Emulation versus Simulation: A Case Study of TCP-Targeted Denial of Service Attacks

Roman Chertov, Sonia Fahmy, Ness B. Shroff
Purdue University

***Abstract*—In this paper, we investigate the applicability of *simulation* and *emulation* for denial of service (DoS) attack experimentation. As a case study, we consider low-rate TCP-targeted DoS attacks. We design constructs and tools for emulation testbeds to achieve a level of control comparable to simulation tools. Through a careful sensitivity analysis, we expose difficulties in obtaining meaningful measurements from the DETER and Emulab testbeds with default system settings, and find dramatic differences between simulation and emulation results for DoS experiments. Our results also reveal that software routers such as Click provide a flexible experimental platform, but require understanding and manipulation of the underlying network device drivers. We compare simulation and testbed results to a simple analytical model for predicting the average size of the congestion window of a TCP flow under a low-rate TCP-targeted attack, as a function of the DoS attack frequency. We find that the analytical model and ns-2 simulations closely match in typical scenarios. Our results also illustrate that TCP-targeted attacks can be effective even when the attack frequency is not tuned to the retransmission timeout. The router type, router buffer size, attack pulse length, attack packet size, and attacker location have a significant impact on the effectiveness and stealthiness of the attack.**

***Keywords*—simulation, emulation, testbeds, TCP, congestion control, denial of service attacks, low-rate TCP-targeted attacks**

I. Introduction

Denial of Service (DoS) attacks have become increasingly prevalent [27], [29]. Significant damage, such as network partitioning, can potentially be caused by attacks that target the Internet infrastructure, such as inter-domain routing protocols, key backbones, and Domain Name System (DNS) servers (which were attacked in 2002). In this work, we address the question of *when simulation* and *emulation* are inadequate for studying DoS attacks. A key component of the answer to this question is the *sensitivity* of simulation and emulation results to parameter settings and testbed capabilities. As a case study, we take an in-depth look at *low-rate TCP-targeted attacks* [16], [22]. In particular, we consider a scenario where an attacker transmits short pulses at an arbitrary frequency. This attack exploits the TCP Additive Increase Multiplicative Decrease (AIMD) mechanism to cause TCP performance degradation. We use a simple analytical model for predicting the average size of the congestion window of a TCP flow under attack, as a function of the attack frequency. The model serves as a lower bound (for the case with no timeouts) when each pulse causes loss. The results from the analytical model are compared to the results from the ns-2 simulator, and the DETER and Emulab emulation testbeds.

TCP-targeted attacks are an interesting case study because they are a major cause for concern (they are easy to launch, stealthy, and may be extremely damaging), and produce traffic oscillations. In a deterministic simulation environment, such oscillations can lead to phase synchronization effects [15] with slight parameter changes. Such effects might appear interesting, but they are not representative of real systems. Additionally, experiments with this attack on testbeds may yield large variations in the results depending on parameter settings and testbed capabilities.

A number of recent studies [16], [24], [36], [10] have studied defenses against TCP-targeted attacks. Among these studies, only [16], [24] conducted testbed experiments, but these experiments were conducted with (*tc*, *iproute2*), NIST-net, or DummyNet [35] for link shaping and queue management, without investigating system parameters, or relating the results to simulations. In contrast, our work investigates emulation environments via a more careful sensitivity analysis, and highlights the danger of default system parameter settings. To the best of our knowledge, this is the first work to quantify and explain key differences between simulation and emulation results for DoS experiments.

The contribution of this work is twofold: (1) We compare results from simulation and emulation experiments with different attack *and* system parameters. We design constructs for the emulation testbeds to achieve a level of control comparable to simulation tools, and configure PC routers via *Click* for high fidelity emulation. We explore key problems that arise due to differences in testbed capabilities and default parameter settings. (2) We use a simple analytical model of TCP performance degradation, in the presence of a special case of TCP-targeted DoS attacks, as a lower bound in our comparisons. We find that the analytical and simulation results closely match for a set of values of the attack pulse length and the router buffer size. Testbed results *only* match the analysis and simulations when routing nodes are overloaded. This overload occurs when the received flow rates exceed the node forwarding capacity. Our results also illustrate that TCP-targeted attacks can still be effective even when the attack frequency is not precisely tuned to the retransmission timeout, as suggested in [16], [22]. The router type, router buffer size, attack pulse length, and attacker location have a significant impact on both the effectiveness *and* stealthiness of the attack. Finally, we find that configurable software routers such as Click provide a flexible experimental platform, but require careful manipulation of the underlying network device drivers.

– This research has been sponsored in part by NSF/DHS grant 0335247, NSF grants 0523249 and 0238294 (CAREER), and USC-ISI. The authors would like to thank Mark Allman (ICSI) and Paul Barford (U. of Wisconsin) for their helpful comments on earlier versions of this work.

– Roman Chertov and Sonia Fahmy are with the Department of Computer Science, 250 N. University St., West Lafayette, IN 47907–2066, USA. Tel: +1-765-494-6183. Fax: +1-765-494-0739, E-mail: {rchertov,fahmy}@purdue.edu. Ness B. Shroff is with the School of Electrical and Computer Engineering, 465 Northwestern Ave., West Lafayette, IN 47907–2035, USA. E-mail: shroff@ecn.purdue.edu

The remainder of this paper is organized as follows. Section II surveys related work on TCP-targeted attacks. Section III discusses a simple analytical model of the performance degradation caused by the attacks. Section IV describes the emulation environment we use, and tools we developed for the DETER emulation testbed. Section V describes our experimental setup. Sections VI and VII discuss our results from DETER, Emulab, and ns-2 simulations. Section VIII describes our experiences with the Click router. Finally, Section IX summarizes our findings and discusses defenses against such attacks.

II. TCP-Targeted Attacks

Most well-publicized DoS attacks have utilized a large number of compromised nodes to create constant high-rate flows towards the victims. Such "flooding attacks" are effective, but have major shortcomings from the attacker's perspective. First, the attacks are easy to detect due to the high volume of uniform traffic, e.g., UDP or ICMP. Several defense mechanisms against these (and more sophisticated) DoS attacks have been proposed in the literature [33], [39], [18], [8], [20]. Second, the attacks can self-congest at some bottleneck and not reach the intended destination. Finally, users of the compromised machines typically notice a performance degradation, prompting these machines to be examined by system administrators, who can then eliminate the vulnerabilities that caused the machines to be compromised in the first place.

An attack that is less susceptible to these limitations is the low-rate TCP-targeted attack, introduced in [22][1]. This attack has generated significant interest due to its potential to do great harm, go undetected, and the ease by which it can be generated. The basic idea of low-rate TCP-targeted attacks [22] is that an attacker transmits short pulses, i.e., square waves, with periodicity close to the Retransmission-Timeout (RTO) interval [19] of ongoing TCP connections. These short pulses induce sufficient packet loss to force the TCP flows under attack to time out, and to continually incur loss as they attempt to begin TCP slow start. Therefore, the goodput of these TCP flows virtually goes to zero. Such an attack can be used to strategically target key routers or servers in the network, thus causing wide-spread degradation of TCP performance.

A key feature of this attack is that it is *stealthy*, i.e., it does not continuously generate significant traffic, and thus cannot be easily distinguished from other legitimate flows (e.g., video or other bursty traffic). Moreover, an attacker does not have to be highly sophisticated to generate these attacks. It is straightforward to generate UDP pulses, or use raw sockets to bypass the TCP congestion avoidance mechanism altogether.

A recent study [16] has considered a more general class of low-rate TCP attacks, referred to as the Reduction of Quality (RoQ) class of attacks. In a RoQ (pronounced "rock") attack, the attacker sends pulses at arbitrary frequencies, rather than trying to precisely match the RTO periodicity. The attack exploits the TCP Additive Increase Multiplicative Decrease (AIMD) mechanism to cause TCP goodput degradation, rather than focusing on timeouts. The premise is that during the *congestion avoidance phase*, when packet losses occur due to attack pulses, TCP halves its congestion window, but when a successful transmission occurs, it only linearly increases its window size. The motivation behind RoQ attacks is that they *need not* be precisely tuned to the RTO frequency, since RTO may be difficult to ascertain, and can be changed for different TCP sessions. While RoQ attacks may not cause TCP goodput to virtually go to zero, as in the case of [22], they can still significantly degrade the service quality. Moreover, these attacks may be even more difficult to detect, since they do not operate at a known frequency. Therefore, we use these attacks as a case study in our work. As previously discussed, these attacks have not been experimentally studied in prior work, except in extremely limited settings, with no sensitivity analysis, or comparisons to analytical or simulation results.

[1] We do not consider other types of application-specific, protocol-specific, or implementation-specific DoS attacks, such as SYN attacks, BGP attacks, LAND, or TEARDROP, in this work. We only focus on attacks against TCP congestion control.

III. Simple Analytical Model

In this section, we describe a simple analytical model, which is a special case of a model in [24]. The model characterizes TCP performance degradation as a function of the TCP-targeted attack frequency. In prior work, e.g., [32], models of TCP throughput as a function of the round-trip time and loss event rate were developed. These models, however, do not consider the presence of periodic attacks. In contrast, we compute the average TCP window size as a function of the TCP-targeted attack parameters. The analysis assumes that TCP Reno [7] in the congestion avoidance phase is being employed for a single flow under attack.[2] As discussed in Section II, the objective of this attack is to exploit the TCP AIMD mechanism and not to cause RTOs. Since Reno can typically recover from a single packet loss without an RTO, it is assumed that *every* attack pulse will induce a packet loss. A loss of a single data packet will cause a reduction of the congestion window by half in TCP Reno, after which additive increase will be employed. For simplicity of the analysis, the short fast recovery phase is ignored. The resulting TCP congestion window saw-tooth pattern is depicted in Figure 1 for a fixed attack frequency. Observe that the model also gives a close approximation of the behavior of TCP New Reno [14] or TCP SACK [26] *even* with a few packet losses with every pulse, since these TCP flavors can typically recover from multiple packet losses without RTOs.

Let W_i be the size of the congestion window (Cwnd) right before the reduction caused by pulse i, $i \geq 0$. Let rtt be the flow round trip time (RTT). Let α be the growth in Cwnd size during the attack *sleep time* t between pulses i and $i+1$. Then, $W_{i+1} = \frac{W_i}{2} + \alpha$, where α (the growth of the window during t) is equal to $\frac{t}{2rtt}$ (assuming that every *other* packet is acked; $\frac{t}{rtt}$ if every packet is acked).

Let W_I be the initial Cwnd size before the attack starts. We need to compute W_{max}, the maximum congestion window size

[2] We have generalized our model to multiple flows under attack, but, for simplicity of illustration, present only the single-flow case here.

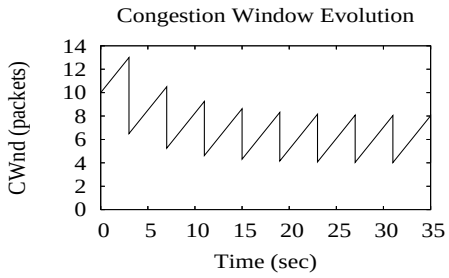


Fig. 1. Saw-tooth pattern of congestion window evolution due to periodic loss every 4 seconds.

after the attack reaches steady state, as well as the average window size W_{avg}. From the above equation, one can easily compute W_1, W_2, $\cdots$. For example, W_3 can be expressed as:

$$W_3 = \frac{\frac{\frac{W_I}{2}+\alpha}{2}+\alpha}{2}+\alpha.$$

Therefore, W_{max} (assuming the limit on the receiver window size is not reached) can be expressed as:

$$W_{max} = \lim_{i\to\infty}(2^{-i}W_I + \alpha(\sum_{j=0}^{i-1}2^{-j})) = 2\alpha.$$

The steady state minimum window size is simply $W_{max}/2 = \alpha$. Since $\alpha = \frac{t}{2rtt}$, therefore, $W_{avg} = \frac{\alpha+2\alpha}{2} = \frac{3t}{4rtt}$.

IV. Emulation Environment and Tools

In order to experiment with this DoS attack in a high fidelity – but quarantined – setting, we leverage the DETER (www.isi.deterlab.net) and Emulab (www.emulab.net) testbeds. Emulab is a time- and space-shared network emulator located at the University of Utah [38]. The system is comprised of hundreds of linked PCs that can be connected in any specified topology, and a suite of software tools that manage them. The Cyber Defense Technology Experimental Research Network (DETER) is an emulation testbed – based on Emulab – that allows researchers to evaluate Internet security technologies [13]. DETER can be accessed remotely, but is quarantined from the Internet. The Evaluation Methods for Internet Security Technology (EMIST) project, in which we are participating, is a companion project that designs testing methodologies and benchmarks for the DETER testbed.

The primary advantage of using a network *emulator* – as opposed to a *simulator* – for security experiments is that an emulation environment affords higher fidelity, and real security appliances can be tested on it. This can expose unforeseen implementation vulnerabilities, protocol interactions, and resource constraints. This is because an emulation testbed uses real computers with limited resources, and real applications and operating systems running on them, to faithfully represent every host in an experiment. Flaws and vulnerabilities are not abstracted by a simplified simulation model. One exception to this rule are router nodes. In the current versions of Emulab and DETER, routers are represented by regular PCs that act as forwarding gateways. We refer to these as *PC routers*. Our experiences with the Wisconsin Advanced Internet Laboratory (WAIL) at www.schooner.wail.wisc.edu/ have demonstrated that a regular commodity PC running Linux with Fast Ethernet or Gigabit Ethernet cards can outperform a Cisco 3600 series router. Specialty PC routers like the ones created by ImageStream [2] are usually created from high end PCs that have multiple PCI buses, SMP, fast memory, and industrial network cards [4], [5]. The performance of specially configured PC routers can easily challenge that of Cisco 7000s/7500s as well as Juniper M-5/M-10 routers according to [2], [3]. Therefore, using PC routers is not unrealistic.

Developing efficient PC routers has been the subject of significant research, e.g., [12], [28], [21]. In these studies, polling and/or DMA are used as an alternative to packet receive interrupts to eliminate receive livelock at high packet rates. This is because interrupts can consume much of the CPU and bus capacity of mid-range machines (i.e., Pentium III and below) at 100 Mbps+ speeds. In [21], programmed I/O (PIO) interaction with the Ethernet controllers is eliminated using Direct Memory Access (DMA). We have found that livelock can occur on Cisco 3600 series routers, limiting the forwarding performance to 10-15 Kpackets/sec per interface.

We first experiment with PC routers without using such systems, since not all device drivers support polling, and hence understanding their behavior is important. Section VIII describes our experiences with a polling-based PC router [21].

Event Control System. In network simulators such as ns-2 [37] and iSSF/iSSFNet [34], it is easy to create a topology, assign tasks to the nodes, and monitor every single packet. A basic testbed – without any software support that mirrors some of these capabilities – is limited in its usefulness, since it requires the experimenters to be experts in system-level programming. Achieving the same level of control provided by a simulator on physical testbed machines is a significant undertaking. Basic topology creation capabilities are provided by emulation testbeds, such as Emulab and DETER, but an experimenter only acquires bare machines that form the desired topology, without any tools running on them.

A natural approach to describe the tasks that must be performed on the testbed nodes is to use event scripts, much like events in an event-driven simulator. The Emulab software implements a few event types such as link failures; however, most of the interaction with the nodes must be performed via a secure shell (SSH) session. We have designed a flexible mechanism to control all test machines from a central location, since manually using each computer is impossible, especially when timed events are involved. We have developed a multi-threaded utility, which we refer to as a *Scriptable Event System*, to parse the script of timed events and execute it on the test machines (communicating with them on the *control network*). Our utility is capable of receiving *callbacks* for event synchronization.[3]

Measurement Tools. Instrumentation and measurement on a

[3]This software can be freely downloaded from http://www.cs.purdue.edu/∼fahmy/software/emist/

testbed pose a significant challenge. The capability to log and correlate different types of activities and events in the test network is essential. Not only are packet traces important, but also system statistics must be measured for DoS attacks. We have developed a set of tools to log events on the test nodes on a per second basis. Statistics such as CPU utilization, packets per second, and memory utilization are logged to the local disk for later manipulation. Scripts for measuring, merging, and plotting system data are also available for download.

V. Experimental Setup

The topology used for both simulation and emulation experiments is depicted in Figure 2. This is a simple dumb-bell topology with four end systems and two routers that connect via a link with 60 ms delay. The attacker and the attack sink are varied from one side of the topology to another.[4] *The same* basic ns-2 script is used for both simulations and testbed experiments. All testbed nodes run the zombie process that forms the basis of our *Scriptable Event System*.

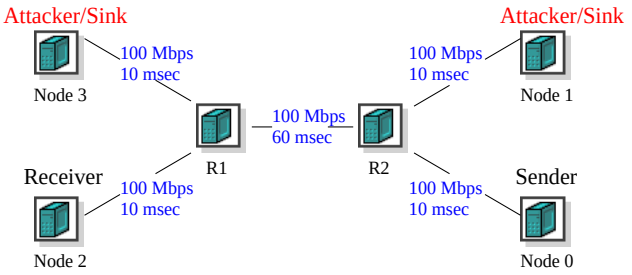


Fig. 2. Simple dumb-bell topology with 160 ms round-trip-time and 100 Mbps links.

On the DETER testbed, all nodes use a Pentium III 733 MHz processor and have 1024 MB of RAM and Intel Pro/1000 cards. On Emulab, end systems are 600 MHz, while nodes representing R1 and R2 are 850 MHz. The nodes have Intel EtherExpress Pro 100 cards in different configurations. In our first set of experiments (Section VII), all nodes run Linux 2.4.20 with IRQ-driven packet processing, thus being susceptible to receive livelock [28]. We use TCP SACK [26] with delayed acknowledgments in ns-2, and on the testbed machines, for the flow under attack. The ns-2 TCP receiver window size is set to 70 packets, and (drop tail) buffer sizes for routers are set to 50 packets (except in experiments when we vary the queue size). The buffer size of 50 was chosen because it is the default value on DETER/Emulab and, according to DummyNet documentation, it is a typical queue size for Ethernet devices. We found that by default the receive and transmit device driver buffers on DETER can hold 256 descriptors (64 on Emulab), according to *ethtool*.

Link delays on the DETER/Emulab testbeds are emulated by DummyNet, meaning that there is a hidden "delay" node on every link. Special care was taken to manually select and configure the delay nodes to be at least as fast as the rest of the test nodes, so that no significant packet loss occurs at the delay nodes. To accomplish this task, we have selected Dual Pentium 4 Xenon 2.8 GHz machines with PCI-X, running SMP and *polling-enabled* FreeBSD-4.10 to act as delay nodes on DETER, and 850 MHz uniprocessor nodes on Emulab. Although shaping methods such as DummyNet or *tc* have been found to induce artifacts (due to their bursty behavior and their not being always true to the desired properties) [1], [6], our current experiments use a single TCP flow, and hence the bursty behavior of DummyNet is not significant.

A. Attack Parameters

On DETER and Emulab, the attack packet sizes we use (as well as all header fields and transmission rates) can be easily configured in our event script. Since most queues are composed of *Maximum Transmission Unit (MTU)-sized slots*, we use *small* UDP packets for our DoS attacks. Attacks with smaller packets (and hence higher packet rates) may be more damaging since each packet requires a certain amount of header processing at routers and packet processing at end systems. (Analysis of the publicly available DoS agents by David Dittrich [11] shows that Stacheldraht and TFN by default use 1024 and 789 byte packets respectively.) Although the attack tool is easily able to use IP source and destination address spoofing, we do not employ spoofing in our experiments, in order to avoid the additional Address Resolution Protocol (ARP) overhead, and avoid backscatter [29] overhead on PC routers.

In our ns-2 experiments, the attack is created by a CBR agent that pulses at regular intervals, transmitting 38-byte packets at full link capacity. To achieve the same behavior on DETER nodes running Linux, we use raw sockets for packet transmission at link rate. System timers were used to measure the duration of the attack pulse, and the sleep time when no packets are sent by the attacker. Using real clock timers is crucial when sub-second granularity is required. However, the attack pulse on Linux is less precise than its ns-2 counterpart, since CPU scheduling, etc, affect the pulse precision. This produces small variations in the experimental data during identical runs on DETER and Emulab. Therefore, we repeat each experiment ten times and average the results. We also compute the variance which we find to be low.

B. Traffic Generation and Measurement Tools

To gauge the impact of the attack on DETER and Emulab, we use *ttcp* [30] to create a single long-lived TCP flow by transferring a large file. We chose to have a single "good" flow as this creates the worst case for the attacker, since the attacker has to send enough traffic to cause loss. A long-lived TCP flow is not unrealistic, as there are several sites that offer large downloads, e.g., DVD-quality movies. In addition, this scenario simplifies the comparison between emulation and simulation results. The *ttcp* tool we use reports statistics such as total time, transfer rate, and other system-specific information. In our case, the transfer rate is a good indicator of the attack potency. Traces collected by *tcpdump* are processed by *tcptrace* to produce an estimated weighted average of the congestion window. To validate the estimated weighted average of the congestion window derived from

[4] This simple topology is not representative of the Internet, but we have selected it in order to be able to analyze the results in depth. Our future work plans include experiments with multiple bottleneck confi gurations and other traffi c patterns.

tcptrace, we have developed a simple tool that records the window for a specific connection from */proc/net/tcp* several times per second. We found that the average of these polled values is close to the value derived from the *tcptrace* output. Therefore, we are confident in the window size results we report.

On DETER and Emulab, we first start the measurement tools as well as *tcpdump*. Then, we trigger the attack agent and, later, the file transfer. The sending node, $Node0$, is instructed to transfer a 30 MB file to $Node2$ via *ttcp*. Upon successful completion of the task, the attacker ceases the attack, and the measurement and tcpdump files are transferred to the project account. The ns-2 simulations and the DETER/Emulab experiments use *the same* basic *tcl* script, and we log the attributes of the ns-2 TCP agent directly. The simulation time is 500 seconds to ensure that steady state has been reached. To conduct accurate transfer rate comparisons between ns-2 and DETER/Emulab, we set the ns-2 TCP packet size to 1447 bytes, as 1447 is the average size reported by *tcptrace* when it analyzes DETER/Emulab tcpdump files.

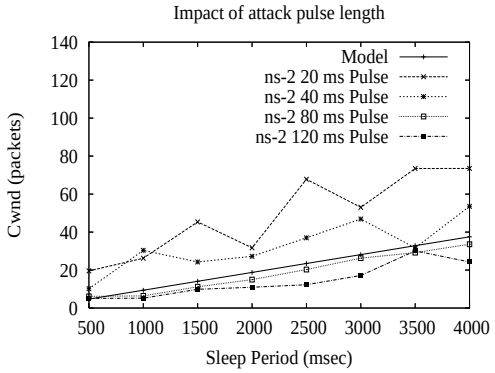


Fig. 3. ns-2 simulation results with varying attack pulse lengths. The RTT is 80 ms in this experiment. Attack packet payload size is 10 bytes.

C. Experimental Design

In our experiments, we investigate the impact of varying the following parameters:

(i) The attack sleep time from 500 ms to 4000 ms in increments of 500 ms;
(ii) The attack pulse length to be 20, 40, 60, 80, 120, 160, or 200 ms;
(iii) The attack packet payload size to be 2, 10, 80, 100, or 700 bytes;
(iv) The round-trip time of the attack flow and the long-lived TCP flow to be 60, 80, 120, 160, or 200 ms;
(v) Router buffer sizes on ns-2 routers to be 25, 50, 100, or 150 packets;
(vi) Router buffers sizes on Click to be 25, 50, or 256 packets;
(vii) The transmission ring buffer size of the *e1000* driver to be 80 or 256 packets; and
(viii) The placement of the attacker to be either $Node1$ or $Node3$.

We compute the following metrics:

(i) Average goodput in KBps (KBytes/sec), computed by dividing the transfer size by the transfer completion time;
(ii) Average congestion window size in packets, computed by dividing the weighted congestion window average by the average MSS reported by *tcptrace*;
(iii) CPU percentage utilization from */proc/uptime*; and
(iv) Packets per second received and sent on the *test network* interfaces from */proc/net/dev*.
System-level measurements, e.g., CPU utilization, cannot be collected in simulations, since ns-2 does not model host CPUs, etc.

VI. Simulation Results

In this section, we compare results from the analytical model in Section III and the ns-2 simulator. We first investigate the impact of varying the length of the attack pulse on the average congestion window size. The length of the attack pulse controls the tradeoff between attack damage and attack stealthiness. Further, it determines the probability of packet loss(es). Recall that the analytical model had assumed that *every attack pulse results in cutting the window by half*. By varying the length of the attack pulse, we study the conditions under which the analytical results match the simulations. The RTT is set to 80 ms in this experiment (R1 to R2 link delay in Figure 2 is changed to 20 ms), while the length of the attack pulse is set to 20, 40, 80, or 120 ms.

From Figure 3, it is clear that longer attack pulses reduce TCP performance (at the cost of reduced attack stealthiness). Extremely short pulses, such as 20 ms pulses, are ineffective because the TCP sender sends packets in bursts. If the attack pulse does not overlap with the burst of legitimate data, no packets are lost. In this case, the legitimate flow Cwnd size can continue to increase with no cuts, and the frequency of Cwnd cuts is lower than the attack frequency. Increasing the length of the attack pulse increases the probability of overlap of the two bursts. We find that attack pulses that are of length greater than or equal to the RTT of the legitimate flow result in packet drop(s) and window cut(s) with virtually every attack pulse.

Another interesting observation from Figure 3 is the non-monotonic increase of the average congestion window for ns-2 with the increase of the sleep time. This can be explained as follows. In ns-2, the lack of overlap between sender and attacker traffic can lead to fewer Cwnd cuts than expected for certain values of sleep time, thus causing the average window to be higher (as discussed above). However, for other values of sleep time, synchronization of the sender and attacker or RTOs can result in a smaller average Cwnd value. Since the ns-2 simulator components in this experiment are deterministic, such synchronization effects are amplified, which is consistent with the observations in [15]. Adding randomization to the attack traffic is likely to reduce or eliminate such effects.

The figure also shows that there is a close match between the analytical model and simulation results in the experiments where the length of the attack pulse is close to or longer than the RTT. This is because pulses longer than or equal to the RTT guarantee that each pulse causes a window cut(s). This observation is confirmed by Figure 4, where we fix the attack pulse length to

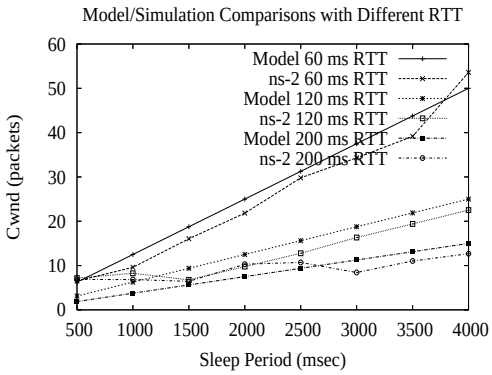


Fig. 4. ns-2 simulation results with varying round trip times (60, 120, or 200 ms). The attack pulse length is set to be equal to the round trip time. The attack packet payload size is 10 bytes.

be equal to the RTT, for different values of RTT. The matches between the analytical model and simulations hold for different RTT values.

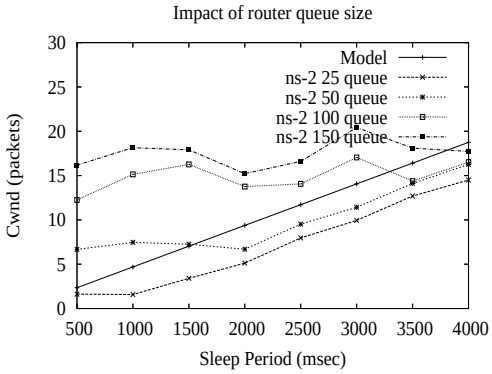


Fig. 5. ns-2 simulation results with varying router buffer sizes (25, 50, 100, or 150 packets). The attack pulse length is set to be equal to the round trip time (160 ms). Attack packet payload size is 10 bytes.

Figure 5 illustrates the impact of varying the sizes of router buffers (queues) on the average congestion window size. For a queue size of 100 or 150 packets, an attack pulse of length equal to the RTT is inadequate to cause goodput degradation for a single TCP flow under attack. The attack is effective with smaller queue sizes, such as 25 and 50 packets. With a longer attack pulse length or more flows under attack, the attack is likely to be effective, even when the router buffer sizes are larger.

VII. DETER/EMULAB TESTBED RESULTS

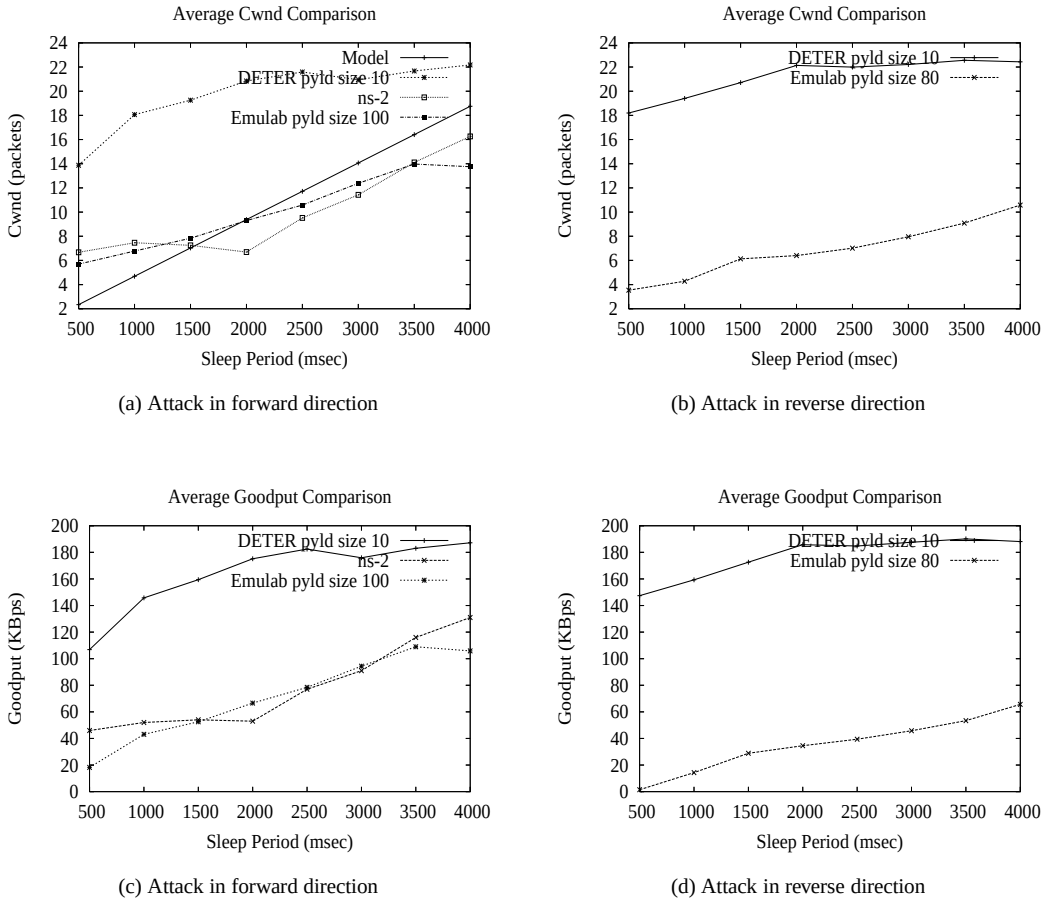
In this section, we undertake the more challenging task of comparing results from the analytical model (Section III) and ns-2 simulator to the DETER and Emulab testbeds. We first use the default system settings for the DETER/Emulab nodes, since the ns-2 configuration was derived from these values. In this set of experiments, the attack packet payload size is 10 bytes on DETER and ns-2, but it is set to 100 bytes on Emulab, as smaller packet sizes on Emulab cause the experiments to take several days, which is problematic in a shared and heavily used testbed like Emulab. The RTT is set to 160 ms, as depicted in Figure 2. Figure 6 gives the average congestion window (Figures 6(a) and 6(b)) and average goodput (Figures 6(c) and 6(d)), for an attacker at node $Node1$ (forward direction) or node $Node3$ (reverse direction). From Figures 6(a) and 6(c), we observe key discrepancies between the DETER results and the analytical, simulation, and Emulab results. The analytical, simulation, and Emulab results are similar, except for small sleep time periods, when ns-2 windows and goodput exceed the analytical results for the same reasons discussed in Section VI (the attack pulse does not overlap with the short burst of data traffic, and thus not every pulse causes a window cut).

From Figures 6(a) and 6(c), we find that for all values of sleep time, DETER results are *not* affected by the attack as much as ns-2 results. This is because the DETER PC router nodes are able to handle the attack pulse and the single TCP flow under attack. The DETER results are comparable to ns-2 results with a router buffer size of 100 packets (Figure 5). For larger values of sleep time, the DETER curve levels off instead of increasing as with ns-2. This is because the goodput in these cases starts approaching the goodput value when no attack is present (203 KBps) for an RTT of 160 ms. This goodput value corresponds to a receiver window size of 34715 bytes (24 segments), which is the value reported by the receiver in our experiments. This receiver window size, set by the *ttcp* application, limits the maximum goodput when no attack is present. Results on the Emulab testbed (even though attack packets are larger) appear to be similar to the analysis and ns-2 results, since the attack creates overload on the Emulab PC routers, causing packet loss and window cuts. We found that the attack causes a significant number of RTOs on Emulab for sleep times 500–1500 ms, while the number of RTOs is negligible for other sleep times on Emulab, and for all cases on DETER and ns-2.[5] More details on why the attack is much more effective on the Emulab testbed are given below.

Observe that in the DETER and Emulab experiments, real data is pushed between the nodes, so there is a certain overhead for reading data from disk, sending it over the network through PC routers, and redirecting it to /dev/null. In contrast, *packet loss in ns-2 only occurs in case of buffer overflow*. The ns-2 nodes themselves have "infinite CPU and bus capacity," and are capable of processing any flow without contention. Since the queue service times are thus faster in ns-2 than on the testbeds, packet drops are less frequent. Another difference is that, due to the bounded capacities of the physical devices on the testbed, we find that maximum testbed packet rates cannot exceed 148 Kpackets/s, while ns-2 reports up to 250 Kpackets/s.

Packet sizes. Our DETER experiments with different attack packet sizes (results not shown here for brevity) have shown that, in case of packets with 700 byte-payload, there is an even less significant goodput degradation, confirming that small packets can cause more damage on PCs and PC routers due to higher

[5] We have observed that the Emulab and DETER goodputs are similar over this range of sleep times when we induce higher loss on DETER by using randomly-generated source and destination addresses in attack packets. This results in significant ARP traffic, as well as load from backscatter packets, which reduces goodput on DETER to the values we see on Emulab.



(a) Attack in forward direction

(b) Attack in reverse direction

(c) Attack in forward direction

(d) Attack in reverse direction

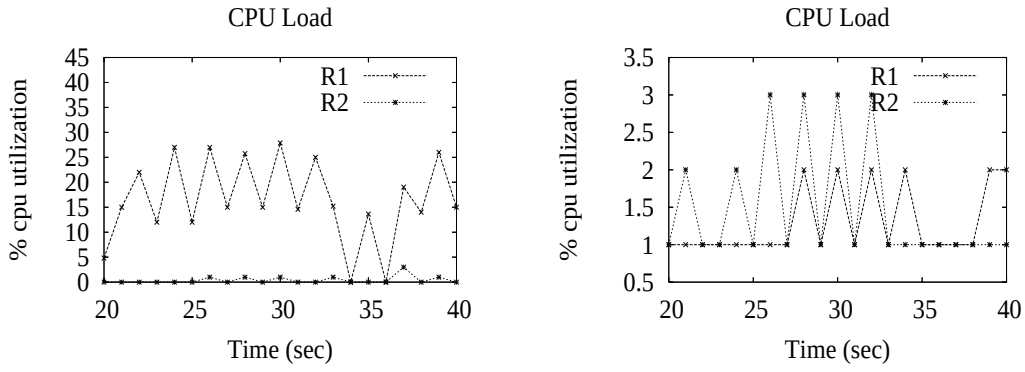
Fig. 6. Comparison of the average congestion window size and the average goodput from analysis, simulations, DETER and Emulab for different sleep time periods, and an attack pulse of length 160 ms. RTT is 160 ms. ns-2 results are not plotted in the reverse case because the attack has little impact.

packet rates, packet processing overhead, and slot based queues. Results with a payload size of 2 bytes show a slightly higher goodput degradation than with a payload of 10 bytes. In contrast, with SMP-enabled dual-CPU PC routers, goodput degradation for the same experiment was minimal, *even* with an attack packet payload size of 2 bytes, as processing can still be completed on such PC routers while executing an interrupt handler.

Attacker location. Another important point is that the ns-2 attack flow does not interfere with the file transfer if it is flowing in the opposite direction (i.e., attacker at $Node3$), since links are full-duplex, port buffers are not shared, and there is no CPU or IRQ (interrupt) overhead per packet in ns-2. Since there is no interference and one cumulative ACK loss does not typically cause a multiplicative reduction of Cwnd (just a potentially slower Cwnd increase), ns-2 is unaffected when the attack traffic flows in the opposite direction of the data traffic. We do not plot ns-2 results on Figures 6(b) and 6(d), since the system is only receiver window-limited, and the average goodput goes up.

Observations from DETER and Emulab experiments tell a different story: the average window and goodput are clearly affected by an attack in the opposite direction (e.g., Emulab goodput with an 80-byte attack packet payload size is reduced to 1.41 KBps for a 500 ms sleep time period, as shown in Figure 6(d)). The interference on DETER and Emulab is due to the fact that the Network Interface Card (NIC) on a PC router which receives a high bandwidth packet flow will consume all available system resources, and other NICs will starve. This results in interference among flows in opposing directions. Since PC routers [2] are used on Internet edges today due to their flexibility, e.g., acting as firewalls and address translators, as well as their low cost, our observation is important in understanding their operation under high loads or DoS attacks.

PC router measurements. To understand the dependence of emulation testbed results on hardware and software, we conduct the same experiment on both DETER and Emulab (but with different attack packet sizes) but we now measure CPU utilization at the two PC router nodes R1 and R2. Figure 7 depicts the results. The only difference between the setup of the DETER experiment and the Emulab experiment is that on Emulab, the attack packet includes an 83-byte payload, compared to a 2-byte payload on DETER. The nodes acting as the two routers are Pentium III 850 MHz with 256 MB RAM on Emulab; and Pentium III 733 MHz with 1024 MB RAM on DETER. The delay nodes on both testbeds are at least as fast as the router nodes. On Em-



(a) Emulab CPU utilization

(b) DETER CPU utilization

Fig. 7. Comparison of Emulab and DETER results with attack packet payload sizes of 83 versus 2 bytes respectively (attack pulse length and RTT are 160 ms; sleep time is 500 ms). In (a) the CPU utilization for R2 is always below 5% as R1 fails to forward a signifi cant portion of the traffi c.

ulab, the attacker ($Node3$ in this experiment) is a Pentium III 600 MHz with 256 MB RAM, while on DETER the attacker is of the same type as a router nodes. With faster routers, a slower attacker, and a larger attack packet payload size on Emulab, plus identical OS images, one would expect the router CPU utilization to be lower on Emulab PC router nodes than on DETER PC router nodes. The plots show that, surprisingly, the CPU load on one of the Emulab PC router nodes is much higher. This causes the attack to be much more effective on Emulab than on DETER. This behavior is likely to be due to differences in testbed hardware and system software, e.g., network devices and buses. Therefore, we conclude that testbed measurements can *significantly vary* based on hardware and system software attributes.

Analysis of our DETER and Emulab measurements has alerted us to a number of interesting issues. We quickly learned that it is crucial to measure several network and system parameters to get a clear picture of each experiment, as *ttcp* goodput/window results are insufficient to judge the fidelity of measurements. For example, we have uncovered an interesting problem with using delay nodes: since our experiments overload nodes, it is impossible to use the default Emulab optimization of delaying (shaping) *two* links using a single delay node. The Emulab and DETER teams were quick to provide us with a new command that allows mapping each link to a single delay node.

VIII. Using Click Routers

To increase the fidelity of the experiments and reduce dependence on default system settings, the PC routers were configured to run an SMP-enabled Linux-2.4.26 kernel with a multi-thread-enabled *Click-1.4.3* [21] Linux module. Because the machines on DETER have the Intel Pro/1000 Ethernet cards, it was possible to use Click's *e1000-5.7.6* NAPI polling driver to make sure that receive livelock does not occur, and Click has the most direct access to the driver. Since Emulab machines we used did not have the Intel Pro/1000 cards, we were unable to conduct experiments with Click on Emulab, since the performance would be *worse than the default Linux IP stack*. Nodes R1 and R2 in Figure 2 were configured to run as IP routers using Click's programming language.

In Click, the entire packet path is easily described, and one can easily configure a simple IP router that bypasses the OS IP stack. Simplification of the packet path yields a performance boost, making the PC router less vulnerable to overload under high packet flows. When the router is configured, each affected network device has to be included into the configuration. It is easy to change the queuing discipline and the queue depth for the queue at each output port. This feature allows more realistic queuing since the queuing is done at the actual node and not on the next DummyNet node as in previous scenarios. We will, however, show that it is insufficient to change the Click *Queue* element depth. This is because Click (or any software system for that matter) has to go through a device driver when it accepts or outputs a packet. Like any Linux network device driver, the driver for the Intel Pro/1000 card has internal transmit (TX) and receive (RX) buffers. The Click *Queue* elements serve as intermediaries between these.

In a baseline experiment, the Click routers were configured with a queue size of 50 per output port. The transmit (TX) buffer on the device drivers was left at the default value of 256 MTU sized packets.[6] With Click routers, the TCP sender was able to achieve an average rate of 74 KBps during a non-stop flood, showing that the PC router can cope with the load. An attack with a pulse length of 160 ms and a sleep period of 500 ms yielded *no perceivable difference* from the average goodput of 206 KBps when no attack was present (206 KBps is not far from DETER results for large sleep periods in Figure 6(c), but much better than DETER for short sleep periods).[7] Since Click was able to cope with high rates and the TCP flow was receiver window limited, *ttcp* was configured to use large buffers (and hence

[6]The fboding agent used in the experiments was compared with Click's *udpgen* to ensure that our generator is on par with that of Click and can produce over 148 Kpackets/sec, when using UDP packets with a 10-byte payload.

[7]Interference in the reverse direction was found to be much less than without using Click (Figure 6(d)).

larger receiver windows) in order to investigate the effectiveness of the attack. The large buffers increased the maximum TCP transfer to rate to 560 KBps. In this case, the attack with a 160 ms pulse length and a sleep period of 500 ms was reported to generate 37960 packets per second or 1.97 MBps by Click's *udpcount*. This attack reduced the TCP goodput to almost one fifth of the value without an attack. This confirms that the attack has a potential for great damage while still being perceived as low rate on the average.

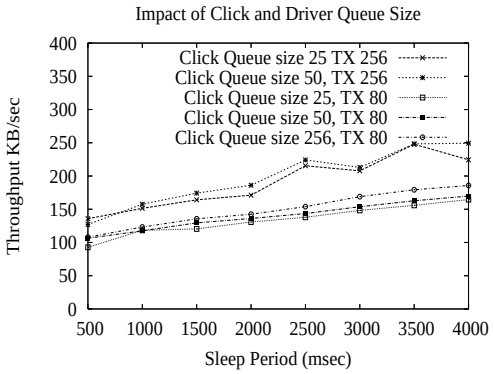


Fig. 8. Impact of varying the Click router queue and the transmit buffer ring of the device driver on DETER. The attack packet payload size is 10 bytes.

Our final set of experiments aim at understanding the effect of varying Click and device driver buffer sizes. The Click *Queue* element size was set to 25, 50, or 256 slots, while the driver transmit (TX) buffer was set to 80 (minimum setting) or 256 (default). We did not experiment with varying the receive (RX) buffer sizes since [9] demonstrated that receiving is much faster than transmitting on PCs, and hence drops do not occur due to RX buffer overflow. The default drop-tail queuing discipline was used. Figure 8 demonstrates that varying the TX buffer size produces *significant* variation in the results. It is also important to note that the TX buffer size has a *much more profound* impact than the Click queue size. Figure 8 clearly shows that a TX of 256 and a Click *Queue* of 50 performs much better than a TX of 80 and a Click *Queue* of 256. This implies that it is *crucial* to be aware of the driver settings.

IX. Summary and Lessons Learned

In this paper, we have investigated the applicability of simulation and emulation for DoS attack experimentation. Our results give insights into the effectiveness of low-rate TCP-targeted attacks, as well as more generally into how to conduct high fidelity experiments. We summarize our findings below.

Effectiveness and stealthiness of TCP-targeted attacks. Our results validate that variants of low-rate TCP-targeted attacks can still be effective even when the attack frequency is not precisely tuned to the retransmission timeout [16], [22], [24]. The router type and buffer size, attack pulse length and packet size, and location of the attacker have a significant impact on the effectiveness and stealthiness of the attack. Simulation results have indicated that the attack is most successful when it is multiplexed with TCP data and not ACKs; however, this was not true when PC routers were bottlenecks in DETER and Emulab experiments. As in previous studies [28], we have demonstrated the superiority of the NAPI (polling) approach under high packet rates, underscoring the need to have more NAPI capable drivers. Results with Click PC routers have shown that we need a better understanding of how to compare performance with certain Click Queue sizes and device driver buffer sizes to performance with other types of routers and to simulation models. Results have also revealed that attack pulses much shorter than the RTT are ineffective and cannot throttle the connection at a desired rate, due to non-overlapping attack pulse and file transfer bursts. Even attack pulses of length equal to the RTT are ineffective when router buffers are large, and there is a single TCP flow under attack.

Defending against TCP-targeted attacks. The stealthiness of variants of TCP-targeted low-rate attacks makes defending against them a challenging task. In the original TCP-targeted low-rate attacks work [22], the authors studied active queue management mechanisms such as FRED [23], RED-PD [25], and SRED [31], for defending against the attacks. They concluded that these mechanisms only reduce – but not eliminate – the effectiveness of the attacks. There are several important considerations for defense against TCP-targeted attacks: (i) *Unknown frequency of attack:* The attack in [22] is generated with a frequency matching the RTO. This attribute is exploited in [36] to detect the attack by examining the autocorrelation function of traffic measurements at the attacked link. The attacker(s), however, can send pulses at an unknown or random frequency. (ii) *More general attacks:* An attacker need not use a precise square wave to generate the attack (e.g., [22] discusses a possible staircase pulse). (iii) *Detecting attacker(s):* Once it is determined that an attack has occurred, misbehaving flows can be prevented access into the network. A promising direction in [17] uses spectral analysis techniques to determine the number of attackers.

Simulation versus emulation. Our comparisons between simulation and emulation experiments with seemingly identical configurations have revealed *key differences* in the results. Some of these differences occur because simulators abstract a number of system attributes, and make several assumptions about packet handling. For example, since PCs are used on the Emulab and DETER testbeds, their CPUs, buses, devices, and device drivers may be bottlenecks that simulators do not model. Another important observation from comparing data from the Emulab and DETER emulation testbeds is that even though the hardware and software on both testbeds may appear similar, the nodes on Emulab experience a much higher CPU load than the DETER nodes for the same packet rate. This means that *the same experimental setup (configuration files, etc.) may produce widely different outcomes* (specifically, a much more effective attack on Emulab than on DETER), as results are highly dependent on the details of underlying hardware and software, and their default settings. These different settings do *not* cause widely different outcomes in typical networking and operating systems experiments, but cause dramatic differences under DoS attacks that overload the system. Although this appears to be undesirable, it has a positive side: Since the Internet is constantly evolving, and is comprised

of *heterogeneous* hardware and software – and not simplified models as in simulators – it is of great value to identify cases where the same experiment yields very different results depending on the components of the test hardware/software, and unforeseen interactions among them. In addition, PC routers like those used on the testbeds are used on the Internet today, and hence understanding their behavior is important.

Future work. We plan to repeat our experiments with different commercial routers, and quantify the differences in the results. WAIL contains 34 Cisco routers ranging from Cisco GSR 12000 to Cisco 2600. DETER recently started supporting Juniper MI70 routers. We also plan to investigate how various devices can be modeled in simulators. Finally, we plan to study more complex topologies and traffic patterns, with multiple flows (with different bottlenecks and RTTs) and multiple attackers and attack traffic rates.

REFERENCES

[1] Classful Queueing Disciplines. http://www.tldp.org/HOWTO/Adv-Routing-HOWTO/lartc.qdisc.classful.html.

[2] ImageStream and Cisco Comparison. http://www.imagestream.com/Cisco_Comparison.html.

[3] OSS network routers. http://www.siriusit.co.uk/docs/doc.php?pageID=8&typeID=3.

[4] PC300 - The Router Killer. http://www.cyclades.com/resources/?wp=6.

[5] Sangoma. http://www.sangoma.com.

[6] S. Agarwal, J. Sommers, and P. Barford. Scalable network path emulation. In *Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, September 2005.

[7] M. Allman, V. Paxson, and W. Stevens. TCP congestion control. RFC 2581, April 1999.

[8] D. G. Andersen. Mayday: Distributed filtering for internet services. In *Proceedings of USITS*, 2003.

[9] A. Bianco, R. Birke, D. Bolognesi, J. Finochietto, G. Galante, M. Mellia, M. Prashant, and F. Neri. Click vs. linux: Two efficient open-source IP network stacks for software routers. In *IEEE Workshop on High Performance Switching and Routing*, May 2005.

[10] Y. Chen, K. Hwang, and Y.-K. Kwok. Collaborative defense against periodic shrew DDoS attacks in frequency domain. *Submitted*, 2005. Available at: http://gridsec.usc.edu/files/TR/ACMTISSEC-LowRateAttack-May3-05.pdf.

[11] D. Dittrich. The "stacheldraht" distributed denial of service attack tool. http://staff.washington.edu/dittrich/misc/stacheldraht.analysis, 1999.

[12] P. Druschel, L. Peterson, and B. Davie. Experiences with a high-speed network adaptor: A software perspective. In *Proceedings of the ACM SIGCOMM Conference*, pages 2–13, August 1994.

[13] R. Bajcsy et. al. Cyber defense technology networking and evaluation. *Communications of the ACM*, 47(3):58–61, March 2004.

[14] S. Floyd and T. Henderson. The NewReno modification to TCP's fast recovery algorithm. RFC 2582, April 1999.

[15] S. Floyd and E. Kohler. Internet research needs better models. *SIGCOMM Comput. Commun. Rev.*, 33(1):29–34, 2003.

[16] M. Guirguis, A. Bestavros, and I. Matta. Exploiting the transients of adaptation for RoQ attacks on internet resources. In *Proceedings of ICNP*, Oct 2004.

[17] A. Hussain, J. Heidemann, and C. Papadopoulos. A framework for classifying denial of service attacks. In *Proc. of ACM SIGCOMM*, August 2003.

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

[19] V. Jacobson. Congestion avoidance and control. In *Proc. of the ACM SIGCOMM*, volume 18, pages 314–329, August 1988.

[20] C. Jin, H. Wang, and K. G. Shin. Hop-count filtering: An effective defense against spoofed DoS traffic. In *Proc. of ACM CCS*, pages 30–41, Oct 2003.

[21] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The Click modular router. *ACM Transactions on Computer Systems*, 18(3):263–297, August 2000.

[22] A. Kuzmanovic and E. W. Knightly. Low-rate tcp-targeted denial of service attacks (the shrew vs. the mice and elephants). In *Proc. of ACM SIGCOMM*, August 2003.

[23] D. Lin and R. Morris. Dynamics of random early dectection. In *Proc. of the ACM SIGCOMM*, volume 27, pages 127–136, September 1997.

[24] X. Luo and R. K.-C. Chang. On a new class of pulsing denial-of-service attacks and the defense. In *Network and Distributed System Security Symposium (NDSS)*, February 2005.

[25] R. Mahajan, S. Floyd, and D. Wetherall. Controlling high-bandwidth flows at the congested router. In *Proc. of IEEE ICNP*, November 2001.

[26] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. TCP selective acknowledgement options. RFC 2018, October 1996.

[27] J. Mirkovic and P. Reiher. A taxonomy of DDoS attacks and DDoS defense mechanisms. *ACM Computer Communications Review*, 34(2):39–54, April 2004.

[28] J. Mogul and K. K. Ramakrishnan. Eliminating receive livelock in an interrupt-driven kernel. *ACM Transactions on Computer Systems*, 15(3):217–252, August 1997.

[29] D. Moore, G. Voelker, and S. Savage. Inferring Internet denial-of-service activity. In *Proceedings of USENIX*, 2001.

[30] M. Muuss. The story of the TTCP program. http://ftp.arl.mil/~mike/ttcp.html.

[31] T. J. Ott, T. V. Lakshman, and L. H. Wong. SRED: Stabilized RED. In *Proc. of IEEE INFOCOM*, volume 3, pages 1346–1355, March 1999.

[32] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP throughput: A simple model and its empirical validation. In *Proc. of the ACM SIGCOMM*, volume 28, pages 303–314, September 1998.

[33] K. Park and H. Lee. On the effectiveness of route-based packet filtering for distributed DoS attack prevention in power-law internets. In *Proceedings of ACM SIGCOMM*, August 2001.

[34] MOSES Project. iSSF and iSSFNet network simulators. http://www.linklings.net/MOSES/?page=software, 2005.

[35] L. Rizzo. DummyNet. http://info.iet.unipi.it/~luigi/ip_dummynet/.

[36] H. Sun, J. Lui, and D. Yau. Defending against low-rate TCP attacks: Dynamic detection and protection. In *Proceedings of ICNP*, Oct 2004.

[37] UCB/LBNL/VINT groups. UCB/LBNL/VINT Network Simulator. http://www.isi.edu/nsnam/ns/, 2005.

[38] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. An integrated experimental environment for distributed systems and networks. In *Proc. of OSDI*, pages 255–270, December 2002.

[39] A. Yaar, A. Perrig, and D. Song. SIFF: A stateless internet flow filter to mitigate DDoS flooding attacks. In *IEEE Symposium on Security and Privacy*, May 2004.