Comparative Analysis
of
Low Rate Denial of Service Attack in MANETs

Thesis submitted in partial fulfilment of the requirements for the award of degree of

Master of Engineering
in
Software Engineering

Submitted By
Arvind Sharma
(Roll No. 801131006)

Under the supervision of
Dr. Neeraj Kumar
Assistant Professor

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004
July 2013

Comparative Analysis
of
Low Rate Denial of Service Attack in MANETs

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering
in
Software Engineering

Submitted By
Arvind Sharma
(Roll No. 801131006)

Under the supervision of

Dr. Neeraj Kumar

Assistant Professor

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

THAPAR UNIVERSITY

PATIALA – 147004

July 2013

CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled, "***Comparative Analysis of Low Rate Denial of Service Attack in MANETs***", in partial fulfillment of the requirements for the award of degree of Master of Engineering in ***Software Engineering*** Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of ***Dr. Neeraj Kumar*** and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

Signature:

Arvind Sharma

(801131006)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

Dr. Neeraj Kumar

Assistant Professor

Computer Science and Engineering Department

Countersigned by

(Dr. Maninder Singh)
Head
Computer Science and Engineering Department
Thapar University
Patiala

(Dr. S. K. Mohapatra)
Dean (Academic Affairs)
Thapar University
Patiala

ACKNOWLEDGMENT

No volume of words is enough to express my gratitude towards my guide, ***Dr. Neeraj Kumar****, Assistant Professor, Computer Science and Engineering Department, Thapar University, who have been very concerned and have supervised the work presented in this thesis report. He has helped me to explore this vast field in an organized manner and provided me with all the ideas on how to work towards a research oriented venture.*

I am also thankful to ***Dr. Maninder Singh****, Head of Department, CSED and* ***Mr. Karun Verma****, P.G. Coordinator, for the motivation and inspiration that triggered me for the thesis work.*

I would also like to thank the staff members and my colleagues who were always there in the need of the hour and provided with all the help and facilities, which I required, for the completion of my thesis.

Most importantly, I would like to thank my ***parents****,* ***friends*** *and the* ***Almighty*** *for showing me the right direction out of the blue, to help me stay calm in the oddest of the times and keep moving even at times when there was no hope.*

Arvind Sharma

(801131006)

Abstract

Mobile MANETs (MANET) is example of wireless mobile communication. Network consists of mobile nodes that are multi-hop, self-configured connected by wireless links. Each node is free to move randomly and change their positions arbitrarily. So, the topology of network may change frequently. In MANETs dynamic mobile nodes are stationed in such a manner that communication between nodes does not rely on any existing network infrastructure. These networks are highly vulnerable to various security threats. One of major thread to current networks security is Denial of service (DoS) attacks. In this type of attacks, a single attacker or group of attacker try to gain access to network in term of interrupting legitimate user to serve by an application running on a mobile node.

This thesis focuses on detection and mitigation of flood type attack that results DoS attack in wireless network. There is new flavour of Dos attack, also known as Low Rate DoS attack. A victim node fails to detect this attack due to very low average rate of attack. In this attack, attacker sends Low Rate network traffic periodically to target node. So far, there is not any solution to counter Low Rate DoS attack without degrading the performance of Transmission Control Protocol (TCP).

A node based solution to mitigate the effect of Low Rate DoS attack discussed in this work. As per follow approach includes Random Early Detection (RED) variants Robust-RED (RRED) for controlling congestion control at network layer and TCP variants Selective Acknowledgement (SACK) at transport layer are used to detect mitigate effect of Low Rate DoS. Also, analyse the result these variants during attack. The solution provides rapid detection and mitigation procedure during attack.

Table of Contents

List of Figures

List of Tables

Abbreviation

ABR	Associativity Based Routing
ACK	Acknowledgement
AQM	Active Queue Management
AODV	Ad-hoc On Demand Distance Vector
ARED	Adaptive-Random Early Detection
CBR	Continuous Bit Rate
CBRP	Cluster based Routing Protocols
CGSR	Cluster head Gateway Switch Routing Protocol
CXCC	Cooperative Cross Layer Congestion Control
DoS	Denial of Service
DSR	Dynamic Source Routing
DSDV	Destination Sequenced Distance Vector
FRED	Flow Random Detection
FSR	Fisheye State Routing
GSR	Global State Routing
HAWK	Halting Anomalies with Weighted Choking
HSR	Hierarchical State Routing
IETF	Internet Engineering Task Force
IP	Internet Protocol
MANET	Mobile Ad-hoc Network
MSS	Maximum Segment Size

NS-2	Network Simulator
OSLR	Optimized Link State Routing
RED	Random Early Detection
RERR	Route Error
RED-PD	RED Preferential Dropping
RRED	Robust -Random Early Detection
RREP	Route Reply
RREQ	Route Request
RTO	Retransmission Time Out
RTT	Round Trip Time
SACK	Selective Acknowledgement
SAP	Shrew Attack Protection
SRED	Stabilized Random Early Detection
SSR	Signal Stability Routing
SYN	Synchronize
TCP	Transmission Control Protocol
TORA	Temporally Ordered Routing Algorithm Protocol
UDP	User Datagram Protocol
VOIP	Voice over Internet Protocol
WLAN	Wireless Local Area Network
WRP	Wireless Routing Protocol
ZHLS	Zone-based Hierarchical Link State Routing Protocol
ZRP	Zone routing protocol

Chapter 1
Introduction

1.1 Background

The field of Wireless networks has experienced rapid growth since 1970s. In fact, the combination of radio communications and computer networks were first introduced by the University of Hawaii in 1971 in an experimental network named ALOHANET.

This was the first Wireless Local Area Network (WLAN) that offered star topology based bidirectional communications [1]. During the 80s, the technology was drastically improved. At the end of the 90s, wireless networks made great revolution and reached a peak due to the constant growth of the Internet.

Figure1.1: A Simple Wireless network [1]

Mobile Ad-hoc networks (MANETs) have been widely researched during last few years, gathering lots of attention due to rapid increase in mobile devices. Today's world of dynamic changing technology of communication networks, MANETs play a vital role in wireless communication. MANETs are collection of wireless mobile nodes that acts as dynamic network without use of fix infrastructure and centralized control to authorise other entities in network. MANET comprises of mobile nodes that

cooperate with each other using wireless connections to route both data and control packets within the wireless network [5].

Unlike a wired network, nodes in an ad hoc network can free to move in random and arbitrary direction, so frequent changes in topology. These networks are self-configuring network and nodes within MANETs provide a peer-level multi-hopping routing service because each node acts as a router [2]. Also, source to destination communication may require routing information via several intermediate nodes to route a packet to the destination node due to limited transmission range of a node. Each mobile node that communicates with other node via radio wave and can communicate directly to those nodes that is in transmission range of each other. Each participating node in MANETs is independent and makes routing decision like route request, route selection, route update and making new communication link with their neighbours as well as serving old established. However, all network functions are based on the nodes mutual effort. A simple example of MANETs is shown below.

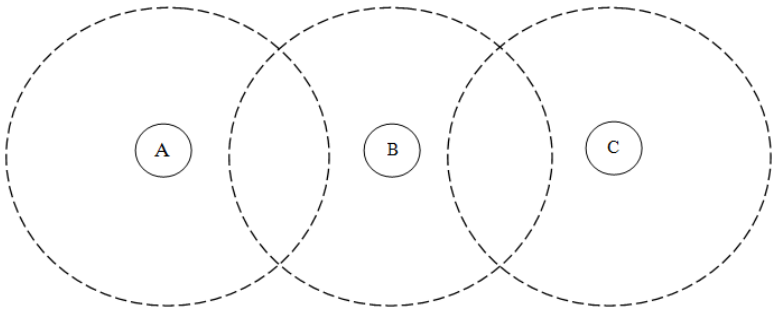

Figure 1.2: A simple MANETs with 3 participating nodes

In figure 1.2, node A wants to communicate to node C. However, node C is not in the direct transmission range of node A. So, node A and C must discover route through node B in order to communicate with each other.

However, there is no dependency on infrastructure that makes it robust and low-cost.

A MANETs has many benefits, such as and adaptability to highly variable characteristics, namely power, transmission conditions, traffic distribution variations, and load balancing. However, those benefits come with many challenges. New algorithms, protocols have to be designed and developed to create a truly flexible and decentralized network. The system may operate in isolation, or may have gateways to and interface with a fixed network.

1.2 Characteristics of MANETs

A MANET is an autonomous system of mobile nodes. MANETs node are equipped with wireless transmitters and receivers using antennas which may be highly-directional (point to point), unidirectional (broadcast), possibly steerable, or some combination thereof. The characteristics of these networks are summarized as follow [3].

- **Dynamic topology**

  MANETs are autonomous system of mobile nodes. Mobile nodes can move arbitrary in any direction thus network topology may change randomly and arbitrarily at unpredictable time intervals.

- **Energy-constrained operation**

  Unlike wired network, Wireless links have significantly lower link capacity. In addition, channel capacity is often much less than a radio's maximum transmission rate so that the realized throughput of wireless communications after accounting for the effects of multiple access, fading, noise, and interference conditions, etc.

- **Bandwidth constrained, variable capacity link**

  Some or Each of mobile nodes in MANETs relies on battery that has limited power to operate continuously. Basically, optimization of energy conservation may be the important system design criteria for these nodes.

A wireless connectivity in the form of a random, multi-hop exists between the nodes at a given point in time, depending on the nodes positions and their transmitter and receiver coverage patterns, transmission power levels and co-channel interference levels.

1.3 Advantages of MANETs

Some of the applications of MANETs are as follows.

- Tactical networks
- Emergency services
- Commercial operation
- Education
- Entertainment

1.4 Routing in MANETs

Routing is the act of moving information across an inter-network from a source node to a destination node. Along the way, at least one intermediate node typically is encountered. It's also referred to carry out two main tasks, first is to determine optimal path over and second is to send the packets through this optimal path from source to destination. The routing algorithm is the part of the network layer software responsible for deciding which output line an incoming packet should be transmitted on, i.e. what should be the next intermediate node for the packet. Routing protocols use metrics to evaluate what path will be the best for a packet to travel from source to destination node. A metric is a standard of measurement; such as hop count, reliability, delay etc; that is used by routing algorithms to determine the optimal path to a destination node in inter-network.

To aid the process of path determination, routing algorithms initialize and maintain routing tables, which contain route information. Route information varies depending on the routing algorithm used. Routing algorithms fill routing tables with a variety of information like hop count, source-destination logical and physical addresses etc. Mainly Destination address/Next hop associations tell a router that a particular destination can be reached optimally by sending the packet to a particular node representing the "next hop" on the way to the final destination. When a router receives an incoming packet, it checks the destination address and attempts to associate this address with a next hop.

1.4.1 MANETs protocol desirable properties

There are some properties required for routing protocol.

- **Distributed operations**

  The ad-hoc routing protocol should be distributed in nature. In MANETs mobile nodes are free to move randomly and there is not any centralise control of nodes; any time nodes can join/leave the network easily.

- **Unidirectional link support**

  In MANETs, nodes have limited transmission range and limited bandwidth of radio channel. So, links between nodes should be bidirectional to utilise the radio channel and also improves the routing protocol performance.

- **Demand based operation**

  In MANETs mobile nodes are independent entity that helps in cooperation for data transfer between source and destination while it is just an intermediate node. So, it is necessary to minimize the use of network resources and control overhead in MANETs. So, the ad-hoc routing protocol should be reactive. This means that when a source wants to send data to destination node, protocol will perform operation rather than periodically broadcast control information.

- **Security**

  Ad-hoc routing protocols are vulnerable to various kind of attacks because it works in radio environment. So, there is some sort of security measure to prevent these types of attacks. There are several mechanisms like authentication, encryption, and trust based routing, threshold cryptography etc. being used for prevention and mitigation of security attacks.

- **Loop free**

  Ad-hoc routing protocol should be loop-free to improve the overall performance of the network. It also gives guarantee that route should be looping free to avoid any waste of channel bandwidth and recourse utilization.

- **Power conservation**

  In MANETs, mobile nodes do not rely on fix infrastructure. So, there is not fix power supply for the mobile nodes. The mobile nodes in MANETs are laptop, mobile, PDAs that works on battery power. So, some sort of mechanism required to save battery consumption. Therefore, it is important that routing protocol has support for sleep mode.

- **Quality of service support**

  It is necessary for ad-hoc routing protocol to support some sort of quality of service to incorporate in routing protocol. There are some quality of service parameter like throughput, end to end delay, network load etc that are being used to understand real time traffic in network.

However, many routing protocol have been proposed for MANETs till now. None of proposed routing protocol from MANETs has all these properties because these protocols are still under development and are being extended with more functionality. The main objective of routing protocol is still to find and maintain route between source and destination nodes.

1.4.2 Classification of MANETs routing protocol

Classification of MANETs routing protocol can be done based many way. In most of the cased this can be done based on the network topology and routing protocol strategy. The routing protocol can be categorised as bellow.

There are some special limitation and properties of MANETs such as limited bandwidth and power, highly dynamic topology, high error rates etc. Moreover, in MANET all nodes are mobile and can be connected dynamically in an arbitrary manner when it compared to infrastructure based networks.

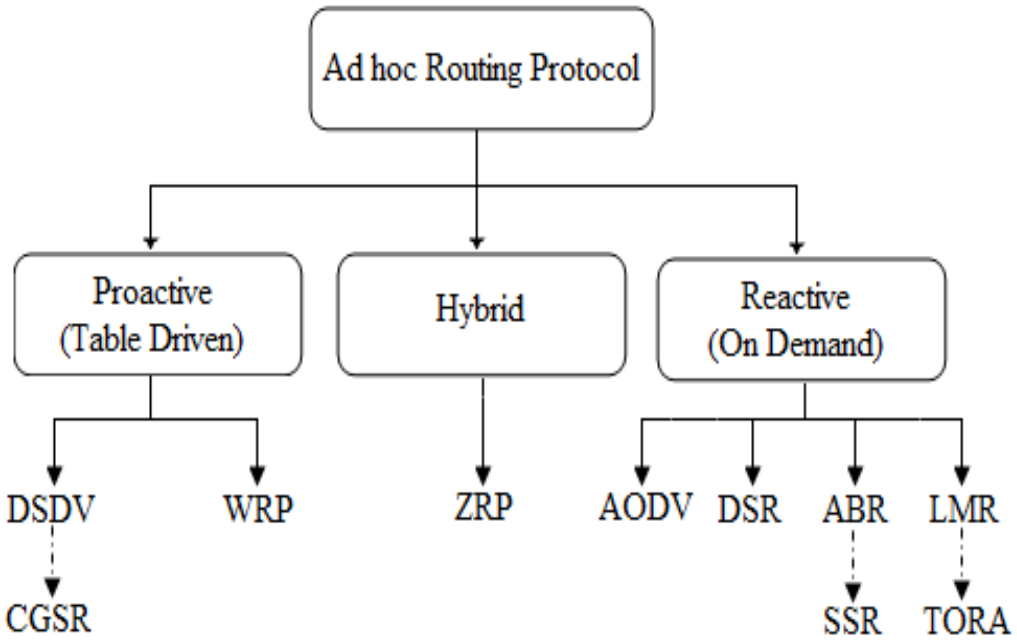

Figure 1.3: Classification of MANETs Routing Protocol [5]

Nodes of MANET act as router and take part in discovery and maintenance to establish a reliable route between source and destination node. Therefore, routing protocols for wired networks cannot be directly used in wireless networks and many protocols have been developed for MANETs. These routing protocols are divided into two categories based on management of routing tables. These categories are Table Driven Routing Protocols (proactive) and On-Demand Routing Protocols (reactive).

1.4.2.1. Proactive (Table Driven) Routing Protocol

Proactive routing protocol [5] means nodes periodically registers changes in the topology and updates routing information. Each node has a routing table that must be keeping up-to-date. Every node propagates the update messages to the network when

there is any change in network topology changes so that nodes can maintain reliable routing table. These protocols have the advantage that there is little latency since routes are already available. Also every node in the network has some sort information about network topology. Every node in network sends periodically update about the network topology that increases bandwidth overhead. Also, Periodic update for route tables keeps the nodes awake and quickly exhausts their batteries. However, there may be chance of many redundant route entries to the specific destination needlessly take place in the routing tables due to periodic updates.

Destination-Sequenced Distance Vector Routing Protocol (DSDV), Wireless Routing Protocol (WRP), Global State Routing (GSR), Fisheye State Routing (FSR), Hierarchical State Routing (HSR), Zone-based Hierarchical Link State Routing Protocol (ZHLS) and Cluster head Gateway Switch Routing Protocol (CGSR) are Table Driven Routing Protocols.

1.4.2.2. Reactive (On-Demand) Routing Protocols

Reactive routing protocol [5] is an opposite approach to proactive protocol. In this protocol, Routes are first discovered on demand, when data needs to be transmitted to a node where no route has yet been discovered. When a node wants to sends data to the destination node, it first checks the destination route in route cache. If destination route available in cache, it immediately sends data to destination node. If require route is not available in route cache then propagates the route request packet to its neighbours. When neighbouring nodes of the source node receive the broadcasted request packet, they forward the packet to their immediate neighbours and this action is happen until the destination is found. Afterward, the destination node sends a replay packet to the source node in the shortest path. The route remains in the route tables of the nodes through shortest path until the route is no longer needed. The major advantage of on demand routing is that it saves bandwidth because it limits the routing overhead. The disadvantage is the latency at the beginning of transmission to nodes when no route, have yet been discovered.

Ad-Hoc On-Demand Distance Vector Routing (AODV), Dynamic Source Routing Protocol (DSR), Cluster based Routing Protocols (CBRP), Temporally Ordered Routing Algorithm Protocol (TORA), Associativity Based Routing (ABR), Signal Stability Routing (SSR) are On-Demand Routing protocols.

1.4.2.3. Hybrid Routing Protocol

Hybrid routing protocol combines the advantage of proactive and reactive protocol. Route is being initially established with some proactive routing protocol and then serves the demand from additionally activated nodes through reactive flooding. Each proactive and reactive protocol work best in different scenario. Proactive protocols are restricted to small domain, whereas reactive routing protocols are used for locating nodes that are from outside the domain [4]. Example of hybrid routing protocols are: Zone routing protocol (ZRP).

Table 1.1: Classification of MANETs Routing Protocol

MANETs Routing Protocol	
Table Driven Routing Protocol	**On Demand Routing Protocol**
Destination-Sequenced Distance Vector Routing Protocol (DSDV)	Ad-Hoc On-Demand Distance Vector Routing (AODV)
Wireless Routing Protocol (WRP)	Cluster based Routing Protocols (CBRP)
Global State Routing (GSR)	Dynamic Source Routing Protocol (DSR)
Hierarchical State Routing (HSR)	Associativity Based Routing (ABR)
Zone-based Hierarchical Link State Routing Protocol (ZHLS)	Signal Stability Routing (SSR**)**
Clusterhead Gateway Switch Routing Protocol (CGSR)	Temporally Ordered Routing Algorithm (TORA)

1.5. Security Issues for MANETs

This security issues are to be considered in MANETS because of its characteristics like vulnerability of channels, nodes, absence of infrastructure and dynamically changing topology. In MANETs, a mobile node has some limitations with respect to bandwidth, computing power, and battery that can lead to application-specific trade-offs between security and resource consumption of the mobile device. However, to do this intermediate node achieves no benefits. So there may be a possibility that some nodes refuse to forward packets and thereby decrease the efficiency of the network in term of throughput and packet delivery ratio.

However, malicious behaviour of the nodes is selfishness. A selfish node may try to save their resources like battery power and computation ability by not participation in network operation like data forwarding [6]. With increasing the number of malicious

nodes, there may be result of making a non-collaboration environment between other nodes and also affected the network performance they do not correctly process the network packets. Therefore, ensure that everything is correctly working in the network to support overall security and know how an insider malicious node is able to attack the wireless MANETs.

Vulnerabilities of operating systems and upper layer applications that belong to user programs such as databases, browsers or client-server applications are not considered as a security issue for MANETs. In MANETs, there are different type of attacks that belongs to different network layers such as physical layer, medium access control layer, network layer and transport layer.

1.5.1 Attacks in MANET's

Any attack on ad hoc networks can be categorized as active and passive attacks. In an active attack, the misbehaving node actively disturbs the normal operation of the network with attempts to alter or destroy the data being exchanged in the network. It can also be classified into two categories, external attacks and internal attacks. External attacks are carried out by nodes that do not belong to the network. These attacks can be prevented by using standard security mechanisms such as encryption techniques and firewalls. Internal attacks are carried out by compromised nodes that are actually part of the network. Since the attackers are already part of the network as authorized nodes, internal attacks are more severe and difficult to detect when compared to external attacks.

In passive attack the malicious entity only listens to the traffic without disturbing proper operation of the network. An attacker is also able to interpret the data gathered through snooping to violet confidentiality requirement. Detection of passive attacks is very difficult since the operation of the network itself does not get affected. In MANETs, the common attack in MANETs is discussed below [7].

1.5.1.1 Passive Eavesdropping

An attacker can listen to any wireless network to know what is going on in the network. It first listens to control messages to infer the network topology to understand how nodes are located or are communicating with another. Therefore, it can gather intelligent information about the network before attacking. It may also listen to the information that is transmitted using encryption although it should be confidential belonging to upper layer applications.

Eavesdropping is also a threat to location privacy. An unauthorized node can notice a wireless network that exists within a geographical area, just by detecting radio signals. To combat this, traffic engineering techniques have been developed.

1.5.1.2 Grayhole Attack (Routing Misbehaviour)

Grayhole attack is an active type of attack, which lead to dropping of messages. Attacking node first agrees to forward packets and then fails to do so. Initially the node behaves correctly and replays true RREP messages to nodes that initiate RREQ message. This way, it takes over the sending packets. Afterwards, the node just drops the packets to launch DoS attack. If neighbouring nodes that try to send packets over attacking nodes lose the connection to destination then they may want to discover a route again, broadcasting RREQ messages. Attacking node establishes a route, sending RREP messages. This process goes on until malicious node succeeds its aim (e.g. network resource consumption, battery consumption). This attack is known as routing misbehaviour.

1.5.1.3 Blackhole Attack

In this attack, an attacker uses the routing protocol to advertise itself as having the shortest path to the node whose packets it wants to intercept. An attacker listen the requests for routes in a flooding based protocol. When the attacker receives a request for a route to the destination node, it creates a reply consisting of an extremely short route. If the malicious reply reaches the initiating node before the reply from the actual node, a fake route gets created. Once the malicious device has been able to insert itself between the communicating nodes, it is able to do anything with the packets passing between them. To carry out a black hole attack, malicious node waits for neighbouring nodes to send RREQ messages. When the malicious node receives an RREQ message, without checking its routing table, immediately sends a false RREP message giving a route to destination over itself, assigning a high sequence number to settle in the routing table of the victim node, before other nodes send a true one. Therefore requesting nodes assume that route discovery process is completed and ignore other RREP messages and begin to send packets over malicious node.

Malicious node attacks all RREQ messages this way and takes over all routes. Therefore all packets are sent to a point when they are not forwarding anywhere. To succeed a black hole attack, malicious node should be positioned at the centre of the wireless network. The difference of Black Hole Attacks [7] compared to Gray Hole Attacks is that malicious node never send true control messages initially. Gray hole

attacks against one or two nodes in the network to isolate them, where as black hole attack affects the whole network.

1.5.1.4 Wormhole Attack

In wormhole attack [8], a malicious node receives packets at one location in the network and tunnels them to another location in the network, where these packets are resent into the network. This tunnel between two colluding attackers is referred to as a wormhole. It could be established through wired link between two colluding attackers or through a single long-range wireless link. In this form of attack the attacker may create a wormhole even for packets not addressed to itself because of broadcast nature of the radio channel. This attack can even prevent routes more than two hops long from being discovered. Though no harm is done if the wormhole is used properly for efficient relaying of packets, it puts the attacker in a powerful position compared to other nodes in the network, which the attacker could use in a manner that could compromise the security of the network.

Possible ways for the attacker to then exploit the wormhole include discarding rather than forwarding all data packets, thereby creating a permanent DoS attack or selectively discarding or modifying certain data packets. So, if proper mechanisms are not employed to protect the network from wormhole attacks, most of the existing routing protocols for ad hoc wireless networks may fail to find valid routes.

1.5.1.5 Impersonation

Due to lack of authentication in MANETs, only MAC or IP addresses uniquely identify hosts. These addresses are not adequate to authenticate the sender node. Therefore non-repudiation is not provided for MANETs protocol. MAC and IP spoofing are the simplest methods to pretend as another node or hide in the network. Malicious nodes achieve impersonation only by changing the source IP address in the control message. Another reason for impersonation is to persuade nodes to change their routing tables pretending to be a friendly node, such as attacks against routing table. Malicious node performs this attack by combining spoofing and dropping attacks. Physically, it must be placed as the only node within the range for destination, in the middle of the route or victim node must be prevented from receiving any other route information to the destination. Malicious node may also change the routing tables of the victim node to redirect its packets, using attacks against the routing table. At this point, malicious node waits for an RREQ message to the destination node from source node. When source node sends an RREQ message, malicious node drops

the RREQ and replays a spoofed RREP message to source node as if it is coming from the destination node. At the same time, malicious node sends a RREQ message to the destination node and drops the RREP message from the destination node. By doing this; malicious node manages to establish a route both to the source and the destination node and attacker controls the communication between the source and destination. If the communication is encrypted or entails an authentication as to MAC or IP address, malicious node can easily get the up layer communication.

1.5.1.6 Routing Table Attacks

Every node has its own routing table to find other nodes easily in the network. At the same time, this routing table draws the network topology for each node for a period (max. 3 seconds, duration of ACTIVE_ROUTE_TIMEOUT constant value of AODV protocol). If malicious node attacks against this table, attacked nodes do not find any route to other nodes that it wants to connect. This attack is always performed by fabricating a new control message. Therefore it is also named fabricating attack. There are many attacks against routing tables. Each one is done by fabricating false Control messages. For example; to attempt a black hole attack, malicious node first invades into the routing table of the victim, sending false RREP message. Malicious node also spreads false RERR messages to the network so that valid working links are marked as broken. Another attack type against the routing table is to attempt to create lots of route entries for non-existent nodes, using RREQ messages. As a result, routing table of the attacked node is full and does not have enough entry to create a new one. This attack type is known as routing table overflow.

Attacks against the routing tables also affect the network integrity, changing the network topology established in the routing tables. Incorrect control messages are disseminated quickly in the network due to route discovery process and influence the network integrity in a wide area.

1.5.1.7 DoS Attacks

In this type of attack, an attacker attempts to prevent legitimate and authorized users from the services offered by the network. A DoS attack can be carried out in many ways. The classic way is to flood packets in the network so that services provided be intermediate node is no longer available to other participating nodes in the network, as a result of which the network no longer operating in the manner it was designed to operate. This may lead to a failure in the delivery of guaranteed services to the end users. Due to the unique characteristics of MANETs, there exist many more ways to

launch a DoS attack in such a network. DoS attacks can be launched against any layer in the network protocol stack [10]. On the physical and MAC layers, an attacker could employ jamming signals which disrupt the on-going transmissions on the wireless channel. On the network layer, an attacker could take part in the routing process and exploit the routing protocol to disrupt the normal functioning of the network. For example, an adversary node could participate in a session but simply drop a certain number of packets, which may lead to degradation in the Quality of Service being offered by the network. On the higher layers, an attacker could bring down critical services by Low Rate DoS attack. Some of the DoS attacks are described below:

Jamming: In this form of attack, the attacker initially keeps monitoring the wireless medium in order to determine the frequency at which the destination node is receiving signals from the sender. It then transmits signals on that frequency so that error-free reception at the receiver is hindered. Frequency hopping spread spectrum (FHSS) and direct sequence spread spectrum (DSSS) are two commonly used techniques that overcome jamming attack.

SYN flooding: In this form of attack, a malicious node sends a large amount of SYN packets to a victim node, spoofing the return addresses of the SYN packets. The SYN-ACK packets are sent out from the victim right after it receives the SYN packets from the attacker and then the victim waits for the response of ACK packet. Without any response of ACK packets, the half-open data structure remains in the victim node. If the victim node stores these half-opened connections in a fixed-size table while it awaits the acknowledgement of the three-way handshake, all of these pending connections could overflow the buffer, and the victim node would not be able to accept any other legitimate attempts to open a connection. Normally there is a time-out associated with a pending connection, so the half-open connections will eventually expire and the victim node will recover. However, malicious nodes can simply continue sending packets that request new connections faster than the expiration of pending connections.

Distributed DoS: Distributed DoS is more severe form of DoS attack because in this attack several attackers that are distributed throughout the network try to prevent legitimate users from accessing the services offered by the network.

1.6 Motivation

MANETs are vulnerable to various security attacks due to absence of centralised control and dynamic changing topology. One of these attacks is the Low Rate TCP DoS attack [13]. In this type of DoS attack, the attacker is responsible for exploiting the standard TCP congestion control mechanism by overflows the bottleneck link periodically. As a result, the re-transmitted packets will be dropped so that sender's packets will not reach the receiver. However, attacker's overall traffic rate can potentially be significantly lower. This property makes Low Rate DoS attack more difficult to detect by counter mechanism.

Low Rate DoS attack is being simulated in MNETs and evaluated its effect in the network. For simulation purpose, NS-2 (Network Simulator version 2) is used that consists of the collection of all network protocols to simulate many of the existing network topologies. To simulate Low Rate TCP DoS, a node that belongs to network acts as attacker in the hierarchical based scenario in MANETs.

Having implemented a new security mechanism which simulates the Low Rate Targeted DoS attack, implementation performed tests on different parameter to compare the network performance with and without attack in the MANETs. As a result, the throughput in the network would deteriorate considerably in the presence of Low Rate attack.

Afterwards, thesis implements solutions to mitigate the effects of Low Rate attack in the MANETs. Thesis implements the mitigation technique like TCP variants at transport layer and RED variants data link layer for network congestion control into the NS-2 and analyse the results.

1.7 Thesis Outline

This thesis consists of 6 chapters and these chapters are organised as follow:

Chapter 1 explains the introduction MANETs, Routing protocol, security issue and various attacks in MANETs. Chapter 2 describes the survey of Low Rate DoS attack, overview of TCP, congestion control, timer and recent work. Chapter 3 introduce about problem statement and objectives. Chapter 4 describes the simulation study required to carry out this work. Chapter 5 shows the result of simulation and chapter 6 include conclusion and future scope.

Chapter 2
State of Art

2.1 Literature Survey

Numerous studies have been carried out in past to study the performance of TCP and UDP over wired and wireless networks. Many detection and protection scheme have been proposed against Low Rate DoS attacks.

X. Luo et al. [10] investigate the performance of TCP flows is affected by DOS attacks under the Drop Tail and various Active Queue Management (AQM) schemes. This study attempts to test the effectiveness of Drop Tail and AQM mechanism in between the flood based DoS attack and pulse based DoS based on both analytical and simulation scenario. Moreover, the Drop Tail surprisingly outperforms the RED-like AQMs when the router is under Pulse based DoS attack; whereas the RED-like AQMs perform better under a severe flood based DoS attack. On the other hand, the Adaptive Virtual Queue algorithm can retain a higher TCP throughput during Pulse based DoS attacks as compared with the RED-like AQMs. However, RED-like schemes are not provide complete mitigation against pulse based attack.

S. Savage et al. [11] proposed a solution for detecting and preventing optimistic acknowledgement by adding extra field in TCP header. An instant request and reply header are added in TCP. When sender sends new data it generates a random number and appends with existing number in nonce field. When receiver replies he has to put this cumulative nonce received in new reply field of TCP header. When acknowledgement is received by the receiver which matches it with expected nonce if it matches then acknowledgement is accepted otherwise it is rejected and attack is detected. The problem with this solution is that extra fields are to be added in TCP header. Therefore change in TCP protocol in each system needs to be done which is not an appropriate task.

C. Zhang et al. [12] describes an adaptive queuing management called Robust Random early detection (RRED) algorithm to improve the TCP throughput against low rate DoS attacks. The basic idea behind the RRED is to detect and filter out attack packets before a normal RED algorithm is applied to incoming flows. However, router based

RRED scheme gives better performance under low rate attack but it cannot completely mitigate the effect of attack.

A. Kuzmanovic et al. [13] proposed two different solutions. In First solution they proposed the solution of randomizing Retransmission Time Out (RTO) value. As minimum RTO value of 1 second was experimentally found to be best [9]. But there is some issue with this solution. First issue is to initially change the TCP implementation of every server. Second issue is randomizing minimum RTO value cause degradation of performance when there is not any attack. Third one, attacker can also adjust attack period according to range of minimum RTO value. Second solution is applying fair queue scheduling. Fair queue scheduling gives equal chances to packet of each flow in the queue of router to be processed. Therefore the attack will not be able to effect to such a great extent.

M.M. Morshed et al. [14] have studied the performance of TCP variants such as Reno, New-Reno, Vegas and Tahoe under AODV, DSR, and DSDV and Optimized Link State Routing (OLSR) routing protocols.

H. Sun et al. [15] proposed a scheme which has to be applied in all routers. Boundary router sees traffic flow going out from it towards target server. If the low rate TCP targeted attack is discovered by analyzing the traffic, router has to find which of the input port the attack is being carried out. Suppose, there is the port from which the attack is coming. Then the affected router will push back the detection to the upstream router connected to the input to that port. This detection mechanism causes two things to happen. Firstly it pushes the detection of attack as close to the source as possible. Secondly it is able to minimize the damage to the legitimate TCP flows. The above mentioned is procedure for the case when attack is done by single attacker. If attack is done in distributed fashion then attack will be coming from multiple input ports therefore above procedure will fail.

This solution has deficiency that the detection and prevention mechanism has to be applied in multiple routers between source and destination or to all routers in the world, as source can be anywhere in the world. Also their detection scheme required lot of steps that is sampling, feature extraction, noise filtering and pattern matching which is not scalable in high speed network links.

G. Yang et al. [16] describe effectiveness of the low rate DoS attack and RTO randomization in network. This study attempts to test the scalability of the attack on a real system by using an analytical model on the throughput of randomised TCP, and also implement and measure the effectiveness of RTO randomization. Simulation showed that the attack can be very effective and randomization significantly improves, but cannot solve the problem entirely.

Y. Kwok et al. [17] proposed a scheme called HAWK (Halting Anomalies with Weighted Choking). This scheme has been applied in bottleneck routers. Two time scales were maintained one for finding burst length and one for attack period. When queue of router crosses some threshold value, Flows are monitored and stored in table if that flow is found to be come at higher rate. Then it is seen that this particular flow is periodic and matching signature of low rate attack flow over a period of time that is generally chosen as 5 seconds and called as HAWK window size. If flow is classified as attack flow it is stored in a table and whenever packet of this flow comes it is dropped. The deficiency of this solution is that it cannot detect the attack when it was carried out in distributed manner.

A. Shevatker et al. [18] proposed a solution which is deployed in boundary router which connects network to the Internet. In the edge router, each TCP packet has to pass through three modules. First module is called flow classifier which classifies packet into particular flow by examining the IP address and port number of source and destination. Then it is passed through object module. This module keeps information of active flows necessary to detect attacks. First it notes the arrival time of each packet and observes the time difference of packets for each flow. If characteristics of shrew attack are found that is sudden burst then quiet period is observed with certain periodicity, then the burst length (L) is calculated of the peak rate and time period (T) at which repetition takes place. If $L \geq$ RTT of nearly all flows and T is approximately equal to minimum RTO that is one second, this flow is classified as attack and filters this flow out in the next module which is filter module. Deficiency of this solution is that it has to maintain information for all TCP flows, which requires more processing and memory resources in each router. Secondly it cannot detect attack when it is done in distributed fashion.

X. Luo et al. [19] proposed a scheme in which they studied incoming TCP flow traffic and outgoing traffic. They placed there detection scheme at boundary router. If attack is going on then there will be decrease in rate of incoming flows due to attack and

there will also be decrease in rate of outgoing ACK flow from victim server. But due to attack flow the incoming rate of TCP flow may not fall significantly than in normal case. They studied incoming and outgoing signals at regular interval of time and attack has been detected if signal was inhibiting a wavelet form. The deficiency of the solution is that it is unclear how to find optimal set of parameters that are sensitive enough to detect distributed Low Rate TCP-targeted attack while keeping low false positive rate.

Chia-Wei Chang et al. [21] proposed a solution called SAP (Shrew Attack Protection), for defending against a Shrew attack, which has to be applied in routers and which requires small modification in dropping techniques of packets in the queue of router. This solution is basically based on classifying TCP flows based on port to which packets are destined and then tagging packet of those flows which are arriving faster than average dropping rate of all flows. Packet drop rate of every class flow is constantly observed. Deficiency of this solution is that it lessens the intensity of attack but cannot detect and prevent the attack. Secondly it is less effective if attacker's flow is of same class as of most of TCP flow destined to that server. For example, a server is running service in port 80. All TCP flows are destined with destination port as port 80. If attacker also attacks with bogus port 80 destined packets then there is no effect of this scheme.

Rob Sherwood et al. [22] proposed a solution for stopping optimistic acknowledgement. In this solution server intentionally drops a segment of TCP flow and then observe, if attack is going on, attacker will not send duplicate acknowledgement otherwise he will send duplicate acknowledgement. In this way attack is detected. Deficiency of this solution is that it decreases the throughput of genuine TCP flow as they have to slow down their rate when server intentionally drops a segment.

2.2 Low Rate DoS:

This class of DoS attacks is unique in nature. As the name implies that the attack rate is very low to signal a confirmed congestion or attack. This attack is mainly targeted at TCP services and hence the attack is termed as "Low Rate Denial of Service". This unique nature makes this attack remain is being unidentified by detection systems designed to detect the DoS or Distributed DoS attacks.

These attacks do not apply abundant packets to flood the network. Instead, it exploits the working mechanism of TCP timers thus affect the throughput of a system. These low-rate attacks are specially crafted to generate packets in very minimal quantity after different period of time [13]. Thus the attacking packets can easily cover up with the legitimate packets and difficult to detect from the Anti-DoS traffic monitoring systems. The attacks carried out this way exploiting the TCP timers are also called "Shrew Attacks".

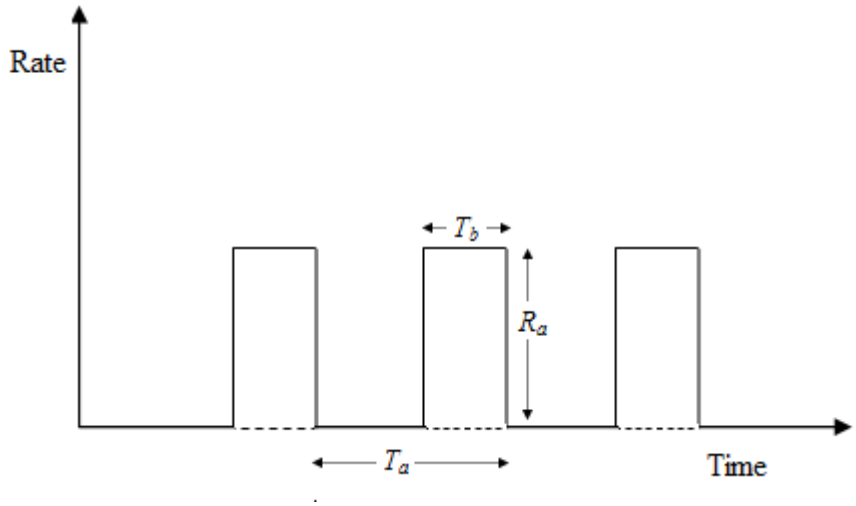

Figure 2.1: Low Rate DoS Attack

During the congestion control mechanism in TCP, congestion window gradually reduced its size until network is free from congestion. So, congestion in network makes sender's rate at lower level and also the reason for potential reduction in throughput. When network is congested, there may be chance to drop packets in the network. For drop packets, the data is sent again after the RTO expires. When the congestion is more in the network, the RTO timer is doubled after which the packets are retransmitted. Thus during a low rate attack, an attacker is able to calculate this RTO time and sends low rate attacking packets traffic to create packet collision at router and there may result of packet loss. So, the attacker can push the TCP into waiting state. Hence, there is no need for flooding the network with packets, but only send low rate packets traffic when the RTO timer is about to expire and push it again into waiting. This type of attack can escape the traffic monitors due to its low traffic rate and is a serious challenge for the security experts.

2.3 TCP Overview

Transport layer forms the backbone for transferring large amount of data across the network. This layer is also responsible for connection establishment and connection maintenance services. It is also used to control the flow of data in both directions between the source and the destination and provide continuous reliable data delivery services over unreliable communication networks. This layer multiplexes data from the application layer to lower layers and de-multiplexes the data from the network layer to the higher layers. The Transport layer protocols are the TCP and the User Datagram Protocol (UDP). TCP is a connection oriented and reliable protocol while UDP is connectionless protocol. In TCP Applications, a formal connection is being established before transferring data over the link. At the end of the data transfers, when an application (client/server) does not want to transfer data then it performs a connection closure operation to indicate the end of data transfer. The TCP protocol is also responsible for validate the correct delivery of data from the client to the server and vice versa. Sometimes, network conditions affect data transmission as there may be lost or error in data that are transmitted over the link. TCP adds support to detect errors or lost data, and to start retransmission mechanism until the data are correctly and completely reached to destination. At transport layer, TCP breaks the upper layer incoming data into IP-sized pieces called segments and these segments pass to network layer where these segments are being handled by Internet Protocol (IP) protocol to send across network in form of packets.

However, there are many algorithms implement in TCP protocol to flow in network. On the way, there is not any formal connection establishment and closure procedure while using UDP as the transport layer protocol. The protocol does not guarantee the delivery of packets and hence the reliability of this protocol is not guaranteed. However, one can observe that the processing overhead is significantly less using UDP than that using TCP. Each protocol selected based on the application and advantages. However, TCP is not particularly suitable for real-time applications such as Voice over IP, scientific experiment, health care etc. Since, TCP sometimes induce relatively long delays in order of seconds while waiting for out-of-order messages or retransmissions of lost segments because it is optimized for accurate delivery rather than timely delivery for such applications, protocols like the Real-time application running over the User Datagram Protocol (UDP) are usually recommended instead.

This following discussion will provides the basic information required to understand the working of the TCP flow control mechanism and timers and will also highlight how the schemes are being exploited by the attackers to create successful denial of service attacks

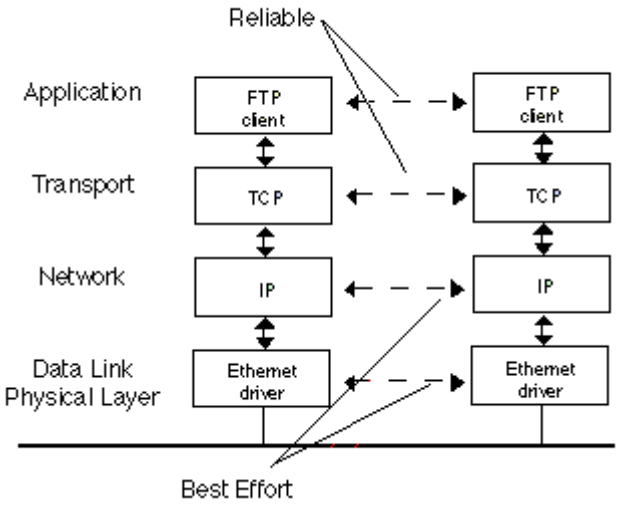

Figure 2.2: TCP/IP Protocol Architecture.

2.3.1 TCP Congestion Control Algorithms

At transport layer protocol, one of the main advantages of using the TCP over the UDP is its reliability. It means that the layer is responsible for confirm the packet delivery to the estimation. This is achieved by using various data flow control techniques at transport layer. In this section we will be highlighting the congestion control mechanism provided by the TCP. The TCP congestion control mechanism uses various algorithms to implement the data flow control. The standard TCP implementations today can be found in [23]. The four algorithms, Slow Start, Congestion Avoidance, Fast Retransmit and Fast Recovery are described below.

2.3.1.1 Slow Start

Slow Start, is a mechanism used by the sender to control the transmission rate also known as sender-based flow control. This is achieved through the rate of acknowledgements from the receiver. In other words, the sender data transmission rate is being determined by the acknowledgements returned from the receiver determines. At the beginning of TCP connection, the Slow Start algorithm initializes a congestion window size to one segment, which is the maximum segment size (MSS)

initialized by the receiver during the connection establishment phase. The congestion window increases by one segment for each acknowledgement returned from receiver. Thus, the sender can transmit the minimum of the congestion window and the advertised window of the receiver, which is simply called the transmission window.

Slow Start is actually progress exponentially when the network is not congested and network response time is good. For example, the first successful transmission and acknowledgement of a TCP segment increases the window to two segments. After successful transmission of these two segments, the window size is being increased to four segments when acknowledgements received by sender. Then eight segments, then sixteen segments and increase exponentially up to the maximum window size advertised by the receiver or until congestion finally does occur. At some point the congestion window may become too large to put huge traffic in the network or network conditions may change such that packets may be dropped. Packets lost will trigger a timeout at the sender side. When this happens, the sender goes into congestion avoidance mode as described in the next section.

2.3.1.2 Congestion Avoidance

Initially, a TCP connection is used the Slow Start algorithm. However, there may be in Slow Start phase, network is forced to drop one or more packets due to congestion. If this happens, Congestion Avoidance is used to slow the transmission rate. However, Slow Start is used in conjunction with Congestion Avoidance as the means to get the data transfer going again so it doesn't slow down and stay slow. In the Congestion Avoidance algorithm, a retransmission timer expiring or the reception of duplicate ACKs can signal the sender that congestion has occurred in the network. The sender immediately puts its transmission window to one half of the current window size that is the minimum of the congestion window and the receiver's advertised window size. However, current window size at least two segments. Moreover, the congestion window is reset to one segment if congestion was indicated by a timeout. So, it automatically puts the sender into Slow Start mode [26].

If congestion was indicated by duplicate ACKs, the Fast Retransmit and Fast Recovery algorithms are invoked as discussed bellow. As data is received during Congestion Avoidance, the congestion window is increased. However, Slow Start is only used up to the halfway point where congestion originally occurred. This halfway point was recorded earlier as the new transmission window. After this halfway point, the congestion window is increased by one segment for all segments in the

transmission window that are acknowledged. This mechanism will force the sender to more slowly grow its transmission rate, as it will approach the point where congestion had previously been detected.

2.3.1.3 Fast Retransmit

When a duplicate ACK is received, the sender does not know if it is because a TCP segment was lost or simply that a segment was delayed and received out of order at the receiver. If the receiver can re-order segments, it should not be long before the receiver sends the latest expected acknowledgement. Typically no more than one or two duplicate ACKs should be received when simple out of order conditions exist. If however more than two duplicate ACKs are received by the sender, it is a strong indication that at least one segment has been lost. The TCP sender will assume enough time has lapsed for all segments to be properly re-ordered by the fact that the receiver had enough time to send three duplicate ACKs. When three or more duplicate ACKs are received, the sender does not even wait for a retransmission timer to expire before retransmitting the segment as indicated by the position of the duplicate ACK in the byte stream.

2.3.1.4 Fast Recovery

Since the Fast Retransmit algorithm is used when duplicate ACKs are being received, the TCP sender has implicit knowledge that there is data still flowing to the receiver. Why? The reason is because duplicate ACKs can only be generated when a segment is received. This is a strong indication that serious network congestion may not exist and that the lost segment was a rare event. So instead of reducing the flow of data abruptly by going all the way into Slow Start, the sender only enters Congestion Avoidance mode. Rather than start at a window of one segment as in Slow Start mode, the sender resumes transmission with a larger window, incrementing as if in Congestion Avoidance mode.

This allows for higher throughput under the condition of only moderate congestion [25]. To summarize this section of the thesis, figure 2 below depicts what a typical TCP data transfer phase using TCP congestion control might look like. We can notice the period of exponential window size increase, linear increase and drop-off. Each of these scenarios depicts the sender’s response to implicit or explicit signals it receives about network conditions. Congestion is a state of severe delay in data transfer caused by overload of packets at the intermediate nodes. TCP congestion control mechanism makes sure of end-to-end data delivery without causing congestion route to

destination. TCP makes use of sequence numbering, congestion window and retransmission timer mechanisms to achieve sequential, congestion less and reliable services.

2.3.2 TCP Timers

Each TCP segment sent over network towards destination, the sender expects to receive an acknowledgement within some period of time otherwise an error in the form of a timer expiring signals that that something is wrong. A sender's implicit knowledge of network conditions may be achieved through the use of a timer. Somewhere in the end-to-end path of a TCP connection a segment can be lost along the way. Often this is due to congestion in network where excess packets must be dropped. TCP not only correct for this situation, but it can also learn something about network conditions from it.

Whenever TCP transmits a segment the sender starts a timer which keeps track of how long it takes for an acknowledgment for that segment to return. This timer is known as the retransmission timer. If an acknowledgement is returned before the timer expires, which by default is often initialized to 1 second, the timer is reset with no consequence. If however an acknowledgement for the segment does not return within the timeout period, the sender would retransmit the segment and double the retransmission timer value for each consecutive timeout up to a maximum of about 64 seconds [23]. If there are serious network problems, segments may take a few minutes to be successfully transmitted before the sender eventually times out and generates an error to the sending application.

Fundamentally, the timeout and retransmission strategy of TCP is the measurement of the round-trip time between two communicating TCP hosts. The round-trip time may vary during the TCP connection as network traffic patterns fluctuate and as routes become available or unavailable. TCP keeps track of when data is sent and at what time acknowledgements covering those sent bytes are returned. TCP uses this information to calculate an estimate of round trip time. As packets are sent and acknowledged, TCP adjusts its round-trip time estimate and uses this information to come up with a reasonable timeout value for packets sent. If acknowledgements return quickly, the round-trip time is short and the retransmission timer is thus set to a lower value. This allows TCP to quickly retransmit data when network response time

is good, alleviating the need for a long delay between the occasional lost segment. The converse is also true. TCP does not retransmit data too quickly during times when network response time is long. If a TCP data segment is lost in the network, a receiver will never even know it was once sent. However, the sender is waiting for an acknowledgement for that segment to return. In one case, if an acknowledgement does not return, the sender's retransmission timer expires which causes a retransmission of the segment. If however the sender had sent at least one additional segment after the one that was lost and that later segment is received correctly, the receiver does not send an acknowledgement for the later, out of order segment. The receiver cannot acknowledgement out of order data; it must acknowledge the last contiguous byte it has received in the byte stream prior to the lost segment. In this case, the receiver will send an acknowledgement indicating the last contiguous byte it has received. If that last contiguous byte was already acknowledged, we call this a duplicate ACK. The reception of duplicate ACKs can implicitly tell the sender that a segment may have been lost or delayed. The sender knows this because the receiver only generates a duplicate ACK when it receives other, out of order segments. In fact, the Fast Retransmit algorithm described later uses duplicate ACKs as a way of speeding up the retransmission process.

2.3.3 TCP Variants

TCP includes eleven variants like Tahoe, TCP/Full, TCP/Asym, Reno, Reno/Asym, Newreno, Newreno/Asym, SACK, FACK, Vegas as source and five- other variants as destination, implemented in Network Simulator (NS-2). Each has unique characteristics and functionality. Most common TCP variants discussed below:

2.3.3.1 TCP-Tahoe

The congestion window is initially increased exponentially till slow-start threshold is reached if there is a normal non-congested traffic condition. After the congestion window has reached slow-start threshold, the congestion window is increased linearly until congestion is detected. TCP-Tahoe treats a timeout as indication of congestion. A non-arrival of acknowledgement before retransmission timeout indicates timer has been expired. At this time, TCP-Tahoe tries to lessen congestion by initiating slow start mechanism by setting the congestion window to one and sets congestion window half of slow start threshold. Congestion window is increased exponentially till slow-

start threshold is reached, then increased linearly until a packet loss is encountered. Figure shows the TCP-Tahoe congestion control mechanism. One of the major issues with TCP-Tahoe is that it significantly reduces the available bandwidth as the congestion window is reduced to one and has to be rebuilt with every acknowledgement received. Also, it takes considerably long time to detect a packet loss.

2.3.3.2 TCP-Reno

It uses the same basic principle of TCP-Tahoe with introduces some intelligence to detect packet loss earlier and not to reduce the congestion window drastically. Basically, TCP-Reno tries to solve the issues present in TCP-Tahoe. TCP-Reno expects immediate acknowledgements for packets sent. A duplicate packet from the receiver indicates that the next packet in line has reached. If a considerable number of duplicate acknowledgements are received, the sender presumes that the packet has been lost. TCP-Reno uses this logic of duplicate acknowledgements (dupacks) to trigger Fast Retransmit. After receiving a predetermined number dupacks, usually set to three, TCP Reno takes it as a sign of segment lost and retransmits the packet immediately and enters Fast Recovery. In Fast Recovery, slow-start threshold and congestion window is set to half the value of current congestion window. For each subsequent dupack, the congestion window is increased by one and a new segment transmitted if the new value permits it. TCP-Reno remains in fast recovery phase until it receives acknowledgements for all the segments in the transmit window when it entered congestion, after which it enters congestion avoidance. TCP-Reno and TCP-Tahoe treat a timeout in the same fashion by triggering slow-start. TCP-Reno overcomes the problems of TCP-Tahoe but cannot detect multiple packet loss within the same window and sometimes reduces the congestion window more than once for packet losses occurred within the same transmit window.

2.3.3.3 TCP New-Reno

The issues present in TCP-Reno are being overcome in TCP New-Reno by modifying the fast recovery mechanism [25]. When a fresh ACK is received during fast recovery, TCP New-Reno handles them as below:

- If receiver ACKs all the segments which were outstanding when TCP New-Reno entered Fast Recovery, then it exits Fast Recovery and sets *cwnd* and *ssthresh* and continues congestion avoidance as in Tahoe.
- If the ACK is partial then it deduces that the next segment in line was lost and retransmits that segment and sets dupacks to 0. It exits Fast Recovery when all

the data segments in the window are acknowledged, until then every progress in sequence number generates a redundant packet retransmission which is instantly acknowledged. TCP New Reno suffers from the fact that it takes one RTT to detect each packet loss. When the ACK for the first retransmitted segment is received, only then can we deduce which other segment was lost.

2.3.3.4 TCP-Vegas

Unlike other TCP variants like TCP-Tahoe, TCP-Reno and TCP New-Reno, TCP-Vegas do not wait for packet loss to happen before reducing the sending rate. Instead, it uses delay in packet arrival as congestion indication. Also, it is known as delay-based TCP. TCP-Vegas use a refined bandwidth estimation technique to perform congestion control. It calculates the available bandwidth in the network as the difference between actual data flow rate and the expected data flow rate. In situations where there is no congestion, these two rates should be essentially more or less the same. In case of congestion, the actual data flow rate would be less than expected data flow rate. The difference between the data flow rates are calculated using the round trip times as given below:

$$Diff = (Expected \mid Actual)\ baseRTT$$

Where *Expected* is the expected rate, *Actual* is the observed rate and *baseRTT* is the minimum round trip time. Based on the calculated *Diff*, TCP-Vegas sender updates the congestion window as follows:

$$Cwnd = \begin{cases} cwnd + 1, & if\ Diff < \propto \\ cwnd - 1, & if\ Diff > \beta \\ cwnd\ \ \ , & Otherwise \end{cases}$$

Where, $\propto$ and β are predefined thresholds.

2.3.3.5 TCP- SACK

Multiple packet losses from a window of data can have a catastrophic effect on TCP throughput. TCP uses a cumulative acknowledgment scheme in which received segments that are not at the left edge of the receive window are not acknowledged. This forces the sender to either wait a roundtrip time to find out about each lost packet, or to unnecessarily retransmit segments which have been correctly received. With the cumulative acknowledgment scheme, multiple dropped segments generally cause TCP to lose its ACK-based clock, reducing overall throughput.

SACK is a strategy which corrects this behaviour in the face of multiple dropped segments. With selective acknowledgments, the data receiver can inform the sender

about all segments that have arrived successfully, so the sender need retransmit only the segments that have actually been lost.

SACK mechanism was denned in [24]. With New-Reno optimization, TCP ends up retransmitting at most one dropped segment per round-trip time. SACK helps TCP recover faster by providing additional information about the state of congestion. SACK-permit option is an enabling initially that may be sent in a SYN segment to indicate that the SACK option may be used once the connection is established. The other one is the SACK option itself, which may be sent over an established connection once permission has been given by the SACK permission option. SACK options will be included in all ACKs that do not acknowledge the highest sequence number in the data receiver's queue. In this situation the network has lost or disordered data, so that the receiver holds non-contiguous blocks of data in its queue. Each non-contiguous block of data queued at the data receiver is defined in the SACK option by two 32-bit unsigned integers. The SACK option does not change the meaning of the "Acknowledgement Number" field. A SACK option that specifies "n" non contiguous blocks will have a length of "8*n+2" octets, so the 40 bytes available for TCP options would allow TCP to specify a maximum of 4 blocks. Of course, if other TCP options are introduced, they will compete for the 40 bytes, and the limit of 4 may be reduced further. We note that the receiver is permitted to discard data in its queue that has not been acknowledged to the data sender, even if the data has already been reported in a SACK option. This situation might happen if the receiver runs out of buffer space; hence, the sender will not discard the data reported in the SACK option until it gets an ACK for that data.

Several studies have been conducted on the performance of SACK. SACK was shown to perform better than TCP Tahoe and TCP Reno in [10]. Even though, SACK is more efficient than TCP Reno and TCP New-Reno, the throughput obtained by SACK is much less than optimal. SO, from the analysis of different TCP variants SACK performs better than other variants.

2.4 Active Queue Management

Congestion is an important issue which effects on TCP in network environment. To keep the stability of the whole network, congestion control algorithms have been extensively examined. Queue management method employed by the routers is one of the important issues in the congestion control study. The IETF proposed Active Queue Management (AQM) as a router-based mechanism for early detection of congestion inside the network. When there are too many coming packets contending for the limited shared resources, such as the queue buffer in the router and the outgoing bandwidth, congestion may happen in the data communication. During congestion, large amounts of packet experience delay or even be dropped due to the queue overflow. Congestion will also decrease efficiency and reliability of the whole network; furthermore, if at very high traffic, performance collapses completely and almost no packets are delivered.

2.4.1 Drop tail

It is a simple queue mechanism that is used by the routers that when packets should to be drop. In this mechanism each packet is treated identically and when queue filled to its maximum capacity the newly incoming packets are dropped until queue have sufficient space to accept incoming traffic. This mechanism uses first in first out policy. When a queue is filled, the router start to discard all extra packets thus was dropping the tail of mechanism. The loss of packets (datagram's) causes the sender to enter slow start which decreases the throughput and thus increases its congestion window.

2.4.2 RED

RED is a congestion avoidance queuing mechanism. The RED active queue management algorithm allows network operators to simultaneously achieve high throughput and low average delay particularly in high-speed transit networks [28].It is active queue management mechanism. It operates on the average queue size and drop packets on the basis of statistics information. If the buffer is empty all incoming packets are acknowledged. As the queue size increase the probability for discarding a packet also increase. When buffer is full probability becomes equal to 1 and all incoming packets are dropped.

RED is capable to evade global synchronization of TCP flows, preserve high throughput as well as a low delay and attains fairness over multiple TCP connections, etc. It is the most common mechanism to stop congestive collapses. When the queue in the router starts to fill then a small percentage of packets are discarded. This is deliberate to start TCP sources to decrease their window sizes and hence suffocate back the data rate. This can cause low rates of packet loss in Voice over IP streams. There have been reported incidences in which a series of routers apply RED at the same time, resulting in bursts of packet loss.

The RED gateways detect incipient congestion by computing the average queue size. RED computes the average queue length *q*avg using an exponentially weighted moving average [28].

$$q_{avg} = (1 - w_q)q_{avg} + w_q * q_{instantaneous}$$

In this equation, w_q defines weight for calculating average queue, min_{th} is the lower threshold below which no packet is dropped, max_{th} describes upper threshold above which all incoming packets are dropped. The packets are dropped based on the above q_{avg}. The averaging process smoothens out temporary traffic fluctuations and allows small bursts to pass through unharmed, dropping packets only during sustained overloads. RED maintains two queue thresholds: min_{th} and max_{th}. If q_{avg} exceeds max_{th}, all incoming packets are dropped, whereas if q_{avg} is less than min_{th} no packet is dropped. If q_{avg} lies between the two thresholds, packets are dropped with a probability p which increases linearly from 0 min_{th} to max_{th}. By dropping packets before the buffer is completely full, the RED gateway attempts to warn the TCP sources of incipient congestion.

2.4.2.1 Adaptive RED (ARED)

ARED algorithm infers whether to make RED more or less aggressive based on the observation of the average queue length [29]. If the average queue length oscillates around *min* threshold then early detection is too aggressive. On the other hand if the average queue length oscillates around *max* threshold then early detection is being too conservative. The algorithm changes the probability according to how aggressive it senses it has been discarding traffic.

2.4.2.2 Robust RED

RRED algorithm was proposed to improve the TCP throughput against DoS attacks, particularly Low Rate DoS attacks. The detailed description of RRED algorithm has

been given in [12]. Experiments have confirmed that the existing RED-like algorithms are notably vulnerable under Low Rate DoS attacks due to the oscillating TCP queue size caused by the attacks. RRED algorithm can significantly improve the performance of TCP under Low Rate DoS.

2.5 Recent work

The attack detection monitors in place today are all designed to detect high rate attacks, and hence the thresholds to confirm an attack are very high. As mentioned earlier, the solutions already in place to detect DoS attacks but cannot detect this low rate attack. RTO Randomization and router-based solution using queuing schemes are the only two solutions suggested so far in [13] and [16] to tackle the low rate attacks.

2.5.1 RTO Randomization

Randomizing the minimum value of RTO value for TCP is the solution proposed in [13]. It is known that the TCP does not perform a doubling of the RTO value after n=6, implying that once the RTO value touches 64, the RTO value is no longer updated as 2n, where n is the n^{th} consecutive timeout period. A random value between 2" and 2n+1 is chosen for this RTO. It is said that the attacker no longer can synchronize the attack. Since this scheme does not involve any attack detection activity, the RTO randomization has to be performed even when there is no attack. This is a fundamental drawback, which affects the TCP performance, while it is still vulnerable to a different rate attack.

2.5.2 Node based Solution

The second approach is the node-based scheme which uses adaptive queue management schemes to tackle the attack [13, 16]. This approach is based on the preferential dropping of DoS flow packets such as Robust Random Early Detection (FRED), RED with preferential dropping (RED-PD). In this approach, the attack packets are monitored and dropped with a probability, which is dependent on the sending rate. This approach fails when the TCP traffic rate to the queues reduces to very low values.

According to [13], the existing AQM schemes such as RED and RED-PD cannot detect burst lengths shorter than 300ms. If the attack is coordinated from different

locations, then the burst length at a particular bottleneck goes completely undetected, thus leading to complete low down of TCP connections in that region or network. The solutions suggested could only mitigate an attack but cannot eliminate them. Apart from these solutions, there is no complete solution, which can defend low rate attacks effectively.

In the following chapter, a node assisted approach to detect and mitigate the low rate attacks is proposed. As discussed earlier, alone an existing AQM schemes is not effective for detection and mitigation purpose. However, the combination of AQM and TCP congestion control variant like Reno, New-Reno, SACK, and VAGAs gives better detection and mitigation as compare to drop-tail AQM. The advantage of this solution is its scalability. Simulation results show that this approach can also detect low profile attacks effectively. Details of the approach and the architecture design are elaborated in the next chapter 3.

Chapter 3

Problem Statement and Objectives

In Chapter 2, the detailed survey about Low Rate denial of service attack has been presented. Chapter 3 discus about issues that have untouched in detailed survey.

3.1 Problem Statement

Thesis carried out detailed survey about Low Rate DoS and other counter mechanism for detection and mitigation. The unique characteristics of Low Rate DoS make it effective to degrade the performance of network. This work discuss about the issue related to router based solution. At data link layer, Active queue management scheme used to control congestion router as discussed in chapter 2. At transport layer, TCP variants used for congestion control. As per work survey, there is not any mechanism that completely mitigates the effect of Low Rate DoS. This work combines the RRED variant of AQM with most efficient TCP variant SACK as per survey in chapter 2. Also, analyse the performance of network with these variants.

3.2 Objective 1

Objective1 of this thesis is to implement the complete scenario of Low Rate DoS attack in MANETs. The scenario includes topology design, TCP application and Implementation of attacker module. Attacker module includes random start mechanism in deferent bust time. For different burst period, simulation is being carried out. For analysis of Low Rate DoS effect in MANETs, thesis used router based approach that is hierarchical network architecture in MANETs.

3.3 Objective 2

At data link layer, RRED gives batter network performance as compare to other AQM technique as discussed in chapter 2. At transport layer, SACK tacks advantage as compare to other congestion control mechanism. Second objective of this work is to implement both congestion control mechanism RRED, SACK in wireless scenario. Also, analysis the effect of congestion control technique separately and try to improve congestion control mechanism by taking advantage of SACK over RRED.

Chapter 4

Simulation Study

This chapter covers the simulation of low rate denial of service attack and mechanism that detect and mitigate them. Implementation covers the topology design, attack modelling under RED and SACK. For the implementation purpose, NS-2 simulator is used because it is easy available, open source tool and also is compatible with C++ programming language.

4.1 Network Simulators (Ns-2)

This section gives a brief introduction to the Ns-2 simulator [30]. The Ns-2 simulator has been around since 1989 and several institutions and societies has supported and contributed to its development as a variant of real network simulator for studying the dynamic behaviour of flow and congestion control schemes in wired and wireless networks. Ns-2 is an event driven simulator targeted especially at network research. It offers implementations of many famous and less famous protocols, traffic sources, etc. NS-2 also provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite) networks.

At the simulation layer NS uses OTcl (Object oriented Tool Command Language) programming language to interpret user simulation scripts [30]. OTcl language is in fact an object oriented extension of the Tcl Language. At the top layer, NS is an interpreter of Tcl scripts of the users. The Tcl language is fully compatible with the C++ programming language.

NS creates two main analysis reports simultaneously while OTcl script is being interpreted. One of them is NAM (Network Animator) object that shows the visual animation of the simulation. The other is the trace object that consists of the behaviour of all objects in the simulation.

NS project is normally distributed along with various packages (ns, nam, tcl, otcl etc.) named as “all-in-one package”, but they can also be found and downloaded separately. In this study, stable version 2.33 of ns all-in-one package is used and installed the package in the operation environment Red Hat Enterprise Linux 5. This work, “.tcl” files have written in text editor and analyzed the results of the “.tr” file.

4.2 Implementation

Implementation includes various parameters that have to configure first before start simulation.

4.2.1 Mobility Model

The mobility model describes the movement of nodes within the simulation area. The Random Waypoint mobility model is commonly used in most simulations. In mobility model the nodes move from one waypoint to the next with a randomly chosen speed (uniformly distributed between 0–20 m/s). A specific speed and duration is chosen for every transition. After the stipulated transition duration ends the node may pause for a specific duration of time before starting its transition towards the next waypoint. Nodes in the simulation set up move according to a model that is well known as the "random waypoint" model selects a rectangular field. Mobility models were created for the simulations using 31 nodes, maximum speed of 20 m/s, topology boundary of 1000 × 1000 and simulation time of 50 sec.

4.2.2 Simulation Set up

This network model will be generated with the help of network animator tool, after running TCL script considering the following simulation parameters as shown in table

Table 4.1: Simulation Parameters

Channel	Channel/Wireless
Interface	Wireless
Packet Size	512 Byte
Queue Length	50
No. of Nodes	31
Simulation Area	1000x1000
Simulation Time	50 Second
Mobility Model	Random Waypoint
Transmission Range	250m
Traffic Model	CBR
Bandwidth	2 mbps
Protocol	DSDV

4.2.3 Topological Design

Topological design of MANET presents in this section. A hierarchical architecture implemented to create wireless scenario. This architecture includes a root node and two clusters or two sub network. Each cluster includes 15 mobile nodes. The hierarchical architecture is also private addressing scheme. A pool of private address is being used for assigning private address to each node in each cluster. At the start of simulation each time an address is being picked up from address pool and assign to the current node. When, a cluster node wants to communicate with other node that resides in other cluster, all the traffic flows from root node. Inside cluster node can communicate directly without forwarding traffic to gateway.

Hierarchical architecture is being used in implementation because attacker node wants to attack a node that flow maximum traffic of the network. In this scenario, maximum network traffic flows from root node.

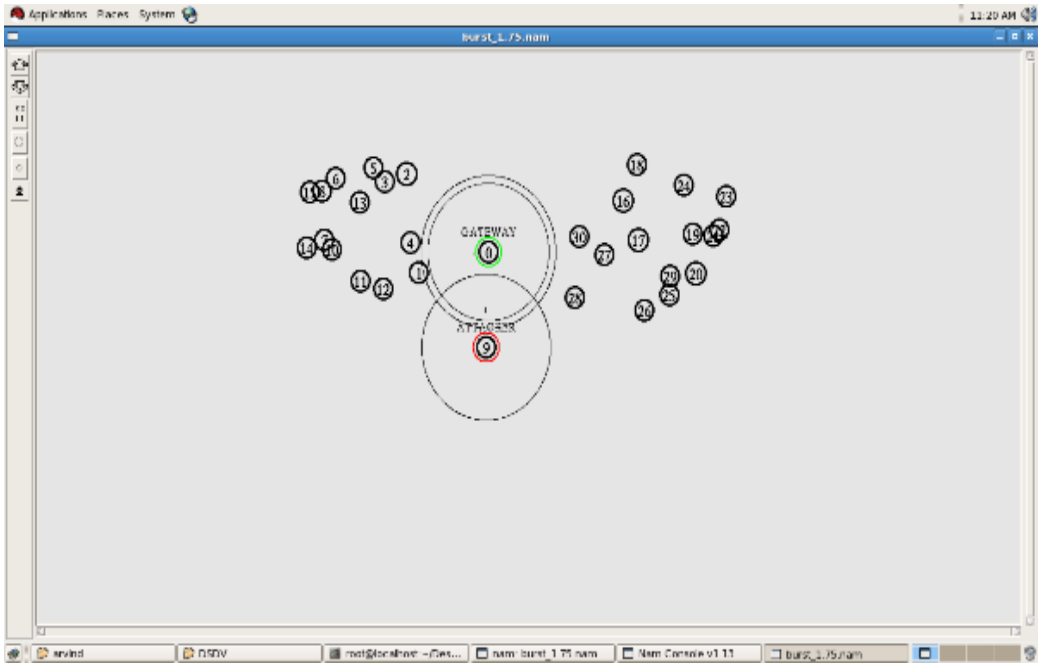

Figure 4.1: Network Topology

As Topology concern, there are only two data connection between clusters for experiment purpose, one from cluster 1 to 2 and another with 2 to 1.

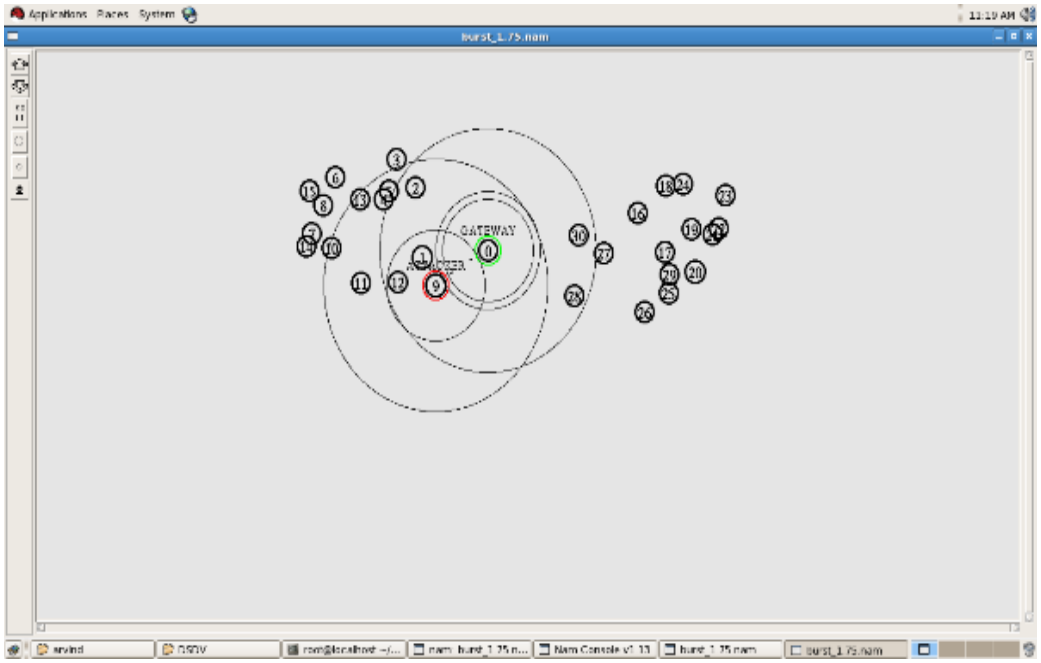

Figure 4.2: Network Topology with TCP Flow

4.2.4 Attack Modelling

In this thesis, before implementing any application take assumption of Table Driven protocol. Counter mechanism with RED variant required Table Driven Protocol to store network information at router. Low Rate attack is modelled by a square wave in

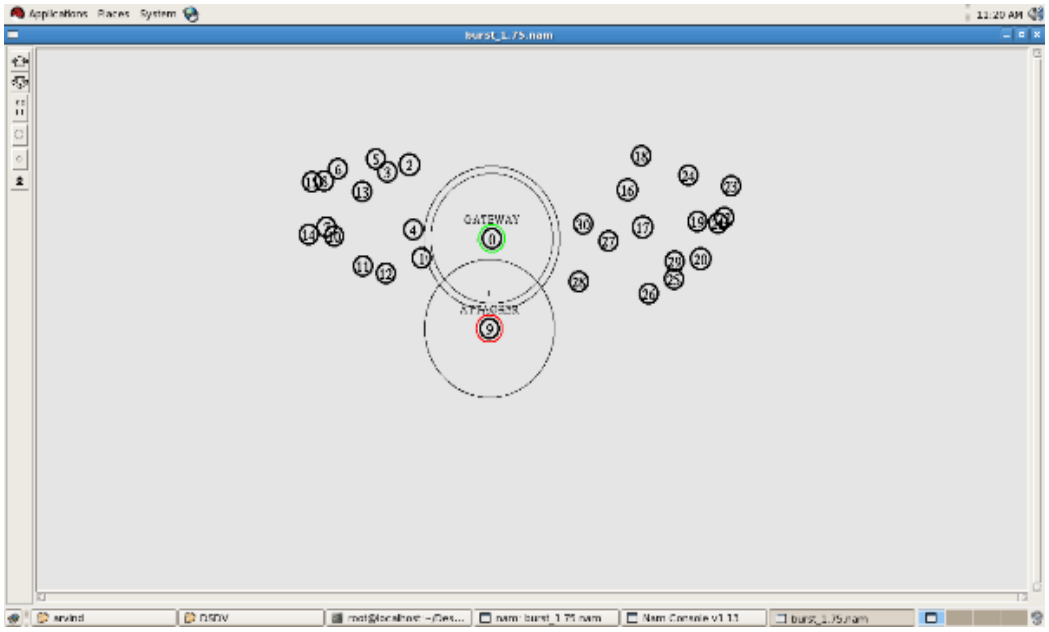

Figure 4.3: Low Rate TCP Flow from Attacker

which the attacker transmits bursts of duration L at rate R in a deterministic on-off pattern that has period T. When the rate R coupled with existing traffic becomes greater than the link capacity loss is incurred.

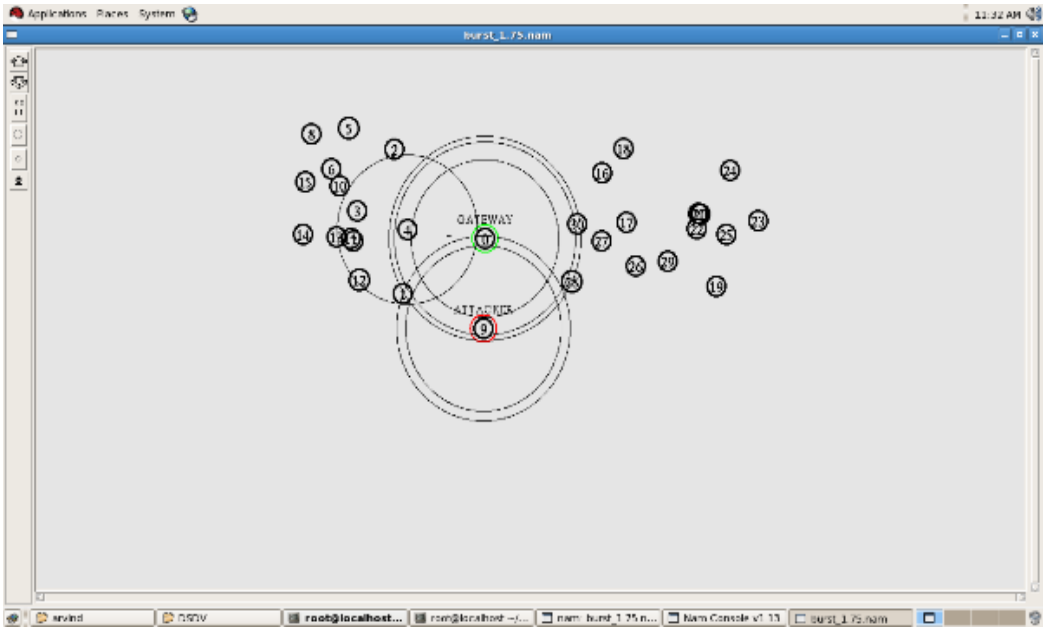

Figure 4.4: Congestion at Gateway

By setting the duration L to be more than the RTT of the flows and period T to be slightly more than minimum RTO value, TCP flows can be forced to repeatedly time out, thus obtaining virtually zero throughputs.

Chapter 5

Results and Discussion

5.1 Performance metric

There are different metrics are used to performance of network. Some of them discussed here. This chapter describes about result carried out based different parameter between Drop tail and RRED with SACK in DSDV Protocol.

Throughput

Throughput is the measure of how fast we can actually send through network. The number of packets delivered to the receiver provides the throughput of the network.

Average End-to-End Delay

It includes all possible delays caused by buffering during queuing at the interface queue, retransmission delays, and propagation and transfer times of data packets.

Routing Overhead

Defined as ratio of total number of routing and reputation related packets and total number of data packets

5.2 Results and Performance Analysis

Figure 5.1, shows throughput comparison between DSDV with Drop tail and DSDV

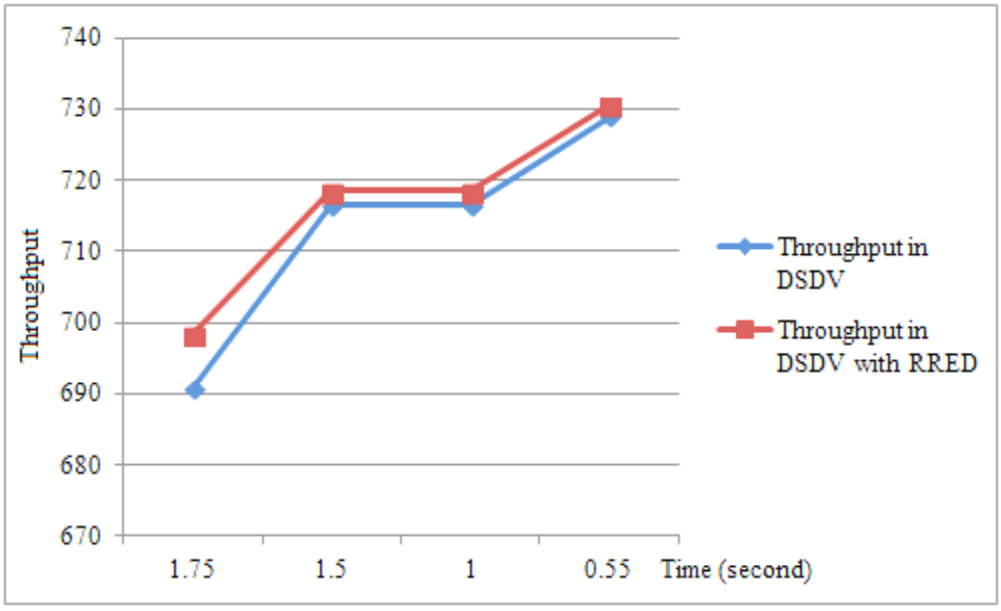

Figure 5.1: Throughput comparison between Drop tail and RRED

with RRED at different burst length of Attack. At burst period 1.75 RRED gives better performance as compare to Drop tail AQM. As burst period of attacks decreases means attacker tacks small time to bottleneck the capacity of channel at victim node, gradually increase in throughput. When burst period of attacker is 1.5 and 1 there is not any improvement in throughput. It slowly increases when attacker puts minimum traffic on the network.

Figure 5.2 shows performance of DSDV protocol with SACK and RRED variant. As the result seen DSDV with SACK and RRED combination gives better performance

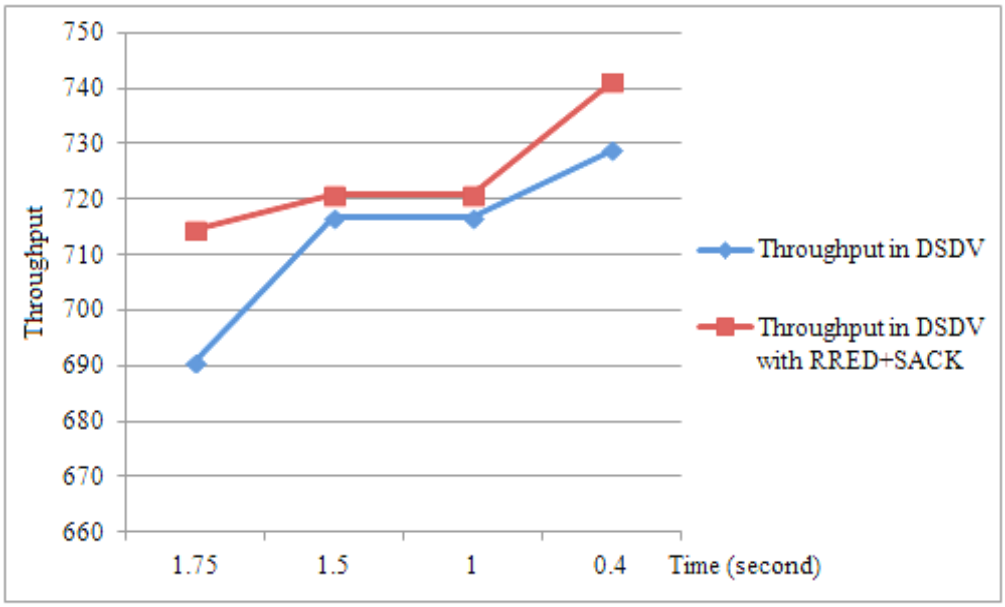

Figure 5.2: Throughput between Drop tail and RRED with SACK

at different burst length of attack. There is remarkable increase in throughput at burst length 1.75. However, at burst length 1.5 and 1, they not meet any improvement but when burst period is 0.4 second, slightly improvement is shown.

Figure 5.3 shows the comparative delay in DSDV with RRED. As the figure state that DSDV Protocol with Drop tail AQM shows slightly better result as compare to RRED AQM. Consider the result at bust length 1.5 and 1 both AQM mechanisms give same result. Initially RRED performance degrades but at bust length 0.4 sec remarkable improvement has been seen.

Figure 5.4, delay in DSDV with RRED and SACK is slightly more as compare to Drop tail DSDV because RRED with SACK suffer to deliver packets.

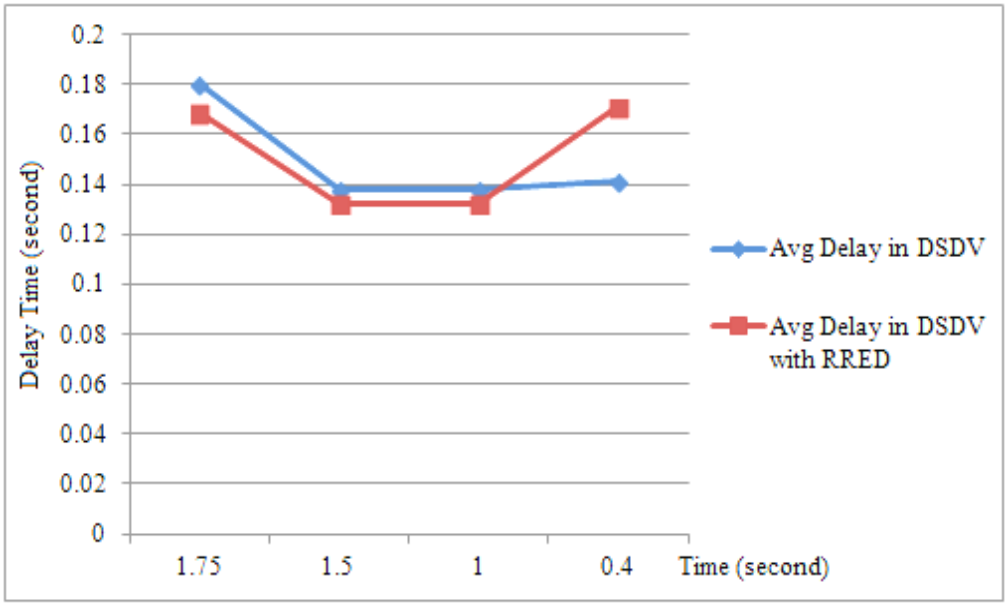

Figure 5.3: Delay between Drop tail and RRED

When network is congested due to the overflow of network traffic, control packets are injucted in the network to inform sender about congestion. RRED and SACK both uses control packets for sending network status to the end point

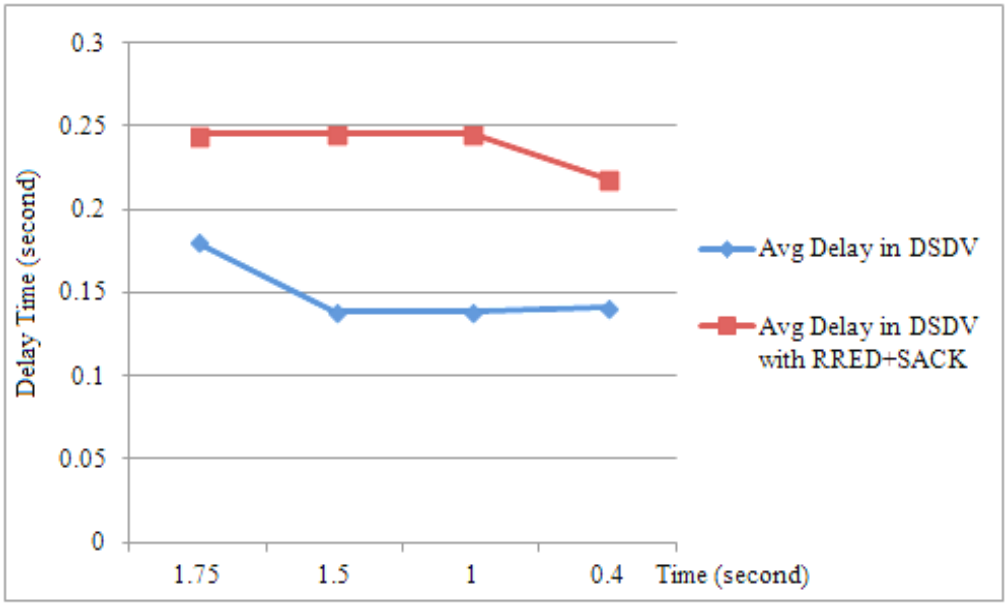

Figure 5.4: Delay between Drop tail and RRED with SACK.

Routing overhead of transmitted packets from sender to receiver as shown in figure 5.5, which gives description about routing overhead in case of DSDV with drop tail and RRED. As the burst length of attack decreases overhead come almost same in both cases.

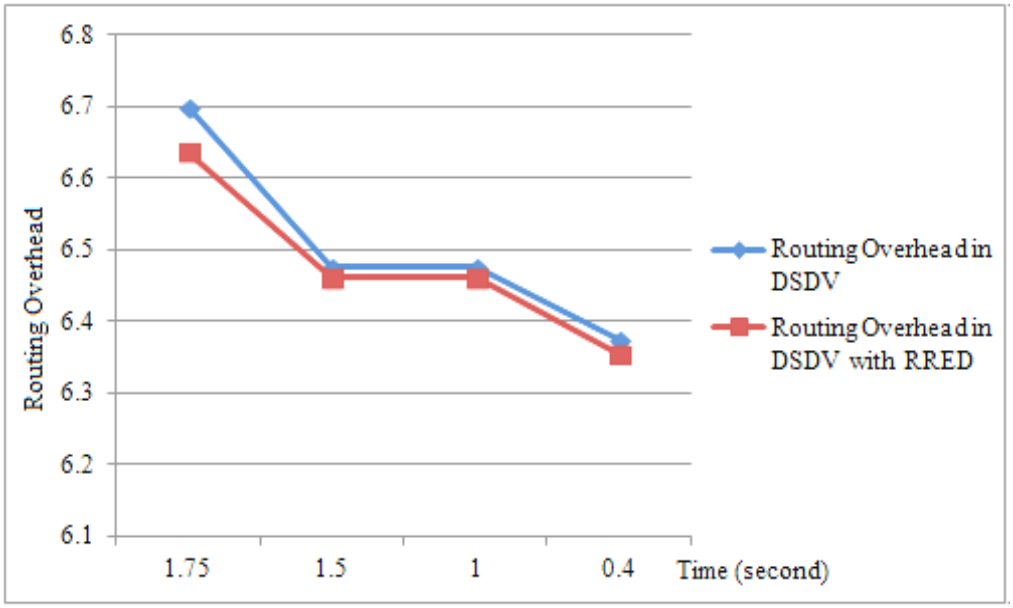

Figure 5.5: Routing Overhead between Drop tail and RRED

There is slight deviation at bust length of 1.5 where Drop tail performs better as compare to RRED.

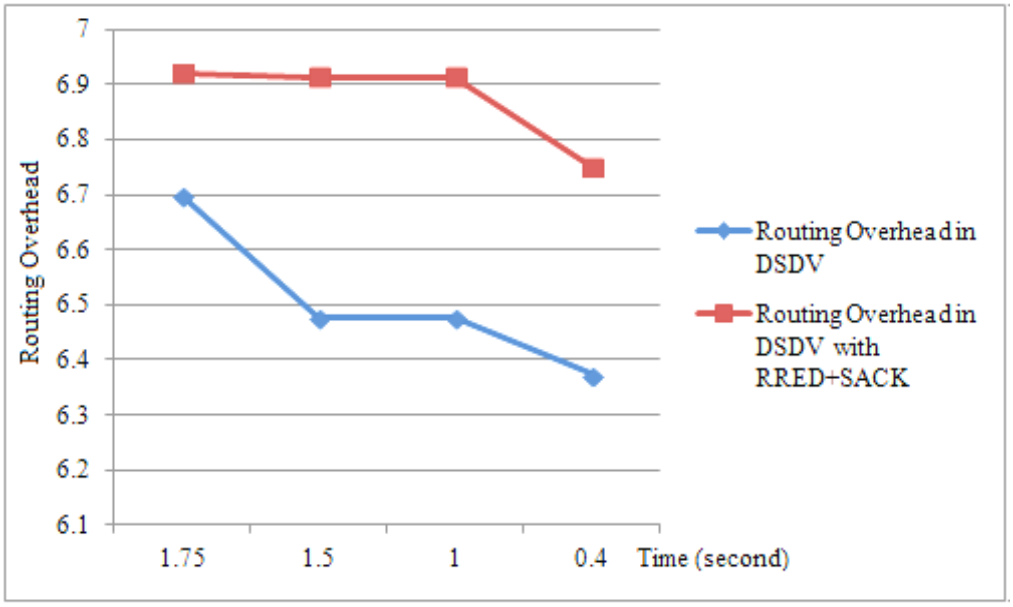

Figure 5.6: Routing Overhead between Drop tail and RRED with SACK

Figure 5.6 describes DSDV with Drop tail performs better over DSDV with RRED and SACK combination. Due to control packets, those have been generated in RRED and SACK. Simple RRED have performed better then it combines with SACK.

Based on analysis of different network parameter conclusion is carried out as given below:

Table 5.1: Performance Comparison 1

	Drop Tail	**RRED**
Throughput	Low	High
Delay	Low	High
Routing Overhead	High	Low

Table 5.2: Performance comparison 2

	Drop Tail	**RRED+SACK**
Throughput	Low	High
Delay	Low	High
Routing Overhead	Low	High

Chapter 6

Conclusion and Future Scope

6.1 Conclusion

In This thesis, Low Rate DoS attacks have been discussed and analysed. In this work examine the impact configurations are vulnerable to such Low Rate attacks. It showed that TCP exhibits of Low-Rate DoS attacks on DSDV under various variant of AQM and TCP. Using detailed experimentation, we show that routers using default performance degradation in term of delay and routing over head when DoS stream multiplexed with a maliciously chosen burst periodic to exploit TCP's retransmission timeout mechanism. An extensive set of simulations and experiments showed in previous section. There is improvement in term of throughput when use RRED mechanism. When apply RRED with SACK, throughput increases at most of the burst periods of attack. But there is slightly decrement in terms end to end delay and routing over head due to control packets triggered by SACK in the network. However, RRED perform better without SACK for above parameters. As defence mechanisms, we advice router based prevention techniques to mitigate the possibility of Low Rate DoS attacks which exploit traffic congestion to impact routing protocols. Using tested experiments we demonstrate the effectiveness of such solutions to prevent Low Rate attacks. Network-router and end-point-based mechanisms can only mitigate, but not eliminate the effectiveness of the attack. To completely defend the system in the presence of such attacks, one would necessarily have to significantly sacrifice system performance in their absence.

6.1 Future Scope

Low Rate DoS attacks are successful against both short and long lived TCP aggregates at different burst period and thus represent a realistic threat to today's wireless networks. For further study, different other variants of AQM and TCP would be implemented to increase performance of network. Also, RTO randomization mechanism will give better performance with RRED and SACK to efficiently counter Low Rate DoS attack in wireless network.

References

[1] W. Stallings, "Wireless LAN Technology" in *Wireless Communication and Networks*, Pearson Education, 2002.

[2] I. Chlamtac, M. Conti, and J. J.-N Liu, "Mobile ad hoc networking: imperatives and challenges," Ad Hoc Networks 1, pp. 13–64, 2003.

[3] Charles E. Perkins, *Ad Hoc Networking*, Addition-Wesley, pp. 125-160, 2001.

[4] Pearlman et al., "Determining the Optimal Configuration for the Zone Routing Protocol," IEEE Journal on Selected Areas in Communications, Vol. 17, No. 8, August 1999.

[5] Elizabeth, Royer et al., "A Review of Current Routing Protocols for Ad-Hoc Mobile Wireless Networks," IEEE Personal Communications, April 1999.

[6] Yau, P-W & Mitchell, CJ, "Security vulnerabilities in ad hoc networks," Proceedings of ISCTA 2003, 7th International Symposium on Communications Theory and Applications, July 2003.

[7] B. Wu, J. Chen, J. Wu, M. Cardei, "A Survey on Attacks and Countermeasures in Mobile Ad Hoc Networks," Wireless/Mobile Network Security, Springer, 2006.

[8] Y. Hu, A. Perrig, D. Johnson, "Packet Leashes: A Defence against Wormhole Attacks in Wireless Ad Hoc Networks," Proc. of INFOCOM, 2003.

[9] M. Allman and V. Paxson, "On estimating end-to-end network path properties," Computer Communication Review, vol. 31, no. 2-Supplement, pp. 124–151, 2001.

[10] X. Luo, R. Chang, and E. Chan, "Performance Analysis of TCP/AQM under Denial-of-Service Attacks, Modelling, Analysis, and Simulation of Computer and Telecommunication Systems," 13th IEEE International Symposium, 2005.

[11] S. Savage, N. Cardwell, D. Wetherall, and T. Anderson, "Tcp congestion control with a misbehaving receiver," SIGCOMM Comput. Commun. Rev., vol. 29, pp. 71–78, October 1999.

[12] C. Zhang, J. Yin, Z. Cai, and W. Chen "RRED: Robust RED Algorithm to Counter Low-Rate Denial-of-Service Attacks", IEEE COMMUNICATIONS LETTERS, VOL. 14, NO. 5, MAY 2010.

[13] A. Kuzmanovic and E. W. Knightly, "Low-rate tcp-targeted denial of service attacks and counter strategies," IEEE/ACM Trans. *Netw.*, vol. 14, no. 4, pp. 683–696, 2006.

[14] M. Morshed, M. Rahman, M. Rahman and M. Islam,"Performance Comparison of TCP variants over AODV, DSDV, DSR, OLSR in NS-2, " International Conference on Informatics, Electronics and Vision, ICIEV'12, pp. 1069-1074, 2012.

[15] H. Sun, J. C. S. Lui, and D. K. Y. Yau, "Defending against low-rate tcp attacks: Dynamic detection and protection," in ICNP, pp. 196–205, 2004.

[16] G. Yang,M. Gerla, andM. Y. Sanadidi, "Defense against lowrate tcp-targeted denial-of-service attacks," In *ISCC '04:* Proceedings of the Ninth International Symposium on Computers and Communications 2004 Volume 2 (ISCC"04), IEEE Computer Society, pp. 345–350, 2004.

[17] Y.-K. Kwok, R. Tripathi, Y. Chen, and K. Hwang, "Hawk: Halting anomalies with weighted choking to rescue well behaved tcp sessions from shrew DDoS attacks," in *ICCNMC*, pp. 423–432, 2005,

[18] A. Shevtekar, K. Anantharam, and N. Ansari, "Low rate tcp denial-of-service attack detection at edge routers," Communications Letters, IEEE, vol. 9, no. 4, pp. 363–365, april 2005.

[19] X. Luo and R. K. C. Chang, "On a new class of pulsing denial-of-service attacks and the defense," in *NDSS*, 2005.

[20] S. Bansal, R.Shorey, S. Chugh, A.Goel, K.Kumar and A Misra, "The Capacity of Multi-hop Wireless Networks with TCP Regulated Traffic," IEEE Global Telecommunications Conference, GLOBECOM'02, pp. 133-137, 2002

[21] C.W. Chang, S. Lee, B. Lin, and J. Wang, “The taming of the shrew: mitigating low-rate tcp-targeted attack,” IEEE Transactions on Network and Service Management, vol. 7, no. 1, pp. 1–13, 2010.

[22] R. Sherwood, B. Bhattacharjee, and R. Braud, “Misbehaving tcp receivers can cause internet-wide congestion collapse,” in ACM Conference on Computer and Communications Security, pp. 383–392, 2005.

[23] M. Allman, V. Paxson, W. Stevens, ’TCP Congestion Control’, RFC2581, April 1999.

[24] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. “TCP Selective Acknowledgment Options,” 1996.

[25] S. Floyd, T. Henderson, A. Gurtov, "The NewReno Modification to TCP's Fast Recovery Algorithm," RFC 3782, April 2004.

[26] M. Allman V. Paxson “TCP Congestion avoidance algorithm”, Sept. 2009.

[27] M. Zin-Oo, M. Othman, “The Effect of Packet Losses and Delay on TCP Traffic over Wire-less Ad Hoc Networks,”.

[28] S. Floyd and V. Jacobson, “Random Early Detection Gateways for Congestion Avoidance,” IEEE/ACM Transactions on Networking, August 1993.

[29] S. Floyd, R. Gummadi, and S. Shenker, “Adaptive RED: An Algorithm for Increasing the Robustness of RED’s Active Queue Management,” August 1, 2001.

[30] “Ns-2,” http://nnam.isi.edu/nsnam/index.php/Contributed_Code#Documentation. [25 June 2013]

List of Publication

1. Arvind Sharma, Neeraj Kumar, "Trust Based Theoretical Framework for Mobile Ad-Hoc Network", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 6, June 2013.

```
#=====================================================

                    # Default Options

#=====================================================

  set val(chan)       Channel/WirelessChannel      ;# channel type
  set val(prop)       Propagation/TwoRayGround  ;#radio-propagation model
  set val(netif)      Phy/WirelessPhy              ;# network interface type
  set val(mac)        Mac/802_11                   ;# MAC type
  set val(ifq)        Queue/DropTail/PriQueue      ;# interface queue type
  set val(ifq)         Queue/RED/Robust            ;# Robust RED
  set val(ll)         LL                           ;# link layer type
  set val(ant)        Antenna/OmniAntenna          ;# antenna model
  set val(ifqlen)      50                          ;# max packet in ifq
  set val(nn)         30                           ;# number of mobilenodes
  set val(rp)         DSDV                         ;#ruting protokol
  set val(x)          1000                         ; # x dimension of the topography
  set val(y)          1000                         ;# y dimension of the topography
  set val(nam)         low_rate.nam                ; # animation file
#=========================================================


Mac/802_11 set basicRate_ 1Mb  ;
Mac/802_11 set dataRate_  1.5Mb  ;
Phy/WirelessPhy set CPThresh_ 10.0 ;
# Capture threshold
Phy/WirelessPhy set CSThresh_ 1.55924e-11 ;
# Carrier sense threshold
Phy/WirelessPhy set RXThresh_ 3.65262e-10;
# Receive power threshold
Phy/WirelessPhy set Rb_ 2*1e6 ;
# Bandwidth
Phy/WirelessPhy set Pt_ 0.2818 ;
# Transmission power
Phy/WirelessPhy set freq_ 914e6 ;
# frequency
Phy/WirelessPhy set L_ 1.0 ;
# system loss factor

#------------default parameter-------------------------------------------------------------#
```

```
Agent/TCP set precisionReduce_ false ;
Agent/TCP set rtxcur_init_ 6.0 ;
Agent/TCP set updated_rttvar_ false ;
Agent/TCP set tcpTick_ 0.1
Agent/TCP set rfc2988_ false
Agent/TCP set minrto_ 1
Queue/RED set bytes_ true
Queue/RED set queue_in_bytes_ true
Queue/RED set q_weight_ 0.001
                Queue/RED set thresh_ 5
                Queue/RED set maxthresh_ 15
                Agent/TCP set singledup_ 0
                Agent/TCP set useHeaders_ false
                Agent/TCP set bugfix_ss_ 0

#----------RRED-------------------------------------------------------------------------------#
                Queue/RED/Robust set hash_bins_ 28
                Queue/RED/Robust set hash_levels_ 2
                Queue/RED/Robust set score_max_ 20
                Queue/RED/Robust set score_min_ 0
                Queue/RED/Robust set score_pass_ 1
                Queue/RED/Robust set last_drop_time_ 0ms
                Queue/RED/Robust set drop_related_period_ 10ms
                Agent/TCP set minrto_ 1;

#---------- Hierachial Topology Structure-----------------------------------------------------#

        $ns_ node-config -addressType hierarchical
        AddrParams set domain_num_ 2        ;# number of sub-domain
        lappend clusterNbr 1                ;#number of cluster in each domain
        AddrParamsset cluster_num_ $clusterNbr
        lappend eilastlevel 14 19           ;#number of nodes in each cluster
        AddrParams set nodes_num_ $eilastlevel
#create trace objects for ns and nam
        set nstrace [open $val(trace) w]
        $ns_ trace-all$nstrace
        set namtrace [open $val(nam) w]
        $ns_ namtrace-all-wireless $namtrace $val(x) $val(y)
#create a topology object and define topology
         set topo [new Topography]
         $topo load_flatgrid $val(x) $val(y)

#create God (General Operations Director)
        create-god [expr $val(nn)+$val(gw)]
```

```
        set chan_1 [new $val(chan)]
#configure for mobile nodes and gateways
        $ns_ node-config -adhocRouting $val(rp)\
                         -llType $val(ll) \
                        -macType $val(mac) \
                        -ifqType $val(ifq) \
                        -ifqLen $val(ifqlen) \
                        -antType $val(ant) \
                        -propType $val(prop) \
                        -phyType $val(netif) \
                        -channel $chan_1 \
                        -topoInstance $topo \
                        -agentTrace ON \
                        -routerTrace OFF \
                        -macTrace ON \
                        -wiredRouting OFF \
#create gateway node
 set #create gateway
set gw(0) [$ns_ node 1.0.0]
#set initial coordinates of gateway node
 $gw(0) set X_ 400.050446
 $gw(0) set Y_ 665.753296
$gw(0) set Z_ 0.0
 $ns_ initial_node_pos
 $gw(0) 40.000000
gw(0) set temp {0.0.1 0.0.2 0.0.3 0.0.4 0.0.5 0.0.6 0. 0.7 0.0.8 0.0.9 0.0.10 0.0.11
          0.0.12 0.0.13 0.0.14 0.0.15 0.0.16 0.1.1 0.1.2 0.1.3 0.1.4 0.1.5 0.1.6 0.1.7
          0.1.8 0.1.9 0.1.10 0.1.11 0.1.12 0.1.13 0.1.14 0.1.15 0.1.16 1.1.1 1.1.2 1.1.3
          1.1.4 1.1.5 1.1.6 1.1.7 1.1.8 1.1.9 1.1.10 1.1.11 1.1.12 1.1.13 1.1.14 1.1.15
          1.1.16 1.0.1 1.0.2 1.0.3 1.0.4 1.0.5 1.0.6 1.0.7 1.0.8 1.0.9 1.0.10 1.0.11
          1.0.12}
for {set i 2} {$i < $val(nn)+2} {incr i}
       {
          set node_($i) [$ns_ node [lindex $temp [expr $i-2]]]
          $node_($i) base-station [AddrParams addr2id [$gw(0) node-addr]]
         }
 ## node_(31) at 684.043762,695.341064
 $node_(31) set X_ 600.043762
 $node_(31) set Y_ 695.341064
 $node_(31) set Z_ 0.0
$node_(31) color "black"
$ns_ initial_node_pos $node_(31) 40.000000

## node_(30) at 1098.283691,620.160583
```

```
$node_(30) set X_ 798.283691
$node_(30) set Y_ 620.160583
$node_(30) set Z_ 0.0
$node_(30) color "black"
$ns_ initial_node_pos
$node_(30) 40.000000
## node_(29) at 889.446899,578.376099
$node_(29) set X_ 589.446899
$node_(29) set Y_ 578.376099
$node_(29) set Z_ 0.0
$node_(29) color "black"
$ns_ initial_node_pos
$node_(29) 40.000000
## node_(28) at 753.570923,661.221680
$node_(28) set X_ 653.570923
$node_(28) set Y_ 661.221680
$node_(28) set Z_ 0.0
$node_(28) color "black"
$ns_ initial_node_pos
$node_(28) 40.000000
# node_(27) at 943.858643,547.126343
$node_(27) set X_ 743.858643
$node_(27) set Y_ 547.126343
$node_(27) set Z_ 0.0
$node_(27) color "black"
$ns_ initial_node_pos
$node_(27) 40.000000
## node_(26) at 748.649048,582.298523
$node_(26) set X_ 796.649048
$node_(26) set Y_ 582.298523
$node_(26) set Z_ 0.0
$node_(26) color "black"
$ns_ initial_node_pos
$node_(26) 40.000000
## node_(25) at 927.925598,794.823853
$node_(25) set X_ 827.925598
$node_(25) set Y_ 794.823853
$node_(25) set Z_ 0.0
$node_(25) color "black"
$ns_ initial_node_pos
$node_(25) 40.000000
## node_(24) at 1021.379272,774.276489
$node_(24) set X_ 921.379272
$node_(24) set Y_ 774.276489
$node_(24) set Z_ 0.0
```

```
$node_(24) color "black"
$ns_ initial_node_pos
$node_(24) 40.000000
## node_(23) at 905.762695,708.384705
$node_(23) set X_ 905.762695
$node_(23) set Y_ 708.384705
$node_(23) set Z_ 0.0
$node_(23) color "black"
$ns_ initial_node_pos
$node_(23) 40.000000
## node_(22) at 1033.135132,697.936707
$node_(22) set X_ 893.135132
$node_(22) set Y_ 697.936707
$node_(22) set Z_ 0.0
$node_(22) color "black"
$ns_ initial_node_pos
$node_(22) 40.000000
## node_(21) at 955.794739,624.773743
$node_(21) set X_ 855.794739
$node_(21) set Y_ 624.773743
$node_(21) set Z_ 0.0
$node_(21) color "black"
$ns_ initial_node_pos
$node_(21) 40.000000
## node_(20) at 846.695801,707.216553
$node_(20) set X_ 846.695801
$node_(20) set Y_ 707.216553
$node_(20) set Z_ 0.0
$node_(20) color "black"
$ns_ initial_node_pos
$node_(20) 40.000000
## node_(19) at 840.668213,783.884888
$node_(19) set X_ 804.668213
$node_(19) set Y_ 783.884888
$node_(19) set Z_ 0.0
$node_(19) color "black"
$ns_ initial_node_pos
$node_(19) 40.000000
## node_(18) at 825.787598,645.756897
$node_(18) set X_ 825.787598
$node_(18) set Y_ 645.756897
$node_(18) set Z_ 0.0
$node_(18) color "black"
$ns_ initial_node_pos
$node_(18) 40.000000
```

```
## node_(17) at 754.516541,717.609497
$node_(17) set X_ 754.516541
$node_(17) set Y_ 717.609497
$node_(17) set Z_ 0.0
$node_(17) color "black"
$ns_ initial_node_pos $node_(17) 40.000000
## node_(16) at 8.315422,782.846252
$node_(16) set X_ 8.315422
$node_(16) set Y_ 782.846252
$node_(16) set Z_ 0.0
$node_(16) color "black"
$ns_ initial_node_pos
$node_(16) 40.000000
## node_(15) at 2.643661,673.418396
$node_(15) set X_ 2.643661
$node_(15) set Y_ 673.418396
$node_(15) set Z_ 0.0
$node_(15) color "black"
$ns_ initial_node_pos
$node_(15) 40.000000
## node_(14) at 119.563614,765.391663
$node_(14) set X_ 119.563614
$node_(14) set Y_ 765.391663
$node_(14) set Z_ 0.0
$node_(14) color "black"
$ns_ initial_node_pos
$node_(14) 40.000000
## node_(13) at 221.328583,610.518616
$node_(13) set X_ 221.328583
$node_(13) set Y_ 610.518616
$node_(13) set Z_ 0.0
$node_(13) color "black"
$ns_ initial_node_pos
 $node_(13) 40.000000
## node_(12) at 121.785881,603.259338
$node_(12) set X_ 121.785881
$node_(12) set Y_ 603.259338
$node_(12) set Z_ 0.0
$node_(12) color "black"
$ns_ initial_node_pos $node_(12) 40.000000
## node_(11) at 56.638927,671.640564
$node_(11) set X_ 56.638927
$node_(11) set Y_ 671.640564
$node_(11) set Z_ 0.0
$node_(11) color "black"
```

```
$ns_ initial_node_pos
$node_(11) 40.000000
## node_(10) at 183.472870,706.771484
$node_(10) set X_ 183.472870
$node_(10) set Y_ 706.771484
$node_(10) set Z_ 0.0
$node_(10) color "black"
$ns_ initial_node_pos
$node_(10) 40.000000
## node_(9) at 39.262756,753.706970
$node_(9) set X_ 39.262756
$node_(9) set Y_ 753.706970
$node_(9) set Z_ 0.0
$node_(9) color "black"
$ns_ initial_node_pos
$node_(9) 40.000000
## node_(8) at 14.260864,698.584656
$node_(8) set X_ 14.260864
$node_(8) set Y_ 698.584656
$node_(8) set Z_ 0.0
$node_(8) color "black"
$ns_ initial_node_pos
$node_(8) 40.000000
## node_(7) at 64.885574,808.397583
$node_(7) set X_ 64.885574
$node_(7) set Y_ 808.397583
$node_(7) set Z_ 0.0
$node_(7) color "black"
$ns_ initial_node_pos
$node_(7) 40.000000
## node_(6) at 192.396423,768.221191
$node_(6) set X_ 192.396423
$node_(6) set Y_ 768.221191
$node_(6) set Z_ 0.0
$node_(6) color "black"
$ns_ initial_node_pos $node_(6) 40.000000
## node_(5) at 119.779785,839.464722
$node_(5) set X_ 119.779785
$node_(5) set Y_ 839.464722
$node_(5) set Z_ 0.0
$node_(5) color "black"
$ns_ initial_node_pos
$node_(5) 40.000000
## node_(4) at 214.249802,867.037354
$node_(4) set X_ 214.249802
```

```
$node_(4) set Y_ 867.037354
$node_(4) set Z_ 0.0
$node_(4) color "black"
$ns_ initial_node_pos
$node_(4) 40.000000
## node_(3) at 256.997345,765.497498
$node_(3) set X_ 256.997345
$node_(3) set Y_ 765.497498
$node_(3) set Z_ 0.0
$node_(3) color "black"
$ns_ initial_node_pos
$node_(3) 40.000000
## node_(2) at 264.491150,673.345398
$node_(2) set X_ 264.491150
$node_(2) set Y_ 673.345398
$node_(2) set Z_ 0.0
$node_(2) color "black"
$ns_ initial_node_pos
$node_(2) 40.000000
puts ""
puts ""
$ns_ at 0.0 "
$gw(0) label \"GATEWAY\""
$ns_ at 2.0 "$node_(10) label \"ATTACKER\""
#########################create links between basestation nodes
#-----------------------------------------------------------------------------------
                                        #Setup Traffic
#-----------------------------------------------------------------------------------
set agent1 [new Agent/TCP/Sack1] ;
set sink [new Agent/TCPSink/DelAck];
 $ns_ attach-agent
$node_(2) $agent1;
$ns_ attach-agent
$node_(29) $sink;
$ns_ connect $agent1
 $sink;
set app1 [new Application/FTP];
$app1 set packetSize_ 512
$app1 attach-agent
 $agent1;
set agent2 [new Agent/TCP/Sack1];
set sink2 [new Agent/TCPSink/DelAck];
$ns_ attach-agent $node_(31) $agent2;
$ns_ attach-agent $node_(5) $sink2;
```

```
$ns_ connect $agent2 $sink2;
set app2 [new Application/FTP];
$app2 set packetSize_ 512
$app2 attach-agent $agent2;
set agent3 [new Agent/TCP/Sack1];
set sink3 [new Agent/TCPSink/DelAck];
$ns_ attach-agent $node_(9) $agent3;
$ns_ attach-agent $node_(15) $sink3;
$ns_ connect $agent3 $sink3;
set app3 [new Application/FTP];
$app3 set packetSize_ 512
$app3 attach-agent $agent3;
$ns_ at 5.4 "$app1 start";
$ns_ at 27.0 "$app1 stop";
$ns_ at 27.4 "$app2 start";
$ns_ at 50.4 "$app2 stop";
#$ns_ at 45.4 "$app3 start";
#$ns_ at 50.0 "$app3 stop";
set Sender [new Agent/TCP/Sack1]
$ns_ attach-agent $node_(10)
$Sender
set Sender_ap [new Application/Traffic/CBR]
$Sender_ap attach-agent
$Sender
set Reciver [new Agent/TCPSink/DelAck]
$ns_ attach-agent $gw(0)
 $Reciver
$ns_ connect
$Sender $Reciver
$Sender_ap set packet_Size_ 20
$Sender_ap set rate_ 2048Mb
set  a 250
set b 500
set c 1000
set d 1450
for {set it 5} {$it < 50}
{
            set it [expr $it+2000/1000.0]}
     {
          set tr 0;
          set tl
             [expr (2000-$d)/1000.0]
             puts "$tl"
             set strt [integer [expr 2000-$d]];
```

```
                set tr [expr $strt/1000.0];
                set atktm [expr $it+$tr] puts
               "attacker random start and stop --------> $atktm"
                $ns_ at [expr $it+$tr] "
               $Sender_ap start";
               $ns_ at [expr $it+$tr+$tl]
               "$Sender_ap stop";
                set atkstp [expr $it+$tr+$tl] puts  "$atkstp"
s               et nsnow [$ns_ now];
 }
#--------------------- #Setup Node Movement #---------------------
$ns_ at 1.0 "$node_(2) setdest 200 500.21 4"
$ns_ at 1.0
"$node_(3) setdest 200 850.21 5"
$ns_ at 2.5 "$node_(4) setdest 120 720.98 7"
$ns_ at 3.5 "$node_(5) setdest 239 670.671 30"
$ns_ at 4.5 "$node_(6) setdest 100 896.59 8"
#$ns_ at 0.5 "$node_(7) setdest 45 600.71 2"
$ns_ at 6.5 "
$node_(8) setdest 232 613.877 4"
$ns_ at 7.5 "$node_(9) setdest 10 964.987671 5"
$ns_ at 1.5 "$node_(10) setdest 400 476.79 30"
$ns_ at 15.5 "$node_(11) setdest 135 974.967 6"
$ns_ at 11.5 "$node_(12) setdest 80 790.10 3"
$ns_ at 2.5 "$node_(13) setdest 124 580.24 5"
$ns_ at 13.5 "$node_(14) setdest 75 664.91 7"
$ns_ at 39.5 "$node_(15) setdest 245 873.11 12"
$ns_ at 0.9 "$node_(17) setdest 654 800.62 6"
$ns_ at 1.9 "$node_(18) setdest 707 700.165 9"
$ns_ at 5.0 "$node_(19) setdest 700 850.8 11"
$ns_ at 13.5 "$node_(20) setdest 905 565.61 10"
$ns_ at 14.5 "$node_(21) setdest 892 866.9 5"
$ns_ at 15.5 "$node_(22) setdest 745 800.17 2"
$ns_ at 16.5 "$node_(23) setdest 859 683.77 7"
$ns_ at 17.5 "$node_(24) setdest 998 700 10"
$ns_ at 17.5 "$node_(25) setdest 995 810.19 7"
$ns_ at 15.5 "$node_(26) setdest 925 674.967 9"
$ns_ at 11.5 "$node_(27) setdest 700 720.21 3"
$ns_ at 0.1 "$gw(0) add-mark m1 green circle"
$ns_ at 0.1 "$node_(10) add-mark m1 red circle"
#-------------------------------------------------------------------------------
            # Tell Nodes When The Simulation Ends
#-------------------------------------------------------------------------------
for {set i 2} {$i < $val(nn)+2} {incr i}
```

```
 {
   $ns_ at $val(stop).0 "$node_($i) reset";
}
$ns_ at $val(stop).0 "$gw(0) reset";
$ns_ at $val(stop).0001 "
stop"
$ns_ at $val(stop).0002 "puts \"NS EXITING...\" ;
 $ns_ halt"
proc stop {}
 {
        global ns_ tracefd
  # Close Trace File
        exec gawk -f pdf1.awk burst_1.75.tr &
        exec gawk -f routingLoad.sh burst_1.75.tr &
        exec gawk -f tt1.sh burst_1.75.tr &
        exec gawk -f e2edelay.sh burst_1.75.tr &
}
puts "Starting simulation..."
$ns_ run
```