VNU Journal of Science: Comp. Science & Com. Eng., Vol. 40, No. 1 (2024) 1–17

VNU Journal of Science:
Computer Science and Communication Engineering

Journal homepage: http://www.jcsce.vnu.edu.vn/index.php/jcsce

Original Article

A Threshold Adaptation Mechanism for Detecting and Filtering Low-Rate DDoS Attacks

Minh Viet Kieu, Dai Tho Nguyen*, Thanh Thuy Nguyen

VNU University of Engineering and Technology, 144 Xuan Thuy, Cau Giay, Hanoi, Vietnam

Received 01 December 2023
Revised 28 February 2024; Accepted 19 March 2024

Abstract: TCP-targeted low-rate distributed denial-of-service (LDDoS) attacks pose a serious challenge to the reliability and security of the Internet. Among various proposed solutions, we are particularly interested in the Congestion Participation Rate (CPR) metric and the CPR-based approach. Through a simulation study, we show that if the algorithm makes use of a fixed CPR threshold, it cannot simultaneously preserve high TCP throughput under attacks and achieve good fairness performance for TCP flows in attack-free periods. Then, we propose a method for adaptively changing the threshold over time to obtain both the objectives. Simulation results show that our adaptive CPR-based approach can effectively protect TCP flows under attacks while keeping fairness for the flows when attacks are not present.

Keywords: Low-rate DDoS, TCP protocol, congestion control, active queue management

1. Introduction

Distributed denial-of-service (DDoS) attacks have been identified as a major threat to the current Internet. The attacks usually consist of a few to hundreds of thousands of compromised computers which send a massive number of packets toward a victim. The victim and the gateway in front of the victim's network are forced to handle intense attack traffic as they have no mechanism to differentiate attack packets from normal ones sent by benign computers, which finally leads to a depletion of their resources such as network bandwidth, processor cycles, and memory. As a result, the victim is no longer able to serve the benign computers and legitimate users suffer a degradation or a denial of the victim's service until the attack stops. Traditional DDoS attacks can be very dangerous if they are not properly handled, however their coarse behavior of sending a large volume of packets toward a victim usually makes them easy to be detected and mitigated.

*Corresponding author.
E-mail address: nguyendaitho@vnu.edu.vn

https://doi.org/10.25073/2588-1086/vnucsce.1861

In 2003, Kuzmanovic and Knightly introduced a class of more clever DDoS attacks called TCP-targeted low-rate DDoS (LDDoS) attacks [1]. This kind of DDoS attacks is difficult to detect as they assault a victim with a smaller rate thanks to the exploitation of the TCP's retransmission timeout mechanism. The attack periodically sends high-rate, short-duration bursts of packets, leading to a low average attaking rate and therefore the ability to avoid detection from existing counter-DDoS mechanisms, but at the same time it can force TCP flows to continually timeout with extremely low throughput.

The CPR-based approach [2] is a router support mechanism for detection and filtering LDDoS attacks. The Congestion Participation Rate (CPR) metric is proposed to measure the contribution of a flow to network congestion. LDDoS flows are expected to have higher CPR values than TCP flows as they don't use any form of congestion control and actively induce congestion in a network. Through a series of experiments, the study reveals that the average CPR of normal TCP flows is around 0.2, whereas the average CPR of LDDoS flows is much larger, around 0.8. A CPR threshold is used to determine whether a packet flow is an attack flow or not. If a flow's CPR is greater than the threshold, it is considered as an LDDoS flow and its subsequent packets will be dropped. Otherwise, it is considered as a normal TCP flow and its packets won't be dropped and will be processed as usual. However, like other threshold-based LDDoS attack detection methods [3, 4], the CPR threshold represents the tradeoff between the detection rate and the false positive rate.

In this paper, we conduct a rather different but complementary study to the one in [2] in which we investigate both LDDoS and TCP flows' CPR by varying TCP flows' propagation delay and the number of TCP flows under a specific attack pattern, rather than keeping the number of TCP flows and their propagation delay fixed and varying LDDoS attack parameters as carried out in [2]. Experiments are performed in which TCP flows start transmitting packets at random times. This is intended to simulate the random behavior of downloading files that occurs on the Internet. In each of the first three sets of experiments, TCP flows have the same propagation delay and the delay is varied over the sets. In the last set of experiments, TCP flows are configured to have different propagation delay, which intends to better simulate the diversity of the flows in the network. Next, through more experiments, we prove the existence of another tradeoff caused by the CPR-based approach, the one between optimizing TCP throughput under attacks and providing fairness to TCP flows in attack-free periods. To overcome all of the tradeoffs, we propose a method that adapts the CPR threshold according to whether the network is under attack or not. Performance of the adaptive CPR-based approach is compared to that of the three fixed threshold instances of the approach. The results show that the adaptive CPR-based approach can preserve TCP throughput under attacks fairly well while keeping fairness for TCP flows under no attack condition. Consequently, the adaptive method for the CPR threshold increases the feasibility and the applicability of the CPR-based approach in real networks.

This paper consists of eight sections. The next section, 2, reviews background knowledge. Section 3 presents the investigation of the CPRs of TCP and LDDoS flows while Section 4 analyzes the tradeoffs of the approach. Section 5 describes in detail the adaptive method applied for the CPR threshold. Section 6 presents simulation results. Section 7 presents related work and we conclude the paper in Section 8.

2. Background

2.1. TCP's Timeout Mechanism

Transmission Control Protocol (TCP) has been widely used in the current Internet and its

congestion control mechanism is one of the main factors behind the success of today's Internet. To deal with congestion in the network, TCP uses two schemes as part of its body: AIMD (Additive Increase Multiplicative Decrease) policy and retransmission timeout mechanism. The two schemes operate interwinningly to make TCP flexible and robust in diverse network conditions. When packet loss is rare, the AIMD policy is used, and vice versa when packet loss is dense, TCP makes use of the timeout mechanism.

AIMD policy allows TCP flows to adapt their sending rates to the network conditions (available network bandwidth, current congestion status). To do that, each TCP flow maintains a congestion window, whose size is denoted by *cwnd*. The size of the window dictates the number of packets each flow can send into the network without acknowledgment. When a TCP flow starts, it has a window of one packet and then speeds up the sending rate by doubling the congestion window size every round trip time. This phase is called a slow start. When the window size exceeds a slow start threshold, $ssthresh$, TCP enters a new phase called congestion avoidance. In this phase, *cwnd* is increased at a much slower rate of one packet per round trip time.

The congestion window size of a TCP flow cannot increase forever. When the size is too large, in particular, it reaches the capacity of the path from sender to receiver, at that time any increase in the congestion window size can result in a packet loss. Packets can also get lost due to damage in transit [5].

TCP has two different ways to detect and recover packet losses. One way is through the receipt of three duplicate acknowledgments for a data packet. In this case, a TCP sender sets the $ssthresh$ threshold to one-half of the current *cwnd*, and *cwnd* is then set to $ssthresh$ plus 3, after that, the sender retransmits the seemingly lost packet and enters the fast recovery phase.

The fast recovery phase only ends when either a retransmission timeout occurs or all the packets in the last window have been acknowledged [6]. If a retransmission timeout occurs, TCP enters the slow start phase with *cwnd* set to one packet. If the latter case happens, TCP returns to the congestion avoidance phase with *cwnd* set to $ssthresh$, i.e. one-half of the congestion window size at the time TCP enters the fast recovery phase.

The second way for TCP to detect and recover packet losses is through the timeout mechanism. The mechanism is used when network congestion occurs and a large number of packets are lost, the sender cannot receive enough duplicate ACKs to trigger fast recovery. After a certain time, if the sender has not received an ACK for a data packet sent previously while receiving less than three duplicate ACKs, it gets a timeout, the data packet is assumed to be lost and needs to be retransmitted. At that time, TCP sender sets its new $ssthresh$ value of one-half of the current *cwnd* and reduces the *cwnd* to one packet, the packet is resent with RTO doubled and the sender enters a slow start phase. Upon further timeout, RTO (Retransmission time out) is doubled with each subsequent timeout along with the resending of the lost packet.

Typically we have $RTO = minRTO =$ 1 second (as recommended in [7]). Moreover, when a timeout occurs, and assuming that the current RTO value is 1 second, TCP reduces its *cwnd* to one packet, RTO is doubled to 2 seconds, and the lost packet is resent. After 2 seconds if the packet has not been acknowledged then RTO is set to 4 seconds and so on. RTO values may be limited by an upper bound of at least 60 seconds (as specified in [8]). The sequence of doubling operations, also known as the exponential back-off algorithm, was originally formed for TCP to respond to timeout but has now become the target of LDDoS attacks.

2.2. Modeling LDDoS Attacks

A flow is typically defined by the tuple (source IP, source port, destination IP, destination

port, protocol). Each LDDoS flow has four additional parameters T_a, T_b, R_b and s. T_a is the inter-burst period, T_b is the burst length, R_b is the burst rate and s is the starting time (see Figure 1).

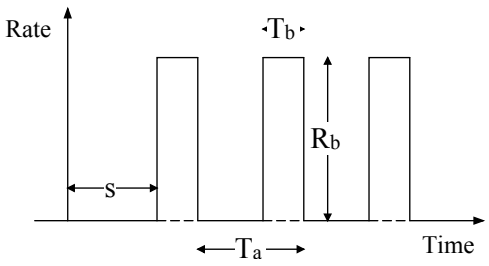


Figure 1. A LDDoS flow.

Assume that there is only one LDDoS flow, the attack flow must have rate R_b large enough to induce loss (R_b should exceed bottleneck bandwidth), duration T_b of scale RTT, and period T_a of scale minRTO. In the most common case, an LDDoS attack consists of many attacking flows, then it can be modeled as in [2], where an LDDoS attack is represented as a 4-tuple (n, g, m, σ) with n is the number of attack flows, g is the number of attack groups, m is the number of flows in each attack group and σ – the starting gap between consecutive attack groups. Attack flows in a group are assumed to start at the same time.

2.3. Calculating CPR of a Flow

According to [2], the CPR of a flow F_i is calculated by a router where the CPR-based approach is deployed as follows:

$$\theta_i = \sum_{t \in T^*} S_{i,t} \Big/ \sum_{t \in T} S_{i,t} \qquad (1)$$

in which $S_{i,t}$ is the number of packets from flow F_i arriving at the router in the interval $[t, t + d]$, d is chosen empirically to be 1 ms corresponding to a sampling frequency of 1000 Hz. T^* is the set of sampling periods when the outgoing link is congested, and T is the set of all sampling periods. The outgoing link is considered to be

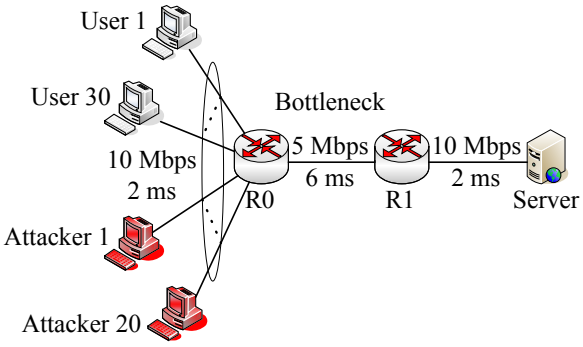


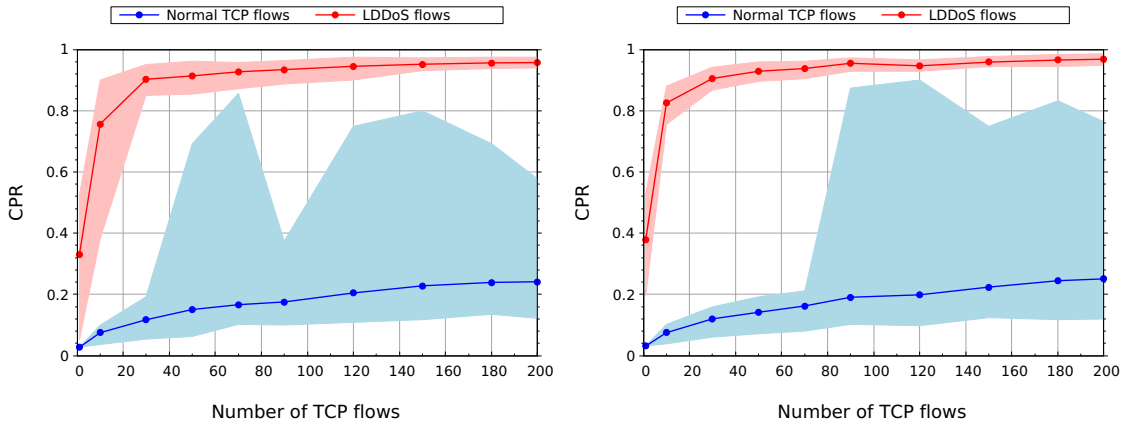
Figure 2. Network topology.

congested in a sampling period if there is at least one packet dropped at the packet queue by the RED algorithm during the period. Moreover, we term such a sampling period as congested.

3. Investigation of TCP and LDDoS Flows' CPRs

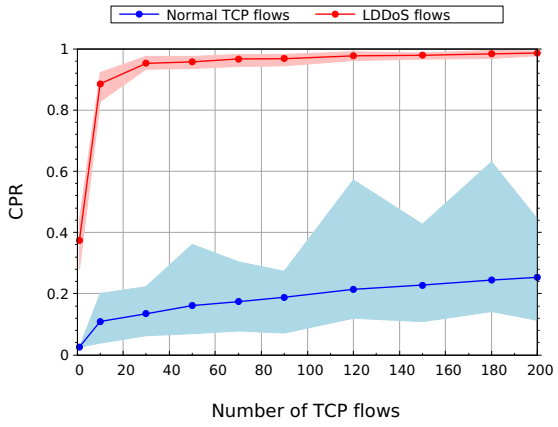
To examine the average, minimum and maximum CPR values of TCP and LDDoS attack flows, in this section we conduct four sets of simulations. We use the platform taken from the address of [9]. For the first set, we use the network in Figure 2, in which the leftmost and rightmost links have a one-way propagation delay of 2 ms. In the second set, the links have a propagation delay of 7 ms while in the third set, they have a propagation delay of 22 ms. In the fourth set, the rightmost link has a one-way propagation delay of 2 ms but the leftmost links have a random propagation delay in the range [2, 92] ms so the two-way propagation delay of the TCP flows is randomized in the range [20, 200] ms.

The leftmost and rightmost links have the same bandwidth of 10 Mbps. The link between router $R0$ and router $R1$ has a smaller bandwidth of 5 Mbps and a one-way propagation delay of 6 ms so that it becomes the congestion point of the network. The queue size of the bottleneck link is 50 packets. CPR-based approach is deployed at router R0 on the queue of the link, whereas other

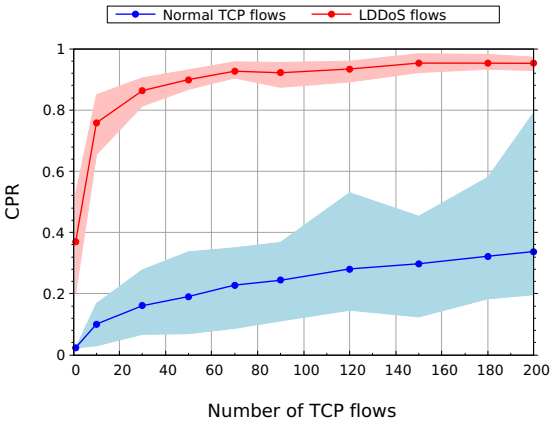


(a) Results of the first set.

(b) Results of the second set.



(c) Results of the third set.



(d) Results of the fourth set.

Figure 4. Results of the four sets of simulations.

links use DropTail queues. The CPR threshold, which we denote as τ in this paper, is set to 2 because we only want to collect flows' CPR and don't want the approach to drop packets as CPR of a flow is always between 0 and 1.

Figure 2 only illustrates the case when there are 30 long-lived TCP flows in the network. However, to see how the number of TCP flows affects the CPRs of TCP and LDDoS flows, we vary the number of TCP flows during each simulation set. In total, each set consists of 10 simulations with the number of TCP flows ranging from 1 to 200.

Each of the TCP flows originates at one of the leftmost computers from User 1 to User 200 (not shown in Figure 2) and terminates at Server, using an FTP application with unlimited data to send. The TCP version is NewReno with a packet size of 1040 bytes. TCP flows randomly start in the period [20, 120] s and end at time 240 s.

We create an LDDoS attack scenario with parameters $n = 20$, $g = 20$, $m = 1$, $\sigma = 1$ s. Each LDDoS flow originates at one of 20 attack computers from Attacker 1 to Attacker 20 and also terminates at Server, sending UDP packets of 50 bytes, and having parameters $T_a = 20$ s, $T_b = 200$ ms and $R_b = 5$ Mbps. The attack starts at time 120 s and stops at time 220 s.

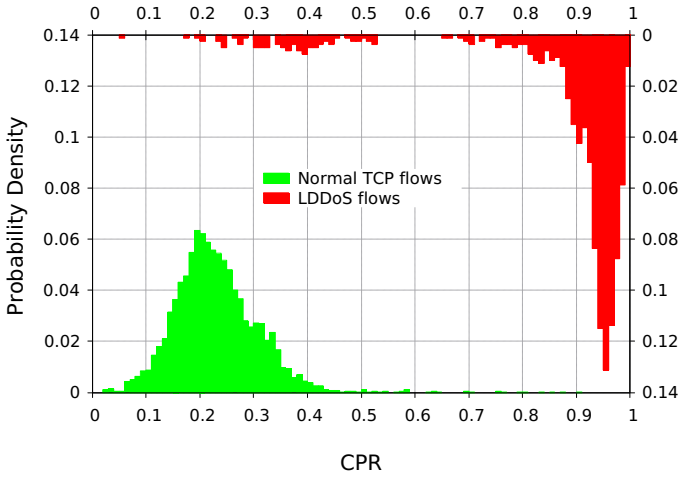


Figure 5. Probability distribution of CPR of normal TCP flows and LDDoS flows.

Information of various flows passing router R0 will be stored in a Bloom filters structure which is similar to one in RRED algorithm [10]. The Bloom filters structure we use in this study has the number of levels $L = 1$, the bins in each level $N = 4200$, and a perfect hash function is used to map each flow to a different bin of the only level. All simulations start at time 0 s and end at time 240 s. At the finish time, we record the average, minimum, and maximum CPRs for all the flows in the network, including normal and attack ones, and the obtained results are demonstrated in Figure 4. The average CPRs of LDDoS flows are depicted by the red lines and the corresponding ones for TCP flows are depicted by the blue lines. We fill the area from the minimum CPR line to the maximum CPR line of LDDoS flows with pink color and for the corresponding range of TCP flows we use the light blue color. One can see that when the number of TCP flows increases, the average, minimum, and maximum CPRs of TCP and LDDoS flows increase accordingly. The average CPR of normal TCP flows passes 0.2 when the number of TCP flows reaches 120 for the first three sets and 70 for the fourth set. The CPR range of TCP flows tends to be wider while that of LDDoS flows becomes thinner as the number of TCP flows increases. The average, minimum, and maximum CPRs of LDDoS flows drop sharply when the number of TCP flows is less than 30 and they are larger than 0.8 for all other cases.

However, Figure 4 cannot fully describe the distribution of CPRs of TCP and LDDoS flows, instead it only shows the CPR range along with the mean CPR of the two types of flows. For the simulations, we have collected a total of 3604 CPR values for TCP flows and 800 CPR values for LDDoS flows and the probability distribution of the CPRs is shown in Figure 5. The CPRs of TCP flows are small and concentrated around 0.2 while the CPRs of LDDoS flows are larger and concentrated mainly in the range [0.8, 1]. Some CPRs of LDDoS flows are scattered in the range [0.2, 0.5]. With the probability distribution, one can choose a CPR threshold τ such that it maximizes the detection rate (i.e., the ratio of the red area on the right side of τ to the whole red area) and minimizes the false positive rate (i.e., the ratio of the green area on the right side of τ to the whole green area). However, the tradeoff between the detection rate and the false positive rate always exists as the figure shows that CPRs of LDDoS flows can be as small as those of TCP flows and vice versa, CPRs of TCP flows can be as large as those of LDDoS flows in

some situations, which is shown in some overlap between the red and green areas.

4. Analysis of the CPR-Based Approach

4.1. Performance of the CPR-Based Approach

To evaluate the performance of the CPR-based approach, we perform 9 simulations with the network in Figure 2, in which TCP flows start simultaneously at time 20 s and stop at time 240 s while LDDoS attacks start at time 120 s and stop at time 220 s. Each simulation uses the approach with a certain value of τ ranging from 0.1 to 0.9.

The approach is configured so that it can only drop packets after time 120 s,[1] which enables TCP flows to share the bottleneck bandwidth freely until the attack starts. This intends to isolate the effect of setting τ only on TCP throughput under attacks and make no effect to TCP flows in the period from 20 s to 120 s. Note that we still calculate CPR for each TCP flow from time 0 although the approach can only drop packets after time 120 s. Figure 6 presents the results of the simulations in which the TCP throughput is normalized with the bottleneck link's bandwidth.

One can observe that the TCP throughput decreases gradually as τ increases from 0.2 to 0.9, as τ equals 0.1 there is a substantial drop in the performance. This is because 0.1 is lower than the average CPR of TCP flows, as has been pointed out in [2] and in the previous section. This in turn results in an increased number of TCP packets incorrectly dropped by the approach.

4.2. Impact of the Approach on Fairness of New TCP Flows

As LDDoS attacks do not always happen, every queue management algorithm and in particular the CPR-based approach, should provide fairness to TCP flows under no attack conditions. If not, a particular user could experience a long period waiting for a data transfer to end. This subsection will explore the fairness amongst new TCP flows when there is no attack. We conduct another set of simulations with the network in Figure 2, in which the CPR-based approach is deployed at router R0 with a certain value of τ from 0.1 to 0.9. As in the previous subsection, we only assign each of these values in turn to the threshold at time 120 s. Each value of τ is associated with four TCP packet sizes, which produces 36 simulations in total. 4 other simulations that corresponds to $\tau = 1$ will be described later in subsection 6.2. In each simulation, there are 10 TCP flows, all start at time 20 and stop at time 240. At time 120 ten new TCP flows start, one every 0.1 seconds, all stop at time 220. The goodput[2] of each new TCP flow is recorded. The standard deviation of the goodputs is calculated with the expected mean value of 1/20-th of the bottleneck link's bandwidth multiplied by 100 (equal to 25 Mbit). The results are shown in Figure 7, in which each data point depicted by square, circle, diamond, or triangle shows the result from a single simulation, with the x-axis showing the fixed value of τ

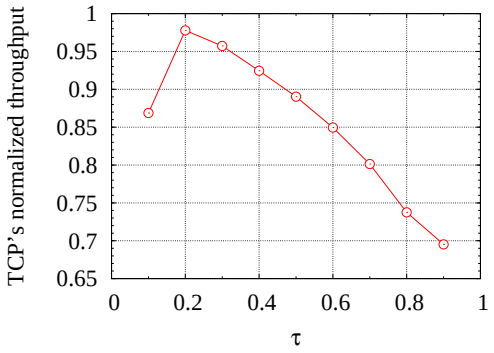


Figure 6. TCP's normalized throughput under LDDoS attacks.

[1]The CPR threshold is set to 2 before time 120 s. At time 120 s, the threshold is re-assigned to one of the nine values from 0.1 to 0.9. The basis of the assignment is that the CPR of a flow is always less than or equal to 1.

[2]The goodput of a flow is the bandwidth delivered to its destination, excluding duplicate packets.

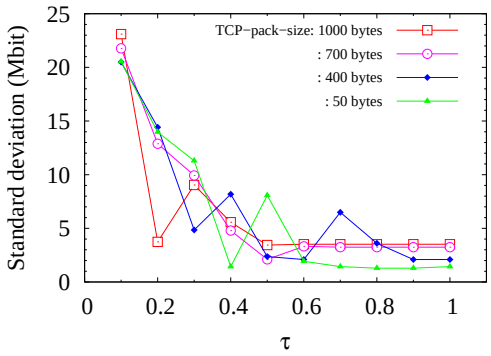


Figure 7. The standard deviation of 10 new TCP flows.

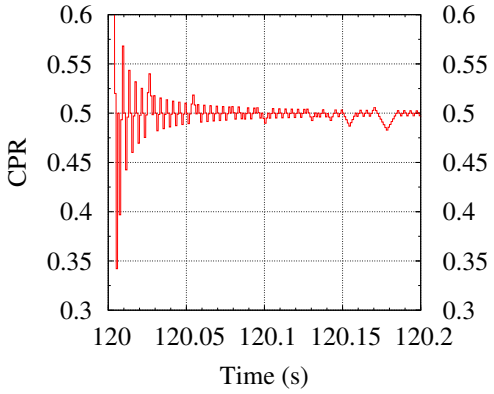


Figure 8. The CPR of the first attack flow.

used by the approach, and the y-axis showing the standard deviation. Each line shows the results of simulations with a specific TCP packet size ranging from 50 bytes to 1000 bytes (not including TCP packets' header size of 40 bytes).

The previous and current subsections clearly show the tradeoff between maximizing TCP throughput during attacks and ensuring fairness for new TCP flows in attack-free periods when the CPR-based approach is in use. If network operators want to use the approach with τ fixed, they must be chosen between high TCP throughput under attacks (requiring small values of τ), or fairness to new TCP flows (requiring large values of τ). Our adaptive method is proposed to solve the problem. Before going into the details of the method, in the next two subsections, we present two groundworks on which the study depends directly.

4.3. Convergence of LDDoS Flows' CPRs

Consider the simulation in subsection 4.1, in which the CPR-based approach is configured with $\tau = 0.5$ and there is such an LDDoS attack. Figure 8 shows the CPR of the first attack flow during the 200 ms period from time 120 to time 120.2 (exactly over its first burst of packets arriving at the router). In the figure, each horizontal line presents a period of 1 ms in which the flow's packets arrive with the same CPR value. Such a period can contain exactly 12.5 attack packet arrivals on average, meaning that every time a period contains 12 packet arrivals, the next period must contain 13 packet arrivals. Horizontal lines with CPR greater than or equal to 0.5 mean that every packet of the flow arriving in the periods will be dropped by the approach, otherwise they will be forwarded to the RED block. We have omitted the first and second periods when the flow starts, which have CPR equal to 0 and 1 respectively. Vertical lines connecting a high horizontal line with a lower one indicate that the previous period is not congested, and vertical lines connecting a low horizontal line with a higher one depict the situation that the previous period is congested. As the figure shows, the CPR roughly converges to the τ's value, in this case, 0.5, from both sides, below and above. All subsequent bursts of the flow happening at time 140, 160, 180, and 200 (not shown in the figure) have CPR that still oscillates around 0.5, but in narrower ranges. CPR of all other attack flows also converges to the value in a similar way.

4.4. RED with Congestion Sample Rate

Recall that a sampling period is considered to be congested if during the period there is at least one packet dropped by the RED algorithm. We define congestion sample rate (CSR) as the

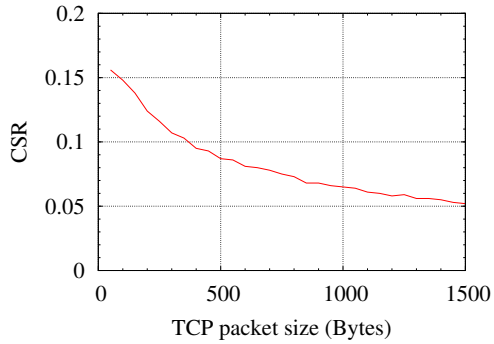


Figure 9. The CSR of the bottleneck link in attack-free periods.

number of congested periods divided by the total number of sampling periods. The CSR is, in fact, the ratio of $|T^*|$ and $|T|$:

$$CSR = |T^*| / |T| \quad (2)$$

where $|T^*|$ and $|T|$ are notations for the number of elements in the set T^* and T respectively. The two sets T and T^* have been defined in equation (1). The difference between CSR and CPR is that CSR is global while CPR is flow-specific. Here, we repeat the simulations in subsection 4.1 without LDDoS attacks, and at the bottleneck link's queue, instead of using the CPR-based approach, we use pure RED only. In each simulation, all TCP flows use the same packet size varying from 50 bytes to 1500 bytes, and at time 240 the CSR of the link is recorded. This is to survey the CSR of the bottleneck link in attack-free conditions. Figure 9 shows the obtained results.

We can see that the CSR of the link is smaller than 0.2 in all cases, it gradually decreases from slightly above 0.15 at a TCP packet size of 50 bytes to about 0.05 when the TCP packet size reaches 1500 bytes.

5. Our Proposed Method

Section 3 shows that it is necessary to have a high CPR threshold to facilitate packets from TCP flows getting through the bottleneck link. However, it can let LDDoS attacks to be more effective as the performance of the CPR-based approach deteriorates when τ increases as demonstrated in subsection 4.1. Furthermore, there is a tradeoff between maximizing TCP throughput under attacks and enforcing fairness for new TCP flows in attack-free periods if the CPR-based approach is used. Therefore, we are clearly aware that we need to use the CPR threshold in a more sophisticated way to balance the aspects. This section will present our method of adjusting τ over time proposed to overcome the tradeoff, as given in Algorithm 1. In the method, τ is restricted in the range [0.2, 0.8] because the average CPR of normal TCP flows is slightly smaller than 0.2 and the average CPR of LDDoS flows is slightly greater than 0.8 [2]. Once an attack burst is launched, the bottleneck link is more likely to be congested, and thus τ is reduced by the adaptive method, in steps of α. To react quickly to the attack, τ should be

Algorithm 1 – Our proposed method.

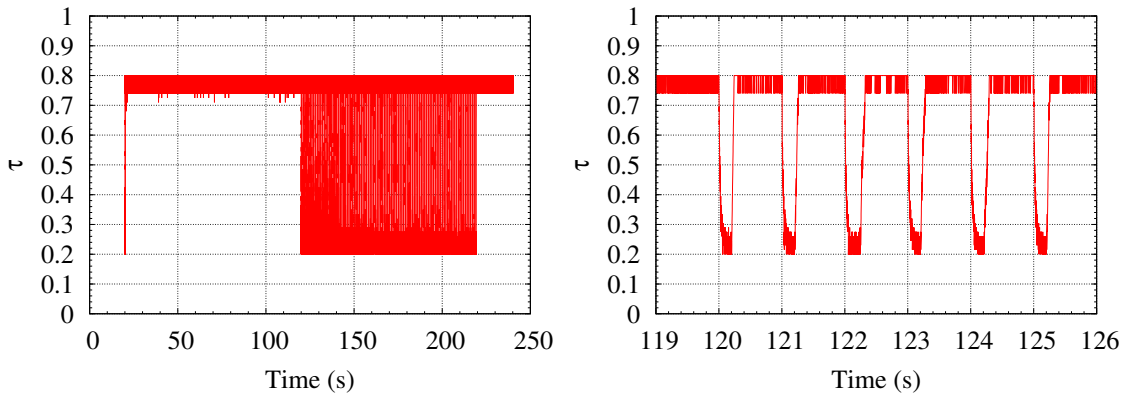
```
Initialization:
  τ = 0.8;
Every sampling period (miliseconds):
  if the outgoing link is congested then
    τ = τ − α;
    if (τ < 0.2) then
      τ = 0.2;
    end if
  else
    τ = τ + β;
    if (τ > 0.8) then
      τ = 0.8;
    end if
  end if

Fixed parameters:
  sampling period: time; 1 (milisecond)
  α: decrement; 0.06
  β: increase factor; 0.015
```

reduced to 0.2, or close to 0.2, over some time smaller than a typical attack burst length, so we use α set to 0.06, meaning that τ can be decreased from 0.8 to 0.2 just after 10 sampling periods, corresponding to 10 ms. The setting of β to 0.015, one-fourth of α, can be justified as follows. As the convergence of CPRs of LDDoS flows to the τ's value described in subsection 4.3, and τ cannot be reduced to a value smaller than 0.2, the CPR of each LDDoS flow tends to converge to 0.2. Each LDDoS flow has a constant attack rate in the attack burst periods as well as its CPR near 0.2, resulting in the CSR of the link about 0.2 over the attack burst periods. By setting β to one-fourth of α, we enforce τ to roughly deviate no more than 0.06 from 0.2 once τ reaches 0.2 during the attack burst periods. When an attack burst period elapses, the network returns to normal, and τ is expected to increase to 0.8. This can happen as the CSR of the link in normal operation without attacks is less than 0.2. The ability of τ to increase to 0.8 and oscillate close to the peak value when an attack burst subsides does not only depend on the CSR but also on the distribution of congested periods over time. If the congested periods are distributed equally, τ would be constantly swung in the range [0.74, 0.8] once it reaches 0.8. For example, during attack-free periods, there are two congested periods every ten consecutive sampling periods on average, and assume that the first one is located at the first position and the second one is located at the sixth position in the chain based on the assumption that congested periods are uniformly distributed over time. After the first sampling period, the CPR threshold is reduced by α because the period is congested. After four following non-congested periods, the threshold increases by $4 \times \beta = \alpha$. The same thing happens for the rest of the chain. If τ has reached 0.8 before, it will continuously fluctuate in the range [0.74, 0.8] by our adaptive threshold algorithm.

6. Experimental Evaluation

6.1. Threshold Adaptation

To examine the threshold adaptation method presented in Section 5, we re-run the simulation in subsection 4.1 with a change of using the method from time 0 instead of using τ fixed. The adaptation of τ over the entire simulation time and over a short period from time 119 to time 126 are given in Figure 10. It shows that when there is no attack, τ mainly fluctuates in the range [0.74, 0.8], except the time when 30 normal TCP flows start (time 20). At that time τ is decreased to 0.2. This means that on average, there is one congested period, followed by four or more non-congested periods. When the attack begins at time 120, τ is



(a) Adaptation over the simulation time.

(b) Adaptation over [119, 126].

Figure 10. The adaptation of τ.

Table 1. Parameters of LDDoS attacks.

Categories	LDDoS attack				Single flow			Aggregate flow		
	n	**g**	**m**	σ	T_a **(s)**	T_b **(ms)**	R_b **(Mbps)**	T_a^+ **(s)**	T_b^+ **(ms)**	R_b^+ **(Mbps)**
AFI	20	20	1	$T_a/20$	[4, 40]	200	5	[0.2, 2]	200	5
AWI	20	20	1	T_b	1	[0, 50]	5	1	[0, 1000]	5
ARI	20	1	20	0	1	200	[0, 0.5]	1	200	[0, 10]

quickly decreased to 0.2 and then fluctuates close to 0.2, mainly in the range [0.2, 0.26], over the first attack burst period. When the attack burst ends, τ increases and again fluctuates near 0.8 as usual. This behavior repeats until time 220 when the attack finishes.

6.2. *Performance of the Adaptive CPR-Based Approach*

To evaluate the throughput performance of the CPR-based approach using the adaptation method, in this subsection, we perform three sets of simulations as in the paper [2], which are called Attack Frequency Intensification (AFI), Attack burst Width Intensification (AWI), and Attack burst Rate Intensification (ARI). The difference is that we use broader ranges of values for parameters T_a, T_b, and R_b as shown in Table 1. T_a is varied in the range [4, 40] (s) instead of the range [20, 40] (s), T_b is varied in the range [0, 50] (ms), not the range [0.1, 10] (ms), and R_b is varied in the range [0, 0.5] (Mbps) rather than the range [0.01, 0.25] (Mbps). This aims to examine the robustness of the threshold adaptation method not only against low-rate DDoS attacks but also against general DDoS attacks. In each simulation, the setting for TCP flows is the same as in subsection 4.1, the attack also starts at time 120 and stops at time 220. The threshold adaptation method is turned on at time 0. For comparison, we use three instances of the original and thus non-adaptive CPR-based approach with $\tau = 0.2$, $\tau = 0.6$ and $\tau = 0.8$. For the instances, τ is set to 2 at time 0, and to one of the three values from time 120 onward. We don't assign the three values for τ from time 0 because this may affect fairness between TCP flows during normal operation, especially with small values like 0.2. Simulation results are shown in Figure 11 and we observe that the adaptive CPR-based approach

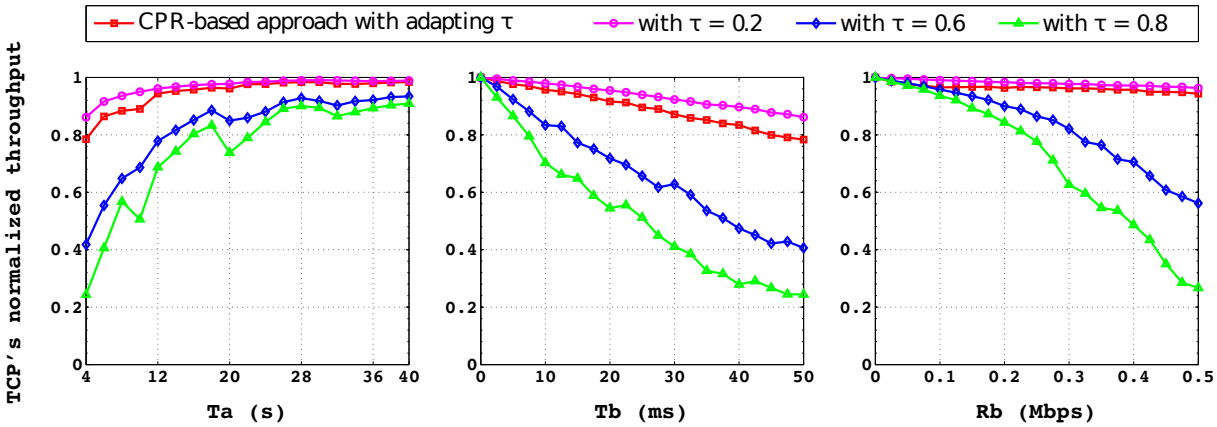


Figure 11. TCP's normalized throughput under attacks.

(depicted by red lines) is slightly worse than the non-adaptive CPR-based approach with $\tau = 0.2$ (depicted by magenta lines), whereas the other non-adaptive variants with $\tau = 0.6$ and $\tau = 0.8$ (depicted by blue and green lines respectively) are considerably worse. The performances of all the variants drain at different rates when the attacks become more aggressive (T_a decreases, T_b increases, R_b increases), with a slower rate for the non-adaptive variants with a smaller τ and for the adaptive variant. In two extreme cases of $T_a = 4$ s and $T_b = 50$ ms when a full-time DDoS happens, the adaptive variant retains about 80% of the bottleneck bandwidth for normal TCP traffic, and the remaining 20% of the bandwidth is wasted for attack traffic, which means that the attack traffic is reduced to 1/5-th of its original size. Non-adaptive variants with $\tau = 0.2$, $\tau = 0.6$ and $\tau = 0.8$ retain about 85%, 40%, and 25% of the bottleneck bandwidth for TCP traffic, respectively. With the adaptive variant, there are usually about 20% of sampling periods during an attack burst in which the attack traffic still passes through the bottleneck link. This is an inherent drawback of the original CPR-based approach as well as the adaptive variant. Fixing the drawbacks will be part of our future work.

Let us come back to Figure 7 for the simulation results about the fairness of our proposed adaptive CPR-based method. The data points corresponding to $\tau = 1$ do not show results from simulations with that value of τ, but from simulations in which the approach uses our method to adjust τ (presented in Section 5). We can observe that the standard deviation varies both with τ and with TCP packet size, with a smaller standard deviation for those simulations with larger τ and smaller packet size. The figure also confirms that our method keeps the standard deviation low regardless of TCP packet sizes, just like as $\tau = 0.6$, 0.8, or 0.9. This means that by using the adaptive method, the goodputs of the 10 new TCP flows tend to be close to the fair bandwidth shares, thus providing fairness to the flows. In contrast, using τ fixed results in high and unpredictable standard deviation (except the three values of τ above), meaning that the goodputs spread out over a wider range. The goodputs and the standard deviations of the TCP flows corresponding to various values of τ are fully shown in Table 2 on the next page.

7. Related Work

In this section, we will discuss and analyze some existing mechanisms that counter LDDoS attacks [11]. In recent years, there has been a lot of research aimed at detecting LDDoS attacks [12–18] but very little research on countering the attacks. Existing counter-LDDoS mechanisms can be divided into two categories: end-point and router-assisted.

Since LDDoS attacks aim at the fixed minRTO and the dependence of the current RTO on minRTO in timeout state (RTOs are multiple of the minRTO), there are two apparent ways to deal with the attacks. First, we should use a random instead of a fixed value for the minRTO parameter as suggested in [19]. In the paper, the authors built a model of TCP throughput under attacks in which end-points randomize their minRTO in a range to eliminate the constancy of TCP's null frequency to the attacks. In general, the randomization of minRTO can only mitigate the attack effect and it cannot eliminate the root cause of the TCP's vulnerability to the attacks. This model can be applied to real networks as some operating systems (OSs), especially the old ones, still use coarse-grained clock granularity (or clock tick) of 500 ms to check timeout events. TCP flows coming from these old systems recover randomly in the range $[1, 1.5]$ sec after a timeout. Consequently, using the large clock granularity has some counter-effects against the attacks.

Another way to deal with the attacks is to choose RTO independently of minRTO. In doing so, it becomes more difficult for the

Table 2. Goodputs and the standard deviation – SD (in Mbit) of 10 new TCP flows vs. τ

TCP packet size	Flow	τ									
		0.1	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**Adapting τ**
50 bytes	1st flow	0	20.28	0	24.98	23.30	25.37	24.59	26.71	26.71	24.59
	2nd flow	0.01	0.01	27.96	26.02	25.42	26.26	25.21	24.35	24.35	25.21
	3rd flow	0.03	27.03	26.98	23.94	25.90	24.14	22.06	24.44	24.44	22.06
	4th flow	0.02	28.66	21.99	23.00	0	24.70	23.87	23.52	23.52	23.87
	5th flow	36.76	28.89	26.08	25.83	27.08	26.14	23.55	27.37	27.37	23.55
	6th flow	0	27.55	27.19	26.04	27.09	21.38	24.01	24.29	24.29	24.01
	7th flow	33.28	26.52	25.46	22.82	25.78	24.56	24.39	24.53	24.53	24.39
	8th flow	0.10	28.76	0.01	23.93	26.88	27.35	23.17	24.21	24.21	23.17
	9th flow	36.02	0	24.91	22.49	24.60	25.43	24.53	23.22	23.22	24.53
	10th flow	36.96	0.01	25.71	25.63	21.88	21.22	23.17	24.14	24.14	23.17
SD		**20.53**	**13.97**	**11.30**	**1.43**	**8.07**	**1.93**	**1.43**	**1.29**	**1.29**	**1.43**
400 bytes	1st flow	37.01	26.32	28.09	27.32	22.90	26.71	31.18	27.72	23.01	26.71
	2nd flow	0.02	0.02	25.08	28.03	26.50	24.38	23.37	26.68	25.53	24.38
	3rd flow	36.64	30.01	27.61	26.15	26.14	21.95	22.28	23.86	25.18	21.95
	4th flow	0.15	28.76	28.68	29.53	30.63	22.93	23.79	27	24.44	22.93
	5th flow	32.93	26.15	24.42	26.83	23.83	27.46	28.88	22.88	22.81	27.46
	6th flow	0	0.02	25.38	0.01	25.97	29.15	31.17	25.26	25.08	29.15
	7th flow	0.01	0	30.20	24.88	25.75	24.39	26.47	26.16	27.73	24.39
	8th flow	0.01	34.66	21.99	22.69	24.84	24.62	22.93	16.81	25.99	24.62
	9th flow	0	21.13	12.18	25.22	21.84	24.24	24.61	25.86	21.45	24.24
	10th flow	35.86	32.52	27.07	24.99	23.11	26.52	7.36	18.60	28.63	26.52
SD		**20.50**	**14.42**	**4.85**	**8.18**	**2.37**	**2.09**	**6.49**	**3.61**	**2.09**	**2.09**
700 bytes	1st flow	0.01	30.90	16.83	22.76	24.76	31.74	30.66	30.66	30.66	30.66
	2nd flow	0.04	25.64	30.29	24.98	20.70	25.13	21.21	21.21	21.21	21.21
	3rd flow	0	0	22.42	28.43	25.60	20.12	20.40	20.40	20.40	20.40
	4th flow	0	28.83	30.35	28.47	24.94	27.73	27.04	27.04	27.04	27.04
	5th flow	34.79	25.58	29.32	26.05	23.58	25.14	24.27	24.27	24.27	24.27
	6th flow	38.69	23.73	0.01	27.05	27.05	22.12	24.75	24.75	24.75	24.75
	7th flow	0.07	30.83	28.91	25.24	26.80	24.62	23.40	23.40	23.40	23.40
	8th flow	35.31	32.54	15.24	11.75	21.05	21.48	21.05	21.05	21.05	21.05
	9th flow	0.25	8.70	29.08	21.12	24.22	28.54	24.02	24.02	24.02	24.02
	10th flow	0.13	0	15.76	27.08	25.38	24.84	21.31	21.31	21.31	21.31
SD		**21.77**	**12.88**	**9.93**	**4.79**	**2.11**	**3.32**	**3.24**	**3.24**	**3.24**	**3.24**
1000 bytes	1st flow	0.31	29.19	25.58	24.31	25.81	22.65	22.65	22.65	22.65	22.65
	2nd flow	0.26	26.02	19.62	26.44	28.37	30.75	30.75	30.75	30.75	30.75
	3rd flow	40.76	25.09	27.11	8	18.62	27.58	27.58	27.58	27.58	27.58
	4th flow	0.37	25.78	20.67	25.69	25.19	28.24	28.24	28.24	28.24	28.24
	5th flow	0.10	25.64	32.41	26.32	24.70	18.22	18.22	18.22	18.22	18.22
	6th flow	0	19.83	17.31	25.99	19.35	25.72	25.72	25.72	25.72	25.72
	7th flow	0.01	20.73	0.02	25.38	29.99	29.35	29.35	29.35	29.35	29.35
	8th flow	0.07	25.75	23.57	28.45	25.18	24.02	24.02	24.02	24.02	24.02
	9th flow	0	23.14	29.83	23.35	22.25	23.72	23.72	23.72	23.72	23.72
	10th flow	36.94	33.43	23.77	25.94	23.92	24.98	24.98	24.98	24.98	24.98
SD		**23.10**	**3.74**	**9.04**	**5.57**	**3.44**	**3.52**	**3.52**	**3.52**	**3.52**	**3.52**

attackers to predict the exact moment when TCP flows retransmit packets after timeout, and hence the attackers cannot synchronize their malicious traffic with RTO of TCP flows to maximize their attack effect. The TCP flows are partially released as their packets can pass the bottleneck link when the queue of the link is not assaulted by attack bursts. Finally, they can achieve a considerable fraction of the bottleneck bandwidth as compared to the very small throughput obtained in the case RTOs are multiple of the minRTO. It is the core idea of a typical end-point mechanism called RTO randomization proposed by Yang et al. in [20] and is re-examined in a real system (Linux) by Efstathopoulos [21]. The randomization of RTO can effectively mitigate the attack effect while keeping both fairness for randomized TCP flows and friendliness for randomized and non-randomized ones.

Randomization on either minRTO or RTO can help TCP flows to avoid LDDoS attacks, but the effect comes too late as TCP flows have entered the timeout state after only one attack burst, which causes an inevitable decline in the TCP throughput. It has been proven in theory through mathematical models and clearly illustrated through simulation and practical results. The end-point mechanisms are fundamentally passive, implying the need for active defense. That's the rationale for various router-assisted mechanisms including active queue management (AQM) algorithms. Since 1998, in RFC 2309 [22] the Internet Engineering Task Force (IETF) has suggested the deployment of the algorithms in network routers to replace the conventional drop-tail algorithm and that Random Early Detection (RED) [23] should be used by default for routers in the Internet.

Using drop-tail discipline for packet queue management in routers can result in some unwanted phenomena, such as: (1) lock-out problem in which a single or a group of flows can monopolize queue space of a common link and prevent other flows from sharing the queue and utilizing the link's bandwidth [24], (2) packet queues in network routers are chronically full, no matter how much the buffer size is, and (3) the global synchronization of TCP flows halving their window size at the same time when a common packet queue is full, followed by a sustained period of lowered link utilization that may cause a reduction in overall network throughput.

If a router deploys RED for its links, TCP flows traversing the links can avoid the global synchronization, data packets often get smaller RTTs because the average queue size of the links is reduced, and the lock-out problem can be mitigated as there will almost always be a buffer available for any incoming packet. However, as shown in [25, 26], it is difficult to set RED's parameters to have it work well in all circumstances. Sometimes RED gateways can deteriorate and behave like a traditional drop-tail gateway when the RED's parameters do not match the requirements of the network load. Therefore, as of 2015 in RFC 7567 [27], IETF no longer recommends RED or any other specific algorithm, such as Flow Random Early Detection (FRED) [28], CHOKe [29], RED with Preferential Dropping (RED-PD) [30], Stabilized Random Early Drop (SRED) [31], or Stochastic Fair Blue (SFB) [32], to be used by default although the RFC continues to recommend the deployment of AQM algorithms in the network.

The design goal of RED is to accompany and complement an end-to-end congestion control protocol like TCP. RED is better than the drop-tail algorithm in cooperating with TCP flows. The algorithm will have the best performance when it is deployed in an environment dominated by responsive flows, i.e., flows that throttle back their transmission rate upon receipt of congestion notification from the network. Historically, RED is designed in the absence of considering its performance under DDoS attacks. Such an attack can involve hundreds of thousands of unresponsive flows, i.e., those that do not use any

form of end-to-end congestion control and do not slow down in response to congestion notification. Indeed, the study in [10] demonstrated that TCP throughput across a bottleneck link still drops sharply when LDDoS attacks happen even though RED is used at routers.

LDDoS attacks create a rather different lock-out phenomenon against TCP traffic where TCP flows are prevented from using a link's bandwidth while the attack traffic takes up only a small fraction of the bandwidth rather than monopolizing it. In search of an AQM algorithm that can detect and regulate the attack traffic, the authors of [19] investigated two representatives: RED-PD and CHOKe. RED-PD uses the RED packet drop history at a router to detect high-bandwidth flows. If a flow has a large number of recently dropped packets, it is likely to have a high arrival rate [33]. Flows above a configurable target bandwidth are identified as high-bandwidth ones. The target bandwidth is determined by the TCP throughput model in [34]. Only high-bandwidth flows are memorized and monitored by the router. RED-PD then restricts the bandwidth allocated to a monitored flow by dropping its packets with a probability depending on the excess sending rate of the flow. However, as RED-PD operates on large time scales of congestion epoch lengths of a reference TCP flow while LDDoS attacks can operate on much smaller time scales, and the response of RED-PD to the attacks is not strong enough, it cannot detect nor throttle the attack traffic. Simulation results showed that RED-PD can only detect and mitigate unnecessarily high-rate attacks with an inter-burst period of less than 0.5 sec. Also, it can only detect LDDoS attack flows with long attack burst lengths (e.g., 300 ms) or high burst rates (e.g., more than twice the bottleneck bandwidth) while much shorter attack burst lengths or much lower burst rates are sufficient to throttle the entire aggregates of the TCP traffic. CHOKe has been shown to have comparable performance as RED-PD against LDDoS attacks.

RED needs additional mechanisms to better protect TCP flows when the attacks are present. In this direction, Zhang et al. proposed the Robust RED (RRED) algorithm [10]. The main idea of RRED is to detect and filter out attack packets before they are fed into the RED algorithm for further processing. RRED has two main function blocks: detection and filtering block and pure RED block. An incoming packet from a flow f is suspected to be an attacking packet if it arrives within a short time interval after a packet from the same flow that is dropped by the detection and filtering block or after a packet from any flow that is dropped by the RED block. An indicator $f.I$ is used to determine whether flow f is an LDDoS attack flow or a normal TCP flow. If a packet from flow f is considered to be an attacking packet, $f.I$ is decreased by one; otherwise, the packet is considered a normal TCP packet, and $f.I$ is increased by one. When a packet arrives at a router, the $f.I$ indicator of the associated flow f is updated. After that if $f.I$ is negative, the packet is dropped; otherwise, the packet will be forwarded to the RED block. The algorithm uses Bloom filters data structure [35] to store statistical information of various flows passing the router. Simulation results showed that Robust RED with a carefully chosen suspicion time interval can effectively filter out attack packets and preserve TCP throughput.

Zhang and his fellows in [2] also proposed a novel metric called Congestion Participation Rate (CPR) to identify LDDoS flows based on their intention to congest the network. Normal TCP flows actively avoid network congestion as they use the TCP congestion control mechanism and hence tend to send fewer packets during network congestion. Whereas, LDDoS flows deliberately induce network congestion and do not slow down their rate when the network congestion occurs. The authors also proposed and implemented the CPR-based approach to detect and filter LDDoS attacks. Simulation results and test-bed experiments showed that

the CPR-based approach can effectively identify LDDoS flows while the normalized cumulative amplitude spectrum (NCAS) approach [3], a Discrete Fourier Transform (DFT)-based method, is only effective for a small set of the attacks.

The study in [4] proposed and evaluated two new information metrics to differentiate LDDoS attack flows from normal TCP flows called Fourier Power Spectrum Entropy and Wavelet Power Spectrum Entropy, and then proposed an idea of integration of the metrics into the RRED algorithm. The simulation results proved that the combination can reduce RRED's false positive rate and improve TCP throughput under attacks.

8. Conclusion

In this paper, we have shown that when using the CPR-based approach to counter LDDoS attacks, there is a tradeoff between maximizing TCP throughput under attacks and providing fairness to TCP flows in attack-free periods. We have proposed an adaptive method for the CPR threshold and the simulation results confirm that our proposition helps the approach to simultaneously achieve high TCP throughput under attacks and fairness for TCP flows when attacks are not present, thereby increasing the approach's feasibility and applicability in real networks.

References

[1] A. Kuzmanovic, E. Knightly, Low-Rate TCP-Targeted Denial of Service Attacks (The Shrew vs. the Mice and Elephants), Proceedings of ACM SIGCOMM, 2003, pp. 75–86, https://doi.org/10.1145/863955.863966.

[2] C. Zhang, Z. Cai, W. Chen, et al., Flow Level Detection and Filtering of Low-Rate DDoS, Computer Networks, Vol. 56, No. 15, 2012, pp. 3417–3431, https://doi.org/10.1016/j.comnet.2012.07.003.

[3] Y. Chen, K. Hwang, Collaborative Detection and Filtering of Shrew DDoS Attacks Using Spectral Analysis, Journal of Parallel and Distributed Computing, Vol. 66, No. 9, 2006, pp. 1137–1151, https://doi.org/10.1016/j.jpdc.2006.04.007.

[4] Z. Chen, C. Yeo, B. Lee, C. Lau, Power Spectrum Entropy Based Detection and Mitigation of Low-Rate DoS Attacks, Computer Networks, Vol. 136, 2018, pp. 80–94, https://doi.org/10.1016/j.comnet.2018.02.029.

[5] V. Jacobson, M. Karels, Congestion Avoidance and Control, ACM Computer Communication Review, Vol. 18, No. 4, 1988, pp. 314–329, https://doi.org/10.1145/52324.52356.

[6] T. Henderson, S. Floyd, A. Gurtov, Y. Nishida, The NewReno Modification to TCP's Fast Recovery Algorithm, RFC 6582, 2012, https://doi.org/10.17487/RFC6582.

[7] V. Paxson, M. Allman, On Estimating End-to-End Network Path Properties, ACM Computer Communication Review, Vol. 29, No. 4, 1999, pp. 263–274, https://doi.org/10.1145/316194.316230.

[8] V. Paxson, M. Allman, J. Chu, M. Sargent, Computing TCP's Retransmission Timer, RFC 6298, 2011, https://doi.org/10.17487/RFC6298.

[9] AQM&DoS Simulation Platform, `https://github.com/mleoking/LeoDoS`, 2016 (accessed on March 1st, 2024).

[10] C. Zhang, J. Yin, Z. Cai, W. Chen, RRED: Robust RED Algorithm to Counter Low-Rate Denial-of-Service Attacks, IEEE Communications Letters, Vol. 14, No. 5, 2010, pp. 489–491, https://doi.org/10.1109/LCOMM.2010.05.091407.

[11] Z. Wu, W. Li, L. Liu, M. Yue, Low-Rate DoS Attacks, Detection, Defense, and Challenges: A Survey, IEEE Access, Vol. 8, No. 19442570, 2020, pp. 43920–43943, https://doi.org/10.1109/ACCESS.2020.2976609.

[12] M. Yue, Z. Wu, J. Wang, Detecting LDoS Attack Bursts Based on Queue Distribution, IET Information Security, Vol. 13, No. 3, 2019, pp. 285–292, https://doi.org/10.1049/iet-ifs.2018.5097.

[13] S. Zhan, D. Tang, J. Man, R. Dai, X. Wang, Low-Rate DoS Attacks Detection Based on MAF-ADM, Sensors, Vol. 20, No. 1, 2019, pp. 189, https://doi.org/10.3390/s20010189.

[14] D. Tang, R. Dai, L. Tang, X. Li, Low-Rate DoS Attack Detection Based on Two-Step Cluster Analysis and UTR Analysis, Human-Centric Computing and Information Sciences, Vol. 10, No. 1, 2020, pp. 6, https://doi.org/10.1186/s13673-020-0210-9.

[15] D. Tang, L. Tang, R. Dai, J. Chen, X. Li, J. Rodrigues, MF-Adaboost: LDoS Attack Detection Based on Multi-Features and Improved Adaboost, Future Generation Computer Systems, Vol. 106, No. 7, 2020, pp. 347–359, https://doi.org/10.1016/j.future.2019.12.034.

[16] D. Tang, J. Man, L. Tang, Y. Feng, Q. Yang, WEDMS: An Advanced Mean Shift Clustering Algorithm for LDoS Attacks Detection, Ad Hoc Networks, Vol. 102, 2020, pp. 102145, https://doi.org/10.1016/j.adhoc.2020.102145.

[17] D. Tang, Y. Feng, S. Zhang, Z. Qin, FR-RED: Fractal Residual Based Real-Time Detection of the LDoS Attack, IEEE Transactions on Reliability, Vol. 70, No. 3, 2021, pp. 1143–1157, https://doi.org/10.1109/TR.2020.3023257.

[18] D. Tang, X. Wang, X. Li, P. Vijayakumar, N. Kumar, AKN-FGD: Adaptive Kohonen Network Based Fine-Grained Detection of LDoS Attacks, IEEE Transactions on Dependable and Secure Computing, Vol. 20, No. 1, 2023, pp. 273–287, https://doi.org/10.1109/TDSC.2021.3131531.

[19] A. Kuzmanovic, E. Knightly, Low-Rate TCP-Targeted Denial of Service Attacks and Counter Strategies, IEEE/ACM Transactions on Networking, Vol. 14, No. 4, 2006, pp. 683–696, https://doi.org/10.1109/TNET.2006.880180.

[20] G. Yang, M. Gerla, M. Sanadidi, Defense against Low-Rate TCP-Targeted Denial-of-Service Attacks, IEEE Symposium on Computers and Communications, 2004, pp. 345–350, https://doi.org/10.1109/ISCC.2004.1358428.

[21] P. Efstathopoulos, Practical Study of a Defense against Low-Rate TCP-Targeted DoS Attack, Proceedings of IEEE ICITST, 2009, pp. 49–54, https://doi.org/10.1109/ICITST.2009.5402593.

[22] B. Braden, D. Clark, et al., Recommendations on Queue Management and Congestion Avoidance in the Internet, RFC 2309, 1998, https://doi.org/10.17487/RFC2309.

[23] S. Floyd, V. Jacobson, Random Early Detection Gateways for Congestion Avoidance, IEEE/ACM Transactions on Networking, Vol. 1, No. 4, 1993, pp. 397–413, https://doi.org/10.1109/90.251892.

[24] S. Floyd, V. Jacobson, Traffic Phase Effects in Packet-Switched Gateways, ACM Computer Communication Review, Vol. 21, No. 2, 1991, pp. 26–42, https://doi.org/10.1145/122419.122421.

[25] W. Feng, D. Kandlur, D. Saha, K. Shin, A Self-Configuring RED Gateway, Proceedings of IEEE INFOCOM, 1999, pp. 1320–1328, https://doi.org/10.1109/INFCOM.1999.752150.

[26] S. Floyd, R. Gummadi, S. Shenker, Adaptive RED: An Algorithm for Increasing the Robustness of RED's Active Queue Management, `https://www.icir.org/floyd/papers/adaptiveRed.pdf`, 2001 (accessed on March 1st, 2024).

[27] F. Baker, G. Fairhurst, IETF Recommendations Regarding Active Queue Management, RFC 7567, 2015, https://doi.org/10.17487/RFC7567.

[28] D. Lin, R. Morris, Dynamics of Random Early Detection, ACM Computer Communication Review, Vol. 27, No. 4, 1997, pp. 127–137, https://doi.org/10.1145/263109.263154.

[29] R. Pan, B. Prabhakar, K. Psounis, CHOKe - A Stateless Active Queue Management Scheme for Approximating Fair Bandwidth Allocation, Proceedings of IEEE INFOCOM, 2000, pp. 942–951, https://doi.org/10.1109/INFCOM.2000.832269.

[30] R. Mahajan, S. Floyd, D. Wetherall, Controlling High-Bandwidth Flows at the Congested Router, Proceedings of IEEE ICNP, 2001, pp. 192–201, https://doi.org/10.1109/ICNP.2001.992899.

[31] T. Ott, T. Lakshman, L. Wong, SRED: Stabilized RED, Proceedings of IEEE INFOCOM, 1999, pp. 1346–1355, https://doi.org/10.1109/INFCOM.1999.752153.

[32] W. Feng, K. Shin, D. Kandlur, D. Saha, The BLUE Active Queue Management Algorithms, IEEE/ACM Transactions on Networking, Vol. 10, No. 4, 2002, pp. 513–528, https://doi.org/10.1109/TNET.2002.801399.

[33] S. Floyd, K. Fall, K. Tieu, Estimating Arrival Rates from the RED Packet Drop History, Draft Paper, 1998.

[34] S. Floyd, K. Fall, Promoting the Use of End-to-End Congestion Control in the Internet, IEEE/ACM Transactions on Networking, Vol. 7, No. 4, 1999, pp. 458–472, https://doi.org/10.1109/90.793002.

[35] B. Bloom, Space/Time Trade-Offs in Hash Coding with Allowable Errors, Communications of the ACM, Vol. 13, No. 7, 1970, pp. 422–426, https://doi.org/10.1145/362686.362692.