Using CPR Metric to Detect and Filter Low-Rate DDoS Flows

Minh Viet Kieu
University of Engineering and Technology
Vietnam National University, Hanoi
15028023@vnu.edu.vn

Dai Tho Nguyen
University of Engineering and Technology
Vietnam National University, Hanoi
UMI UMMISCO 209 (IRD/UPMC), Hanoi, Vietnam
nguyendaitho@vnu.edu.vn

Thanh Thuy Nguyen
University of Engineering and Technology
Vietnam National University, Hanoi
nguyenthanhthuy@vnu.edu.vn

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 the existing algorithm cannot simultaneously achieve high TCP throughput while under attack and good fairness performance for new legitimate TCP flows in normal times. We then propose a new version of the CPR-based approach to overcome the tradeoff. Simulation results show that it preserves TCP throughput while under attack fairly well, yet maintains fairness for new TCP flows in normal times.

CCS CONCEPTS

• **Networks → Denial-of-service attacks**;

KEYWORDS

Low-rate DDoS attack, TCP, AQM, RED.

ACM Reference Format:
Minh Viet Kieu, Dai Tho Nguyen, and Thanh Thuy Nguyen. 2017. Using CPR Metric to Detect and Filter Low-Rate DDoS Flows. In *SoICT '17: Eighth International Symposium on Information and Communication Technology, December 7–8, 2017, Nha Trang City, Viet Nam.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3155133.3155161

1 INTRODUCTION

Distributed denial-of-service (DDoS) attacks have been identified as a major threat to the current Internet. Traditionallly, DDoS attacks consist of numerous compromised computers sending packets as fast as possible toward a victim. The victim or the router in front of the victim's network can be forced to handle intense attack traffic because they cannot distinguish attack packets from normal ones generated by benign computers, resulting in a situation where they do not have enough time and resources, such as network bandwidth, processor cycles, and memory, to serve these computers. Consequently, the legitimate users suffer a degradation or a denial of the victim's service until the attacks stop. Traditional DDoS attacks, on one hand, appear quite effective, but on the other hand, are abnormal due to their coarse behavior of injecting a large volume of packets into the network so they can potentially be detected and alleviated.

In [6], Kuzmanovic and Knightly introduced a new kind of DDoS attack called TCP-targeted low-rate distributed denial-of-service (LDDoS) attack. Compared to the traditional flooding-based DDoS attacks, LDDoS attacks operate in a more stealthy manner by exploiting TCP's retransmission timeout mechanism. As the attack periodically sends high-rate, short duration bursts of packets, it has overall average sending rate sufficiently low to elude existing counter-DDoS mechanisms, but at the same time it can still cause TCP flows to continually timeout leading to near zero throughput.

Since then, there have been a number of mechanisms proposed to defend against this kind of DDoS attacks. Such mechanisms can be divided into two categories: end-point and router-assisted. Since the LDDoS attacks aim at the fixed minRTO and the dependence of the current RTO on minRTO in timeout state, one of the ways to deal with the LDDoS attacks is to choose RTO independently of minRTO. This is the basic idea of a typical end-point mechanism called RTO randomization [9]. It could help TCP flows to avoid LDDoS attacks, but its effect comes too late when almost all TCP flows have entered the timeout state, already causing a decline of the TCP throughput. The mechanism is essentially passive, implying the need of active defense mechanisms. That's the rationale for various router-assisted mechanisms and, in particular, active queue management (AQM) algorithms. Since 1998, the IETF has suggested the deployment of such algorithms in network routers [3] in place of the conventional drop-tail algorithm. One of the most well-known AQM algorithms is Random Early Detection (RED) [4]. RED provides many benefits on network traffic, including almost no bias against bursty traffic and an overall reduction of network delay resulted from its packet dropping policy based on statistical probabilities, avoiding global synchronization when all TCP flows enter timeout and then recover from timeout simultaneously. However, some studies, such as [11], showed that the TCP throughput across a bottleneck link is still vulnerable to LDDoS attack even if RED is used at routers.

RED still needs additional mechanisms to protect TCP flows in the presence of LDDoS attacks. In this sense, Zhang *et al.* [10] proposed a congestion participation rate (CPR) metric for measuring each flow's contribution to network congestion, and showed that the average CPR of normal TCP flows is slightly smaller than 0.2, whereas the average CPR of LDDoS flows is much larger, slightly greater than 0.8. Based on a CPR threshold τ, the authors proposed

SoICT '17, December 7–8, 2017, Nha Trang City, Viet Nam

ACM ISBN 978-1-4503-5328-1/17/12…$15.00
https://doi.org/10.1145/3155133.3155161

an approach integrated into RED (more precisely in front of the RED module), to detect and filter LDDoS flows. The main idea of the approach is that if a flow having CPR greater than τ, it will be classified as an attack flow and consequently all of its arriving packets will be dropped, otherwise the flow will be classified as normal and all of its packets won't be dropped by the approach, but can still be dropped by the RED module. The authors also provided a guidance on how to choose τ that makes the tradeoff between the detection rate and the false positive rate. As τ increases, the probability that a normal TCP flow is misclassified as an attack flow will decrease, but the probability that an attack flow is treated as a normal flow will increase.

As pointed out in [10], when many TCP flows compete for a bottleneck link's bandwidth, and in the presence of a LDDoS attack, the CPR-based approach can effectively distinguish between normal TCP flows and LDDoS flows, but the authors have not considered the actual efficiency of the approach in terms of TCP throughput under attack, have neither figured out how to use the threshold τ, either a best-suited fixed τ or a time-varying τ, to optimize the network performance. Besides the tradeoff between the detection rate and the false positive rate, by performing simulations with NS-2 simulator [1], we find out that there is another tradeoff between maximizing the TCP throughput and providing fairness to new TCP flows in normal times when using this approach. When using a fixed τ network operators are forced to choose between a high TCP throughput under attack (requiring small values of τ), or fairness to new TCP flows (requiring large values of τ). To overcome this tradeoff, we propose a method that adapts τ according to whether the network is under attack or not. The idea behind our method is as follows. In attack times τ should be small to increase TCP throughput, whereas in normal times it should be large to provide more fairness to TCP flows. We also evaluate the performances of our adaptive version of the CPR-based approach by using simulations, and the experimental results show that it can preserve TCP throughput under attack fairly well, while still providing fairness to new TCP flows in normal times. Our contributions makes the CPR-based approach more feasible to be used in real networks.

This paper consists of six sections. The next section, 2, reviews background knowledge. Section 3 presents our analysis of the CPR-based approach. Section 4 describes in detail our method. Section 5 presents simulation results. We conclude our contributions in Section 6.

2 BACKGROUND

2.1 TCP's Timeout Mechanism

TCP congestion control procedures are one of the main reasons for the success of today's Internet despite frequent disturbances and bottlenecks. Two schemes are provided to make TCP flexible and robust in diverse network conditions. When packet loss is rare, the control uses additive increase multiplicative decrease (AIMD) policy, and vice versa when packet loss is frequent, it makes use of TCP's retransmission timeout mechanism.

AIMD control allows TCP flows to adapt their sending rates to the network conditions (available network bandwidth, current congestion status). Concretely, each TCP flow maintains a congestion window *cwnd*, corresponding to the limit total number of packets that can be sent into the network before receiving an acknowledgement. When a TCP flow starts, it has a congestion window of one packet. TCP judiciously speeds up the sending rate by doubling the congestion window every round trip time. This phase is called slow start. When the congestion window 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. One way in which TCP realizes a packet loss is through the receipt of three duplicate acknowledgements for a packet, it then sets its *ssthresh* threshold to one-half of the current *cwnd*, sets *cwnd* to *ssthresh* plus 3, then resends the seemingly lost packet and enters fast recovery phase in which it waits for an acknowledgment of the resent packet before returning to congestion avoidance phase. If the resent packet has not been acknowledged, TCP enters slow start phase with *cwnd* set to one packet.

TCP's retransmission timeout mechanism deals with heavy network congestion by interpreting a packet loss and timeout as a sign of congestion. If so, TCP halves its *ssthresh* and reduces its *cwnd* to one packet, the packet is resent with RTO doubled and TCP enters slow start phase. Upon further timeout, RTO is doubled with each subsequent timeout along with the resending of the lost packet.

Abbreviations:

RTO: Retransmission time out

RTT: Round trip time

RTTVAR: Round trip time variation

SRTT: Smoothed round trip time

G: Clock granularity (usually $\leq$ 100 ms)

The calculation of the current RTO [8] is based on the following formulas:

Initial:

$RTO = 1$ second.

When the first RTT measurement R is made, TCP sender sets:

$$SRTT = R,$$

$$RTTVAR = R/2,$$

$$RTO = SRTT + max(G, 4 \times RTTVAR).$$

When a subsequent RTT measurement R' is made, TCP sender sets:

$$RTTVAR = (1 - \beta) \times RTTVAR + \beta \times |SRTT - R'|,$$

$$SRTT = (1 - \alpha) \times SRTT + \alpha \times R',$$

where $\alpha = 1/8$ and $\beta = 1/4$ (as recommended in [5]),

$$RTO = max(minRTO, SRTT + max(G, 4 \times RTTVAR)),$$

where $minRTO = 1$ second (as recommended in [7]).

Typically we have $minRTO > SRTT + max(G, 4 \times RTTVAR)$, resulting in $RTO = minRTO = 1$ second according to the last equation setting RTO above. Moreover, when timeout occurs, and assume 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

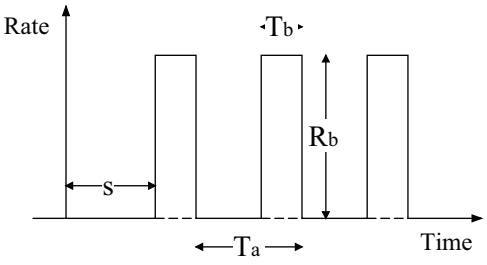


Figure 1: A LDDoS flow.

by an upper bound of at least 60 seconds (as specified in [8]). The sequence of doubling operations, also known as 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 attack cycle, T_b is the attack burst width, R_b is the attack burst rate, and s is the starting time (see Figure 1). Assume that there is only one LDDoS flow in the attack, the flow must have rate R_b large enough to induce loss (R_b should exceed bottleneck link's bandwidth), duration T_b of scale RTT, and period T_a of scale minRTO. In the most common case, a LDDoS attack consists of many LDDoS flows, then it can be modeled as in [10], where a LDDoS attack is represented as a 4-tuple (n, g, m, σ) with n being the total number of attack flows, g the number of attack groups, m the number of flows in each attack group, and σ the starting gap between consecutive attack groups. Refer to [10] for more details.

2.3 Calculating CPR of a Flow

According to [10], the CPR of a flow F_i is calculated by a router in which the CPR-based approach is deployed as follows:

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

where $S_{i,t}$ 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 congested in a sampling period if there is at least one packet dropped at the packet queue during this period. Moreover, we term such a sampling period as congested.

3 ANALYSIS OF THE CPR-BASED APPROACH

The CPR-based approach with τ fixed has not been thoroughly discussed in the paper [10]. Therefore in this section we would like to present some issues that we consider unclear.

3.1 Performances of the CPR-Based Approach

In order to examine the performances of the CPR-based approach, we perform 9 simulations with the platform in [2], each uses the approach with a certain value of τ ranging from 0.1 to 0.9. All simulations start at time 0 and end at time 240, using network topology as in Figure 2. There are 30 long-lived TCP flows, each originates at one of the leftmost computers from User 1 to User 30 and terminates at Server, using FTP application with unlimited data to send. The TCP version is of NewReno with packet size of 1000 bytes. TCP flows all start transmitting packets at time 20 and stop at time 240. We create a LDDoS attack scenario with parameters $n = 20$, $g = 20$, $m = 1$, $\sigma = 1$ second. 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 $T_a = 20$ seconds, $T_b = 200$ ms, $R_b = 5$ Mbps. The attack starts at time 120 ($s = 120$) and stops at time 220.

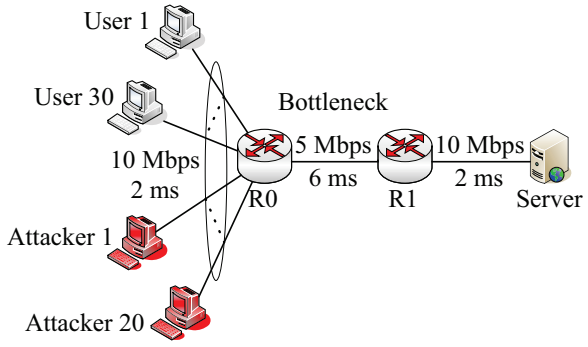


Figure 2: Network topology.

All links from the net have bandwidth of 10 Mbps and one-way propagation delay of 2 ms, except the link between router R0 and router R1 that has bandwidth of 5 Mbps and one-way propagation delay of 6 ms, making it the point of congestion. The queue size of the congested link is 50 packets. RED with the CPR-based approach is deployed at router R0 on the queue of the link, whereas other links use drop-tail queues. The sampling frequency of the approach is 1000 Hz, the same as [10], resulting in sampling periods of 1 ms. We only allow the approach to drop packets after time 120,[1] leaving the TCP flows to share the link's bandwidth freely until the attack starts. This intends to isolate the effect of setting τ only on TCP throughput under attack, making no affect to TCP flows in normal time from time 20 to time 120. Note that we still calculate the CPR for each flow from time 0 although the approach can only drop packets after time 120. To store information of various flows passing through router R0, we use Bloom filters technique that is similar to one in RRED algorithm [11]. In our simulations, we set the number of levels $L = 1$, the bins in each level $N = 4200$, and we use a perfect hash function mapping each flow to a different bin of the only level. TCP throughput in the attack period from time 120 to time 220 is normalized to the link's bandwidth to obtain the result as in Figure 3. As the figure shows, 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 due to the fact that a value of 0.1 is smaller than the average CPR of TCP flows, as

[1]This is done by setting the threshold τ to a value smaller than 1 at time 120, before that τ is assigned to a value greater than 1, in this case, 2. This is due to the CPR of a flow that is always smaller than or equal to 1.

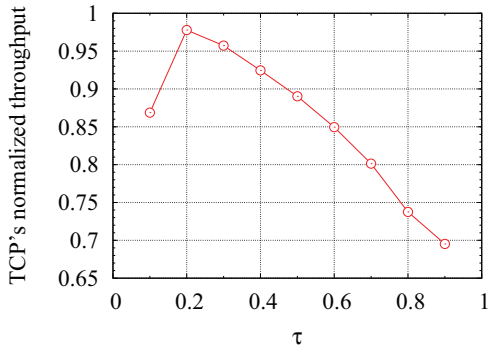


Figure 3: TCP's normalized throughput under LDDoS attack.

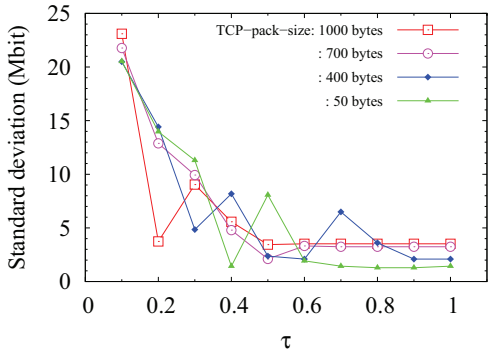


Figure 4: The standard deviation of 10 new TCP flows.

has been pointed out in [10]. This, in turn, results in an increased number of TCP packets incorrectly dropped by the approach.

3.2 Impact of the Approach on Fairness of New TCP Flows

As LDDoS attack does not always happen, every online queue management algorithms and in particular RED with the CPR-based approach, should provide fairness to TCP flows in normal time. If this is not the case, a particular user could experience a long period of time waiting for a data transfer to end. This subsection will explore the fairness among new TCP flows when there is no attack. We perform a set of simulations that still use the network topology in the previous subsection, in which the approach is deployed at router R0 with a certain value of τ ranging from 0.1 to 0.9. As the previous simulations, we only assign each of these values in turn to the threshold τ at time 120. Each value of τ is associated with four TCP packet sizes, this produces total 36 simulations. 4 other simulations correspond to $\tau = 1$ will be described later. In each simulation, there are 10 TCP flows 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 4, 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 used by the approach, and the y-axis showing the standard deviation. Each line shows results from simulations with a specified TCP packet size ranging from 50 bytes to 1000 bytes (not including TCP packet's header size of 40 bytes). The data points corresponding to $\tau = 1$ do not show results from simulations with that value of τ, but show results from simulations in which the approach uses our method for adapting τ (as discussed later in Section 4). The figure shows that the standard deviation varies both with τ and with TCP packet size, with smaller standard deviation for those simulations with a larger τ. 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 our method, the goodputs of the 10 new TCP flows tend to be close to the fair bandwidth share, 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 deviation of the TCP flows corresponding to various values of τ are fully shown in the Table 2 in the last page of this paper.

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

3.3 Convergence of LDDoS Flows' CPRs

Consider the simulation in the first subsection, in which the CPR-based approach is configured with $\tau = 0.5$ and there is such a LDDoS attack. Figure 5 shows the CPR of the first attack flow over a period of 200 ms from time 120 to time 120.2 (exactly over its first burst of packets arriving at the router). In this figure each horizontal line presents a period of 1 ms in which the flow's packets arrive with the same value of CPR. Such a period can contain exactly 12.5 attack packet arrivals on average, this means that every time a period containing 12 packet arrivals, the next period must contain 13 packet arrivals. Horizontal lines with CPR greater than or equals 0.5 mean that every packets 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, that 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, 200 (not shown in this 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.

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

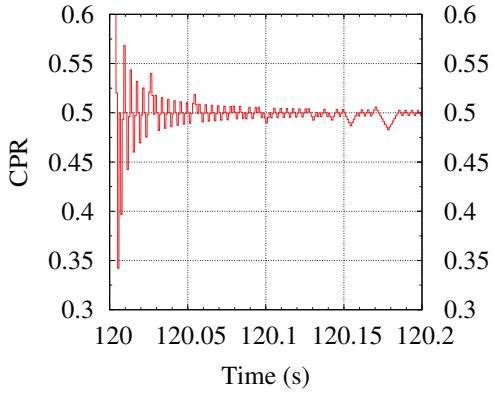


Figure 5: The CPR of the first attack flow.

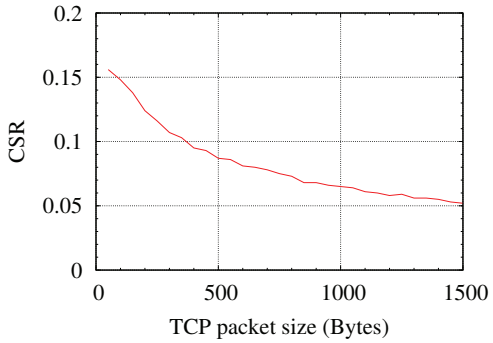


Figure 6: The CSR of the bottleneck link in normal time.

3.4 RED with Congestion Sample Rate

Recall that a sampling period is considered to be a congested period if during this period there is at least one packet dropped at the queue of the outgoing link. We define congestion sample rate (CSR) as the number of congested periods divided by the total number of sampling periods. The CSR is, in fact, the ratio of $|T^*|$ and $|T|$:

$$CSR = |T^*| \Big/ |T| \tag{2}$$

where $|T^*|$ and $|T|$ are notations for the number of elements in the set T^* and T respectively. The difference between CSR and CPR is that CSR is global while CPR is flow specific. Here, we repeat the simulations in the first subsection without LDDoS attack, and at the bottleneck link's queue, instead of using the CPR-based approach, we use pure RED only. In each simulation, for all TCP flows, we use the same TCP packet size ranging from 50 bytes to 1500 bytes, and record the CSR at time 240. This intends to investigate the CSR of the bottleneck link in normal time. Figure 6 shows the obtained results. We can see that the CSR is smaller than 0.2 in all cases, it gradually decreases from slightly above 0.15 at TCP packet size of 50 bytes to about 0.05 when TCP packet size reaches 1500 bytes.

4 OUR PROPOSED METHOD

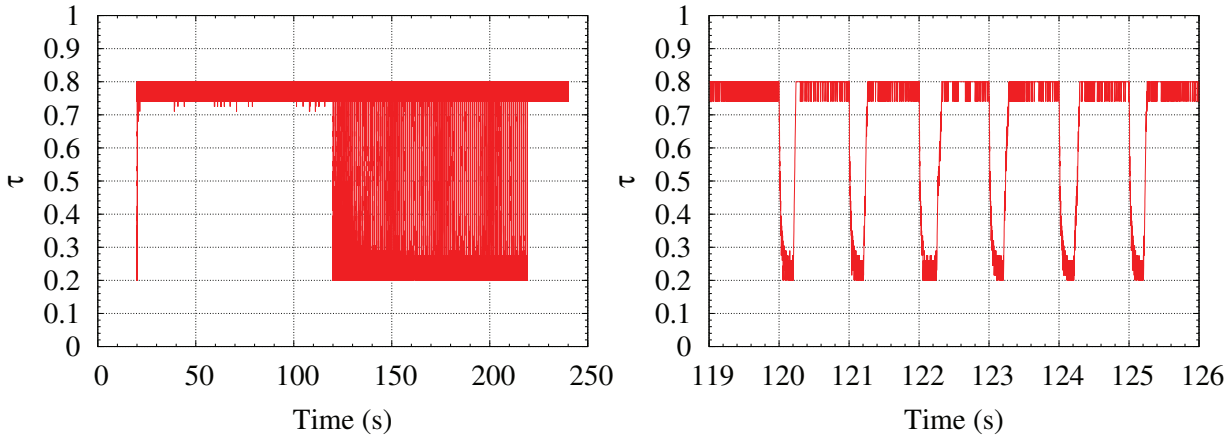
As discussed in detail in previous section, there exists a tradeoff between TCP throughput under attack and fairness among new TCP flows when using the CPR-based approach. This section will present our method of adapting τ proposed to overcome this tradeoff, as given in Figure 7. In our method, τ is 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. Once a LDDoS attack has been launched, the bottleneck link is more likely to be congested, and thus τ is reduced by our method, in steps of α. To react quickly to the attack, τ should be reduced to 0.2, or close to 0.2, over a time period 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 CPR of LDDoS flows to the τ's value described in the previous section, and τ cannot be reduced to a value smaller than 0.2, the CPR of each LDDoS flow tends to converge to 0.2. Because each LDDoS flow has 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 time is less than 0.2. The ability of τ increasing to 0.8 and oscillates close to the peak value when an attack burst ends 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 swinged in the range [0.74, 0.8] once τ reaches 0.8 in normal time.

```
τ is initiated to 0.8;
Every sampling period miliseconds:
      If the outgoing link is congested then
            τ = τ − α;
            If (τ < 0.2) then τ = 0.2;
      Else
            τ = τ + β;
            If (τ > 0.8) then τ = 0.8;
      End if

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

Figure 7: Our proposed method.



(a) Adapting over the simulation time. **(b) Adapting over [119, 126].**

Figure 8: The adapting of τ.

5 EXPERIMENTAL EVALUATION

5.1 Threshold Adaptation

To validate the analysis, we re-run the simulation in the first subsection of section 3 with a change of using our method for adating τ from time 0 instead of using τ fixed. The adapting of τ over the simulation time and over a short time period from time 119 to time 126 are given in Figure 8. It shows that when there is no attack τ mainly fluctuates in the range [0.74, 0.8], except the case when 30 normal TCP flows start up at 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 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.

5.2 Performances of the Adaptive CPR-Based Approach

In order to evaluate the performances of the adaptive CPR-based approach, in this subsection we perform three set of simulations as in the paper [10] called Attack Frequency Intensification (AFI), Attack burst Width Intensification (AWI), and Attack burst Rate Intensification (ARI). The only difference is that we use wider ranges of values for parameters T_a, T_b, R_b as in Table 1. T_a is varied in [4, 40] (s) instead of [20, 40] (s), T_b is varied in [0, 50] (ms) instead of [0.1, 10] (ms), and R_b is varied in [0, 0.5] (Mbps) instead of [0.01, 0.25] (Mbps). This aims to examine the robustness of the adaptive CPR-based approach not only against LDDoS attacks, but also against general DDoS attacks. In all simulations we still use the setting for TCP flows as in the first subsection of section 3, the attack still starts at time 120 and stops at time 220. Adapting method is always turned on at time 0. For comparison, we also use three instances of non-adaptive CPR-based approach with $\tau = 0.2$, $\tau = 0.6$, and $\tau = 0.8$. In each instance 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 in normal time, especially with small values such as 0.2. In simulation results shown in Figure 9, we see that adaptive CPR-based approach (depicted by red lines) is slightly worse than non-adaptive CPR-based approach with $\tau = 0.2$ (depicted by magenta lines), whereas non-adaptive CPR-based approach with $\tau = 0.6$ or $\tau = 0.8$ (depicted by blue and green lines respectively) is considerably worse. Performance of all variants is decreased at different rates when the attack becomes more aggressive (T_a

Table 1: Parameters of LDDoS attack.

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]

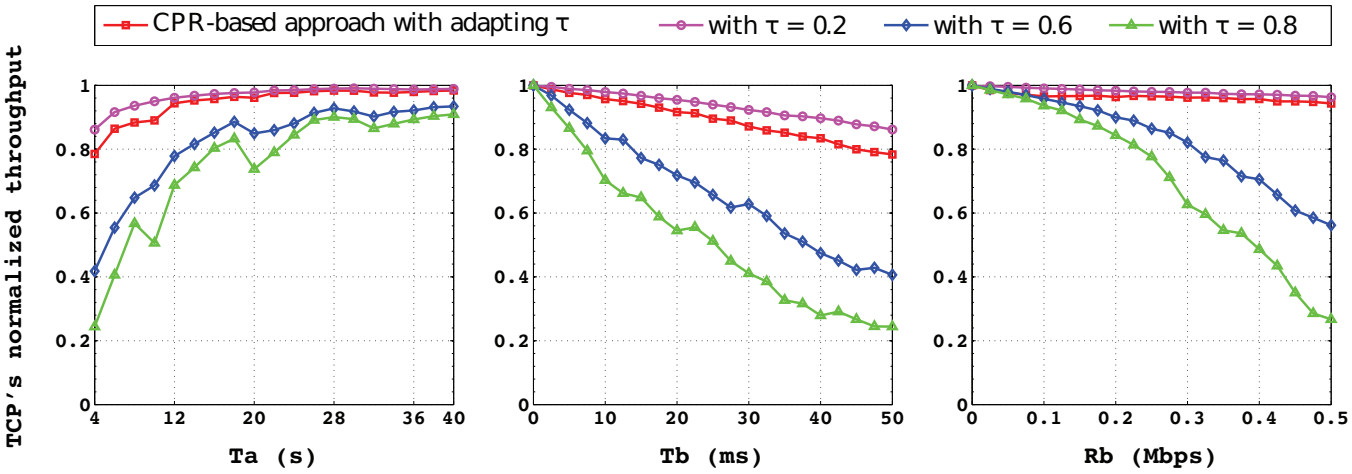


Figure 9: TCP's normalized throughput under attack.

decreases, T_b increases, R_b increases), with smaller rate for non-adaptive CPR-based approach with smaller τ and for adaptive one. At two extreme cases of $T_a = 4$ s and $T_b = 50$ ms when a full-time DDoS happens, adaptive CPR-based approach retains about 80% of the bottleneck link's bandwidth for TCP throughput, and the remaining 20% of the bandwidth is wasted for attack traffic, this means that attack traffic is reduced to 1/5-th of its original size. Non-adaptive ones with $\tau = 0.2$, $\tau = 0.6$, and $\tau = 0.8$ retain about 80%, 40%, and 20% of the link's bandwidth for TCP throughput, respectively. In operation of our adaptive CPR-based approach there are usually about 20% of samping periods in each attack burst in which attack traffic still goes through the link. This is an inherent drawback of our adapting method as well as the original CPR-based approach. Fixing the drawback will be part of our future work.

6 CONCLUSION

Based on simulations, we have showed that when using the CPR-based approach to counter LDDoS attack, there is a tradeoff between maximizing TCP throughput while under attack and providing fairness to new TCP flows in normal times. We have also proposed an adaptive version of the CPR-based approach. The simulation results show that our proposition helps the approach to simultaneously achieve high TCP throughput while under attack and fairness among new TCP flows, thus increasing its feasibility for being used in real networks.

REFERENCES

[1] 2005. NS-2 simulator. http://www.isi.edu/nsnam/ns/. (2005).
[2] 2011. AQM&DoS simulation platform. https://sites.google.com/site/cwzhangres/home/posts/aqmdossimulationplatform/. (2011).
[3] B. Braden, D. Clark, and many others. 1998. *Recommendations on queue management and congestion avoidance in the Internet.* RFC 2309.
[4] S. Floyd and V. Jacobson. 1993. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking* 1, 4 (1993), 397–413.
[5] V. Jacobson and M. Karels. 1988. Congestion avoidance and control. *ACM Computer Comm. Review* 18, 4 (1988), 314–329.
[6] A. Kuzmanovic and E. Knightly. 2003. Low-rate TCP-targeted denial of service attacks (The shrew vs. the mice and elephants). In *Proceedings of ACM SIGCOMM.*
[7] V. Paxson and M. Allman. 1999. On estimating end-to-end network path properties. In *Proceedings of ACM SIGCOMM.*
[8] V. Paxson, M. Allman, J. Chu, and M. Sargent. 2011. *Computing TCP's retransmission timer.* RFC 6298.
[9] G. Yang, M. Gerla, and M. Sanadidi. 2004. Defense against low-rate TCP-targeted denial-of-service attacks. In *IEEE Symposium on Computers and Communications.*
[10] C. Zhang, Z. Cai, W. Chen, X. Luo, and J. Yin. 2012. Flow level detection and filtering of low-rate DDoS. *Elsevier Computer Networks* (2012).
[11] C. Zhang, J. Yin, Z. Cai, and W. Chen. 2010. RRED: Robust RED algorithm to counter low-rate denial-of-service attacks. *IEEE Communications Letters* 14, 5 (2010).

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**