August 2006

# A Double Horizon Defense Design for Robust Regulation of Malicious Traffic

Ying Xu
*University of Pennsylvania*, yingx@seas.upenn.edu

Roch A. Guérin
*University of Pennsylvania*, guerin@acm.org

## Recommended Citation

Ying Xu and Roch A. Guérin, "A Double Horizon Defense Design for Robust Regulation of Malicious Traffic", . August 2006.

# A Double Horizon Defense Design for Robust Regulation of Malicious Traffic

**Abstract**
Deploying defense mechanisms in routers holds promises for protecting infrastructure resources such as link bandwidth or router buffers against network Denial-of-Service (DoS) attacks. However, in spite of their efficacy against bruteforce flooding attacks, existing router-based defenses often perform poorly when confronted to more sophisticated attack strategies. This paper presents the design and evaluation of a system aimed at identifying and containing a broad range of malicious traffic patterns. Its main feature is a double time horizon architecture, designed for effective regulation of attacking traffic at both short and long time scales. The short horizon component responds quickly to transient traffic surges that deviate significantly from regular (TCP) traffic, i.e., attackers that generate sporadic short bursts. Conversely, the long horizon mechanism enforces strict conformance with normal TCP behavior, but does so by considering traffic over longer time periods, and is therefore aimed at attackers that attempt to capture a significant amount of link bandwidth. The performance of the proposed system was tested extensively. Our findings suggest that the implementation cost of the system is reasonable, and that it is indeed efficient against various types of attacks while remaining transparent to normal TCP users.

**Keywords**
Network, denial-of-service

**Comments**
In *Proceedings of the Second IEEE Communications Society/CreateNet International Conference on Security and Privacy in Communication Networks (SecureComm 2006)*.

# A Double Horizon Defense Design for Robust Regulation of Malicious Traffic

Ying Xu and Roch Guérin
Department of Electrical and Systems Engineering
University of Pennsylvania
Philadelphia, PA 19104
Emails: yingx@seas.upenn.edu, guerin@ee.upenn.edu

***Abstract*— Deploying defense mechanisms in routers holds promises for protecting infrastructure resources such as link bandwidth or router buffers against network Denial-of-Service (DoS) attacks. However, in spite of their efficacy against brute-force flooding attacks, existing router-based defenses often perform poorly when confronted to more sophisticated attack strategies. This paper presents the design and evaluation of a system aimed at identifying and containing a broad range of malicious traffic patterns. Its main feature is a double time horizon architecture, designed for effective regulation of attacking traffic at both short and long time scales. The short horizon component responds quickly to transient traffic surges that deviate significantly from regular (TCP) traffic, i.e., attackers that generate sporadic short bursts. Conversely, the long horizon mechanism enforces strict conformance with normal TCP behavior, but does so by considering traffic over longer time periods, and is therefore aimed at attackers that attempt to capture a significant amount of link bandwidth. The performance of the proposed system was tested extensively. Our findings suggest that the implementation cost of the system is reasonable, and that it is indeed efficient against various types of attacks while remaining transparent to normal TCP users.**

## I. Introduction and Background

Network Denial-of-Service (DoS) attacks, or bandwidth attacks, that directly target network infrastructure resources such as link bandwidth and/or router buffer are an important threat to the reliability of the Internet. Network DoS attacks are usually perpetrated by compromising a large number of "zombie" hosts and commandeering them to source flooding traffic towards a target link or network. During an attack, the amount of flooding traffic is often large enough to overwhelm the target, and disrupt the performance of normal users.

To combat network DoS attacks, it is natural to deploy mechanisms around critical network resources to serve as a first "line of defense". This approach is taken in *router-based* defenses [19], [18], [13], [14], [27] that directly monitor the usage of resources such as bandwidth or buffer. Router-based defenses rely on certain pre-defined "normal" system behaviors, usually expressed in terms of incoming traffic patterns and/or local router conditions, to identify DoS attack(ers). Activities deviating from this "norm", once detected, are automatically classified as malicious and subjected to further regulation. This facilitates *on-line detection and mitigation* of DoS attacks, which makes such defenses more desirable than more reactive solutions such as traceback [4], [22].

In spite of their efficacy against brute-force, high-rate attacks, one major concern with router-based DoS defenses is their *robustness* under more sophisticated real-world attacks. To shed some light on this issue, we explored in [23] the extent to which several existing router-based defenses could be defeated by attackers employing more complex traffic flooding strategies. Based on the granularity at which abnormal behaviors are defined, router-based defenses can be categorized into aggregate level defenses and flow level defenses. In [23], we focused on two specific systems, the Aggregate Congestion Control (ACC) [18] proposal and the RED preferential dropping (RED-PD) proposal [19], which are representative of aggregate level defenses and flow level defenses, respectively. The investigation revealed a number of vulnerabilities in both designs. In particular, because aggregate level defenses rely on coarse grain traffic descriptors, namely, "traffic aggregates," to detect attackers, they are often not effective at defending against more subtle malicious behaviors posed by individual users, or flows [1]. For example, the ACC system, which is designed to regulate bandwidth hogs that induce significant congestion, can be defeated by an attacker that carefully controls its rate increase to avoid ever triggering severe losses. In contrast, flow level defenses rely on fine-grain descriptors that specify the expected traffic profile of a well-behaving user. For example, the RED-PD system incorporates the notion of "TCP-friendliness," and regulates flows transmitting at a rate (abnormally) higher than a standard TCP flow. Unfortunately, though this is effective against high bandwidth attackers, it remains vulnerable to low-rate attackers that periodically blast short traffic bursts. The fact that bursty ON-OFF attackers can defeat a base RED-PD system was first identified in [17], with the "Shrew" attacker, and subsequently in [11]. Our study [23] confirmed these results, but focused on understanding the *reasons* behind these failures. Our findings suggest that the failure of the RED-PD defense stems from its slow response, as thwarting highly transient attackers calls for the rapid identification of suspicious traffic and the swift imposition of

This work was supported by NSF grant ITR00-85930.

[1] The traffic of an individual user is often represented by a "flow", which is a *fine-grain* traffic descriptor associated with a particular combination of source/destination addresses, port numbers and protocol ID.

steep penalties.

The goal of this paper is to leverage the understanding of [23] to build a router-based defense that remains effective under a broad range of attacks. Based on [23], we select flow level defenses that enforce standard TCP behavior as our starting point and seek to strengthen them along two main dimensions. First, the defense system should be *efficient* against a wide variety of attacks. Ideally, it should keep the impact of an attacker to a level similar to that of a "normal" TCP flow, *independent* of the flooding strategy. In addition, the defense system should be *transparent* to legitimate TCP users, i.e., impose minimal penalties on them. In order to understand how to build defenses that can fulfill these goals, we first evaluate a set of schemes that represent likely choices for a successful design. We find that performance is heavily dependent on the time period (horizon) used to detect malicious behaviors, as well as the mechanism employed to filter suspicious traffic. In particular, a short horizon mechanism coupled with a stringent regulation strategy, e.g., dropping all out-of-profile traffic, is very effective at eliminating malicious traffic. However, as flow rates commonly fluctuate over small time scales, such a scheme often ends up wrongly penalizing legitimate users. A long horizon mechanism that averages traffic over longer time intervals fares better in that respect, but is ineffective at throttling transient attackers, because it either fails to detect them or reacts too slowly to their presence.

To overcome this dilemma, we propose a *double horizon* defense architecture. The short horizon component relies on a traffic profile looser than the standard TCP behavior, and allows for quick reaction to transient and high intensity attacks without sacrificing transparency. However, this does not protect against steady and less aggressive attackers. This is the responsibility of the long horizon component that enforces strict TCP conformance, but over a longer time interval to avoid falsely identifying regular TCP rate fluctuations as malicious. We provide evidences in support of this design, and demonstrate that it not only meets our design goals, but also that it can be implemented using simple hash table data structures, with a processing and storage cost affordable even for very high link speeds.

The investigation carried out in this paper is a comprehensive exploration of the design space associated with building robust router-based defenses that remain efficient across a broad range of attacks. Specifically, the defense is able to significantly curtail the impact of an attacker to no more than that of a regular TCP user, no matter how it varies its traffic. This being said, attackers can still be successful in disrupting normal traffic if they manage to summon a large enough number of hosts, e.g., as in the case of BotNets attacks [15] or reflector attacks [21]. In such cases, it is virtually impossible to distinguish an attacking host from a legal host, solely on the basis of their individual traffic. Defending against such attacks calls for different measures to identify attackers. We discuss this further in Section VI, in which we argue that combining the defense mechanism proposed in this paper with certain aggregate mechanisms, e.g., the detection and regulation techniques used by the ACC defense, is a promising approach for dealing with *both* such large-scale DDoS attacks and the more pernicious attacks we focus on in this paper.

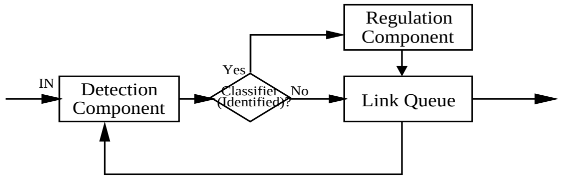


Fig. 1. Flow Level Defenses: Conceptual Representation

The rest of the paper is structured as follows. In Section II, we present the broad design space we consider for developing defense mechanisms. Section III explores different design choices within this space and motivates our choice of a double horizon design. Section IV provides specific details on the proposed design, analyzes its complexity, and discusses configuration issues. In Section V, we demonstrate that the design is effective against a wide variety of attacks in different configurations. We discuss related studies in Section VI, and conclude the paper in Section VII.

## II. Design Space and Options

In this section, we introduce a generic design framework for flow level defenses. This framework allows us to capture the major factors that affect defense performance, and to identify specific schemes of interest.

Flow level defenses rely on an a priori definition of the "normal" behavior of individual flows, which they constantly strive to enforce. Since TCP users dominate the Internet user community, it is reasonable to define "normal" user behavior based on how a TCP flow consumes link resources, i.e., to enforce TCP-friendliness. As shown in Fig. 1, a typical flow level defense consists of two major components: a detection component and a regulation component, which are connected through a classifier. A flow violating the standard TCP profile, once identified by the detector, will be classified as potentially malicious and have its traffic filtered inside the regulation component before entering the link queue. In this paper, we assume the defense systems relies on a RED queue [8] that implements random dropping. As we will discuss, this facilitates estimating the rate of TCP flows. In the following, we elaborate on the design choices for the detection and regulation components, respectively.

### *A. Detection Strategy*

The detection component measures the resource usage of all active flows traversing the link and identifies flows that do not conform to "normal" TCP behavior. Fulfilling this task requires two different operations: tracking the baseline behavior of a TCP flow and measuring the resource usage of incoming flows.

*1) Profiling TCP-friendly Behavior:* A common approach for characterizing normal TCP behavior is the "TCP-friendly" concept [7], which categorizes as conformant any flow sending

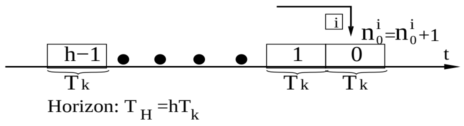


Fig. 2. Measuring Traffic: Illustration

at a lower rate than a long-lived TCP flow under the same conditions. The stable rate of a persistent TCP flow is well-known to be roughly equal to:

$$f(r,p) = \frac{\sqrt{1.5}}{r\sqrt{p}} \tag{1}$$

where $r$ and $p$ are the round-trip time and the steady state loss event rate experienced by that flow [7].

To identify non-conformant users, it is necessary to obtain an accurate estimation of $f(r,p)$. Since the parameter $r$ is typically pre-configured by the defense [2], estimating $p$ becomes the main challenge. There are two factors that make this task non-trivial. First, the loss levels seen by distinct flows may be different. Fortunately, this should be less of an issue for a RED queue, since RED drops packets randomly, and thus helps evenly distribute loss events across flows. Consequently, the loss event rate experienced by a single flow can be inferred from the overall link loss rate observed at the local router [10]. The second issue regarding estimating $p$ is the selection of an accurate observation duration $T_L$. As the overall link loss rate oscillates at small time scales, e.g., several round-trip times, we expect that $T_L$ should be reasonably large in order to filter out rate fluctuations. We found that when the link load is stationary, as long as $T_L$ is larger than a few hundreds $r$, the accuracy of the estimation is quite insensitive to the specific value of $T_L$. Therefore, we set the default $r$ to 40ms and use a fixed $T_L$ of 20 seconds in the rest of the paper. In practice, such a value would also be small enough to quickly detect variations in the steady state loss level, as the congestion level of most Internet paths usually stays unchanged for several minutes [28].

*2) Estimating Resource Usage of Flows:* The detection component also needs to be aware of the rate of incoming flows so as to detect malicious traffic. Estimating the rate of a flow is equivalent to estimating its traffic volume within a certain time period. Fig. 2 depicts an architecture for realizing this task. Specifically, the system time is divided into a series of consecutive *measurement intervals* with equal length of $T_k$ seconds. Without loss of generality, the current measurement interval is indexed with 0, and the previous with 1 and so on. Each interval keeps its own set of counters for recording the volume of transit flows. Let $n_k^i$ denote the value of the counter assigned to flow $i$ in the $k$th measurement interval. Then, the expected volume of traffic flow $i$ sends in $T_k$ seconds, denoted as $\hat{n}^i$, can be estimated using a moving average (MA) estimator [3] with a history length of $h$ intervals:

$$\hat{n}^i = \frac{1}{h}\sum_{k=0}^{h-1} n_k^i \tag{2}$$

The total time span over which $\hat{n}^i$ is estimated, i.e., $hT_k$, is denoted as the *time horizon* $T_H$. Irrespective of how $n_k^i$ is calculated, $T_H$ has a significant impact on how well the detection component can differentiate legitimate TCP flows from non-conformant flows. As with the link loss rate, the rate of TCP flows also exhibits natural fluctuations at small time scales (See Fig.3(b) in [24] for a demonstration). Hence, the detector may wrongly estimate the traffic volume of flows if that volume is measured over a relatively short horizon. Using a very long horizon alleviates this problem, but raises the risk of ignoring flows that significantly violate TCP behavior at short time scales. Therefore, it is crucial to select the "right" time horizon(s) to efficiently trade off transparency and accuracy.

We discuss next options for measuring traffic. A straightforward approach is to update $n_k^i$ upon each packet arrival. Obviously, this is not a scalable solution. We argue that a defense system should rely on *sampling* techniques to reduce the number of packets or flows that needs to be remembered. In this paper, we focus on a simple random sampling technique [4]. Random sampling selects every incoming packet with a certain probability $s$. As shown in Fig. 2, all counters are initialized to zero at the beginning of a measurement interval. A new sample of flow $i$ in the current interval increases the counter $n_0^i$ by 1, and also triggers the update of $\hat{n}^i$. By comparing $\hat{n}^i$ with $\hat{n}^T$, the maximum average number of samples a TCP-friendly flow would generate in the same period, one can decide whether a flow violates the standard TCP behavior. *A flow is identified as TCP-unfriendly if* $\hat{n}^i > \hat{n}^T = f(r,p)T_k s$ [5].

### *B. Regulation Strategy*

The role of the regulation component, is to filter malicious traffic and turn it into conformant traffic. Next, we introduce several regulation strategies that differ in both their reaction speed and their regulation intensity.

*1) Progressive Dropping With Delayed Response:* The first regulation mechanism we introduce tracks the amount of out-of-profile traffic generated by a non-conformant flow and drops its packets accordingly. In particular, upon the completion of *every* measurement interval, the regulator retrieves the latest

[2] We explore the impact of a pre-configured $r$ on TCP flows with different round-trip times in Section IV-B.

[3] Other moving average schemes with unequal weights, e.g., the exponential weighted moving average (EWMA) algorithm, can also be used. We found that the choice of a specific averaging scheme has minor impact on the performance differences across distinct defenses.

[4] There have been a number of recent works [6], [16], [12] that address the issue of how to apply advanced sampling schemes to measure traffic efficiently. Though these schemes can increase the accuracy of traffic measurement in particular instances, we expect the impact of a specific sampling technique to be smaller than that of the horizon setting. As we shall see, an effective defense scheme can be built upon a random sampling technique as long as it operates with an appropriate horizon(s).

[5] To account for variable size packets, $n_0^i$, $\hat{n}^i$ and $\hat{n}^T$ can all be byte counts.

measurement results $n_0^i$ and $\hat{n}^T$ from the detector, and assigns to flow $i$ a dropping probability $p_i$ equal to:

$$\left[1 - \frac{\hat{n}^T}{\hat{n}^i}\right]^+ \tag{3}$$

*throughout* the next interval. It is easy to see that when the system is stable ($f(r,p)$ is a constant), and there are no sampling errors, Eq. (3) computed *at the end* of a measurement interval is exactly equal to the fraction of excess traffic flow $i$ generates in the most recent horizon. Thus, by probabilistically dropping packets accordingly, the regulator attempts to filter out all excess traffic of a non-conformant flow and bring its rate back to $f(r,p)$.

We name the above regulation policy the "Progressive Dropping with Delayed Response" (*PD-DR*) strategy, as it progressively tracks down how much a flow violates the standard TCP behavior. For a "new" non-conformant flow appearing in the system, the PD-DR strategy exhibits a natural delay in applying its penalty, as obtaining an accurate estimation of the amount of excess traffic can only be achieved after that flow has been monitored for an *entire* horizon. Thus, we suspect when $T_H$ is relatively large, this type of defense may not effectively regulate malicious flows with a highly variable rate process. For instance, it is possible that when the regulator finally comes up with a correct dropping decision, the attacking flow has already disappeared, making the regulator's efforts futile.

*2) Cut-Tail Dropping With Instant Response:* The second strategy we consider attempts to overcome the intrinsic limitations of the PD-DR scheme, by reacting to suspicious traffic at the earliest possible time. To achieve a fast response, this strategy only monitors traffic seen in the current measurement interval, i.e., using a history length $h$ equal to 1. [6]. Moreover, instead of waiting until the end of a measurement interval, this regulation mechanism starts penalizing a flow *immediately* after it has been classified as malicious. Specifically, for any given flow $i$, the regulator compares $n_0^i$ with $n_T$ upon every update, and if $n_0^i > n_T$, *all* subsequent packets from flow $i$ are dropped for the rest of the interval. This process repeats anew in each measurement interval. In an ideal situation without sampling errors, this mechanism can also correctly filter out all excess traffic.

We name this regulation scheme the "Cut-tail Dropping with Instant Response" (*CT-IR*) strategy. Clearly, in order for a CT-IR regulator to respond quickly to sudden traffic violations, a small measurement interval length must be used. However, because of intrinsic TCP rate fluctuations, this is also likely to trigger frequent false detections and penalizations, which is detrimental to achieving a high degree of transparency.

*3) CT-IR with Slack:* To reduce the potential damage a CT-IR regulator could impose on regular TCP flows, we extend the basic CT-IR scheme by introducing a slack factor $\delta$ ($\delta \geq 1$). A regulator that employs both the CT-IR scheme and the slack only applies cut-tail dropping on a flow $i$ when $n_0^i$ exceeds an inflated TCP-friendly profile of $\delta \cdot n_T$. This enables the regulation component to focus on flows that significantly violate the standard TCP profile, and therefore makes it possible for a regulator to achieve both transparency and fast response even when a very small horizon length is used. On the other hand, as this scheme relies on a looser definition of "normal traffic", it alone would not be able to enforce strict TCP-friendliness.

We have thus far reviewed several possible design choices for flow level defenses. Specifically, defenses detect malicious traffic by measuring the resource consumption of flows and comparing it with the maximally allowable TCP-friendly rate estimated at the local link. Once identified, a flow can be regulated according to either one of three strategies, namely, PD-DR, CT-IR, or CT-IR with slack.

## III. Evaluation of Design Choices

In this section, we evaluate the performance of the aforementioned defense schemes through a number of case studies. Our evaluation is qualitative in nature, as our focus is on comparing the relative performance of these schemes and exploring the impact of various design parameters, so as to understand how to build a successful system. For both the PD-DR and the CT-IR schemes, we let $T_H = T_K$ ($h = 1$), and mainly explore the influence of the horizon length $T_H$. [7]. As for the CT-IR with slack, we configure its horizon length to $r = 40$ms, i.e., one round-trip time of a standard TCP flow, and investigate how its performance changes with the slack $\delta$. For this initial investigation, all schemes use a sampling probability of $s = 0.1$.

We carry out our investigation using the topology of Fig. 3, which consists of one bottleneck link with a capacity of $C$ Mb/sec. The RED queue on the link has configurable parameters $min_{th}$, $max_{th}$ and $q_{weight}$, and has a maximal size of $B$ packets. The link is shared by $N_T$ long-lived TCP flows (with no window size limitations) and $N_A$ malicious flows, and we test the efficacy of the defenses in throttling attacking traffic on that link. The access links of all TCP users are assumed to also have a speed of $C$ Mb/sec and a delay of $y$ms. We assume that each attacking source is connected over a high-speed access link of, say 622 Mb/sec, and with a 5ms delay. Parameters such as $N_A$, $N_T$, $C$ $y$, as well as $min_{th}$ and $max_{th}$ can all be varied. Unless stated explicitly, we use the following default setting: $N_A = 1$ (a single attacker), $N_T = 16$, $y = 10$ms (40ms round-trip propagation delay), $C = 45$Mb/sec, and consider TCP Reno users sending $500$ byte packets. In addition, the default configuration of the RED queue assumes $B = 1000$ packets, $min_{th} = 100$ packets, $max_{th} = 300$ packets and $q_{weight} = 0.002$. The effect of changing $min_{th}$ and $max_{th}$ will be discussed later.

[6] Such a regulation mechanism may also be configured with $h > 1$. However, this will slow down the reaction speed and can be shown to reduce the regulation scheme's efficacy against bursty attackers. We therefore do not consider such cases.

[7] Though the PD-DR scheme can be configured with $h > 1$, the specific value of $h$ was found to have a minor impact for a given $T_H$.

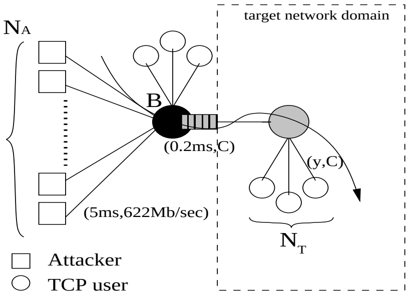
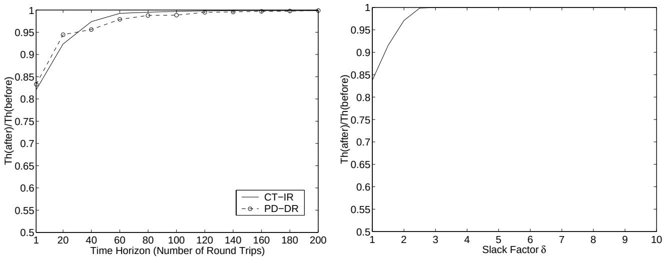
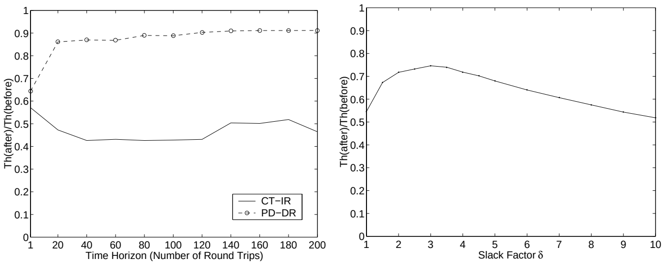


Fig. 3. Topology



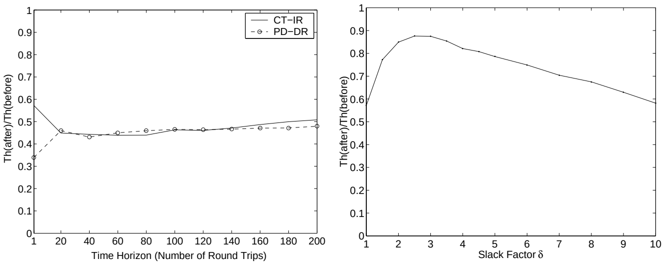
(a) Impact of $T_H$ (b) Impact of $\delta$

Fig. 4. Transparency to TCP flows



(a) Impact of $T_H$ (b) Impact of $\delta$

Fig. 5. Efficacy Against One Persistent Attacker



(a) Impact of $T_H$ (b) Impact of $\delta$

Fig. 6. Efficacy Against One Transient Attacker

### A. Scenario 1: Transparency

The first scenario we study, is a configuration similar to the default setting but with no attacker ($N_A = 0$). In this configuration, we are mainly concerned with whether the defense system can stay transparent and allow TCP traffic to consume the full link capacity. Fig. 4(a) reports, for both the PD-DR and the CT-IR schemes, the ratio of the average TCP throughput after and before the defense is turned on. A first observation from the figure is that the horizon length $T_H$ has a dominant impact on the level of transparency of a defense. In contrast, the choice of the specific scheme has little effect. In particular, when the horizon length is small, e.g., $T_H = 1r$, activating either defense causes TCP flows to lose over $15\%$ of their original throughput, while this amount diminishes to less than $1\%$ when $T_H$ exceeds $100r$. Overall speaking, both schemes perform similarly for a given $T_H$.

The poor transparency seen at small horizons is in part due to intrinsic TCP rate fluctuations. As discussed in Section II-A.2, TCP traffic oscillates substantially over small time horizons, which makes measurements collected over these intervals less representative of the real long term rate of a TCP flow. Another contributing factor is the use of random sampling. Specifically, a flow with a stable long-term rate is expected to send fewer packets in a shorter measurement interval. According to the central limit theorem, the number of samples collected over such a period is, therefore, expected to have a larger variance. When combining these two effects, it is not difficult to see that a smaller horizon results in a higher degree of rate estimation errors. This in turn translates into more aggressive filtering, which prevents TCP flows from fully saturating the link.

Fig. 4(b) plots the throughput ratio of TCP flows as a function of the slack factor $\delta$, for the extended CT-IR strategy. As shown in the figure, a large $\delta$, e.g., $\delta > 2$ helps TCP flows retain most of their initial throughput. This is because a larger $\delta$ corresponds to a looser rate constraint, and thus reduces the probability of false classification. In short, Fig. 4(b) validates the use of a loose traffic profile as another means for improving transparency.

### B. Scenario 2: One Persistent Attacker

We now evaluate the performance of the defense when adding a *single* attacker. We first test a constant bit rate (CBR) attacking source flooding at $1.5C$. This type of attacker captures the brute-force behavior of many typical attacks. Fig. 5(a) compares the PD-DR scheme and the CT-IR scheme. As one can observe, the two curves quickly diverge as $T_H$ increases. The PD-DR scheme causes an initial TCP throughput drop of more than $30\%$ when $T_H = 1r$. This relatively large drop is again mainly due to improper classification and therefore penalization of TCP flows. The residual attacking traffic surviving the defense also has an impact, but this impact is minor as more than $90\%$ of the flooding traffic was dropped. When $T_H \geq 140r$, the impact of mis-classification becomes negligible and the defense succeeds in keeping throughput reduction below $10\%$. The CT-IR defense, in contrast, shows a very different behavior. Even if it starts at a similar level of throughput reduction, this value actually increases with $T_H$. We verified that this was not because the defense could not identify the attacker and filter its traffic. The actual cause is that CT-IR does not continuously penalize identified flows, but allows a chunk of traffic to enter the queue before activating the cut-tail dropping. As this amount is roughly equal to

$f(r,p)T_H$ (proportional to $T_H$), a longer horizon allows an attacker to dump a larger burst within every measurement interval. Since this burst arrives with a high rate of $1.5C$, it triggers heavy losses and greatly degrades TCP performance.

Fig. 5(b) further evaluates a slack CT-IR defense. As we can observe, increasing $\delta$ initially helps improve TCP performance, since using a $\delta$ somewhat larger than $1$ improves transparency without greatly weakening the defense's ability to filter out-of-profile traffic. However, this trends reverses when $\delta > 3$, as the release of more attacking traffic induces more disruptions than what can be compensated for by the lower undue penalization on TCP. This explains the "hump" in the figure, which prevents TCP flows from attaining more than $75\%$ of their original throughput. As a result, the overall performance of a slack CT-IR defense remains worse than that of a PD-DR defense.

### C. Scenario 3: One Transient Attacker

In this sub-section, we consider a highly transient ON-OFF attacker that floods at $1.5C$ for $40$msec, and then stays idle for $960$msec before turning active again. As the attacker's 1-second attack cycle synchronizes with the TCP retransmission timeout duration, it can frequently force TCP flows into timeout. In [17], it was shown that this type of attacker can almost shut off TCP flows without being detected by RED-PD. We therefore, intend to test if one of the three defense strategies can be used to throttle this type of attacker.

From Fig. 6(a), we see that neither the CT-IR scheme nor the PD-DR scheme is efficient in thwarting the attacker. By checking trace data, we found that the CT-IR defense did filter out most of the attacking traffic when $T_H$ is small, e.g., $T_H = 1r$, primarily because of its fast response speed. However, this was negated by its poor transparency, which when combined with the remaining attacking traffic, caused TCP flows to lose more than $40\%$ of their initial throughput. Compared with the CT-IR scheme, the PD-DR defense responds even slower for a given $T_H$, yet exhibits similar transparency issues. Hence, it results in even worse TCP performance. When $T_H$ becomes large, both schemes perform poorly in regulating the attacker. This is because the long-term rate of the attacker is merely $0.06C$, a value similar to that of a standard TCP flow in steady state. Consequently, most of the attacking traffic manages to evade the defense and significantly reduce TCP throughput.

Fig. 6(b) identifies the slack CT-IR mechanism running at $T_H = 1r$ as the best defense scheme among the three when dealing with bursty ON-OFF attackers. In particular, its short horizon length enables the defense to rapidly react to transient traffic surges, while the inflated traffic profile significantly reduces the possibility of wrongly penalizing legitimate TCP users. As a result, the defense is able to drop most of the attacking traffic without degrading TCP performance. Fig. 6(b) also shows there exists a broad range of values for $\delta$ that yield a relatively small throughput reduction. For instance, TCP throughput decrease can be kept within $25\%$ for $\delta$ between $1.5$ and $5.5$. As we will see later, such insensitivity facilitates the selection of a default $\delta$.

### D. Towards a Double Horizon Design

Our investigation so far has revealed that no single defense scheme is capable of being both efficient against all attacks we consider and transparent to legitimate TCP users. Specifically, achieving transparency usually calls for a defense that either maintains a relatively long horizon (the PD-DR or the CT-IR scheme), or that relies on a slackened traffic profile (the slack CT-IR scheme). However, using a long horizon limits visibility at short time scales, and as a result both the PD-DR and the CT-IR schemes are unable to detect low-rate attackers that significantly violate TCP behavior over short time periods. The slack CT-IR scheme fares better in this respect, but its use of a slack factor allows out-of-profile traffic to pass through the regulator. This translates into poor performance when dealing with persistent high-rate attackers. Efficiently throttling this type of attacker calls for a long horizon mechanism that continuously filters traffic, such as the PD-DR mechanism.

Because both the PD-DR scheme and the slack CT-IR scheme achieve transparency and efficiency for different types of attacks, it seems appealing to combine them to leverage their complementary strengths. In the next section, we present the design of such a hybrid architecture.

## IV. A Double Horizon Defense Design

In this section, we review the architecture design and configuration of our double horizon defense. Due to space constraint, we only highlight the main ideas. Full details can be found in an extended technical report [24].

### A. Overview of Architecture Design

Our double horizon design includes two different mechanisms: the slack CT-IR scheme and the PD-DR scheme. The goal of the slack CT-IR scheme is to throttle short term traffic spikes. Hence, we configure its horizon length to $1r$ to ensure a fast response speed. In contrast, the PD-DR scheme should rely on a relatively long horizon to enforce strict long-term TCP behavior. In our design, we assume $T_K = kr$ so that it has a total horizon length of $hk$ round-trips. In the rest of the paper, we denote these two mechanisms as the "short horizon mechanism" and the "long horizon mechanism" respectively.

Both the short horizon and the long horizon mechanisms calibrate their baseline behavior using the estimated loss rate $p$ provided by the router queue. In all other respects, the two mechanisms operate *independently*, as they rely on separate sampling processes and generate their own dropping decisions. Both mechanisms track active flows using a standard *hash table* data structure, in which each flow is represented by a specific flow entry. Fig. 7 shows sample formats for the short horizon flow entry (top) and the long horizon flow entry (bottom). Specifically, a flow is characterized by a flow ID field (FID), which is a hash value computed over the source and destination addresses of its packets. The short horizon flow entry uses one counter to record $n_0^i$, while the long horizon flow entry contains $(h+1)$ counters, one for recording $\hat{n}^i$, and the others reserved for the $h$ most recent measurement intervals. For a newly arrived packet, the system checks both

| FID | rnd | h | k | $n^i_0$ |
|---|---|---|---|---|
| 4 | 2 | B(h,k) | | 2 |

| FID | rnd | h | k | $n^i$ | 0 | 1 | ● ● ● | h−1 |
|---|---|---|---|---|---|---|---|---|
| 4 | 2 | B(h,k) | | 3 | 2 | 2 | | 2 |

Fig. 7. Flow Entry Format: Short Horizon and Long Horizon

hash tables, using the packet's FID, to determine whether it should be dropped by either of them. A packet passes the defense *if and only if* it survives both mechanisms.

The 3-tuple <**rnd,h,k**> in each flow entry records the *most recent* horizon, measurement interval and round-trip in which a packet of that flow was seen. Based on such information, we proposed in [24] a simple strategy to clear obsolete counters and flow entries that do not contain information in the current time horizon. The strategy acts only upon new packet arrivals, thus avoids synchronized operations at the end of each measurement interval.

*1) Overall Complexity:* We have implemented the proposed design into the NS-2 simulator [1]. As with hash tables, our design has an *O(1)* packet processing complexity. Specifically, every packet arrival incurs at most 1 `read` operation, for retrieving corresponding flow entries[8], and at most 1 `write` operation, for updating specific counters or purging the entire flow entry.

The main overhead of our design lies in the amount of memory needed for holding flow entries. Specifically, we dimension the hash table based on the *maximal* number of valid flows it would accommodate at $100\%$ link utilization. Since the number of distinct flows reaches its maximum when every sample maps to a different flow, the maximal number of flow entries needed for the short horizon and the long horizon hash tables are $Cs_s r/P$ and $Cs_l(h+1)kr/P$ respectively, where $s_s$ and $s_l$ are their sampling rates, and $P$ is the average packet size. We specify the resource usage of both types of flow entries in Fig. 7. Specifically, we assign 4 bytes to the FID field, which is enough to represent 4 billion different flows. Each counter is assigned 2 bytes, except for the counter used for $\hat{n}^i$, which is allocated 3 bytes. We use 2 bytes to store the **rnd** field. For the **h** and the **k** fields, using a total of $b(h,k) = \lceil \log_2 hk \rceil + 1$ bits, so that $B(h,k) = \lceil b(h,k)/8 \rceil$ bytes is sufficient. The total storage requirement of the two hash tables in bytes, can therefore be obtained by multiplying the maximum expected number of flow entries together with the amount of memory each entry occupies, that is:

$$\begin{aligned} M(h,k,s_s,s_l) &= (9 + B(h,k) + 2h)Cs_l(h+1)kr/P \\ &\quad + (8 + B(h,k))Cs_s r/P \end{aligned} \qquad (4)$$

### B. Parameter Selection and Validation

We now summarize how we select the default parameters. Our main goal is to ensure adequate transparency for regular TCP flows, while keeping memory requirements affordable. We contend that ensuring transparency calls for a rate of loss events caused by the defense, denoted as $p^f$, to be *small* compared with $p$, the stable loss event rate when there is no defense. In [24], we estimated $p^f$ based on a simplified stationary model ([3], [7]), which assumes a TCP window size uniformly distributed within: $\left[\lfloor \frac{2}{3} f(r,p) r \rfloor, \lceil \frac{4}{3} f(r,p) r \rceil\right]$. Our analysis showed that increasing the sampling rates $s_s$ and $s_l$, using a larger slack $\delta$, and selecting a longer horizon length $hk$ all help to reduce $p^f$. This is consistent with the qualitative results observed in Section III-A. However, based on Eq.(4), choosing larger values of $s_s$, $s_l$ and $hk$ is at a cost of consuming more memory. A key finding in [24] is that selecting a very large $s_l$, e.g., $s_l > 0.1$ is of limited benefit in terms of memory usage, as the higher sampling rate translates into only a modest reduction in the minimal horizon length $hk$ needed to achieve a given $p^f$. This is again due to intrinsic TCP rate fluctuations, which can only be averaged out by using a sufficiently long horizon, e.g., over a hundred $r$. Hence, we select $s_l = 0.01$ and $hk = 500$, which is further split into the optimal combination of $h = 2$ and $k = 250$. As for the short horizon mechanism, we select a large sampling rate of $s_s = 0.5$, which allows us to use a relatively small $\delta$, i.e., $\delta = 3.0$, to achieve sufficient transparency. Using a large $s_s$ only has a minor effect on the overall memory cost, as the major consumption comes from the long horizon hash table. We summarize the default parameter setting in Fig. 8.

| Long Horizon | | | Short Horizon | | Baseline | |
|---|---|---|---|---|---|---|
| $s_l$ | $h$ | $k$ | $s_s$ | $\delta$ | $r$ | $T_L$ |
| 0.01 | 2 | 250 | 0.5 | 3.0 | 40ms | 20sec |

Fig. 8. Default Parameter Setting

*1) Validation of Memory Consumption:* Based on Eq.(4), Fig. 8, and assuming an average packet size $P$ of $500$ bytes, our default defense design needs $(0.001175C)$ bytes to store flow information. This is a value much smaller than today's router buffer setting. According to [5], [3], most existing router designs use $(RTTC/8)$ bytes buffer, where $RTT$ is the estimated round-trip time of a flow and is usually assumed to be 250ms. This results in a buffer size of $(0.03125C)$ bytes. Our design only requires less than $4\%$ of this amount [9]. For instance, our design requires less than 12MB of storage for an OC-192 (10Gbps) link,.

*2) Validation of Transparency:* We evaluated the impact of the default defense on TCP flows in various settings. Results are reported in what follows as well as in [24]. We first consider the default topology, and vary $N_T$, the number of standard TCP flows, from $4$ to $320$. This yields a link loss rate $p$ ranging from $0.01\%$ to $5\%$. Fig. 9(a) plots the resulting TCP throughput ratio as a function of $p$. The figure shows that TCP throughput reduction is less than $5\%$ in all scenarios, and is less than $1\%$ when $N_T > 8$. The throughput loss is higher for smaller values of $N_T$ because in these cases, the chance that multiple TCP flows are simultaneously penalized is higher.

[8] We assume a CAM-based implementation of the hash tables as in [20].

[9] To achieve efficiency at ultra-high link speeds, e.g., $160Gb/sec$ or beyond, [3] advocates using a buffer size that is as small as $1-2\%$ of what is adopted in current router desi gns. Investigating how to make our design efficient for such high speeds is left for future work.

Real Internet traffic exhibits a wide range of round-trip latencies [9]. We consider next a scenario where TCP flows with heterogeneous round-trip propagation delays are multiplexed. Specifically, we consider aggregating $46$ TCP flows, which are divided into $23$ groups with their round-trip delays evenly distributed between $20$ms and $460$ms. Fig. 9(b) shows for each group, the throughput before and after turning on the defense. It is easy to observe that the defense heavily penalized flows that were more aggressive than standard TCP flows, e.g., (group 1) TCP flows with a $20$ms round-trip delay. Nevertheless, the link bandwidth released by those flows was mostly absorbed by other groups of less aggressive TCP flows. Consequently, the overall TCP throughput drop is only a minor amount of $1.2\%$. In [24], we further studied the effect of varying the target round-trip time $r$. The results indicate that increasing $r$ tends to distribute link capacity more evenly among flows, i.e., towards the max-min fair allocation, but also results in larger throughput reductions as a wider range of flows are regulated. From the perspective of sustaining high throughput, the current choice of $r$ appears to be reasonable.

We have also tested the default parameter choice in two other configurations, one involving the aggregation of *web-like* traffic, and the other involving a topology with *multiple bottlenecks*. In [24], we showed that in both scenarios, this default configuration results in only very limited throughput reduction for TCP flows.

## V. Efficacy of the Double Horizon Design

This section is devoted to the evaluation of the defense capabilities of the proposed double horizon design, as well as comparing its efficiency to that of another well-known flow level defense, namely, the RED-PD system. In contrast to Section III, the evaluation in this section is *quantitative* in nature. In particular, our goal is to measure the extent to which the defense system can regulate attacking traffic, based on the notion of *"impact factor"*.

### A. Performance Metrics

The "impact factor" is a measure that captures the average impact of a single attacking flow, in terms of the equivalent number of regular TCP flows that would have the same effect. Specifically, assuming that launching $N_A$ attackers on a link with $N_T$ *standard* TCP flows yields a TCP throughput ratio of $TH_r$, i.e., the total TCP throughput is reduced by $100(1 - TH_r)\%$. If further assuming that standard TCP flows share link capacity in an equal manner, this level of performance degradation can also be achieved by adding $N_T^{'} = N_T(1/TH_r - 1)$ standard TCP flows. The impact factor $\mathcal{F}$, is thus defined as the ratio of $N_T^{'}$ over $N_A$:

$$\mathcal{F} = \frac{N_T}{N_A}\left(\frac{1}{TH_r} - 1\right) \tag{5}$$

Our implicit notion of an ideal defense system is that it should limit the impact of any attacking flow to no more than that of a normal TCP flow. This translates into a requirement for a defense to achieve $\mathcal{F} \leq 1$. When this is not feasible, the closer $\mathcal{F}$ is to 1, the better the defense.

### B. Defense Efficacy: Single Attacking Flow

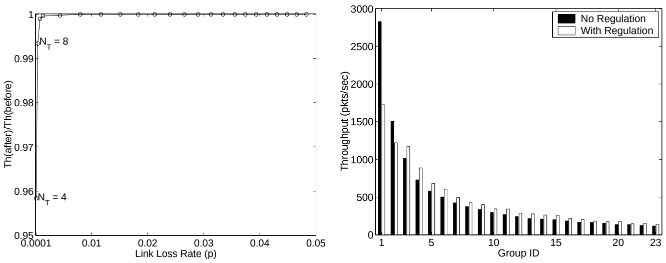
In the following, we explore the efficacy of the double horizon design based on the impact factor it achieves. As a benchmark, we start with the default configuration of Section III where there is only a single attacking flow. For comparison purposes, we also measure the impact factor of the RED-PD defense.

*1) Attacking Schemes:* We characterize the traffic pattern of a single attacking source using a generic ON-OFF model. An ON-OFF source can be represented by the three-tuple: $< R, b, I >$, where $b$ and $I$ are the lengths of its active (ON) and idle (OFF) periods, and $R$ specifies the rate at which the source sends traffic when active. Despite its simplicity, this ON-OFF model captures a broad range of traffic patterns. In our experiments, we tested two specific situations. Our first scenario assumes a basic CBR attacker $(I = 0)$ and varies its rate $R$ from $0.05C$ to $1.5C$. In the second scenario, we fix $R$ to $1.5C$ and set $(b + I)$ to 1 second, while increasing $b$ from 10ms to 1000ms. This gives us a family of ON-OFF sources that exhibit different levels of burstiness in their traffic. Particularly, when $b$ is small, these sources are essentially the bursty "Shrew" attackers we tested before. At the other extreme, when $b = 1000ms$ the source always stays "ON" and becomes a CBR attacker [10].

*2) CBR Attacker:* Fig. 10(a) reports the impact factor of a CBR attacker with a variable flooding rate, for both the double horizon defense and the RED-PD defense. The figure shows that the impact factor peaks at $1.3$ for $R = 0.15C$ and is smaller than $1$ after that, which shows that the defense was reasonably successful in limiting the impact of the attacker to a level close to that of a regular TCP flow. This value is still a bit larger than $1$, because the standard TCP profile, i.e., $f(r, p)$, slightly overestimates the actual TCP sending rate by not considering time-out events [7]. Another interesting phenomenon is that when $R$ becomes sufficiently large, e.g., larger than $0.3C$, the impact of the attacker actually diminishes. This is because the attacking traffic then triggers both the short and long horizon mechanisms, and is thus penalized twice. As the amount of extra penalty increases with the degree of violation, an attacker with a higher rate injects a smaller amount of traffic into the queue. We believe this to be a desirable property, as it discourages an attacker from using high flooding rates. Fig. 10(a) also plots the impact factor of the attacker under the RED-PD defense. As we can observe, the RED-PD defense can also limit the impact of the attacker below a certain level. However, the filtering of out-of-profile traffic is not as effective as in our design, as $\mathcal{F}$ can reach a value greater than 2. Also, this value stays roughly constant as $R$ increases. This is because RED-PD relies only on a long horizon mechanism to filter traffic, and thus does not apply extra penalty on attackers with higher rates.

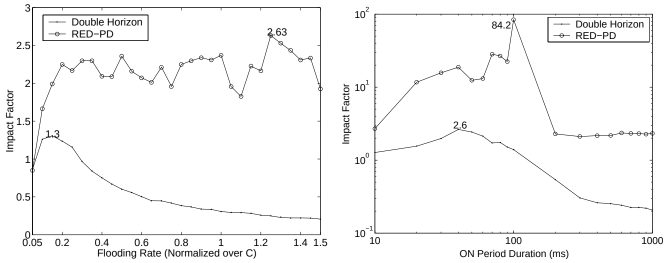
*3) ON-OFF Attacker:* Results for an ON-OFF attacker are reported in Fig. 10(b), via a log-log plot of the impact

[10] There is another case where both $R$ and $b/I$ are fixed, but $b$ varies. Given the minor effect of varying $b$, we omit the results here.



(a) Homogeneous TCP Flows (b) Heterogeneous TCP Flows

Fig. 9. Validation of Transparency



(a) Varying $R$ (CBR) (b) Varying $b$ ($b + I = 1$ sec)

Fig. 10. Impact of A Single Attacking Source

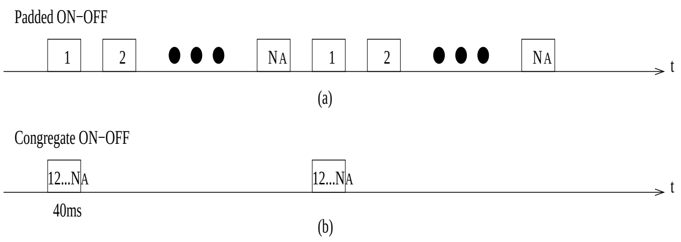
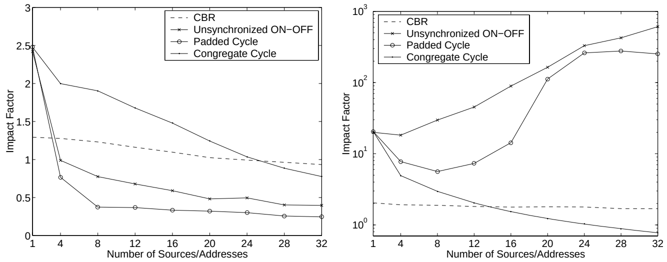


Fig. 11. Synchronized Multi-Address Flooding Strategies



(a) Double Horizon (b) RED-PD

Fig. 12. Impact of Multiple Attacking Sources/Addresses

factor. The figure demonstrates that the RED-PD system performs poorly when dealing with highly transient attackers. In particular, the impact factor is larger than $10$ when $b \in [20\text{msec}, 100\text{msec}]$, and peaks at a value about $90$. This confirms earlier results in [17], [23]. As pointed out in [23], the RED-PD design cannot effectively penalize bursty ON-OFF attackers because of its slow response speed. In contrast, the double horizon design reduces the impact of such attackers to a level an order of magnitude smaller. Specifically, when $b$ is small, it is the short horizon mechanism that acts against transient traffic violations. When $b$ increases from $10$msec, the impact factor initially grows as more traffic is allowed to enter the queue. However, when $b$ exceeds $40$msec, the attacker also triggers the long-horizon defense and gets penalized by both mechanisms, which causes $\mathcal{F}$ to quickly diminish to less than $1$. The peak value of $\mathcal{F}$ in this case is about $2.5$, which is larger than that in the CBR case. This implies periodically sending a large burst has a greater, albeit bounded, impact than continuously loading the system.

### *C. Defense Efficacy: Multiple Attacking Flows*

We now extend our study to cases where there are multiple attacking flows involved in the attack. In practice, this can happen in two different scenarios. First, an attacker can commandeer multiple compromised hosts to launch distributed DoS (DDoS) attacks, in which each host contributes one specific attacking flow. In addition, a single attacking host assuming multiple source addresses (legal or spoofed) can also generate traffic corresponding to multiple flows. This is because most flow-level defense systems, including our design and the RED-PD system, classify packets into flows based on a combination of source and destination addresses. An attacker flooding with multiple source addresses can thus fool those systems into believing that the attacking traffic belongs to distinct flows. In this sub-section, we consider both scenarios.

*1) Attacking Schemes:* We first consider the multi-host scenarios. As observed in V-B, a single host can have an impact factor larger than $1$ in two cases, either by using a CBR flow with a moderate rate of $0.15C$, or by crafting ON-OFF patterns with a small active period. We concentrate on these two strategies, because of their potentially bigger impact. Specifically, we assume that there are $N_A$ attacking hosts, all using one of the following strategies:

**i. Constant Bit Rate:** In this case, each attacking host floods with a constant bit rate of $R = 0.15C$.

**ii. Unsynchronized ON-OFF:** In this case, every host creates bursty ON-OFF patterns with $< R, b, I >$ equal to $< 1.5C, 40msec, 960msec >$. Attacking hosts are assumed to act independently and start with a random phase uniformly distributed within the 1-second period. We name such a scheme the "Unsynchronized ON-OFF" strategy.

We consider next an attacker that has access to $N_A$ different addresses, and floods at a fixed rate of $1.5C$ whenever active. In such a context, the attacker has the ability to coordinate timing between active addresses, or flows. For instance, by using one address for a certain period before switching to the next address, the attacker can craft synchronized traffic patterns across distinct flows. In our study, two address switching strategies are considered:

**iii. Padded Cycle:** The first strategy, the "padded cycle" strategy, is shown in Fig. 11(a). In this scenario, each address is used for $b = 40msec$ before the next address is activated, and there is a padded idle period of $\left[\frac{1}{N_A} - b\right]^+$ seconds between successive addresses. This strategy generates a series of staggered ON-OFF sources as it cycles through all $N_A$ addresses in a period of $\max(1, bN_A)$ seconds. When $N_A < 1/b$, the padded idle period ensures that the impact of every address is evenly distributed across the attack cycle.

**iv. Congregate Cycle:** Instead of distributing the effect of different addresses across the attack cycle, the second strategy, i.e., the "congregate cycle" strategy attempts to aggregate their impact in a short time period. As shown in Fig. 11(b), within a cycling round the attacker floods with each address for $b/N_A$ ($b = 40$msec) seconds consecutively. Upon the completion of one round, the attacker stays idle for $960$msec before blasting again, which creates a 1-second attack cycle. The rationale of this strategy, is to maximize the impact of every traffic burst. In particular, recall that if one address is used continuously for $40$msec, most of the associated traffic will be filtered. Hence, rather than distributing many such "weakened" bursts over time, as is the case in the "padded cycle" strategy, this strategy attempts to inject the entire burst into the defense by using a smaller "ON" period for each address.

*2) Results:* Fig. 12 reports the resulting $\mathcal{F}$ for cases where $N_A$ increases from 1 to $32$. Fig. 12(a) focuses on the double horizon defense, while Fig. 12(b) studies the RED-PD defense. The main observation from Fig. 12(a) is that all four strategies exhibit an $\mathcal{F}$ that decreases with $N_A$. Moreover, when $N_A$ becomes large, all curves settle at a level below 1. Among the three flooding strategies creating ON-OFF traffic patterns, the curve corresponding to the "Padded Cycle" strategy decays the fastest, In contrast, the "Congregate Cycle" strategy exhibits the slowest decay in $\mathcal{F}$. This suggests that TCP traffic is more susceptible to large bursts that are relatively far apart, than to many small, evenly distributed bursts. An off-line examination indicates that this is because a large burst can shut off a significant fraction of TCP flows by forcing them into timeout, while small bursts only decrease (halve) the TCP window size, which has a lesser effect. Nevertheless, when $N_A > 25$, even the "congregate" strategy ends up with an $\mathcal{F}$ smaller than 1. The "Unsynchronized ON-OFF" strategy has a behavior that is in between that of the above two strategies. This is due to the random activation of attacking flows, which results in the ON periods of some sources being spread apart, while others cluster together creating a congregate effect.

The results of Fig. 12(a) confirms the efficacy of the double horizon design. Fig. 12(b) paints an opposite picture for the RED-PD system. Specifically, as ON-OFF patterns are not effectively regulated, attackers using the "Unsynchronized ON-OFF" or the "Padded Cycle" strategies can achieve an impact factor larger than $200$. Only the "Congregate Cycle" strategy yields an $\mathcal{F}$ that actually decreases with $N_A$. This is because most of the attacking traffic already bypasses the defense when $N_A = 1$. As the strategy does not introduce "new" traffic, the average impact of each address diminishes as $N_A$ gets larger.

*3) Alternative Configurations:* Apart from the default configuration, we also evaluated the efficacy of our defense system in several other settings, e.g., multiplexing *web-like traffic*, aggregating *heterogeneous TCP flows*, as well as *changing link queue configurations*. We verified that in all these scenarios, the default defense can successfully identify all four attacking strategies, and limit their impact to a level similar to that of TCP-friendly traffic. We refer to [24] (Section V.D) for a complete presentation of results.

## VI. Related Works

In this section, we briefly survey related network DoS countermeasures, and compare them with the scheme proposed in this paper in terms of both efficacy and applicability.

Earlier works such as traceback [22], [4] attempt to identify the source(s) of attacking traffic in order to shut them off. Traceback mechanisms allow a victim to reconstruct the incoming path of attacking traffic, based on information elicited from intermediate routers. Traceback is, however, not effective in rapidly stopping an ongoing attack, as collecting enough information to identify the sources often takes a long time.

Router-based defenses explicitly tackle the issue of online throttling of DoS attacks. As previously mentioned, router-based defenses can be categorized into aggregate level defenses and flow level defenses. Aggregate level defenses rely on coarse-grain descriptor, namely, "traffic aggregates" to identify suspicious traffic. For example, during extreme congestion, the ACC system [18] classifies high-bandwidth aggregates sharing a common destination address (or address clusters) as malicious. This, as demonstrated in [18], enables the defense to detect and contain large-scale, brute-force DDoS attacks in many typical scenarios. The main challenge faced by aggregate level defenses, is the difficulty of defining suitable aggregates and their expected overall behavior as well as setting proper triggers for enacting traffic regulation. This makes such defenses vulnerable to sophisticated attackers relying on carefully crafted traffic patterns (bursts) [23]. In contrast, as flow-level schemes monitor every user for conformance with a particular profile, e.g., TCP friendliness, they are more effective against individual attackers. In particular, the double horizon defense proposed in this paper can successfully confine attackers so that their individual impact does not exceed that of a regular TCP flow. On the other hand, a large number of attackers, even if constrained to be TCP-friendly, can eventually overwhelm any flow level defense. Our contention is not only that the two represent very different types of attacks, but that it is possible to design a reasonably effective overall defense by combining efficient flow-level schemes, such as the one proposed in this paper, together with aggregate level mechanisms. For example, upon detecting that the double horizon defense alone fails to limit link congestion to a certain level, extra scanning could be activated to detect and regulate suspicious traffic aggregates that are likely associated with large-scale DDoS attacks. Validating the feasibility and efficacy of such a combination is part of the work we are currently pursuing.

Kill-Bots [15] is a server side defense mechanism aimed at thwarting CyberSlam, i.e., massive DDoS attacks targeting web server resources such as CPU, memory, disk and database bandwidth. Kill-Bots applies reverse Turing test such as CAPTCHA graphical tests [2] to distinguish zombie hosts (bots) from human users. Since performing these tests usually requires higher layer services such as HTTP, Kill-Bots cannot be used directly by routers to protect network infrastructure resources such as bandwidth or buffer.

SIFF [25] and TVA [26] are recent proposals that rely on the notion of "capabilities" to thwart DoS attacks. These proposals are designed based on a simple rationale, i.e., a sender should first gain permissions from the receiver before it can make any of its packets reach that receiver. Such permissions, represented by tokens, or capabilities, are special fields embedded inside every packet sent out by participating hosts. Routers supporting capabilities check incoming packets and drop all packets with invalid capabilities. Therefore, during an attack, all attacking traffic would be blocked unless the attacker could cheat the victim and obtain sending permissions. Existing capability-based solutions mostly focus on designing host and router functionalities for securely distributing and validating capabilities, but have not explicitly addressed how to efficiently issue capabilities to distinguish legal traffic from unwanted traffic. We expect that some of the understandings gained in this paper might help resolve this problem. For example, a profile-driven approach based on "TCP friendliness" could be used for issuing "fine-grained" capabilities [26] and rejecting out-of-profile traffic. Also, a system architecture similar to our defense, i.e., enforcing the standard profile at different time horizons, might help attain a reasonable tradeoff between transparency and efficiency. How to incorporate our design into capability-based framework, however, is beyond the scope of this paper.

## VII. Conclusions

In this paper, we presented a novel double horizon router defense for efficiently regulating a broad range of network attacks. The design was shown to have a reasonably low implementation cost while remaining transparent to normal TCP users, and most importantly being able to thwart a broad range of attacking strategies. In particular, the mechanism was found to prevent any attacker from having an impact much greater than that of a regular TCP flow. Obviously, by mustering a very large number of attacking hosts/identities, it is still possible to significantly disrupt service, but the impact of individual attackers has now been significantly curtailed. We are currently working on implementing the design in a Linux router to test its performance in more realistic settings, and are also investigating the feasibility of combining it with other router-based mechanisms such as ACC-Pushback.

## References

[1] The network simulator – ns-2. http://www.isi.edu/nsnam/ns/.

[2] L. V. Ahn, M. Blum, N. J. Hopper, and J. Langford. CAPTCHA: Using hard AI problems for security. In *Proc. of EUROCRYPT'03*, May 2003.

[3] G. Appenzeller, I. Keslassy, and N.McKeown. Sizing router buffers. In *Proc. ACM SIGCOMM'04*, Portland, OR, August 2004.

[4] S. M. Bellovin. ICMP traceback messages. Internet draft, IETF, 2000. http://www1.cs.columbia.edu/ smb/papers/draft-bellovin-itrace-00.txt.

[5] R. Bush and D. Meyer. Some Internet architecture guidelines and philosophy. RFC 3439, IETF, December 2003.

[6] C. Estan and G. Varghese. New directions in traffic measurement and accounting. In *Proc. ACM SIGCOMM'02*, pages 323–336, 2002.

[7] S. Floyd and K. Fall. Promoting the use of end-to-end congestion control in the Internet. *IEEE/ACM Trans. Netw.*, 7(4):458–472, 1999.

[8] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Trans. Netw.*, 1(4):397–413, 1993.

[9] S. Floyd and E. Kohler. Internet research needs better models. In *Proc. HotNets-I*, October 2002.

[10] M. Goyal, R. Guérin, and R. Rajan. Predicting TCP throughput from non-invasive network sampling. In *Proc. IEEE INFOCOM'2002*, pages 180–189, New York, NY, June 2002.

[11] M. Guirguis, A. Bestavros, and I. Matta. Exploiting the transients of adaptation for RoQ attacks on Internet resources. In *Proc. IEEE ICNP'04*, pages 184–195, 2004.

[12] F. Hao, M. Kodialam, T. V. Lakshman, and H. Zhang. Fast, memory efficient traffic estimation by coincidence counting. In *Proc. IEEE INFOCOM'05*, Miami, Fl, March 2005.

[13] J. Ioannidis and S. M. Bellovin. Implementing pushback: Router-based defense against DDoS attacks. In *Proc. NDSS'02*, February 2002.

[14] H. Jiang and C. Dovrolis. Guardian: A router mechanism for extreme overload prevention. In *Proc. ITCOM'02*, August 2002.

[15] S. Kandula, D. Katabi, M. Jacob, and A. Berger. Botz-4-sale: Surviving organized DDoS attacks that mimic flash crowds. In *Proc. of NSDI'05*, May 2005.

[16] A. Kumar, J. Xu, J. Wang, O. Spatschek, and L. Li. Space-code bloom filter for efficient per-flow traffic measurement. In *Proc. of IEEE INFOCOM'04*, Hong Kong, China, March 2004.

[17] A. Kuzmanovic and E. W. Knightly. Low-rate TCP-targeted denial-of-service attacks: the shrew vs. the mice and elephants. In *Proc. ACM SIGCOMM'03*, pages 75–86, 2003.

[18] R. Mahajan, S. M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker. Controlling high bandwidth aggregates in the network. *SIGCOMM Comput. Commun. Rev.*, 32(3):62–73, 2002.

[19] R. Mahajan and S. Floyd. Controlling high-bandwidth flows at the congested router. In *Proc. IEEE ICNP'01*, pages 192–201, 2001.

[20] R. Pan, L. Breslau, B. Prabhakar, and S. Shenker. Approximate fairness through differential dropping. *SIGCOMM Comput. Commun. Rev.*, 33(2):23–39, 2003.

[21] V. Paxson. An analysis of using reflectors for distributed denial-of-service attacks. *SIGCOMM Comput. Commun. Rev.*, 31(3):38–47, 2001.

[22] S. Savage, D. Wetherall, A. Karlin, and T. Anderson. Practical network support for IP traceback. In *Proc. ACM SIGCOMM'00*, pages 295–306, 2000.

[23] Y. Xu and R. Guérin. On the robustness of router-based denial-of-service (DoS) defense systems. *SIGCOMM Comput. Commun. Rev.*, 35(2):47–60, July 2005.

[24] Y. Xu and R. Guérin. A double horizon defense design for robust regulation of malicious traffic. Technical report, University of Pennsylvania, January 2006. Available on-line at: http://einstein.seas.upenn.edu/mnlab/publications.html.

[25] A. Yaar, A. Perrig, and D. Song. SIFF: A stateless internet flow filter to mitigate DDoS flooding attacks. In *Proc. IEEE Security and Privacy Symposim*, May 2004.

[26] X. Yang, D. Wetherall, and T. Anderson. A DoS-limiting network architecture. In *Proc. ACM SIGCOMM'05*, Philadelphia, PA, August 2005.

[27] D. Yau, J. Lui, and F. Liang. Defending against distributed denial-of-service attacks with max-min fair server-centric router throttles. In *Proc. IWQoS'02*, May 2002.

[28] Y. Zhang, N. Duffield, V. Paxson, and S. Shenker. On the constancy of Internet path properties. In *Proc. ACM IMW'01*, 2001.