Identifying Traffic Prioritization on the Internet

Vahab Pournaghshband
Computer Science Department
University of San Francisco
vahab.p@usfca.edu

***Abstract*—ISPs report that the use of VR, video streaming, online gaming, and video conferencing is on the rise in recent years. This rise in the popularity of bandwidth-hungry applications, coupled with resource-constrained networks, has reignited discussion about how different applications' network traffic is treated (or mistreated) by ISPs. There is already evidence that some ISPs are discriminating against certain traffic by prioritizing others. In most cases, ISPs do not reveal the details about their potentially discriminatory behavior. In this paper, we propose an active probing tool that enables end-users to detect whether their ISPs are implementing any prioritization scheduling mechanism. Our technique is non-intrusive and robust to cross traffic. It is entirely end-to-end, requiring no special network support. We have used simulations and Internet experiments to evaluate our approach. Our findings demonstrate an accurate detection of prioritization.**

***Index Terms*—Traffic Prioritization, Internet Measurement, Priority Queueing.**

I. Introduction

Since its early days, the internet was designed from the end-to-end principle. Networks deliver traffic with best effort and do not treat traffic preferentially based on properties such as port number, IP address, or packet content. Recent violations of this principle by many internet intermediaries has led to significant interest in the "network neutrality" debate [1]. In fact, an ongoing debate questions whether the network should be neutral with respect to the packets it carries—that is, not to discriminate against or to alter traffic except for lawful purposes (e.g., blocking illegal content). Network neutrality states that all traffic on the internet must be treated with impartiality, regardless of its origin, destination, or content. This essentially means that unfair traffic differentiation, such as prioritizing or degrading specific traffic flows, is prohibited [2].

In recent years, bandwidth-hungry and delay-sensitive applications such as VR, video streaming, online gaming, and video conferencing have gained substantial popularity. In fact, in a recent report [3] it is estimated that in developed markets, metaverse could generate about 13% of total traffic by 2025. To cope with the growth in their network use, many ISPs have widely adopted Traffic differentiation [4]. Traffic prioritization is one of various popular approaches to traffic differentiation used to manage scarce network resources.

Strict Priority Queuing (SPQ) is one of the primary methods of providing traffic prioritization on edge networks. In fact, SPQ has been available in commercial routers for quite a while, and various models from popular brands, such as the Cisco and Juniper Networks [5] , offer support for it. In a simple two-class model, SPQ classifies network packets as either high priority or low priority traffic, which are then filtered into separate FIFO queues. The SPQ scheduler then ensures that higher priority traffic (if present) is always served before lower priority. In other words, the high priority queue must be completely empty before the regular queue is served. The advantage of this method is that high priority packets are guaranteed delivery so long as their inflow does not exceed the transmission rate of the network. The potential disadvantage is that a high proportion of high priority traffic will cause the low priority traffic to suffer extreme performance degradation, referred to as "starvation" [6].

SPQ's hostile nature toward lower priority flow is particularly undesirable in delay-sensitive network applications. Also, starvation has detrimental effects on TCP-based applications. In a regular TCP data transmission, the sender of the low priority traffic will not receive acknowledgement about the packets sent, because the packets are accumulating in the low priority queue. In fact, they may never reach the receiver, or they may arrive so late that the sender may have already marked them as lost, retransmitted them, and adjusted its congestion window size to a lower value. Mangiante et al. [7] outlines the bandwidth and latency requirements for the streaming of a 360 degrees VR experience. Their findings show QoS mechanisms, such as traffic prioritization, that adversely affect latency and bandwidth, consequently degrade users' quality of experience of these applications.

The ability to detect traffic prioritization enables customers to develop appropriate strategies for improving their application performance. Large content providers, for instance, strive to ensure that their internet applications outperform those offered by their competitors. If a content provider realizes its traffic is being deprioritized by an ISP, it may want to negotiate better service-level agreements with that ISP. Small customers will also benefit from such information. For example, they may change port numbers or encrypt packets to circumvent deprioritization.

In addition to the aforementioned benefits to metaverse users and content providers, there are also broader applications in detecting traffic differentiation by ISPs. Several countries around the world have established network neutrality regulations to prevent unfair traffic differentiation [8]. However, ISP compliance cannot be ensured by regulations alone, and thus detection tools are required to regularly check compliance. Furthermore, even in a nonregulated environment it is important to ensure the transparency of traffic management practices

adopted by ISPs. Internet users in the US already speculate that ISPs may be concealing their discriminatory behavior by prioritizing popular bandwidth measurement tools [9].

Lastly, measuring network path characteristics is critical to diagnosis, optimization, and the development of distributed services. SPQ can severely affect the accuracy of measurement tools' output and the effectiveness of network troubleshooting procedures.

Though details about ISP network policies and configuration are valuable—and, in some cases, critical to end users, administrators, and content providers—most ISPs do not reveal the details about their potentially discriminatory behavior such as traffic prioritization. In light of this, in this paper we present *TarProbe*, a lightweight, accurate tool to detect traffic prioritization by the end hosts and without ISP cooperation. Through active probing methods, TarProbe empowers the stakeholders to understand and characterize the performance degradation caused by traffic prioritization. Our technique is non-intrusive and robust to cross traffic. It is entirely end-to-end, requiring neither changes to nor information from intermediate nodes.

Moreover, despite the potential impact on users and content providers, there is currently a lack of internet-wide studies quantifying its prevalence. The existence of a robust detection tool is the first step in conducting a large-scale study.

The remainder of the paper is organized as follows: Section II presents related work, followed by detection methodology in Section III. Implementations, Simulations, and Internet Evaluation are presented in Sections IV, V, and VI, respectively. Section VII concludes the paper.

II. Related Work

Detecting traffic differentiation is a research topic widely explored in the literature [10]. Most existing traffic differentiation approaches, however, focus on detecting other forms of discrimination than traffic prioritization—most notably, traffic shaping and traffic blocking. These approaches only measure the performance for one flow class at a given time. As a result, these approaches cannot detect SPQ, since the effect of SPQ is observable only when both classes of traffic are present.

Despite the importance of detecting SPQ, only three studies have focused on detecting strict priority queueing: DiffProbe [11], POPI [12], and Packsen [13]. All three approaches use a series of bursty measurements comprising interleaving patterns of packets of different class types and then analyze the results.

POPI [12] uses loss ranks as a statistical means to detect traffic loss differentiation. DiffProbe [11] detects prioritization by comparing the latency between exposed and control traffic. DiffProbe employs the Kullback-Leibler divergence for delay distributions. Similar to DiffProbe, Packsen framework [13] measures the difference in latency between two flows but uses Mann-Whitney U-test to detect traffic prioritization. All three approaches use a fixed set of parameters when compared to the path characteristics. This lack of configurability has made these tools unnecessarily intrusive in some scenarios, and inaccurate in some other scenarios.

III. Methodology

A. Notations

The following set of notations will be used throughout this paper.

- P_H: The packet that is being prioritized and size ℓ.
- P_L: The packet that is being deprioritized and size ℓ.
- P: The number of packets of interests (POI), as defined in Section III.C.
- N': The number of packets in the separation packet train, as defined in Section III.C.
- $|Q_H|, |Q_L|$: The size of high and low priority queues (in packets).
- c_p: The path capacity from the sending to the receiving host.
- Z: The link capacity of the outgoing link from SPQ.
- r: The sending rate of the probe packets as perceived by SPQ. In other words, it is the minimum of the sending rate at the sender and the capacity of the narrow link between the sender and SPQ.

B. Assumptions

In our approach to detecting SPQ on a path, we assumed that the network consists of a series of store-and-forward nodes, each of them is equipped with a FIFO queue and having a constant service rate. We also assume $r > Z$. If the inequality does not hold, then there will be no observable differentiation by SPQ, since Z is not the bottleneck.

C. Approach Overview

To detect whether SPQ is provided on the network, we use active probing in two phases: low priority phase ($\mathcal{LP}$) and high priority phase ($\mathcal{HP}$). In each phase the performance of only one priority class is measured. We use loss as our performance metric. Once both phases are complete, their perceived loss rates are then compared with one another.

In $\mathcal{LP}$, the sender sends P low priority packets (defined as packets of interest), each separated by N' high priority packets (defined as separation packet train). The pattern in which the packet train in low priority phase is constructed is formalized in Algorithm 1. Once the packets are sent, the perceived loss of the packets of interest is measured and recorded at the receiving side. In contrast, the priority class of the packets is flipped in $\mathcal{HP}$. That is, P high priority packets of interest are sent, each separated by N' low priority packets. The rationale behind the proposed pattern is that if there is no prioritization, the loss rate for both high and low priority packets of interest (denoted as l_L, l_H, respectively) should be almost equal. On the other hand, in the presence of prioritization (and with carefully selected values of N' and P), this pattern can create two very different scenarios for packets of interest of each priority class. We discuss these two scenarios here.

In the presence of SPQ, in $\mathcal{LP}$, the high priority packets in the separation packet train maintain a non-empty high priority queue. This results in the starvation and subsequent dropping of low priority packets of interest. On the other hand, the

Algorithm 1 Construct Low Priority Probe Sequence

constructLProbeSequnce(N', P)

```
(P_{L_j})_{j=1}^{P} ← ∅
for i = 1 to P do
    for k = 1 to N' do
        (P_{L_j})_{j=1}^{P} ← (P_{L_j})_{j=1}^{P} || (P_H)
    end for
    (P_{L_j})_{j=1}^{P} ← (P_{L_j})_{j=1}^{P} || (P_{L_i})
end for

return (P_{L_j})_{j=1}^{P}
```

scenario is very different in $\mathcal{HP}$. The low priority packets in the separation packet train outspread the high priority packets of interest so that the high priority packets of interest arriving in the high priority queue experience an empty queue and therefore are served immediately.

As a result of sending probing packets with the described pattern, in the presence of SPQ, low priority packets of interest experience a higher loss rate, compared to that of high priority. This suggests that, to detect the presence of SPQ, a threshold should be specified to distinguish the effects of SPQ from normal internet variabilities:

$$\Delta l = l_L - l_H > \tau. \tag{1}$$

IV. Implementations

We used UDP packets to generate our active probing train. With the help of two TCP connections before and after the UDP train, the sender sends the experiment parameters to the receiver, and the receiver shares the perceived loss of the packets of interest for each priority class to the sender for further analysis.

In the remainder of this section, we detail the selection of parameters.

A. Number of measurements (Γ)

On average, the presence of cross traffic differs at certain times of the day and follows a particular pattern [14]. By performing measurements throughout the day, we hope to capture the effects of time-dependent cross traffic variation. Therefore, we ran each scenario at every hour of a full day, resulting in a total of 24 measurements.

B. Threshold (τ)

Since our approach is loss-based, it requires a predetermined static threshold to compare the loss rates between the packets of interest. Introducing a correct loss ratio threshold requires careful consideration, since this value is a major factor in the accuracy of our detection mechanism. In order to come up with a correct loss ratio threshold, we need to understand the root causes of loss on the internet—and, in contrast, what is considered normal loss, and in contrast, what is considered unusually high loss rate. Studies [15] suggest that 20% is a suitable value for τ, meaning that loss rates larger than that may not be attributed to normal internet variabilities.

C. Separation Packet Train Length (N')

We aim to develop a detection tool that is non-intrusive. Given that the size of the probing train is a factor of N' large, in order for TarProbe to be lightweight, we need to find the smallest value for N' that achieves a high degree of detection accuracy. In this section, we derive a lower bound for N'.

Recall from Section III.B our assumption $r > Z$. Additionally, from Section III.C, our objective is that in $\mathcal{HP}$, Q_H to never build up. On the other hand, in $\mathcal{LP}$, our objective is for Q_H never to be empty. To accomplish these objectives, the following conditions must hold:

(i) High priority packets in $\mathcal{HP}$ are sent at the rate $\frac{r}{N'+1}$. Hence, for Q_H to never build up, $\frac{r}{N'+1} \leq Z$ must hold.

(ii) High priority packets in $\mathcal{LP}$ are sent at the rate $\frac{N'}{N'+1}r$. Therefore, for Q_H to never be empty, $\frac{N'}{N'+1}r > Z$ must hold.

Combining (i) and (ii): $\frac{r}{N'+1} \leq Z < \frac{N'}{N'+1}r$
Lastly, we solve for N':

$$N' \geq \left\lceil max(\frac{r-Z}{Z}, \frac{Z}{r-Z}) \right\rceil \tag{2}$$

Therefore, an optimal value for N' is achievable when the values of r and Z are known. r is the sending rate at the sender and thus is known to the sender. Assuming that SPQ is typically implemented on the bottleneck, Z would then be equal to the path capacity. If $c_P = Z$, then we can use existing tools such as pathrate [16] to estimate the path capacity—therefore, Z.

Moreover, inequality (2) entails a larger N' is suitable when the sending-rate-to-bottleneck ratio (r/Z) is either too small or too large. Though an optimal value for N' is desirable, it is unnecessary in most scenarios. In Section V.B, we show through simulations that a high degree of detection accuracy is still achievable by near-optimal values for N'.

D. Packets of Interest (P)

The total number of packets of interest should be sufficiently large so that the difference in the perceived loss rates (Δl) is greater than the fixed loss rate threshold, τ. In Section V.A, through simulations, we show that 1000 is a reasonable value for P to achieve a high degree of detection accuracy.

V. Simulations

We simulated our approach using the ns-3 simulator [17] under different network scenarios using the simple topology: $Sender \rightarrow SPQ \rightarrow Receiver$.

We used simulation to investigate how network characteristics and parameter selection can influence our detection rate. In our simulation, we used the values presented in Table I for the parameters. Each of the following subsections refers to a particular scenario. In order to observe the effects of a

particular parameter on detection accuracy, in each scenario we modify the value of that parameter, while keeping the remaining parameters constant.

Table I
PARAMETERS USED IN SIMULATION.

Parameter	Value
Sending Rate (r)	10 Mbps
Bottleneck (Z)	2 Mbps
Inter-Packet Time	0.1 ns
$\|Q_H\|$, $\|Q_L\|$	75
Packet Size	200B
Separation Packet Train Length (N')	4
Number of Packets of Interest (P)	1000

A. The Effects of the Number of Packets of Interest (P)

As illustrated in Figure 3, it is clear that for the values larger than 1000, detection accuracy improvement is only marginal.

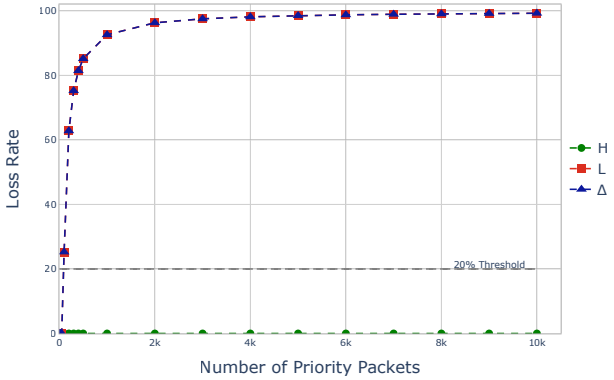

Figure 1. The effects of total number of packets of interests (P) on detection accuracy.

B. The Effects of the Sending-Rate-to-Bottleneck Ratio (r/Z)

In Section IV.C we showed that N', r/Z, and TarProbe detection accuracy have direct correlations. In this simulation scenario, we examine this relationship.

Figure 3 demonstrates this relationship for a range of values for N' and Z, while r is constant. If r/Z is too large ($Z <$ 2Mbps in this scenario), then the high priority packets of interest are also lost, because the link is too slow. As shown in Figure 4, this can potentially affect the detection rate, as the difference in loss rate between high and low priority packets of interest decreases.

On the other hand, if r/Z is too small ($Z >$ 8Mbps in this case), then the high priority packets in the separation packet trains in $\mathcal{LP}$ are processed so fast that they are unable to keep the high priority queue constantly busy. This allows most low priority packets of interest to exit the low priority queue. This leads to a decrease in Δl, which can negatively affect the detection accuracy negatively. In this case, however, where r/Z is close to 1, the end hosts do not noticeably experience the effects of SPQ. To summarize, simulation has confirmed that our approach detects only *effective* SPQ.

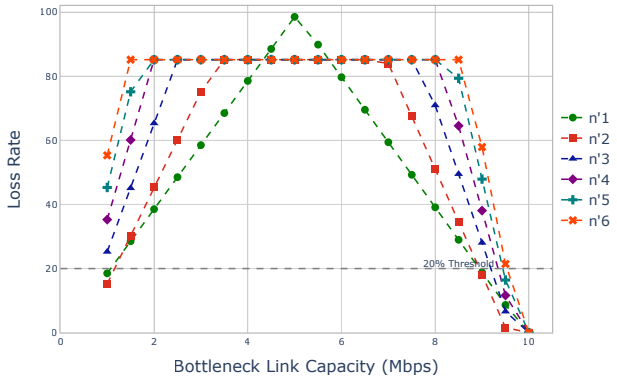

Figure 2. The effects of the separation packet train length (N') and the sending-rate-to-bottleneck ratio (r/Z) on detection accuracy.

C. The Effects of the Separation Packet Train Length (N')

Simulations (Figure 3) confirm that the larger the values of N', the wider the range of scenarios our tool can detect. However, it is evident that $N' = 4$ is a suitable optimal or near-optimal value for all detectable scenarios.

VI. INTERNET EVALUATION

In this section, we present our internet evaluation to confirm that our method works on a real network. Here, we begin with the experiment setup, and follow with a demonstration of the results and an analysis.

A. Experiment Setup

The experiment environment and topology setup are depicted in Figure 4. To enable the effect of SPQ, we used a Cisco Catalyst 3750 switch, which has SPQ enabled. The SPQ-enabled switch and the receiver were all located in the Anonymous University network. The senders, however, are remote PlanetLab [18] nodes. Also, we reduced the sending transmission rate from the switch, making that link the narrow link of the path. Note that to confirm that our SPQ link is indeed the bottleneck, we used pathrate [16]. This capacity estimation tool has been proven experimentally to work well with PlanetLab nodes [19].

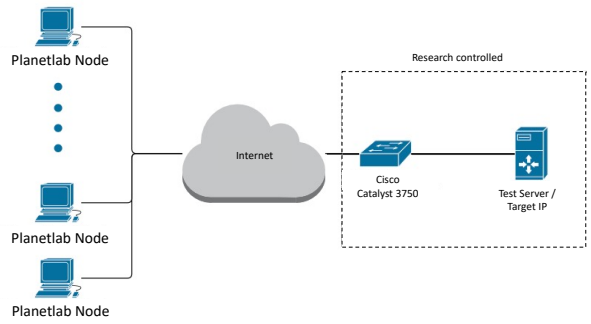

Figure 3. Environment used in our experiments.

We could have placed both the sender and the receiver remotely and simulated a single bottleneck link on the path by routing the traffic stream through our local network. But by using a remote sender and a local receiver, we simulate a more common scenario, in which traffic prioritization is typically present in the edge network. This arrangement also ensures that the experiment matches the no-valley property, which derives from the provider-to-customer relationship and is a commonly adopted routing policy by ASes [20], [21].

B. Results

We selected a set of four geographically distributed PlanetLab nodes connected via the open internet for our experiment. We define an experiment scenario uniquely by a remote PlanetLab node. For each scenario, we performed 24 individual experiments, within a span of 24 hours, running only one experiment every hour.

We used the following parameters in our internet experiments: in the switch, we used the default setting for the queue size in Cisco Catalyst 3750, which is 60 packets [5]. Also, in the switch, we set $Z = 2$Mbps. VM Slices in PlanetLab nodes are typically rate-limited to 10Mbps [19]—thus, $r = 10$Mbps. Provided with this information, N' was set to 4. Finally, we used $P = 1000$.

Figure 5 summarizes our results for all the experiment scenarios we performed by presenting the normal distribution parameters of each scenario. The results confirm the existence of a significant gap in the difference in loss ratio between the high and low priority packets, in the presence of a traffic prioritizer for all scenarios tested.

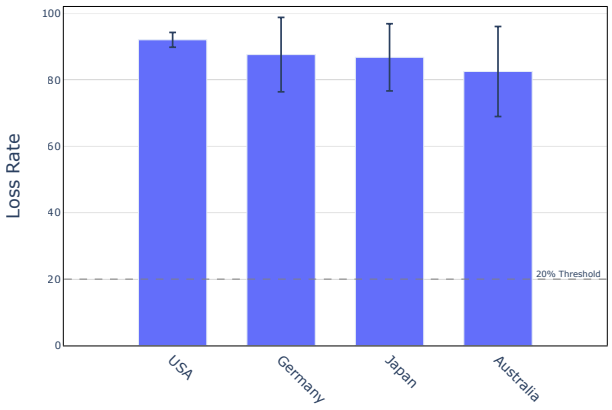

Figure 4. Summary of the results from the Internet experiments: planet-lab2.cs.ucla.edu (USA), pl1.uni-rostock.de (Germany), pl1.sos.info.hiroshima-cu.ac.jp (Japan), and planetlab.research.nicta.com.au (Australia).

VII. Conclusions

Traffic prioritization is a popular traffic discrimination approach employed by ISPs. Though most ISPs do not reveal their treatment (or mistreatment) of certain traffic, this information is valuable and, in some cases, critical for stakeholders (e.g., metaverse end-users and content providers). In this paper, we developed and evaluated a new tool for accurately detecting strict priority queueing. Our proposed approach is a packet-train-like active probing tool that infers *perceivable* discrimination by measuring relative loss rates. Our active probing tool is entirely end-to-end, non-intrusive, and robust against cross traffic. The simulation and internet experiments verified the accuracy of the detection methods, as long as our probing traffic can create some queueing at the discriminatory link and when the loss differentiation is non-negligible. Finally, we are making our code publicly available to empower and inform end-users and bring empirical data to discussions of net neutrality regulations.

References

[1] Timothy Karr, "Net Neutrality Violations: A Brief History," https://www.freepress.net/our-response/expert-analysis/explainers/net-neutrality-violations-brief-history, July 2021.

[2] S. Dustdar and E. P. Duarte, "Network neutrality and its impact on innovation," *IEEE Internet Computing*, vol. 22, no. 6, pp. 5–7, 2018.

[3] STL Partners, "Metaverse and VR to drive traffic growth," https://stlpartners.com/articles/consumer/metaverse-and-vr-to-drive-traffic-growth/, June 2022.

[4] T. Flach, P. Papageorge, A. Terzis, L. Pedrosa, Y. Cheng, T. Karim, E. Katz-Bassett, and R. Govindan, "An internet-wide analysis of traffic policing," in *Proc of ACM SIGCOMM Conference*. ACM, 2016.

[5] Cisco System, "Cisco IOS Quality of Service Solutions Configuration Guide," https://www.cisco.com/c/en/us /td/docs/ios/qos/configuration /guide/12_2sr/qos_12_2sr_book.pdf, November 2009.

[6] M. Rahimi and V. Pournaghshband, "An improvement mechanism for low priority traffic tcp performance in strict priority queueing," in *2016 International Conference on Computer Communication and Informatics (ICCCI)*, 2016, pp. 1–5.

[7] S. Mangiante, G. Klas, A. Navon, Z. GuanHua, J. Ran, and M. D. Silva, "VR is on the edge: How to deliver 360° videos in mobile networks," in *Proc. of Workshop on VR and Augmented Reality Network*, 2017.

[8] H. Habibi Gharakheili, A. Vishwanath, and V. Sivaraman, "Perspectives on net neutrality and internet fast-lanes," *SIGCOMM Comput. Commun. Rev.*, vol. 46, no. 1, p. 64–69, jan 2016.

[9] "Is there any evidence that ISPs prioritise traffic to Speedtest.net?" https://www.reddit.com/r/sysadmin/comments/4tg6si/is_there_any_evidence_that_isps_prioritise/.

[10] T. Garrett, L. E. Setenareski, L. M. Peres, L. C. Bona, and E. P. Duarte, "Monitoring network neutrality: A survey on traffic differentiation detection," *IEEE Communications Surveys & Tutorials*, 2018.

[11] P. Kanuparthy and C. Dovrolis, "Diffprobe: Detecting isp service discrimination," in *2010 Proceedings IEEE INFOCOM*, March 2010.

[12] G. Lu, Y. Chen, S. Birrer, F. E. Bustamante, and X. Li, "Popi: a user-level tool for inferring router packet forwarding priority," vol. 18, no. 1. IEEE Press, 2010, pp. 1–14.

[13] U. Weinsberg, A. Soule, and L. Massoulie, "Inferring traffic shaping and policy parameters using end host measurements," in *INFOCOM, 2011 Proceedings IEEE*. IEEE, 2011, pp. 151–155.

[14] A. Soule, A. Nucci, R. Cruz, E. Leonardi, and N. Taft, "How to identify and estimate the largest traffic matrix elements in a dynamic environment," in *Proc. of 2004 ACM SIGMETRICS*.

[15] H. X. Nguyen and M. Roughan, "Rigorous statistical analysis of internet loss measurements," *IEEE/ACM Transactions on Networking*, vol. 21, no. 3, pp. 734–745, 2013.

[16] C. Dovrolis, P. Ramanathan, and D. Moore, "What do packet dispersion techniques measure?" in *Proceedings IEEE INFOCOM*, vol. 2. USA: IEEE, 2001, pp. 905–914.

[17] "ns-3: An Open Simulation Environment," https://www.nsnam.org/.

[18] "PlanetLab: An Open Network Platform," https://www.planet-lab.org/.

[19] S.-J. Lee, P. Sharma, S. Banerjee, S. Basu, and R. Fonseca, "Measuring bandwidth between planetlab nodes," in *Proc. of the 6th International Conference on Passive and Active Network Measurement*, 2005.

[20] L. Gao, "On inferring autonomous system relationships in the internet," *IEEE Transactions on Networking*, vol. 9, no. 6, pp. 733–745, 2001.

[21] F. Wang, Z. M. Mao, J. Wang, L. Gao, and R. Bush, "A measurement study on the impact of routing events on end-to-end internet path performance," in *Proc. of ACM SIGCOMM*, 2006.