

SYN FLOODING ATTACK PREVENTION USING A NOVEL APPROACH: HRTE ALGORITHM AND COMPARATIVE ANALYSIS WITH OPTIMIZING ALGORITHM

by

Banalata Paul
CSE 05006441
Umma Zinat Jahan Swarna
CSE 05006450



A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of
BSc in Computer Science and Engineering.

**DEPARTMENT OF COMPUTER SCIENCE
STAMFORD UNIVERSITY BANGLADESH**

March, 2017

Chapter 1

Introduction

Introduction

Over the past few years many sites on the Internet have been the target of denial of service (DoS) attacks, among which TCP SYN flooding is the most prevalent. Indeed, recent studies have shown an increase of such attacks, which can result in disruption of services that costs from several millions to billions of dollars. The aim of denial of service attacks is to consume a large amount of resources, thus preventing legitimate users from receiving service with some minimum performance. TCP SYN flooding exploits the TCP's three-way handshake mechanism and its limitation in maintaining half-open connections. Any system connected to the Internet and providing TCP-based network services, such as a Web server, FTP server, or mail server, is potentially subject to this attack. A TCP connection starts with the client sending a SYN message to the server, indicating the client's intention to establish a TCP connection. The server replies with a SYN/ACK message to acknowledge that it has received the initial SYN message and at the same time reserves an entry in its connection table and buffer space. After this exchange, the TCP connection is considered to be half open. To complete the TCP connection establishment, the client must reply to the server with an ACK message. In a TCP SYN flooding attack, an attacker, from a large number of compromised clients in the case of distributed DoS attacks, sends many SYN messages, with fictitious (spoofed) IP addresses, to a single server (victim). Although the server replies with SYN/ACK messages, these messages are never acknowledged by the client. As a result, many half-open connections exist on the server, consuming its resources.

Security has been always an important issue in communication and computation systems. In these systems security has different aspects. Sometimes the threat is in the form of disclosing of our confident information. In this case we use some security algorithms and protocols to protect the confidentiality. Cryptanalysis and robust key management and distribution algorithms play important roles from this point of view. On the other hand sometimes security is defined as continuous and uninterrupted service. This definition is taken when the threat is in form of a DoS attack. The goal of a DoS attack is to completely tie up certain resources so that legitimate users are not able to access a server. A successful DoS attack overpowers the victim and conceals the offender's identity. A DoS attack can be regarded as an explicit attempt of attackers to prevent legitimate users from gaining a normal network service. DoS attacks typically trust on the misuse of exact susceptibility in such a way that it consequences in a denial of the service.

Now arithmetical assessment show that DoS positions at the quarter place in the list of the most poisonous attack classes in contradiction of information systems. Allen and Marin use estimates of the Hurst parameter to identify attacks which cause a decrease in the traffic's self-similarity. Two statistical methods of analyzing network traffic to find DOS attacks are provided in. One monitors the entropy of the source addresses found in packet headers, while the other monitors the average traffic rates of the most active addresses. Many proposals have been made for IDSs intended to detect DoS attacks, most of them being based on the arithmetical detection of high traffic rate spending from the interloper or interlopers.

Generally DoS attacks could have two major forms. In the first one, the malicious user crafts very carefully a packet trying to exploit vulnerabilities in the implemented software (service or a

protocol). In the second form, the malicious user is trying to overwhelm system's resources of the provided service-like memory, CPU or bandwidth, by creating numerous of useless well-formed requests. This type of attack is well known as flooding attack. One of the most common DoS flooding attacks is SYN flooding attacks. It works at the conveyance layer. A TCP connection is recognized in what is known as a 3-way handshake. When a client labors to start a TCP connection to a server, first, the client needs a connection by distribution an SYN packet to the server. Note that SYN flooding attacks goal to use TCP buffer space and do not touch the parameters such as link bandwidth, dispensation capitals and so on. We believe that in a SYN flooding attack an unfair scheduler gives more opportunity to attack requests but prevents legal connections from getting service. Hence, we propose a scheduling-based solution in which when an arriving request faces with a full queue, the oldest half connection is terminated and its resources are assigned to the new connection. When a computer or network resource receives multiple service requests (jobs) at a given time, a scheduling algorithm is necessary to determine the order in which requests are serviced. Scheduling algorithms work based on several job features such as processing time, priority, due date, and so regarding their design goals. In this section, we present Highest **Residence Time Ejection (HRTE)** scheduling algorithms and which is used to design a defense strategy against SYN flooding attacks.

HRTE algorithm is a preemptive two-phase scheduling algorithm. This algorithm is useful for scenarios in which the service time of requests is unknown. According to this scheduling algorithm, while input queue isn't full, HRTE is in its first phase and acts exactly like round robin algorithm. But upon queue becomes full and arriving requests are blocked, HRTE switches to its second phase during which ejects the job with the highest residence time and assigns the released capacity to the arriving requests. HRTE remains in this phase until a free capacity is available in the waiting queue. In other words, it has some similarities with SRTF, but when remaining time is unknown SRTF cannot be used. In this situation we assume that those requests that have the longest duration in the past will have the longest remaining time, as well and hence will be rejected.

Our theoretical analysis and simulations show that the proposed algorithm HRTE shows similar and sometimes better performance in defending SYN flood attack than PSO algorithm. Therefore it is more efficient to use a scheduling algorithm HRTE to detect SYN flooding attack then implementing another algorithm for the attack mitigation purpose for a server with limited resources. At the same time, our proposed algorithm uses threshold to detect the attack requests and ejects them directly, where PSO only adapts some parameters of the server to defend the attack.

Chapter 2

RELATED WORK

There are many researches focusing on SYN flooding attack, several defense mechanisms have been proposed, such as:

Shahram et al. [2] This paper proposes that SYN flooding attack can be viewed metaphorically as result of an unfair scheduling that gives more opportunity to attack requests but prevents legal connections from getting services. In this paper, a scheduling algorithm that ejects the half connection with the longest duration. When number of half open connections reaches to the upper bound. The simulation results show that the proposed defense mechanism improves performance of the under attack system in terms of loss probability of requests and share of regular connections from system resources.

J. Lemon et al. [3] have analyzed the traffic at an Internet gateway and the results showed that we can model the arrival rates of normal TCP SYN packets as a normal distribution. Using this result, we described a new attack detection method taking the time variance of arrival traffic into consideration. Simulation results show that our method can detect attacks quickly and accurately regardless of the time variance of the traffic.

Long et al. [4] proposed two queuing models for the DoS attacks in instruction to get the pack postponement jitter and the loss probability.

D.J Bernstein et al. [5] present a simple and robust mechanism, called *Change-Point Monitoring* (CPM), to detect denial of service (DoS) attacks. The core of CPM is based on the inherent network protocol behaviors, and is an instance of the Sequential Change Point Detection. To make the detection mechanism insensitive to sites and traffics patterns, a non-parametric Cumulative Sum (CUSUM) method is applied.

C. L. Schuba, I. V. Krsul, M. G. Kuhn, E. H. Spafford, A. Sundaram and D. Zamboni, et al. offers protection against SYN flooding for all hosts connected to the same local area network, independent of their operating system or networking stack implementation[6].

Vasilios A. Siris et al. proposed the two algorithms considered are an adaptive threshold algorithm and a particular application of the cumulative sum (CUSUM) algorithm for change point detection [7]. The performance is investigated in terms of the detection probability, the false alarm ratio, and the detection delay. Particular emphasis is on investigating the tradeoffs among these metrics and how they are affected by the parameters of the algorithm and the characteristics of the attacks.

Gholam Shaker et al[8] This paper proposes a self-managing approach, in which the host defends against SYN flooding attack by dynamically tuning off its own two parameters, that is, m (maximum number of half open connections) and h (hold time for each half-open connection). In this way, it formulates the defense problem and optimization problem and then employs the particle swarm optimization (PSO) algorithm to solve it. The simulation results show that the proposed defense strategy improves performance of the under attack system in terms of BUE and PSA.

All of these defense mechanisms are installed at the firewall of the victim server or inside the victim server, thereby providing no hints about the sources of the SYN flooding. They have to rely on the expensive IP trace back to locate the flooding sources. Because of the defense line is at, or close to, the victim, the network resources are also wasted by transmitting the flooding packets.

Moreover, these defense mechanisms are state full, i.e., states are maintained for each TCP connection or state computation is required. Such a solution makes the defense mechanism itself vulnerable to SYN flooding attacks. Recent experiments have shown that a specialized firewall, which is designed to resist SYN floods, became futile under a flood of 14,000 packets per second [9]. The shameful defense mechanisms also degrade the end-to-end TCP performance, e.g., incurring longer delays in setting up connections. In the absence of SYN flooding attacks, all the overheads introduced by the defense mechanism become superfluous. We, therefore, need a simple stateless mechanism to detect SYN flooding attacks, which is immune to the SYN flooding attacks. Also, it is preferred to detect an attack early near its source, so that one can easily trace the flooding source without resorting to expensive IP trace back.

Most of the algorithms used to detect SYN Flood attack, shows the presence of an SYN Flood attack on the server and then tries to increase the space for requests or to decrease the time taken for the processing of each request. But both of these actions might not be of much use as more space allocation might result in resource exhaustion and reducing processing time may hamper the execution of non attacking requests. There, we tried to handle the SYN attack using a novel approach that eliminates the requests which it suspects to be potential attacking requests rather than extracting more resource or reducing allotted processing times.

In our paper we have used Highest Residence Time Ejection (HRTE) defense against SYN flood attack. We have used an objective value which was evaluated by the HRTE by updating the present value of the time for which a request is holding idly in the buffer queue and is it completing the full data transmission or not in every iteration for each request. And based on this, we have maximized the buffer space so that more valid request can arrive and minimized the buffer time so that no request can stay in the buffer queue for a long time.

Chapter: 4

Overview of Scheduling Algorithms Used to prevent SYN Flood Attack: Scope of Our Approach

4.1 Scheduling Algorithms:

The scheduler is an operating system module that selects the next jobs to be admitted into the system and the next process to run. Scheduling is a process which allows one process to use the CPU while the execution of another process is on hold (in waiting state) due to unavailability of any resource like I/O etc, thereby making full use of CPU. The aim of CPU scheduling is to make the system efficient, fast and fair.

When a computer or network resource receives multiple service requests (jobs) at a given time, a scheduling algorithm is necessary to determine the order in which requests are serviced. Scheduling algorithms work based on several job features such as processing time, priority, due date, and so regarding their design goals.

4.2 A Review over Scheduling Algorithms:

There are many different scheduling algorithms proposed for scheduling in different operating systems or switches. A Process Scheduler schedules different processes to be assigned to the CPU based on particular scheduling algorithms. Below we cover some popular algorithms,

- First-IN, First-Out (FIFO) Scheduling
- Shortest-Processing-Time (SPT) Scheduling
- Hard Fairness
- Max-Min Fairness
- Priority Scheduling
- Shortest Remaining Time First(SRTF)
- Round Robin(RR) Scheduling
- Multiple-Level Queues Scheduling

These algorithms are either **non-preemptive or preemptive**. Non-preemptive algorithms are designed so that once a process enters the running state; it cannot be preempted until it completes its allotted time, whereas the preemptive scheduling is based on priority where a scheduler may preempt a low priority running process anytime when a high priority process enters into a ready state.

4.2.1 FIFO:

A common method of job scheduling for computer and network resources is First-Come-First-Serve (FCFS) or FIFO, where jobs are serviced in the order in which they arrive. Using FIFO, the job with the earliest arrival time is served first. Jobs with earlier arrival times are served before jobs that arrive later.

The first in, first out (FIFO) method of inventory valuation is a cost flow assumption that the first goods purchased are also the first goods sold. In most companies, this assumption closely matches the actual flow of goods, and so is considered the most theoretically correct inventory valuation method. The FIFO flow concept is a logical one for a business to follow, since selling off the oldest goods first reduces the risk of obsolescence.

Under the FIFO method, the earliest goods purchased are the first ones removed from the inventory account. This results in the remaining items in inventory being accounted for at the most recently incurred costs, so that the inventory asset recorded on the balance sheet contains costs quite close to the most recent costs that could be obtained in the marketplace. Conversely, this method also results in older historical costs being matched against current revenues and recorded in the cost of goods sold; this means that the gross margin does not necessarily reflect a proper matching of revenues and costs. For example, in an inflationary environment, current-cost revenue dollars will be matched against older and lower-cost inventory items, which yield the highest possible gross margin.

First-In-First-Out (FIFO) Replacement On a page fault, the frame that has been in memory the longest is replaced. **FIFO** is not a stack **algorithm**. In certain cases, the number of page faults can actually increase when more frames are allocated to the process.

The First-in-First-out Algorithm (FIFO) In the first step, the pages are loaded in the main memory. If the page is in the memory, we pass in the other page and is increased by one.

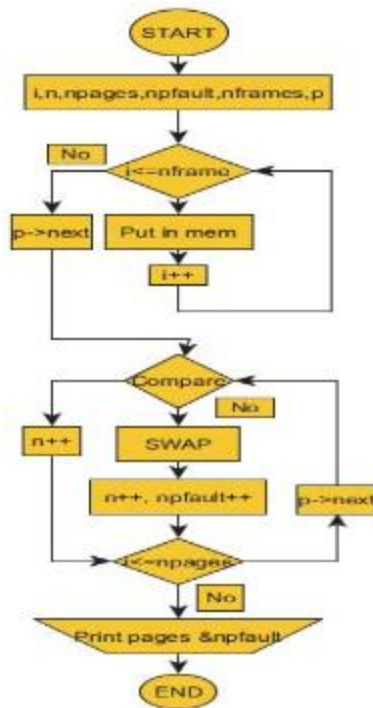


Fig 4.1 FIFO flowchart diagram

4.2.2 SPT (Shortest Processing Time):

Using SPT, jobs are processed in ascending order of processing times. It is well known that SPT minimizes the total completion times of a set of jobs. SPT produces an optimal job sequence for minimizing the total, and thus mean, of job waiting times.

The shortest processing time rule orders the jobs in the order of increasing processing times. Whenever a machine is freed, the shortest job ready at the time will begin processing. This algorithm is optimal for finding the minimum total completion time and weighted completion time. In the single machine environment with ready time at 0 for all jobs, this algorithm is optimal in minimizing the mean flow time, minimizing the mean number of jobs in the system, minimizing the mean waiting time of the jobs from the time of arrival to the start of processing, minimizing the maximum waiting time and the mean lateness.

4.2.3 Hard fairness:

Hard fairness is also known as round-robin scheduling. Homogeneity of resources is not a requirement in this type of scheduling. It is the fairest scheme since each process is guaranteed exactly equal amount of time in order.

Hard fairness” we mean a system where each user transmits at its own desired rate in every channel condition. This corresponds to the so-called delay-limited capacity of fading multi-access channels. When no fairness is imposed, the notion of throughput (or ergodic) capacity region becomes relevant: this is the long-term average rate region achievable when the users adapt their rate and power according to the instantaneous channel conditions .It is well-known that the maximum long-term average

Throughput is achieved by letting only the user with the best channel transmit on each time-frequency coding interval (referred to as slot in the following). However, in a cellular environment

Where users are at different distance from the base station, this strategy would result in a very unfair resource allocation: basically, only the users closest to the base station would be allowed to transmit. Hence, various scheduling algorithms aiming at maximizing the long-term average throughput subject to some fairness constraint have been proposed. Among these, the Proportional Fair Scheduling (PFS) algorithm enjoys many desirable properties and was adopted in some evolutionary 3G wireless communication standards for delay-tolerant data-oriented communications.

4.2.4 Max–Min fairness:

Max–min fairness allocates resources in order of increasing demand. The minimum amount of resources assigned to each process is maximized. So if there are more than enough resources for each process, every process gets what it needs. If there is not, the resources are split evenly. This means that the process which require fewer resources get a higher proportion of their need satisfied. This type of scheme works best when there is not large differences in amount of resources requested by different processes.

In communication networks, multiplexing and the division of scarce resources, **max-min fairness** is said to be achieved by an allocation if and only if the allocation is feasible and an attempt to increase the allocation of any participant necessarily results in the decrease in the allocation of some other participant with an equal or smaller allocation.

4.2.5 Priority Scheduling:

Priority scheduling is a method of scheduling processes based on priority. In this method, the scheduler chooses the tasks to work as per the priority, which is different from other types of scheduling, for example, a simple round robin. Priority scheduling involves priority assignment to every process, and processes with higher priorities are carried out first, whereas tasks with equal priorities are carried out on a first-come-first-served (FCFS) or round robin basis. An example of a general-priority-scheduling algorithm is the shortest-job-first (SJF) algorithm.

Priorities can be either dynamic or static. Static priorities are allocated during creation, whereas dynamic priorities are assigned depending on the behavior of the processes while in the system. To illustrate, the scheduler could favor input/output (I/O) intensive tasks, which lets expensive requests to be issued as soon as possible.

Priorities may be defined internally or externally. Internally defined priorities make use of some measurable quantity to calculate the priority of a given process. In contrast, external priorities are defined using criteria beyond the operating system (OS), which can include the significance of the process, the type as well as the sum of resources being utilized for computer use, user preference, commerce and other factors like politics, etc.

4.2.6 Shortest Remaining Time First (SRTF):

Shortest Remaining Time First, also known as Shortest Job First (SJF), is a scheduling method that is a preemptive version of shortest job next scheduling. In this scheduling algorithm, the process that needs the smallest amount of remaining time to be completed is selected to execute. Since the currently executing process has the shortest amount of remaining time and since that time should only reduce as execution progresses, processes will always run until they complete or a new process is added that requires a smaller amount of time.

Shortest remaining time is advantageous because short processes are handled very quickly. The system also requires very little overhead since it only makes a decision when a process completes or a new process is added, and when a new process is added the algorithm only needs to compare the currently executing process with the new process, ignoring all other processes currently waiting to execute.

Like shortest job first, it has the potential for process starvation; long processes may be held off indefinitely if short processes are continually added. This threat can be minimal when process times follow a heavy-tailed distribution. A similar algorithm which avoids starvation at the cost of higher tracking overhead is highest response ratio next.

Like shortest job next scheduling, shortest remaining time scheduling is rarely used outside of specialized environments because it requires accurate estimations of the runtime of all processes that are waiting to execute. It can be categorized into two parts:

1. Non-preemptive: Once selected for execution, a process continues to run until the end of its CPU burst. It is also known as Shortest Job First (SJF).

2. Preemptive: The process which is currently in execution, runs until it complete or a new process is added in the CPU Scheduler that requires smaller amount of time for execution. It is also known as shortest remaining time first (SRTF).

4.2.7 Round Robin (RR) Scheduling:

Round robin is the scheduling algorithm used by the CPU during execution of the process. Round robin is designed specifically for time sharing systems. It is similar to first come first serve scheduling algorithm but the preemption is the added functionality to switch between the processes.

A small unit of time also known as time slices or quantum is set/ defined. The ready queue works like circular queue. All processes in this algorithm are kept in the circular queue also known as ready queue. Each New process is added to the tail of the ready/circular queue. By using this algorithm, CPU makes sure, time slices (any natural number) are assigned to each process in equal portions and in circular order, dealing with all process without any priority.

The main advantage of round robin algorithm over first come first serve algorithm is that it is starvation free. Every process will be executed by CPU for fixed interval of time (which is set as time slice). So in this way no process left waiting for its turn to be executed by the CPU.

Round robin algorithm is simple and easy to implement. The name round robin comes from the principle known as round robin in which every person takes equal share of something in turn. This method is quite same as the FCFS but the difference is the in this case the processor will not process the whole job (process) at a time. Instead, it will complete an amount of job (quantum) at a turn and then will go to the next process and so on. When all jobs have got a turn, it will again start from the first job and work for a quantum of time/cycle on each job and proceed.

One of the oldest, simplest, fairest and most widely used algorithm is round robin (RR). In the round robin scheduling, processes are dispatched in a FIFO manner but are given a limited amount of CPU time called a time-slice or a quantum. If a process does not complete before its CPU-time expires, the CPU is preempted and given to the next process waiting in a queue. The preempted process is then placed at the back of the ready list.

Round Robin Scheduling is preemptive (at the end of time-slice) therefore it is effective in time-sharing environments in which the system needs to guarantee reasonable response times for interactive users. The only interesting issue with round robin scheme is the length of the quantum. Setting the quantum too short causes too many context switches and lower the CPU efficiency. On the other hand, setting the quantum too long may cause poor response time and approximates FCFS. In any event, the average waiting time under round robin scheduling is often quite long.

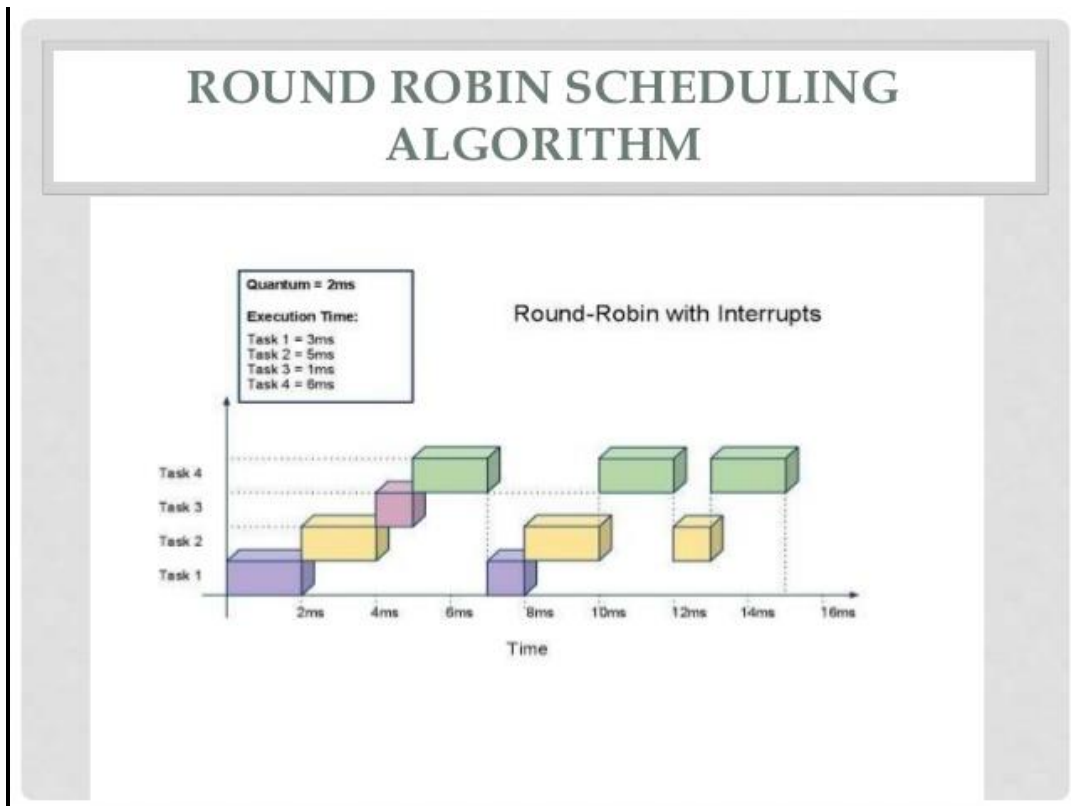


Fig 4.2 Round Robin Scheduling

4.3 Implemented Algorithms

There are many different scheduling algorithms proposed for scheduling in different operating systems or switches. In our paper we cover some popular algorithms, they are

- * Round Robin,
- * First in First out (FIFS),
- * Shortest Remaining Time first (SRTF)

In mat lab, first we choose five process and they were specific time for those scheduling Algorithm. Our process and its bust time are here.

Process	Burst time	Arrival time
P1	3	0
P2	4	0
P3	2	0
P4	6	0
P5	4	0

Table-4.1: Table of process

FIFO, SRTE and RR's Gantt chart:

FIFO

P1	P2	P3	P4	P5
0	3	7	9	15
				19

SRTE

P3	P1	P2	P5	P4
0	2	5	9	13
				19

Round Robin (RR)

Quantum=2

P1	P2	P3	P4	P5	P1	P2	P4	P5	P4	
0	2	4	6	8	10	11	13	15	17	19

Then we execute those programs, here is there output.

FIFO:

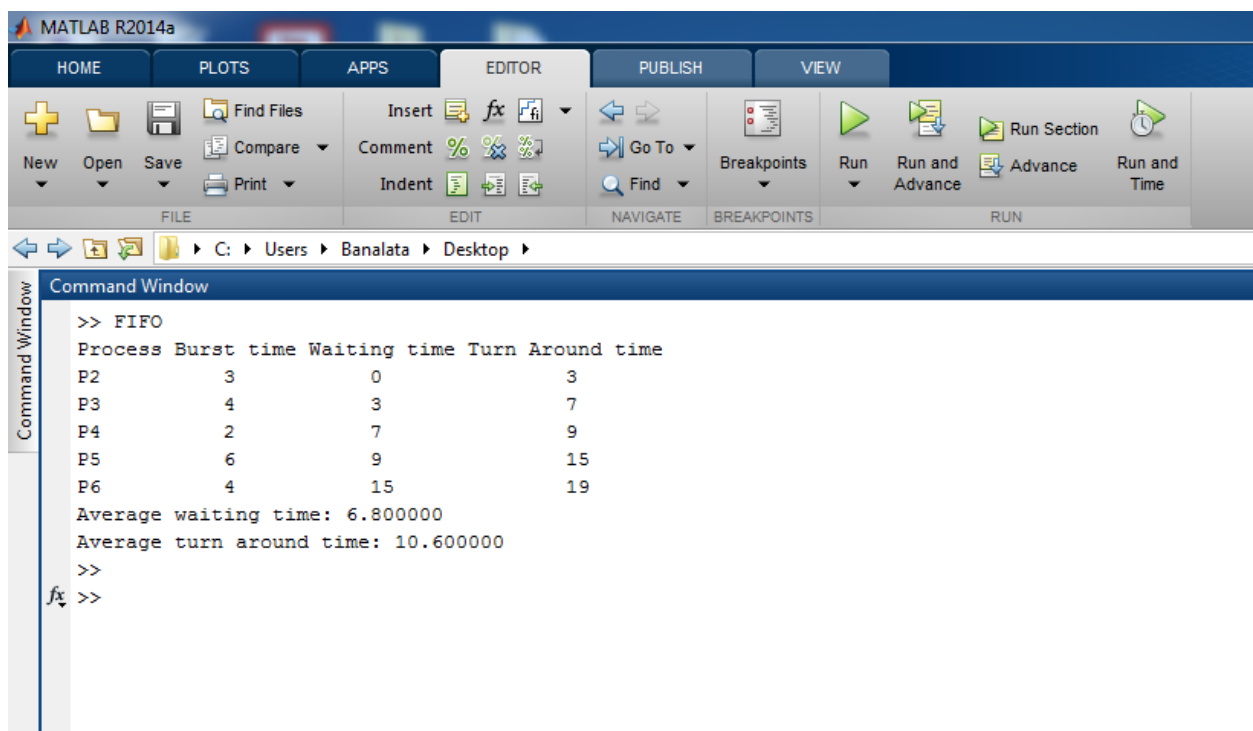
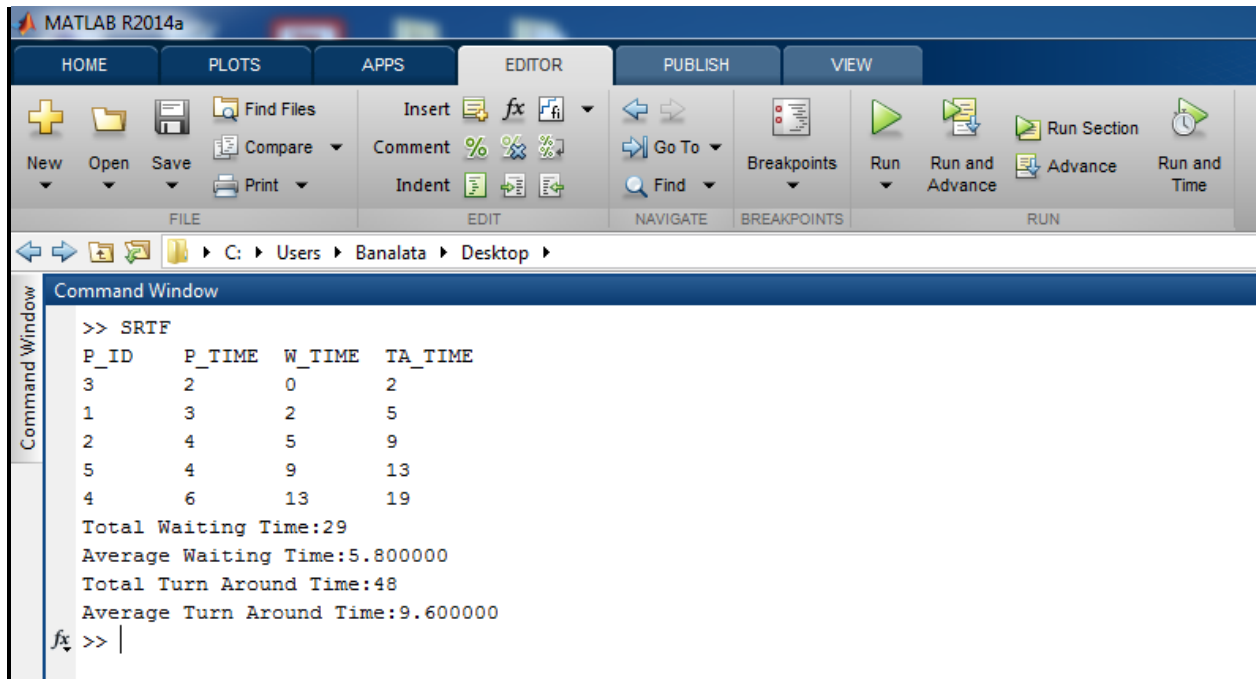


Fig 4.3 FIFO in matlab

SRTE:

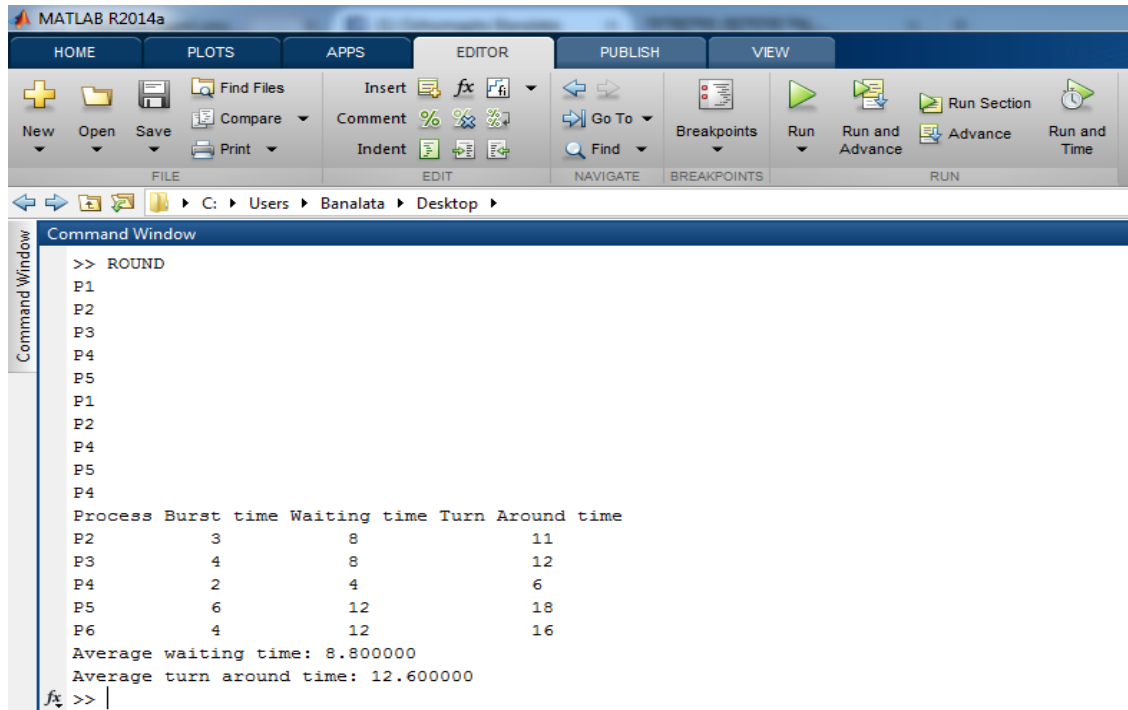


The image shows the MATLAB R2014a interface with the Command Window open. The Command Window displays the output of the SRTE function, which includes a table of process times and summary statistics.

```
>> SRTE
P_ID   P_TIME  W_TIME  TA_TIME
3       2        0        2
1       3        2        5
2       4        5        9
5       4        9       13
4       6       13       19
Total Waiting Time:29
Average Waiting Time:5.800000
Total Turn Around Time:48
Average Turn Around Time:9.600000
fx >> |
```

Fig 4.4 SRTE in matlab

Round Robin (RR):



The image shows the MATLAB R2014a Command Window with the following output:

```
>> ROUND
P1
P2
P3
P4
P5
P1
P2
P4
P5
P4
Process Burst time Waiting time Turn Around time
P2          3             8          11
P3          4             8          12
P4          2             4           6
P5          6            12          18
P6          4            12          16
Average waiting time: 8.800000
Average turn around time: 12.600000
fx >> |
```

Fig 4.5 Round Robin in matlab

Our analysis of three algorithms shows the different average waiting time. The FCFS is better for a small burst time. The SJF is better if the process comes to processor simultaneously. The last algorithm, Round Robin, is better to adjust the average waiting time desired.

FIFO and SJF not suitable for times sharing systems. On the other hand, Round Robin is suitable for time sharing systems. For this reason, we considered Round Robin is better suited than those algorithms for our further work.

We proposed a novel approach to scheduling for detecting SYN Flood attack: **Highest Residence Time Ejection (HRTE)**. HRTE mainly works in two phases. When it is in its first phase, it works exactly like round robin. But its second phase is ejects the job with the highest residence time. We chose a threshold from the burst time of all the process/job to determine.

4.4 Proposed Scheduling Algorithms: Highest Residence Time Ejection (HRTE):

This algorithm is a preemptive two-phase scheduling algorithm. This algorithm is useful for scenarios in which the service time of requests is unknown. According to this scheduling algorithm, while input queue isn't full, HRTE is in its first phase and acts exactly like round robin algorithm. But upon queue becomes full and arriving requests are blocked, HRTE switches to its second phase during which ejects the job with the highest residence time and assigns the released capacity to the arriving requests. HRTE remains in this phase until a free capacity is available in the waiting queue. In other words, it has some similarities with SRTF, but when remaining time is unknown SRTF cannot be used. In this situation we assume that those requests that have the longest duration in the past will have the longest remaining time, as well and hence will be rejected.

We tried to prevent SYN flood attacks using Round Robin and HRTE algorithm. We used service time of requests for this analysis. We consider that the request having highest residence time as attack request and normal residence time as regular request. We got some results and we showed the efficiency with the proposed algorithm with two parameters. They are AT, RT. We will describe them in the next chapter.

CHAPTER 3

Denial of Service (DOS) attack: TCP SYN Flood attack & Mitigation Methods

3.1 Introduction

Security has been always an important issue in communication and computation systems. In these systems security has different aspects. Sometimes the threat is in the form of disclosing of our confident information. In this case we use some security algorithms and protocols to protect the confidentiality. Security has become necessary in a world where more services are relying on internet technology. For this reason, it has attracted a lot of attention in various areas of communication networks.

There are several types of important attacks, such as the worm, virus, Trojan horse and especially Denial of Service (DoS), each of which causes crucial problems to usual business operations. In spite of extensive efforts to provide robustness for the systems against DoS attack, this attack is yet a serious problem on the Internet. Traditionally, DoS attacks aim at degrading the availability and quality of services, by consuming the service resources to make it unavailable. In doing this, DoS attacks may send to the victim a high-rate traffic that exhausts service resources. Statistical evaluations show that DoS ranks at the fourth place in the list of the most venomous attack classes against information systems. Recently, many efforts have been made, in parallel with the evolution of DoS attacks, in the field of prevention and detection in networking security. In terms of prevention, some of the approaches that have been proposed include egress or ingress filtering, disabling unused services, and honey pots. In other works a congestion pricing approach and a router-based technique have been employed to neutralize DoS attacks.

In this chapter we have discussed about types of DOS specially TCP syn and the methods used to prevent .Because the recent DoS attacks on popular web sites like Yahoo and eBay and their consequent disruption of services have exposed the vulnerability of the Internet to Distributed Denial of Service (DDoS) attacks . It has been shown that more than 90% of the DoS attacks use TCP. The TCP SYN flooding is the most commonly-used attack. Not only the Web servers but also any system connected to the Internet providing TCP-based network services, such as FTP servers or Mail servers is susceptible to the TCP SYN flooding attacks.

3.2 What Is a DoS Attack?

A DoS attack refers to a Denial of Service attack in which attackers use one computer and one Internet connection to send numerous requests to flood the target server. It is a malicious attempt that overloads resources to make them unavailable to other traffic, or at least to slow down their response to visitors.

Along with the rapid development of the online world, DoS attacks are becoming increasingly sophisticated, which makes them hard to be detected. They can even utilize the vulnerabilities of applications.

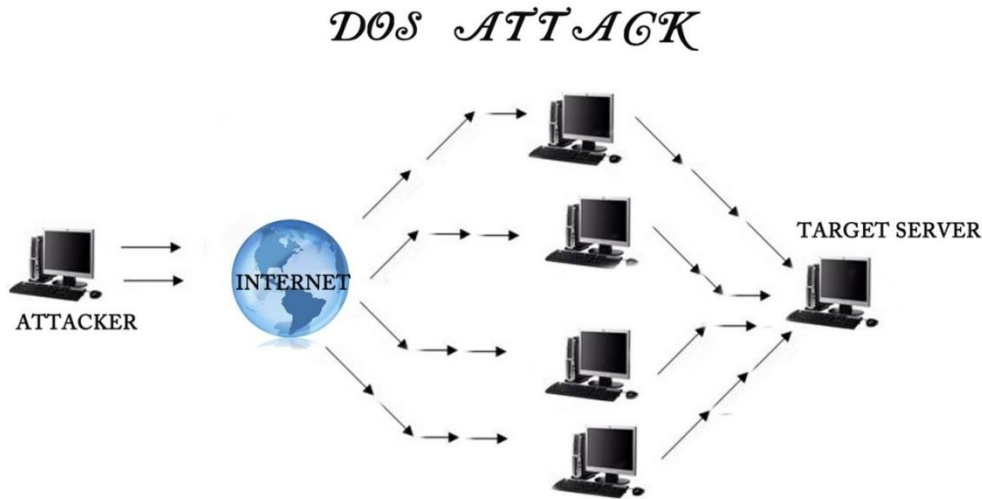


Fig 3.1 DOS attack

The targets and reasons for DoS attacks vary. The target can be a single computer, a port of a system, a network, and also some resources like bandwidth and disk space. Moreover, the attacks can be used to execute malware, cause errors and destroy the operating system. As for the reasons, it is proved that most attacks originate from people who are extremely unsatisfied about a service, cyber criminals and business competitors.

3.3 What Is a DDoS Attack?

The full name of a DDoS attack is a Distributed Denial of Service attack which uses a network of computers and connections distributed around the world to overload a service. These computers usually belong to a botnet, a large group of devices that are infected and hijacked by a malicious group or individual through involuntarily installed malware. Compared with DoS attacks, it is even harder to withstand DDoS attacks. DDoS is a type of DOS attack where multiple compromised systems, which are often infected with a Trojan, are used to target a single system causing a Denial of Service (DoS) attack. Victims of a DDoS attack consist of both the end targeted system and all systems maliciously used and controlled by the hacker in the distributed attack.

In a DDoS attack, the incoming traffic flooding the victim originates from many different sources – potentially hundreds of thousands or more. This effectively makes it impossible to stop

the attack simply by blocking a single IP address; plus, it is very difficult to distinguish legitimate user traffic from attack traffic when spread across so many points of origin.

3.4 What Is the Main Difference Between DoS and DDoS Attacks?

A Denial of Service (DoS) attack is different from a DDoS attack. The DoS attack typically uses one computer and one Internet connection to flood a targeted system or resource. The DDoS attack uses multiple computers and Internet connections to flood the targeted resource. DDoS attacks are often global attacks, distributed via bonnets. The most significant difference, as is mentioned above, is that in a DoS attack, the attackers use only one computer and one Internet connection, while those launching DDoS attacks use a globally distributed network of computers.

In addition, it is much more difficult to fight against DDoS attacks as there are hundreds or thousands of sources sending out requests to flood the target, especially when a website or server is under a specifically targeted DDoS attack. It is nearly impossible to block out the sources.

But in a DoS attack, if the incoming traffic is identified as being malicious instead of a normal traffic spike, hosts can take actions to absorb and attack and block the source as soon as it is identified. This kind of attacks can be stopped in a short time.

DoS and DDoS attacks both are significant security issues that can take down a whole server. For this reason, when selecting a web host, you should pay attention to the availability of DDoS protection. Although the attacks cannot be prevented, advanced technologies will contribute to the effective mitigation. Nowadays, an increasing number of web hosts, including the ones listed below, are making efforts continuously in improving the security of their servers and the hosted websites.[4]

3.5 Some Specific DOS /DDoS Attacks Types:

There are several types of Dos attack .Our research we have to find some types of DDoS/ DoS methods or attacks.

3.5.1 UDP flood attack

A **UDP flood attack** is a denial-of-service (DoS) attack using the User Datagram Protocol (UDP), a session less/connectionless computer networking protocol.

Using UDP for denial-of-service attacks is not as straightforward as with the Transmission Control Protocol (TCP). However, a UDP flood attack can be initiated by sending a large number of UDP packets to random ports on a remote host. As a result, the distant host will:

- Check for the application listening at that port;
- See that no application listens at that port;
- Reply with an ICMP Destination Unreachable packet.

Thus, for a large number of UDP packets, the victimized system will be forced into sending many ICMP packets, eventually leading it to be unreachable by other clients. The attacker(s) may also spoof the IP address of the UDP packets, ensuring that the excessive ICMP return packets do not reach them, and a minimizing their network location(s). Most operating systems mitigate this part of the attack by limiting the rate at which ICMP responses are sent. Software such as Low Orbit Ion Cannon and UDP Unicorn can be used to perform UDP flooding attacks.

3.5.2 Ping Flood (ICMP Flood):

Ping flood, also known as ICMP flood, is a common Denial of Service (DoS) attack in which an attacker takes down a victim's computer by overwhelming it with ICMP echo requests, also known as pings. The attack involves flooding the victim's network with request packets, knowing that the network will respond with an equal number of reply packets. Additional methods for bringing down a target with ICMP requests include the use of custom tools or code, such as hping and scapy.

An ICMP request requires the server to process the request and respond, so it takes CPU resources. Attacks on the ICMP protocol, including smurf attacks, ICMP floods, and ping floods take advantage of this by inundating the server with ICMP requests without waiting for the response. This attack seeks to overwhelm the server's ability to respond, thereby blocking valid requests. Since ICMP packets should be rare in a normal traffic situation, F5 BIG-IP Local Traffic Manager (LTM) and BIG-IP Advanced Firewall Manager (AFM) are able to mitigate ICMP floods by limiting the rate of all ICMP traffic, and then dropping all ICMP packets above this limit. BIG-IP LTM and BIG-IP AFM provide the ability to set a limit on the maximum number of ICMP packets to prevent the server from ever getting flooded.

This strains both the incoming and outgoing channels of the network, consuming significant bandwidth and resulting in a denial of service.

3.5.3 Teardrop Attack:

A teardrop attack is a denial-of-service (DoS) attack that involves sending fragmented packets to a target machine. Since the machine receiving such packets cannot reassemble them due to a bug in TCP/IP fragmentation reassembly, the packets overlap one another, crashing the target network device. This generally happens on older operating systems such as Windows 3.1x, Windows 95, Windows NT and versions of the Linux kernel prior to 2.1.63.

One of the fields in an IP header is the “fragment offset” field, indicating the starting position, or offset, of the data contained in a fragmented packet relative to the data in the original packet. If the sum of the offset and size of one fragmented packet differs from that of the next fragmented packet, the packets overlap. When this happens, a server vulnerable to teardrop attacks is unable to reassemble the packets - resulting in a denial-of-service condition.

While much more popular on older versions of Windows, the teardrop attack is also possible on Windows 7 and Windows Vista machines that have SMB enabled. This attack causes fragmented packets to overlap one another on the host receipt; the host attempts to reconstruct them during the process but fails. Gigantic payloads are sent to the machine that is being targeted, causing system crashes.

3.5.4 Peer-to-peer attacks:

A flaw in the design of a popular peer-to-peer network software has given attackers the ability to create massive denial-of-service attacks that can easily overwhelm corporate Web sites, a security firm warned last week. A peer-to-peer (P2P) network is a distributed network in which individual nodes in the network (called “peers”) act as both suppliers (seeds) and consumers (leeches) of resources, in contrast to the centralized client–server model where the client server or operating system nodes request access to resources provided by central servers.

Peers make a portion of their resources, such as processing power, disk storage or network bandwidth, directly available to other network participants, without the need for central coordination by servers or stable hosts. Peers are both suppliers and consumers of resources, in contrast to the traditional client-server model in which the consumption and supply of resources is divided. Emerging collaborative P2P systems are going beyond the era of peers doing similar

things while sharing resources, and are looking for diverse peers that can bring in unique resources and capabilities to a virtual community thereby empowering it to engage in greater tasks beyond those that can be accomplished by individual peers, yet that are beneficial to all the peers.

While P2P systems had previously been used in many application domains, the architecture was popularized by the file sharing system Napster, originally released in 1999. The concept has inspired new structures and philosophies in many areas of human interaction. In such social contexts, peer-to-peer as a meme refers to the egalitarian social networking that has emerged throughout society, enabled by Internet technologies in general.

3.5.5 TCP SYN Flood Attacks:

A SYN flood DDoS attack exploits a known weakness in the TCP connection sequence (the “three-way handshake”), wherein a SYN request to initiate a TCP connection with a host must be answered by a SYN-ACK response from that host, and then confirmed by an ACK response from the requester. In a SYN flood scenario, the requester sends multiple SYN requests, but either does not respond to the host’s SYN-ACK response, or sends the SYN requests from a spoofed IP address. Either way, the host system continues to wait for acknowledgement for each of the requests, binding resources until no new connections can be made, and ultimately resulting in denial of service.

In this type of attacks, the attacker sends a vast amount of “Please start a connection with me” packets but no follow-up packets. When a server receives such packets, it allocates certain memory resources for the new session and hence if there are no follow up packets and a lot of request packets coming in, the server resources are exhausted and it is not able to allocate resources for real traffic coming in. Connection flood attacks happen when acknowledgement packet is sent to the server to complete a three way handshake (that completes the setting up of connection) but no more packets are sent to the server, causing unused connections.

The IPS systems first analyze the source of such packets. If the source has had previous transactions and is trust worthy, then those packets are allowed to pass through. For untrustworthy sources, it attaches a cookie to the response messages and challenges the source to send back a response. If there is no response, then that session is dropped and packets from those sources are no longer allowed. Even if a source first builds a trustworthy relationship, before starting an attack, the sampling of the ratio of the SYN packets and acknowledgement packets are used to identify attacks. Connection flood attacks are mitigated by limiting the number of TCP/UDP connections opened per client. These connections are generally limited per application port.

In addition to these techniques, the bandwidth available per application is sometimes limited so that a DoS attack on that application becomes ineffective, as the packets are slowed down if there is an abnormal rise in traffic and this also ensures that the other applications have a certain bandwidth reserved. Some vendors also have an Access Control List to allow only certain pre defined applications and denying all other types of traffic.

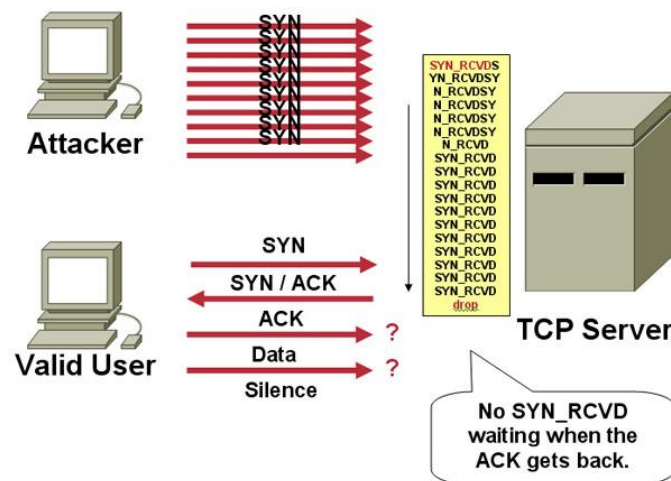


Fig 3.2 TCP SYN Flood attack

3.5.6 Ping of Death Attack:

Ping of Death (a.k.a. PoD) is a type of Denial of Service (DoS) attack in which an attacker attempts to crash, destabilize, or freeze the targeted computer or service by sending malformed or oversized packets using a simple ping command.

While PoD attacks exploit legacy weaknesses which may have been patched in target systems. However, in unpatched systems, the attack is still relevant and dangerous. Recently, a new type of PoD attack has become popular. This attack, commonly known as a Ping flood, the targeted system is hit with ICMP packets sent rapidly via ping without waiting for replies. The size of a correctly-formed IPv4 packet including the IP header is 65,535 bytes, including a total payload size of 84 bytes. Many historical computer systems simply could not handle larger packets, and would crash if they received one. This bug was easily exploited in early TCP/IP implementations in a wide range of operating systems including Windows, Mac, UNIX, Linux, as well as network devices like printers and routers.

The TCP connection management protocol sets a position for a classic Denial of Service (DoS) attack, called the SYN flooding attack. In this attack attacker sends a large number of TCP SYN segments, without completing the third handshaking step to quickly exhaust connection resources of the victim server. Therefore it keeps TCP from handling legitimate requests. SYN flooding attack can be viewed metaphorically as result of an unfair scheduling that gives more opportunity to attack requests but prevents legal connections from getting services. In this reason, we present TCP_SYN flood.

3.6 TCP-SYN flood:

A SYN flood is a form of denial-of-service attack in which an attacker sends a succession of SYN requests to a target's system in an attempt to consume enough server resources to make the system unresponsive to legitimate traffic.

The attack involves having a client repeatedly send SYN (synchronization) packets to every port on a server, using fake IP addresses. When an attack begins, the server sees the equivalent of multiple attempts to establish communications. The server responds to each attempt with a SYN/ACK (synchronization acknowledged) packet from each open port, and with a RST (reset) packet from each closed port.

SYN/TCP Flood

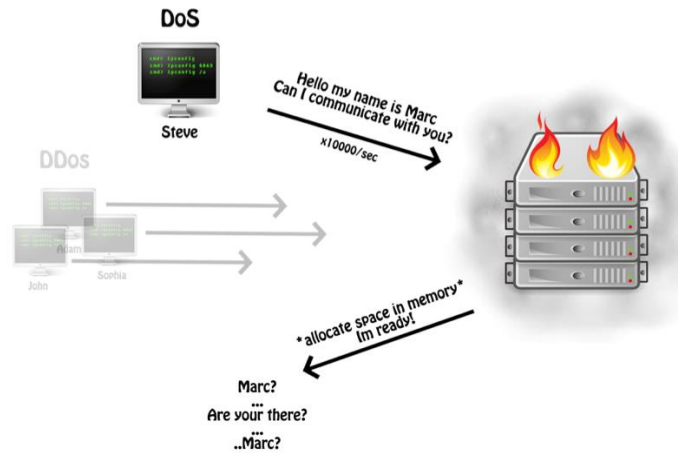


Fig 3.3 TCP SYN Flood

The aim of denial of service attacks is to consume a large amount of resources, thus preventing legitimate users from receiving service with some minimum performance. TCP SYN flooding exploits the TCP's three-way handshake mechanism and its limitation in maintaining half-open connections. Any system connected to the Internet and providing TCP-based network services, such as a Web server, FTP server, or mail server, is potentially subject to this attack. A TCP connection starts with the client sending a SYN message to the server, indicating the client's intention to establish a TCP connection.

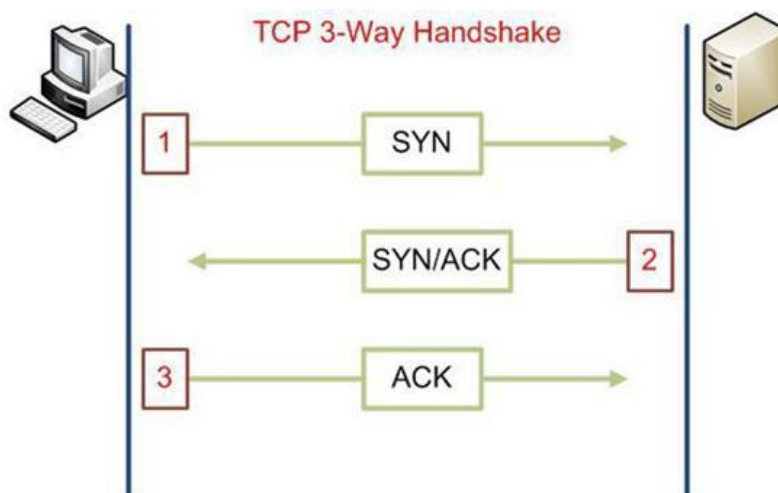


Fig 3.4 Three-way Handshake

When a client labors to start a TCP connection to a server, first, the client needs a connection by distribution an SYN packet to the server. Then, the server returns a SYN-ACK, to the client. Lastly, the client admits the SYN-ACK with an ACK, at which point the connection is recognized and data transfer commences. In an SYN flooding attack, attackers use this protocol to their advantage. The attacker directs a large number of SYN packets to the server. All of these packets have to be touched like a connection request by the server, so the server must response with a SYN-ACK. The attacker then has two choices. One is just not to response to the SYN-ACK, which will reason the server to have a half-open connection. This would let the server to chunk any further packets from the attacks IP address, ending the attack hastily.

Then again, the attacker parodies the IP address of some unwary client. The server rationally responses to this IP address, but the genuine client really exist in at this IP address will weakening this SYN-ACK as it did not pledge the connection. The result is that the server is left waiting for a reply from a large amount of connections. Since reserve of any system is imperfect, then, there are a limited number of connections a server can handle. Once all of these are in use, waiting for connections that will not ever come, no new connections can be made whether valid or not. It is clear that though this is a conveyance layer attack, it touches all TCP-based requests in the victim server. When the server cannot handle new connections, any request that tries to establish TCP connections with the server, fails in its effort. Note that SYN flooding attacks goal

To use TCP buffer space and do not touch the parameters such as link bandwidth, dispensation capitals and so on.[2]

3.7 Methods Used To Prevent SYN Flood:

SYN flooding attack is the most widespread of the DoS attacks. In these attacks normal SYN packets can't be distinguished from the SYN attack packets. SYN attacks are one of the major types of problems in the computer network security. Because they include many other type of attacks they are one of the most frequently used attack methods. In general, SYN attacks are used to block access to the computer networks or personal computers. Detection SYN flood we can use some methods .Such as.

3.7.1 Intrusion Detection Systems (IDS):

Intrusion Detection is defined as the process of intelligently monitoring the events occurring in a computer system or network and analyzing them for signs of violations of the security policy. The primary aim of Intrusion Detection Systems (IDS) is to protect the availability, confidentiality and integrity of critical networked information systems. Intrusion Detection Systems (IDS) are defined by both the method used to detect attacks and the placement of the

IDS on the network. IDS may perform either misuse detection or anomaly detection and may be deployed as either a network-based system or a host-based system. These results in four general groups: misuse-host, misuse-network, anomaly-host and anomaly-network. Misuse detection relies on matching known patterns of hostile activity against databases of past attacks. They are highly effective at identifying known attack and vulnerabilities, but rather poor in identifying new security threats. Anomaly detection will search for something rare or unusual by applying statistical measures or artificial intelligence methods to compare current activity against historical knowledge. Common problems with anomaly-based systems are that, they often require extensive training data for artificial learning algorithms, and they tend to be computationally expensive, because several metrics are often maintained, and need to be updated against every systems activity. Some IDS combine qualities from all these categories (usually implementing both misuse and anomaly detection) and are known as hybrid systems.

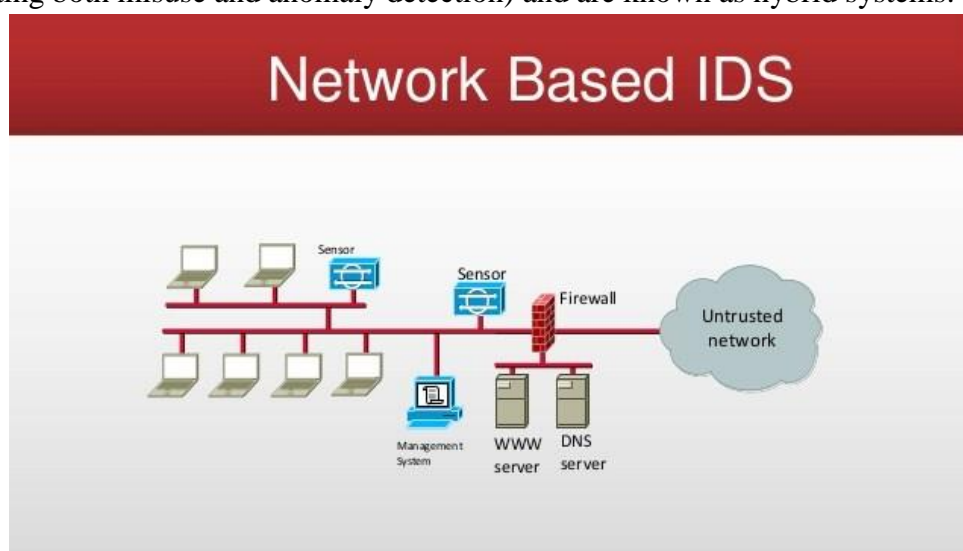


Fig 3.5 Intrusion Detection Systems (IDS)

3.7.2 Fuzzy Logic:

Applying fuzzy methods for the development of IDS yield some advantages, compared to the classical approach. Therefore, Fuzzy logic techniques have been employed in the computer security field since the early 90's. The fuzzy logic provides some flexibility to the uncertain problem of intrusion detection and allows much greater complexity for IDS. Most of the fuzzy IDS require human experts to determine the fuzzy sets and set of fuzzy rules. These tasks are time consuming. However, if the fuzzy rules are automatically generated, less time would be consumed for building a good intrusion classifier and shortens the development time of building or updating an intrusion classifier. The model suggested in building rare class prediction models for identifying known intrusions and their variations and anomaly/outlier detection schemes for detecting novel attacks whose nature is unknown. The latest in fuzzy is to use the Markov model.

As suggested in a Window Markov model is proposed, the next state in the window equal evaluation to be the next state of time t, so they create Fuzzy window Markov model. As discussed in proposes a technique to generate fuzzy classifiers using genetic algorithms that can detect anomalies and some specific intrusions. The main idea is to evolve two rules, one for the normal class and other for the abnormal class using a profile data set with information related to the computer network during the normal behavior and during intrusive (abnormal) behavior. Fuzzy preference relation is another method applied to intrusion detection based on fuzzy satisfaction function. This is applied for comparison of attack signatures.

Fuzzy Logic Example

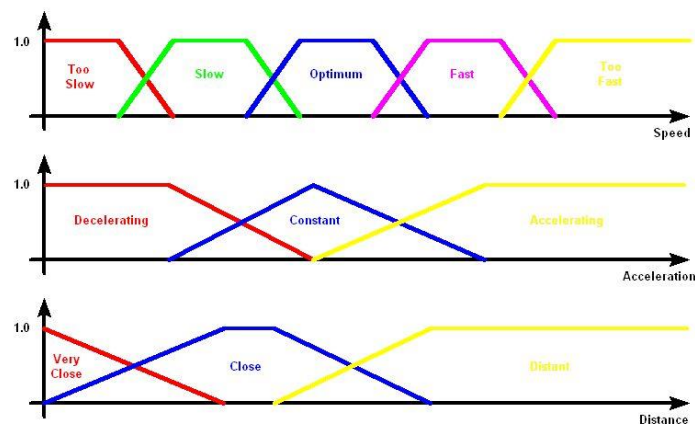


Fig 3.6 Fuzzy Logic

3.7.3 Router Configuration:

Configure router to protect TCP servers from TCP SYN-flooding attacks, a type of denial-of-service attack. This is accomplished by configuring the Cisco IOS feature known as TCP Intercept. For a complete description of TCP Intercept commands. To locate documentation of other commands that appear in this chapter, use the command reference master index or search online. To identify the hardware platform or software image information associated with a feature, use the Feature Navigator on Cisco.com to search for information about the feature or refer to the software release notes for a specific release. For more information, see the chapter "Identifying Supported Platforms" section in the "Using Cisco IOS Software." The TCP intercept feature implements software to protect TCP servers from TCP SYN-flooding attacks, which are a type of denial-of-service attack.

The TCP intercept feature helps prevent SYN-flooding attacks by intercepting and validating TCP connection requests. In intercept mode, the TCP intercept software intercepts TCP synchronization (SYN) packets from clients to servers that match an extended access list. The software establishes a connection with the client on behalf of the destination server, and if successful, establishes the connection with the server on behalf of the client and knits the two half-connections together transparently. Thus, connection attempts from unreachable hosts will never reach the server. The software continues to intercept and forward packets throughout the duration of the connection. The number of SYNs per second and the number of concurrent connections peroxide depends on the platform, memory, processor, and other factors

In the case of illegitimate requests, the software's aggressive timeouts on half-open connections and its thresholds on TCP connection requests protect destination servers while still allowing valid requests.

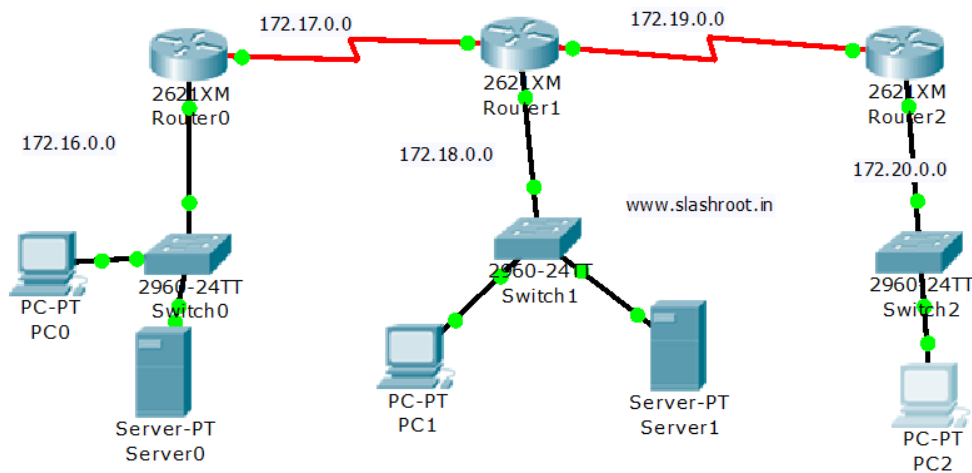


Fig 3.7 Router Configurations

3.7.4 Scheduling Algorithm:

In computing, **scheduling** is the method by which work specified by some means is assigned to resources that complete the work. The work may be virtual computation elements such as threads, processes or data flows, which are in turn scheduled onto hardware resources such as processors, network links or expansion cards.

A scheduler is what carries out the scheduling activity. Schedulers are often implemented so they keep all computer resources busy (as in load balancing), allow multiple users to share system

resources effectively, or to achieve a target quality of service. Scheduling is fundamental to computation itself, and an intrinsic part of the execution model of a computer system; the concept of scheduling makes it possible to have computer multitasking with a single central processing unit (CPU).

A scheduler may aim at one of many goals, for example, maximizing throughput (the total amount of work completed per time unit), minimizing response time (time from work becoming enabled until the first point it begins execution on resources), or minimizing latency (the time between work becoming enabled and its subsequent completion), maximizing fairness (equal CPU time to each process, or more generally appropriate times according to the priority and workload of each process). In practice, these goals often conflict (e.g. throughput versus latency), thus a scheduler will implement a suitable compromise. Preference is given to any one of the concerns mentioned above, depending upon the user's needs and objectives.

A Process Scheduler schedules different processes to be assigned to the CPU based on particular scheduling algorithms.

- First-Come, First-Served (FCFS) Scheduling
- Shortest-Job-Next (SJN) Scheduling
- Priority Scheduling
- Shortest Remaining Time
- Round Robin(RR) Scheduling
- Multiple-Level Queues Scheduling

These algorithms are either **non-preemptive or preemptive**. Non-preemptive algorithms are designed so that once a process enters the running state; it cannot be preempted until it completes its allotted time, whereas the preemptive scheduling is based on priority where a scheduler may preempt a low priority running process anytime when a high priority process enters into a ready state.

3.8 Our Approach to SYN flood

Our research focuses on finding an effective method to mitigate the effects of SYN flooding attack by proposing a novel scheduling method which can also act as a defense against this type of attacks. To detect TCP SYN flooding attack we tested the different scheduling approach to schedule the tasks, specifically, SYN requests sent to a server. In order to detect TCP SYN flooding, scheduling algorithms work based on several job features such as processing time, priority, due date, and so regarding their design goals. After testing at least 3 different approaches, we came to the conclusion that Round Robin was the most effective scheduling approach for a server to manage its tasks which is not under any attack. Therefore, we proposed a novel scheduling algorithm named “Highest Residence Time Ejection” (HRTE) algorithm. Using this algorithm on a server which has been attacked by TCP-SYN flood, we were able to see an improvement in the performance of the server under attack. In this section, we first present some of these scheduling algorithms and then present a novel scheduling algorithm which is used to design a defense strategy against SYN flooding attacks.

Chapter 5

Our Proposed Scheme to Prevent SYN Flood: HRTE Algorithm

5.1 Introduction:

TCP SYN flooding attacks, which has been well-known to the community for several years. Various countermeasures against these attacks have been proposed. In this type of attack, the attacker sends a vast amount of “Please start a connection with me” packets but no follow-up packets. When a server receives such packets, it allocates certain memory resources for the new session and hence if there are no follow up packets and a lot of request packets coming in, the server resources are exhausted and the server cannot handle new connections. In this case, we can use scheduling algorithm for preventing this attack.

A scheduler is what carries out the scheduling activity. Schedulers are often implemented so they keep all computer resources busy (as in load balancing), allow multiple users to share system resources effectively, or to achieve a target quality of service. Scheduling is fundamental to computation itself, and an intrinsic part of the execution model of a computer system; the concept of scheduling makes it possible to have computer multitasking with a single central processing unit .

According to the scheduling algorithms, we proposed a scheduling algorithm which is Highest Residence Time Ejection (HRTE) algorithm.

5.2 How our algorithm (HRTE) works:

HRTE algorithm is useful for scenarios in which the service time of requests is unknown. This algorithm is a two-phase scheduling algorithm. It is a preemptive. According to this scheduling algorithm, while input queue isn't full, HRTE is in its first phase and acts exactly like round robin algorithm. But upon queue becomes full and arriving requests are blocked, HRTE switches to its second phase during which ejects the job with the highest residence time and assigns the released capacity to the arriving requests. HRTE remains in this phase until a free capacity is available in the waiting queue. In other words, it has some similarities with SRTF, but when remaining time is unknown SRTF cannot be used. In this situation we assume that those requests that have the longest duration in the past will have the longest remaining time, as well and hence will be rejected. [2]

5.3 The parameter used to determine efficiency of algorithm:

When a connection request arrives at a TCP-based server, receives a buffer space of the backlog queue upon finding an inactive buffer space and is blocked otherwise. Now, consider a server under the SYN flooding attacks. Assume that in this computer each half-open connection is held for at most a period of holding time (h), and at most a number of maximum half-open connections (m) are allowed. We assume that a half-open connection for a regular request packet

is held for a chance time which is exponentially distributed with parameter μ . The arrivals of the regular request packets and the attack packets are both Poisson processes with rates λ and λ , respectively. The two arrival processes are independent. Obviously, when the system is under attack then number of pending connections increases and in a point in which there is no more room for pending connection to be saved the arriving requests will be blocked. In the other words, when a server is under SYN flooding attacks, half-open connections quickly consume all the memory allocated for the pending connections and prevent the victim from further accepting new requests, leading to the well-known buffer overflow problem.

We believe that the defense against this attack can be considered as a queue scheduling algorithm that differentiates attack requests from regular requests and then ejects the attack requests. This defense scheme, called HRT_SYN, tries to block attack connections and to prevent the system from allocating buffer space to attack connections. To this end, we use the proposed scheduling algorithm and keeps attack request from occupying system resources for a long time. According to this approach while there is free capacity for coming connection requests they will be accepted and inserted in the queue. But, when a connection arrives and faces with a full queue, then the HRTE scheduling algorithm is invoked and ejects the connection with the highest residence time. The ejected connection is likely an attack request that leaves the system to make the capacity free. . To measure efficiency of defense algorithms we use three parameters. RT and AT:

RT:

RT is Regular requests residence Time that can be described as mean ratio of the occupied resources by regular connections to total available connection resources. It is computed by using the following equation

$$RT = \frac{\sum \text{All regular connections duration}}{\text{Simulation time} * m}$$

In which, m is maximum number of request /queue capacity that are allowed to be established in the system.

AT:

AT is Attack residence Time that can be described as mean ratio of connection opportunistic that is occupied by Attack connections. It is computed by using following equation

$$AT = \frac{\sum \text{All attack connections duration}}{\text{Simulation time} * m}$$

This HRTE algorithm can be described by flowchart of Figure:

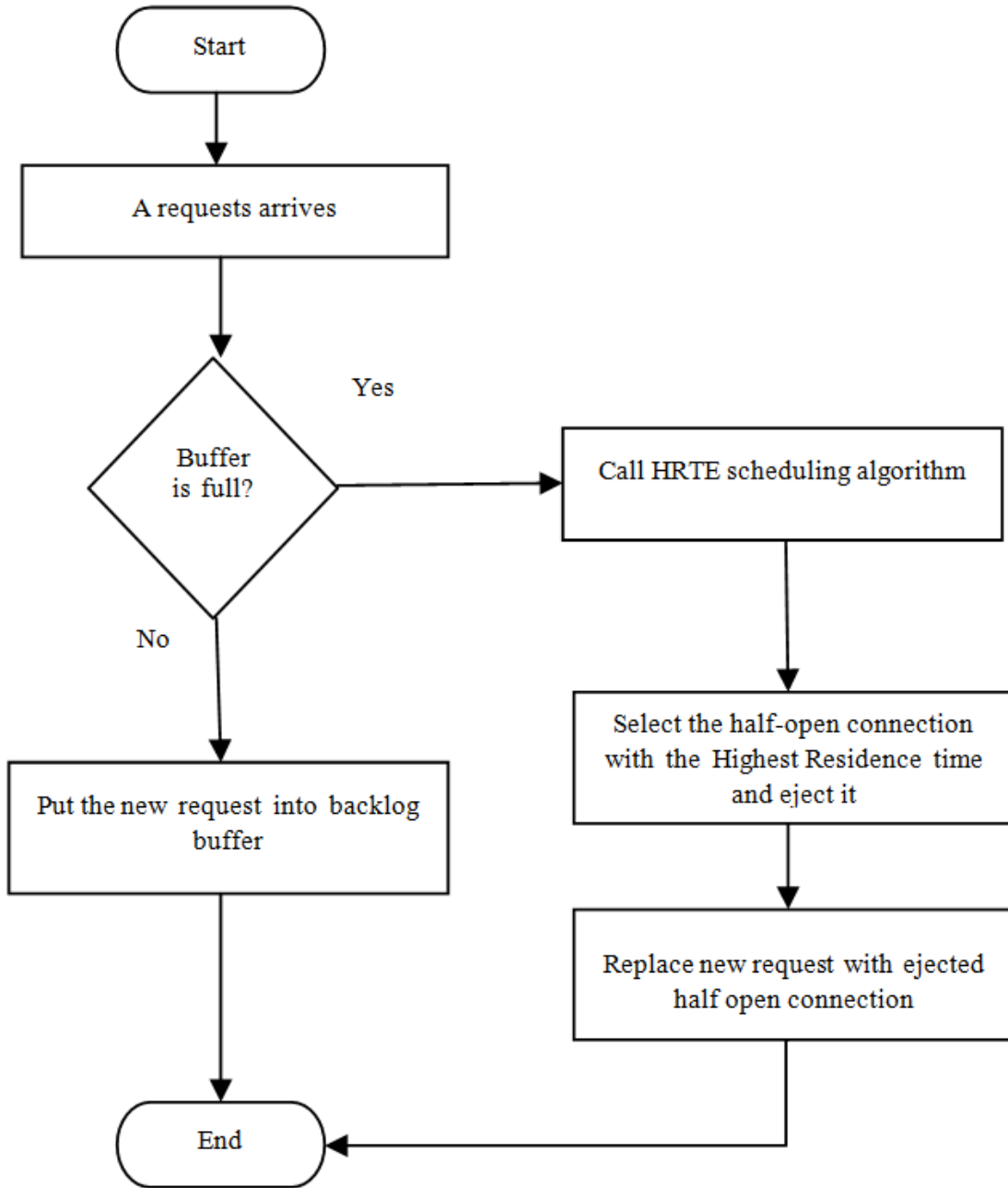


Fig 5.1 HRTE_SYN in operation

We considered a queue capacity $m=20$ and we considered 10 TCP SYN request with different turnaround time and burst time as our experiment data. We create three scenarios for TCP SYN request, scenarios are

- High attack
- Medium attack
- Low attack

In those three scenarios we used each average turnaround time for selecting attack request and regular request. If request value is greater than average turnaround time, then we considered it Attack request otherwise request value is less than average turnaround time then we considered it Regular request.

High attack:

In these scenarios from 10 requests, there are 8 attack request and 3 regular requests. We choose 10 process and they were specific Burst time for this Algorithm. Our process and its burst time are here.

Process	Burst time	Arrival time
P1	3	0
P2	4	0
P3	5	0
P4	8	0
P5	6	0
P6	5	0
P7	7	0
P8	9	0
P9	4	0
P10	2	0

Table 5.1: Table of High attack

Medium attack:

In these scenarios from 10 requests, there are 4 attack request and 6 regular requests. We choose 10 process and they were specific Burst time for this Algorithm. Our process and its bust time are here.

Process	Burst time	Arrival time
P1	3	0
P2	4	0
P3	2	0
P4	1	0
P5	4	0
P6	5	0
P7	7	0
P8	9	0
P9	4	0
P10	2	0

Table 5.2: Table of Medium attack

Low attack:

In these scenarios from 10 requests, there are 2 attack request and 8 regular requests. We choose 10 process and they were specific Burst time for this Algorithm. Our process and its bust time are here.

Process	Burst time	Arrival time
P1	1	0
P2	1	0
P3	1	0
P4	1	0
P5	1	0
P6	1	0
P7	1	0
P8	2	0
P9	8	0
P10	9	0

Table 5.3: Table of Low attack

We have used the same process with same request time and implemented PSO algorithm to compare the strategies. As we know PSO is better for handling servers with more dynamic allocation systems but we targeted servers having limited capacity and our analysis shows better results with HRTE algorithm.

Chapter 6

Implementation, Simulation and Comparison Evaluations

6.1 Simulation Result:

In this section, to prevent TCP SYN flood we implement three scenarios. For this purpose three scenarios of simulations are presented. In first scenario attack intensity is considered as $k = 0.2-0.3$ which means High attack intensity, in second one it is considered as $k = 0.5-0.6$ which means medium attack intensity, in third scenario it is set as $k = 0.7-0.8$ which means Low attack intensity and finally in three scenario of our simulations, we consider a variable attack intensity.

Low attack:

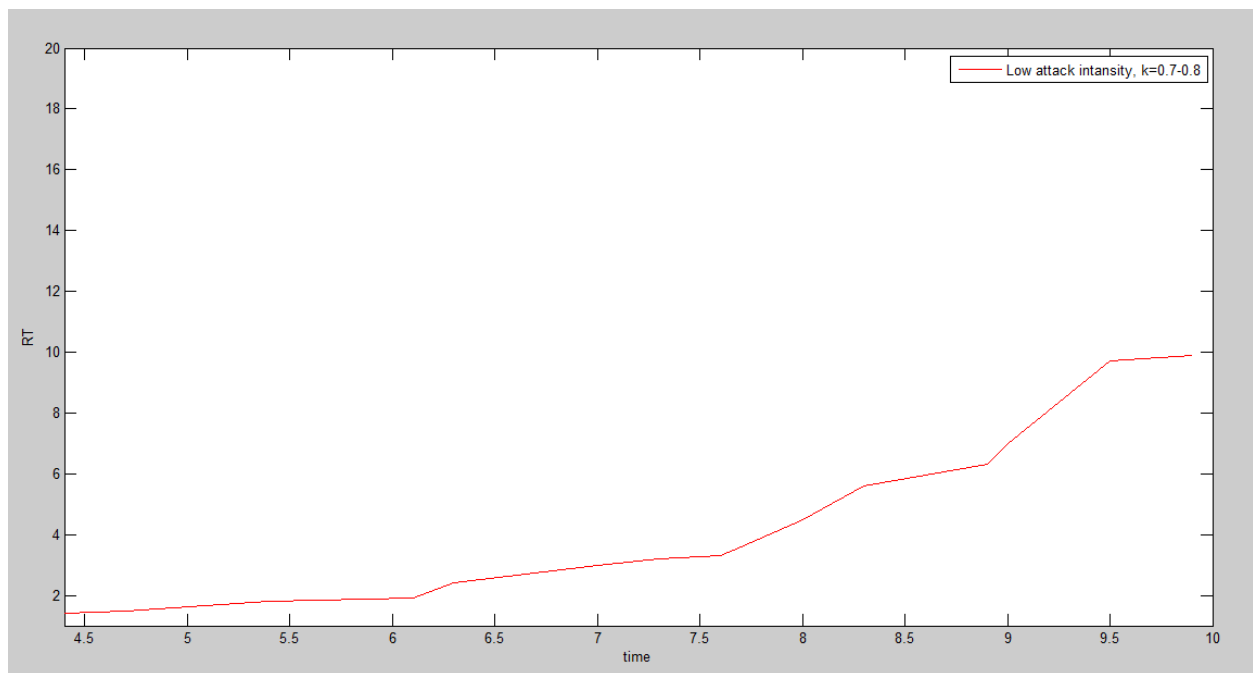


Fig 6.1 Regular request residence (RT) by HRTE_SYN in low attack intensity

In the figure we have used $k = 0.7-0.8$ which defines low attack intensity. In low attack intensity, the RT increases slightly for the requests, which means the waiting time for the non-attack requests increases when we are using our algorithm.

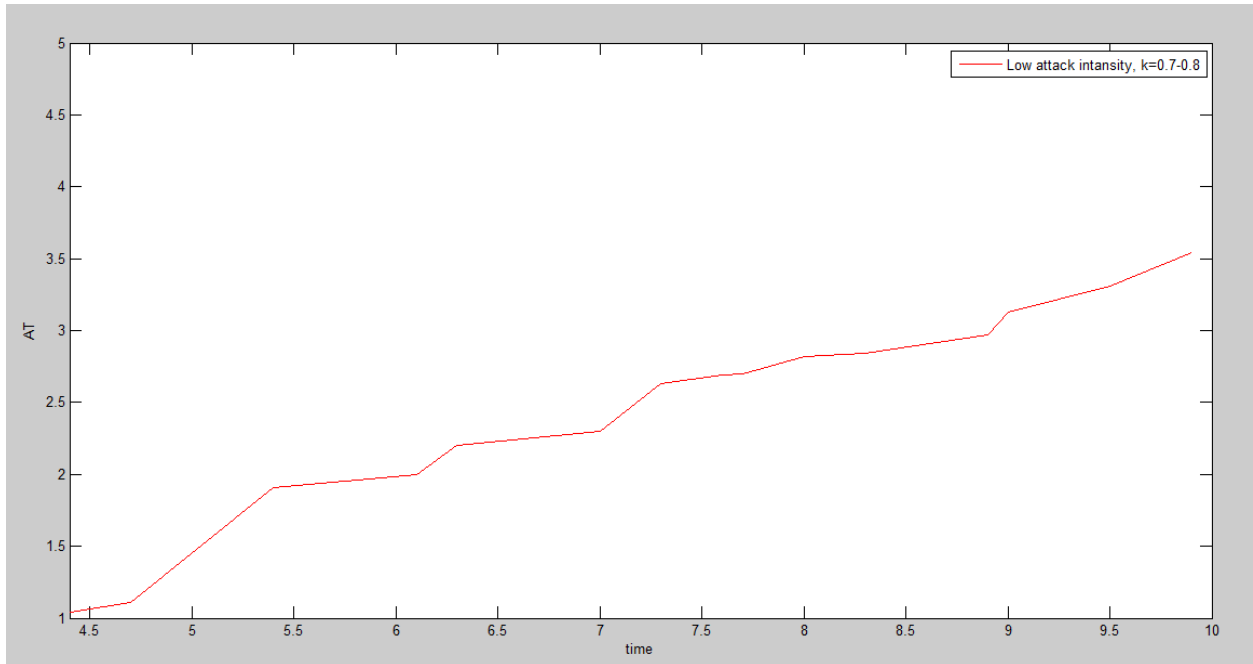


Fig 6.2 Attack request residence (AT)by HRTE_SYN in low attack intensity

In the figure we have used $k=0.7-0.8$ which defines low attack intensity. When attack intensity is low, the waiting time for attack requests slightly increases as there are limited numbers of attacks.

Medium attack:

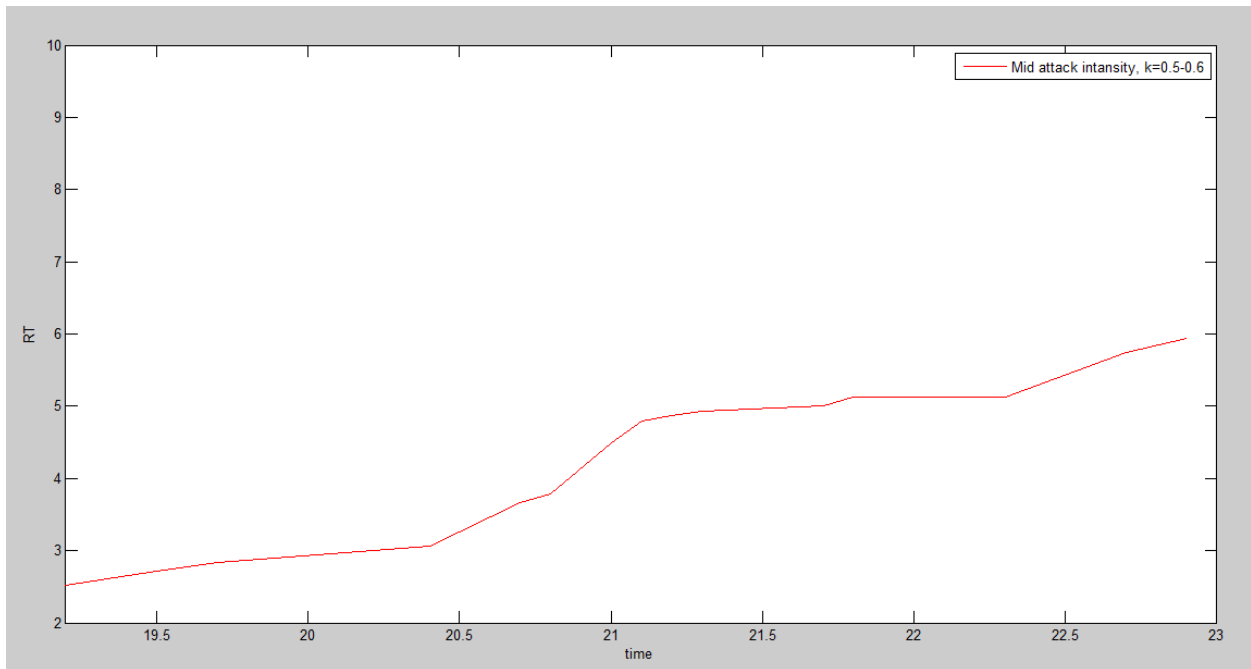


Fig 6.3 Regular request residence (RT)by HRTE_SYN in medium attack intensity

In the figure we have used $k=0.5-0.6$ which defines medium attack intensity. In medium attack intensity, the RT increases little bit for the requests, which means the waiting time for the non-attack requests increases when we are using our algorithm.

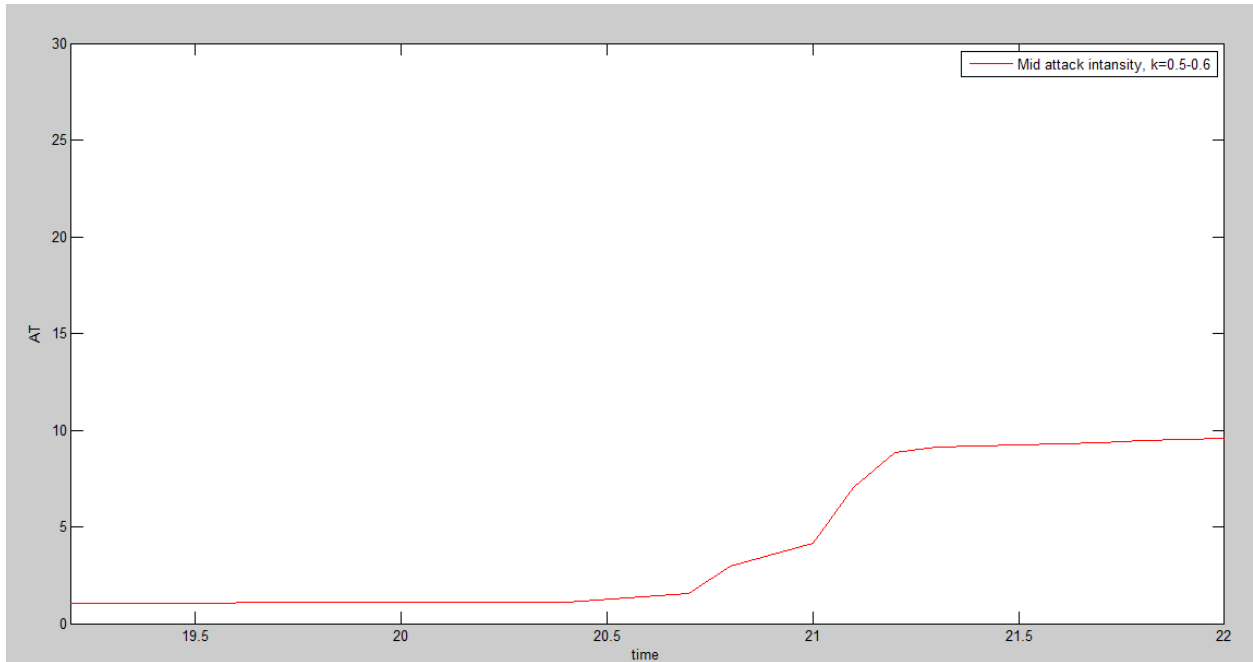


Fig 6.4 Attack request residence (AT) by HRTE_SYN in medium attack intensity

In the figure we have used $k=0.5-0.6$ which defines medium attack intensity. When attack intensity is medium, the AT is reduced as the number of attack increase.

High Attack:

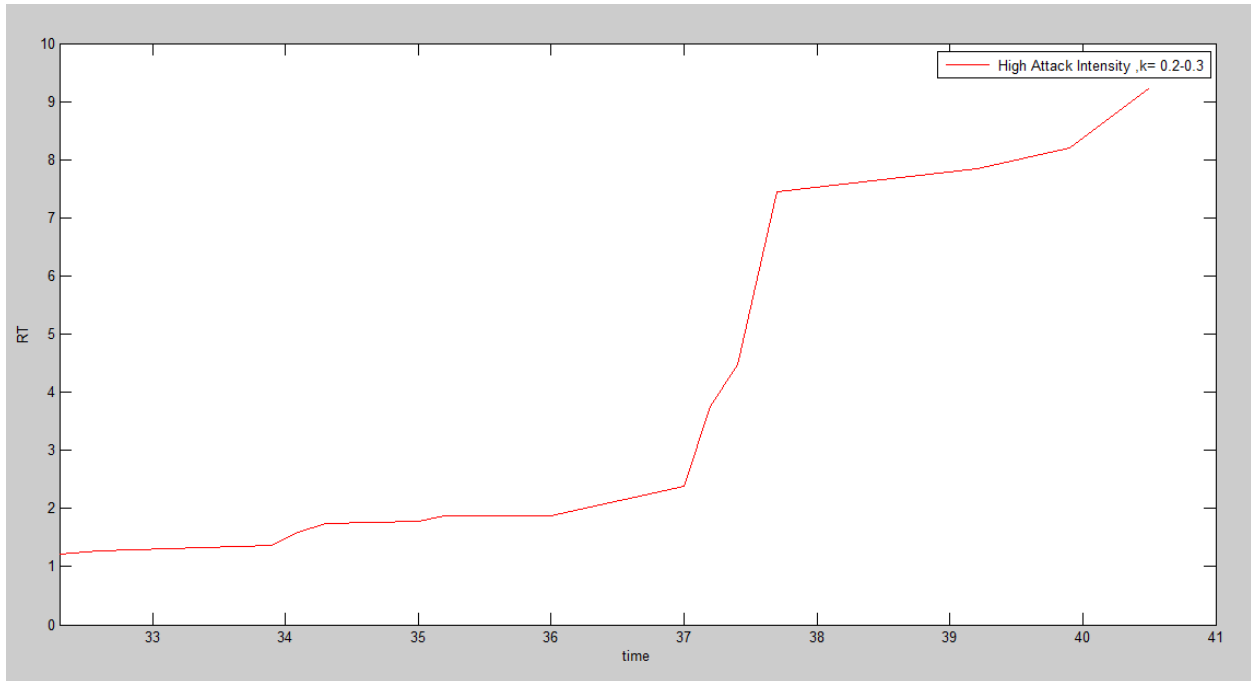


Fig 6.5 Regular request residence (RT)by HRTE_SYN in high attack intensity

In the figure we have used $k=.2-.3$ which defines high attack intensity. In high attack intensity, the RT increases for the requests, which means the waiting time for the non-attack requests increases when we are using our algorithm.

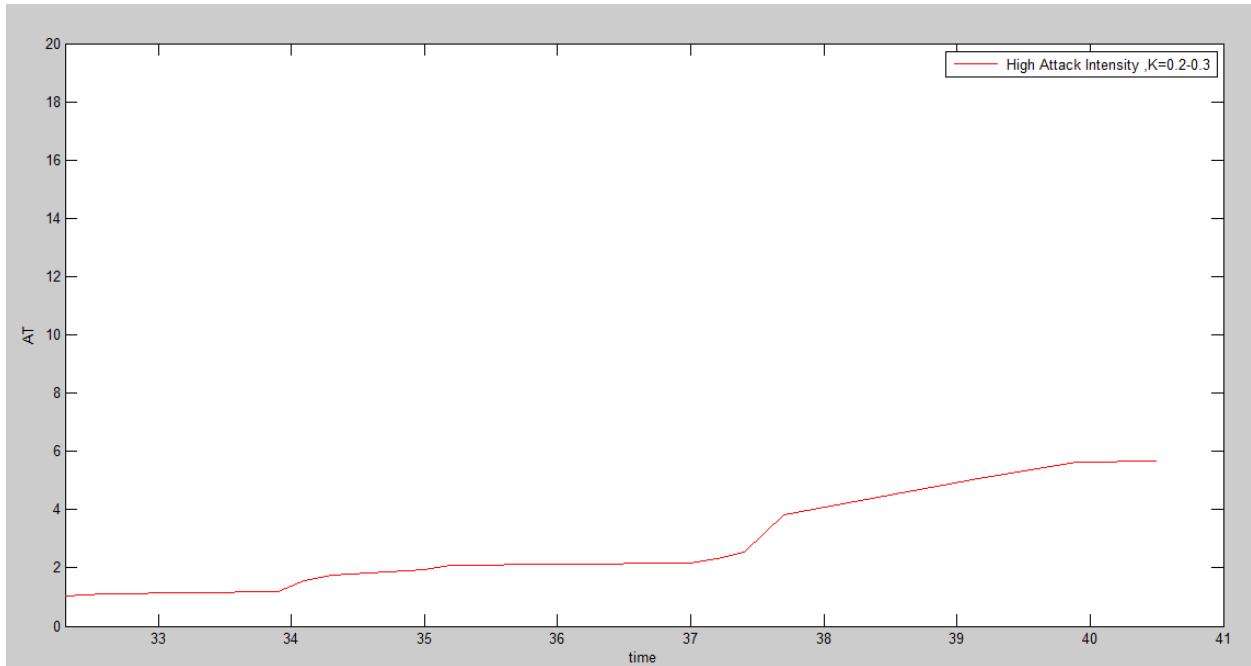


Fig 6.6 Attack request residence (AT) by HRTE_SYN in high attack intensity

In the figure we have used $k=0.2-0.3$ which defines high attack intensity. When attack intensity is high, the AT is further reduced as the number of attack increase.

6.2 Comparison of HRTE Algorithm with Particle Swarm Optimization (PSO) Algorithm

6.2.1 PSO algorithm:

Particle swarm optimization is a search algorithm that has been inspired from bird flocking and fish schooling. This population based algorithm has been designed and introduced by Kennedy and Eberhart in 1995. The basic PSO has found many successful applications in a number of problems including standard function optimization problems, solving permutation problems and

training multi-layer neural networks. The PSO algorithm contains a swarm of particles in which each particle indicates a potential solution. The particles fly through a multidimensional search space in which the position of each particle is adjusted according to its own experience and the experience of its neighbors. PSO system combines local search methods (through self experience) with global search methods (through neighboring experience), attempting to balance exploration and exploitation.

6.2.2 How PSO works:

Consider a server offering some TCP-based services that is subject to SYN flooding attacks. As mentioned before, SYN flooding attack uses the 3-way handshaking protocol running in the TCP connection establishment phase. In a SYN flooding attack, attacker sends a large number of SYN packets to the server. Each of these packets has to be handled like a connection request by the server, so the server must answer with a SYN-ACK and must allocate a memory space to this half-open connection. In other words, attacker tries to exhaust the memory space allotted to the TCP protocol. By this background, in modeling of this attack, we consider only one resource i.e. the memory space of the victim server and since it has limited capacity we consider it as a queuing system. Employing queuing theory, we give an abstraction for modeling SYN flooding attack. In this model, all connection requests share a same backlog queue. When a request arrives at the system, receives a buffer space of the backlog queue upon finding an inactive buffer space and is blocked otherwise. Assume that in this computer each half open connection is held for at most the period of time h seconds and at most m concurrent half-open connections are allowed. It is also assumed that a half-open connection for a regular request packet is held for a chance time which is exponentially distributed with parameter. The arrivals of the regular connection request packets and the attack connection request packets are both Poisson processes with rates k_1 and k_2 , respectively. The two arrival processes are independent of each other and of the holding times for half-open connections. Obviously, when the system is under SYN flooding attack, number of pending connections increases and in a point in which there is no more room for new connections to be saved, the arriving connection requests will be blocked. In the other word, when a server is under SYN flooding attacks, half-open connections quickly consume all the memory allocated for the pending connections and prevent the victim from further accepting new requests[10].

6.3 Comparison:

Particle Swarm Optimization (PSO) is a technique used to explore the search space of a given problem to find the settings or parameters required to maximize a particular objective. PSO considered time t and queue size m , that values of t and m play important roles in SYN flooding attack, t and m as its control parameters and employs the PSO algorithm to tune them

dynamically toward the best defense position. PSO increase or reduce time t and queue size m when it's needed.

However, PSO does not identify which are the particular attack requests. It adjusts the parameters of the server (t and m) according to its best value. This might be little inconvenient for servers which have fixed request capacity or time span for each request. This is where our approach comes in handy.

HRTE is a simple algorithm; it can identify the attack request and regular request based on a threshold which is dynamically selected. When the server is not under any SYN attack, it does the scheduling using Round Robin algorithm. When attacks are detected by using threshold value, it switches its approach which in turn helps to identify the attack request based on their burst time and ejects the requests with highest time in the queue. HRTE identify the attack based on threshold and clerically ejects the attack request. It doesn't change time and queue size.

We showed the efficiency of both the algorithms using same set of time for TCP requests and for a fixed number of requests and time, HRTE has exhibited improved result than PSO.

PSO LOW RT and HRTE LOW RT:

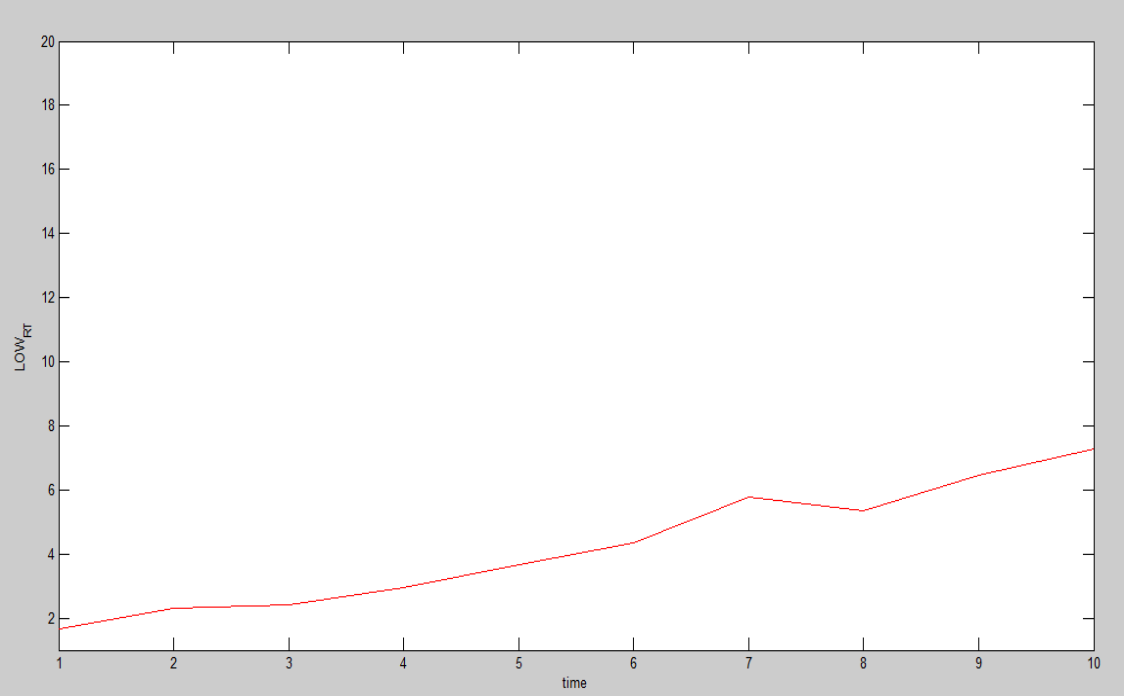


Fig 6.7 PSO LOW RT

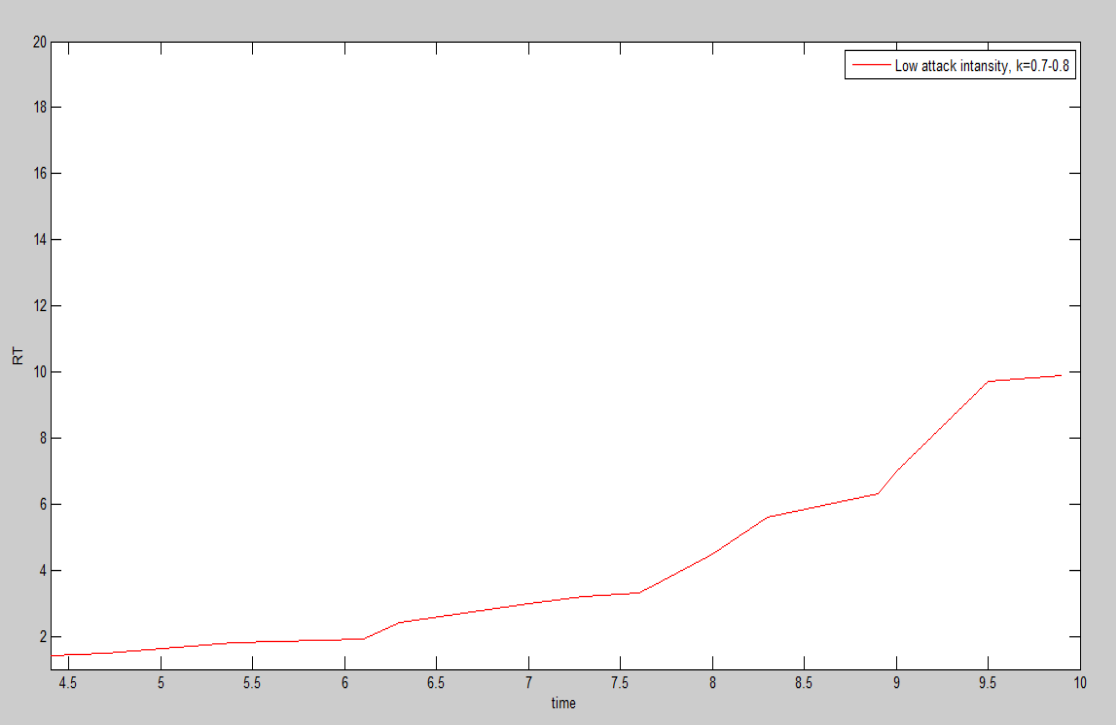


Fig 6.8 HRTE LOW RT

PSO LOW AT and HRTE LOW AT:

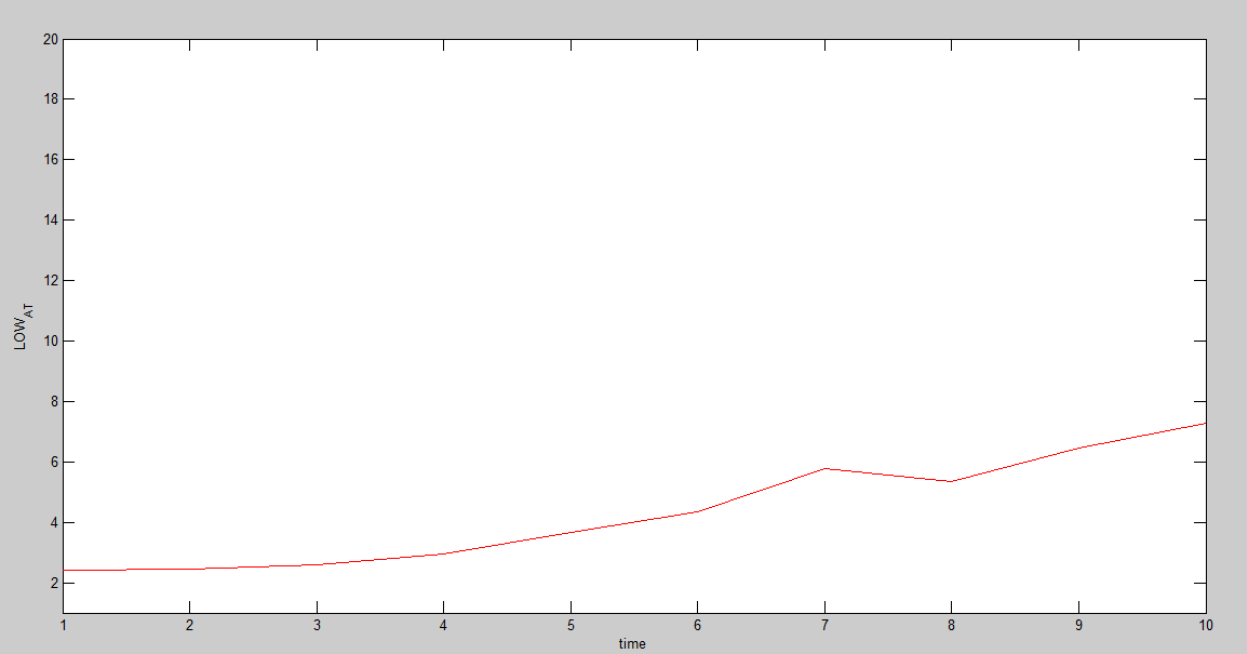


Fig 6.9 PSO LOW AT

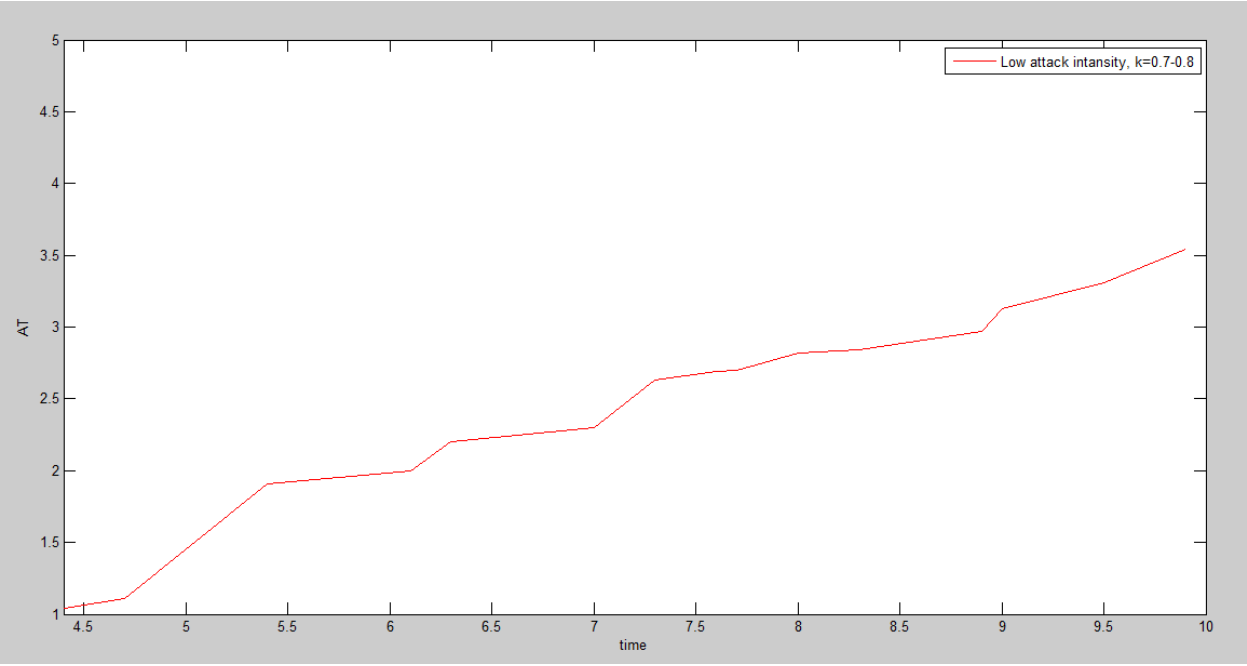


Fig 6.10 HRTE LOW AT

PSO MIDEUM RT and HRTE MIDEUM RT:

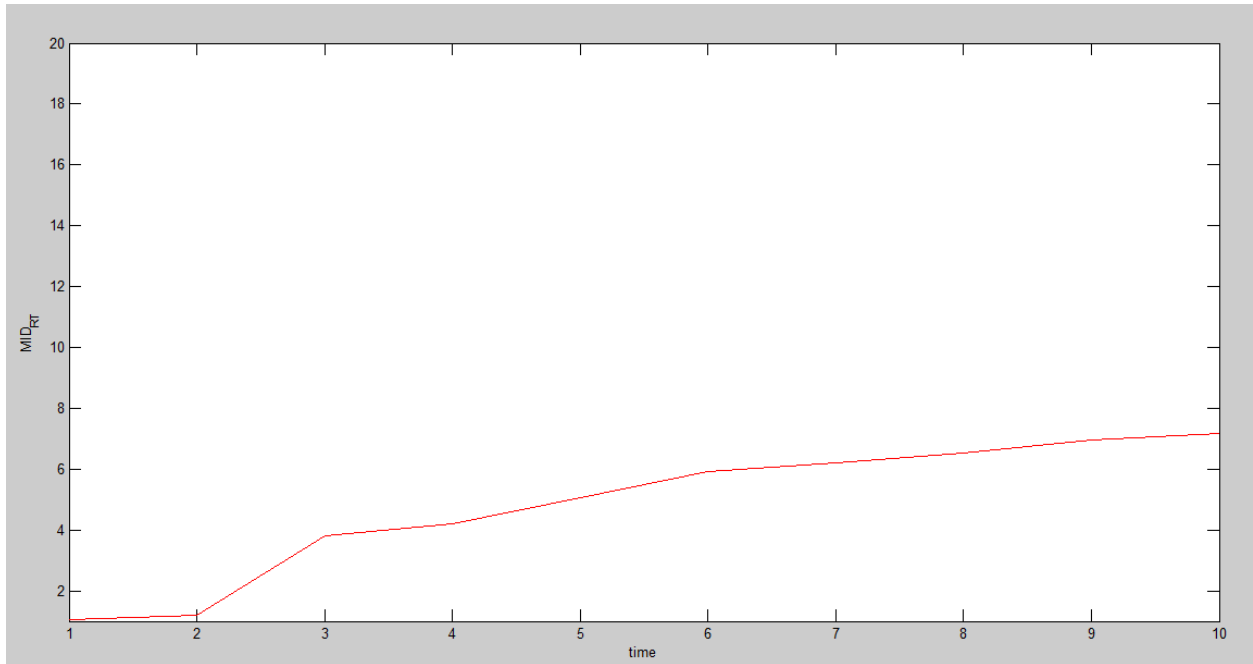


Fig 6.11 PSO MIDEUM RT

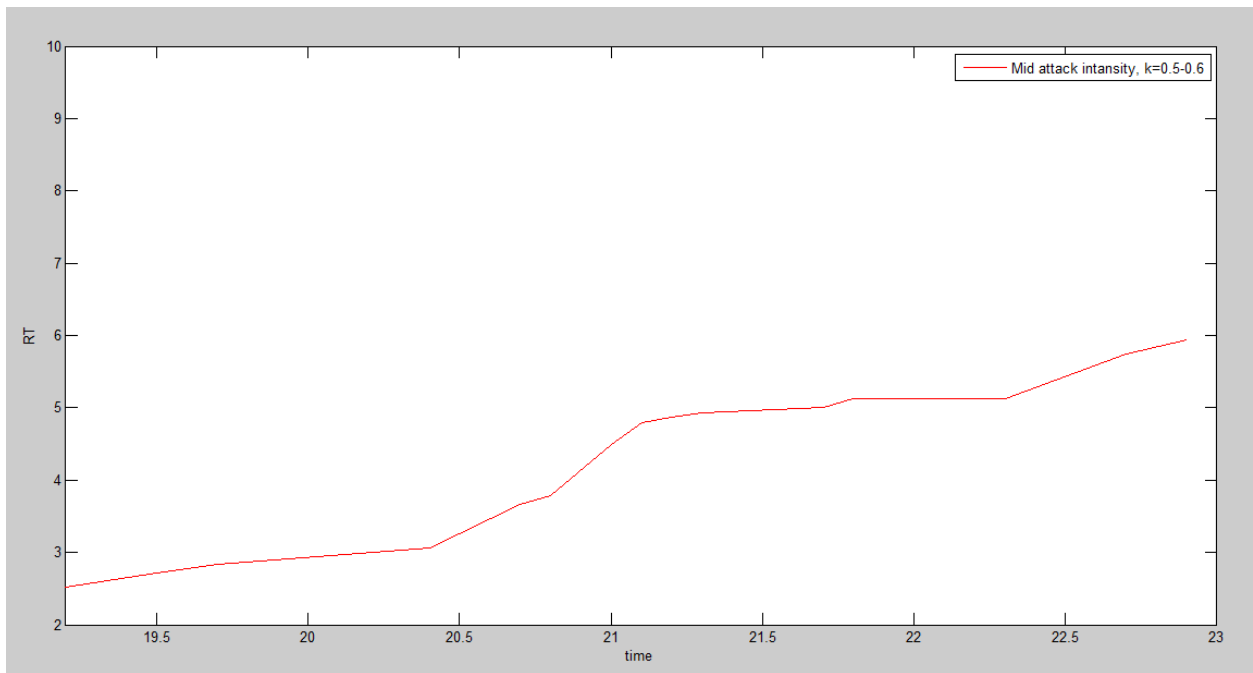


Fig 6.12 HRTE MIDEUM RT

PSO MIDEUM AT and HRTE MIDEUM AT

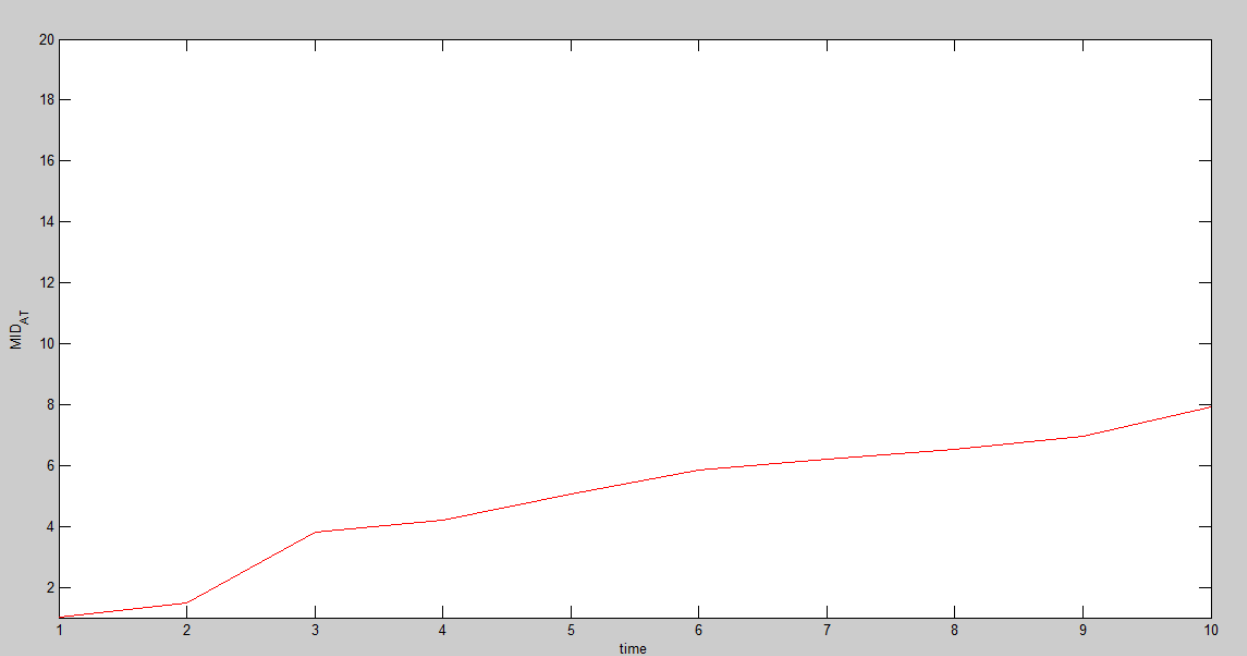


Fig 6.13 PSO MIDEUM AT

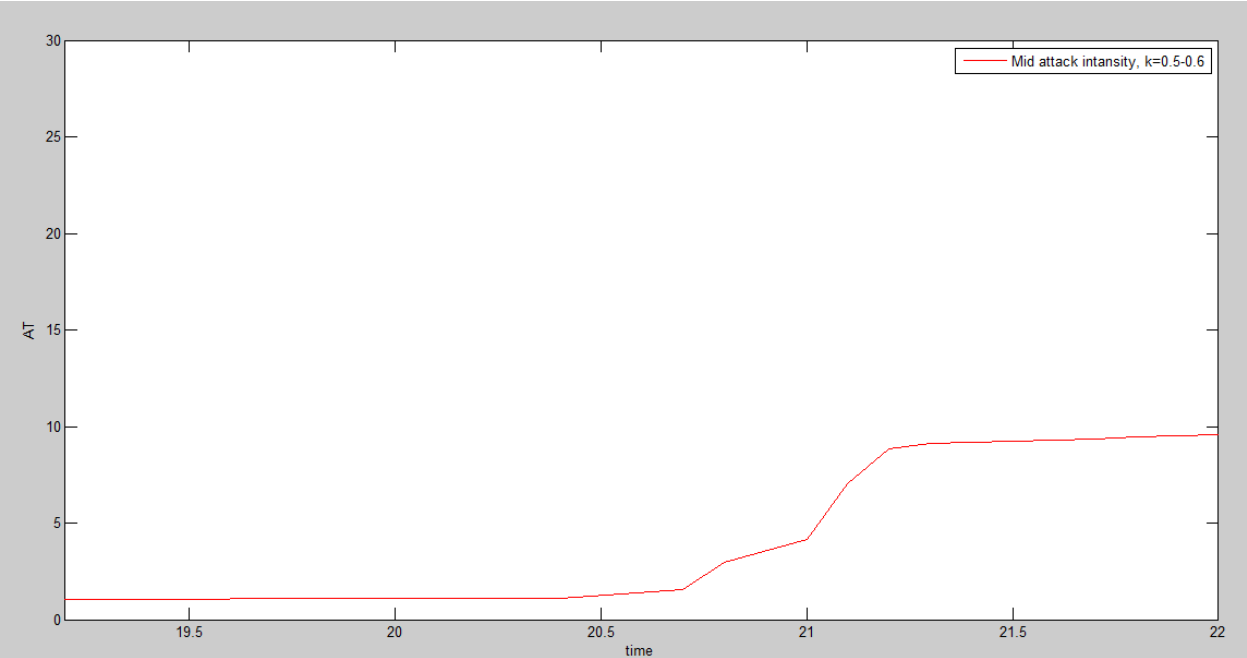


Fig 6.14 HRTE MID AT

PSO HIGH RT and HRTE HIGH RT

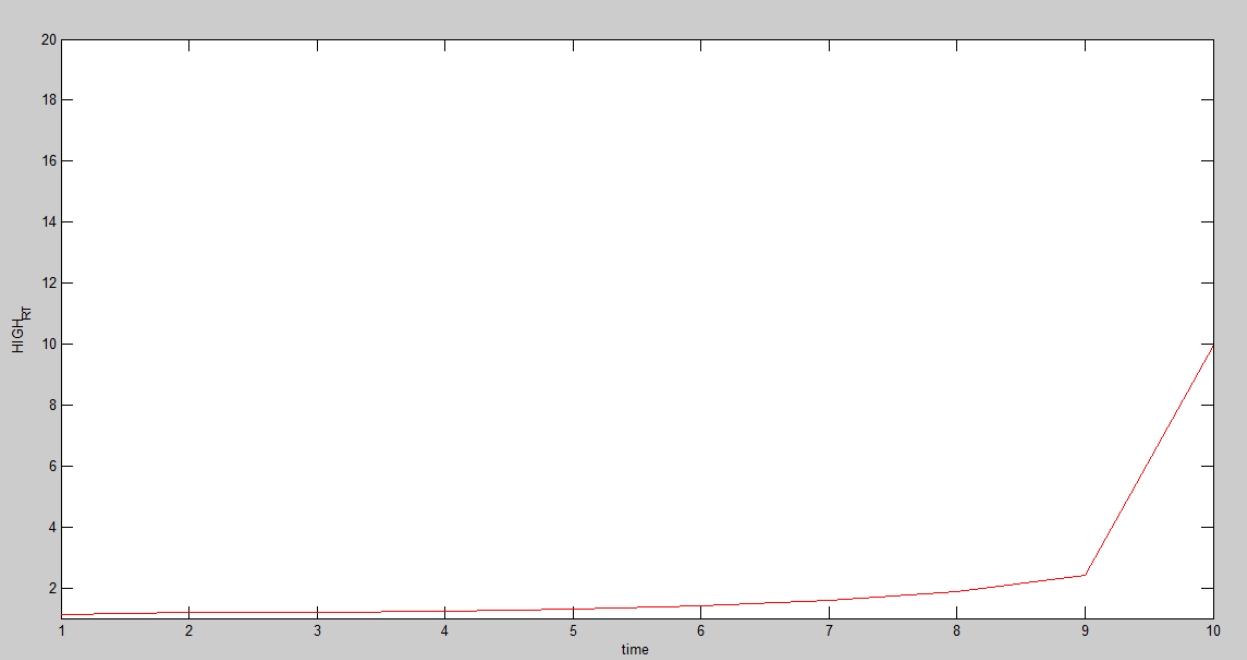


Fig 6.15 PSO HIGH RT

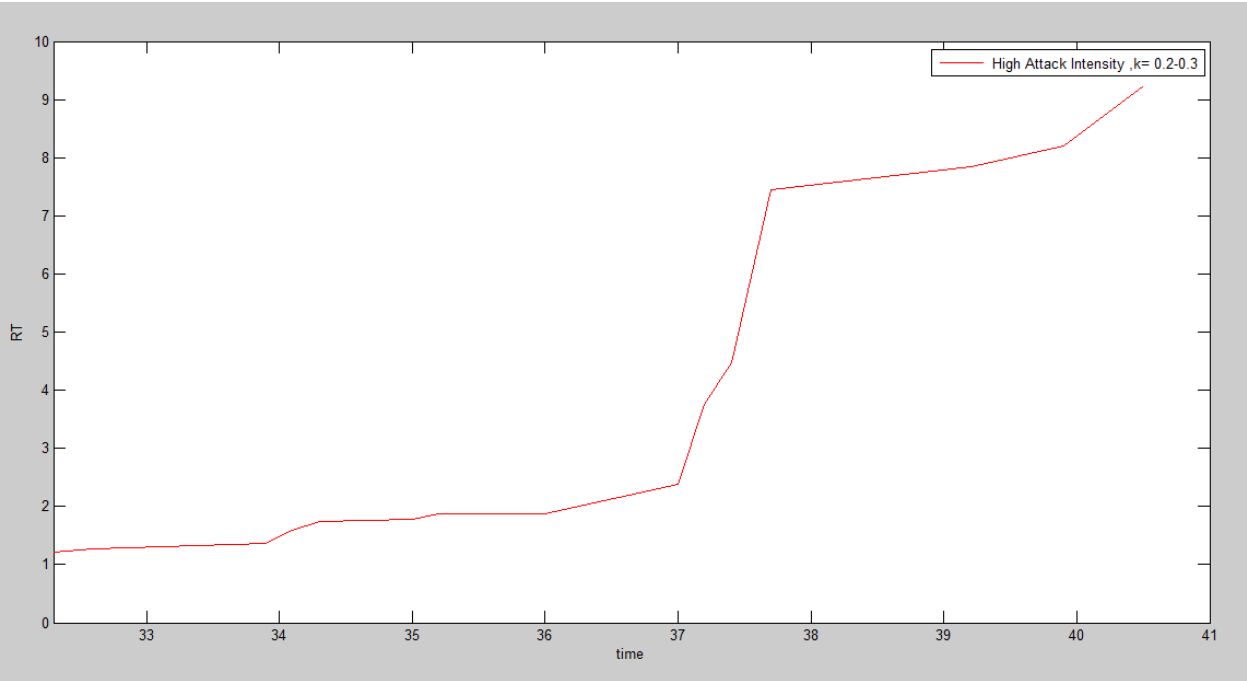


Fig 6.16 HRTE HIGH RT

PSO HIGH AT and HRTE HIGH AT

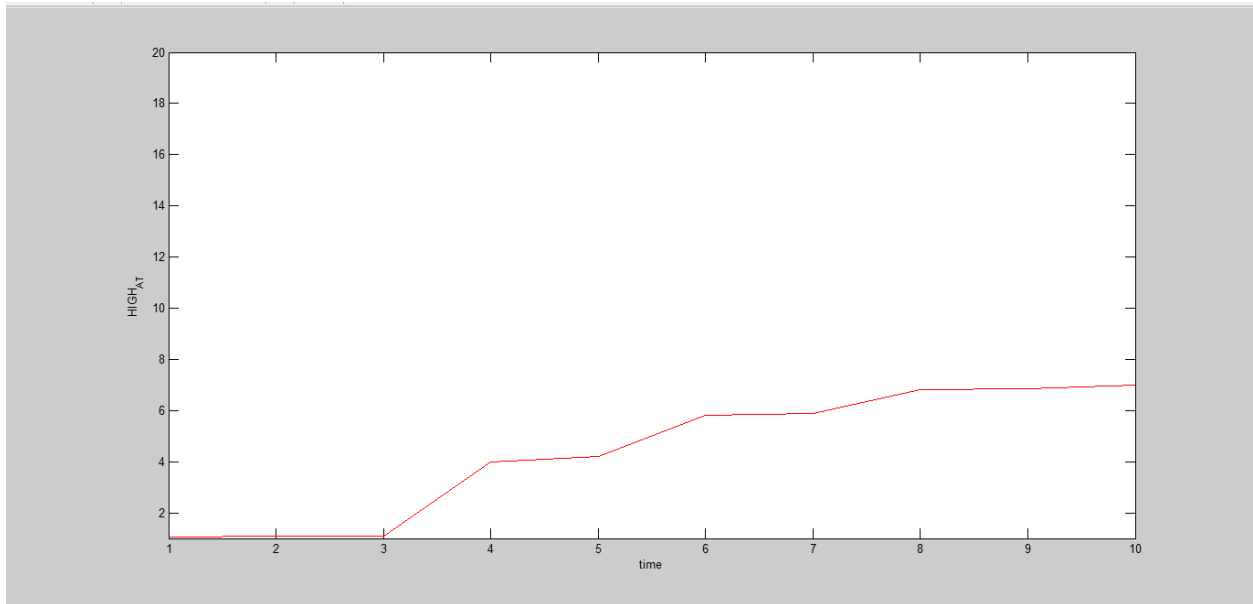


Fig 6.17 PSO HIGH AT

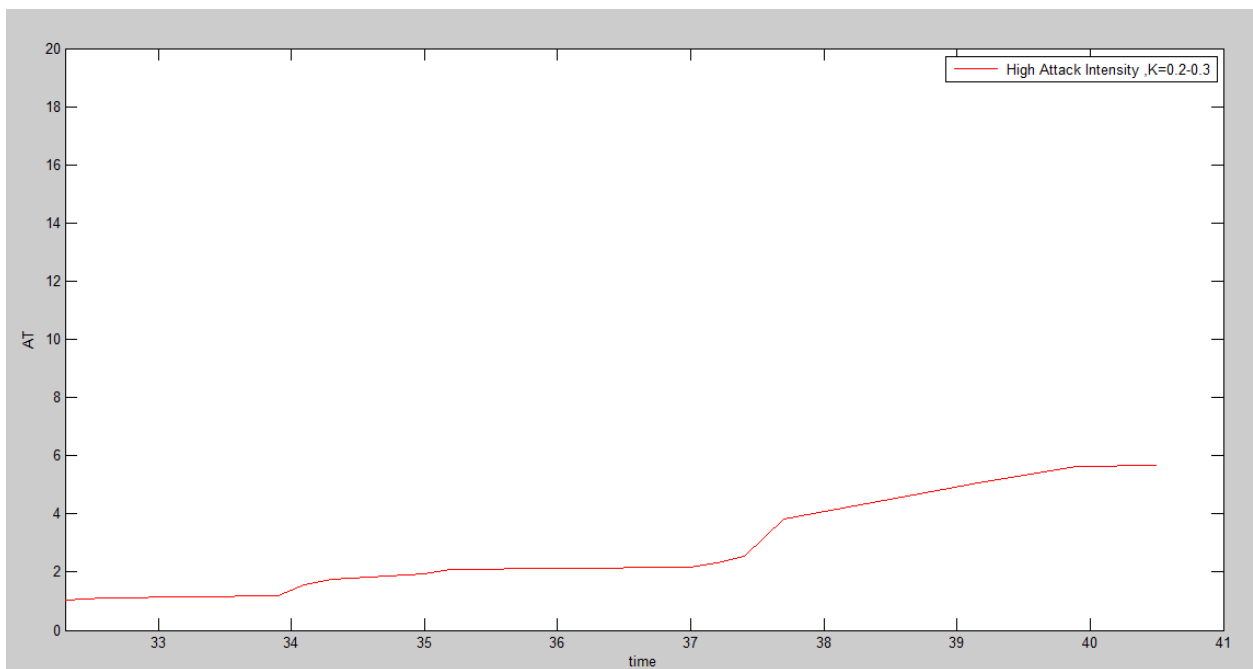


Fig 6.18 HRTE HIGH AT

From this figure we can understand, sometime HRTE performance better than PSO and sometime HRTE performance similar as HRTE. PSO algorithm provides better dynamic solution toward the best defense position. But PSO algorithm that's control parameters t and m can't work everywhere. For this reason, whether server can't reduce or increase time and queue size or whether server queue size is fixed their HRTE show better performance.

Chapter 7

Conclusion

7.1 Final Analysis and Decision:

Theoretical analysis shows that HRTE identify the attack based on threshold and ejects the attack request. HRTE which ejects the job with the highest residence time. On the other hand PSO finds out the best value and adjusts the time and queue size accordingly. Although both of them have shown similar performance in defending SYN attack, but since HRTE is already doing the scheduling for the requests, we can use it for SYN attack as well. We don't need an extra algorithm for defending the attacks. This is useful for the servers having limited capacity and time constraints to use HRTE than PSO.

We finally conclude that server whose queue or network capacity is not subjected too much drift. In these case HRTE show better performance than PSO.

7.2 Limitation and Future Plan:

We could not work with real time data in actual server. Therefore we cannot see the actual effect of our approach on a victim server. For our future plans, we would like to work with real time data in an actual server.

Chapter 8

APPENDIX

Code for HRTE Algorithm

```
tic
m=20;
n= 10;
btime=[3 4 5 8 6 5 7 9 4 2 ];
%burst time

q=2;
%quantum time

tatime=zeros(1,n);
%turn around time

wtime=zeros(1,n);
%waiting time

residence_time=zeros(1,n);
rtime=btime;
%intially remaining time= waiting time

b=0;

t=0;

flag=0;
%this is set if process has burst time left after
%quantum time is completed
if(n<m)

for i=1:1:n
%running the processes for 1 quantum

if(rtime(i)>=q)

fprintf('P%d\n',i);
```

```

for j=1:1:n

if(j==i)

    rtime(i)=rtime(i)-q;                %setting the
remaining time if it is the process scheduled

else if(rtime(j)>0)

    wtime(j)=wtime(j)+q;                %incrementing
wait time if it is not the process scheduled

end

end

end

else if(rtime(i)>0)

fprintf('P%d\n',i);

for j=1:1:n

if(j==i)

    rtime(i)=0;                        %as the remaining time is
less than quantum it will run the process and end it

else if(rtime(j)>0)

    wtime(j)=wtime(j)+rtime(i);        %incrementing wait
time if it is not the process

% scheduled
end

end

end

end

end

end

end

```



```

for i=1:1:n

    if(rtime(i)>0)
    %if remaining time is left set flag

        flag=1;

    end

end

while(flag==1)
%if flag is set run the above process again

    flag=0;

    for i=1:1:n

        if(rtime(i)>=q)

            fprintf('P%d\n',i);

            for j=1:1:n

                if(j==i)

                    rtime(i)=rtime(i)-q;

                else if(rtime(j)>0)

                    wtime(j)=wtime(j)+q;

                end

            end

        end

    end

    else if(rtime(i)>0)

        fprintf('P%d\n',i);

        for j=1:1:n

            if(j==i)

                rtime(i)=0;

```

```

else if(rtime(j)>0)
wtime(j)=wtime(j)+rtime(i);

end

end

end

end

end

end

for i=1:1:n
if(rtime(i)>0)
flag=1;

end

end

end

for i=1:1:n

tatime(i)=wtime(i)+btime(i);                                %calculating turn
around time for each process

% by adding waiting time and burst time
end

disp('Process Burst time Waiting time Turn Around time');    %displaying
the final values

for i=1:1:n

fprintf('P%d\t\t%d\t\t%d\t\t%d\n', (i+1),btime(i),wtime(i),tatime(i));

b=b+wtime(i);

t=t+tatime(i);

residence_time(i)=wtime(i)+btime(i);

```



```

p5= 47;
p6= 40;
p7= 41;
p8= 47;
p9= 49;
p10= 36;

AT =((p4+p5+p6+p7+p8+p9+p10)/(simulation_time *m));
fprintf('AT = %d',AT);

```

Plotting Code in MATLAB

```

time = [32.30
        32.60
        33.90
        34.10
        34.30
        35.00
        35.20
        36.00
        37.00
        37.20
        37.40
        37.70
        39.20
        39.90
        40.50];

RT =
[1.22,1.27,1.36,1.60,1.75,1.78,1.87,1.88,2.38,3.75,4.48,7.44,7.85,8.20,9.24];
% my independent vector
plot(time, RT, 'r')

xlabel ('time');
ylabel('RT');

axis([32.30 41 0 10]);
legend(' High Attack Intensity ,k= 0.2-0.3')

```

Code for PSO Algorithm

```
%% Initialization
% Parameters
clear
clc
iterations = 30;
inertia = 1.0;
correction_factor = 2.0;
swarm_size = 49;

% ----- initial swarm position -----
index = 1;
for i = 1 : 7
    for j = 1 : 7
        swarm(index, 1, 1) = i;
        swarm(index, 1, 2) = j;
        index = index + 1;
    end
end

swarm(:, 4, 1) = 0;           % best value so far
swarm(:, 2, :) = 0;         % initial velocity

%% Iterations
for iter = 1 : iterations

    %-- evaluating position & quality ---
    for i = 1 : swarm_size
        swarm(i, 1, 1) = swarm(i, 1, 1) + swarm(i, 2, 1)/1.3;    %update x
        position
        swarm(i, 1, 2) = swarm(i, 1, 2) + swarm(i, 2, 2)/1.3;    %update y
        position
        x = swarm(i, 1, 1);
        y = swarm(i, 1, 2);
        x=3;
        y=20;

        lambda1= 3;
        r1= poissrnd(lambda1);

        lambda2= 7;
        r2= poissrnd(lambda2);

        val = (r1*x)/(r2*x)*(2*y-(r1*x+r2*x));           % fitness evaluation
        (you may replace this objective function with any function having a global
        minima)

        if val < swarm(i, 4, 1)                         % if new position is better
            swarm(i, 3, 1) = swarm(i, 1, 1);           % update best x,
            swarm(i, 3, 2) = swarm(i, 1, 2);           % best y postions
            swarm(i, 4, 1) = val;                       % and best value
        end
    end
end
```

```

end

[temp, gbest] = min(swarm(:, 4, 1));           % global best position

%--- updating velocity vectors
for i = 1 : swarm_size
    swarm(i, 2, 1) = rand*inertia*swarm(i, 2, 1) +
correction_factor*rand*(swarm(i, 3, 1) - swarm(i, 1, 1)) +
correction_factor*rand*(swarm(gbest, 3, 1) - swarm(i, 1, 1)); %x velocity
component
    swarm(i, 2, 2) = rand*inertia*swarm(i, 2, 2) +
correction_factor*rand*(swarm(i, 3, 2) - swarm(i, 1, 2)) +
correction_factor*rand*(swarm(gbest, 3, 2) - swarm(i, 1, 2)); %y velocity
component

end

%% Plotting the swarm
clf
plot(swarm(:, 1, 1), swarm(:, 1, 2), 'x') % drawing swarm movements
axis([-2 30 -2 30]);
pause(.2)
end

```

BIBLIOGRAPHY

- [1]SeungJae Won, “TCP SYN Flood - Denial of Service”, web2.uwindsor.ca/courses/cs/aggarwal/cs60564/Assignment1/Won.pdf
- [2]Shahram et al. “Defense against SYN Flooding Attacks: A Scheduling Approach ”
- [3]J. Lemon et al. “DETECTION OF SYN FLOODING ATTACKSUSING LINEAR PREDICTION ANALYSIS”
- [4] J. Lemon, “Resisting SYN Flooding DoS Attacks with a SYN Cache”, *Proceedings of USENIX BSDCon '2002*, February, 2002.
- [5] C. L. Schuba, I. V. Krsul, M. G. Kuhn, E. H. Spafford, A. Sundaram and D. Zamboni, “Analysis of a Denial of Service Attack on TCP”, *Proceedings of IEEE Symposium on Security and Privacy*, May 1997.
- [6]Vasilios A. Siris and Fotini Papagalou et al. “Application of Anomaly Detection Algorithms for Detecting SYN Flooding Attacks”
- [7] S. M. Bellovin, “ICMP Traceback Messages”, *Internet Draft: draftbellovin-itrace-00.txt*, March 2000.
- [8] K. Park and H. Lee, “On the Effectiveness of Probabilistic Packet Marking for IP Traceback under Denial of Service Attack”, *Proceedings of IEEE INFOCOM 2001*, March 2001.
- [9] S. Savage, D. Wetherall, A. Karlin and T. Anderson, “Practical Network Support for IP Traceback”, *Proceedings of ACM SIGCOMM'2000*, August 2000
- [10]Shahram Jamali et al. “Defense against SYN flooding attacks: A particle swarm optimization approach”
- [11] A. C. Snoren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, S. T. Kent and W. T. Strayer, “Hash-Based IP Traceback”, *Proceedings of ACM SIGCOMM'2001*, August 2001.
- [12] S. F. Wu, L. Zhang, D. Massey, and A. Mankin, “Intention-driven ICMP traceback”, *Internet Draft: draft-wu-itrace-intention-00.txt*, February 2001.