Fidelity of Network Simulation and Emulation: A Case Study of TCP-Targeted Denial of Service Attacks

Roman Chertov, Sonia Fahmy, Ness B. Shroff

In this paper, we investigate the differences between *simulation* and *emulation* when conducting denial of service (DoS) attack experiments. 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, Emulab, and WAIL testbeds with default system settings. We 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. Our experiments with commercial Cisco routers demonstrate that they are highly susceptible to the TCP-targeted attacks when ingress/egress IP filters are used.

Categories and Subject Descriptors: C.2.3 [**Network Operations**]: Network monitoring; C.4 [**Performance of Systems**]: Performance attributes; I.6.4 [**Simulation and Modeling**]: Model validation and analysis

General Terms: Measurement, Security, Performance

Additional Key Words and Phrases: Simulation, emulation, testbeds, TCP, congestion control, denial of service attacks, low-rate TCP-targeted attacks

1. INTRODUCTION

Denial of Service (DoS) attacks have become increasingly prevalent [Mirkovic and Reiher 2004; Moore et al. 2001]. 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 2007 and in 2002). In this work, we address the question

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.

Authors' addresses: Roman Chertov and Sonia Fahmy are with the Department of Computer Science, Purdue University, 305 N. University St., West Lafayette, IN 47907–2107, USA. Tel: +1-765-494-6183. Fax: +1-765-494-0739, E-mail: {rchertov,fahmy}@purdue.edu. Ness B. Shroff is with the departments of ECE and CSE, The Ohio State University, 2015 Neil Ave., Columbus, OH 43210, USA. E-mail: shroff@ece.ohio-state.edu.

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* [Guirguis et al. 2004; Kuzmanovic and Knightly 2003]. 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 performance degradation to TCP flows traversing the same routers. We compare results from a simple analytical model of TCP goodput under this attack scenario to results from the popular ns-2 simulator, and the DETER, Emulab, and WAIL testbeds under the same scenario.

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). Further, in a deterministic simulation environment, traffic oscillations induced by these attacks can lead to phase synchronization effects with slight parameter changes [Floyd and Kohler 2003]. Such effects may appear interesting, but they are not representative of real systems. Additionally, experiments with this attack on real testbeds may be sensitive to the testbed capabilities and parameter settings.

A number of recent studies [Guirguis et al. 2004; Luo and Chang 2005; Sun et al. 2004; Chen et al. 2005] have experimented with TCP-targeted attacks, and proposed defenses against them. Among these, only [Guirguis et al. 2004; Luo and Chang 2005] conducted testbed experiments, but these experiments were conducted with (*tc*, *iproute2*), NIST-net, or DummyNet [Rizzo 2000] 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.

This paper is a significant extension of our preliminary study [Chertov et al. 2006], which did not include any results using commercial routers. The primary contribution of this paper is a careful comparison of results from simulation and emulation experiments with different DoS attack *and* system parameters. We also compare PC routers to commercial Cisco routers. We explore key problems that arise due to differences in testbed capabilities and default parameter settings. We design constructs for the emulation testbeds to achieve a level of control comparable to simulation tools. Our results reveal that significant differences can exist between simulation and emulation experiments even if the experimental setups are almost identical. Finally, we find that configurable software routers such as Click [Kohler et al. 2000] provide a flexible experimental platform, but require careful manipulation of the underlying network device drivers.

The remainder of this paper is organized as follows. Section 2 provides background on the attack we use as our case study. Section 3 gives a simple analytical model of the performance degradation caused by TCP-targeted attacks. Section 4 describes the emulation environment we use, and the tools that we have developed for the DETER emulation testbed. Section 5 explains our experimental setup. Sections 6 and 7 discuss our results from ns-2, Emulab, and DETER experiments. Section 8 describes our ex-

periences with the Click router. Section 9 discusses our experiments with commercial Cisco routers on the WAIL testbed. Section 10 presents the results when multiple TCP flows are present. Finally, Section 11 summarizes our findings.

2. TCP-TARGETED ATTACKS

TCP-targeted DoS attacks are an ideal case study for understanding the pros and cons of different evaluation methods and testbeds. 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 [Park and Lee 2001; Yaar et al. 2004; Ioannidis and Bellovin 2002; Andersen 2003; Jin et al. 2003]. 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 [Kuzmanovic and Knightly 2003][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 is that an attacker transmits short pulses, i.e., square waves, with periodicity close to the Retransmission-Timeout (RTO) interval [Jacobson 1988] 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 widespread 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 and 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 [Guirguis et al. 2004] 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

[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.

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. Moreover, these attacks may be even more difficult to detect, since they do not operate at a known frequency. While RoQ attacks may not cause TCP goodput to virtually go to zero, as in the case of [Kuzmanovic and Knightly 2003], they can still significantly degrade service quality. Table I demonstrates the bandwidth[2] of a TCP-targeted attack averaged over a one second interval. The table demonstrates that the average bandwidth that the attack flow consumes is a small portion of a 100 Mbps Ethernet link. Therefore, we use these attacks as a case study in our work.

Packet rate	Packet size	Sleep (ms)	Pulse (ms)	Bandwidth (Mbps)
16 Kpackets/s	64	500	160	1.99
16 Kpackets/s	64	2000	160	0.61
85 Kpackets/s	64	500	160	10.55
85 Kpackets/s	64	2000	160	3.22
140 Kpackets/s	64	500	160	17.38
140 Kpackets/s	64	2000	160	5.31

Table I. Average bandwidth of a TCP-targeted attack

3. SIMPLE ANALYTICAL MODEL

In this section, we give a simple analytical model that we use in our comparisons to gauge the effectiveness of an attack. The model can be considered as a special case of the model given in [Luo and Chang 2005]. The model characterizes TCP performance degradation as a function of the TCP-targeted attack frequency. In prior work, e.g., [Padhye et al. 1998], models of TCP throughput as a function of the round-trip time and loss event rate were developed. These models, however, did 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 [Allman et al. 1999] in the congestion avoidance phase is being employed for a single flow under attack.[3] As discussed in Section 2, 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

[2]The bandwidth calculation includes the Ethernet, IP, and UDP headers.

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

will induce a packet loss, resulting in Cwnd being cut in half but not inducing an RTO. 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 [Floyd and Henderson 1999] and TCP SACK [Mathis et al. 1996] *even* with a few packet losses with every pulse, since these TCP flavors can typically recover from multiple packet losses without RTOs.

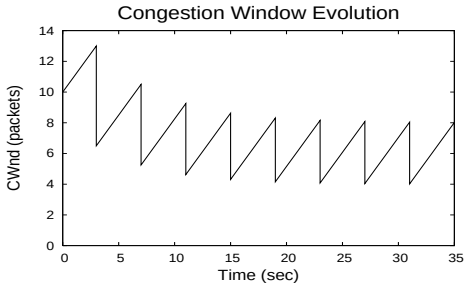


Fig. 1. Saw-tooth pattern of congestion window evolution due to periodic loss every 4 seconds.

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 the default TCP behavior where every *other* packet is acked [Kurose and Ross 2001]; $\frac{t}{rtt}$ 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 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, $W_{avg} = \frac{\alpha+2\alpha}{2} = \frac{3t}{4rtt}$.

From the computed W_{avg}, we can approximate the average throughput as: $Rate = W_{avg} \times MSS \times 1/rtt$ [Kurose and Ross 2001], where *MSS* is the Maximum Segment Size and *rtt* is the connection RTT.

4. EMULATION ENVIRONMENT AND TOOLS

In order to experiment with this DoS attack in a high fidelity – yet quarantined – setting, we leverage the DETER (www.isi.deterlab.net), Emulab (www.emulab.net), and WAIL (www.schooner.wail.wisc.edu) testbeds. Emulab is a time- and space-shared network emulator located at the University of Utah [White et al. 2002]. The system comprises hundreds of linked PCs that can be connected in any specified topology, and a suite of software tools that manage them. When an experiment gets allocated, the management system allocates PCs and then connects them via VLANs, creating a desired topology. A key advantage of such emulation testbeds is that results are reproducible, allowing for detailed comparisons and careful sensitivity analysis.

The Cyber Defense Technology Experimental Research Network (DETER) is an emulation testbed – based on Emulab – that allows researchers to evaluate Internet security technologies [Bajcsy and et al 2004]. DETER can be accessed remotely, but is quarantined from the Internet. This allows for experimentation with live malware without infecting real Internet hosts. 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 Wisconsin Advanced Internet Laboratory (WAIL) is another testbed based on Emulab that has a variety of commercial Cisco routers available to the researchers.

The primary advantage of using a network *emulator* – as opposed to a *simulator* – for security experiments is that an emulation environment affords much higher fidelity, provided that it is correctly configured. Further, real security appliances (e.g., off-the-shelf hardware) can be tested on it. This can expose unforeseen implementation vulnerabilities, protocol interactions, and resource constraints. This is because an emulation testbed uses real devices 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 or idealized simulation model.

In addition to emulating link delay and bandwidth, network emulation testbeds may include one component that differs from real Internet components: 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 WAIL have demonstrated that a regular commodity PC running Linux may outperform low to mid-range Cisco routers. Specialty PC routers such as the ones from ImageStream are usually created from high-end PCs that have SMP, multiple buses, fast memory, and industrial network cards [Cyclades Corporation 2000; Sangoma Technologies Corp 2006]. The performance of specially configured PC routers can challenge that of Cisco 7000 series as well as Juniper M-5/M-10 routers according to [ImageStream Internet Solutions, Inc. 2006; Sirius Corporation 2004]. Therefore, it is important that we

understand the behavior of PC routers under attack. We also use Cisco 3600/7000-series routers on WAIL to compare the results.

Developing efficient PC routers has been the subject of significant research, e.g., [Druschel et al. 1994; Mogul and Ramakrishnan 1997; Kohler et al. 2000]. In these studies, polling and/or Direct Memory Access (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.

We first experiment with PC routers with interrupt-based packet processing, since not all device drivers support polling, and hence understanding their behavior is important. Section 8 describes our experiences with a polling-based Click router.

Event control system. In network simulators such as ns-2 [UCB/LBNL/VINT groups 2005] and iSSF/iSSFNet [MOSES Project 2005], 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 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, 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. We have developed a multi-threaded utility, which we refer to as a *Scriptable Event System*, to parse a 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.[4]

Measurement tools. Instrumentation and measurement on a 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 log statistics such as CPU utilization, packets per second, and memory utilization to the local disk for later manipulation. Scripts for measuring, merging, and plotting system data are also available for download.

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

5. EXPERIMENTAL SETUP

The topology used for both simulation and testbed experiments is depicted in Figure 2.[5] This is a simple dumb-bell topology with four end-systems and two routers. Unlike traditional dumb-bell topologies, the backbone link has no delay. The reason for this is that the current setup on the WAIL testbed makes it impossible to introduce any propagation delay on a router-to-router link. To make accurate testbed comparisons, we have moved all delays to the access links. The links for the attack nodes have no delays since the attack traffic is delay-agnostic. The attacker and the attack sink are varied from one side of the topology to the other. *The same* basic ns-2 script is used for both simulations and testbed experiments.

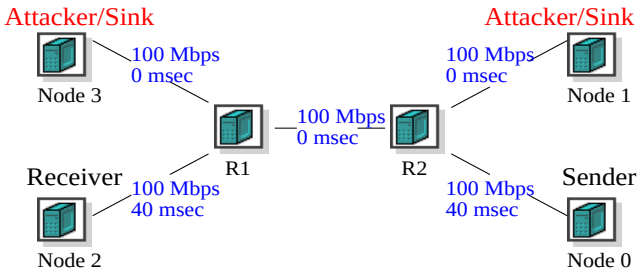


Fig. 2. Simple dumb-bell topology with 160 ms round-trip-time and 100 Mbps links.

5.1 Testbed Setup

Machine specifics. On the DETER testbed, we use dual Pentium III (P3) 733 MHz with 1024 MB of RAM and dual P4 Xeon 2.8 GHz with 2048 MB of RAM. Both machine types have Intel Pro/1000 cards. P3s are used for *Node0*, *Node2*, *R1*, and *R2*, while P4s are used for *Node1* and *Node3*. On Emulab, *Node0* and *Node2* are P3 600 MHz with 256 MB of RAM; *Node1* and *Node3* are P4 2.0 GHz with 512 MB of RAM; and *R1* and *R2* are P3 850 MHz with 256 MB of RAM. The P3s have Intel EtherExpress Pro 100 cards while the P4s have Intel Pros. On WAIL, all hosts are P4 2.0 GHz with 1024 MB of RAM with Intel Pro cards, while *R1* and *R2* are varied between Cisco 3640s and a 7206VXR-7500 pair. The choice of the specific machine types on the testbeds was driven by the need to minimize cross-switch hops: since the emulation testbeds contain a very large number of machines, multiple switches are used to connect them together, potentially leading to loss at cross-switch hops.

In our first set of experiments (Section 7), all nodes ran Linux 2.4.26 with IRQ-driven packet processing, thus being susceptible to receive livelock [Mogul and Ramakrishnan 1997]. The default NIC driver is e1000-5.2.30.1-k1 on DETER, and

[5]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 includes experiments with multiple bottleneck configurations and complex traffic patterns.

eepro100.c:v1.09j-t9/29/99 on Emulab. We used TCP SACK [Mathis et al. 1996] with delayed acknowledgments in ns-2, as well as on the testbed machines. The ns-2 (drop tail) buffer sizes for routers were set to 50 packets (except in experiments when we vary the queue size). The buffer size of 50 was chosen because, according to DummyNet documentation [Rizzo 2000], 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. Shaping methods such as DummyNet [Rizzo 2000] or *tc* have been found to induce artifacts, due to bursty behavior and not being always true to the desired properties [Hubert and et al 2002; Agarwal et al. 2005]. To avoid these problems, link delays on the DETER/Emulab/WAIL testbeds are emulated by running the Click router on delay nodes. 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 packet loss occurs at the delay nodes. To accomplish this task, we selected Dual Pentium 4 Xeon 2.8 GHz machines with PCI-X, running SMP and *polling-enabled* Linux 2.4.26 to act as delay nodes on DETER, and 2.0 GHz uniprocessor nodes on Emulab/WAIL. To avoid bursty behavior, we configured the kernel clock to run at a higher frequency of 1000 Hz. Unlike the default Emulab-type delay nodes, each of our delay nodes are assigned to shape a single link instead of two.

5.2 Attack Generation

On the testbeds, the attack packet sizes (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 most of our DoS attacks (unless we are experimenting with the attack packet size). 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. Although our attack generator 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 [Moore et al. 2001] overhead on PC routers.

In our ns-2 experiments, the attack is created by a CBR agent that pulses at regular intervals. To achieve the same behavior on the testbed nodes running Linux, we use raw sockets for packet transmission. System timers are 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. We use the real clock to generate desired packet rates and space packets equally in time as much as possible. 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 the testbeds. In the case of ns-2, it has been observed that deterministic simulations can lead to synchronization artifacts [Floyd and Kohler 2003]. To break this synchronization, we have added random jitter to the attack pulse and sleep periods that can extend them by $U(0, 10)$ ms, approximating the

coarseness of a system clock. Thus, we repeat each simulation or testbed experiment ten times and report mean values as well as 95-percent confidence intervals.

Our attack generator, *flood*, is capable of generating packets at variable rates. To ensure that the desired rate is achieved, the packets are evenly spaced in time, and a high-end machine is required to generate rates of over 140 Kpackets/s.[6] The flooding agent used in the experiments was compared with Click's kernel element *udpgen* to ensure that our generator is on par with that of Click. Attack packets take 64 bytes at the data link layer, as 64 bytes is the smallest possible Ethernet frame. Since ns-2 does not explicitly increase packet size as the packet traverses the protocol stack (i.e., packet size does not increase as headers are added), we have explicitly fixed the size of ns-2 attack packets to 64 bytes.

5.3 Traffic Generation and Measurement

To gauge the impact of the attack on the three testbeds, we use *iperf* [Tirumala and et al 2005] to create a single TCP flow for 250 seconds. We chose to have a single "good" flow (except in Section 10) as this creates the worst case for the attacker, since the attacker must send enough traffic to fill the queues and induce losses. Such a long-lived TCP flow is not unrealistic, since several Internet sites offer large downloads, e.g., DVD-quality movies. In addition, this scenario simplifies the comparison between simulation and testbed results. The *iperf* tool reports statistics such as total time and transfer rate. In our case, the transfer rate is a good indicator of the attack potency. We have configured the experimental nodes as suggested in [Mathis and Reddy 2006], and used an 11 MB receive window in *iperf* to allow full link utilization. TCP traces collected by *tcpdump* are processed by *tcptrace* to derive the following statistics: number of sent/received data packets, number of sent/received ACKs, number of retransmits, and congestion window estimates. Since we are interested in the effects of the attack during the TCP *congestion avoidance phase*, we have developed a tool *cwnd_track* that records the window for a specific connection from */proc/net/tcp* several times per second (once per RTT).

On the testbeds, 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 send a TCP stream to *Node2* via *iperf* for 250 seconds. Upon successful completion of the task, the attacker ceases the attack, and the measurement and *tcpdump* files are transferred to the project account for analysis. The ns-2 simulations and the testbed experiments use *the same* basic *tcl* topology script, and we log the attributes of the ns-2 TCP agent directly. The simulation time is 250 seconds and Cwnd values are collected 40 seconds after the simulation start to discount the effects of slow start. We report Cwnd as the number of outstanding segments (instead of bytes) to correspond to the analytical model. To conduct accurate transfer rate comparisons between ns-2 and the testbeds, we set the ns-2 TCP packet size to 1505 bytes. The average payload size reported by *tcptrace* when it analyzes the tcpdump files is 1447. This means

[6] The limit of 140 Kpackets/s is due to the limitation of the hardware on the network card itself.

that the average packet size is 1505 bytes if Ethernet (14+4 bytes), IP (20 bytes), and TCP (20 bytes) overhead is included.

We encountered two problems in our initial experiments. First, we noticed a large number of hardware duplicates in the *tcpdump* files when we captured traces at the sending node. We suspect the problem is due to *libpcap* as the duplicates were never sent on the wire. To resolve this problem, we use the delay nodes on the testbeds to collect TCP traces via *tcpdump* in a similar fashion as was described in [Chertov 2005]. However, we have combined the tasks of delaying and monitoring into a single node as the number of available testbed nodes is usually limited. Since the delay/capture nodes only deal with TCP traffic, they can easily capture 100 Mbps TCP traffic as the packet rates are less than 9000 Kpackets/s [Chertov 2005].

Second, we experienced cases where the window results from *tcptrace* and *cwnd_trak* appeared erroneous, and did not correspond to the observed goodput. To make comparisons among the analysis, simulations, and testbed experiments possible, we use goodput instead, applying a simple transformation to the analytical model that gives an approximate analytical goodput, as described in Section 3.

5.4 Experimental Design

We investigate the impact of varying the following parameters:

(i) The attack sleep time from 500 ms to 4000 ms in 500 ms increments, and from 2000 ms to 20000 ms in 2000 ms increments;

(ii) The attack pulse length to be 20, 40, 60, 80, 120, 160, or 200 ms;

(iii) The attack packet size to be 64, 1184, or 1400 bytes;

(iv) The attack packet rate to be 8.3, 16, 85, or 140 Kpackets/s;

(v) The round-trip time of the attack flow and the long-lived TCP flow to be 60, 80, 120, 160, 200 or 420 ms;

(vi) Router buffer sizes on ns-2 routers to be 25, 50, 100, 150 or 256 packets;

(vii) Router buffer sizes on Click to be 50 or 250 packets;

(viii) The transmission ring buffer size of the *e1000* driver to be 80 or 256 packets; and

(ix) The placement of the attacker to be either *Node1* or *Node3*.

We compute the following metrics:

(i) Average goodput in Kbps (Kbits per second) or Mbps, as reported by *iperf* or computed from ns-2 results;

(ii) Average congestion window size in packets, computed for testbed experiments by taking an average of the congestion window values tracked by our *cwnd_trak* utility;

(iii) CPU percentage utilization from */proc/uptime*; and

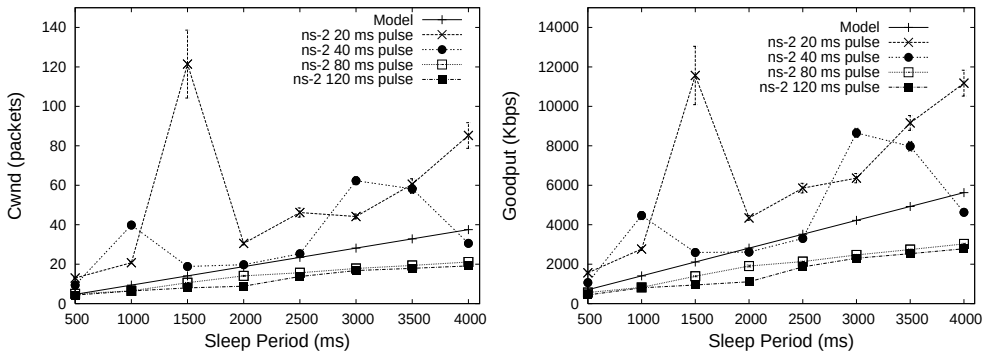
(iv) Packets per second received and sent on the *test network* interfaces, computed for testbed experiments by using Click's *udpcount* utility.

System-level measurements, e.g., CPU utilization, cannot be collected in simulations, since ns-2 does not model host CPUs.

6. SIMULATION RESULTS

In this section, we compare results from the analytical model in Section 3 and the ns-2 simulator. To compute the analytical goodput, we set the *MSS* to the size of ns-2 TCP packets (1505 bytes). We compare both the analytical Cwnd and goodput to the simulation results to verify that the Cwnd to goodput transformation defined in Section 3 is sufficiently accurate.

Impact of attack pulse length. We first investigate the impact of varying the length of the attack pulse. The length of the attack pulse controls the trade-off 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, while the length of the attack pulse is set to 20, 40, 80, or 120 ms. To increase the chances that each attack pulse results in the loss of a TCP packet, we have set the attack rate at just over 195 Kpackets/s. This rate consumes an entire 100 Mbps link with 64-byte packets.



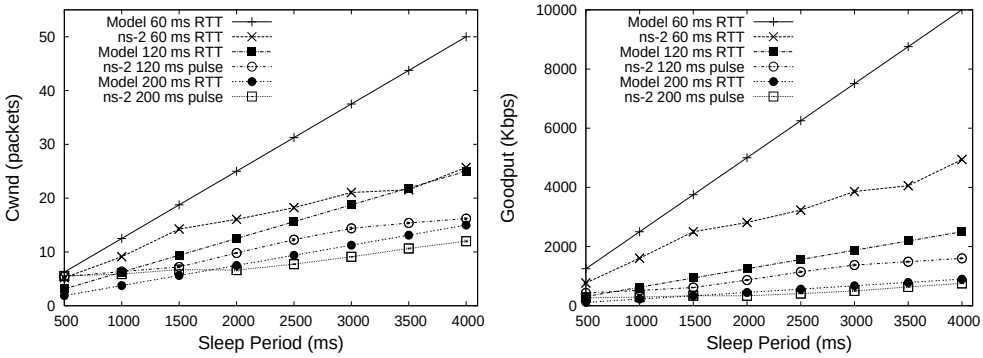
(a) Congestion Window (b) Goodput

Fig. 3. ns-2 simulation results with varying attack pulse lengths. The RTT is 80 ms in this experiment. Attack packets are 64 bytes.

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 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 in 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. Even the added randomness in the attack traffic does not entirely remove these synchronization effects. Adding randomness to the TCP sender/receiver nodes as well as the routing nodes may potentially eliminate synchronization effects. Additionally, creating random inter packet gaps between the attack packets can reduce the synchronization effect. In the real world, it is almost impossible to have precisely equal inter packet gaps in a CBR flow.

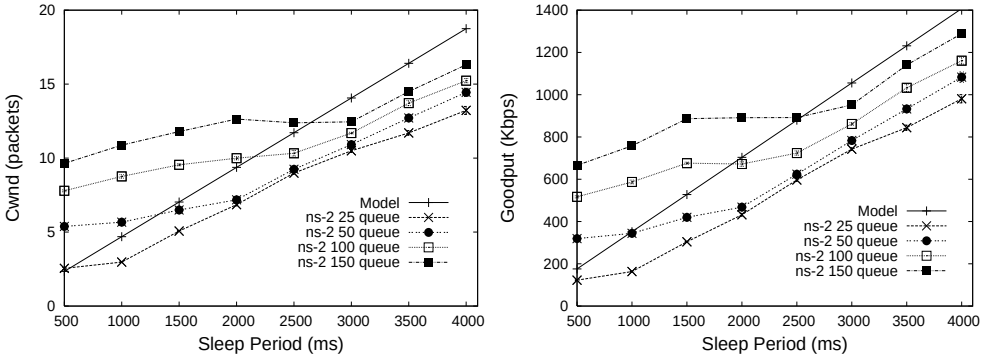


(a) Congestion Window (b) Goodput

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. Attack packets are 64 bytes.

The figure also shows that there is a good 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 be equal to the RTT, for different values of RTT. The matches between the analytical model and simulations hold, except for small RTT values. The large deviation between the model and the simulation results for the

60 ms RTT is due to a large number of RTOs which significantly reduce the goodput and violate the assumptions of the model.



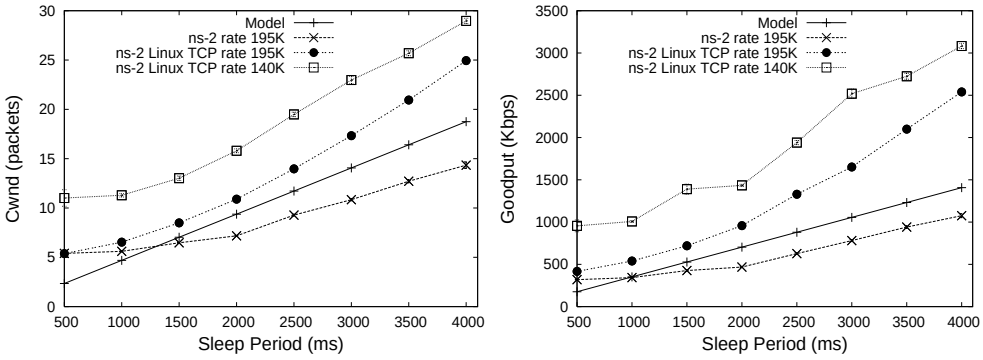
(a) Congestion Window (b) Goodput

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 packets are 64 bytes.

Impact of queue size. Figure 5 illustrates the impact of varying the sizes of the router buffers (queues). 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 or more flows under attack, the attack is likely to be effective, even when the router buffer sizes are larger.

Comparing Figures 3(a) and (b), 4(a) and (b), and 5(a) and (b) demonstrates that using the analytical goodput computed from the analytical W_{avg} is sufficiently accurate for our comparisons. Therefore, in the remainder of the paper, we will use goodput instead of window size in our plots.

Impact of TCP type and attack rate. In the previous simulations, we used the ns-2 TCP SACK agent. However, TCP behavior in the Linux operating system is somewhat different, and the Linux algorithm is more aggressive in the face of congestion [Sarolahti and Kuznetsov 2002]. The work in [Wei and Cao 2006] gives a module that allows using the TCP code from Linux 2.6.16-3 from within ns-2. Their kernel version is newer than what we use in our testbed experiments (2.4.26), but we have verified that the code for Reno congestion avoidance and slow start is fundamentally unchanged between the two versions. As stated in Section 5.2, the maximum packet rate that we could achieve on the testbed was 140 Kpackets/s, but our earlier simulations used 195 Kpackets/s. To have a fair comparison of simulation results with various testbed results in the following sections, we have reduced the attack rate in the simulations



(a) Congestion Window (b) Goodput

Fig. 6. ns-2 simulation results with using Linux Reno TCP SACK and ns-2 TCP SACK. Attack rate is varied between 195 Kpackets/s and 140 Kpackets/s. The round trip time is 160 ms. Attack packets are 64 bytes.

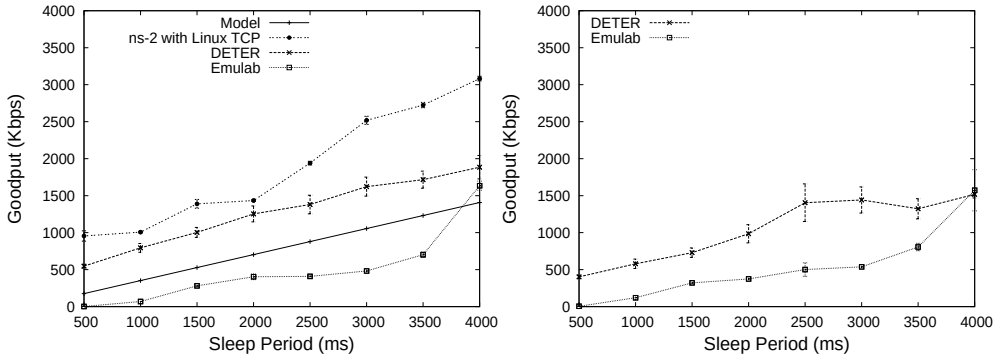
down to 140 Kpackets/s. Figure 6 demonstrates the differences between using two different TCP agents and also varying the attack rate. It is interesting to note that at the 195 Kpackets/s attack rate, Linux TCP performs much better than the regular ns-2 TCP SACK agent. The difference confirms that Linux TCP is more aggressive in the face of congestion. Not surprisingly, a reduction in the attack's pulse height results in better TCP performance. The results also indicate that Linux TCP is much less sensitive to packet loss, compared to the assumption of a window cut with every attack pulse in our model. In the remainder of this paper, our simulation experiments will use ns-2 Linux TCP and an attack rate of 140 Kpackets/s.

7. EMULAB AND DETER TESTBED RESULTS

In this section, we undertake the more challenging task of comparing results from the analytical model (Section 3) and ns-2 simulator to Emulab and DETER testbed results.

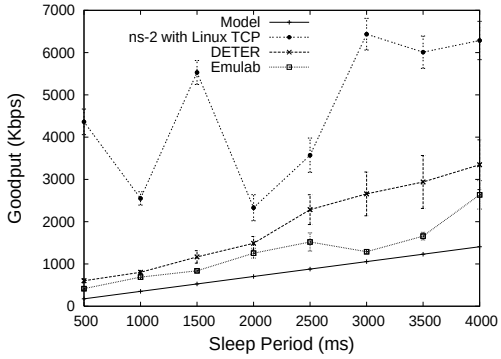
TCP without attack. We first measure the performance of the TCP flow without any attacks on Emulab and DETER. On Emulab, the average of ten runs was 83.86 Mbps with a standard deviation of 5.871 Mbps. On DETER, we have observed an average transfer rate of 88.28 Mbps with a std. deviation of 0.082 Mbps. In both cases, examining the *tcpdump* files via *tcptrace* revealed that the sender congestion window was close to the bandwidth delay product of the path, and no loss occurred at the delay nodes.

Emulab/DETER versus simulation. We initially use the default system settings for the Emulab and DETER nodes, meaning that we do not alter the default Linux configurations. In this set of experiments, the attack packets are 64 bytes. The RTT is set to 160 ms, as depicted in Figure 2. We conduct two series of experiments on Emulab

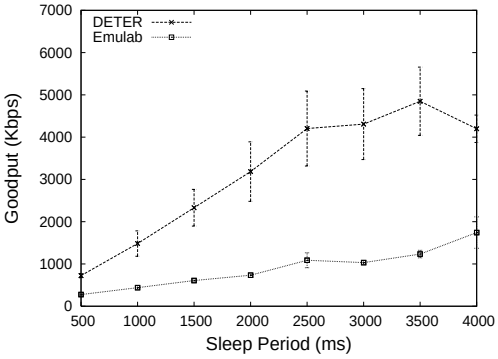


(a) Attack in forward direction at 140 Kpackets/s.

(b) Attack in reverse direction at 140 Kpackets/s.



(c) Attack in forward direction at 85 Kpackets/s.



(d) Attack in reverse direction at 85 Kpackets/s.

Fig. 7. Comparison of 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. Attack packets are 64 bytes. ns-2 results are not plotted in the reverse case because the attack has little impact.

and DETER where we vary the attack packet rate from 85,000 to 140,000 packets/s, which is the maximum transmission rate of our attack nodes. Figure 7 gives the average goodput for an attacker at node *Node1* (forward direction) or node *Node3* (reverse direction). The figures show key discrepancies between the simulation, Emulab, and DETER results.

From Figure 7(a), we find that for all values of sleep time, the goodput with ns-2 is typically higher than in other scenarios. Results on the Emulab testbed when the packet rate is at 140 Kpackets/s are mostly below the analysis and the DETER results. This is because the attack creates significant overload on the Emulab PC routers, causing significant packet loss. For the 500 ms sleep time, the goodput on Emulab is almost

zero. Examination of *tcptrace* and *netstat* output revealed that very few packets were sent throughout the duration of the connection and there was a very large number of RTOs with large idle transmit periods. This indicates that the attack behaved as in the original low-rate attack paper [Kuzmanovic and Knightly 2003]. When the attack rate is at 85 Kpackets/s, the results on Emulab are above the analytical results indicating that the attack has lost some of its potency. The simulation results reveal that synchronization effects appear even more pronounced due to non-monotonic increases in throughput. Additionally, the simulation results reveal that the attack with these packet rates and Linux TCP has a more significant impact on the testbed nodes compared to the simulated nodes.

To understand the significant differences between Emulab and DETER results, especially with 140 Kpackets/s attacks, we measure CPU utilization at the two PC router nodes *R1* and *R2*. The attackers on both testbeds are similar (2.0 GHz versus 2.8 GHz machines). We find that the load experienced on DETER PC routers is much smaller. This is because DETER PC routers have dual CPUs and an SMP configured OS, a PCI-X bus, and better NIC devices than the Emulab machines used in our experiments. This causes the attack to be more effective on Emulab than on DETER.[7] Therefore, we conclude that testbed measurements can *significantly vary* based on hardware and system software attributes.

In addition, we observe that packet loss in router nodes on the testbed can occur at a number of bottlenecks, since packets traverse multiple components and buffers, and any of these buffers can overflow. In contrast, *packet loss in ns-2 only occurs in case of output buffer overflow*. The ns-2 nodes themselves have "infinite CPU and bus capacity," and are capable of processing any flow without contention. This highly simplified and idealized router model yields a dramatically different behavior when router components such as the CPU, buses, or routing fabric become bottlenecks as in these experiments.

Impact of packet size. Our experiments with different attack packet sizes (results not shown here for brevity) have shown that, in case of packets with 1400 byte-payload, there is a less significant goodput degradation, confirming that small packets can cause more damage on PCs and PC routers due to higher packet rates, packet processing overhead, and slot-based queues.

Impact of attacker location. Another key observation is that the ns-2 attack flow does not interfere with the TCP flow 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

[7] We have observed that the Emulab and DETER goodputs are similar over this range of sleep times when we induce higher loss on DETER. For example, we used 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.

traffic flows in the opposite direction of the data traffic. We do not plot ns-2 results on Figure 7(b) or (d) since the system is only bandwidth limited, and the average goodput consumes the entire link.

Observations from Emulab and DETER experiments tell a different story: the average goodput is clearly affected by an attack in the opposite direction (e.g., Emulab goodput with a 140 Kpackets/s attack rate is almost the same as when the attack was in the forward direction as shown in Figure 7(a)). The interference on Emulab and DETER 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.

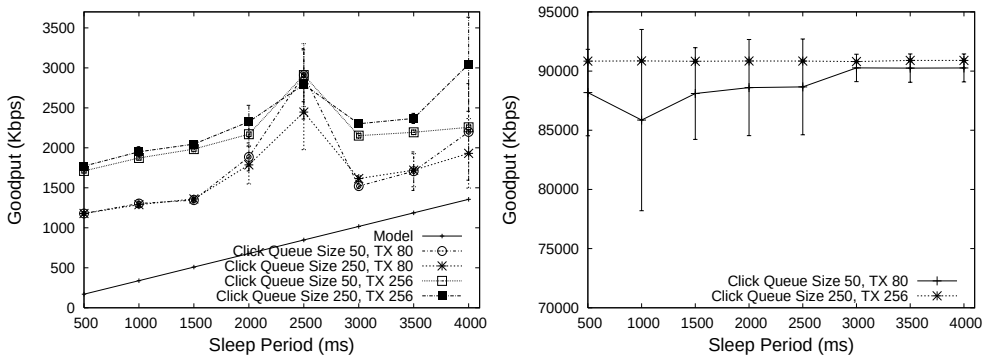
8. USING THE CLICK MODULAR ROUTER

In this section, nodes *R1* and *R2* in Figure 2 are configured to run the *Click* Linux module post version 1.5.0. Because DETER machines have Intel Pro/1000 Ethernet cards, it was possible to use Click's *e1000-5.7.6* NAPI polling driver to ensure that receive livelock does not occur, and Click has the most direct access to the driver. Additionally, we use dual 2.8 GHz P4 Xeon PC routers to leverage their faster CPUs and buses. Using this more complex IP router configuration caused stability issues with SMP. To overcome this problem, we had to compile the kernel and Click for a single CPU. Further, we changed the kernel version to Linux-2.4.33 to take advantage of patches that may increase stability under heavy load.

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 loads. 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. 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. As with 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. We have created a simplified version of an IP router as described in [Kohler et al. 2000], where the input ports are connected to a routing table. The routing table then sends the packets to the appropriate *Queue* element from which the output port pulls the packets.

Forwarding performance. We configure the Click routers with a queue size of 250 packets per output port. The transmit (TX) buffer on the device drivers was left at the default value of 256 MTU-sized packets. Without any attacks, the TCP flow achieved an average goodput of 90.83 Mbps with a std. deviation 0.054 Mbps. With a non-stop flood of over 140 Kpackets/s at *Node1*, Click's *udpcount* element at *Node3* verified that the receiving rate was the same as the sending rate with minimal packet loss. We confirmed the results with the Emulab's *portstats* utility.

Impact of Click queue and device queue sizes. To understand the effect of varying Click (OS intermediates) and device driver buffer sizes, the Click *Queue* element size was set to 50 or 250 slots, while the driver transmit (TX) buffer was set to 80 (minimum setting) or 256 (default). We did not experiment with varying receive (RX) buffer sizes since [Bianco et al. 2005] demonstrated that receiving is much faster than transmitting on PCs, and hence drops do not typically occur due to the RX buffer overflow. The default drop-tail queuing discipline was used. Figure 8(a) 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 250. This implies that it is *crucial* to be aware of the driver settings. We have also experimented with an attack in the reverse direction as the TCP flow. Figure 8(b) clearly indicates the advantage of polling versus interrupts in Click: there is little interference between the opposing flows.



(a) Attack in forward direction (b) Attack in reverse direction

Fig. 8. Impact of varying the Click router queue and the transmit buffer ring of the network device driver on DETER. Attack packets are 64 bytes. The attack pulse length and RTT are set to 160 ms.

Click versus simulation. Since we know the exact size of the default TX output queue (256 slots), we ran a set of ns-2 simulations with router output buffer sizes set to 256 packets. We decided not to include the size of intermediate Click queue as the previous experiments have shown that the TX buffer sizes are more significant. In the simulations, we set the attack packet rate to 140 Kpackets/s (as it is on the testbed). Figure 9 depicts the results. The simulation results follow Click's results *after the 10 second mark* for this case. The simulation and Click results indicate that the attack is weaker than the analysis predicts.

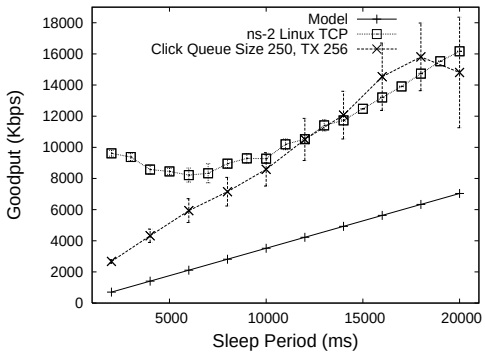


Fig. 9. Comparison of the average goodput from analysis, simulation, and Click (on DETER) for long sleep periods. Attack packets are 64 bytes. Queue size in ns-2 and output device buffer ring in Click is 256. The attack pulse length and RTT are set to 160 ms.

9. WAIL TESTBED RESULTS

In this section, we conduct experiments with commercial Cisco routers on the WAIL testbed.

9.1 Cisco 3640 Routers

Cisco 3640 is a multi-service access platform for medium and large-sized enterprises and small Internet Service Providers [Cisco Systems 2006c]. The Cisco 3640 router is essentially a PC that runs on a 100 MHz RISC-type CPU and uses DRAM and NVRAM for buffer and OS storage. This router uses interrupts for packet processing, making it susceptible to receive livelock. The routers used for our experiments have been configured to use Cisco IOS Version 12.4(5). By default, the input queue size is 75 slots and the output queue size is 40 slots.

Forwarding performance. The router supports "Process Switching" mode [Cisco Systems 2006d] and "Interrupt Context Switching" [Cisco Systems 2006d], as well as several caching options. In process switching mode, the router uses IRQ handlers for receiving and sending packets. Additionally, the router schedules an *ip_input* process that performs the necessary look-ups and makes switching decisions. Since the router has to schedule the *ip_input* process for each packet and the CPU is quite underpowered, the router cannot forward a full 100 Mbps flow composed of MTU-sized packets. In our tests, the router was able to forward a maximum of 1248 64-byte packets per second.

The interrupt context switching mode is an optimized version of the process switching mode. The optimization happens when all the switching work is performed via the interrupt handlers and the *ip_input* process does not get scheduled. In our experiments, the router was able to process 14.4 K 64-byte packets per second in the interrupt context switching mode with "Cisco Express Forwarding" (CEF) [Cisco Systems 2006d], which is a significant improvement over the process switching mode.

TCP without attack. Cisco IOS offers the ability to install IP filters on the input and output ports to provide egress/ingress filtering capability. Since using IP filters is advised to prevent IP spoofing, we ran our experiments with and without filters. Figure 2 depicts the topology that we have used, but *R1* and *R2* are now Cisco 3640 routers. The experiment was run in exactly the same manner as was described in Section 5. We first ran ten experiments without the attack to determine TCP performance under normal conditions. The filters only allowed the packets with IPs that were part of the test network to be forwarded. Without the filters, the TCP flow was able to achieve an average transfer rate of 83.3 Mbps with a std. deviation of 0.0213 Mbps; however, with the filters the goodput dramatically dropped to an average of 7.3 Mbps with a std. deviation of 0.111 Mbps. Enabling IP filters forces the packets to be "process"- instead of "fast path"-switched. The difference in switching paths can result in a performance difference of an order of magnitude [Cisco Systems 2006a].

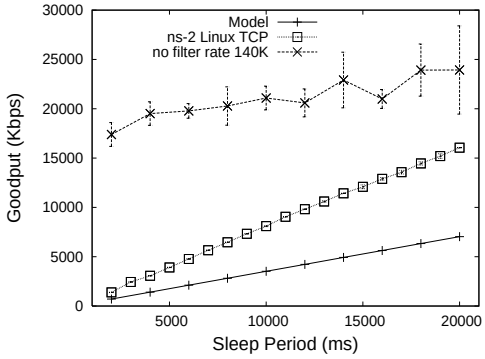
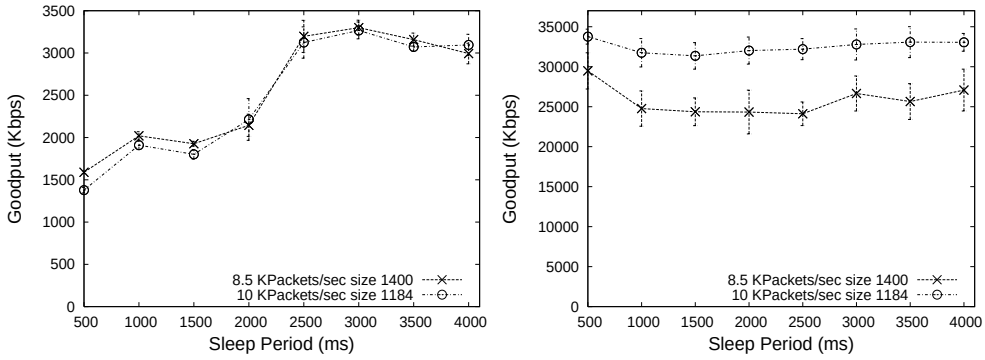


Fig. 10. Comparison of the average goodput from analysis, simulation, and Cisco 3640s when long sleep periods are used. The IP filters on the Cisco routers are disabled. Attack packets are 64 bytes. The attack pulse length and RTT are set to 160 ms.

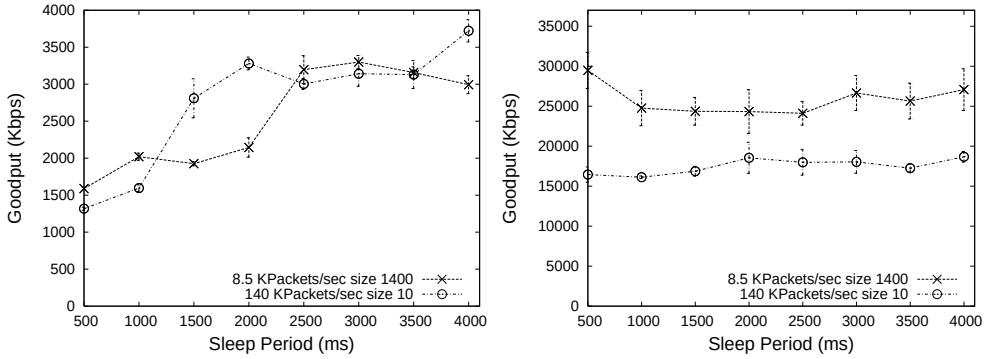
3640 versus simulation. Since we know the exact size of the router output queues (40 slots), we ran a set of simulations with this size. In the simulations, we set the attack packet rate to 140 Kpackets/s exactly as on the testbed. We disabled the IP filters on the routers and varied the sleep times from 2 seconds to 20 seconds in 2 second increments to capture a variety of attack scenarios. Figure 10 depicts the results. The router results indicate that the attack is much *less effective*, compared to the simulations and analysis. This is because most of the attack traffic is dropped at the input queue of the interface that receives the attack traffic. If the attack traffic is multiplexed with the TCP traffic on the same input port, then the attack potency is significantly higher.[8]

[8]We have also observed that if we use TCP packets instead of UDP packets for the attack, the TCP goodput degradation is closer to the model. This is because a stream of unsolicited TCP data packets causes the sink to respond with RST packets, causing significant congestion at the router.



(a) With IP filters (b) Without IP filters

Fig. 11. Effect of varying packet size and packet rate – while maintaining the same bit rate of 98 Mbps – with and without IP filters on Cisco 3640s. The attack is sent at 8.5 Kpackets/s with 1400 byte payload, or at 10 Kpackets/s with 1184 byte payload. The attack pulse length and RTT are set to 160 ms.



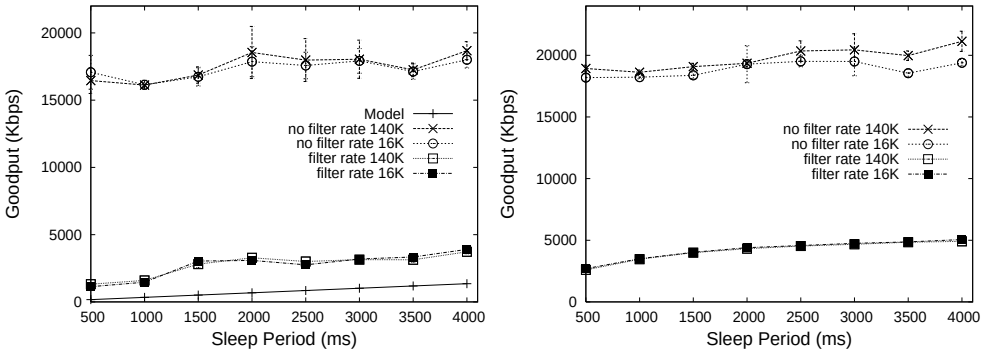
(a) With IP filters (b) Without IP filters

Fig. 12. Impact of 140 Kpackets/s (64-byte packet size) versus 8.5 Kpackets/s (1400 byte payload) attack flows on Cisco 3640s with and without IP filters. The attack pulse length and RTT are set to 160 ms.

Impact of packet size. We now examine the effect of maintaining the same attack bit rate while changing the packet rate, by varying the packet size. We observe that achieving the same bit rate is not always possible by varying packet rates and sizes: for instance, 8500 Kpackets/s with 1400-byte payload creates a 98 Mbps flow (includes UDP/IP/Ethernet headers). The smallest packet size to use to achieve the same bit rate is 1184 bytes. Any smaller size fails to meet the desired bit rate.

Figure 11 demonstrates TCP goodput when the attack rate is 8.5 Kpackets/s with

1400-byte payload versus 10 Kpackets/s with 1184-byte payload packets. The experiment was run with and without IP filters. The results surprisingly indicate that slightly larger packets at a slightly lower packet rate cause more damage to TCP. To investigate the impact of varying packet rates, we conducted another experiment where the packet rate was 140 Kpackets/s and packets were 64 bytes, and compared it to the 8.5 Kpackets/s with 1400-byte payload results. Figure 12 demonstrates the results. It is interesting to note that when IP filters were present, larger packets sometimes induced more damage than smaller packets; however, when IP filters were removed, smaller packets at a higher rate clearly had a more profound impact.



(a) Attack in forward direction (b) Attack in reverse direction

Fig. 13. Comparison of the average goodput from analysis, simulations, and WAIL Cisco 3640s for different sleep periods, attack rates, and a pulse length of 160 ms. RTT is 160ms. Attack packets are 64 bytes. The reported results for the Cisco routers are with and without IP filters. ns-2 results are not plotted in the reverse case because the attack has little impact.

Impact of IP filters. Figure 13 illustrates the results with the TCP-targeted attack in forward and reverse directions. As in Section 7, we use UDP packets that create 64-byte frames at the data link (Ethernet) layer. The attack rate is either 140 Kpackets/s or 16 Kpackets/s (14 Kpackets/s is the measured maximum loss free forwarding rate). The results confirm that the *use of IP filters has a profound effect on the impact of the attack*. Figure 13(a) and (b) show that an attack with a maximum pulse height is not meaningful if the device can be overloaded at lower attack intensity levels. By observing this fact, an attacker can make the attack much more stealthy and harder to detect. Finally, *the results indicate that the attack has an impact in the reverse direction, even when commercial routers are used.*

9.2 Cisco 7000 Series Routers

A few rungs up the ladder from the 3600 series are the 7000 series routers. Such routers are designed as aggregation points for large enterprises or Points of Presence (POP) for small ISPs. The routers support a variety of network modules that include Gigabit Ethernet, Fast Ethernet, SONET, and Serial. Router documentation [Cisco Systems 2006b] reveals these routers to be similar to multi-bus interrupt-driven PCs. Similar routers are advertised by ImageStream [ImageStream Internet Solutions, Inc. 2006] based on commodity PCs with special network cards running a modified version of the Linux operating system. Unfortunately, the router layout on the WAIL testbed does not allow us to use the same topology as in Section 5. The link connecting the two routers is a 155 Mbps POS-3 instead of a 100 Mbps Fast Ethernet. We used a Cisco 7206VXR with IOS Version 12.3(10) as *R2* and a Cisco 7500 with IOS Version 12.3(17b) as *R1* in Figure 2.

Forwarding performance. Unlike the case of the 3640s, the 7000 series routers are capable of forwarding almost 140 Kpackets/s when IP filters are disabled.

TCP without attack. Without the attack and with no filters, the TCP flow achieved an average rate of 24.99 Mbps with 4.53 Mbps std. deviation. With the filters, the average rate decreased to 6.37 Mbps with a std. deviation of 0.179 Mbps. Note that the TCP flow did not achieve a full link utilization in the no-filter run; this was due to occasional packet loss in the routers. However, with multiple flows using the "-P" option in *iperf*, the aggregate of the flows did reach full link utilization. We suspect that some hardware issue is causing occasional packet loss. For high-speed TCP flows, even an occasional loss is detrimental as the congestion window is significantly cut, requiring many consecutive round trips to reach full link capacity.

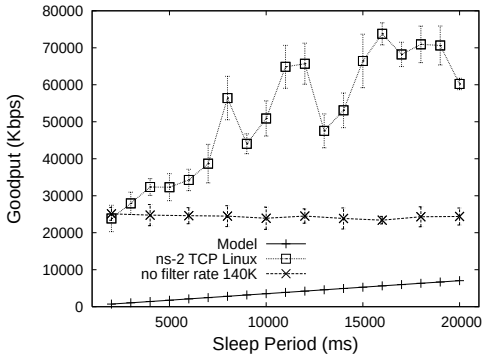
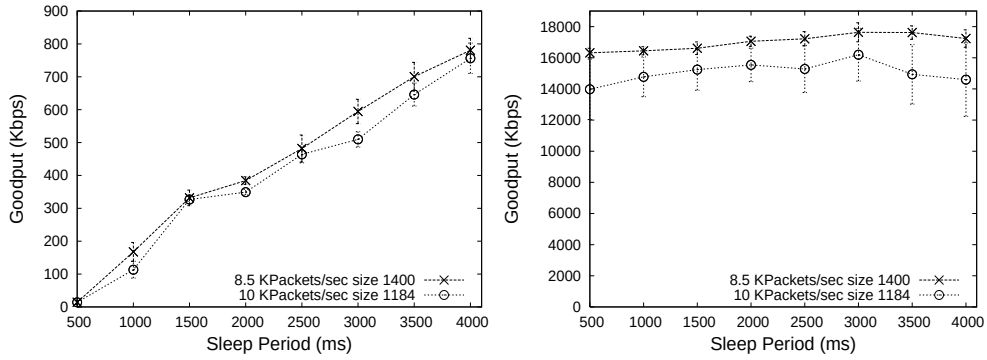


Fig. 14. Comparison of the average goodput from analysis, simulation, and Cisco 7000s when long sleep periods are used. The IP filters on the Cisco routers are disabled. Attack packets are 64 bytes. The attack pulse length and RTT are set to 160 ms.

7000s versus simulation. Since the link between the routers was a 155 Mbps SONET link and the output queues were of size 40, we ran another set of simulations with these settings. In the simulations, the attack packet rate is set to 140 Kpackets/s as on the testbed. Again, we ran the routers without IP filters and varied the sleep times from 2 seconds to 20 seconds in 2 second increments. Figure 14 illustrates the results. In the simulation results for the 140 Kpackets/s attack, there is substantially more damage initially, but then the attack impact is quickly reduced as the throughput climbs to almost 80 Mbps in some cases. The reason for the discrepancy between simulation and the analytical model is due to the fact that the link speed difference invalidates the assumption that each attack pulse will result in a Cwnd cut. The results on the routers are very close to results when no attack was present, showing that *the attack is not very effective* in this case.

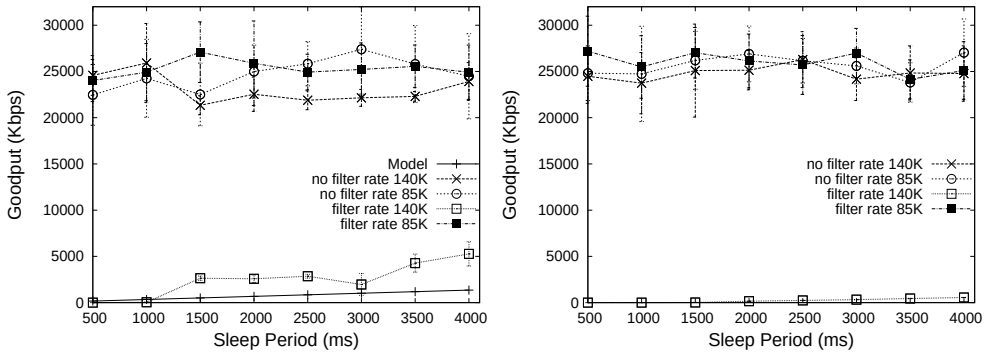


(a) With IP filters (b) Without IP filters

Fig. 15. Effect of varying packet size and packet rate while maintaining the same bit rate of 98 Mbps with and without IP filters on Cisco 7000s. The attack is sent at 8.5 Kpackets/s (1400 byte payload) or 10 Kpackets/s (1184 byte payload). The attack pulse length and RTT are set to 160 ms.

Impact of packet size. As with the 3640 routers, we compare the effect of maintaining the same bit rate while changing the packet rate by varying the packet size. Figure 15 demonstrates the results when the attack rate is 8.5 Kpackets/s with 1400-byte payload packets versus a 10 Kpackets/s rate with 1184-byte payload packets. As before, the experiment was run with and without IP filters. Unlike the 3640 results, the 7000 results clearly indicate that even slightly smaller packets at higher packet rates produce more damage.

Impact of IP filters. Figure 16 shows the results for forward and reverse direction attacks with varying attack packet rates, with and without IP filters. It is interesting to compare the results for forward and reverse directions when IP filters are used. The first router to receive the attack flow drops some of the attack packets; hence, the



(a) Attack in forward direction (b) Attack in reverse direction

Fig. 16. Comparison of the average goodput from analysis, simulations and WAIL Cisco 7000s for different sleep periods, attack rates, and a pulse length of 160 ms. RTT is 160 ms. Attack packets are 64 bytes. bytes. The reported results for the Cisco routers are with and without IP filters. ns-2 results are not plotted in the reverse case because the attack has little impact.

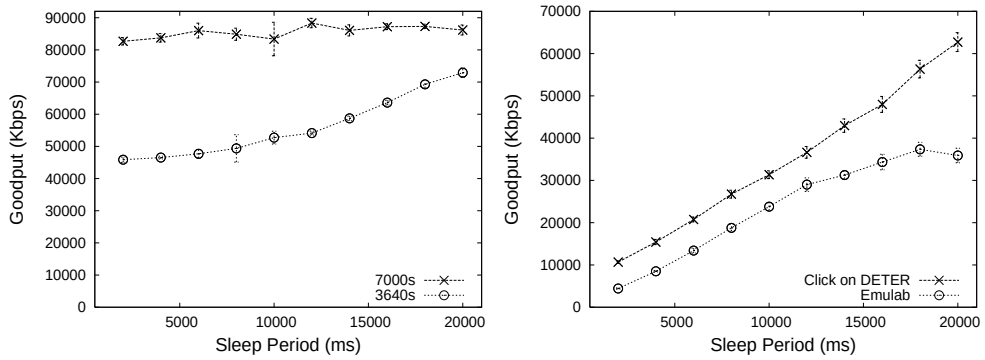
downstream router receives a reduced attack flow. By changing direction of the attack, we change which router experiences the higher load. The large difference between forward and reverse directions at 140 Kpackets/s when using IP filters is due to the fact that the Cisco 7206VXR merges the TCP and the attack flows in the forward direction, while the Cisco 7500 merges the TCP ACK and the attack flows in the reverse direction. The Cisco 7206VXR has a R7000 CPU at 350 MHz, while the Cisco 7500 has a R5000 CPU at 200 MHz. It is not surprising that the slower CPU on the 7500 led to higher packet loss when IP filtering was enabled. It is also interesting to note that the impact of the attack on the TCP goodput is higher when larger packets at low rates are used (Figure 15(b)) compared to the attack with smaller packets but higher rates (Figure 16(a)) (e.g., 16 Mbps vs. 25 Mbps). The results indicate that *in this case, large attack packets cause more damage than smaller ones.*

The results imply two important conclusions. First, as in all previous experiments except for Click, the attack has an impact in both the forward and *reverse* directions. Second, IP filters dramatically increase the impact of the attack.

10. THE CASE OF MULTIPLE TCP FLOWS

All our experiments thus far have dealt with only a single TCP flow under attack. Not surprisingly, TCP is highly sensitive to loss, especially on fast links, as it halves its window in the face of a single packet loss. In reality, rarely is there only a single TCP flow on a link. For the sake of completeness, we ran a series of experiments where 9 TCP flows were present. To avoid flow synchronization, we created three different RTT paths: 80 ms, 160 ms, and 420 ms. This was accomplished by creating a LAN with 3 nodes attached to *R2*. The delays were constructed in the same manner as

previously discussed. Each LAN node had three flows between it and *Node2* using the "-P" *iperf* option.



(a) WAIL routers (b) Emulab versus Click on DETER

Fig. 17. Impact of the attack on the aggregate goodput of 9 TCP fbws with different RTTs (80 ms, 160 ms, and 420 ms). The attack pulse length is set to 160 ms. Attack packets are 64 bytes.

TCP without attack. On DETER, when running Click PC routers, we observed the total goodput of the flows (averaged over several experiments) to be 91.6 Mbps with a 0.28 Mbps std. deviation. On Emulab (without Click PC routers), the total goodput of the flows was 45.29 Mbps with a 2.6 Mbps std. deviation.[9] On the Cisco 3640s on WAIL, the total goodput was 75.74 Mbps with a 1.13 Mbps std. deviation. On the Cisco 7000s, the total was 89.83 Mbps with a 1.03 Mbps std. deviation.

Impact of the attack. We vary the attack sleep period from 2 to 20 seconds in 2 seconds increments on DETER (with Click PC routers), Emulab (without Click PC routers), and WAIL. The attack was configured to generate 140 Kpackets/s in the forward direction with an attack pulse duration of 160 ms. On WAIL routers, we disabled IP filters to obtain the best performance. Figures 17(a) and (b) depict the aggregate results for the 9 flows. There are two key observations to be made. First, the attack on the 7000-series routers appears to do almost no damage (Figure 17(a)). Second, comparing Emulab and DETER-with-Click, Emulab results show the goodput to be lower than on DETER as expected. However, since the TCP flows did not achieve full link utilization on Emulab when no attack was present, it is hard to make a concrete conclusion.

The experiments highlight a key point: A single TCP flow can be highly susceptible to the low-rate attack; however, *the aggregate of multiple flows is much more immune*

[9]Nodes *R1* and *R2* were the sources of occasional loss as the aggregate packet rate of the fbws was higher than that of a single fbw.

especially when the sleep period exceeds 5 seconds. Additionally, when the TCP flows have different RTTs, the small RTT flows are affected very little, thus increasing the aggregate goodput.

11. SUMMARY AND LESSONS LEARNED

In this paper, we have investigated the applicability of simulators and emulation testbeds for DoS attack experimentation. Our results give interesting insights into how to conduct high fidelity experiments.

Our comparisons between simulation and emulation experiments with seemingly identical configurations have revealed *key differences* in the results. Such dramatic differences occur because simulators and emulated components abstract a number of system attributes, and make several assumptions about packet handling. For example, in PC routers on the Emulab and DETER testbeds, the CPUs, buses, devices, and device drivers may become bottlenecks that simulators do not model. Results are also highly sensitive to the particular software version. For example, Linux TCP flows on the testbeds and in the simulation were less susceptible to the attack than the SACK implementation on ns-2. Another key result is that the attack is effective in the reverse direction in the case of Cisco routers and PC routers, but not in simulations or using Click.

An important observation on the data from the Emulab and DETER emulation testbeds is that although the hardware and software on both testbeds may appear similar, the nodes on Emulab typically experience a 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* typically cause widely different outcomes in typical networking and operating systems experiments, but cause dramatic differences under DoS attacks that overload the system. As in previous studies [Mogul and Ramakrishnan 1997], 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 intermediate "Queue" sizes and device driver buffer sizes.

Our experience with commercial Cisco routers has demonstrated that some, but not all, can be similar to software PC routers in performance. Use of IP ingress/egress filters can be detrimental to the network performance under a DoS attack.

The main lesson from this study is the need to carefully configure denial of service experiments, and the importance of not generalizing results beyond the specific configurations used in the experiments. Although the diversity of results on different testbeds appears to be undesirable, it has a positive side: Since the Internet is constantly evolving, and comprises *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. Additionally, results for the same attack on various platforms define the scope in which DoS attacks range from being effective to ineffective.

REFERENCES

AGARWAL, S., SOMMERS, J., AND BARFORD, P. 2005. Scalable network path emulation. In *Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*.

ALLMAN, M., PAXSON, V., AND STEVENS, W. 1999. TCP congestion control. RFC 2581.

ANDERSEN, D. G. 2003. Mayday: Distributed filtering for Internet services. In *Proceedings of USITS*.

BAJCSY, R. AND ET AL. 2004. Cyber defense technology networking and evaluation. *Communications of the ACM 47*, 3 (March), 58–61.

BIANCO, A., BIRKE, R., BOLOGNESI, D., FINOCHIETTO, J., GALANTE, G., MELLIA, M., PRASHANT, M., AND NERI, F. 2005. Click vs. linux: Two efficient open-source IP network stacks for software routers. In *IEEE Workshop on High Performance Switching and Routing*.

CHEN, Y., HWANG, K., AND KWOK, Y.-K. 2005. Collaborative defense against periodic shrew DDoS attacks in frequency domain. *Submitted*. Available at: http://gridsec.usc.edu/files/TR/ACMTISSEC-LowRateAttack-May3-05.pdf.

CHERTOV, R. 2005. Performance of a software link monitor. http://www.cs.purdue.edu/homes/rchertov/reports/click.pdf.

CHERTOV, R., FAHMY, S., AND SHROFF, N. B. 2006. Emulation versus simulation: A case study of TCP-targeted denial of service attacks. In *Proceedings of the 2nd International IEEE CreateNet Conference on Tesbeds and Research Infrastructures (TridentCom)*.

CISCO SYSTEMS. 2006a. Cisco 3600 series router architecture. http://www.cisco.com/en/US/products/hw/routers/ps274/products_tech_no%te09186a00801e1155.shtml.

CISCO SYSTEMS. 2006b. Cisco 7200 series router architecture. http://www.cisco.com/en/US/products/hw/routers/ps341/products_tech_no%te09186a0080094ea3.shtml.

CISCO SYSTEMS. 2006c. Cisco routers. http://www.cisco.com/warp/public/cc/pd/rt/index.shtml.

CISCO SYSTEMS. 2006d. How to choose the best router switching path for your network. http://www.cisco.com/en/US/tech/tk827/tk831/technologies_white_paper0%9186a00800a62d9.shtml.

CYCLADES CORPORATION. 2000. Cyclades-PC300: The Router Killer. http://www.cyclades.com/resources/?wp=6.

DRUSCHEL, P., PETERSON, L., AND DAVIE, B. 1994. Experiences with a high-speed network adaptor: A software perspective. In *Proceedings of the ACM SIGCOMM Conference*. 2–13.

FLOYD, S. AND HENDERSON, T. 1999. The NewReno modification to TCP's fast recovery algorithm. RFC 2582.

FLOYD, S. AND KOHLER, E. 2003. Internet research needs better models. *SIGCOMM Comput. Commun. Rev. 33*, 1, 29–34.

GUIRGUIS, M., BESTAVROS, A., AND MATTA, I. 2004. Exploiting the transients of adaptation for RoQ attacks on internet resources. In *Proceedings of ICNP*.

HUBERT, B. AND ET AL. 2002. Linux Advanced Routing & Traffic Control HOWTO. http://www.tldp.org/HOWTO/Adv-Routing-HOWTO/lartc.qdisc.classful.html.

IMAGESTREAM INTERNET SOLUTIONS, INC. 2006. ImageStream and Cisco Comparison. http://www.imagestream.com/Cisco_Comparison.html.

IOANNIDIS, J. AND BELLOVIN, S. M. 2002. Implementing pushback: Router-based defense against DDoS attacks. In *NDSS*.

JACOBSON, V. 1988. Congestion avoidance and control. In *Proc. of the ACM SIGCOMM*. Vol. 18. 314–329.

JIN, C., WANG, H., AND SHIN, K. G. 2003. Hop-count filtering: An effective defense against spoofed DoS traffic. In *Proc. of ACM CCS*. 30–41.

KOHLER, E., MORRIS, R., CHEN, B., JANNOTTI, J., AND KAASHOEK, M. F. 2000. The Click modular router. *ACM Transactions on Computer Systems 18*, 3 (August), 263–297.

KUROSE, J. F. AND ROSS, K. W. 2001. *Computer Networking – A top-down approach featuring the Internet*. Addison Wesley.

KUZMANOVIC, A. AND KNIGHTLY, E. W. 2003. Low-rate TCP-targeted denial of service attacks (the shrew vs. the mice and elephants). In *Proc. of ACM SIGCOMM*.

LUO, X. AND CHANG, R. K. C. 2005. On a new class of pulsing denial-of-service attacks and the defense. In *Network and Distributed System Security Symposium (NDSS)*.

MATHIS, M., MAHDAVI, J., FLOYD, S., AND ROMANOW, A. 1996. TCP selective acknowledgement options. RFC 2018.

MATHIS, M. AND REDDY, R. 2006. Enabling high performance data transfers. http://www.psc.edu/networking/projects/tcptune/.

MIRKOVIC, J. AND REIHER, P. 2004. A taxonomy of DDoS attacks and DDoS defense mechanisms. *ACM Computer Communications Review 34*, 2 (April), 39–54.

MOGUL, J. AND RAMAKRISHNAN, K. K. 1997. Eliminating receive livelock in an interrupt-driven kernel. *ACM Transactions on Computer Systems 15*, 3 (August), 217–252.

MOORE, D., VOELKER, G., AND SAVAGE, S. 2001. Inferring Internet denial-of-service activity. In *Proceedings of USENIX*.

MOSES PROJECT. 2005. iSSF and iSSFNet network simulators. http://www.linklings.net/MOSES/?page=software.

PADHYE, J., FIROIU, V., TOWSLEY, D., AND KUROSE, J. 1998. Modeling TCP throughput: A simple model and its empirical validation. In *Proc. of the ACM SIGCOMM*. Vol. 28. 303–314.

PARK, K. AND LEE, H. 2001. On the effectiveness of route-based packet filtering for distributed DoS attack prevention in power-law internets. In *Proceedings of ACM SIGCOMM*.

RIZZO, L. 2000. DummyNet. http://info.iet.unipi.it/$\sim$luigi/ip_dummynet/.

SANGOMA TECHNOLOGIES CORP. 2006. http://www.sangoma.com.

SAROLAHTI, P. AND KUZNETSOV, A. 2002. Congestion control in linux TCP. In *In Proceedings of Usenix 2002/Freenix Track*. 49–62.

SIRIUS CORPORATION. 2004. OSS network routers. http://www.siriusit.co.uk/index.php/documentation/internet_connectivit%y/routing.

SUN, H., LUI, J., AND YAU, D. 2004. Defending against low-rate TCP attacks: Dynamic detection and protection. In *Proceedings of ICNP*.

TIRUMALA, A. AND ET AL. 2005. Iperf - the tcp/udp bandwidth measurement tool. http://dast.nlanr.net/Projects/Iperf/.

UCB/LBNL/VINT GROUPS. 2005. UCB/LBNL/VINT Network Simulator. http://www.isi.edu/nsnam/ns/.

WEI, D. X. AND CAO, P. 2006. Ns-2 tcp-linux: an ns-2 tcp implementation with congestion control algorithms from linux. In *WNS2 '06: Proceeding from the 2006 workshop on ns-2: the IP network simulator*. ACM Press, New York, NY, USA, 9.

WHITE, B., LEPREAU, J., STOLLER, L., RICCI, R., GURUPRASAD, S., NEWBOLD, M., HIBLER, M., BARB, C., AND JOGLEKAR, A. 2002. An integrated experimental environment for distributed systems and networks. In *Proc. of OSDI*. 255–270.

YAAR, A., PERRIG, A., AND SONG, D. 2004. SIFF: A stateless internet fbw filter to mitigate DDoS fbooding attacks. In *Proc. of IEEE Symposium on Security and Privacy*.