Techniques for Improving Performance of the CPR-Based Approach

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 have created an opportunity for attackers to reduce their total attaking rate (and hence, the detection probability of the attacks) while inflicting the same damage to TCP flows as traditional flooding-based DDoS attacks. CPR-based approach has been proposed by Zhang et al. to detect and filter this kind of DDoS attacks, but its performance in terms of TCP throughput under attack is shown to be limited by the way it calculates CPR for each flow. In this paper, we will propose some modifications to the CPR-based approach in order to increase its performance. Simulation results show that the modifications can increase performance significantly.

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. 2018. Techniques for Improving Performance of the CPR-Based Approach. In *SoICT '18: Ninth International Symposium on Information and Communication Technology, December 6–7, 2018, Da Nang City, Viet Nam.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3287921.3287940

1 INTRODUCTION

The Internet was built several decades ago and has become very popular everywhere in the world. Although the performance of the Internet in terms of network delay, throughput, or network congestion has been improved significantly since its inception so far, its operational principle remains nearly unchanged, that is, it is still operating based on two cornerstones: best-effort service and end-to-end paradigm. In the center of the Internet, routers are responsible for relaying packets from source to destination. These source and destination are actually end computers where packets are sent and received. Best-effort service means that the intermediate routers simply perform the task of storing and forwarding packets. All other tasks, for example and most prominently, network congestion control, are left for end computers. So end-to-end paradigm, and especially end-to-end congestion control, plays an indispensable role in avoiding and mitigating network congestion in the current Internet.[1] Currently, TCP is still the dominant transport protocol in the Internet and so does its congestion control. Without TCP congestion control, the Internet can become severely congested and unusable as in October 1986 when the Internet suffered a series of congestion collapses [5].

But the problem has not stopped there yet. If a computer becomes malicious by continually sending packets into the network at a very high rate or just simply fails to use end-to-end congestion control, it will hurt other computers that are comunicating as some resources, such as network bandwidth, computer's or router's processor cycles, memory, are exhausted. In the meanwhile, intermediate network still passively transmits packets to their destination and does nothing to prevent traffic from the misbehaving computer. This well-known phenomenon is called denial-of-service (DoS) attack. If there are more than one computers involved in the attack, it is called distributed denial-of-service (DDoS) attack. Internet can bring us a convenient way to access information, communicate with each others but it also carries within its functional architecture the origin of the DDoS threat.

The problem of DDoS attacks has finally been recognized and there has been an increasing agreement that additional mechanisms are needed at routers to protect the Internet from DDoS attacks and computers that send more than their fair share. Since 1998, the IETF has suggested the deployment of active queue management (AQM) algorithms, such as Random Early Detection (RED) [4], in network routers for congestion avoidance purposes and in order to replace traditional drop-tail queue management algorithm [3]. RED has almost no bias against bursty traffic[2] and also offers an overall reduction of network delay resulted from its packet dropping policy

SoICT '18, December 6–7, 2018, Da Nang City, Viet Nam

ACM ISBN 978-1-4503-6539-0/18/12…$15.00
https://doi.org/10.1145/3287921.3287940

[1] In recent years, due to the tremendous growth of traffic demand, a technique called time-dependent pricing (TDP) is being developed to reduce network congestion further. TDP provides incentives for delay-tolerant, price-sensitive users to shift their traffic demand from congested (with high price) to less-congested (with lower price) periods, thereby reducing network congestion. Besides, there are some new paradigms, such as software-defined networks (SDNs) or network functions virtualization, which enable end-to-end QoS guarantee. To see a combination of TDP and SDN in cellular networks, please refer [10].

[2] In [4] the authors refer to bursty traffic as traffic from a connection where the amount of data transmitted in one roundtrip time is small compared to the delay-bandwidth

based on statistical probabilities where the exact probabilities are computed as a function of the average queue size. By adding the randomness factor in packet dropping decisions, RED routers help TCP flows to avoid global synchronization as happened with drop-tail routers in which TCP flows enter timeout and then recover from timeout simultaneously every time packet queue is full. The advantages of RED are also the shortcomings of the combination model between TCP congestion control and the use of traditional drop-tail algorithm at routers.

RED is designed to accompany a transport-layer congestion control protocol, such as TCP, and is shown to be better than drop-tail algorithm in cooperation with TCP flows, but the design has been done with the lack of considering RED's performance under DDoS attacks. In [9], the authors showed that TCP throughput across a bottleneck link is still decreased sharply under low-rate DDoS (LDDoS) attacks, a new kind of DDoS attacks, even if RED is used at routers. LDDoS attacks have been introduced for the first time by A. Kuzmanovic and E. Knightly in [7]. By exploiting TCP's retransmission timeout mechanism, a LDDoS attacker can reduce his total attacking traffic rate manyfold while inflicting the same damage as traditional flooding-based DDoS attacks on TCP flows, so his traffic flows are very much like well-behaved TCP flows, even better with respect to some metric (e.g., sending rate, bursty characteristic) and in a large-scale LDDoS attack.

Recently, CPR-based approach [8] has emerged as an effective algorithm to counter LDDoS attacks. It is a flow-based technique in the sense that it calculates Congestion Participation Rate (CPR) metric for each flow passing a router. CPR-based approach is deployed in front of the RED module in routers. If a flow having CPR greater than a CPR threshold, 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 a normal TCP flow and all of its packets won't be dropped by the approach, but can still be dropped by the RED module. The approach can effectively detect and filter LDDoS flows in the presence of a LDDoS attack, but its performance in terms of TCP throughput under attack is limited by the way it calculates CPR (see [6] for more details). In this paper, we will introduce some modifications to the CPR-based approach in order to increase its performance. Simulations with NS-2 simulator [1] will be used to demonstrate our comparison. The rest of this paper is organised as follows. In section 2, we discuss in details about the current problem and then present our ideas to solve this issue. Section 3 is the simulation results. We conclude this paper in Section 4.

product, but where multiple packets from that connection arrive at a router in a short period of time.

2 CURRENT PROBLEM AND OUR IDEAS

CPR-based approach divides time into consecutive small non-overlapped periods. Consider a particular period of time and assume that at the starting of the period a flow has CPR smaller than the current CPR threshold (e.g., when it traverses the router for the first time), its packets can definitely add to the queue length of the router without being filtered by the approach until the period ends. This is due to the fact that CPR of a flow is always updated at the end of each sampling period. This shortcoming of the CPR-based approach can lead to a full queue at the router after only one sampling period and the queue remains full thereafter if an attacker arranges a large-scale LDDoS attack and keeps the total sending rate sufficiently high (higher than the rate at which the bottleneck link can serve). We think that an improvement to the performance of the CPR-based approach can be lied in updating CPR for each flow not only at the end of each period, but also at the time when a packet is dropped. This updating technique can prevent an attack flow from flooding router during a period when its CPR is less than the current CPR threshold.

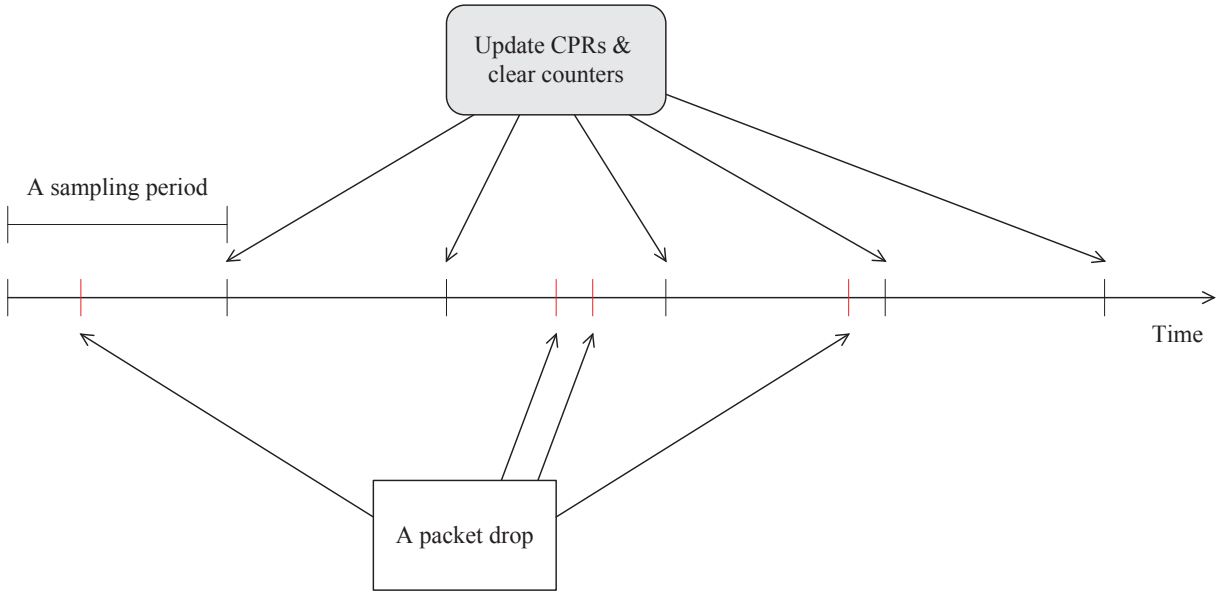

Figure 1: Calculation of the CPR metric.

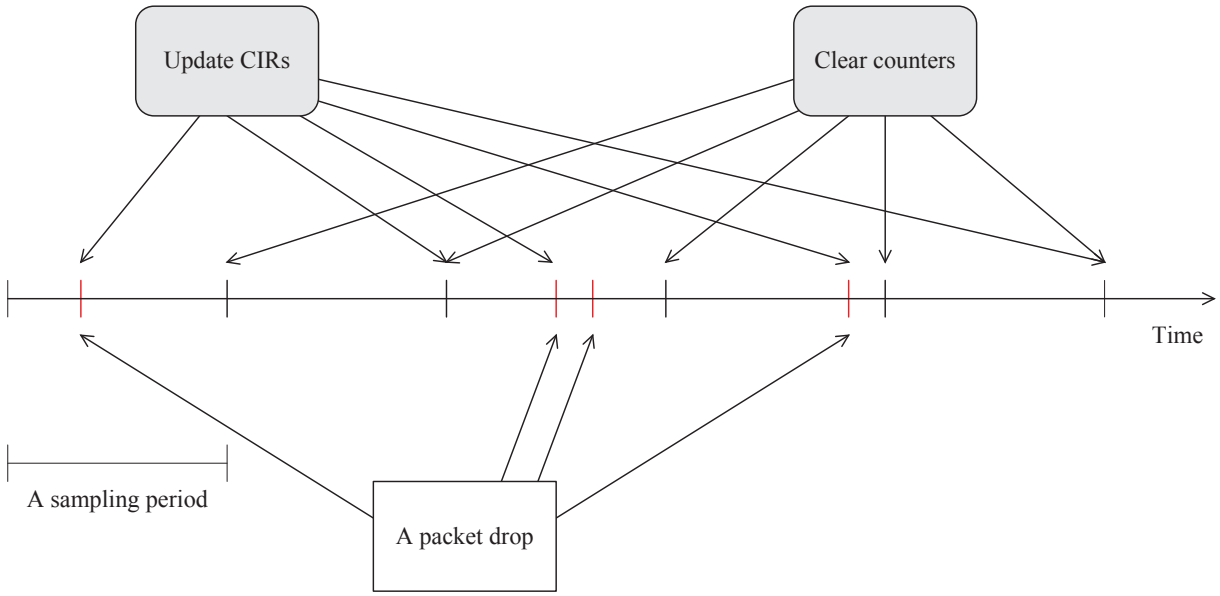

Figure 2: Calculation of the CIR metric.

To accomplish the idea above, in this paper we propose Congestion Interval Rate (CIR) metric. The CIR of a flow F_i is calculated by:

$$\zeta_i = |T^*| \Big/ |T| \tag{1}$$

where $|T^*|$ and $|T|$ are notations for the number of elements in the set T^* and T respectively. T^* is the set of sampling periods when flow F_i is active and the outgoing link is congested. T is the set of sampling periods when flow F_i is active. A flow is considered to be active in a sampling period if it has at least one packet arriving at router during this period. 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. If the outgoing link is congested in a sampling period, the period is called congested period. Each period of time can be represented by $[t, t + d]$, in which t is the starting time and d is the duration of the period and is empirically chosen to be 1 ms.

Difference between the CPR and CIR metrics is shown in Figures 1 and 2. The Figure 1 shows that CPR of every flow is only updated at the end of each period (using Equation 1 in the original paper [8]) and then all of its counters, e.g. counter for storing the number of packets that have been arrived since the starting time of the current period, are cleared. With CIR metric shown in the Figure 2, if there is at least one packet dropped during a sampling period, then CIRs of all currently active flows (considered from the starting time of the period to the arrival time of the first packet dropped) are updated at the time of the first packet dropped (using Equation 1 in this paper). These active flows do not have to wait until the end of the period to update their CIRs. For convenience, we name this technique as *early update*. Next, the counters for these flows are not erased, they are marked to indicate that the flows' CIRs are already updated. With subsequent incoming packets, the CIRs of their associated flows will be updated only if these values have not been already updated. At the end of the period, all counters will be erased to reuse in the next period. In the case of a period with no packet drop, updating CIRs and clearing counters all happen at the end of the period.

Detailed algorithm for CIR-based approach is presented in Figure 5 in which the CIR threshold is still denoted by τ as in the CPR-based approach. The CIR-based approach operates as follows. When a packet, denoted by pkt, arrives at router, its associated flow, denoted by f, will be computed using hash function. If the current sampling period is not congested ($congested = 0$) and f is not active then f is marked as active. If the current sampling period is congested ($congested = 1$) and $f.CIR$ has not been updated then $f.CIR$ is updated and marked as updated. Next, $f.CIR$ is compared to τ. If $f.CIR$ is greater than or equal to the threshold, f will be classified as an attack flow and pkt will be dropped, otherwise the flow will be classified as a normal TCP flow and pkt will be passed to the RED block (pkt can still be dropped by the RED block). When RED block drops one packet and the current sampling period is not congested, the period is set to be congested by setting the *congested* variable to 1. After that, all CIRs of currently active flows will be updated and marked as already updated. CIR-based approach has a routine running at the end of every sampling period. At that time, if the current sampling period is congested, indicating that CIR of all flows has been updated and we only have to clear the counters to mark all flows as inactive and their CIRs as not updated for the next sampling period. If the current sampling period is not congested, we have to update CIR for all active flows and then clear the counters. The last action of the routine is setting *congested* variable back to 0.

3 SIMULATION RESULTS

3.1 CIRs of TCP flows in normal time

To examine the CIRs of normal TCP flows in normal time (i.e. when there is no LDDoS attack), in this subsection we perform

a simulation with the platform in [2] in which instead of using CPR-based approach we use CIR-based approach with a CIR threshold still denoted by τ. The simulation starts at time 0 and ends at time 120, using the network topology as in Figure 3.

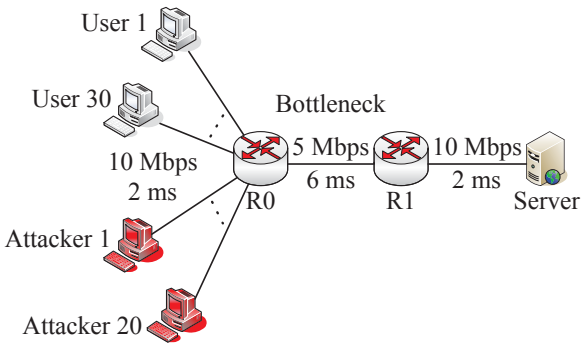

Figure 3: Network topology.

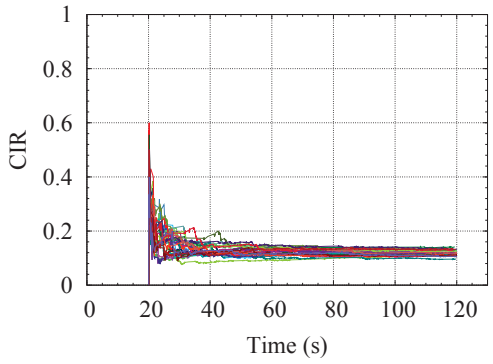

Figure 4: CIRs of 30 normal TCP flows going through the bottleneck link.

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 120. 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 a 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 CIR-based approach[3] 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 [8], resulting in sampling periods of 1 ms. To store information of various flows passing through router R0, we use Bloom filters technique that is similar to one in RRED algorithm [9]. In our simulation, 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.

[3]In this simulation CIR threshold is a constant and is set to 2. With this threshold the CIR-based approach does not drop packets during simulation time because CIR of every flow is always less than or equal to 1, thereby dropping packets is only managed by RED algorithm.

```
For each incoming packet pkt:
f is the associated flow of the packet
   If congested == 0 then
      If f is not active then
         f is marked as active;
      End if
   Else
      If f.CIR has not been updated then
         update f's CIR;
         f.CIR is marked as updated;
      End if
   End if
   If f.CIR ≥ τ then
      drop(pkt);
   Else
      pass pkt to the RED block;
      If RED drops pkt && congested == 0 then
         congested := 1;
         update CIR of all currently
         active flows;
         mark all their CIRs as updated;
      End if
   End if

At the end of each sampling period:
   If congested == 1 then
      mark all flows as inactive and
      their CIRs as not updated;
   Else
      update CIR of all active flows;
      mark all flows as inactive and
      their CIRs as not updated;
   End if
   congested := 0;

Parameters:
congested: the congestion status of the
current sampling period
τ: the CIR threshold
sampling period: time; 1 miliseconds
```

Figure 5: Detailed algorithm for CIR-based approach.

Figure 4 shows that the CIR of each normal TCP flow converges to an equilibrium point, with a slight change in value from one to another, but all points are below 0.2.

3.2 CIR difference between normal TCP flows and LDDoS flows

In this subsection we will investigate the difference between CIRs of normal TCP flows and CIRs of LDDoS flows. We perform three sets of simulations called Attack Frequency Intensification (AFI), Attack burst Width Intensification (AWI), and Attack burst Rate Intensification (ARI) with parameters shown in Table 1 (please refer [8] for more details about how to model LDDoS attacks). There are 20 attack flows, each originates at one of the 20 attacking computers from Attacker 1 to Attacker 20 and terminates at Server (see Figure 3). All attack flows send UDP packets with packet size of 50 bytes. In each set of simulations, for each single attack flow, we vary one parameter and fix two others. All simulations start at time 0 and end at time 240 in which TCP flows are configured the same as in previous subsection, except that they stop at time 240 instead of time 120. LDDoS attacks start at time 120 and stop at time 220. At time 240 of each simulation, we record the minimum, maximum, and average CIRs of normal TCP flows and those of LDDoS flows. The results are shown in Figure 6 in which the blue lines depict the average CIRs of normal TCP flows and the red lines depict the average CIRs of LDDoS flows. These lines are extended (depicted by orange and green colors respectively) to the minimum and maximum lines that connect minimum CIRs and maximum CIRs of the corresponding groups of flows. The central and rightmost subfigures of Figure 6 don't plot the points corresponding to LDDoS flows with T_b = 0 ms and R_b = 0 Mbps because these values are just 0. In the rightmost subfigures, the average line of LDDoS flows seems to be not extended. In fact, we still plot the minimum and maximum lines and fill the area between them with orange color but the lines are very close to each other. The reason behind this is that in the set ARI where R_b is varied, 20 attacking computers are scheduled to start transmitting packets at the same time and with the same rate, so the attack flows' CIRs are nearly equal. From Figure 6 we can conclude that the CIR metric can differentiate LDDoS flows from normal TCP flows. It also shows that when an attack becomes more aggressive by reducing attack cycle T_a or by increasing attack burst width T_b, the maximum and average CIRs of normal TCP flows tend to increase while their minimum CIRs are more stable. As R_b increases, the maximum and average CIRs of normal TCP flows increase slightly.

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]

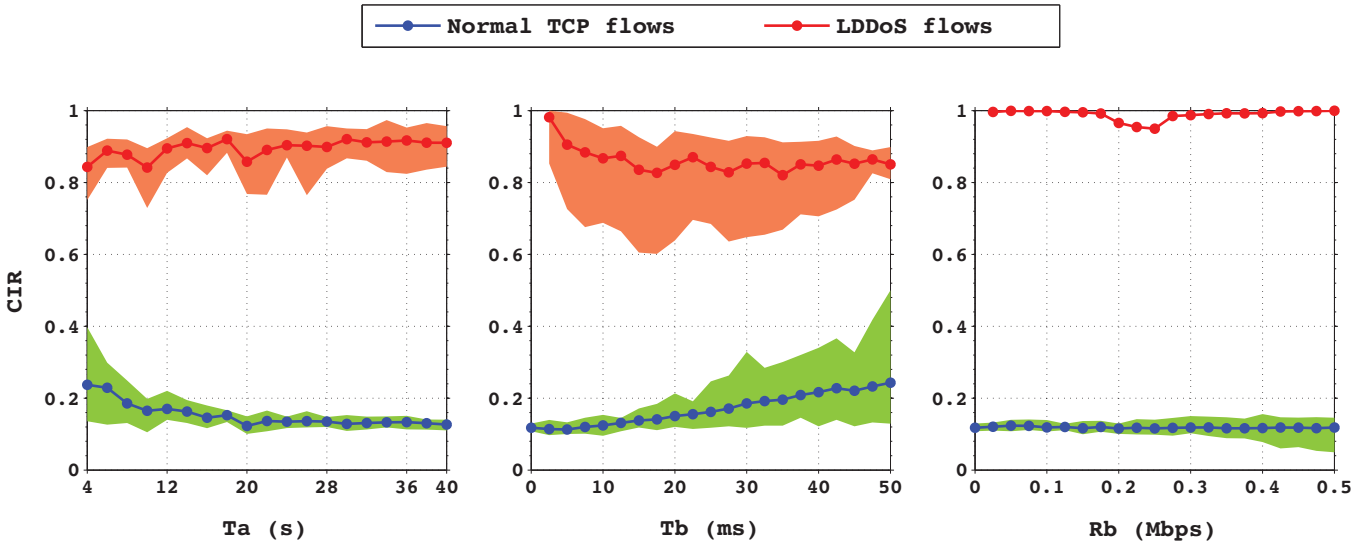

Figure 6: CIR difference between normal TCP flows and LDDoS flows.

The minimum CIR tends to decrease as R_b increases because at time 220 when the attacks finish, there is a small number of TCP flows (fewer than 30) recovering from timeout state and competing to utilize the bottleneck link's bandwidth while other TCP flows are still in timeout state with the CIRs nearly unchanged, so during the time period from time 220 to time 240 when simulations finish, the flows can get smaller CIRs than before when there are 30 competing TCP flows, and the higher the attack rate is the fewer the number of the flows is, thereby the minimum CIR of TCP flows is decreased.

3.3 Performance comparison of CPR-based approach and CIR-based approach

To compare the performance of the approaches, in this subsection we perform two sets of simulations, each corresponds to the use of one of the approaches at router R0. Each set consists of 9 simulations corresponding to the use of τ with values ranging from 0.1 to 0.9. All simulations start at time 0 and end at time 240. The setting for TCP flows is the same as in the previous subsection. 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 and stops at time 220. We only allow the two approaches to drop packets after time 120,[4] 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. 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 7.

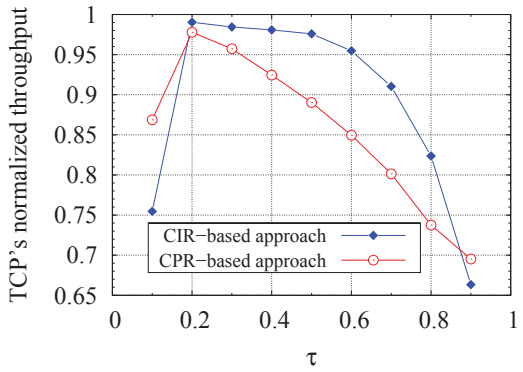

Figure 7: TCP's normalized throughput under LDDoS attack.

In this figure, the blue and red lines respectively represent the performance of the CIR-based and CPR-based approaches with different values of τ. The blue line is always lying on the red line, except the two marginal values of τ, $\tau = 0.1$ and $\tau = 0.9$. This shows that the performance of the CIR-based approach is higher than that of the CPR-based approach.

[4]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 fact that the CPR and the CIR of a flow are always smaller than or equal to 1.

4 CONCLUSION

In this paper, we have shown that the performance of the CPR-based approach in terms of TCP throughput under attack can be improved. Derived from the CPR metric in [8], we have proposed a new metric, called CIR, for differentiating LDDoS attack flows and normal TCP flows and a technique called *early update* to help the CPR-based approach to achieve an increased performance. The simulation results show that our modifications to the approach can improve its performance significantly under a regular LDDoS attack scenario where each attack flow from 20 attack flows takes on a stage of one second in every cycle of 20 seconds in the attack. Our future work will be discovering relationship between link's drop rate and the CIR metric.

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 Communication Review* 18, 4 (1988), 314–329.
[6] M. Kieu, D. Nguyen, and T. Nguyen. 2017. Using CPR Metric to Detect and Filter Low-Rate DDoS Flows. In *Proceedings of ACM SoICT*. 325–332.
[7] 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*. 75–86.
[8] C. Zhang, Z. Cai, W. Chen, X. Luo, and J. Yin. 2012. Flow level detection and filtering of low-rate DDoS. *Elsevier Computer Networks* 56, 15 (2012), 3417–3431.
[9] 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), 489–491.
[10] Z. Zhou, L. Tan, B. Gu, Y. Zhang, and J. Wu. 2018. Bandwidth Slicing in Software-Defined 5G: A Stackelberg Game Approach. *IEEE Vehicular Technology Magazine* 12, 2 (2018), 102–109.