An Approach for Mitigating Denial of Service Attack

Chetan Chauhan

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, Odisha, India

An Approach for Mitigating Denial of Service Attack

Thesis submitted in
May 2013
to the department of
Computer Science and Engineering
of
National Institute of Technology Rourkela
in partial fulfillment of the requirements
for the degree of

Bachelor of Technology

in

Computer Science and Engineering

by

Chetan Chauhan
[Roll: 109CS0069]

with the supervision of

Prof. Manmath Narayan Sahoo

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, Odisha, India

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, Odisha, India.

May 21, 2013

Certificate

This is to certify that the work in the thesis entitled ***An Approach for Mitigating Denial of Service Attack*** by ***Chetan Chauhan*** is a record of an original work carried out under my supervision and guidance in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering. Neither this thesis nor any part of it has been submitted for any degree or academic award elsewhere.

Prof.Manmath Narayan Sahoo
Assistant Professor
Department of Computer Science and Engineering
NIT Rourkela

Acknowledgment

My first thanks are to the Almighty, without whose blessings I wouldnt have been writing this acknowledgments. I then would like to express my heartfelt thanks to my supervisor, Prof. Manmath Narayan Sahoo, for his guidance, support, and encouragement during the course of my bachelor study at the National Institute of Technology, Rourkela. I am especially indebted to him for teaching me both research and writing skills, which have been proven beneficial for my current research and future career. Without his endless efforts, knowledge, patience, and answers to my numerous questions, valuable suggestions without which the successful completion of this project would not have been possible. The experimental methods and results presented in this thesis have been influenced by him in one way or the other. It has been a great honor and pleasure for me to do research under Prof. Manmath supervision.

Further i am also thankful to all the faculty members and staff of Department of Computer Science and Engineering, National Institute of Technology, Rourkela for their constant help and support during the entire project work.

Finally, I want to dedicate this thesis to my parents, my family members and my beloved one for their unlimited support and strength. Without their dedication and dependability, I could not have pursued my BTech. degree at the National Institute of Technology Rourkela.

Last but not the least I would like to thank for the constant motivation of my classmates who in one way or another have helped us in the successful completion of this work.

Chetan Chauhan

Abstract

Denial of Service or Distributed Denial of Service attacks are the most common types of cyber-attack on the internet and are rapidly increasing. Two general form of Dos attacks are - those attacks that crashes services (computer attack) and those that flood services (network attack).Flooding DDoS attacks produce adverse effects for critical infrastructure availability, integrity and confidentiality. Current defense approaches cannot efficiently detect and filter out the attack traffic in real time. Based on the assumption that the attacker flows are very aggressive than the legitimate users, the proposed work provides limit to the rate of the burst of packets and provides sufficient bandwidth to genuine users during flooding DoS attack. In this thesis, we have followed an approach for mitigating DoS/DDoS attack based on The Interface Based Rate Limiting (IBRL) algorithm, applied on Interfaces at the server-side, used to mitigate the identified DoS attacks. The implementation is carried out on a simulation tool Omnet++ installed on linux machine. The results show that there is considerable decrease in the two host and network based performance metrics that are Packet drop and Response time under DoS and DDoS attacks. We also analyzed our approach in a simulation environment against one existing end system based mitigation strategy.

Contents

List of Figures

List of Abbreviations

DoS: Denial of Service attack

DDoS: Distributed Denial of Service attack

LACC: Local Aggregate-based Congestion Control

AITF: Active Internet Traffic Filtering mechanism

IBRL: Interface Based Rate limiting algorithm

DefCOM: Defensive Cooperative Overlay Mesh

PD: Packet Drop

RT: Response Time

IP:Internet Protocol

ICMP:Internet Control Message Protocol

UDP: User Datagram Protocol

–

Chapter 1

Introduction

Introduction

DDoS Impact

Problem Addressed

Design Approach

1.1 Introduction

Before computers were there, the information was stored in physical files. As any other asset, information has significant value and required to be secured from its ill use. The files are required to have three properties confidentiality, integrity, and availability. In the current age of technology, information become electronic and stored in computers and new age devices. But these three requirements for security does not changed. They remained in demand. To be secured, information needs to be hidden from unauthorized access (confidentiality), protected from unauthorized changes (integrity), and available to an authorized entity when it is needed (availability)[1]. The implementation of these requirements, however, are challenging and difficult. And led to the creation of new field of security and cryptography techniques under computer science technology.

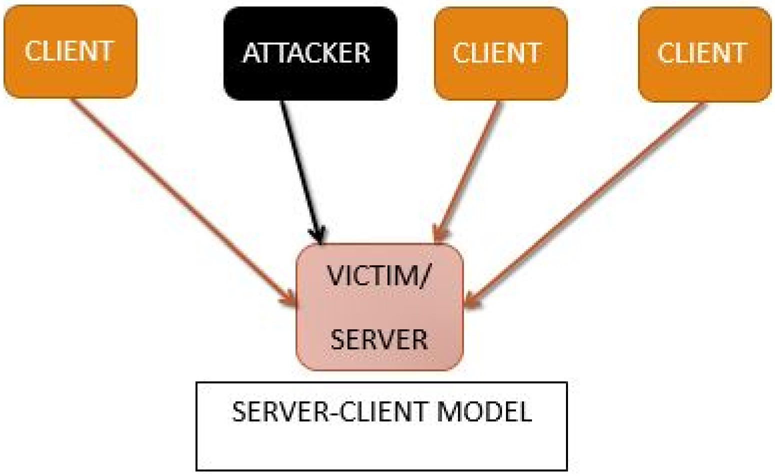


FIGURE 1.1: Denial of Service attack with server-client scenario

A Denial of Service (DoS) attack can be characterized as an attack with the purpose of preventing legitimate users from using a victim computing system or network resource. A Distributed Denial of Service (DDoS) attack is a large scale, coordinated attack on the availability of services of a victim system or network resource, launched indirectly through many compromised computers on the Internet. The machine whose services are under attack is normally called as victim (primary victim), while the compromised systems used to launch the attack are often called the Zombies(secondary victims)[2]. In some cases these zombies machines are unaware of the accomplishment of attack. The use of secondary victims in performing a DDoS attack provides the attacker with the ability to wage a much larger and more disruptive attack, while making it more difficult to track down the original attacker.

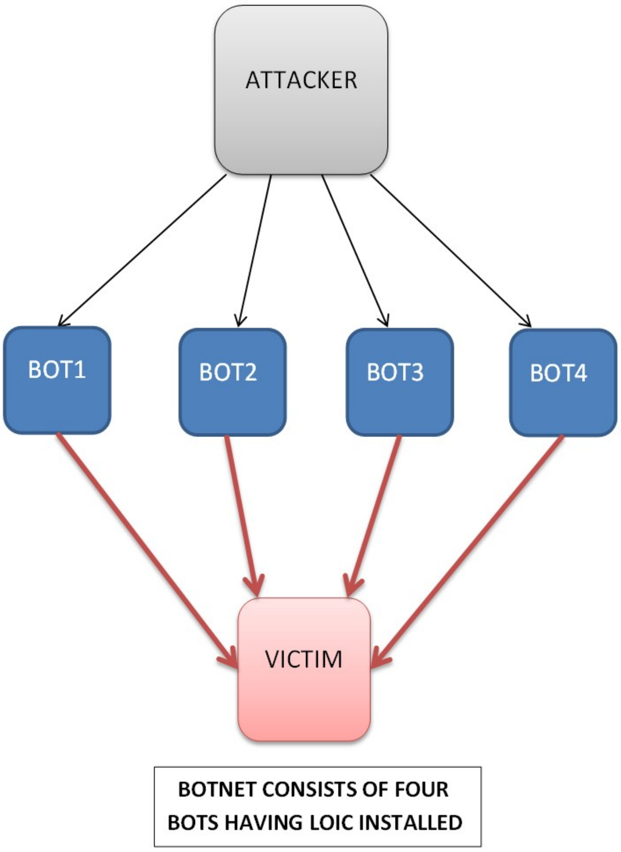


Figure 1.2: Distributed Denial of Service attack four zombies or bots controlled by one attacker

A DDoS [3]attack not only attack on single target victim but also many simultaneously.The attack can accomplished on both wired networks as well as wireless sensor networks.A wireless Sensor Network is a collection of nodes deployed mostly in the range of hundreds to thousands connected to the same network. Each node has its own processing capability, memory, power source and sensors. The nodes are designed in such a manner that they can communicate and organize themselves through the network. The nodes have a wide range of cost depending on the need. The concept of WSN is getting popular day by day. WSN is mostly used to monitor environment and physical conditions.

DDoS attack is a threat to the availability which is one of the security goals. As Distributed Denial of Service (DDoS) attack will make the system not to be available for the legitimate purpose. System under attack is unable to provide legitimate user with its service. This attack may slow down or totally interrupt the service of a system. There are various strategies used by the attackers to accomplish this kind of attack. Even attacker has developed their own tools to attack server in different ways. There are many tools emerging every season. LOIC, HOIC, SLOWLORIS, PYLORIS, TORSHAMMER and HULK are some of the tools used last few years. All of which follow different strategies to accomplish attack. This is one of the main reasons that why DoS/DDoS attack is so dangerous. It might send so many bogus requests to a server that the server crashes because of the heavy load. The attacker may also intercept requests from the clients, causing the clients to send requests many times and overload the system. The attacker might intercept and delete a servers response to a client, making the client to believe that the server is not responding.

1.2 DDoS Impact

Denial-of-service (DoS) attack in two different forms[4] (1) exploiting bugs in network clients or server applications which is an attempt to crash the application or host on which application is running or (2) flooding a network server with fake traffic due to which server is unable or it becomes difficult to server to receive and process legitimate traffic. The former typically are buffer overrun attacks in which a large amount of data packets requests are sent to the network application which it unable to handle properly with respective reply. At the server side, in main memory a block of memory is assigned an automata state which changes its state only when server reply with respective acknowledgement. If server is unable to send ack signal for some requests in high traffic scenario then the automata states of memory block piled up as unable to change states. Huge amount of memory is being wasted and in-turn memory and processor crashes.

Hacker targets those companies which do not take security seriously enough and their systems, in general, are easily compromised. They become threat, not only to themselves but also to other targeted companies through their systems. Using secure operating systems such as Unix - nginx which offer process protection (to prevent an application crash from crashing the whole system), with security patches are kept up-to-date and vulnerability alerts (to avoid running applications which are vulnerable to buffer overrun attacks), and controls and monitors network

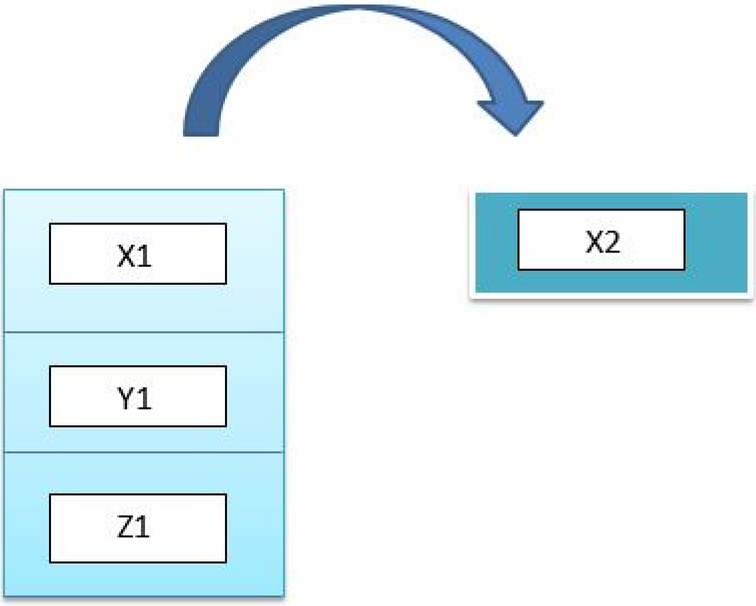


Figure 1.3: Automata states of main memory blocks at server side

traffic (for flood attacks) DoS attacks can be defended. DDoS attacks are a variation of DoS type attack. A DDoS attack uses hundreds/thousands of machines simultaneously, leads to flooding at higher scale which makes it harder to defend against. And also its identification is difficult because rather than appearing as an burst of traffic from a single machine/node, a DDoS attack appears instead as normal traffic from a large number of machines in network or outside network.

It is very tedious to trace back to origin of the DDoS attack as thousands of compromised machines are involved in attack. In this condition, it is enormous to stemming the attack with logistical problems and identifying its real origin. Attackers use address spoofing within the attacking tool to spoof the attack that is, fake source addresses are used in the packets that they send to victim. This makes tracing not possible. Tracing back involves examining the logs of all of the intermediate routers, one by one, to trace the packets back one hop at a time. It is no surprise that no harrests have yet been made for these attacks. These attacks does not has any comprehensive defense against them[4]. A real solution would involve re-engineering the entire network architecture as the Internet was simply not designed with these vulnerabilities in mind. This means, to take pre-emptive measures to reduce the possibility of these attacks and minimize their impact is critical.

Password cracking and buffer overruns are the two methods to prevent our system from becoming compromised systems. Preventing address spoofing can also help to make the origins of attacks harder to conceal[4]. Ingress and egress packet filtering should be used on firewalls or routers to prevent packets with spoofed addresses from crossing these boundaries. Dictionary-based attack programs are common, so single word passwords are not a good choice. Passwords should contain a mix of letters and digits or punctuation, be hard to guess, kept secure and changed regularly. Buffer overrun attacks used to compromise systems are a more sophisticated variation of the DDoS buffer overrun attacks[4], in which the excess data is carefully constructed to be meaningful instructions which are then executed by the host under attack. To protect against these, you need to use servers that have been well written and carefully audited for such bugs. A firewall can often provide a good defense here, if your servers are located behind it. A good firewall will have all buffer operations carefully controlled, and will identify, log and handle overrun attempts. A firewall can also issue alerts, for example by sending SMS messages to mobile phones, which will allow countermeasures to be taken as early as possible. Routing information at the various routers available in the route has good possibilities to have a DDoS attack. The routing table can be maintained according to the attackers will, it is done when the packets entering into router, contain some codes and ongoing attack becomes impossible to catch.

Basically, DDoS attacks make the network components busy so that these cannot respond to the legitimate requests. Therefore, the conclusion of the discussion is the impact of this attack is on three components of the system: Disruption of configuration information, such as routing information, Disruption of physical network components, Consumption of computational resource, such as bandwidth, disk space, or CPU time.

1.2.1 Problem Addressed

From the above discussion it was assessed that how much dangerous the DDoS attack is. In two different ways this attack hampers the server. It hampers in terms of resources available at server side and the bandwidth of the channel used to avail the service from the server. Many solutions to solve the problem to mitigate the attack from hampering the resources at server have been proposed. Amongst them so many are proposed for saving the server from being attacked is of having the support from network. Implementation of a mitigation or controlling technique for denial of service attack so that system (server) works properly or is able to provide service to its intended users even under Denial of Service Attack.

1.2.2 Design Approach

The algorithm proposed is based on interfaces at the server side through which traffic is passed prior to entering into server. The algorithm applies rate limiting rules to a particular interface. In this way by controlling in coming burst of packets with lesser packet drop and lesser response time, the effect of denial of service attack decreased to good extent.

Chapter 2

Related Work

- Local Aggregate-based Congestion Control (LACC)
- Cooperative pushback mechanism
- Adaptive throttle algorithm
- Active Internet Traffic Filtering (AITF) mechanism
- DefCOM
- Traceback-based rate limit algorithm.

The reason for the development of DoS/DDoS defense mechanisms is increasing vulnerability and recurrence of DDoS attack. All these mechanisms address to a specific type of DDoS attack. These are attacks on Web servers or authentication servers. Most of the proposed approaches require certain features to achieve their peak performance. All these approaches perform quite differently if deployed in an environment where these requirements are not met.It is very important to understand that in what way all these approaches can be combined together and can efficiently solve the problems.

2.1 Local Aggregate-based Congestion Control (LACC)

As not all traffic going to a server under attack is malicious, ACC aims at protecting the innocent traffic within the aggregate when the high-bandwidth aggregate is malicious[5]. An aggregate is a collection of packets sharing a common property. some of the examples of an aggregate are all packets with a given source prefix, and all ICMP ECHO packets destined for a particular address.

Aggregate-based congestion control (ACC) [6], which works at a different granularity - that of an aggregate. The primary goal of ACC is to protect the network and the rest of the traffic from severe congestion caused by high-bandwidth aggregates. The level of congestion is monitored by a router implementing ACC. Using either drop history or random samples, the router tries to identify the aggregate responsible for it after detecting congestion. Properties considered are source and destination prefixes. The identified aggregates are rate-limited to a level that is dynamically decided based on the arrival rate of non-rate-limited aggregates, and the congestion level at the router. This is done such that the aggregate is not punished too harshly, while significantly reducing the drop rate at the congested router.

2.2 Cooperative pushback mechanism

LACC can be supplemented at the routers with a cooperative ACC mechanism called pushback. Using pushback, the congested router can request its upstream routers to rate-limit the aggregate on its behalf. Pushback can be recursively propagated further upstream.

Pushback has two advantages in addition to those of LACC. First, by taking rate-limiting upstream, pushback reduces bandwidth consumption of packets that would eventually be dropped downstream. Second, and more important, pushback can help focus rate-limiting on traffic coming from directions that are more likely to be pumping in malicious traffic. This can be achieved by intelligently computing the rate-limits sent upstream (can be different for different upstream routers), and would protect the innocent traffic in the aggregate specification. Looking at issues like implementation complexity (a FreeBSD prototype implementation is also in progress), incremental deployment of pushback, policy issues, attack topologies (utility of pushback depends on it), and finer time-scale effects of LACC.

2.3 Adaptive throttle algorithm

Adaptive throttle algorithm [7] protects a server from resource overload and also increases the ability of legitimate traffic to its intended server. Even though server-centric router throttling is a promising approach to counter DDoS attacks, non-trivial challenges prevent immediate deployment in the Internet. The approach is proactive: Before aggressive traffic flows can converge to overwhelm a server, a subset of routers along the forwarding paths is activated to regulate the incoming traffic rates1 to more moderate levels. The basic mechanism is for a server under attack, say S, to install a router throttle at a set of upstream routers several hops away. The throttle limits the rate at which packets destined for S can be forwarded by a router. Traffic that exceeds the throttle rate will be dropped at the router.

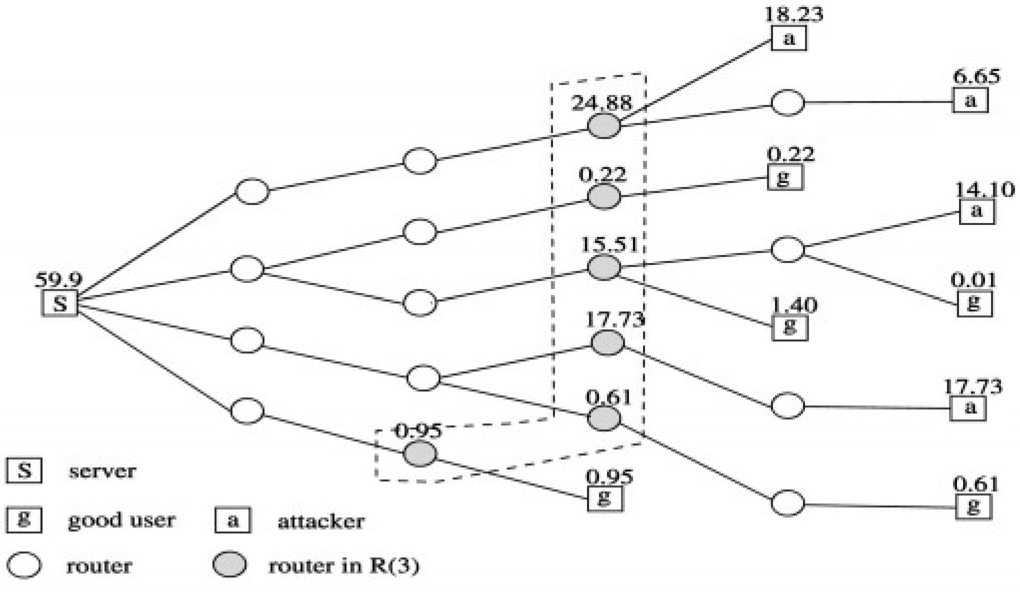


FIGURE 2.1: Adaptive throttle mitigation mechanism

2.4 IP Traceback-Based Intelligent Packet Filtering

This methodology proposes to leverage the attack graph information obtained through IP Traceback to preferentially filter out packets that are more likely to come from attackers.

IP Traceback-Based Intelligent Packet Filtering [8] a protocol-independent DDoS defense scheme that is able to dramatically improve the throughput of legitimate traffic during a DDoS attack. It works b performing smart filtering: dropping DDoS traffic with high probability while allowing most of the legitimate traffic to go through. This clearly requires the victim to be able to statistically distinguish legitimate traffic from DDoS traffic. The proposed scheme leverage on and extends IP traceback techniques to gather intelligence: information such as whether or not a network edge is on the path from an attacker. By preferentially filtering out packets that are inscribed with the mark of an infected edge, the proposed scheme filters out most of the traffic from attackers since each and every edge on an attackers path to the victim is infected. Packets from a legitimate client, on the other hand, with high probability will not be filtered out, since, typically, most of the edges on the clients path to the victim are not infected[9]. To evaluate its effectiveness in defending against DDoS attacks, the proposed scheme

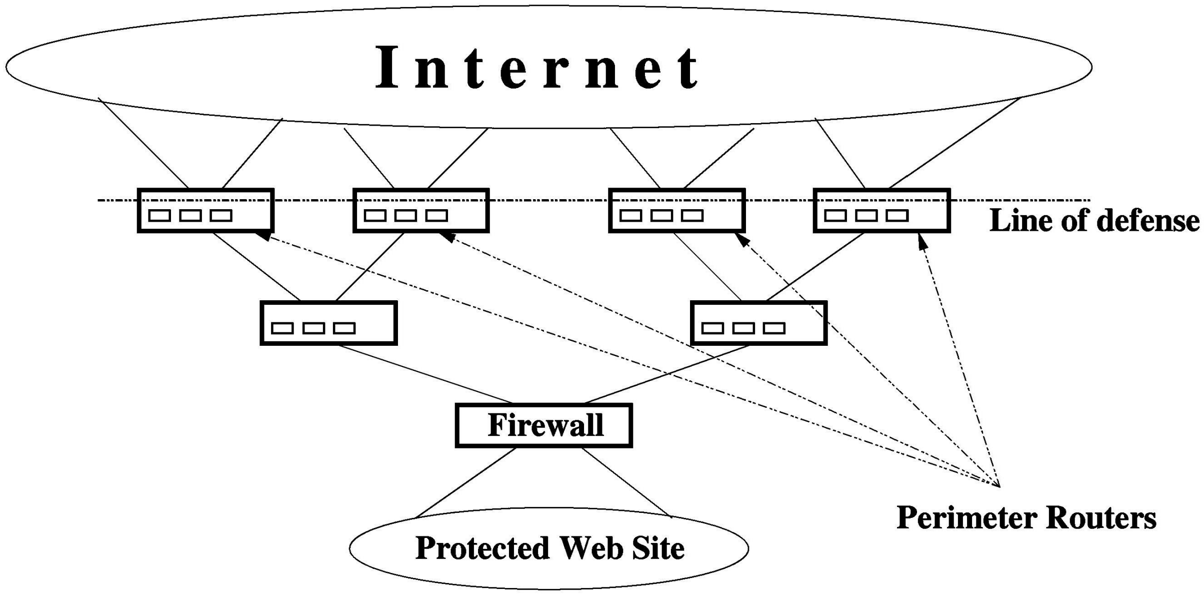


FIGURE 2.2: Implementation of IP Traceback-Based Intelligent Packet Filtering mechanism

is simulated on three sets of real-world Internet topologies with varying operating parameters. Simulation results demonstrate that the throughput of the legitimate traffic can be increased by three to seven times.

2.5 Active Internet Traffic Filtering (AITF) mechanism

Our main contribution is Active Internet Traffic Filtering (AITF) [10], a protocol that leverages recorded route information to block attack traffic. An AITF-enabled receiver uses the route recorded on incoming packets to identify the last point of trust on each attack path and causes attack traffic to be blocked at that point, i.e., as close as possible to its sources. We provide a way to do this securely. AITF prevents abuse by malicious nodes seeking to disrupt other nodes' communications. We show that our approach can selectively block a million attack sources, yet requires only tens of thousands of TCAM memory entries and a few megabytes of DRAM memory from each participating router; these numbers correspond to the specifications of real products [11][12]. We also provide an incremental deployment scenario, in which even early adopters receive a concrete benefit; this benefit is compounded by further deployment.

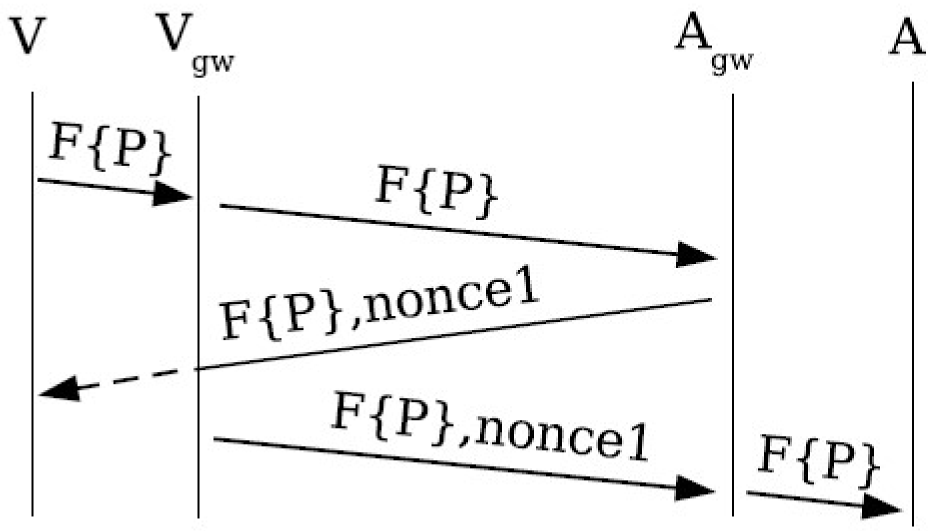


Figure 2.3: AITF entities and message exchange

Vgw proves its location on the path to V by intercepting the nonce sent to V. This prevents malicious node M, located off the path from Agw to Vgw, from causing alter to be installed at Agw and block traffic to V. By picking a sufficiently large and properly random value for the nonce, it can be made arbitrarily difficult for M to guess it.

2.6 DefCOM

DefCOM is a distributed cooperative system for DDoS defense through a flat overlay to detect and stop attacks. Its design has an economic model where networks deploying defense nodes directly benefit from their operation. But how to authenticate and establish economic cooperative relationship across different management domains has not been described clearly. DefCOM leverages the classifier nodes near attack sources to differentiate between legitimate and attack packets. However, if attack packets have no distinct signature, then classifier nodes will not work. Unfortunately, the flooding-style attack traffic produced by modern attack tools usually has no distinct characteristics at the attack source end.

Our system, called DefCOM (Defensive Cooperative Overlay Mesh) [13], deploys defense nodes distributed in the Internet core and through the edge networks. All nodes form a peer-to-peer overlay to securely exchange attack-related messages.

When an attack occurs, nodes close to the victim detect this and alert the rest of the DefCOM overlay. Core nodes and those in vicinity of attack sources then suppress the attack traffic through coordinated rate limiting. Source nodes are also tasked with traffic profiling, making sure that their share of limited bandwidth is fully dedicated to traffic they deem legitimate or important. They further work in concert with core nodes to ensure that this legitimate traffic is safely delivered to the victim The novel contribution of DefCOM is that legitimate clients of this network can also achieve DDoS attack transparency and reach the victim anytime if they deploy a classifier node in their network.

2.7 Traceback-based rate limit algorithm

A meek DDoS attack is considered that is finding out the subtle difference between attackers and legitimate users and provided an elaborate IP Traceback-based rate limit algorithm [14]. The bottleneck link bandwidth between the victim and the last hop router is assumed. Also it is assumed that the available bandwidth is abundant for legitimate traffics. The DDoS rate limiting mechanisms described above is based upon classifying the attack packets and legitimate packets and stored as log files and then offline analysis is carried out. In some mechanisms, the legitimate packets drop rate is high during rate limiting. In order to overcome the above said drawbacks online monitoring is proposed in this paper. It classifies legitimate and illegitimate packets, considering the network performance metrics. Then rate limiting is applied to the attack traffic to mitigate the vulnerability of the attack.

An IP traceback-based rate limit algorithm, which can not only mitigate the DDoS attack effect as close to the attack source end as possible, but also improve the throughput of legitimate traffic more effectively than MaxMin based rate limit algorithm even under a meek attack.

IP traceback technique allows the victim to identify the sources of DoS or DDoS attacks even in the presence of IP spoofing. Although IP traceback itself could not mitigate attack effect, it can assist other countermeasures such as router throttles to defeat attacks in the best places. Since DDoS attacks came forth, a number of IP traceback approaches have been proposed, such as link testing, ICMP-based iTrace, probabilistic packet marking (PPM) and so on. The detail

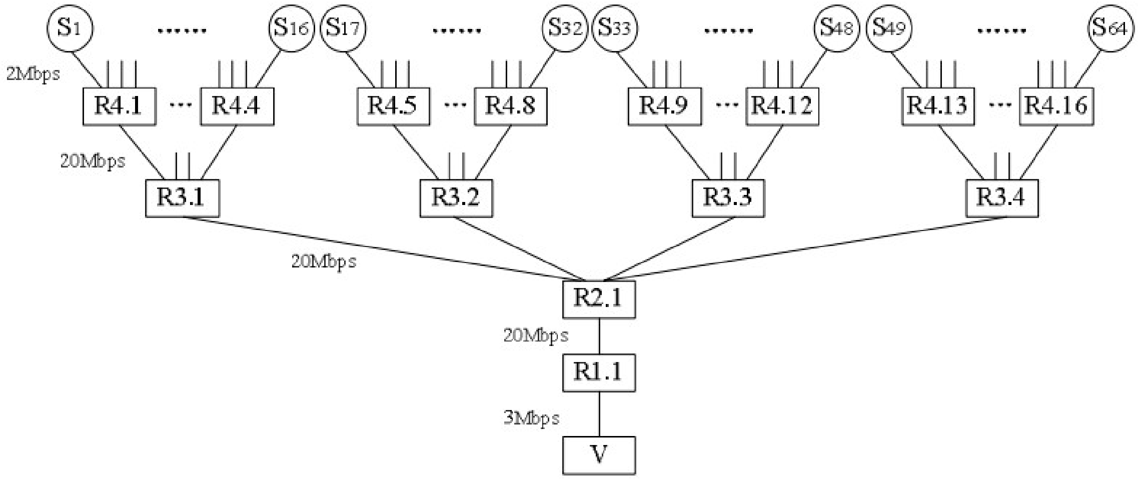


Figure 2.4: Traceback-based rate limit Mechanism

of IP traceback process is out of the scope of this paper. In this paper, we focus on how to use IP traceback results to improve the performance of rate limiting. By leveraging one concrete IP traceback technique, the victim could reconstruct a traffic tree rooted itself and identify the attack sub-tree. The leaf nodes of the traffic tree are the best place to rate limit. Hence, i could install throttles at those leaf nodes and allocate the victims limited resource among those throttles. Unless otherwise noted, we assume the victims limited resource is the bandwidth of the bottleneck link between the victim and the last hop router.

Chapter 3

Motivation

DoS/DDoS is a dangerous weapon which harm cyber world in various aspects. By extortion or financial gain through damaging a brand of competitor, or by raising ones profile in the hacker community, DDoS proves its usage in creating mess is vast. It is not only related to cyber world but also to social world. Its usage became recently a weapon to take revenge by disrupting an organisations or indeed a countrys. For critical infrastructure organisations, the cost of DoS attacks can be significant. A loss of 8 million was the result of a single DoS incident was reported by an Australian Computer Crime and Security Survey in 2005. A significant and prolonged period of system unavailability could result in losses with magnitude higher than what is considered for many critical infrastructure companies. More than the potential for significant financial losses, some critical infrastructure organisations losses extend to social and human costs through an inability to deliver essential services. A loss of life could be an indirect impact such as through a DoS impact on the health system, or delays in emergency service dispatch. even intangibles such as decreased morale and loss of reputation are the other costs suffered.

To mitigate the risks of DoS and DDoS attacks procedures, software and hardware can be put in place that will protect systems prior to attack. It can detects malicious activity as it occurs. Then support the organisation in reacting appropriately as required. As a result of the nature of DoS attacks, it is often the case that strong reactive mechanisms are the best form of defence. These aspects motivates this project to give an approach based on Interface based rate limiting algorithm to control attack.

3.1 Issues in the existing algorithms

Many algorithm have been proposed and they all cover only limited aspects of DDoS attack. Most of the approaches have post attack analysis and countering measures. These methodologies track the log files of server machines and after the attack is accomplished, the analysis is done. If next season other kind of attack arises, then these methodologies fails. But ours approach based on rate limiting mechanism works efficiently as it is applied on interfaces not on server. Therefore, server is prevented. My approach is basically a prevention measure not a post attack prevention measure. In this way, server and its resources are protected which makes this approach one or the other way better.

Chapter 4

Proposed Approach

4.1 Introduction

Currently, internet that we are using is prone to attacks. The three internets critical infrastructures which are availability, integrity and confidentiality are yet to be achieved completely. The existing network infrastructure and their benefits are illegally exploited by attackers. Denial of service attack is an active type of attack that affects availability infrastructure of the internet. Dos attacks a system in various ways which are discussed earlier. The DoS which is considered here creates flood which uses bandwidth of the channel to be used by clients for legitimate work from server machine.

DDoS attack which uses millions of zombies machines mostly with forged source addresses creates a surge of traffic without packet content signature. The available link bandwidth varies in accordance with the statistics of the input traffic[15]. These statistics of arriving traffic are not stationary as internet parameters like network traffic load, mix of traffic, mix of congestion control actions and on/off flows keeps on changing. The bottleneck link in the victim network is consumed by the huge volume of unwanted traffic created by various tools used to attack server. The defense technique proposed here, aims to provide enormous bandwidth to legitimate users at the time of attack. The satisfactory efficiency to detect and filter out attack traffic is not being fully achieved by most of the current defense approaches.

The approach used here to defend DoS/DDoS is to rate limit the attack traffic so that legitimate users are not affected. Rate limiting assigns restriction to bandwidth for traffic like ICMP, UDP or specific connection types[16]. It proves itself an effective countermeasure to control rate oriented attacks on condition that attacker send more traffic than the legitimate client. INTERFACE BASED RATE LIMITING ALGORITHM [17][18] proposed here with leaky bucket algorithm as strategy applied on specific interface selected by the IBRL Algorithm is to mitigate the vulnerabilities of DDoS attack. Two important parameters that are: response time and packet drop are measured and the attack traffic is mitigated using IBRL algorithm.

4.2 Terms Used

Taking a server machine (node) having three interfaces using which it receives traffic of packets from clients and even attackers. Interfaces here are:
S(0/0), S(0/1) and S(0/2).
Some terms used in algorithm:

- Throughput (P): In communication networks, throughput is the average rate of successful message delivery over a communication channel. This data may be delivered over a physical or logical link, or pass through a certain network node. The throughput is usually measured in bits per second (bit/s or bps), and sometimes in data packets per second or data packets per time slot.
- Bandwidth: In computer networking bandwidth, network bandwidth, is a measurement of bit-rate of available or consumed data communication resources expressed in bits per second or multiples of it (bit/s, kbit/s, Mbit/s, Gbit/s, etc.).
  B= (Consumed bandwidth of interface/total channel capacity)*100
- b=constant (set 95 %)
- Packet Drop: Packet loss occurs when one or more packets of data travelling across a computer network fail to reach their destination.

  Packet loss can be caused by a number of factors including signal degradation over the network medium due to multi-path fading, packet drop because of channel congestion, corrupted packets rejected in-transit, faulty networking hardware, faulty network drivers or normal routing routines.

  When caused by network problems, lost or dropped packets can result in highly noticeable performance issues or jitter with streaming technologies, voice over IP, online gaming and videoconferencing, and will affect all other network applications to a degree. However, it is important to note that packet loss does not always indicate a problem. If the latency and the packet loss at the destination hop are acceptable then the hops prior to that one don't matter.
- Response Time: Response time is defined as the interval from when a user initiates a request to the instant at which the first part of the response is received at by the application.

- Interface: A Network interface is a systems (software and/or hardware) interface between two pieces of equipment or protocol layers in a network. A network interface will usually have some form of network address. This may consist of a node Id and a port number or may be a unique node Id in its own right.

4.3 The System

variables description:

- S(0/0)- Serial interface 1 of the edge router
- S(0/1)- Serial interface 2 of the edge router
- S(0/2)- Serial interface 3 of the edge router
- RL- Rate limiting rules
- B- Bandwidth of a router
- b- Maximum Bandwidth
- P- Throughput

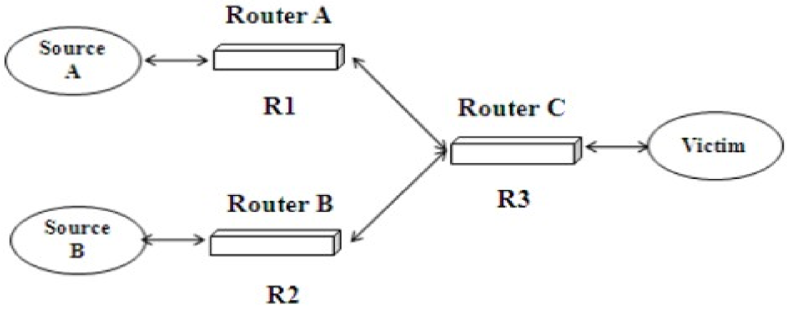


Figure 4.1: Router having interfaces at server side

4.4 Proposed Algorithm

```
if (P(S(1/0))>P(S(1/1)) && P(S(1/0))>P(S(1/2))) then
    check B(S(0/1)) if (B(S(1/0))>b) then
        RL → S(1/0)
    end
end
else if (P(S(1/1))>P(S(1/0)) && P(S(1/1)>P(S(1/2))) then
    check B(S(1/1))
    if (B(S(1/1))>b) then
        RL → S(1/1)
    end
end
else if (P(S(1/2))>P(S(1/0)) && P(S(1/2))>P(S(1/1))) then
    check B(S(1/2))
    if (B(S(1/2))>b) then
        RL → S(1/2)
    end
end
```

Algorithm 1: INTERFACE BASED RATE LIMITING ALGORITHM

4.5 RL- RATE LIMITING USING LEAKY BUCKET ALGORITHM

Consider a Bucket with a small hole at the bottom, whatever may be the rate of water pouring into the bucket, the rate at which water comes out from that small hole is constant. Once the bucket is full, any additional water entering it spills over the sides and is lost.

The same idea of leaky bucket can be applied to packets. Conceptually each network interface contains a leaky bucket. And the following steps are performed

- When the host has to send a packet, the packet is thrown into the bucket.
- The bucket leaks at a constant rate, meaning the network interface transmits packets at a constant rate.
- Bursty traffic is converted to a uniform traffic by the leaky bucket.

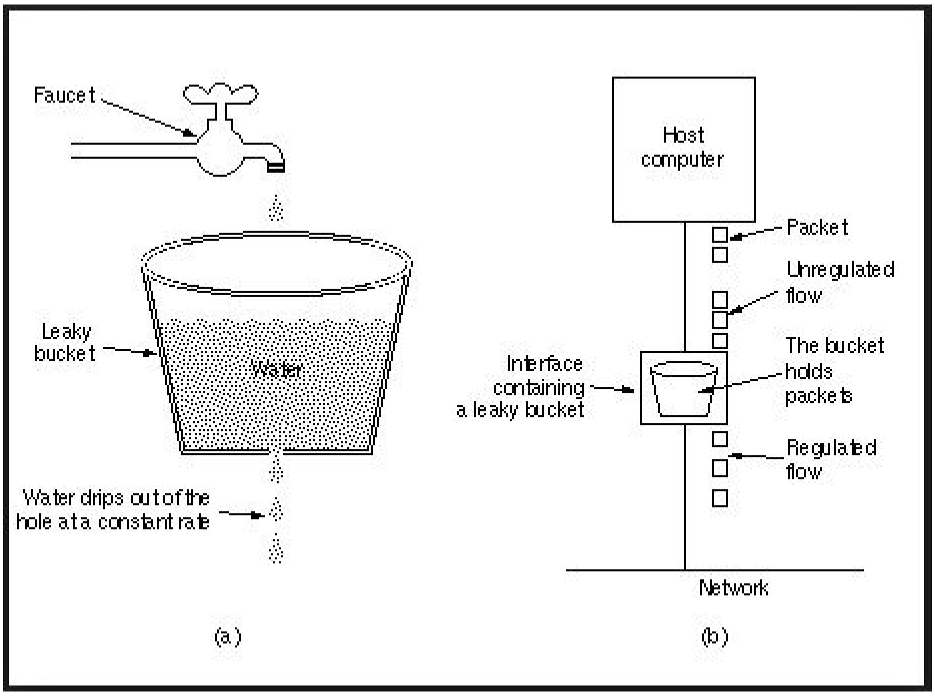


FIGURE 4.2: Leaky bucket analogy

- Here, the bucket is a finite queue that outputs at a finite rate.
- This arrangement can be simulated in the operating system or can be built into the hardware. Implementation of this algorithm consists of a finite queue.
- Whenever a packet arrives, if there is room in the queue it is queued up and if there is no room then the packet is discarded.

Chapter 5

Simulations and Results

The Setup

Simulation Snapshots

Outputs

Results and Analysis

5.1 The Setup

The simulation set up used is Intel Dual Core Processor with 2.10GHz Clock speed and Memory of 4 GB. The algorithm was simulated in OMNeT++ Version 4.2.2 Network simulator.

The Wireless Sensor Network used in the simulation is shown below:

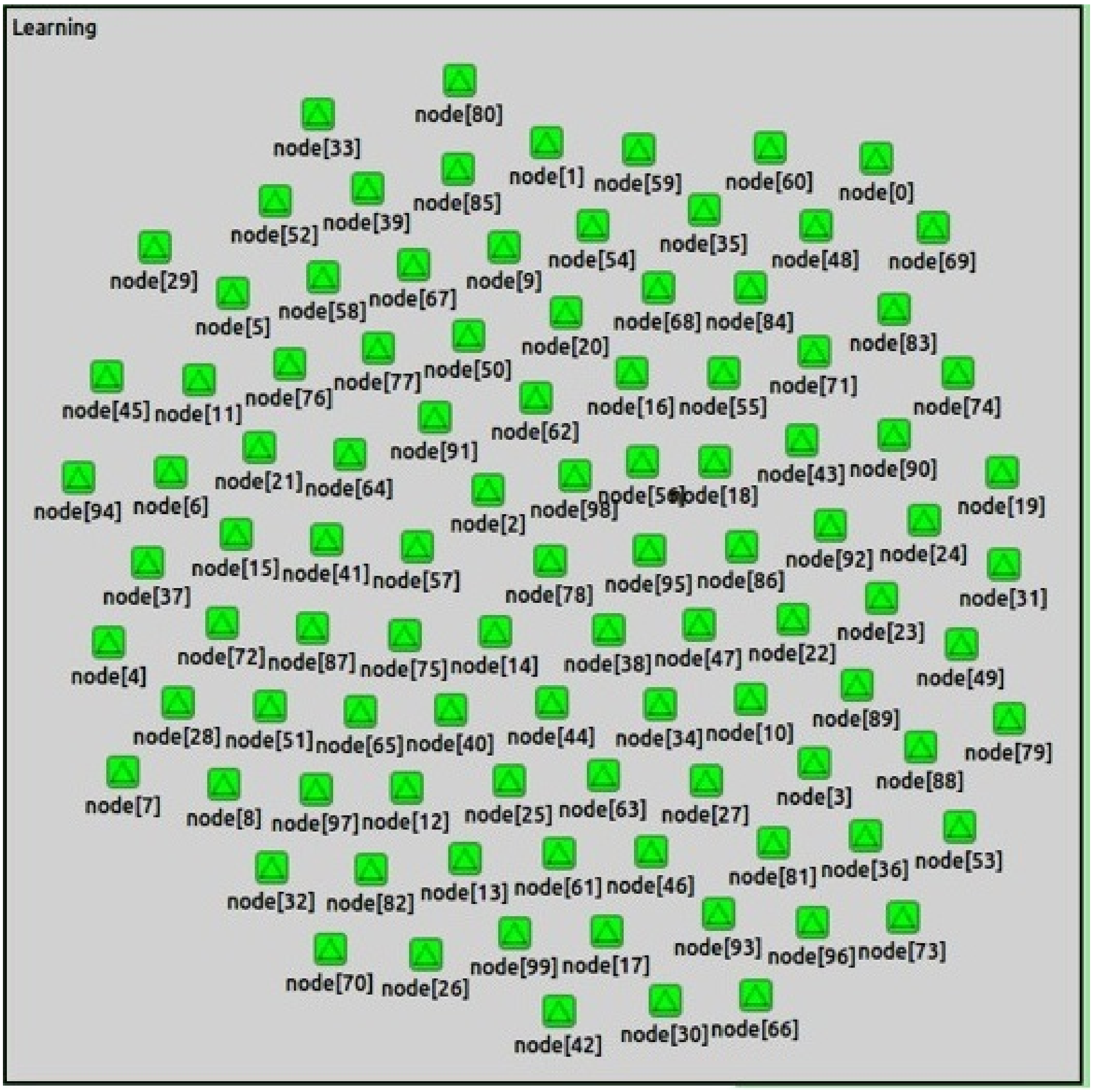


FIGURE 5.1: Simulation Network

5.2 Simulation Snapshot

The Simulation output trace at run-time from the OMNeT++ simulator is as shown below:

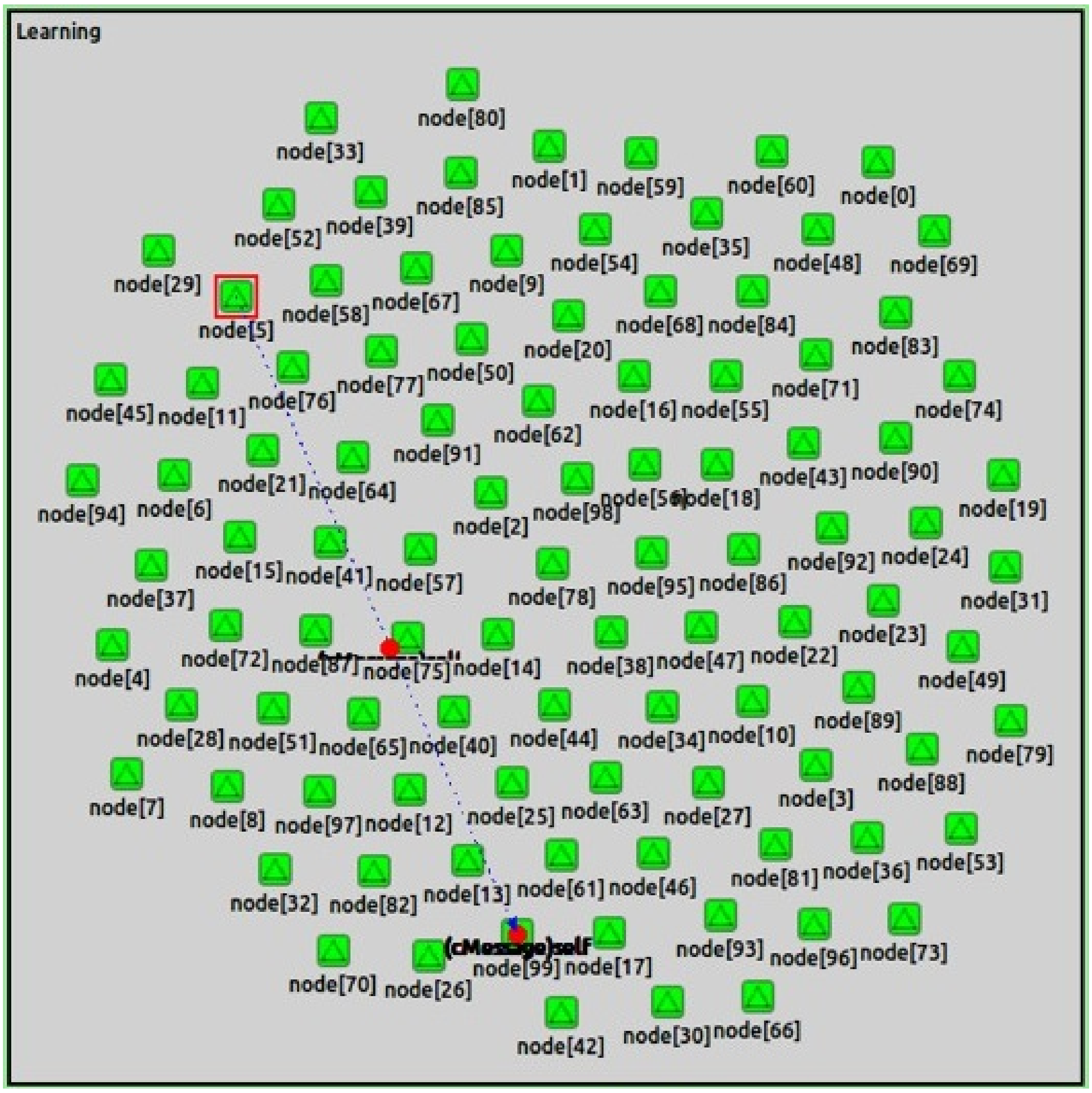
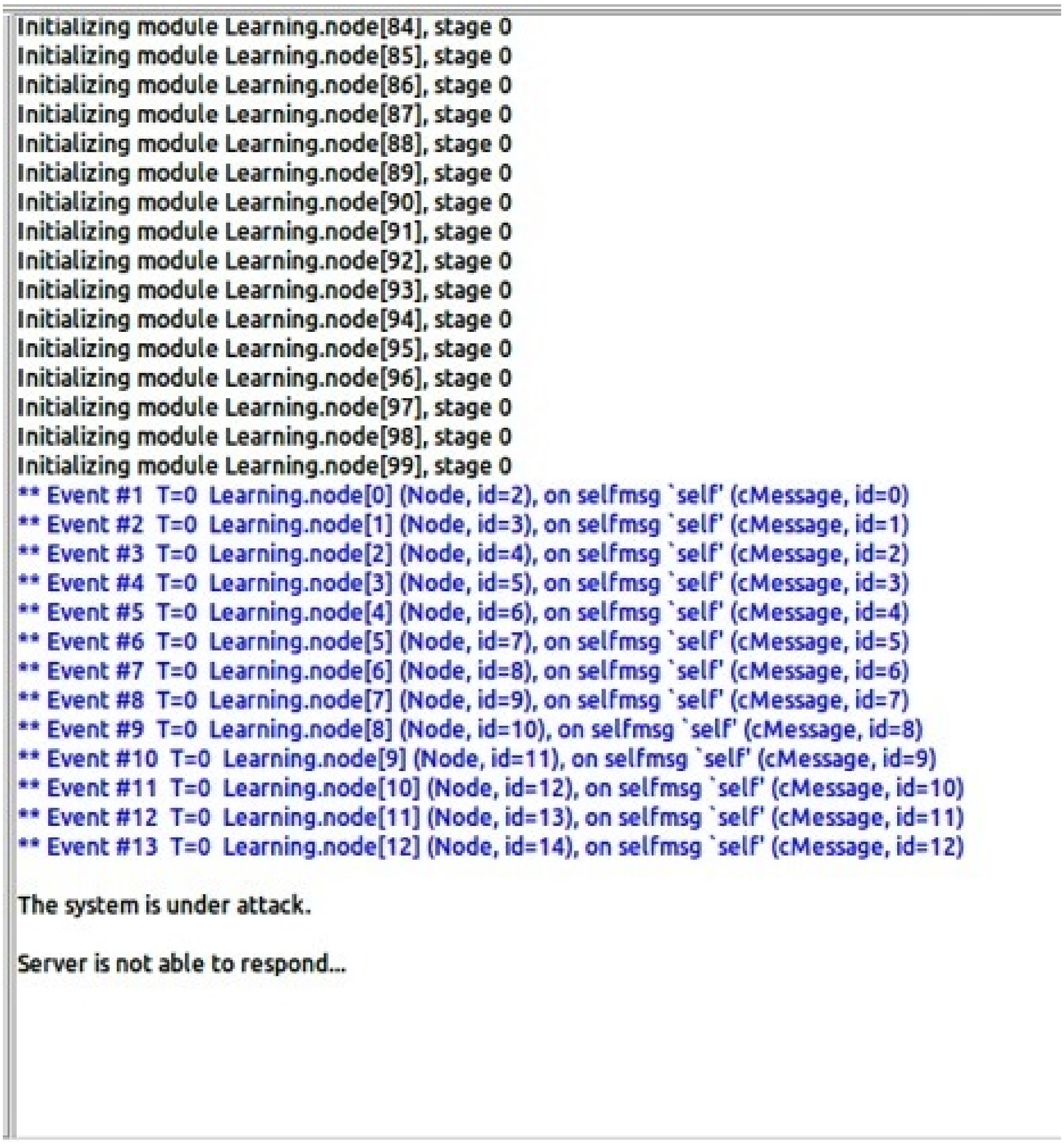


FIGURE 5.2: Run-time window of simulation

5.3 Outputs

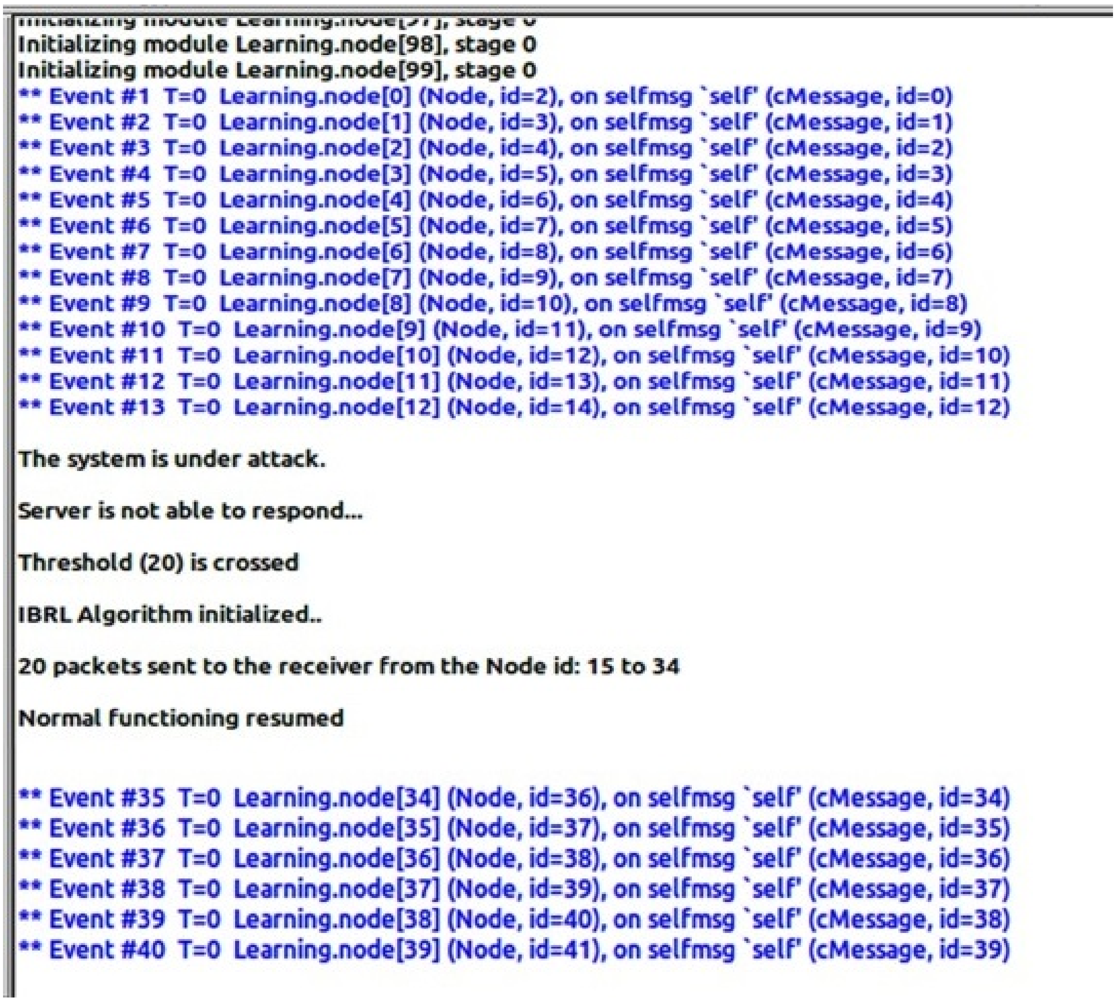
Outputs are shown by the OMNeT++ Output Windows. Detection of attack is been done on the basis of particular value of number of packets received at the victim server side. If number of packets received crosses this value then the system becomes unable to respond and the packet passing stopped. This is shown by the snapshot Fig 5.3.



```
Initializing module Learning.node[84], stage 0
Initializing module Learning.node[85], stage 0
Initializing module Learning.node[86], stage 0
Initializing module Learning.node[87], stage 0
Initializing module Learning.node[88], stage 0
Initializing module Learning.node[89], stage 0
Initializing module Learning.node[90], stage 0
Initializing module Learning.node[91], stage 0
Initializing module Learning.node[92], stage 0
Initializing module Learning.node[93], stage 0
Initializing module Learning.node[94], stage 0
Initializing module Learning.node[95], stage 0
Initializing module Learning.node[96], stage 0
Initializing module Learning.node[97], stage 0
Initializing module Learning.node[98], stage 0
Initializing module Learning.node[99], stage 0
** Event #1  T=0  Learning.node[0] (Node, id=2), on selfmsg `self' (cMessage, id=0)
** Event #2  T=0  Learning.node[1] (Node, id=3), on selfmsg `self' (cMessage, id=1)
** Event #3  T=0  Learning.node[2] (Node, id=4), on selfmsg `self' (cMessage, id=2)
** Event #4  T=0  Learning.node[3] (Node, id=5), on selfmsg `self' (cMessage, id=3)
** Event #5  T=0  Learning.node[4] (Node, id=6), on selfmsg `self' (cMessage, id=4)
** Event #6  T=0  Learning.node[5] (Node, id=7), on selfmsg `self' (cMessage, id=5)
** Event #7  T=0  Learning.node[6] (Node, id=8), on selfmsg `self' (cMessage, id=6)
** Event #8  T=0  Learning.node[7] (Node, id=9), on selfmsg `self' (cMessage, id=7)
** Event #9  T=0  Learning.node[8] (Node, id=10), on selfmsg `self' (cMessage, id=8)
** Event #10  T=0  Learning.node[9] (Node, id=11), on selfmsg `self' (cMessage, id=9)
** Event #11  T=0  Learning.node[10] (Node, id=12), on selfmsg `self' (cMessage, id=10)
** Event #12  T=0  Learning.node[11] (Node, id=13), on selfmsg `self' (cMessage, id=11)
** Event #13  T=0  Learning.node[12] (Node, id=14), on selfmsg `self' (cMessage, id=12)

The system is under attack.

Server is not able to respond...
```

Figure 5.3: Omnet Output Window at attack time

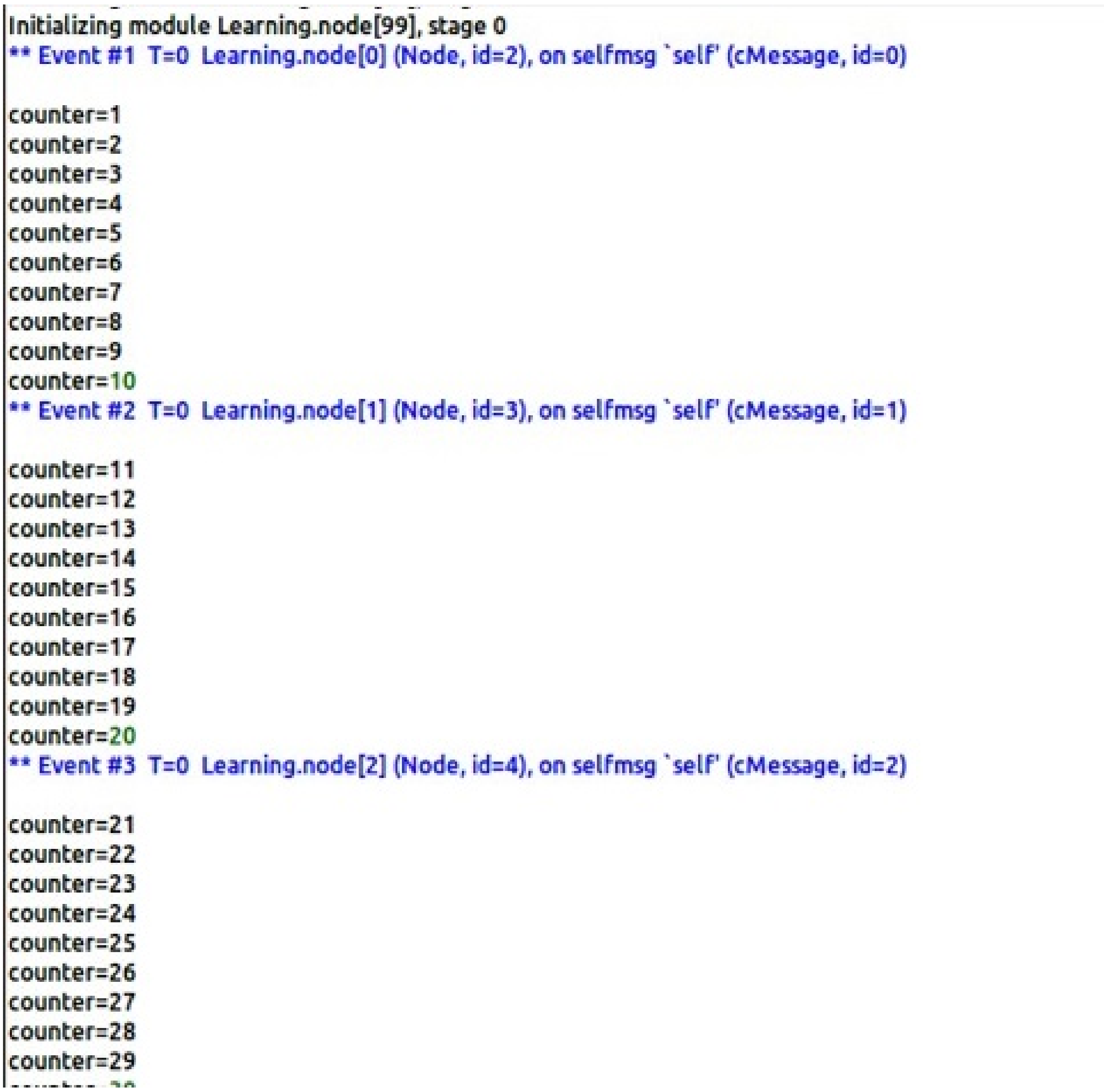
Now, after applying IBRL approach with Leaky bucket rate limiting technique, when attack is detected, the normal functioning of message passing is resumed. This is done by storing extra packets for some time in QUEUE BUFFER and sending them altogether then after i.e in next slot. All this is shown in snapshot Fig 5.4



```
Initializing module Learning.node[97], stage 0
Initializing module Learning.node[98], stage 0
Initializing module Learning.node[99], stage 0
** Event #1  T=0  Learning.node[0] (Node, id=2), on selfmsg `self' (cMessage, id=0)
** Event #2  T=0  Learning.node[1] (Node, id=3), on selfmsg `self' (cMessage, id=1)
** Event #3  T=0  Learning.node[2] (Node, id=4), on selfmsg `self' (cMessage, id=2)
** Event #4  T=0  Learning.node[3] (Node, id=5), on selfmsg `self' (cMessage, id=3)
** Event #5  T=0  Learning.node[4] (Node, id=6), on selfmsg `self' (cMessage, id=4)
** Event #6  T=0  Learning.node[5] (Node, id=7), on selfmsg `self' (cMessage, id=5)
** Event #7  T=0  Learning.node[6] (Node, id=8), on selfmsg `self' (cMessage, id=6)
** Event #8  T=0  Learning.node[7] (Node, id=9), on selfmsg `self' (cMessage, id=7)
** Event #9  T=0  Learning.node[8] (Node, id=10), on selfmsg `self' (cMessage, id=8)
** Event #10  T=0  Learning.node[9] (Node, id=11), on selfmsg `self' (cMessage, id=9)
** Event #11  T=0  Learning.node[10] (Node, id=12), on selfmsg `self' (cMessage, id=10)
** Event #12  T=0  Learning.node[11] (Node, id=13), on selfmsg `self' (cMessage, id=11)
** Event #13  T=0  Learning.node[12] (Node, id=14), on selfmsg `self' (cMessage, id=12)

The system is under attack.

Server is not able to respond...

Threshold (20) is crossed

IBRL Algorithm initialized..

20 packets sent to the receiver from the Node id: 15 to 34

Normal functioning resumed


** Event #35  T=0  Learning.node[34] (Node, id=36), on selfmsg `self' (cMessage, id=34)
** Event #36  T=0  Learning.node[35] (Node, id=37), on selfmsg `self' (cMessage, id=35)
** Event #37  T=0  Learning.node[36] (Node, id=38), on selfmsg `self' (cMessage, id=36)
** Event #38  T=0  Learning.node[37] (Node, id=39), on selfmsg `self' (cMessage, id=37)
** Event #39  T=0  Learning.node[38] (Node, id=40), on selfmsg `self' (cMessage, id=38)
** Event #40  T=0  Learning.node[39] (Node, id=41), on selfmsg `self' (cMessage, id=39)
```

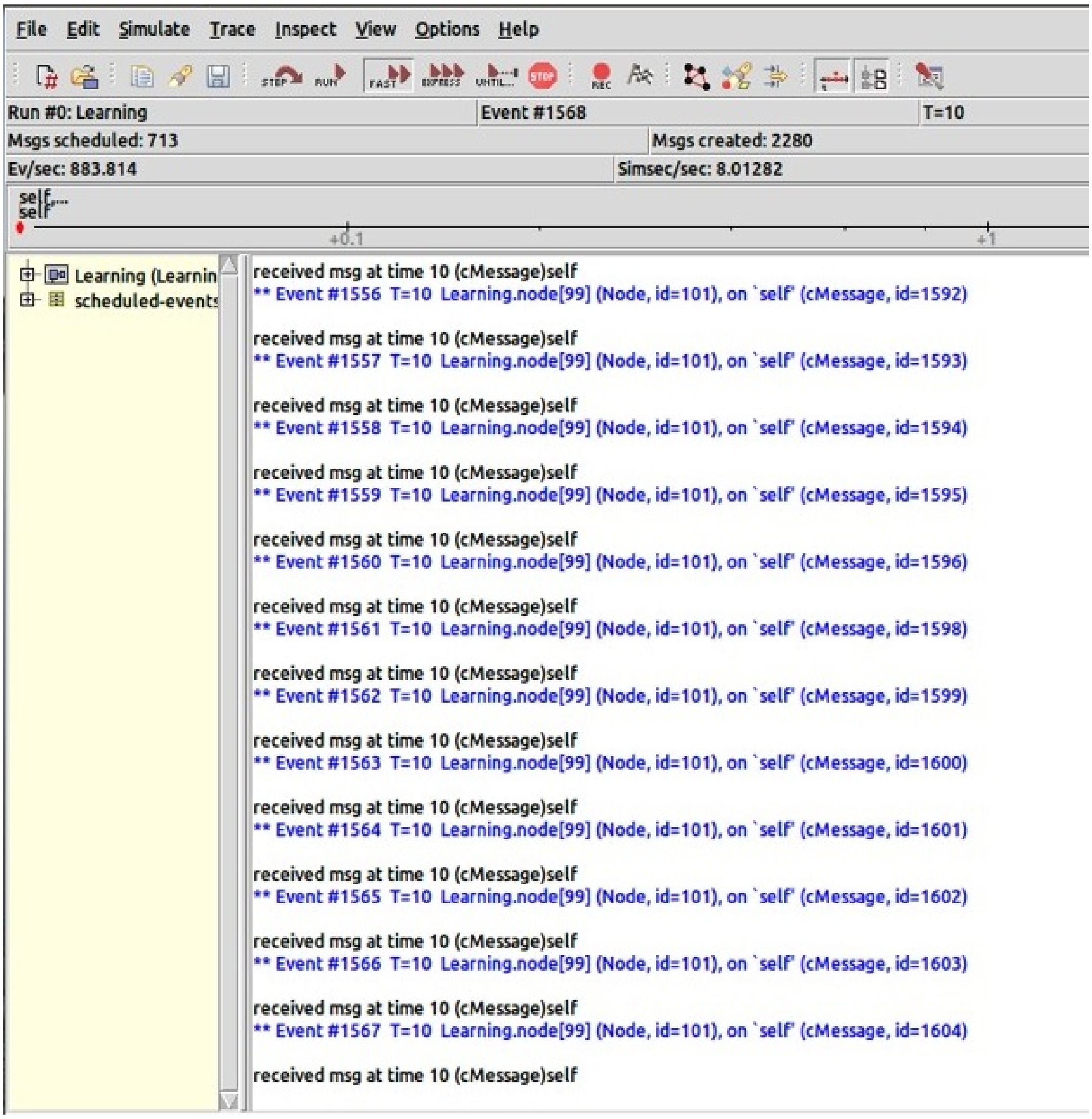
Figure 5.4: Omnet Output Window at attack time with algorithm applied

The previous two snapshots are for small values of threshold packets number and queue buffer. But main simulation is done with high values that are similar to actual values that are faced by real network server client system under attack. Fig 5.5 and Fig 5.6 shows this simulation with 99 sender-clients and 1 receiver-server and with every node sending 10 packets(Fig 5.5 showing normal traffic) and 1000 packets(Fig 5.6 showing attack traffic) at a time instant T simultaneously.



```
Initializing module Learning.node[99], stage 0
** Event #1  T=0  Learning.node[0] (Node, id=2), on selfmsg `self' (cMessage, id=0)

counter=1
counter=2
counter=3
counter=4
counter=5
counter=6
counter=7
counter=8
counter=9
counter=10
** Event #2  T=0  Learning.node[1] (Node, id=3), on selfmsg `self' (cMessage, id=1)

counter=11
counter=12
counter=13
counter=14
counter=15
counter=16
counter=17
counter=18
counter=19
counter=20
** Event #3  T=0  Learning.node[2] (Node, id=4), on selfmsg `self' (cMessage, id=2)

counter=21
counter=22
counter=23
counter=24
counter=25
counter=26
counter=27
counter=28
counter=29
```

Figure 5.5: Simulation of normal traffic flow with every client sending 10 packets at a given time slot



Run #0: Learning | Event #1568 | T=10

Msgs scheduled: 713 | Msgs created: 2280

Ev/sec: 883.814 | Simsec/sec: 8.01282

```
received msg at time 10 (cMessage)self
** Event #1556  T=10  Learning.node[99] (Node, id=101), on `self' (cMessage, id=1592)

received msg at time 10 (cMessage)self
** Event #1557  T=10  Learning.node[99] (Node, id=101), on `self' (cMessage, id=1593)

received msg at time 10 (cMessage)self
** Event #1558  T=10  Learning.node[99] (Node, id=101), on `self' (cMessage, id=1594)

received msg at time 10 (cMessage)self
** Event #1559  T=10  Learning.node[99] (Node, id=101), on `self' (cMessage, id=1595)

received msg at time 10 (cMessage)self
** Event #1560  T=10  Learning.node[99] (Node, id=101), on `self' (cMessage, id=1596)

received msg at time 10 (cMessage)self
** Event #1561  T=10  Learning.node[99] (Node, id=101), on `self' (cMessage, id=1598)

received msg at time 10 (cMessage)self
** Event #1562  T=10  Learning.node[99] (Node, id=101), on `self' (cMessage, id=1599)

received msg at time 10 (cMessage)self
** Event #1563  T=10  Learning.node[99] (Node, id=101), on `self' (cMessage, id=1600)

received msg at time 10 (cMessage)self
** Event #1564  T=10  Learning.node[99] (Node, id=101), on `self' (cMessage, id=1601)

received msg at time 10 (cMessage)self
** Event #1565  T=10  Learning.node[99] (Node, id=101), on `self' (cMessage, id=1602)

received msg at time 10 (cMessage)self
** Event #1566  T=10  Learning.node[99] (Node, id=101), on `self' (cMessage, id=1603)

received msg at time 10 (cMessage)self
** Event #1567  T=10  Learning.node[99] (Node, id=101), on `self' (cMessage, id=1604)

received msg at time 10 (cMessage)self
```

FIGURE 5.6: Simulation of attack traffic flow with every attackers sending 1000 packets at a given time slot

5.4 Results and Analysis

To see how the implemented algorithm is able to control or mitigate DoS or DDoS attack (packet flood) by limiting rate of number of packets, two network important parameters at server side are analysed.

These are:

1. Response time
2. Packet drop

These two analysed and their changes for different scenarios are plotted against number of nodes(clients). The three different scenarios are:

1. Normal traffic
2. DDoS attack traffic
3. Attack with Proposed IBRL algorithm applied.

5.4.1 Packet drop Vs number of nodes

As number of nodes increases, number of packets increases and therefore load at server packet drop also increases with it. In Fig 5.7, it can be observed that how drastically packets are dropped as compared to normal traffic scenario. this is due to the increase in number of nodes. These packets are to be prevented from getting dropped as they could be sent by legitimate clients.

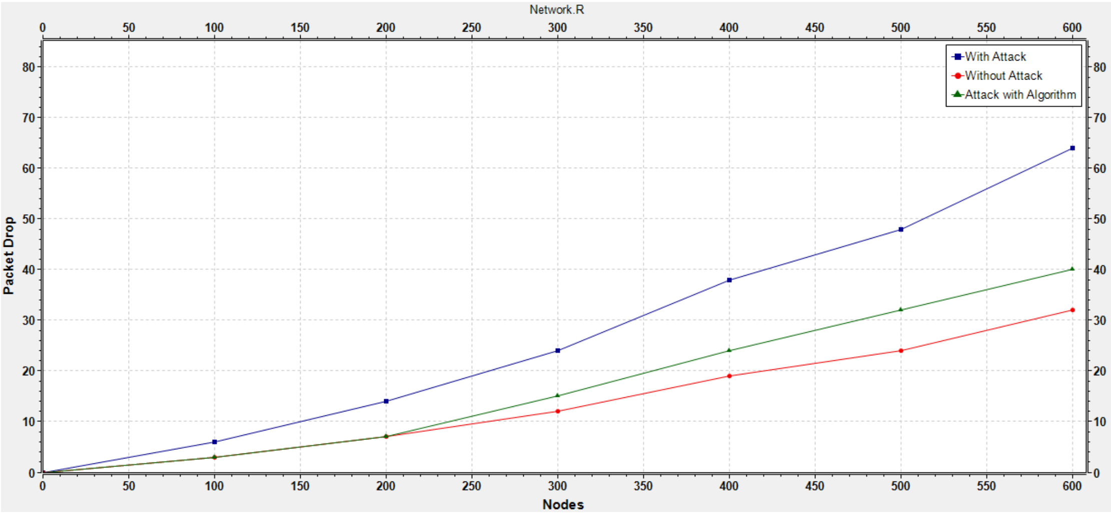


Figure 5.7: Packet drop Vs. Number of nodes plot

when the algorithm is applied, the change in packet drop with respect to number of nodes becomes much more like the initial normal traffic plot. That tells the algorithm is working and giving positive and intended results.

5.4.2 Response time Vs. number of nodes

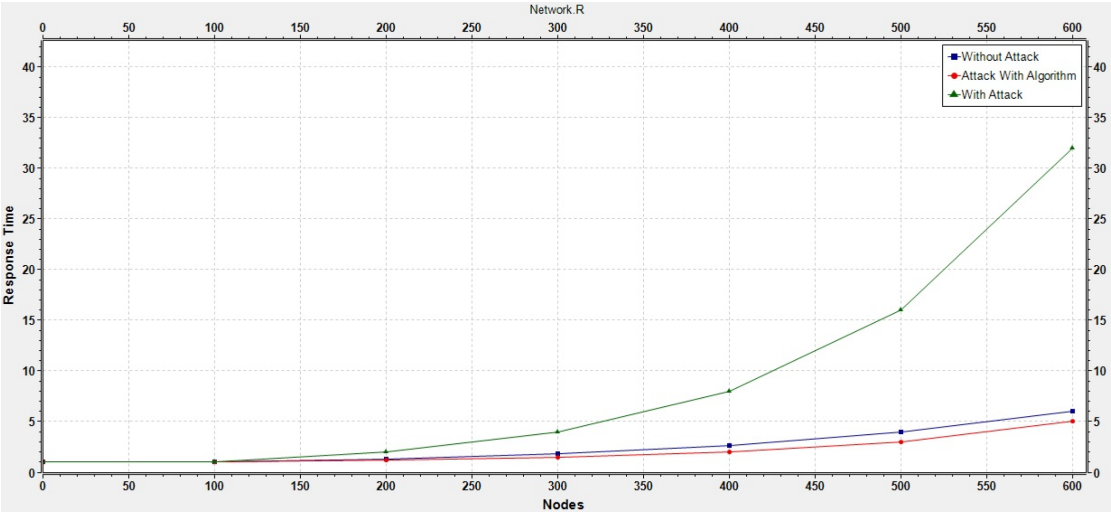


FIGURE 5.8: Response time Vs. Number of nodes plot

As number of nodes increases, number of packets increases, load to respond to all request at server increases which in turn leads to increase in the response time, which is really not good as user does not like to wait for its request to be served by server.

From Fig 5.8, following can be analysed:

1. In normal traffic, the system is able to serve the clients within a good response time. And as the number of nodes increases, although there is a increase in response time but not significant i.e system is working gently.
2. At the time of attack, due to high load the rapid change in response time can be observed from the plot. This increase the response time could so high that sometimes system is supposed to be failed.
3. In algorithm implementation plot, the change in response time with respect to number of nodes becomes nearly same to that of without attack i.e.normal traffic. This observations and analysis tells that the algorithm with rate limiting strategy is able to decrease the impact of DDoS in terms of both Response time and packet drop.

Chapter 6

Conclusions and Future Works

6.1 Conclusions

The proposed scheme consists of a Interface based rate limiting algorithm with the application of leaky bucket algorithm at specific interface of server's router. The interface on which the algorithm is to be applied is chosen by IBRL algorithm. The procedure is activated or initialised when the Denial of service attack(Flooding Upstream)is accomplished and the number of packets crosses the receiving capacity of server. At the time of attack, the system server is unable to respond to its legitimate clients. When the proposed mitigation procedure is activated, the rate limiting strategy at interface level proves to control the rate of incoming packets and mitigate the attack effects. This is shown in the output as OMNeT++ Window at the time of attack and attack with algorithm applied. This whole procedure is analysed in the simulation using two parameters. Response time and packet drop are analysed and their changes with respect to number of nodes are plotted. By the simulation results and comparisons we conclude that the algorithm is able to bring decrease in response time and number of packets dropped and hence the using through the implementation it is concluded that even under denial of service attack, system resumed working properly.

6.2 Future Works

In future we intend to optimize the algorithm by focusing upon the optimised rate limiting technique which could have better and efficient buffer in many aspects. There are various network based parameters which can also be analysed and could help in improvements in the procedure.

Bibliography

[1] Behrouz A. Forouzan. Cryptography and network security. *Tata McGraw Hill Publication*, 2008.

[2] S. Specht and R. Lee. Taxonomies of distributed denial of service networks, attacks, tools, and countermeasures. *Technical Report CE-L2003-03*, page 164, 2003.

[3] http://www.w3.org/security/faq/wwwsf6.html.

[4] Graham Wheeler. Denial-of-service:courting disaster and technical report. *Technical Re- port,CEQURUX Technologies.*

[5] S. Savage, D. Wetherall, A. Karlin, and T.Anderson. Practical network support for ip traceback. *ACM SIGCOMM*, 2000.

[6] R.Mahajan, S. Bellovin, S. Floyd, V. Paxson, and S. Shenker. Controlling high bandwidth aggregates in the network. *ACM Computer Communications Review*, 32(3):62–73, 2002.

[7] D.K.Yau, J.C. Lui, F. Liang, and Y. Yam. Defending against distributed denial-of-service attacks with max-min fair server- centric router throttles. *ACM Transaction on Networking*, 13(1):29– 42, 2005.

[8] M.Sung and J. Xu. Ip traceback-based intelligent packet filtering: A novel technique for defending against internet ddos attacks. *Proc. of 10th IEEE ICNP*, 2002.

[9] Shubha Kher Jinran Chen and Arun Somani. Mitigating denial of service attack using proof of work and token bucket algorithm. *Proceedings of the 2006 workshop on Dependability issues in wireless ad hoc networks and sensor networks*, pages 65–72, 2011.

[10] K. Argyraki, D. Cheriton, Sergio Marti, T. J. Giuli, Kevin Lai, and Mary Baker. Active internet traffic filtering: Real-time response to denial-of-service attacks. *USENIX*, 2005.

[11] F. Liang and D. Yau. Using adaptive router throttles against distributed denial-of -service attacks, journal of software. *IEEE Transactions on Computers*, 13(7):1120–1127, 2002.

[12] J. Mirkovic, E. Arikan, S. Wei, S.Fahmy, R. Thomas, and P. Reiher. Benchmarks for ddos defense evaluation. *MILCOM*, 2006.

[13] J.Mirkovic, M. Robinson, P. Reiher, and G. Oikonomou. Distributed defense against ddos attacks. *University of Delaware CIS Department Technical Report CIS-TR-2005-02*, 2005.

[14] Yinan Jing, Xueping Wang, Xiaochun Xiao, and Gendu Zhang. Defending against meek ddos attacks by ip traceback-based rate limiting. *Global Telecommunications Conference, GLOBECOM '06. IEEE*, 2006.

[15] Monika Sachdeva, Krishan Kumar, Gurvinder Singh, and Kuldip Singh. Performance analysis of web service under ddos attacks. *IEEE International Advance Computing Conference*, pages 6–9, 2009.

[16] Ming Li, Jun Li, and Wei Zhao. Simulation study of flood attacking of ddos. *Internet Computing in Science and Engineering*, pages 189–199, 2008.

[17] Nithya Ramanathan, Kevin K. Chang, Rahul Kapur, Lewis Girod, Eddie Kohler, and Deborah Estrin. Sympathy for the sensor network debugger. *SenSys*, pages 255–267, 2005.

[18] G. Preetha, B.S. Kiruthika Devi, and S. Mercy Shalinie. Combat model-based ddos detection and defence using experimental testbed: a quantitative approach. *International Journal of Intelligent Engineering Informatics*, pages 261–279, 2011.