechanisms - if any - are deployed, we claim that the performance of each single non-differentiated flow should present the same behaviour as that of the rest of our traffic as a whole. Differentiated flows, on the other hand, should stand out when compared to all other flows grouped together, where a large fraction of non-differentiated flows should mitigate the impact of differentiated ones.

4. UPSTREAM SPECIFICATION

We replay user's outgoing traffic and evaluate if their access ISP performs any differentiation. The metric we use is the round-trip time (RTT) between the user and a selected router on their access ISP. A brief outline follows (also refer to Figure 1)

1. CAPTURE TRAFFIC. We capture the user's outgoing packets for a pre-determined time window, generally for a few minutes.

2. ARRANGE PACKETS INTO FLOWS. We arrange packets into 5-tuple flows, which display a sufficiently fine granularity for the meaningfulness of our next analysis.

3. SHUFFLE PACKETS. We want to make sure that all flows see the same network conditions, regardless of any possible transient congestion

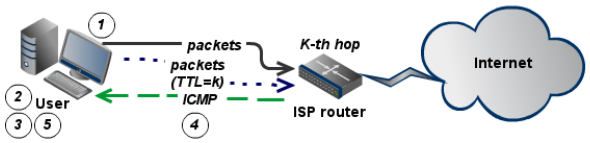

Figure 1: Upstream scenario. 1) Capture traffic 2) Arrange packets into flows 3) Shuffle packets 4) Replay with TTL=k and get ICMP packets back 5) Apply statistical test

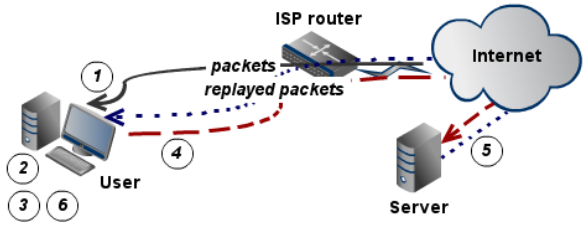

Figure 2: Downstream scenario. 1) Capture traffic 2) Arrange packets into flows 3) Shuffle packets 4) Upload packets to measurement server 5) Packets replayed from server to user 6) Apply statistical test.

along the path, and at the same time not introduce any bias by the order in which we inject packets. We now show how flows exhibit such property after shuffling. We assign a weight to each flow, according to its initial size in packets, and we craft the trace to replay by picking packets from all flows according to flows' weights. By popping packets in the above fashion, we obtain a Bernoulli process for each flow. The waiting time between the occurrence of packets from each flow clearly follows a geometric distribution, which is known to be the discrete version of the exponential distribution. Consequently, each shuffled flow follows a Poisson Process and the PASTA property (Poisson Arrivals See Time Averages) finally holds.

4. REPLAY. We need to target the user's access ISP. In the IP header of each packet, the Time-To-Live field needs to be forged with a value of k, with k being the number of IP hops to the router under examination. This way, each replayed packet will trigger an ICMP Time Exceeded Message on the k-th router along its path, which lets us easily compute the RTT. Care must be taken in choosing the inter-packet delay so as to bypass ICMP-rate limitation.

5. APPLY STATISTICAL TEST. We only consider large enough flows ($>$ 20 packets) as meaningful for testing, while smaller flows only contribute to form our baseline for comparison. To detect traffic differentiation, we use a method that was first introduced by A. Soule et al. [5] in the context of flow classification. Our goal is to be able to decide the likelihood of one flow being generated by the same *source* that generated all other flows, considered as a whole. This can be achieved by converting our samples into histograms, which have the desirable property of incorporating the *entire* distribution of one flow's delays. We then fit the entire set of histograms to a Dirichlet random distribution, which is able to capture the behaviour of almost any type of distribution, without making any a priori assumption on them. The probability that each flow belongs to the same Dirichlet random distribution as all other flows will let us claim whether or not that flow was differentiated.

5. DOWNSTREAM SPECIFICATION

We replay the user's incoming traffic and evaluate if their access ISP performs any differentiation. The metric we use is the one-way delay between our measurement server and the end user. A brief outline follows (also refer to Figure 2).

1. CAPTURE TRAFFIC. As in 4.1
2. ARRANGE PACKETS INTO FLOWS. As in 4.2
3. SHUFFLE PACKETS. As in 4.3
4. UPLOAD TRAFFIC TO MEASUREMENT SERVER. We deploy a measurement server in order to replay spoofed packets and measure one-way delays to our end user. We don't want to modify any important IP packet fields (e.g. source address, payload), but we have to distinguish at destination our replayed packets from current traffic. We thus forge the IP identification field with a known value that is recognized by a firewall at the user side (such firewall is also responsible for not letting replayed packets go up the user's network stack). This is only a minor alteration: differentiation techniques based on IP flows will still see the original IP addresses and those based on Deep Packet Inspection will still see the original payloads.
5. REPLAY TRAFFIC FROM SERVER TO USER. The server can finally send our shuffled packets back to the user at a constant rate ρ (with $\rho < available\,bandwidth$, to avoid stressing the network), starting at instant t_0. Once the user-side application knows these two values, it can directly estimate when each packet was sent and can compute one-way delays.
6. APPLY STATISTICAL TEST. As in 4.5

6. ASSESSMENT AND FUTURE WORK

Only the upstream method is currently available, with the implementation of the downstream part being among future work. The current version of Chkdiff successfully manages to identify differentiation on flow type when packet delays are directly affected. Simple tests in a testbed environment with Dummynet [1] as our traffic differentiator yielded very promising results.

Development possibilities in this domain are rather vast. An easy improvement would be to include an analysis of lost packets on a per-flow basis. This could enable us to detect differentiation when triggered by injection of RST packets into a TCP connection, modification of TCP advertisement window or alteration of application messages. Post-analysis mapping of differentiated flows to applications could give us a better understanding of how user applications might be impacted. Also, while our first prototype only targets a selected router along the path of each packet, the next step would be to test the first few k routers and classify flow delays accordingly. The statistical analysis would then be among delays experienced to the same router. This, together with a large-scale deployment of Chkdiff, could let us map the behaviour of routers at each ISP.

References

[1] M. Carbone and L. Rizzo. Dummynet revisited. *SIGCOMM Comput. Commun. Rev.*, 40(2):12–20, apr 2010. ISSN 0146-4833.

[2] M. Dischinger, M. Marcon, et al. Glasnost: Enabling end users to detect traffic differentiation. In *Proceedings of the 7th Symposium on Networked Systems Design and Implementation (NSDI)*. San Jose, CA, Apr 2010.

[3] P. Kanuparthy and C. Dovrolis. Diffprobe: detecting isp service discrimination. In *Proceedings of the 29th conference on Information communications*, INFOCOM'10, pp. 1649–1657. IEEE Press, Piscataway, NJ, USA, 2010. ISBN 978-1-4244-5836-3.

[4] —. Shaperprobe: end-to-end detection of isp traffic shaping using active methods. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, IMC '11, pp. 473–482. ACM, New York, NY, USA, 2011. ISBN 978-1-4503-1013-0.

[5] A. Soule, K. Salamatia, et al. Flow classification by histograms: or how to go on safari in the internet. In *ACM SIGMETRICS Performance Evaluation Review*, vol. 32, pp. 49–60. ACM, 2004.

[6] M. B. Tariq, M. Motiwala, et al. Detecting network neutrality violations with causal inference. *ACM SIGCOMM CoNext*, p. 289, 2009.

[7] U. Weinsberg, A. Soule, et al. Inferring traffic shaping and policy parameters using end host measurements. In *INFOCOM*, pp. 151–155. 2011.