QUEUEING ANALYSIS FOR NETWORKS UNDER DoS ATTACK

A Project Report Submitted

in Partial Fulfilment of the Requirements

for the Degree of

BACHELOR OF TECHNOLOGY

in

Mathematics and Computing

by

Nikhil Agarwal

(Roll No. 11012323)

to the

DEPARTMENT OF MATHEMATICS

INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

GUWAHATI - 781039, INDIA

April 18, 2015

CERTIFICATE

This is to certify that the work contained in this project report entitled "**QUEUEING ANALYSIS FOR NETWORKS UNDER DoS ATTACK**" submitted by **Nikhil Agarwal** (**Roll No.: 11012323**) to Indian Institute of Technology Guwahati towards partial requirement of **Bachelor of Technology** in Mathematics and Computing has been carried out by him/her under my supervision and that it has not been submitted elsewhere for the award of any degree..

Guwahati - 781 039	(Dr. N. Selvaraju)
April 18, 2015	Project Supervisor

ABSTRACT

With the rapid development of information technology, internet has affected the people in all aspects such as public utilities, telecommunication, financial transaction and defense system, all depends on information technology and their security. By using latest technology and internet, attackers may perform malicious activities. Botnet is the most serious emerging threat. Botnet performs various kinds of malicious activities and one of them is Denial of Service. DoS degrades the performance of a network, disconnects the host and performs bandwidth depletion and resource depletion attack. The main aim of the project is to have queueing analysis of networks under such attacks i.e to understand how each parameter is affected under DoS attacks or to detect an attack in progress as soon as possible because the servers definitely have some mechanisms to store and process the requests.

Contents

Chapter 1

Introduction

1.1 Security

Security is an important topic now a days. The protection from harm, or degree of resistance is called security. Every vulnerable and valuable assets like person, computer, organization needs security of one or other kind. Out of these, computer security is the most important issue these days.

1.1.1 Computer Security

Computer Security is the combination of processes and mechanisms by which digital equipment, information and services are protected from unintended or unauthorized access, change or destruction. It is of growing importance in line with the increasing reliance on computer systems of most societies worldwide. Bank transactions are quite common these days, everything is purchased using credit cards, so building a bullet proof secure network is very necessary these days.

1.1.2 Why do we need Security?

Security is becoming more and more important issue these days due to inexpensive Internet connections, changing world of global data communications and fast-paced software development. Since the global computing is largely insecure, security has become a basic requirement of day by day. As our data moves from point X to point Y on the Internet, it passes through large number of other points along the way, which gives other users opportunity to intercept, interfere and even alter it. Even, other users may maliciously transform our data into something which we didn't intend. The intruders may obtain unauthorized access to our system and can impersonate us, steal information from us and can even deny access to our own resources.

1.1.3 Vulnerabilities

Every attack can be classified into either of these categories:

- **Backdoors:** Using Backdoor, an attacker bypasses normal authentication and gets remote access to a computer.
- **Denial of Service:** Denial of Service(DoS) is used to make system resources unusable by mostly taking the victim system down.
- **Direct Access Attacks:** An unauthorized user after gaining access to a victim computer may perform various malicious activities like installing different types of devices to compromise security, modifying operating system, key loggers, software worms, etc. Hard drives can also be read using bootable media.
- **Eavesdropping:** Eavesdropping is the act of listening to private conversations between hosts on a network.
- **Indirect Attacks:** Indirect attaks are generally caused by using someone else's computer.
- **Exploits:** These are the small programs designed to take advantage of a software flaw that has been discovered, either remote or local. The code from the exploit program is frequently reused in trojan horses and computer viruses.

We will focus our study on Denial of Service.

1.2 Denial of Service

Various kind of attacks like Backdoors, Direct Access attacks, Eavesdropping, etc can be made on an internet. Denial of Service(DoS) is one such important attack. DoS attacks are not used to gain unauthorized access or control of a system. They are instead designed to render it unusable. Attackers can deny service to individual victims, such as by deliberately entering a wrong password enough consecutive times to cause the victim account to be locked, or they may overload the capabilities of a machine or network and block all users at once. These types of attack are, in practice, very hard to prevent, because the behavior of whole networks needs to be analyzed, not only the behavior of small pieces of code. DoS attacks basically are used to:

- Slow network performance
- Make some website unavailable
- Restricting users to access any web site
- Increase the number of spam emails received
- Disconnect a wireless or wired internet connection
- Long term denial of web services.

1.2.1 Types of DoS Attack:

- **UDP Flooding:** UDP flooding is mainly host based DoS attack. The attack is generally initialized by sending large number of UDP packets to the random port on the victim system. By delivering large number of packets to the victim system, the attacker takes down the victim system.
- **TCP SYN Flooding:** For establishing a TCP connection we use 3 way handshake mechanism. This attack also uses this method to initialize a new TCP

connection. For this attack attackers rapidly send SYN segments with or without spoofing their IP address causing the system to go down.

- **Ping of Death:** In this type of attack, the attacker generates a packet more than 65,536 bytes to take down the victim system. He uses a ping system utility to create an IP packet that exceeds the maximum 65,536 bytes of data allowed by the IP specification. This oversized packet is then sent to an unsuspecting victim system. Systems may crash, hang, or reboot when they receive such a maliciously crafted packet.

- **Smurf Attack:** Smurf attack is a kind of an amplification attack. In this attack, the attacker generates large amount of traffic on a victim computer to take it down. Attacker sends a packet as a direct broadcast to a subnet of a network. All the system in the subnet sends the reply of this broadcast. Attacker uses spoofed IP address. By spoofing the source IP address of the packet, all the responses will get sent to the spoofed IP address.

Why do we need Queueing Theory? We will mainly concentrate on studying the system parameters when large number of malicious packets are sent to the victim system to take the victim down so that it is left unusable. DoS is related to security which in turn is related to cryptographic issues but Queueing theory also plays a vital role. Every network serves request for clients. The memory allocation of the networks are also fixed i.e huge number of requests can't be served together. So networks must have some mechanism to store and serve the request arriving at the system i.e some queuing mechanism is the necessity of every network. Since server is everything in a network, it is certainly the key point for the attacker to launch the attack and somehow disable the server to provide the service to its legitimate users. Thus, queueing theory conceptually holds the power to detect some kind of Denial of Service attacks.

What is Queueing theory? Queueing theory is the mathematical study of waiting lines, or queues. In queueing theory a model is constructed so that queue lengths and waiting times can be predicted. Queueing theory examines each and every component of waiting in line to be served, including the number of customers, arrival process, service process, number of servers, number of systems (which might be people, data packets, cars, etc.). Queues are used everywhere in real life. Queuing theory mainly deals with providing resources efficiently to avoid large queues. Real-life applications of queuing theory include providing faster customer service, improving traffic flow, shipping orders efficiently from a warehouse or serving requests quickly on a website. In our context, queues are only used as a mechanism to store the packets in the server and based on that we want to predict, the probability of packet being dropped or the service being denied to some clients.

Motivation: Thus, we have theoretically covered that Queueing Theory is necessary for studying DoS. In the next chapter, we take the intrusion elimination model described in [3] written by P. Kammas, T. Komninos, Y.C. Stamatiou1 and try to study the basic framework of that model and the usefulness of it. In the last chapter we predict some results using the model described in chapter 2 and try to study the the probability of the packet being dropped and other network parameters. Then we present the future scope of the project in full detail.

Chapter 2

The intrusion propagation and elimination model:

2.1 Introduction:

As described in [1], written by P. Kammas, T. Komninos, Y.C. Stamatiou1, we study mathematical model for the co-evolution of the populations of virus and antivirus software modules based on a queue theoretical formulation. Our computer network is modeled as interconnected service centers (network of queues) with incoming/outgoing connections to the outside world. Each such service center is modeled as an M/M/1 queue servicing the agents that move within the network. These agents are the virus and antivirus agents that move in the network as network customers. The idea behind this model is that antivirus agents are simply users of all network resources and try to exploit all network servers in order to propagate further. Thus, this model provides a link between a networks characteristics (e.g. queue policies, service times and server utilization) with the speed with which virus agents propagate. A further element of this model is the virus-antivirus agent interaction. The rule we adopt is simple: if an antivirus agent meets a virus agents then it kills the virus agent and then kills itself (i.e. the two agents annihilate) so as to reduce the network load progressively as more and more virus agents are eliminated. We are interested for the co-evolution of the populations of virus and antivirus software agents across the infected network when they follow this mode of interaction. We show that the distribution of the numbers of virus/antivirus in the network nodes can be written in a product form much like the solution of the open Jackson networks of M/M/1 queues

2.2 Basic Model Design

Our Computer Network is modeled as an open Jackson network(i.e having product form solution) with incoming/outgoing connections to the outside world. We have N nodes in our system and we model every node as an M/M/1 queue with infinite size, such that no packets rejection takes place. The service times of the queue follow the exponential distribution and the arrivals follow the Poisson distribution. It is assumed that the service time of the packet in each queue is independent of service time of packet at other queues. It is assumed that the packet transmission time to a network queue is the same for all queues and approximately equal to the inverse of the transmission speed of the link leading to the queue. It is also assumed that the service time of the packet in each queue is independent of service time of packet at other queues. Whenever a packet is serviced at queue, it chooses the next node to visit with probability or exits the network with a certain probability which is almost similar to that of Markov routing techniques. The choice for the next node is predetermined hence the model allows deterministic routing. At any time any packet can enter from outside or from some other queue and can be transmitted to some other queue or out of the network. We use antivirus and virus agents in this model. They are just two different kind of customers where the malicious one can affect the network utilization of the other one. This model finds the probability of the network to be in certain state given the number of virus and antivirus agents.

The model parameters are:

- N: It denotes the number of queues in the network or the number of network nodes.

- λ_i : This is the parameter of the Poisson distribution used to model the arrival of agents in the network. The model doesn't differentiate between virus and antivirus when they arrive at the network and, thus, are considered to form Poisson distribution with single parameter.

- μ_i : This is the parameter of the exponential distribution assumed for the service time of the i^{th} queue i.e. service time for the queues follow exponential distribution. Since we have assumed service time is same for all the queues, so we denote it with this single parameter.

- ρ_i : Studying the utilization of a queue is very necessary for studying the queuing theory which is equal to λ_i/μ_i

- (a_i, v_i, d_i) : a_i denotes number of antivirus and v_i denotes number of virus agents where as d_i denotes number of virus antivirus interactions at the i^{th} queue, at equilibrium state.

- $Pr[a]$: It denotes the probability that the job queued carries an antivirus agent.

- $Pr[v]$: It denotes the probability that the job queued carries a virus agent.

- l_{ij} : It denotes the rate at which jobs leave queue i and enters queue j. If $i = 0$ it denotes that the job is coming from some outside source not from some other queue of the network. Similarly if $j = 0$ it denotes that the job is going outside of the entire network not to some other queue in the network.

- q_{ij} : It denotes the probability that a job leaves queue i entering queue j. If $i = 0$ it denotes that the job is coming from some outside source not from some other queue of the network. Similarly if $j = 0$ it denotes that the job is going outside of the entire network not to some other queue in the network.

- $\mathbf{n}(\mathbf{t}) = ((a_1(t), v_1(t), d_1(t)), ..., (a_N(t), v_N(t), d_N(t)))$

  $\mathbf{n}(\mathbf{t})$ is the state vector and it consist of three tuples. Since we have N nodes in our network, each node has some v_i number of viruses, a_i number of antiviruses and d_i number of interactions between viruses and antiviruses at time t. More precisely we can say that total number of viruses and antiviruses in state i should be $v_i + a_i + 2d_i$.

- The probability density function for the possible state vectors is defined as follows: $P_{((a_1,v_1,d_1),.....,(a_N,v_N,d_N):t)} = Pr[\mathbf{n(t)} = ((a_1, v_1, d_1), ..., (a_N, v_N, d_N))]$
- The steady state distribution for the state vector $\mathbf{n(t)}$ is defined as follows:

$$P_{((a_1,v_1,d_1),.....,(a_N,v_N,d_N))} = \lim_{t\to\infty} P((a_1, v_1, d_1), ..., (a_N, v_N, d_N) : t)$$

Our basic model looks somewhat like this:

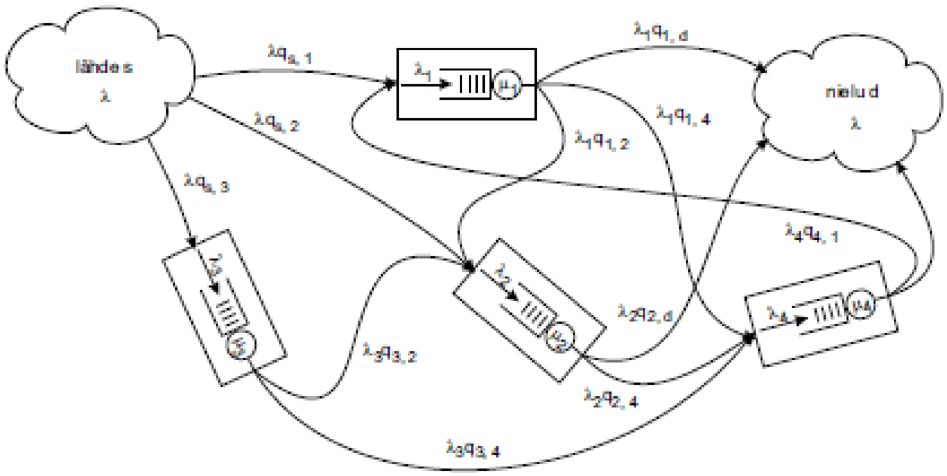


Figure 2.1: Basic Structure of our model (click on image for source)

2.3 Steady State Distribution

Theorem 2.3.1. *Given a computer network modeled as above i.e containing N nodes and each of the model parameters in the above described sense, the probability distribution function for the state vector in the steady state is given by:*

$$P_{((a_1,v_1,d_1),.....,(a_N,v_N,d_N))} = \prod_{i=1}^{N}(1-\rho_i){\rho_i}^{a_i+v_i+2d_i}$$

Proof. We start the proof by enumerating the possible events that can occur during an infinitesimal small time interval dt and will deal with each cases accordingly:

- **If a job arrives in the network in some of it′s queues is given by:**

  For antivirus:

  $$\sum_{j=1}^{N} P_{((a_1,v_1,d_1),...,(a_j-1,v_j,d_j),...,(a_N,v_N,d_N):t)} l_{0j} Pr[a]dt$$

  For virus:

  $$\sum_{j=1}^{N} P_{((a_1,v_1,d_1),...,(a_j,v_j-1,d_j),...,(a_N,v_N,d_N):t)} l_{0j} Pr[v]dt$$

  Since l_{0j} denotes the rate at which a job enters state j directly, we multiply it with corresponding probability of being a virus or an antivirus. Thus, as the antivirus enters the state j, $\mathbf{n(t)}$ at state j changes from (a_j-1, v_j, d_j) to (a_j, v_j, d_j). Same thing happens for the virus also.

- **If a job leaves a queue and exits the network is given by:**

  For antivirus:

  $$\sum_{i=1}^{N} P_{((a_1,v_1,d_1),...,(a_i+1,v_i,d_i),...,(a_N,v_N,d_N):t)} \mu_i q_{i0} Pr[a]dt$$

  For virus:

  $$\sum_{i=1}^{N} P_{((a_1,v_1,d_1),...,(a_i,v_i+1,d_i),...,(a_N,v_N,d_N):t)} \mu_i q_{i0} Pr[v]dt$$

  q_{i0} is the probability that after leaving state i, the job get out of the network and μ_i is the parameter for service time of i_{th} queue. Thus multiplying this two along with the probability of virus or antivirus we get the probability of a job which leaves the queue and exits the network.

- **If a job leaves one queue and enters another queue within the network is given by:**

  For antivirus:

$$\sum_{i=1}^{N}\sum_{j=1}^{N} P_{((a_1,v_1,d_1),...,(a_i+1,v_i,d_i),....,(a_j-1,v_j,d_j),...,(a_N,v_N,d_N):t)}\mu_i q_{ij} Pr[a]dt$$

  For virus:

$$\sum_{i=1}^{N}\sum_{j=1}^{N} P_{((a_1,v_1,d_1),...,(a_i,v_i+1,d_i),....,(a_j,v_j-1,d_j),...,(a_N,v_N,d_N):t)}\mu_i q_{ij} Pr[v]dt$$

  Multiplying μ_i with q_{ij} we get the probability for a job that moves from state i to state j along with the corresponding probability of being a virus or an antivirus.

- **If a pair of antivirus-virus agents annihilate together is given by**

$$\sum_{i=1}^{N} P_{((a_1,v_1,d_1),...,(a_i+1,v_i+1,d_i-1),...,(a_N,v_N,d_N):t)}{\mu_i}^2 Pr[a]Pr[v]dt$$

  When one virus and antivirus combine with each other, d_i increases by 1 and a_i and v_i decreases by 1. Thus, when we multiply probability of virus with that of antivirus along with their service time we get the probability of annihilation.

- **None of the above occurs is given by:**

$$P_{((a_1,v_1,d_1),...,(a_i,v_i,d_i),...,(a_N,v_N,d_N):t)}(1 - dt\sum_{j=1}^{N}(l_{0j} + \mu_j + {\mu_j}^2 Pr[a]Pr[v]))$$

  Thus this probability is given by adding no job enters in a queue along with no annihilation and no job leaving the queue. Thus the probability of none of the above case occurring is given by subtracting it with 1.

Now, we know that

$$P_{((a_1,v_1,d_1),.....,(a_N,v_N,d_N):t+dt)}$$

is given by adding the probability of all the above cases. In the last term when we open the bracket and take

$$P_{((a_1,v_1,d_1),.....,(a_N,v_N,d_N):t)}$$

to the left side and divide both sides by dt, and taking the limit as $dt \to 0$, then we obtain the left side as

$$\frac{d}{dt}P_{((a_1,v_1,d_1),.....,(a_N,v_N,d_N):t)}$$

which becomes 0 because no variation of population occurs in steady state while on the right hand side we cancel out dt from each term. After making sum of the terms on the right hand side equal to 0, and substituting Theorem 2.2.1, case 1 for antivirus becomes

$$P_{((a_1,v_1,d_1),..(a_j-1,v_j,d_j)...,(a_N,v_N,d_N):t)} = \frac{(1-\rho_j){\rho_j}^{a_j-1+v_j+2d_j}}{(1-\rho_j){\rho_j}^{a_j+v_j+2d_j}}$$

$$P_{((a_1,v_1,d_1),..(a_j-1,v_j,d_j)...,(a_N,v_N,d_N):t)} = \frac{1}{\rho_j}$$

Now substituting similarly for all other case we finally get

$$\sum_{j=1}^{N}(l_{0j} + \mu_j + {\mu_j}^2 Pr[a]Pr[v]) = \sum_{j=1}^{N}\frac{l_{0j}Pr[a]}{\rho_j} + \sum_{j=1}^{N}\frac{l_{0j}Pr[v]}{\rho_j} + \sum_{i=1}^{N}\mu_i q_{i0}\rho_i Pr[a] +$$

$$\sum_{i=1}^{N}\mu_i q_{i0}\rho_i Pr[v] + \sum_{i=1}^{N}\sum_{j=1}^{N}\frac{\rho_i}{\rho_j}\mu_i q_{ij} Pr[a] + \sum_{i=1}^{N}\sum_{j=1}^{N}\frac{\rho_i}{\rho_j}\mu_i q_{ij} Pr[v] + {\mu_i}^2 Pr[a]Pr[v]$$

Since we want to look at only those queues which has either virus or an antivirus. Thus $Pr[a] + Pr[v] = 1$.. Substituting this in the above equation, we are left with only four terms on the right hand side and now we analyze each of them.

- First term can be rewritten by substituting ρ_j as:

$$\sum_{j=1}^{N} \frac{l_{0j}}{\rho_j} = \sum_{j=1}^{N} \frac{l_{0j}\mu_j}{\lambda_j}$$

- Second therm can be reduced to:

$$\sum_{i=1}^{N} \mu_i q_{i0} \rho_i = \sum_{i=1}^{N} \lambda_i q_{i0} = \sum_{i=1}^{N} \lambda_i (1 - \sum_{j=1}^{N} q_{ij}) = \sum_{i=1}^{N} \lambda_i - \sum_{i=1}^{N}\sum_{j=1}^{N} \lambda_i q_{ij} \tag{2.1}$$

The first one follows from simply substituting $\mu_i \rho_i = \lambda_i$ while the second one is the probability that the job goes out of the system multiplied by λ_i. We find this probability by subtracting from 1 that the job goes to some other queue. The third one is just the multiplication of the terms.

Summation of $\lambda_i q_{ij}$ over all i and j denotes that that the job arrives at the rate λ_i and with probability q_{ij} makes it's transition form queue i to queue j. So, it can also be written as the negation of it's going out of the system or annihilating. Thus we can write:

$$\sum_{j=1}^{N} \lambda_i q_{ij} = \sum_{i=1}^{N} \lambda_i - \sum_{i=1}^{N} l_{i0} - \sum_{i=1}^{N} {\mu_i}^2 Pr[a]Pr[v] \tag{2.2}$$

A process arriving at rate λ_i either goes to some other queue or it goes out of the system or it annihilates itself. So we can write:

$$\lambda_i = \sum_{j=0}^{N} l_{ij} + {\mu_i}^2 Pr[a]Pr[v]$$

Now calculating and substituting we get

$$\sum_{i=1}^{N} \lambda_i = \sum_{i=1}^{N}\sum_{j=0}^{N} l_{ij} + \sum_{i=1}^{N} {\mu_i}^2 Pr[a]Pr[v]$$

$$\Rightarrow \sum_{i=0}^{N}\sum_{j=1}^{N} l_{ij} = \sum_{i=1}^{N}\sum_{j=0}^{N} l_{ij} + \sum_{i=1}^{N} {\mu_i}^2 Pr[a]Pr[v]$$

$$\Rightarrow \sum_{i=1}^{N}\sum_{j=1}^{N} l_{ij} + \sum_{j=1}^{N} l_{0j} = \sum_{i=1}^{N}\sum_{j=1}^{N} l_{ij} + \sum_{i=1}^{N} l_{i0} + \sum_{i=1}^{N} {\mu_i}^2 Pr[a]Pr[v]$$

$$\Rightarrow \sum_{j=1}^{N} l_{0j} = \sum_{i=1}^{N} l_{i0} + \sum_{i=1}^{N} {\mu_i}^2 Pr[a]Pr[v]$$

Now substituting in eqn 2.2 we get,

$$\sum_{i=1}^{N}\sum_{j=1}^{N} \lambda_i q_{ij} = \sum_{i=1}^{N} \lambda_i - \sum_{j=1}^{N} l_{0j} \tag{2.3}$$

Now substituting in eqn 2.1 we get,

$$\sum_{i=1}^{N} \mu_i q_{i0} \rho_i = \sum_{i=1}^{N} \lambda_i - \sum_{i=1}^{N}\sum_{j=1}^{N} \lambda_i q_{ij} = \sum_{j=1}^{N} l_{0j} \tag{2.4}$$

- Third term

$$\sum_{i=1}^{N}\sum_{j=1}^{N} \frac{\rho_i}{\rho_j} \mu_i q_{ij} = \sum_{i=1}^{N}\sum_{j=1}^{N} \lambda_i q_{ij} \frac{\mu_j}{\lambda_j}$$

Now substituting 2.3 we get

$$\sum_{j=1}^{N} \frac{\mu_j}{\lambda_j}(\lambda_j - l_{0j})$$

$$\Rightarrow \sum_{j=1}^{N} \mu_j - \sum_{j=1}^{N} \frac{\mu_j l_{0j}}{\lambda_j}$$

Now when we add all of these four terms, first term cancels out with the last term of third term and we are left with

$$\sum_{j=1}^{N} l_{0j} + \sum_{j=1}^{N} \mu_j + \sum_{j=1}^{N} {\mu_j}^2 Pr[a]Pr[v]$$

This is equal to left hand side, thus our theorem is correct as it satisfies the steady state equation.

$\square$

2.4 Properties and Assessment of our basic model:

In the above above described model, we have assumed that antivirus and virus agents enter the network as normal tasks as they need networks resources to serve their purpose. Both antivirus and virus agents destroy one another. Although more drastic elimination techniques can be used for the model, but this simple elimination rule provide some advantages like it can be mathematically solvable, we can detect how serious a virus can be in an network with similar server characteristics. The model doesn't deal with any complex differential equations for tracking the probability of antivirus and virus agents in a queue. The resulting probability distribution function for the described network state is quite simple and it can be easily computed and analysed by varying some parameters. In addition, each parameter plays some kind of role in the derivation and the dependency of the network and the queues on a particular parameter can be easily studied. In this case, we have assumed that one virus and one antivirus interacts, if one antivirus interacts with s viruses then our d_i for queue i will be become $(1 + s)d_i$ and corresponding probability distribution function will be given by

$$P_{((a_1,v_1,d_1),.....,(a_N,v_N,d_N))} = \prod_{i=1}^{N} (1 - \rho_i){\rho_i}^{a_i+v_i+(1+s)d_i}$$

Thus our described model is pretty simple to study and we will extend this model to study DoS attacks.

Chapter 3

Denial of Service

3.1 Basic Model Design

In the previous section, we discussed a basic model where the system had N queues, arrival rate was λ_i, service rate was μ_i, utilization was ρ_i and we described the probability of the system containing a_i number of antivirus, v_i number of virus and d_i number of antivirus and virus interactions. The probability was given by :

$$\mathbf{P}_{((\mathbf{a_1},\mathbf{v_1},\mathbf{d_1}),.....,(\mathbf{a_N},\mathbf{v_N},\mathbf{d_N}))} = \prod_{\mathbf{i=1}}^{\mathbf{N}} (\mathbf{1} - \rho_{\mathbf{i}})\rho_{\mathbf{i}}{}^{\mathbf{a_i+v_i+2d_i}}$$

Now extending this analogy, suppose a_i denotes the number of normal request packets entering the system, v_i denotes the number of malicious packets entering the system. If there is no interaction between these packets, then d_i is 0 and the expected result will be:

$$\mathbf{P}_{((\mathbf{a_1},\mathbf{v_1}),.....,(\mathbf{a_N},\mathbf{v_N}))} = \prod_{\mathbf{i=1}}^{\mathbf{N}} (\mathbf{1} - \rho_{\mathbf{i}})\rho_{\mathbf{i}}{}^{\mathbf{a_i+v_i}}$$

(a_i, v_i) denotes the queue i having a_i number of normal packets and v_i number of malicious packets. So for N nodes the probability to be in a particular state $\mathbf{n(t)}$

$= ((a_1, v_1), ...(a_N, v_N))$ is given by the product form solution of

$$\prod_{i=1}^{N}(1-\rho_i){\rho_i}^{a_i+v_i}$$

3.2 Single Node Network:

If we have single node in the network and we model that node as an M/M/1 queue with infinite size, Then the probability of server having a normal packets and v malicious packets is given by:

$$\mathbf{P_{(a,v)} = (1-\rho)\rho^{a+v}}$$

where a denotes the number of normal packets in the node where as v denotes the number of malicious packets in the node.

3.2.1 Numerical Results

We try to study the above described result by varying the parameters.

Varying ρ : For a single server system, ρ is a measure of utilization as well as traffic intensity of a network. For a stable system ρ should be between 0 and 1. If we vary ρ to find the maximum probability of reaching a given a and v, we know that maximum probability occurs when:

$$\tfrac{\partial P}{\partial \rho} = 0 \Rightarrow \quad \rho = \tfrac{a+v}{a+v+1}$$

Initially when ρ is near to 0, i.e. when mean arrival time is quite large compared to mean service time, then the traffic intensity is very low i.e as soon as the packet arrives it is served, hence probability to reach a certain state a, v is also very low. Now as the system become more traffic intense, mean arrival time decreases and the mean service time increases, and the probability to reach a certain state a, v also increases upto a certain limit. Now as ρ tends to 1, i.e the system become most traffic intense, then the probability to reach that state a, v again goes to 0.

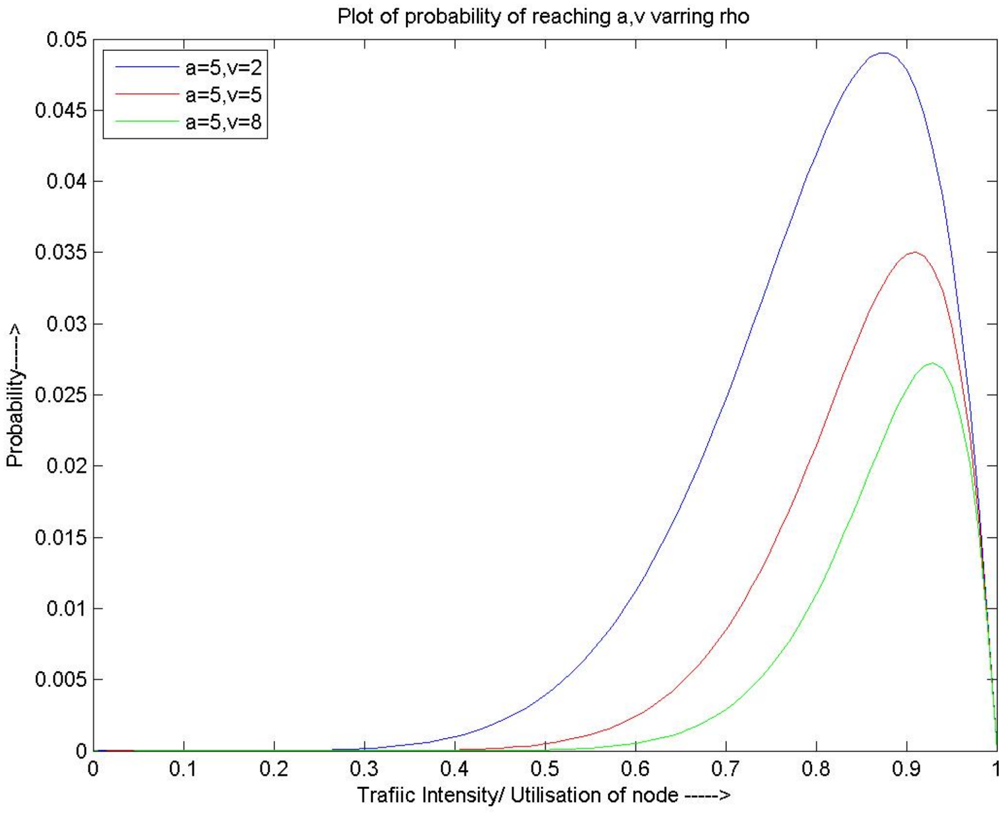


Figure 3.1: Probability of reaching a,v by varying traffic intensity

From the above graph, we see that the maximum probability to reach a certain state a, v for all the three cases occurs when ρ is near to 1. Our theoretical result also points that maximum probability is at $\frac{a+v}{a+v+1}$ which is quite nearer to 1. Apart from that, we observe that if the number of malicious packets increase, probability to reach a, v goes down further. Thus, as the value of v increases, maximum probability occurs more nearer to 1, but the maximum probability goes on decreasing. With less number of packets, we have more probability to reach certain state, compared to more number of packets. All the three cases have similar behavior when ρ is less than 0.3 and very very close to 1. So, to study the behavior of a node, we should mainly focus on the traffic intensity between 0.3 and 1;

Varying v: If we vary number of malicious packets keeping traffic intensity and number of normal packets fixed:

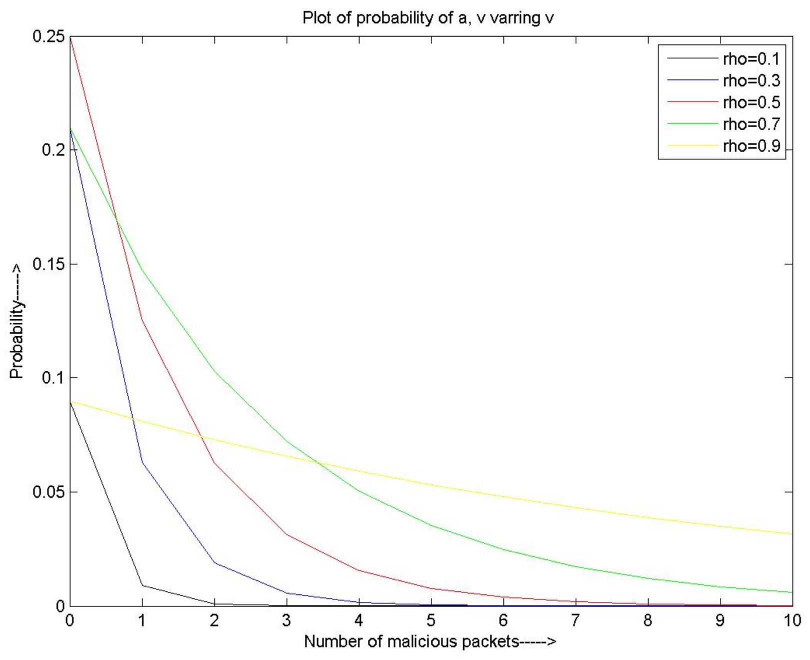


Figure 3.2: Probability of reaching a,v by taking a=1 and varying v

As the number of malicious packets increase, the probability to be on a certain state a, v goes on decreasing. We assume that our system has only one normal packet. We know that low values of ρ indicate that mean service time is faster than mean arrival time. So, as the number of malicious packets increase, probability to reach certain state a, v goes to 0 quickly for low traffic intense systems. For a given malicious packets, if we draw a vertical line in our graph, we see that for $\rho = 0.5$ we get the maximum probability to reach a certain state a, v. If we increase the number of normal packets, probability to reach a particular state converges to 0 more rapidly.

n$^{\text{th}}$ packet gets dropped: Now let us assume that the n^{th} packet received by system is dropped after the start i.e denial of service occurs for the n^{th} packet. Then, we can say that when the n^{th} packet arrives the system contains either $(0, n-1)$ or $(1, n-2)$ or ...... or $(n-1, 0)$ number of normal and malicious packets. Probability that the service is denied is given by :

$$P_{denial} = (1-\rho)\rho^{0+n-1} + (1-\rho)\rho^{1+n-2} + ... + (1-\rho)\rho^{n-1+0}$$

$$\Rightarrow P_{denial} = n(1-\rho)\rho^{n-1}$$

Varying ρ: If we vary ρ, taking n fixed we get:

$$\frac{\partial P_{denial}}{\partial \rho} = 0 \Rightarrow \quad \rho = \frac{n-1}{n}$$

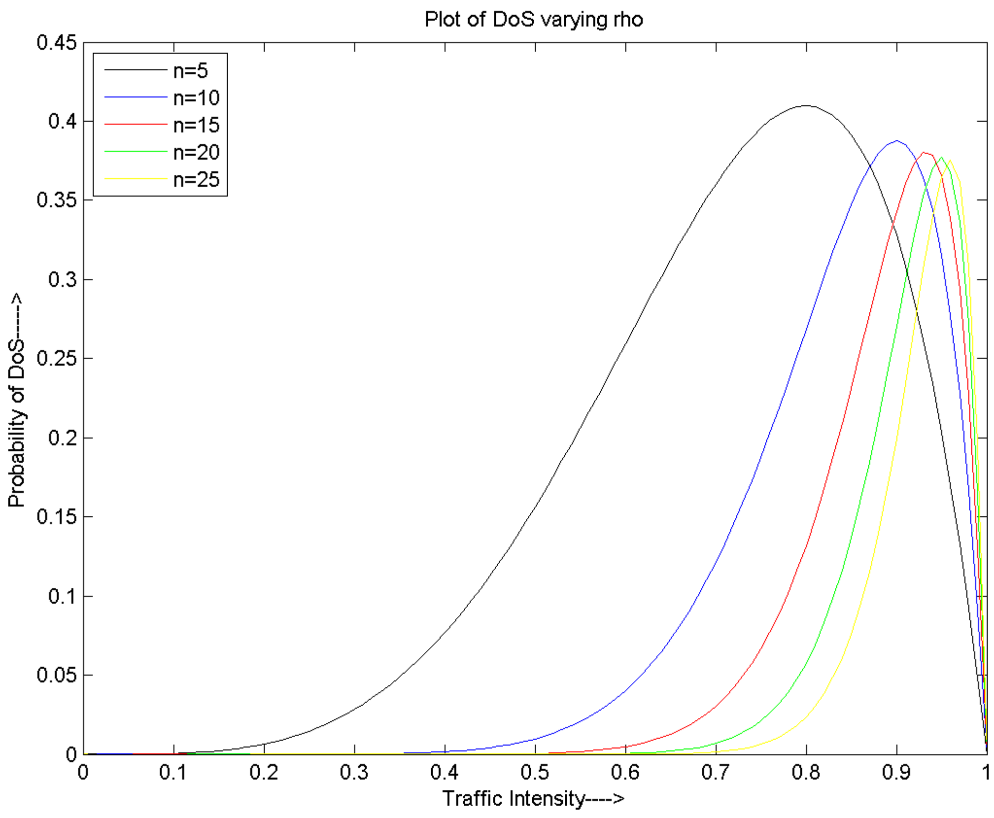


Figure 3.3: Probability of packet loss Varying rho for different n

For a fixed value of n, the value of ρ for which we get maximum probability of n^{th} packet being dropped is given by $\frac{n-1}{n}$. As the value of n increases, more and more, the ρ value to have maximum probability of the n^{th} packet being dropped tends to 1, while the probability in itself goes on decreasing. This is because if we substitute ρ in the equation for which maximum probability occurs, we get the maximum probability as $(\frac{n-1}{n})^{n-1}$ and as n increases the value become smaller and smaller. For, low values of traffic intensity, service time is faster than the arrival time, so the probability of packet being dropped is very less. As the value of traffic intensity increases, the probability to drop packets increases slowly. Once the probability reaches maximum, it decreases very rapidly. Thus, for lower values of traffic intensity change in probability is lower compared to higher values of traffic intensity.

Varying n: If we vary n taking fixed ρ :

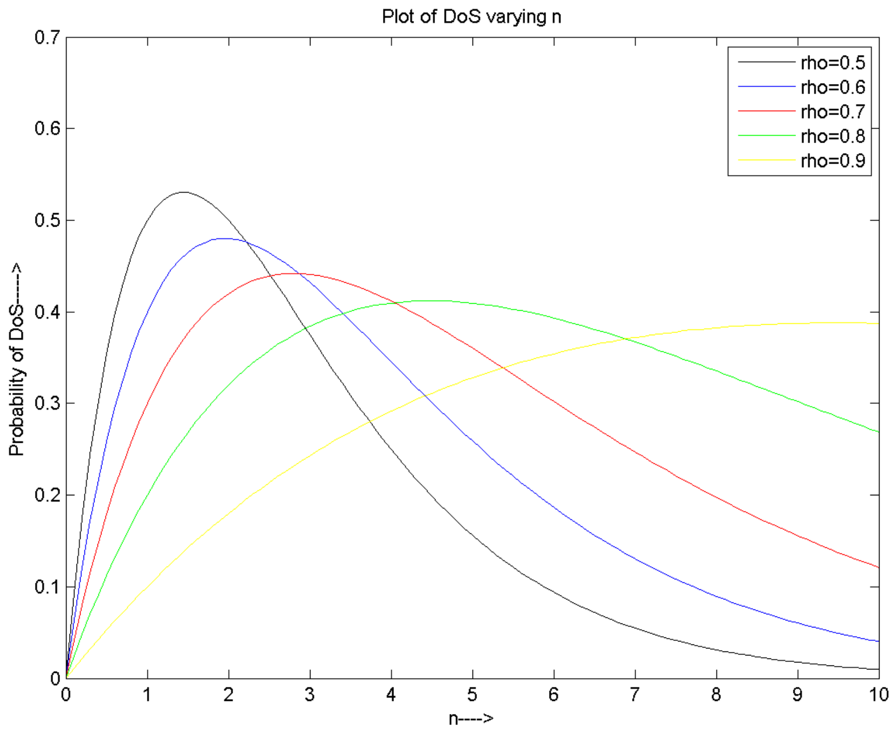


Figure 3.4: Probability of packet drop by varying n taking different rho

If we take smaller values of ρ, graph is mainly distorted and since we have shown that our main point of concentration should be when ρ varies from 0.4 to 1. so we take five different values of ρ and try to vary n. If ρ is small then for very small value of n, we get the maximum probability of packet being lost. As the value of ρ increases, we get a little larger value of n, for the maximum probability of a packet to be lost while the probability in itself goes on decreasing. If we draw a vertical line after n=8 packets, we see that higher the value of ρ, higher is the probability of a packet being lost. This result holds in with the fact that higher value of ρ indicate that mean arrival time is less and mean service time is more and hence probability of packet being dropped is more. If we want to find the probability of a packet being lost for smaller values of n, say 2, then we see that the probability of packet being lost for lower values of ρ is much more than that of higher value of ρ. So, we conclude that if for a smaller n, the packets are dropped, then the probability to drop packets for smaller values of ρ is more than larger values of ρ and for larger n, larger values of ρ are more probable to drop packets.

Varying n and ρ: If we vary n and ρ together we observe that for smaller values of n, for a very small ρ we get very high probability of the packet being dropped compared to larger values of ρ while for values of n greater than 5, we have high probability of packet being dropped for larger values of ρ.

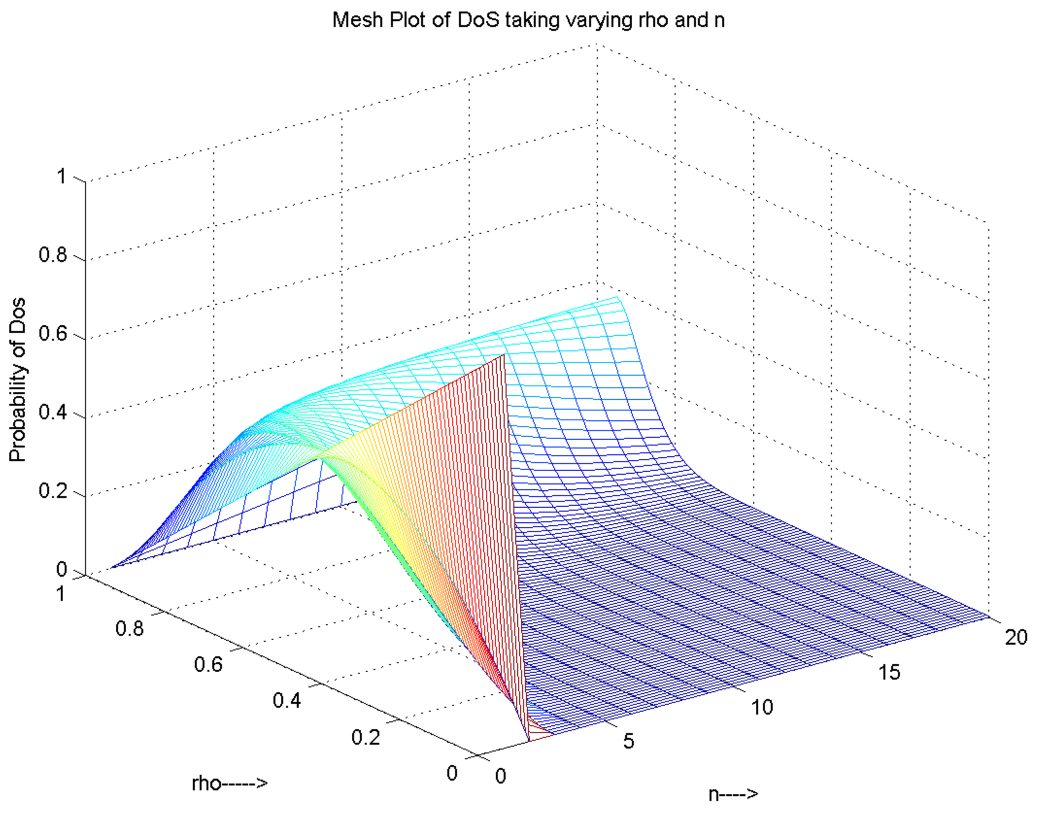


Figure 3.5: Probability of packet being dropped varying n and rho

Thus, the probability of packet being dropped is similar to denying service to that packet and we have studied the probability of packet being dropped by varying ρ and n and both of them together.

3.3 Network Simulations

For practical purposes, we present network simulations of Denial of Service Attacks. We basically focus on two fields: UDP attack over UDP client and UDP attack over TCP client. We do not consider TCP attacker because TCP protocol is for reliable transmission of the packets and an attacker's only aim is to prevent normal packets from reaching the destination not to ensure the delivery of attack packets.

3.3.1 UDP Attacker over UDP client

Here we study the system parameters in both presence and absence of attacker. We take a simple topology as described below:

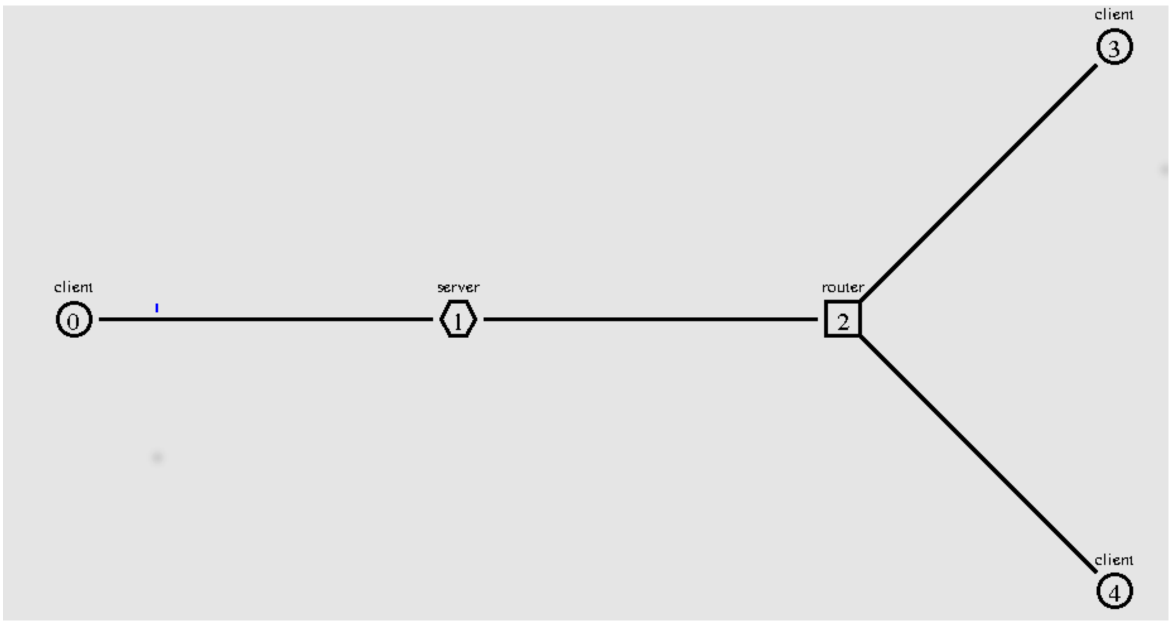


Figure 3.6: Absence of Attacker Configuration

3.3.1.1 Description of our Topology

- **Client Packets:** Packets are generated at the client side at an exponenial rate i.e the packets follow exponential arrival rate at the server and the packets have

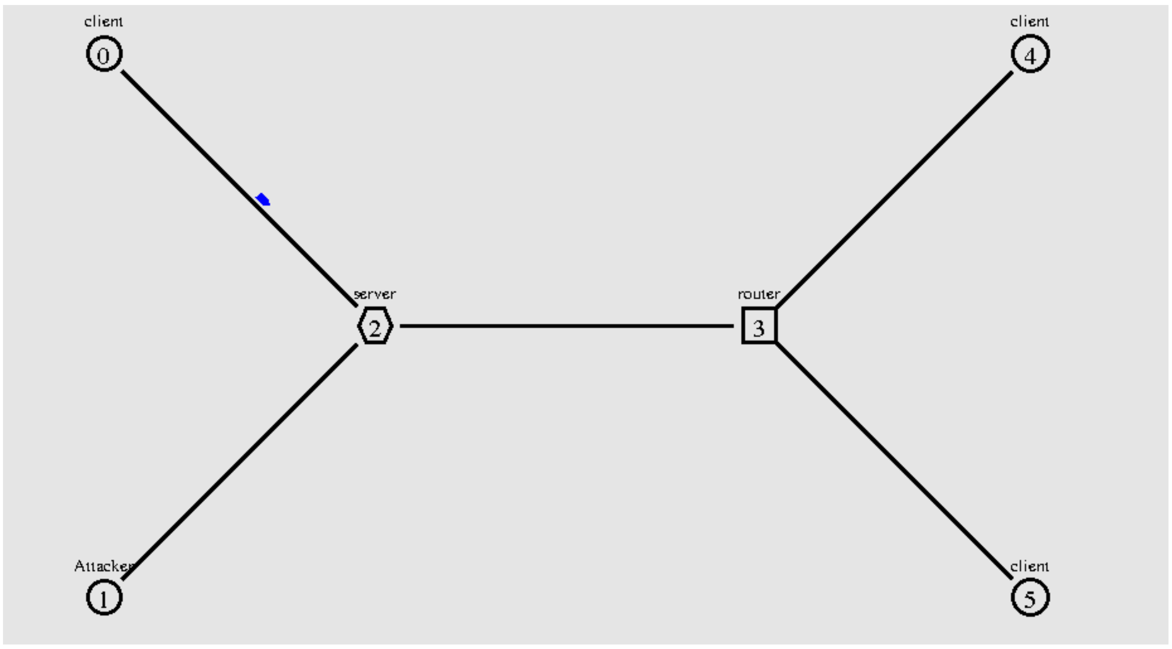


Figure 3.7: Presence of Attacker Configuration

varying length and the lengths are simulated by exponential distribution. Varying packet lengths produces exponential service rate.

- **Attacker Packets:** The attacker packets also follow exponential service rate but have uniform packet size and hence uniform service rate.
- **Link Description:** Each links have 100 ms propagation delay and 0.5Mbps link speed in a fixed Bandwidth case i.e Each links are of same length and Each link provides equal service.
- **Queue Description:** Our Queuing model follows M/D/1/K model i.e the arrival rate is exponential, service rate is uniform, single server and queue size is k. We fix our Queuing size to be 15.
- **Protocol Description:** Our client and attacker on the left hand side (as shown in figure 3.7) both connects with User Datagram Protocol with both the clients on the right hand side. Each packet has a certain random probability which decides the receiver of the packet.

- **Simulation Time:** We simulate the entire duration for 100 secs. For the first 10 secs normal packet generation takes place. After 10 secs the attacker starts working. At t=40 sec, the attacker stops working and at t=60 sec it starts again. At t=90 sec the attacker finishes itself and finally at t=100 sec normal packet generation also stops.

3.3.1.2 Network Simulation Code

```
# Create a simulator object
set ns [new Simulator]

# Define different colors for data flows (for NAM)
$ns color 1 Blue
$ns color 2 Red

# open the trace file
set filename [lindex $argv 0]
set tracefile1 [open $filename w]
$ns trace-all $tracefile1

# Open the NAM trace file
set namfile [open out.nam w]
$ns namtrace-all $namfile

# Define a 'finish' procedure
proc finish {} {
    global ns tracefile1 namfile
    $ns flush-trace
    close $tracefile1
    close $namfile
   #exec nam out.nam &
```

```
    exit 0
}

set lamda 50
set mu [lindex $argv 2]

set MyRng [new RNG]
$MyRng seed 0
set iat_udp [new RandomVariable/Exponential]
$iat_udp set avg_ [expr 1.0/$lamda]
$iat_udp use-rng $MyRng

set MyRng1 [new RNG]
$MyRng1 seed 0
set iat_tcp [new RandomVariable/Exponential]
$iat_tcp use-rng $MyRng1
$iat_tcp set avg_ [expr 1.0/$lamda]

set MyRng2 [new RNG]
$MyRng2 seed 0
set pktsize [new RandomVariable/Exponential]
$pktsize use-rng $MyRng2
$pktsize set avg_ [expr 10000.0/($mu)]

set MyRng3 [new RNG]
$MyRng3 seed 0
set prob [new RandomVariable/Uniform]
$prob use-rng $MyRng3

# Create 6 nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]

# Create links between nodes
$ns duplex-link $n0 $n2 0.5Mb 100ms DropTail
$ns duplex-link $n1 $n2 0.5Mb 100ms DropTail
$ns duplex-link $n2 $n3 0.3Mb 100ms DropTail
$ns duplex-link $n3 $n4 0.2Mb 100ms DropTail
$ns duplex-link $n3 $n5 0.2Mb 100ms DropTail

# Setting node positions
$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n2 $n3 orient right
$ns duplex-link-op $n3 $n4 orient right-up
$ns duplex-link-op $n3 $n5 orient right-down

# Set Queue positions
$ns simplex-link-op $n2 $n3 queuePos 0.5
$ns duplex-link-op $n0 $n2 queuePos 0.5
$ns duplex-link-op $n1 $n2 queuePos 0.5

# Set Queue Sizes
$ns queue-limit $n0 $n2 10
$ns queue-limit $n1 $n2 15
$ns queue-limit $n2 $n3 15
$ns queue-limit $n3 $n4 5
```

```
# Labelling
$ns at 0.0 "$n2 label server"
$ns at 0.0 "$n3 label router"
$ns at 0.0 "$n0 label client"
$ns at 0.0 "$n1 label Attacker"
$ns at 0.0 "$n4 label client"
$ns at 0.0 "$n5 label client"

# Shape
$n2 shape hexagon
$n3 shape square

# Monitoring queue
set filenam [lindex $argv 1]
set qmon [$ns monitor-queue $n2 $n3 [open $filenam w] 1.0];
[$ns link $n2 $n3] queue-sample-timeout;

# Defining a Random Uniform Generator
proc Random_Generator_Uniform {} {
    set MyRng4 [new RNG]
    $MyRng4 seed 0
    set r4 [new RandomVariable/Uniform]
    $r4 use-rng $MyRng4
    $r4 set min_ 0
    $r4 set max_ 4
    puts stdout "Testing Uniform Random Variable inside function"
    global x
    set x [$r4 value]
    return x
}
```

```
proc Random_Generator_Exponential {} {
    set MyRng5 [new RNG]
    $MyRng5 seed 0
    set r5 [new RandomVariable/Exponential]
    $r5 use-rng $MyRng5
    $r5 set avg_ 100
    puts stdout "Testing Exponantial Random Variable inside function"
    global y
    set y [$r5 value]
    return y
}

# Setup a TCP connection
set tcp [new Agent/UDP]
$ns attach-agent $n0 $tcp
set sink [new Agent/Null]
$ns attach-agent $n4 $sink
$ns connect $tcp $sink
$tcp set fid_ 1
#$tcp set packetsize_ 552

set tcp2 [new Agent/UDP]
$ns attach-agent $n0 $tcp2
set sink2 [new Agent/Null]
$ns attach-agent $n5 $sink2
$ns connect $tcp2 $sink2
$tcp2 set fid_ 1

# Setup a UDP connection
set tcp1 [new Agent/UDP]
$ns attach-agent $n1 $tcp1
```

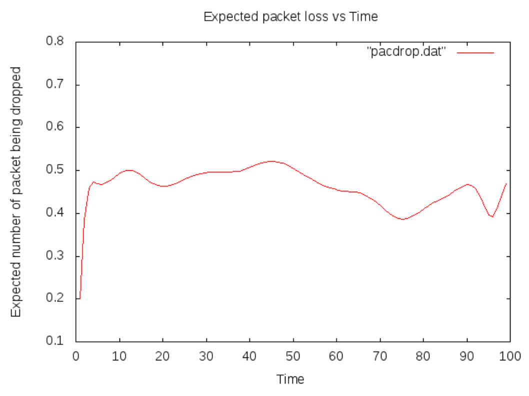
```
set sink1 [new Agent/Null]
$ns attach-agent $n4 $sink1
$ns connect $tcp1 $sink1
$tcp1 set fid_ 2

set tcp3 [new Agent/UDP]
$ns attach-agent $n1 $tcp3
set sink3 [new Agent/Null]
$ns attach-agent $n5 $sink3
$ns connect $tcp3 $sink3
$tcp3 set fid_ 2

proc sendpacket_udp {} {
    global ns tcp iat_udp pktsize prob tcp2
    set time [$ns now]
    $ns at [ expr $time + [$iat_udp value]] "sendpacket_udp"
    set bytes [expr round ([$pktsize value])]
    if [expr [$prob value] < 0.5] {
        #puts "1"
        $tcp send $bytes
    } else {
        #puts "2"
        $tcp2 send $bytes
    }
}

proc sendpacket_tcp {tm} {
    global ns tcp1 iat_tcp tcp3 prob sendtime
    set time [$ns now]
    set sendtime [ expr $time + [$iat_tcp value]]
    if { $tm > $sendtime } {
        $ns at $sendtime "sendpacket_tcp $tm"
        set bytes 512
        if [expr [$prob value] < 0.5] {
            $tcp1 send $bytes
        } else {
            $tcp3 send $bytes
        }
    }
}

$ns at 0.0001 "sendpacket_udp"
$ns at 10.0001 "sendpacket_tcp 40"
$ns at 60.0000 "sendpacket_tcp 90"

#Call the finish procedure after 5 seconds of simulation time
$ns at 100.0 "finish"

#Run the simulation
$ns run
```

Listing 3.1: NS2 Code which simulates the network

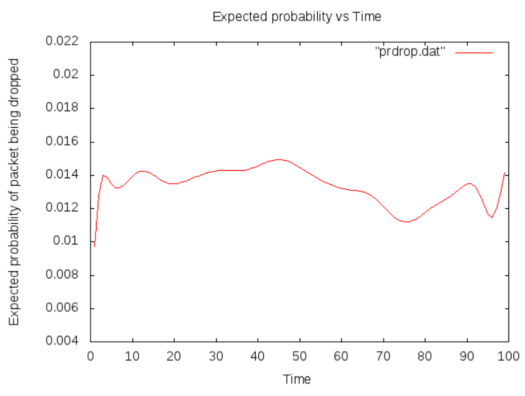
3.3.1.3 Experimental Analysis and Results:

For simulation purpose we run 100 simulations to produce our results. We basically study five things: Expected Throughput, Expected probability of packet being dropped, Expected number of packets being dropped, Expected size of queue, Expected waiting time in a queue. The parameters, we vary to study the system are: time, Arrival Rate, Service Rate and Bandwidth. The parameters are studied in both presence and absence of attacker.

Fixed Arrival Rate, Service Rate and Bandwidth:

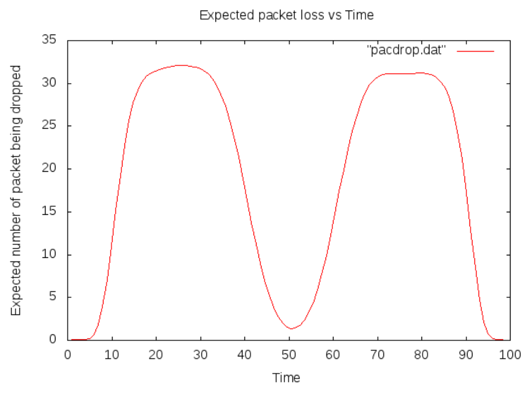


(a) Number of packets dropped

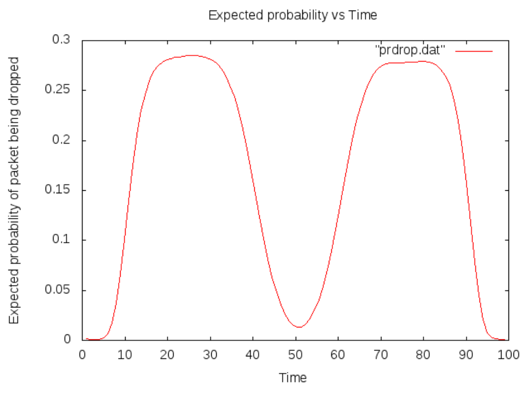


(b) Probability of packets dropped

Figure 3.8: Number of Packets and probability of packet being dropped in absence of Attacker Configuration



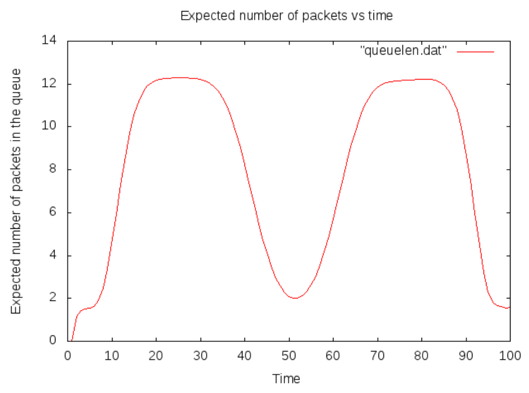
(a) Number of packets dropped



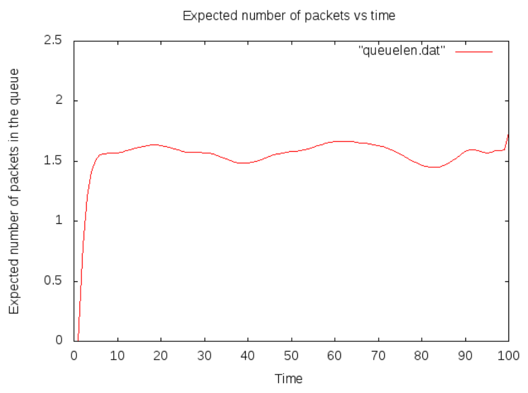
(b) Probability of packets dropped

Figure 3.9: Number of Packets and probability of packets dropped in presence of Attacker Configuration

As attack starts at time t=10 sec, the number of packets being dropped starts increasing. As we switch off the attack, the packets dropped almost tends to 0 which is similar to the case when there is no attacker and the number of packets being dropped is almost 0. Similar situation exists for the probability of packet being dropped.



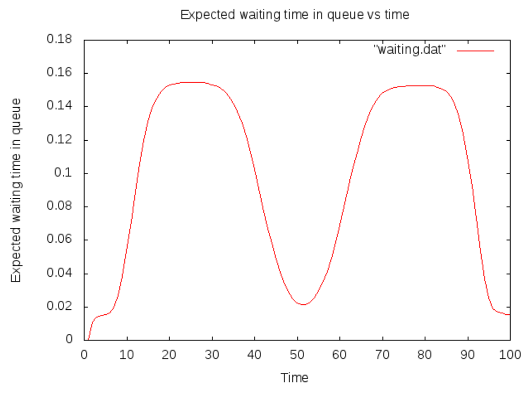
(a) Presence of Attacker



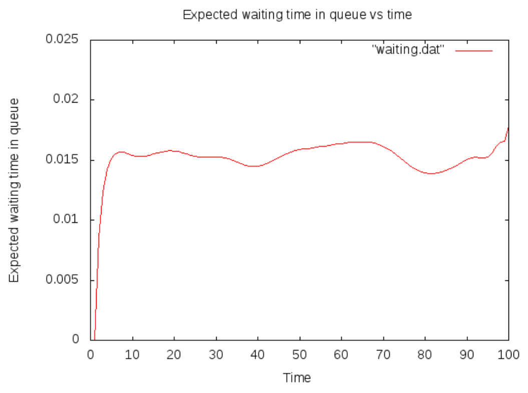
(b) Absence of Attacker

Figure 3.10: Expected Queue length in terms of packets

During the presence of attack, large number of packets are generated and hence the queue size is almost full. When we switch off the attack, queue size tends to almost 0.



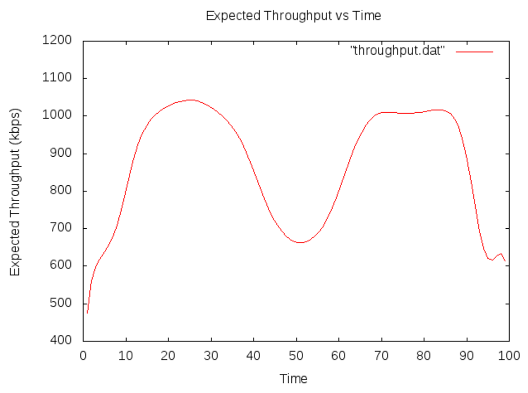
(a) Presence of Attacker



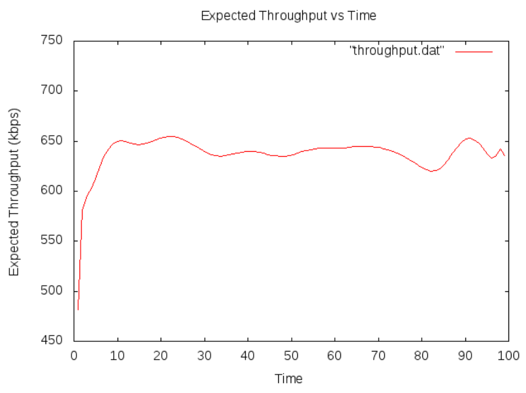
(b) Absence of Attacker

Figure 3.11: Expected waiting time in a queue

We know that waiting time is proportional to the queue size. More the queue size, more is the waiting time. In absence of attacker both queue sizes and waiting time is very low. Waiting time is almost 10 times in presence of attacker compared to the absence of attacker.



(a) Presence of Attacker



(b) Absence of Attacker

Figure 3.12: Expected Throughput vs Time

If number of packets flowing through the system increases, system throughput increases due to more and more utilisation. Higher the link utilisation, higher the number of packets dropped. System throughput is almost 1.5 times in presence of attacker compared to the absence of attacker.

Varying Arrival Rate, fixed Service Rate and Bandwidth:

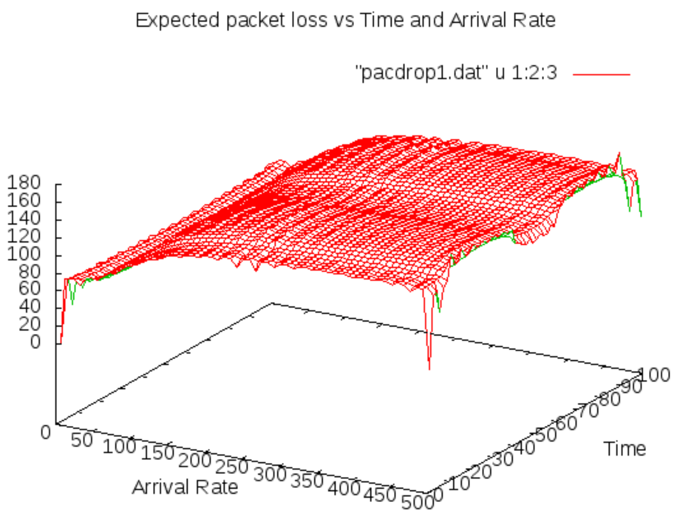


Figure 3.13: Expected number of packets dropped in presence of Attacker Configuration varying Arrival Rate

In absence of attacker, expected number of packets dropped is less than 1 but in presence of attacker, as the arrival rate increases, number of packets dropped goes on increasing. As the time increases, expected number of packets dropped also goes on increasing. When we switch off the attacker between 40 and 60 secs, packets dropped decreases so we find a buldge in our graph. Expected probability of packets being lost behave in a similar manner.

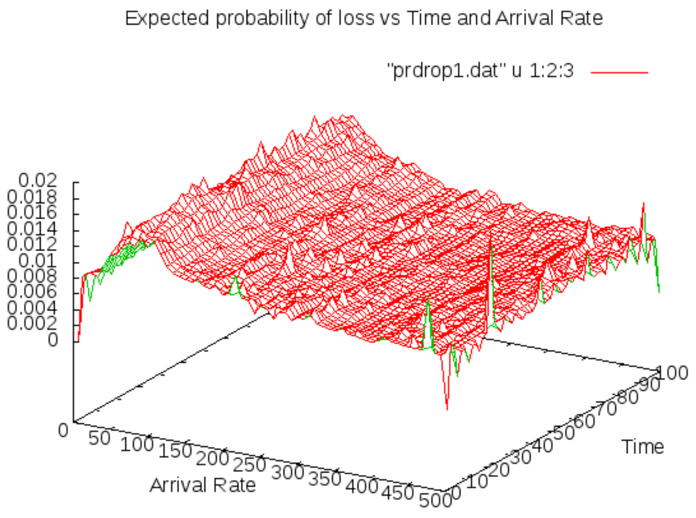


Figure 3.14: Expected probability of packets dropped in absence of Attacker Configuration varying Arrival Rate

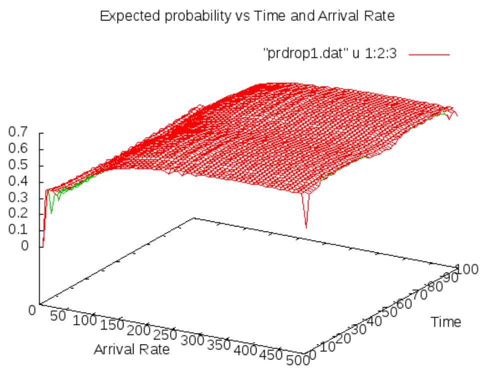


Figure 3.15: Expected probability of packets dropped in presence of Attacker Configuration varying Arrival Rate

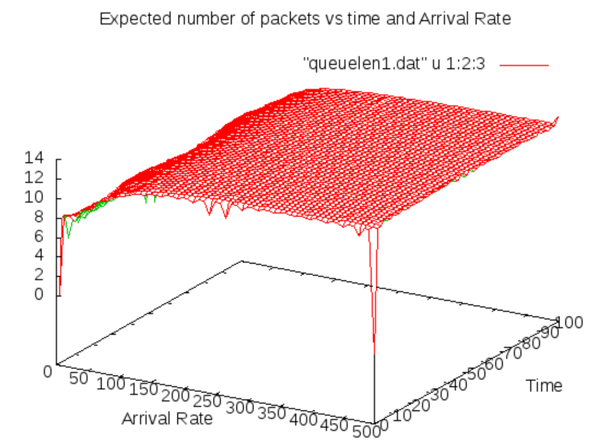


Figure 3.16: Expected number of packets in queue in presence of Attacker Configuration varying Arrival Rate

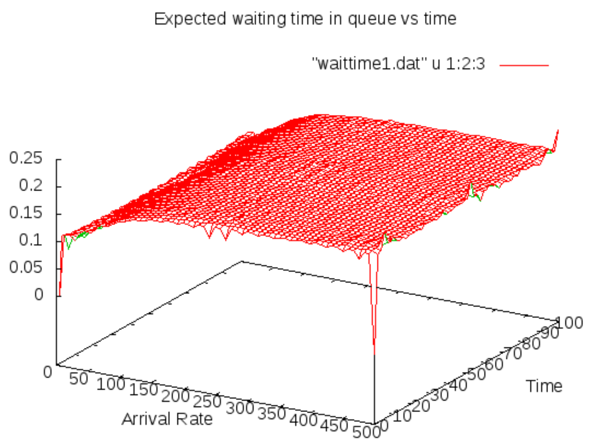


Figure 3.17: Expected waiting time in presence of Attacker Configuration varying Arrival Rate

As the arrival rate increases, queue length increases because the number of packets increases in the system. Hence the waiting time also goes on increasing. Queue length is almost full for large value of arrival rate. Without the presence of attacker, queue length is uniform through out time.

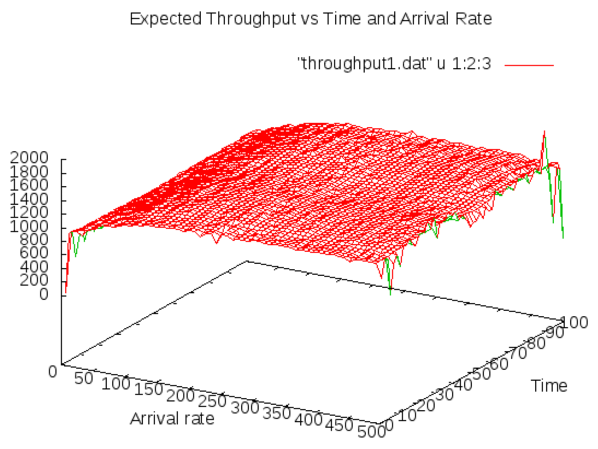


Figure 3.18: Expected Throughput in presence of Attacker Configuration varying Arrival Rate

Increase in arrival rate results in increase of number of packets which in result increases the link utilisation.

Varying Service Rate, fixed Arrival Rate and Bandwidth:

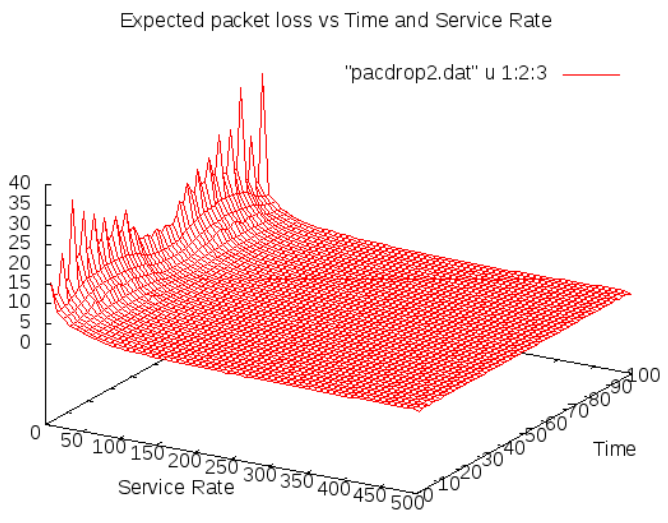


Figure 3.19: Expected number of packets dropped in presence of Attacker Configuration varying Service Rate

For smaller values of service rate, expected number of packets dropped is very large. As the value of service rate increases, as soon as the packet arrives, it is served hence queue is almost empty and packet loss goes on decreasing and it almost goes to 0 for large value of service rate. Similar behaviour occurs for expected probability of packet being dropped.

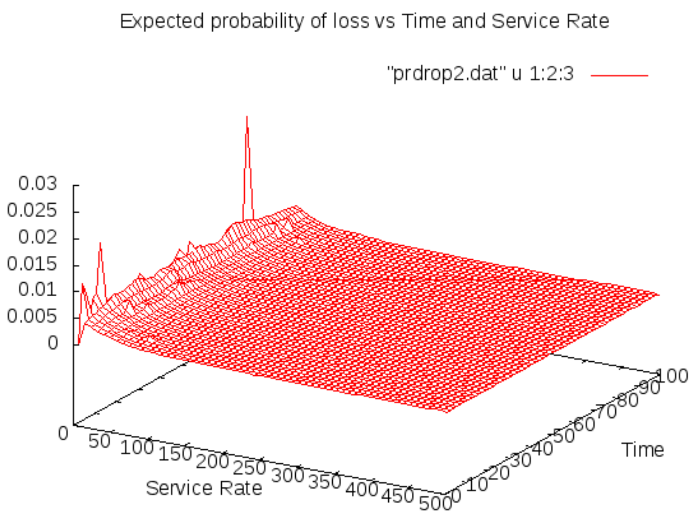


Figure 3.20: Expected probability of packets dropped in absence of Attacker Configuration varying Service Rate

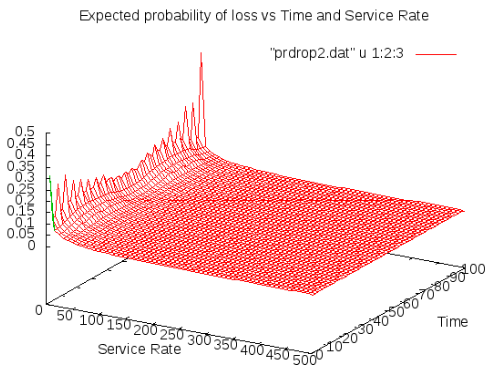


Figure 3.21: Expected probability of packets dropped in presence of Attacker Configuration varying Service Rate

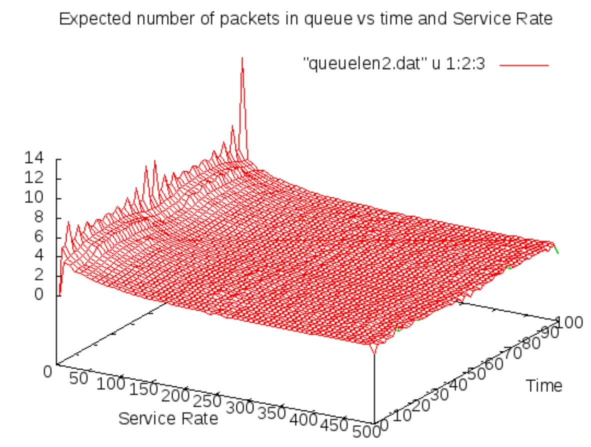


Figure 3.22: Expected number of packets in queue in presence of Attacker Configuration varying Service Rate

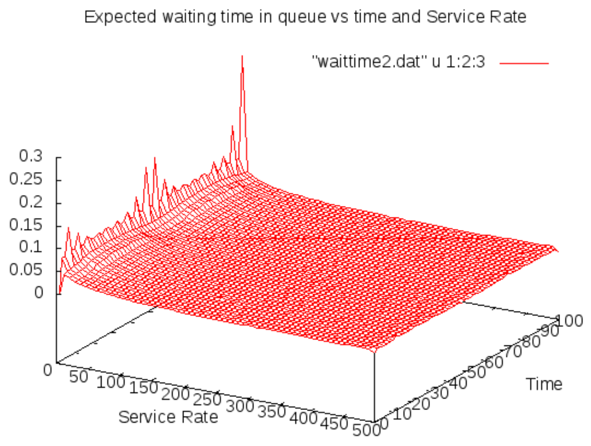


Figure 3.23: Expected waiting time in presence of Attacker Configuration varying Service Rate

As the service rate increases, queue length decreases because the number of packets decreases in the system because the packets are served very quickly. Hence the waiting time also goes on decreasing. Queue length is almost full for small values of service rate. Without the presence of attacker, queue length is uniform through out time. Between 40.0 and 60.0 secs, when we stop the attacker queue length further goes down for small values of survival rate. For larger values of service rate, queue length is uniform through out time and hence it is almost 0.

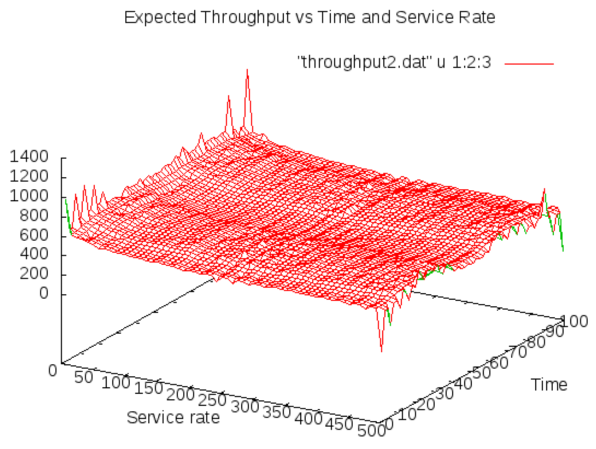


Figure 3.24: Expected Throughput in presence of Attacker Configuration varying Service Rate

Increase in service rate results in decrease of number of packets which in result decreases the link utilisation.

Varying Bandwidth, fixed Arrival Rate and Service Rate:

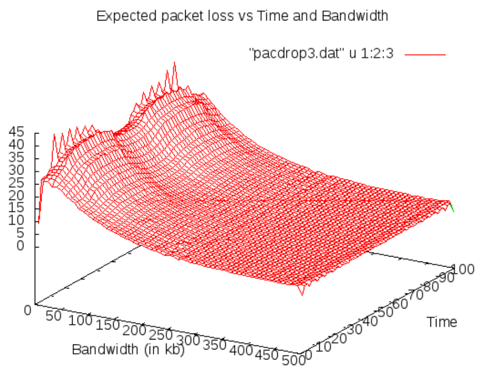


Figure 3.25: Expected number of packets dropped in presence of Attacker Configuration varying bandwidth

For low values of Bandwidth, expected number of packets dropped is very large. As the value of Bandwidth increases, packets are served quickly, hence queue is almost empty and packet loss goes on decreasing and it almost goes to 0 for large value of Bandwidth. Similar behaviour occurs for expected probability of packet being dropped.

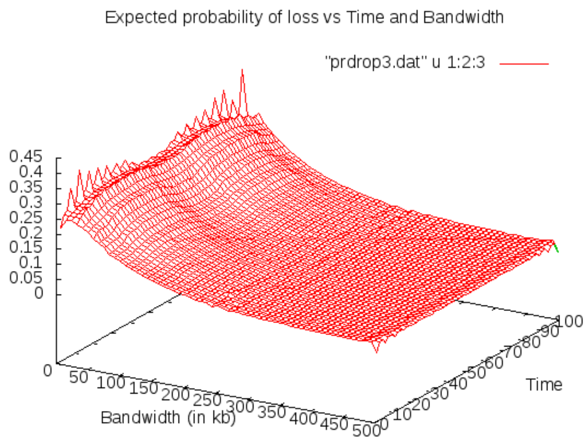


Figure 3.26: Expected probability of packets dropped in presence of Attacker Configuration varying bandwidth

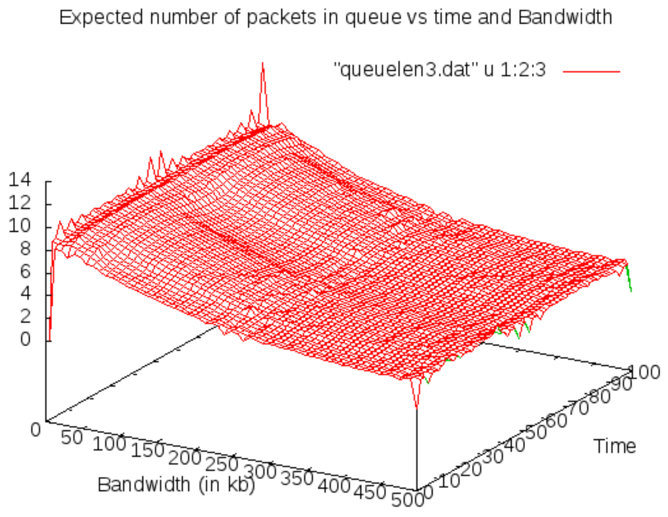


Figure 3.27: Expected number of packets in queue in presence of Attacker Configuration varying bandwidth

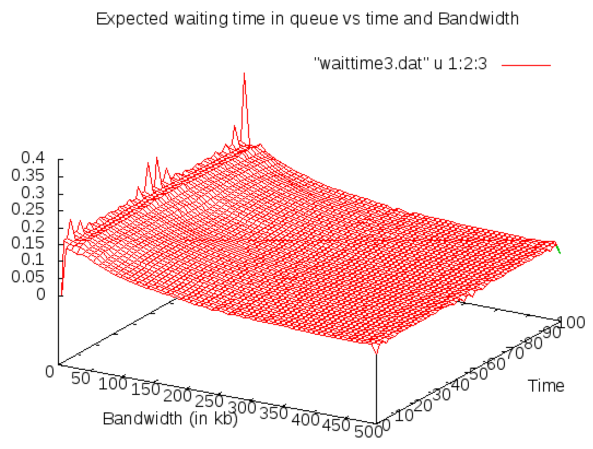


Figure 3.28: Expected waiting time in presence of Attacker Configuration varying bandwidth

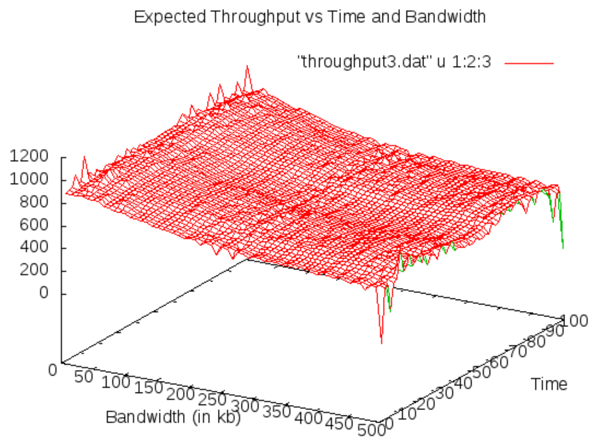


Figure 3.29: Expected Throughput in presence of Attacker Configuration varying Bandwidth

As the Bandwidth increases, queue length decreases because the number of packets decreases in the system because the packets are served very quickly. Hence the waiting time also goes on decreasing. Queue length is almost full for small values of Bandwidth since packets are served very slowly. Increase in Bandwidth results in decrease of number of packets which in result decreases the link utilisation.

3.3.2 UDP Attacker over TCP client

We take the same model as described in figure 3.7. But instead of UDP client we take a TCP client on the left side and try to study system performance. Transmission Control Protocol provides better flow control, congestion control and error detection mechanism compared to UDP. Once the packets are lost we have no control over the packet in UDP but in TCP due to retransmission system study becomes more complicated to study.

3.3.2.1 Description of our Topology

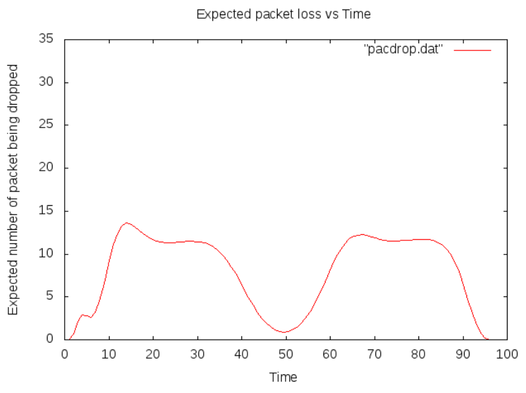
- **Client Packets:** Packets are generated at the client side at an exponenial rate i.e the packets follow exponential arrival rate at the server. Since our client is a TCP connection, packets are generated at the multiple of 2 unless packet drop occurs in which case generation takes place linearly.
- **Attacker Packets:** The attacker packets also follow exponential service rate but have uniform packet size and hence uniform service rate.
- **Link Description:** Each links have 100 ms propagation delay and 0.5Mbps link speed in a fixed Bandwidth case i.e Each links are of same length and Each link provides equal service.
- **Queue Description:** Our Queuing model follows M/D/1/K model i.e the arrival rate is exponential, service rate is uniform, single server and queue size is k. We fix our Queuing size to be 10.
- **Protocol Description:** Our client and attacker on the left hand side (as shown in figure 3.7) both connects with Transmission Control Protocol with both the clients on the right hand side. Each packet has a certain random probability which decides the receiver of the packet.
- **Simulation Time:** We simulate the entire duration for 100 secs. For the first 10 secs normal packet generation takes place. After 10 secs the attacker starts

working. At t=40 sec, the attacker stops working and at t=60 sec it starts again. At t=90 sec the attacker finishes itself and finally at t=100 sec normal packet generation also stops.

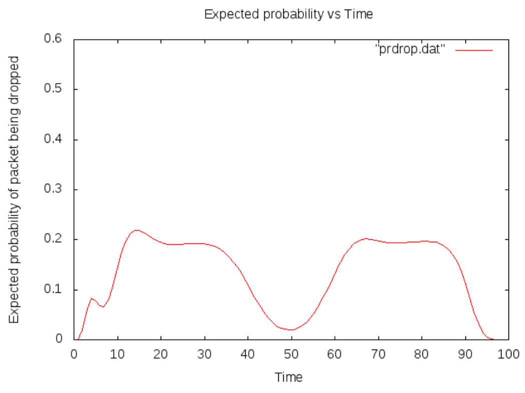
3.3.2.2 Experimental Analysis and Results:

For simulation purpose we run 100 simulations to produce our results. We basically study five things: Expected Throughput, Expected probability of packet being dropped, Expected number of packets being dropped, Expected size of queue, Expected waiting time in a queue. The parameters, we vary to study the system are: time, Arrival Rate, Service Rate and Bandwidth. The parameters are studied in both presence and absence of attacker.

Fixed Arrival Rate, Service Rate and Bandwidth:



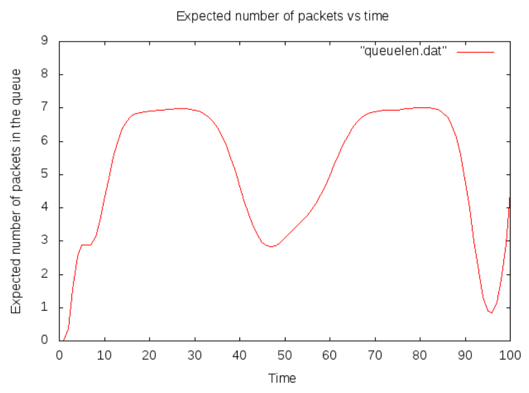
(a) Number of packets dropped



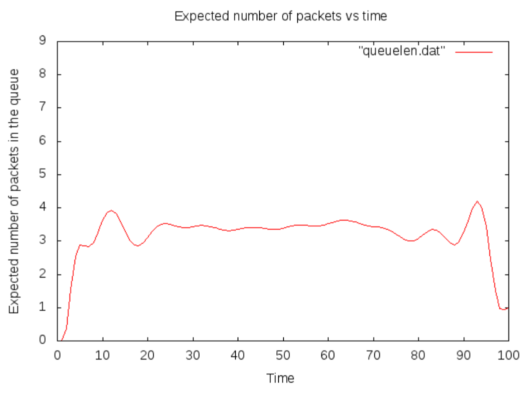
(b) Probability of packets dropped

Figure 3.30: Number of Packets and probability of packets dropped in presence of Attacker Configuration

As attack starts at time t=10 sec, expected number of packets being dropped starts increasing. As we switch off the attack, the packets dropped almost tends to 0 which is similar to the case when there is no attacker and the number of packets being dropped is almost 0. Similar situation exists for the expected probability of packet being dropped. The expected number of packets dropped in TCP case is almost half of that of UDP case.



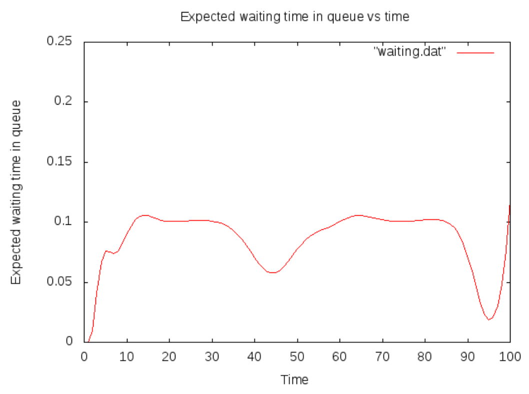
(a) Presence of Attacker



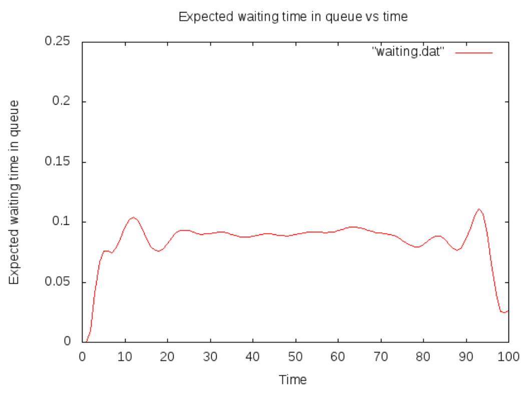
(b) Absence of Attacker

Figure 3.31: Expected Queue length in terms of packets

During the presence of attack, large number of packets are generated and hence the queue size is almost full. When we switch off the attack, queue size tends to almost 0.



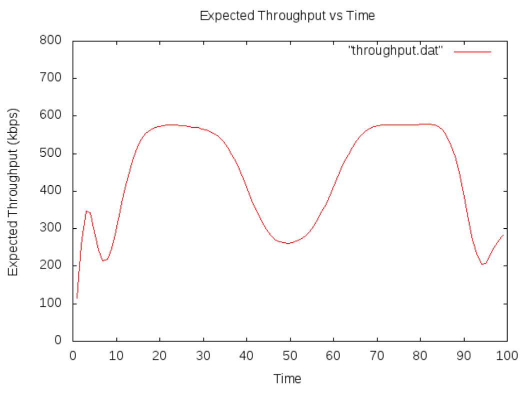
(a) Presence of Attacker



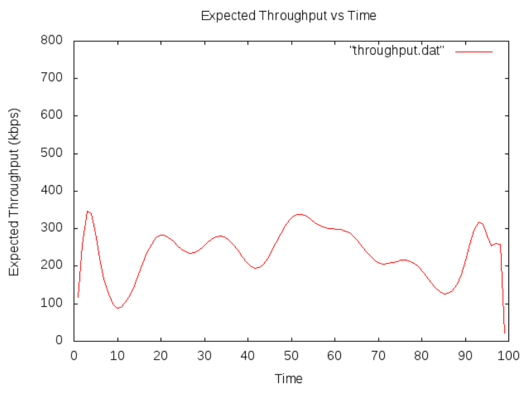
(b) Absence of Attacker

Figure 3.32: Expected waiting time in a queue

We know that waiting time is proportional to the queue size. More the queue size, more is the waiting time. In absence of attacker both queue sizes and waiting time is very low.



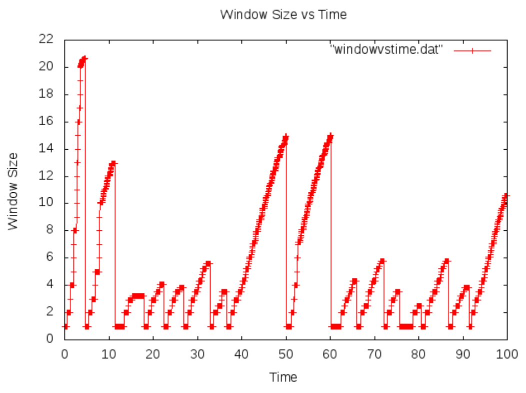
(a) Presence of Attacker



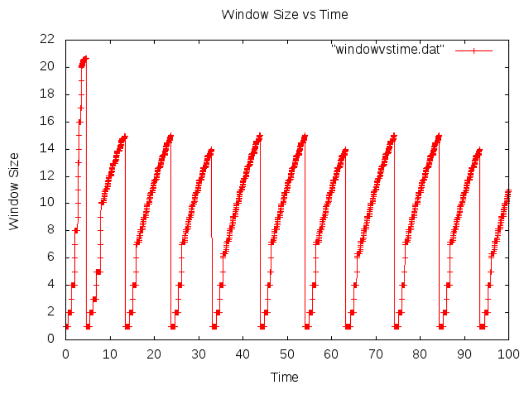
(b) Absence of Attacker

Figure 3.33: Expected Throughput vs Time

If number of packets flowing through the system increases, system throughput increases due to more and more utilisation. Higher the link utilisation, higher the number of packets dropped. System throughput is almost 2 times in presence of attacker compared to the absence of attacker.



(a) Presence of Attacker



(b) Absence of Attacker

Figure 3.34: Congestion Window vs Time

For TCP connection, packets are send exponentially and if drop occurs half of that number of packets are send exponentially and then linear sending occurs. In absence of attacker, the pattern is similar to the actual TCP result but as we introduce the attacker, TCP window size decreases, when we switch off the attacker between 40.0 and 60.0 there is linear increase in the number of packets until it gets dropped.

Varying Arrival Rate, fixed Service Rate and Bandwidth:

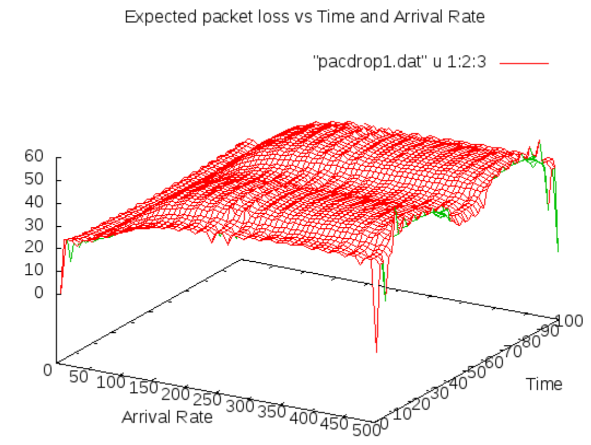


Figure 3.35: Expected number of packets dropped in presence of Attacker Configuration varying Arrival Rate

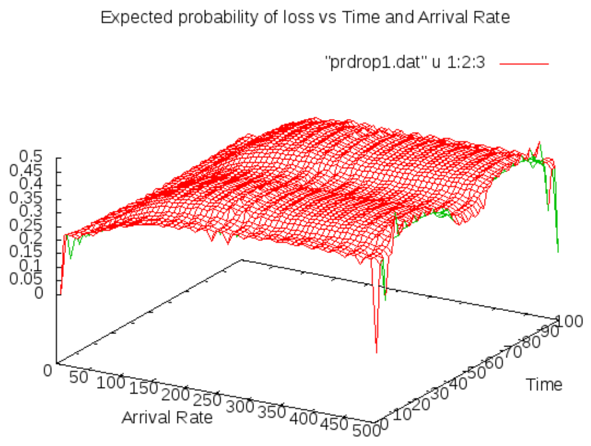


Figure 3.36: Expected probability of packets dropped in presence of Attacker Configuration varying Arrival Rate

As the arrival rate increases, expected number of packets being dropped goes on increasing. As the time increases, expected number of packets dropped also goes on increasing. When we switch off the attacker between 40 and 60 secs, packets dropped decreases so we find a buldge in our graph. Expected probability of packets being lost behave in a similar manner. Expected number of packets being dropped in case of UDP were thrice in comparision with TCP.

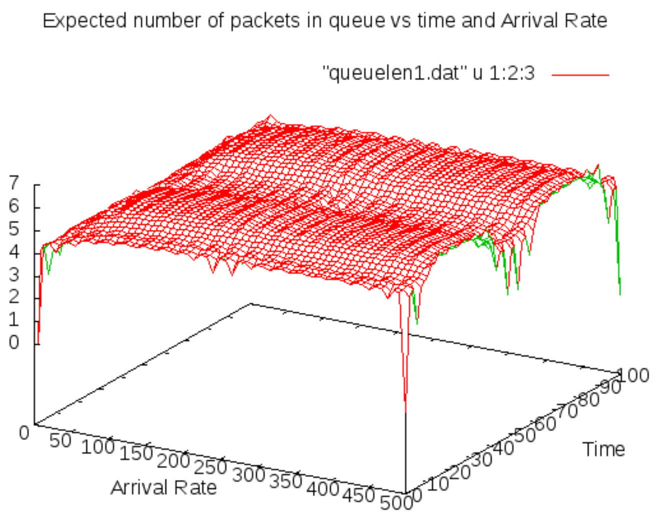


Figure 3.37: Expected number of packets in queue in presence of Attacker Configuration varying Arrival Rate

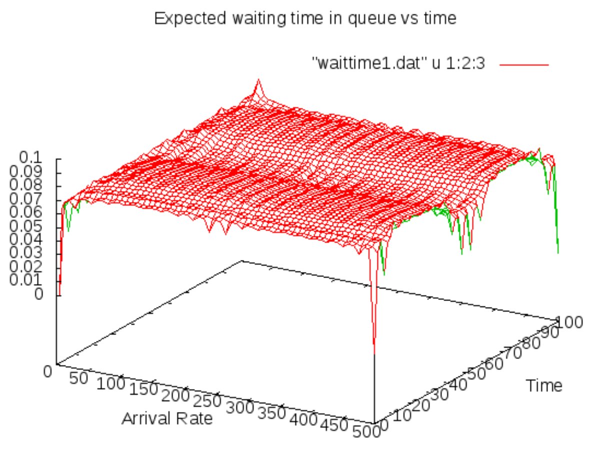


Figure 3.38: Expected waiting time in presence of Attacker Configuration varying Arrival Rate

As the arrival rate increases, expected queue length increases because the number of packets increases in the system. Hence the waiting time also goes on increasing. Here the expected queue length does not increase abruptly as that of in UDP connection. Similarly the average waiting time does not change much as the arrival rate increases because once the packets are dropped, TCP starts checking the congestion window and start retransmitting only one packet. Thus TCP congestion control mechanism helps the average waiting time and expected queue size to change uniformly with the increase in arrival rate.

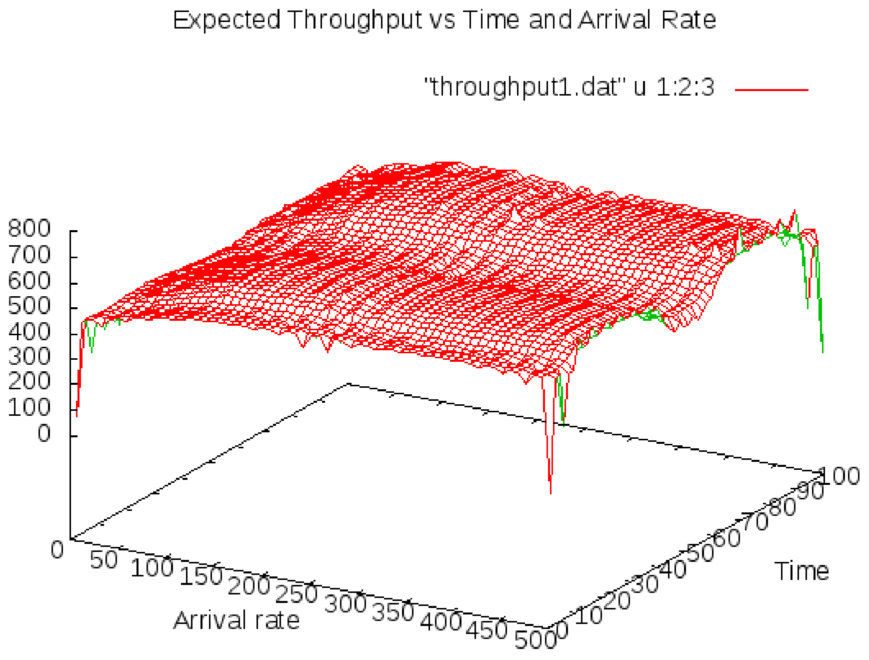


Figure 3.39: Expected Throughput in presence of Attacker Configuration varying Arrival Rate

Increase in arrival rate results in increase of number of packets which in result increases the link utilisation. The link utilisation is almost half of that of UDP connection due to better flow and congestion control mechanism.

Varying Service Rate, fixed Arrival Rate and Bandwidth:

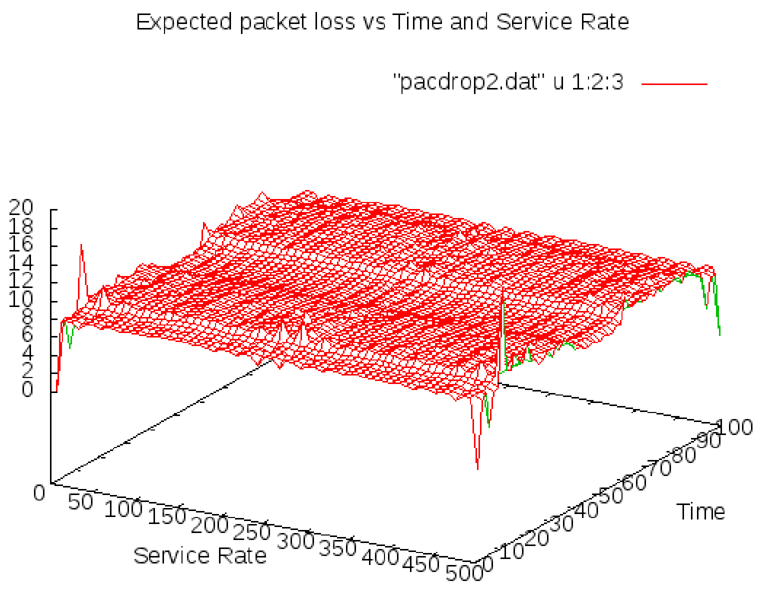


Figure 3.40: Expected number of packets dropped in presence of Attacker Configuration varying Service Rate

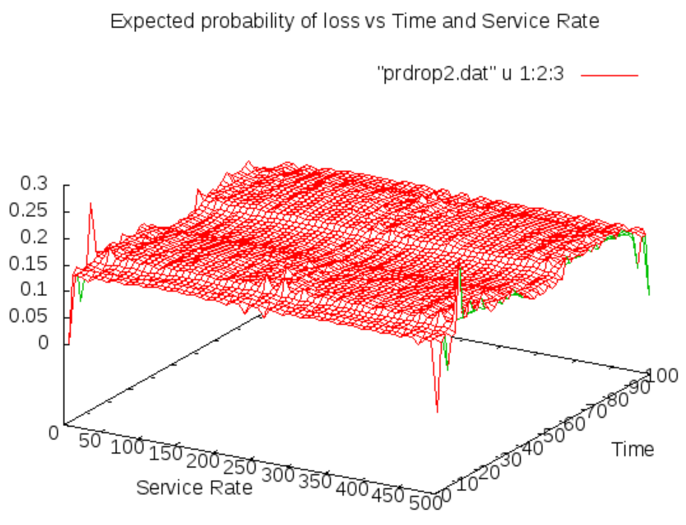


Figure 3.41: Expected probability of packets dropped in presence of Attacker Configuration varying Service Rate

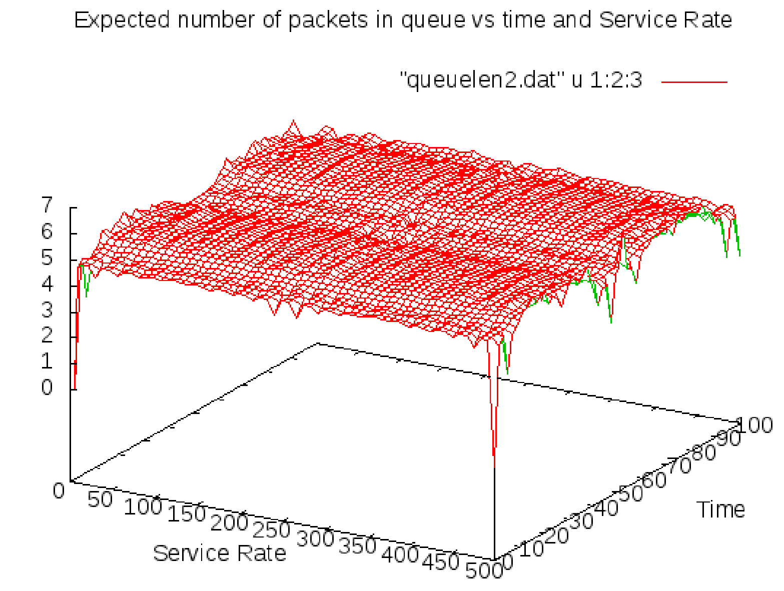


Figure 3.42: Expected number of packets in queue in presence of Attacker Configuration varying Service Rate

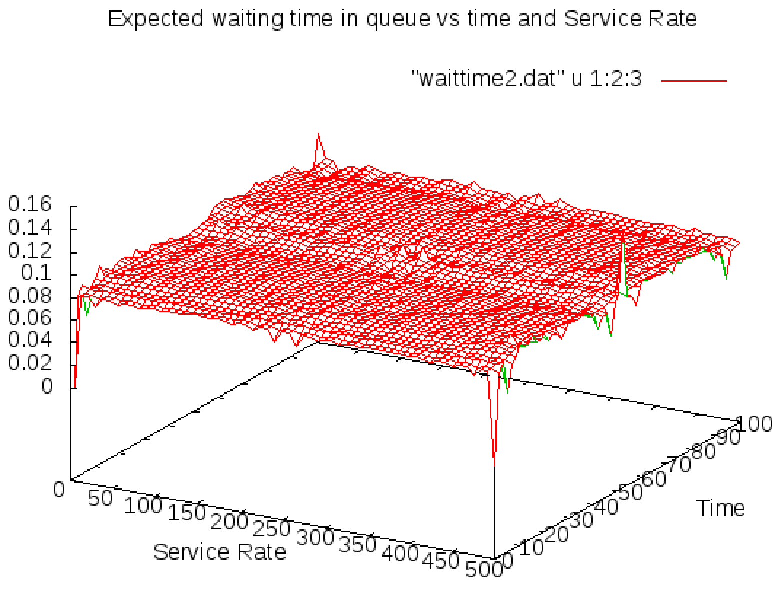


Figure 3.43: Expected waiting time in presence of Attacker Configuration varying Service Rate

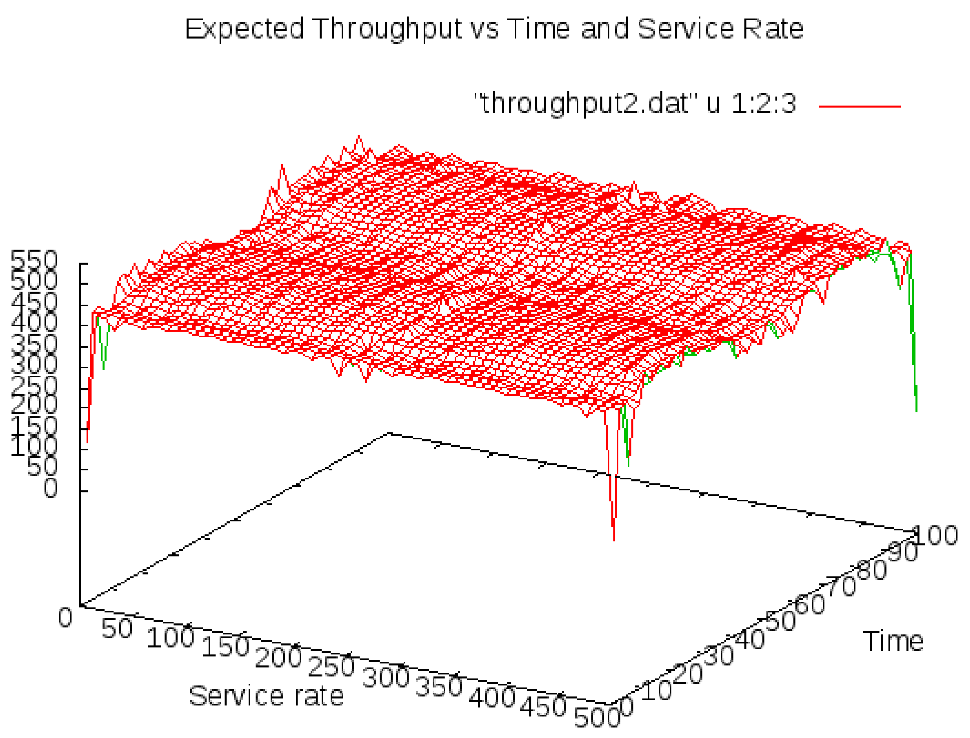


Figure 3.44: Expected Throughput in presence of Attacker Configuration varying Service Rate

For smaller values of service rate, packet drop occurs very rapidly. Due to presence of control mechanisms of TCP, packets send to the server reduces and hence the expected number of packets dropped also. As the value of service rate increases, packets are served quickly and hence the expectation of packet being dropped also decreases. So, both the graphs of expected number of packets being dropped and expected probability of packets being dropped is almost uniform through out the arrival rate and time. Similar logic holds true for expected queue length, average waiting time and system throughput. These results are very different from that of UDP connection and hence the packets dropped for smaller values of service rate for UDP connection is almost 2.5 times, system throughput for UDP is amost 1.5 times that of TCP connection.

Varying Bandwidth, fixed Arrival Rate and Service Rate:

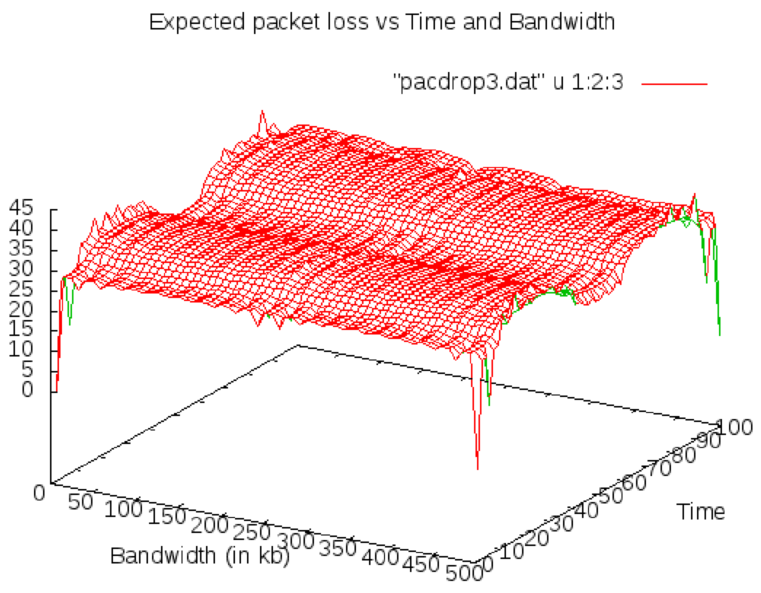


Figure 3.45: Expected number of packets dropped in presence of Attacker Configuration varying bandwidth

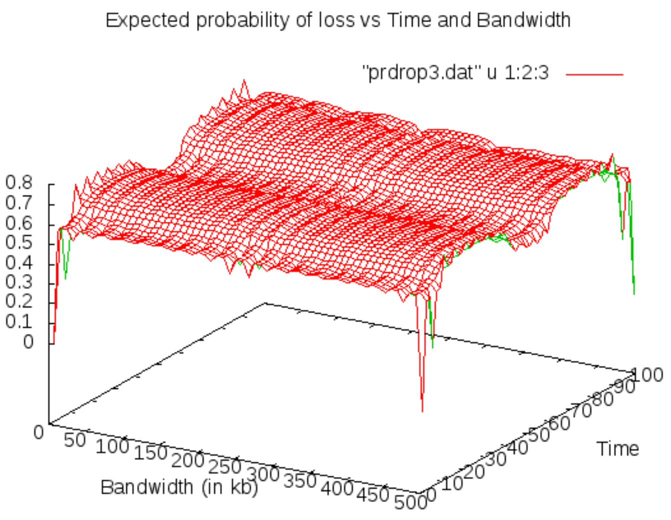


Figure 3.46: Expected probability of packets dropped in presence of Attacker Configuration varying bandwidth

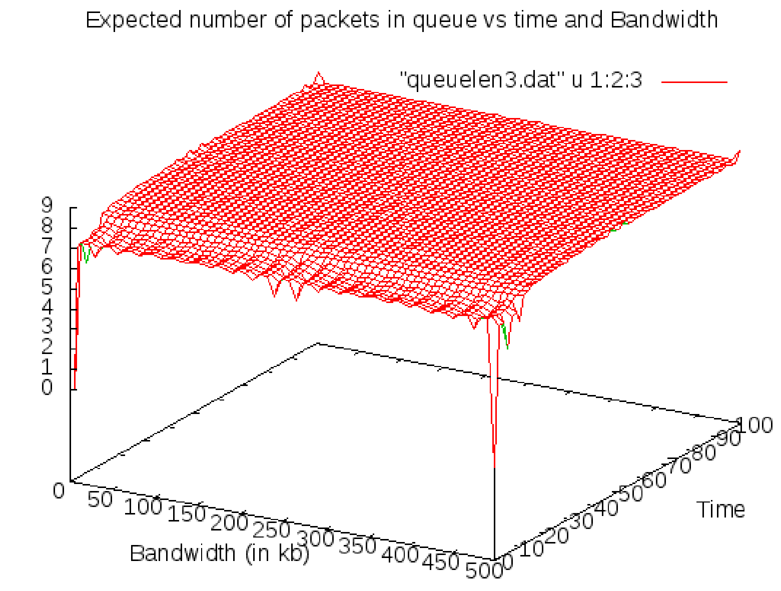


Figure 3.47: Expected number of packets in queue in presence of Attacker Configuration varying bandwidth

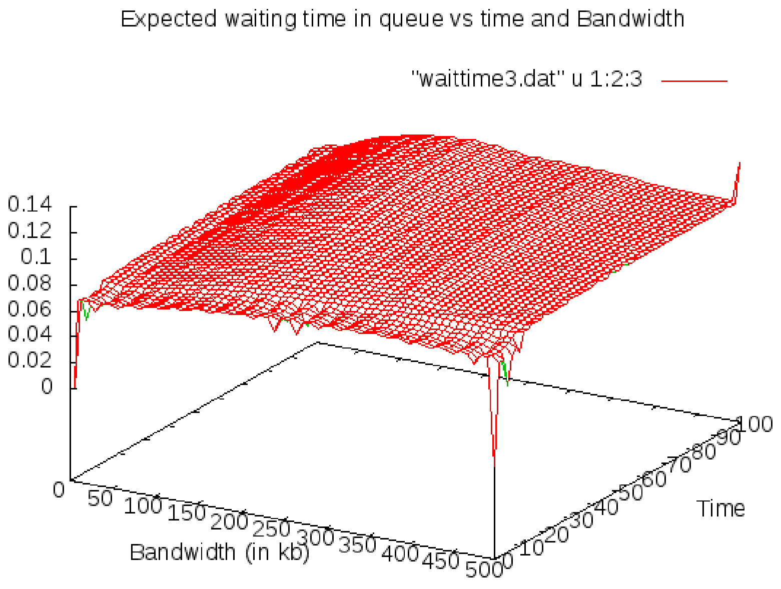


Figure 3.48: Expected waiting time in presence of Attacker Configuration varying bandwidth

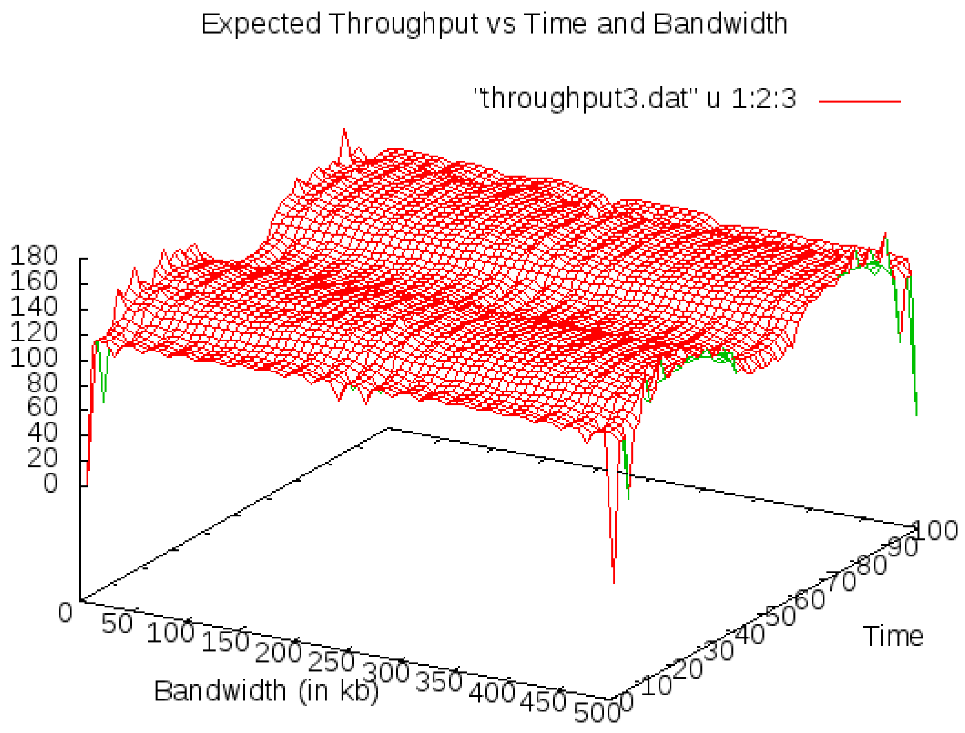


Figure 3.49: Expected Throughput in presence of Attacker Configuration varying Bandwidth

For smaller values of link Bandwidth, packet drop occurs very rapidly. Due to presence of control mechanisms of TCP, packets send to the server reduces and hence the expected number of packets dropped also. As the value of Bandwidth increases, packets are served quickly which results in sending large number of packets due to TCP mechanism. So, both the graphs of expected number of packets being dropped and expected probability of packets being dropped is almost uniform through out the arrival rate. Now at the time when attacker is switched on, more packets are generataed and the expected number of packet being dropped increases. Similar logic holds true for expected queue length, average waiting time and system throughput. Compared to UDP, expected number of packets being dropped is almost half and system throughput is almost one-fourth for the TCP.

Bibliography

[1] T. K. Pantelis Kammas and Y. Stamatiou, "Queuing theory based models for studying intrusion evolution and elimination in computer networks," *Journal of Information Assurance and Security 4 200-208*, 2009.

[2] A. Aissani, "Queueing Analysis for Networks Under DoS Attack," 2009.

[3] S. G. Neetu Singh and P. Chaudhuri, "Denial of Service Attack: Analysis of Network Traffic Anomaly Using Queing Theory," *Journal of Computer Science & Engineering*, vol. 1, no. 2, 2010.

[4] S. Singh and M. Gyanchandani, "Analysis of Botnet Behavior Using Queuing Theory," *International Journal of Computer Science & Communication*, vol. 1, no. 2, 2010.

[5] http://www-sop.inria.fr/members/Eitan.Altman/COURS-NS/n3.pdf.

[6] http://www.mathcs.emory.edu/~cheung/Courses/558-old/Syllabus/90-NS/trace.html.

[7] http://zetcode.com/lang/tcl/procedures/.

[8] http://nile.wpi.edu/NS/simple_ns.html.

[9] http://csis.bits-pilani.ac.in/faculty/murali/resources/tutorials/ns2.htm.