End-to-End Detection of Middlebox Interference

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

Peter Reiher
Computer Science Department
University of California, Los Angeles
Los Angeles, USA
reiher@cs.ucla.edu

***Abstract*—Internet middleboxes are an increasingly common network element. They can significantly alter the handling of traffic streams. Therefore, it is beneficial—and in some cases, crucial—to enable end-hosts to detect them. While *transparent middleboxes* interfere with the traffic, they do not change traffic content, giving the appearance that the data have merely been routed. This transparency makes end-to-end detection of such intermediaries particularly challenging. While existing ad hoc detection approaches apply only to a specific middlebox type, we present a general framework to detect a broad class of middleboxes. We use detecting transparent middleboxes as an example to illustrate our idea. We demonstrate our results by detecting three common middleboxes: network compression, traffic prioritization, and traffic shaping, using analysis, network simulations, and live Internet experiments with a real middlebox.**

***Index Terms*—Middlebox, Detection, Internet measurement.**

I. Introduction

Abstractly, the Internet adheres to the end-to-end principle, characterized by smart endpoints and a simple network. However, the real Internet landscape is much more intricate, primarily due to the widespread deployment of middleboxes at various network points [1]. Extensive research has shed light on the prevalence and functionality of these middleboxes [1], [2]. Other studies [3] show that the number of different middleboxes in an enterprise network often exceeds the number of routers. Today's enterprise networks rely on a diverse range of specialized middleboxes. These middleboxes manifest in various forms, including proxies, firewalls, intrusion detection systems, WAN optimizers, network address translation (NAT) devices, and application gateways. They represent an integral component of today's Internet and significantly contribute to delivering high levels of service for many applications.

A middlebox—defined as "any intermediary device performing functions other than the normal, standard functions of an IP router on the datagram path between a source host and destination host" [4]—manipulates traffic for purposes other than packet forwarding. This interference extends beyond mere routing and can seriously impact traffic flow, resulting in significant implications for both senders and receivers.

Detecting the presence of middleboxes is important for several reasons, including enhancing performance, improving security and privacy, and debugging network issues. Several practical examples highlight the significance of middlebox detection. For instance, detecting middleboxes that impose traffic differentiation can unveil potentially discriminatory behavior by some ISPs. This is particularly crucial in the context of net neutrality, the principle advocating for equal service treatment of all network traffic by ISPs. Moreover, ISPs typically do not publicly disclose such discriminatory behavior, emphasizing the need for detection mechanisms. In the realm of performance, detecting compression in network traffic can save resources on constrained devices by identifying redundant double compression [5]. In the context of debugging, having knowledge about what is present along the communication path is fundamentally important in handling network issues. Generally, tools and methodologies that assist in mapping the Internet prove valuable.

We define *transparent middleboxes* as a class of middleboxes that make no changes to the content of traffic, giving the appearance that nothing has been done to the traffic stream other than routing it to the destination. Either they do not modify the payload of packets or they reverse their changes before the packets reach the receiver. Transparent middleboxes can perform many functions, such as eavesdropping, delaying packets, compressing or decompressing payloads, encrypting or decrypting payloads, combining or splitting packets, and modifying link, IP, and transport layer headers. The transparency property of transparent middleboxes makes end-to-end detection of such intermediaries harder in most cases, raising the question, "Can end hosts determine whether something of this kind has been done if they pay attention?"

The existing literature has proposed mechanisms to detect several transparent middleboxes, such as network compression [5], TCP/IP header modification [6], and various forms of traffic differentiation [7]–[12], including traffic blocking [13], [14], traffic shaping [15]–[19], and traffic prioritization [20]–[22]. But these detection mechanisms focus on detecting only a specific middlebox by presenting a dedicated tool for that purpose. Therefore, they lack studies that explore the detection of a broad set of middleboxes. Moreover, it may be beneficial to determine whether a middlebox is detectable without the necessity of developing a specialized tool for that detection.

Our contribution is novel in two key aspects:

- Detecting a broad class of middleboxes.
- Introducing a reduction framework to determine if a middlebox MB is detectable via only analytical means. Specifically, we reduce the problem of "is MB detectable" to the problem of "is MB' detectable," by

showing that (1) MB's behavior can be analytically described as MB' and (2) MB' is detectable.

We illustrate our approach through a concrete example. We define an abstract class of transparent middleboxes termed *discriminators* in Section II, and show they are detectable (Section III). In Sections IV, V, and VI, we show that three distinct middleboxes—network compression, traffic shaping, and traffic prioritization—are detectable by showing that their behavior can be described analytically as a discriminator. Subsequently, we demonstrate how our detection framework enabled us to design detection tools for these middleboxes. We implemented these tools and present the results obtained from our Internet evaluations in Section VII. Section VIII outlines potential directions for future work and concludes the paper.

II. Preliminaries

In this section, we introduce the notations, formal definitions, and terminologies that we will use throughout the paper.

A. Notations

- $\mathcal{SND}$: The sending host.
- $\mathcal{RCV}$: The receiving host.
- P_A: A packet with predefined property A and size $|P_A|$.
- P_B: A packet with predefined property B and size $|P_B|$.
- MB: The middlebox in question.
- $t_d(P)$: Departure time (time to transmit the first bit of packet P into the wire).
- $t_a(P)$[1]: Arrival time of the last bit of packet P to the middlebox.
- $d_{MB}(P) := t_d(P) - t_a(P)$: Total delay imposed by the middlebox on packet P.
- t_T: Transmission time (time to transmit the entire packet P into the MB's outbound link to the destination).
- c_p: The path capacity—the link capacity of the narrow link between $\mathcal{SND}$ and $\mathcal{RCV}$.
- Z: The link capacity of the outgoing link from MB.
- r: $\mathcal{SND}$'s sending rate of the probe packets as perceived by MB. This is the minimum of the sending rate at the sender and the capacity of the narrow link between $\mathcal{SND}$ and MB.

B. Definitions

An *influence* indicates the observable and measurable effects of the third party's middlebox interference on the path properties. For instance, in the case of interference involving packet encapsulation, where packets become larger, the influence manifests as the additional delay resulting from the increased transmission time of each packet.

An influence I is *detectable* if there exists an approach to detect I under normal network conditions,[2] using the existing measurement tools available to typical hosts, considering the inherent accuracy and precision limitations of these tools.

[1] $t_a(P)$ and $t_d(P)$ are based on relative clocks in a single system, MB; thus, synchronization is not required.

[2] Normal network condition entails that none of the following is present for a long duration: network failure, network instability, or persistent cross traffic.

We define *network flow discriminators* (or simply *discriminator*) as middleboxes that influence traffic flows by delaying (or dropping) packets of different flows in a discriminatory way. In the case of two network flows, one with property A and the other with property B, a discriminator introduces additional delays to each packet in the flow with property A compared to the flow with property B. Below, we formally define discriminators:

An ordered pair of sets of packets $(\tilde{P_A}, \tilde{P_B})$ are said to be *delay-discriminatory* if:

- $\mid \tilde{P_A} \mid = \mid \tilde{P_B} \mid \neq 0$.
- $\exists P_{A_i} \in \tilde{P_A}, \exists P_{B_j} \in \tilde{P_B}$:
  $\delta_i := d_{MB}(P_{A_i}) - d_{MB}(P_{B_j}) \geq \delta_{min} > 0$ [3]
  and
  $t_T(P_{A_i}) \geq t_T(P_{B_j})$
  and
  $(\tilde{P_A} - \{P_{A_i}\}, \tilde{P_B} - \{P_{B_j}\})$ is also delay-discriminatory or is $(\emptyset, \emptyset)$.

This definition implies the existence of at least one bijective mapping between the elements of set $\tilde{P_A}$ and set $\tilde{P_B}$, such that when $d_{MB}(P_{A_i})$ and $d_{MB}(P_{B_i})$ are compared, a positive delay is observed. Furthermore, this definition guarantees that the transmission delay for P_{A_i} is greater than or equal to that of P_{B_i}.

This definition also includes middleboxes that drop packets in a discriminatory way since $\delta_i = +\infty$. Common examples of such middleboxes are traffic policers and routers that employ discriminatory buffer management mechanisms such as WRED [23]. Note that this definition excludes cases where both P_{A_i} and P_{B_i} are dropped.

Network Flow Discriminator: A middlebox MB is a *network flow discriminator* if, given $d_{MB}()$, there exists a pair of non-empty sets of packets $(\tilde{P_A}, \tilde{P_B})$ that is delay-discriminatory.

III. Detecting Discriminators

In this section, we first show that discriminators are detectable by presenting a detection approach. We then introduce a framework to show that any middlebox whose influence can be analytically described as a discriminator is also detectable, provided certain requirements are met.

A. Assumptions

In our analysis in this section, we make several simplifying assumptions. The rationale behind them is to avoid unnecessary complexity in describing discriminators' influence deterministically and in presenting the fundamental aspects of our analytical presentation.

We assume that the network comprises a series of store-and-forward nodes, each of them equipped with a FIFO queue and has a constant service rate. We further assume that the transmission delay is linear with respect to packet size. In addition, we assume all packets in the network flow between the end-hosts go through the middlebox and that

[3] Following this definition: $\delta_{min} := \min_{\forall i} \delta_i$,

other traffic does not cause the measurement packets to queue. Furthermore, we assume the length of all P_As and P_Bs is fixed and equal to $|P|$. Lastly, we assume that $d_{MB}(P_{A_i})$ and $d_{MB}(P_{B_i})$ are constant for all i's, so we simply refer to them as $d_{MB}(P_A)$ and $d_{MB}(P_B)$.

B. Approach Overview

In this section, we introduce our proposed approach for detecting discriminators, followed by an elucidation of its underlying rationale.

To detect discriminators, we use active probing in two rounds. In each round, a packet train consisting exclusively of packets with property either only A or B is sent. To detect discrimination, the loss rate in each round is measured and then compared.

To investigate the detectability of discriminators, we look at the problem from a unique perspective. We imagine that the imposed delay by the discriminator can be analogously likened to the effects of transmission delays introduced by a virtual link inside the middlebox. In this conceptualization, the middlebox is represented as two virtual nodes, denoted as C and D, connected by a link of bandwidth S (as depicted in Figure 1). Notably, it is crucial to recognize that C and D are not actual routers. Therefore, C has a receive buffer but not a transmission queue, and D has a transmission queue but does not have a receive buffer.

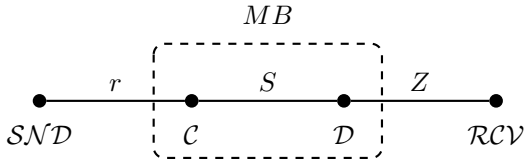

Fig. 1: Decomposition of the discriminator node into two virtual nodes connected by a virtual link.

Recall from Section III.A, we assume $d_{MB}(P_{A_i})$ and $d_{MB}(P_{B_i})$ are constant for all i's. Therefore,

$$\forall i, \delta_i = \delta_{min} = d_{MB}(P_A) - d_{MB}(P_B)$$

For arriving packets with property A and B, their respective link capacities of the virtual link are[4]

$$S_A = \frac{8 \cdot |P|}{d_{MB}(P_A)}, S_B = \frac{8 \cdot |P|}{d_{MB}(P_B)}$$

To facilitate the creation of discriminatory effects by MB, we focus on the following scenario: $S_A < r < S_B$. This is because when packets are sent at the rate r, some P_As are expected to be dropped at MB, since $S_A < r$. Conversely, all P_Bs are unaffected by the middlebox interference since $S_B > r$. This observation suggests a loss-based approach to detect discriminators.

To ensure that the observed packet losses are solely attributable to the interference introduced by MB, we make the assumption that $r \not> c_p$. This ensures that there are no other queue buildups (and hence no packet losses) because of a fast sender. We thus assume $r = c_p$[5], and consequently, $S_A < c_p < S_B$.

Since $c_p < S_B$, the loss rate attributed to MB for P_Bs is 0, whereas for P_As, it is $1 - S_A/c_p$, considering $c_p > S_A$. These loss rates are denoted as l_A and l_B. Consequently, the difference in loss rate (denoted as Δl) equals to $1 - S_A/c_p$. This suggests that if a threshold—say τ— exists so that it distinguishes normal Internet loss and induced loss of this kind by MB, then for the discriminator to be detectable, it is sufficient to show that the following holds:

$$\Delta l = 1 - \frac{S_A}{c_p} \geq \tau \quad (1)$$

In Section III.C, we show that such a threshold exists. Note that the Inequality (1) demonstrates the relationship between δ_{min} (since $\delta_{min} = f(S_A, S_B)$), c_p, and MB's detectability.

C. Implementation Details

In Section III.B, we showed that a loss-based approach to detect discriminators exists. In this section, we present the details pertaining to our proposed approach.

In summary, the detection process comprises four distinct phases: pre-probing, probing, post-probing, and detection. The pre-probing phase involves the construction of packets for the probing phase. In the probing phase, the sender sends back-to-back packets with property A, followed by a brief pause to avoid inter-measurement bias, and then sends a train of packets with property B. In the post-probing phase, the receiver forwards information about the collected measurement data back to the sender. The three probing phases are iterated 24 times (once every hour). The rationale behind repeating the experiment in a span of 24 hours is that, on average, the presence of cross traffic differs at certain times of the day and follows particular patterns [25]. By conducting measurements throughout the day, we hope to identify and eliminate the effects of non-persistent network failures, instabilities, and time-dependent cross traffic variations from our detection results. Once all measurement rounds are completed, in the last phase, the end-hosts make the detection decision by analyzing the collected measurement data.

1) Pre-Probing Phase: In this phase, the sender constructs the detection packet-train-like probe. The detection probe consists of two probe sequences. The first probe sequence consists of ρ packets with property A. These packets are denoted as $(P_{A_j})_{j=1}^{\rho}$, where $(P_{A_j})_{j=1}^{\rho} := (P_{A_1}, P_{A_2}, ..., P_{A_\rho})$ is a finite sequence of ρ-ordered elements. The second probe sequence, denoted as $(P_{B_j})_{j=1}^{\rho}$, consists of packets with property B. It is important to note that while, in this instance, all packets within each probe sequence are identical and share the same property, this may not always be the case (for an alternative scenario, refer to Section VI, for example).

In the next phase (probing phase), the sender sends the constructed packets in the exact order in which they are organized within each sequence.

[4]The factor "8" matches the byte-to-bit unit.

[5]In practice, to ensure $r = c_p$, the sender first estimates c_p using existing tools such as pathrate [24]. The sender then reduces its sending rate to c_p.

Recall from Section II.B that the probe sequences are meticulously crafted so that $(\tilde{P_A}, \tilde{P_B})$ is delay-discriminatory. Since $d_{MB}(P_{A_i}) - d_{MB}(P_{B_i}) \geq \delta_{min}$, $(\tilde{P_A}, \tilde{P_B})$ should be constructed to achieve the largest δ_{min}. This is because, as Inequality (1) shows, δ_{min} and MB's detectability are positively correlated.

Lastly, during the pre-probing phase, the sender and receiver establish a TCP connection to exchange the measurement parameters required for the subsequent probing phase.

2) Probing Phase: In this phase, $\mathcal{SND}$ transmits the packets in $(P_{A_j})_{j=1}^{\rho}$, with each packet being separated by a time interval of γ. It then pauses for a Λ period of time, before sending the packets in $(P_{B_j})_{j=1}^{\rho}$, with each packet in the probe γ apart.

We chose a very small Inter-Departure Time (γ), ensuring that the packets in each probe sequence are sent back-to-back. This results in the perceived sending rate, r, being the highest possible value. Furthermore, the Inter-Measurement Time (Λ) must be configured to maintain independence between measurement probes, preventing any form of inter-measurement bias. Λ should be set to a value that is sufficiently large to allow MB's queue to dissipate from the preceding experiment's probes. However, it should not be excessively high to maintain relatively consistent network conditions across both measurements. We choose Λ to be 30 seconds to meet these criteria effectively.

3) Post-Probing Phase: In this phase, through a TCP connection, $\mathcal{RCV}$ shares with $\mathcal{SND}$ the information regarding the receipt of each packet in each measurement round. This information is used in the detection phase.

4) Detection Phase: In this phase, the end-hosts use the collected data from the 24 measurement rounds to calculate Δl from the loss rates, l_A and l_B, for P_{A_i}s and $P_{B_i}s$ in each round. If the median of Δl is larger than a threshold, τ, then discrimination is detected.

To generate a correct threshold, we need to understand the root causes of loss on the Internet and what is considered normal loss—and, in contrast, what is considered an unusually high loss rate. Studies measuring normal loss on the Internet [26] suggest 20% as a suitable value for τ. Losses larger than that may not be attributable to normal Internet variabilities.

D. Simulation Results

We used ns-3 [27] to simulate the discriminator and our proposed approach to detect them. Figure 2 demonstrates the loss rates caused by MBs that introduce d_{MB} delays. Recall that δ_{min} is the difference in d_{MB} values. Figure 2 illustrates an important observation: when $d_{MB}(P_A)$ and $d_{MB}(P_B)$ are configured such that $l_A > \tau$ and $l_B = 0$ for a particular MB, then the loss rate $\Delta l = l_A - l_B = l_A$. In this scenario, discriminators with Δl values exceeding the 20% threshold (τ) are detectable. Therefore, Figure 2 demonstrates how specific values of d_{MB} and c_p impact the ability to detect discriminators.

To summarize, the simulations confirmed that the discriminator is detectable under the following conditions:

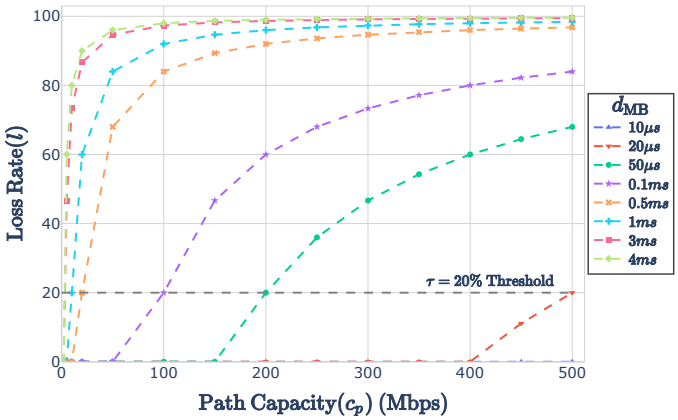

Fig. 2: The relationship between d_{MB} and c_p in detecting a discriminator ($\Delta l > \tau$), where $\rho = 1000, c_p = r$.

(1) We observe that when $\delta_{min} \geq$ 1ms, MB is detectable for nearly all values of c_p. We refer to this threshold as Θ.

(2) We also observe that the requirement $\delta_{min} = \Theta \geq$ 1ms alone, however, is not sufficient to guarantee that the discriminator is detectable. This is because, in some cases, although the introduced loss rate is substantial, the difference in loss rate may not meet the detectable threshold. For instance, if $d_{MB}(P_A) =$ 4ms and $d_{MB}(P_B) =$ 3ms, resulting in $\delta_{min} =$ 1ms, Δl might fall below τ, rendering it undetectable. Hence, there should be a complementary requirement governing the maximum value of $d_{MB}(P_B)$. To address this, we introduce a different threshold, θ, requiring $d_{MB}(P_{B_i}) < \theta \; \forall i$. Figure 2 illustrates that for all practical path capacities ($c_p \leq$ 500Mbps), the delay of 10μs imposed by MB has minimal impact on the traffic.

E. Detection Framework

In this section, we present the main contribution of our paper: a reduction framework to determine if a middlebox MB is detectable via only analytical means. Essentially, we reduce the problem of "is MB detectable" to the problem of "is discriminator detectable" by showing that MB is a discriminator. Given the premise that a discriminator is detectable under certain conditions, we only need to show that MB is a detectable discriminator.

To show that a particular middlebox MB, which influences network flows by imposing discriminatory delays (or drops), is detectable, it is sufficient to show:

1) MB is a discriminator. In other words, there exists a pair of non-empty sets ($\tilde{P_A}$,$\tilde{P_B}$) that is delay-discriminatory.
2) $\delta_{min} \geq \Theta$, where $\delta_{min} := \min\{\delta_i : 1 \leq i \leq \rho\}$.
3) $\forall i, d_{MB}(P_{B_i}) \leq \theta$.

Alternatively, these conditions can be described as follows:
$\exists \tilde{P_A}$,$\tilde{P_B}$, such that
$\forall i, (d_{MB}(P_{A_i}) - d_{MB}(P_{B_i}) \geq \Theta$ and $d_{MB}(P_{B_i}) < \theta)$.
We empirically choose Θ=1ms and θ=10μs.

F. Discussions

In this section, we presented a straightforward detection approach to illustrate the detectability of middleboxes that

cause discriminatory delays. We acknowledge that a more sophisticated detection engine can be developed to employ complex statistical tools to enhance robustness against outliers and normal network variations while requiring fewer rounds than 24—ideally just one. However, this paper's objective is not to devise such optimal detection approach but rather to demonstrate that *a* solution for detecting discriminators exists.

In the remaining sections, we use the demonstrated fact that "discriminators are detectable under specific conditions" as the premise to determine the detectability of different MBs via analytical means.

IV. Detecting Network Compression

Routers and nodes within the Internet, including WAN optimizers, often implement compression techniques at the link-layer or IP-level to enhance network throughput and reduce latency between endpoints. These compression methods, whether at the link layer [28], [29] or IP level [30], typically involve compressing both the payload and upper-layer header information of traffic flows. In this section, we show that network compression is a discriminator and thus detectable by demonstrating that it discriminates between packets with high entropy payloads and those with low entropy payloads.

A. Notations

- P_L, P_H: The packets with a low entropy (e.g., filled with zero) and a high entropy (e.g., filled from `/dev/urandom`) payload, both with size $|P|$.
- P'_L, P'_H: The resulting packet after compression is applied to P_L and P_H. Their respective sizes are $|P'_L|$ and $|P'_H|$.
- $\alpha_P := \dfrac{|P|}{|P'|}$: Compression efficiency for P.
- $d_c(P)$: Processing delay to compress P.
- $|Q|$: Size of transmission queue at the compressing side.
- $t_d(P)$: Departure time (time to transmit the last bit of packet P into the wire).[6]

B. Network Compression is Detectable

To confirm that network compression is detectable, we show that the three conditions in the detection framework detailed in Section III.E hold.

1) Network Compression is a Discriminator: We achieve this by constructing ($\tilde{P_A}$, $\tilde{P_B}$) that is delay-discriminatory. $\tilde{P_A}$ is a set of ρ packets of P_{H_i}. In contrast, $\tilde{P_B}$ is a set of ρ packets of P_{L_i}. This construction allows us to create a scenario where packets with high entropy payloads (P_H) and packets with low entropy payloads (P_L) are subjected to network compression, ultimately highlighting the delay discrimination characteristic of network compression.

It is widely acknowledged that an effective compression algorithm should uphold the following two statements: $d_c(P_H) \geq d_c(P_L)$, and $|P'_H| > |P'_L|$. Recognize that the second statement implies $t_T(P'_H) > t_T(P'_L)$:

$$\begin{aligned}\delta_i &= d_{MB}(P_{H_i}) - d_{MB}(P_{L_i}) \\ &= \{d_c(P_{H_i}) - d_c(P_{L_i})\} + \{t_T(P'_{H_i}) - t_T(P'_{L_i})\} > 0\end{aligned}$$

Therefore, $\forall i, \delta_i > 0$; hence, network compression is a discriminator.

2) $\delta_{min} \geq \Theta$ and $\forall i, d_{MB}(P^i_B) \leq \theta$: First, we assume $Z = c_p$, with this assumption arising from the widely accepted understanding that implementing link-layer compression on non-bottleneck links is a poor network management practice since it degrades network performance. Additionally, we assume the following conditions hold:

$$c_p \cdot \alpha_{P_H} < r < c_p \cdot \alpha_{P_L} \tag{2}$$

$c_p \cdot \alpha_P$ should be viewed as the *perceived link capacity* of the compression link for P. Given that $|P'_H| \approx |P|$, it follows that $\alpha_{P_H} = \dfrac{|P'|}{|P|} = 1$. Conversely, α_{P_L} is typically significantly larger for any effective compression algorithm. For instance, consider a packet with a payload consisting only of zeros and a size of 1100 bytes. When compressed with LZ Deflate [31], it reduces to 56 bytes, resulting in $\alpha_{P_L} \approx 20$.

The left side of (2) holds because $Z = c_P < r$. The right side of (2) holds since the sender is able to reduce its sending rate to ensure $r < c_p \cdot \alpha_{P_L}$, in case α_{P_L} is still not sufficiently large. Since $r < c_p \cdot \alpha_{P_L}$, P_L's do not experience queuing delay. Therefore, P_H's queuing delay is the only contributing factor in calculating δ_i.

To compute δ_i, we initially assume that the queue at the compression link has become saturated in the case of P_H's (due to $r > c_p$). Consequently, for each P_{H_i}, δ_i can either be $+\infty$ (indicating that the packet is dropped because the queue is full) or $(|Q|-1)|P'_H|/c_p$ (representing the time required to transmit all packets preceding the last queued packet). Given that the value of $|Q|$ is typically unknown to the end-hosts, we introduce and employ the concept of super-packets to ensure our analysis remains independent of this variable.

λ-super-packets: The j-th λ-super-packet ($\dot{P}_j$) comprises the j-th grouping of λ consecutive packets within the probing packet sequence: $\dot{P}_j := (P_{\lambda j}, P_{\lambda j+1}, \ldots, P_{\lambda(j+1)-1})$.
We then define $d_{MB}(\dot{P}_j)$ as follows:

$$d_{MB}(\dot{P}_j) := \begin{cases} 0 & \text{if all packets in the group are received} \\ \infty & \text{if at least one packet in the group is lost} \end{cases}$$

Hence, we evaluate the discriminatory impact of compression on the super-packets instead of the original low and high entropy packets. Recognizing that the loss rates for P_L and P_H are 0 and $1 - c_p/r$, respectively, our objective is to determine the smallest natural number, λ, where at least one packet is lost in every $\dot{P}_{H_j}$:

$$\frac{c_p}{r} \leq \frac{\lambda - 1}{\lambda}$$

Solving for λ,

$$\lambda \geq \left\lceil \frac{1}{1 - \frac{c_p}{r}} \right\rceil \tag{3}$$

[6]Notice that we slightly modified the definition of $t_d(P)$ from its original definition in Section II.A. With this revised definition of $t_d(P)$ and considering the characteristics of the compression process: $d_{MB}(P) = d_c(P) + t_T(P)$. This accounts for the specific impact of compression on the delay experienced by the packet P as it passes through MB.

(3) shows the existence of such λ so that $\forall j, d_{MB}(\dot{P}_{H_j}) = +\infty$. Simultaneously, $\forall j, d_{MB}(\dot{P}_{L_j}) = 0$, demonstrating that $\delta_{min} \geq \Theta$ since $\delta_{min} = +\infty$. Additionally, since $\forall i, d_{MB}(\dot{P}_{L_i}) = 0$, $\forall i, d_{MB}(P_B^i) \leq \theta$ also holds.

In conclusion, all three steps in the detection framework hold; therefore, network compression is detectable.

V. Detecting Traffic Shaping

Traffic shaping is widely used to help optimize performance and improve latency by controlling the amount of data that flows into and out of the network. Many access ISPs deploy traffic shaping to limit the peak network traffic on their transit links and thereby reduce their wide-area bandwidth costs [15].

A traffic shaper employing the token bucket algorithm with shaping profile (σ, Σ) functions as a forwarding module with the following behavior: Consider a link with a capacity of νbps, associated with a token bucket of size Σ tokens. Whenever the bucket is not full, tokens are generated at a rate of σ tokens per second, where $\sigma < \nu$. The link can transmit an arriving packet of size m bits only if the token bucket has at least m tokens; upon the transmission of the packet, the shaper depletes m tokens from the bucket, equivalent to the size of the transmitted packet.

In this section, we show that traffic shaping is a discriminator and thus detectable.

A. Notations

- P_σ: The packet (with size $|P|$) that is being shaped adhering to the shaping profile (σ, Σ).
- P_ν: The packet (with size $|P|$) that is not being shaped and consequently serviced at the link capacity rate, ν.
- $|Q|$: The size of the transmission queue at the shaper's outbound link.

B. Traffic Shaping is Detectable

To confirm that traffic shaping is detectable, we show that the three conditions in the detection framework detailed in Section III.E hold.

1) Traffic Shaper is a Discriminator: We achieve this by constructing $(\tilde{P_A}, \tilde{P_B})$ that is delay-discriminatory. $\tilde{P_A}$ is a set of ρ packets of P_σ, and $\tilde{P_B}$ is a set of ρ packets of P_ν.[7]

In our analysis, we assume the following scenario:

$$\sigma < r \leq \nu \tag{4}$$

The left side of (4) stems from the fact that if $r \leq \sigma$, in the absence of cross traffic, the shaper is not interfering with the traffic flow, and thus there is no interference to detect. The right side of (4) holds because if $r > \nu$, then the sender can reduce its sending rate to ν.

We further assume that a packet train consists of shaped traffic that has been sent with the objective of depleting all tokens at the shaper and saturating the shaper's queue.

We extend our assumption to consider a packet train comprising shaped traffic intentionally sent to deplete all tokens at the shaper and fill its queue to capacity. Consequently, each subsequent P_σ encounters some queuing delay or face a drop if the queue is already full. In contrast, under the assumption $r < \nu$, P_νs are promptly served without any delays. Thus, there exists a positive delay between each packet of P_σ and P_ν, demonstrating that the traffic shaper is a discriminator.

2) $\delta_{min} \geq \Theta$ and $\forall i, d_{MB}(P_B^i) \leq \theta$: In our analysis, we maintain the assumption of constant normal forwarding and routing for all packets. Thus, the queueing delay remains the only contributing factor in our calculation of δ_i. Given $r < \nu$, P_{ν_i}s are serviced immediately, implying $d_{MB}(P_{\nu_i}) = 0$. Consequently, for the calculation of δ_i, our focus is on determining $d_{MB}(P_{\sigma_i})$.

Under the assumption of a full queue and depletion of all tokens, each incoming P_σ results in δ_i being either $+\infty$ (indicating a drop due to the full queue) or $(|Q| - 1)|P|/\sigma$ (representing the time required to transmit all packets ahead of the last queued packet). As $|Q|$ remains unknown to $\mathcal{SND}$ and $\mathcal{RSV}$, similar to our approach in the analysis presented in Section IV.B, we employ the concept of super-packets to ensure that the analysis is independent of this variable.

Furthermore, our objective is to find the minimum λ such that $d_{MB}(\dot{P}_{\sigma_j}) = +\infty$ for all j's. Given that the loss rate for P_σs is $1 - \sigma/r$, employing a similar analysis to the one presented in Section IV.B, the following inequality validates the existence of such λ:

$$\lambda \geq \left\lceil \frac{1}{1 - \frac{\sigma}{r}} \right\rceil$$

Lastly, since $r < \nu$, in the absence of cross traffic, P_νs are never lost, hence $\forall j, d_{MB}(\dot{P}_{\nu_j}) = 0$. Therefore, $\forall i, d_{MB}(P_B^i) \leq \theta$ holds.

In conclusion, all three steps in the detection framework hold; therefore, traffic shaper is detectable.

VI. Detecting Traffic Prioritization

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. Strict Priority Queuing (SPQ) is one of the widely-known methods of providing traffic differentiation on edge networks [32].

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.

In this section, we show that SPQ is a discriminator and thus detectable.

A. Notations

- P_H: The packet (with size $|P|$) that is prioritized.
- P_L: The packet (with size $|P|$) that is not prioritized.
- Q_H, Q_L: High and low priority queues with their respective sizes: $|Q_H|, |Q_L|$.

[7] P_σs are constructed to match the alleged classification criteria at shaper.

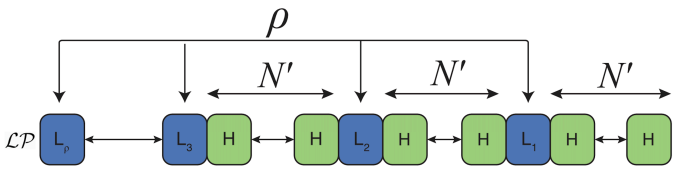

Fig. 3: Illustration of packet probe train to detect SPQ constructed in pre-probing phase for the low priority probing phase. (Green) High priority packets. (Blue) Low priority packets. L_is represent the packets of interest.

B. SPQ is Detectable

To confirm that SPQ is detectable, we show that the conditions in the framework detailed in Section III.E hold.

1) SPQ is a discriminator: To show that SPQ is a discriminator, we create interleaving patterns of packets in our probe sequences involving two distinct phases: low priority phase ($\mathcal{LP}$) and high priority phase ($\mathcal{HP}$). In $\mathcal{LP}$, the sender sends ρ low priority packets (referred to as packets of interest), each separated by N' high priority packets (referred to as separation packet train). Figure 3 illustrates the pattern used to construct the probe sequence during the low priority phase. In contrast, in $\mathcal{HP}$, the priority class of the packets is flipped. Specifically, ρ high priority packets of interest are sent, each separated by N' low priority packets.

Now we show that if $\tilde{P_A}$ and $\tilde{P_B}$ consist of the ρ low and high priority packets of interests during the $\mathcal{LP}$ and $\mathcal{HP}$ phases, respectively, then ($\tilde{P_A}$, $\tilde{P_B}$) is delay-discriminatory. To achieve this, our aim to create significant discriminatory effects is twofold: (1) during $\mathcal{HP}$, Q_H never to build up, hence no P_{H_i} is dropped at Q_H, (2) during $\mathcal{LP}$, Q_H never to be empty, hence few P_{L_i}s are either starved at Q_L or dropped when Q_L is full. We seek to determine if there exist values for ρ and N' that satisfy these objectives.

Assuming $r > Z$,[8] to accomplish these objectives, the following conditions must hold:

(i) High priority packets in $\mathcal{HP}$ are sent at the rate of $\frac{r}{N'+1}$. 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 of $\frac{N'}{N'+1}r$. Therefore, for Q_H to never be empty, the condition $\frac{N'}{N'+1}r \geq Z$ must be satisfied.

Combining conditions (i) and (ii), we have $\frac{r}{N'+1} \leq Z \leq \frac{N'}{N'+1}r$. We can now solve for N':

$$N' \geq \left\lceil \max\left\{\frac{r-Z}{Z}, \frac{Z}{r-Z}\right\} \right\rceil \tag{5}$$

Therefore, there exists a value for N' that satisfies both (i) and (ii). In this scenario, P_{H_i}s in $\mathcal{HP}$ are serviced immediately, while P_{L_i}s in $\mathcal{LP}$ experience starvation, hence delayed or dropped. This effect shows that SPQ is a discriminator.

2) $\delta_{min} \geq \Theta$ and $\forall i, d_{MB}(P_B^i) \leq \theta$: Assuming that (i) and (ii) are met, $\forall i, d_{MB}(P_B^i) \leq \theta$ holds, since all P_{H_i} in $\mathcal{HP}$ are served immediately, experiencing negligible delay.

[8]If $r \leq Z$, then there will be no observable discrimination by SPQ.

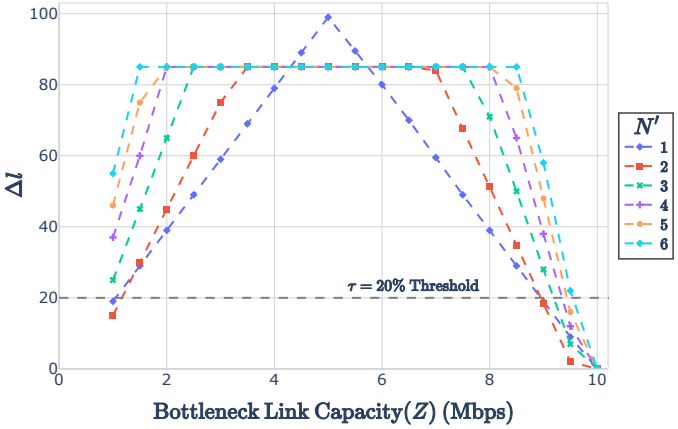

Fig. 4: The effects of the separation packet train length (N') and the sending-rate-to-bottleneck ratio (r/Z) on detection accuracy, where r = 10Mbps, $|Q_H|$ = $|Q_L|$= 75, and ρ = 1000, using a simple topology $\mathcal{SND} \rightarrow MB \rightarrow \mathcal{RCV}$.

As for $\delta_{min} \geq \Theta$, to compute δ_{min}, we need to calculate δ_i values. We make the assumption that normal forwarding and routing are constant for all packets, hence excluding them from the calculation of δ_i. Therefore, in $\mathcal{LP}$, $\delta_i = d_{MB}(P_{L_i})$.

Assuming that condition (ii) is met (hence P_{L_i}s experience starvation), P_{L_i}s are either dropped (if Q_L is full), or queued. In the former scenario, $\delta_i = +\infty$. In the latter case, δ_i is the time required to transmit all high priority packets in all separation packet trains in $\mathcal{LP}$. Specifically, $\delta_i \geq N' \cdot \rho \cdot t_T(P_H)$. Selecting suitably large values for N' and ρ will guarantee $\delta_{min} \geq \Theta$.

In conclusion, all three steps in the detection framework hold; therefore, SPQ is detectable.

C. Simulation Results

Through simulations (ns-3), we illustrate how network characteristics (r/z) and the parameter N' can impact the detectability of SPQ. We chose SPQ in our simulation because, unlike network compression and traffic shaping, multiple parameters can affect the tool's detection rate.

Inequality (5) shows the relationship between N' and r/Z in detecting SPQ. Figure 4 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 degrade the detection accuracy. 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.

VII. Internet Evaluation

In this section, we present our Internet evaluation conducted to confirm the effectiveness of our detection methods to detect network compression, traffic shaper, and SPQ.

A. Experiment Setup

The experiment environment and topology setup are depicted in Figure 5. To enable the effect of shaping and SPQ, we used a Cisco Catalyst 3750 switch, which has both QoS features: SPQ and traffic shaping. To enable the effect of network compression, we emulated network compression by implementing our own,using Click Modular Software Router [33]. The switch and receiver were located in our institution network. The senders, however, are a set of four geographically distributed remote PlanetLab [34] nodes, connected via the open Internet [35]. Notably, we reduced the sending transmission rate from the middlebox, thereby making $MB \rightarrow \mathcal{RCV}$ link the narrow link of the path. To confirm that this link is indeed the bottleneck, we used the capacity estimation tool Pathrate [24]. Pathrate has been proven experimentally to work well with PlanetLab nodes [36].

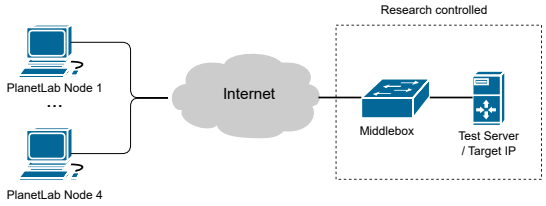

Fig. 5: Environment used in our experiments.

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 simulated a more common scenario in which such middleboxes are typically present in the edge network.

We used the following parameters in our Internet experiments: in the switch, we set Z = 2Mbps, and we used the default setting of 60 packets for the queue size in Cisco Catalyst 3750 [37]. VM Slices in PlanetLab nodes are typically rate-limited to 10Mbps [36]—thus, r = 10Mbps. We set the total number of packets of interest, ρ, to be 1000. Moreover, when constructing the SPQ detection probe, we used $N' = 4$. Lastly, we used $\lambda = 1$ in our calculations involving network compression and traffic shaping.

B. Results

We uniquely define an experiment scenario by pairing a remote PlanetLab node with the specific middlebox the end hosts are trying to detect. As outlined in Section III.C, each measurement scenario consists of 24 individual measurement conducted over a 24-hour period, executing one measurement per hour. We record the median Δl for each of the 24-round measurements and compare it against the detectable threshold.

In our experiment, we conducted each measurement scenario ten times. The normal distribution of recorded medians

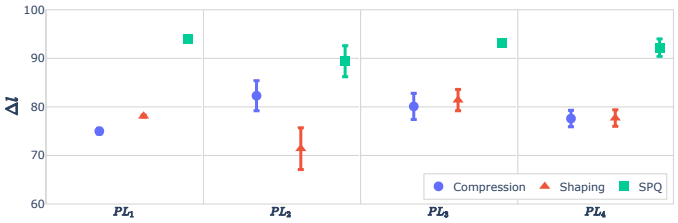

Fig. 6: Summary of the Internet experiments results in normal distribution form. All median values are above the 20% detectable threshold. (PL_1 : *planetlab2.cs.ucla.edu (USA),* PL_2 *:pl1.uni-rostock.de (Germany),* PL_3 *:pl1.sos.info.hiroshima-cu.ac.jp (Japan),* PL_4 *:planetlab.research.nicta.com.au (Australia)).*

for each scenario is depicted in Figure 6. The results confirmed the existence of a significant gap exceeding the 20% threshold, in the presence of the middlebox interference for all tested scenarios. On the other hand, in the absence of these middleboxes, none of the median Δls values surpassed the detectable threshold.

It is important to note that in the case of traffic shaping, our tool excludes the packets used to deplete the tokens at the shaper in the loss rate calculations during the detection phase.

Recalling from Section III.F, while a more robust detection approach could mitigate biases introduced by factors such as cross traffic, routing dynamics, and load balancing, in fewer rounds, the primary objective of this paper is to demonstrate the existence of a solution rather than an optimal one.

VIII. Concluding Remarks

In today's Internet landscape, middleboxes are ubiquitous and can significantly alter the handling of traffic streams. In this paper, we presented a general framework to detect a broad class of middleboxes. While this work constitutes a significant advance, we view it as just the initial step in a continued exploration of this research direction.

In this paper, we showed an example of one abstract class—network discriminator, along with one solution to detect the abstract class. We also identified three middleboxes whose interference are described as this abstract class. Our ongoing research encompasses the exploration of various avenues within each of these categories. This includes the formulation of additional abstract classes, the exploration of alternative solutions for detecting discriminators, and the identification of more middleboxes that exhibit characteristics of discriminators.

Furthermore, our approach assumes a static third party middlebox whose discriminatory behavior remains unchanged during the course of the probing (measurement) phase. Our ongoing research investigates addressing dynamic environments where third parties actively attempt to evade detection by closely monitoring traffic, identifying measurement traffic, and potentially introducing bias into the measurement results.

REFERENCES

[1] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar, "Making middleboxes someone else's problem: Network processing as a cloud service," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 13–24, 2012.

[2] Z. Wang, Z. Qian, Q. Xu, Z. Mao, and M. Zhang, "An untold story of middleboxes in cellular networks," in *Proceedings of the ACM SIGCOMM 2011 Conference*, ser. SIGCOMM '11. New York, NY, USA: ACM, 2011, pp. 374–385. [Online]. Available: http://doi.acm.org/10.1145/2018436.2018479

[3] J. Sherry, S. Ratnasamy, and J. S. At, "A survey of enterprise middlebox deployments," 2012. [Online]. Available: https://www2.eecs.berkeley.edu/Pubs/TechRpts/2012/EECS-2012-24.pdf

[4] B. E. Carpenter and S. W. Brim, "RFC 3234: Middleboxes: Taxonomy and Issues," vol. 3234, pp. 1–27, 2002.

[5] V. Pournaghshband, A. Afanasyev, and P. L. Reiher, "End-to-end detection of compression of traffic flows by intermediaries," in *IEEE/IFIP Network Operations and Management Symposium*, ser. NOMS '14. IEEE, 2014.

[6] G. Detal, B. Hesmans, O. Bonaventure, Y. Vanaubel, and B. Donnet, "Revealing middlebox interference with tracebox," in *Proceedings of the 2013 Conference on Internet Measurement Conference*, ser. IMC '13. New York, NY, USA: ACM, 2013, pp. 1–8. [Online]. Available: http://doi.acm.org/10.1145/2504730.2504757

[7] 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.

[8] R. Ravaioli, G. Urvoy-Keller, and C. Barakat, "Towards a general solution for detecting traffic differentiation at the internet access," in *2015 27th International Teletraffic Congress*, Sept 2015, pp. 1–9.

[9] M. M. B. Tariq, M. Motiwala, and N. Feamster, "NANO: network access neutrality observatory," in *HotNets*. ACM SIGCOMM, 2008, pp. 127–132.

[10] M. Dischinger, M. Marcon, S. Guha, P. K. Gummadi, R. Mahajan, and S. Saroiu, "Glasnost: Enabling end users to detect traffic differentiation." in *NSDI*, 2010, pp. 405–418.

[11] A. Molavi Kakhki, A. Razaghpanah, A. Li, H. Koo, R. Golani, D. Choffnes, P. Gill, and A. Mislove, "Identifying traffic differentiation in mobile networks," in *Proceedings of the 2015 Internet Measurement Conference*, ser. IMC '15. New York, NY, USA: ACM, 2015, pp. 239–251. [Online]. Available: http://doi.acm.org/10.1145/2815675.2815691

[12] Z. Zhang, O. Mara, and K. Argyraki, "Network neutrality inference," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, pp. 63–74, aug 2014. [Online]. Available: http://doi.acm.org/10.1145/2740070.2626308

[13] M. Dischinger, A. Mislove, A. Haeberlen, and K. P. Gummadi, "Detecting bittorrent blocking," in *Proceedings of the 8th ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '08. New York, NY, USA: ACM, 2008, pp. 3–8. [Online]. Available: http://doi.acm.org/10.1145/1452520.1452523

[14] S. B. R. Beverly and A. Berger, "The Internet Is Not a Big Truck: Toward Quantifying Network Neutrality." *pp. 135–144., Springer Berlin Heidelberg*, 2007.

[15] F. Li, A. A. Niaki, D. Choffnes, P. Gill, and A. Mislove, "A large-scale analysis of deployed traffic differentiation practices," in *Proceedings of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '19. Association for Computing Machinery, 2019, p. 130–144.

[16] 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 *Proceedings of the 2016 ACM SIGCOMM Conference*, ser. SIGCOMM '16. ACM, 2016, p. 468–482.

[17] P. Kanuparthy and C. Dovrolis, "Shaperprobe: End-to-end detection of isp traffic shaping using active methods," in *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, ser. IMC '11. New York, NY, USA: ACM, 2011, pp. 473–482. [Online]. Available: http://doi.acm.org/10.1145/2068816.2068860

[18] Y. Zhang, Z. M. Mao, and M. Zhang, "Detecting traffic differentiation in backbone isps with netpolice," in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement*. ACM, 2009, pp. 103–115.

[19] 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.

[20] Kanuparthy and C. Dovrolis, "Diffprobe: Detecting isp service discrimination," in *2010 Proceedings IEEE INFOCOM*, Mar 2010, pp. 1–9.

[21] 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.

[22] V. Pournaghshband, "Identifying Traffic Prioritization on the Internet," in *Proceedings of IEEE International Conference on Metaverse Computing, Networking and Applications (IEEE MetaCom)*, ser. IEEE MetaCom '23, 2023.

[23] Cisco, "Qos: Congestion avoidance configuration guide," 2016. [Online]. Available: https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/qos_conavd/configuration/xe-16/qosconavdxe16book.pdf

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

[25] 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 *Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems*, ser. SIGMETRICS '04. ACM, 2004, pp. 73–84.

[26] 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.

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

[28] R. Friend and W. Simpson, "RFC1974: PPP Stac LZS Compression Protocol," 1996.

[29] D. Rand, "RFC1978: PPP Predictor Compression Protocol," 1996.

[30] A. Shacham, B. Monsour, R. Pereira, and M. Thomas, "RFC3173: IP payload compression protocol (IPComp)," 2001.

[31] R. Pereira, "RFC2394: IP payload compression using DEFLATE," 1998.

[32] 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.

[33] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, "The Click modular router," *Trans. on Comp. Systems*, vol. 18, no. 3, pp. 263–297, 2000.

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

[35] P. Kirth and V. Pournaghshband, "Pldetect: a testbed for middlebox detection using planetlab," in *EAI International Conference on Testbeds and Research Infrastructures for the Development of Networks Communities (TRIDENTCOM)*, 2019.

[36] S.-J. Lee, P. Sharma, S. Banerjee, S. Basu, and R. Fonseca, "Measuring bandwidth between planetlab nodes," in *Proceedings of the 6th International Conference on Passive and Active Network Measurement*, ser. PAM'05. Berlin, Heidelberg: Springer-Verlag, 2005, p. 292–305.

[37] 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.