RRED: Robust RED Algorithm to Counter Low-Rate Denial-of-Service Attacks

Changwang Zhang, Jianping Yin, Zhiping Cai, and Weifeng Chen

***Abstract*—The existing Random Early Detection (RED) algorithm and its variants are found vulnerable to emerging attacks, especially the Low-rate Denial-of-Service (LDoS) attacks. In this letter we propose a Robust RED (RRED) algorithm to improve the TCP throughput against LDoS attacks. The basic idea behind the RRED is to detect and filter out attack packets before a normal RED algorithm is applied to incoming flows. We conduct a set of simulations to evaluate the performance of the proposed RRED algorithm. The results show that, compared to existing RED-like algorithms, the RRED algorithm nearly fully preserves the TCP throughput in the presence of LDoS attacks.**

***Index Terms*—AQM, low-rate DoS attack, RED, robust.**

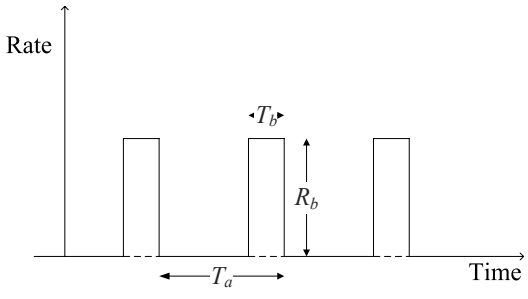

Fig. 1. LDoS attack stream.

I. Introduction

In the past decades, quite a few Active Queue Management (AQM) algorithms such as Random Early Detection (RED) [1] and its variants have been proposed to handle congestion and to improve the TCP performance ([1], [2], [3], [4]). Although these AQM algorithms are highly robust to diverse network conditions, most of them were designed without considering their robustness against network attacks, such as the Denial-of-Service (DoS) attacks that have been identified as a major threat to today's Internet services. Example DoS attacks include TCP SYN attacks, ICMP directed broadcasts and DNS flood attacks. These attacks normally generate high-rate transmission of packets toward the victim node. They can be detected and alleviated [5]. Recently a new kind of DoS attack, low-rate DoS attack, has been proposed in [6] that exploits TCP's retransmission timeout mechanism to reduce TCP throughput without being detected. Compared to traditional flooding based DoS attacks, the low-rate DoS attack does not employ a "sledge-hammer" approach of high-rate transmission of packets, and consequently eludes detection. RED-like algorithms have already been found to be notably vulnerable to LDoS attacks [7].

In this letter, we propose a novel Robust RED (RRED) algorithm to thwart the LDoS attacks. The RRED algorithm consists of a new detection algorithm and a traditional RED algorithm. The basic idea behind the RRED is to detect and filter out LDoS attack packets from incoming flows before they feed to the RED algorithm. Bloom filter techniques [8] have been used in the implementation of the RRED algorithm to manage potentially numerous incoming flows and increase the detection accuracy. We conduct a set of simulations to evaluate the performance of the proposed algorithm. Experiment results show that the RRED algorithm is highly robust when the router is under an LDoS attack. The TCP traffic remains its ideal rate and TCP throughput is nearly fully preserved.

Manuscript received July 6, 2009. The associate editor coordinating the review of this letter and approving it for publication was F.-N. Pavlidou.

This work is supported in part by the National Natural Science Foundation of China (No.60970034, No.60603062, and No.60903040) and Natural Science Foundation of Hunan Province (06JJ3035).

C. Zhang, J. Yin, and Z. Cai are with the School of Computer Science, National University of Defense Technology, Changsha, China (e-mail: mleoking@gmail.com).

W. Chen is with the Department of Math & Computer Science, California University of Pennsylvania, USA.

II. LDoS Attacks

Following the notations in [6] and [9], we describe an LDoS attack using three parameters (T_a, T_b, R_b). As shown in Fig. 1, T_a represents the attack period, T_b represents the attack burst width, and R_b represents the attack burst rate.

The LDoS attack exploits TCP's slow-time-scale dynamics of retransmission time-out (RTO) mechanisms to reduce TCP throughput [6]. Basically, an attacker can cause a TCP flow to repeatedly enter a RTO state by sending high-rate (R_b), but short-duration bursts (T_b), and repeating periodically at slower RTO time-scales (T_a). The TCP throughput at the attacked node will be significantly reduced while the attacker will have low average rate making it difficult to be detected.

A critical observation needs to be noted here. Within a benign TCP flow, the sender will delay sending new packets if loss is detected (e.g., a packet is dropped). Consequently, a packet is suspected to be an attacking packet if it is sent within a short-range after a packet is dropped. This is the basic idea of our detection algorithm presented in Section III.

III. Robust RED (RRED)

In this section, we explain the design and implementation of the RRED algorithm. Fig. 2 describes the basic architecture of the RRED algorithm. A detection and filter block is added in front of a regular RED block on a router. The basic idea behind the RRED is to detect and filter out LDoS attack packets from incoming flows before they feed to the RED algorithm. How to distinguish an attacking packet from normal TCP packets

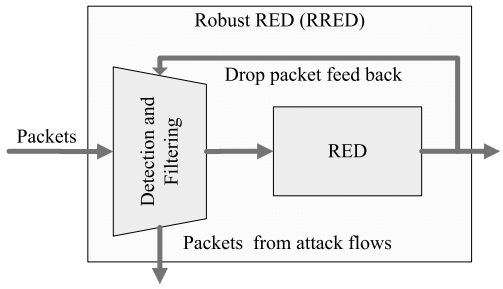

Fig. 2. Architecture of Roust RED (RRED).

is critical in the RRED design. This is achieved based on the observation mentioned in Section II.

All incoming TCP packets to a router belong to different flows. Here, a flow is defined by a 5-tuple *(Source IP, Source Port, Destination IP, Destination Port, Protocol)*. We use an indicator $f.I$ to judge whether flow f is an LDoS attack flow or a normal TCP flow. Specifically, $f.I$ is calculated as follows. If a packet from flow f is considered to be an attacking packet (described below), $f.I$ is decreased by one; if it is considered to be a normal packet, $f.I$ is increased by one. Then an incoming packet from a flow with a negative $f.I$ is filtered. Packets from a flow with a positive or zero $f.I$ will further feed to the RED block.

An incoming packet from flow f is suspected to be an attacking packet if it arrives within a short-range after a packet from f that is dropped by the detection and filter block or after a packet from any flow that is dropped by the RED block. The following process is used to define this short-range. For every flow f (either a normal TCP flow or an LDoS flow), let $f.T_1$ be the arrival time of the last packet from f that is dropped by the detection and filter block. Let T_2 be the arrival time of the last packet from any flow that is dropped by the RED block. The short-range is defined as $[T_{max}, T_{max} + T^*]$, in which $T_{max} = MAX(f.T_1, T_2)$. If the arrival time of an incoming packet from flow f falls into this range, the packet is suspected to be an attacking packet. Note that T_1 is flow specific while T_2 is global, which capture the fundamental characteristics of an LDoS attack flow and the global impact of the attack on the whole network, respectively.

A proper value should be chosen for T^* to (i) filter most attacking packets, and to (ii) pass most normal packets. In this letter, we empirically choose T^* to be 10ms, which works quite well for diverse LDoS attacks, as will be seen in Section IV. Our future work is to design a mechanism that can adaptively change T^* on the fly.

Fig. 3 shows the pseudo codes of the RRED algorithm. We have implemented the RRED algorithm in C. Anyone interested in the code can contact the authors. In Fig. 3, pkt denotes an incoming packet; f is the flow index hashed using pkt's source-destination address pair via function $RRED - FLOWHASH()$.

A router may receive packets from potentially numerous flows whereas we need to keep T_1 and I for each flow. To solve this problem we use Bloom-filters technique that is similar to SFB [3]. The Bloom filters in our RRED implementation are constructed with L levels with each level containing N

$RRED - ENQUE(pkt)$
1: $f \leftarrow RRED - FLOWHASH(pkt)$
2: $T_{max} \leftarrow MAX(Flow[f].T_1, T_2)$
3: **if** $pkt.arrivaltime \in [T_{max}, T_{max} + T^*]$ **then**
4: $\quad reduce\ local\ indicator\ by\ 1\ for\ each\ bin\ of\ f$
5: **else**
6: $\quad increase\ local\ indicator\ by\ 1\ for\ each\ bin\ of\ f$
7: **end if**
8: $Flow[f].I \leftarrow maximum\ of\ local\ I\ from\ bins\ of\ f$
9: **if** $Flow[f].I >= 0$ **then**
10: $\quad RED{-}ENQUE(pkt)\ //pass\ pkt\ to\ the\ RED\ block$
11: $\quad$ **if** $RED\ drops\ pkt$ **then**
12: $\quad\quad T_2 \leftarrow pkt.arrivaltime$
13: $\quad$ **end if**
14: **else**
15: $\quad Flow[f].T_1 \leftarrow pkt.arrivaltime$
16: $\quad drop(pkt)$
17: **end if**
18: **return**

Fig. 3. Pseudo codes of the RRED algorithm.

bins. Each level has an independent hash function. A flow is mapped to a total of L bins, each in one level. Specifically, an L-tuple $(b_1, b_2, ..., b_L)$ where $bj \in [1, ..., N]$ uniquely identifies a flow. Each bin maintains a local indicator. The filter is updated following the procedure below. If a packet from a flow f is suspected to be an attacking packet, all the L bins corresponding to f reduce their local indicators by 1. Similarly, if a packet is a normal packet, all the local indicators are increased by 1 (lines 4 and 6 in Fig. 3). Note that an LDoS flow may share a bin with a normal flow on a particular level. To avoid the situation that a normal flow is polluted by an LDoS flow due to the shared bins, we empirically set an upper bound 10 and lower bound -1 for each bin's local indicator. More specifically, an LDoS flow cannot pollute a normal flow by reducing the local indicator of the shared bin to a low negative number. Finally, the indicator $f.I$ of flow f equals to the maximum value of the L local indicators from the L bins corresponding to f (line 8 in Fig. 3).

IV. Performance Evaluation

In this section, we use NS-2 simulator [10] to conduct a set of simulations to evaluate the performance of proposed RRED algorithm in the presence of LDoS attacks. Several other AQM algorithms including RED [1], RED-PD [2], SFB [3], AVQ [4], and DropTail are used in the comparison. Note that LDoS attacks exploit TCP's retransmission timeout mechanism thus we only consider TCP flows in this letter.

Fig. 5 shows the experimental topology. The queue size of the bottleneck link is 50 packets. AQM algorithms are used on the bottleneck queue, and other queues use DropTail. A TCP ($Newreno$) based FTP flow with packet size of 1000 bytes is generated from each user (User 1 to User 30). LDoS traffic is generated from Attacker 1 to Attacker 20 by sending UDP packets with packet size of 50 bytes.

The parameters of RRED are set as L=2, N=23, and T^*=10ms. Here, L and N of RRED are set as the same values

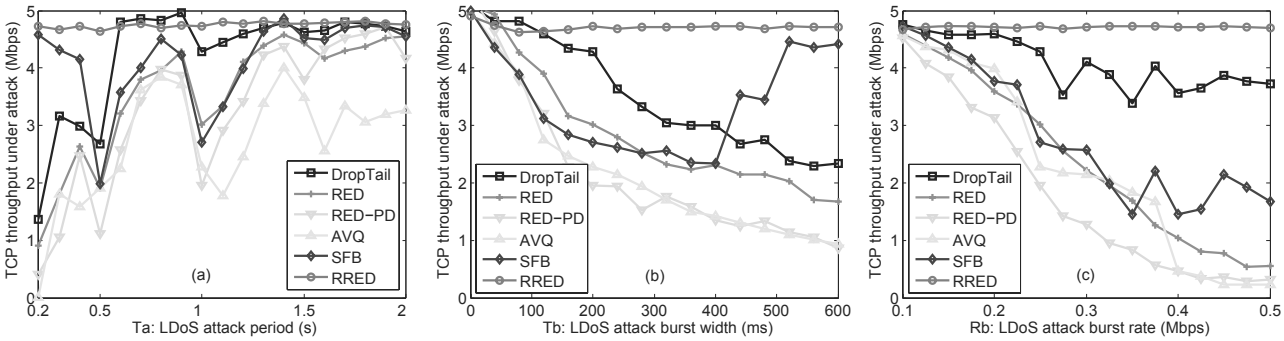

Fig. 4. TCP throughput under attack. (a) Ta =[0.2, 2], Tb=200 and Rb=0.25; (b) Ta =1, Tb=[0, 600] and Rb=0.25, (c) Ta =1, Tb=200 and Rb=[0.1, 0.5].

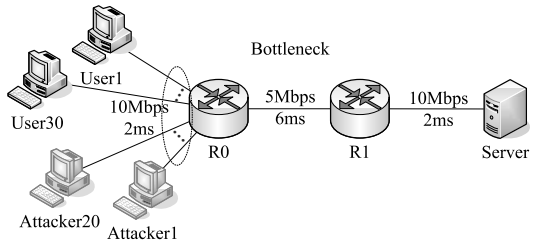

Fig. 5. Experimental topology.

TABLE I
EXPERIMENTAL PARAMETERS

Experiments	T_a**(s)**	T_b**(ms)**	R_b**(Mbps)**
Set one	[0.2, 2]	200	0.25
Set two	1	[0, 600]	0.25
Set three	1	200	[0.1, 0.5]

as SFB [3] for comparison, and T^* is chosen empirically. The RED we employ here is based on packet count rather than packet byte, which is more sensitive to the small packets of LDoS attack flows. The other parameters of the AQM algorithms are all NS-2 default values.

For the three parameters of the LDoS attack, we choose T_a=1s since [6] reported that LDoS attacks with $T_a \approx 1s$ are most effective. T_b is set to 200ms and R_b is set as 0.25Mbps so that the aggregate R_b of 20 attackers is equal to the bottleneck bandwidth of the network (5Mbps). With these three parameters, we conduct three sets of experiments to evaluate and compare the performance of the AQM algorithms. For each set, we fix two values and vary the other value (see Table I). For example, for set one, we vary T_a from 0.2 to 2 while fixing T_b and R_b. Varying these three parameters aims to investigate the robustness of the RRED algorithm if an attacker changes its resending behavior during an attack.

Fig. 4 shows the experimental results corresponding to the three sets of parameters. The results show that the RRED algorithm is highly robust. The TCP throughput, which is close to the link capacity, is nearly fully preserved in diverse LDoS attacks. The results also confirm that the existing RED-like algorithms are notably vulnerable under LDoS attacks due to the oscillating TCP queue size caused by the attacks [7]. Their performance is worse than DropTail in the presence of LDoS attacks. The TCP throughput of RED-like algorithms declines as T_b or R_b increases.

Further experiments show that the false positive of RRED on bursty but non malicious HTTP flows is less than 2% when we fix T_a=1, T_b=200 and R_b=0.25 while varying the average number of new HTTP connections started in each second from 100 to 1000.

Although we focus on TCP flows in this letter, an interesting issue will be to consider the performance of RRED on UDP flows due to the fact that UDP senders do not reduce their transmission rate. The RRED algorithm shown in Fig. 3 declares a flow to be an attacking flow only if major packets in the flow are sent within the short-range after a packet is dropped. Thus we believe that the current RRED will suspect a UDP flow if the UDP flow behaves non-responsively. Further investigation on the fairness issue will be part of our future work.

V. CONCLUSION

We have proposed and implement a Robust RED (RRED) to counter LDoS attacks in this letter. Simulations and analysis show that the RRED algorithm (i) is highly robust, (ii) can significantly improve the performance of TCP under LDoS attacks, (iii) obviously outperforms existing RED-like algorithms against LDoS attacks.

REFERENCES

[1] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Networking*, vol. 1, no. 4, pp. 397–413, 1993.
[2] R. Mahajan, S. Floyd, and D. Wetherall, "Controlling high-bandwidth flows at the congested router," in *IEEE ICNP*, 2001.
[3] F. Wu-Chang, D. D. Kandlur, D. Saha, and K. G. Shin, "Stochastic fair blue: a queue management algorithm for enforcing fairness," in *IEEE INFOCOM*, 2001.
[4] S. S. Kunniyur and R. Srikant, "An adaptive virtual queue (AVQ) algorithm for active queue management," *IEEE/ACM Trans. Networking*, vol. 12, no. 2, pp. 286–299, 2004.
[5] R. K. C. Chang, "Defending against flooding-based distributed denial-of-service attacks: a tutorial," *IEEE Commun. Mag.*, vol. 40, no. 10, pp. 42–51, 2002.
[6] A. Kuzmanovic and E. W. Knightly, "Low-rate TCP-targeted denial of service attacks and counter strategies," *IEEE/ACM Trans. Netw.*, vol. 14, no. 4, pp. 683–696, 2006.
[7] M. Guirguis, A. Bestavros, and I. Matta, "Exploiting the transients of adaptation for RoQ attacks on Internet resources," in *IEEE ICNP*, 2004.
[8] M. Paynter and T. Kocak, "Fully pipelined bloom filter architecture," *IEEE Commun. Lett.*, vol. 12, no. 11, pp. 855–857, 2008.
[9] A. Shevtekar and N. Ansari, "A router-based technique to mitigate reduction of quality (RoQ) attacks," *Computer Networks*, vol. 52, no. 5, pp. 957–970, 2008.
[10] S. McCanne and S. Floyd, "The network simulator - ns-2," 2008.