On the Effect of Router Buffer Sizes on Low-Rate Denial of Service Attacks

Sandeep Sarat Andreas Terzis
sarat@cs.jhu.edu terzis@cs.jhu.edu
Johns Hopkins University

Abstract

Router queues buffer packets during congestion epochs. A recent result by Appenzeller *et al.* showed that the size of FIFO queues can be reduced considerably without sacrificing utilization. While Appenzeller showed that link utilization is not affected, the impact of this reduction on other aspects of queue management such as fairness, is unclear. Recently, a new class of low-rate DoS attacks called *shrews* was shown to throttle TCP connections by causing periodic packet drops. Unfortunately, smaller buffer sizes make shrew attacks more effective and harder to detect since shrews need to overflow a smaller buffer to cause drops.

In this paper, we investigate the relation between buffer size and the shrew sending rate required to cause damage. Using a simple mathematical model, we show that a relatively small increase in the buffer size over the value proposed by Appenzeller is sufficient to render the shrew attack ineffective. Intuitively, bigger buffers require the shrews to transmit at much higher rates to fill the router queue. However, by doing so, shrews are no longer low-rate attacks and can be detected by Active Queue Management (AQM) techniques such as RED-PD. We verified our analysis through simulations showing that a moderate increase in the buffer size, coupled with an AQM mechanism is adequate to achieve high link utilization while protecting TCP flows from shrew attacks.

***Index Terms*—Routers, Buffer Sizing, Active Queue Management, Denial of Service Attacks**

1 Introduction

Internet routers employ queues to buffer packets during periods of congestion. Until recently, the size of buffers for TCP dominated links was determined using the rule of thumb proposed by Villamizar *et al.* in [11]. According to this rule, the size B of a buffer is given by $B = \overline{RTT} \times C$, where $\overline{RTT}$ is the average round-trip time of the flows traversing the link and C is the link capacity. While this rule of thumb was widely accepted, Appenzeller *et al.* recently showed, based on TCP flow de-synchronization dynamics, that queue size can be actually reduced without sacrificing utilization [2]. Given N flows, they show that a buffer of size $B' = \frac{\overline{RTT} \times C}{\sqrt{N}}$, suffices to maintain utilization close to 100% for drop tail queues. Since this result depends primarily on the de-synchronization of TCP flows sharing the same queue, it is believed to extend to other queuing schemes such as RED [4].

RED was the first in a series of Active Queue Management (AQM) schemes that use increases in queue size to detect *incipient congestion* before the queue becomes full. Subsequent extensions to RED, e.g. RED-PD [9] etc., attempt to achieve a fair allocation of resources among potentially selfish or malicious flows sharing the same link. Malicious flows may violate the TCP congestion control algorithm in order to selfishly maximize their throughput or cause *Denial of Service* attacks, thereby minimizing the throughput received by TCP flows sharing the same link. Since the majority of AQM schemes maintain partial flow state for reasons of scalability, larger buffer sizes translate to more accurate per-flow statistics and therefore higher probability of detecting misbehavers. This brings us to the main question we address in this paper: *While buffer size can be reduced without affecting link utilization, does this reduction make the detection of misbehavers harder?* To test the vulnerability of smaller buffer queues to misbehaving sources, we use a recently proposed class of DoS attacks called *shrews* [7]. These malicious flows send short periodic bursts of traffic trying to fill up the buffer and force TCP timeouts, thus throttling the throughput of TCP flows. We chose this type of attack because shrews are difficult to detect due to their low average sending rate.

We use a mathematical model to show that smaller queues are indeed vulnerable to shrew attacks. However, increasing the buffer to $B'' = mB', m << \sqrt{N}$, is sufficient to drive the shrews' average transmitting rate required to cause a DoS attack considerably higher than the min-max fair rate. When this happens, shrews can be detected by an AQM scheme such as RED-PD and consequently penalized without affecting compliant flows. We validate our analysis using simulations in two different scenarios: (a) a 10 Mbps link shared by 20 flows and, (b) a 155 Mbps link shared by 250 flows.

2 The Shrew Attack

We begin with a brief description of the shrew attack. A detailed discussion on shrews can be found in [7]. Consider a bottleneck link shared by a large number of TCP flows. A low-rate shrew DoS attack is a periodic burst of traffic (e.g. square wave pattern) such as the one shown in Fig. 1. The shrew transmits at a high rate of P bps, for a short period of time l sec. For the rest of the time, it transmits

at a much lower rate (almost zero). This behavior repeats with a period of T sec. The average rate of a typical shrew is given by by $P * l/T$. Since the ratio l/T is small, the shrew appears to be a well-behaved over larger timescales, thus evading detection.

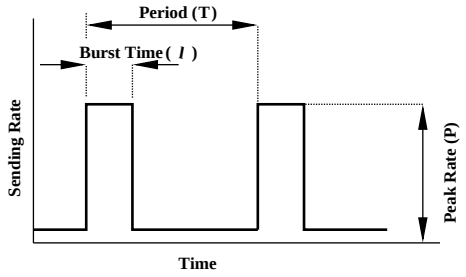

Figure 1. Square-wave shrew

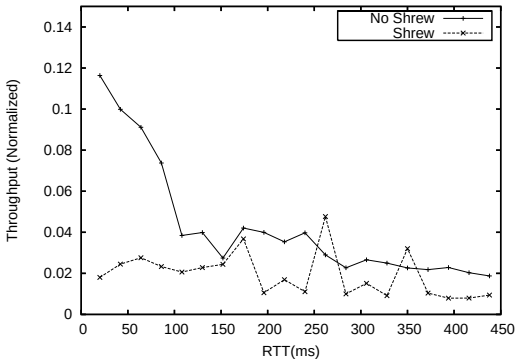

Figure 2. Effect of a single shrew on TCP throughput as a function of the RTT of flows sharing a DropTail queue.

The shrew works by keeping the buffer full for a sufficiently long time (typically in the time scale of the flows' RTT), causing the router to forcefully drop multiple packets from the same TCP flow. At this point, TCP flows will try to retransmit the packets after a retransmission timeout (RTO). By setting its period T to be equal to the TCP flow's minRTO (the authors of [1] suggest that all TCP flows should set their minRTO to 1 second) the shrew causes the retransmitted packets to also be dropped. Subsequently, TCP does an exponential back-off, dropping its congestion window to one and doubling its RTO. Since the new RTO is also a multiple of T, the flow experiences repeated packet losses. Typically, the lower RTT flows are penalized more heavily than the higher RTT ones. As Figure 2 shows (recreated from Figure 7 of [7]) shrews can considerably decrease the throughput of competing TCP flows.

N	The number of flows sharing the link.
$\overline{RTT}$	The average RTT of the flows.
C	The link capacity.
B	The buffer size given by $B = m \cdot \frac{\overline{RTT} \cdot C}{\sqrt{N}}$, $m \geq 1$.
$B_0 = \gamma \cdot B$	The instantaneous queue size when the shrew attack is launched.
s	The number of shrews.
P	The burst rate of a single shrew.
l	The burst time.
T	The period of the shrew.

Table 1. Notation used in the mathematical analysis of the shrew attack.

3 Mathematical Analysis

We present here a simple fluid model used to analyze the effect of increasing buffer size on shrew attacks. In our analysis we assume an idealized AQM scheme that is able to detect and penalize flows sending traffic at rate higher than their fair rate. The notation we use is listed in Table 1.

Consider an attack with s shrews, is launched on a link used by N TCP flows. We further assume that the goal of the shrew attack is to limit all the flows with RTT $< \rho$ sec. Then, the minimum amount of total incoming traffic required to keep the buffer full for ρ sec, is given by:

$$\text{Input traffic} = B - (B_0 - C \cdot \rho) = C \cdot \rho + (B - B_0) \quad (1)$$

As shown in [7], if a shrew attack is launched on a link shared by a large number of TCP flows, and the shrew attack limits all flows with RTT $< \rho$, TCP flows with larger RTT may consume the additional capacity. Furthermore, other background traffic sharing the link such as short TCP and UDP flows, which are unaffected by the shrew, also aid the shrew in filling up link. Consequently, the shrews need to send less traffic than shown in Eq.(1). In the worst case, when the link is completely utilized by background traffic, the shrews must at least account for $(B - B_0)$. Therefore, for a time period ρ:

$$\text{Shrew traffic} \geq B - B_0 = m \cdot (1 - \gamma) \cdot \frac{\overline{RTT} \cdot C}{\sqrt{N}} \quad (2)$$

where γ is the fraction of the buffer that was full at the beginning of the shrew attack. The fraction γ depends on factors such as the queue type and the traffic mix traversing the link. Note that the total traffic sent by the shrews during this time is equal to $(P * l) * s$. Thus, we can rewrite Eq.(2) as:

$$(P \cdot l) \cdot s \geq m \cdot (1-\gamma) \cdot \frac{\overline{RTT} \cdot C}{\sqrt{N}} \quad (3)$$

One can see from Eq.(3) that if we increase the size of the buffer by using a larger constant $m' = m + \Delta m$, the peak rate of each shrew must increase by:

$$\Delta P \geq \Delta m \cdot (1-\gamma) \cdot \frac{\overline{RTT} \cdot C}{\sqrt{N} \cdot l \cdot s} \quad (4)$$

Eq.(4) reveals that with a unit increase in the multiplicative factor m, each individual shrew needs to increase its sending rate by an order of $O(1/\sqrt{N})$. Given that the fair bandwidth of a fbw is $fbw = O(\frac{C}{N})$, a small increase in m, causes the sending rate of each shrew to be higher than fbw, whereby the shrew is no longer a low-rate attack and will therefore be detected by the AQM mechanism. Note that for high speed links, $\Delta m << \sqrt{N}$, and so the buffer size still remains $<< \overline{RTT} \cdot C$. Furthermore, as N increases, the fair bandwidth allotted to each fbw decreases. Consequently, the average sending rate of the shrew is much higher than the fair bandwidth and the shrew is easier to detect.

We use a typical scenario as an illustration. Consider an OC3 link (155 Mbps) carrying 150 TCP fbws, with γ = 0.7, $\overline{RTT}$ = 250 ms, and l = 100ms. In this case the additional buffer space that needs to be filled by the shrews for a unit increase of m, is $\Delta = (1-\gamma) * \frac{\overline{RTT} * C}{\sqrt{N}} \approx 1$ Mb. Therefore, if $s = 5$, each individual shrew needs to increase its peak sending rate by 2 Mbps for a unit increase in m. Consequently, the average sending rate of a shrew increases by $2Mbps * l/T = 2 * 0.1/1 = 0.2$ Mbps. The min-max fair bandwidth for the link is 155/150 $\approx$ 1 Mbps. Thus, choosing m = 5 ($< \sqrt{150} \approx 12.24$) is sufficient to drive the sending rate of a single shrew sufficiently high, whereby the shrew will be detected by an AQM scheme (e.g. RED-PD) which provides approximate fairness.

4 Evaluation

We use ns-2 simulations to verify our mathematical analysis. Figure 3 shows the classic dumb-bell topology we used in our simulations, with two sets of sources and sinks: the first set consists of TCP source/sink pairs while the second set consists of shrews. All TCP fbws are long duration SACK fbws. We used SACK because it was found to be the most resistant version of TCP to the shrew attack [7]. TCP sources start at a random time between [0,10] sec while the shrew attack starts at 100 sec, to allow the TCP fbws to reach steady state.

All the source-sink pairs are inter-connected by the bottleneck link $r_0 \rightarrow r_1$. The links delays and speeds are shown in Figure 3. All the sinks have a one-way delay of 1 msec to router r_1. The one-way propagation delay of the TCP sources to r_0, uniformly increases from 0 to 220 msec. Therefore, the round trip time ranges uniformly from 20 msec to 460 msec as suggested in [5].

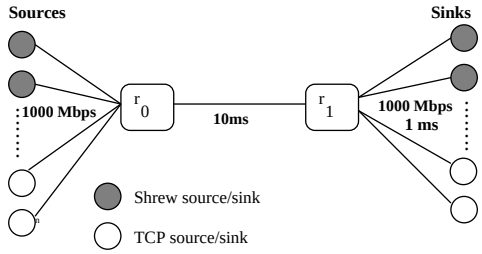

Figure 3. Dumb-bell configuration.

We set the buffer size of the $r_0 \rightarrow r_1$ link to be equal to $B = m * \frac{\overline{RTT} \times C}{\sqrt{N}}$ and we vary m from 1 to $\sqrt{N}$, to measure the effect of increasing buffer size on the throughput of the TCP fbws and the sending rate of the shrew. All the links have drop-tail queues except the $r_0 \rightarrow r_1$ link which uses RED-PD [9]. RED-PD uses a configurable *target round trip time* R to derive the average sending rate of compliant TCP fbws using the deterministic model of TCP from [3]. According to this model, the sending rate of a compliant TCP fbw is $B_R = \frac{\sqrt{1.5}}{R\sqrt{p}}$, where p is the ambient loss rate computed over the recent history of packet losses. Flows whose sending rate is higher than B_R are identified as misbehaving and are monitored. The advantage of increasing R is that more misbehaving fbws can be identified. On the other hand, doing so increases the required amount of per-fbw state which is proportional to the increase in R and the number of fbws traversing the link. In our simulations we use multiple values of R to evaluate the sensitivity of RED-PD in detecting the shrews.

In the following paragraphs, we present the results from two different scenarios, with different link speeds and number of fbws to investigate the effect of increasing buffer size on the throughput of TCP and the transmitting rate of the shrews. All the results are based on at least 400 sec of simulations.

4.1 Low Speed Link

This scenario is similar to the one used in the original shrew attack paper [7]. The capacity of bottleneck link is set to 10 Mbps and link is shared by 20 TCP fbws and a single shrew. The RED-PD threshold R is set to 40 msec. We use the shrew parameters from [7], where P = 10 Mbps, l = 200 msec, and T = 1.2 sec, for an average sending rate of 1.67 Mbps. Given a shrew with parameters p, l, T, we define an equivalent CBR to be a CBR fbw transmitting at

a constant rate of pl/T, equal to the average sending rate of the shrew. We then compare the normalized throughput as a percentage of the link capacity that each TCP fbw achieves when it competes with a shrew to the throughput achieved when the shrew is replaced by the equivalent CBR fbw.

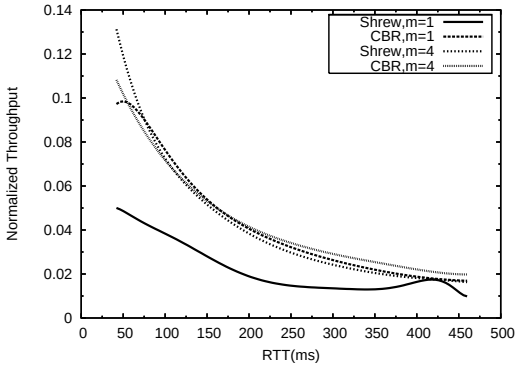

Figure 4. TCP throughput as a function of the RTT under increasing buffer sizes. Unless otherwise specified, $R = 40$ msec.

Figure 4 plots the throughput obtained by the different TCP fbws as a function of their RTT. The first point to be noted in the graph is the well-known negative bias of TCP to fbws with high RTT. The more interesting point however, is the reduced sending rate of TCP fbws across the whole RTT range when the shrew is active and the buffer size is small $(m = 1)$. The low RTT fbws are more adversely affected. However, TCP sending rate increases with m. When $m = 4$, the throughput of the TCP fbws is approximately equal to that, when the shrew is replaced by the equivalent CBR source. This result indicates that the higher buffer size is indeed effective in minimizing the effect of the shrew.

	R=40 msec		R=120msec	
m	CBR	Shrew	CBR	Shrew
1	83%	43%	86%	79%
2	86%	65%	87%	80%
3	86%	78%	88%	81%
4	86%	82%	88%	81%
$\sqrt{20} \approx 4.5$	86%	83%	88%	81%

Table 2. Aggregate link utilization from 20 TCP flows.

Figure 5 shows the effect of a unit increase in m on the throughput of the TCP fbws. When m = 2, the throughput of TCP fbws with low RTT increases. However, there is still some negative effect of the shrews on the higher RTT fbws. Table 2 shows the percentage of link capacity utilized by the 20 TCP fbws for different values of R. The Shrew column corresponds to the throughput obtained by the TCP fbws under a shrew attack while the CBR column shows the throughput for the equivalent CBR fbw. As seen from the table, the TCP throughput is gradually restored with increase in m. When $m = 4$, the negative effect of the shrew on the TCP fbws is minimal [1]. The table also shows that using a larger R value (120msec) requires a smaller increase in the buffer size to mitigate the shrew attack because RED-PD with R=120msec is a better fairness approximator than RED-PD with R=40msec. The reason is that when R = 40msec, RED-PD only detects fbws whose RTT $\leq$ 40 msec. For the same reason, that the results (not shown here) for a simple RED queue are similar to that of RED-PD with R=40msec. Since most of the fbws in this experiment have higher RTT, RED-PD emulates RED for the majority of the fbws. Consequently, when R = 120 msec, the throughput attained by the TCP fbws is closer to the fair throughput allocation (C/N = 10/21 $\approx$ 470Kbps). Of course, setting R equal to the maximum RTT among the TCP fbws would achieve perfect fairness and completely mitigate the shrews, with the least increase in the buffer size. The downside is that since the amount of packet drop information stored by RED-PD is proportional to the number of fbws and R, larger R values result in higher state overhead.

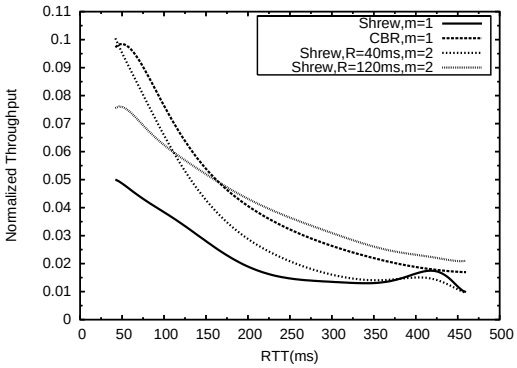

Figure 5. When R increases to 120 msec, it is possible to have small buffer size $(m = 2)$ without penalizing the TCP flows sharing the link with the shrew.

From this first experiment, one may incorrectly suspect that in order for the shrew to be neutralized, $m \approx \sqrt{N}$. This is however an artifact of the small number of TCP fbws (20) sharing the link in this experiment. To show that small values of m are indeed adequate, we repeated the experiment

[1] Utilization of 86% and 88% for TCP when $m = 4.5$, indicates that the competing fbw is having high number of losses and the TCPs are able to utilize the additional capacity

using a higher number of flows on a faster link.

4.2 High Speed Link

Next, we consider a more realistic scenario, where the bottleneck link is an OC-3 (155Mbps) link shared by 250 TCP flows. We use ten synchronized shrews ($\approx$ 4% of the total number of flows). This way any single shrew has a lower average sending rate and is more difficult to detect. For each shrew $P = 20$ Mbps, $l = 200$ msec, and $T = 1.2$ sec, implying an average sending rate of 3.33 Mbps. Therefore, all the synchronized shrews have an aggregate peak rate of 200Mbps for a burst time of 200 ms. Table 3 shows the link utilization due to TCP flows obtained with shrews and with shrews replaced by equivalent CBR flows. As seen in the previous simulation, the utilization steadily increases with m. The corresponding bandwidth plot in Figure 6, illustrates that as m increases to ≈ 3, the sending rate of the low RTT flows is restored to the no-shrews scenario. As in the previous subsection, we repeat the experiment with $R = 120$ msec. As expected, increasing R increases the throughput obtained by the high RTT flows, thereby improving the overall link utilization considerably. In this case, we see that with $R = 120$ msec and $m = 5$, we achieve TCP utilization as good as when $m = 16$. The same effect is evident in Figure 6.

	R=40 msec		R=120msec	
m	CBR	Shrew	CBR	Shrew
1	82%	52%	81%	53%
3	82%	62%	82%	58%
5	83%	68%	83%	82%
8	82%	70%	82%	81%
12	81%	76%	81%	81%
$\sqrt{250} \approx 16$	80%	77%	80%	80%

Table 3. Aggregate TCP link utilization for 250 flows.

Eq.(4) in Section 3 shows that for the shrew attack to be effective, the peak rate P as well as the shrew's average rate has to increase linearly with m. We verified this analysis via simulation. The graphs in Figure 7 plot the shrews normalized sending rate, required to maintain the same low link utilization as when $m = 1$. We see that in the case of 20 flows, when $m = 3$ the average sending rate of the shrew is approximately 50% of the link capacity. For 250 flows, with $m = 5$, each shrew needs to send at about 5% of the link capacity, i.e 50% (since $s = 10$) of the link traffic must be shrew traffic. This implies that the shrew traffic is no longer a low rate traffic but actually a high rate DDoS attack, even with a relatively poor choice of $R = 40$ ms. Unlike shrews, high rate DDoS attacks are easy to detect and several schemes exist to contain them, e.g. [6]. The required shrew sending rate increases more slowly when $m > 5$. The reason for this is as follows: To maintain TCP utilization equal to that when $m = 1$ ($\approx 50\%$), it is enough for the competing shrews to fill up 50% of the link, irrespective of the buffer size. This can be achieved by sending at a rate slightly greater than 50% of the link speed.

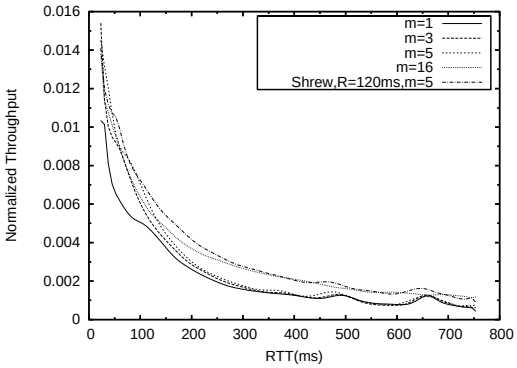

Figure 6. TCP throughput under increasing buffer sizes.

As seen from the last two subsections, a small increase in m is sufficient to improve the throughput of the high RTT TCP flows to almost the levels of no-DoS attack. However, there is still some effect of the shrew attack seen in terms of the reduced utilization of the link capacity. As described in [9], setting the target bandwidth R of the RED-PD router to be very large ($\geq maxRTT$) guarantees perfect fairness, but the state to be maintained is of $O(N*R)$, where N is the number of flows and hence scalability is an issue. Based on the mathematical analysis and simulations setting a moderate target bandwidth of $R(\approx 120ms)$ and increasing m by a small value (≈ 5) provides comparable performance with the no-shrew scenario, both in terms of utilization and fairness. We believe that the effect of shrew attacks can mitigated by using this double pronged strategy. In order for DoS attacks to throttle TCP traffic, shrews need to send at a considerably high proportion of the link capacity, whence the attack is no longer a low rate attack and easier to detect.

5 Related Work

There has a been a plethora of AQM schemes inspired by RED, the seminal work in this area [4]. Quite a few of the schemes aim to provide approximate fairness. However, fairness comes at the cost of maintaining additional state at the router. To minimize state overhead, AQM schemes use only partial flow state. Hence, it is possible for false negatives to occur and malicious flows can exploit these

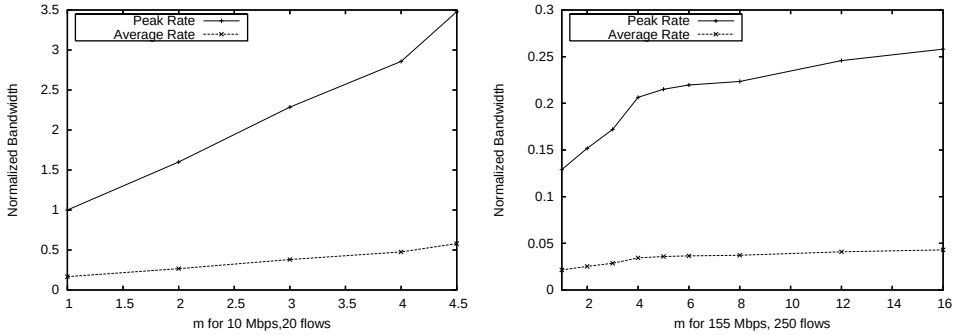

Figure 7. Peak and Average Shrew sending rate needed to maintain reduced link utilization.

weaknesses to gain undue resource advantages. An example of such malicious flows, is the *shrew attack*, a low-rate TCP targeted denial of service attack [7]. Various countermeasures to the shrew attack have been proposed. The notable being: **(a)** RTO randomization; however [7] argues that shrews can still filter out portions of TCP traffic. **(b)** Router level DDoS solutions such as IP traceback [10] and Pushback [6], and **(c)** AQM modifications such as HAWK [8], which makes the policy decision to penalize all bursty flows. However, not all bursty traffic is malicious. For example, short TCP flows would be unduly penalized by HAWK due to their bursty nature. In this paper rather than proposing a new AQM scheme we show that a moderate increase in buffer size, coupled with the use of RED-PD is sufficient to minimize the impact of shrew attacks on TCP traffic.

6 Summary

This paper studies the effect of buffer sizes on the power of low rate TCP targeted DoS attacks to "shut-down" competing TCP traffic. Using a simple mathematical analysis coupled with simulations we showed that a relatively small increase in the buffer size can mitigate the effect of shrew attacks on TCP traffic. The intuition behind our result is simple: As the buffer size increases, shrews need to fill a larger buffer to cause multiple TCP packet drops. This means that the shrews need to transmit at high speeds, at which point they are no longer low-rate attacks and can be detected by existing AQM schemes such as RED-PD.

References

[1] M. Allman and V. Paxson. On estimating end-to-end network path properties. In *Proceedings of ACM SIGCOMM*, Aug. 1999.

[2] G. Appenzeller, I. Keslassy, and N. McKeown. Sizing router buffers. In *Proceedings of ACM SIGCOMM*, Aug. 2004.

[3] S. Floyd and K. Fall. Promoting the Use of End-to-End Congestion Control in the Internet. *IEEE/ACM Transactions on Networking*, 7(4):458473, Aug. 1999.

[4] S. Floyd and V. Jacobson. Random Early Detection gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, 1993.

[5] S. Floyd and E. Kohler. Internet research needs better models. In *Proceedings of HotNets-I*, Oct. 2002.

[6] J. Ioannidis and S. M. Bellovin. Implementing pushback: Router-based defense against DDoS attacks. In *Proceedings of NDSS*, February 2002.

[7] A. Kuzmanovic and E. Knightly. Low-rate TCP-targeted denial of service attacks (the shrew vs. the mice and elephants. In *Proceedings of ACM SIGCOMM*, Aug. 2003.

[8] Y.-K. Kwok, R. Tripathi, Y. Chen, and K. Hwang. HAWK: Halting Anomalies with Weighted Choking to Rescue Well-Behaved TCP Sessions from Shrew DoS Attacks. Technical Report 2005-5, USC Internet and Grid Computing Lab, Feb. 2005.

[9] R. Mahajan, S. Floyd, and D. Wetherall. Controlling high-bandwidth flows at the congested router. In *ICNP*, Nov. 2001.

[10] S. Savage, D. Wetherall, A. R. Karlin, and T. Anderson. Practical network support for IP traceback. In *Proceedings of ACM SIGCOMM*, pages 295–306, 2000.

[11] C. Villamizar and C. Song. High performance TCP in ANSNET. *SIGCOMM Computer Communications Review*, 24(5):45–60, 1994.