A Distributed Denial of Service Testbed

Desmond Schmidt, Suriadi Suriadi, Alan Tickle, Andrew Clark,
George Mohay, Ejaz Ahmed, and James Mackie

Information Security Institute, Queensland University of Technology,
126 Margaret Street, Brisbane, 4001, Australia
d.schmidt@qut.edu.au, s.suriadi@qut.edu.au,
ab.tickle@qut.edu.au, a.clark@qut.edu.au, g.mohay@qut.edu.au,
e.ahmed@qut.edu.au, j.mackie@qut.edu.au

Abstract. The Denial of Service Testing Framework (dosTF) being developed as part of the joint India-Australia research project for 'Protecting Critical Infrastructure from Denial of Service Attacks' allows for the construction, monitoring and management of emulated Distributed Denial of Service attacks using modest hardware resources. The purpose of the testbed is to study the effectiveness of different DDoS mitigation strategies and to allow for the testing of defense appliances. Experiments are saved and edited in XML as abstract descriptions of an attack/defense strategy that is only mapped to real resources at run-time. It also provides a web-application portal interface that can start, stop and monitor an attack remotely. Rather than monitoring a service under attack indirectly, by observing traffic and general system parameters, monitoring of the target application is performed directly in real time via a customised SNMP agent.

Keywords: Distributed Denial of Service, Testbed Development.

1 Introduction

This paper discusses the design of the Distributed Denial of Service testbed being developed as part of a joint India-Australia research project entitled 'Protecting Critical Infrastructure from Denial of Service Attacks: Tools, Technology and Policy'.

It is divided into four parts. Part 1 provides a brief background to the problem and the role it plays within the research project. Part 2 critically assesses existing testbeds for studying DDoS attacks. Part 3 describes the our current testbed, Part 4 describes some experiments already carried out using it, and in Part 5 we describe future directions for the testbed.

Distributed Denial of Service (DDoS) is a serious and growing problem for corporate and government services doing business on the Internet. Some botnets now number in millions of compromised machines [1, 2]. As well as for other nefarious purposes, these botnets can be used to launch Distributed Denial of Service attacks, such as those recently carried out against Twitter, Facebook [3], and government websites in the US and South Korea [4]. Modern DDoS attacks can muster 49GBps of

J. Berleur et al. (Eds.): HCC9/CIP 2010, IFIP AICT 328, pp. 338–349, 2010.

attack traffic, but more recently this type of flooding attack is giving way to more sophisticated, stealthy attacks designed to cripple a particular service [5]. The India-Australia project aims to address various aspects of this problem, and is divided into five sub-projects:

1. Probabilistic Packet Processing to Mitigate High-rate Flooding Attacks
2. DoS Defence Appliance for Web Services
3. Puzzles for DoS Mitigation in Protocols for Authenticated Key Exchange
4. Denial of Service Vulnerabilities and Challenges in Emerging Technologies
5. Harmonisation of Policy, Legal and Regulatory Environments for National Information Infrastructure Protection

Of these, the first four subprojects all require the use of a testbed facility. A DDoS testbed is an essential tool for preparing and testing the defensive strategies, appliances and protocols against such attacks as we are intending to research.

2 Existing Testbeds

Judging by existing implementations, a DDoS testbed needs to provide facilities to:

1. Specify, save and replay an experiment
2. Deploy, run and stop an experiment
3. Monitor the simulated DDoS attack in progress and to save the results to disk for later replay or analysis.

`There have been three basic strategies used for building a DDoS testbed:

1. Simulation. In this technique a network simulator such as ns-2 [6, 7] or OPNET [8] is used to specify and then instantiate a simulation on a single computer. The accuracy of such simulations and their suitability for DDoS experimentation has, however, recently been called into question [9, 10]. The attraction of simulation is that virtually any network topology can be created quickly and inexpensively; the prime disadvantage is that simulated networks when under attack may behave very differently from real or emulated networks [9].

2. Emulation. In this technique real machines are connected together to form the topology of the test network. Although the end-points of the network are mostly physical computers, the connections between networks are normally provided by soft-routers. Although more realistic than simulation, emulation suffers from scalability: it is hard to extend a local Ethernet network of PCs to model the performance of entire ISP networks that use powerful hardware routers, ATM, and multi-gigabit links.

3. The use of real networks cannot be discounted, but would seem to pose too many problems: (a) it is not possible to change the network to suit the experiment, (b) certain experiments, e.g. those involving Internet worms, could escape from the test and infect or damage the wider Internet, and (c) collateral degradation of network links may result from flooding attacks. Certain types of DDoS attacks, such as low-level stealth attacks, however, could conceivably be tested on a real section of the Internet like PlanetLab, a world-wide network of virtual machines [11].

2.1 The DETER Testbed

The DETER testbed [12, 13] is closest to the kind of design we are seeking, for a small to moderate size facility that allows experiments to be safely contained, and uses reconfigurable hardware and software. However, there are a number of reasons why we chose to deviate from the DETER design:

1. DETER uses the Emulab software. This has a GPL license, which only permits modifications under the same license. Since the terms of the India-Australia project specify that any software produced shall be licensed to the respective governments, not the general public, this is less useful to us.
2. The Emulab and DETER testbeds [6, 12] use a relatively large number of physical machines. We needed to build something with more modest resources.
3. The Emulab software design would be too complex to mimic, since it probably would cost more than the hardware it would run on [12]. For example, the ability to share and partition the testbed is not needed.
4. Our experiments comparing soft routers with hardware routers have shown that soft routers, even properly tuned, perform poorly in comparison to hardware routers with small-packet traffic, apparently because the host computer cannot process interrupts from the ethernet card fast enough to avoid dropped packets [14]. Under a DDoS flood attack a soft-router might thus introduce a serious anomaly.

The DETER testbed uses VLANs and soft-routers (but also some hardware routers) to provide flexibility. Experiments recorded as ns-scripts (in Tcl) can be quickly recreated by programming the VLANs and routers to generate the desired network topology. The advantages of ease of use, sharability, and remote access however, must be balanced against the disadvantages of higher cost in constructing, administering and maintaining the testbed.

3 The dosTF Testbed

The dosTF testbed has evolved organically in response to our own research needs. It may thus provide a useful model and alternative approach for other research groups wanting to construct their own small scale testbed for DDoS experimentation.

An example of our current experimental setup is shown in Fig. 1. The same or additional components could be rearranged as desired for a particular experiment. A total of 8 average PCs, installed with a mixture of Linux and Windows, are each fitted with two ethernet cards. These two interfaces ensure that all physical machines are dual-homed. The monitor network is on a single subnet, and is used to install software, launch and stop attacks and to monitor services during attacks. The attack network, consisting of two subnets joined by a physical router, is used to carry out attacks on particular services, to generate background traffic, or to host defense applications or devices. The physical PCs are intended to act as targets, although they may also participate as agents. One of these also acts as a point of remote access, and

another as a base for launching attacks. Three VMWare servers provide around 200 virtual hosts that may be used in simulated DDoS attacks. The driver of this design has been cost: it seems wasteful to maintain hundreds of physical machines, when most of them will only send low levels of data to the attack target. We intend eventually to evolve this design by acquiring more physical routers to study aggregation of traffic at focal points in the network topology. For now, it suffices to study the direct effects of flood and stealth attacks on applications from a range of IP-addresses.

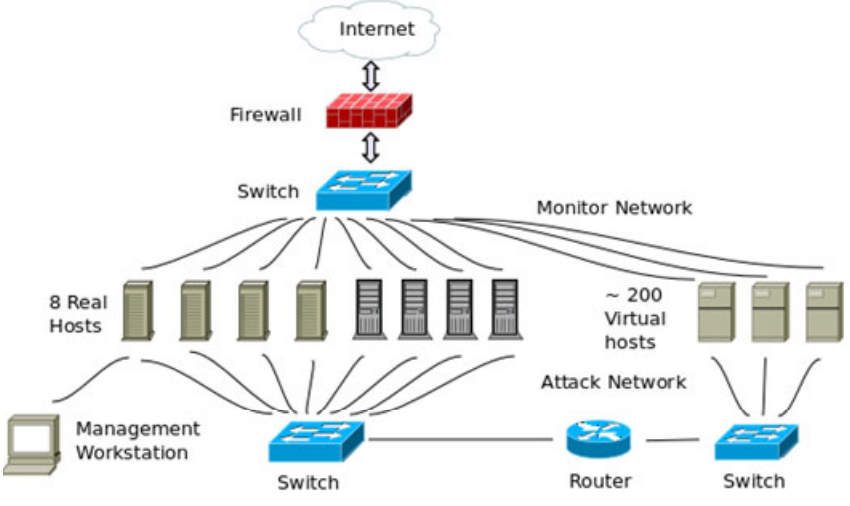

Fig. 1. Example dosTF Topology

The main difference between this design and the DETER model is shown in Fig. 2-3. Whereas in DETER a software layer is effectively introduced by the programmable VLANs and soft-routers, in our design the structure of the network topology has to be reconfigured manually. Our XML description of the experiment (or *scenario*) has no intervening software layer, and hence must refer directly to the physical testbed. This has the disadvantage that every change in the setup of the testbed will invalidate previously saved experiments.

Our solution to this deficiency is shown in Fig. 3: The scenario only stores an abstract description of the experiment: the number of attacking hosts, their operating systems, their preferred type (virtual or physical), the characteristics of the target and the software to be installed on them. When an experimenter launches an attack, the control application maps the abstract description of the experiment to physical machines using standard network discovery techniques. This approach yields the same degree of flexibility as in the DETER testbed, although the topology has to be wired manually.

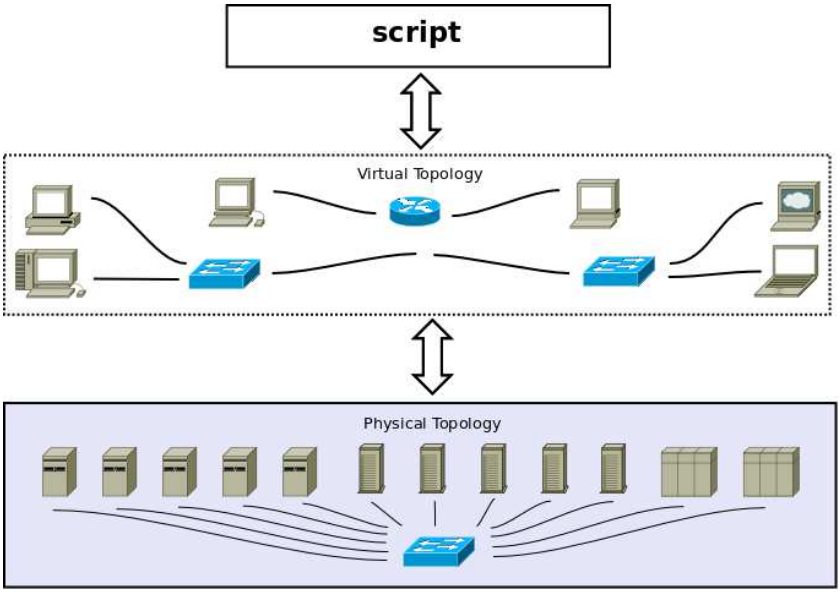

Fig. 2. DETER Testbed Model

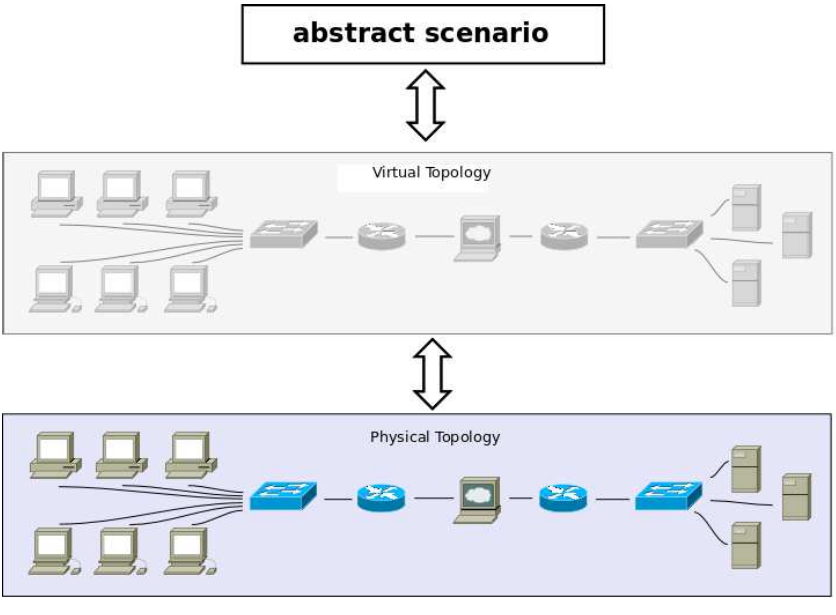

Fig. 3. dosTF Abstract Scenario

3.1 Monitoring

The monitoring of a machine under a DDoS attack involves an obvious and serious problem: how can an application respond promptly with information about its status, when it is already under attack? Even in severe attacks, however, the multi-tasking design of modern operating systems should allow enough responsiveness in the overall system to enable the gathering of basic statistics at regular intervals. The saturation of the attack network can then be easily dealt with by installing a second Ethernet card to use as the monitoring interface. This is also a feature of the DETER Testbed [12]. The advantage of using live feedback during an attack is that it becomes possible to see the performance of service applications as they buckle under the strain imposed by a DDoS attack, or recover as defensive measures are engaged.

The method we chose for live monitoring was to install an SNMP (Simple Network Management Protocol) service on each potential target. We could then query the target for a wealth of built-in MIB (Management Information Base) variables such as, for example, tcpOutRsts (requests to resend a TCP segment). Such information has sometimes been used to detect the presence of a DDoS attack [15]. However, these system-wide values are less useful when monitoring the effect of an attack against a single service. According to Mircovic *et al.* [16] Denial of Service can be effectively measured by monitoring only a select few application-specific parameters: chiefly memory and CPU usage, as well as responsiveness and goodput (the amount of data actually being received and sent out by the application).

Another drawback with the standard SNMP installation is that per-second monitoring of network-related MIB-values tends to tie up the CPU. In our case we observed 30% CPU utilisation with the default Linux SNMP agent running, when querying IF-MIB variables.

Our solution was to write a small custom MIB that would discreetly measure the following parameters on a per-second basis for any named service:

1. Percentage of system memory being used
2. Percentage of CPU being used
3. Number of active threads or forked children of a process
4. Response time in milliseconds to a generic query
5. Goodput - the actual data throughput of the service
6. Does the response to a given challenge match the expected value?

This is all we currently measure, but the custom MIB can be extended at any time. It is written as a separate agent that can be brought up or down without disruption to the main SNMP agent. Parameters 1, 2 and 3 can be measured by system commands that take only a few milliseconds to run.

Response time is measured in nanoseconds, up to a maximum of 15 secs, of a named service to a given challenge string, which usually consists of binary digits. The values for HTTP, TELNET, FTP, SSH and DNS query generic properties of their respective services, e.g. the HTTP challenge merely requests the server's options, and the DNS challenge requests the service's status. But it is also possible to override the default challenge, to define new services, to change ports and protocols *etc*.

Goodput can be computed without modification of the service, at least in the case of Linux, by using the Systemtap tool, or by modifying the kernel. This may be preferable to 'instrumenting' the service, *i.e.* by modifying it [16].

The advantage of using SNMP is that existing software libraries for querying and setting values, as well as command-line tools can be used. We also envisage that using these specific MIB values, rather than the general ones, may provide a more accurate way for an alarm system to *detect* Denial of Service.

3.2 The Scenario

An experimenter needs to specify what form an attack will take, and to save that information so it can be edited and replayed later. DETER uses Tcl largely for historical reasons [9], but this is a programming language with a fixed syntax, and is not ideally suited to the recording of an abstract experiment. XML [17], on the other hand, is a widely used markup language suitable for a variety of programming tasks. Many tools for reading and writing XML files already exist, and changing the scenario schema or structure in response to design changes is easy. Our schema contains the following basic elements:

Agents: may be one of attacker, traffic generator, defender or service. These are programs that can be launched from the command line. Each agent is specified by a set of runtime parameters, the system requirements, the number of hosts it should be copied to, and the type of hosts required (e.g. real or virtual, and desired operating system). The control application (described below) will then choose an appropriate binary to copy to the specified host.

Targets: There may be more than one, and each is specified simply by an operating system type and optionally by an IP-address. This latter facility is needed because otherwise the default choice of target may not be what is desired.

Views: These describe the layout of portlet windows in the testbed software, to be described below. This section is entirely optional, but otherwise there would no way to save the screen layout of the tools used for monitoring a particular experiment.

3.3 Command and Control

The experimental scenario described above needs to be activated by some means. Agents will first have to be copied to their assigned targets. This is achieved in dosTF via SFTP. This can be configured so that the payload will only be copied to the targeted host if the current file is more recent.

The attack command is then issued by the control workstation via ssh, using a standard username and password. Since the interface is simply the commandline, this allows us to leverage most existing tools. For example, the traffic generator D-ITG [18] can be easily configured within the scenario to generate various types of traffic.

In the third phase the experiment is monitored in real time by the control workstation via SNMP. The results are displayed on the workstation as a series of graphs and the values are logged to local files automatically.

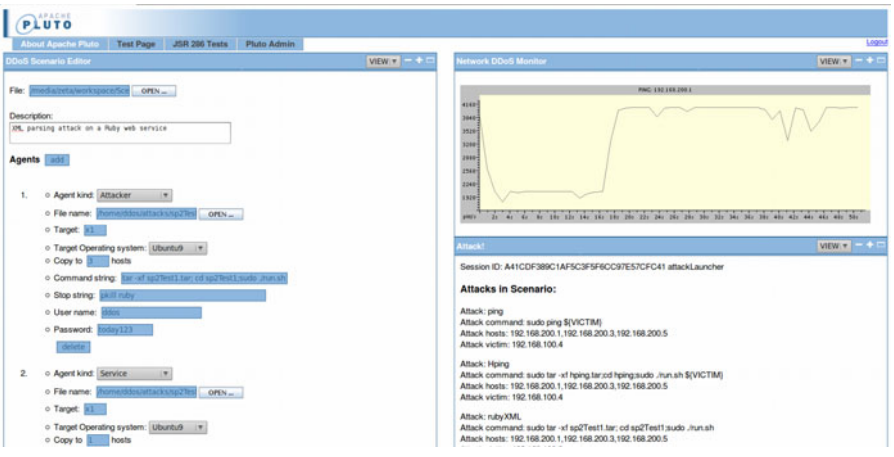

Fig. 4. Pluto Portal Interface

3.4 Web Interface

DETER uses a desktop based GUI application to monitor and control experiments, and RPC for communications [13]. This introduces some complications, since user authentication has to be programmed in, and the application must be available on the computer connecting to the testbed. GUI development is inherently complex and expensive [19]. Changes to the functionality of the underlying program can cause expensive alterations to the GUI. This could easily lead to the GUI taking more time to develop than the underlying functionality.

Our solution to this problem was to use a modular web application interface for GUI development, such as a portal server. The GUI is accessed via an ordinary web browser either remotely or locally. The web interface is divided into a number of sub-windows, or portlets, which can be installed and arranged, or duplicated by the user to suit the experiment. Authentication is already built-in, as also (on some servers) user filespace management for saving experimental results. Fig. 4 shows the current state of the interface, and includes three portlets, one which edits the XML scenario file, another on the right that launches attacks, and a monitor portlet that graphs the chosen SNMP MIB variables and also logs them to disk.

4 Experimental Results

The dosTF testbed is designed to run experiments for any of the subprojects listed in Section 1. For example, Subproject 4 investigates DoS vulnerabilities in emerging technologies such as web services. This section describes two experiments from this area of the overall project as examples of simulated DoS attacks carried out on the

testbed. Each experiment starts as an XML scenario, which is then executed, causing the attack program to be copied to its respective hosts, then the attack was launched, and the results monitored using the custom SNMP MIB parameters.

4.1 Experiment 1

The first experiment exploits a vulnerability in the Ruby XML parser. The attack uses an invalid web service request payload containing a *deeply-nested* meaningless XML message (up to 100,000-levels deep), and then sends a flood of such requests to the Ruby server. The payload size is around 1.5 MB. A vulnerable XML parser will try to load each of the XML messages sent. The goal is to consume all of the memory available on the server, causing a potential denial of service to legitimate clients.

The single attack and victim machines are physical hosts on the same subnet as shown in Fig. 1. The victim server's resources include a dual-core 3 GHz CPU with 3.7 GB of memory. The SNMP monitoring provided by the dosTF testbed was used to track memory usage and CPU usage throughout the attack and after. The attack itself lasted for about 10 minutes. Requests were sent in bursts of 500, followed by a sleep of 0.1 of a second. The result of the experiment is shown in Fig. 5. Although the CPU usage quickly reached a maximum, the memory usage took longer to become exhausted, eventually causing the Web Service to fail, before restarting.

4.2 Experiment 2

The second experiment exploits a vulnerability in many web services that respond to unauthenticated requests for their service description files, or WSDL documents. This

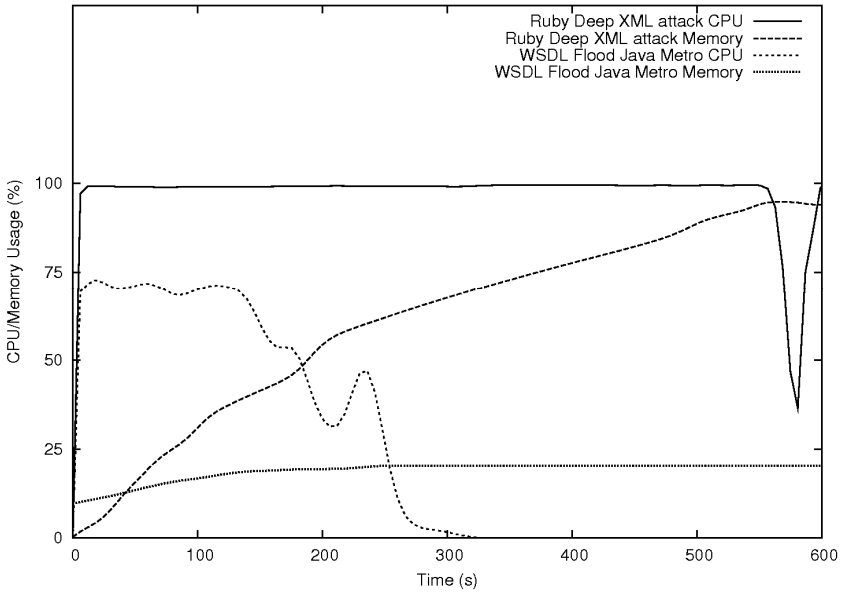

Fig. 5. CPU and Memory Usage for Experiments 1 and 2

experiment tested the effect of performing repeated requests for a WSDL document on a web service developed using the Java Metro library, and deployed on the Glassfish application server. By default, the WSDL document will be dynamically generated, requiring some processing by the server. Successive requests may thus have a significant impact on CPU and memory consumption, and on the response time for legitimate web service requests.

The three attack machines were real hosts, and the victim was a virtual host, as shown in Fig. 1, but without the intervening router. The victim machine's resources included a dual-core 4.8 GHz CPU and 3 GB of memory. Fig. 5 shows the application server response as tracked via the SNMP monitoring provided by the testbed. The attack lasted roughly 4 minutes, consisting of 250 bursts of 150 requests per machine. In contrast with the first experiment, memory consumption was virtually unchanged. After the attack CPU consumption soon fell back to normal levels.

5 Future Developments

As well as being a general DoS testing facility, the dosTF testbed can also provide a flexible framework for carrying out experiments involving specific devices. Subproject 1, for example, aims to develop a 'network flooding attack mitigation tool' capable of protecting security devices, such as an application-aware firewall. As shown in Fig. 6, the device will monitor the state of the firewall and, once a high-rate flooding attack is detected, will initiate corrective action to mitigate the impact of the attack.

Another application of the testbed will be the testing of vulnerabilities introduced by the use of the IPv6 protocol in an emulated SCADA network controlling a set of distributed resources, similar to those encountered in the monitoring and control of critical infrastructure, such as in the electricity and water utilities. The aim is to study the behaviour of such SCADA systems when known IPv6 vulnerabilities are exploited and to evaluate the effectiveness of potential mitigation techniques.

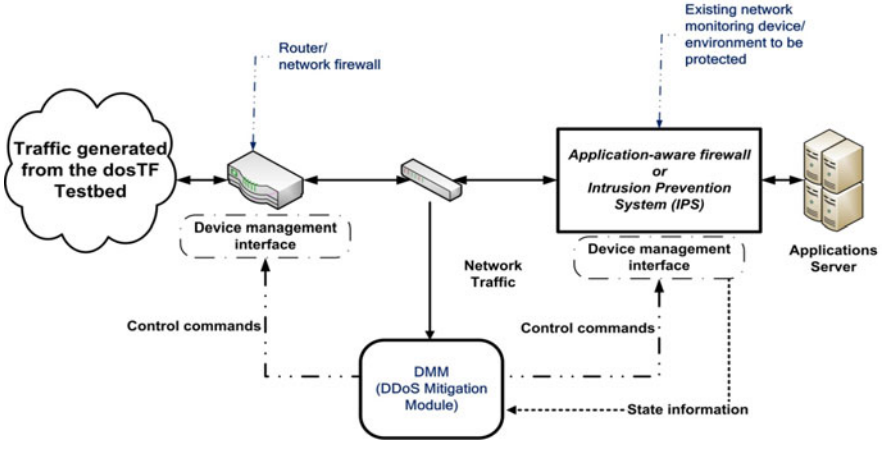

Fig. 6. DDoS Mitigation Module

6 Conclusion

The dosTF testbed is designed to provide an abstract means for specifying an experiment that can be run unchanged or with insignificant changes on various physical network topologies. By dispensing with the need to construct a virtual topology within the physical layout of a dedicated testbed it enables experiments involving new network appliances. This design, being simpler than the DETER model, allows for the construction and management of a private DoS/DDoS test facility at minimal cost, with some sacrifice in ease of use.

This research is supported by the Australia-India Strategic Research Fund.

References

1. Leyden, J.: Conficker botnet growth slows at 10m infections. The Register 26/1/09 (2009), http://www.theregister.co.uk/2009/01/26/conficker_botnet/
2. Chapman, S.: Massive 2 Million PCs Botnet Uncovered. Computerworld, UK, 23/4/09 (2009), http://www.cio.com/article/490454/Massive_2_Million_PCs_Botnet_Uncovered
3. Van Buskirk, E.: Facebook Confirms Denial-of-Service Attack. Wired, 6/8/09 (2009), http://www.wired.com/epicenter/2009/08/facebook-apparently-attacked-in-addition-to-twitter/
4. Nakashima, E., Krebs, B., Harden, B.: U.S., South Korea Targeted in Swarm Of Internet Attacks. The Washington Post, 9/7/09 (2009), http://www.washingtonpost.com/wp-dyn/content/article/2009/07/08/AR2009070800066.html
5. Worldwide Infrastructure Security Report. Arbor Networks (2009), http://www.arbornetworks.com/report
6. White, B., Stoller, L., Ricci, R., Guruprasad, S., Newbold, M., Hibler, M., Barb, C., Joglekar, A.: An Integrated Experimental Environment for Distributed Systems and Networks. In: Proc. of the Fifth Symposium on Operating Systems Design and Implementation, Boston, MA, pp. 255–270 (2002)
7. Lam, H.-Y., Li, C.-P., Chanson, S.T., Yeung, D.-Y.: A Coordinated Detection and Response Scheme for Distributed Denial-of-Service Attacks. In: IEEE International Conference on Communications, ICC 2006, vol. 5, pp. 2165–2170 (2006)
8. Blackert, W.J., Gregg, D.M., Castner, A.K., Kyle, E.M., Hom, R.L., Jokerst, R.M.: Analyzing Interaction Between Distributed Denial of Service Attacks And Mitigation Technologies. In: Proceedings of DARPA Information Survivability Conference and Exposition, April 22-24, vol. 1, pp. 26–36 (2003)
9. Chertov, R., Fahmy, S., Shroff, N.B.: Emulation versus Simulation: A Case Study of TCP-Targeted Denial of Service Attacks. In: Proceedings of 2nd International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities. TRIDENTCOM, pp. 315–325 (March 2006)
10. Mircovic, J., Fahmy, S., Reiher, P., Thomas, R.K.: How to Test DoS Defences. In: Proceedings of the Cybersecurity Applications & Technology Conference For Homeland Security, CATCH (2009)
11. Peterson, L., Bavier, A., Fiuczynski, M.E., Muir, S.: Experiences Building PlanetLab. In: OSDI 2006: 7th USENIX Symposium on Operating Systems Design and Implementation, pp. 351–366 (2006)

12. Benzel, T., Braden, R., Kim, D., Neuman, C., Joseph, A., Sklower, K.: Experience with Deter: A Testbed for Security Research. In: Proceedings of 2nd International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities. TRIDENTCOM, pp. 378–388 (March 2006)
13. Mircovic, J., Reiher, P., Thomas, R., Schwab, S.: Automating DDoS Experimentation. In: Proceedings of the DETER Workshop (August 2007)
14. Bolla, R., Bruschi, R.: RFC 2544 Performance Evaluation for a Linux Based Open Router. In: Proc. of the 2006 IEEE Workshop on High Performance Switching and Routing (HPSR 2006), Poznan, Poland, pp. 9–14 (June 2006)
15. Yu, J., Lee, H., Kim, M.-S., Park, D.: Traffic flooding attack detection with SNMP MIB using SVM. Computer Communications 31, 4212–4219 (2008)
16. Mircovic, J., Hussain, A., Fahmy, S., Reiher, P., Thomas, R.K.: Accurately Measuring Denial of Service in Simulation and Testbed Experiments. IEEE Transactions on Dependable and Secure Computing 6(2), 81–95 (2009)
17. Bray, T., Paoli, J., Sperberg-McQueen, C.M., Maler, E., Yergeau, F.: Extensible Markup Language (XML) 1.0, 5 edn. (2008), http://www.w3c.org/TR/REC-xml
18. Botta, A., Dainotti, A., Pescapè, A.: Multi-protocol and multi-platform traffic generation and measurement. In: INFOCOM 2007 DEMO Session, Anchorage (May 2007), http://www.grid.unina.it/software/ITG/
19. Shneidermann, B.: Designing the User Interface: Strategies for Effective Human-Computer Interaction, 3rd edn. Addison-Wesley, Reading (1998)